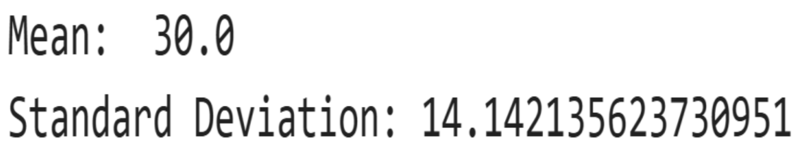
1. Calculate the mean and standard deviation.

First.py

```
import numpy as np

data = [10, 20, 30, 40, 50]

mean_value = np.mean(data)

std_deviation = np.std(data)

print("Mean: ",mean_value)

print("Standard Deviation:", std_deviation)
```

OUTPUT



```
Mean:  30.0
Standard Deviation: 14.142135623730951
```

2. Read CSV and Calculate Mean & Standard Deviation of Scores

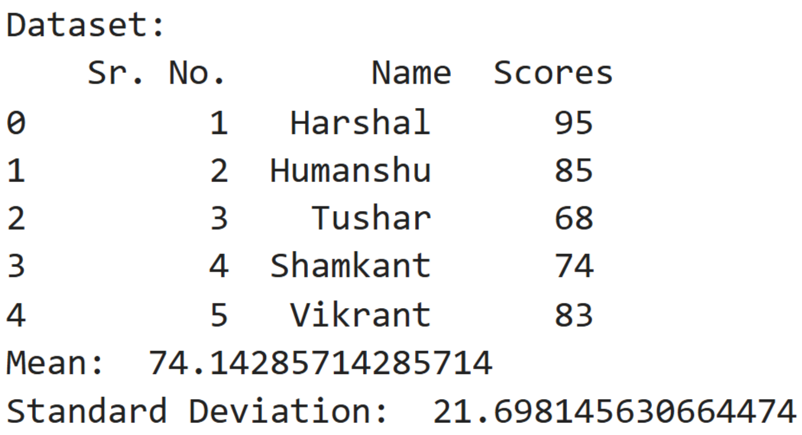
data.csv

Sr. No.	Name	Scores
1	Harshal	95
2	Humanshu	85
3	Tushar	68
4	Shamkant	74
5	Vikrant	83
6	Milind	29
7	Mayur	85

Second.py

```
import pandas as pd

df = pd.read_csv("data.csv")

print("Dataset:\n", df.head())


mean_value = df['Scores'].mean()

std_deviation = df['Scores'].std()


print("\nMean: ", mean_value)

print("Standard Deviation: ", std_deviation)
```

OUTPUT



```
Dataset:
    Sr. No.       Name  Scores
0         1    Harshal      95
1         2   Humanshu      85
2         3     Tushar      68
3         4   Shamkant      74
4         5    Vikrant      83
Mean:  74.14285714285714
Standard Deviation:  21.698145630664474
```

3. Perform data filtering, and calculate aggregate statistics.

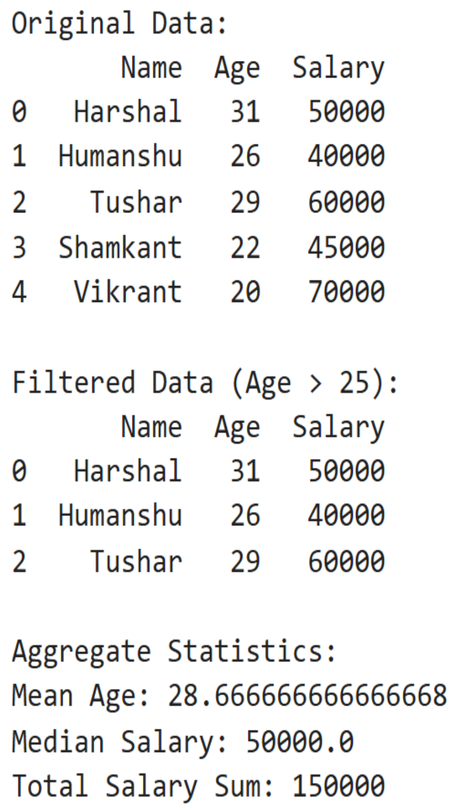
data.csv

Name	Age	Salary
Harshal	31	50000
Humanshu	26	40000
Tushar	29	60000
Shamkant	22	45000
Vikrant	20	70000

Third.py

```
import pandas as pd
# Load data
df = pd.read_csv('data.csv')
print("Original Data:")
print(df.head())
# Perform data filtering (Age > 25)
filtered_df = df[df['Age'] > 25]

# Calculate aggregate statistics
mean_age = filtered_df['Age'].mean()
median_salary = filtered_df['Salary'].median()
total_salary = filtered_df['Salary'].sum()

# Display results
print("\nFiltered Data (Age > 25):")
print(filtered_df)
print("\nAggregate Statistics:")
print("Mean Age:", mean_age)
print("Median Salary:", median_salary)
print("Total Salary Sum:", total_salary)
```

OUTPUT



```
Original Data:
       Name  Age  Salary
0   Harshal   31   50000
1  Humanshu   26   40000
2    Tushar   29   60000
3  Shamkant   22   45000
4   Vikrant   20   70000

Filtered Data (Age > 25):
       Name  Age  Salary
0   Harshal   31   50000
1  Humanshu   26   40000
2    Tushar   29   60000

Aggregate Statistics:
Mean Age: 28.666666666666668
Median Salary: 50000.0
Total Salary Sum: 150000
```

4. Calculate total sales by month.

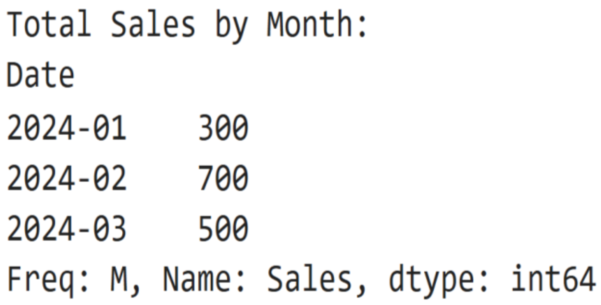
sales_data.csv

Date	Sales
15-01-2024	100
20-01-2024	200
10-02-2024	300
25-02-2024	400
05-03-2024	500

Fourth.py

```
import pandas as pd


data = pd.read_csv('sales_data.csv')

data['Date'] = pd.to_datetime(data['Date'], dayfirst=True)


monthly_sales = data.groupby(data['Date'].dt.to_period('M'))['Sales'].sum()


print("Total Sales by Month:")

print(monthly_sales)
```

OUTPUT



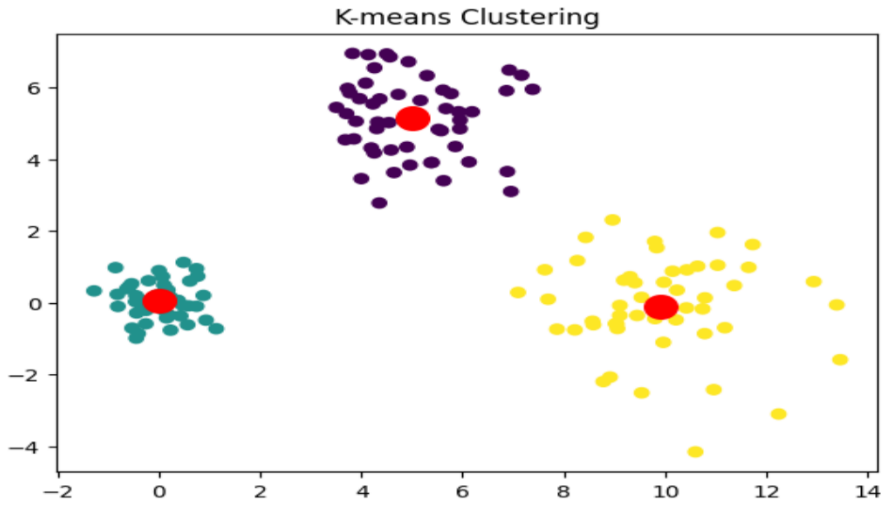
```
Total Sales by Month:
Date
2024-01    300
2024-02    700
2024-03    500
Freq: M, Name: Sales, dtype: int64
```

5. Implement the Clustering using K-means.

Fifth.py

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans


# Generate data

np.random.seed(0)

X = np.vstack([

   np.random.normal([0, 0], 0.5, (50, 2)),

   np.random.normal([5, 5], 1, (50, 2)),

   np.random.normal([10, 0], 1.5, (50, 2))

])


# K-means clustering and plot

kmeans = KMeans(n_clusters=3).fit(X)

plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis')

plt.scatter(*kmeans.cluster_centers_.T, c='red', s=200)

plt.title('K-means Clustering')

plt.show()
```

OUTPUT

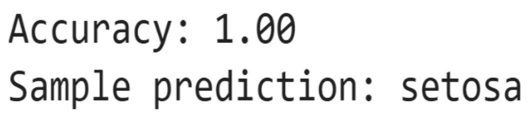


6. Classification using Random Forest.

Sixth.py

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score


# Load data and split

iris = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2,
random_state=42)


# Train model and predict

clf = RandomForestClassifier(n_estimators=10).fit(X_train, y_train)

y_pred = clf.predict(X_test)


# Print results

print("Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred)))

print("Sample prediction:", iris.target_names[clf.predict([[5.1,3.5,1.4,0.2]])[0]])
```

OUTPUT



```
Accuracy: 1.00
Sample prediction: setosa
```

7. Regression Analysis using Linear Regression.

data.csv

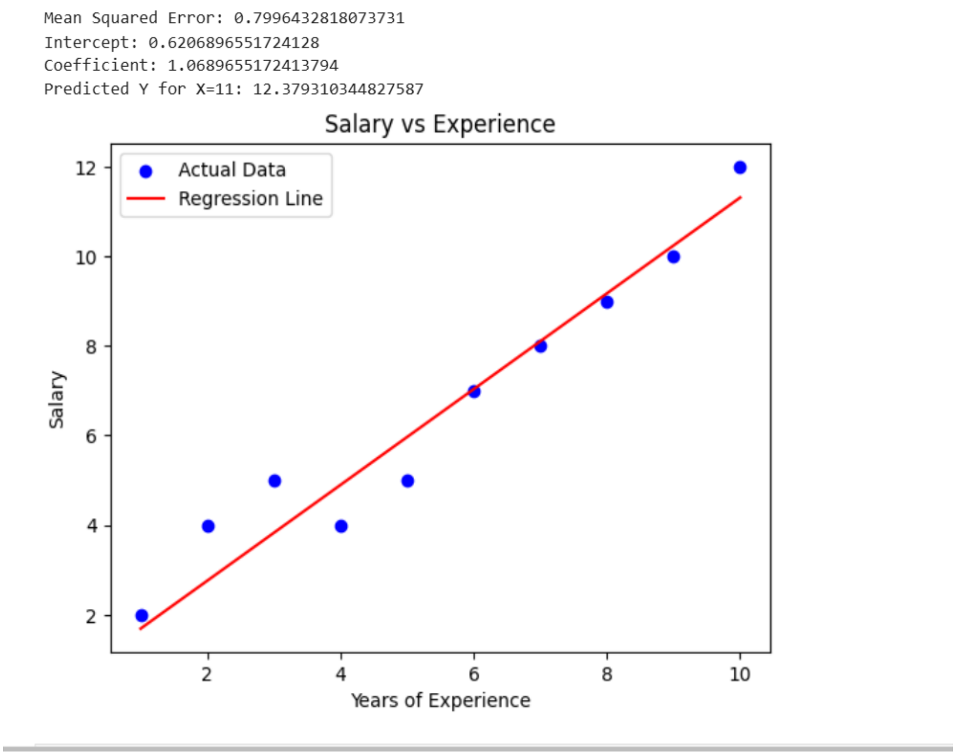
X	Y
1	2
2	4
3	5
4	4
5	5
6	7
7	8
8	9
9	10
10	12

Seventh.py

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

# Load data from CSV file

df = pd.read_csv('data.csv')

# Splitting data into features and target variable

X = df[['X']]

y = df['Y']

# Splitting dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating and training the model

model = LinearRegression()

model.fit(X_train, y_train)

# Making predictions

y_pred = model.predict(X_test)

# Evaluating the model

mse = mean_squared_error(y_test, y_pred)
```

```
print("Mean Squared Error:", mse)

print("Intercept:", model.intercept_)

print("Coefficient:", model.coef_[0])

# Predicting for a new value

new_X = pd.DataFrame({'X': [11]})  # Ensure new input has feature names

predicted_Y = model.predict(new_X)

print("Predicted Y for X=11:", predicted_Y[0])

# Plotting Salary vs Experience

plt.scatter(X, y, color='blue', label='Actual Data')

plt.plot(X, model.predict(X), color='red', label='Regression Line')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.title('Salary vs Experience')

plt.legend()

plt.show()
```

OUTPUT



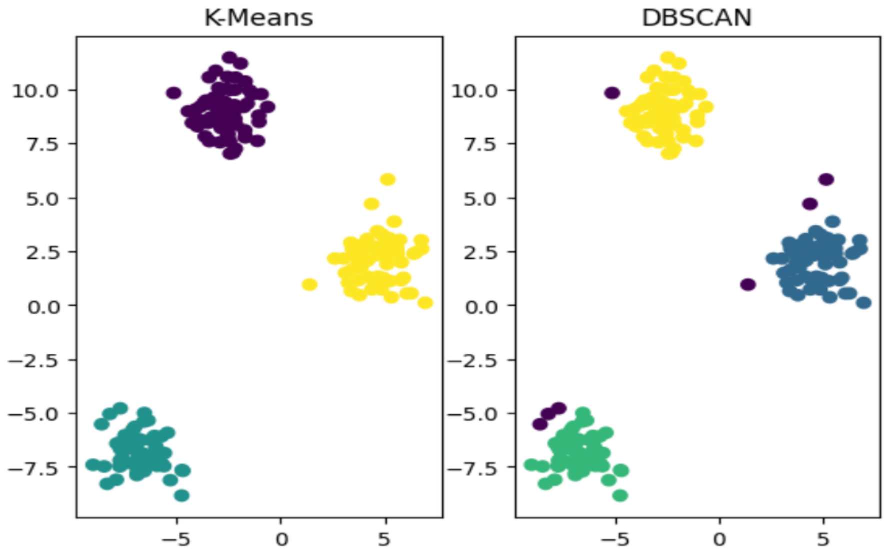
```
Mean Squared Error: 0.7996432818073731
Intercept: 0.6206896551724128
Coefficient: 1.0689655172413794
Predicted Y for X=11: 12.379310344827587
```

9. Visualize the result of the clustering and compare.

Ninth.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.datasets import make_blobs

X, _ = make_blobs(n_samples=200, centers=3, random_state=42)

plt.subplot(1, 2, 1)
plt.scatter(X[:, 0], X[:, 1], c=KMeans(n_clusters=3).fit_predict(X), cmap='viridis')
plt.title("K-Means")

plt.subplot(1, 2, 2)
plt.scatter(X[:, 0], X[:, 1], c=DBSCAN(eps=1.0, min_samples=5).fit_predict(X), cmap='viridis')
plt.title("DBSCAN")
plt.show()
```

OUTPUT

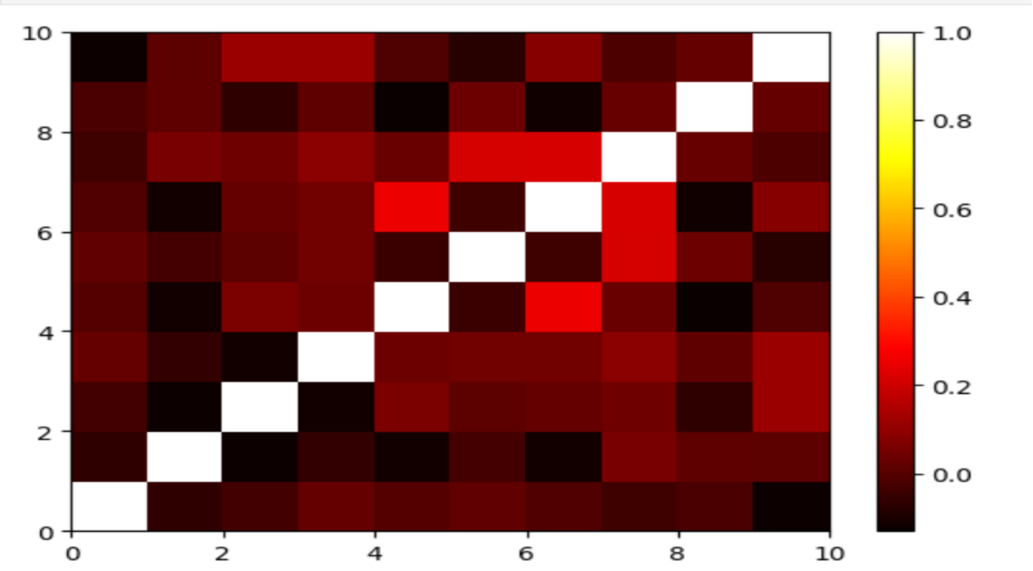


10. Visualize the correlation matrix using a pseudocolor plot.

Tenth.py

```
import numpy as np

import matplotlib.pyplot as plt


# Generate some random data

np.random.seed(0)  # For reproducibility

data = np.random.rand(10, 100)


# Create a correlation matrix

corr_matrix = np.corrcoef(data)


# Plot the correlation matrix

plt.pcolor(corr_matrix, cmap='hot')

plt.colorbar()

plt.show()
```

OUTPUT

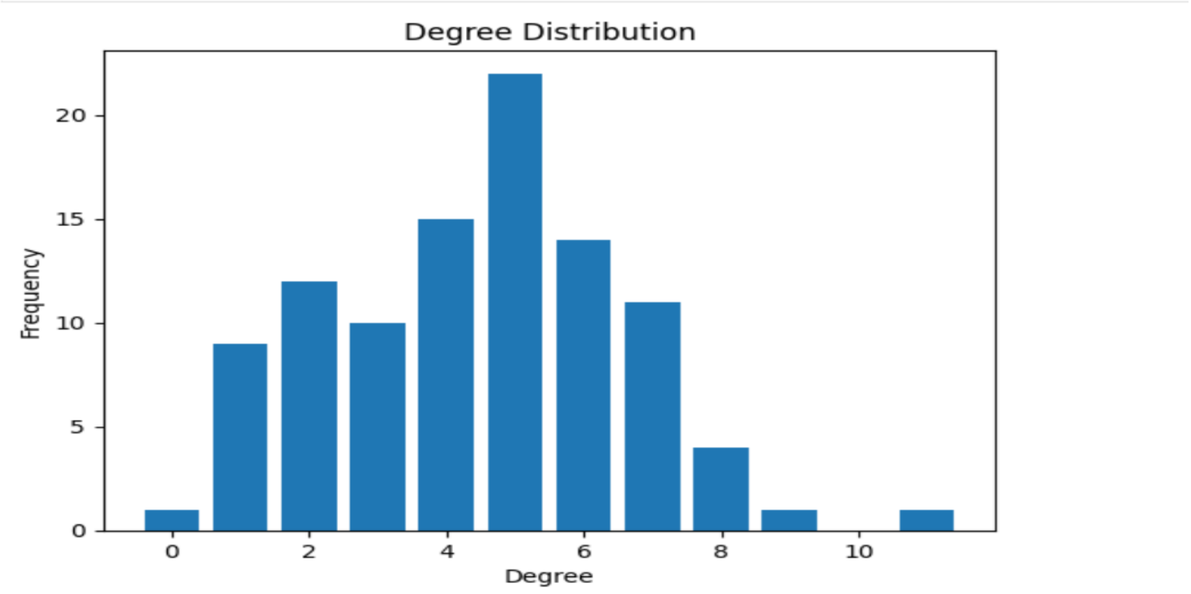


11. Use of degrees distribution of a network.

Eleventh.py

```
import networkx as nx
import matplotlib.pyplot as plt

# Create a random network
G = nx.erdos_renyi_graph(100, 0.05)

# Calculate degrees
degrees = dict(G.degree)

# Extract degree values
degree_values = list(degrees.values())

# Plot degree distribution
plt.hist(degree_values, bins=range(max(degree_values)+2), align='left', rwidth=0.8)
plt.xlabel('Degree')
plt.ylabel('Frequency')
plt.title('Degree Distribution')
plt.show()
```

OUTPUT

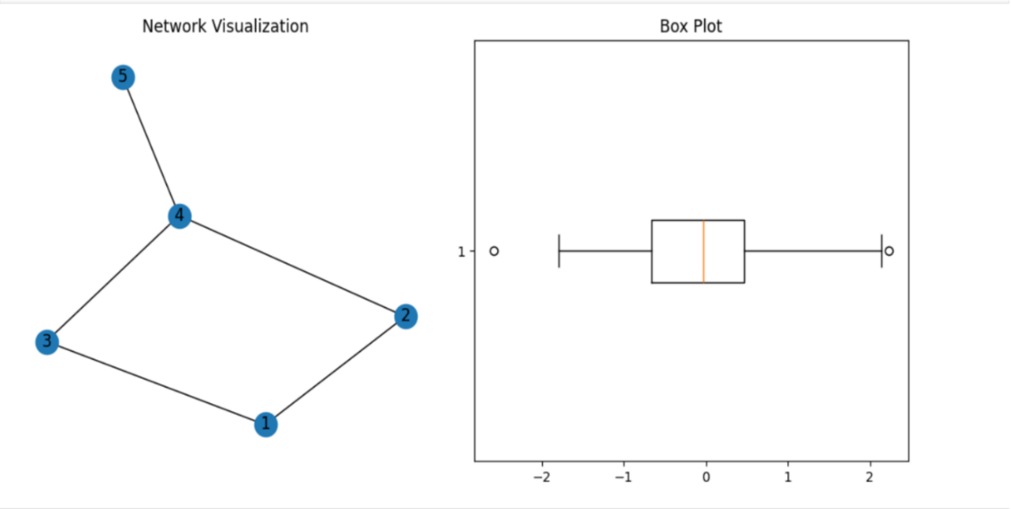


12. Graph visualization of a network using maximum, minimum, median, first quartile and third quartile.

Twelve.py

```
import networkx as nx

import matplotlib.pyplot as plt

import numpy as np

# Create and visualize the network

G = nx.Graph([(1, 2), (1, 3), (2, 4), (3, 4), (4, 5)])

pos = nx.spring_layout(G)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

nx.draw(G, pos, with_labels=True)

plt.title("Network Visualization")

# Generate data and plot box plot

data = np.random.randn(100)

plt.subplot(1, 2, 2)

plt.boxplot(data, vert=False)

plt.title("Box Plot")

plt.tight_layout()

plt.show()
```

OUTPUT


Network Visualization
Box Plot