23/11/2018 Yuxin SHI

Apprentissage automatique

A l'aide des méthodes vues en cours, et à partir des données disponibles, proposer au moins trois classifieurs (prédicteur de SPAM). Vous prendrez soin de détailler toutes les démarches faite pour construire le classifieur notamment:

Choix de la méthode. Calibration et éventuel choix des variables utilisées. Estimation de la qualité de classification.

In [37]:

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn import cross_validation

#Load data
spam_train = pd.read_csv('spamdataset_train.csv')
spam_test = pd.read_csv('spamdataset_test.csv')
#split dataset
test_features = spam_test
train_target = spam_train['Spam']
train_features = spam_train.drop('Spam', axis=1)
```

In [29]:

```
from sklearn.svm import SVC

# Evaluation of quality score
# Split the train dataset in using cross validation method
eval_features, test_eval_features, eval_target, test_eval_target = \
cross_validation.train_test_split(train_features, train_target, test_size=0.33)
modelSVC = SVC()
modelSVC.fit(eval_features, eval_target)
resEval = modelSVC.predict(test_eval_features)
score = accuracy_score(test_eval_target, resEval)
print("SVC quality is : " + str(score*100) + "%")

# Prediction of test dataset
modelSCV.fit(train_features, train_target)
resSCV = modelSCV.predict(test_features)
print(resSCV)
# Add it as a new column
spam_test['Spam'] = resSCV
spam_test.to_csv('scvSapmTest.csv', encoding='utf-8')
```

```
SVC quality is : 83.2778147901%
[0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0
1 0 1
 0 0 1 0 1 0 0 0 1 1 0 1 0]
```

In [31]:

```
from sklearn.tree import DecisionTreeClassifier

#Evaluation of quality score
eval_features, test_eval_features, eval_target, test_eval_target = \
cross_validation.train_test_split(train_features, train_target, test_size=0.33)
modelTree = DecisionTreeClassifier()
modelTree.fit(eval_features, eval_target)
resTreeEval = modelTree.predict(test_eval_features)
scoreTree = accuracy_score(test_eval_target, resTreeEval)
print("CART quality is " + str(scoreTree*100) + "%")

# Prediction of test dataset
modelSCV.fit(train_features, train_target)
resTree = modelTree.predict(test_features)
print(resTree)

# Add it as a new column
spam_test['Spam'] = resTree
spam_test.to_csv('treeSapmTest.csv', encoding='utf-8')
```

```
CART quality is 90.6728847435%
[0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0
1 0 1
 1 1 1 0 1 0 1 0 1 0 0 1 1]
```

In [38]:

```
from sklearn.naive_bayes import GaussianNB
#It runs quick, but a low quality

#Evaluation of quality score
eval_features, test_eval_features, eval_target, test_eval_target = \
cross_validation.train_test_split(train_features, train_target, test_size=0.33)
modelGNB = GaussianNB()
modelGNB.fit(eval_features, eval_target)
resGNBEval = modelGNB.predict(test_eval_features)
scoreGNB = accuracy_score(test_eval_target, resGNBEval)
print("Native bayes in gausin quality is " + str(scoreGNB*100) + "%")

# Prediction of test dataset
modelGNB.fit(train_features, train_target)
resGNB = modelGNB.predict(test_features)
print(resGNB)

# Add it as a new column
spam_test['Spam'] = resTree
spam_test.to_csv('GNBSapmTest.csv', encoding='utf-8')
```

```
Native bayes in gausin quality is 80.6129247169%
[0 0 1 1 1 0 0 1 0 0 1 0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0
1 1 1
 0 0 1 0 1 1 1 1 1 0 0 1 0]
```

In [36]:

```
from sklearn.linear_model import LogisticRegression

#Evaluation of quality score
eval_features, test_eval_features, eval_target, test_eval_target = \
cross_validation.train_test_split(train_features, train_target, test_size=0.33)
modelLR = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multino
mial')
modelLR.fit(eval_features, eval_target)
resLREval = modelLR.predict(test_eval_features)
scoreLR = accuracy_score(test_eval_target, resLREval)
print("Logistic regression quality is " + str(scoreLR*100) + "%")

# Prediction of test dataset
modelLR.fit(train_features, train_target)
resLR = modelLR.predict(test_features)
print(resLR)

# Add it as a new column
spam_test['Spam'] = resTree
spam_test.to_csv('LRSapmTest.csv', encoding='utf-8')
```

```
Logistic regression quality is 93.4710193205%
[0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0
1 1 1
 0 0 1 0 1 0 0 0 1 0 0 1 0]
```

Conclusion

Selon le résultat de chaque $\hat{R}(f)$, la regression logistique a une meilleure qualité pour la prédiction.

In [44]:

```
# Export result to csv
data = {'result SCV':resSCV, 'result CART':resTree, 'result Native Bayes':resGNB
, 'result logistic regression':resLR}
resTotal = pd.DataFrame(data)
resTotal.to_csv("ALL_RES.csv", encoding='utf-8')
```