Projet RAMP : Air Passengers

Thierry GAMEIRO MARTINS Samir LAZZALI

---

1 – INTRODUCTION

Le défi de ce RAMP est de trouver de bonnes données qui peuvent être corrélées au trafic aérien et de le prédire pour les vols US entre la période 2011 et 2013. La qualité de la prédiction est mesurée par RMSE.

Notre score obtenu sur la plateforme est le suivant :

3	titigmr	civil_warssss	0.266	176.062299	228.050452
4	samirLazzali	The_Notorious	0.266	178.159817	226.813781

2 – AJOUTS DE DONNÉES EXTERNES

Pour ajouter de nouvelles données, nous étions contraints d'utiliser le fichier « external data » présent dans le dossier « submission ». Celui-ci contenait déjà les données de météo.

1 – Les données météorologiques : le fichier des données externes contient des informations météorologiques journalières (Température, Vitesse du vent, Visibilité, Pluviométrie, etc…) entre la période 2011 à 2013 pour chaque aéroport. Toutefois, on remarque que les colonnes « Max gust speed km/h » et « Précipitation » contiennent des valeurs manquantes et la variable « Events » contient beaucoup de modalités différentes et des NA. Notre pré-processing est le suivant :

> I – Pour les évènements (Events) : lorsque la valeur est manquante, nous supposons qu'il n'y a pas de temps « spécial », donc nous attribuons la modalité « Sunny ». Toutefois, si la valeur de la variable « Cloud Cover » est supérieure à 4 nous attribuons à la place « Cloudy » (Ceci retire l'ensemble des NA). Ensuite, nous avons regrouper dans une seule même modalité les catégories multiples en deux modalités : Snow ou Extrem.
>
> II – Pour la variable « Max Gust Speed », nous avons attribuer la valeur de la variable vitesse du vent lorsque qu'il n'y a pas de rafale de vent.
>
> III – Nous avons décidé de retirer la colonne « Précipitation ».

Toutefois, après un premier modèle, ces données ne suffisaient pas pour obtenir un score convenable (au maximum un RMSE de 0,450 avec un Random Forests). Nous avons donc ajouté de nouvelles données, que nous devions « merger » avec le fichier des données externes. Les données sont les suivantes :

2 – La distance : la distance en kilomètres entre l'aéroport d'arrivée et de départ est un facteur à prendre en compte car il représente la taille de l'avion selon s'il s'agit d'un vol long-courrier,

ou non. Pour cela, nous avons utilisé une base *Kaggle* pour récupérer la latitude et la longitude de chaque aéroport et une fonction *distance* pour la calculer. Pour éviter de les intégrer dans le jeu de données externes, nous avons créé un dictionnaire au sein de la fonction, prenant pour clé l'aéroport et en valeur la position géographique.

3 – L'ajout de la population de la ville : le nombre de vols entre deux aéroports dépend probablement du nombre de personnes présentes dans les villes respectives. À partir des données de Wikipédia provenant de Census.gov, nous avons ajouté au dictionnaire précédent la population de chaque aéroport (plus précisément la ville dont dépend l'aéroport), sur l'année 2010.

4 – Les jours fériés : Cette nouvelle variable montre si la date de départ de l'avion est un jour fériée aux USA, ou proche d'un jour. Pour cela on applique une fenêtre de deux jours (2 avant et 2 après afin de capturer cet effet). On suppose que autour de ces jours-là, le volume de passagers va être supérieur à la normale. Toutefois, sachant que les réservations peuvent se faire à l'avance, il aurait été possible d'utiliser une fenêtre plus large. La base est disponible ici.

5 – Trafics des aéroports : une variable à prendre en compte dans ce type de prédiction est l'historique du nombre de passagers (*Pour notre cas, nous avons utilisé ce nombre à la même date, mais dans un vrai enjeu métier nous n'aurions pas ces informations. Toutefois, nous pourrions par exemple à la place utiliser le trafic de l'année précédente, ce qui fonctionnerait aussi bien*). Cette démarche s'est faite en deux étapes :

- **Scraping des données** : nous avons collecter le trafic de chaque aéroport de manière mensuelle pour les arrivées et les départs, sur le site de l'aviation américaine (ici).
- **Merge sur external data** : nous avons sélectionner les dates utiles pour notre prédiction (2011 – 2013) et « merger » dans le fichier externe les informations pour les départs et ensuite les arrivées (deux colonnes supplémentaires dans notre jeu de données).

6 – Indices économiques et démographique : pour capter l'effet économique de la période, nous avons intégré le taux de chômage de chaque État sur ces mêmes années (ici), le « Transportation Service Index » (TSI) pour les passagers (ici), le pouvoir d'achat de la population à cette période (ici), le cours du pétrole (WTI) et une variable concernant la migration (ici). *Pour cette dernière variable, les données n'étaient pas disponibles aux dates du jeu de données. Nous avons donc utilisé des données actuelles en supposant que celles-ci ne fluctuent que légèrement. Il s'agissait ici de capturer le degré de migration d'un État*. Nous avons ensuite « merger » toutes ces données au fichier externe à partir du mois et de l'année en question.

3 – PRÉ-PROCESSING DES DONNEES

Pour ce projet, nous avons effectué toutes les étapes de prétraitement dans le cadre d'une *pipeline scikit-learn* qui enchaîne les étapes de transformation puis d'estimation. Nous allons créer des fonctions de préprocessing et utiliser **FunctionTransformer** pour la rendre compatible avec l'API *scikit-learn*. En effet, le retour du programme sur la plateforme RAMP doit être une *pipeline scikit-learn* (donc un objet qu'on peut *fit* et *transform*).

Transformeur : dates

La mise en colonnes numériques des dates est une opération courante lorsque les données des séries chronologiques sont analysées à l'aide de prédicteurs non paramétriques. On effectue les transformations suivantes : colonnes numériques pour l'année (2011-2012), le mois (1-12), la semaine de l'année (1-52) et le jour de la semaine (0-6).

Transformeur : calcul des distances et population

À partir du dictionnaire (position géographique et population des aéroports), on applique la fonction de calcul des distances entre les aéroports de départ et d'arrivé, pour chaque ligne (une variable). On fait de même avec la population en utilisant ce même dictionnaire qui contient la population de chacune des villes des aéroports de départ et d'arrivé (deux variables).

Transformeur : données externes

Ce transformeur permet de « merger » à notre jeu de données initial, toutes les variables contenues dans les données externes (météo, jours fériés, indices économiques et sociaux, trafic des aéroports). Lorsque nous possédions l'information concernant un aéroport de départ et d'arrivé, nous avons « merger » les données successivement pour chaque aéroport (par exemple, température de départ et d'arrivé, etc…). Nous avons ensuite « merger » les variables seulement indexées par le temps (Vacances, TSI, etc…).

Transformeur : one hot encoding

Sachant que certaines variables sont catégorielles (contiennent du texte, ou la valeur numérique ne correspond pas à quelque chose de quantifiable), il a fallu « binariser » nos variables (départ, arrivée, évent).

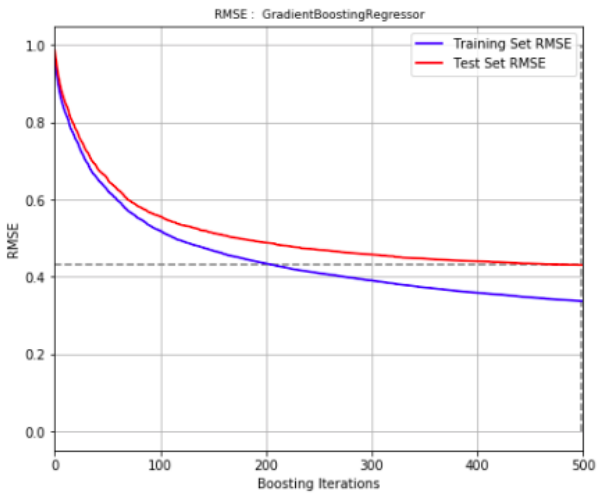
Transformeur : soustraction

La dernière étape de préprocessing a été le calcul des différences entre les variables où nous avions l'information au départ et à l'arrivée (lorsqu'elles étaient quantitatives). Ceci permettait de capter l'information concernant le voyage sur la variable en question.

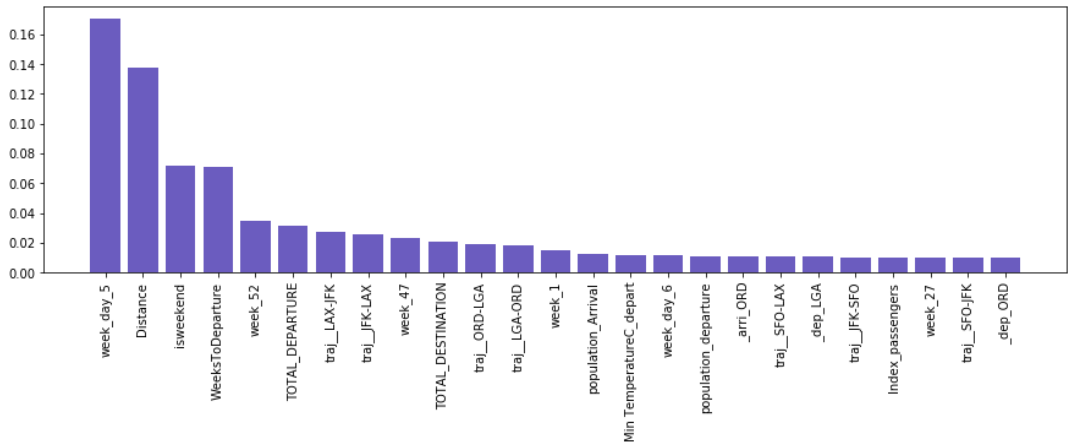
Il s'agit ici de notre **pipeline finale** en termes de préprocessing. Nous l'avons itérativement amélioré tout au long du projet lorsque nous ajoutions de nouvelles variables. Elle renvoie une matrice de données explicatives de dimension 8902 x 319. Maintenant, l'étape importante est : la sélection de variables.

4 – FEATURES SELECTION

Nous avons commencé par un premier modèle : **GradientBoostingRegressor** afin d'obtenir un aperçu de l'effet de l'ensemble des variables (RMSE de 0,4201).

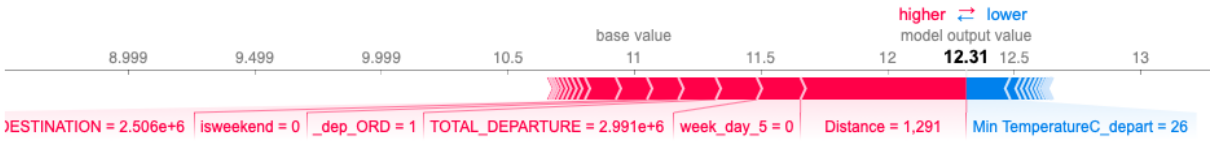


1 – Features importances : il est possible de récupérer les importances de chaque variable à la construction du modèle et d'en afficher les 25 premières :



On remarque que plusieurs de nos variables externes font partie des variables ayant le plus d'importance (Distance, Trafic, population, TSI). Après plusieurs modèles, nous ne retiendrons pas cette de sélection.

2 – SelectFromModel : l'API scikit-learn propose des fonctions de **Features sélection**, qui permettent à partir d'un estimateur (LGBM, mais nous pouvions également choisir un Lasso) de sélectionner les meilleures variables selon une métrique (la moyenne des importances). Le modèle conservait 144 variables.



Le graphique ci-dessus montre les caractéristiques qui contribuent chacune à faire passer la sortie du modèle de la valeur de base (la sortie moyenne du modèle sur l'ensemble des données de train que nous avons passées) à la sortie du modèle. Les caractéristiques qui poussent la prédiction vers le haut sont indiquées en rouge (Distance, le vendredi), celles qui poussent la prédiction vers le bas sont en bleu (la température). Il s'agit ici davantage une visualisation pour comprendre les variables du modèle.

3 – Corrélation statistique : Après avoir fait quelques tests sur nos modèles, nous nous sommes rendu compte que le nombre de variables restaient très conséquent. À partir des corrélations, nous avons conservé seulement un petit nombre de variables dans notre modèle : une seule variable météo, les variables économiques et sociaux, la distance et les jours fériés, les aéroports de départs et d'arrivés.

Les variables concernant les différences entre le départ et l'arrivé n'ont pas été utilisées pour la suite. Elles n'ont pas été intégrée car diminuait notre score et donc la complexité du modèle. On se retrouve finalement avec un modèle parcimonieux avec moins de 30 variables.

5 – MODEL SELECTION AND TUNNING

La phase finale du projet consiste à la recherche du meilleur modèle et des hyperparamètres associés. Les RMSE de chacun des modèles (hyperparamètres par défaut) sur notre jeu de données (par étapes) sont les suivants

Étape	Modèle	Échantillon d'entraînement	Échantillon de test
1	Random Forests Regressor	0.4565 +/- 0.0266	0.4616
	Extra Trees Regressor	0.4258 +/- 0.0182	0.4147
	Lasso	0.5808 +/- 0.0288	0.6057
2	Stacking (Lasso, Gradient Boosting Regressor, Random Forest et Extra Trees Regressor)	0.4677 +/- 0.0244	0.4747
3	XGBoost Regressor	0.3816 +/- 0.0177	0.3836
	LightGBM Regressor	**0.3541 +/- 0.0225**	**0.3475**

À partir de l'ensemble de nos résultats précédent sur nos **Features** sélectionnées, on peut conclure que le modèle **LightGBM** semble être le meilleur pour notre problématique. En résumé, il s'agit d'un modèle d'arbre optimisé qui se révèle performant autant en termes de score que de temps de calcul. Pour affiner notre score, nous avons utilliser la fonction GridSearchCV afin de trouver les hyperparamètres optimaux de ce modèle :

- Les paramètres du **« leaves »** et de **« child »** qui jouaient beaucoup sur notre score (respectivement 35 et 7) ;
- La paramètre **« estimators »** permettait d'augmenter le nombre d'arbres à notre modèle et donc augmenter le score également. Pour notre soumission finale, nous l'avons fixé à 2000 ;

6 – CONCLUSION

Pour conclure, ce projet nous a aidé à réaliser l'importance des différentes étapes d'un projet de machine learning dans un objectif de prédiction mais également d'interprétation d'un modèle (savoir quelles sont les variables discriminantes).

- Tout d'abord, bien définir le périmètre du projet afin de trouver des données externes adaptées et pertinente.
- Dans une seconde étape, le choix des variables par un processus itératif, où l'objectif est d'assurer une parcimonie au sein du modèle.
- Enfin, une dernière étape de « Tuning » permet une nette amélioration des métriques retenues mais toujours en laissant la possibilité de revenir à des étapes précédentes (ajout, retrait ou modification de certaines variables).