DTCS-620: Statistics for Data Science

New York Institute of Technology

Project - I

Student Name: Niral Patel
Student ID: 1303276
Code Link: https://github.com/niral-patel/Statics-for-DS-Assignment-1/blob/main/Assingment%20-%201.ipynb
Python code: https://github.com/niral-patel/Statics-for-DS-Assignment-1/blob/main/Assignment-1.py

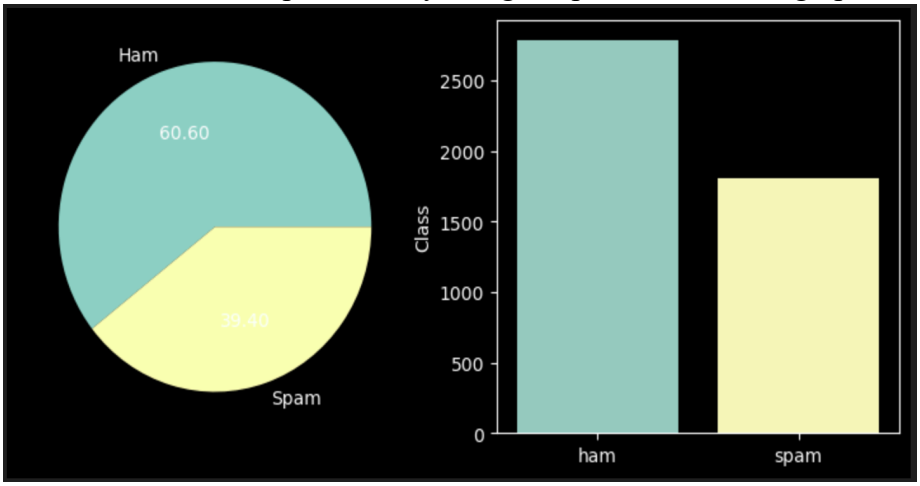
To perform tasks, I import different important classes such as pandas, numpy, sklearn, etc. Moreover, to remove the warnings I imported the warnings class and use action method to ignore the filter warnings.

```
[46]  ✓ 0.4s                                                    Python

import pandas as pd
import numpy as np

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings(action='ignore')

#Other additional imports
warnings.filterwarnings('ignore')
```

Now It's time to upload the data-set and storing it in df and get more information about it by using info() method

```
[47]  ✓ 0.3s                                                    Python

df = pd.read_csv("spam.csv")

[48]  ✓ 0.3s                                                    Python

df.info()

...  <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 4601 entries, 0 to 4600
     Data columns (total 58 columns):
      #   Column      Non-Null Count  Dtype
     ---  ------      --------------  -----
```

Check the total number of Ham data and Spam data by using the pie chart and bar graph.



As per requirement, I split the data in to two parts, one is train data set which cover the first 1000 rows of data to train the model and second one is rest below 1000 rows for the testing purpose.

```
# Createae the Test and Train Data using basic python sysntex
train_df = df.iloc[:1000,:]
test_df = df.iloc[1000:,:]

X_train = train_df.drop(columns='Class')
Y_train = train_df['Class']
X_test = test_df.drop(columns='Class')
Y_test = test_df['Class']
print("Total Train Data", train_df.shape,'\nTotal Test Data',test_df.shape,"\nTest Data set x:",X_test.shape,'\nT
```

[49] ✓ 0.2s Python

```
Total Train Data (1000, 58)
Total Test Data (3601, 58)
Test Data set x: (3601, 57)
Test Data set y: (3601,)
Train Data set x: (1000, 57)
Train data set y: (1000,)
```

Reporting Task: 1

Compare the accuracies of the Random Forest classifier as a function of the number of base learners (e.g., 10, 50, 100, 500, 1000, and 5000) and the number of features to consider at each split (e.g., auto or sqrt). Report your observations/conclusions and provide evidence to support your conclusions.

To compare the accuracy, I created the two for loop, first loop contains features which are 'Auto' and 'SQRT', and second loop contains estimators which are '10, 50, 100, 500, 1000, 5000'. In those two loops I perform steps to get accuracy from 'RandomForestClassifier' model and store them in a Data Frame.

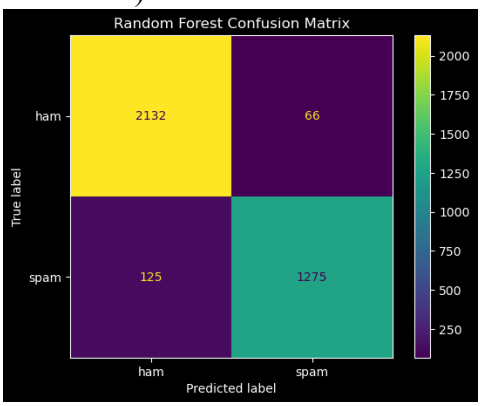
```python
# Random Forest classifier
#--------------------------
from sklearn.ensemble import RandomForestClassifier
#--------------------------

estimator = [10,50,100,500,1000,5000] #estimate values
features = ['auto', 'sqrt'] #feature values
rf_accuracy_auto = [] # Empty array to store the Accuracy score
rf_accuracy_sqrt = []
print('\nRandom Forest Classification model','\n')
for i in features: #for loop for feature selection
    print('\nRandom Forest model with Feature:',i,'\n')
    for j in estimator: #for loop for estimator selection
        random_forest_clf = RandomForestClassifier(n_estimators=j, max_features=i) #create a RF model with feature and estimator
        random_forest_clf = random_forest_clf.fit(X_train, Y_train) #Train the model
        rf_score = random_forest_clf.score(X_test, Y_test) # Test the model and get the accuracy score
        print('Classification Accuracy with ',j,' estimators:', round((rf_score),4), "\n")
        if i == 'sqrt':
            rf_accuracy_sqrt.append(rf_score) #store the accuracy score if feature is squrt
        else:
            rf_accuracy_auto.append(rf_score) #store the accuracy score if feature is auto

accuracy1 = pd.DataFrame(rf_accuracy_sqrt,columns = ['accuracy_sqrt'])
accuracy2 = pd.DataFrame(rf_accuracy_auto,columns = ['accuracy_auto'])
accuracy = pd.merge(accuracy1,accuracy2,left_index=True,right_index=True)
print (accuracy)
```

✓ 22.1s Python Python Python

After performing this code, the output will show as below. As per the output the Maximum accuracy is gained when the number of base learners(n_estimators) is set to 1000 and the 'max_features' parameter is set to 'sqrt'. The lowest accuracy occurs when 'n_estimators' parameter is set to 10 and 'max_features' is set to 'auto'. In other words, when n_estimator is less the accuracy will decrease and the sqrt in max_features provide the better accuracy than auto. The accuracy result was **0.9336**.

```
Random Forest Classification model

Random Forest model with Feature: auto

Classification Accuracy with  10  estimators: 0.9284

Classification Accuracy with  50  estimators: 0.9342

Classification Accuracy with  100  estimators: 0.9309

Classification Accuracy with  500  estimators: 0.9345

Classification Accuracy with  1000  estimators: 0.9334

Classification Accuracy with  5000  estimators: 0.9317

Random Forest model with Feature: sqrt

Classification Accuracy with  10  estimators: 0.9242

Classification Accuracy with  50  estimators: 0.935

Classification Accuracy with  100  estimators: 0.9334

Classification Accuracy with  500  estimators: 0.9334

Classification Accuracy with  1000  estimators: 0.9339

Classification Accuracy with  5000  estimators: 0.9336

   accuracy_sqrt  accuracy_auto
0       0.924188       0.928353
1       0.935018       0.934185
2       0.933352       0.930853
3       0.933352       0.934463
4       0.933907       0.933352
5       0.933630       0.931686
```

Create classification report and confusion matrix to get the precision and recall information to compare the two classification (Decision tree and Random Forest).




```
# Metrics - Random Forest classifier
rf_predict = random_forest_clf.predict(X_test)

print("Random Forest classifier with SQRT and 5000  estimators\n")
print ("Prediction:",rf_predict,'\n')
print("Accuracy:", round((accuracy_score(Y_test, rf_predict)),4), "\n")
print("Report: \n", classification_report(Y_test, rf_predict))
print("Confusion Matrix: \n", confusion_matrix(Y_test, rf_predict), "\n")
```

✓ 1.5s Python

```
Random Forest classifier with SQRT and 5000  estimators

Prediction: ['ham' 'ham' 'spam' ... 'ham' 'ham' 'spam']

Accuracy: 0.9336

Report:
              precision    recall  f1-score   support

         ham       0.93      0.96      0.95      2182
        spam       0.94      0.89      0.91      1419

    accuracy                           0.93      3601
   macro avg       0.93      0.93      0.93      3601
weighted avg       0.93      0.93      0.93      3601

Confusion Matrix:
 [[2095   87]
 [ 152 1267]]
```

To compute the Decision Tree Classification, import the 'Decisiontreeclassification' model form sklearn library and tree class, store the model in variable and train the model and check the accuracy score using this model.

```
# Decision Tree classifier
#--------------------------
from sklearn.tree import DecisionTreeClassifier
#--------------------------

# Fit the training model to the desired classifier
decision_tree_clf = DecisionTreeClassifier()
decision_tree_clf = decision_tree_clf.fit(x_train, y_train)
dt_accuracy = decision_tree_clf.score(x_test, y_test)
print("Classification Accuracy: ", round((dt_accuracy),4), "\n")
```

✓ 0.4s Python

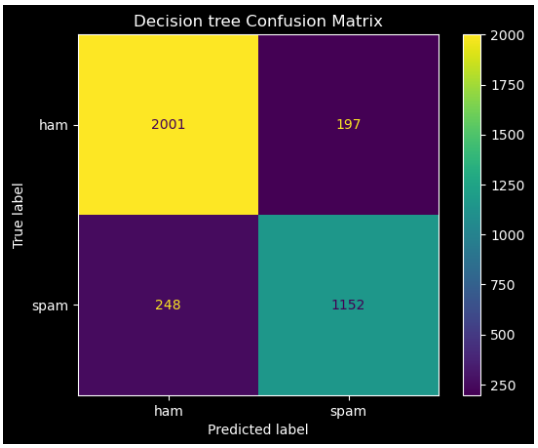
```
Classification Accuracy:  0.876
```

Preform the same functions to get the Classification report and confusion matrix as shown below.

```
# Metrics - Decision Tree classifier
dt_predict = decision_tree_clf.predict(x_test)
print("Decision Tree classifier\n")
print ("Prediction:",dt_predict,'\n')
print("Accuracy:", round((accuracy_score(y_test, dt_predict)),4), "\n")
print("Report: \n", classification_report(y_test, dt_predict))
print("Confusion Matrix: \n", confusion_matrix(y_test, dt_predict), "\n")
```

✓ 0.1s Python Python Python Python

```
Decision Tree classifier

Prediction: ['ham' 'ham' 'ham' ... 'ham' 'spam' 'spam']

Accuracy: 0.876

Report:
              precision    recall  f1-score   support

         ham       0.89      0.90      0.90      2175
        spam       0.85      0.84      0.84      1423

    accuracy                           0.88      3598
   macro avg       0.87      0.87      0.87      3598
weighted avg       0.88      0.88      0.88      3598

Confusion Matrix:
 [[1960  215]
 [ 231 1192]]
```

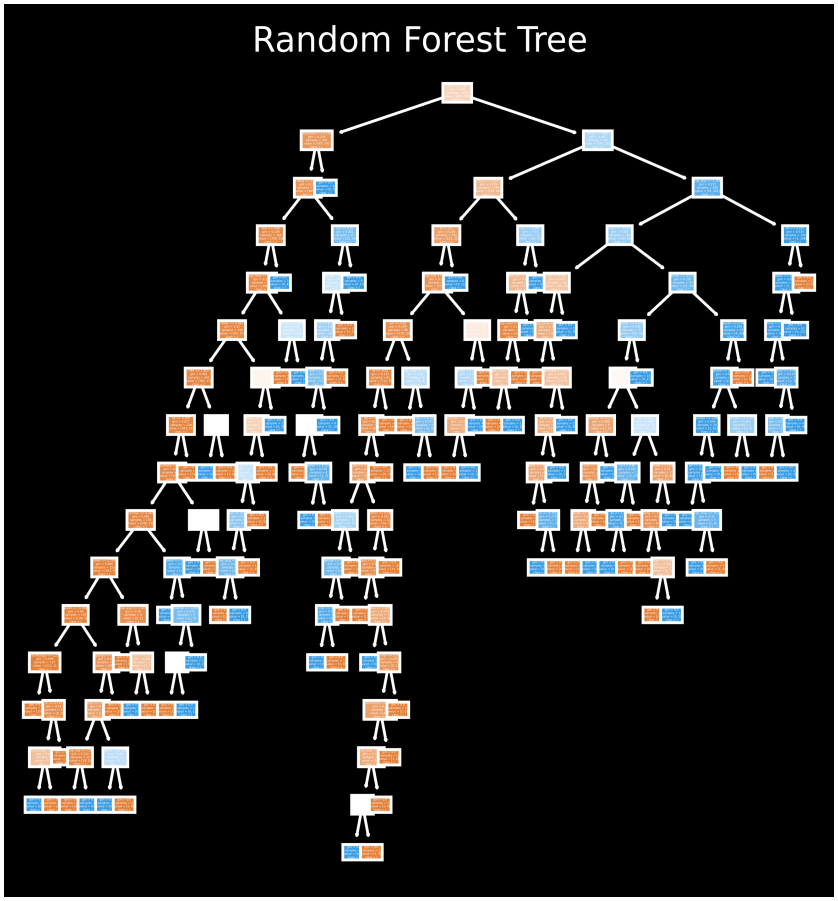


Reporting Task 2:

Compare of the results of all the classifiers (with the best possible parameter setting for each classifier). Use classification accuracy (# of instances correctly classified/total # of instances presented for classification), per class classification accuracy, and confusion matrix to compare the classifiers.

As demonstrated in the code, and in the table below, I obtained **Random Forest classifier** as the best classifier with maximum overall accuracy,

Classifier	Classification Accuracy	Per Class Classification Accuracy		Confusion Matrix
		Precision	Recall	
Decision Tree	0.876	Ham – 0.89 Spam – 0.85	Ham – 0.90 Spam – 0.84	array([[2001, 197], [248, 1152]])
Random Forest	0.9336	Ham – 0.93 Spam – 0.94	Ham – 0.96 Spam – 0.89	array([[2132, 66], [125, 1275]])



Decision Tree