**Storefront : Inquiry Prediction Contest**

**Participants** : Suss Romain, Adnin Aurélien, Flaus Ambroise

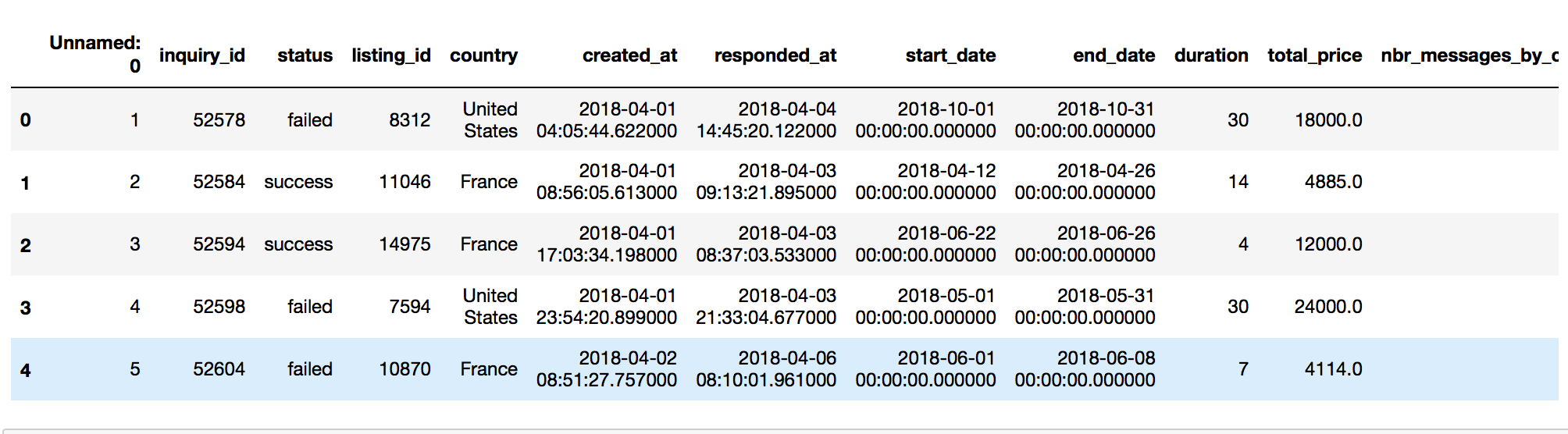
**Objectifs** : Prédire si une inquiry va success ou failed

**Dataset** : 7160 inquiry effectuées (dont le succès est connu) concernant les Etats-Unis et la France

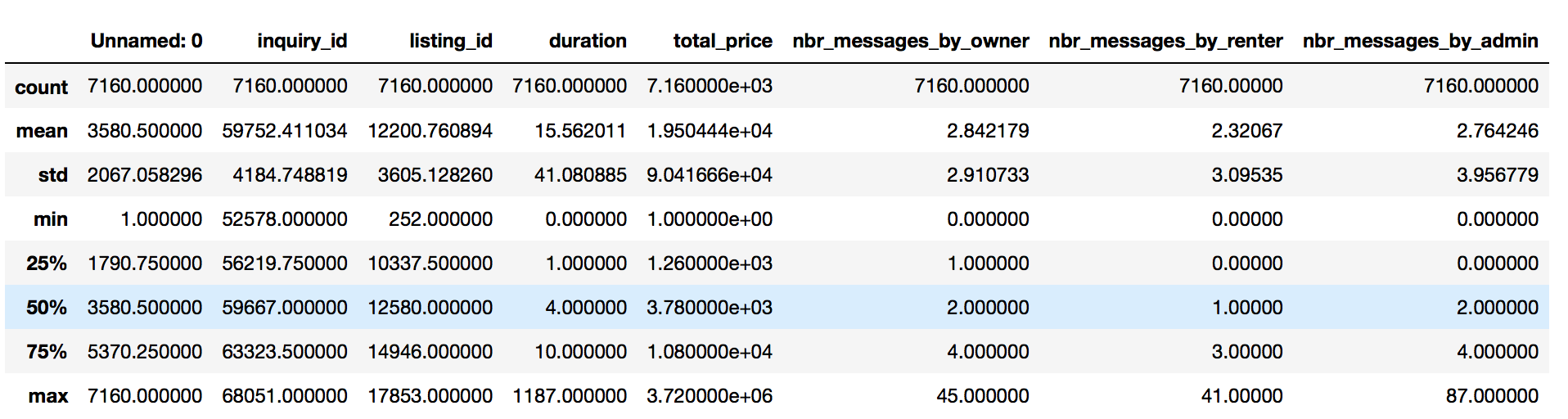
Partie 1 : Exploration des données

Dans cette partie, nous allons juste nous familiariser avec notre dataset. Il est important de bien savoir de quoi est composé notre dataset et de commencer à tirer quelques informations pour avoir une petite idée de la répartition de nos données.

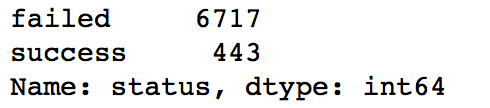
Premièrement, on regarde les 5 premières lignes de notre dataset. On peut voir les 13 colonnes et le type de valeurs qui les composent. Il y a des dates, des valeurs numériques et des chaines de caractères. On remarque aussi rapidement qu’il y a des colonnes qui servent d’id. Ces colonnes ne seront pas à prendre en compte lors de la modélisation.



Ensuite on regarde ce que contiennent les colonnes avec des valeurs numériques avec des éléments statistiques basique pour se faire une première idée des valeurs et de l’importance de la variable.

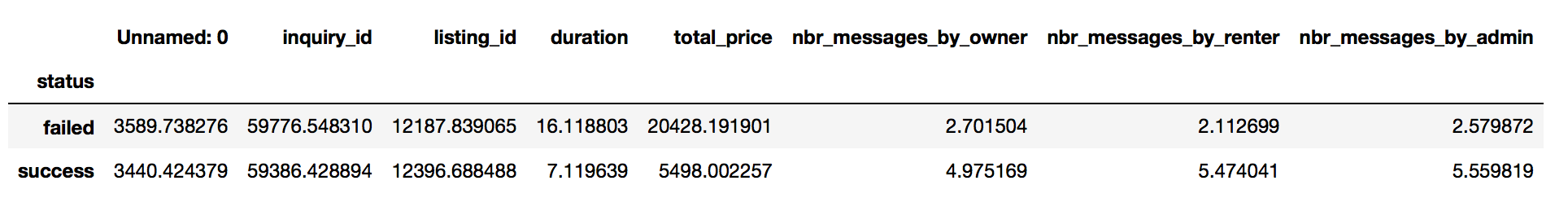


Puis on regarde la répartition des données que l’on veut prédire.



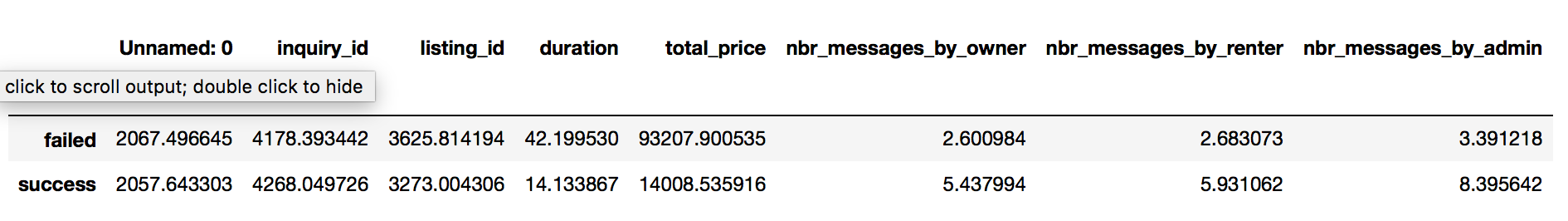
Il y a 6.19% des inquiry qui vont success. Donc, un algorithme qui va prédire failed à chaque fois aura bon à 93,81%. Il va donc falloir faire attention à cela par la suite.

Enfin, on peut regarder la moyenne des données suivant si l’inquiry est success ou failed.



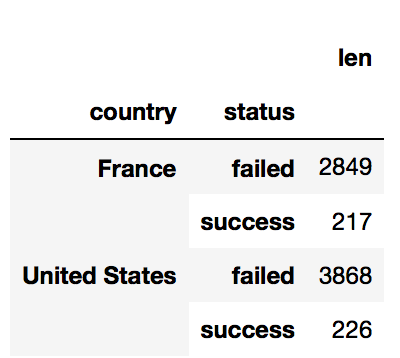
On remarque que les variables duration et total price sont bien inférieures en cas de success. Donc les inquiry ont tendance à success si la durée de location est courte (environ une semaine) et si le prix est peu élevé (5500 $/euros). On remarque également que plus le nombre de messages échangés entre les trois parties est élevé, plus on a de chance de success.

Enfin, on peut également regarder la déviation standard des données suivant si l’inquiry est success ou failed.



Cela confirme les hypothèses émises précédemment.

Regardons plus particulièrement la variable pays :



Pour la France, on a donc 7% des inquiry qui success et pour les Etats-Unis 5%. Cette variable ne va donc pas avoir un poids important dans notre modèle.

Ensuite, nous avons les variables created\_at : date où l’inquiry a été créée par le locataire et responded\_at: date où le propriétaire a répondu pour la première fois sur la conversation. Il ne nous semble pas intéressant de regarder ces deux variables séparément. Cependant, on peut calculer combien de temps le propriétaire a mis pour répondre. Cela est une idée de nouvelle variable pour la suite.

On peut aussi déjà dire que les variables qui représentent les id sont inutiles. De plus les variable start\_date et end\_date nous semble utile uniquement lorsque l’on fait la différence entre les deux. Cette variable existe déjà : c’est la variable duration, on peut donc retirer les deux autres.

Nous avons donc une meilleure idée de la composition de notre dataset. On a donc plus 7 variables qui peuvent expliquer la variable status (qui nous dit si on a success ou failed). Et on a déjà émis l’hypothèse que les variables duration et total\_price sont importantes et qu’il va surement falloir transformer les variables created\_at et responded\_at.

Partie 2 : Un premier modèle

Nous avons choisi pour ce premier modèle d’utiliser un algorithme de régression logistique. C’est un algorithme d’apprentissage supervisé qui permet la classification. La régression logistique est un algorithme très répandu, offrant pouvoir explicatif très fort du fait de sa linéarité.

De plus, nous n’avons pas transformé les deux variables created\_at et responded\_at. Nous verrons dans la dernière partie quelles sont les next steps pour améliorer notre modèle. Nous avons alors 5 variables.

Dans notre cas, le jeu de données n’est pas très grand. Pour augmenter la significativité de la validation, nous utiliserons la moyenne de plusieurs validations croisées, disons 8. Nous testons cela avec notre algorithme et nous obtenons :

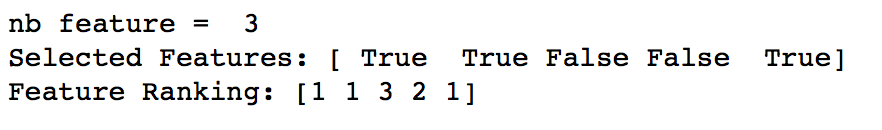


Comme nous utilisons un modèle linéaire, il est intéressant d’observer le poids que la régression logistique accorde à chaque variable :

L’interprétation est simple : un poids positif augmente la probabilité, un poids négatif la diminue et un poids proche de zéro signifie qu’il est très peu discriminant par rapport à la variable à prédire.

Ici on remarque que les variables 1,2 et 5 sont intéressantes pour notre problème, et qu’au contraire les variables 3 et 4 sont peu discriminantes et n’apporte pas grand-chose.

Pour confirmer cela il existe d’autres outils pour nous aider dans la sélection de variable. Avec le RFE on obtient :



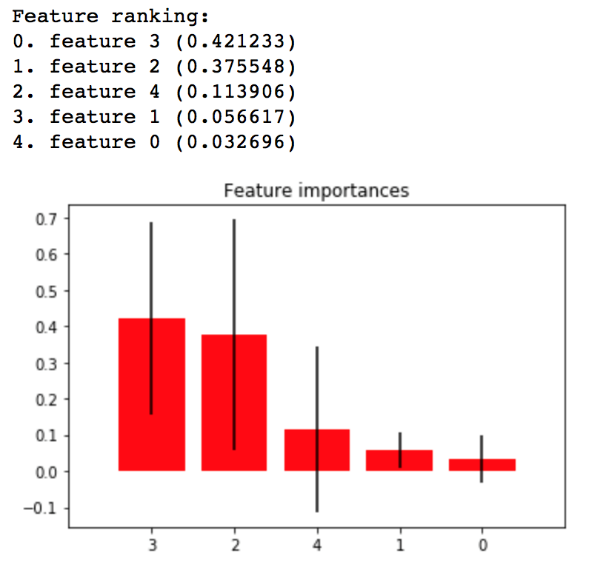
Cela nous confirme bien les variables intéressantes.

Partie 3 : Un modèle plus performant

Nous avons décidé d’utiliser deux autres algorithmes pour notre modélisation. D’abord le random forest, c’est un algorithme passe-partout que l’on trouve naturellement parmi les solutions des top-Kagglers. Il est rapide à entrainer, car parallélisable et robuste. De plus il est beaucoup plus intuitif à comprendre que d’autres algorithmes non linéaires.

Tout comme pour la régression logistique nous utilisons la cross validation et nous obtenons : 

Le random forest nous permet de savoir quelle importance il donne à chaque variables. Nous pouvons après entrainement du modèle obtenir ceci :



Cela nous permet donc de voir les variables qui peuvent être importantes.

Nous avons aussi voulu tester l’algorithme de gradient boosting. Il correspond à une forme de régression (et classification) non linéaire extrêmement performante. Le gradient boosting, en particulier, partage avec le random forest le gène commun d’être une méthode ensembliste, qui se base sur des arbres de décisions comme méthode sous-jacente à l’ensemble.

Tout comme les algorithmes précédents nous utilisons la cross validation et nous obtenons : 

Partie 4 : L’overfitting et le compromis biais variance

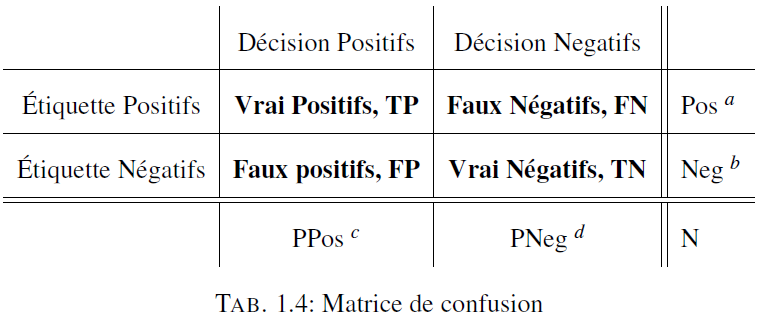
Le surapprentissage est la bête noire dans tout problème de data science. Dans la réalité, la distinction entre un bon modèle et un modèle overfitté n’est pas toujours évidente. Pour qualifier le surapprentissage, on s’appuie souvent sur ce que l’on nomme le compromis biais variance. Ce compromis va permettre de trouver un juste équilibre entre la complexité du modèle et sa capacité à se généraliser.

Ici nous n’allons pas beaucoup changer les paramètres de base des algorithmes. Nous perdrons certes de la performance mais nous risquerons moins de faire du surapprentissage.

Partie 5 : Reporting de performance

1) Matrice de confusion

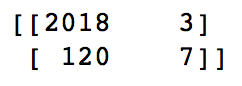
Pour les problèmes de classification, il existe différentes métriques de performance.

L’évaluation d’un problème de classification se base sur une matrice de confusion, qui met en regard des données prédites et des données observées. Une matrice de confusion représente ceci : 

Cependant, c’est certainement la courbe ROC qui est la méthode d’évaluation des problèmes de classification la plus utilisée.

**La régression logistique**

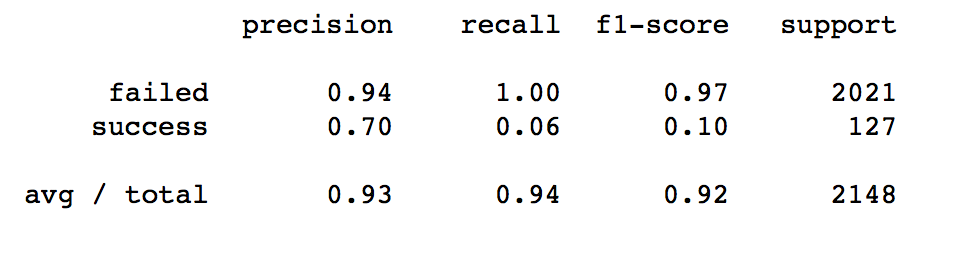
Pour la régression logistique, nous avons la matrice de confusion suivante :



De nombreuses mesures peuvent être tirées à partir de cette matrice, afin de décrire plus généralement le comportement du modèle par rapport aux données réelles.

On peut calculer le recall qui est le taux de vrais positifs et on peut aussi calculer la précision. Il en existe d’autres mais ces deux là permettent de se faire une bonne idée générale de la qualité du modèle. Le rappel permet de mesurer la proportion de positifs prédits parmi tous les positifs de la population. La précision permet de mesurer la proportion de positifs de la population parmi tous les positifs prédits. Ainsi un modèle parfait aura un recall égale à 1 (il prédit la totalité des positifs) et une précision égale à 1 (il ne fait aucune erreur, tous les positifs prédits sont des vrais positifs). De plus, il existe un autre indicateur qui est agrégé. Il est composé à partir du recall et de la précision : le f1-score. On calcule pour cela la moyenne harmonique de la précision et du recall.

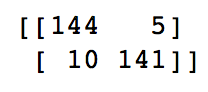
Pour la régression logistique nous obtenons :



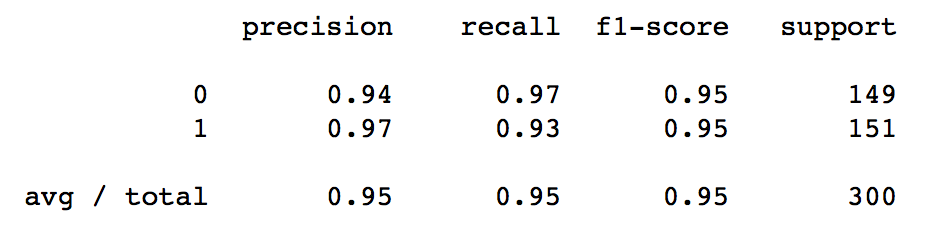
On remarque que notre modèle est très mauvais pour prédire les success, mais qu’au final il obtient quand même un bon score. Comme nous l’avons dit précédemment nous avons très peu de success (6.19% des inquiry) et si l’on utilise un modèle qui prédit uniquement failed on aura un très bon score. Pourtant ce qui nous intéresse le plus c’est de prédire quand une inquiry va success. Donc malgré un bon score ce modèle est plutôt mauvais. Ici nous sommes un peu dans le même cas avec un modèle performant mais qui est mauvais sur les prévisions de success. La régression logistique n’a donc pas de très bonne performance pour nous.

**Gradient boosting**

Pour le gradient boosting, nous avons la matrice de confusion suivante :



Tout comme pour la régression logistique nous pouvons effectuer les mêmes calculs à savoir le recall, la précision et le f1-score. Nous obtenons :



Ici, nous remarquons que nous sommes à chaque fois proche de 1. Nous avons donc un modèle performant qui va bien prédire les success.

2) Courbe ROC

Nous l’avons évoqué plus haut : la classification repose sur un arbitrage entre rappel et précision. Cet arbitrage se base sur le choix d’un seuil de décision qui va favoriser l’un ou l’autre. La matrice de confusion dépend donc de la valeur de seuil s. La courbe ROC va permettre de systématiser l’analyse des résultats d’un classifieur, en fournissant une vue synthétique de sa performance pour toutes les valeurs de s possibles.

Pour construire une courbe ROC, deux indicateurs de performances sont requis :

* La sensibilité a : le taux de vrais positifs
* La spécificité b : le taux de vrais négatifs

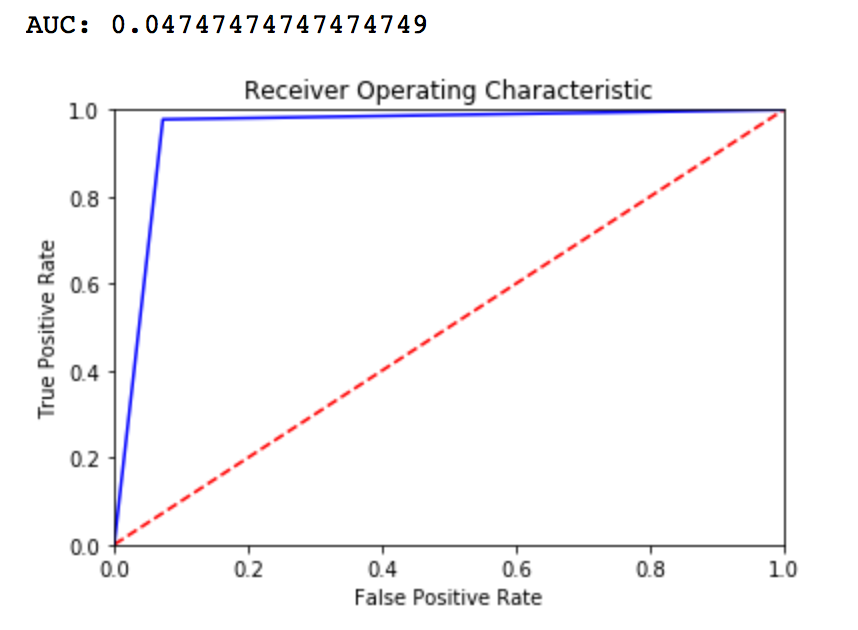
La courbe ROC est alors tracée dans un espace à deux dimensions avec a en ordonnée et 1-b en abscisse : cela revient à tracer le taux de vrais positifs en fonction du taux de faux positifs. Ces valeurs sont tracées pour les différentes valeurs d’un seuil de décision s.

L’analyse de la courbe va permettre de choisir le seuil de décision optimal. Ce choix peut se faire analytiquement, mais on peut tout aussi bien appliquer une règle empirique simple : le seuil optimal est celui qui est au point le plus proche de l’idéal (1,1) et au plus loin de la diagonale.

Le grand attrait de la courbe ROC est qu’elle offre un cadre commun qui permet de comparer des modèles de différents types. Cette comparaison peut se faire localement (pour certains seuil de décisions donnés), ou globalement, quel que soit le seuil de décision. Pour ce dernier cas, on considère la surface sous la courbe ROC : l’AUC.

La comparaison est d’une facilité déconcertante : plus grande est l’AUC, meilleur est le modèle. Cette valeur s’interprète en effet comme la probabilité de classer un exemple positif choisi au hasard comme positif.

Prenons par exemple notre modèle construit avec le gradient boosting, on obtient :



Partie 6 : Difficultés/Next steps

La difficulté pour résoudre ce genre de problèmes réside dans le type de dataset. Nous avions peu de données et la distribution des classes à prédire était très asymétrique.

Différentes méthodes que nous pouvons regrouper en deux catégories principales sont souvent utilisées pour traiter le déséquilibre des classes en apprentissage supervisé :

- Des stratégies d’échantillonnage visant à rééquilibrer les données. Ces méthodes pouvant par ailleurs être utilisées avec n’importe quelle méthode d’apprentissage supervisé.

- La prise en compte explicite de l’asymétrie des classes au sein même de l’algorithme généralement sous la forme de matrice de coûts (ou des objectifs précis spécifiés par l’utilisateur). Des approches ensemble permettent de rendre n’importe quel type d’algorithme sensible à l’asymétrie, notamment par des méthodes de boosting (nous avons fait le gradient boosting) ou du bagging.

Ensuite nous n’avons pas fait de data engineering (nous avons estimé que ce n’était pas le but ici), nous pouvons améliorer notre dataset grâce à cela. Comme évoqué plus haut, certaines variables méritent d’être traitées car cela peut nous apporter de l’information supplémentaire. C’est une des next steps possibles.

Une autre step est de rajouter des données manquantes. Cela n’est pas toujours évident d’aller chercher de la data mais notre modèle peut devenir plus pertinent. Par exemple nous avons pensé au type d’entreprise qui fait l’inquiry, ou le type de personne, son poids économique etc. Aussi trouver des variables sur le type de bien qui est loué, sa taille, ses particularités. Ou encore pour quel type d’événement la demande est faite. Il y a donc pas mal d’idées intéressantes que l’on peut creuser à ce niveau.

Enfin nous pouvons travailler sur nos algorithmes. Il existe beaucoup de paramètres que l’on peut modifier pour que la modélisation soit plus performante. Toujours en faisant bien attention à des chose comme l’overfitting.

En tout cas cela était très intéressant de travailler sur ce mini projet. Nous nous rendons bien compte de toute les ouvertures possibles ce qui le rend encore plus excitant. Nous espérons donc pouvoir travailler avec vous, pouvoir discuter de tout cela et trouver les meilleures solutions possibles.