# TP 2 – Jeu de données SALARY

*Rudy LOMBARD*
*Nono Armel TCHIASSO*
*Wissam GORCHANE*

I. L'objectif est de prédire le salaire S d'un individu. Le jeu de données est composé de 32560 observations. Nous disposons de deux classes : 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 faut 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()

| | age | statut | etudes | nombre_annee_etudes | situation_familiale_un |
|---|---|---|---|---|---|
| 0 | 39 | State-gov | Bachelors | 13 | Never-married |
| 1 | 50 | Self-emp-not-inc | Bachelors | 13 | Married-civ-spouse |
| 2 | 38 | Private | HS-grad | 9 | Divorced |
| 3 | 53 | Private | 11th | 7 | Married-civ-spouse |
| 4 | 28 | Private | Bachelors | 13 | Married-civ-spouse |

| | age | etudes | nombre_annee_etudes | plus_values | moins_value |
|---|---|---|---|---|---|
| 0 | 39 | 9.0 | 13 | 2174 | 0 |
| 1 | 50 | 9.0 | 13 | 0 | 0 |
| 2 | 38 | 11.0 | 9 | 0 | 0 |
| 3 | 53 | 1.0 | 7 | 0 | 0 |
| 4 | 28 | 9.0 | 13 | 0 | 0 |

5 rows × 92 columns

*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 effectuer le one-hot-encoding. 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.

## I. 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. 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.

## A. Multi Layer Perceptron Classifier

Parmi plusieurs paramètres (hidden_layer_size & fonction d'activation) GridSearchCV nous renvoie comme paramètres optimaux :

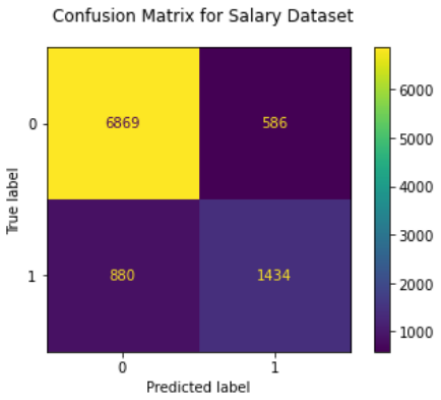
```
MLPClassifier(activation='logistic', hidden_layer_sizes=(10, 2))
```

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 : $(22792 \times 10) + (10 \times 10) + 10$ soit 228030 poids dans notre réseau
- Biais du réseau : $10 \times 10 + 10$ soit 110 biais

```
Accuracy à chaque Fold sur la base train :  [0.85216056 0.85325729 0.85125055 0.84795963 0.85015358]
Accuracy sur l'ensemble de la base train :  0.8528429702479153
Accuracy sur la base test :  0.8509418509418509
```

*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.

| | MLP Classifier |
|---|---|
| % Accuracy en Test | 85 |
| % Accuracy en Train | 85 |
| Couche cachées | 2 |
| Neuronnes par couches cachées | 10 |
| Nombre de biais | 110 |
| Nombre de poids | 228030 |

*Résultats MLP*

## B. Decision Tree Classifier

Parmi plusieurs paramètres (criterion, max_depth, min_sample_split) GridSearchCV nous renvoie comme paramètres optimaux :

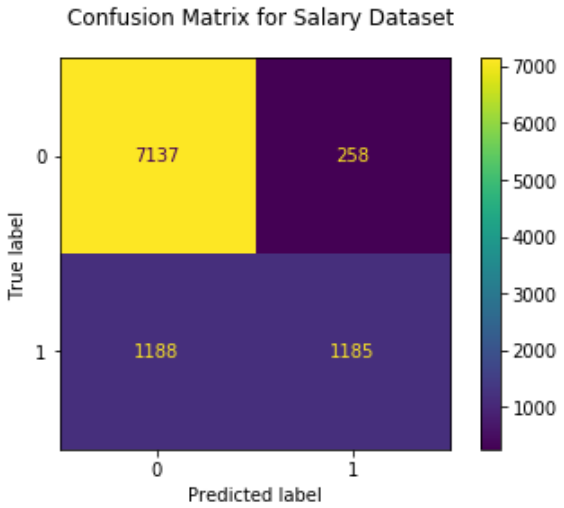
```
DecisionTreeClassifier(criterion='entropy', max_depth=9, min_samples_split=16)
```

A partir de cela, on note en paramètres optimaux :

- Un critère d'entropie
- Une profondeur maximale de 9
- Une séparation de données minimale de 16 sur le classifier

```
Accuracy à chaque Fold sur la base train :  [0.8561088  0.86225049 0.85256692 0.85344449 0.85432207]
Accuracy sur l'ensemble de la base train :  0.8555630382058219
Accuracy sur la base test :  0.851965601965602
```

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




*Matrice de confusion pour le Decision Tree Classifier*

On observe ici, comme pour le MLP de bonnes prédictions en ce qui concerne les salaires inférieurs à 50K (96% de reconnaissance en base test), cependant, le taux de reconnaissance pour les salaires supérieurs à 50K sont moins bons que le MLP : 0.49% de taux de reconnaissance sur la base de test.

| | Decision Tree Classifier |
|---|---|
| % Accuracy en Test | 85 |
| % Accuracy en Train | 85 |
| max_depth | 9 |
| min_samples_split | 16 |

*Résultats Decision Tree Classifier*

## C. Linear SVC

Parmi le paramètre de la souplesse de la marge C, GridSearchCV nous renvoie comme paramètre optimal :

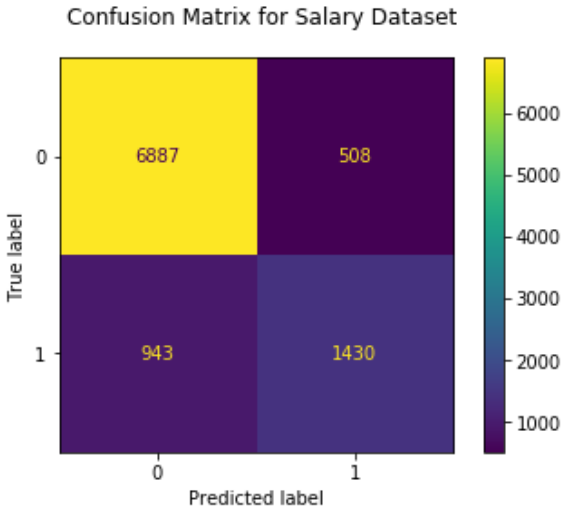
```
LinearSVC(C=0.1)
```

Notre paramètre optimal ici est une marge souple :

- C = 0.1

```
Accuracy à chaque Fold sur la base train :  [0.84974775 0.85457337 0.84861781 0.85278631 0.849276  ]
Accuracy sur l'ensemble de la base train :  0.8510002588075162
Accuracy sur la base test :  0.8514537264537264
```

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




*Matrice de confusion pour le Linear Support Vector Classifier*

On observe ici un taux de prédiction sur la base test de 93% pour les salaires inférieurs à 50K, et un taux de prédiction pour les salaires supérieurs à 50K de 56%.

| | LinearSVC |
|---|---|
| % Accuracy en Test | 85.0 |
| % Accuracy en Train | 85.0 |
| C | 0.1 |

*Résultats LinearSVC*

## D. SVC à Kernel Gaussien

Parmi les paramètres de la souplesse de la marge C et le gamma, GridSearchCV nous renvoie comme paramètres optimaux :

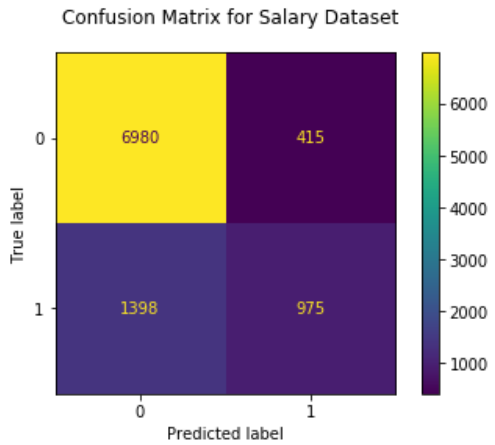
```
SVC(C=1.1, gamma=1)
```

Nos paramètres optimaux ici sont :

- C = 1.1
- Gamma = 1

```
Accuracy à chaque Fold sur la base train :  [0.81092345 0.82364554 0.81834138 0.81834138 0.81483107]
Accuracy sur l'ensemble de la base train :  0.8172165612556197
Accuracy sur la base test : 0.8143939393939394
```

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




*Matrice de confusion pour le Gaussian Support Vector Classifier*

Pour notre dernier classifieur, on observe un taux de prédiction sur la base test sur les salaires inférieur à 50K de 94%, quant à celui sur les salaires supérieurs à 50K, le taux est de 41% ce qui est le taux le plus bas des 4 classifieurs.

| | Gaussian SVC Classifier |
|---|---|
| % Accuracy en Test | 81.0 |
| % Accuracy en Train | 82.0 |
| C | 1.1 |
| Gamma | 1.0 |

*Résultats Gaussian SVC*

# II. Observations

On observe que dans l'ensemble, les scores de nos classifieurs sur la base de test sont relativement les mêmes. Cependant, nous avons observé un mauvais équilibrage entre nos classes, la classe des salaires supérieurs à 50K est inférieure à celle des salaires supérieurs à 50K. Nous retrouvons ce biais sur nos matrices de confusions qui dans l'ensemble n'ont pas de résultats satisfaisants : moins de 80% de prédiction pour cette classe. Nous avons décider d'établir un tableau qui rassemble les taux de reconnaissance mais aussi les taux de prédiction pour chaque classe afin de choisir le classifieur qui serait le plus optimal.

| | MLP Classifier | Decision Tree Classifier | Linear SVC | Gaussian SVC |
|---|---|---|---|---|
| % Accuracy en Train | 85 | 85 | 85 | 81 |
| % Accuracy en Test | 85 | 85 | 85 | 81 |
| % Prédiction classe 0 | 92 | 96 | 93 | 94 |
| % Prédictions classe 1 | 62 | 49 | 56 | 41 |

*Tableau comparatif des performances des Classifieurs*

Nous constatons que dans l'ensemble, le MLP Classifier est le classifier qui généralise le mieux nos données, c'est ce classifieurs que nous conserverons afin de prédire les salaires de manière optimale.

Nous constatons aussi que les performances globales sur la prédiction de la classe 1 sont assez médiocres, ce qui révèle l'important d'avoir un bon équilibrage entre nos données en entrée afin d'entrainement au mieux nos classifieurs.