SPAM MAIL PREDICTION

Mails are of two types: spam and non-spam also known as ham mails.

1. Mail data
2. Data pre-processing
3. Train test split
4. New mail is given to the trained model and then we predict if it is spam or ham

For training we use logistic regression model because for binary classification (spam or ham) it is best preferred.

Step 1:

Importing the dependencies (importing the required libraries)-

1. import numpy as np

for creating numpy arrays

1. import pandas as pd

for using dataframes which are useful in making the data in structured manner

the csv files are made into a more structured table content with the help of data frames

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

the train\_test\_split function is useful for training and testing of data

1. from sklearn.feature\_extraction.text import Tfidfvectorizer

the tfidfvectorizer is useful in generation of feature vectors (numerical values)

this is done because the numerical data is easily understood by the machine than the text information

1. from sklearn.linear\_model import LogisticRegression

the training data is useful for training the logistic regression model

the logistic regression is best for binary classification

1. from sklearn.metrics import accuracy\_score

the test data is useful for prediction of the accuracy score

this accuracy score is useful for evaluating the model

Step 2:

Data collection and pre-processing

1. rawdata = pd.read\_csv(‘ path ‘)

loading the data from csv file to pandas dataframe

1. print(rawdata)

for printing the raw data

1. now we have to replace the null values (missing values) with a null string

maildata = rawdata.where((pd.notnull(rawdata)), ‘ ‘)

the where function is useful for a condition and here it helps in filling the missing values with and empty string (‘ ‘)

1. printing the first five rows

print(maildata.head())

printing the no. of rows and columns

print(maildata.shape)

o/p: (x,y) where x is the no. of rows and y is the no. of columns

We have labels for all mails i.e. ham or spam

Now we do **LABEL ENCODING**

WE ENCODE THE LABEL TO NUMERICAL VALUES

“REPLACE TEXT VALUE WITH NUMERICAL VALUE”

REPLACE HAM WITH 1

REPLACE SPAM WITH 0

Step 3:

1. Label encoding

maildata.loc[maildata[‘category’] == ‘spam’,’category’,] = 0

maildata.loc[maildata[‘category’] == ‘ham’,’category’,] = 1

1. Separating the data into to texts and labels

x = mailData[‘Message’]

y = mailData[‘Category’]

1. Separating the data into training data and test data

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state = 3)

The x\_train,x\_test,y\_train,y\_test are arrays which hold the mails as given by the train\_test\_split function

Here the test\_size tells us that the 20% of the data is used for testing purpose

The random\_state is optional

The train\_test\_split function works in different manner i.e., for each time of its execution it separates different mails into training and testing data

In order to make sure that it separates the same mails in all cases we can use the random\_state

The random\_state can hold any numerical value.

Step 4:

Feature Extraction

1. Transform the text data to feature vectors that can be used as input to the logistic regression model

feature\_extraction = TfidfVectorizer(min\_df = 1,stop\_words = ‘english’, lowercase=True)

Here we load the vectorizer which is used for generation of feature vectors into the feature\_extraction variable. The min\_df tells us that the least occurring words can be ignored. The stop\_words=’english’ makes sure that words such as is, did, are, as etc can be ignored as they can commonly exist in all mails so for them, we don’t need to create feature vectors. The lowercase converts all the words into lowercase for better understanding of data.

x\_train\_features = feature\_extraction.fit\_transform(x\_train)

A new variable x\_train\_features would be useful for storing the x\_train values (textual form) as numerical values i.e., as feature vectors. Here we use two functions fit and transform. Fit for fitting the vectorizer and transform for converting the text data to feature vectors.

x\_test\_features = feature\_extraction.transform(x\_test)

1. Convert the y\_train and y\_test values to integer type (by default they are of object type)

y\_train = y\_train.astype(‘int’)

y\_test = y\_test.astype(‘int’)

Step 5:

Training the model: Logistic Regression

model = LogisticRegression()

Train the logistic regression model with the training data

model.fit(x\_train\_features,y\_train)

Step 6:

Evaluating the trained model:

1. Prediction on training data-

prediction\_on\_training\_data = model.predict(x\_train\_features)

accuracy\_on\_training\_data = accuracy\_score(y\_train,prediction\_on\_training\_data)

print(“Accuracy on trained data:”, accuracy\_on\_training\_data)

1. Prediction on test data-

prediction\_on\_test\_data = model.predict(x\_test\_features)

accuracy\_on\_test\_data = accuracy\_score(y\_test,prediction\_on\_test\_data)

print(“Accuracy on test data:”, accuracy\_on\_test\_data)

Step 7:

A new mail is given as input and validated if it is spam or ham

Example:

input\_mail = [“mail\_data”]

input\_mail\_features = feature\_extraction.transform(input\_mail)

prediction = model.predict(input\_mail\_features)

print(prediction)

The prediction can have 2 values,0 or 1, and is a list.

if prediction==1:

print(“Ham mail”)

else:

print(“Spam mail”)

THE END