
AMITY UNIVERSITY NOIDA

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

# ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

## PRACTICAL FILE

Made By-

Shourya Solanki

A2305223569

B.Tech 4CSE-3Y

Submitted To-

Dr Pintu Kumar Ram

# PRACTICAL – 1

AIM:

1) To read the Iris Dataset.
2) To find any missing values.
3) To find the average value of each column.
4) To find the maximum value in each column.
5) To find the minimum value in each column.

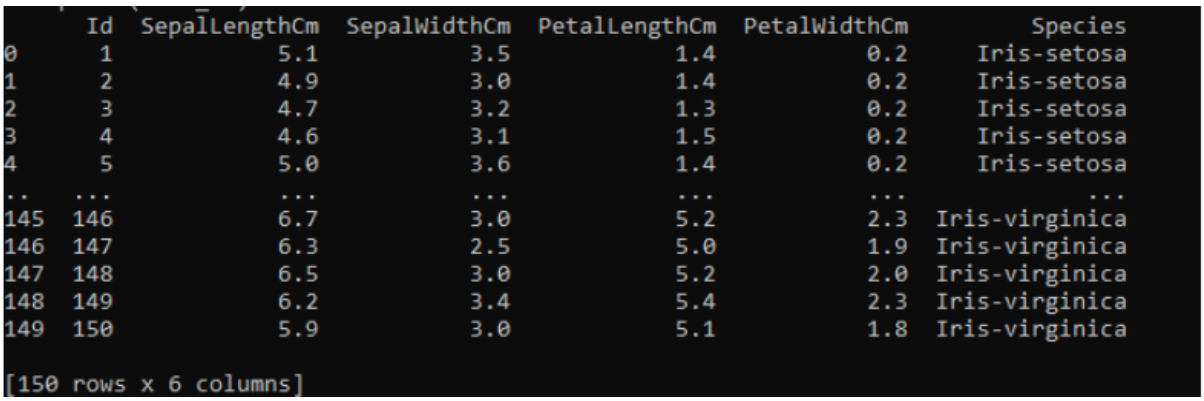
LIBRARIES IMPORTED: Pandas

INPUT:

1) Import pandas as pd

```
>>>
>>> #Correct file path to the CSV file
>>> file_path = r'C:\Users\Peter|Downloads\Iris.csv'
>>>
>>> #Read the entire CSV file into a DataFrame
>>> iris_df = pd.read_csv (file_path)
>>>
>>> #Display the entire dataset (if its not too large)
>>> print(iris_df)
```

OUTPUT:



| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| .. | ... | ... | ... | ... | ... | ... |
| 145 | 146 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| 146 | 147 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| 147 | 148 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| 148 | 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| 149 | 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

[150 rows x 6 columns]

2) import pandas as pd

>>>

>>> #File path to the CSV file

>>> file_path = r'C:\Users\Peter\Downloads\Iris.csv'

>>>

>>> #Read the CSV file into a DataFrame

>>> iris_df = pd.read_csv(file path)

>>>

>>> #Check for missing values in each column

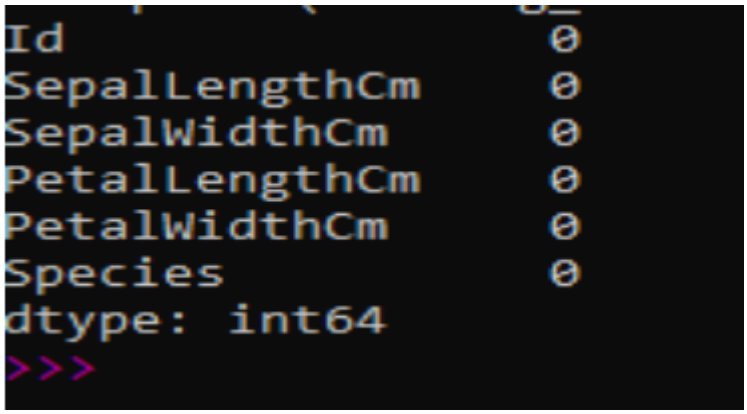
>>> missing_values = iris_df.isna().sum()

>>>

>>> #Display missing values count for each column

>>> print(missing_values)

OUTPUT:




3) # Select only numeric columns from the dataset

>>> numeric_columns = iris_df.select_dtypes(include=['float64', 'int64'])

>>>

>>> # Calculate the mean (average) of each numeric column

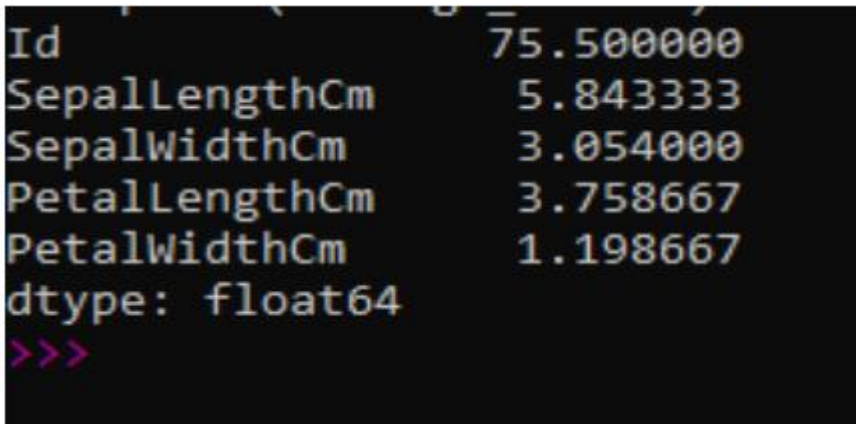
>>> average_values = numeric_columns.mean()

>>>

>>> # Display the average for each numeric column

>>> print(average_values)

OUTPUT:




4) # Select only numeric columns from the dataset

>>> numeric_columns = iris_df.select_dtypes(include=['float64', 'int64'])

>>>

>>> # Find the maximum value of each numeric column

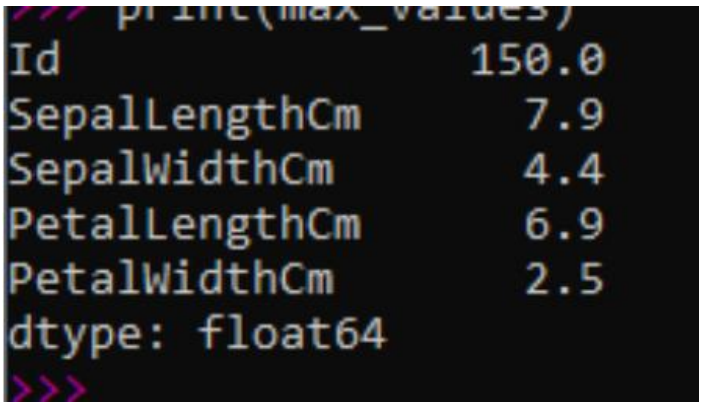
>>> max_values = numeric_columns.max()

>>>

>>> # Display the maximum value for each numeric column

>>> print(max_values)

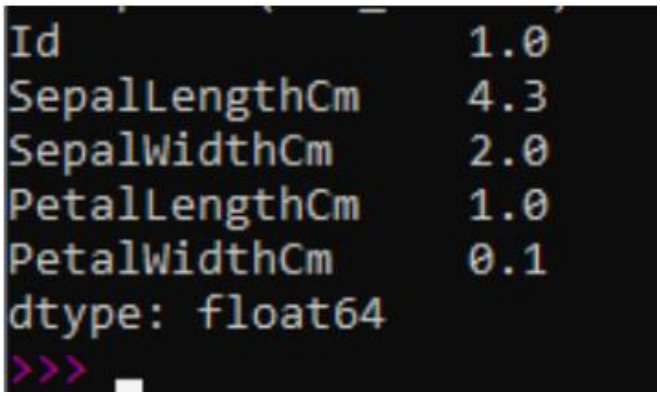
OUTPUT:




5) # Select only numeric columns from the dataset

```
>>> numeric_columns = iris_df.select_dtypes(include=['float64', 'int64'])
>>>
>>> # Find the minimum value of each numeric column
>>> min_values = numeric_columns.min()
>>>
>>> # Display the minimum value for each numeric column
>>> print(min_values)
```

OUTPUT:



# PRACTICAL – 2

AIM:

Implementing k-nearest neighbour algorithm in python.

INPUT:

```
Import matplotlib.pyplot as plt
From sklearn.neighbors import KNeighborsClassifier as knc
#dataset
x = [4, 5, 10, 4, 3, 11, 14, 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
# plt.plot(x, y, '*')
# plt.show()
printf('Plotting the given DataSet')
plt.scatter(x = x, y = y, c = classes, marker = '*')
plt.show()
data = list(zip(x, y)
printf('Making the Array')
print(data)
k = knc(n_neighbors = 3)
k.fit(data, classes)
new_x = 8
new_y = 21
np = [(new_x, new_y)]
class
prediction = k.predict(np)
printf('Predicting the new point class')
print(prediction)
```

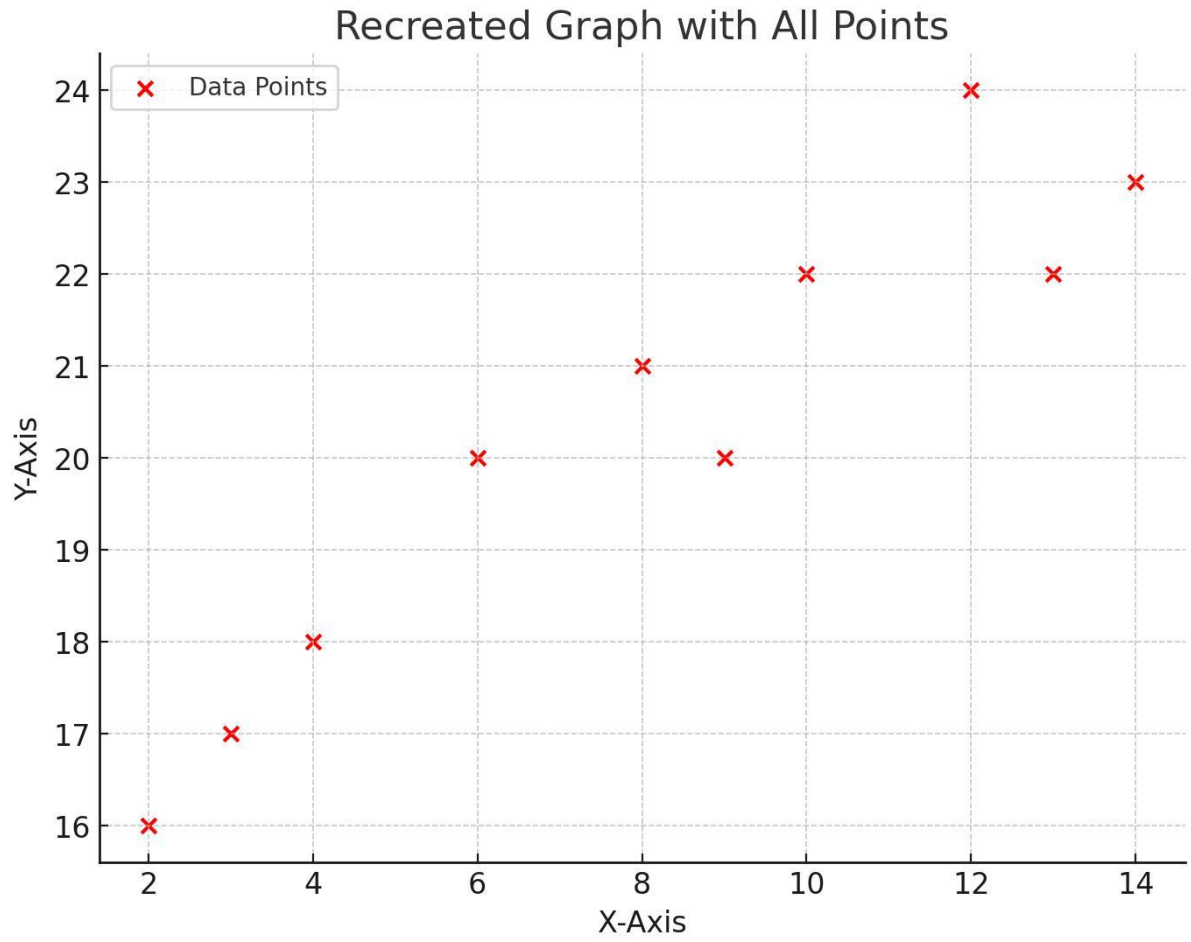
print('Plotting full data')

plt.scanner(x = x + [new_x], y = y + [new_y], c = classes

[prediction[0]], marker = '*')

plt.show()

OUTPUT:



## EXPERIMENT: 3

**AIM:** Implement K mean clustering algorithm.

**LIBRARIES IMPORTED**: Numpy, Sklearn

**INPUT:**

```
import numpy as np
from sklearn.cluster import KMeans

#data points using numpy
data = np.array([
    [2, 1], [3, 4], [2, 4], [8, 8], [4, 1], [4, 3],[5,6],
    [7, 8], [6, 5],[6,2],[8,1],[6,1],[5,2], [4,2],[2,5]
    ])

# Number of clusters (k)
k = 4

# Apply K-Means clustering
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
kmeans.fit(data)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

print("Final Centroids:") #printing centroids
print(centroids)

print("\nClusters Formed:")
for i in range(k):
    # Grouping data points by clusters
    print(f"Cluster {i}: {data[labels == i]}")
```

**OUTPUT SCREEN:**

```
Final Centroids:
[[3.8        1.8       ]
 [6.5        6.75      ]
 [2.33333333 4.33333333]
 [6.66666667 1.33333333]]

Clusters Formed:
Cluster 0: [[2 1]
 [4 1]
 [4 3]
 [5 2]
 [4 2]]
Cluster 1: [[8 8]
 [5 6]
 [7 8]
 [6 5]]
Cluster 2: [[3 4]
 [2 4]
 [2 5]]
Cluster 3: [[6 2]
 [8 1]
 [6 1]]
>>>
```

# PRACTICAL 5

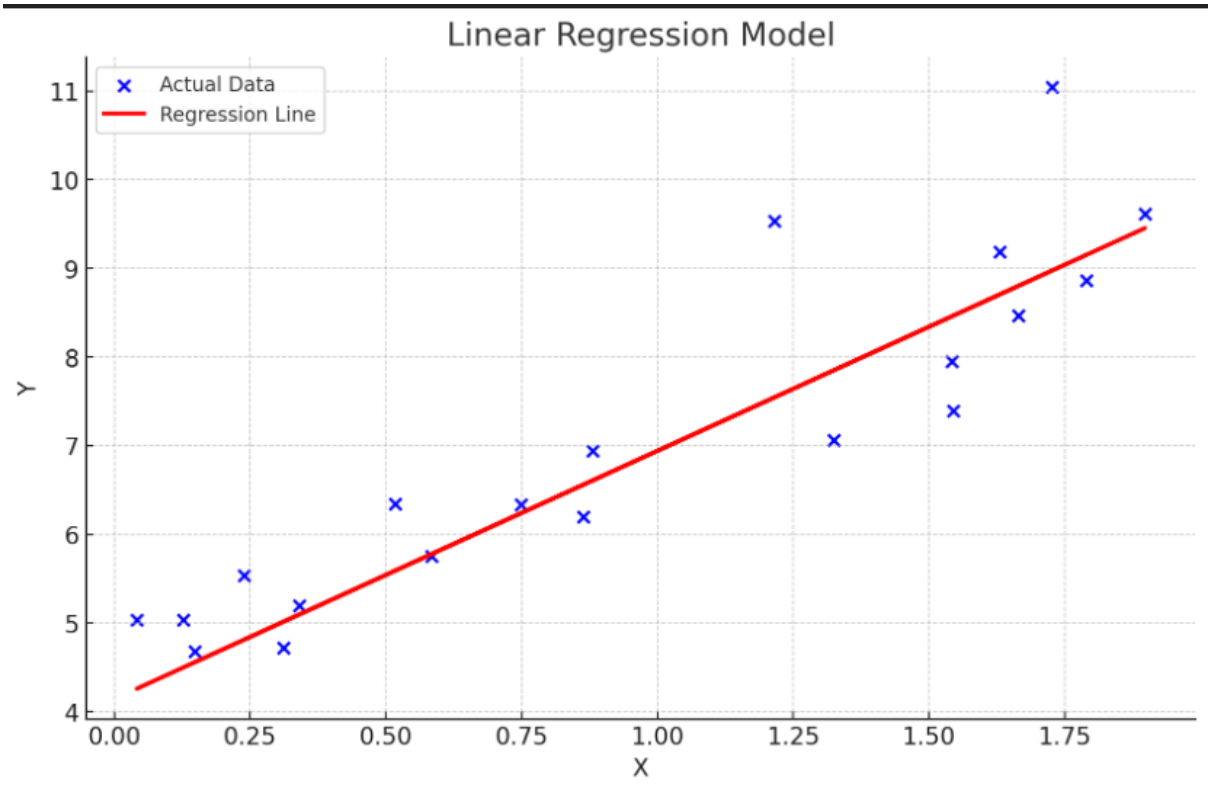
AIM: Linear Regression

LIBRARIES IMPORTED: Numpy, Sklearn

INPUT:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_errornp.random.seed(42)

x = 2 * np.random.rand(100, 1)

y = 4 + 3 * x + np.random.randn(100, 1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)

model = LinearRegression()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")

print(f"Intercept: {model.intercept_}, Coefficient: {model.coef_}")

plt.scatter(x_test, y_test, color='blue', label='Actual Data')

plt.plot(x_test, y_pred, color='red', linewidth=2, label='Regression Line')

plt.xlabel("X")

plt.ylabel("Y")

plt.legend()

plt.show()
```

OUTPUT:


Linear Regression Model
Actual Data
Regression Line
Y
X
11
10
9
8
7
6
5
4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

# PRACTICAL 6

AIM: Logistic Regression

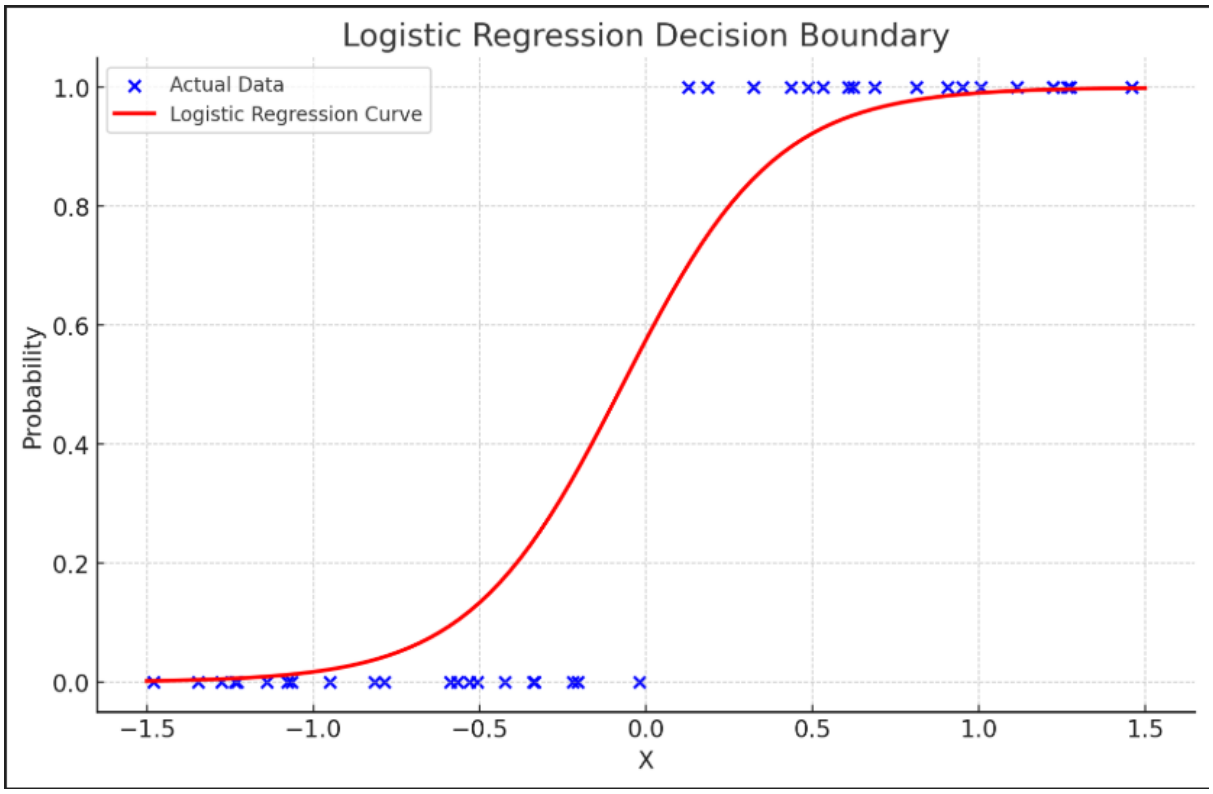
LIBRARIES IMPORTED: Numpy, Sklearn

INPUT:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

np.random.seed(42)

x = 3 * np.random.rand(200, 1) - 1.5  # Random values between -1.5 and
1.5

y = (x > 0).astype(int).ravel()  # Label: 1 if x > 0, else 0

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)

model = LogisticRegression()

model.fit(x_train, y_train)y_pred = model.predict(x_test)accuracy =
accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred)


print(f"Accuracy: {accuracy:.2f}")

print("Confusion Matrix:")

print(conf_matrix)

print("Classification Report:")

print(report)

x_values = np.linspace(-1.5, 1.5, 100).reshape(-1, 1)
```

```
y_prob = model.predict_proba(x_values)[:, 1]

plt.scatter(x_test, y_test, color='blue', label='Actual Data')

plt.plot(x_values, y_prob, color='red', linewidth=2, label='Logistic
Regression Curve')

plt.xlabel("X")

plt.ylabel("Probability")

plt.legend()

plt.title("Logistic Regression Decision Boundary")

plt.show()
```

OUTPUT:



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.95 | 0.98 | 21 |
| 1 | 0.95 | 1.00 | 0.97 | 19 |
| accuracy | | | 0.97 | 40 |
| macro avg | 0.97 | 0.98 | 0.97 | 40 |

# PRACTICAL 7

AIM: Support Vector Machine

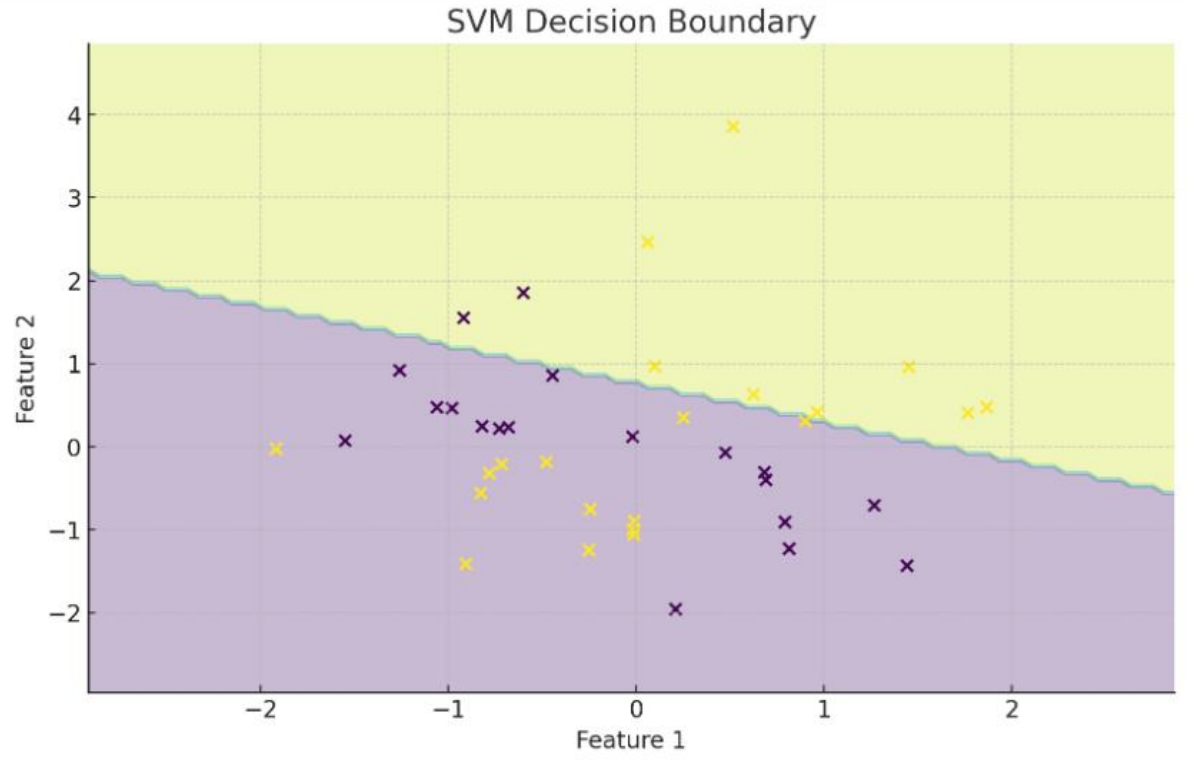
LIBRARIES IMPORT: Numpy, Sklearn

INPUT:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

np.random.seed(42)

x = np.random.randn(200, 2)  # 2D data for visualization

y = (x[:, 0] * x[:, 1] > 0).astype(int)  # Label: 1 if x1*x2 > 0, else 0x_train,
x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)model = SVC(kernel='linear')

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred)


print(f"Accuracy: {accuracy:.2f}")

print("Confusion Matrix:")

print(conf_matrix)

print("Classification Report:")

print(report)

def plot_decision_boundary(model, X, y):

   x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
```

```
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.title("SVM Decision Boundary")
    plt.show()

plot_decision_boundary(model, x_test, y_test)
```

OUTPUT:



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.57 | 0.89 | 0.69 | 19 |
| 1 | 0.80 | 0.38 | 0.52 | 21 |
| accuracy | | | 0.62 | 40 |
| macro avg | 0.68 | 0.64 | 0.61 | 40 |

# PRACTICAL 8

AIM: Wine Quality Prediction

LIBRARIES: Pandas, Numpy

INPUT:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')
```

OUTPUT:

```
    type  fixed acidity  volatile acidity  citric acid  residual sugar  \
0  white            7.0              0.27         0.36            20.7
1  white            6.3              0.30         0.34             1.6
2  white            8.1              0.28         0.40             6.9
3  white            7.2              0.23         0.32             8.5
4  white            7.2              0.23         0.32             8.5

   chlorides  free sulfur dioxide  total sulfur dioxide  density    pH  \
0      0.045                 45.0                 170.0   1.0010  3.00
1      0.049                 14.0                 132.0   0.9940  3.30
2      0.050                 30.0                  97.0   0.9951  3.26
3      0.058                 47.0                 186.0   0.9956  3.19
4      0.058                 47.0                 186.0   0.9956  3.19

   sulphates  alcohol  quality
0       0.45      8.8        6
1       0.49      9.5        6
2       0.44     10.1        6
3       0.40      9.9        6
4       0.40      9.9        6
```