Investigation of a framework for seasonal time series forecasting

Colin Leverger

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

« Colin LEVERGER »

« Investigation of a framework for seasonal time series forecasting »

Thèse présentée et soutenue à « IRISA Rennes », le « 16 novembre 2020 »
Unité de recherche : UMR 6074 IRISA
Thèse N° :

Rapporteurs avant soutenance :

Anthony BAGNALL	Professeur, University of East Anglia
Antoine CORNUEJOLS	Professeur, AgroParisTech

Composition du Jury :

Président :	Béatrice DUVAL	Professeure, Université d'Angers
Examinateurs :	Usue MORI	Maître de Conférences, Université Basque
	Themis PALPANAS	Professeur, Université de Paris
	Simon MALINOWSKI	Maître de Conférences, Université de Rennes 1
	Thomas GUYET	Maître de Conférences, Institut Agro/IRISA Rennes
Dir. de thèse :	Alexandre TERMIER	Professeur, Université de Rennes 1

Invité(s) :

Laurence ROZÉ	Maître de Conférences, INSA Rennes
Vincent LEMAIRE	Chercheur, Orange Labs Lannion
Alexis BONDU	Chercheur, Orange Labs Paris
Régis MARGUERIE	Manager technique, Orange

Remerciements

Cela fait six ans que je travaille chez Orange. J'ai débuté comme apprenti ingénieur logiciel et à la fin de mon cursus, lorsque le monde commençait à être habitué à l'intelligence artificielle, j'ai eu l'opportunité de continuer mon expérience avec cette thèse en Data Science, dont voici le manuscrit. Trois années mouvementées où j'ai eu l'occasion d'apprendre la recherche, le *machine learning* et tant d'autres choses encore. Ce qui est sûr, c'est que j'étais loin de me douter de ce qui m'attendait lors de ces trois années ; cette expérience, riche, mais intense, n'aurait pas été possible sans l'aide de différentes personnes et sur différents plans (techniques comme personnels). Ces quatre pages de remerciements leur sont dédiées. Pour entrer directement dans le vif du sujet de ma thèse, je vous invite à aller à la page 21 pour l'introduction.

Tout d'abord, je remercie l'entreprise Orange pour l'opportunité donnée lors de cette thèse : confier un projet de recherche de cette envergure à un apprenti non formé à la recherche était un pari risqué, que j'ai eu le plaisir de relever en étant bien accompagné. Je remercie Régis MARGUERIE pour avoir cru en moi lors de la création du sujet et pour toute l'aide apportée lors de ces trois années, pour chaque réunion mensuelle et chaque point de synchronisation. Je remercie Frédéric JAY pour la gestion du projet PerForecast et pour son aide lors des deux dernières années de la thèse, ainsi que l'équipe des métrologues et des experts pour ces six années de bonne ambiance et de formation continue et bienveillante. Vous allez me manquer (Stéphane avec son légendaire optimisme, David avec ses coding games, Laurence avec ses bons conseils, Bello pour son enthousiasme et sa grande gentillesse... et tout le monde !).

Lors de la thèse, j'ai eu la chance de pouvoir profiter d'une mobilité internationale en officiant comme chercheur détaché pendant trois mois à Tokyo au Japon. Cette expérience n'aurait pas été possible sans l'aide de Régine ANGOUJARD MIET (Orange), Jean-Luc TAGLIAMONTE (Orange), Caroline SURQUAIN (INRIA), que je remercie très chaleureusement. Nous en avons passé des heures, à nous battre avec le NDA... mais nous avons réussi ! Par la même occasion, je remercie l'Institut National d'Informatique (NII) à Tokyo pour leur accueil, ainsi que Ryota KOBAYASHI : *thank you very much for your support and welcoming for those three perfect months at NII labs at Tokyo—hope to see you soon.* Je remercie aussi l'Université de Bretagne Loire-Atlantique et le Conseil Régional de Bretagne pour l'obtention d'une bourse de mobilité qui m'a permis de financer ce séjour de recherche. Enfin, je remercie chaleureusement Sawako TERMIER pour son aide lors de l'apprentissage du japonais : dans un laps de temps aussi court, avec mes recherches en parallèle, j'ai été capable grâce à toi de lire quelques Kanjis et de me présenter en japonais. Avec du recul, mon programme d'apprentissage de la langue était très (trop) ambitieux... mais c'était un plaisir de préparer ce séjour avec ton aide.

Je remercie Vincent LEMAIRE d'avoir su m'orienter lors des questions techniques de recherche et bien sûr pour toutes les réunions mensuelles et les idées que tu as pu me suggérer en temps opportuns. Tu te plais à dire que tu n'as été qu'un « aiguilleur et donneur d'avis » lors de cette expérience, je te confirme

que ton rôle est allé bien au-delà. Par ailleurs, je te remercie d'avoir rendu possible mon intégration ponctuelle dans une équipe de Data Scientistes à Orange Gardens, qui m'a clairement permis de faire de belles rencontres et de collaborer avec des talents du groupe pour ma recherche, c'était un atout pour l'avancement des travaux (merci à Bruno KAUFFMANN et ses équipes pour l'accueil !).

Je remercie tout particulièrement Alexis BONDU (Orange) pour son accompagnement lors de l'établissement de l'état de l'art et dans ma recherche d'une direction. Ta participation et ton aide à ce moment charnière dans la thèse ont été comme une bouée de sauvetage pour moi, qui était dans le brouillard le plus profond à ce moment précis [1]. Tu m'as réellement aidé à construire un projet de recherche concret sur lequel expérimenter, me lancer, et cela nous a permis ensuite de faire deux publications (et une troisième à venir) avec un contenu de recherche intéressant et de surcroît avec de bons résultats. Cette aide a énormément compté pour moi et je t'en remercie encore. Par ailleurs, c'est toujours un réel plaisir de collaborer avec toi — au plaisir de continuer dans le futur :-)

Ces trois années de thèse ont été l'occasion pour moi de découvrir le monde de la recherche, notamment en étant immergé la majeure partie du temps au laboratoire IRISA/INRIA dans l'équipe LACODAM. J'ai partagé mon bureau avec Kevin, Yichang et Raphael, que je remercie pour la bonne ambiance et pour leur sympathie. Nos sujets étaient tous différents, mais nos discussions techniques ont pu faire émerger de nouvelles idées et aider la correction de certains problèmes. Je remercie également affectueusement l'équipe des doctorants (dans l'ordre d'apparition à LACODAM : Yann, Clément, Mael[s], Alban, Johanne, Heng, Grégory, Camille, Josie, Julien) ainsi que Luis (qui s'est infiltré dans la « team des jeunes » assez rapidement à son arrivée ;-)) — nous avons pu nous soutenir lors des moments difficiles, prendre des cafés et faire des soirées pour fêter les papiers, aller en conférence ensemble... et ce ne sont que de supers souvenirs. Évidemment, je remercie l'équipe LACODAM plus largement (en incluant la « team des vieux ») pour tous les conseils, la bienveillance et la bonne ambiance. Nous avons la chance d'être dans une équipe dynamique, avec plein d'horizons différents et avec toujours de bonnes discussions — à la fois en recherche, mais aussi dans un cadre plus détendu à TAHITI par exemple. Merci à tous et au plaisir de collaborer dans le futur pour continuer de faire vivre la recherche ensemble. Je remercie plus particulièrement Gaelle, pour m'avoir aidé à organiser mes séjours de conférence, ainsi que la soutenance de thèse : ton aide était indispensable pour aborder ces événements sereinement.

Pour faire ma recherche, j'ai été accompagné par de super encadrants le long de ces trois ans. Tout d'abord, merci à Alexandre TERMIER, directeur de la thèse, pour ton support. C'était un plaisir de partager avec toi toutes les réunions mensuelles (avec ces fameuses illustrations corporate dont tu es si empreint), c'était aussi très enrichissant d'apprendre à ton contact à mener la recherche, trouver de nouvelles pistes et à penser stratégie. Tu m'as aussi permis d'organiser le séjour au Japon (je me souviens encore du deuxième jour de ma thèse ou je suis allé te voir en te demandant comment on pouvait travailler pour ce séjour, que j'avais planifié avant même le début de la thèse), cela a rendu l'expérience de thèse encore plus forte. Je remercie chaudement Laurence ROZE, pour ta participation à toutes les réunions hebdomadaires. Tu as toujours été très positive et rassurante lors de cette expérience et cela m'a permis plus d'une fois de reprendre confiance lors des moments difficiles, merci. Je remercie chaleureusement

1. ... Engagez-vous, qu'y disaient... M'enfin, les six premiers mois d'une thèse, quelle aventure, vraiment !

Simon MALINOWSKI, pour le suivi plus technique des algorithmes en R et bien sûr pour l'encadrement de la thèse. Je me souviens particulièrement de notre soirée à Nice où nous avons pu profiter de bonnes bières avant le séminaire séries temporelles à Orange (bien que le déroulement même de la soirée me semble encore aujourd'hui assez flou) ; ta patience, ta pédagogie et ton calme placide m'ont inspiré dans mon apprentissage d'une posture plus posée, merci pour cela. Un énorme merci pour Thomas GUYET pour la gestion de la majorité de la thèse (sur le plan organisationnel, administratif, recherche, développement). C'est sûr, sans ton aide, je ne serai pas arrivé aussi loin. Merci de m'avoir répété 15 fois de générer des données (la quinzième fois fut la bonne), de m'avoir donné tant de conseils sur les présentations et sur les objectifs de telle ou telle piste, de m'avoir soutenu lors de mes moments difficiles pour mes problèmes de santé, et merci d'avoir massivement contribué à la qualité du code de recherche pour la dernière année. Nous avons tous deux appris à travailler ensemble ces trois dernières années et cela commence à devenir bien huilé — dommage que ça se termine si tôt (?!) ;-). Évidemment, je remercie mes quatre encadrants pour l'aide à la rédaction d'articles, pour la structuration et rédaction de la thèse et pour leurs remarques qui ont toutes eu leur importance.

Je remercie Dominique GAY ainsi que Themis PALPANAS pour leur participation au comité de suivi de thèse. Je remercie le jury de la thèse pour leur participation à la soutenance et à la relecture du manuscrit, ainsi que pour les échanges que nous avons pu avoir avant, pendant ou après la soutenance (Anthony BAGNALL, Antoine CORNUEJOLS, Béatrice DUVAL, Usue MORI, Themis PALPANAS).

Ma thèse s'est inscrite dans un moment charnière de ma vie : je venais de terminer mes études d'ingénieur, je me demandais quelle voie suivre pour ma carrière et je venais en plus de me découvrir une condition de santé relativement compliquée à gérer, spécialement lors d'un travail de recherche aussi intense qu'une thèse. Pour affronter ces moments parfois difficiles, mes proches et ma famille ont étés particulièrement importants pour moi. Je remercie tout d'abord Djo et Mel, pour toutes ces soirées et repas organisés rue de la pinterie, où il fait décidément bon vivre (« Quelle vue, les gars ! »). Djo, ton rôle a été primordial, notre amitié et notre projet musical commun m'ont vraiment aidé à souffler et à prendre l'air au long de ces années. Cela a certainement insufflé de nouvelles idées et directions pour toute ma recherche : encore merci et pourvu que notre projet continue de grandir tel qu'il le fait déjà. Merci à Ladjez de nous avoir rejoints dans l'aventure musicale, perfectionnant encore cette soupape d'évacuation de la pression qu'était la musique. Merci à Joss, Lilie, Victor, Elé, Paulo, Marie, Fanch, pour votre écoute et pour votre franche camaraderie, pour toutes nos soirées passées à siroter de la liqueur en refaisant le monde : là encore un exutoire nécessaire pour continuer ma recherche sereinement.

Merci à Ninon et Léonie, mes très chères sœurs, pour votre soutien durant ces trois années. Nous sommes tous trois à nouveau diplômés au cours de ces quelques années passées, je suis fier de nous :-) *Big up* particulier à Ninon et Ivan pour leur accueil à Birmingham lors de ma conférence IDEAL2019, c'était un plaisir de voir comment vous vivez de l'autre côté de la Manche. Merci à Valérie, pour ton soutien inconditionnel et pour ton aide pour mettre au clair mes idées dans les temps difficiles ; tu as su me donner ton goût du travail bien fait et tu as su sincèrement et réellement supporter chacune de mes décisions depuis toutes ces années (bonnes ou mauvaises — après tout, l'expérience est une lanterne qui n'éclaire que le chemin parcouru. . .), et je t'en remercie encore mille fois. Ce cheminement et ces 8 années

d'études n'auraient pas étés possibles sans ton soutien. Merci à Jean-Louis, pour le soutien, les discussions stratégiques sur les tenants et les aboutissants de telles ou telles décisions, pour m'avoir transmis ton talent de vulgarisateur qui m'a été de grandes utilités ces derniers temps, pour les merveilleux petits plats du dimanche midi en famille, mais aussi pour le bricolage étiopathique que tu as su me prodiguer tout au long de la thèse et depuis l'émergence de mes soucis de santé. Sans toi, c'est sûr, je serai encore en train de boiter, sourd comme un pot, et ma recherche aurait sans aucun doute été plus difficile encore à mener... Mille mercis (et encore plus) pour tout cet amour familial qui m'a toujours porté vers l'avant.

Je remercie du fond du cœur Ketsia, qui partage ma vie depuis le début de mes études supérieures. La période de la thèse n'était pas la plus simple pour toi : tu as du supporter mes changements d'humeurs liés à une recherche si imprévisible. La seconde année de la thèse ayant vraiment été difficile, tu as du mérite, de m'avoir supporté – et de m'avoir aidé à continuer d'y croire. Merci, pour tous tes conseils, pour toutes les répétitions de présentation auxquelles tu as gentiment participé, pour toutes les relectures, pour ton accueil à Paris pendant la moitié de la thèse. Merci aussi d'avoir été là pour me soutenir lors des moments incertains et d'avoir été là pour célébrer les moments de joie. Merci, pour tous les petits plats de chef que nous avons partagé, confinés que nous étions, lors de la rédaction de la thèse. Et merci de m'avoir appris à réfléchir autrement, à remettre en question les choses, à prendre du recul... et pour tant d'autre chose encore ! Je te dois beaucoup pour tout cela.

Last but not least, je dédie cette thèse à mes grands-parents Papé et Mamé, qui m'ont accompagné, tout au long de ma vie, mais aussi lors de la rédaction finale du manuscrit — la dernière semaine passée à Brett', bercé par les vagues, a permis à bien des tournures de phrases de se conclure et à la rédaction de la thèse de se terminer (...enfin ! après ces 9 semaines de rédaction intenses !). Merci pour votre enthousiasme sur mon sujet, pour votre fierté d'avoir un docteur dans la famille et pour la quantité impressionnante de gentillesse distillée çà et là au grès de ma vie : cette thèse, elle résulte de tout cela, vous en êtes aussi les auteurs, car vous m'avez aidé à devenir la personne que je suis aujourd'hui.

«La prédiction est un art difficile, surtout lorsqu'elle concerne l'avenir.»

– Jean-Louis, citant Pierre DAC

RÉSUMÉ EN FRANÇAIS

Les logiciels et les applications web font partie intégrante de notre vie quotidienne. Il est en effet difficile d'imaginer un monde sans Internet : nous vivons à une époque connectée, où même les services essentiels (tels que les systèmes de santé, les systèmes gouvernementaux, les systèmes militaires et de sécurité, les systèmes financiers, *etc.*) sont hébergés sur des milliers de serveurs à travers le monde. Par exemple, il est estimé que l'entreprise Google dispose d'environ 1 million de serveurs pour supporter ses services. [2] Si les serveurs viennent à manquer, les applications deviennent lentes et difficiles à utiliser. Au contraire, s'ils sont trop nombreux, les performances sont meilleures, mais les coûts induits [3] pour la gestion des serveurs superflus ne sont pas négligeables et devraient être réduits dans le meilleur des cas. La maintenance des infrastructures, assurée par les entreprises ou les institutions, n'est pas une tâche aisée. Les serveurs informatiques sont nombreux, les applications de plus en plus complexes, les technologies évoluent rapidement et le nombre d'utilisateurs est en constante augmentation. Le défi consiste à offrir le meilleur service au public à tout moment et en toutes circonstances.

Pour répondre au défi de la gestion des infrastructures, il est nécessaire de planifier les capacités requises à leur bon fonctionnement. Le planning capacitaire représente la façon dont une entreprise gère, met à jour, ajoute ou retire des serveurs physiques et virtualisés de son infrastructure. Un bon planning capacitaire réduit les coûts opérationnels et améliore la qualité des services fournis. Ses tâches et ses objectifs dépendent fortement du type d'infrastructure (baies de serveurs, ou infrastructure *cloud*), du type d'application hébergée (applications critiques ou avec moins de contraintes), des objectifs de l'entreprise et du budget alloué.

Les différents services proposés par Orange (accès internet grand public, réseau 4G, applications entre entreprises, *etc.*) sont hébergés sur des milliers de serveurs et la qualité du service est primordiale pour proposer la meilleure expérience aux clients. Cet opérateur téléphonique français historique est l'une des marques internationales majeures dans le domaine des télécommunications. Le groupe Orange est présent dans 29 pays à travers le monde et concentre ses activités sur la téléphonie et la vente d'accès à Internet – avec pour objectif principal d'offrir une qualité de service sans faille.

Au sein d'Orange, les infrastructures sont gérées par des gestionnaires de projet et des experts, via des directives de planification des capacités majoritairement manuelles. Cette approche est dispendieuse pour l'entreprise : afin d'éviter les interruptions d'activité, elle tend à surestimer les besoins. Elle est également coûteuse en maintenance : plus le système est complexe, plus il est difficile de le gérer manuellement. Ainsi, le développement d'outils aidant les ingénieurs à anticiper les besoins futurs est en pleine expansion. Bodik et *al.* [Bod+09] montrent que l'analyse de données d'indicateurs de performance des infrastructures (comme le nombre d'utilisateurs, l'utilisation du CPU ou de la RAM) peut être utilisée pour améliorer la planification des capacités. En effet, les données générées par les serveurs, sous forme de série temporelle (données numériques ordonnées dans le temps), pourraient être de précieuses sources d'information. Chez

2. Voir estimation: `https://www.quora.com/How-many-servers-does-Google-have-1`
3. En termes d'argent dépensé, mais aussi d'énergie utilisée, *etc.*

Orange, les données fonctionnelles peuvent concerner l'utilisation des services, ou des problématiques financières, comme l'illustre la Fig. 1.

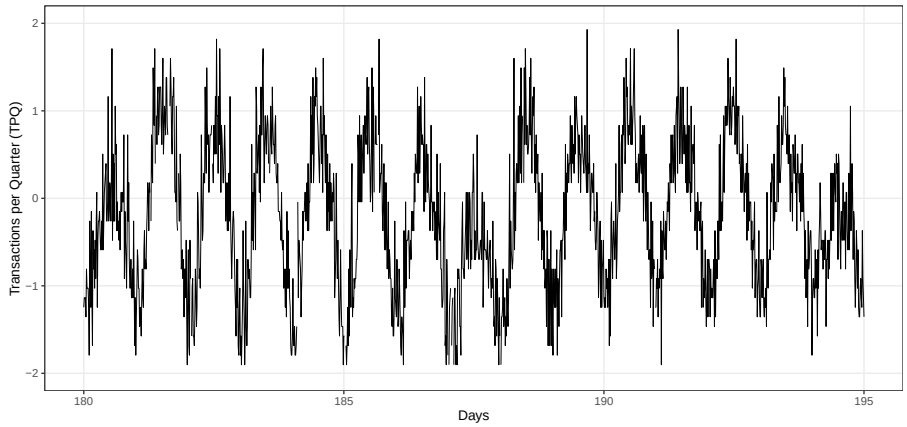


Figure 1: Exemple de métriques fonctionnelles, transactions financières par quinzaine de minutes pour un projet Orange (Orange Money).

Un des enjeux majeurs de l'analyse est de prévoir l'évolution des données dans le temps. L'anticipation du comportement futur des consommateurs est une application importante en matière de marketing, mais aussi pour la gestion des serveurs. Plus les prévisions sont précises, plus les décisions concernant la planification des capacités seront éclairées : des calculs fiables permettent par exemple de décider quel système mettre à niveau. Les prévisions de séries temporelles sont particulièrement utiles lorsque l'on dispose de peu de connaissances sur le processus de génération des données, ou lorsqu'il n'existe pas de modèle explicatif satisfaisant qui relie la variable à prédire à d'autres variables explicatives [Zha03]. Les approches statistiques les plus populaires sont les processus autorégressifs [Aka98], les modèles ARIMA [Box+15] ou le lissage exponentiel type Holt-Winters [Win60].

De nombreuses séries temporelles, notamment celles liées aux activités humaines ou aux phénomènes naturels, présentent un caractère saisonnier : leurs valeurs sont périodiques et régulières. Les comportements liés à l'activité humaine présentent souvent une saisonnalité quotidienne et hebdomadaire. C'est particulièrement vrai pour les ensembles de données de planification des capacités Orange. Sur la Fig. 1 ci-dessus, la saison quotidienne est très nette. Elle est due à l'objectif de l'application, qui est de transférer de l'argent entre les utilisateurs, et il y a naturellement plus de transferts le jour que la nuit, les transferts ayant lieu entre des populations situées sur les mêmes fuseaux horaires.

Il est essentiel de connaître le caractère saisonnier d'une série, car cela peut restreindre l'espace de recherche de son modèle mathématique. STL [Cle+90] est l'approche classique pour traiter de la saisonnalité ; elle construit un modèle en tenant compte de trois composantes : la saisonnalité, la tendance (évolution à long terme de la série temporelle : augmentation ou diminution) et les résidus (écart par rapport aux tendances et à la saisonnalité). Il est alors supposé que la série temporelle ne présente qu'un seul comportement périodique (ex : périodicité quotidienne des températures). Les modèles Holt-Winters et ARIMA ont été étendus pour traiter de la saisonnalité, sous condition d'un comportement périodique unique et clair. Cependant, cette hypothèse est en pratique peu fréquemment vérifiée. Considérons la

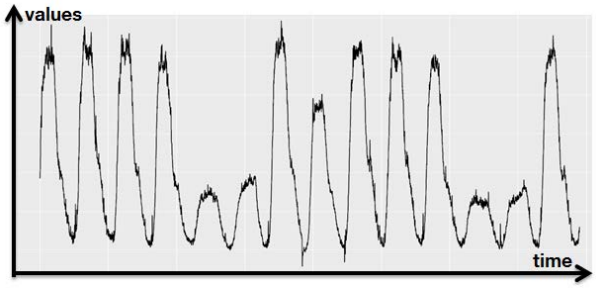


Figure 2: Exemple de série temporelle saisonnière (source: [Hyn11]). Deux semaines de données sur le trafic internet (en bits) provenant d'un fournisseur d'accès privé ayant des centres dans 11 villes européennes. L'ensemble des données correspond à une liaison transatlantique et a été collecté de 6 h 57 le 7 juin à 11 h 17 le 31 juillet 2005. La série temporelle est évidemment saisonnière, mais l'hypothèse d'un schéma périodique unique ne semble pas appropriée dans ce cas.

série temporelle de la Fig. 2, qui montre une mesure du trafic Internet pendant deux semaines. S'il existe effectivement une périodicité quotidienne, il existe également deux types de comportements quotidiens : les comportements en semaine et les comportements en week-end. Le cadre de la STL et les méthodes statistiques associées ne permettent pas de bien saisir ces caractéristiques.

Cette thèse étudie ce problème de prévision des séries temporelles saisonnières dans le contexte de la planification des capacités à Orange. Techniquement parlant, les séries temporelles vont être découpées, groupées, puis il va s'agir de s'aider des séries temporelles et des groupes pour apprendre à prédire le groupe de la saison suivante. Enfin, une série temporelle représentant au mieux le groupe prédit va être utilisée pour produire la prévision.

Un framework de prévisions de séries saisonnières

Dans cette thèse, un *framework* consacré aux séries temporelles saisonnières est proposé. Ce *framework* est basé sur la combinaison d'algorithmes de clustering et de classification pour produire des prévisions déterministes et probabilistes. Trois instanciations en seront étudiées.

La Fig. 3 explicite les différents blocs qui composent le *framework*. Les séries temporelles sont d'abord séparées en un ensemble d'apprentissage et un ensemble de test, et les saisons composant les séries sont extraites (la longueur d'une saison peut être connue de l'utilisateur ou trouvée automatiquement par un algorithme). Ensuite, les différentes saisons composant l'ensemble d'apprentissage sont groupées selon leurs ressemblances – à l'aide d'un algorithme de clustering ou de coclustering. L'étape suivante consiste en l'apprentissage du type de groupe de la saison à venir grâce aux données d'une ou de plusieurs saisons précédentes – à l'aide d'un classifieur par exemple. Enfin, le classifieur et les informations des saisons de l'ensemble de test sont utilisés pour effectuer les prédictions. Ces dernières sont créées en utilisant une combinaison d'une ou plusieurs séries temporelles représentatives des groupes prédits.

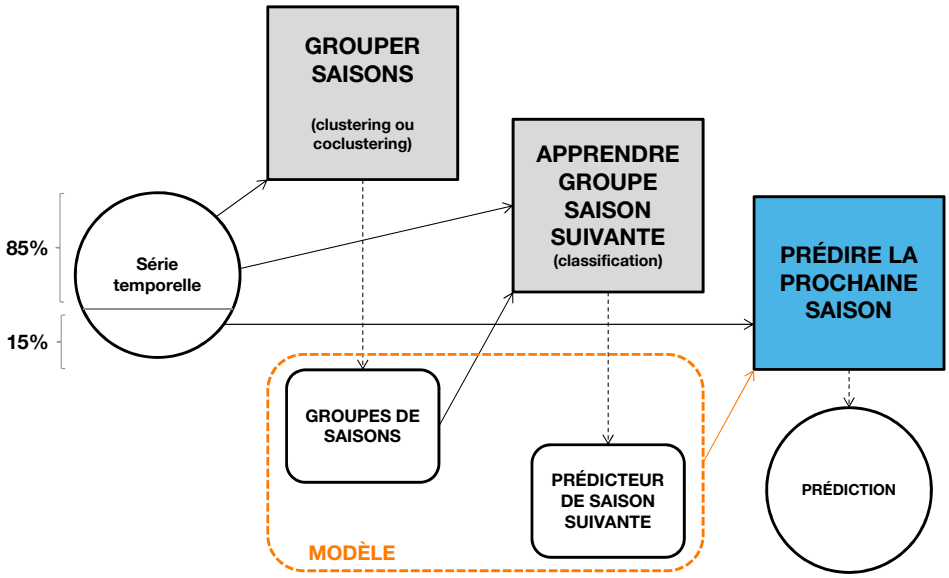


Figure 3: Vue simplifie du *framework* proposé, avec en gris l'apprentissage du modèle, et en bleu l'utilisation du modèle pour créer les prévisions.

Contributions et organisation du manuscrit

Les contributions de cette thèse sont les suivantes.

- Le **premier chapitre** présente le contexte dans lequel cette thèse s'inscrit. Plus particulièrement, les notions de performances, de planification des capacités et de prévisions de séries temporelles pour la planification des capacités sont abordées ;
- Le **second chapitre** donne un état de l'art de la prévision des séries temporelles et introduit les définitions formelles nécessaires pour aborder le reste de la thèse ;
- Le **troisième chapitre** présente *un framework de prévisions de séries temporelles saisonnières*. Seuls les concepts, la formalisation et les idées générales sont introduits. Toutes les notions utilisées avec le *framework* sont définies, car les trois instanciations proposées suivent toutes les mêmes principes et le même formalisme ;
- Ensuite, trois instanciations du *framework* sont présentées.
  - Le **quatrième chapitre** présente une première implémentation déterministe. Il utilise le *clustering K-means* ainsi que des *Chaines de Markov* pour créer des *prévisions déterministes*. Le choix des K-means et des modèles de Markov est naturel, car ils sont tous deux bien connus de la communauté des chercheurs et constituent des étalons, ce qui est utile pour évaluer les hypothèses. Cette première implémentation a été introduite dans la publication [Lev+18].
  - Le **cinquième chapitre** introduit la seconde implémentation déterministe. Le *framework* de référence est élargi grâce à l'utilisation de *plusieurs algorithmes de clustering* (K-means, K-shape, GAK et MODL), et les chaînes de Markov sont remplacées par des *classifieurs*

(classifieurs naïfs bayésiens, arbres de décisions, forêt aléatoire et régression logistique). Cette seconde implémentation a été introduite dans la publication [Lev+19].

— Le **sixième chapitre** présente une dernière implémentation basée sur le *coclustering* et la *classification* probabiliste pour produire des *prévisions probabilistes*. L'expérimentation avec divers algorithmes et diverses combinaisons était primordiale, en particulier pour introduire un *framework* en *portfolio* qui s'adapterait à chaque topologie de données, de manière à avoir des algorithmes pilotés par les données. Les prévisions probabilistes sont en principe intéressantes dans un contexte où l'incertitude est grande, comme dans les problèmes de planning capacitaire.

— L'utilisation pratique du *framework* est illustrée sur un cas d'utilisation concret dans le **septième chapitre**, en utilisant un ensemble de données fourni par la société Orange issue de l'application Orange Money. Ce cas d'utilisation est utile pour explorer les résultats des différentes parties du *framework* (*e.g.* clustering, prévisions, *etc.*).

Expérimentations

Le Tableau 1 explicite comment le *framework* peut être instancié en fonction des paradigmes choisis. Globalement, les deux différences fondamentales entre l'implémentation déterministe et l'implémentation probabiliste résident dans *l'utilisation du clustering ou du coclustering* pour créer les groupes de jours, et dans *l'utilisation des centroïdes/medoïdes ou des densités* pour créer les prévisions.

Dans l'étape A) (Clustering), différents algorithmes de clustering peuvent être utilisés, notamment K-means, K-shape, GAK et MODL. Dans l'étape B) (Classification), différents algorithmes de classification peuvent être utilisés, tant qu'ils sont capables de donner des prévisions probabilistes, notamment classifieurs naïf bayésiens, arbres de décisions, forêt aléatoire et régression logistique. Différentes combinaisons de ces algorithmes de clustering et de classification sont testées lors des expériences menées dans cette thèse.

Table 1: Configuration du *framework* proposé.

		Forecasts déterministes	**Forecasts probabilistes**
Processus d'apprentissage	A) Clustering	Clustering	Coclustering
	B) Classifieur	Classification	
Processus de prévision	C) Prototypes	Centroïde ou medoïde	Densité
	D) Prévision	Hard ou Soft	

Lors des expériences, l'erreur de prédiction est quantifiée en calculant la *Mean Absolute Error* (MAE) entre les valeurs prédites et les valeurs réelles. Chaque série temporelle est divisée chronologiquement en trois parties : 70% des données sont utilisées comme données d'entraînement, 15% comme données de validation et 15% comme données de test. Les données utilisées pour les expérimentations sont présentées de manière exhaustive dans un tableau en annexe 7.2. Notez que les ensembles de données présentés sont utilisés tout au long de la thèse pour toutes les expérimentations. Il y a 49 séries temporelles utilisées au moment des expérimentations, provenant de diverses sources telles que la bibliothèque TSDL [Hyn11], deux ensembles de données proviennent des projets Orange Money (ceux-ci seront étudiés plus en profondeur

dans le dernier chapitre de cette thèse), des villes de Porto ou Melbourne, *etc.* Naturellement, les séries temporelles utilisées dans cette thèse ont été sélectionnées pour leur saisonnalité évidente.

Enfin, l'ensemble du code du *framework* a été développé en Python 3.5. Pour le développement des algorithmes de clustering, la bibliothèque *tslearn* [Tav+20] a été utilisée. Elle constitue un ensemble d'outils utiles pour le traitement des données temporelles. Pour développer la classification, la bibliothèque *sklearn* [Ped+11] a été utilisée. Pour le développement de l'algorithme de coclustering, *Khiops* [Bou16] a été utilisé.

Résultats principaux

Le *framework* proposé est implémenté en trois versions. Deux versions produisent des prévisions déterministes (voir Chap. 4 et Chap. 5), une version produit des prévisions probabilistes (voir Chap. 6). Les performances des trois versions du *framework* sont comparées aux performances d'opposants de l'état de l'art. En résumé,

— Les performances de la première base déterministe FC2M présentée dans le Chap. 4 sont satisfaisantes contre des opposants non spécialisés dans les séries temporelles saisonnières (ARIMA, AR, Holt Winters), mais en dessous d'une méthode purement saisonnière (SARIMA) – voir Fig. 4.

— Les performances de la seconde version déterministe F2C présentée dans le Chap. 5 sont meilleures que tous les opposants déterministes utilisés pour les comparaisons – voir Fig. 5. L'influence du nombre de saisons utilisé lors de l'apprentissage des classifieur (paramètre γ) est étudiée et utiliser plus d'historique permet d'avoir de meilleures prévisions.

— Les performances de la version probabiliste présentée dans le Chap. 6 sont satisfaisantes: la création de prévisions probabilistes visuellement intéressante (*e.g.* qui suivent bien les vraies données – voir Fig. 6) est constatée, et la méthode PF2C utilisée en portfolio arrive en troisième position – voir Fig. 5.

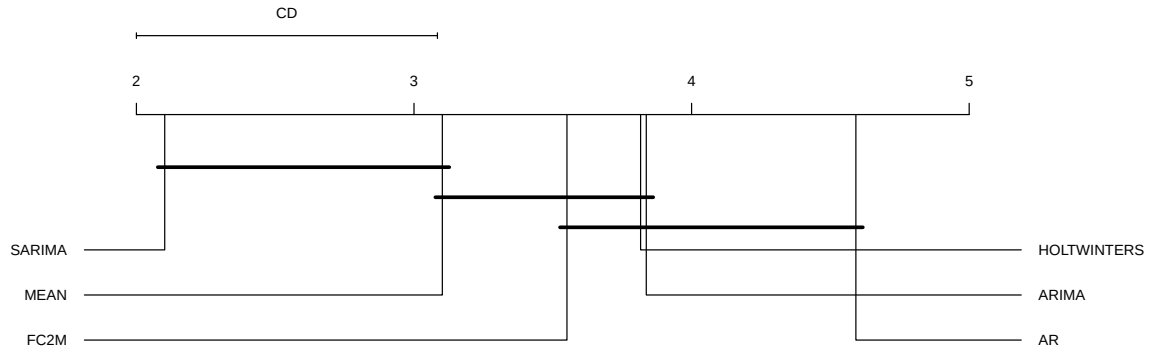


Figure 4: Classement global pour la méthode FC2M contre quatre opposants (AR, ARIMA, SARIMA, Holt Winters) et une méthode *baseline* (MEAN) en utilisant la MAE.

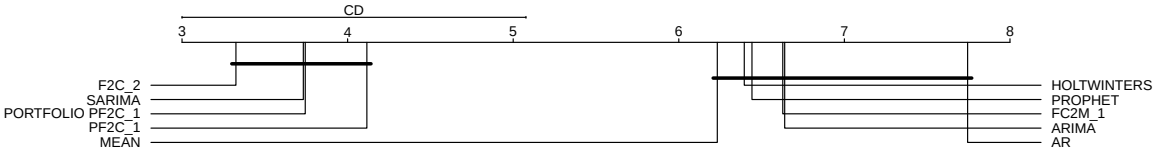


Figure 5: Classement global des différentes approches de prédiction pour différentes valeurs de $\gamma \in [1,2]$ (ex: F2C_γ)

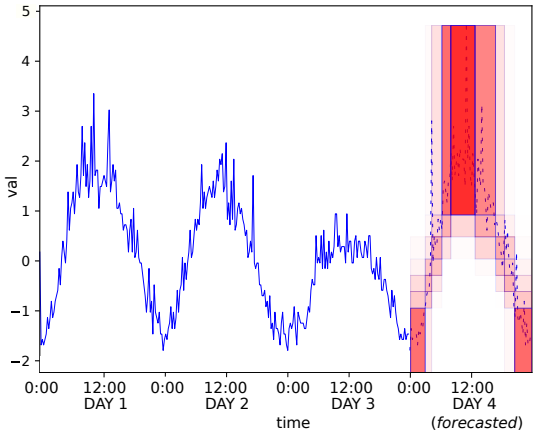


Figure 6: Un exemple de prévision probabiliste. En bleu, les trois jours utilisés pour alimenter le classificateur, en rouge la distribution de probabilités associée. Dans ce cas particulier, la plupart des valeurs bleues se trouvent dans les zones ayant de fortes valeurs de probabilités.

Table of Contents

INTRODUCTION

Undoubtedly, computing, software and web applications are part of our everyday life. It is hard to imagine a world without the Internet: we live in a hyper-connected era, full of social networks, shopping websites, and even critical services (such as health care systems, government systems, military and security systems, finance systems, *etc.*) are hosted on thousands of servers across the globe. For example, it is estimated that the Google company has approximately 1 million servers to support its services. [4] If there is not enough severs deployed, the applications become slow and difficult to use. On the contrary, if there are too many servers, performances are better, but the costs incurred [5] for managing superfluous servers are not negligible and should be reduced in the best-case scenario. The maintenance of the infrastructure is done by companies or institutions providing digital services, and is not an easy task. Servers are numerous, applications more and more complex, technologies are evolving fast and, furthermore, the number of users is constantly increasing. The challenge is to offer the best service for the public at any time and under any circumstances.

Capacity planning is a major preoccupation for companies of the digital economy in order to address this challenge. Capacity planning is the way an IT service manages, updates, adds or removes physical/virtualised servers from its infrastructure. A well-made capacity planning helps to reduce operational costs, and improves the quality of provided services. Capacity planning tasks and objectives highly depend on the type of infrastructure (*e.g.* bare metal servers or cloud-enabled infrastructure), the type of application hosted (*e.g.* critical applications or more relaxed environment), the goals of the company and the budget allocated. In this thesis, we are not directly interested in optimising resource usage but on forecasting the demand to have more time to deploy in advance an optimal infrastructure.

The various services offered by Orange (*e.g.* general public internet access, 4G network, business to business applications, *etc.*) are hosted on thousands of servers, and the quality of service is paramount for proposing the best experience to the clients. This historic French telephone operator is one of the major brands in the telecommunications field. In 2019, it was the thirteenth-largest company in the world in this field, with a global turnover of 43.7 billion euros. Orange Group operates in 29 countries around the world and focuses its business on telephony and on the sale of internet access – with a big concern on the quality of the services offered.

At Orange, project owners, managers and experts often apply manual capacity planning guidelines to manage their infrastructure. This approach is cumbersome and to prevent from any business interruptions, the server capacity is often overestimated. The more complex the system, the more difficult for engineers to set it up manually. Thus, there are more and more interest in developing tools to support engineers on part of the capacity planning problem. Bodik et *al.* [Bod+09] show that data analytics on datacenters Key Performance Indicators (KPI, *e.g.*, capacity planning, number of users, RAM) may be used to improve capacity planning. Indeed, the data generated by servers or user activity could be valuable sources of

4. See estimate: `https://www.quora.com/How-many-servers-does-Google-have-1`
5. In terms of money spent and in terms of energy used, for example.

information. The KPI data is often collected in the form of time series datasets (numerical data ordered by time). KPI are classified in two categories: the functional KPI measures the level of activity of service; they depend on hosted services and user behaviour (the number of people on a given website) but is not related to hardware, whereas the technical KPI measure resource consumption in the infrastructure itself (evolution of CPU use). An example of a functional KPI may be found in Fig. 7: each wave represents one day in the activity of the Orange Money project.

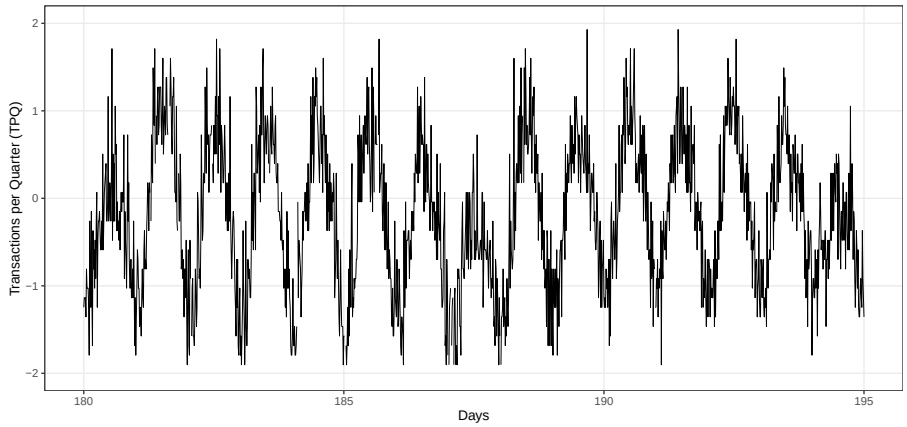


Figure 7: Example of functional metric transactions per quarter for one Orange project (Orange Money).

One particular data analytic task that may help resource managers in the daily activity is to forecast the evolution of the KPI. Having a view on the future evolution of a system could be very valuable for systems managers. Important applications are to forecast future consumer behaviour in marketing, as well as also servers load for popular applications. The more accurate are forecasted the KPI, the more informed will be the management decisions regarding capacity planning. The objective is to provide accurate pieces of information to anticipate resources needs. Reliable forecasts enable for example to take decision of which system to upgrade, in order to seamlessly operate the maintenance.

Forecasting the evolution of a temporal process is a critical research topic with many challenging applications. This modelling approach is particularly useful when little knowledge is available on the underlying data generating process or when there is no satisfactory explanatory model that relates the prediction variable to other explanatory variables [Zha03]. In addition, having information on future evolution of a given system is important for making useful decisions at the right time, regardless of the application studied.

The most popular time series forecasting methods come from statistics, and are the following: Autoregressive [Aka98], ARIMA [Box+15] or Holt-Winters [Win60]. These methods build a mathematical model of time series. This model is then applied on new time series to predict future values. Recently, in many approaches coming from machine learning [BTL12], new methods emerged. Neural networks have for example been used extensively [ZQ05], often in conjunction with ARIMA statistical models [Zha03]. Some probabilistic forecasting toolsets have also been proposed (Prophet [TL18] or GluonTS [Ale+19]) and they often propose a way of configuring complex models more automatically.

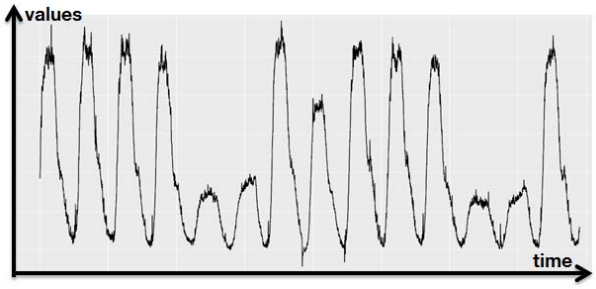


Figure 8: Seasonal time series example borrowed from [Hyn11]. Two weeks of the internet traffic data (in bits) from a private ISP with centres in 11 European cities. The whole data corresponds to a transatlantic link and was collected from 6:57 a.m. on 7 June to 11:17 on 31 July 2005. The time series is obviously seasonal, but the assumption of having one unique periodic pattern seems not suitable in this case.

Many time series, especially those related to human activities or natural phenomena, exhibit seasonality. This means that there is some periodic regularity in the values of time series. Human activity behaviours often exhibit daily and weekly seasonality. This is particularly true in Orange capacity planning datasets. As seen in Fig. 7 above, the daily season is very clear: every day, the activity grows up until noon and then it decreases. It is due to the application purpose, which is to transfer money between users, and there are naturally more transfers at daytime than at night-time. Most transfers are made between populations at the same time zones.

Knowing or discovering that a time series is seasonal is a valuable information for forecasts, as it can restrict the search space for the mathematical model of the time series. The classical approach to deal with seasonality is called STL [Cle+90], it builds a model while taking into account three components: seasonality, trends (long-term evolution of the time series: increase or decrease) and residual (deviation from trends and seasonality). It assumes that the time series exhibits a single periodic behaviour (ex: daily periodicity of temperatures).

Using time series seasonality in the capacity planning process could enable the study of the seasons (*e.g.* detecting several season types such as 'normal season' or 'season that will cause a problem'). Then, being able to forecast several future seasons of a time series and discovering that one of them is a problematic season can allow managers to act proactively and to anticipate the problem.

Holt-Winters and ARIMA models have been extended to deal with seasonality. But they also assume a single and clear periodic behaviour. We distinguish here periodicity and seasonality. One concern with the method concerning periodic behaviour is that it is often violated in practical cases. Consider the time series in Fig. 8, which shows an internet traffic measurement for two weeks. While there is indeed a daily periodicity, there are two types of daily patterns: weekday patterns and weekend patterns. This cannot be well captured by the STL framework and the associated statistical methods.

This thesis studies this problem of seasonal time series forecasting in the context of capacity planning at Orange. The objective is to exploit the idea of having different types of season to make the forecast accurate. Technically speaking, time series are split in multiple seasons, similar seasons are grouped together in order to identify typical seasons, and then the time series and groups are used to learn how to predict the next season's group. Finally, a time series that best represents the predicted groups yields the

forecast. This original way of doing tie series forecasting is the sketch of the generic framework investigated in this thesis to state whether it can help to improve capacity planning procedures.

Contributions and outlines

In this thesis, a framework dedicated to seasonal time series is proposed. The framework is based on the combination of clustering and classification algorithms for producing both deterministic and probabilistic forecasts, and three instantiations of the framework are studied.

The contributions of this thesis are the following.

— **Chap. 1** introduces the context in which this thesis will take place. More especially, the notions of performances, capacity planning and time series forecasts for capacity planning are discussed.

— **Chap. 2** gives a state of the art of time series forecasting and all the formal definitions needed to approach the rest of the thesis.

— **Chap. 3** introduces *a framework for seasonal time series forecasting*. The framework proposed aims at producing the seasonal forecast in *only one shot for the entire season*. Only concepts, formalisation and general ideas are introduced. All the notions used with the framework are defined, because the three implementations proposed all follow the same principles and same formalism. The Chap. 3 first formalises the problem of seasonal time series forecasting, then presents the learning steps of the model, then the forecasting step and finally introduces the experimental setup followed during the entire thesis.

— Then, three implementations of the framework are depicted.

  — **Chap. 4** introduces the baseline deterministic implementation. It uses *K-means clustering algorithms* alongside with *Markov Chains* for creating *deterministic forecasts*. The choice of K-means and Markov Models is natural because they are both well known by the research community and can give a first assessment on the interest of the approach. This first implementation was introduced in the publication [Lev+18].

  — **Chap. 5** introduces the second deterministic implementation. The baseline framework is extended with the use of *several clustering algorithms* (K-means, K-shape, GAK and MODL), and the Markov Chains are replaced by *classifiers* (logistic regression, naive-bayes, decision trees and random forests). This second implementation was introduced in the publication [Lev+19].

  — **Chap. 6** introduces a last implementation based on *probabilistic coclustering and classifiers* for producing *probabilistic forecasts*. Experimenting with various algorithms and various combinations is paramount, especially to introduce a *portfolio framework* that would adapt to each data topology in a data-driven way. Probabilistic forecasting is interesting to handle uncertainty, like in capacity planning problems.

— The practical use of those frameworks is illustrated on one main use case in **Chap. 7**, using one dataset provided by Orange company, collected on servers deployed for the Orange Money project. This use case is used to evaluate qualitatively the various parts of the framework (*e.g.* clustering, forecasting, *etc.*)

Chapter 1

Data driven capacity planning

Applications, and more especially online services, are often under pressure while more and more people are adopting them. The performances of servers and infrastructure are closely related to the *workload*. The more people using a service, the more requests and computations for the infrastructure, thus the higher the workload. The workload is not only related to CPU usage, but also to other technical features such as RAM usage, network capacity or IO access, *etc.* There is a complex relation between the number of people using a service and the overall use of servers. This relation is not straightforward and managing infrastructure efficiently according to the load is not an easy problem. The management of servers is made using *capacity planning* techniques, that help infrastructure managers in the decision-making process.

In this chapter, the goal is to clarify the technical and industrial notions used in this thesis, and to motivate choices made for a focused research direction. Concepts such as performances, capacity planning and forecasting algorithm are introduced.

Section 1.1 explicits the problems behind infrastructure maintenance, and why it is necessary to have good capacity plans for a company. Capacity planning is introduced in the Sec. 1.2, which also exhibits the main challenges of capacity planning and gives details about the technical challenges of such solutions. Section 1.3 motivates the data driven approach introduced in the thesis. Section 1.4 finally gives an overview of lots of possible solutions that could have been explored during the thesis and exhibits the selected direction.

1.1 Context

In this section, the problems faced and solutions adopted by companies while dealing with a massive infrastructure are introduced. Sec. 1.1.1 gives an idea of some major problems common in infrastructure management. Then, Sec. 1.1.2 gives an overview of how the performances of infrastructure are assessed with tests. Test data may also be used to configure the capacity of infrastructure, and therefore are useable for capacity planning tasks.

1.1.1 Problems of infrastructure maintenance

During its lifecycle, an application will hopefully meet a certain success and the number of users will gradually increase over time. Servers and infrastructure are configured to support a maximum number of users simultaneously, and may show long response time or even crash over this limit. When a service is under a high workload, it is important to ensure that there will be no operating problems (such as machines overutilized, or not enough machines to answer user requests, *etc.*) [All08].

Problems for consumers

As an example, think about the Christmas period, where a lot of people would like to buy presents on the internet. Imagine you are one of these buyers. And imagine that, after hours of browsing and seeking on your favourite website, you have found the best gift card for your old aunt. It is now time to pay, you have finally entered the three security digits of your faithful credit card, and suddenly: the website is not answering anymore. After a while, a pop-up eventually appears, showing that the website is currently under maintenance, due to an unpredicted number of people using it at the same time (well, it's Christmas for everyone...). You still have questions on your mind: is your payment finalised? Will you receive this gift – will Aunty be happy at Christmas Day? Those unanswered questions mostly cause frustration and anxiety, which can make you churn (change of website/provider). Meanwhile, all engineers are working on the rush to re-establish the services: infrastructure has to be fixed, the sooner the better. Those kinds of situations are often stressful for technical crews, and technicians only wished to have taken their holidays right before Christmas...

Now imagine that you are a fan of the famous 'Game of Thrones' series. Imagine that a website is actually providing a VOD service which includes access to 'Game of Thrones'. As you know, this show is very popular, and because you love it, you are eager to watch the new episodes as soon as possible so you can debrief with your friends and colleagues the next day at a coffee break. If all of your (many) friends and *a fortiori* all the fans of this show think this way, and if there is only one provider for the show at the television, there will be a massive peak of people that will connect at the very same moment using their TV and laptops to watch the episode. And if the infrastructure is not well calibrated, it could have consequences on the quality of the diffusion. As a fan, you will probably not be happy with lags and outages occurring between two crucial scenes! Ultimately, you could even show your unhappiness on social networks of any kind. The project owner does not like such a bad buzz about his project on the internet. It is known that it can be long and costly to order, install and configure new servers (several months and thousands of dollars). This is why she/he may be interested to have a more precise idea of when to buy new servers – when is the next season of 'Game of Thrones'?

Problems for companies

Undeniably, for the manager of operational teams and its infrastructure engineers, one of the worst scenarios that could append is a problem that would put the service down. It could have dramatic impacts on any company. Indeed, people are not buying things on websites that are down. In many companies, the cost of one hour of downtime is estimated and it is often more than hundreds of thousands dollars. For example, during the famous Amazon Prime Days in 2018, the shopping website was down for more than one hour and it has cost to the company several millions [1].

Always having oversized servers could be a good solution. Although, companies usually don't make this kind of bet, because larger and faster servers are more expensive. Every resource not consumed because of a low activity is wasted and that waste has a cost. P. Bodik models peaks and workload in [Bod+10] and uses the analogy between high user peaks and earthquakes to show that it is not reasonable to systematically over estimate capacity: it is way too expensive to have very strong buildings ready to

1. See: `https://bit.ly/2lIIhBI`, a webpage from Businessinsider that agues that the one-hour downtime cost 100 million USD to Amazon.

face every situation (even the one pretty improbable, say, disastrous earthquakes in a non-volcanic area), in analogy with huge, oversized servers, ready to welcome thousands of users. But serving a website with too few servers, which are not powerful enough to answer in a reasonable delay to user requests, is not a good scenario either. Users are not inclined to wait too long until results to their queries are shown. As stated in [Nie94], a user starts noticing delays after 1 second waiting time and lose attention after more than 10 secondes. Competition is hard and buyers can change their mind (and their providers) in no time.

Some cloud management softwares are able to automatically balance the use of servers so that there is less underused servers. The applications hosted are automatically transferred to unused servers depending on the needs. It allows doing some overallocation of resources [XSC12] where the software hypervisor takes profit of inactivity in some applications to reduce the overall number of servers used. Although, it can be dangerous if all the applications become very active at the same moment.

Using cloud infrastructure is more flexible: resources might be adjusted on demand, sometimes even automatically. But even if using cloud infrastructure allows to automatically adjust the architecture of an application by making its resources growing at peaks, it cannot grow indefinitely, and it has an impact on the application price. In theory, companies can pay as they grow, as long as they are able to afford costs induced by improvements.

1.1.2 Performances and tests

Computers, smartphones and tablets with an internet connection are becoming cheaper day after day, and every aspect of the customers modern lives is web-connected. It means that there are more devices browsing web services. This revolution of use has started more than a decade ago. Internet giants have then been developing technologies to cope with the always-increasing number of online consumers. Their goal is to maintain *good performances*. This section is dedicated to the presentation of the infrastructure testing and the various techniques used.

What is infrastructure performance?

Tech leaders have a great interest in the performances of applications. There exists several ways of assessing them. One of them is to *test* the applications and infrastructure. Infrastructure is being tested to ensure that it is correctly calibrated, to check the quality and robustness of the application. The goal is to validate that it could be deployed to the real world in front of real people.

Testers are specialists who bench and stress the infrastructure to ensure that they behave as expected under pressure. They use tools to simulate users browsing a website, create complex test scenarios to ensure that servers can handle in typical conditions. Tests are usually executed on a complete replica of the infrastructure, because it is not recommended to stress the production servers as they will suffer and might crash during tests, causing an unwanted unavailability. Test data is often gathered and analysed offline, because test procedures are to be distinguished of monitoring activities. In the latest, the aim is to fix problems in real time rather than preparing the infrastructure for any situation.

The *load tests* give the light on what can be addressed on the application infrastructure, regarding the customers' usage. With *limit tests*, testers try to find the limits (in terms of data flow, number of users, *etc.*) of the application. Testers could also benchmark the platforms during hours with *long tests*, to assess whether infrastructure can last long when stressed.

Part of the tester's job is to define *test scenarios*. Plausible situations which can occur in real life are simulated. Behaviour of potential customers that browse the website in a very specific way is modelled, reproduced and ultimately replayed. Scenarios can be very long and complex, can be randomised to introduce variability in the process, and are run using sometimes thousands of users at the same time. Of course, being exhaustive in scenarios is not possible: some users will find a way to use the website not exactly as planned.

One of the first indicators studied during analysis of tests results is the *response time*. It represents the time spent by the servers and infrastructure to reply to a user's query and to send responses (which can be HTML, or of any other type). At a second hand, the *overall availability* of the servers represents the duration where the servers operate normally. System with a high availability label will theoretically be up 99.99% of the time [GS91]. The *robustness* of a server represents the capacity of the server not to bug and not to lose data. Finally, the *scalability* of the systems defines how well the servers and infrastructure react to a sudden rise of activity, and specifies the response times for different numbers of users.

How to achieve good performances?

First of all, it is important that the infrastructure containing the application is *well dimensioned* [Tia09]. Dimensions of infrastructure is linked both to the number of servers available (that is highly dependant on the architecture and use of the application), but also to the quality of the servers available (servers with lots of CPUs and RAM are more efficient).

Secondly, the *configuration* of the servers is very important, as a bad configuration can dramatically impact the behaviour of servers under high load [Eil+06]. A well-dimensioned infrastructure but with a poor configuration is sometimes unable to handle several users simultaneously, because of the overconsumption of resources that are being wasted. On the other hand, the reverse is also true, and a small infrastructure but very well tuned/configured may handle more load.

Finally, *data flows between servers* play an important role [Bal+11]. Indeed, the footprint of data is huge: network links have to be adapted in the infrastructure, servers have to be ready to process the massive incoming data flows (for storage, running algorithms, *etc.*). Data centre architects should also take into account that data flows might change over time, according to many factors (for example, launch of a new product, sudden increase in the interest of one item, *etc.*).

1.1.3 Wrap up

The necessity of having the best infrastructure management systems to prevent problems on services is not to be proven. Such systems will be able to ensure that servers are enough and well calibrated, that users don't experience any latency, that networks are operating fine... And any problem will rise notifications which will help infrastructure managers to fix problems, proactively if possible. A good start for configuring infrastructure is performance testing, which is useful to configure brand new servers before production.

1.2 Capacity planning

Capacity planning is the way servers are managed once they are on set. The goal is to maintain performances on the long term and not to ensure at one given instant in time that performances are sufficient.

In this section, the definition of capacity planning is given on Sec. 1.2.1. A view on how capacity planning may be applied on two different infrastructure architectures is depicted on Sec. 1.2.2.

1.2.1 Definition

Capacity Planning (CP) consists in managing running infrastructure and servers for the purpose of reducing latency, enhancing the overall service quality and avoiding problems on servers during production. The goal is to determine the production capacity needed to meet changing demands. Some concrete examples of CP tasks are to find the right number of servers for an application, to find the size of some network links, to detect poor configuration on servers, *etc.*

Data from tests is often used to grade capacity models. For example, tests can reveal that systems cannot go above 10 transactions per seconds and 10 people connected to the database is the maximum. That information is to be used to enhance capacity plans. Those two processes are independent but infrastructure is often tested before creating capacity plans.

CP relies on the study of a particular set of monitored information. They assess the global health of a given system at a certain instant in time. Capacity is a very wide concept: network bandwidth, hard drive or CPU use, number of transactions per seconds, *etc.* CP can be done in various ways, depending on the target infrastructure and whether it is cloud enabled or not.

1.2.2 Capacity planning for two types of infrastructure

Bare metal and legacy infrastructure

In the early ages of computer science, services were deployed on huge *bare metal servers* [Moo+05]. Systems maintenance was mainly manual as the servers had to be ordered and installed by humans. If a threshold of users browsing a service was reached, it was difficult to react rapidly and anticipation was an important asset to ensure the stability of a service. On another hand, fewer people were browsing services at this time, which made the scaling process less dangerous.

CP analysts were collecting data, sometimes manually on legacy monitoring systems, and taking actions about anticipated flaws. They were asking questions such as 'will my system support the load for the next month?' and answering them by manipulating the datasets and creating experts models [Sne02].

Models were condensed of human expertise, mostly because applications managers knew very well their behaviour. Experts were aware that every year, during certain religious events, people traditionally has a different use of the application. For example, it is known that people tend to buy more goods than usual during a few weeks before Christmas. It has to be anticipated for systems hosting merchant websites not to crash.

Finally, capacity reports were produced every day/week to keep a track of every action and decision and have an overall view of the system health.

Legacy infrastructure and systems are still present in companies, for different reasons. At Orange, some critical systems have been up and running for decades, and are sometimes too critical to be shut down. For systems that run telecommunications services, it is indeed tricky to even think about a possible downtime. Those servers are proof-tested, and some mainframes used to host those old (but critical) applications sometimes could cost thousands or even millions of euros.

Cloud infrastructure

Nowadays, a lot of companies are using fully *cloud-enabled infrastructure*. The management of those is easier. Adding or deleting new machines is doable in minutes from a web interface for most cloud providers, whereas it could take months with traditional servers that should be installed manually by human operators. It is simple to reduce the anticipation time needed to scale up a service, and decisions can be taken in real time. There is less focus on the solid infrastructure than before, because cloud infrastructure is offering numerous interesting properties by design (*e.g.* replicated services, automatic virtual machines backups, *etc.* – see [MG+11]).

The maintenance of the real infrastructure is often on the cloud provider side, which is not necessarily the one deploying the application. More broadly, the objective of capacity planners is still to anticipate the future activity of servers and application but the horizon is shorter because everything goes faster.

Some advanced CP software sometimes manage the infrastructure in autonomy, by using some mechanisms such as *auto scaling* [MH11]. This technique allows the software that manages the cloud hardware to pop new virtual machines automatically if it feels that the performances are too low, based on some parameters defined by engineers. It also enables the optimisation of the number of servers in real time: if one server is not used at all for a certain amount of time, it could be removed and re-created when the need arises [DWS12].

Migrating legacy solutions to the cloud can take a lot of resources. The transition is not always fast. Orange already owns many bare metal infrastructure that are easier to exploit than clouds infrastructure. Teams are already trained and used to those paradigms, and more efficient. Therefore, most of the cloud CP techniques (auto scalability, real-time placement of virtual machines, *etc.*) are easier to implement in companies that are less dependant to massive, already in place bare metal infrastructure.

1.2.3 Wrap up

In this section, the concept of CP has been introduced, and its application to two different types of infrastructure explained (namely bare metal and cloud infrastructure). For those two types of infrastructure, the challenge is to anticipate the need for more resources in order to ease their management.

This thesis aims in developing some machine learning (ML) algorithms for improving the CP at Orange. A special interest is shown about forecasting algorithms. Next Sec. 1.3 gives a view on a possible data-driven CP tool for Orange needs.

1.3 Toward data-driven capacity planning tool

The solution to a fully automated and complete infrastructure management software is certainly not straightforward. Many different directions might be followed in the path of improving the CP toolset at

Orange, and more especially for the OM project. CP solutions are usually focused on the management of servers (auto-scaling, technical management of infrastructure, real time peak detection, *etc.*). Although, the use of time series and particularly forecasted values is a real support that can be used with CP tools to improve their accuracy and even their scope of application. That is why this thesis focus will be entirely on the *TS forecasting* and its data science aspects. Less emphasis will therefore be made on technical tools. An industrial use of the forecasting framework introduced in the thesis is mentioned in the ultimate Chap. 7 which is dedicated to a use case for the OM project.

In this section, the interest of Orange for developing data driven CP tools is shown. In a first Sec. 1.3.1, the industrial problem and the datasets that will be used for case study are presented. Then, an exploration of the data is made in Sec. 1.3.2. The hypothesis made on the datasets manipulated are shown in Sec. 1.3.3. The interest of data driven algorithms for CP task is explained in next Sec. 1.4.

1.3.1 Industrial problem and datasets

For this thesis, data from a fast-growing Orange asset was provided: Orange Money project (OM). [2] This latest was established in late 2008. It aims in providing an easy access to bank transfers for African customers. The bank exchange system is hosts by a large Orange infrastructure and is used by more than 31 million customers across all of Africa and Europe.

Technically speaking, the application is present on 14 African countries, and 162 servers were in place between 2014 and mid-2018; some of them were hosted in France and others directly on the targeted countries in Africa.

The datasets were collected from this infrastructure using Nagios supervisor [Bar08]. It collected technical and operational metrics every five minutes:

— *Technical data*: data concerning the server's performance such as percentages of CPU/memory use across all servers,

— *Functional data*: data concerning users browsing the service; number of financial transactions every minute, number of people on a website, *etc.*

Examples of the data collected on (OM) servers can be found in Fig. 1.1. OM is providing various datasets, coming from multiple sources: technical sources (practical field reality such as evolution of servers' technical Key Performance Indicators - KPIs - over time: CPU, MEM, ...) or operational sources (evolution of business KPIs over time, often in relation with money and/or users).

For the OM project, the need of having a very well-made capacity plan is important. Indeed, this application is crucial as it concerns money transfers between individuals. Also, the application knows a very high growth rate, which means that more people are adopting it every day and it impacts the server's usage. Finally, the relatively long-term forecast required by CP applications is challenging. It is more useful to have a view on the long-term evolution of OM applications rather than one unique point forecast for next day maximum value.

Until now, the raw data was used by OM managers to feed massive excel spreadsheets, where the data was displayed and handmade forecasts being computed. Those tasks were time consuming: the growth rate (workload and augmentations of this latest) of two countries is always different, and some rules had to be inferred by experts to create models that were more suited. This method was also less resilient to

2. See: `https://orangemoney.orange.fr/`

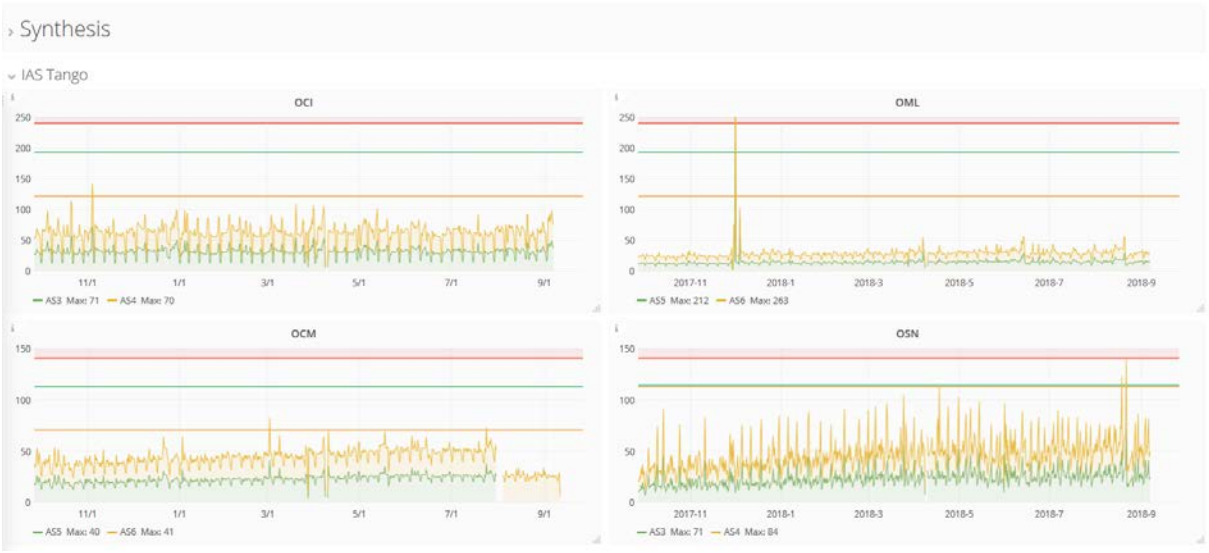


Figure 1.1: Examples on how OM project uses the raw data coming from servers to visualise with dashboards the good health of systems. This data shows the number of people using the service on a daily basis on four different African countries. Upper right graph exhibits one data peak. Lower right graph exhibits one increasing trend.

fast change in the trends, because models were updated weekly. Such approach is mostly ad hoc, and do not scale to a growing number of deployment countries and use cases.

Because of those limitations and prior to this thesis, project managers decided to create a tool to support them by forecasting the evolution of some variables of interest in the data (transaction per quarters, CPU usage, RAM usage, *etc.*). This tool is called PerForecast [LML18]. It consists in an interface that allows users to upload data then to process the data using some state-of-the-art forecasting models and finally selects the best model, all automatically. This approach enables the use of complex models for non-initiate and non-statistician managers for the conception of capacity plans.

1.3.2 Data exploration

Prior to the experiments with the framework, an analysis of the data at hand was conducted. Two of the most emblematic pieces of data manipulated for the experiments are the Transactions Per Quarters (TPQ: transactions every fifteen minutes) and the CPU datasets. In Fig. 1.2 are provided graphs and analysis about one particular TPQ metric, which is an aggregation of all the TPQ metric monitored on all the 162 servers.

The methodology presented is a classic of TS analysis. It aims at verifying if the TS at hand is stationary, seasonal, shows a trend, *etc.* The analysis is conducted on the last 1000 points of the TS. There are 96 points per day. The methodology followed on this particular TPQ TS was repeated as many times as necessary for further analysis of other TS of the dataset.

— The Subfig. (a) displays the raw data in blue and a smoothed version of it in red. The blue line below the data is the trend observed, which in this case is practically non-existent, and stable.

— The Subfig. (b) displays the autocorrelation graph. The latest show a very clear, almost perfectly sinusoidal correlation patterns between points, which suggest a strong seasonality; it makes sense with the plot of the data and also with the *a priori* knowledge of OM application which is driven by the daily human activity. Two upper peaks of the sinusoid are separated by approximately 96 points in the autocorrelation graph.

— The Subfig. (c) displays the partial autocorrelation graph. It gives the partial correlation of the TS with its own lagged values. The stabilisation of the plot is observed at lag 5.

— The Subfig. (d) is a QQ plot [GW68]. The QQ plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a Normal or exponential [For15]. This QQ plot is light-tailed, meaning that the data is not generated by a fully Gaussian process.

— The Subfig. (e) is a histogram which displays the repartition of the values observed in the raw TS. Consistency might be found between the QQ plot and this histogram in the extremities of the two plots. There is a regular distribution of the points for medium values but a rather low number of extreme values.

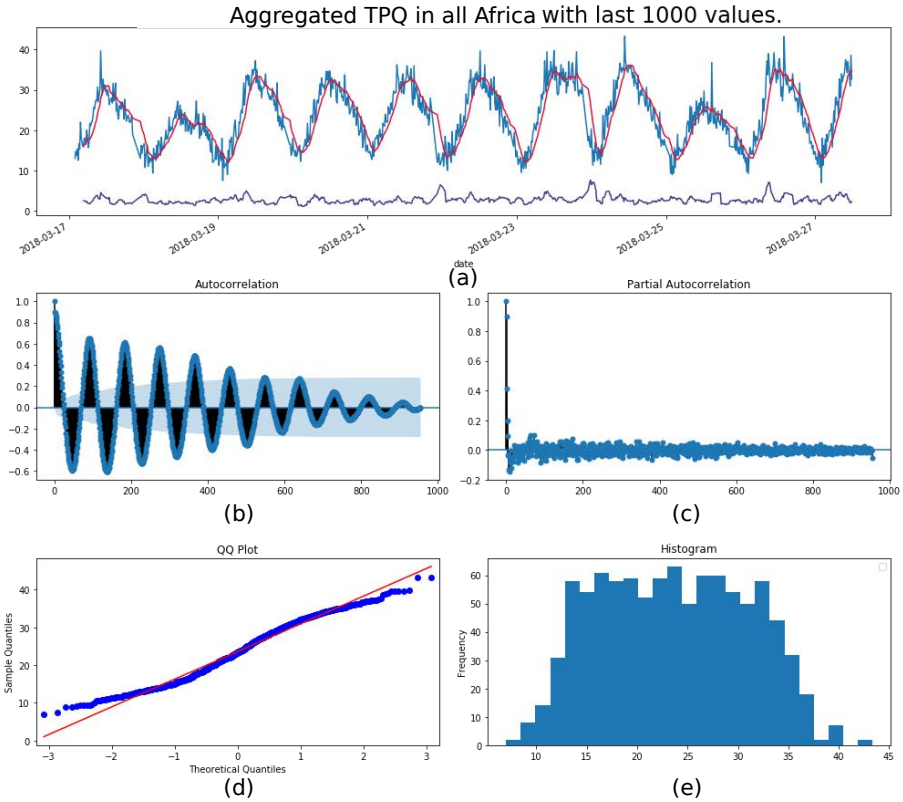


Figure 1.2: Analysis of TS at hand: the situation for the aggregated metric 'Aggregated TPQ in all Africa'.

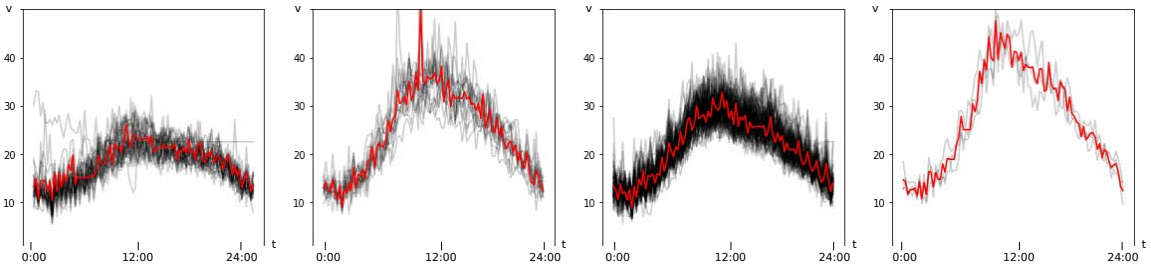


Figure 1.3: Analysis of transactions per second TS: display clusters found by a K-means clustering algorithms with 4 clusters. t is time, v is values.

A second step of this analysis was to assess the diversity of clusters in the data. For this task, one K-means clustering algorithms with four clusters was applied to the seasons that have been previously split from each other. Fig. 1.3 exhibits the results and 4 different types of days are clearly identified. First cluster at the left of the figure represents the low activity day. Even at noon there is not that much activity and the maximum value is around 30. On the other side, second cluster and last cluster represent the high activity days; the last cluster is filled with very few days, most probably because it represents the very exceptional days that did not occur often on the learning data. Finally, the third cluster represents the average day, and is filled with more TS than other clusters.

1.3.3 Nature of the data and hypothesis

The analysis of the data in the previous section exhibits some interesting characteristics: seasonality, correlation between points, etc. In all the remaining of this thesis, several hypotheses are made on the datasets found in the CP problematic. They are important because they frame the scope of the developed algorithms. The hypotheses made on the data are listed below.

— $\mathcal{H}_0$: time series are regularly sampled, there is no missing data and the data collection process is reliable;

— $\mathcal{H}_1$: time series are diverse by nature and many different types of data coexist;

— $\mathcal{H}_2$: time series are seasonal;

— $\mathcal{H}_3$: there are no trends in the time series;

— $\mathcal{H}_4$: season length is known by the forecaster prior to forecasting;

— $\mathcal{H}_5$: there exists different types of season, each type being different from each other and representing a collection of similar seasons;

— $\mathcal{H}_6$: it is possible to guess one season type based on the knowledge of the few past seasons observed, because there is a correlation between successive types of seasons.

Hypothesis $\mathcal{H}_1$ is important in the context of CP because sources of data are multiple: they can for example be technical or functional, those two data types being fundamentally different. Also, most of the CP datasets at hand are seasonal, thus the Hypothesis $\mathcal{H}_2$. Hypothesis $\mathcal{H}_3$ is weak because it is easy to remove trends from time series (see Sec. 2.1.1). Finally, in the CP datasets, it has often been observed

two low-profile days occurring after five normal days (the occurrence of weekends being a probable cause). Generalising this observation led to the Hypothesis $\mathcal{H}_6$ that introduces the correlations between successive types of seasons.

1.4 Data driven algorithms

Sec. 1.3.1 showed that the sources of data for the OM project are quite diverse: money transfer information as well as server status data. *Data driven* algorithms are interesting because they mostly rely on the data itself, and do not require complex parameter setting. It means that their configuration is easier and also that they can be applied on various data types seamlessly. This is why they are particularly interesting for CP tasks, where the data is extremely diverse by nature and where the need to have automatic and reliable algorithms is strong.

Sec. 1.4.1 explains how forecasting algorithms might be used by capacity planners to manage infrastructure. After that, a diagram that exhibits research directions considered for this thesis is shown, and choices explained in Sec. 1.4.2.

1.4.1 Forecasts and capacity planning

Forecasting algorithms are common in the ML field. They are used to predict the future values taken by one given data source. It is interesting in the context of CP. The study of data coming from the servers, and estimations on how they can evolve could give infrastructure manager crucial information about potential evolution of systems, and thus help to proactively maintain them.

Forecasts have for example been used to maintain the internet backbone traffic [Pap+05b]. Authors use a linear TS models (ARIMA) to predict when and where link additions/upgrades have to take place in an Internet protocol backbone network. The identification of the long-term trend was important in this work and authors used the wavelet multiresolution analysis to do so.

Still in the context of network capacity, [Pap+05a] use spectrum analysis and linear models (ARIMA) for creating short-term forecasts of the WiFi use in a campus. Authors insist that short-term forecasting (*e.g.*, next minute) can assist in designing more energy-efficient clients, whereas long-term forecasting is essential for CP and understanding the evolution of the wireless traffic and networks.

More recently and to manage cloud infrastructure in a better way, linear models have been used by [RDG11; VKJ15] as a support for enhancing auto-scaling tools, where the need to predict future workload based on a limited horizon for adjusting resources allocated to users ahead-of-time is paramount. [CDM10] use some pattern matching to identifying resource usage patterns that have occurred in the past, and to forecast what pattern will be next.

Having an exhaustive view on how can evolve one unique data source is a challenge. Taking into account several sources of data (*e.g.* CPU, RAM, disk, *etc.*) in one unique forecasting algorithm is another yet harder challenge, addressed by [Ye+14] where authors propose a multivariate prediction model for quality of service. However, many bare metal servers are still in use at Orange, and most of them are monitored by capacity planners that often rely on the study of one unique technical metric (such as CPUs or Memory). Then, there exist an interest in *forecasting each variable independently* for improving CP algorithms.

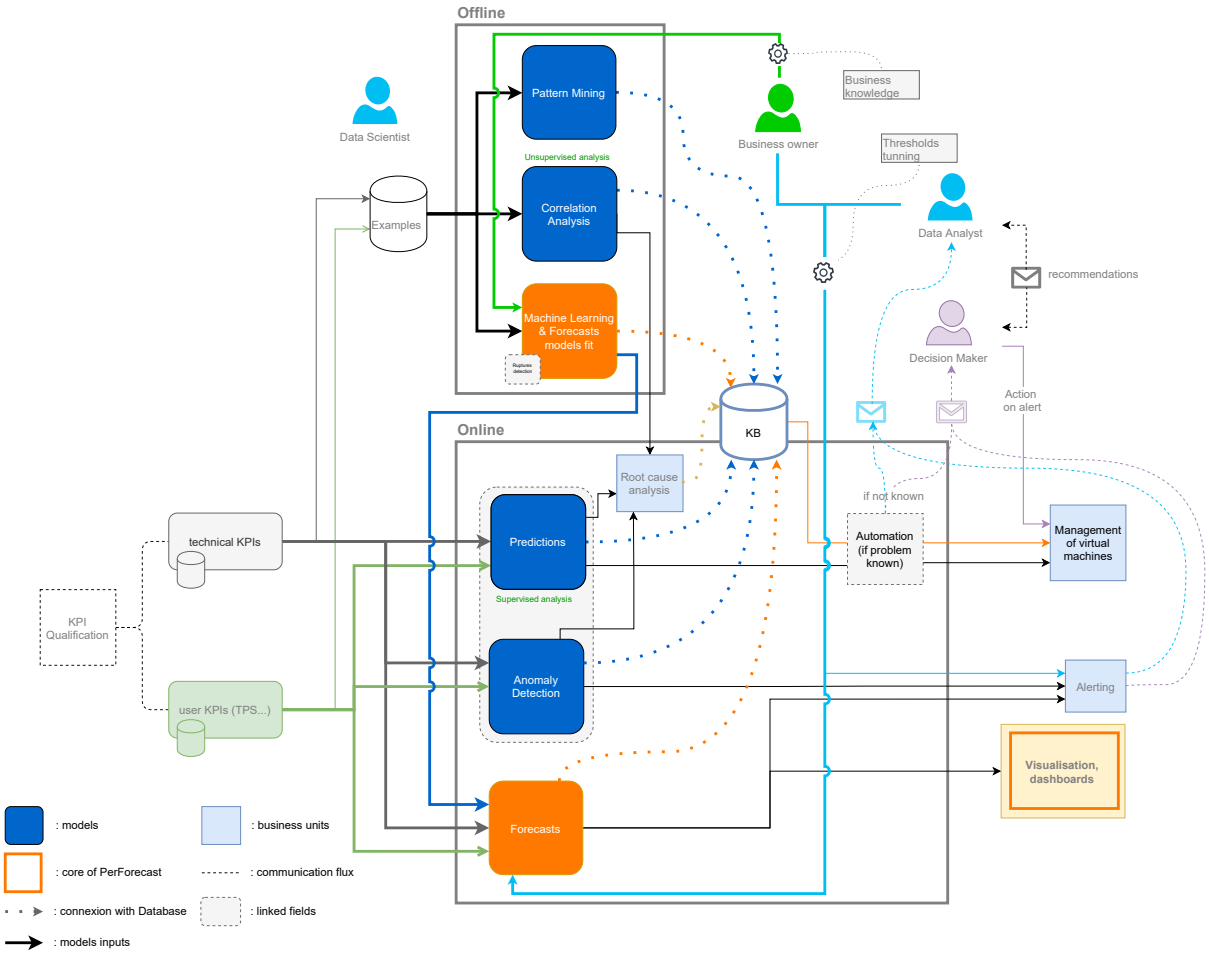


Figure 1.4: Functional diagram of one hypothetical and ideal ML toolkit for improving CP.

The configuration of forecasting algorithm is not a trivial task and data scientists are often required to find good parameters. Bad parameters could lead to wrong forecasts, and the latest are not very useful. The need for more out-of-the-box, automatic forecasting models for CP is explicit. A state of the art about forecasting algorithm will be proposed in Chap. 2.

1.4.2 What could be done?

Fig. 1.4 shows a big picture of all the systems entangled together in one hypothetical, ideal and complete data science tool for enhancing CP. It explicit inputs and possible outputs. A detailed explanation of this figure is given below.

This figure is decomposed in various boxes which represent some data science techniques that could be used for improving CP toolbox. They are spread in two categories which are 'offline' and 'online'. 'Online' treatments are done with data which is continuously incoming from the sensors. 'Offline' treatments are done with data already collected.

Inputs

The input sources, that may be found in the left of the figure in light grey and light green, are at the basis of every data science project. Obviously, the data collected directly on servers and infrastructure is the first source to be used.

Human actors, represented with people in the figure, can also have relations with ML models by configuring parameters and giving insights about business knowledge. Indeed, those insights are not necessarily shown in a comprehensive way on the raw dataset.

Finally, some ML models might output results that will be used by other algorithms. The orange box 'Machine learning & Forecasts models fit' in the upper, offline section of the figure is linked to the orange 'Forecast' box in the online section. Indeed, models might be trained offline and used afterward online.

A knowledge database (KB) is represented in the centre of the figure, fed by all the algorithms in place in the system. It can be used to optimise servers, tune configurations, but also to store the computations and use them afterwards.

Offline and online paradigms

Two main categories for algorithms are represented: the ones that are trained offline, and the ones that are trained and then used online.

Offline

Offline algorithms are designed to be trained with complete datasets and reused afterwards once trained. Once trained, models can be used instantly.

In the offline section which is at the upper part of the figure, the three following models have been identified:

— Pattern Mining models: used to extract patterns from the data, to find relevant pieces of TS that can be used to trigger alarms, predict system failure on the basis of pattern succession... Real world TS often exhibits patterns that may be useable for modelling. *Pattern mining* methods discover those patterns, often by using past data in an offline mode. A pattern could be repeating peaks on the data, strange behaviours of customers, *etc.* [IOB09] gives an overview of some pattern mining algorithms used for production planning.

— Correlation analysis: in some technical TS, variables are correlated: percentages of CPU use vs memory RAM use, two or more functional transactions, *etc.* The sudden rise of RAM use on one server could be correlated to a later rise in CPU use, and this could lead to a problematic situation to be detected – and avoided. *Correlations analysis* of the multiple series that compose the learning ensemble could reveal unknown links between variables that may affect the performances.

— Forecast models fit: known data is used to train forecasting models. They are used after hand for forecasting purpose. Models can then be used online in real time, or offline and trained periodically (updated every night, for example). It is worth noticing that forecasting is the most difficult task but being able to accurately forecast answer the two other objectives.

Online

Online algorithms are characterised by real-time data processing using continuous streams. The data does not need to be 'complete' for the analysis to begin. Those methods are particularly adapted in the context of massive data analysis when results of computations are needed fast.

Algorithms can be trained offline and used online. They can also be trained only online while being used. Those models can adapt to new situations and are always evolving.

In the online section which is on the lower part of the figure, the three following models have been identified:

— Predictions: this thesis distinguishes *prediction tasks* and forecast tasks. Predictions are made to answer a binary question: 'is my infrastructure well dimensioned?', 'will my servers crash tomorrow?'. Forecasts are more suited if the need is to have a view of the overall evolution of KPIs, using dashboards, for example.

— Anomaly detection: finding *anomalies* in huge data streams enable faster reaction to problems. This is especially useful when applied online, to react before the problems. Anomaly analysis could also be done retrospectively but CP usually focus on real-time usage. *Peaks* are notable anomalies that should be detected because they represent unusual situations that could be dangerous for the quality of the services.

— Forecasts: this box is used jointly with models fitted offline. Models fitted are used and they produce the forecasts to be consumed by the end user.

Business goals and outputs

Creating a fully automated CP ML toolbox is a fairly complex task. A tool used by humans (project owners and managers) to enhance existing CP processes is a more reasonable goal for this thesis. This section identifies four major business goals. They are represented in light blue rectangles in the figure.

— Root cause analysis: models could help in the *root cause analysis*. This information is the holy grail, as knowing which elements have failed in complex infrastructure could accelerate next similar maintenance operations, or even problem resolution.

— Automatic virtual machine placement: because CP can also concern virtual machines, a module which takes care of their *automatic management* inside a complex data centre could be intended.

— Alerting: raising *alerts* at a right time is also something of interest. Those are good indicators of incoming problems and can be used to take rapid actions.

— Visualisation: (in light orange) the *visualisation* of the knowledge inferred by models is a major goal. It can be displayed graphically, in a nice and understandable way for non-technical users (ex: managers in charge of maintenance of data centres). Such visualisation includes technical graphs with maximum value not to exceed, peak detection... One visualisation example can be found on the Fig. 1.1.

1.4.3 Wrap up

Developing all the elements depicted in Fig. 1.4 is a challenge, because each individual functionality introduced (*e.g.* anomaly detection, root cause analysis, *etc*) is a complex subject on its own and all subjects might be studied in several independent theses.

In this thesis, the goal is to provide a tool that will help to improve the capacity planning toolset at Orange. Forecasting is seen a support for better decisions-making, in the management of the infrastructure.

Therefore, key aspects that will be studied in this work are shown by orange-coloured elements in the figure.

— The development of *offline machine learning algorithms* (upper part of the figure in the offline section);

— The offline machine learning algorithms are then used to *generate forecasts* (lower part of the figure in the online section);

— The results of the forecasts are provided using a *visualisation tool* at the end.

1.5 Wrap up and objectives

For this thesis, certain choices had to be made to narrow down the possible research directions. The focus will be made on one core aspect of the Fig. 1.4, that is the *forecasting of time series*.

Many challenges arise with the use of time series forecasting algorithms for capacity planning. More precisely, this thesis first aims at proposing a unique algorithm that is adapted to many types of incoming time series: technical ones, such as CPU or RAM use, or business ones, such as the number of people browsing a service at a certain time of the day, *etc.* It makes sense to tackle *the high diversity of data available for capacity planning problematic* in relation with the hypothesis $\mathcal{H}_1$.

Secondly, this thesis aims at proposing a data-driven algorithm that requires less human inputs as possible. The algorithm and forecasts should be easy to manipulate, both for expert data scientist but also for end users that are less used to technical aspects. It makes sense because *users of capacity planning algorithms are not necessarily data scientists*, and not always able to find parameters for complex algorithms.

Finally, this thesis aims at creating a forecasting algorithm able to foresee several points in one forecasting batch. It makes sense because *the more information about the future available for capacity planning, the more precise will be the planning.* The state of the art of forecasting algorithms in Chap. 2 shows that most of the methods are rather built for sequential point by point forecasting, or the forecast of a usually low and limited number of points.

For creating a framework that is adapted to time series, and especially in the context of capacity planning, the hypothesis depicted in Sec. 1.3.3 will be paramount because they shape the needs and create constraints. They will therefore be used as guidelines for the entire thesis.

Next Chap. 2 is dedicated to a state of the art about those time series, and more precisely about time series forecasting algorithms.

Chapter 2

Time series and forecasting models

As explained in Sec. 1.4, time series forecasting is an important asset for supporting good capacity planning. Indeed, having reliable forecasts facilitates anticipation, enables faster decision-making, and makes processes of infrastructure maintenance easier. Time series forecasting is an old research interest: the 'Journal of Forecasting' was for example set in 1982 [DH06], long before the emergence of 'big data'. Furthermore, weather forecasts [Bul01] or even financial forecasts [Wor27] were already common at the beginning of the 20th century. This field of research is relatively mature and has proposed several approaches, all adapted to various types of data and having different requirements.

Leading example

During the state of the art, a certain number of examples to support the definitions or concepts will be given. It is a golden thread that will concern one fictional infrastructure composed of one or several servers, with several indicators (CPU, RAM, *etc.*). Those examples will be emphasised in this type of grey box. This fictional infrastructure is nonetheless close to the real one which is studied in Chap. 7.

In this chapter, a view on the state of the art of time series forecasting algorithms is presented. Sec. 2.1 is dedicated to the presentation of time series. In Sec. 2.2, differences between deterministic and probabilistic forecasting are explained. It is useful to understand those for a better understanding of the next sections of this thesis. Some major deterministic models are listed and presented in the Sec. 2.3. Some major probabilistic models are listed and presented in the Sec. 2.4. Sec. 2.5 gives some details about exogenous data and their importance for time series forecasting. Finally, some limits of time series forecasting algorithms are shown in the concluding Sec. 2.6.

2.1 Time series

Time series (TS) are sequences of time-ordered values. Usually, values are numerical (real numbers). Some classical examples of numerical TS can be the evolution of the temperature at a given place, the number of heart beat per minute for a human, or the index value of companies over time in the exchange markets. TS represent the evolution of continuous processes over time. The study of TS is particularly

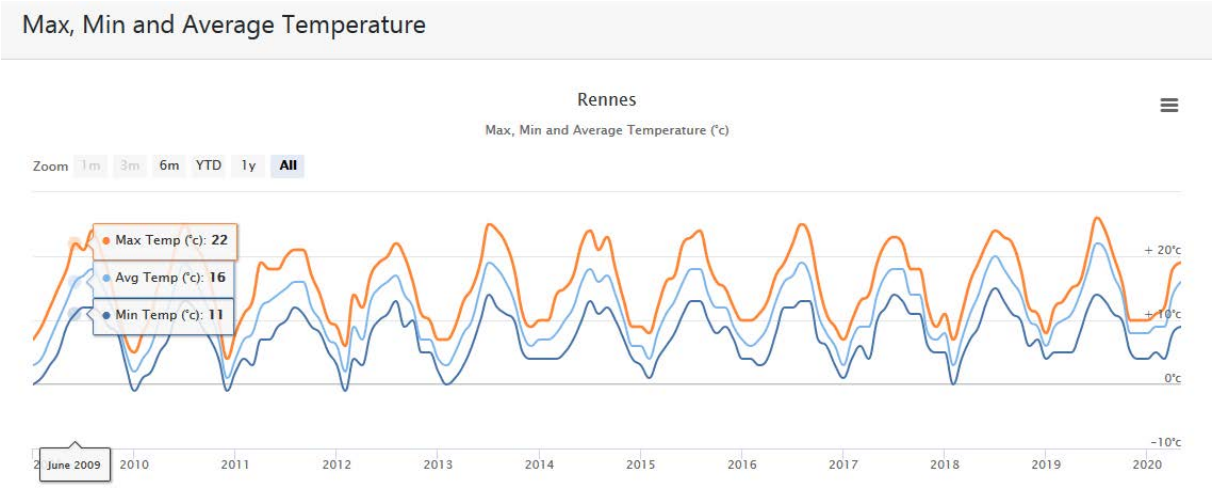


Figure 2.1: Monthly max, min and average temperatures for Rennes City for the past 11 years. Image borrowed from the following website: `https://www.worldweatheronline.com/rennes-weather-history/bretagne/fr.aspx`.

useful to model the temporal evolution of variables, or to understand the underlying processes. In this thesis, a focus on the forecasting of numerical TS is made.

For this work, the TS of interest are seasonal: they model processes that exhibit seasonal repetition over time. A classical example of seasonal TS is the measurement of temperature at a given location for several years: a lower temperature will naturally be observed at night-time, higher temperature at day time, but also higher temperatures during summer. An example of such TS might be found in Fig. 2.1, where variations between summer and winter temperatures are explicit. This type of TS is interesting because they are often found in Capacity Planning (CP) datasets due to the users daily consumption habits.

Sec. 2.1.1 below aims at giving formal definitions, which will be used afterward in the remaining of this thesis. Seasonal TS will be defined in this section. Sec. 2.1.2 gives some examples of use cases and machine learning tasks that may be applied on TS. One of which is the forecasting of TS, and will be described in further details in the next section because of its major role in the algorithms introduced in this work.

2.1.1 Definitions

First, a general definition of TS is given.

Definition 1. *Time series*
A TS $\{(t_1, \vec{y}_1), ..., (t_n, \vec{y}_n)\}$ *is a sequence of couples* $(t_i, \vec{y}_i)$ *with* t_i *the time at instant* i*,* $\vec{y}_i$ *the value at instant* i *and* n *the length of the TS.*

In this thesis, it is reminded that hypothesis $\mathcal{H}_0$ given in Sec. 1.3.3 states that points are regularly sampled, and that TS have no missing values. Thus, the t_i can be removed from the definitions below.

Moreover, a TS that describes the variation over time of one unique indicator is said to be univariate, as stated by the following definition.

Definition 2. *Univariate time series*
A univariate TS Y is a temporal sequence of values $Y = \{y_1, \ldots, y_n\}$, where $y_i \in \mathbb{R}$ is the value of Y at time i and n denotes the length of the TS.

An example of univariate TS could be the monitoring of the evolution of available RAM for a given computer.

Sometimes, multiple indicators are monitored over time. For example, one could be interested in both following the evolution of RAM use and CPU use with the same timescales. Those TS are said to be multivariate.

Definition 3. *Multivariate time series*
A multivariate TS is a vector of values $Y = \langle \vec{y}_1, \ldots, \vec{y}_n \rangle$ of length n. For all $i \in [1, n]$, $\vec{Y}_i \in \mathbb{R}^p$, where p denotes the dimensionality of $\vec{Y}$.

Some TS models are only able to deal with TS that are stationary over time. As stated in [Tab13]:

Definition 4. *Stationary time series*
The properties of the stationary TS do not depend on the time at which the series is observed [WMH98]. For a stochastic process to be stationary, the mean value and the variance do not depend on time [Har93]. Stationary TS are not seasonal and do not have trends. Non-stationary TS can have nonconstant means, time-varying variances, or all of these properties occurring simultaneously. Trendy, seasonal and cyclical TS are types of non-stationary TS [Wei06].

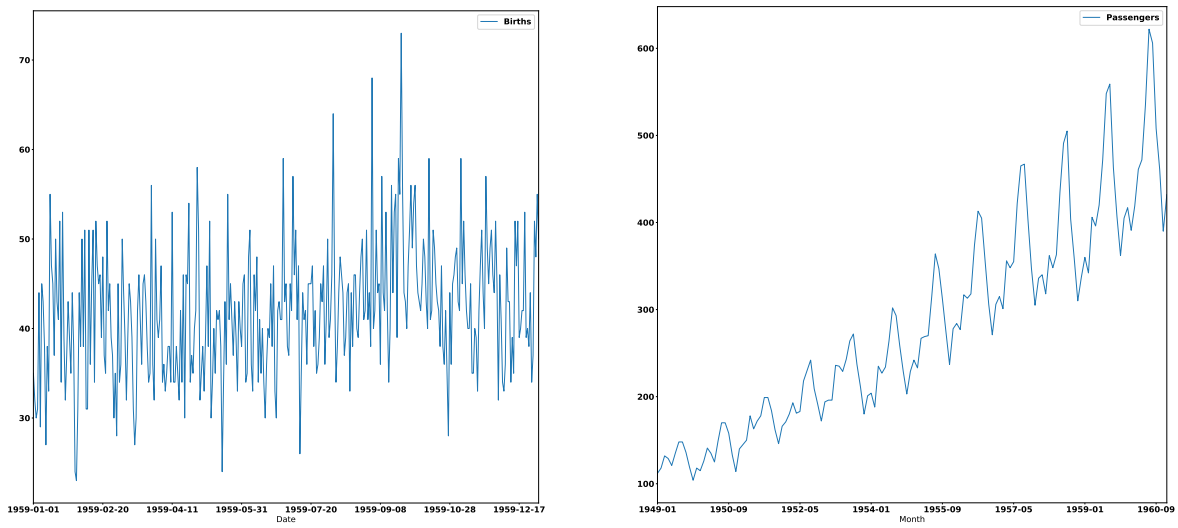
As stated in [MS96], it is more difficult to treat data where correlation between points is high, with high trends or even seasonality. Most of TS theory applies only to stationary variables.

Fig. 2.2 gives an example of a stationary TS (left) and a non-stationary TS (right). It is possible to make TS stationary. The objective is often to isolate the non-stationary part and remove it from the original signal. Among those techniques are the detrending [Raf94], deseasonalising [Nel+99], differencing [DP87] and log transforming [LX12] of TS.

Autocorrelograms are often used to find out if TS are stationary, or if there is some seasonality. They measure the linear relationship between lagged values of a TS. Lagged valued are constructed by repeating the TS with a slight delay for comparison purposes. A strong correlation between lagged points results in the graph shown in Fig. 2.3. Without seasonality and thus less correlation, the values of the autocorrelogram would quickly be close to 0.

Some TS contain a periodic pattern that repeats over periods. It is particularly the case for TS that are related to human activities or natural phenomena (for instance the average monthly temperature measured at some location on earth follows somewhat the same cycle every year). This kind of series can be denoted periodic.

Definition 5. *Season*
Let s be the seasonal periodicity of the considered univariate TS. A season is a TS $\tilde{Y} = \{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_s\}$ of length s.



(a) Stationary TS

(b) Non-stationary TS

Figure 2.2: Comparison between (a) stationary and (b) non-stationary TS. Subfig. (a) is the Daily Female Births dataset, Subfig. (b) is the Airline Passengers dataset, which shows both trend and seasonal components.

As expected, a seasonal TS is composed of one or more season that repeats over time.

Definition 6. *Seasonal time series*
Let $\mathcal{Y} = \{\tilde{y}^{1..k}\}$ be a collection of k typical seasons, then a seasonal TS $Y = \{y_1, y_2, \dots, y_n\}$ is a univariate TS of length $n = k \times s$ such that:

$$y_i = \tilde{y}^{k_{\sigma(i)}}_{i-s\times\sigma(i)} + \epsilon,\ \forall i \in [1..n]$$

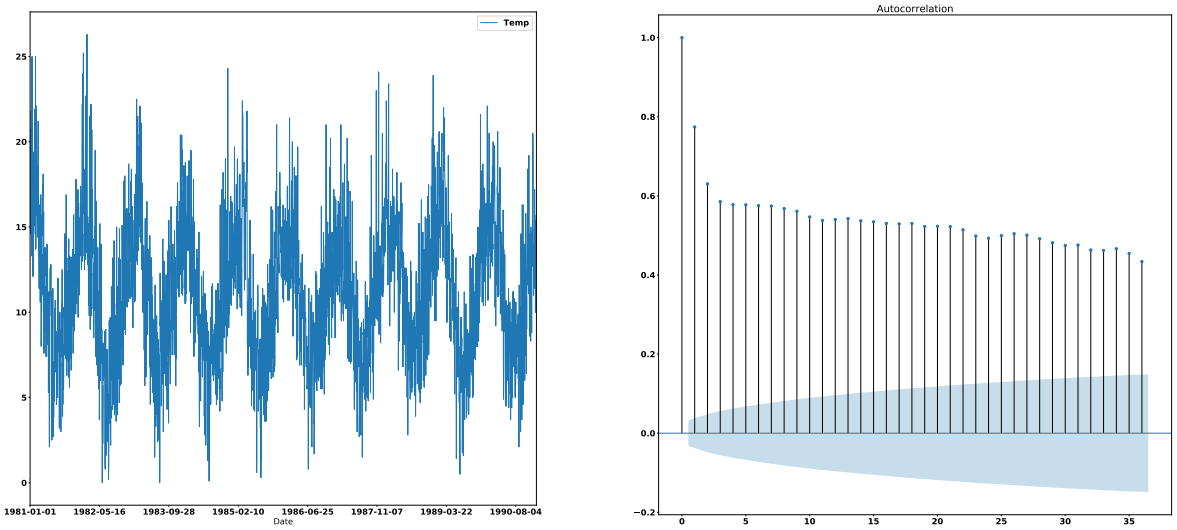
where $\sigma(i) = \lfloor \frac{i}{s} \rfloor$ is the season index of the i-th timestamp of the series, k_i is type of i-th typical seasonal TS and $\epsilon \sim \mathcal{N}(0,1)$ is a Gaussian noise.

Often, TS exhibit more complex structures than just a pattern that periodically repeats. A TS might contain different repetitive patterns that do not always occur with the same regularity.

Week days vs weekends

For instance, in a TS representing the number of people browsing a given service, week days and weekends can often be considered as two different patterns. Week day seasonality is of length 5, whereas weekend seasonality is of length 2.

Finding seasons in TS data is challenging. Seasonality can be multiple and an accurate detection is the key to better forecasts. One of the oldest methods for handling seasonality in TS is to extract them using a seasonal decomposition such as the X-11 method [DH06; LQ12] and its many variants [Fin+98;



(a) Minimum daily temperature dataset

(b) Autocorrelation plot

Figure 2.3: Open source seasonal TS (a). This dataset describes the minimum daily temperatures over 10 years (1981-1990) in the city of Melbourne, Australia. The units are in degrees Celsius and there are 3650 observations. The source of the data is credited as the Australian Bureau of Meteorology. Autocorrelogram associated (b), first 50 points.

Dag78]. Seasonal unit root tests are also a key for discovering seasons in a TS [DHF84]. Some methods are natively adapted to seasonal TS. It is the case of the nonparametric method STL [Cle+90], which automatically estimates trends, seasonal and remainder components of a TS for better forecasting. It is a simple yet efficient model based on the decomposition of the TS. [Ahd+05] propose a robust testing procedure for finding periodic sequences in multiple TS data. For long TS, it suggests the use of the Fisher's g-statistics [WFS04] for the detection of periodic patterns.

Some TS may exhibit not only one but several seasons. For example, sells might be influenced both by weekly rhythms (more sales during the week end), but also by a monthly rhythm (payday at the end of the month) and finally by the annual rate of national holidays. Recent work [Bor+17] copes with this problem by using several AR and MA models (presented below in Sec. 2.3.1) together for different time resolution of the TS. If the data is seasonal for a given resolution, AR models are used, if not MA models are used. [Gou+08] uses state-space models to model both hourly and daily patterns.

2.1.2 Machine learning tasks for time series

TS analysis [BDC02] has become a recent challenge since more and more sensors collect data with high rates. There are many attempts in the data science research community to apply modern techniques to tackle TS analysis. This section outlines non-exhaustively four of the more common machine learning (ML) problems faced by data scientists for technical TS analysis: anomaly detection, classification, clustering, forecasting.

Anomaly detection

Anomalies represent unusual and sometimes undesirable events that could be caused by problems in the systems monitored (in a computing infrastructure context or even when using sensor data on industrial machines), by unpredicted events (in the context of software security where attacks are not easily predictable) or simply on the unexpected nature of some particular datasets (anomalies in genomic data could represent some mutations) [CBK10]. Anomalies can be detected offline with datasets that are no longer evolving. The stakes here are a better understanding of the data. Anomalies can also be detected online if the need to react fast arises [Wan+11].

Classification

Generally speaking, classification algorithms aim to assign labels to given data. TS classification differs from traditional classification, mainly because TS data is ordered. The relations between features are thus different, which adds information. Nevertheless, the use cases are fairly similar: the goal is to assign a class to a given TS.

In supervised cases, labels are known, and usually represent one characteristic of the data. For example, daily temperatures could be classified as 'very cold', 'cold', 'normal' or 'hot', depending on the type of variations of the TS.

Classification is made using a classifier: the data is used at learning time, with labelled examples. Classifiers are then used with unlabelled data and predict their classes, sometimes with a relative degree of certainty for probabilistic classifiers.

In some recent work [Lar+19], authors compare the performances of two dictionary-based classifiers for TS classification. In [DBC15], authors study a specific problem that is early classification of TS, where the goal is to classify TS as early as possible in continuously incoming data. A detailed analysis of most of the current TS classification algorithms and their performances may be found in [Bag+17], where authors reimplemented most of the classifications algorithms published in the last decade and made extensive experiments.

Clustering

Clustering algorithms aim at identifying groups of TS that are similar in a given ensemble of TS.

More formally, and as stated in [ASW15]:

Definition 7. *Given a dataset of n TS data $D = \{Y_1, Y_2, \ldots, Y_n\}$, TS clustering is the process of unsupervised partitioning of D into $C = \{C_1, C_2, \ldots, C_k\}$, in such a way that similar TS are grouped together. C_i is called a cluster, where $\forall i \in [1,n] \subset i \subseteq \{Y_1, ..., Y_n\}, D = \cup_{i=1}^{k} C_i$ and $C_i \cap C_j = \varnothing$ for $i \neq j$.*

Clustering algorithms are unsupervised, and the ground truth about clusters composition is unknown *a priori*. One problematic that quickly arises is the question of distance between two TS. Such distance might indeed be computed in various ways. It is a challenging task because of the noisiness of TS and the presence of outliers [Lin+04]. There exists several distances. Two of the most widely used TS distance metrics are the Euclidean distance [FRM94] and the Dynamic Time Warping distance [SC78].

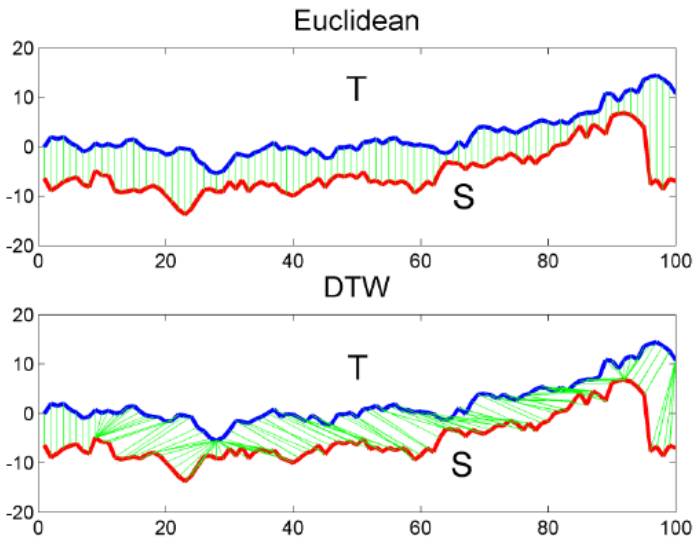


Figure 2.4: Difference between Euclidean and DTW measures. Image borrowed from [Cas+12].

— Euclidean distance consists in computing the square root of squared pairwise difference (point by point) for two TS of same length. For two TS Y and Z of length n: $Y = \{y_1, \ldots, y_n\}$ and $Z = \{z_1, \ldots, z_n\}$, the Euclidean distance ED is equal to:

$$ED(Y, Z) \equiv \sqrt{\sum_{i=1}^{n} (y_i - z_i)^2}$$

— DTW is a method that calculates an optimal match between two given TS by computing their optimal alignment. The latest should minimise the Euclidean distance between aligned points. DTW is useful when the need is to compare two TS where very similar events happened but not simultaneously (see grey box example below). Fig. 2.4 shows visually the difference between DTW and Euclidean distance.

Product launch, timezones and DTW

Data peaks that happened on the launch of a new product on two different time zone might be comparable but they do not happen on the same time scale. The use of Euclidean distance with two peaks in two timescales not synchronised will dramatically increase the error. Although, there is a need to detect those two TS as similar, through the use of DTW for instance.

Forecasting

TS forecasting consists in predicting future values for one or several given TS. It relies on the study of the data already known, and possibly of exogenous data, to infer possible evolution and make predictions.

The first question that the forecaster naturally faces is to know the desired term for the forecast. The horizon is linked to the number of points forecasted. To state the obvious, the further away are the points forecasted, the more uncertain is becoming the forecast.

2.2 Deterministic vs probabilistic forecast

This thesis goal is to provide a reliable mid term to long-term forecasting algorithms for seasonal TS. In this work, TS are assumed to be regularly sampled and univariate.

Definition 8. *With Y a TS of length n, forecasting up to the horizon h is the task which consists in predicting the h next points such as $\hat{Y} = \hat{y}_{n+1}, ..., \hat{y}_{n+h}$.*

There exists several forms of forecasting and several paradigms for TS forecasting. Experimentations have been conducted using the two following paradigms: (1) deterministic and (2) probabilistic forecasting. The concepts for those two different paradigms are illustrated with the Fig. 2.5, and detailed respectively in the next two Sec. 2.3 and 2.4.

Those two methodologies achieve two different goals.

— In the deterministic paradigm (also known as point forecast [Gne11]), the aim is to obtain a numerical estimation of the future values that the TS will exhibit.

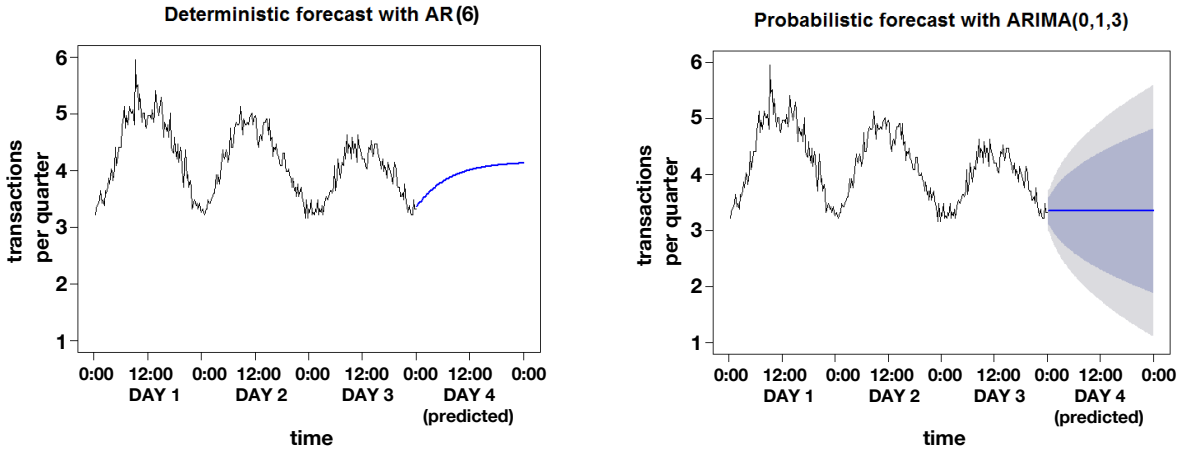
— In the probabilistic paradigm, the aim is also to predict future values, but forecasts are obtained in the form of probability distributions of the value of the TS at a given instant in time, rather than plain deterministic values.

Most of the algorithms of the state-of-the-art are adapted for a short term forecasting that either consists in forecasting the next few points of a TS or some points not too far from the last observed point. It is easy to imagine that longer forecasts are also more difficult to produce, because the uncertainty rises with broader horizons and the error can quickly accumulate [Sor+07]. More factors that cannot be foreseen nor controllable are engaged when the horizon is larger.

Multi-step forecasting is a concept which is closely linked to long-term forecasting. Indeed, one often needs to predict several points in time in the future for retrieving a long-term horizon. But multi-step ahead forecasting tasks are more difficult [TT94], since they have to deal with various additional complications like accumulation of errors, reduced accuracy, and increased uncertainty [Wei18; Sor+07; Tai+12]. The oldest and most intuitive multi-step strategy is the recursive strategy [Tai+12], where forecasted values are fed back to the algorithm in order to predict the future values. It is risky if one of the forecasted values is wrong because the error will propagate quickly.

2.3 Deterministic forecasting

A TS forecast is considered *deterministic* [Zam+14] when it provides numerical values for the next points to appear in the future. More formally, a deterministic forecast can be defined as follows:



(a) Deterministic forecast

(b) Probabilistic forecast

Figure 2.5: Comparison between (a) deterministic and (b) probabilistic forecast. Historical values are in black and forecasts are in blue. In Subfig. (b), the blue line is the expected value, and blue regions are probabilistic intervals. The darker the region, the more likely the values.

Definition 9. *Deterministic time series forecasting*
With $Y = \{y_1, \ldots, y_n\}$ *a TS of length* n*, forecasting the future of* Y *up to horizon* h *consists in giving numerical values estimations such as* $Y = \{\hat{y}_{n+1}, \ldots, \hat{y}_{n+h}\}$.

Future values at time $n+h$ for each forecasting horizon h can be formalised as follows: $\hat{y}_{n+h} = g_h(y_1, ..., y_n) + \epsilon_{t+h}$ where g_h is a forecasting model specific of the horizon h and ϵ_{t+h} denotes the model error. At this point, g_h is a single-valued forecasting model.

Data centre performance

As an example of a deterministic forecast, if the problem is to forecast data centres performances, forecast could say that tomorrow's maximum use of the RAM will be 32,760 MO, or that the maximum number of people browsing will be 752 at the end of the day.

The remaining of this section is dedicated to several deterministic methodologies. The well-known autoregressive models is first introduced in Sec. 2.3.1, then Sec. 2.3.2 focus on seasonal TS forecasting algorithms, Sec. 2.3.3 presents neural networks methods and finally Sec. 2.3.4 makes a focus on ensemble models for TS forecasting.

2.3.1 Autoregressive models

Autoregressive forecasting models are often state of the art for many applications, from finance to weather forecasts. Models are numerous and each has specific assumptions about data shape and format.

They can be seasonal or not, adapted to short or long forecasts, can be used for one step ahead forecast or multistep forecasts, *etc.*

The list below gives an overview of some major linear models. Those are ordered by increasing complexity. The first one on that list is among the simplest and easiest models to create TS forecasts.

— **Simple average/mean forecast**: this is the most naive method, where future point $\hat{y}_{n+1}$ is equal to the mean of all the past points present in y. Then,

$$\hat{y}_{n+1} = \frac{1}{n}\sum_{j=1}^{n} y_j$$

— **Autoregressive model (AR)**: introduced in [Aka70; Aka98], AR makes the assumption that there is a linear relationship between the output variable and the past values of the TS. The parameters can be learnt while minimising least squares error. AR model is not always stationary as it may contain a unit root. The order of an AR model is noted p, such as:

$$y_t = c + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \varepsilon_t$$

where $\varphi_1, \ldots, \varphi_p$ are the parameters of the model, c is a constant, and ε_t is white noise. The advantages of those models are the model simplicity, the computational efficiency during the training phase and the handling of the noise. AR models might also be combined. [CHB07] use a sum of two AR models for creating forecasts in a capacity planning purpose, with manufactures data, which can be highly seasonal and thus similar to the type of data processed. Note that AR forecasting is more precise when it deals with non-seasonal datasets.

— **Moving Average (MA)**: this method assumes that less recent points may have less importance than the most recent ones. The order of a MA model is noted q, such as:

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

where μ is the mean of the series, the $\theta_1, \ldots, \theta_q$ are the parameters of the model and the $\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}$ are error terms.

— **ARMA**: an ARMA model is obtained by combining one autoregressive model along with a moving average model. One ARMA model of order (p, q) is a discrete temporal process $(Y_t, t \in \mathbb{N})$ such as

$$Y_t = \varepsilon_t + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where φ_i and θ_i are the model parameters and ε_i error.

  — An AR model AR(p) is actually an ARMA($p, 0$)

  — A MA model MA(q) is actually an ARMA $(0, q)$

— **ARIMA [Box+15]**: Including an integrated part of an ARMA model results in ARIMA model. The integrated part of the ARIMA model aims at transforming any non-stationary data to stationary

data by differencing raw observations; data values are replaced by the difference between the data values and the previous values.

There are three parameters to set: p, for the order of the autoregressive model, d for the degree of differencing, and q for the order of the moving-average model. Those parameters may be set following the Box and Jenkins's methodology [MH97]. Grid search methods are able to find the best combination of (p, d, q) parameters in a defined range. It allows using fine-tuned ARIMA models without a deep knowledge of the Box and Jenkins's methodology. In [JAS11], authors use several ARIMA models to predict day-ahead electricity prices. Kavasseri et al. [KS09] use fractional-ARIMA to generate a day-ahead and two days ahead wind forecasts.

2.3.2 Seasonal models

The *seasonal TS deterministic forecasting* is a forecasting of a seasonal TS at a horizon $h = s$, *i.e.*, the prediction of the TS values for the whole next season ahead. One season is necessarily composed of several points.

ARIMA expects data that is either not seasonal or has the seasonal component removed, *e.g.* seasonally adjusted via methods such as seasonal differencing [Box+15]. The SARIMA model [GET06] extends the autoregressive model to deal with seasonality. It is an ARIMA model that can deal with seasonal aspects in TS.

In [CS12], SARIMA is being used for forecasting the daily peak electricity in South Africa. Electricity forecasting is one big interest in the forecasting community [IM05; KM06; CPB09; Koo+13; CPM10; JAS11].

It is possible to use seasons with SARIMA models, that are based on the ARIMA models. In SARIMA, values that are far away back in time have a limited impact on the forecasting, whereas each and every season is valued with the same interest for producing forecasts in this thesis framework. It is also possible to enrich SARIMA models with exogenous data, like in [Xie+13] for a study on the electrical market in Sweden.

In [Esp+05], authors developed a Periodic Autoregression (PAR) model to deal with highly seasonal data: electricity grid data in Belgium, monitored for 5 years. They create a short-term forecasting algorithms which rely on the PAR model to take into account multiple seasonality (weekly, monthly, *etc.*). They also use it to identify multiple 'daily profiles' and use clustering in the goal of diminishing dimensions.

The difficulty of producing reliable forecasts arises when multiple seasonality enters into play.

Weekly, monthly and yearly data

In massive infrastructure and TS related to the use of the servers, it is often observed that TS data depends on the weekly habits of the consumers (*e.g.* less activity during the weekend for an application linked to work for employees); it is easy to imagine that monthly events such as paydays will also impact the systems. Finally, yearly events such as national holidays are also important.

In [DHS11], authors use exponential smoothing for leveraging this multiple seasonality problematic. Their framework, which relies on maximum likelihood, was successfully tested on various TS datasets, both linear and non-linear. Adding exogenous and empirical data is also possible.

2.3.3 Neural networks models

In the ML communities, one of the objectives is to create data analytic tools that would require fewer modelling efforts from data scientists. *Neural networks models* can be efficient for forecasting tasks, especially when the data is more complex and when the process is non-linear. They can be seen as nonparametric data driven approaches which can capture nonlinear data structures without prior assumptions about the underlying relationship in a particular problem [ZPH01]. One drawback of those models is their tendency to over-fit [Zha03], which may cause lower performances. Interested readers may find more specific and detailed information about neural network applied for TS forecasting in the following [HOR96; AE17].

Feedforward networks are the simplest neural network available. In the latest, data flows from one point of the network to another with no retropropagation. They are for example used in [KHM05] for forecasting lynx population, in [BS12; Saâ17; An+13; Dud16; Cat+07] for electricity forecasting matters.

Long Short Term Memory (LSTM) neural network [GSC99] is a classical architecture used for TS forecasting, but not dedicated to seasonal TS. The interest of LSTM is that they benefit from feedback connections between neuron, which gives them some memory of the past. LSTM are largely used, with various applications: forecasting petroleum production [SK19], financial forecasts [ZXZ17; CLL19; Sel+17; SWW18], electricity price and demand [Zhe+17; Kum+18], *etc.*

Finally, neural networks are used in conjunction with autoregressive models, linear part of the data being modelled by the latest. Examples of hybrid cohabitation between neural networks and ARIMA models might be found in [Zha03; Far10; TYT02; KB11; KAV15; WM12].

2.3.4 Ensemble and combination models

Ensemble forecasting methods and hybrid models are created from several state of the art, independent models, that are mixed to create more complex chains. This strategy is the one proposed in the Arbitrated Dynamic Ensemble [Cer+17]. This metalearning method combines different models, regarding to their specifies against target datasets. In [PL05], authors use both ARIMA and SVMs models to forecast stock prices problem, and to tackle the non-linearity of some datasets.

Ensemble methods take advantage of several forecasting algorithms for creating better forecasts. Indeed, each algorithm may be more precise with certain types of datasets, or in certain identified situations. These hybrid models are created from several state of the art, independent models, that are mixed to create more complex chains. They are able to take advantage of each model specificities to yield better forecasts. The general idea is to select which algorithm to apply to which data (or even to which portion of the data) in a transparent way for the user. It does not require manual settings. In [WS10], many methods are being used for crafting forecasts: neural networks, auto regressive models, GARCH, Monte Carlo, wavelets.

2.4 Probabilistic forecasting

The popularity of *probabilistic forecasting algorithms* is increasing over the past few years. In 2014, a worldwide probabilistic forecasting (GEFCom 2014) competition took place and started to drive the attention to probabilistic forecasting for international data scientists. The organisers provided electricity and weather information to the contestant with the goal of producing rolling energy forecasts over several weeks of contest.

A TS forecast is considered probabilistic when it provides a probability distribution over several future possible values instead of one unique value but a probability for each possible output value.

Definition 10. *Probabilistic time series forecasting*
With $Y = \{y_1, \ldots, y_n\}$ a TS of length n, forecasting the future of Y consists for h in $\mathbb{N}$ in predicting a random variable $\hat{y}_{n+h}$ with an explicit density distribution : $P(\hat{y}_{n+h} \leq y|Y)$.

A general introduction to probabilistic forecasting might be found in the book [Gne08].

In the early 2000, probabilistic forecast was mainly used for weather forecasts [GRG04; SGR10; Pin+07; TG10; Dob+08; BRG07] (mostly for wind speed forecasting). Despite the public familiarity with probabilistic forecasts mainly due to its generalised public used for weather forecasts, this research field is only recently driving the attention of the forecaster community.

[GGN16] were the grand winner of the GEFCom forecasting competition of 2014 mentioned above, and proposed a model called quantile generalised additive model (quantGAM). It relies on the forecast of the temperature in the future - it is easy to imagine that more energy is required during winter for heating houses and thus it has an impact on the overall energy consumption.

A lot of methods presented in this competition also rely on two major elements to produce good forecasts.

— First, human expertise is required for configuring those very complex models. Being knowledgeable of one specific field is useful after all, and experts are really able to make a difference for tuning parameters and algorithms perfectly.

— Second, the volume and availability of exogenous data are paramount: weather data are exogenous to electricity consumption curves, but mandatory in those methods for producing good forecasts. The choice of exogenous data to be included in the learning process is not trivial. Picking data which is not adapted would probably not improve the performances and could even lower them.

In this family of forecasting methods can also be found quantile estimators [GWK89], Prediction Intervals for Regression [Sti85], Fuzzy Prediction Interval Models [Sáe+14].

Some methods already cited above natively provide intervals in their forecasts: SARIMA [Zho+06] (here for electricity price forecasting), or even Prophet [TL18].

Data centre performance with probabilistic forecasts

In an huge infrastructure, it is possible to forecast three different scenarios for the incoming day: one scenario with a low use of CPU sc_1, a scenario with a high but sustainable use of CPU sc_2 or a catastrophe scenario sc_3 - the one that will put the service down for sure. Probabilistic forecasting algorithms then attach probabilities to each of the cases (for instance $P_{sc_1} = 0.32, P_{sc_2} = 0.08, P_{sc_3} = 0.60$). With those prognosis engaged, tomorrow has a high probability of being messy and decisions could be taken in advance for preparing the servers. This is a simple case of probabilistic forecasting where the predicted probability distribution is non-zero only at three places (sc_1, sc_2 and sc_3).

2.4.1 Interval forecasts

One major approach is *interval forecasts* [GWK89; Cha93]. The principle is to forecast an interval in which the future data points may lay, with a certain probability (a confidence score). For instance, the prediction for a TS at the horizon h will be $[\gamma,\omega]$ to state that the value $\hat{y}_{n+h} \in [\gamma,\omega]$. As stated in [Wer14], a prediction interval is associated with a random variable (*e.g.* electricity price) that is yet to be observed, while a confidence interval is a specific case for an interval forecast.

One second common use of probabilistic forecast is to follow the evolution of the population [KPH02; Alh99; Kei01]. Note that interval forecasts are carrying more information and thus are more subject to interpretation. Indeed, according to [GÖT10], a well-configured interval forecast (around 85% of confidence) will help the decision makers in a production planning problematic, but a too high or too low confidence interval rather complicate the decision-making process.

GluonTS [Ale+19] is a TS probabilistic forecasting framework based on neural networks. This framework has several pre-trained models that can be used as-is or retrain, depending on the users' needs.

2.4.2 Density forecasts

Density forecasts are more complete than interval forecasts because they provide the entire distribution and probabilities of the possible future values. It is especially useful for forecasts that do not follow the normal distribution [DH06]. It is a more complex task, both in terms of realisation and in terms of interpretation for the end user. To the best of my knowledge, the literature is rather sparse on this specific type of probabilistic forecast. Interested readers might find more information in the survey [TW00]. Cumulative Distribution Functions (CDF) (see Sec. 3.5.3) are sometimes used to represent density forecasts.

2.5 Exogenous data

Exogenous data are not included in the raw TS. They represent any information that might add new knowledge to the studied problem. This information comes from external sources sometimes not directly related to the problem studied. Those can be used to improve forecasts.

Exogenous data for an application

For example, in the case of a TS of transactions per seconds on a given web service, it could be interesting to know that some days in the learning sets are holidays or that one religious event that will dramatically change the user behaviour will happen. The latest is often happening with OM application with the Aïd event in Africa which has different dates every year. That information is not included in the original raw TS but could enrich the model at learning time.

Most of the time, using exogenous data is not easy, as the analyst has to configure the models for including them. [TL18] introduces a decomposable TS model (called Prophet) that uses the holiday period (exogenous data), trend and seasons for providing seasonal interval forecasts. Similar to a generalised additive model (GAM [Has17]), with time as a regressor, Prophet fits several linear and non-linear functions of time as components. This algorithm is largely impacted by the quality and number of exogenous data available at training time. It follows the 'analyst-in-the-loop' paradigm, which implies that a human intervention is highly desirable for tuning the models and improving results in testing. With less information and with less expertise from the data scientist tuning the model, the performances might dramatically drop.

2.6 Wrap up and perspectives

As seen in this chapter, there exists many different forecasting algorithms for TS. They are all adapted to some situations but also all have some specific limitations. This thesis will especially focus on the study of *seasonal TS forecasting* algorithms, both for *deterministic* and *probabilistic* paradigms. The three major limitations this work aims to tackle are listed below.

As seen in the Sec. 2.2, multi-step forecasting is usually *not the most precise way of producing reliable midterm forecasts for seasonal TS*. Due to the sequential nature of those forecasts, the link between forecasted values may be too strong. If an error is made on the first point forecasted, all the future values might inherit of this mistake and be biased. The longer the forecast horizon, the more the process repeats. The error then grows faster. Also, the longer the horizon, the more time step to be forecasted. It means that the risk to produce wrong forecasts is also more important.

A second limit of this kind of forecasting algorithm is *the way models are set up*. Most of the time, human intervention and expertise is needed to configure the models. There are some parameters to tune, which may be set with the knowledge of the data, that could be acquired with data scientists task such as data exploratory analysis. Those tasks are tedious and not necessarily easy to handle by automated scripts. Human interpretation is often key for setting useful parameters and therefore, to obtain good forecasts.

Finally, most of the state-of-the-art algorithms are *unable to take into consideration multiple seasonalities* in an easy way. This is something that is easier for us, because higher-level seasonal variations might be included in the low-level split seasons and learning is made with this bias.

Next Chap. 3 presents the general framework proposed during this thesis in an attempt to remove those limitations.

Chapter 3

Conceptual view of the framework

This chapter presents a conceptual view of the framework proposed during this thesis. Abstractions and ideas behind the algorithms are introduced. Different concrete implementations [1] will be depicted in the next chapters Chap. 4, 5, and 6. The proposed framework aims to provide reliable medium to long-term forecasts for seasonal time series.

Theoretically, this framework is suited to seasons of any length and aims at being as *generic* as possible in order to be useable for any type of data (under hypothesis Hyp. $\mathcal{H}_1$ in Sec. 1.3.3 page 34). The framework also aims at being *easy to use* for non-expert users. It requires only one parameter (season length known in advance, according to Hyp. $\mathcal{H}_4$ in Sec. 1.3.3), except the internal hyperparameter optimisations that the end user will not deal with. Seasonality is paramount for this methodology, and the entire framework is built upon their automatic extraction. The notion of season will be specified below in Sec. 3.2, and the hypothesis on the data clarified.

The framework is composed of two learning steps and one forecasting step, as depicted in Fig. 3.1. First, it *groups similar seasons* of a given time series into homogeneous groups. Groups may be built with clustering or with coclustering algorithms. Then, the groups are used to learn to *predict the most probable groups of the next season*. This can be done using information from the past seasons. Once the model is learned, next season forecast is created by *using the groups and trained next season predictors*.

In this chapter, Sec. 3.1 first formalises the problem solved by the aforementioned framework. Sec. 3.2 then synthetises the main motivation in the development of a new seasonal forecasting framework. The different parts of the framework are then presented in Sec. 3.3. Details about the data used in all experiments are given in Sec. 3.4. Measurements, evaluation strategies and protocols that will be used in this thesis for assessing the quality of the forecasting algorithm will be presented in the Sec. 3.5.

3.1 Problem statement

In addition to the hypothesis depicted in Sec. 1.3.3 in page 34 which only concerned the data at hand, two additional hypotheses are formalised below. They aim to give an intuition about the mechanisms that will be used in the framework. More especially:

— $\mathcal{H}_7$: knowing $\mathcal{H}_5$, a clustering algorithm can be useful to groups similar type of seasons contained in time series;

— $\mathcal{H}_8$: knowing $\mathcal{H}_6$, a classification algorithm can be used to learn links between various season types contained in time series, in order to model the occurrence of season types.

1. In this context, implementation is analogous to instantiation, *e.g.* technical development of the conceptual framework.

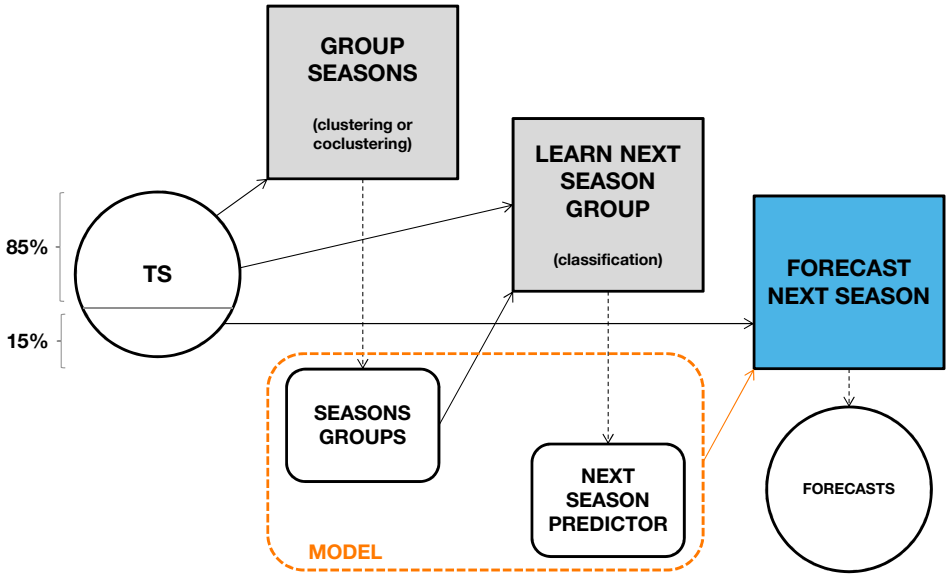


Figure 3.1: Simplified view of the learning (in grey) and forecasting (in blue).

Given $Y = \{y_1, \ldots, y_n\}$ a seasonal TS and $s \in \mathbb{N}^*$ its season length (see Sec. 1.3.2 for an example of seasonal TS data in the context of capacity planning (CP)), it is interesting to forecast the next season of Y, *i.e.* computing $\hat{y}_{n+1}, \ldots, \hat{y}_{n+s}$ that represents the prediction of the s values in the next season of Y. It is recalled that multi-step forecasting is a challenge (see Sec. 2.2) and existing solutions such as a recursive strategy are not always satisfactory. Forecasting an entire season at once is a way to limit the error propagation.

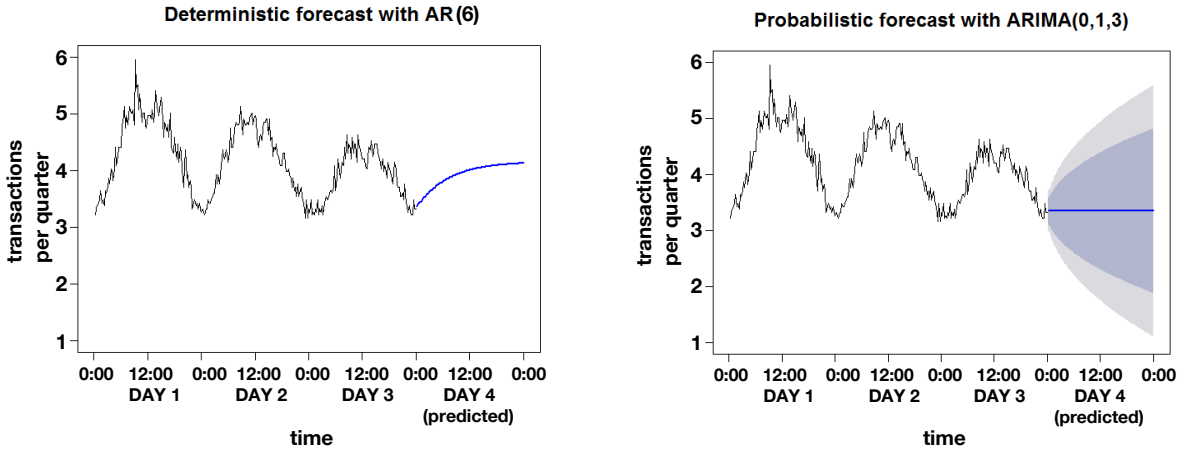
These predictions are made using the history of Y (values 1 to n). Computing the prediction of the value of Y at time instant $n+i$ can be written as:

$$\hat{y}_{n+i} = F_{n+i}(y_1, \ldots, y_n), \forall\, i \in [1, s],$$

where F_{n+i} is a model that needs $y_1, \ldots, y_n$ as input to predict $\hat{y}_{n+i}$. In the case where $F_{n+i}(y_1, \ldots, y_n) \in \mathbb{R}$, the prediction is said to be *deterministic*. The model predicts the value of the time series. A visual example of a deterministic forecast is given in the left part of Figure 3.2 and more explanations may be found in Sec. 2.3.

This framework also proposes on another kind of forecasting : *probabilistic* forecasting. In this case, the model predicts a probability distribution of the domain of TS values. In this case, $F_{n+i}(y_1, \ldots, y_n)$ is no more a real value. It is a function that represents a probability distribution. $F_{n+i}(y_1, \ldots y_n)$ represents the probability distribution $\mathbb{P}(y_{n+i} = x \,|\, y_1, \ldots, y_n), x \in \mathbb{R}$.

A visual example of a probabilistic forecast is given in the right part of Figure 3.2 and more explanations may be found in Sec. 2.4.



(a) Deterministic forecast

(b) Probabilistic forecast

Figure 3.2: Presentation of (a) deterministic and (b) probabilistic forecast. Historical values are in black and forecasts are in blue. In Subfig. (b), the blue line is the expected value, and blue regions are probabilistic intervals. The darker the region, the more likely the values.

3.2 Model for seasonal time series

Sec. 1.3 has shown that some TS exhibit clear seasonal behaviour (see Hyp. $\mathcal{H}_2$ in Sec. 1.3.3), and that the behaviour is not exactly the same from one season to another (see Hyp. $\mathcal{H}_5$ in Sec. 1.3.3). Many seasonal TS exhibit various recurring seasons types. It is common to find several recurring season types in long TS. For a weekly dataset, some specific days could show similar user behaviours. For example, in a financial context, non-working days like Saturdays and Sundays might exhibit a higher activity for shoppers that tend to go shopping on their free time. In contrast, certain areas like business districts, that are usually busy during weekdays, might exhibit a much lower activity level during the weekend. Recurrences are not necessarily as simple as this working day/holiday example. The overall activity for a given day is a mix between shoppers that are in free time (more likely during the weekend) and shoppers that have shifted days off.

The Def. 11 introduces the model of seasonal TS.

Definition 11. *Model for season and seasonal time series*

A season is assumed to have a fixed length s. Without loss of generality, it is assumed that the length of Y is a multiple of s ($n = m \times s$). Thus, Y can also be written $Y = \{D_1, \ldots, D_m\}$ where $\forall i \in [1, m]$, D_i is a subsequence of Y (of length s) that represents the i^{th} season of Y, i.e. *$D_i = y_{(i-1)\times s+1}, y_{i\times s}$.*

Y is said to be seasonal (with season length s) if there exists $\mathcal{S} = \{S^1, \ldots, S^p\}$ a finite collection of p sub-series (of length s) called typical seasons such that

$$\forall i \in [1, m], D_i = \sum_{j=1}^{p} \sigma_{i,j} S^j + \epsilon_i,$$

where ϵ_i represents a $s-$dimensional vector of noise samples and where $\sum_j \sigma_{i,j} = 1$. In other words, it means that every season in a seasonal TS Y is a weighted linear combination of typical seasons plus some noise.

Intuitively, $(\sigma_{i,j})$ give the proportions of the combination of type j in the i-th season of the TS. If S^j is seen as an individual behaviour, then $\sigma_{i,j}$ can be seen as a proportion of individuals of type j that contribute to cumulative value D_i. Alternatively, it can be seen as a probability of having the individual behaviour and D_i the esperance of the individual behaviour.

Fig. 3.3 illustrates a seasonal TS with six seasons of length 3. Two typical seasons can be found in this TS (blue and orange), and one season (green) is a mix of the two others (50% of each). Here, $\sigma_{1,1} = 1$, $\sigma_{1,2} = 0$, $\sigma_{2,1} = 0$, $\sigma_{2,2} = 1$, $\sigma_{3,1} = 0$, $\sigma_{3,2} = 1$, $\sigma_{4,1} = 1$, $\sigma_{4,2} = 0$, $\sigma_{5,1} = 0.5$, $\sigma_{5,2} = 0.5$, $\sigma_{6,1} = 0.5$, $\sigma_{6,2} = 0.5$.

According to the above definition, this TS is seasonal as it is the succession of typical seasons, but it is not periodic as the season is not always the same. Note that seasonal TS has to be distinguished from cyclical TS. Cyclic patterns exist when data exhibit rises and falls that are not of fixed period, unlike the seasons that are necessary of fixed and known length. [2]

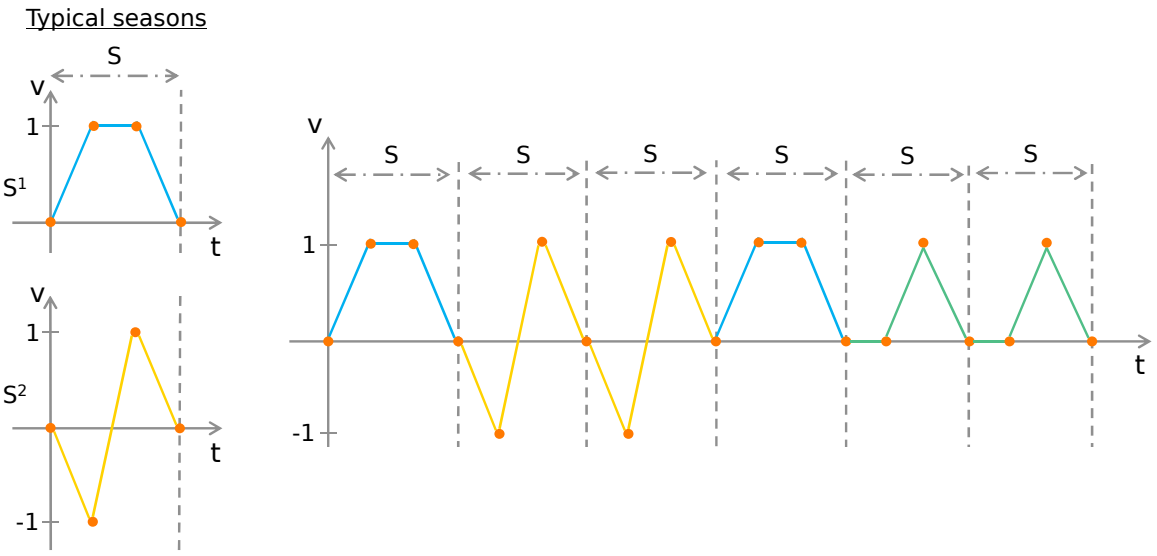


Figure 3.3: Examples of seasonal TS with two different typical seasons (illustrated on the left) and one mixed season. Each season is repeated twice, but not always in the same order.

The intuition is that extracting seasons and studying their occurrence patterns could help create a forecasting algorithm more versatile and able to cope with several identified seasons. It is one strong assumption on the informativeness of those patterns. More precisely, *TS could mainly be driven by a two-order temporal process*. It is worth noticing that if the TS is stationary, it is assumed that only the $\sigma_{i,j}$ are changing in time.

For daily datasets, the circadian scale drives the daily evolution of the TS (first order) but, at the second order, this daily behaviour is itself driven by some hidden rules. The evolution within a week

2. See https://robjhyndman.com/hyndsight/cyclicts/.

could be considered. During the weekdays (from Monday to Friday) TS have a daily behaviour which is different from the weekend daily behaviour. The same reasoning can be applied for any size of seasons and nested seasons of smaller size.

It has been shown in Def. 11 above (and in Hyp. $\mathcal{H}_5$ in Sec. 1.3.3) that in the TS of interest, several types of seasons coexist and they are different enough to be gathered in several different groups (two seasons that would not be different enough to be easily differentiated can be seen as the same season, their differences are then considered as noise). Such groups are called *typical seasons*.

Fig. 3.3 exhibits a sequence of 6 seasons with 3 typical seasons, but some longer TS could show longer sequences with more seasons. Still in Fig. 3.3, it can for instance be learned simply by observing the figure that there is 50% of chances to see a yellow season and 50% of chances to see a green season after a blue season. This example is generalisable using more season types and with more data while using a classifier.

3.3 Framework for seasonal TS forecasting

This section introduces the general framework proposed for seasonal TS forecasting. For the sake of clarity, deterministic forecasting is the only paradigm considered. The aim is to predict the next season of Y, *i.e.* $\{\hat{y}_{n+1}, \ldots, \hat{y}_{n+s}\}$.

The general sketch of the approach is composed of two classical data driven processes: a learning process (see Fig. 3.4) and a forecasting process (see Fig. 3.5). The learning process learns a forecasting model made of a clusterer and a classifier. This model is used for predicting future TS values during the forecasting process.

The method proposed is to use the seasonal behaviour of the TS by defining some typical seasons denoted $\{S^1, \ldots, S^p\}$ in Def. 11 and thus to detect second-order behaviours by analysing sequences of typical seasons.

Some technical details about these two processes are given in the next two subsections.

3.3.1 Learning process

The learning process first identifies typical seasons in the TS using clustering. Classifiers are then used to learn the next season type. The combination of clustering and classification for learning the group's appearance probability based on the seasons themselves is intuitive – if the hypothesis $\mathcal{H}_5$ about the presence of various and well-defined seasons in the data is respected.

The learning process is composed of three steps (see Fig. 3.4):

1. **Data splitting:** The data splitting step simply extracts, from $Y = \{y_1, \ldots, y_{m \times s}\}$, a set of m seasons $\mathcal{D} = \{D_1, \ldots, D_x, \ldots, D_m\}$, where D_x represents the subseries of length s corresponding to the x^{th} season in Y.
2. **Season grouping:** The m elements of $\mathcal{D}$ are given to a clustering algorithm that gathers similar seasons into p clusters of TS named $\{C_1, \ldots, C_p\}$. Each cluster is hence composed of similar TS. They each may be summarised using a prototype (*i.e.* a representative TS for a given cluster). *Centroids* or *medoids* could be chosen as prototypes.
   — The *centroid* is the mean of all the seasons present in the cluster. It does not necessarily represent a real-TS found in the set.

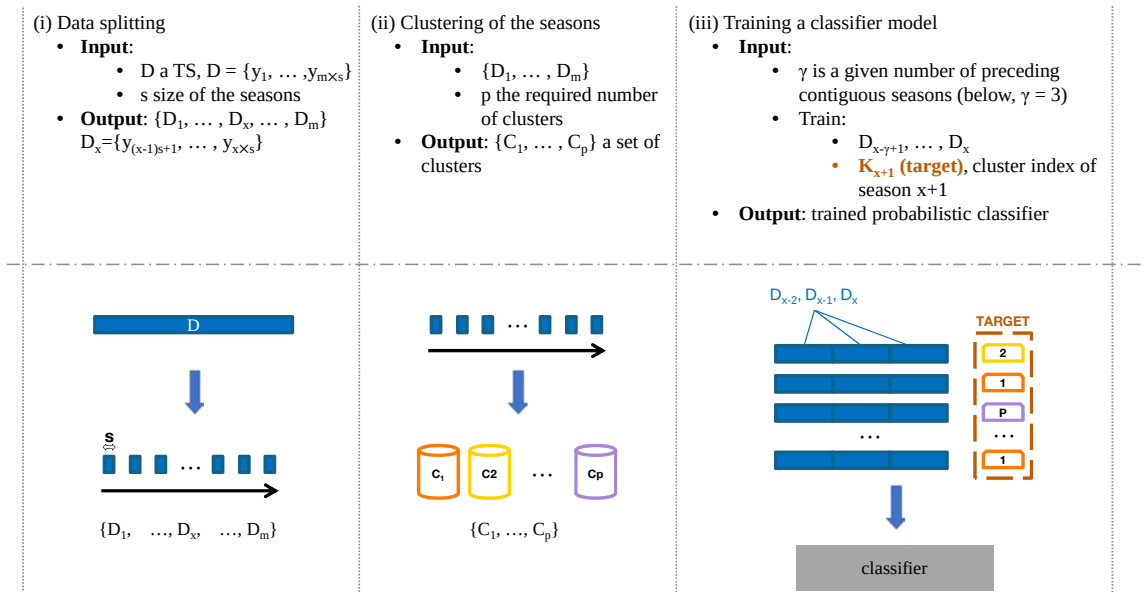


Figure 3.4: Illustration of the learning process (see text for explanation).

— The *medoid* is the most representative season (TS) of a cluster. It is a real, tangible TS and it could enhance interpretability.

These prototypes represent typical seasons, they will be called $\{\hat{S}^1,\ldots,\hat{S}^p\}$.

3. **Next-season group learning:** A classifier is then trained. The input of this classifier contains the values of γ consecutive seasons in Y (*i.e.* D_i, $i \in [x-\gamma+1,x]$).

Once learned, the model may be applied several times for several forecasts without the need of training it again. It should be noted that only the first 85% of the data are used for learning in the experimental setup.

3.3.2 Forecasting process

Once a couple $(clusterer, classifier)$ has been learned, it can be used to make forecasts from the last γ seasons.

First, a single prototype per cluster is computed (see above in Sec. 3.3.1 for more details). To predict season $x+1$, prototypes are needed together with the γ previous seasons (*i.e.* D_i, $i \in [x-\gamma+1,x]$).

The prediction of the next season is generated from both these prototypes and the vector of probability. The forecasting process is composed of three steps (see Fig. 3.5):

1. **Classifier feeding:** The last D_i, $i \in [x-\gamma+1,x]$ are given to the classifier.
2. **Next-season group estimating:** Two paradigm may be followed: *hard method* and *soft method*.
   — *Hard method*: the output of this classifier is the typical season $\hat{K}_{x+1}$ (an integer between 1 and p) of the next season (*i.e.* the one that occurs right after the γ consecutive seasons of the input vector).

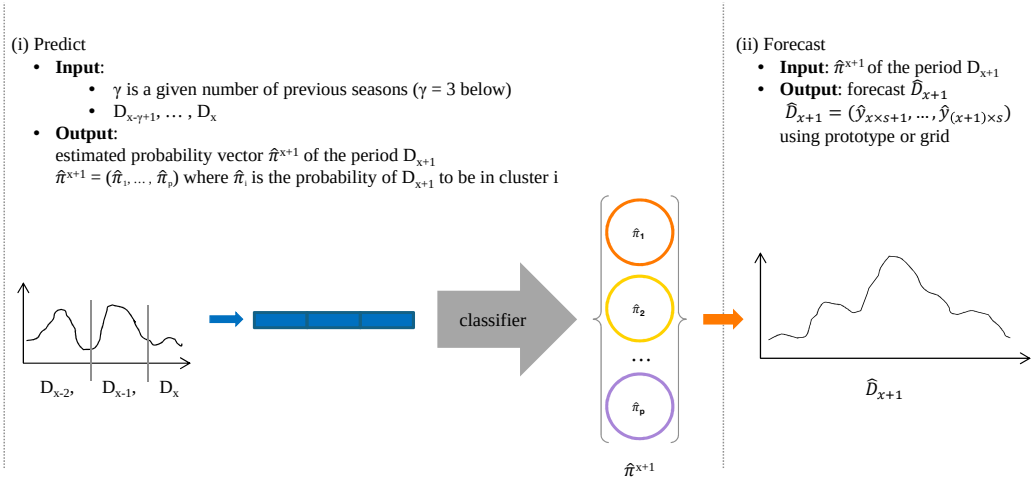


Figure 3.5: Illustration of the forecasting process (see text for explanation). γ equals 3.

— *Soft method*: classifier with probabilistic outputs may be used. In other words, rather than just predicting $\hat{K}_{x+1}$, the classifier outputs a vector of probabilities $\hat{\pi}^{x+1} = \{\hat{\pi}_1, \ldots, \hat{\pi}_p\}$, where $\hat{\pi}_i$ represents the probability that the season $x+1$ is in cluster C_i given the input data.

3. **Forecast computing:** compute $\hat{D}_{x+1}$ the predicted season using the set $\{\hat{S}^1, \ldots, \hat{S}^p\}$ of prototypes and $\hat{\pi}^{x+1}$.
   — For hard methodology: $\hat{D}_{x+1} = \hat{S}^{\hat{K}_{x+1}}$
   — For soft methodology: $\hat{D}_{x+1} = \sum_{i=1}^{p} \hat{\pi}_i * \hat{S}^i$

Depending on the kind of algorithm used for clustering, the kinds of prototypes and the classifier, the framework leads to different forecasting solutions.

3.4 Data used for experiments

The data used for experimentations is presented exhaustively in a table in Appendix 7.2. Note that the datasets presented are used along the entire thesis for all experimentations. There are 49 TS used at testing time, from various sources such as the time-series data library [Hyn11], two datasets from Orange Money projects (those will be studied more in depth in the last chapter of this thesis in Chap. 7), from the cities of Porto or Melbourne, etc. Obviously, the TS used in this thesis have been selected because seasonality was identified, and several number of points per seasons were used. The Fisher test presented in Sec. 2.1.1 is used to validate the size of the seasons used at learning time. Each TS is normalised using a z-normalisation prior to data splitting, in order to have comparable results.

3.5 Metrics and performances assessment

For any algorithm, once the forecast is generated, the forecaster need to *assess the quality of predictions*. Assessing the performances of algorithms is interesting: it allows one to *select the most accurate model* for a given task, by comparing performances of all the tested models during hyperparameter tuning. Metrics

are also important for the end users to *see how reliable is a forecasting algorithm before starting to use it.* It is not because such algorithm is better than all other ones for one particular task that the results are necessarily useable, because the performances might still be too low regarding to the needs and standards associated. For some application (*e.g.* spaceship landing or plane trajectory, *etc.*), it is required to have a very low error because of their critical aspect, for others having a biased trend for the data is sufficient.

This section first presents typical setups and methodologies used by data scientists to assess quality of forecasts, in Sec. 3.5.1. Then, the two metrics used for deterministic forecasting are presented 3.5.2. The metric used for probabilistic forecasting is presented in Sec. 3.5.3. One diagram used for comparing ranks of methods is presented in Sec. 3.5.4. Finally, win/lose diagrams used for comparing two given methods side by side using an arbitrary numerical metric are presented in Sec. 3.5.5.

3.5.1 Typical setup for performance assessment

The first step for computing model performance is to *optimise its hyperparameters.* Bad parameters settings can lead to bad forecasts and thus bad performances. For ARIMA, the main parameters are only three (p, d, q), but for some other, more complex models, they can be numerous: sometimes thousands, or even millions for very complex neural networks.

One classical way to find the best hyperparameters (that will be used in the experiments of this thesis) is to split the TS at hand into several ensembles.

— One training ensemble will be used to train the algorithm and usually represents 70% of the data.

— One validation ensemble will be used in the process of parameter tuning, before actual tests, and usually represent 15% of the data. Models are trained with the training data, forecasts are made on the validation data, and forecasts are compared to known values.

— One test ensemble is finally used in the actual performance review process, and usually represents 15% of the data. When hyperparameters are optimised, it could be interesting to broaden the learning ensemble by mixing the train and validation ensembles, in order to create a training ensemble of size 85% of the data.

If performances at testing time are acceptable, the tested algorithm may be used 'in the real world' with 100% of the data for training, and forecasts made for the actual future. In this case, it is obviously only possible to compute performances when the real data becomes available. If the performances are bad, one could attempt to improve hyperparameters or to check in the data if something has changed that could cause the forecasts to be biased.

In some cases, it is interesting to shuffle the data for less biased learning. It mostly depends on the underlying task. In the case of TS forecasting and because of the strong correlation between temporal values, it is not necessarily relevant, as it is important to learn the relations between contiguous values and the shuffle would remove those relations.

3.5.2 Metrics for deterministic forecasts: MSE, MAE

Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics are the most standard metrics available. They are not specific to the TS prediction problematic. They are both fairly easy to understand and to compute, and are really used for lots of machine learning applications.

The MSE measures the average of the squared prediction errors. Let $y = y_1, \ldots, y_p$ be a vector of ground truth values and $\hat{y} = \hat{y}_1, \ldots, \hat{y}_p$ the corresponding prediction vector. The MSE between y and $\hat{y}$ is defined as

$$MSE(y, \hat{y}) = \frac{1}{p} \times \sum_{j=1}^{p} (y_j - \hat{y}_j)^2$$

The MAE between y and $\hat{y}$ is defined as

$$MAE(y, \hat{y}) = \frac{1}{p} \times \sum_{j=1}^{p} |y_j - \hat{y}_j|$$

As these two measures represent prediction errors, the lower they are, the more accurate the predictions.

If the need to compare performances between two different datasets arises, it is judicious to normalise the data with the same function because MSE and MAE are highly dependant on the real value carried by the TS.

The MAE will be preferred for performance comparisons in this thesis, because it will facilitate the comparisons with probabilistic algorithms (see CRPS metric in Sec. 3.5.3 just below).

3.5.3 Metric for probabilistic forecasting: CRPS

Continuous Ranked Probability Score (CRPS) is often used for probabilistic forecasts assessment. For computing the CRPS, the notion of *Cumulative Distribution Function* (CDF) must be recalled. It is the probability that the studied variable takes a value less than or equal to a given threshold x. An example of a CDF could be observed on Fig. 3.6.

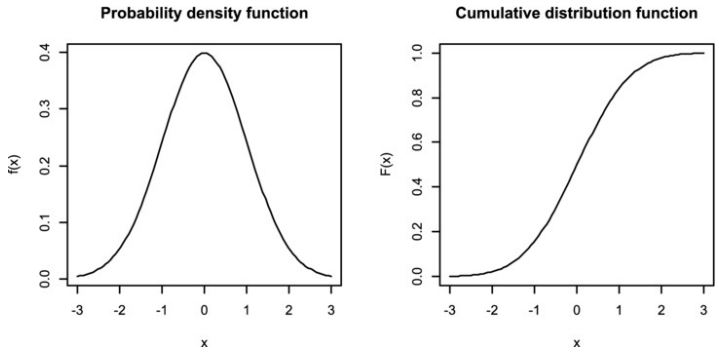


Figure 3.6: Probability Density Function vs Cumulative Distribution Function.

In the probabilistic model, $\forall i \in [1, \ldots, h]$, $\hat{y}_{n+i}$ is a probability distribution, *i.e.*: $\hat{y}_i : \mathbb{R} \to [0, \ldots, 1]$. Let F_i be the cumulative distribution function (CDF) of $\hat{y}_{n+i}$ and x the observation:

$$F_i(x) = P\left[\hat{y}_{n+i} \leq x\right]$$

The CRPS metric for a cumulative function is

$$CRPS(F_i, x) = \int_{-\infty}^{\infty} (F_i(t) - \mathbb{1}(t-x))^2 \, \mathrm{d}t$$

where $\mathbb{1}$ is the Heaviside step function and denotes a step function along the real line that reaches the value of 1 if the real argument is positive or zero, the value of 0 otherwise.

The metric for the prediction of the whole TS $\hat{y} = (\hat{y}_{n+1}, \ldots, \hat{y}_{n+h})$ is

$$CRPS(F_i, \hat{y}) = \sum_{j=1}^{h} CRPS(F_i, \hat{y}_{n+j}) * \frac{1}{h}$$

As CRPS represents prediction errors, the lower they are, the more accurate the predictions. Note that the CRPS is equivalent to the MAE [Her00] for deterministic forecasts.

3.5.4 Critical diagram

Critical diagrams (CD) [Dem06] have been introduced to compare method on several independent datasets in terms of ranks. It is useful to compare several methods where it is meaningless to mean the metric results because of the use of different scales for different datasets.

It is based on Nemenyi tests. An example might be found in Fig. 3.7. They compare on a graduated scale the average ranks obtained by several competing methods. The value of the evaluation criterion is not explicitly taken into account, but the rank (such as podium places) obtained for each method and for each dataset. Then, the ranks obtained over all datasets are averaged and displayed on the scale.

The critical diagram is displayed along with the critical difference in rank. Critical diagram is the minimum rank difference that allows to state that two approaches have significantly different performances. The critical difference is calculated and depends in particular on the number of datasets. This critical difference is used on the critical diagram to link together approaches that are not statistically discernible (hence the horizontal lines below the graduated scale). In the Fig. 3.7, results for method 1, 2 and 3 are not statistically different, but the performances of method 4 are definitely poor.

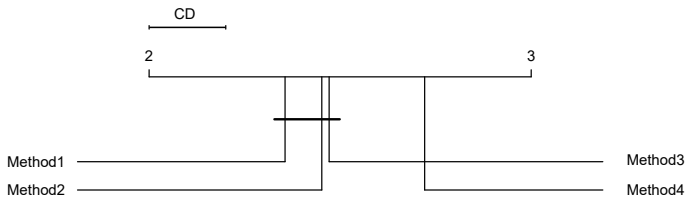


Figure 3.7: An example of a critical diagram. It compares the average ranks of the different methods (from Method1 to Method4).

3.5.5 Win lose diagram

The critical diagrams presented just above are interesting to compare ranks of methods but they are not used directly to visualise the numerical performances. For this purpose, the 'win lose' diagrams are introduced. An example is provided in Fig. 3.8. The goal is to make pairwise comparison of the performances of methods. In the case of Fig. 3.8, the two methods compared are method 1 and method 2. The metric used (MAE here) of the two selected methods is plotted against each other for each dataset. A line which separates the graph in two is shown. Therefore, each point plotted below the line is in favour of method 1, each point plotted above in favour of method 2, one point representing the results on one given dataset. This type of diagram enables a visual analysis of the results, and plot real performances (not only rank), which is complementary from the analysis conducted with CD. On those plot is also shown the p-value. It represents the significance of the difference between the two compared methods.

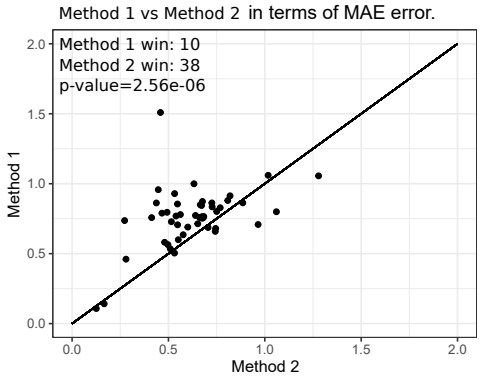


Figure 3.8: Win/lose graph that shows the number of times method 1 won against method 2.

3.6 Wrap up

This chapter presented the framework developed in this thesis. This framework is composed of one learning step and one forecasting step. The learning step is itself composed of three steps, and rely on clustering and classification algorithms. The forecasting step is also composed of three steps that use the groups of the season created during learning time.

The assumption of seasonality in the time series makes the framework intuitive. More especially, grouping seasons using clustering is a non-supervised way of discovering several seasons types, and this information may be used later on at prediction time by training classifiers.

This framework might be implemented with various clustering algorithms and various classification algorithms. This chapter did not give any details about possible approaches, because the next chapters are dedicated to this task. Three different implementations will be assessed in the next three chapters.

Chapter 4

Baseline deterministic approach

In this chapter, a first baseline of the framework presented in Chap. 3 is studied. It combines the use of K-means clustering algorithms (that will create groups of seasons) and Markov Models (that will model the temporal relation between seasons and act as a classifier). The choice of the two aforementioned algorithms is natural for developing a baseline because they are widely used. This baseline allows validating some assumptions introduced in Sec. 1.3.3 about the data and the problem itself.

The baseline implementation details are given in Sec. 4.1. The interest of this first framework implementation is demonstrated with experimentations in Sec. 4.2. The performances of the method are compared against another baseline (mean season) and four state-of-the-art methods (autoregressive model, ARIMA, SARIMA and Holt Winters), on a collection of open source datasets. Sec. 4.3 finally concludes and explicits some of the limits in the method, then gives an idea of possible enhancements of the baseline that will be developed in next chapters. This baseline has been published in the AALTD workshop at ECML 2018, Dublin [Lev+18]. Experiments have been extended for the thesis.

4.1 The FC2M implementation

The version of the framework presented in this chapter is called 'Forecasting with Clustering and Markov Models' (FC2M). For extracting knowledge about the various seasons, a solution is to use clustering algorithms. One of the most used clustering algorithms is K-means and it is appropriate for a baseline. For learning sequences and understand how those seasons are chained in time, a solution is to use Markov Chains [KS76]. The intuition is that there is an underlying structure in the sequences of the σ_i introduced in Def. 11, and thus that $\sigma_{i+1} \sim f(\sigma_j, j < i)$. The Markov Chain is a simple linear model of the sequential relation between the types of seasons. The Markov Models are then a straightforward solution to model these sequences. They can be used in replacement of the classifier depicted in Chap. 3 for a simple learning. This section presents the approach to produce one-season-ahead TS forecasting using K-means and a first order Markov Models.

This approach is an instantiation of the framework presented in Chap. 3. Therefore, Sec. 3.3 and Fig. 3.4 and 3.5 (respectively for learning and forecasting steps) are used for explanations, the specific implementation is exhibited below.

4.1.1 Learning process

The learning of the framework is depicted in Fig. 3.4.

1. **Data splitting:** First, the seasons are split. The splitting is straightforward and compliant with the original framework presentation.

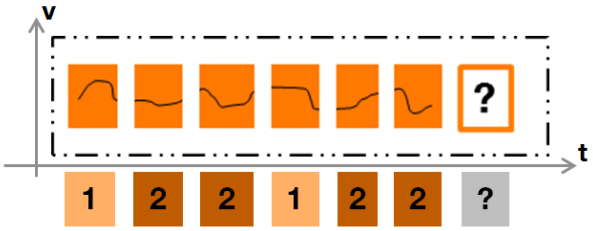


Figure 4.1: Encoding a seasonal TS using the cluster occurrences of the seasons.

2. **Season grouping:** The elements of $\mathcal{D}$ are then given to a clustering algorithm. In this case, the multidimensional K-means based on the Euclidean distance is used. The prototypes used in this baseline are the centroids. They are denoted $\{S^1,\ldots,S^p\}$. They correspond to typical seasons. The choice of the number p of clusters will be discussed in Sec. 4.2.
3. **Next-season group learning:** Here, the classifier originally depicted in Sec. 3.3 is implemented by a first-order Markov Model. The set $\mathcal{D}$ is encoded into $K_1,\ldots,K_m$, where $K_i \in [1,p]$ corresponds to the index of the cluster that contains the season D_i (as seen in the Figure 4.1). This sequence is modelled by a Markov Model of transition matrix $\Pi = (\pi_{i,j})$, where $\pi_{i,j}$ is the probability that the next season belongs to cluster j given that the current season belongs to cluster i. As seen in Fig. 4.2, this Markov Model enables the estimate of the most probable cluster to which will belong TS next season.

This first baseline aims at verifying the assumptions about the interest of the seasons, and more especially *the interest of modelling seasons chronological sequences*. Thus, the only information required is the labels attributed by clustering algorithms and no numerical information about the TS itself. Markov Chains are then more indicated for this need to learn seasons occurrence. Numerical information such as TS values will be added to the models in next versions of the framework.

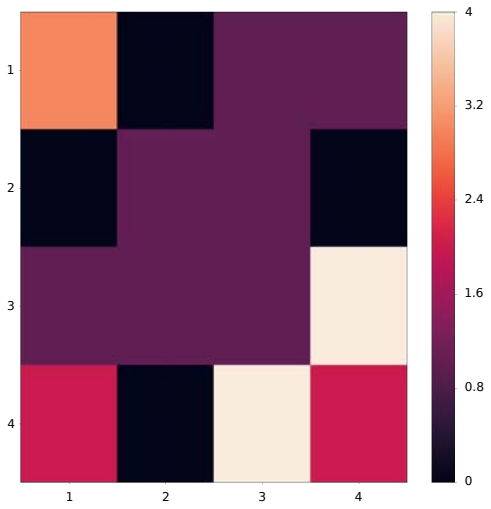


Figure 4.2: Matrix which depicts the probabilities linked to one real Markov Model. The lighter the colour, the more probable to jump from one cluster type (left) to another cluster type (down) for incoming season. For example, reading this graph indicates that the probability to jump from cluster 3 to cluster 4 is pretty high, as well to jump from cluster 4 to cluster 3. However, there is practically no chance to go from season 2 to season 1.

4.1.2 Forecasting process

This forecasting step (described in Fig. 3.5) uses the model learned above to predict the next season measurements ($\hat{D}_{x+1}$) given the current season measurements (D_x).

1. **Markov Model feeding:** First, the index of the closest cluster to D_x is computed, and denoted K_x.
2. **Next-season group estimating:** The most probable cluster for the next season is estimated using the transition matrix $\Pi : \hat{K}_{x+1} = \arg \max_{j \in 1,\dots,p} \pi_{K_x,j}$.
3. **Forecast computing:** Finally, the forecasting of the next season is given by the centroid $\hat{S}^{\hat{K}_{x+1}}$.

4.2 Experiments

In this section, results obtained with the FC2M methodology are commented. It is reminded that Sec. 3.4 of previous Chap. 3 introduces some preliminary details about the protocol followed for experiments for all versions of the framework presented in this thesis.

The entire code of the framework was developed in Python 3.5. For developing the K-means clustering, the library *tslearn* [Tav+20] was used. The Markov Models were handcrafted. The only hyperparameter optimised during learning was p (number of clusters). The validation dataset is used to select the best number of clusters p (ranging from 2 to 200) based on the MAE retrieved from the forecast itself.

4.2.1 Opponents

This study has been conducted using univariate TS (presented in Sec. 3.4, datasets listed in Appendix 7.2), where the next season is forecasted using only past data from current TS. The performance of the approach gives indication about the two-order temporal scale of the TS.

In this chapter, one baseline (mean season) and four classical forecasting techniques (autoregressive model, ARIMA, SARIMA and Holt Winters) are used to compare the quality of the forecasts given by the approach:

Mean season Mean season is a simple baseline mean season calculation: all season present in learning ensemble $\mathcal{D}$ are used to compute the average season. This latest is then given as a forecast result for next season. Whatever the history, their method always predicts the same season.

Autoregressive (AR) model [Llo82] [Aka69] AR model is a representation of a random process that can be used to describe some TS.

(S)ARIMA [Box+15] ARIMA and its seasonal counterpart SARIMA. An ARIMA or SARIMA model is trained with the training set, and this model is applied to forecast the seasons of the test set (given the past). Models are fitted using the `auto.arima` method that selects the models that best fit the training data according to an information criterion.

Holt Winters (HW) [Win60] Holt Winters Triple Exponential Smoothing is a rule of thumb technique for predicting TS data. It extends the Holt's method to capture seasonality.

4.2.2 Results

Fig. 4.3 shows the Critical Diagram (CD) for the MAE for all the methods cited in Sec. 4.2.1. CD have been presented in Sec. 3.5.4 and show the mean rank of the methods. It is noticeable that the FC2M method outperforms three opponents that are Holt Winters, ARIMA and AR method (although not by a significant margin). It can be explained by a weak adequacy of these last methods to the task. AR, ARIMA and HW are more used to forecast next few points of the TS while this task requires to forecast the TS for the entire season (several points). Fig. 4.4 confirms the good performances of FC2M against AR and its bad performances against both MEAN and SARIMA. A point above the segment means good performances for FC2M method.

It is also interesting to observe that the mean season baseline was able to outperform AR, ARIMA, Holt Winters and FC2M. It can be explained by the fact that mean seasons are representative enough of the next season to come so that the MAE is effectively low compared to methods that are not adapted to seasonal data (*e.g.* AR, ARIMA and Holt Winters) or methods that make more clear-cut decisions about the next type of season to come (*e.g.* FC2M), which could cause greater errors in case of bad next season type prediction.

Finally, the good performances of SARIMA model are shown. This method is state-of-the-art for seasonal TS forecasts and is the opponent to beat for next implementations of the proposed framework. Next Sec. 4.3 discusses how FC2M could be improved to do so.

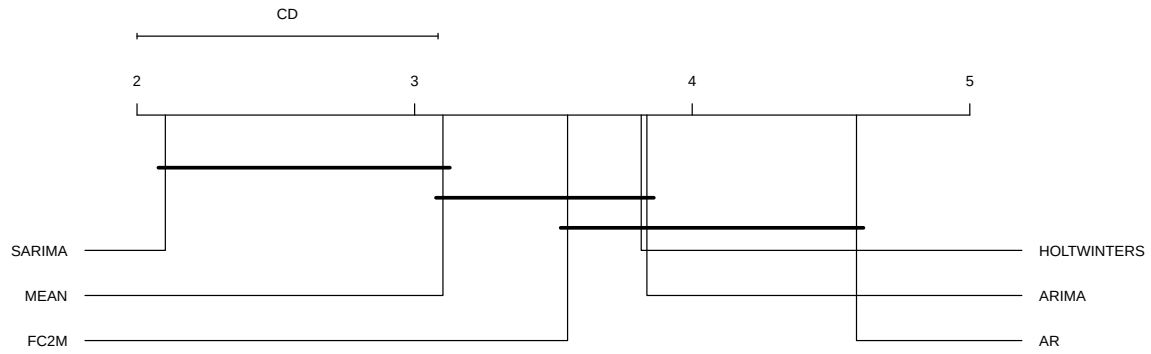
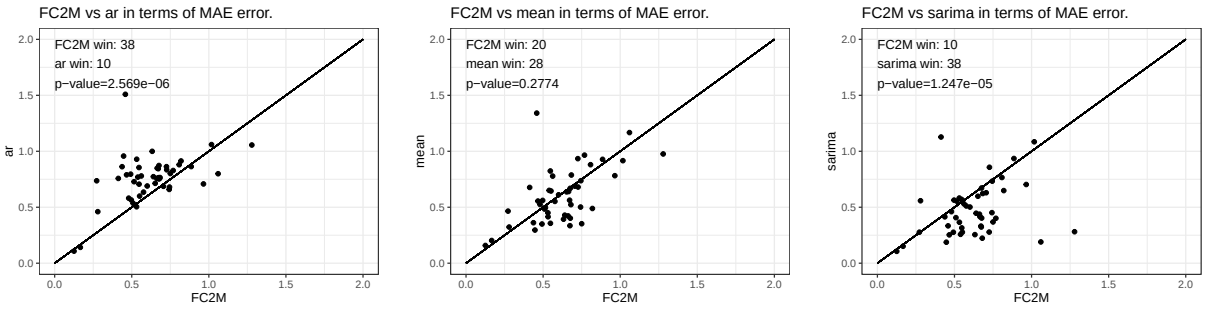


Figure 4.3: Global ranks for the FC2M framework against four opponents (AR, ARIMA, SARIMA, Holt Winters) and one baseline (MEAN) using the MAE metric.



(a) FC2M vs AR (b) FC2M vs MEAN (c) FC2M vs SARIMA

Figure 4.4: Win/lose graphs that show the number of times FC2M won against three selected opponents ordered by performances: AR, MEAN and SARIMA.

4.3 Discussion

Experimental results show that the approach performs better than three of the state-of-the-art methodologies used for comparisons (AR, ARIMA and Holt Winters) on 49 real dataset (see Appendix 7.2), but it is an early proposal that has some limitations and weaknesses.

It would be interesting to know which of the learning steps described on the Fig. 3.4 is lowering forecast performances. This could help in improving the chain by tuning well-identified steps and parameters. In the first stage of the model, seasonal TS are clustered. Enhancing the quality of clustering could be a key to better results.

— For this first version of the framework, experiments have been made with the K-means algorithm with a Euclidean distance. In practice, if detecting temporally aligned data peaks is something of interest, this distance is to be privileged. But if the goal is to detect peaks in a season without precise information about their timings, DTW [Nie04] is a better candidate. There is not a unique good choice but some choices that fit the data characteristics like Douzal-Chouakria et Amblard [DA12] suggest it for a classification task.

— In addition, the clustering strategy could also be evaluated. One of the weaknesses of the current approach using a K-means algorithm is the critical choice of p. In this study, the optimal number of clusters is found using the validation set. Various K-means sizes are tested on the training dataset, and select the one which helps the algorithm in having a lower MSE. Fewer empirical techniques such as David et Bouldin criteria [DB79] or even Silhouette [Rou87] could help to select a priori the best number of clusters with a lower computational cost.

In the second stage of the model, guessing the next season cluster type accurately is paramount for generating good forecasts. A more accurate solution than Markov models can enhance the centroid selection at forecasting time.

— In this chapter, a Markov model takes only into account the season before the one to predict. This simple model has been preferred to higher order Markov models because of the required quantity of training data. With 7 different seasons, transition matrix of size 49 are to learn in this case, but of size 343 for a 2-order Markov model. This requires long sequences of seasons to accurately estimate them. All the datasets at hand are not long enough to perform this kind of computation. This raises the question of the *number of seasons to take into account at learning time*, thus the question of the *importance of the γ parameter* (see Sec. 3.3 for more details about γ).

— It is also noticeable that despite the label of the group of next day, no numerical data coming from the TS itself is used at learning time. It seems logical to think that having this knowledge is crucial for guessing next season type, as in TS, the next values are often influenced by past values. Replacing the Markov Chains used in this baseline implementation with classifiers that are able to take more data at learning time (including the data from the TS itself) could also be interesting for better performances.

4.4 Wrap up

In this chapter was presented FC2M, the first implementation of the framework which addresses a specific forecasting problem, *i.e.* the forecast of the evolution of seasonal time series indicators a season ahead.

The method proposed is a time series forecasting framework that is based on the assumption that the time series are implicitly structured as a sequence of typical seasons. The experiments comparing baseline approaches and classical time series forecasting methods to the FC2M framework show that this assumption is verified by most of the time series of the dataset. The FC2M methodology is better than 3 out of 5 opponents when comparing mean ranks using the MAE metric.

Although, the proposed method may benefit from improvements in the two main stages: the clustering of time series and the next type of day prediction. Markov models may be improved by some more recent works on sequence prediction [BV18], or by their replacement with classification algorithms. This latest point is the subject of the next Chap. 5.

Chapter 5

Deterministic approach

This chapter aims to enhance the first implementation of the framework proposed (FC2M), and to introduce a second version of it. In particular, the performance of *various clustering algorithms* for grouping seasons in input time series will be assessed. To address the first limitation of the framework's initial implementation presented, *i.e.* the *lack of options concerning the way groups of seasons are created*. Experiments have only been made using K-means. It is the most widely used clustering algorithm, consequently a first choice for a baseline. But the problem of grouping time series is complex and many other methods have been proposed. It would be interesting to adapt the grouping method to the dataset considered for having the best measure of similarity. Experimenting with non-Euclidean based algorithms and also with time series-specific algorithms would help to strengthen the grouping process.

One second limitation is the *relative simplicity of the Markov Chains for modelling the occurrence of types of seasons*. Indeed, it only takes into account clustering labels of the seasons and nothing more. Once again, this simplicity was an asset for developing a baseline but it has clear limits. Adding more information about the season's composition might bring knowledge and make the forecasts of next days more accurate. This improvement is possible when using classifiers that are able to deal with those new sources of data. The specific problem of seasonal time series forecasting could then be *framed as a classification problem*. Therefore, Markov Chains will be *replaced by classifiers* for learning the sequence of days.

First Sec. 5.1 presents the framework new implementation. The various elements used in the framework (clustering algorithms and classifiers) are introduced. Sec. 5.2 shows the experiments conducted to assess this new approach. Comparisons between various parameter combinations are made, along with opponents comparisons. Final Sec. 5.3 concludes on the advantages of the framework, its flaws, and gives clues about next studies that will be conducted in last chapters. This version of the framework has been published in the IDEAL conference 2019, Manchester [Lev+19]. Experiments have been extended for the thesis.

5.1 The F2C implementation

The deterministic framework presented in this chapter is called 'Forecasting with Clustering and Classification' (F2C). The general sketch of this second approach is analogous to the one presented in Chap. 3. Therefore, Fig. 3.4 and 3.5 (respectively for learning and forecasting steps) originally found in Sec. 3.3 are used. The specific implementation is shown below.

5.1.1 Learning process

The learning stage is composed of three steps (see Fig. 3.4):

1. **Data splitting:** The data splitting step consists in constructing a set of m seasons. It is assumed that observations of $\mathcal{D}$ are independent. This step is unchanged compared to Chap. 3.
2. **Season grouping:** The m elements (seasons) of $\mathcal{D}$ are given to a clustering algorithm that gathers similar seasons into p groups of typical seasonal TS. In this implementation, four algorithms have been assessed: K-Means, K-Shape, GAK and MODL. Also, two prototypes have been used: medoids and centroids. More details are given in Sec. 5.1.1 page 76 below.
3. **Next-season group learning:** A classifier is then trained to estimate, using the knowledge of the TS, the expected group of the next season. In this implementation, the number of past and known seasons used for learning the classifier is $\gamma = 1$ and a unique season D_x is being used. Also, four classifiers have been tested: naive-bayes, decision trees, random forests and logistic regression. More details are given in Sec. 5.1.1 page 77 below.

Clustering step: details and algorithms

In this step, a clustering algorithm is used to group the m seasons of $\mathcal{D}$ into p groups. The choice of p will be discussed at the end of this section; in the following, p is given. A representative series is computed inside each group. These series represent typical seasons that occur in the dataset. Hence, at the end of this step, every season of $\mathcal{D}$ can be assigned a label (that represents in which group it has been clustered) and p typical seasons are computed. In this chapter, four different clustering algorithms are studied: K-means, K-shape, GAK and MODL.

K-Means for TS [Llo82] K-means algorithm aims at creating a partitioning of the data by minimising the intra-cluster variance. The use of K-means implies the use of a distance measure between two TS. Two of the major distance measures available for TS are Euclidean, and Dynamic Time Warping (DTW) [PKG11]. The Euclidean distance is used for experiments. K-means is essential in the clustering literature and including it in the experiments was natural, even if it is not fully dedicated to TS clustering. K-means was the clusterer used in the FC2M implementation of the framework presented in Chap. 4.

K-Shape for TS [PG15] K-Shape algorithm uses a distance measure based on a normalised version of the cross correlation (invariant to TS scale or shifting). It can be seen as a K-means algorithm but that uses a shape-based similarity measure. It is finally important to note that this algorithm is notably known for being dedicated to TS clustering, thus the choice of adding it to the experiments.

Global Alignment Kernel K-means (GAK) [Cut11] Kernel K-means is a version of K-means that computes the similarity between TS by using kernels. It identifies clusters that are not linearly separable in the input space. The Global Alignment Kernel is a modified version of DTW. This clustering algorithm was included because it does not use Euclidean distance, and also because it is dedicated to TS clustering. Experimenting with various cluster similarity measure is important, especially because Euclidean distance have some limits, especially when the TS at hand shows some slight offsets (see Sec. 2.1.2 for more details about distance measures for clustering).

Clustering with MODL [Bou12] MODL is a nonparametric method that uses a piecewise constant density estimation to group similar TS. TS are partitioned into clusters and the TS values are discretised into intervals. The cross-product of these discretisation is an estimation of the joint

density of the TS and points. This clustering algorithm was picked because of its innovative distance measure based on TS density, and also because the choice of number of clusters p is fully data driven, and there is no need for the user to provide p. For now, MODL is used as a clustering algorithm. Coclustering aspects will be introduced in next Chap. 6, in Sec. 6.1.2.

The choice of the number of clusters is of particular importance. A tuning approach is used: for each candidate number of clusters, the training set is used to build the overall model. This model is then used to predict the seasons of the validation set. The number of clusters that leads to the lowest error on the validation set is selected.

For K-means, K-shape and GAK algorithms, candidates number of cluster are systematically chosen in a pre-defined range [2,300]. Partitions with empty clusters are discarded. On the other hand, MODL coclustering estimates in a nonparametric way the best number of clusters for each input TS. The number of cluster finding procedure is regularised. This estimated number then usually leads to the best description of input TS, regarding to the coclustering task. However, this model can be simplified to reduce the number of clusters. From now on, the procedure is similar: different numbers of clusters are evaluated and the overall model that leads to the lowest error on the validation set is kept.

Predicting the cluster index of the next season

A classifier is then trained to predict, using the knowledge of the current season D_x, the expected group of the next season. To train this classifier, index of groups created at learning time as described above are used. A learning set is then created to feed the classifier: each line of the learning set corresponds to TS values of a season D_x (explanatory variables) and a target variable which corresponds to the group of the next season (the group of season D_{x+1}). Including more data in the classifier would probably be beneficial: data about past few days, and not only the last one, data about national holidays, and maybe other exogenous data related to the problem at hand.

In the Fig. 3.5, $\hat{\pi}^{x+1}$ denotes a vector of probabilities (*i.e* $\{\hat{\pi}_1,\ldots,\hat{\pi}_p\}$). Four different types of classifiers are investigated: naive-bayes classifier, decision trees, random forests and logistic regression.

Naive-bayes [Ris+01] First introduced in the early 60ies in [Mar61], naive-bayes classifiers make the assumption that the features of the data are totally independent. Although independence is generally a poor assumption, in practice naive-bayes often competes well with more sophisticated classifiers.

Decision tree [SL91] Decision trees process the data in the form of trees which are composed of leaves and nodes. The data is described with binary statements and the path created by answering those binary questions while traversing the tree is followed to find most probable outcomes.

Random forest classifiers [Ho95] Random forest are an ensemble technique that creates different random trees to improve the overall accuracy. They combine tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [Bre01].

Logistic regression [Kle+02] Logistic Regression is a linear classification model which is the counterpart of linear regression. The aim is to determine how a set of numerical properties (numbers) can be associated with one of the classes under consideration.

Those four classifiers were selected because they are standard in the community, widely used and are also able to provide probabilities vectors to appear in each class. Due to the various sizes of TS processed,

a minimum size of learning for classifier cannot be guaranteed. Therefore, the use of neural networks classifiers, which would require a massive amount of data for training, was not implemented.

5.1.2 Forecasting process

To forecast the next season, the classifier learned to predict the group index of the next season plays a central role. It uses the knowledge of the current season and of representative series of the different groups to generate the prediction of the next season.

The forecasting of the next season is done in three steps (see Fig. 3.5).

1. **Classifier feeding:** First, the current season (D_x with $\gamma = 1$) is given to the classifier.
2. **Next-season groups estimating:** Then, this classifier computes the probabilities $\hat{\pi}^{x+1}$ of the next season to belong to each group ($\{\hat{\pi}_1, \ldots, \hat{\pi}_p\}$).
3. **Forecast computing:** Finally, next season is predicted. Two paradigms are studied in this chapter:
   — *Hard method*: season $x+1$ is predicted as the prototype of the most probable cluster, *i.e.* $\hat{S}^k$ with $k = \arg\max_{1 \leq i \leq p} \hat{\pi}_i$.
   — *Soft method*: season $x+1$ is predicted as a weighted combination of the prototypes. The weights correspond to the probabilities in $\hat{\pi}^{x+1}$.

   Experiments are being made with centroid and medoid prototypes.

5.2 Experiments

During the experiments were investigated (1) the impact of the choice of both the clustering and the classification algorithms on the performance of the overall method, (2) the performances against competitors and (3) the impact of parameters γ.

The entire code of the framework was developed in Python 3.5. For developing the TS clustering, the library *tslearn* [Tav+20] was used. It provides access to most of state-of-the-art TS clusterer, and is, furthermore, a useful toolset for TS processing. For developing the classification, the library *sklearn* [Ped+11] was used. For the developing the coclustering algorithm, *Khiops* [Bou16] was used.

In the learning process, the number p of clusters is found using the validation ensemble, as stated in Sec. 3.4. All other hyperparameters stayed identical during all the process. The sole optimisation of the number of clusters was already time consuming. Adding other parameters in the hyperparameter optimisation process would really increase the learning time. The parameters provided for the clusterers and classifiers are given below for information. Note that for the sake of reproducibility, a fixed random seed was used.

TimeSeriesKMeans: the used distance metric is Euclidean.

RandomForestClassifier: the max depth is two and the number of estimators is 100.

DecisionTreeClassifier: the criterion used is gini, the max depth is 96 and min samples by leaf is 5.

LogisticRegression: the solver is 'lbfgs' and the classifier multinomial. The maximum number of iterations is 1000.

5.2.1 Best parameters for F2C method

Prototypes and prototype mixing paradigm study

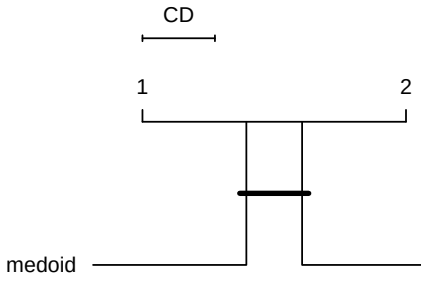
The results shown in this section concern all the possible framework parameters mixed together (*e.g.* all the clusterers, all the classifiers, for all found values of p) for the validation ensembles. Those results are used to find the best parameters overall – for all possible use cases and for all possible clusterer and classifier combinations. It is recalled that the CD figures compare mean ranks of methods, thus the lower is the better.

As seen in Sec. 5.1.1, it is possible to use two prototypes for generating predictions out of clusters: centroids and medoids. The Subfig. 5.1 (a) is a critical diagram which describes the accuracy of the forecasts generated using those two prototypes. It shows that better results are observed using medoids – not significantly.

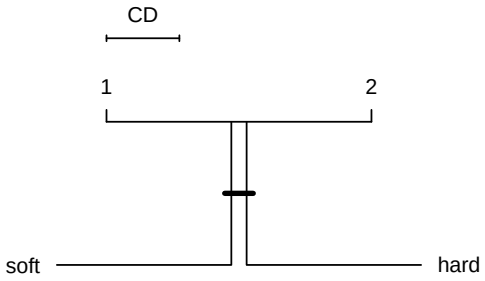
Centroids and medoids are indistinguishable but medoid show slightly better results in average, therefore, the remaining of this experimental section Sec. 5.2 is written only using those, and all centroid results are discarded.

As seen in Sec. 5.1.2, it is possible to use two paradigms for mixing prototypes for generating predictions: hard and soft. The Subfig. 5.1 (b) is a critical diagram which describes the accuracy of the forecasts generated with both those paradigms. It shows that better results are observed using soft methodology – not significantly.

Soft and hard paradigms are indistinguishable but soft show slightly better results in average, therefore, the remaining of this experimental section Sec. 5.2 is written only using soft, and all hard results are discarded.



(a) Performance of the various prototypes for the F2C method.



(b) Hard methodology VS soft methodology, performances for the F2C method.

Figure 5.1: Performances of (a) various prototypes, (b) hard and soft methodology, overall for any combination of clusterer and classifier for the MAE metric. Subfig. (b) generated with medoid results.

Sixteen possibilities

The F2C framework uses one clustering algorithm along with one classification algorithm. Fig. 5.2 illustrates all possible combinations while using four clusterers and four classifiers. Any clustering algorithm

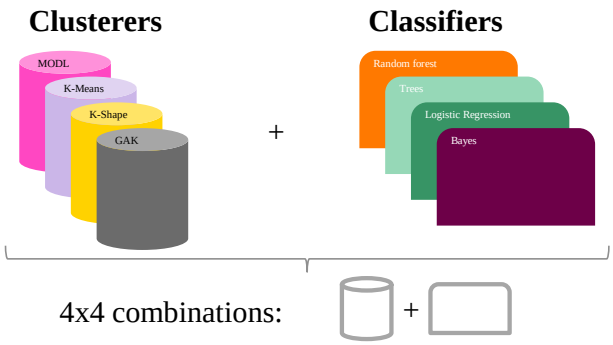


Figure 5.2: Illustrations of the four classifiers (rounded rectangles) and four clustering algorithms (cylinders) combined in the experiments.

and any classifier might be used as long as the classifier provides probabilistic decisions – it means that the number of possible combination is effectively greater than 16.

Impact of the clustering algorithms choice

Clustering is the first external element used in the chain. The performances of all the clusterers used during the F2C experiences are compared: K-means, K-shape, GAK and MODL. The experiment is made with all classifiers to avoid being biased by an especially strong clusterer-classifier combination.

Fig. 5.3 (a) shows that there is not significant differences between K-means and K-shape clusterer, but K-means is first despite the low statistical significance of the CD bound. Though, MODL is a bad choice for the extended deterministic F2C framework, despite the data-driven way to find the best number of cluster which would have been convenient during the learning phase. The relative superiority of K-means cluster is shown.

Impact of the classification algorithm choice

As explained above, an important part of the method is to predict the group (or cluster) of the next season. This prediction is made using a classifier. Once the group is predicted, a weighted combination of different centroids is used to make the forecast.

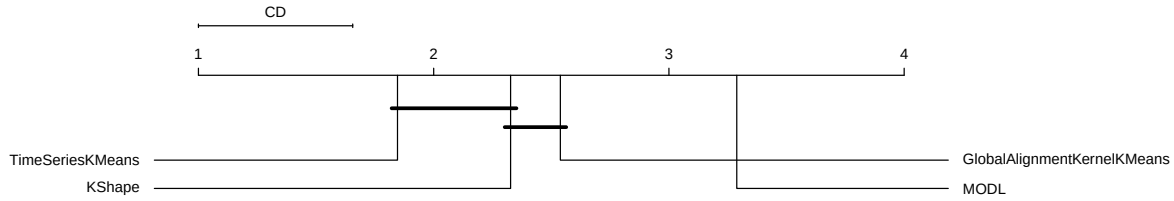
In this section, a performance comparison of the four considered classifiers (naive-bayes, decision tree, random forests and logistic regression) is made. For those tests, all clusterers results are used, because the interest is to assess the classifiers performances only. The critical diagram that compares the performance of the four classifiers in terms of MAE is given in Fig. 5.3 (b).

The random forests are elected as the more efficient classifier, closely followed by the naive-bayes, although results are once again not significant.

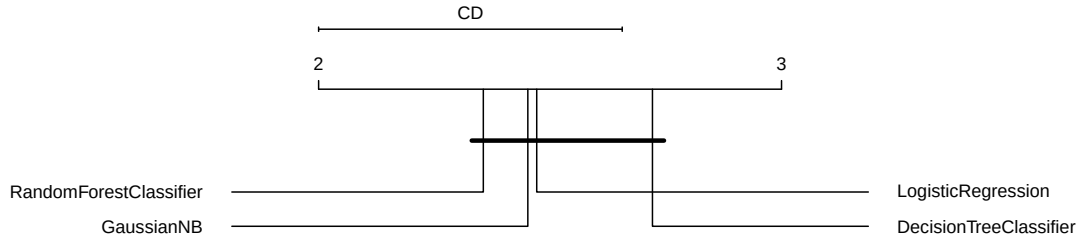
Wrap up

The sections above shown that the best parameters combination found for the extended F2C method are the following:

— K-means clusterer,



(a) Critical diagram, comparison of the average ranks of the difference clustering algorithms.



(b) Critical diagram, comparison of the average ranks of the different classification algorithms.

Figure 5.3: Performances of (a) clusterers, (b) classifiers, Fig. generated with medoid and soft results.

— medoid prototype,

— soft methodology,

— random forests classifier.

5.2.2 Comparison against competitors

This section aims at comparing the performance of the proposed approach (F2C) with the six following competitors prediction methods: mean season, autoregressive models (AR), ARIMA, SARIMA, Holt Winters and FC2M with Markov Model. All the opponents have been presented in Sec. 4.2.1 of previous Chap. 4. It is recalled that between the baseline (FC2M) presented in Chap. 4 and this version of the deterministic framework (F2C), the use of medoids and the soft paradigm have been introduced (respectively in Sec. 5.1.1 and Sec. 5.1.2). Improvements have also be made on the clusterers and the classifiers. Only the value of $\gamma = 1$ stays identical.

Fig. 5.4 in page 83 shows a first CD that exhibits the performance of the F2C methodology with the same configurations from FC2M (*e.g.* K-means clusterer, centroid prototype, hard paradigm), despite the classifier which was not a Markov Model but a naive-bayes. Best opponents (ARIMA and MEAN) are also shown in the figure – other opponents have been filtered for an enhanced readability. It shows that having a better classifier is important, as the performances of this non-optimised version of F2C still outperforms significantly FC2M.

Fig. 5.5 in page 83 compares performances of F2C methodology when it is well configured (F2C_BEST) with the most optimal parameters found in the selection above in Sec. 5.2.1 (K-means algorithm is used as the clustering algorithm with a soft way of mixing medoids, in conjunction with logistic regression classifier), and also when it is bad configured (F2C_WORST) with all the worst parameters possible (by taking the last parameters found in the critical diagram of Sec. 5.2.1 above, *e.g.* use of hard methodology,

MODL clusterer with centroids, and gaussian naive-bayes classifier). It shows that having a proper configuration process is key to better results, and that F2C is sensitive to parameter tuning.

Fig. 5.6 in page 83 shows a CD that demonstrates the performances of the real F2C methodology that have been properly configured (F2C_BEST), against all the opponents available. The figure shows that F2C outperforms significantly all the opponents, including and FC2M, and SARIMA – not significantly.

5.2.3 Results with various γ

Fig. 5.7 compares the performances of F2C methodology with a value of γ in $[1,2,3,4,6]$ ($F2C\_\gamma$). Despite the low significance of this CD plot, it is interesting to observe that the best value of γ is 2; however, it is also observed gradual better performances for values of γ of 1, 3, 4, and 6.

This figure is a good sign of the impact of the size of the historic, as results with $\gamma = 1$ (with only one season used for learning the classifier) are the worst observed. Also, it is known that the structure of the seasons highly depends on each dataset. Therefore, it is observed that a size of $\gamma = 2$ enables a better generalisation and fits a larger number of datasets.

A study of the specific datasets where prediction with a $\gamma = 2$ gave better results [1] was conducted but no common characteristic has been found. At this time, investigations did not identify some global best parameters for γ. In this case it is recommended to proceed through cross-validation to select the best parameter value.

5.3 Wrap up

In this chapter was presented F2C, the second implementation of the framework which addresses a specific problem of seasonal time series forecasting a day ahead. This implementation is still following the frame introduced in Chap. 3, but differs from the version developed in Chap. 4. More particularly, Markov Models have been replaced by classifiers and more clusterers have been tested.

A comparison of different clustering, different prototypes and different classifiers has been provided. A study on the impact of the parameters choices (*e.g.* choice of clustering algorithms, choice of classification algorithms) has been made. It shows that a K-means clustering combined with a random forest classifier gives better results for the deterministic forecasting task. Experiments also show that the proposed approach is competitive with other TS prediction methods and that SARIMA, the opponent to beat in last Chap. 4, was outperformed. It has been shown that the impact of γ is true. Furthermore, it should be noted that using a value of $\gamma = 2$ provides better results and appears to help generalisation of the algorithm for the datasets used in the study.

1. *e.g.* tide, PT08.S4.NO2, PT08.S3.NOx, NOx.GT, C6H6.GT, Total bike rentals, Electricity Recommended retail price, Monthly beer production, Quarter-monthly river flow, Lac St-Jean, Mon pax web, Internet traffic data III, OM CPU – see Appendix 7.2

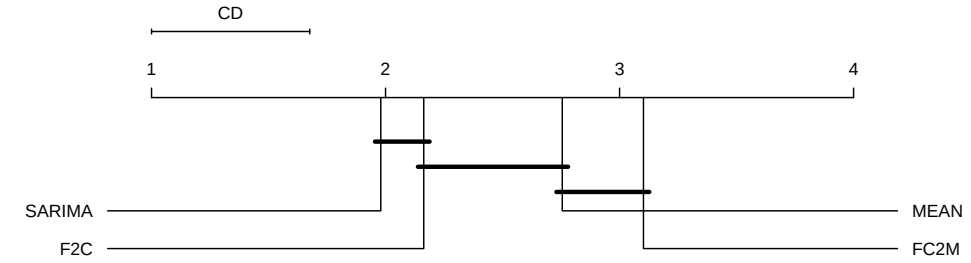


Figure 5.4: Critical diagram of the comparisons of the subset of the four best methods. F2C method configured with K-means clusterer, centroid prototypes, hard paradigm, and naive bayes classifier (like FC2M), and a $\gamma = 1$, in an attempt to make the configurations of F2C and FC2M comparable.

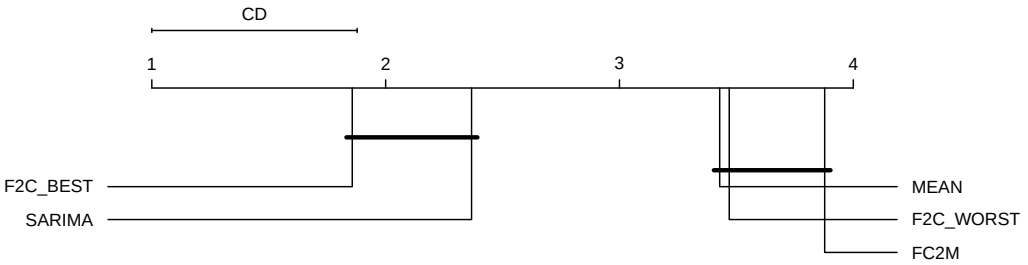


Figure 5.5: Critical diagram of the comparisons of the subset of the four best methods, both for the best combination of parameters found for F2C (F2C_BEST) and for the worst combination of parameters (F2C_WORST), with $\gamma = 1$.

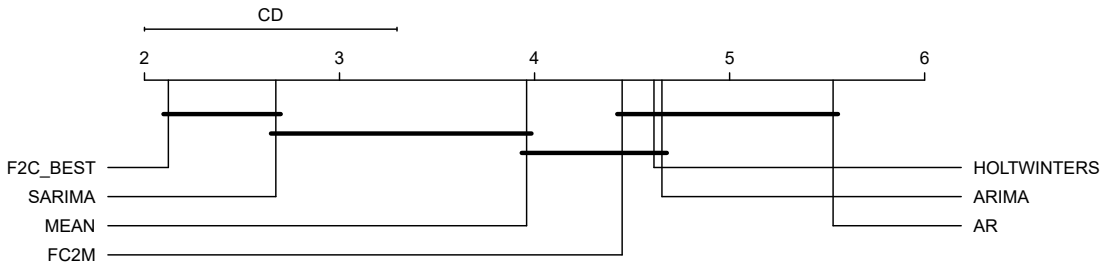


Figure 5.6: Critical diagram of the comparison between different prediction approaches for a value of $\gamma = 1$.

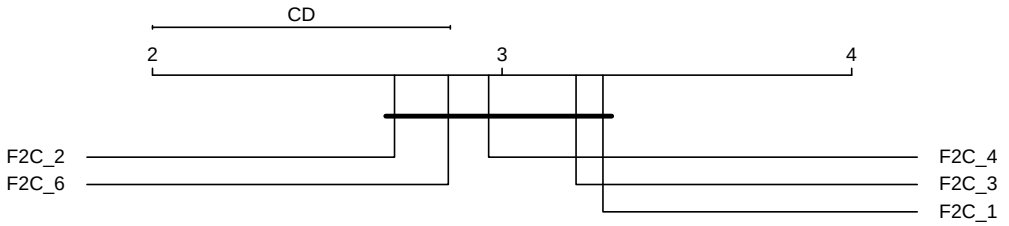


Figure 5.7: Critical diagram of the comparison between different prediction approaches for various values of γ (in $[1, 2, 3, 4, 6]$) only for F2C method.

Chapter 6

Probabilistic approach

This chapter introduces a new version of the framework. Up to now, only deterministic forecasts have been produced with the framework proposed in Chap. 3. However, *probabilistic forecasts would be interesting*, particularly in the capacity planning problematic where uncertainty is very usual. Modelling the probability of having a peak in an incoming season would be useful for anticipation. This is not something which is possible with deterministic algorithms, where only plain values are provided.

To produce probabilistic forecasts, the need to *gather and describe seasons in a probabilistic way* arises. Probabilistic clustering algorithms are mandatory for this purpose. In this chapter, the problem of seasonal time series forecasting is rephrased in a probabilistic manner while using probabilistic time series clustering and probabilistic next type of season classification. The use of MODL coclustering algorithms is chosen for coclustering. This method is able to create probability grids that describe seasons, grids that may be used afterward for producing probabilistic forecasts.

First Sec. 6.1 explains the interest and the stakes of using time series probabilistic forecasting. Sec. 6.2 presents the probabilistic implementation of the framework and its various parts. Sec. 6.3 introduce a portfolio approach to the framework. Sec. 6.4 shows the experiments conducted to assess the performances of this new approach. Comparisons between various parameter combinations are made, and opponents are being challenged. Sec. 6.5 discusses the results.

6.1 Probabilistic seasonal time series forecasting

6.1.1 The stakes of probabilistic time series forecasting

As stated in Sec. 2.2 which showed the differences between deterministic forecasting and probabilistic forecasting, a forecast is probabilistic when it provides probability distributions rather than plain deterministic values. Those probability distributions are useful because they bring additional knowledge to the forecasts, especially in contexts where uncertainty must be dealt with (*e.g.* in electricity production forecasting, in capacity planning, *etc*). The forecasts are richer and can be analysed by specialists and experts of the domain with more room for interpretation.

The goal of seasonal probabilistic forecasting is to estimate

$$P(\hat{y}_{n+1:n+s} \mid y_{1:n}, \Phi)$$

where s is the season length, $\hat{y}_{n+1:n+s}$ are the s next values (next season) of the observed time series $y_{1:n}$ and Φ the model parameters.

Our framework relies on the assumption that there exists typical seasons in time series (see Hyp. $\mathcal{H}_5$

in Sec. 1.3.3). We remind that $\mathcal{S} = \{S^1, \ldots, S^p\}$ denotes the typical seasons in notations introduced in Def. 11. Here, the number of typical seasons is part of the model parameters Φ.

Assuming that prototypes belonging to $S^1, \ldots, S^p$ are conditionally independent events, the law of total probability ensures that:

$$P(\hat{y}_{n+1:n+s} \mid y_{1:n}, \Phi) = \sum_{S \in \mathcal{S}} P(\hat{y}_{n+1:n+s} \mid S).P(S \mid y_{1:n}, \Phi)$$

where $P(\hat{y}_{n+1:n+s} \mid S)$ is the probability of having $\hat{y}_{n+1:n+s}$ known that the season is of type S (and the observed time series) and $P(S \mid y_{1:n}, \Phi)$ is the probability of having season of type S knowing past observations.

This expression relates the seasonal probabilistic forecasting to the general principles of the proposed framework:

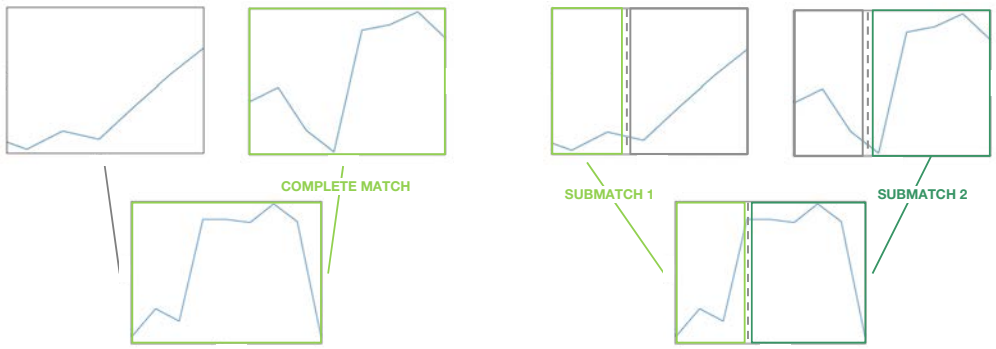
— the first term, $P(\hat{y}_{n+1:n+s} \mid S)$, corresponds to the framework's clustering step. Estimating these probabilities (for each cluster $S \in \mathcal{S}$) is the inference phase of the clustering. Learning a clustering model for seasonal time series consists in estimating the parameters $\Phi_{\mathcal{S}}$ of a clustering model that maximises the likelihood of $P(y_{k+1:k+s} \mid S, \Phi_{\mathcal{S}})$ for $k = 0, \ldots, \frac{m}{s}$. Classical probabilistic clustering approaches are based on Gaussian Mixture Models [Rey09] or adapting distance based clustering to a probabilistic approach [Iyi10]. The next section will introduce a probabilistic model based on time series co-clustering.

— the second term, $P(S \mid y_{1:n}, \Phi)$, corresponds to the classification step of the framework (S is the type of the season coming after n). Estimating $P(S \mid y_{1:n}, \Phi)$ is a classical supervised learning problem that can be addressed by many supervised classifiers such as Bayesian Classifiers [WB98]. But the non-probabilistic classifiers introduced in the previous implementation of the framework can also be used in the probabilistic approach using isotonic calibration [ZE02]. The goal of isotonic calibration is to obtain actual probabilities using the classifiers, a classifier not calibrated being less precise – apart from the bayes classifier which is naturally calibrated.

6.1.2 Coclustering of time series: a probabilistic model

In the previous version of the framework, MODL coclustering was introduced as an alternative for time series clustering. This section goes deeper in the details of this approach that turns out to be a suitable model for probabilistic clustering of time series.

Coclustering is a particular type of unsupervised algorithm which differs from regular clustering approaches by creating co-clusters. The objective of the coclustering approaches consists in simultaneously partitioning the lines and the columns of an input data table. Thus, a co-cluster is defined as a set of examples belonging to both a group of rows and a group of columns. Coclustering identifies clusters based on different local similarities (*i.e.* values for a subinterval of the season). This feature is interesting and new: traditional clustering algorithms gather TS as a whole (see Subfig. 6.1 (a)) and there is not subsplitting of the TS following the time interval (see Subfig. 6.1 (b) with a simple example of two submatches). Hence, coclustering can be used to create groups while matching several subparts of the time series coming from several clusters.



(a) Clustering of a TS. (b) Coclustering of a TS.

Figure 6.1: Difference between clustering (a), which matches the entire TS, with coclustering (b), which is able to match subintervals of the time series of various other time series.

In the literature, coclustering has been applied in the specific case of TS (for example, in [Hea+05] for gene expression analysis, in [WZK15] for geo-referenced TS, in [Kha+12] for cloud workload analysis). The usual input data table of a coclustering approach can be either a contingency table of two categorical variables, or a 'attributes/examples' table. There is a need for a more general setting, by processing more than two variables including numerical variables, in order to make a partition of TS. Indeed, a coclustering approach able to handle three variables is needed in order to make: i) groups of identifiers of times series (the categorical variable C); ii) intervals of timestamp values (the variable T encoded as numerical); iii) intervals of the measurements values of the times series (the variable V also numerical). A co-cluster gathers TS (group of identifiers) that have distinct values during a certain interval of time. Contrary to the clustering approach that is based on the entire TS, the coclustering approach uses a local criterion.

In the suggested TS forecasting approach, a tri-clustering (*i.e.* that extends the coclustering problem to three variables) is needed to deal with both numerical and categorical variables. Among the approaches available in the literature, the choice was made on the MODL framework [Bou12] which matches this need.

The MODL framework makes a constant piecewise assumption (*i.e.* a 3D histogram) to estimate the joint distribution $P(C,T,V)$ by jointly discretising the variables T, V and grouping the TS identifiers of the variable C. The resulting model consists of the Cartesian product [1] of the three partitions of the variables C, T, V. This model can be represented as a 3D grid (see Fig. 6.2, on the left). In this 3D grid, if one considers a given group of TS (*i.e.*, a given group of C), the model provides a bivariate discretisation which estimates $P(T,V \mid C) = \frac{P(C,T,V)}{P(C)}$ as a 2D grid (see Fig. 6.2, on the right). This 2D grid gives the probability to have a given range of values during a given interval of time. Therefore knowing that a TS belongs to a given cluster the corresponding 2D grid may then be used for crafting forecasts (see next section).

In the MODL approach, finding the best tri-clustering model is turning into a model selection problem. To do so, a Bayesian approach called Maximum A Posteriori (MAP) is used to select the most probable model given the data. Details about how this 3D grid model is learned may be found in [Bou12; BBC15].

1. The Cartesian product of the three partitions is used as a constant piecewise estimator – *i.e.*, a 3D histogram.

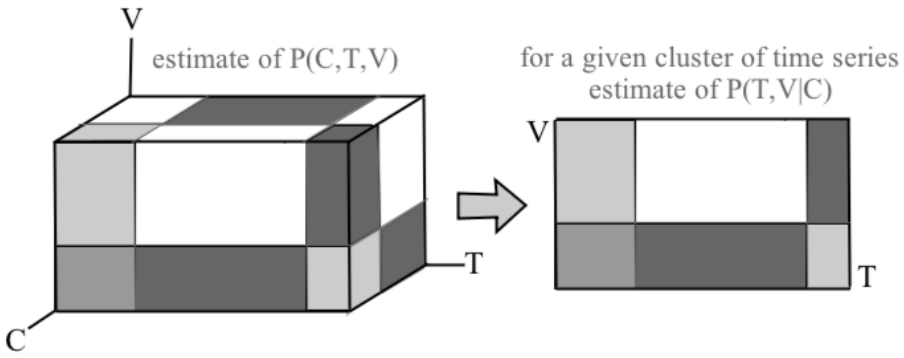


Figure 6.2: Illustration of a trivariate coclustering model where a slice referred to 'grid' is extracted. It it this grid that is used for producing probabilistic forecasts.

The main idea could be summarised as finding the grid which maximises the contrast compared to a grid based on the assumption that T,V and C are independent (*i.e.* $P_{grid_{3D}} = P(V,T,C)$ compared to $P_{Indepedence} = P(V)P(T)P(C)$). Therefore the estimation of this MAP model outputs: (i) ν intervals of values $V_i = [v_i^l, v_i^u]$ for $i = 1,\ldots,\nu$, (ii) τ intervals of times $T_i = [t_i^l, t_i^u]$ for $i = 1,\ldots,\tau$, (iii) groups of TS. These groups of time series corresponds to the typical seasons, denoted $\mathcal{S}$ in the above model. $|\mathcal{S}|$ is the number of clusters at the finer level that is optimal in the sense of the MODL framework.

In the TS forecasting approach, the right number of (tri-)clusters are chosen regarding to the forecasting task. This value could differ from $|\mathcal{S}|$. Therefore the MODL coclustering approach allows applying a hierarchical clustering to the finer level to have a coarse level with a lower number of clusters called C^*, $C^* < |\mathcal{S}|$. Then C^* becomes the retained value considering the global aim. This procedure is similar to the one described for clustering algorithms: different number of clusters are evaluated and the overall model that leads to the lowest forecast error on the validation set is kept.

Let us now come back to the formalisation of probabilistic time series forecasting. $P(\hat{y}_{n+1:n+s} \mid S)$ can be estimated by the MODL model from the conditional probabilities $P(V,T \mid C = S)$ where S denotes one of the time series groups.

In practice, this probability is not interesting to forecast time series. Indeed, an estimate of $P(\hat{y}_{n+i}|S)$ is made for each $i = 1,\ldots,s$. As the grid boundaries are the same for each cluster, this means that for each time instant $i = 1,\ldots,s$, the probabilistic forecast is a weighted distribution

$$P(\hat{y}_{n+i} \mid y_{1:n}, T, \Phi) = \sum_{S \in \mathcal{S}} P(\hat{y}_{n+i} \mid S,T).P(S \mid y_{1:n}, \Phi)$$

where $P(\hat{y}_{n+i} \mid S,T)$ is a piecewise constant distribution given by a slice of the MODL grid. Then, the forecast itself can be represented as a 2D grid with the boundaries of the MODL model.

6.2 The PF2C Framework

The framework implementation presented in this chapter is called 'Probabilistic Forecasting with Clustering and Classifier' (PF2C). Chap 3 gave an overview of the framework architecture with the Fig. 3.4 and 3.5 (respectively presenting the learning phase and the forecasting phase). In this chapter, there is

only one clustering algorithms used: MODL coclustering. The classifiers are similar to the ones presented in Sec. 5.1.1.

6.2.1 Learning process

In this implementation of the framework, MODL coclustering is paramount, since it will be used to generate probabilistic forecasts. Clustering algorithms used with previous implementations FC2M and F2C (*e.g.* K-means, K-shape and GAK) are not useable for that purpose.

Classifiers are also important in this learning process. The classifiers used in the framework (*e.g.* naive-bayes classifier, decision trees, random forests and logistic regression) have already been presented in Sec. 5.1.1.

6.2.2 Forecasting process

Cluster prototypes

As seen in Sec. 3.3.1, the framework first gathers similar seasons into groups by a coclustering algorithm at learning time. One prototype per cluster (that represents a typical season) is needed to generate the forecasts at forecasting time.

With coclustering, *grids* as shown in Fig. 6.2 are used as cluster prototype. A grid represents the joint probability of the values V at time T for a given cluster C ($P(T,V|C)$). The darker the cell is, the more probable it is that the activity lies in this specific cell in the data.

Forecasting paradigms

To forecast the next season (season $x+1$), the framework makes use of the p prototypes of the p found clusters and the probabilistic output of the classifier $\hat{\pi}^{x+1} = \{\hat{\pi}_1, \ldots, \hat{\pi}_p\}$.

Two paradigms are studied in this chapter: hard method and soft method, see Sec. 3.3.2. Note that for F2C, the prediction is a TS, while for PF2C, the prediction is a grid.

6.3 Portfolio framework instance

As always in this thesis, the framework may be instantiated in various ways, depending on which clustering or classification algorithm is used.

To select the best combination of clusterer and classifier for a given dataset, a portfolio approach seems appropriate. This approach selects the best combination of both cluterer and classifier using the validation set, while optimising the number of clusters at the same time. It aims at proposing a complete data driven approach, the framework being automatically adapted for any time series. More especially, the difference between the regular mode and the portfolio mode can be described as follows:

— In the *regular mode*, all the datasets are used to find the best hyper-parameters. More specifically, all the train and validation ensembles are used for finding the best overall clusterer and classifier combination, and this choice is applied on every test ensembles to gather results. When hyper-parameters are found, train and validation ensembles are merged together to make the training set

bigger at testing time, and thus having more reliable results. The best parameters are found using all the datasets at hand, and not dataset per dataset. This solution thus requires less computing.

— In the *portfolio mode*, the best clusterer and classifier combination is selected per dataset. Each elected combination might then be different for each different dataset. Therefore, this solution is more fine-grained and specialised to the data and the need.

Performances of the portfolio approach will be studied later in this chapter in Sec. 6.4.5 below.

6.4 Experiments

In this section, experiments to assess the performance of the proposed framework are proposed: F2C and PF2C methods are compared, as well as other opponents introduced in Sec. 4.2.1.

Sec. 6.4.1 first gives the reader an overview of the setup used for the experiments: datasets, data split, opponents, *etc.* Sec. 6.4.2 shows the interest of using a grid for forecasting and validates some assumption, using generated time series. It also shows an example of probabilistic grid forecasts. Sec. 6.4.3 explicits the best parameters found for the PF2C implementation. Those parameters are used in next Sec. 6.4.4, which presents results on extensive experiments and comparisons with state-of-the-art algorithms are made. Sec. 6.4.5 finally explicits the performances of the portfolio methodology depicted in Sec. 6.3.

6.4.1 Protocol

In this section are introduced the experimental setup used in the experiments. It is first reminded that the protocol followed for all experimental parts of this thesis is presented in Sec. 3.4. The datasets are listed in the Appendix 7.2.

Then, it is recalled that opponents have been presented in Sec. 4.2.1. PF2C being a probabilistic methodology, a new opponent following this paradigm is introduced: **Prophet [TL18]**. Authors use a decomposable time series model that uses the holiday period (exogenous data), trend and seasons for providing seasonal interval forecasts. Similar to a generalised additive model (GAM [Has17]), with time as a repressor, Prophet fits several linear and non-linear functions of time as components. A model is learned to use the past data (up to the last season before the one to predict), and this model is used to predict the next season. No exogenous data has been used for training Prophet. Comparisons will also be made with the F2C method, presented on Chap. 5, which uses K-means clustering algorithm and random forest classifiers to learn the structure in the season sequence.

The entire code of the framework was developed in Python 3.5. For developing the TS clustering, the library *tslearn* [Tav+20] was used. For developing the classification, the library *sklearn* [Ped+11] was used. For the developing the coclustering algorithm, Khiops [Bou16] was used.

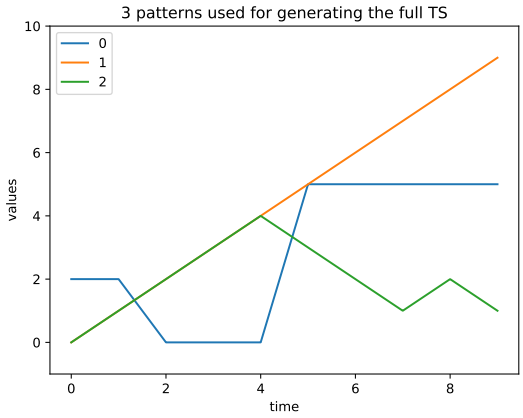
6.4.2 Experiments with synthetic datasets

This section shows preliminary probabilistic results with generated data. The goal is to introduce the probabilistic grid used in PF2C method, and more especially to give intuitions behind probabilistic forecasting and check hypothesis.

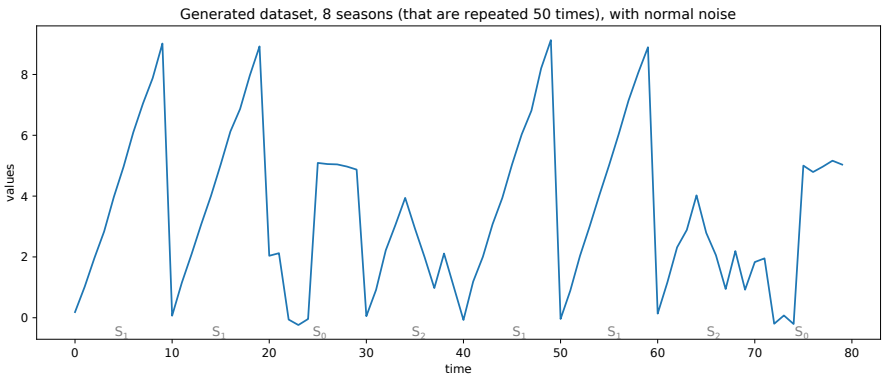
The data generated

Generating data is a good strategy for checking assumptions before launching experiments at scale. Indeed, the shape of the generated data is often simpler, and completely controlled. Experiments may be executed with various parameters, to plot understandable results and to validate basic expectations.

The seasonal data generated for this section follows some well-established seasonal sequences. Three different TS patterns are defined for three different seasons of length 10. In the Fig. 6.3, one season (s_1 in orange) with always increasing values is observed, one season (s_2 in green) with two peaks is observed, *etc.* Those three different seasons are then repeated 50 times in a defined and ad hoc order (*e.g.* $s_1, s_1, s_0, s_2, s_1, s_1, s_2, s_0$, as observed in Subfig. 6.3 (b) which shows the entire sequence that is being repeated), and a noise is added to the final TS to make the forecasting process less straightforward.



(a) Three patterns used for generating the TS – without noise.



(b) Generated TS – with normal noise.

Figure 6.3: The sequences of length 10 used for generating the TS are given in Subfig. (a). Generated TS (b).

Grid probabilistic forecasts

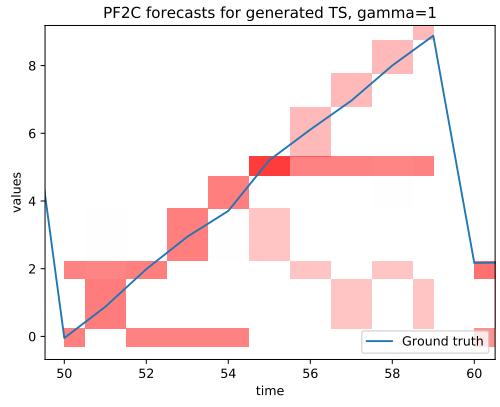
In the Fig. 6.4, two examples of one season ahead probabilistic grid forecasts are shown. In these examples, the real values of the predicted TS are shown in blue. Several cells are shown in a red overlay. The darker the red, the more probable next season ahead lay in this (T,V) interval. The red overlay represents the probabilistic grid created by MODL coclustering.

Subfig. 6.4 (a) illustrates a probabilistic forecast with a lot of uncertainty. Indeed, light red cells are observed in the figure where the data was predicted to lay (with a low probability), and there was no very dark red cells. In this case, the classifier was unable to predict the next season cluster type accurately and therefore mixed the three grid prototypes (33% of grid 0, 33% of grid 1 and 33% of grid 2 in this case). Note that all grids generated (0, 1 and 2) share the same squaring because they were generated with MODL which necessarily create the same cutting for all dimensions (C, V, T).

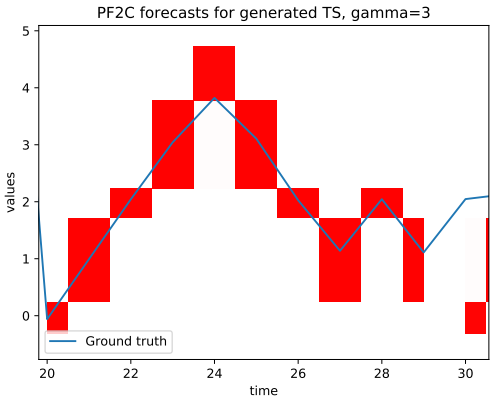
Subfig. 6.4 (b) illustrates a good probabilistic forecast. It is observed that the real values (in blue) often appear in the red boxes where the red is very dark. Furthermore, there is no light red cells shown. It means that season type was both well described by MODL and well predicted by the classifier.

Learning with various windows size

It is interesting to compare the performances of the two aforementioned forecasts in Fig. 6.4 with the value of γ (which is the number of historical seasons used for learning the classifier). In the Subfig. 6.4 (b), $\gamma = 3$ and the sequence prior to the one displayed is $[s_1, s_1, s_0]$. In the given sequence (*e.g.* $s_1, s_1, s_0, s_2, s_1, s_1, s_2, s_0$ repeated 50 times), it is easy to see that it is impossible to be wrong because the only possible shape after $[s_1, s_1, s_0]$ or $[s_2, s_1, s_1]$ is s_2 and nothing else. But in Subfig. 6.4 (a) and to predict s_1 with a γ of 1 (with the sequence $[s_0, s_2, s_1]$ prior to the data displayed), it is hard to tell because both s_0, s_1 and s_2 can all be followed by the season type s_1. After a s_1, it is 50% s_1, 25% s_0, 25% s_2 of chances to be in each of those grids.



(a) Uncertain forecast with many light red cells.



(b) Good forecast with only dark red cells.

Figure 6.4: Two examples of one season ahead grid forecasts for the generated TS. (a) s_1 predicted with some errors, (b) s_2 predicted with a high certainty.

Quality of the forecast

The grid forecasts shown in Fig. 6.4 describe the season ahead for the generated datasets. This approach is valuable because creating grids is a good way to learn the seasonal behaviours encountered in the TS in a data-driven way. Good predictions of next season grid with the classifier hence lead to a good description of the incoming points distributions. It is then an interesting probabilistic alternative to the deterministic one presented in the last two chapters.

As explained in Sec. 2.4, it is not possible to compute the MSE nor the MAE, because the predicted grids cannot be seen as deterministic values. Therefore, the CRPS metric presented in Sec. 3.5 enters into play. Technically, the steps followed to compute the CRPS are as described in the Algo. 1. The goal is to compute the integral of one function F composed of combinations of the Cumulative Distribution Function (CDF) of both the grid and the known values of the TS, for each time step T. The area below the curve F represents the CRPS. An example is seen in Fig. 6.5.

Data: grid forecast and observed values;
Result: how to compute the CRPS with probabilistic grids;
for *each time step* T_i *for* $i \in 1..h$ **do**
  CDF_g=compute the CDF of predicted grid at T_i;
  CDF_r=compute the CDF of known value at T_i;
  $F = (CDF_g - CDF_r)^2$;
  $CRPS_i = \int_{-\inf}^{+\inf} F$;
end
Output: $CRPS = mean(CRPS_1, ..., CRPS_h)$;

Algorithm 1: How to compute the CRPS for probabilistic grid forecasts.

The two steps that consist in computing the CDF are shown below:

— To compute the CDF of the grid CDF_g, it is a matter of using the dimension T in the grid created in order to extract 'slices of grid'. Those represent, for one interval T_i with $i \in \mathbb{N}$, probability distribution for the forecast to appear in value intervals $[V_{low}, V_{high}]$ for all the value intervals identified by MODL. Transforming this information in CDF is trivial.

— To compute the CDF of the real observed value CDF_r, one simply states that the probability to have a value below the given value is equal to 100%, thus creating a step-shaped CDF.

In conclusion, computing the CRPS using the grid is upright, because it is natural to translate the grid itself into a CDF. Also, it is interesting to note that as stated in Sec. 3.5, the CRPS metric is analogous to MAE for deterministic forecasts. Therefore, comparing MAE measure for deterministic forecasts against CRPS values for probabilistic forecasts is technically sound.

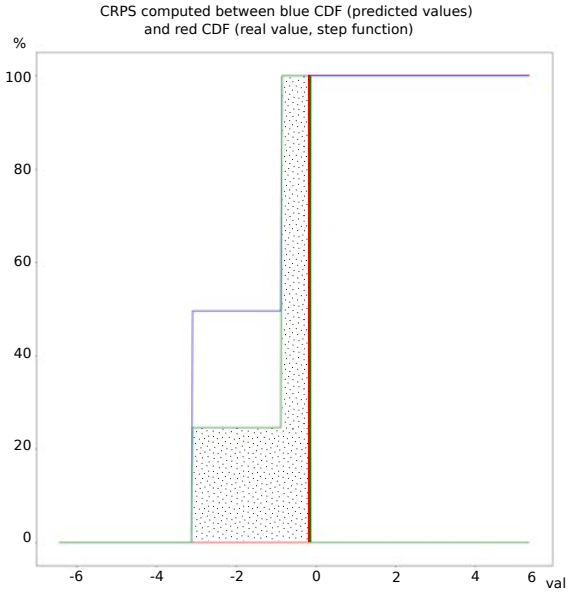


Figure 6.5: Example of CRPS computation, dotted area below the green curve is the value of the CRPS computed with the CDF of the prediction (blue) and the CDF of the real value (red). Reading the blue CDF gives the information that there were 50% of chances to observe values superior to approx. -3 and 100% of the data superior to -0.8, in the observed data. Reading the red CDF gives the information that the observed value was 0.

6.4.3 Best parameters for PF2C method

In this section, an analysis of the alternative settings of the PF2C methodology is conducted. The goal is to find out which is the best way of mixing probabilistic grids (soft vs hard mixing paradigms) and also which is the best classifier for the PF2C method – overall for all the dataset used. Obviously, there is no need to find out which is the best clusterer because MODL is the only clustering algorithm able to produce probabilistic grids for the PF2C methodology. The learning step of the framework is made with $\gamma = 1$.

Fig. 6.6 (a) shows that the best way of mixing grids for producing probabilistic forecasts is the soft methodology. The classifier is thus chosen with soft decision for the remaining of this section.

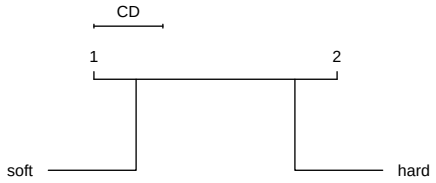
Fig. 6.6 (b) shows that decision trees are in first position, but that the difference between all the other classifiers used is not statistically significant. This last point is enforced by Fig. 6.6 (c) (d) (e), which shows that decision trees are not systematically better than the other classifiers (pairwise CRPS performances) and that results are really tight.

In conclusion, this section shows that the best parameters combination found for the PF2C method are the following:

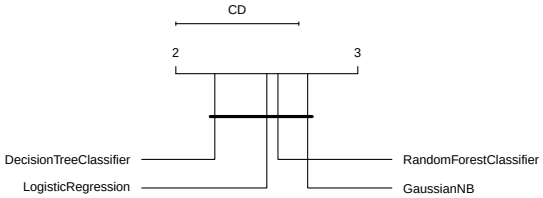
— MODL coclusters (the only clustering algorithm able to create probabilistic grids),

— Soft methodology with grid prototype,

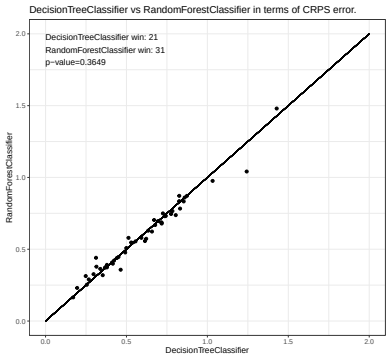
— Decision-trees classifier.



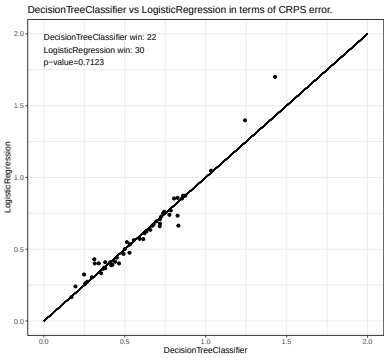
(a) Hard methodology VS soft methodology, performances for the PF2C method.



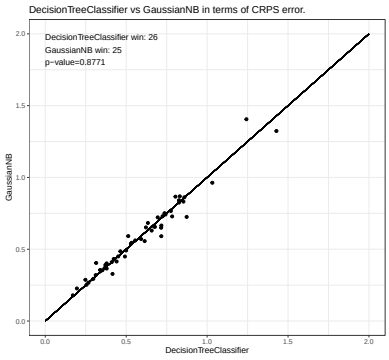
(b) Performance of the various classifiers for the PF2C method.



(c) Decision trees vs random forests



(d) Decision trees vs logistic regression



(e) Decision trees vs naive-bayes

Figure 6.6: Performances of (a) hard VS soft methodology, (b) various classifiers. (c) (d) (e) Win/lose graphs that show in first row the number of times decision trees won against the three other classifiers: logistic regression, random forests and naive-bayes. Comparisons using CRPS. Subfig. (b) (c) (d) (e) generated with soft results.

6.4.4 PF2C vs opponents

Fig. 6.7 is a critical diagram that assesses the performances of the method using the test ensemble. [2] It shows that rank-wise, PF2C is performing well and is in third position. It is worth noticing that the rank difference with F2C is not statistically significant. This is confirmed by the Fig. 6.8 which shows the performance of the method against three selected opponents: ARIMA, Prophet and F2C. One could also notice the bad performances of Propjet which is ranking 7th out of 9 methods compared. This can be explained because Prophet heavily relies on exogenous data for producing forecasts, and no exogenous data was used at all during experiments, in order to have comparable results.

2. The method has been properly configured with the best parameters found in Sec. 6.4.3 using the validation ensemble.

6.4.5 Portfolio approach

Fig. 6.9 compares the performances of the regular and portfolio methodologies. The portfolio has been introduced in Sec. 6.3.

In Subfig. 6.9 (a), it is worth noticing that $\text{PF2C}_{\gamma=1}$ outperforms $\text{PF2C}_{\gamma=3}$. While using grids and probabilistic forecasts, it therefore seems more efficient to learn the classifier with only one season in the historic – in opposition to the previous conclusion drew in Sec. 6.4.2 with generated datasets. The second thing to observe is that the two portfolios PORTFOLIO $\text{PF2C}_{\gamma}, \gamma \in [1,3]$ give better results than their respective counterparts PF2C_{γ}. It is encouraging and makes the portfolio approach meaningful.

In Subfig. 6.9 (b), the PF2C methodology has been configured with $\gamma = 1$, both for the regular and the portfolio method (according to best performances observed in Subfig. 6.9 (a)). The F2C methodology has been configured with $\gamma = 2$ (according to Fig. 5.7 page 83). The first thing to see is that the critical difference shows a significant difference between two groups of methods: the methods introduced in this thesis (F2C and PF2C) along with SARIMA, and the other methods (mean baseline, Holt Winters, Prophet, FC2M baseline, ARIMA and AR). Also, as observed in Subfig. (a), the results of the portfolio PF2C are better than the regular methodology, and they are very close to SARIMA method rank-wise (non-significantly). Finally, even if the portfolio has better results, it is observed that the regular F2C_2 is still the first method ranking-wise. One explanation is that the CRPS measure is not suitable for evaluating grid prediction. The grid model is not a probabilistic model centred around the mean. Non-zero cells represent alternative paths, see grid on Subfig. 6.4 (a). If it first seems intuitively interesting to have these paths, it penalises the CRPS in this particular case.

6.5 Wrap up

In this chapter, a probabilistic instantiation of the framework introduced in this thesis is proposed. Experiments with synthetic datasets have first been conducted. The goal was to check assumptions, as well as presenting the new probabilistic forecasts shaped as a grid. It showed that a probabilistic grid predicted by the classifier really stick to the real data in Fig. 6.4. It shows *the relevance of the grid created by MODL* but also *the precision of the classifier* which was able to predict with a 100% certainty the type of next season.

The real performances of the framework have been compared with deterministic opponents already used in previous chapters, as well as with a probabilistic one (Prophet). Results show that the method is performing well, especially while using the portfolio methodology introduced in Sec. 6.3 (portfolio method in third rank against all opponents).

This chapter finally introduced a portfolio methodology, where the aforementioned framework is algorithmically fine-tuned dataset per dataset, producing a specific parameters combination well fitted for each precise need and each dataset. The portfolio methodology has been studied for PF2C and accuracy improvements are noticed. The conclusion is that the portfolio is beneficial for creating better forecasts, as the parameters are more adapted for each particular datasets.

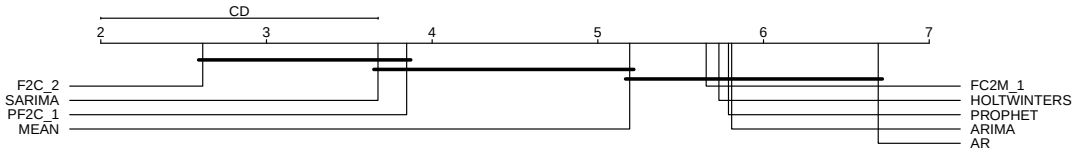
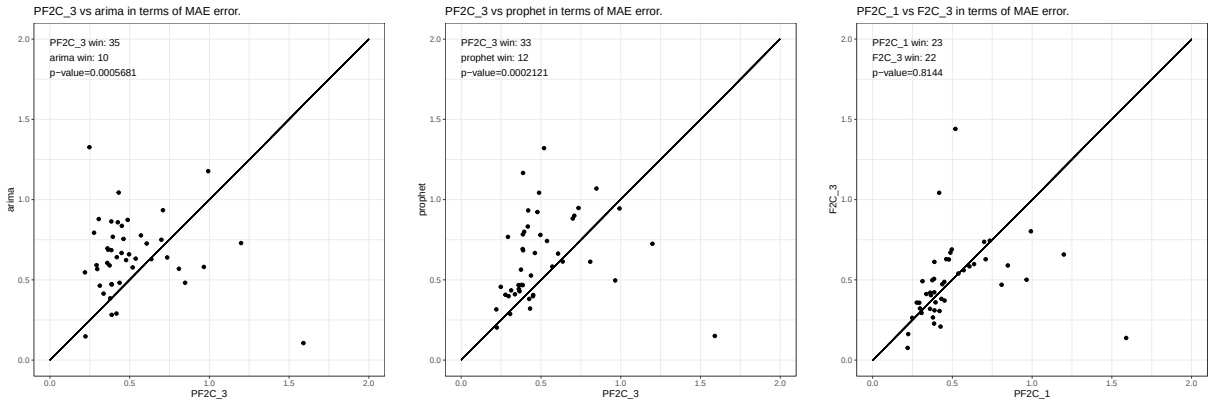
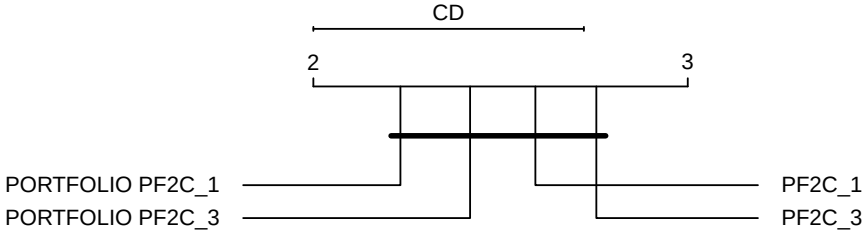


Figure 6.7: Critical diagram of the comparison between different prediction approaches for different values of γ (1 and 2, *e.g.* PF2C_γ for example).

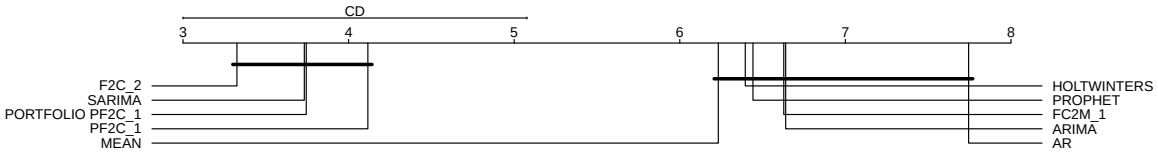


(a) PF2C vs ARIMA (b) PF2C vs PROPHET (c) PF2C vs F2C

Figure 6.8: Win/lose graphs that show the number of times PF2C won against three selected opponents ordered by decreasing performances: ARIMA, PROPHET and F2C.



(a) Critical diagram of the comparison between regular and portfolio approach approaches.



(b) Critical diagram of the comparison between all methods including PF2C portfolio approach.

Figure 6.9: (a) CD that shows a rank analysis of the portfolio methodology performances. Use of various values of γ (*e.g.* PF2C_γ for example). (b) CD that shows ranks of all methodology including PF2C portfolio approach.

Chapter 7

Orange Money Transactions per Quarter Use Case

Chap. 3 introduced a framework for seasonal time series forecasting, and three different implementations of it were proposed on Chap. 4, 5 and 6. The performances of all the implementations have been compared with the use of several state-of-the-art opponents and with various datasets from different sources: nature information such as river flows, monthly beer production in Australia for several years, or technical time series from Orange Money project in the context of capacity planning, *etc.* Appendix 7.2 presented the datasets, that were numerous and did not only concern the capacity planning problems, because the need was to develop a data-driven approach that would fit any type of incoming data (as stated in Hyp. $\mathcal{H}_1$ in Sec. 1.3.3).

However, this thesis is dedicated to the improvement of the capacity planning toolset at Orange. This last chapter studies the interest of the framework in this particular context. The aim is to confirm the relevance of the approach with a study on capacity planning datasets only.

First, some notable results of the framework presented in Chap. 5 for deterministic forecasts and Chap. 6 for probabilistic forecasts are shown in Sec. 7.1. More particularly, the dataset used for this analysis are presented in Sec. 7.1.1. The results of the clustering algorithms are then studied in Sec. 7.1.2, and more specifically the clusters compositions, both for K-means clustering algorithms and for MODL coclustering. Some deterministic forecasts of F2C (see Chap. 5) are compared in Sec. 7.1.3 with some probabilistic grids produced by PF2C (see Chap. 6).

7.1 Case study on Orange Money dataset

The case study is the forecast of server usage that is the backbone of a mobile application. The application is a mobile banking solution proposed to the African continent's customers. They carry a performance problematic, that is to ensure that their infrastructure is well sized and configured. Every day, the infrastructure has to be adapted to fit the expected demand. The possibility to support decision-making by forecasting the user activities a day ahead is inspected. The user activity is monitored via the number of transactions per quarter on the servers' infrastructure.

7.1.1 Orange Money TPQ dataset

As stated in Sec. 1.1.2, the stakes of OM project to maintain a reliable infrastructure follows three axes. First, the servers must be well dimensioned – the right number of servers should be up at the right time for serving all requests without service interruption. Also, the servers must be well configured –

flawless connection to the networks, software up to date, etc. Finally, the data flows involved during the functioning of the servers must be controlled.

To ensure the good performances of the services, load tests and limit tests (introduced in Sec. 1.1.2) were conducted on the entire infrastructure. The major result of those is the maximum number of transactions per quarter Max_{TPQ} that the infrastructure is able to handle. Once Max_{TPQ} is reached, the infrastructure is no longer in control and can crash at any moment.

The seasonal forecasting algorithms presented in this thesis enter into play: they allow forecasting the next day profile, and therefore to check if Max_{TPQ} is reached in a close future. If it is, several decisions could be taken: scale up the infrastructure by adding new servers, for example (difficult to do in short notice especially for a bare metal infrastructure). Reconfigurations of the infrastructure could also be done more rapidly.

This experimental section is focused on the results of the frameworks for one functional dataset, that represents the number of people browsing the service: the transactions per quarter (TPQ). It is then a matter of comparing the TPQ to Max_{TPQ} and check if $TPQ < Max_{TPQ}$ stays correct.

The Fig. 7.1 illustrates 15 days of data (1440 points) for one server from day number 180 to day number 194. The values of the TPQ are z-normalised. This figure shows that the time series is obviously seasonal and it has been confirmed by the Fisher's g tests [WFS04] (see Sec. 2.3.2). Nonetheless, there are at least two different types of days, that may be discriminated by their values in the middle of the season. Days 180, 187 and 194 have low values around noon, while the others have high values around noon. Day 186 is in between.

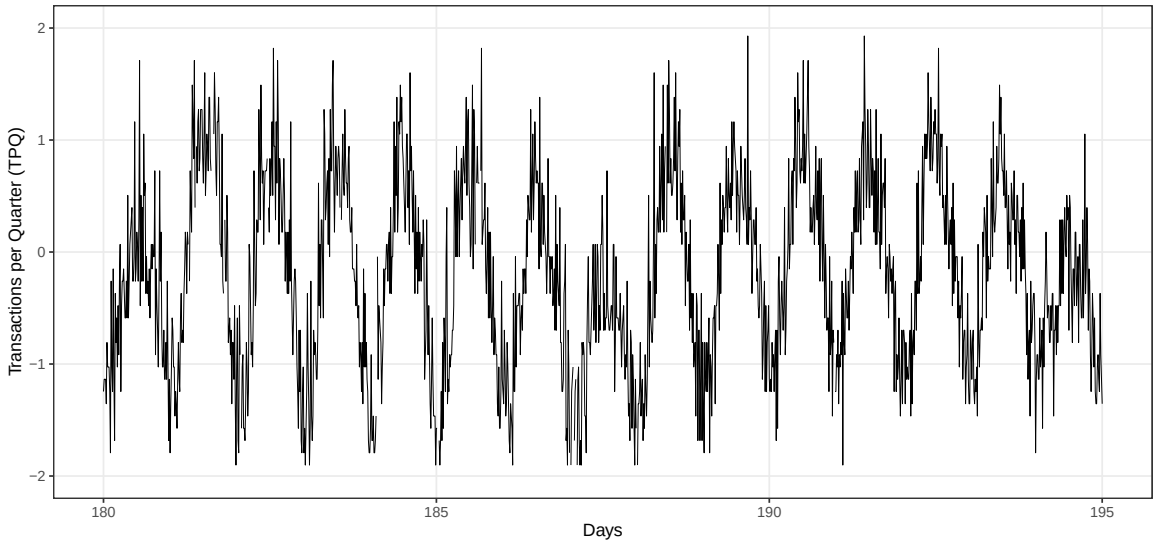


Figure 7.1: Transaction per Quarter (TPQ) dataset for 15 days. One measure per quarter of an hour. The first day is a Monday.

7.1.2 Relevance of using seasons for forecasting

In this section, the results of the clustering algorithms (K-means and MODL coclustering) are shown. It is reminded that in the framework, the number of clusters is chosen by using the result of the forecasting for the whole chain (clustering and the classification steps). In this study, the random forest classifier was used, which lead to four clusters for K-means and seven cocluster for MODL coclustering during the learning process. Note that these numbers of clusters may be different with another classifier.

K-means clustering on TPQ data

The Fig. 7.2 in page 103 shows the clusters found using the K-means clustering algorithms. Table 7.1 gives a precise view of the clusters composition in terms of days of the week. The cluster c_1^k, which was the lowest activity cluster, is composed mainly of Sundays, which could explain the low activity detected. Most days in clusters c_4^k are the end of the week days (Thursdays and Fridays), which could explain the high activity peaks, probably because people transfer money between each other for enjoying the week end.

Cluster	Mon	Tue	Wed	Thu	Fri	Sat	Sun
c_1^k	2	2	0	1	1	7	44
c_2^k	18	18	19	18	16	18	3
c_3^k	17	18	18	17	16	18	7
c_4^k	18	16	18	19	22	12	1

Table 7.1: Number of seasons per weekdays and per K-means cluster.

Finally, the confusions between the weekdays in clusters c_2^k to c_4^k show that knowing only the weekday is not sufficient to accurately predict the right type of day (*i.e.* the cluster). Thus, it justifies the use of a classifier to predict the type of day based on previous observations.

MODL coclustering on TPQ data

Fig. 7.3 in page 103 shows three clusters out of the seven that have been generated by the MODL coclustering algorithm with the TPQ at OM datasets. Table 7.2 gives a precise view of the clusters composition in terms of days of the week.

The three clusters are different, and have been ordered according to the increasing level of activity observed. Data about types of days in each cluster might be found in Fig. 7.3. First, it is observed that the first cluster c_1^m is really smoother than the others. It is most likely because 38 days out of the 42 in this cluster are Sundays. In terms of statistics over the 54 weeks present in the data, more than 70% of the Sundays exhibited the same behaviour and were placed in this cluster.

In clusters c_2^m and c_3^m, there are several different locations for high activity peaks. In cluster c_2^m, there is a nearly even distribution of all types of weekdays including weekend but almost no Mondays. In cluster c_3^m, there is only 14% of the days that are weekend days and the rest evenly distributed along working days. Fig. 7.4 confirms that there is a high activity peak around noon in cluster c_3^m.

Cluster	Mon	Tue	Wed	Thu	Fri	Sat	Sun
c_1^m	1	0	0	1	0	2	38
c_2^m	4	10	8	10	7	14	10
c_3^m	13	14	16	17	14	11	1
c_4^m	17	9	10	9	14	7	3
c_5^m	14	9	10	3	10	6	0
c_6^m	6	12	11	15	10	15	3
c_7^m	23	21	21	24	24	22	6

Table 7.2: Number of weekdays per MODL cluster.

Fig. 7.4 exhibits the MODL grids produced for cluster c_1^m and cluster c_3^m grids. It is interesting to compare their difference because they both represent two different season topology. When the cluster c_1^m represents mostly Sundays, cluster c_3^m represents working days. Figure 7.4 (a) shows that the transactions per quarter are really not high around noon in this particular grid. Figure 7.4 (b) shows that the transactions per quarter have exhibited a lot of high value in red at around noon, sign of a high activity at this hour in this given scenario. On another hand, it is shown that there is a low activity on the servers around 3:00 AM.

7.1.3 Forecast study: examples of deterministic vs probabilistic forecasts

In this section, several forecasts produced by the framework are shown, both for deterministic (F2C) and probabilistic (PF2C) paradigms.

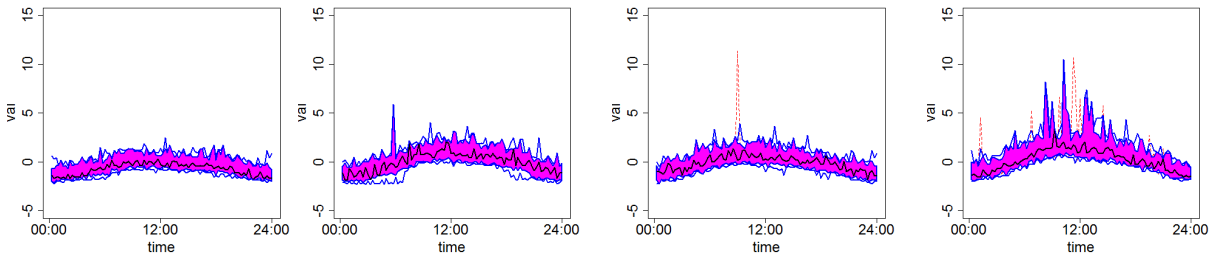
In Fig. 7.5 (a), a deterministic forecast produced with K-means algorithm in conjunction with random forest classifier is shown. In orange, the curve for the forecasted hard approach are depicted, in green for the soft approach, and in blue the real values. It is instantly noticeable that the green curve is smoother than the orange one. An easy explanation would be that there is only one curve represented in orange and several weighted curves in green. In this case, the soft method was closer to real data.

Finally, the Fig. 7.5 (b) shows how the grids (see Fig. 7.4) are used for producing one forecast. It is interesting to see that the real data follows the red path inside the computed MODL grid, sign that the curves followed an expected scenario. The figure is produced under a soft methodology, meaning that the seven grids for cluster c_1^m and c_7^m have been weighted to produce the final grid: the one used for the forecast.

While comparing the two plots in Subfig. 7.5 (a) and (b), the MODL probabilistic grid is more precise than the deterministic forecast that are further away from real values. Indeed, the blue curve that represents the real data really follows the red path described by the probabilistic grid.

7.2 Wrap up

In this chapter, a direct application of the framework elaborated in this thesis was shown on a capacity planning dataset provided by Orange Money project. More especially, the dataset was functional and



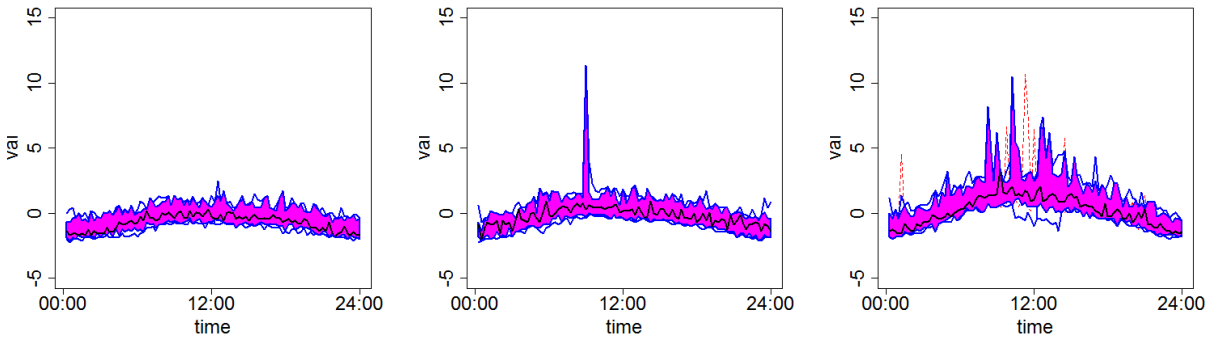
(a) Cluster c_1^k with 57 time series

(b) Cluster c_2^k with 110 time series

(c) Cluster c_3^k with 111 time series

(d) Cluster c_4^k with 106 time series

Figure 7.2: Functional boxplot [SG11] for the seasons detected in Orange Money TPQ dataset with K-means clustering and random forest classifier. In mauve: the envelope of the 50% central region; In black: the median curve; In blue: the maximum non-outlying envelope; red dashed time series are outliers.

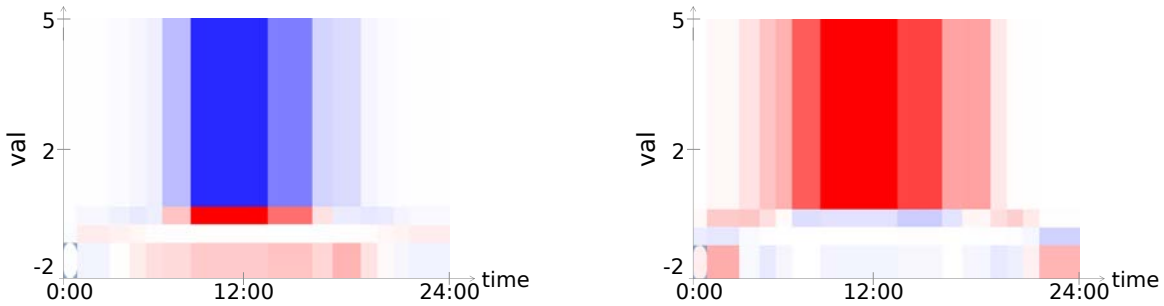


(a) Cluster c_1^m with 42 time series

(b) Cluster c_2^m with 63 time series

(c) Cluster c_3^m with 86 time series

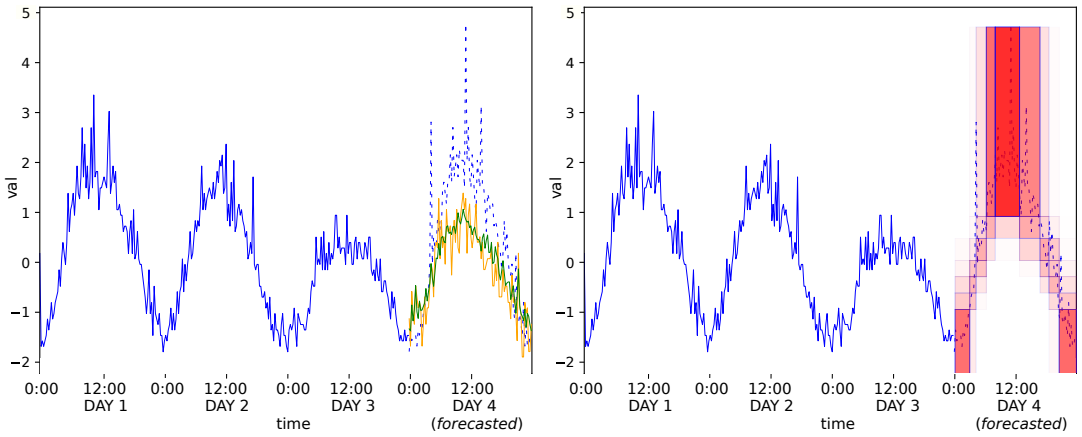
Figure 7.3: Functional boxplot for the seasons detected in Orange Money TPQ dataset with MODL clustering and random forest classifier.



(a) Cluster c_1^m grid with 42 time series

(b) Cluster c_3^m grid with 86 time series

Figure 7.4: Two examples of the MODL probabilistic grid for cluster c_1^m and c_3^m for data TPQ at Orange Money. The redder the cell, the more likely the activity lay in this cell in the data; the bluer the cell, the less likely the data lay in this cell.



(a) Deterministic forecast produced with K-means algorithm in conjunction with random forest classifier for the TPQ dataset. In dashed blue real values, in orange is shown hard forecasts, in green soft forecasts (see Sec. 6.2.2 for more details).

(b) An example of a grid forecast. In blue only, the three days used for feeding in the classifier, in blue with a red overlay the real values and the associated predicted grid. Most of the blue values lay in the predicted grid in this particular case.

Figure 7.5: Examples of deterministic and probabilistic forecasts generated respectively with F2C and PF2C implementations of the framework presented in Chap. 3.

represented the number of people browsing a service every quarter. Study on the composition of clusters created by two clustering algorithms (*e.g.* K-means and MODL) were displayed, and results showed that the hypothesis $\mathcal{H}_4$ introduced in Sec. 1.3.3 (which stated that different types of seasons were present in capacity planning datasets) seems respected in this case. Two forecasts created by both F2C and PF2C methodology were shown and the theoretical interest of the grid forecasts demonstrated with this real use case.

CONCLUSION

This thesis studied seasonal time series forecasting, and was motivated by an industrial capacity planning problem. A framework for seasonal time series prediction is proposed. It involves three steps: the discovering of typical seasons, the prediction of the type of the next season, and finally the prediction of next season itself. The initial motivation for introducing such a framework stems from the infrastructure management context. Indeed, data collected on servers is often seasonal, and reliable forecasts are paramount for having a view of the future evolution and to improve capacity planning. This generic framework has been investigated. Three implementations of the framework were proposed. Each of them has been experimentally studied on a large number of different datasets in order 1) to find the best setting and 2) to conclude on the prediction accuracy of the method.

Table 7.3 gives a concise overview of the main elements of the deterministic forecast (see Chap. 4 and Chap. 5) and the probabilistic forecast (see Chap. 6). The deterministic forecast F2C (column 3 of the table) predicts a single value for all the timestamps. The probabilistic forecast PF2C (column 4 of the table) predicts a probability distribution for each timestamp. The table exhibits two main differences between the deterministic and the probabilistic forecast: (1) the way to create the clusters of TS, (2) the way to define the prototype of a cluster (*i.e.* the representative of a cluster).

Table 7.3: Possible configurations of the framework proposed.

		Deterministic forecast	**Probabilistic forecast**
Learning process	A) Time series Clustering	Clustering	Coclustering
	B) Learning Classifier	Classification	
Forecasting process	C) Time series Prototypes	Centroid or medoid	Density
	D) Forecast	Hard or Soft	

The intensive experiments allow drawing the following conclusion:

— **Chap. 4** introduces a deterministic implementation that is based on K-means and Markov models for learning the framework. Results are encouraging MAE-wise, and better than most of the deterministic opponents selected (AR, ARIMA, Holt Winters). The only opponent not defeated at this stage is SARIMA.

— **Chap. 5** improves the deterministic baseline previously introduced and proposes the F2C implementation. It is based on four different clusterers (*e.g.* K-means, K-shape, GAK and MODL) and four different classifiers (*e.g.* naive-bayes classifier, decision trees, random forests and logistic regression). The performances of each combination of parameters are assessed and the best combination found is the use of K-means with medoid prototypes, using a soft methodology and random forests. Results of the F2C methodology properly configured are good MAE-wise and all the opponents including SARIMA are outperformed. The impact of the number of seasons used for learning the models (γ) is investigated and it is shown that and a bigger γ can helps to produce better forecasts.

— **Chap. 6** proposes a probabilistic implementation of the framework: PF2C. This instantiation takes advantage of MODL coclustering algorithms to create probabilistic grids. Those can be used both to describe time series and to produce forecasts. The same four classifiers as F2C was used. Comparisons using the MAE is not possible for probabilistic forecasts, thus the use of CRPS measure to compare probabilistic forecasts. Results are good CRPS-wise and PF2C ranks at the third place, behind F2C method introduced in previous chapter and SARIMA model. A portfolio approach where the best parameters for each dataset are found using the validation set rather than best overall parameters for all dataset is proposed, and it effectively improves the results. Although, the bad performances of one probabilistic opponent Prophet which is state of the art for seasonal forecasting give the intuition that the CRPS is not really adapted as a performance metric for comparing probabilistic and deterministic forecasts.

One advantage of the framework proposed is that it is able to *produce predictions at the horizon of one season in one shot* (*i.e.*, there is no need to build different models for different horizons). The goal is to fight against the accumulation of error often observed with traditional point forecast methods.

This framework is also a way to *frame the forecasting problem as a classification problem.* The use of clustering to mine information about the various seasons of the time series can create knowledge more easily processed with classification rather than with traditional forecasting algorithms. The study of seasons relies heavily on the Hyp. $\mathcal{H}_5$ in Sec. 1.3.3, which states that there are several types of seasons in a time series. This hypothesis is verified in Chap. 7 with the study of time series provided by Orange Money project that shows explicit seasonality with several profiles of days. The use of clustering to extract seasons is interesting, especially while using the soft paradigm (see Sec. 3.3.2) to mix predicted cluster prototypes at forecasting time. It naturally helps to extract several seasonality from the data.

The framework proposed finally *follows a data-driven approach that only requires few easy-to-set parameters* (such as the length of one season for example). The method is therefore adapted to any type of incoming data – in the capacity planning problem, but also in other fields. This data-driven nature allows the integration of the framework in software like PerForecast [LML18], and it is really a plus for easing the management of complex infrastructure without the hassle of configuring complex algorithms.

Perspectives

Three perspectives are considered for improving the framework: (1) experimenting with exogenous data, (2) making comparisons with more probabilistic forecasting algorithms and (3) predicting several seasons at once instead of only one.

Exogenous data

Exogenous data have not been used in any version of the framework. For instance, for daily datasets, models do not take into account the position of the weekday to make forecasts. However, some observations suggest that it could improve the prediction accuracy.

For example, the clustering of daily time series set up with 2 clusters could extract two types of behaviours: a high activity profile (HA) from Mondays to Fridays and a low activity profile (LA) on Saturdays and Sundays. As a consequence, the trained classifiers will more likely predict a HA day after

another HA day (with probability 0.8). This means that every Friday, it will wrongly predict the day with a HA profile. A model with the weekday information will split the next day prediction rule in two different rules: High activity from Monday to Thursday leads to a high activity the next day while high activity on Friday leads to a low activity. It remains interesting to have information about the types of the days as, in real data, there are several different profiles of days that are not necessarily correlated to the weekday information.

More generally, using exogenous data requires more user intervention (to find the data sources, include them in the framework and classifiers, *etc.*), but this compromise could create less automatic tools with better forecasts, that will be improved by the new sources of data.

Probabilistic algorithm comparisons

In the future, it is paramount to investigate the relevance of using CRPS for comparison performances between deterministic and probabilistic method, and to compare the performances of PF2C against other probabilistic forecasting opponents.

First of all, other metrics should be considered. CRPS is indeed one metric among others: the pinball loss function, that is a quantile score, is for instance used in major probabilistic forecasting competitions such as GEFcom 2014 [Hon+16].

Furthermore, it would be interesting to compare PF2C with probabilistic methods such as the one presented in [GGN16]. The latter is the best probabilistic forecasting proposed in the GEFcom 2014. It was not implemented in this thesis because it requires a lot of non-trivial configurations, and also a lot of exogenous data not necessarily available for the datasets we have used.

Finally, only MODL coclustering has been assessed as probabilistic forecasting algorithms for the PF2C implementation. There exist other paradigms for creating probabilistic groups: fuzzy clustering introduced in [GK79] (sometimes based on autocorrelation like in [DM09]), or even other types of coclustering algorithms not based on MODL, such as [IPM09] which shares the parameterless advantage. It would be beneficial to include them in the framework, in order to follow the path engaged in F2C – an implementation which used several clustering algorithms selected according to their performances on given datasets.

Several seasons prediction

The three implementations of the framework proposed in this thesis provide forecasts for one season ahead. In future work, it would be interesting to forecast several next seasons: the interest for capacity planning is clear, because the more informed will be engineers about future evolution of the systems, the more informed will be their decisions about infrastructure update or improvements.

Modifying the framework for this purpose seems doable, by learning sequences of next seasons rather than one unique next season. It is therefore a matter of modifying the classifier target for learning those sequences. However, this modification raises several questions: (1) how to create the forecasts from the sequence predicted, (2) will this new methodology require more data for learning accurately the sequences and (3) how to make the probabilistic forecasting using MODL grids.

BIBLIOGRAPHY

[AE17] Abdulaziz Almalaq and George Edwards, « A review of deep learning methods applied on load forecasting », *in*: *Proceedings of the international conference on machine learning and applications (ICMLA)*, 2017, pp. 511–516.

[Ahd+05] Miika Ahdesmaki, Harri Lahdesmaki, Ron Pearson, Heikki Huttunen, and Olli Yli-Harja, « Robust detection of periodic time series measured from biological systems », *in*: *BMC bioinformatics* 6.*1* (2005), p. 117.

[Aka69] Hirotugu Akaike, « Fitting autoregressive models for prediction », *in*: *Annals of the institute of Statistical Mathematics* 21.*1* (1969), pp. 243–247.

[Aka70] Hirotugu Akaike, « Statistical predictor identification », *in*: *Annals of the Institute of Statistical Mathematics* 22.*1* (1970), pp. 203–217.

[Aka98] Hirotugu Akaike, « Autoregressive model fitting for control », *in*: *Selected Papers of Hirotugu Akaike*, 1998, pp. 153–170.

[Ale+19] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al., « GluonTS: Probabilistic time series models in python », *in*: *arXiv preprint 1906.05264* (2019).

[Alh99] Juha Alho, « On probabilistic forecasts of population and their uses », *in*: *Bulletin of the International Statistical Institute* 58 (1999).

[All08] John Allspaw, *The art of capacity planning: scaling web resources*, O'Reilly Media, Inc., 2008.

[An+13] Ning An, Weigang Zhao, Jianzhou Wang, Duo Shang, and Erdong Zhao, « Using multi-output feedforward neural network with empirical mode decomposition based signal filtering for electricity demand forecasting », *in*: *Energy* 49 (2013), pp. 279–288.

[AN07] Arthur Asuncion and David Newman, *UCI machine learning repository*, 2007.

[ASW15] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah, « Time-series clustering–a decade review », *in*: *Information Systems* 53 (2015), pp. 16–38.

[Bag+17] Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines, « The great time series classification bake off: An experimental evaluation of recently proposed algorithms », *in*: *Springer* 1602 (2017).

[Bal+11] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron, « Towards predictable datacenter networks », *in*: *Proceedings of the ACM SIGCOMM conference*, 2011, pp. 242–253.

[Bar08] Wolfgang Barth, *Nagios: System and network monitoring*, No Starch Press, 2008.

[BBC15] Alexis Bondu, Marc Boullé, and Antoine Cornuéjols, « Symbolic representation of time series: A hierarchical coclustering formalization », *in*: *Proceedings of the International Workshop on Advanced Analysis and Learning on Temporal Data*, (AALTD), 2015, pp. 3–16.

[BDC02] Peter Brockwell, Richard Davis, and Matthew Calder, *Introduction to time series and forecasting*, vol. 2, Springer, 2002.

[Bod+09] Peter Bodik, Rean Griffith, Charles Sutton, Armando Fox, Michael I Jordan, and David A Patterson, « Automatic exploration of datacenter performance regimes », *in*: *Proceedings of the workshop on Automated control for datacenters and clouds*, ACM, 2009, pp. 1–6.

[Bod+10] Peter Bodik, Armando Fox, Michael J Franklin, Michael I Jordan, and David A Patterson, « Characterizing, modeling, and generating workload spikes for stateful services », *in*: *Proceedings of the symposium on Cloud computing*, 2010, pp. 241–252.

[Bor+17] Kianoosh Boroojeni, Hadi Amini, Shahab Bahrami, SS Iyengar, Arif Sarwat, and Orkun Karabasoglu, « A novel multi-time-scale modeling for electric power demand forecasting: From short-term to medium-term horizon », *in*: *Electric Power Systems Research* 142 (2017), pp. 58–73.

[Bou12] Marc Boullé, « Functional data clustering via piecewise constant nonparametric density estimation », *in*: *Pattern Recognition* 45.*12* (2012), pp. 4389–4401.

[Bou16] Marc Boullé, « Khiops: Outil d'apprentissage supervisé automatique pour la fouille de grandes bases de données multi-tables », *in*: *Actes de la conférence Extraction et Gestion des Connaissances*, 2016, pp. 505–510.

[Box+15] George Box, Gwilym Jenkins, Gregory Reinsel, and Greta Ljung, *Time series analysis: forecasting and control*, John Wiley & Sons, 2015.

[Bre01] Leo Breiman, « Random forests », *in*: *Machine learning* 45.*1* (2001), pp. 5–32.

[BRG07] Veronica Berrocal, Adrian Raftery, and Tilmann Gneiting, « Combining spatial statistical and ensemble information in probabilistic weather forecasts », *in*: *Monthly Weather Review* 135.*4* (2007), pp. 1386–1402.

[BS12] Kanna Bhaskar and Sri Niwas Singh, « AWNN-assisted wind power forecasting using feed-forward neural network », *in*: *transactions on sustainable energy* 3.*2* (2012), pp. 306–315.

[BTL12] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne, « Machine learning strategies for time series forecasting », *in*: *European business intelligence summer school*, Springer, 2012, pp. 62–77.

[Bul01] Frank Bullen, « Weather Forecasting and its Critics », *in*: *The Leisure hour: an illustrated magazine for home reading* (1901), pp. 223–228.

[BV18] Kailash Budhathoki and Jilles Vreeken, « Causal Inference on Event Sequences », *in*: *Proceedings of the SIAM International Conference on Data Mining*, 2018, pp. 55–63.

[Cas+12] Carmelo Cassisi, Placido Montalto, Marco Aliotta, and Alfredo Pulvirenti, « Similarity measures and dimensionality reduction techniques for time series data mining », *in*: *Advances in data mining knowledge discovery and applications* (2012), pp. 71–96.

[Cat+07] João Paulo da Silva Catalão, Sílvio José Pinto Simões Mariano, VMF Mendes, and LAFM Ferreira, « Short-term electricity prices forecasting in a competitive market: A neural network approach », *in*: *Electric power systems research* 77.*10* (2007), pp. 1297–1304.

[CBK10] Varun Chandola, Arindam Banerjee, and Vipin Kumar, « Anomaly detection for discrete sequences: A survey », *in*: *Transactions on Knowledge and Data Engineering* 24.*5* (2010), pp. 823–839.

[CDM10] Eddy Caron, Frederic Desprez, and Adrian Muresan, « Forecasting for grid and cloud computing on-demand resources based on pattern matching », *in*: *Proceedings of the International Conference on Cloud Computing Technology and Science*, 2010, pp. 456–463.

[Cer+17] Vítor Cerqueira, Luís Torgo, Fábio Pinto, and Carlos Soares, « Arbitrated ensemble for time series forecasting », *in*: *Proceedings of the Joint European conference on machine learning and knowledge discovery in databases*, 2017, pp. 478–494.

[Cha93] Chris Chatfield, « Calculating interval forecasts », *in*: *Journal of Business & Economic Statistics* 11.*2* (1993), pp. 121–135.

[CHB07] Argon Chen, C-H Hsu, and Jakey Blue, « Demand planning approaches to aggregating and forecasting interrelated demands for safety stock and backup capacity planning », *in*: *Proceedings of the International Journal of Production Research* 45.*10* (2007), pp. 2269–2294.

[Cle+90] Robert Cleveland, William Cleveland, Jean McRae, and Irma Terpenning, « STL: A seasonal-trend decomposition », *in*: *Journal of official statistics* 6.*1* (1990), pp. 3–73.

[CLL19] Jian Cao, Zhi Li, and Jian Li, « Financial time series forecasting model based on CEEMDAN and LSTM », *in*: *Physica A: Statistical Mechanics and its Applications* 519 (2019), pp. 127–139.

[CPB09] Yacine Chakhchoukh, Patrick Panciatici, and Pascal Bondon, « Robust estimation of SARIMA models: Application to short-term load forecasting », *in*: *Proceedings of the Workshop on Statistical Signal Processing*, 2009, pp. 77–80.

[CPM10] Yacine Chakhchoukh, Patrick Panciatici, and Lamine Mili, « Electric load forecasting based on statistical robust methods », *in*: *Transactions on Power Systems* 26.*3* (2010), pp. 982–991.

[CS12] Delson Chikobvu and Caston Sigauke, « Regression-SARIMA modelling of daily peak electricity demand in South Africa », *in*: *Journal of Energy in Southern Africa* 23.*3* (2012), pp. 23–30.

[Cut11] Marco Cuturi, « Fast global alignment kernels », *in*: *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 929–936.

[DA12] Ahlame Douzal-Chouakria and Cécile Amblard, « Classification trees for time series », *in*: *Pattern Recognition* 45.*3* (2012), pp. 1076–1091.

[Dag78] Estela Bee Dagum, « Modelling, forecasting and seasonally adjusting economic time series with the X-11 ARIMA method », *in*: *Journal of the Royal Statistical Society*, The Statistician 27.*3/4* (1978), pp. 203–216.

[DB79] David Davies and Donald Bouldin, « A cluster separation measure », *in*: *Transactions on Pattern Analysis and Machine Intelligence* 1.*2* (1979), pp. 224–227.

[DBC15] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols, « Early classification of time series as a non myopic sequential decision making problem », *in*: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, (ECML/KDD), 2015, pp. 433–447.

[Dem06] Janez Demšar, « Statistical comparisons of classifiers over multiple data sets », *in*: *Journal of Machine learning research* 7 (2006), pp. 1–30.

[DH06] Jan De Gooijer and Rob Hyndman, « 25 years of time series forecasting », *in*: *International journal of forecasting* 22.*3* (2006), pp. 443–473.

[DHF84] David Dickey, David Hasza, and Wayne Fuller, « Testing for unit roots in seasonal time series », *in*: *Journal of the American Statistical Association* 79.*386* (1984), pp. 355–367.

[DHS11] Alysha De Livera, Rob Hyndman, and Ralph Snyder, « Forecasting time series with complex seasonal patterns using exponential smoothing », *in*: *Journal of the American statistical association* 106.*496* (2011), pp. 1513–1527.

[DM09] Pierpaolo D'Urso and Elizabeth Ann Maharaj, « Autocorrelation-based fuzzy clustering of time series », *in*: *Fuzzy Sets and Systems* 160.*24* (2009), pp. 3565–3589.

[Dob+08] J. Dobschinski, A. Wessel, B. Lange, K. Rohrig, L. von Bremen, and YM Saint-Drenan, « Estimation of wind power prediction intervals using stochastic methods and artificial intelligence model ensembles », *in*: *Proceedings of the German Wind Energy Conference DEWEK*, 2008.

[DP87] David Dickey and Sastry Pantula, « Determining the order of differencing in autoregressive processes », *in*: *Journal of Business & Economic Statistics* 5.*4* (1987), pp. 455–461.

[Dud16] Grzegorz Dudek, « Multilayer perceptron for GEFCom2014 probabilistic electricity price forecasting », *in*: *International Journal of Forecasting* 32.*3* (2016), pp. 1057–1060.

[DWS12] Brian Dougherty, Jules White, and Douglas Schmidt, « Model-driven auto-scaling of green cloud computing infrastructure », *in*: *Future Generation Computer Systems* 28.*2* (2012), pp. 371–378.

[Eil+06] Tamar Eilam, Michael Kalantar, Alexander Konstantinou, Giovanni Pacifici, John Pershing, and Aditya Agrawal, « Managing the configuration complexity of distributed applications in internet data centers », *in*: *Communications Magazine* 44.*3* (2006), pp. 166–177.

[Esp+05] Marcelo Espinoza, Caroline Joye, Ronnie Belmans, and Bart De Moor, « Short-term load forecasting, profile identification, and customer segmentation: a methodology based on periodic time series », *in*: *Transactions on Power Systems* 20.*3* (2005), pp. 1622–1630.

[Far10] Durdu Ömer Faruk, « A hybrid neural network and ARIMA model for water quality time series prediction », *in*: *Engineering applications of artificial intelligence* 23.*4* (2010), pp. 586–594.

[Fin+98] David Findley, Brian Monsell, William Bell, Mark Otto, and Bor-Chung Chen, « New capabilities and methods of the X-12-ARIMA seasonal-adjustment program », *in*: *Journal of Business & Economic Statistics* 16.*2* (1998), pp. 127–152.

[For15] Clay Ford, « Understanding QQ plots », *in*: (2015), p. 18.

[FRM94] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos, « Fast subsequence matching in time-series databases », *in*: *Sigmod Record* 23.*2* (1994), pp. 419–429.

[GET06] Clive William John Granger, Graham Elliott, and Allan Timmermann, *Handbook of Economic Forecasting*, vol. 1, Elsevier, 2006.

[GGN16] Pierre Gaillard, Yannig Goude, and Raphaël Nedellec, « Additive models and robust aggregation for GEFCom2014 probabilistic electric load and electricity price forecasting », *in*: *International Journal of forecasting* 32.*3* (2016), pp. 1038–1050.

[GK79] Donald Gustafson and William Kessel, « Fuzzy clustering with a fuzzy covariance matrix », *in*: *Proceedings of the conference on decision and control*, 1979, pp. 761–766.

[Gne08] Tilmann Gneiting, « Probabilistic forecasting », *in*: *Journal of the Royal Statistical Society*, Statistics in Society (2008), pp. 319–321.

[Gne11] Tilmann Gneiting, « Making and evaluating point forecasts », *in*: *Journal of the American Statistical Association* 106.*494* (2011), pp. 746–762.

[GÖT10] Paul Goodwin, Dilek Önkal, and Mary Thomson, « Do forecasts expressed as prediction intervals improve production planning decisions? », *in*: *European Journal of Operational Research* 205.*1* (2010), pp. 195–201.

[Gou+08] Phillip Gould, Anne Koehler, Keith Ord, Ralph Snyder, Rob Hyndman, and Farshid Vahid-Araghi, « Forecasting time series with multiple seasonal patterns », *in*: *European Journal of Operational Research* 191.*1* (2008), pp. 207–222.

[GRG04] Yulia Gel, Adrian Raftery, and Tilmann Gneiting, « Calibrated probabilistic mesoscale weather field forecasting: The geostatistical output perturbation method », *in*: *Journal of the American Statistical Association* 99.*467* (2004), pp. 575–583.

[GS91] Jim Gray and Daniel Siewiorek, « High-availability computer systems », *in*: *Computer* 24.*9* (1991), pp. 39–48.

[GSC99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, *Learning to forget: Continual prediction with LSTM*, tech. rep., 1999.

[GW68] Ramanathan Gnanadesikan and Martin Wilk, « Probability plotting methods for the analysis of data », *in*: *Biometrika* 55.*1* (1968), pp. 1–17.

[GWK89] Clive William John Granger, Halbert White, and Mark Kamstra, « Interval forecasting: an analysis based upon ARCH-quantile estimators », *in*: *Journal of Econometrics* 40.*1* (1989), pp. 87–96.

[Har93] Andrew Harvey, « Time series models », *in*: (1993).

[Has17] Trevor Hastie, « Generalized additive models », *in*: *Statistical models in S*, Routledge, 2017, pp. 249–307.

[Hea+05] Nicholas Heard, Christopher Holmes, David Stephens, David Hand, and George Dimopoulos, « Bayesian coclustering of Anopheles gene expression time series: study of immune defense response to multiple experimental challenges », *in*: *Proceedings of the National Academy of Sciences* 102.*47* (2005), pp. 16939–16944.

[Her00] Hans Hersbach, « Decomposition of the continuous ranked probability score for ensemble prediction systems », *in*: *Weather and Forecasting* 15.*5* (2000), pp. 559–570.

[Ho95] Tin Kam Ho, « Random decision forests », *in*: *Proceedings of International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282.

[Hon+16] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob Hyndman, *Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond*, 2016.

[HOR96] Tim Hill, Marcus O'Connor, and William Remus, « Neural network models for time series forecasts », *in*: *Management science* 42.*7* (1996), pp. 1082–1092.

[Hyn11] Rob Hyndman, *Time series data library (TSDL)*, 2011, URL: `http://robjhyndman.com/TSDL`.

[IM05] Zuhaimy Ismail and Khairil Asmani Mahpol, « SARIMA model for forecasting Malaysian electricity generated », *in*: *Matematika* 21 (2005), pp. 143–152.

[IOB09] Ruhaizan Ismail, Zalinda Othman, and Azuraliza Abu Bakar, « Data mining in production planning and scheduling: A review », *in*: *Proceedings of the Conference on Data Mining and Optimization*, 2009, pp. 154–159.

[IPM09] Dino Ienco, Ruggero G Pensa, and Rosa Meo, « Parameter-free hierarchical co-clustering by n-ary splits », *in*: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2009, pp. 580–595.

[Iyi10] Cem Iyigun, « Probabilistic distance clustering », *in*: *Wiley Encyclopedia of Operations Research and Management Science* (2010).

[JAS11] Tina Jakaša, Ivan Andročec, and Petar Sprčić, « Electricity price forecasting – ARIMA model approach », *in*: *Proceedings of the International Conference on the European Energy Market (EEM)*, 2011, pp. 222–225.

[KAV15] Ina Khandelwal, Ratnadip Adhikari, and Ghanshyam Verma, « Time series forecasting using hybrid ARIMA and ANN models based on DWT decomposition », *in*: *Procedia Computer Science* 48.*1* (2015), pp. 173–179.

[KB11] Mehdi Khashei and Mehdi Bijari, « A novel hybridization of artificial neural networks and ARIMA models for time series forecasting », *in*: *Applied Soft Computing* 11.*2* (2011), pp. 2664–2675.

[Kei01] Nico Keilman, « Uncertain population forecasts », *in*: *Nature* 412.*6846* (2001), pp. 490–491.

[Kha+12] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis, « Workload characterization and prediction in the cloud: A multiple time series approach », *in*: *Proceedings of the Network Operations and Management Symposium*, IEEE, 2012, pp. 1287–1294.

[KHM05] Yoshio Kajitani, Keith W Hipel, and A Ian McLeod, « Forecasting nonlinear time series with feed-forward neural networks: a case study of Canadian lynx data », *in*: *Journal of Forecasting* 24.*2* (2005), pp. 105–117.

[Kle+02] David Kleinbaum, Dietz, Gail, Mitchel Klein, and Mitchell Klein, *Logistic regression*, Springer, 2002.

[KM06] YH Kareem and Asso Raouf Majeed, « Monthly Peak-load Demand Forecasting for Sulaimany Governorate Using SARIMA. », *in*: *Proceedings of the International Conference on Transmission & Distribution Conference and Exposition*, 2006, pp. 1–5.

[Koo+13] Bon-Gil Koo, Min-Seok Kim, Kyu-Han Kim, Hee-Tae Lee, June-Ho Park, and Cheol-Hong Kim, « Short-term electric load forecasting using data mining technique », *in*: *Proceedings of the International Conference on Intelligent Systems and Control (ISCO)*, 2013, pp. 153–157.

[KPH02] Nico Keilman, Dinh Quang Pham, and Arve Hetland, « Why population forecasts should be probabilistic-illustrated by the case of Norway », *in*: *Demographic research* 6 (2002), pp. 409–454.

[KRA11] Irena Koprinska, Mashud Rana, and Vassilios Agelidis, « Yearly and seasonal models for electricity load forecasting », *in*: *Proceedings of the International Joint Conference on Neural Networks*, 2011, pp. 1474–1481.

[KS09] Rajesh Kavasseri and Krithika Seetharaman, « Day-ahead wind speed forecasting using f-ARIMA models », *in*: *Renewable Energy* 34.*5* (2009), pp. 1388–1393.

[KS76] John Kemeny and Laurie Snell, *Markov chains*, Springer-Verlag, New York, 1976.

[Kum+18] Sumit Kumar, Lasani Hussain, Sekhar Banarjee, and Motahar Reza, « Energy load forecasting using deep learning approach-LSTM and GRU in spark cluster », *in*: *Proceedings of the International Conference on Emerging Applications of Information Technology (EAIT)*, 2018, pp. 1–4.

[Lar+19] James Large, Anthony Bagnall, Simon Malinowski, and Romain Tavenard, « On time series classification with dictionary-based classifiers », *in*: *Proceedings of the Intelligent Data Analysis* 23.*5* (2019), pp. 1073–1089.

[Lev+18] Colin Leverger, Vincent Lemaire, Simon Malinowski, Thomas Guyet, and Laurence Roze, « Day-ahead time series forecasting: application to capacity planning », *in*: *Proceedings of the workshop on Advanced Analytics and Learning of Temporal Data (AALTD)*, 2018.

[Lev+19] Colin Leverger, Simon Malinowski, Thomas Guyet, Vincent Lemaire, Alexis Bondu, and Alexandre Termier, « Toward a framework for seasonal time series forecasting using clustering », *in*: *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, 2019, pp. 328–340.

[Lin+04] Jessica Lin, Michail Vlachos, Eamonn Keogh, and Dimitrios Gunopulos, « Iterative incremental clustering of time series », *in*: *International Conference on Extending Database Technology*, 2004, pp. 106–122.

[Llo82] Stuart Lloyd, « Least squares quantization in PCM », *in*: *Transactions on information theory* 28.*2* (1982), pp. 129–137.

[LML18] Colin Leverger, Régis Marguerie, and Vincent Lemaire, « PerForecast : un outil d'automatisation de planning capacitaire et de prévision de l'évolution de séries temporelles univariées », *in*: *Conférence Extraction et Gestion des Connaissances (EGC) Paris*, 2018.

[LQ12] Dominique Ladiray and Benoit Quenneville, *Seasonal adjustment with the X-11 method*, vol. 158, Springer Science & Business Media, 2012.

[LX12] Helmut Lütkepohl and Fang Xu, « The role of the log transformation in forecasting economic variables », *in*: *Empirical Economics* 42.*3* (2012), pp. 619–638.

[Mar61] Melvin Earl Maron, « Automatic indexing: an experimental inquiry », *in*: *Journal of the ACM (JACM)* 8.*3* (1961), pp. 404–417.

[Mel16] City Of Melbourne, *Pedestrian Counting System*, 2016, URL: `http://www.pedestrian.melbourne.vic.gov.au/`.

[MG+11] Peter Mell, Tim Grance, et al., « The NIST definition of cloud computing », *in*: (2011).

[MH11] Ming Mao and Marty Humphrey, « Auto-scaling to minimize cost and meet application deadlines in cloud workflows », *in*: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12.

[MH97] Spyros Makridakis and Michele Hibon, « ARMA models and the Box–Jenkins methodology », *in*: *Journal of Forecasting* 16.*3* (1997), pp. 147–163.

[Moo+05] Justin Moore, Jeff Chase, Keith Farkas, and Parthasarathy Ranganathan, « Data center workload monitoring, analysis, and emulation », *in*: *Proceedings of the workshop on computer architecture evaluation using commercial workloads*, 2005, pp. 1–8.

[MS96] Radu Manuca and Robert Savit, « Stationarity and nonstationarity in time series analysis », *in*: *Physica D: Nonlinear Phenomena* 99.*2-3* (1996), pp. 134–161.

[Nel+99] Michael Nelson, Tim Hill, William Remus, and Marcus O'Connor, « Time series forecasting using neural networks: Should the data be deseasonalized first? », *in*: *Journal of forecasting* 18.*5* (1999), pp. 359–367.

[Nie04] Ralph Niels, « Dynamic time warping », *in*: *Artificial Intelligence* (2004).

[Nie94] Jakob Nielsen, *Usability engineering*, Elsevier, 1994.

[Pap+05a] Maria Papadopouli, Haipeng Shen, Elias Raftopoulos, Manolis Ploumidis, and Felix Hernandez-Campos, « Short-term traffic forecasting in a campus-wide wireless network », *in*: *Proceedings of the International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 3, 2005, pp. 1446–1452.

[Pap+05b] Konstantina Papagiannaki, Nina Taft, Zhi-Li Zhang, and Christophe Diot, « Long-term forecasting of Internet backbone traffic », *in*: *transactions on neural networks* 16.*5* (2005), pp. 1110–1124.

[Ped+11] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., « Scikit-learn: Machine learning in Python », *in*: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[PG15] John Paparrizos and Luis Gravano, « k-shape: Efficient and accurate clustering of time series », *in*: *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2015, pp. 1855–1870.

[Pin+07] Pierre Pinson, Henrik Aa Nielsen, Jan K Møller, Henrik Madsen, and George N Kariniotakis, « Non-parametric probabilistic forecasts of wind power: required properties and evaluation », *in*: *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology* 10.*6* (2007), pp. 497–516.

[PKG11] François Petitjean, Alain Ketterlin, and Pierre Gançarski, « A global averaging method for dynamic time warping, with applications to clustering », *in*: *Pattern Recognition* 44.*3* (2011), pp. 678–693.

[PL05] Ping-Feng Pai and Chih-Sheng Lin, « A hybrid ARIMA and support vector machines model in stock price forecasting », *in*: *Omega* 33.*6* (2005), pp. 497–505.

[Raf94] Lawrence Raffalovich, « Detrending time series: A cautionary note », *in*: *Sociological Methods & Research* 22.*4* (1994), pp. 492–519.

[RDG11] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale, « Efficient autoscaling in the cloud using predictive models for workload forecasting », *in*: *Proceedings of the 4th International Conference on Cloud Computing*, 2011, pp. 500–507.

[Rey09] Douglas Reynolds, « Gaussian Mixture Models. », *in*: *Encyclopedia of biometrics* 741 (2009).

[Ris+01] Irina Rish et al., « An empirical study of the naive Bayes classifier », *in*: *Proceedings of the Workshop on Empirical Methods in Artificial Intelligence*, 2001, pp. 41–46.

[Rou87] Peter Rousseeuw, « Silhouettes: a graphical aid to the interpretation and validation of cluster analysis », *in*: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65.

[Saâ17] Foued Saâdaoui, « A seasonal feedforward neural network to forecast electricity prices », *in*: *Neural Computing and Applications* 28.*4* (2017), pp. 835–847.

[Sáe+14] Doris Sáez, Fernand Ávila, Daniel Olivares, Claudio Cañizares, and Luis Marín, « Fuzzy prediction interval models for forecasting renewable resources and loads in microgrids », *in*: *Transactions on Smart Grid* 6.*2* (2014), pp. 548–556.

[SC78] Hiroaki Sakoe and Seibi Chiba, « Dynamic programming algorithm optimization for spoken word recognition », *in*: *IEEE transactions on acoustics, speech, and signal processing* 26.*1* (1978), pp. 43–49.

[Sel+17] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman, « Stock price prediction using LSTM, RNN and CNN-sliding window model », *in*: *Proceedings of the international conference on advances in computing, communications and informatics*, IEEE, 2017, pp. 1643–1647.

[SG11] Ying Sun and Marc Genton, « Functional boxplots », *in*: *Journal of Computational and Graphical Statistics* 20.*2* (2011), pp. 316–334.

[SGR10] McLean Sloughter, Tilmann Gneiting, and Adrian Raftery, « Probabilistic wind speed forecasting using ensembles and Bayesian model averaging », *in*: *Journal of the american statistical association* 105.*489* (2010), pp. 25–35.

[SK19] Alaa Sagheer and Mostafa Kotb, « Time series forecasting of petroleum production using deep LSTM recurrent networks », *in*: *Neurocomputing* 323 (2019), pp. 203–213.

[SL91] Rasoul Safavian and David Landgrebe, « A survey of decision tree classifier methodology », *in*: *transactions on systems, man, and cybernetics* 21.*3* (1991), pp. 660–674.

[Sne02] Rob Snevely, *Enterprise data center design and methodology*, Prentice Hall Press, 2002.

[Sor+07] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse, « Methodology for long-term prediction of time series », *in*: *Neurocomputing* 70.*16-18* (2007), pp. 2861–2869.

[Sti85] Robert Stine, « Bootstrap prediction intervals for regression », *in*: *Journal of the American Statistical Association* 80.*392* (1985), pp. 1026–1031.

[SWW18] Shaolong Sun, Yunjie Wei, and Shouyang Wang, « Adaboost-lstm ensemble learning for financial time series forecasting », *in*: *Proceedings of the International Conference on Computational Science*, 2018, pp. 590–597.

[Tab13] Bahman Rostami Tabar, « ARIMA demand forecasting by aggregation », PhD thesis, 2013.

[Tai+12] Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa, « A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition », *in*: *Expert systems with applications* 39.*8* (2012), pp. 7067–7083.

[Tav+20] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Ruswurm, Kushal Kolar, et al., « Tslearn, A Machine Learning Toolkit for Time Series Data », *in*: *Journal of Machine Learning Research* 21.*118* (2020), pp. 1–6.

[TG10] Thordis Thorarinsdottir and Tilmann Gneiting, « Probabilistic forecasts of wind speed: ensemble model output statistics by using heteroscedastic censored regression », *in*: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 173.*2* (2010), pp. 371–388.

[Tia09] Wenhong Tian, « Adaptive dimensioning of cloud data centers », *in*: *Proceedings of the International Conference on Dependable, Autonomic and Secure Computing*, 2009, pp. 5–10.

[TL18] Sean Taylor and Benjamin Letham, « Forecasting at scale », *in*: *The American Statistician* 72.*1* (2018), pp. 37–45.

[TT94] George Tiao and Ruey Tsay, « Some advances in non-linear and adaptive modelling in time-series », *in*: *Journal of forecasting* 13.*2* (1994), pp. 109–131.

[TW00] Anthony Tay and Kenneth Wallis, « Density forecasting: a survey », *in*: *Journal of forecasting* 19.*4* (2000), pp. 235–254.

[TYT02] Fang-Mei Tseng, Hsiao-Cheng Yu, and Gwo-Hsiung Tzeng, « Combining neural network model with seasonal time series ARIMA model », *in*: *Technological Forecasting and Social Change* 69.*1* (2002), pp. 71–87.

[VKJ15] Carlos Vazquez, Ram Krishnan, and Eugene John, « Time Series Forecasting of Cloud Data Center Workloads for Dynamic Resource Provisioning. », *in*: *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 6.*3* (2015), pp. 87–110.

[Wan+11] Chengwei Wang, Krishnamurthy Viswanathan, Lakshminarayan Choudur, Vanish Talwar, Wade Satterfield, and Karsten Schwan, « Statistical techniques for online anomaly detection in data centers », *in*: *Proceedings of the International Symposium on Integrated Network Management and Workshops*, 2011, pp. 385–392.

[WB98] Christopher Williams and David Barber, « Bayesian classification with Gaussian processes », *in*: *Transactions on Pattern Analysis and Machine Intelligence* 20.*12* (1998), pp. 1342–1351.

[Wei06] William WS Wei, « Time series analysis: univariate and multivariate », *in*: *Methods. Boston, MA: Pearson Addison Wesley* (2006).

[Wei18] Andreas Weigend, *Time series prediction: forecasting the future and understanding the past*, Routledge, 2018.

[Wer14] Rafał Weron, « Electricity price forecasting: A review of the state-of-the-art with a look into the future », *in*: *International journal of forecasting* 30.*4* (2014), pp. 1030–1081.

[WFS04] Sofia Wichert, Konstantinos Fokianos, and Korbinian Strimmer, « Identifying periodically expressed transcripts in microarray time series data », *in*: *Bioinformatics* 20.*1* (2004), pp. 5–20.

[Win60] Peter Winters, « Forecasting sales by exponentially weighted moving averages », *in*: *Management science* 6.*3* (1960), pp. 324–342.

[WM12] Xiping Wang and Ming Meng, « A Hybrid Neural Network and ARIMA Model for Energy Consumption Forcasting. », *in*: *JCP* (2012), pp. 1184–1190.

[WMH98] Steven Wheelwright, Spyros Makridakis, and Rob Hyndman, *Forecasting: methods and applications*, John Wiley & Sons, 1998.

[Wor27] Holbrook Working, « Forecasting the price of wheat », *in*: *Journal of Farm Economics* 9.*3* (1927), pp. 273–287.

[WS10] Lei Wu and Mohammad Shahidehpour, « A hybrid model for day-ahead price forecasting », *in*: *Transactions on Power Systems* 25.*3* (2010), pp. 1519–1530.

[WZK15] Xiaojing Wu, Raul Zurita-Milla, and Menno-Jan Kraak, « Co-clustering geo-referenced time series: exploring spatio-temporal patterns in Dutch temperature data », *in*: *International Journal of Geographical Information Science* 29.*4* (2015), pp. 624–642.

[Xie+13] Mengchen Xie, Claes Sandels, Kun Zhu, and Lars Nordström, « A seasonal ARIMA model with exogenous variables for elspot electricity prices in Sweden », *in*: *Proceedings of the International Conference on the European Energy Market (EEM)*, 2013, pp. 1–4.

[XSC12] Zhen Xiao, Weijia Song, and Qi Chen, « Dynamic resource allocation using virtual machines for cloud computing environment », *in*: *Transactions on parallel and distributed systems* 24.*6* (2012), pp. 1107–1117.

[Ye+14] Zhen Ye, Sajib Mistry, Athman Bouguettaya, and Hai Dong, « Long-term QoS-aware cloud service composition using multivariate time series analysis », *in*: *Transactions on Services Computing* 9.*3* (2014), pp. 382–393.

[Zam+14] Michaël Zamo, Olivier Mestre, Philippe Arbogast, and Olivier Pannekoucke, « A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production, part I: Deterministic forecast of hourly production », *in*: *Solar Energy* 105 (2014), pp. 792–803.

[ZE02] Bianca Zadrozny and Charles Elkan, « Transforming classifier scores into accurate multiclass probability estimates », *in*: *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*, 2002, pp. 694–699.

[Zha03] Peter Zhang, « Time series forecasting using a hybrid ARIMA and neural network model », *in*: *Neurocomputing* 50 (2003), pp. 159–175.

[Zhe+17] Jian Zheng, Cencen Xu, Ziang Zhang, and Xiaohua Li, « Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network », *in*: *Proceedings of the Annual Conference on Information Sciences and Systems (CISS)*, 2017, pp. 1–6.

[Zho+06] M. Zhou, Z. Yan, YX Ni, G. Li, and Yi Nie, « Electricity price forecasting with confidence-interval estimation through an extended ARIMA approach », *in*: *Generation, Transmission and Distribution* 153.*2* (2006), pp. 187–195.

[ZPH01] Peter Zhang, Eddy Patuwo, and Michael Hu, « A simulation study of artificial neural networks for nonlinear time-series forecasting », *in*: *Computers & Operations Research* 28.*4* (2001), pp. 381–396.

[ZQ05] Peter Zhang and Min Qi, « Neural network forecasting for seasonal and trend time series », *in*: *European journal of operational research* 160.*2* (2005), pp. 501–514.

[ZXZ17] Qun Zhuge, Lingyu Xu, and Gaowei Zhang, « LSTM Neural Network with Emotional Analysis for Prediction of Stock Price. », *in*: *Engineering letters* 25.*2* (2017).

PUBLICATIONS

[Lev+18] Colin Leverger, Vincent Lemaire, Simon Malinowski, Thomas Guyet, and Laurence Roze, « Day-ahead time series forecasting: application to capacity planning », *in*: *Proceedings of the workshop on Advanced Analytics and Learning of Temporal Data (AALTD)*, 2018.

[Lev+19] Colin Leverger, Simon Malinowski, Thomas Guyet, Vincent Lemaire, Alexis Bondu, and Alexandre Termier, « Toward a framework for seasonal time series forecasting using clustering », *in*: *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, 2019, pp. 328–340.

[LML18] Colin Leverger, Régis Marguerie, and Vincent Lemaire, « PerForecast : un outil d'automatisation de planning capacitaire et de prévision de l'évolution de séries temporelles univariées », *in*: *Conférence Extraction et Gestion des Connaissances (EGC) Paris*, 2018.

List of Figures

List of Tables

APPENDIX

Appendix 1: data used for experiments

The Table below explicits the datasets used for the experimental parts of this thesis.
All the TS have been z-normalised beforehand for comparing different MAE.

Table 7.4: Table which summarise the various datasets used for experimentations.

Dataset	*Origin*	*Acquisition freq.*	*No. pt-s/seas*	*No. seas*
Orange CPUs	Orange	15 mins	96	512
Orange Hits per quarter	Orange	15 mins	96	512
Pedestrian Counting System	City of Melbourne [Mel16]	1 hour	24	1490
Daily rainfall Melbourne from from 1981 to 1990	tsdl [Hyn11]	1 day	7	428
No. of Births in Quebec from Jan. 1, 1977 to Dec. 31, 1990	-	1 day	7	428
Daily maximum temperatures in Melbourne, Australia, 1981-1990	-	1 day	7	470
Internet traffic data I from Jun. 7, 2005 to Jul. 31, 2005	-	1 hour	24	51
Internet traffic data II from Nov. 19, 2004 to Jan. 27, 2005	-	1 hour	24	69
Internet traffic data III from Nov. 19, 2004 to Jan. 27, 2005	-	5 minutes	288	51
Quarter-monthly rainfall, Lac St-Jean Region, 1953 - 1982	-	4 months	4	30
Quarter-monthly river flow, Lac St-Jean Region, 1953 - 1982	-	4 months	4	30
Monthly mean discharge in cubic meters per second for the period of record, Niagara River at Queenston, 1860-1990	-	yearly	1 month	31
Monthly beer production Australia megalitres	-	1 month	12	39
Monthly sunspot Zuerich	-	1 month	12	235

Table 7.4 continued...

Dataset	*Origin*	*Acquisition freq.*	*No. pt-s/seas*	*No. seas*
Flow of Jokulsa Eystri river from Jan. 1, 1972 to Dec. 31, 1974	-	1 day	6	182
Electricity production	kaggle [1]	1 hour	12	32
Weather Canada	kaggle [2]	1 hour	24	57
Rossman Sales	kaggle [3]	1 day	7	82
Currency	kaggle [4]	1 day	3	100
CO.GT	Air quality indicators from Mar. 10, 2004 to Apr. 04 2005 in an Italian city [AN07]	1 hour	12	250
PT08.S1.CO	-	1 hour	12	250
C6H6.GT	-	1 hour	12	250
PT08.S2.NMHC	-	1 hour	12	250
NOx.GT	-	1 hour	12	250
PT08.S3.NOx	-	1 hour	12	250
NO2.GT	-	1 hour	24	125
PT08.S4.NO2	-	1 hour	12	250
PT08.S5.O3	-	1 hour	12	250
RH	-	1 hour	24	125
Humidity	Bike sharing from Jan. 1, 2011 [Cer+17]	1 hour	23	58
Total bike rentals	-	1 hour	23	58
Global horizontal radiation	Solar radiation monitoring from Apr. 25, 2016 to Aug. 25, 2016 [Cer+17]	1 hour	14	214
Direct normal radiation	-	1 hour	14	214
Diffuse horizontal radiation	-	1 hour	14	214
Amial	Porto water consumption from different locations in the city of Porto from Nov. 11, 2015 to Jan. 11, 2016 [Cer+17]	30 minutes	48	62
Preciosa mar	-	30 minutes	48	62

1. See: `https://www.kaggle.com/robikscube/hourly-energy-consumption`
2. See: `https://www.kaggle.com/selfishgene/historical-hourly-weather-data?select=temperature.csv`
3. See: `https://www.kaggle.com/c/rossmann-store-sales`
4. See: `https://www.kaggle.com/kashnitsky/topic-9-part-1-time-series-analysis-in-python`

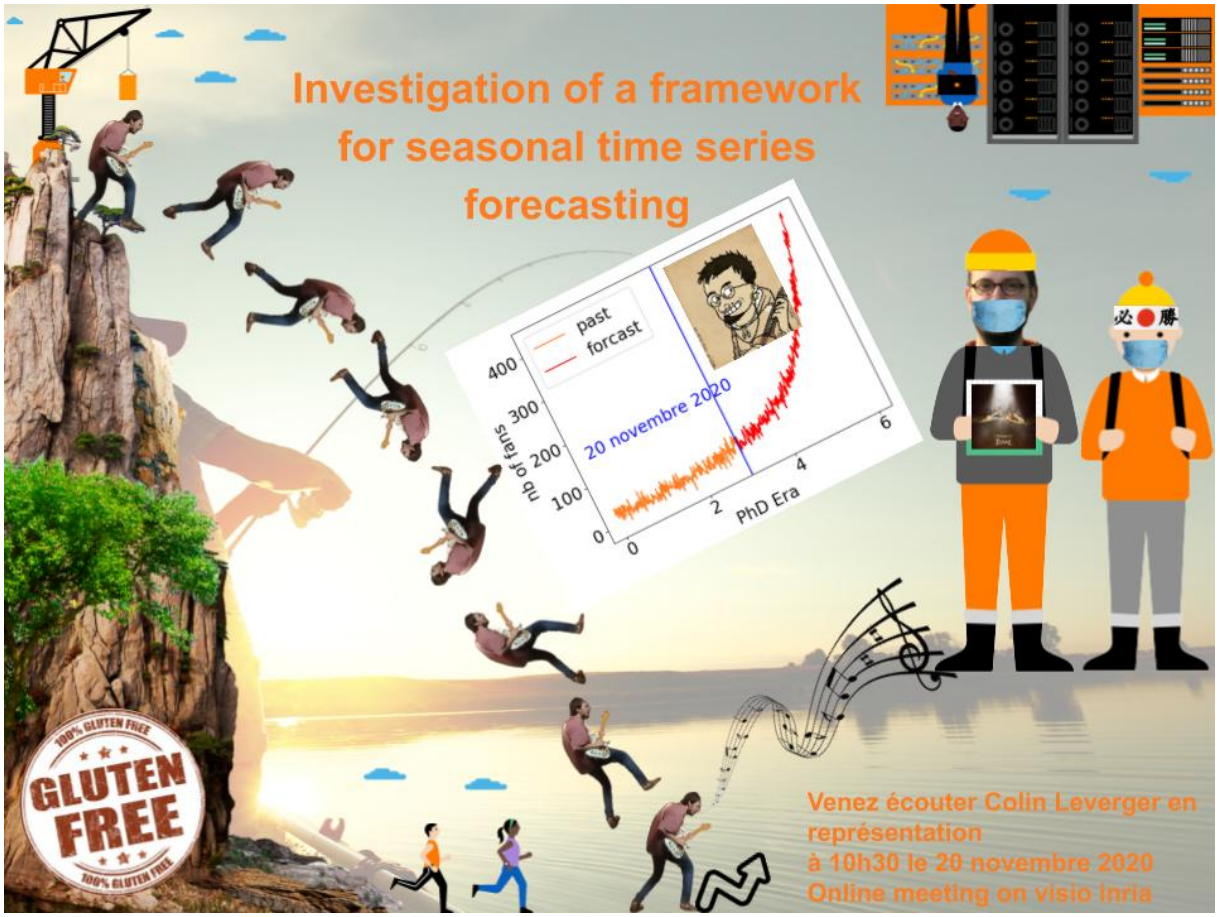
Table 7.4 continued...

Dataset	*Origin*	*Acquisition freq.*	*No. pt-s/seas*	*No. seas*
Electricity total load	Hospital energy loads from Jan. 1, 2016 to Mar. 25, 2016 [Cer+17]	1 hour	24	125
Equipment load	-	1 hour	24	125
Gas energy	-	1 hour	24	125
Gas heat energy	-	1 hour	24	125
Water heater Energy	-	1 hour	24	125
Electricity Recommended retail price	Australian electricity from Jan. 1, 1999 to Mar. 1, 1999 [KRA11]	30 minutes	6	472
Electricity Total demand	-	minutes	48	59
tide	San Francisco sea level (source: NOAA [5])	6 minutes	124	112
311SF	Number of calls for 'Graffitis' cases to the San Francisco call center (source: SF Open data [6])	1 hour	24	75
Bidmc	Electrocardiogram (ECG) (source: Physionet [7])	125 Hz	45	69
Traffic	New York Traffic Volume (source: New York Metropolitan Transportation Council [8])	1 hour	24	106
Enedis	Electricity consumption (source: Enedis [9])	30 mins	48	143
Mon pax web	Adelaide Airport Aircraft Movements (source: Australian Bureau of Infrastructure, Transport and Regional Economics: [10])	monthly	12	114

5. See: `https://coastwatch.pfeg.noaa.gov/erddap/`
6. See: `https://data.sfgov.org/City-Infrastructure/311-Cases/vw6y-z8j6`
7. See: `https://physionet.org/content/bidmc/1.0.0/`
8. See: `https://opendata.cityofnewyork.us/`
9. See: `https://data.enedis.fr`
10. See: `https://data.gov.au/data/dataset/airport-traffic-data`

Appendix 2: affiche de thèse

Affiche à visée humouristique, composée par mes collègues doctorants pour annoncer la date de soutenance de ma thèse.



UNIVERSITÉ DE RENNES 1

Titre : Investigations sur un framework pour des prévisions de séries temporelles saisonnières

Mot clés : Apprentissage automatique, séries temporelles, saisonnalité, clustering, classification

Résumé : Pour déployer des applications web, l'utilisation de serveurs informatique est primordiale. S'ils sont peu nombreux, les performances des applications peuvent se détériorer. En revanche, s'ils sont trop nombreux, les ressources sont gaspillées et les coûts argumentés. Dans ce contexte, les ingénieurs utilisent des outils de planning capacitaire qui leur permettent de suivre les performances des serveurs, de collecter les données temporelles générées par les infrastructures et d'anticiper les futurs besoins. La nécessité de créer des prévisions fiables apparait évidente. Les données des infrastructures présentent souvent une saisonnalité flagrante. Le cycle d'activité suivi par l'infrastructure est déterminé par certains cycles saisonniers (par exemple, le rythme quotidien de l'activité des utilisateurs). Cette thèse présente un *framework* pour la prévision de séries temporelles saisonnières. Ce *framework* est composé de deux modèles d'apprentissage automatique (*e.g.* clustering et classification) et vise à fournir des prévisions fiables à moyen terme avec un nombre limité de paramètres. Trois implémentations du *framework* sont présentées : une *baseline*, une déterministe et une probabiliste. La *baseline* est constituée d'un algorithme de clustering K-means et de modèles de Markov. La version déterministe est constituée de plusieurs algorithmes de clustering (K-means, K-shape, GAK et MODL) et de plusieurs classifieurs (classifieurs bayésiens, arbres de décisions, forêt aléatoire et régression logistique). La version probabiliste repose sur du coclustering pour créer des grilles probabilistes de séries temporelles, afin de décrire les données de manière non supervisée. Les performances des différentes implémentations du *framework* sont comparées avec différents modèles de l'état de l'art, incluant les modèles autorégressifs, les modèles ARIMA et SARIMA, les modèles Holts Winters, ou encore Prophet pour la partie probabiliste. Les résultats de la *baseline* sont encourageants, et confirment l'intérêt pour le *framework* proposé. De bons résultats sont constatés pour la version déterministe du *framework*, et des résultats corrects pour la version probabiliste. Un cas d'utilisation d'Orange est étudié, et l'intérêt et les limites de la méthodologie sont montrés.

Title: Investigation of a framework for seasonal time series forecasting

Keywords: Machine Learning, time series, seasons, clustering, classification

Abstract: To deploy web applications, using web servers is paramount. If there is too few of them, applications performances can quickly deteriorate. However, if they are too numerous, the resources are wasted and the cost increased. In this context, engineers use capacity planning tools to follow the performances of the servers, to collect time series data and to anticipate future needs. The necessity to create reliable forecasts seems clear. Data generated by the infrastructure often exhibit seasonality. The activity cycle followed by the infrastructure is determined by some seasonal cycles (for example, the user's daily rhythms). This thesis introduces a framework for seasonal time series forecasting. This framework is composed of two machine learning models (*e.g.* clustering and classification) and aims at producing reliable midterm forecasts with a limited number of parameters. Three instantiations of the framework are presented: one baseline, one deterministic and one probabilistic. The baseline is composed of K-means clustering algorithms and Markov Models. The deterministic version is composed of several clustering algorithms (K-means, K-shape, GAK and MODL) and of several classifiers (naive-bayes, decision trees, random forests and logistic regression). The probabilistic version relies on coclustering to create time series probabilistic grids, that are used to describe the data in an unsupervised way. The performances of the various implementations are compared with several state-of-the-art models, including the autoregressive models, ARIMA and SARIMA, Holt Winters, or even Prophet for the probabilistic paradigm. The results of the baseline are encouraging and confirm the interest for the framework proposed. Good results are observed for the deterministic implementation, and correct results for the probabilistic version. One Orange use case is studied, and the interest and limits of the methodology are discussed.