Ex 3:
Naive Bayes Classification on Iris Dataset

```
#Step 1: Importing the Libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd



#Step 2: Importing the dataset

dataset = pd.read_csv('Iris.csv')


X = dataset.iloc[:,:4].values

y = dataset['Species'].values

dataset.head(5)


#Step 3: Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)



#Step 4: Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)

#Step 5: Training the Naive Bayes Classification model on the Training Set

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X_train, y_train)


#Step 6: Predicting the Test set results

y_pred = classifier.predict(X_test)

y_pred


#Step 7: Confusion Matrix and Accuracy

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

from sklearn.metrics import accuracy_score

print ("Accuracy : ", accuracy_score(y_test, y_pred))

cm


#Step 8: Comparing the Real Values with Predicted Values

df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})

df
```

Support Vector Machine on Iris Dataset

```
# Imports

from sklearn.datasets import load_iris

from sklearn.svm import SVC
```

```
import pandas as pd

import numpy as np


# Load Data

iris = load_iris()


# Create a dataframe

df = pd.DataFrame(iris.data, columns = iris.feature_names)

df['target'] = iris.target


# Let's see a sample of created df

df.sample(frac=0.01)



# Let's see target names

targets = iris.target_names

print(targets)


# Prepare training data for building the model

X_train = df.drop(['target'], axis=1)

y_train = df['target']

# Instantiate the model

cls = SVC()

# Train/Fit the model

cls.fit(X_train, y_train)
```

```
# Make prediction using the model

X_pred = [5.1, 3.2, 1.5, 0.5]

y_pred = cls.predict([X_pred])

print("Prediction is: {}".format(targets[y_pred]))
```