

# 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.words('english')]

    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

```

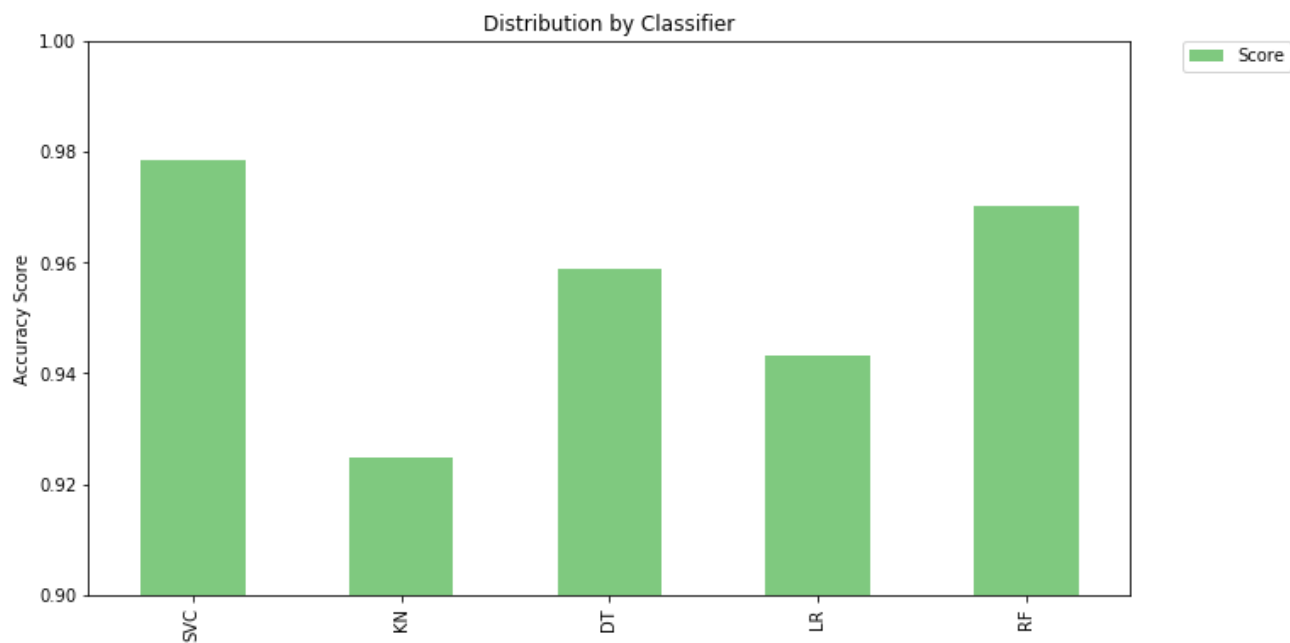
/home/ayaz/anaconda3/lib/python3.7/site-packages/ipykernel\_launcher.py:2: FutureWarning: from\_items is deprecated. Please use DataFrame.from\_dict(dict(items), ...) instead. DataFrame.from\_dict(OrderedDict(items)) may be used to preserve the key order.

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 [ ]: