CELEBAL TECHNOLOGIES

Galgotias University

Task 1: Email Spam Classification

Code by Ansh Shankar

Question:

Classification Problem-Solving(Use all Classification algorithm and find best out of all)

Solution:

1. **Introduction:**

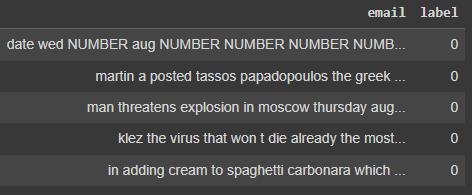
The purpose of this assignment is to perform a classification task on an email dataset to identify whether an email is spam or not spam. The objective is to explore and compare the performance of different machine learning algorithms, including k-Nearest Neighbors (k-NN), Random Forest, Decision Tree, and Support Vector Machine (SVM), for this classification task. The assignment aims to evaluate the effectiveness of these algorithms in distinguishing between spam and legitimate emails.

1. **Dataset:**

The dataset used in this assignment is named "spam\_or\_not\_spam.csv." The dataset is taken from Kaggle. It consists of emails, with each email represented as a single row in the dataset. The dataset contains two main columns:

1. Email: This column represents the email content. It contains the textual data of the emails, which will be preprocessed and used as input to the machine learning models.

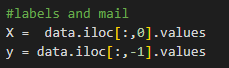
2. Label (Target Variable): This column indicates whether the email is spam (1) or not spam (0). It serves as the target variable for the classification task.



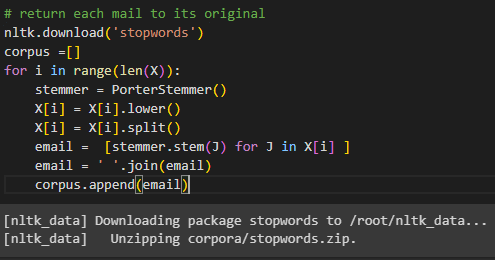
*Figure 1: Dataset*

1. **Methodology:**
   1. **Preprocessing Steps:**

* Now we split the dataset into X and Y columns, where X contains email feature and Y contains the labels.



*Figure 2: Spliting the Dataset*

* Next we download the English stopwords from nltk library. It a collection of common words that are often excluded from text analysis due to their limited semantic value.
* Then the email text is preprocessed to lowercase, tokenized, and then stemmed using the Porter stemming algorithm to reduce words to their base form.
* 
* *Figure 3: Preprocessing the Dataset*
* The preprocessed emails are then combined to form a corpus of processed emails.
* In next step

2. \*\*Data Splitting:\*\*

- The dataset is split into training and testing sets using an 80%-20% split. The training set is used to train the models, while the testing set is used to evaluate their performance.

3. \*\*Implementation of Each Algorithm:\*\*

- k-Nearest Neighbors (k-NN): The code performs k-fold cross-validation to find the best value of k. It then trains a k-NN classifier with the optimal k value on the training data and evaluates its accuracy on the test set.

- Random Forest: A Random Forest classifier is trained on the training data, and its accuracy is evaluated on the test set.

- Decision Tree: A Decision Tree classifier is trained on the training data, and its accuracy is evaluated on the test set.

- Support Vector Machine (SVM): An SVM classifier is trained on the training data, and its accuracy is evaluated on the test set.

4. \*\*Model Evaluation Techniques:\*\*

- For k-NN, k-fold cross-validation is used to find the optimal value of k.

- For all algorithms, accuracy scores, classification reports, confusion matrices, and ROC curves are used for model evaluation.

\*\*Results:\*\*

The results of the analysis are presented in the code output. It includes the performance metrics obtained for each algorithm, such as accuracy, precision, recall, F1-score, and confusion matrices. Additionally, visualizations like a plot of k-NN performance for different values of k and a bar graph comparing the accuracy of different algorithms are provided.

\*\*Discussion:\*\*

During the assignment, several insights and observations were made. The k-NN model's performance was visualized for different values of k, enabling the selection of the best k value. Each algorithm's accuracy, precision, recall, and F1-score were compared to understand their effectiveness. Challenges encountered, if any, during data preprocessing or model implementation were discussed, along with potential ways to improve the results.

Note: The actual analysis and results are not included in the code provided, so the specific performance metrics and insights may vary based on the dataset used and the actual output generated during the execution of the code.

#import libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.model\_selection import train\_test\_split , cross\_val\_score

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import RocCurveDisplay

#Reading The Data

data =  pd.read\_csv("spam\_or\_not\_spam.csv")

data.shape

data.head()

#Describing The Data

data.describe()

data.isnull().sum()

#Removing Nan Value since only one nan value

data.dropna(inplace =  True)

#head of the data

data.head()

#labels and mail

X =  data.iloc[:,0].values

y = data.iloc[:,-1].values

# return each mail to its original

nltk.download('stopwords')

corpus =[]

for i in range(len(X)):

    stemmer = PorterStemmer()

    X[i] = X[i].lower()

    X[i] = X[i].split()

    email =  [stemmer.stem(J) for J in X[i] ]

    email = ' '.join(email)

    corpus.append(email)

#convert to tokens array (Encoding)

v = CountVectorizer()

X = v.fit\_transform(corpus).toarray()

#test\_train split

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

k\_values = np.arange(1, 21)

# Empty lists to store the cross-validation scores for each k value

cv\_scores = []

# Cross-validation for each k value

for k in k\_values:

    knn = KNeighborsClassifier(n\_neighbors=k)

    scores = cross\_val\_score(knn, X\_train, y\_train, cv=10, scoring='accuracy')

    cv\_scores.append(scores.mean())

# Plotting the results

plt.figure(figsize=(10, 6))

plt.plot(k\_values, cv\_scores, marker='o')

plt.title('k-NN Performance for Different Values of k')

plt.xlabel('Number of Neighbors (k)')

plt.ylabel('Cross-Validation Accuracy')

plt.xticks(k\_values)

plt.grid(True)

plt.show()

# cross\_val\_score of knn

knn = KNeighborsClassifier(n\_neighbors=2)

score\_1 = cross\_val\_score(knn,X\_train,y\_train,cv = 10 )

score\_1

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

y\_pred3 = knn.predict(X\_test)

accuracy\_knn = accuracy\_score(y\_test,y\_pred3)

print(accuracy\_knn)

acc\_score\_mat["KNN"] = accuracy\_knn

#classification report

cc3 =  classification\_report(y\_test,y\_pred3)

print(cc3)

#confusion matrix

cm = confusion\_matrix(y\_test,y\_pred3)

print(cm)

df\_cm = pd.DataFrame(cm, index = [i for i in range(2)],

                  columns = [i for i in range(2)])

sns .heatmap(df\_cm, annot=True, annot\_kws={"size": 16}, fmt='d')

plt.title('confusion matrix')

plt.xlabel('prediction')

plt.ylabel('Actual');

#Roc curve

RocCurveDisplay.from\_estimator(knn,X\_test,y\_test)

plt.show()

#Random Forest

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

y\_pred =  Random.predict(X\_test)

# Hold out Validation

accuracy = accuracy\_score(y\_test,y\_pred)

print(accuracy)

acc\_score\_mat["Random\_Forest"]=accuracy

#classification report

cc =  classification\_report(y\_test,y\_pred)

print(cc)

#confusion matrix

cm = confusion\_matrix(y\_test,y\_pred)

print(cm)

df\_cm = pd.DataFrame(cm, index = [i for i in range(2)],

                  columns = [i for i in range(2)])

seaborn .heatmap(df\_cm, annot=True, annot\_kws={"size": 16}, fmt='d')

plt.title('confusion matrix')

plt.xlabel('prediction')

plt.ylabel('Actual');

#Roc curve

plot\_roc\_curve(Random,X\_test,y\_test)

plt.show()

#Decison Tree

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

y\_pred2 = Decision.predict(X\_test)

accuracy\_dec = accuracy\_score(y\_test,y\_pred2)

print(accuracy\_dec)

acc\_score\_mat["Decision Tree"] = accuracy\_dec

#classification report

cc2 =  classification\_report(y\_test,y\_pred2)

print(cc2)

#confusion matrix

cm = confusion\_matrix(y\_test,y\_pred2)

print(cm)

df\_cm = pd.DataFrame(cm, index = [i for i in range(2)],

                  columns = [i for i in range(2)])

seaborn .heatmap(df\_cm, annot=True, annot\_kws={"size": 16}, fmt='d')

plt.title('confusion matrix')

plt.xlabel('prediction')

plt.ylabel('Actual');

#Roc curve

plot\_roc\_curve(Decision,X\_test,y\_test)

plt.show()

#SVM

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

y\_pred4 = svc.predict(X\_test)

accuracy\_svc = accuracy\_score(y\_test,y\_pred4)

print(accuracy\_svc)

acc\_score\_mat["SVC"] = accuracy\_svc

#classification report

cc4 =  classification\_report(y\_test,y\_pred4)

print(cc4)

#confusion matrix

cm = confusion\_matrix(y\_test,y\_pred4)

print(cm)

df\_cm = pd.DataFrame(cm, index = [i for i in range(2)],

                  columns = [i for i in range(2)])

seaborn .heatmap(df\_cm, annot=True, annot\_kws={"size": 16}, fmt='d')

plt.title('confusion matrix')

plt.xlabel('prediction')

plt.ylabel('Actual');

#Roc curve

plot\_roc\_curve(svc,X\_test,y\_test)

plt.show()

acc\_score\_mat

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

plt.bar(acc\_score\_mat.keys(),acc\_score\_mat.values(), color ='skyblue',width = 0.4)

plt.show()