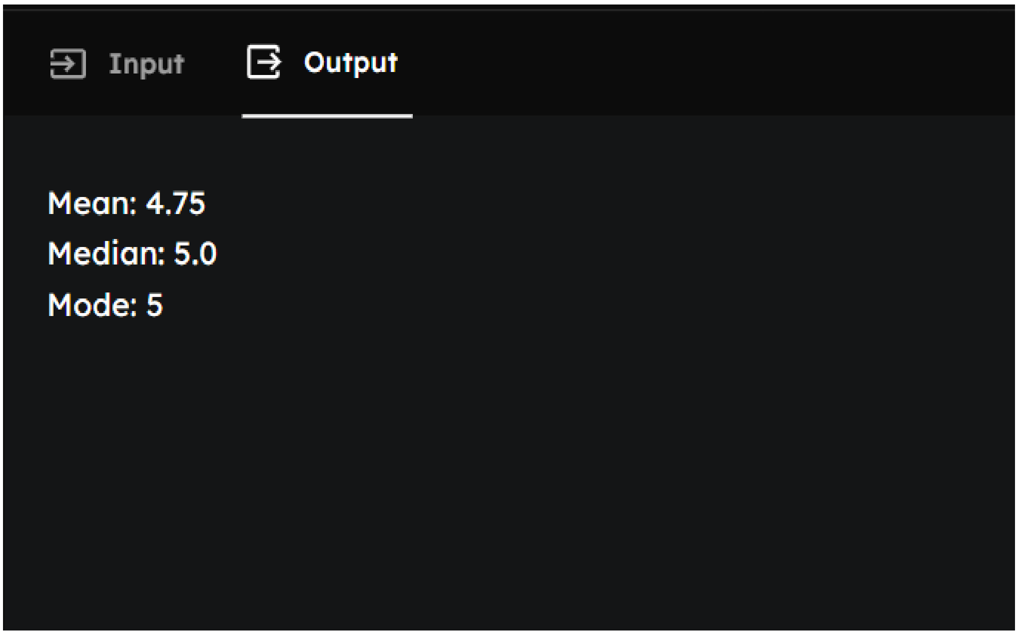
1. Write a python program to find mean, mode, median.

```
import statistics

# Sample data
data = [1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 8, 9]

# Calculate mean
mean_value = statistics.mean(data)
print(f"Mean: {mean_value}")

# Calculate median
median_value = statistics.median(data)
print(f"Median: {median_value}")

# Calculate mode
mode_value = statistics.mode(data)
print(f"Mode: {mode_value}")
```

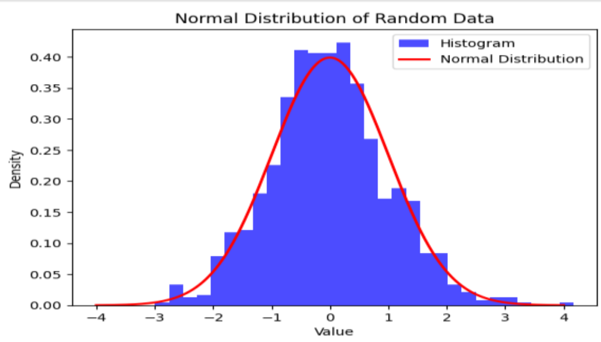
#OUTPUT



2. Write a python program to typical normal data distribution.

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import norm


# Generate random data

data = np.random.normal(size=1000)


# Plot histogram

plt.hist(data, bins=30, density=True, alpha=0.7, color='blue', label='Histogram')


# Plot normal distribution curve

x = np.linspace(-4, 4, 1000)

plt.plot(x, norm.pdf(x), color='red', linewidth=2, label='Normal Distribution')


# Add labels, title, and legend

plt.xlabel("Value")

plt.ylabel("Density")

plt.title('Normal Distribution of Random Data')

plt.legend()

plt.show()
```

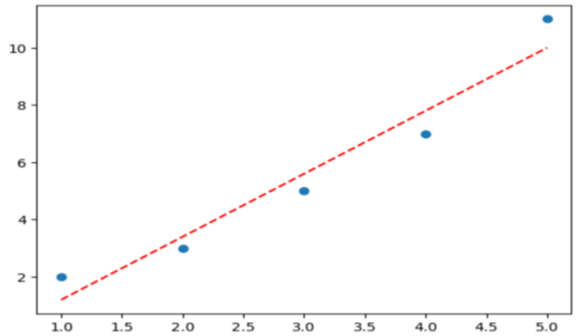
#OUTPUT



3. Write a python program to draw scatter plot of linear regression.

```
# Import necessary libraries
import matplotlib.pyplot as plt
import numpy as np

# Create some data
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 3, 5, 7, 11])

# Calculate the regression line
z = np.polyfit(x, y, 1)
p = np.poly1d(z)

# Plot the data
plt.scatter(x, y)

# Plot the regression line
plt.plot(x, p(x), "r--")

# Show the plot
plt.show()
```

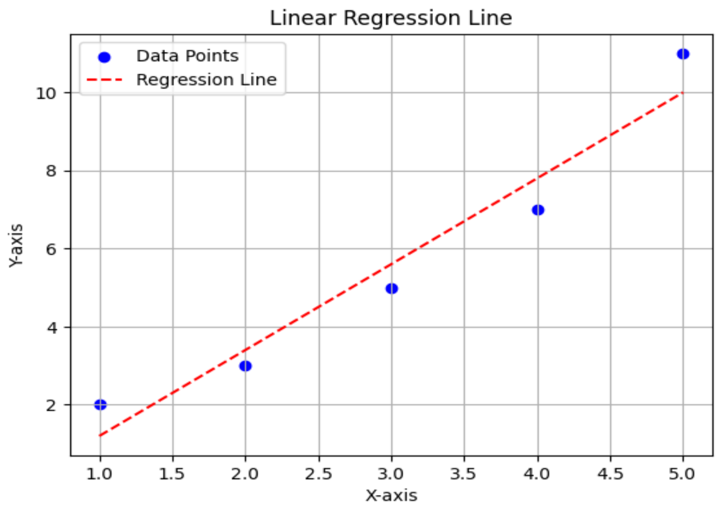
#OUTPUT



4. Write a python program to draw the line of Linear Regression.

```
# Import necessary libraries
import matplotlib.pyplot as plt
import numpy as np

# Create some sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 3, 5, 7, 11])

# Calculate the coefficients for the linear regression line
z = np.polyfit(x, y, 1)  # 1 indicates a linear fit (degree 1)
p = np.poly1d(z)         # Create a polynomial object for the regression line

# Create a scatter plot of the data points
plt.scatter(x, y, color='blue', label='Data Points')

# Plot the linear regression line
plt.plot(x, p(x), color='red', linestyle='--', label='Regression Line')

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Linear Regression Line')
plt.legend()

# Add a grid for better readability
plt.grid()

# Show the plot
plt.show()
```

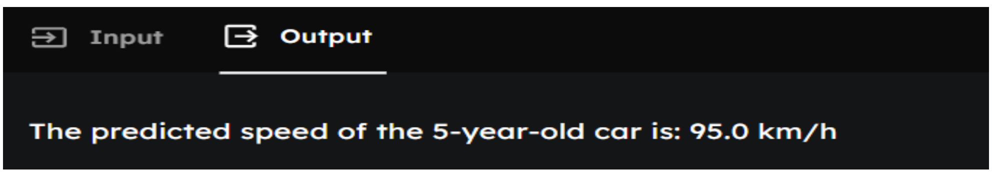
#OUTPUT



5. Write a python program to predict the speed of a 5 years old car.

```
def predict_speed(mileage):
    # Assume the initial speed is 100 km/h
    initial_speed = 100

    # Assume the speed decreases by 1 km/h for every 10,000 km
    speed_decrease = mileage / 10000

    # Calculate the predicted speed
    predicted_speed = initial_speed - speed_decrease

    # Ensure the predicted speed doesn't go below zero
    if predicted_speed < 0:
        predicted_speed = 0

    return predicted_speed

# Set a fixed mileage value
mileage = 50000  # Example: 50,000 km

# Predict the speed
predicted_speed = predict_speed(mileage)

# Print the predicted speed
print("The predicted speed of the 5-year-old car is:", predicted_speed, "km/h")
```

#OUTPUT



6. Write a python program to print the coefficient values of the regression object.

```
# Import necessary libraries
from sklearn.linear_model import LinearRegression
import numpy as np

# Create some sample data
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([2, 3, 5, 7, 11])  # Dependent variable

# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X, y)

# Print the coefficient values
print("Intercept:", model.intercept_)
print("Slope:", model.coef_)
```

#OUTPUT

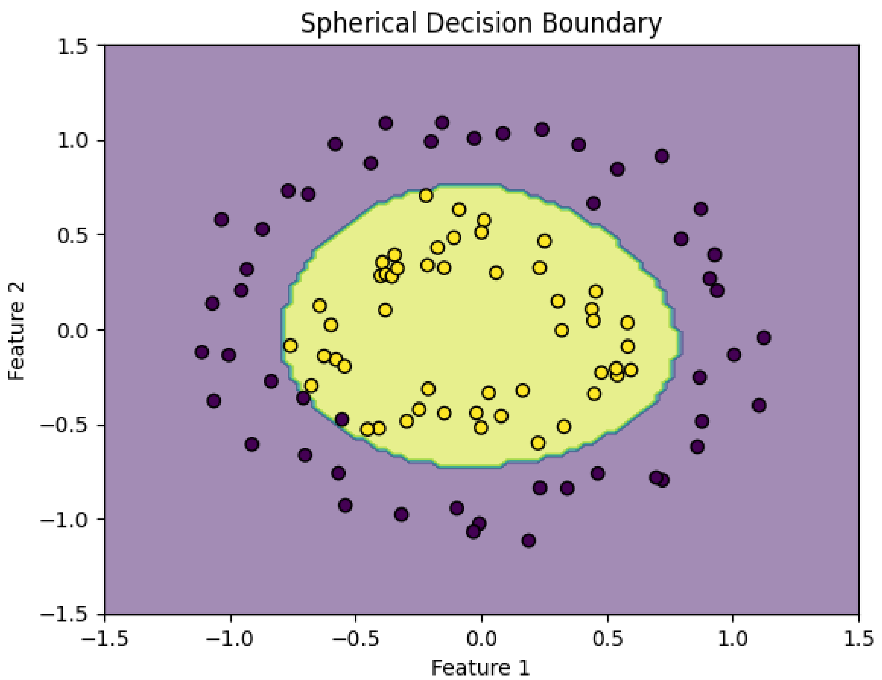
Input Output

```
Intercept: -1.0000000000000018
Slope: [2.2]
```

7. Write a python program to 2d binary classification data generated by make_circles() have a spherical decision boundary.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.svm import SVC

# Generate 2D binary classification data
X, y = make_circles(n_samples=100, noise=0.1, factor=0.5)

# Train an SVM with an RBF kernel
model = SVC(kernel='rbf')
model.fit(X, y)

# Create a grid to plot the decision boundary
xx, yy = np.meshgrid(np.linspace(-1.5, 1.5, 100), np.linspace(-1.5, 1.5, 100))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

# Plot the data and the decision boundary
plt.contourf(xx, yy, Z, alpha=0.5)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
plt.title('Spherical Decision Boundary')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

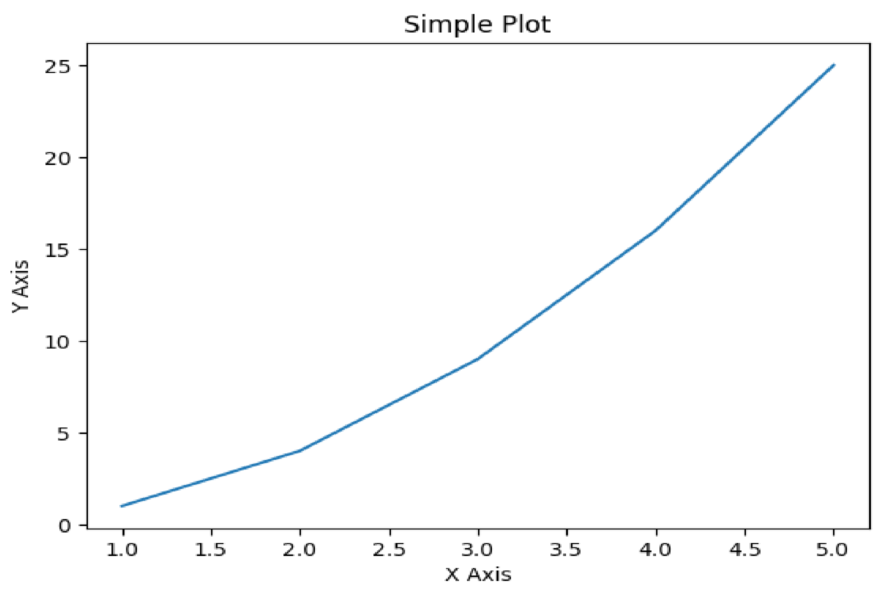
#OUTPUT



8. Write a python program to display the plot we can use the functions plot() and show() from pyplot.

```
import matplotlib.pyplot as plt

# Data for the plot
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

# Create the plot
plt.plot(x, y)

# Add title and labels
plt.title('Simple Plot')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')

# Display the plot
plt.show()
```

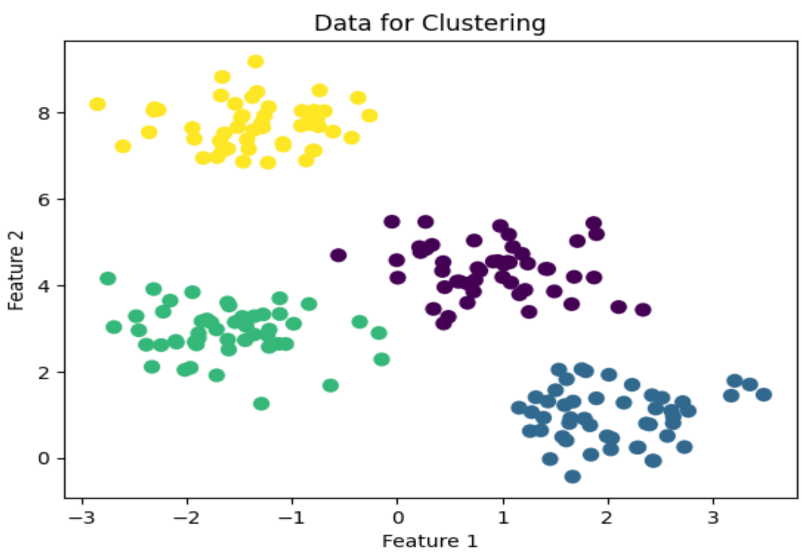
#OUTPUT



9. Write a python program to data generated by the function make_blobs() are blobs that can be utilized for clustering.

```
# Import necessary libraries

from sklearn.datasets import make_blobs

import matplotlib.pyplot as plt


# Generate data for clustering

X, y = make_blobs(n_samples=200, centers=4, cluster_std=0.60, random_state=0)


# Plot the data

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='viridis')


# Add title and labels

plt.title('Data for Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')


# Display the plot

plt.show()
```

#OUTPUT



10. Write a python program to random multi-label classification data is created by the function make make_multilabel_classification().

```
from sklearn.datasets import make_multilabel_classification


# Generate synthetic multi-label classification data

X, y = make_multilabel_classification(n_samples=100, n_features=20, n_classes=5, n_labels=2, random_state=42)


# Display shapes and sample labels

print("Feature matrix shape:", X.shape)

print("Label matrix shape:", y.shape)

print("Sample of labels for first 10 samples:\n", y[:10])
```

#OUTPUT

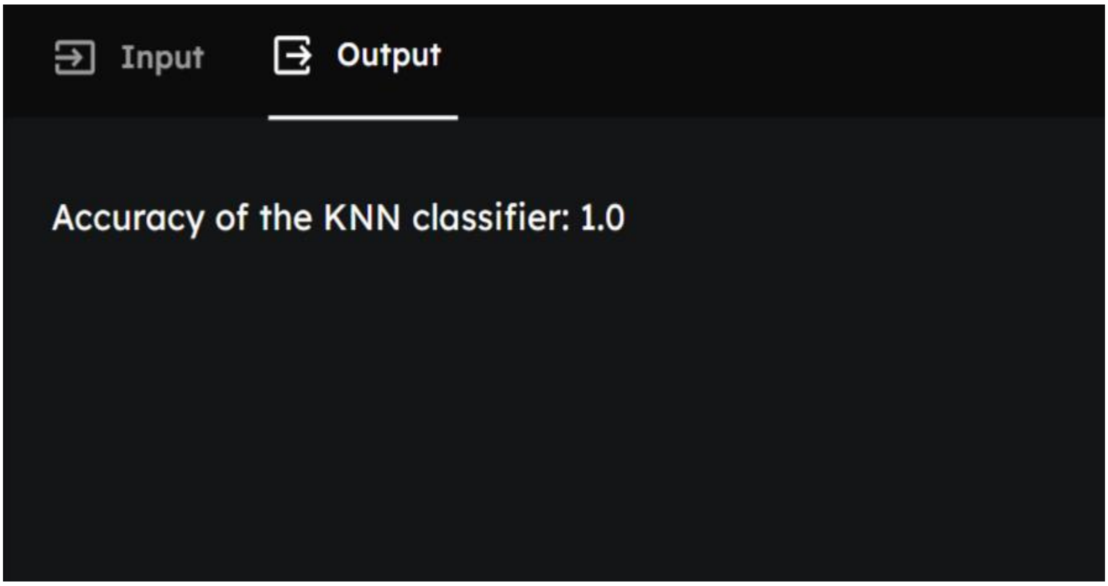
Input   Output

```
Feature matrix shape: (100, 20)
Label matrix shape: (100, 5)
Sample of labels for first 10 samples:
 [[0 0 0 1 0]
 [1 1 1 0 0]
 [0 0 1 1 0]
 [1 0 0 0 0]
 [1 0 1 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 1 0 0 0]
 [1 1 1 1 1]
 [1 1 1 1 0]]
```

11. Write a python program to implement the KNN algorithm.

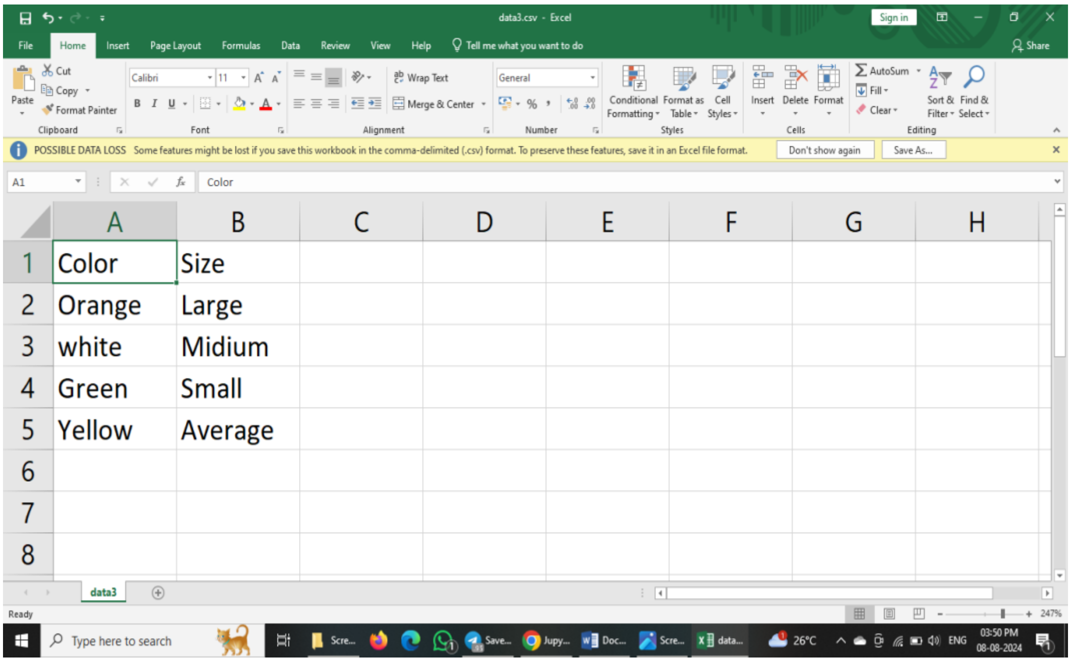
```
from sklearn.datasets import make_blobs

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier


# Generate synthetic data

X, y = make_blobs(n_samples=300, centers=3, random_state=42)


# Split the dataset and fit the KNN model

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

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)


# Make predictions and print accuracy

accuracy = knn.score(X_test, y_test)

print("Accuracy of the KNN classifier:", accuracy)
```

#OUTPUT



12. Write a python program to creating a dataframe to implement one hot encoding from CSV file.

Create a .csv file in an Exel:




```
import pandas as pd

# Load CSV file into a DataFrame
df = pd.read_csv('data3.csv')

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Perform one-hot encoding on categorical columns
df_encoded = pd.get_dummies(df)

# Display the DataFrame after one-hot encoding
print("\nDataFrame after One-Hot Encoding:")
print(df_encoded)
```

#OUTPUT

```
Original DataFrame:
    Color     Size
0  Orange    Large
1   white   Midium
2   Green    Small
3  Yellow  Average

DataFrame after One-Hot Encoding:
   Color_Green  Color_Orange  Color_Yellow  Color_white  Size_Average  \
0        False          True         False        False         False   
1        False         False         False         True         False   
2         True         False         False        False         False   
3        False         False          True        False          True   

   Size_Large  Size_Midium  Size_Small  
0        True        False       False  
1       False         True       False  
2       False        False        True  
3       False        False       False  
```