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:

```
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
X[:5]:
['Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...', 'Ok lar... Joking wif u oni...', "Free entry in 2 a wkly
comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's", 'U dun say so early hor... U c already
 then say...', "Nah I don't think he goes to usf, he lives around here though"]

y[:5]: [0 0 1 0 0]

Label Mapping : ['ham' 'ham' 'spam' 'ham' 'ham']
Accuracy:  0.9865470852017937
 [[964   2]
 [ 13 136]]
```

- **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:

```
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
Accuracy:  0.9748878923766816
 [[965   1]
 [ 27 122]]
```

- **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:

```
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
0.9811659192825112
[[965   1]
 [ 20 129]]
```