**TP 2 – Jeu de données SALARY**

*Rudy LOMBARD*

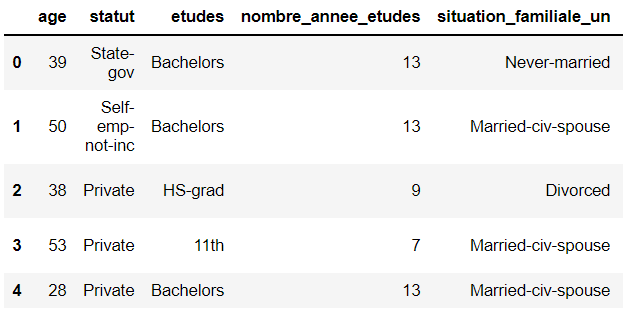
*Nono Armel TCHIASSO*

*Wissam GORCHANE*

1. L’objectif est de prédire le salaire S d’un individu. Les deux labels sont par rapport à ce salaire : inférieur ou suppérieur à 50K. Nous avons ensuite 14 features notées de F1 à F14. En explorant les données du dataset, on remarque que certaines sont quantitatives et d’autres sont qualitatives. On a donc transformé judicieusement les variables qualitatives en variables quantitatives.

Il fallait identifier les features qualitatives ordinales et nominales :

* Ordinales : features « études »
  + OrdinaleEncoder appliqué à cette colonne du dataset
    - Utilisation de OrdinaleEncoder() venant de la librairie sklearn
* Nominales : toutes les autres features
  + OneHotEncoder appliqué à ces colonnes du dataset
    - Utilisation de la fonction get\_dummies()

 Une image contenant table

Description générée automatiquement

A gauche : dataset avant transformation ; A droite : dataset après transformation

En affichant le nouveau dataset avec toutes les variables quantitatives, on se retrouve avec 92 colonnes avec l’ajout de colonnes grâce à la fonction get\_dummies(). Pour la feature « études », un ordre a été donné pour chaque valeur quantitative donc pas de nouvelles colonnes crées dans ce cas.

Avant d’appliquer nos différents classifieurs à ce jeu de données, nous devons normaliser l’ensemble de nos données à l’aide de StandardScaler(). Ce prétraitement est très important car il permet de redimensionner les variables numériques pour qu’elles soient comparables sur une échelle commune.

Ensuite, on sépare nos données en base train et en base test avec les données normalisées.

Enfin, on va entraîner les différents classifieurs avec les hyperparamètres suivants :

# **Entrainement des classifieurs**

La recherche des paramètres optimaux pour nos tous nos classifieurs se fait à l’aide de la fonction GridSearchCV qui test les différents paramètres entrés et nous renvoie les meilleurs paramètres en fonction des performances de chaque classifieur.

## **Multi Layer Perceptron Classifier**

Parmi plusieurs paramètres (hidden\_layer\_size & fonction d’activation) GridSearchCV nous renvoie comme paramètres optimaux :

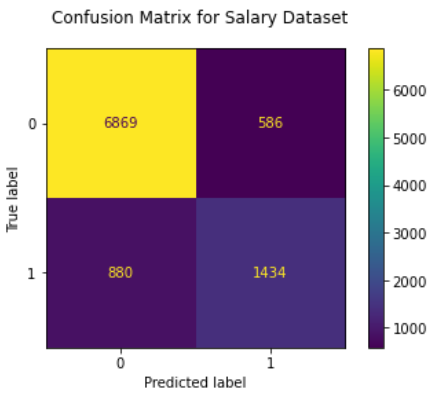


A partir de cela, on note en paramètres optimaux :

* 2 couches cachées
* 10 neurones par couche
* Nous avons en entrées du model 22792 exemples
* Poids du réseau : soit 228030 poids dans notre réseau
* Biais du réseau : soit 110 biais



*De haut en bas les résultats sur la base de validation, d’entrainement et de test*



*Matrice de confusion pour le MLP Classifier*

On observe une bonne reconnaissance sur les salaires inférieurs à 50K (environ 92%) mais un assez faible taux de reconnaissance sur les salaires supérieurs à 50K (environ 62%) cela pourra s’expliquer par la faible présence de salaires supérieurs à 50K dans notre base d’apprentissage.

Une image contenant texte

Description générée automatiquement

*Résultats MLP*

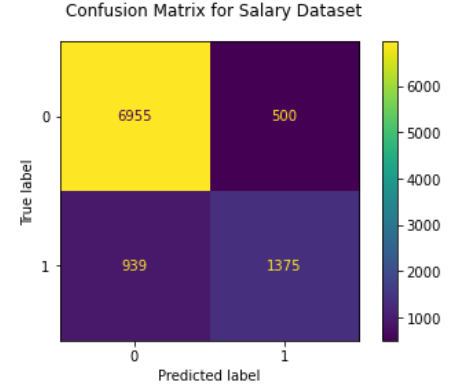
1. Pour chaque entrainement de classifieur, on va split nos données en base d’apprentissage en utilisant la cross-validation. Elle va permettre de tester la fiabilité du modèle. Le processus consiste à écarter en amont une partie des données du dataset d’entrainement. Ces données ne seront pas utilisées pour entraîner le modèle, mais plus tard pour tester et valider le modèle.

Ci-dessous les différents résultats des classifieurs avec la matrice de confusion associée :

SVM linéaire :

Une image contenant texte

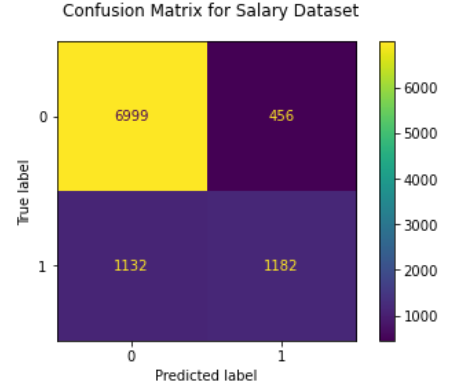
Description générée automatiquement



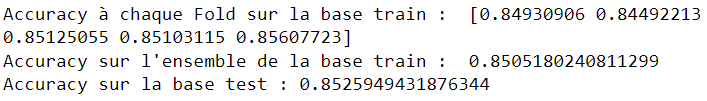
SVM non linéaire :

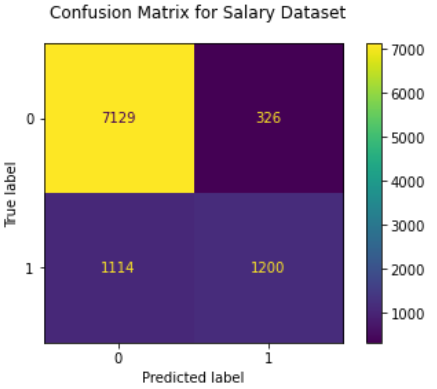
Une image contenant texte

Description générée automatiquement



Arbre de décision :





1. Encore une fois, le but est de prédire une sortie binaire (soit un salaire de moins de 50K, soit de plus) et nous voulons identifier une relation entre nos sorties et les features indépendantes. C’est un scénario classique de classification.

Ici, le SVM linéaire est le plus adapté pour cette tâche demandée car :

* On prédit une catégorie
* On a des données labélisées
* Nos échantillons ne dépassent pas 100k

De plus, on remarque que le SVM linéaire a de meilleures performances en base train et test avec une accuracy de 0.85 pour les deux bases.