**Open Ended Experiment**

**Aim: Fraud Detection in SMS using:**

* **Naïve Bayes Method**
* **Logistic Regression**
* **Support Vector Machine**

**Theory:**

* **Fraud detection in SMS:** This experiment focuses mainly on the spam SMS everyone receives in their daily lives. Spam detection is the process of identifying and filtering unwanted and unsolicited SMS, also known as spam. Spam detection is a critical task in the field of information security, as it helps protect users from malicious content, phishing attacks and other threats.
* **Naïve Bayes Method:** It is a classification technique based on Bayes’ Theorem with an independence assumption among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’. An NB model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.
* **Logistic Regression:** Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables. In short, the logistic regression model computes a sum of the input features (in most cases, there is a bias term), and calculates the logistic of the result. The output of logistic regression is always between (0, and 1), which is suitable for a binary classification task. The higher the value, the higher the probability that the current sample is classified as class=1, and vice versa.
* **Support Vector Machine:** 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 the 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 boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

**Program:**

* **Naïve Bayes Method:**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import LabelEncoder

from sklearn.naive\_bayes import MultinomialNB

from sklearn.feature\_extraction.text import CountVectorizer

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

from sklearn.metrics import confusion\_matrix

data\_path = 'dataset.csv'

data\_frame = pd.read\_csv(data\_path)

print(data\_frame.head())

# Extract X and y from the data frame

X = data\_frame['Message'].tolist()

y = data\_frame['Category'].tolist()

# Initialize label encoder

label\_encoder = LabelEncoder()

y = label\_encoder.fit\_transform(y)

# Print the first 5 elements of X and y

print(f'X[:5]: \n{X[:5]}\n')

print(f'y[:5]: {y[:5]}\n')

print(f"Label Mapping : {label\_encoder.inverse\_transform(y[:5])}")

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

clf=Pipeline([ ('vectorizer',CountVectorizer()),

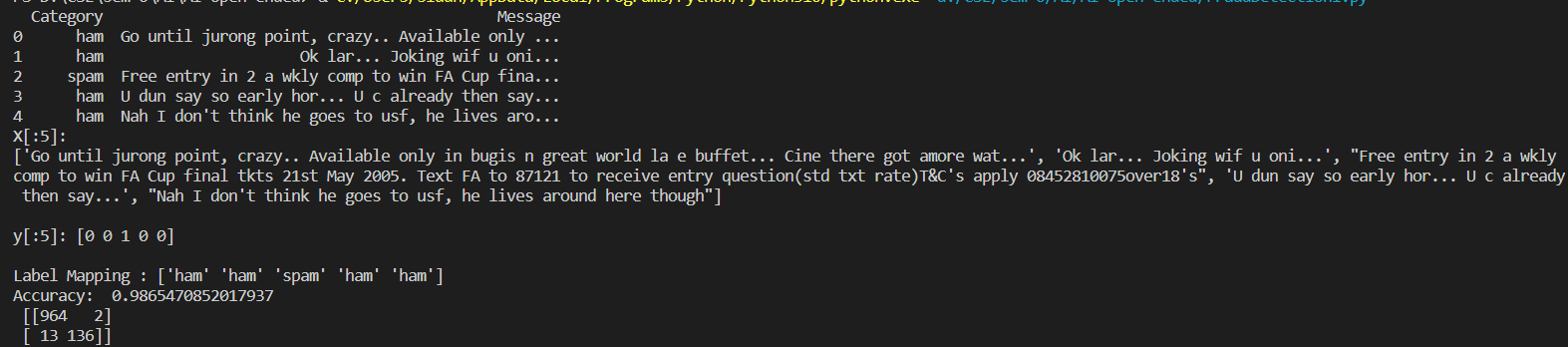
('nb',MultinomialNB())])

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

print("Accuracy: ",clf.score(X\_test,y\_test))

y\_pred=clf.predict(X\_test)

print("",confusion\_matrix(y\_test,y\_pred))

**O/P:** 

* **Logistic Regression**

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score,precision\_score

from sklearn.metrics import confusion\_matrix

# Specify the path to the SPAM text message dataset

data\_path = 'dataset.csv'

# Load the dataset using the load\_data function

df= pd.read\_csv(data\_path)

# Print the first five rows of the dataset

print(df.head())

vectorizer=CountVectorizer(stop\_words="english")

X=vectorizer.fit\_transform(df['Message'])

y=df['Category']

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

lr=LogisticRegression()

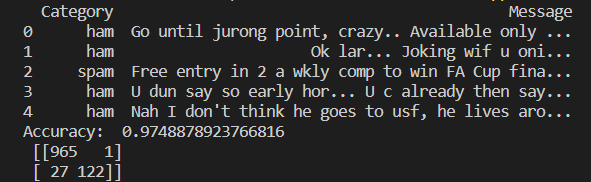
lr=lr.fit(X\_train,y\_train)

y\_pred=lr.predict(X\_test)

print("Accuracy: ",accuracy\_score(y\_test,y\_pred))

print("",confusion\_matrix(y\_test,y\_pred))

**O/P:**



* **Support Vector Machine**

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn.feature\_extraction.text import CountVectorizer

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

data\_path = 'dataset.csv'

# Load the dataset using the load\_data function

data\_frame = pd.read\_csv(data\_path)

# Print the first five rows of the dataset

print(data\_frame.head())

X = data\_frame['Message'].values

y = data\_frame['Category'].values

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

vectorizer = TfidfVectorizer()

X\_train=vectorizer.fit\_transform(X\_train)

X\_test=vectorizer.transform(X\_test)

svm=SVC()

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

print(svm.score(X\_test,y\_test))

y\_pred=svm.predict(X\_test)

print(confusion\_matrix(y\_test,y\_pred))

**O/P:**

