VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

DIGITAL ASSIGNMENT 1

SPAM SMS CLASSIFIER

NAME: SYED AYAZ IMAM

REGISTRATION NUMBER: 18BCE0660

COURSE NAME: MACHINE LEARNING

COURSE CODE: CSE4020

FACULTY: NAGA RAJA G

PROBLEM STATEMENT

To develop a classifier that could classify SMS as SPAM or NOT SPAM using various algorithms as comparing the accuracy.

PROBLEM DESCRIPTION

An average smartphone user in India receives at least 5 spam messages daily and no phone manufacturer/carrier has taken measures to address this issue.This model could be implemented in text messaging app for android/iOS and thus all incoming messages will be classified between spam and not spam. If spam, the message will go in the spam section and user will get a silent notification only.
The app could sort spam messages into a different folder and as a result the user will not be disturbed due to frequent spam messages.

DATASET DESCRIPTION

- The dataset is chosen from UCI Machine Learning Repository at *archive.ics.uci.edu*
- The dataset consists of 5573 columns and two columns.
- The first column has the classification whether the SMS is spam or not as ham/spam.
- The second column has the SMS in string form.

MACHINE LEARNING TECHNIQUES USED

The machine learning algorithms used for implementation and comparative analysis are:

- Logistic Regression
- Support Vector Classifier
- Decision Tree Classifier
- K Neighbors Classifier
- Random Forest Classifier

APPROACH:

To predict whether an SMS is SPAM or NOT SPAM:
1. Using natural language preprocessing tools, the string of SMS is formatted.
2. The dataset will be split into test and training set.
3. The training data will be used to train various classifiers.
4. The test data is fed to check the accuracy score.
5. Comparative analysis is done for all the algorithms.

SYED AYAZ IMAM (18BCE0660)

In [15]:
```
#IMPORTING LIBRARIES
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from nltk.stem import SnowballStemmer
from nltk.corpus import stopwords
%matplotlib inline
```

In [16]:
```
#Reading data from csv
sms = pd.read_csv('spam.csv', encoding='latin-1')
sms.head()
```

Out[16]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [17]:
```
#Dropping unnamed columns and renaming
sms = sms.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
sms = sms.rename(columns = {'v1':'label','v2':'message'})
```

In [18]:
```
#Text processing
text_feat = sms['message'].copy()
text_feat.head()
```

Out[18]:
```
0    Go until jurong point, crazy.. Available only ...
1                        Ok lar... Joking wif u oni...
2    Free entry in 2 a wkly comp to win FA Cup fina...
3    U dun say so early hor... U c already then say...
4    Nah I don't think he goes to usf, he lives aro...
Name: message, dtype: object
```

In [19]:
```
#Defining text processing function and processing the messages
def text_process(text):

    text = text.translate(str.maketrans('', '', string.punctuation))
    text = [word for word in text.split() if word.lower() not in stopwords.wor

    return " ".join(text)

text_feat = text_feat.apply(text_process)
vectorizer = TfidfVectorizer("english")
features = vectorizer.fit_transform(text_feat)
```

In [20]:
```
#Splitting the data into training and test sets
features_train, features_test, labels_train, labels_test = train_test_split(fe
                                                                             te
```

In [21]:
```
#Importing classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [22]:
```
#Configuring the classifiers
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier(n_neighbors=49)
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=31, random_state=111)
```

In [23]:
```
#Assigning key values to classifier
clfs = {'SVC' : svc,'KN' : knc, 'DT': dtc, 'LR': lrc, 'RF': rfc}
```

In [24]:
```
#Defining a function to fit the classifiers to the data
def train_classifier(clf, feature_train, labels_train):
    clf.fit(feature_train, labels_train)
```

In [25]:
```
#Defining a function to predict from the given features
def predict_labels(clf, features):
    return (clf.predict(features))
```

In [26]:
```
#Iterating through the classifiers
pred_scores = []
for k,v in clfs.items():
    train_classifier(v, features_train, labels_train)
    pred = predict_labels(v,features_test)
    pred_scores.append((k, [accuracy_score(labels_test,pred)]))
```

In [27]:
```
#Making a dataframe of all the accuracy scores
df = pd.DataFrame.from_items(pred_scores,orient='index', columns=['Score'])
df
```

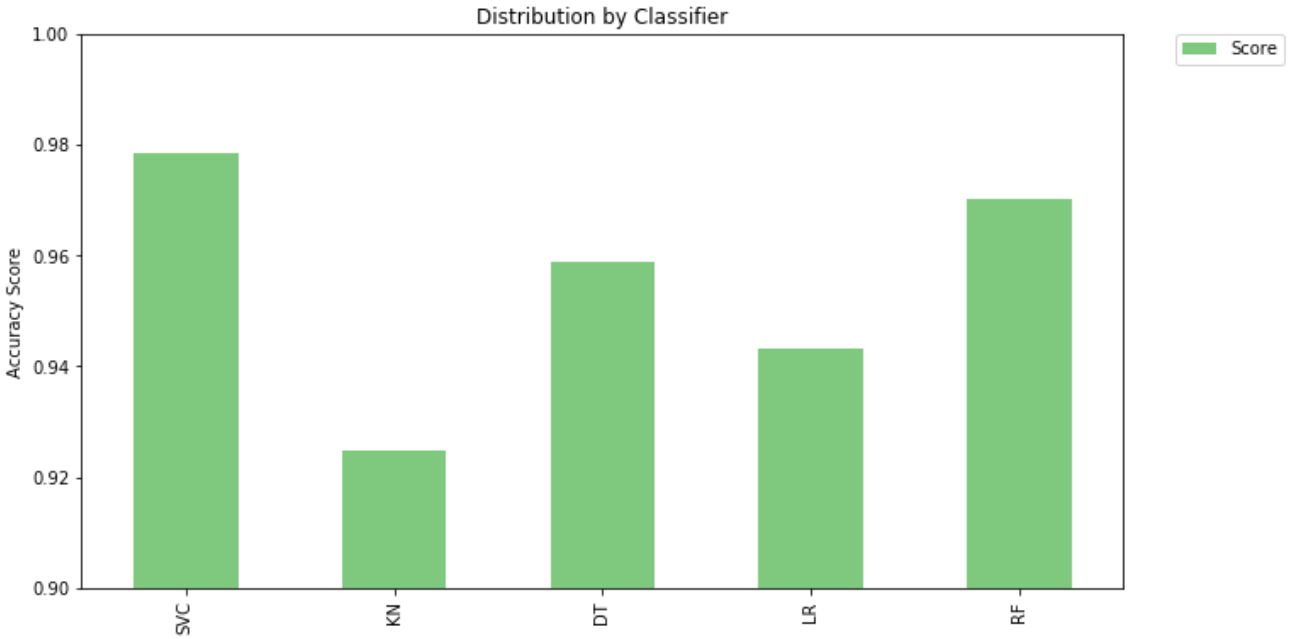
```
/home/ayaz/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: Fut
ureWarning: from_items is deprecated. Please use DataFrame.from_dict(dict(ite
ms), ...) instead. DataFrame.from_dict(OrderedDict(items)) may be used to pre
serve the key order.
```

Out[27]:

	Score
SVC	0.978469
KN	0.924641
DT	0.958732
LR	0.943182
RF	0.970096

In [28]:
```
#Plotting the accuracies on bar graph
df.plot(kind='bar', ylim=(0.9,1.0), figsize=(11,6), align='center', colormap='
plt.xticks(np.arange(5), df.index)
plt.ylabel('Accuracy Score')
plt.title('Distribution by Classifier')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[28]: <matplotlib.legend.Legend at 0x7fd79d72e8d0>



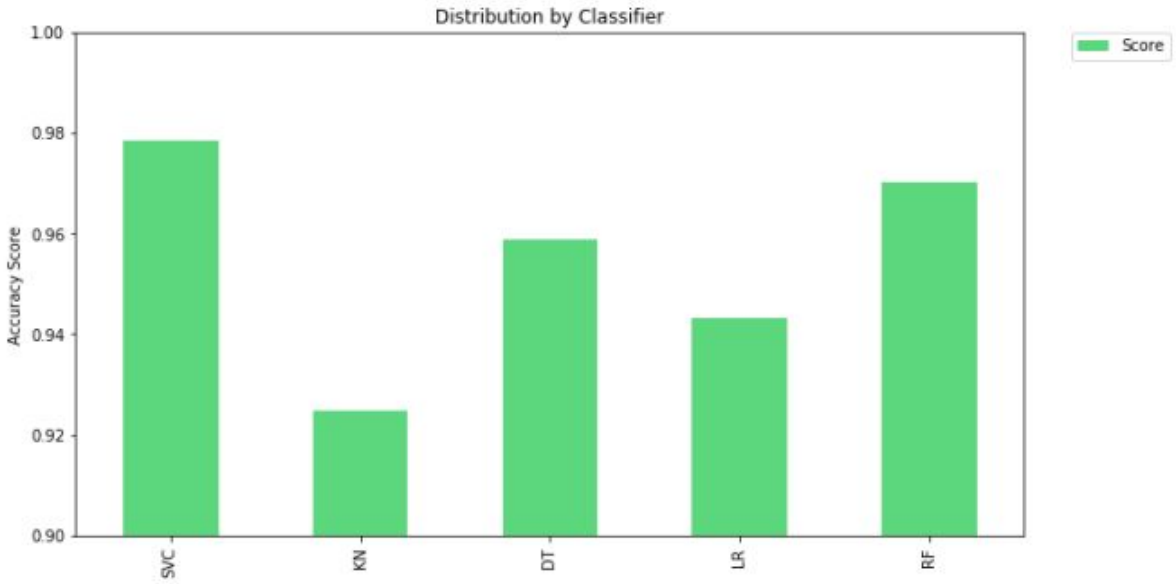
In [ ]:

/

RESULTS

ALGORITHM	ACCURACY SCORE
SVC	0.978469
KN	0.924641
LR	0.943182
DT	0.958732
RF	0.970096

CONCLUSION



From the above bar-graph, it is evident that:

1. Support Vector Classifier gives the best accuracy.

2. KNN gives least accuracy(still over 90%).
3. Support Vector Classifier and Random Forest Classifier have similar accuracy scores.