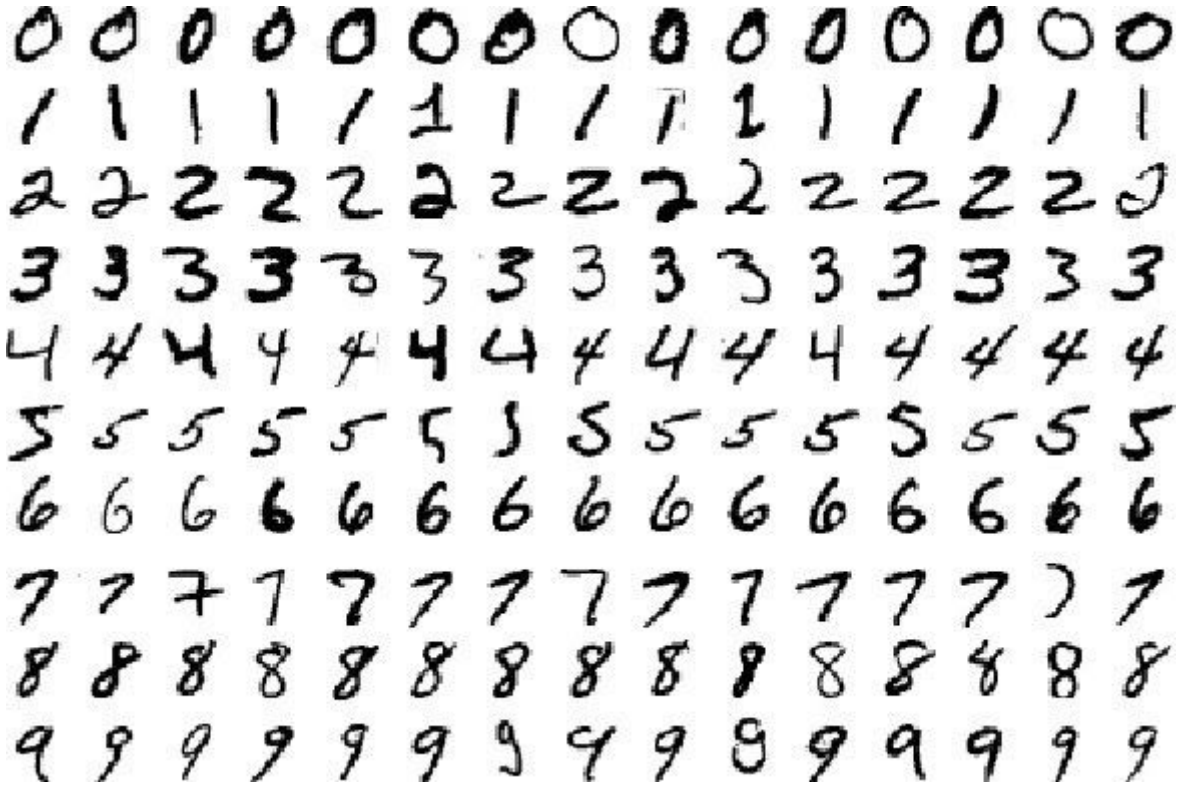
Classification des chiffres manuscrits



Christian MBARGA MVOGO,Eva MOULARD, Céline GUILLEN

1. Information sur les données

On affiche les clés des données:
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_names', 'DESCR', 'details', 'url'])

On détermine les features :
X = data1["data"]

On affiche la première instance: X[0]

On détermine la cible :
Y = data1["target"]

On affiche les classes de Y :
array(['5', '0', '4', ..., '4', '5', '6'], dtype=object)

On affiche la première image:

On transforme la liste de string de Y en integer avant la répartition des données.

2. Répartition des données

```
# Répartition de la base d'entraînement et de la base de test:
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 1/7 ,
random_state=4
```

3. Apprentissage d’un classificateur binaire

3.1 Apprentissage des données

On crée une liste de booléen pour savoir si la cible est égale à la première instance(5) ou pas :

```
ytrain5 = (Y_train == 5)
```

3.2 Évaluation du modèle d’apprentissage sur les données d’apprentissage

On instancie la classe SGDClassifier :

```
import numpy as np
from sklearn.linear_model import SGDClassifier
sgdc = SGDClassifier(max_iter=5, tol=-np.infty)
```

On entraîne le modèle:

```
sgdc.fit(X_train, ytrain5)
```

On évalue le modèle:

```
from sklearn.model_selection import cross_val_score
cvs = cross_val_score(sgdc, X_train, ytrain5,scoring ="accuracy", cv = 3)
array([0.964  , 0.96695, 0.96375])
```

La moyenne des scores :

```
cvs.mean()
0.9649
```

3.3 Taux de classification

```
# Validation croisée 1 pour le taux de classification de chaque fold:
from sklearn.model_selection import cross_val_score
cvs = cross_val_score(sgdc, X_train, ytrain5,scoring ="accuracy", cv = 3)

# La moyenne des taux de classification
cvs.mean()
0.9649
```

Pour évaluer notre modèle, on va utiliser la validation croisée en utilisant la méthode 3fold_cross_validation. On va donc partager le jeu d'entraînement en 3 blocs, puis on effectue des prédictions et on évalue ces prédictions sur chaque bloc en utilisant un modèle entraîné sur le reste des blocs.

Création de la classe Never5Classifier

```
from sklearn.base import BaseEstimator
```

```
class Never5Classifier(BaseEstimator):
    def fit():
        pass
    def predict():
        return np.zeros((len(X), 1), dtype=bool)
```

On obtient un taux de classification de plus de 90% sur les 3 folds et un taux de classification moyen de 91%. Peut-on vraiment en déduire que le modèle est performant? Ces bons résultats sont peut être dû au fait que les 5 sont finalement peu représentés et que quoi qu'il arrive la probabilité de ne pas avoir de 5 sera vraie à 90%. On ne peut donc pas se fier à de bons résultats pour un classifieur, ce n'est pas un signe de bonne performance du modèle mais plutôt le résultat d'une sous représentation des données à étudier dans le jeu de données

3.4 Matrice de confusion

```
#Fonction cross_val_predict
from sklearn.model_selection import cross_val_predict
Y_train_predict = cross_val_predict(sgdc, X_train,ytrain5, cv=3)

# Matrice de confusion
from sklearn.metrics import confusion_matrix
confusion_matrix(ytrain5,Y_train_predict)

array([[52968,  1656],
       [ 1209,  4167]], dtype=int64)
```

3.5 Précision et rappel

- **La précision** permet d'étudier l'exactitude des prédictions positives. Sa formule est précision=(TP/(TP + FP))
- **Le rappel (ou sensibilité)** est le taux d'observation positives ayant été correctement détectées par le classificateur. La formule est: rappel= (TP/(TP+FN)).

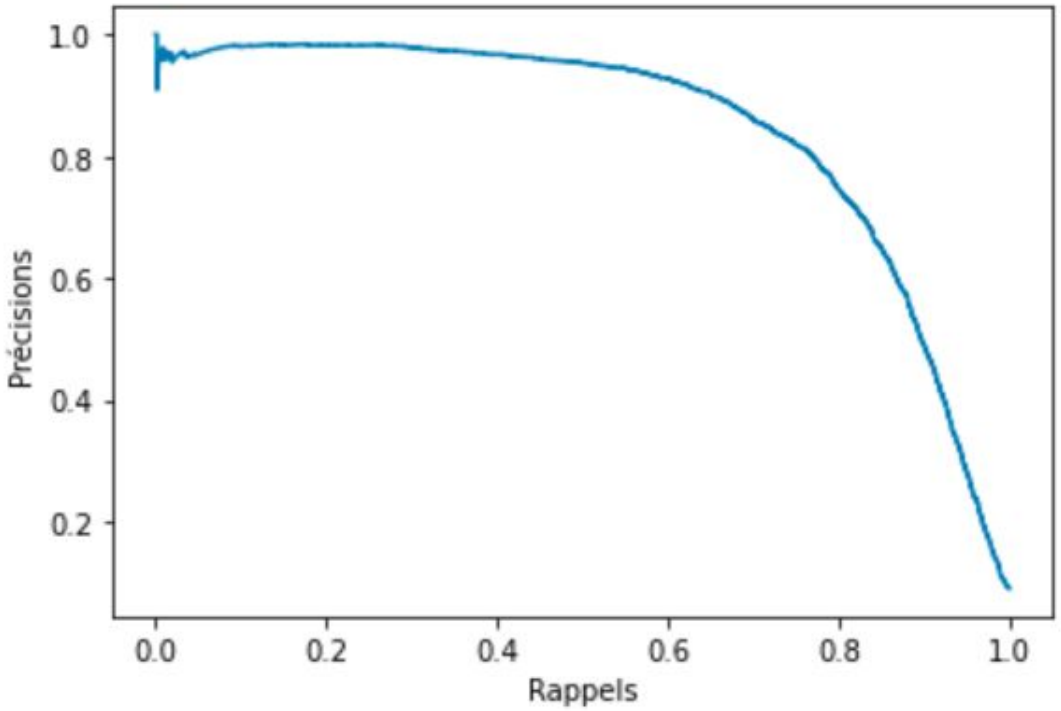
```
# La précision:
from sklearn.metrics import precision_score
precision_score(ytrain5,Y_train_predict)
0.7156105100463679
```

 # Rappel:
from sklearn.metrics import recall_score
recall_score(ytrain5,Y_train_predict)
0.7751116071428571

 # F1_score:
from sklearn.metrics import f1_score
f1_score(ytrain5,Y_train_predict)
0.7441735869274042

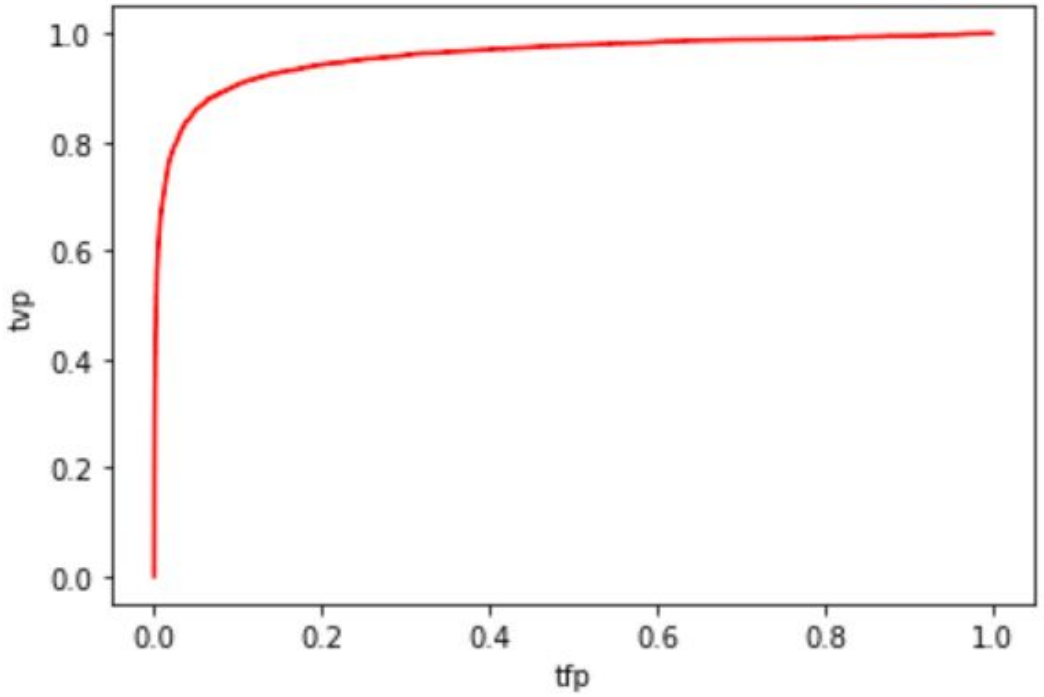
il est logique de trouver un F1_score similaire car on a une précision et un rappel quasiment identique. Pour voir le compromis existant entre Rappel et précision, le SGDClassifier prend des décisions de classification. Il va alors pour chaque observation, calculer un score basé sur une fonction de décision. Si ce score est supérieur à un certain seuil, il affecte la donnée à la classe positive, sinon il l'affecte à la classe négative.

 # Le classifieur SGD calcule un score en se basant sur sa fonction de décision. Si le score est supérieur à un seuil, il affecte la classe positive à l'instance sinon il affecte la classe négative.
from sklearn.model_selection import cross_val_predict
scores1 = cross_val_predict(sgdc, X_train,ytrain5, method= 'decision_function',cv=3)



Le classificateur s'améliore pour passer de 86 % de précision de moyenne à 90 %. Cependant , il peut encore s'améliorer.

3.6 Courbe ROC



Evaluation de la courbe ROC:
from sklearn.metrics import roc_auc_score
roc_auc_score(ytrain5,scores1)
0.9586734255078441

L'évaluation est assez correcte. Il y a des liens avec la courbe précision/rappel.

4. Apprentissage d'un classifieur multi-classes

4.1 Apprentissage des données

On entraîne le modèle :
sgdc.fit(X_train,Y_train)

On fait les prédictions:
Y_pred2 = sgdc.predict(X)
array([5, 0, 4, ..., 4, 5, 6])

scores2 = sgdc.decision_function(X)

array([[-3.60893016e+05, -4.59033339e+05, -2.05427229e+05, ...,
-2.86198455e+05, -4.38039997e+05, -3.60101066e+05],

```
[ 5.59419776e+05, -6.09898159e+05, -3.39405482e+05, ...,
 -3.50388735e+05, -3.62962804e+05, -2.95910653e+05],
[-4.04446748e+05, -4.13663786e+05, -2.90450203e+05, ...,
 -3.58887468e+05, -4.62192037e+05, -2.78766510e+05],
...,
[-9.14262828e+05, -7.48986362e+05, -5.19401462e+05, ...,
 -1.45848587e+05, -1.81065854e+05, -1.01708187e+03],
[-6.99449755e+05, -3.19878790e+05, -3.95793343e+05, ...,
 -4.12620835e+05, -1.19534287e+05, -4.82354096e+05],
[-6.23010730e+05, -1.07105647e+06, -2.83161142e+05, ...,
 -6.46254735e+05, -1.00339377e+06, -8.57926008e+05]])
```

4.2 Évaluation du modèle d'apprentissage sur les données d'apprentissage

4.2.1 Taux de classification

```
from sklearn.model_selection import cross_val_score
cvs2 = cross_val_score(sgdc, X_train, Y_train,scoring ="accuracy", cv = 3)
array([0.86235, 0.87565, 0.87195])
```

La moyenne des scores:

```
cvs2.mean()
0.8699833333333333
```

Mise en échelle de la base d'entraînement:

```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
scaling1 = scale.fit_transform(X_train)
```

On obtient un résultat encore meilleur en normalisant nos données.

Scores par rapport à la mise en échelle:

```
cvs3 = cross_val_score(sgdc, scaling1, Y_train,scoring ="accuracy", cv = 3)
array([0.91025, 0.91165, 0.90805])
```

Moyenne des scores :

```
cvs3.mean()
0.9099833333333333
```

Le classificateur s'améliore pour passer de 86 % de précision de moyenne à 90 %.

4.2.2 Matrice de Confusion

Mise en échelle de la base d'entraînement:
scaling2 = scale.fit_transform(X_train)

Prédiction sur la nouvelle mise en échelle :
Y_train_predict2 = cross_val_predict(sgdc,scaling2 , Y_train, cv = 10)

Matrice de confusion:
mat_confu1 = confusion_matrix(Y_train,Y_train_predict2)

```
array([[5740,    2,   24,   11,   11,   47,   41,    9,   33,    2],
       [   1, 6476,   47,   21,    5,   37,    9,   12,  106,   11],
       [  49,   29, 5383,  110,   88,   27,   81,   66,  171,   19],
       [  43,   33,  123, 5369,    3,  220,   35,   51,  137,   93],
       [  22,   31,   42,    4, 5415,    7,   56,   34,   90,  217],
       [  70,   38,   28,  184,   67, 4563,  119,   33,  174,  100],
       [  33,   25,   47,    1,   39,   99, 5606,    9,   56,    0],
       [  27,   25,   75,   37,   49,    9,    6, 5770,   17,  223],
       [  51,  139,   76,  144,   14,  154,   51,   21, 5065,  141],
       [  44,   30,   25,   85,  156,   38,    2,  204,   83, 5255]],
      dtype=int64)
```

5. Conclusion

Avec les différents modèles utilisés, on voit très clairement qu'ils peuvent être améliorés au fur à mesure de l'entraînement et des évaluations de ces mêmes modèles.