Université Paris 2 Panthéon-Assas
Session 2021–22

Machine Learning

Séance 9

L'objectif de cette session est de (i) rappeler les bases de la classification et (ii) d'appliquer nos connaissances sur python.

Deux bases de données sont à votre disposition :

— `data_X.xlsx`

— `data_y.xlsx`

1 Importation, description et visualisation des données

1. Importez les bases de données en utilisant le package `pandas`
   — Commande : `pandas.read_excel("cheminverslesdonnees")`
   — Exemple ci-dessous (il faut changer le chemin d'accès aux données)

```
import pandas
X = pandas.read_excel("/Users/Assas/machine_learning/data_X.xlsx")
y = pandas.read_excel("/Users/Assas/machine_learning/data_y.xlsx")
```

2. Décrivez la structure des données :
   — Pour la base X : `type(X)`, `X.head()`, `X.shape`, `X.columns`, `X.dtypes`)

```
print(type(X))
print(X.head(10))
print(X.shape)
print(X.columns)
print(X.dtypes)
print(X.info())
```

   — Quelle est la structure de la base X ? Que représentent les colonnes ? Dans quel format les données sont-elles stockées ?

3. Visualisez les données :

```

import matplotlib as mpl
import matplotlib.pyplot as plt
chiffre = X.iloc[1]
chiffre_image = chiffre.values.reshape(28, 28)
plt.imshow(chiffre_image,cmap=mpl.cm.binary,interpolation="nearest")
plt.axis("off")
plt.show()
y.iloc[1]
```

4. Que représente Y ? Que représente la première observation stockée dans la variable Y ?

2 Préparation des données

5. Divisez la base de données en deux sous-bases : une base d'apprentissage et une base de test
   — Utilisez le code suivant :

```
import numpy as np
y = y.astype(np.uint8).values.ravel()
X_train, X_test, y_train, y_test = X[:4000], X[4000:], y[:4000], y[4000:]
```

— Combien d'observations contient la base de test ? Quelle proportion de la base de données cela représente ?

6. Vrai ou faux :
   — La base de test est utilisée pour tester la performance du modèle
   — Les résultats vont dépendre des observations incluses dans la base d'apprentissage et dans la base de test
   — Il suffit de classer aléatoirement les observations avant la division de la base de données en deux pour obtenir toujours les mêmes performances pour un modèle donné
7. Utilisez le code suivant :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

8. Combien d'observations contient la base de test ? Quelle proportion de la base de données cela représente ?
9. Créez la variable de réponse suivante
   — Dans quel cas cette variable vaut 1 ? Dans quel cas vaut-elle 0 ?

```
y_train_0 = (y_train == 0)
y_test_0 = (y_test == 0)
```

3 Classification

3.1 Entraînement

10. Utilisez le code suivant :

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(loss='log',random_state=42)
sgd_clf.fit(X_train, y_train_0)
sgd_clf.predict([chiffre])
```

11. Questions sur le code :
    — Quel modèle a-t-on utilisé ?
    — Avons nous réussi à prédire correctement le chiffre 0 ?
12. Questions sur la descente de gradient :
    — A quoi sert l'algorithme de descente de gradient ? Comment fonctionne-t-il ?
    — Vrai ou faux : l'initialisation de l'algorithme de descente de gradient consiste à choisir des paramètres au hasard ?
    — Vrai ou faux : l'algorithme de descente de gradient converge toujours vers le minimum global
13. Questions sur la régression logistique
    — Quel est le principal avantage de la régression logistique sur la régression linéaire lorsque Y est binaire ?
    — Vrai ou faux : dans le cadre de la régression logistique, on peut utiliser les équations normales pour trouver les paramètres d'intérêt
    — Vrai ou faux : le modèle de régression logistique est un modèle linéaire

3.2 Validation Croisée

14. Utilisez le code suivant :

```
from sklearn.model_selection import cross_val_score
cross_val_score(sgd_clf, X_train, y_train_0, cv=3, scoring="accuracy")
```

15. Questions sur le code
    — En combien de groupes ("folds") la base d'entraînement est-elle divisée ?
    — Quelle est la performance ("accuracy") du modèle sur chacun des groupes ?
16. Question sur la validation croisée ?
    — Pourquoi utilise-t-on la validation croisée ?
    — Quelle est la différence entre la LOOCV (Leave one out cross-validation) et la k-fold cross validation ?
    — Vrai ou faux : lorsque k=n, la LOOCV et la k-fold cross validation donnent les mêmes résultats
    — Vrai ou faux : l'avantage de la LOOCV sur la k-fold cross validation est d'ordre computationnel
    — Vrai ou faux : la LOOCV apporte des résultats différents si on l'implémente plusieurs fois (en utilisant la même base de données et les mêmes modèles)

3.3 Comparaison avec un estimateur naïf

17. Utilisez le code suivant :

```
from sklearn.base import BaseEstimator
class Never0Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)

never_0_clf = Never0Classifier()
cross_val_score(never_0_clf, X_train, y_train_0, cv=3, scoring="accuracy")
```

18. Questions sur le code :
    — Pourquoi cet estimateur est-il considéré comme un estimateur naïf ?
    — Pourquoi est-ce important de comparer nos résultats avec ceux obtenus grâce à cet estimateur ?
    — Obtient-on des meilleurs performances que l'estimateur naïf ?

4 Mesurer la performance de l'estimateur

19. Utilisez le code suivant :

```
from sklearn.model_selection import cross_val_predict
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_0, cv=3)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
confusion_matrix(y_train_0, y_train_pred)
```

20. Questions
    — Qu'est-ce qu'une matrice de confusion ?
    — A l'aide du tableau précédent :

— Donnez le nombre de (i) vrai négatif, (ii) faux positif, (iii) faux négatif et (iv) vrai positif
— Calculez la précision
— Calculez le rappel
— Calculez l'erreur de classification moyenne

— Vrai ou faux
— Pour mesurer la qualité de l'estimation d'un modèle : la précision est un indicateur plus pertinent que le rappel
— Pour mesurer la qualité de l'estimation d'un modèle : le rappel est un indicateur plus pertinent que la précision
— Pour mesurer la qualité de l'estimation d'un modèle : l'erreur de classification moyenne est un indicateur plus pertinent que la précision ou le rappel

— Utilisez le code suivant :

```
print(classification_report(y_train_0,y_train_pred))
```

5 Comparaison K plus proche voisin et régression logistique

21. Vrai ou faux :
— Les K-plus proches voisins désignent une méthode paramétrique
— Les K-plus proches voisins désignent une méthode d'apprentissage non supervisé
— Dans le cadre des K-plus proches voisins, le nombre K de voisins est choisi automatiquement via l'algorithme de descente de gradient

22. Utilisez le code suivant :

```
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report

sgd_clf = SGDClassifier(loss='log',random_state=42)
sgd_clf.fit(X_train, y_train_0)
sgd_clf.predict([chiffre])

y_pred_log = sgd_clf.predict(X_test)
print(classification_report(y_test_0,y_pred_log))

from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, y_train_0)

y_pred_neigh = neigh.predict(X_test)
print(classification_report(y_test_0,y_pred_neigh))
```

23. Pour obtenir le meilleur taux de classification, quel modèle choisiriez-vous ? Une régression logistique ou les K-plus proches voisins ? Pourquoi ?

24. Utilisez le code suivant

```
error_rate = []
for i in range(1,40):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train,y_train_0)
    pred_i = neigh.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test_0))
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate, linestyle='dashed',marker='o', markersize=10)
plt.title('Taux d erreur de classification vs. Nombre de voisins K')
plt.xlabel('K')
plt.ylabel('Taux d erreur de classification')
req_k_value = error_rate.index(min(error_rate))+1
```

25. Quel K choisiriez-vous ? Pourquoi ?