**GROUP – B**

**Assignment No: 2**

====================================================================

**Title:-** 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 Neighbors and

Support Vector Machine for classification. Analyze their performance.

=====================================================================**Objective:-**

-To learn about methods of Binary Classification.

-To understand about KNN and SVM.

=====================================================================

**Theory:-**

* In this practical, In a more digitizing world, building and constructing models that are designed to separate data into categories has become a relevant design space for different models and algorithms that separates emails into two different categories: (1)Normal state – not spam (2) Abnormal state – Spam. This will be done by first constructing a dataset reflecting real world data. The data collected will be used to train and tune the different machine learning algorithms. The 2 different methods were: k-nearest neighbors (KNN) and support vector machine (SVM).
* **Classification**

In machine learning, Classification, as the name suggests, classifies data into different parts/classes/groups. It is used to predict from which dataset the input data belongs to. Classification is the process of assigning new input variables (X) to the class they most likely belong to, based on a classification model, as constructed from previously labeled training data. Data with labels is used to train a classifier such that it can perform well on data without labels (not yet labeled). This process of continuous classification, of previously known classes, trains a machine. If the classes are discrete, it can be difficult to perform classification tasks.

Types of Classification

There are two types of classifications: (1) Binary classification (2) Multi-class classification

* **Binary Classification**

It is a process or task of classification, in which a given data is being classified into two classes. It’s basically a kind of prediction about which of two groups the thing belongs to. Binary classification uses some algorithms to do the task, some of the most common algorithms used by binary classification are:

1. Logistic Regression
2. k-Nearest Neighbors
3. Decision Trees
4. Support Vector Machine
5. Naive Bayes

Term Related to binary classification:

1. Precision - Precision in binary classification (Yes/No) refers to a model's ability to correctly interpret positive observations. In other words, how often does a positive value forecast turn out to be correct? We may manipulate this metric by only returning positive for the single observation in which we have the most confidence.
2. Recall- The recall is also known as sensitivity. In binary classification (Yes/No) recall is used to measure how “sensitive” the classifier is to detecting positive cases. To put it another way, how many real findings did we “catch” in our sample? We may manipulate this metric by classifying both results as positive.
3. F1 Score- The F1 score can be thought of as a weighted average of precision and recall, with the best value being 1 and the worst being 0. Precision and recall also make an equal contribution to the F1 ranking.

* **Dataset Description**

The csv file contains 5172 rows, each row for each email. There are 3002 columns. The first column indicates Email name. The name has been set with numbers and not recipients' name to protect privacy. The last column has the labels for prediction: 1 for spam, 0 for not spam. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding the non-alphabetical characters/words. For each row, the count of each word (column) in that email (row) is stored in the respective cells. Thus, information regarding all 5172 emails are stored in a compact dataframe rather than as separate text files.

* Code Explanation:

*import pandas as pd*

*df= pd.read\_csv('emails.csv')*

*df.shape*

*df.head()*

Import all libraries required for email classification and Read values from dataset with the help of pandas library.

*#input data*

*x=df.drop(['Email No.','Prediction'],axis=1)*

*#output data*

*y=df['Prediction']*

In pandas, drop( ) function is used to remove column(s).axis=1 tells Python that you want to apply function on columns instead of rows.

*x.shape*

*x.dtypes*

*set(x.dtypes)*

NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements. A data type object (an instance of numpy.dtype class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted.

import seaborn as sns

*sns.countplot(x=y)*

*y.value\_counts()*

The Seaborn.countplot() method is used to display the count of categorical observations in each bin in the dataset. The countplot() method takes input data in many forms, such as wide-form data, long-from data, arrays or a list of vectors. The value\_counts() function is used to get a Series containing counts of unique values.

*#feature scaling*

*from sklearn.preprocessing import MinMaxScaler*

*scaler = MinMaxScaler()*

*x\_scaled = scaler.fit\_transform(x)*

*x\_scaled*

Standardizes features by scaling each feature to a given range.This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one. MinMaxScaler scales the data to a fixed range, typically between 0 and 1. The `.fit\_transform()` method is used to actually transform the data, this means that it both fits the scaler to the data and transforms it in one step.

*#cross validation*

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

*x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,random\_state = 0, test\_size=0.25)*

*x\_scaled.shape*

*x\_train.shape*

*x\_test.shape*

The train\_test\_split() method is used to split our data into train and test sets. First, we need to divide our data into features (X) and labels (y). The dataframe gets divided into X\_train, X\_test, y\_train, and y\_test. X\_train and y\_train sets are used for training and fitting the model. The X\_test and y\_test sets are used for testing the model if it’s predicting the right outputs/labels. train\_test\_split() performs the split and returns four sequences (in this case NumPy arrays) in this order:

1. x\_train: The training part of the first sequence (x)
2. x\_test: The test part of the first sequence (x)
3. y\_train: The training part of the second sequence (y)
4. y\_test: The test part of the second sequence (y)

* **KNN**

*#import the class*

*from sklearn.neighbors import KNeighborsClassifier*

*#create an object*

*knn = KNeighborsClassifier(n\_neighbors=5)*

*#Train an algorithm*

*knn.fit(x\_train,y\_train)*

*#Predict on test data*

*y\_pred = knn.predict(x\_test)*

*#import the evaluation metrics*

*from sklearn.metrics import ConfusionMatrixDisplay, accuracy\_score*

*from sklearn.metrics import classification\_report*

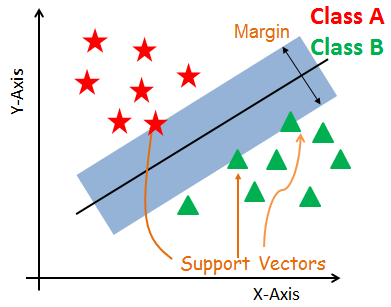
*accuracy\_score(y\_test,y\_pred)*

The kNN algorithm can be considered a voting system, where the majority class label determines the class label of a new data point among its nearest ‘k’ (where k is an integer) neighbors in the feature space. Imagine a small village with a few hundred residents, and you must decide which political party you should vote for. To do this, you might go to your nearest neighbors and ask which political party they support. If the majority of your’ k’ nearest neighbors support party A, then you would most likely also vote for party A. This is similar to how the kNN algorithm works, where the majority class label determines the class label of a new data point among its k nearest neighbors.

In program, a fixed value of 5 for k, but we’ll need to optimize this later on. first create an instance of the kNN model, then fit this to our training data then pass both the features and the target variable, so the model can learn. Predictions on the test dataset, which will be use later to score the model. To evaluate this model is by using accuracy. We check the predictions against the actual values in the test set and count up how many the models got right.

* **SVM**

Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.



*from sklearn.svm import SVC*

*#svm = SVC(kernel = 'linear')*

*#svm = SVC(kernel = 'poly')*

*svm = SVC(kernel = 'rbf')*

*svm.fit(x\_train,y\_train)*

*svm.fit(x\_train,y\_train)*

*y\_pred = svm.predict(x\_test)*

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

Import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function. Then, fit your model on train set using fit() and perform prediction on the test set using predict().Accuracy can be computed by comparing actual test set values and predicted values. There are different types of SVM kernels like –

1. Linear : A linear kernel is a type of kernel function used in machine learning, including in SVMs (Support Vector Machines). It is the simplest and most commonly used kernel function, and it defines the dot product between the input vectors in the original feature space.
2. Polynomial : A particular kind of kernel function utilised in machine learning, such as in SVMs, is a polynomial kernel (Support Vector Machines). It is a nonlinear kernel function that employs polynomial functions to transfer the input data into a higher-dimensional feature space.
3. Gaussian (RBF - radial basis function) : The Gaussian kernel, also known as the radial basis function (RBF) kernel, is a popular kernel function used in machine learning, particularly in SVMs (Support Vector Machines). It is a nonlinear kernel function that maps the input data into a higher-dimensional feature space using a Gaussian function.
4. Laplace : The Laplacian kernel, also known as the Laplace kernel or the exponential kernel, is a type of kernel function used in machine learning, including in SVMs (Support Vector Machines). It is a non-parametric kernel that can be used to measure the similarity or distance between two input feature vectors.

=====================================================================

**Conclusion:-**

Thus we have studied how to classify the email using the binary classification method.

=====================================================================