Test Quantmetry

Nicolas Agnus-Villette

December 18, 2018

Abstract

Ce rapport présente les réponses aux questions de l'énoncé. L'ensemble du code est joint à ce document sous forme de Python Notebook où sont présentées les descriptions détaillées de l'ensemble du code est des résultats ainsi que l'analyse des graphiques.

1 Statistiques descriptives

1.1 Description du jeu de données

Nous commençons l'analyse par un travail de Data Preprocessing. Nous supprimons immédiatement la variable index qui est redondante et la variable date qui ne sera pas traitée pour la conception du modèle. Nous pouvons immédiatement voir que le dataset contient un mélange de données quantitatives et qualitatives.

1.2 Missing Data

Nous commençons l'analyse par le traitement des Missing Data. La commande `data.isnull().values.any()` renvoie `True` ainsi nous savons que le dataset possède des `NaN`. `data.isnull().sum()` affiche un tableau récapitulatif de la somme de ces missing values.

Plusieurs méthodes de traitement des missing values sont envisageables mais dans ce cas, leurs proportions nous permet de simplement supprimer les `rows` ou il figure au moins une de ces `NaN` quelque soit la variable grâce à la commande `data.dropna(inplace=True)`. `data.dropna(inplace=True)` permet d'effectuer cette opération et `data.isnull().sum()` confirme son succès par un compte des valeurs égale à 0 pour chaque variable.

1.3 Data Distribution

Nous pouvons analyser la répartition de la classe `embauche` et nous voyons qu'il y a un nombre significativement plus élevé de candidatures qui n'ont pas abouties.

L'analyse des graphes de densité des variables numériques nous démontre qu'elles suivent toutes une distribution normale.

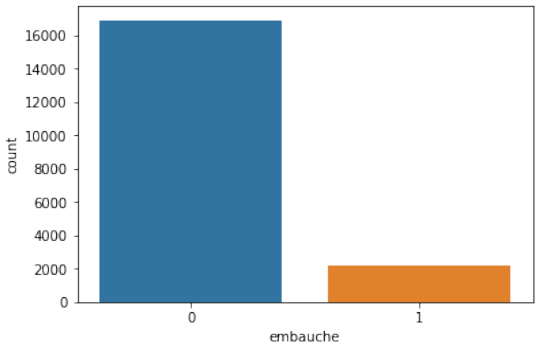

Figure 1: Data Distribution

1.4 Outliers

Un affichage de `data.describe()` nous permet d'afficher les résultats des principaux tests statistiques pour nos variables. En comparant les valeurs `Min` et `Max`, nous pouvons immédiatement conclure sur la présence d'outliers. Ce résultat peut être confirmé par la visualisation des `Box Plots` de nos valeurs numériques.

Nous supprimons une première partie des outliers en définissant des règles fixant des limites de range pour chaque variable. La note est donc restreinte entre 0 et 100, pour l'âge, entre 17 et 62 car il parait bien plus probable de postuler pour un emploi dans ces limites. L'expérience est réduite à $[0; \infty]$ Nous utilisons ensuite les limites des `Box plots` pour établir quelles données seront gardées.

1.5 Variables à Sélectionner

D'après les analyses graphiques des `Box plots`, aucune corrélation n'est évidente entre les valeurs numériques et la classe. En effet, l'ensemble des médianes sont quasiment égales pour chaque catégorie. Les analyses de corrélations entre valeurs numériques nous permettront en revanche de supprimer certaines variables en prévision de la création de notre modèle.

En ce qui concerne les valeurs qualitative. Nous voyons une majorité de spécialité en géologie étant embauché et il semblerait que le diplôme ait un impact également car nous avons une distribution quasi identique de candidatures qui ont abouties pour des détenteurs de licence ou master, bien devant les 2 autres catégories de la variable.

1.6 Dépendance Statistiquement Significative

1.6.1 Spécialité & Sexe

Nous analysons la dépendance de la spécialité et du sexe qui sont deux variables nominales par un test du χ^2. Le résultat et la p-value qui est égale à 0 nous permettent de conclure sur la dépendance de ces deux variables.

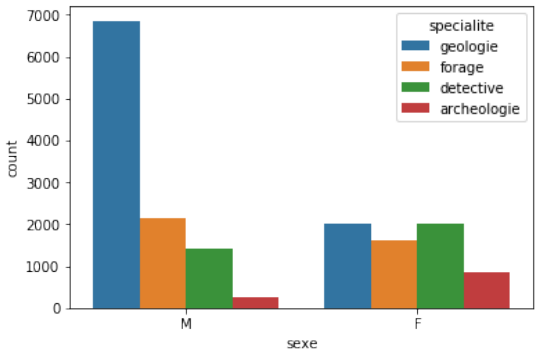

Figure 2: Spécialité en fonction du sexe

1.6.2 Couleur de cheveux & Salaire demandé

Il s'agit de l'analyse d'une variable nominale et d'une variable numérique. L'affichage d'un `Box Plot` du salaire en fonction des catégories de cheveux nous permet d'analyser les médianes de chaque `Box` et nous pouvons conclure sur l'indépendance de ces deux variables.

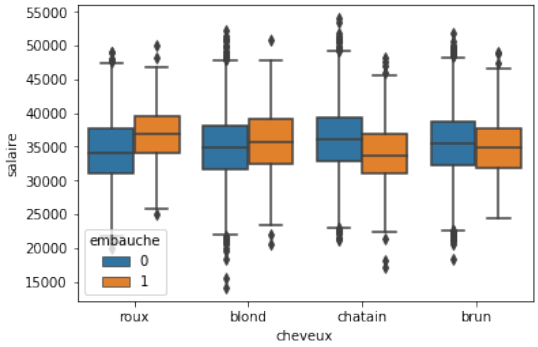

Figure 3: Cheveux en fonction du salaire

1.6.3 Expérience & Note

L'analyse graphique du nuage de points laisse présager qu'il n'y a qu'une faible corrélation entre le salaire et la note à l'exercice. En effet, nous voyons une relation linéaire avec corrélation négative car le salaire diminue en fonction de la note.

Une corrélation de Pearson nous permet de quantifier cette éventuelle dépendance grâce à la commande suivante.

```
stats.pearsonr(data['note'], data['salaire'])
(-0.40330886400499638, 0.0)
```

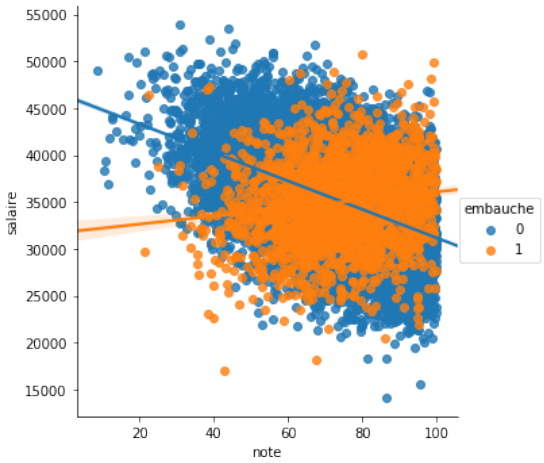

Figure 4: Expérience en fonction de la note

Nous avons confirmation d'une corrélation négative modérée de -0.40 et une p-value de 0.

2 Machine Learning

2.1 Data Preprocessing

Afin de pouvoir travailler sur nos modèles de Machine Learning. Nous effectuons plusieurs étapes de Data Preprocessing. Nous supprimons les variables colinéaires en gardant celles qui sont les plus corrélées avec la classe embauche.

Pour les variables catégoriques, nous effectuons une opération de `One Hot Encoding` afin de créer des nouvelles variables correspondant à chaque catégorie. Cette opération permettra à l'ensemble des modèles de travailler avec nos données.

Nous séparons la donnée en `Training Set` et `Test Set` avec un rapport de 70 : 30. Nous normalisons la données afin qu'elle puisse être traitée par nos modèles. Les valeurs sont des unités et range différents donc nous la reportons dans le range $[0, 1]$ grâce à `MinMaxScaler()`. Les modèles comme de régression linéaire ou Random Forest ne nécessite pas cette étape mais nous allons comparer plusieurs modèles pour trouver le plus optimal donc cela reste une étape obligatoire dans notre cas.

2.2 Comparaison des Modèles

Afin de déterminer quel serait le meilleur modèle à utiliser pour notre tâche, nous allons comparer les valeurs de `Auc` (Area Under Curve) pour la mesure `Receiver Operating Characteristic (ROC)` De cette évaluation, il parait que le `Gradient Boosting Classifier` donne les meilleurs résultats. ce qui est également confirmé par la comparaison des mesures d'`Accuracy`.

2.3 Optimisation Paramétrique

Une fois le modèle défini, nous effectuons une étape de `Hyperparameter Tuning` afin d'optimiser les performances de notre modèle. Nous comparons ensuite ce modèle optimisé avec notre `Baseline`. Nous choisissons donc de chercher la meilleure combination possible pour les paramètres suivant:

```
n_estimators = [1, 2, 4, 8, 16, 32, 64, 100, 200, 300]
max_depth = [2, 3, 5, 10, 15]
min_samples_leaf = [1, 2, 4, 6, 8]
min_samples_split = [2, 4, 6, 10]
max_features = [1, 2, 3, 4, 5, 6]
```

`randomizedsearch_cv` nous donne une première estimation des paramètres donnant les meilleurs résultats en utilisant la mesure `AUC`. Nous prenons donc ces valeurs pour effectuer la même étapes en réduisant le range de recherche autour des ces valeurs et obtenons les paramètres suivants:

```
{'max_depth': 5,
 'max_features': 3,
 'min_samples_leaf': 2,
 'min_samples_split': 3,
 'n_estimators': 150}
```

2.4 Importance des Variables

Grâce à ce nouveau modèle optimisé, nous pouvons récupérer la liste d'importance des variables. Nous pouvons ainsi la convertir en `panda Dataframe` puis nous la visualisons grâce à un `barplot` ordonné selon leur importance respective. Nous pouvons ainsi voir que pour notre modèle de `Gradient Boosting`. La `note`, l'`age` et l'`exp`, sont les 3 variables qui ont été les plus déterminantes. Nous remarquons également que le poids de la `note` est nettement plus conséquent que les autres. Ce résultat est étonnant car ce sont des corrélations qui n'étaient pas mise en évidence lors des analyses graphiques.

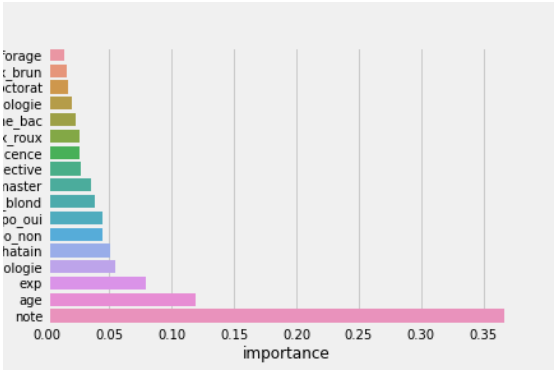

Figure 5: Feature Importance

2.5 Critère de Performance

2.5.1 Receiver Operator Characteristic(ROC)

Nous sommes en présence d'une tâche de classification binaire ou la variable `embauche` peut prendre la valeur 0 ou 1 avec une forte `class imbalance`. L'analyse des
`Receiver Operator Characteristic(ROC)` curves se sont révélées être particulièrement adaptées pour l'évaluation dans ce contexte.

Il s'agit de la courbe du ratio de `True Positive` en fonction des
`False Negative`. Une analyse graphique de la courbe permet d'avoir une vison concrète des performances du système mais ici, nous utiliserons la valeur d'`Aera Under Curve (AUC)`. Plus cette valeur sera proche de 1, plus la courbe suivra une allure supposée parfaite.

2.5.2 F1-Score

En complément, nous évaluons le `f1-score` qui est une mesure adaptée à une classification avec une distribution de la classe majoritaire négative. Cette mesure sera directement liée à la performance du modèle en considérant un rapport entre la précision et le recall.

2.6 Modèle avec Importance des Variables

Grâce aux résultats de l'importance des variables, nous pouvons évaluer un nouveau modèle en ne conservant que nos variables les plus significatives. Nous n'observons pas une amélioration significative des performances.

```
accuracy: 0.8908
mae: 0.1092
auc: 0.6078
f1: 0.3429
```

2.7 Pistes d'Amélioration

Stacking Afin d'améliorer notre modèle, nous pourrions effectuer une étape de `Stacking` ou nous procéderions à une évaluation en 2 étapes. La première consisterait en une évaluation classique de notre classification avec différents modèles puis dans un second temps, nous utiliserions les prédictions obtenues et non les variables comme donnée pour notre second modèle.

Principal component analysis (PCA) Pour notre étape de
`Feature Selection`, nous pourrions effectuer une
`Principal Component Analysis (PCA)` afin de réduire la dimension de l'espace en préservant la variance. On peut ainsi analyser les vecteurs représentant les axes principaux de notre donnée.

Hyperparameter Tuning Nous pourrions améliorer notre étape de `Tuning` en ajoutant des étapes de sélection, en ajoutant des range à évaluer plus précis et en croisant les résultats obetnues pour chaque paramètre.