72017912C
COMPBEB1326

GROUP B : ASSIGNMENT No. 2

Page No.
Date

Title: Email Classification.

Objective: To classify the email using the binary classification method.

Problem Statement:

Classify the email using the binary classification method. Email Spam detection has two states : a) Normal State - Not Spam, b) Abnormal State - Spam. Use K-Nearest Neighbours & Support Vector Machine for classification. Analyze their performance.

Software & Hardware Requirements :

1. Desktop / Laptop
2. Any Operating System
3. Python & Required Libraries
4. Jupyter Notebook.

Theory :

Classification :

Classification is process of categorizing a given set of data into classes. It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

The classification predictive modelling is the task of approximating the mapping function from input variables to discrete output variables. The main goal is to identify which class / category the new data will fall into.

Page No.
Date

Types of Classification Algorithms:

1) Logistic Regression
2) Naive Bayes
3) K-Nearest Neighbours
4) Decision Tree
5) Support Vector Machines.

K-Nearest Neighbours (KNN):

1) It is based on supervised learning technique.
2) It assumes the similarity between the new data & available data & put the new data into the category that is most similar to the available categories.
3) KNN algorithm stores all the available data & classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K NN algorithm.
4) KNN algorithm can be used for Regression as well as for classification but mostly it is used for classification problems.
5) KNN is a non-parametric algorithm, which means it does not make any assumptions on underlying data.
6) It is also called as lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset & at the time of classification, it performs an action on the dataset.

Algorithm:

Step 1: Select the number K of the neighbours.
Step 2: Calculate the Euclidean distance of K number of

neighbours.

Step 3: Take the K nearest neighbours as per the calculated Euclidean distance.

Step 4: Among these K neighbours, count the numbers of the data points in each category.

Step 5: Assign the new data points to that category for which the numbers of neighbour is maximum.

Step 6: Model is ready.

Support Vector Machine (SVM):

Support vector machine or SVM is one of the most popular Supervised learning algorithms, which is used for Classification as well as Regression problems. However, primarily it is used for classification problems in Machine Learning.

The goal of SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision is called a hyperplane.

SVM chooses the extreme points/vectors that helps in creating the hyperplane. These extreme cases are called as support vectors, & hence algorithm is termed as support vector machine.

Page No.
Date

Types of SVM:

1) **Linear SVM:** Linear SVM is used for linearly seperable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly seperable data, & classifier is used called as Linear SVM classifier.

2) **Non-Linear SVM:** Non-linear SVM is used for non-linearly seperated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non linear data & classifier used is called as Non-linear SVM classifier.

Conclusion: Successfully classified emails using K-Nearest Neighbours & Support Vector Machine algorithm.

28/09/22

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
```

In [2]:
```
df=pd.read_csv('emails.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	milit
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	

5 rows × 3002 columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

In [6]:
```
df.dropna(inplace = True)
```

In [7]:
```
df.drop(['Email No.'],axis=1,inplace=True)
X = df.drop(['Prediction'],axis = 1)
y = df['Prediction']
```

In [8]:
```
from sklearn.preprocessing import scale
X = scale(X)
# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_stat
```

KNN classifier

In [9]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [10]:
```
print("Prediction",y_pred)
```

```
Prediction [0 0 1 ... 1 1 1]
```

In [11]:
```
print("KNN accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

```
KNN accuracy =  0.8009020618556701
```

In [12]:
```
print("Confusion matrix",metrics.confusion_matrix(y_test,y_pred))
```

```
Confusion matrix [[804 293]
 [ 16 439]]
```

SVM classifier

In [13]:
```
# cost C = 1
model = SVC(C = 1)

# fit
model.fit(X_train, y_train)

# predict
y_pred = model.predict(X_test)
```

In [14]:
```
metrics.confusion_matrix(y_true=y_test, y_pred=y_pred)
```

Out[14]:
```
array([[1091,    6],
       [  90,  365]], dtype=int64)
```

In [15]:
```
print("SVM accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

```
SVM accuracy =  0.9381443298969072
```