Contents

Exercise 1: Introduction to Numpy

Aim:

1. Write a numpy program to evaluate 2 list.
2. Write a numpy program to generate even number array from 50-90.
3. Write a numpy program to generate a 4x4 identity matrix.
4. Write a program to generate a 5x5 0 matrix with elements on main diagonal: 1,2,3,4,5.
5. Write a numpy program to create a vector with values from 0 to 20 and change the sign of the numbers in the range 9 to 15.
6. Write a numpy program to compute the sum of all elements, sum of each column, and sum of each row for a given array.
7. Write a numpy program to save a given array to a text file and load it.
8. Write a numpy program to check whether 2 arrays are equal (element-wise comparison).
9. Write a numpy program to create a 4x4 array with random values. Create a new array from the set array by swapping 1st and last rows.
10. Write a numpy program to multiply 2 given array of same size element by element.

Python Code:

[1]:
```
import numpy as np
```

1.Write a numpy program to evaluate 2 list.

[2]:
```
l1=np.array([1,2,3,4])
l2=np.array([5,6,1,7])
gr=np.greater(l1,l2)
le=np.less_equal(l1,l2)
ge=np.greater_equal(l1,l2)
lt=np.less(l1,l2)
eq=np.equal(l1,l2)
print("L1 > L2: ",gr,"\nL1 <= L2",le,"\nL1 >= L2",ge,"\nL1 <␣
 ↪L2",lt,"\nL1  = L2",eq)
```

```
L1 > L2:  [False False  True False]
L1 <= L2 [ True  True False  True]
L1 >= L2 [False False  True False]
L1 < L2 [ True  True False  True]
L1  = L2 [False False False False]
```

2. Write a numpy program to generate even number array from 50-90.

[3]:
```
l=np.arange(50,91,2)
print("Array: ",l)
```

```
Array:  [50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90]
```

3. Write a numpy program to generate a 4x4 identity matrix.

[4]:
```
i=np.identity(4,dtype=int)
print("4x4 Identity Matrix")
print(i)
```

```
4x4 Identity Matrix
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
```

4. Write a program to generate a 5x5 0 matrix with elements on main diagonal: 1,2,3,4,5.

[5]:
```
z=np.diag([1,2,3,4,5])
print("5x5 0 matrix with main diagonal 1,2,3,4,5")
print(z)
```

```
5x5 0 matrix with main diagonal 1,2,3,4,5
[[1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]
 [0 0 0 0 5]]
```

5. Write a numpy program to create a vector with the values from 0 to 20 and change the sign of the numbers in the range 9 to 15.

[6]:
```
n=np.arange(0,21)
print("Original Vector", n)
for i in n:
    if i >=9 and i<=15:
        n[i]=i*-1
print("Vector after changing sign", n)
```

```
Original Vector [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17␣
↪18 19 20]
Vector after changing sign [  0   1   2   3   4   5   6   7   8  -9 -10␣
↪-11 -12 -13 -14 -15  16  17 18  19  20]
```

6. Write a numpy program to compute the sum of all elements, sum of each column, and sum of each row for a given array.

[7]:
```
n=[[1,2],[5,3]]
a1=np.array(n)
print("Sum of all element:",a1.sum())
print("Sum of rows:",a1.sum(1))
print("Sum of col:",a1.sum(0))
```

```
Sum of all element: 11
Sum of rows: [3 8]
Sum of col: [6 5]
```

7. Write a numpy program to save a given array to a text file and load it.

[8]:
```
n=np.arange(0,10)
with open("file.txt","w") as f:
    f.write(str(n))
with open("file.txt","r") as f:
    print(f.readlines())
```

```
['[0 1 2 3 4 5 6 7 8 9]']
```

8. Write a numpy program to check whether 2 arrays are equal (element-wise comparison).

[9]:
```
n1=np.array([0,10])
n2=np.array([0,4])
print("Array 1: ",n1)
print("Array 2: ",n2)
a=np.equal(n1,n2)
print("Element-wise comparison",a)
```

```
Array 1:  [ 0 10]
Array 2:  [0 4]
Element-wise comparison [ True False]
```

9. Write a numpy program to create a 4x4 array with random values. Create a new array from the set array by swapping 1st and last rows.

[10]:
```
import random
l1=[]
for i in range(4):
    l2=[]
    for j in range(4):
        l2.append(random.randint(2,9))
    l1.append(l2)
m1=np.array(l1)
print("Original 4x4 matrix")
print(m1)
m1[[0,-1]]=m1[[-1,0]]
print("\nMatrix after swapping 1st and last row")
print(m1)
```

```
Original 4x4 matrix
[[8 8 3 6]
 [4 6 5 4]
 [2 6 3 2]
 [5 5 6 9]]

Matrix after swapping 1st and last row
[[5 5 6 9]
 [4 6 5 4]
 [2 6 3 2]
 [8 8 3 6]]
```

10. Write a numpy program to multiply 2 given array of same size element by element.

[11]:
```
n1=np.array([1,2,3,4,5])
n2=np.array([2,3,4,5,6])
print("Array 1: ",n1)
print("Array 2: ",n2)
print("Element wise product: ",n1*n2)
```

```
Array 1:  [1 2 3 4 5]
Array 2:  [2 3 4 5 6]
Element wise product:  [ 2  6 12 20 30]
```

Exercise 2: Matrix Operation

Aim:

Write a program to input 2 matrix from the user and find the following:

i. Dot Product

ii. Transpose

iii. Determinant

iv. Inverse

v. Trace

vi. Rank

vii. Eigen values and Eigen vectors

Python Code:

[1]:
```
import numpy as np
matrix1=[]
matrix2=[]
r1,c1=list(map(int,input("Enter no of rows and cols for matrix 1:").
 ↪split()))
r2,c2=list(map(int,input("Enter no of rows and cols for matrix 2:").
 ↪split()))
m=list(map(int,input("Enter elements of matrix 1:").split()))
matrix1.append(m)
m1=np.array(matrix1).reshape(r1,c1)
m=list(map(int,input("Enter elements of matrix 2:").split()))
matrix2.append(m)
m2=np.array(matrix2).reshape(r2,c2)
print("Matrix 1:")
print(m1)
print("\nMatrix 2:")
print(m2)
```

```
Enter no of rows and cols for matrix 1:2 2
Enter no of rows and cols for matrix 2:2 2
Enter elements of matrix 1:1 2 3 4
Enter elements of matrix 2:9 8 7 6
Matrix 1:
[[1 2]
 [3 4]]

Matrix 2:
[[9 8]
 [7 6]]
```

i. Dot Product

[2]:
```
dot=m2.dot(m1)
print("\nDot Product:")
print(dot)
```

```
Dot Product:
[[33 50]
 [25 38]]
```

ii. Transpose

[3]:
```
print("\nTranspose of matrix 1:")
print(np.transpose(m1))
```

```
Transpose of matrix 1:
[[1 3]
 [2 4]]
```

iii. Determinant

[4]:
```
print("\nDeterminant of matrix 2")
print(np.linalg.det(m1))
```

```
Determinant of matrix 2
-2.0000000000000004
```

iv. Inverse

[5]:
```
print("\nInverse of matrix 2:")
print(np.linalg.inv(m2))
```

```
Inverse of matrix 2:
[[-3.   4. ]
 [ 3.5 -4.5]]
```

v. Trace

[6]:
```
print("\nTrace of matrix 1")
print(m1.trace())
```

```
Trace of matrix 1
5
```

vi. Rank

[7]:
```
print("\nRank of matrix 1")
print(np.linalg.matrix_rank(m1))
```

```
Rank of matrix 1
2
```

vii. Eigne values and Eigen vectors

[8]:
```
print("\nEigen Values of matrix 1")
print(np.linalg.eig(m2))
```

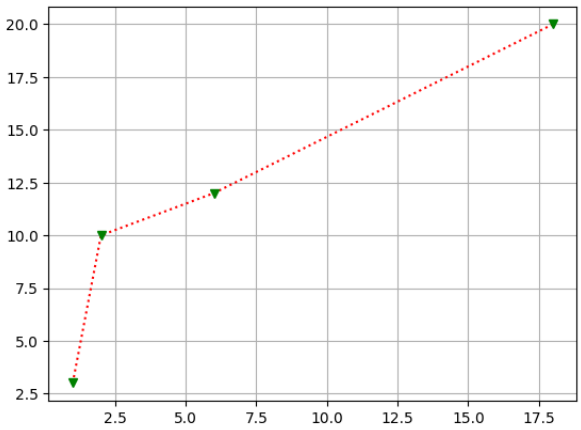
```
Eigen Values of matrix 1
(array([15.13216876, -0.13216876]), array([[ 0.79366214, -0.65894079],
       [ 0.60835878,  0.75219481]]))
```

Exercise 3: Programs Using Matplotlib

Program 1:

Draw a line in a diagram from position (1,3) to (2,10) then to (6,12) and finally to position (18,20). Mark each point with a beautiful green colour and set line colour to red and line style dotted.

[1]:
```
import matplotlib.pyplot as plt
x=[1,2,6,18]
y=[3,10,12,20]
plt.plot(x,y,color="red",marker='v',mfc='g',mec='g',linestyle=":")
plt.grid(True)
plt.show()
```

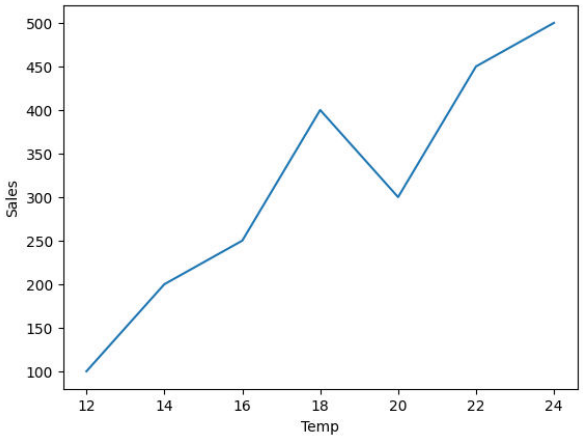


Program 2:

Draw a plot for the following data:

Temperature (°C)	Sales
12	100
14	200
16	250
18	400
20	300
22	450
24	500

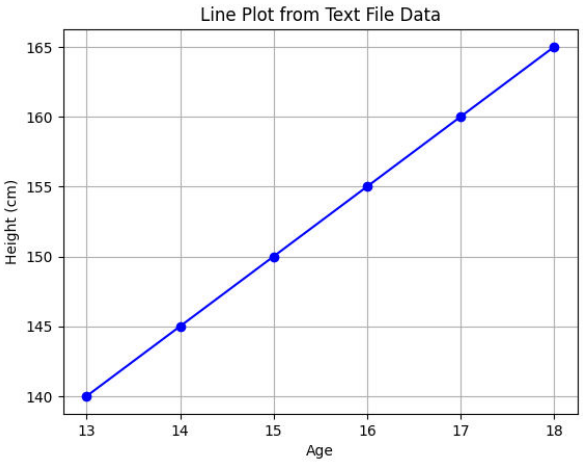
[2]:
```
import matplotlib.pyplot as plt
x=[12,14,16,18,20,22,24]
y=[100,200,250,400,300,450,500]
plt.plot(x,y)
plt.xlabel("Temp")
plt.ylabel("Sales")
plt.show()
```



Program 3:

Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

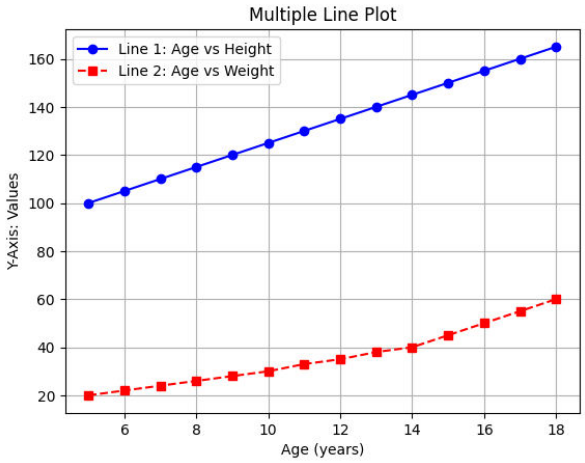
[3]:
```
import matplotlib.pyplot as plt
x_values = []
y_values = []
with open("data.txt", 'r') as file:
    for line in file:
        values = line.strip().split()
        x_values.append(float(values[0]))
        y_values.append(float(values[1]))
plt.plot(x_values, y_values, marker='o', linestyle='-', color='b')
plt.xlabel('Age')
plt.ylabel('Height (cm)')
plt.title('Line Plot from Text File Data')
plt.grid(True)
plt.show()
```



Program 4:

Write a Python program to plot two or more lines on the same plot with suitable legends of each line.

[4]:
```
import matplotlib.pyplot as plt
age = [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
height = [100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150, 155,␣
 ↪160, 165]
weight = [20, 22, 24, 26, 28, 30, 33, 35, 38, 40, 45, 50, 55, 60]
plt.plot(age, height, marker='o', linestyle='-', color='b', label='Line␣
 ↪1: Age vs Height')
plt.plot(age, weight, marker='s', linestyle='--', color='r',␣
 ↪label='Line 2: Age vs Weight')
plt.xlabel('Age (years)')
plt.ylabel('Y-Axis: Values')
plt.title('Multiple Line Plot')
plt.legend()
plt.grid(True)
plt.show()
```



Program 5:

Write a Python program to create multiple plots.

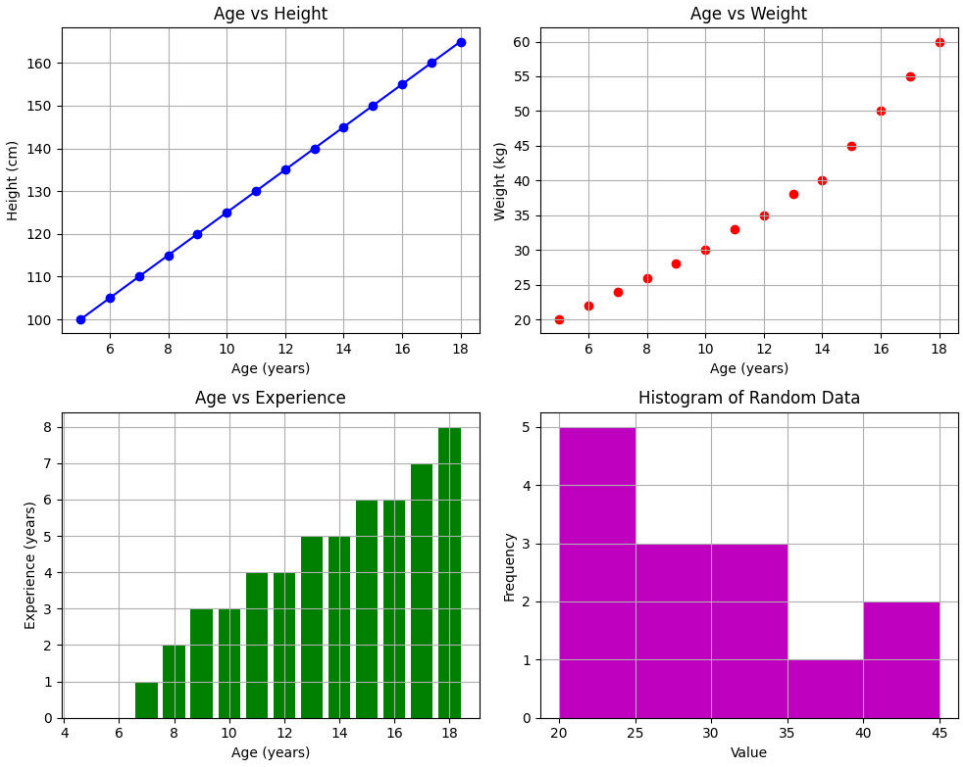
[5]:
```
import matplotlib.pyplot as plt
age = [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
height = [100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150, 155,␣
 ↪160, 165]
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
axs[0, 0].plot(age, height, marker='o', color='b')
axs[0, 0].set_title('Age vs Height')
axs[0, 0].set_xlabel('Age (years)')
axs[0, 0].set_ylabel('Height (cm)')
axs[0, 0].grid(True)

weight = [20, 22, 24, 26, 28, 30, 33, 35, 38, 40, 45, 50, 55, 60]
axs[0, 1].scatter(age, weight, color='r')
```

```
axs[0, 1].set_title('Age vs Weight')
axs[0, 1].set_xlabel('Age (years)')
axs[0, 1].set_ylabel('Weight (kg)')
axs[0, 1].grid(True)

experience = [0, 0, 1, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 8]
axs[1, 0].bar(age, experience, color='g')
axs[1, 0].set_title('Age vs Experience')
axs[1, 0].set_xlabel('Age (years)')
axs[1, 0].set_ylabel('Experience (years)')
axs[1, 0].grid(True)

data = [20, 22, 24, 26, 28, 30, 22, 24, 28, 30, 33, 35, 40, 45]
axs[1, 1].hist(data, bins=5, color='m')
axs[1, 1].set_title('Histogram of Random Data')
axs[1, 1].set_xlabel('Value')
axs[1, 1].set_ylabel('Frequency')
axs[1, 1].grid(True)
plt.tight_layout()
plt.show()
```



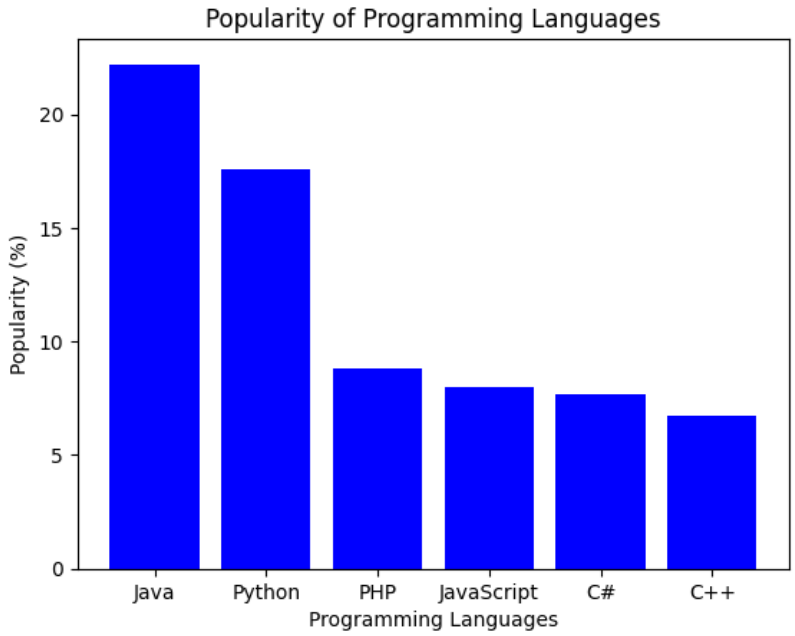
Program 6:

Consider the following data. Programming Languages: Java, Python, PHP, JavaScript, C#, C++ Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

[6]:
```
import matplotlib.pyplot as plt
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
```

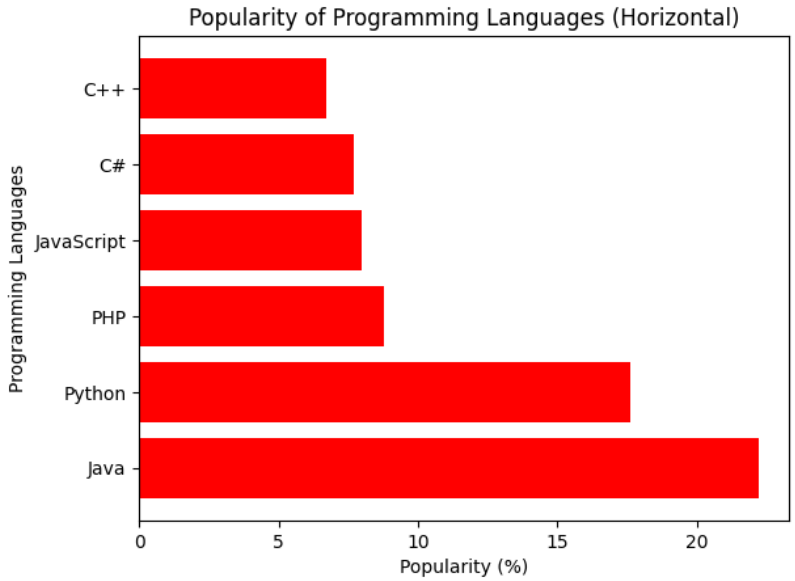
i. Write a Python program to display a bar chart of the popularity of programming Languages.

[7]:
```
plt.bar(languages, popularity, color='blue')
plt.xlabel('Programming Languages')
plt.ylabel('Popularity (%)')
plt.title('Popularity of Programming Languages')
plt.show()
```



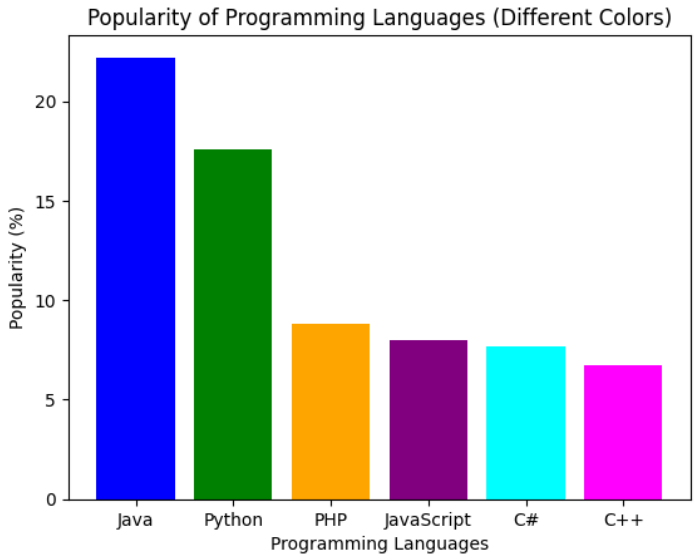
ii. Write a Python program to display a horizontal bar chart of the popularity of programming Languages (Give Red colour to the bar chart).

[8]:
```
plt.barh(languages, popularity, color='red')
plt.xlabel('Popularity (%)')
plt.ylabel('Programming Languages')
plt.title('Popularity of Programming Languages (Horizontal)')
plt.show()
```



iii. Write a Python program to display a bar chart of the popularity of programming Languages. Use a different colour for each bar.

[9]:
```
colors = ['blue', 'green', 'orange', 'purple', 'cyan', 'magenta']
plt.bar(languages, popularity, color=colors)
plt.xlabel('Programming Languages')
plt.ylabel('Popularity (%)')
plt.title('Popularity of Programming Languages (Different Colors)')
plt.show()
```

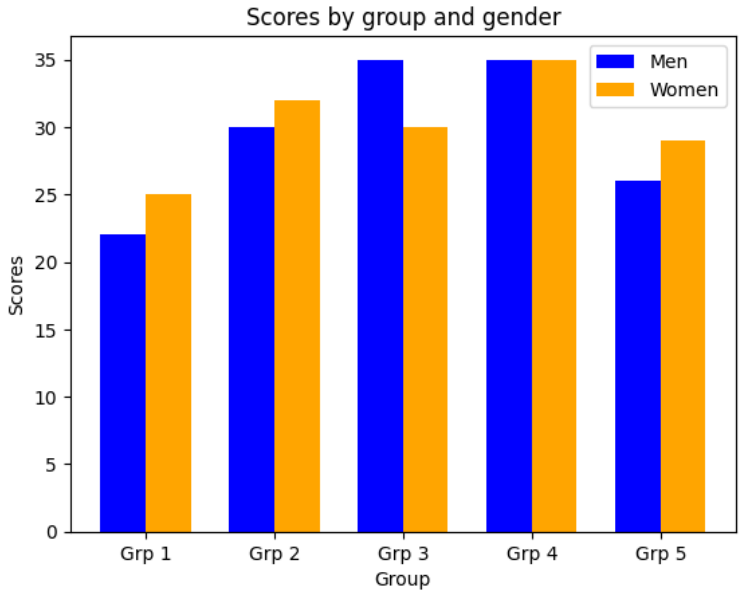


Program 7:

Write a Python program to create a bar plot of scores by group and gender. Use multiple X values on the same chart for men and women.
Sample Data:
Means (men) = (22, 30, 35, 35, 26)

Means (women) = (25, 32, 30, 35, 29)

[10]:
```
import numpy as np
import matplotlib.pyplot as plt
groups = ['Grp 1', 'Grp 2', 'Grp 3', 'Grp 4', 'Grp 5']
means_men = [22, 30, 35, 35, 26]
means_women = [25, 32, 30, 35, 29]
x = np.arange(len(groups))
width = 0.35
fig, ax = plt.subplots()
bars_men = ax.bar(x - width/2, means_men, width, label='Men',␣
 ↪color='blue')
bars_women = ax.bar(x + width/2, means_women, width, label='Women',␣
 ↪color='orange')
ax.set_xlabel('Group')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(groups)
ax.legend()
plt.show()
```



Exercise 4: Introduction to Pandas

Aim:

1. Write a python program to convert list to series.
2. Write a python program to generate series of dates from 1 August 2024 to 15 August 2024.
3. Write a program to convert a dictionary to DataFrame and display it.
4. Write a program to create a 2D list and covert it into DataFrame and display it.

Python Code:

1.Write a python program to convert list to series.

[1]:
```
import pandas as pd
sample_list = [10, 20, 30, 40, 50]
result_series = pd.Series(sample_list)
print("Original List:", sample_list)
print("Converted Series:")
print(result_series)
```

```
Original List: [10, 20, 30, 40, 50]
Converted Series:
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

2. Write a python program to generate series of dates from 1 August 2024 to 15 August 2024.

[2]:
```
import pandas as pd
start_date = '2024-08-01'
end_date = '2024-08-15'
dates = pd.date_range(start_date, end_date)
print("Series of Dates from 1 August 2024 to 15 August 2024:")
print(dates)
```

```
Series of Dates from 1 August 2024 to 15 August 2024:
DatetimeIndex(['2024-08-01', '2024-08-02', '2024-08-03', '2024-08-04',
               '2024-08-05', '2024-08-06', '2024-08-07', '2024-08-08',
               '2024-08-09', '2024-08-10', '2024-08-11', '2024-08-12',
               '2024-08-13', '2024-08-14', '2024-08-15'],
              dtype='datetime64[ns]', freq='D')
```

3. Write a program to convert a dictionary to DataFrame and display it.

[3]:
```
import pandas as pd
data = {
    'Name': ['Dhoni', 'Virat', 'Rohit', 'Sanju'],
    'No:': [7, 18, 45, 11],
    'Team': ['CSK', 'RCB', 'MI', 'RR']}
print("Dictionary: ",data)
df = pd.DataFrame(data)
print("\nConverted DataFrame:")
print(df)
```

```
Dictionary:  {'Name': ['Dhoni', 'Virat', 'Rohit', 'Sanju'], 'No:': [7,␣
↪18, 45,
11], 'Team': ['CSK', 'RCB', 'MI', 'RR']}

Converted DataFrame:
    Name  No: Team
0  Dhoni    7  CSK
1  Virat   18  RCB
2  Rohit   45   MI
3  Sanju   11   RR
```

4. Write a program to create a 2D list and covert it into DataFrame and display it.

[4]:
```
import pandas as pd
data = [
    ['Mbappe', 9, 'FW'],
    ['Vini', 7, 'LW'],
    ['Rodrygo', 11, 'RW'],
    ['Bellingham', 5, 'AMF']]
print("2D List: ",data)
column_names = ['Name', 'No:', 'Position']
df = pd.DataFrame(data,columns=column_names)
print("\nConverted DataFrame:")
print(df)
```

```
2D List:  [['Mbappe', 9, 'FW'], ['Vini', 7, 'LW'], ['Rodrygo', 11, 'RW'],
['Bellingham', 5, 'AMF']]

Converted DataFrame:
         Name  No: Position
0      Mbappe    9       FW
1        Vini    7       LW
2     Rodrygo   11       RW
3  Bellingham    5      AMF
```

Exercise 5: K-Nearest Neighbour (KNN)

Aim:

Write a program to build a KNN model on the Iris dataset.

Python Code:

[1]:
```
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

[2]:
```
iris=datasets.load_iris()
X,y=iris.data,iris.target
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.
 ↪7,random_state=78)
```

[3]:
```
classifier=neighbors.KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 95.56%
```

Prediction on unseen data.

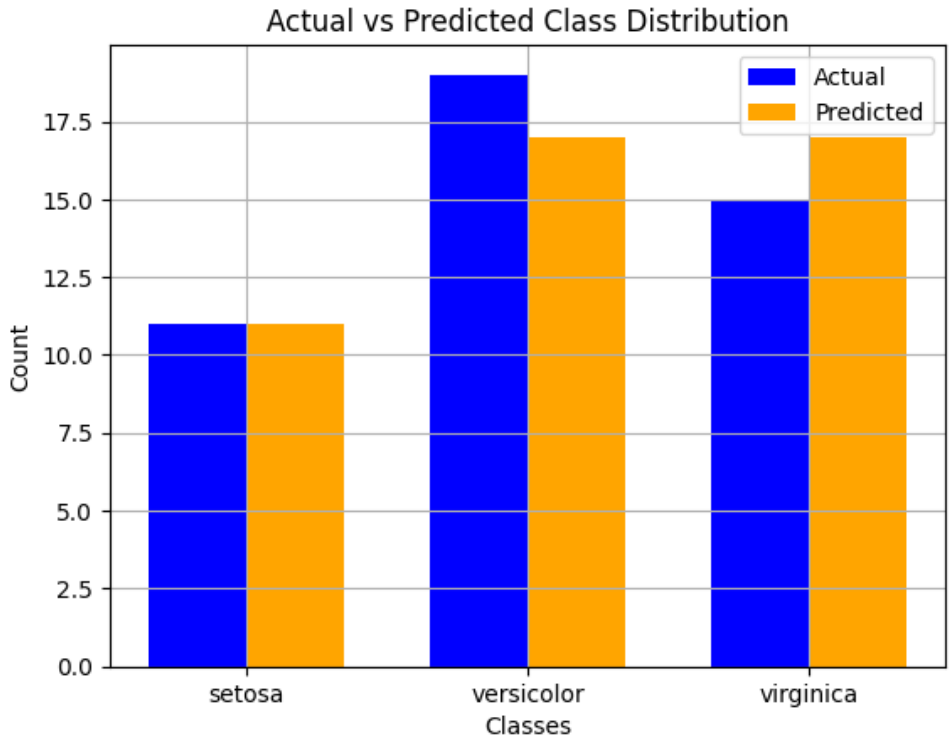
[4]:
```
result=classifier.predict([[3.5,5.1,4.8,2.9]])
print("Predicted Class: ",datasets.load_iris().target_names[result])
```

```
Predicted Class:  ['versicolor']
```

Box Plot to visualize the distribution of predictions versus the true labels.

[5]:
```
actual_counts = np.bincount(y_test)
predicted_counts = np.bincount(y_pred)
classes = iris.target_names
x = np.arange(len(classes))
width = 0.35
plt.bar(x - width/2, actual_counts, width, label='Actual', color='blue')
plt.bar(x + width/2, predicted_counts, width, label='Predicted',␣
 ↪color='orange')
plt.xlabel('Classes')
plt.ylabel('Count')
```

```
plt.title('Actual vs Predicted Class Distribution')
plt.xticks(x, classes)
plt.legend()
plt.grid()
plt.show()
```



Exercise 6: KNN- Diabties Dataset

Aim:

Write a program to implement KNN for Diabeties dataset and create a scatter plot for the result using eda tools.

Python Code:

[1]:
```
import pandas as pd
from sklearn import datasets
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
```

[2]:
```
db=datasets.load_diabetes()
scaler = StandardScaler()
X, y = db.data, db.target
X_scaled = scaler.fit_transform(X)
X,y=db.data,db.target
y_binned = np.digitize(y, bins=[y.mean() - y.std(), y.mean() + y.std()])
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y_binned,␣
 ↪train_size =0.8,random_state=42)
classifier=neighbors.KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 61.80%
```

Prediction on unseen data.

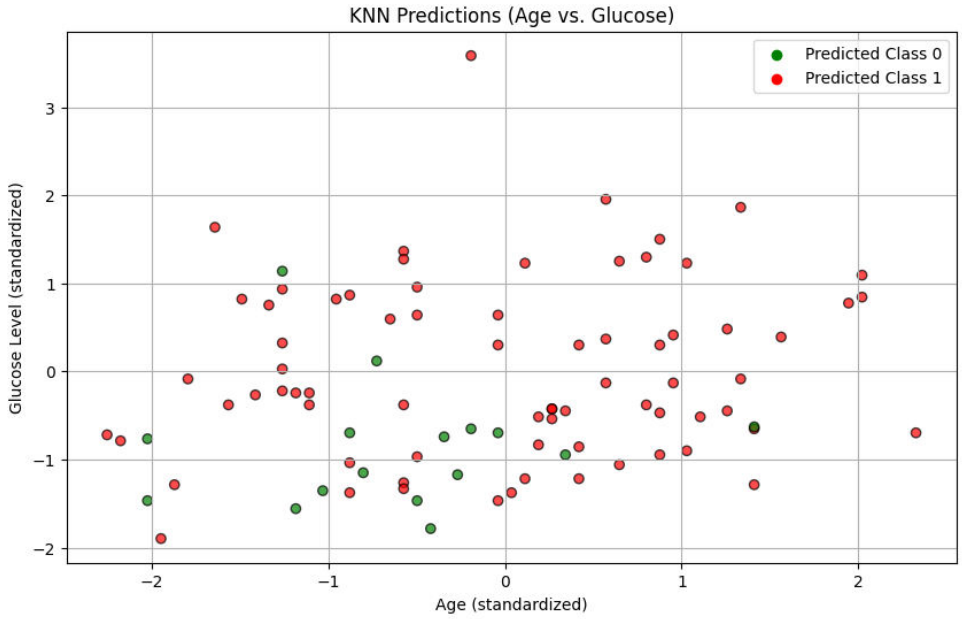
[3]:
```
input_data = np.array([[4, 130, 70, 20, 80, 30.5, 0.4, 45,0,0]])
prediction = classifier.predict(input_data)
print("Prediction:", "Diabetic" if prediction[0] == 1 else␣
 ↪"Non-Diabetic")
```

```
Prediction: Diabetic
```

Scatter Plot Scatter Plot of Actual vs. Predicted Outcomes

[4]:
```
plt.figure(figsize=(10, 6))
colors = np.where(y_pred == 0, 'green', 'red')
plt.scatter(X_test[:, 0], X_test[:, 2], color=colors, alpha=0.7,␣
 ↪edgecolors='k')
plt.title('KNN Predictions (Age vs. Glucose)')
```

```
plt.xlabel('Age (standardized)')
plt.ylabel('Glucose Level (standardized)')
plt.grid()
plt.scatter([], [], color='green', label='Predicted Class 0')
plt.scatter([], [], color='red', label='Predicted Class 1')
plt.legend()
plt.show()
```



Exercise 7: Decision Tree

Aim:

Write a program to implement Decision Tree for Iris dataset.

Python Code:

[1]:
```
import pandas as pd
from sklearn import datasets
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

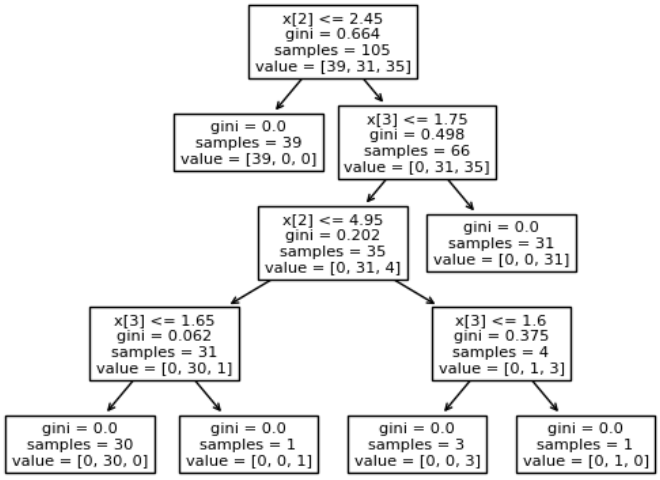
[2]:
```
iris=datasets.load_iris()
X,y=iris.data,iris.target
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.
 ↪7,random_state=78)
dt=tree.DecisionTreeClassifier()
dt.fit(X_train,y_train)
y_pred=dt.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 95.56%
```

Decision Tree Model.

[3]:
```
tree.plot_tree(dt)
```

[3]:



[4]:
```
result=dt.predict([[3,4,5,6]])
print("Predicted Class:", iris.target_names[result])
```

```
Predicted Class: ['virginica']
```

Exercise 8: Naive Bayes Classifier

Aim:

Write a program to implement Naive Bayes Classifier for Iris dataset.

Python Code:

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn import naive_bayes
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[2]:
```
db=datasets.load_iris()
X,y=db.data,db.target
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7)
classifier=naive_bayes.GaussianNB()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 95.56%
```

Prediction on unseen data.

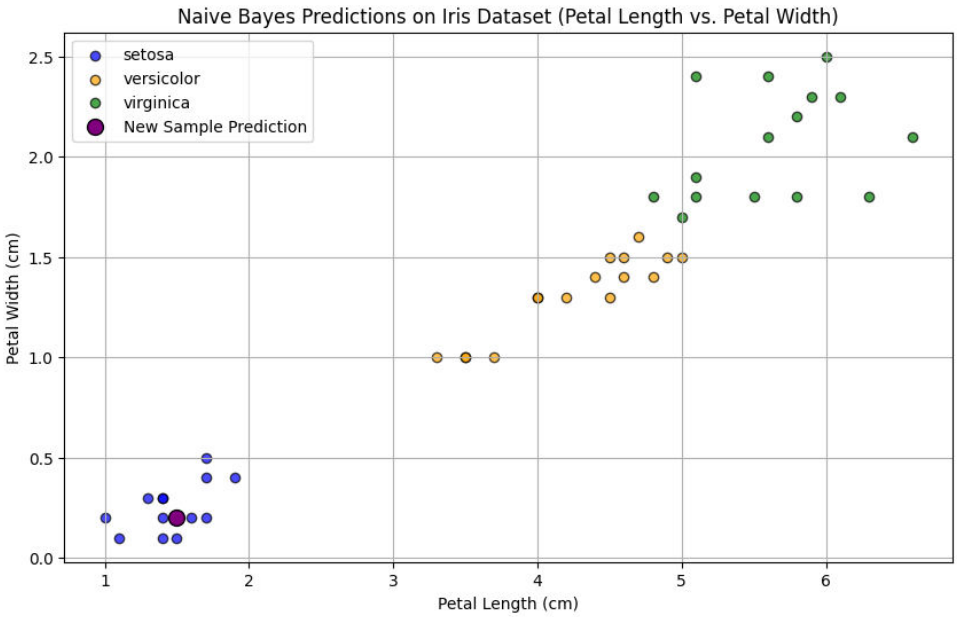
[3]:
```
new_data=np.array([[5.0, 3.5, 1.5, 0.2]])
result = classifier.predict(new_data)
print("Predicted cLass:", db.target_names[result][0])
```

```
Predicted cLass: setosa
```

Scatter Plot to visualize the petal length (X-axis) and petal width (Y-axis) of the test set.

[4]:
```
plt.figure(figsize=(10, 6))
colors = ['blue', 'orange', 'green']
for i in range(3):
    plt.scatter(X_test[y_pred == i, 2], X_test[y_pred == i, 3],␣
 ↪color=colors[i], alpha=0.7, edgecolors='k', label=db.target_names[i])
plt.scatter(new_data[0, 2], new_data[0, 3], color='purple', s=100,␣
 ↪edgecolors='k', label='New Sample Prediction')
plt.title('Naive Bayes Predictions on Iris Dataset (Petal Length vs.␣
 ↪Petal Width)')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.grid()
plt.legend()
```

```
plt.show()
```


Naive Bayes Predictions on Iris Dataset (Petal Length vs. Petal Width)
setosa
versicolor
virginica
New Sample Prediction
Petal Width (cm)
Petal Length (cm)
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6

Exercise 9: Linear Regression

Aim:

Write a program to implement Linear Regression on a random dataset.

Python Code:

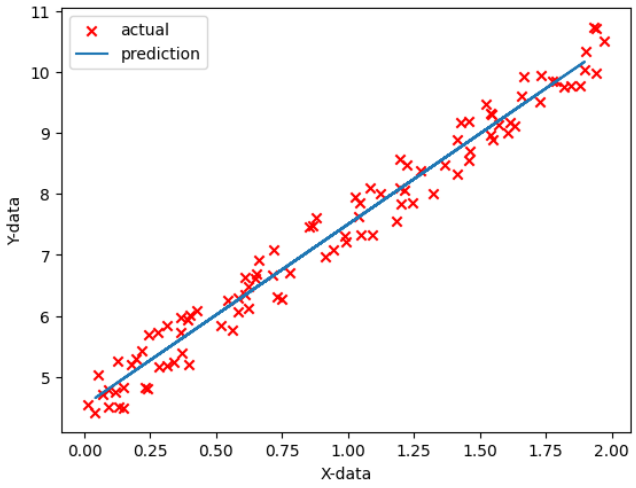
[1]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
import matplotlib.pyplot as plt
```

[2]:
```
np.random.seed(42)
X=2*np.random.rand(100,1)
y=4+3*X+np.random.rand(100,1)
```

[3]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,y,train_size=0.
↪8,random_state=42)
m=LinearRegression()
m.fit(X_train,Y_train)
y_pred=m.predict(X_test)
se=mean_squared_error(Y_test,y_pred)
print("Squared Error: ",se)
print("R^2 Score: ",r2_score(Y_test,y_pred))
```

```
Squared Error:  0.0954430346975895
R^2 Score:  0.9742947589810751
```

[4]:
```
plt.scatter(X,y,color='red',marker='x',label="actual")
plt.plot(X_test,y_pred,label='prediction')
plt.xlabel('X-data')
plt.ylabel('Y-data')
plt.legend()
plt.show()
```



Prediction on unseen data.

[5]:
```
n=float(input("Enter X value to predict: "))
n1=np.array([[n]])
print("Predicted y value",m.predict(n1)[0][0])
```

```
Enter X value to predict: 0.5
Predicted y value 6.017292590766776
```

Exercise 10: K Means Clustering

Aim:

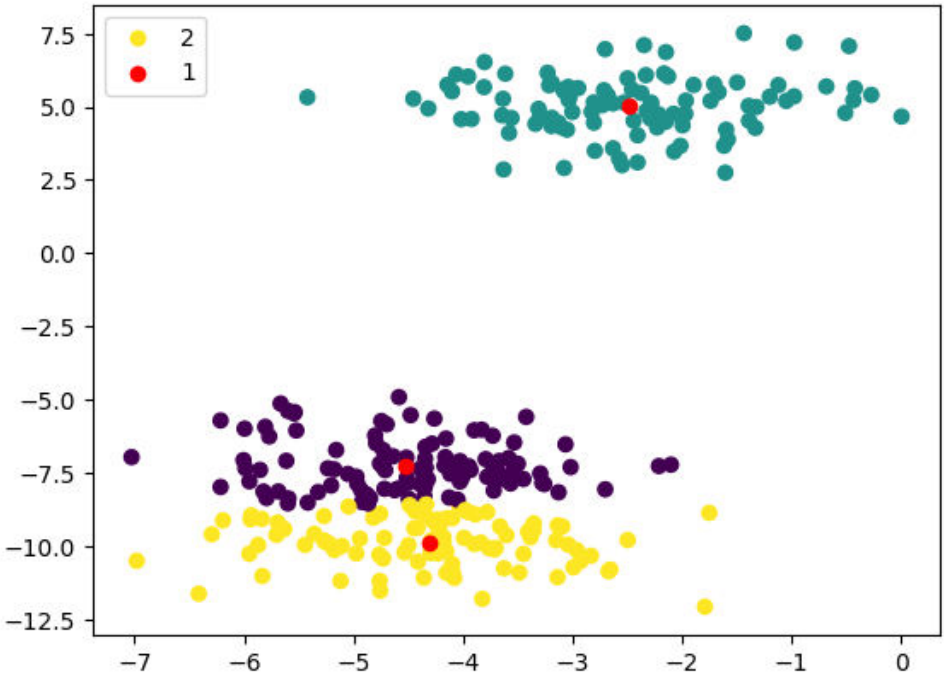
Write a program to implement K Means Clustering on a random dataset.

Python Code:

[1]:
```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

K-Means Clustering with 3 Clusters and Centroids.

[2]:
```
data,_ = make_blobs(n_samples=300)
m=KMeans(n_clusters=3,random_state=42,n_init='auto')
m.fit(data)
labels=m.labels_
set(labels)
centroids=m.cluster_centers_
plt.scatter(data[:,0],data[:,1],c=labels,cmap="viridis")
plt.scatter(centroids[:,0],centroids[:,1],c='red')
plt.legend(list(labels))
plt.show()
```



Prediction on unseen data.

[3]:
```
new_data=[[2,-4],[6,6],[2,5]]
result=m.predict(new_data)
for i, point in enumerate(new_data):
    print(f"The point {point} belongs to cluster {result[i]}.")
```

```
The point [2, -4] belongs to cluster 0.
The point [6, 6] belongs to cluster 1.
The point [2, 5] belongs to cluster 1.
```

Exercise 11: Support Vector Machine- SVM

Aim:

Write a program to implement a Support Vector Machine on a random dataset.

Python Code:

[1]:
```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,classification_report
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
X,y=make_classification(n_samples=1000,n_features=2,n_informative=2,
n_redundant=0,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.
↪8,random_state=42)
```

[3]:
```
s=SVC(kernel='linear')
s.fit(X_train,y_train)
y_pred = s.predict(X_test)
ac=accuracy_score(y_test,y_pred)
print("Accuracy: ",ac)
report=classification_report(y_test,y_pred)
print("\nClassification Report")
print(report)
```

```
Accuracy:  0.88

Classification Report
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       101
           1       0.88      0.88      0.88        99

    accuracy                           0.88       200
   macro avg       0.88      0.88      0.88       200
weighted avg       0.88      0.88      0.88       200
```

Support Vector Machine Decision Boundary Visualization.

[4]:
```
plt.figure(figsize=(8,6))
plt.scatter(X[:,0],X[:,1],c=y,cmap='viridis',s=50,alpha=0.6)
ax=plt.gca()
xlimit=ax.set_xlim()
ylimit=ax.set_ylim()
xx,yy=np.meshgrid(np.linspace(xlimit[0],xlimit[1],100),np.
↪linspace(ylimit[0],ylimit[1],100))
```

```
z=s.decision_function(np.c_[xx.ravel(),yy.ravel()])
z=z.reshape(xx.shape)
plt.contour(xx,yy,z,colors='k',levels=[-1,0,1],alpha=0.
 ↪5,linestyles=['--','-','--'])
plt.title("SVM Decision Boundary")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```