|  |  |
| --- | --- |
| Project name | Building a Smarter AI-Powered Spam Classifier |
| Team ID | Proj\_212177\_team\_1 |
| Date | 25-10-2023 |
| Maximum mark |  |

**Introduction :**

Model Training: The model is trained using the preprocessed data. During training, the model learns to distinguish between spam and legitimate messages by optimizing a defined objective function.

Evaluation: Evaluating the model's performance is a critical step. Common evaluation metrics for spam classification include accuracy, precision, recall, F1-score, and the ROC curve. It's also important to assess the model's robustness to handle imbalanced datasets and its ability to generalize to unseen data.

**Data set:**

Given data set is available in [SMS Spam Collection Dataset (kaggle.com)](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset) is used for our project.

**Program:**

**Import necessary libraries**

In[0]:

import numpy

import pandas as pd

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import re

import nltk

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import wordcloud

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.model\_selection

import cross\_val\_score

from matplotlib.colors import ListedColormap

from sklearn.metrics import precision\_score, recall\_score, plot\_confusion\_matrix, classification\_report, accuracy\_score, f1\_score from sklearn import metrics

**Load the Dataset:**

In[2]:

df=pd.read\_csv("spam.csv")

In[3]:

y = data["Target"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

In[4]:

classifiers = [MultinomialNB(),

RandomForestClassifier(),

KNeighborsClassifier(),

SVC()]

for cls in classifiers:

cls.fit(X\_train, y\_train)

pipe\_dict = {0: "NaiveBayes", 1: "RandomForest", 2: "KNeighbours",3: "SVC"}

In[4]:

for i, model in enumerate(classifiers):

cv\_score = cross\_val\_score(model, X\_train,y\_train,scoring="accuracy", cv=10)

print("%s: %f " % (pipe\_dict[i], cv\_score.mean()))

Out[4]:

NaiveBayes: 0.967552

RandomForest: 0.974537

KNeighbours: 0.911450

SVC: 0.974086

In[5]:

creating lists of varios scores

precision =[]

recall =[]

f1\_score = []

trainset\_accuracy = []

testset\_accuracy = []

for i in classifiers:

pred\_train = i.predict(X\_train)

pred\_test = i.predict(X\_test)

prec = metrics.precision\_score(y\_test, pred\_test)

recal = metrics.recall\_score(y\_test, pred\_test)

f1\_s = metrics.f1\_score(y\_test, pred\_test)

train\_accuracy = model.score(X\_train,y\_train)

test\_accuracy = model.score(X\_test,y\_test)

precision.append(prec)

recall.append(recal)

f1\_score.append(f1\_s)

trainset\_accuracy.append(train\_accuracy)

testset\_accuracy.append(test\_accuracy)

In[6]:

data = {'Precision':precision,

'Recall':recall,'F1score':f1\_score,'Accuracy on Testset':testset\_accuracy,'Accuracy on Trainset':trainset\_accuracy}

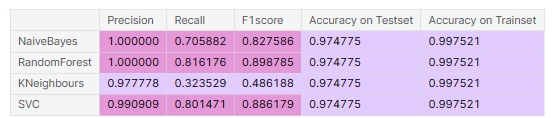
Results = pd.DataFrame(data, index =["NaiveBayes", "RandomForest", "KNeighbours","SVC"])

In[7]:

cmap2 = ListedColormap(["#E2CCFF","#E598D8"])

Results.style.background\_gradient(cmap=cmap2)

Out[7]:



In[8]:

cmap = ListedColormap(["#E1F16B", "#E598D8"])

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,10))

for cls, ax in zip(classifiers, axes.flatten()):

plot\_confusion\_matrix(cls,

X\_test,

y\_test,

ax=ax,

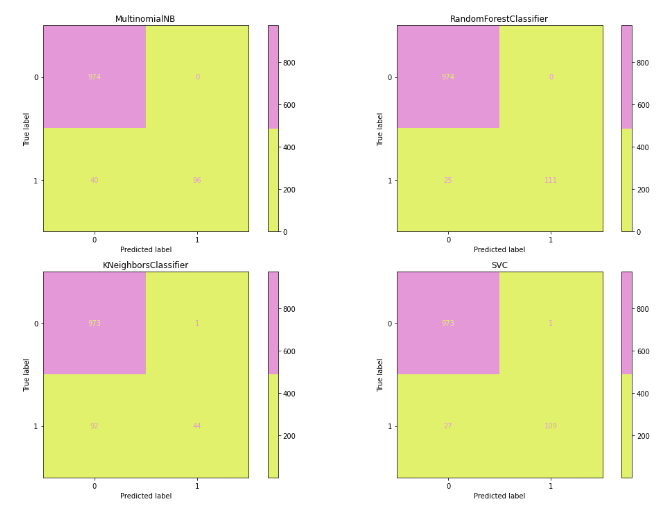
cmap= cmap, )

ax.title.set\_text(type(cls).\_\_name\_\_)

plt.tight\_layout()

plt.show()

Out[8]:



**Explanation:**

Steps in model building:

* Setting up features and target as X and y
* Splitting the testing and training sets
* Build a pipeline of model for four differentclassifiers.
  + Naïve Bayes
  + RandomForestClassifier
  + KNeighborsClassifier
  + Support Vector Machines
* Fit all the models on training data
* Get the cross-validation on the training set for all the models for accuracy

**Testing the models:**

Accuracy Report:

An accuracy report is a document or summary that provides information about the performance of a model, system, or process in terms of accuracy.

Confusion Matrix:

A confusion matrix is a table used in machine learning and statistics to describe the performance of a classification model. It allows you to understand how well a model is classifying instances into different categories, such as "positive" and "negative" for binary classification or multiple classes in multiclass classification.