Report Data Science London + Scikit-learn

March 12, 2017

1 Report: Data Science London + Scikit-learn

In [ ]:
```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar  7 20:39:56 2017

@author: hp
"""

# Pandas is a open source library provides high performance, easy-to-
# use data structures and data amalysis tools for Python
import pandas

# Importing Scatter_matrix from pandas.tools.plotting: help to create
# Scatter plot
from pandas.tools.plotting import scatter_matrix

# Importing Matplotlib.pyplot to create figures and axes to achieve the des
import matplotlib.pyplot as plot

from sklearn.metrics import classification_report

# Importing train_test_split that will quickly split data into train and te
from sklearn.cross_validation import train_test_split

# Importing PCA from sklearn.decomposition to perform pca functionality
from sklearn.decomposition import PCA

import matplotlib.pyplot as plot
# Importing Confusion_matrix from sklearn,metrics to create confusion matri
from sklearn.metrics import confusion_matrix

# model_selection.cross_val_score takes a scoring parameter that controls
# what metric they apply to the estimators evaluated
from sklearn import model_selection

# Importing accuracy from sklearn.metrics to calculate accuracy score
```

```
from sklearn.metrics import accuracy_score

from sklearn.linear_model import Perceptron

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.svm import SVC

# Numpy supports large, multi-dimensional arrays and matrices, along with
import numpy as np
# OS provides a huge range of useful methods to manipulate files and direct
import os


working_directory =r"D:\Aakash_Documents\MS_Collections\AcceptanceFromSaint


############################Section 1: Read the data ####################
# In this section we will read all the 3 datasets

#fetch training data
train_data = np.genfromtxt(open(working_directory + 'train.csv','rb'), deli

#fetch target variable data
target_data = np.genfromtxt(open(working_directory + 'trainLabels.csv','rb'

#fetch testing data
test_data = np.genfromtxt(open(working_directory + 'test.csv','rb'), delimi
############################Section 1: End ##############################

##########Section 2: create different functions for future use##########
"""
function 1: This funtion will divide the training data in 2 parts (80 & 20
            Model will be trained on first part and validation will be perf
            second part.
            Basically it will help us to understand whether selected model
            perform better on final test data or not.
"""
def dividedata(train_data,target_data):
    train,test,train_target,test_target = train_test_split(train_data,targe
    return train,test,train_target,test_target

"""
```

```
function 3: In this function we will do Principal Component Analysis(PCA) i
            important function because if we will check our dataset there a
            40 features or variables. And not every variable is that import
            able to define the target variable. In such cases it becomes im
            to first recognize important variables that can define the maxi
            Basically, this function will help us in Dimensionality Reducti
"""
def dopca(train,train_target,test,test_data):
    pca = PCA(n_components=12,whiten=True)
    train = pca.fit_transform(train,train_target)
    test = pca.transform(test)
    test_data = pca.transform(test_data)
    return train,test,test_data
############################Section 2: End#############################

#################### Section 3: Perform Data Split and PCA #############
train,test,train_target,test_target = dividedata(train_data,target_data)

train,test,test_data = dopca(train,train_target,test,test_data)
#################### Section 3: End ####################################

#################### Section 4: Statistical Model Building and Evaluation
# Testing options and evaluating metric
# Reset the random number seed before each run to ensure that the
# evaluation of each algorithm is performed using exactly the same data spl
seed = 7 # 7 is just the id, can be any number
scoring = 'accuracy'

# Checking Algorithms
# Creating empty list to use it for every model in for loop
models = []

models.append(('LR', LogisticRegression()))
models.append(('PR', Perceptron()))

models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('QDA', QuadraticDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))
#models.append(('RNC', RadiusNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))
models.append(('SVM', SVC()))

# evaluating each model in turn
results = []
names = []
msg=[]
```

```
# Creating for loop to call different models
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    #cv_results = model_selection.cross_val_score(model, train_data, target
    cv_results = model_selection.cross_val_score(model, train, train_target
    results.append(cv_results)
    names.append(name)
    msg.append ("%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())

######### Statistical Model Accuracy and Standard Deviation ###########
#['LR: 0.831040 (0.010621)',
# 'PR: 0.769753 (0.035249)',
# 'LDA: 0.823796 (0.023128)',
# 'QDA: 0.934800 (0.014980)',
# 'KNN: 0.938887 (0.017178)',
# 'CART: 0.779459 (0.021256)',
# 'SVM: 0.938345 (0.008632)']
#########################################################################

# From the above, we can see that KNN, SVM and QDA has least Standard Devia
# good accuracy sore
# Comparing Algorithms
fig = plot.figure()
fig.suptitle('Algorithms Comparison')
a = fig.add_subplot(111)
plot.boxplot(results)
a.set_xticklabels(names)
plot.show()
#################### Section 4: End ##################################

####### Section 5: Statistical Model Fitting and Prediction on 20% of trai
#####let's perform prediction on 20% part of training data and store result
#### at the end we will select model on the basis of accuracy scores
accuracyscore=[]
################## Evaluating Models #################

##################Start: Support Vector Machines ##################
# Making predictions on validation dataset using SVM
svm = SVC()
# Fitting the training data based on target data
svm.fit(train, train_target)
# Predicting the output of test data based on the model
predict_svm = svm.predict(test)
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_svm))
print(confusion_matrix(test_target, predict_svm))
print(classification_report(test_target, predict_svm))
```

```
accuracyscore.append(('svm',accuracy_score(test_target, predict_svm)))
###Classification report of SVM model################
#             precision    recall  f1-score   support
#
#        0.0       0.96      0.92      0.94       106
#        1.0       0.91      0.96      0.93        94

# avg / total       0.94      0.94      0.94       200
###Classification report of SVM model################

###Cofusion Matrix####
#[[97  9]
# [ 4 90]]
######################
#################End: Support Vector Machines #################


##################Start: Perceptron #################
################## Perceptron #################
# Making predictions on validation dataset using SVM
pr = Perceptron()
# Fitting the training data based on target data
pr.fit(train, train_target)
# Predicting the output of test data based on the model
predict_pr = pr.predict(test)
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_pr))
print(confusion_matrix(test_target, predict_pr))
print(classification_report(test_target, predict_pr))

accuracyscore.append(('perceptron',accuracy_score(test_target, predict_pr))

###Classification report of Perceptron model################
#        precision    recall  f1-score   support
#
#        0.0       0.78      0.58      0.66       106
#        1.0       0.63      0.82      0.71        94

#avg / total       0.71      0.69      0.69       200
###Classification report of Perceptron model################

###Cofusion Matrix####
#[[61 45]
# [17 77]]
###Cofusion Matrix####
##################End: Perceptron #################

##################Start: QDA #################
```

```
################## QDA ##################
# Making predictions on validation dataset using SVM
qda = QuadraticDiscriminantAnalysis()
# Fitting the training data based on target data
qda.fit(train, train_target)
# Predicting the output of test data based on the model
predict_qda = qda.predict(test)
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_qda))
print(confusion_matrix(test_target, predict_qda))
print(classification_report(test_target, predict_qda))

accuracyscore.append(('QDA',accuracy_score(test_target, predict_qda)))

###Classification report of QDA model###############
#              precision    recall  f1-score   support
#
#        0.0       0.97      0.89      0.93       106
#        1.0       0.88      0.97      0.92        94

#avg / total       0.93      0.93      0.93       200
###Classification report of QDA model###############

###Cofusion Matrix####
#[[94 12]
# [ 3 91]]
###Cofusion Matrix####
##################End: QDA ##################

##################Start: LDA ##################
################## LDA ##################
# Making predictions on validation dataset using SVM
lda = LinearDiscriminantAnalysis()
# Fitting the training data based on target data
lda.fit(train, train_target)
# Predicting the output of test data based on the model
predict_lda = lda.predict(test)
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_lda))
print(confusion_matrix(test_target, predict_lda))
print(classification_report(test_target, predict_lda))

accuracyscore.append(('LDA',accuracy_score(test_target, predict_lda)))
###Classification report of LDA model###############
#              precision    recall  f1-score   support
#
#        0.0       0.86      0.81      0.83       106
#        1.0       0.80      0.85      0.82        94
```

```
#avg / total       0.83      0.83      0.83       200
###Classification report of LDA model################

###Cofusion Matrix####
#[[86 20]
# [14 80]]
###Cofusion Matrix####
##################End: LDA #################


##################Start: KNN #################
################## K-Nearest Neighbour #################
# Making predictions on validation dataset using KNN
knn = KNeighborsClassifier()
# Fitting the training data based on target data
knn.fit(train, train_target)
# Predicting the output of test data based on the model
predict_knn = knn.predict(test)
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_knn))
print(confusion_matrix(test_target, predict_knn))
print(classification_report(test_target, predict_knn))

accuracyscore.append(('KNN',accuracy_score(test_target, predict_knn)))
###Classification report of KNN model################
#             precision    recall  f1-score   support
#
#        0.0       0.95      0.90      0.92       106
#        1.0       0.89      0.95      0.92        94
#
#avg / total       0.92      0.92      0.92       200
###Classification report of KNN model################

###Cofusion Matrix####
#[[95 11]
# [ 5 89]]
###Cofusion Matrix####
##################End: KNN #################

##################Start: Logistic Regression #################
################## Logistic Regression #################
# Making predictions on validation dataset
l = LogisticRegression()
# Fitting the training data based on target data
l.fit(train, train_target)
# Predicting the output of test data based on the model
predict_lr = l.predict(test)
```

```
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_lr))
print(confusion_matrix(test_target, predict_lr))
print(classification_report(test_target, predict_lr))

accuracyscore.append(('Logistic',accuracy_score(test_target, predict_lr)))
###Classification report of LR model###############
#             precision    recall  f1-score   support
#
#        0.0       0.88      0.82      0.85       106
#        1.0       0.81      0.87      0.84        94
#
#avg / total       0.85      0.84      0.85       200
###Classification report of LR model###############

###Cofusion Matrix####
#[[87 19]
# [12 82]]
###Cofusion Matrix####
##################End: Logistic Regression #################


##################Start: Decision Tree #################
################## Decision Tree #################
# Making predictions on validation dataset
d = DecisionTreeClassifier()
# Fitting the training data based on target data
d.fit(train, train_target)
# Predicting the output of test data based on the model
predict_d = d.predict(test)
# Printing Accuracy Score, Confusion Matrix and Classification Report
print(accuracy_score(test_target, predict_d))
print(confusion_matrix(test_target, predict_d))
print(classification_report(test_target, predict_d))

accuracyscore.append(('Decision',accuracy_score(test_target, predict_d)))
###Classification report of Decision Tree model###############
#             precision    recall  f1-score   support
#
#        0.0       0.91      0.86      0.88       106
#        1.0       0.85      0.90      0.88        94
#
#avg / total       0.88      0.88      0.88       200
###Classification report of Decision Tree model###############

###Cofusion Matrix####
#[[91 15]
# [ 9 85]]
```

```
###Cofusion Matrix####
##################End: Decision Tree #################
####### Section 5: END ########################################

####### Section 6: Model Selection and Prediction of Test Data ##########
###let's check accuracy score of each model and then we will select our fi
(accuracyscore)

#[('Decision', 0.88),
# ('KNN', 0.92000000000000004),
# ('LDA', 0.82999999999999996),
# ('Logistic', 0.84499999999999997),
# ('QDA', 0.92500000000000004),
# ('perceptron', 0.68999999999999995),
# ('svm', 0.93500000000000005)]


#####################################################################
# From the above Accuracy Score and Classification Report
# SVM (Accuracy Score = 0.935),
# QDA (Accuracy Score = 0.925 ),
# KNN (Accuracy Score = 0.92) are the three best models

# So, We will apply SVM Model on Testing Data, will predict the output and
# the results in prediction.csv file

results = svm.predict(test_data)
idcol = np.arange(start=1,stop=9001)
results2 = np.column_stack((idcol,results))

np.savetxt(working_directory + 'prediction.csv',results2,fmt='%d',delimiter
####### Section 6: END ########################################################
```