EXPERIMENT: 1 **Date: 20/03/25**

AIM: How to calculate important numbers based on data sets, how to use various Python modules and how to make functions that are able to predict the outcome based on what we have learned (Small Dataset)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
df = pd.read_csv("score.csv")
df.head()
```

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

```
df.isnull().sum()
```

	0
Hours	0
Scores	0

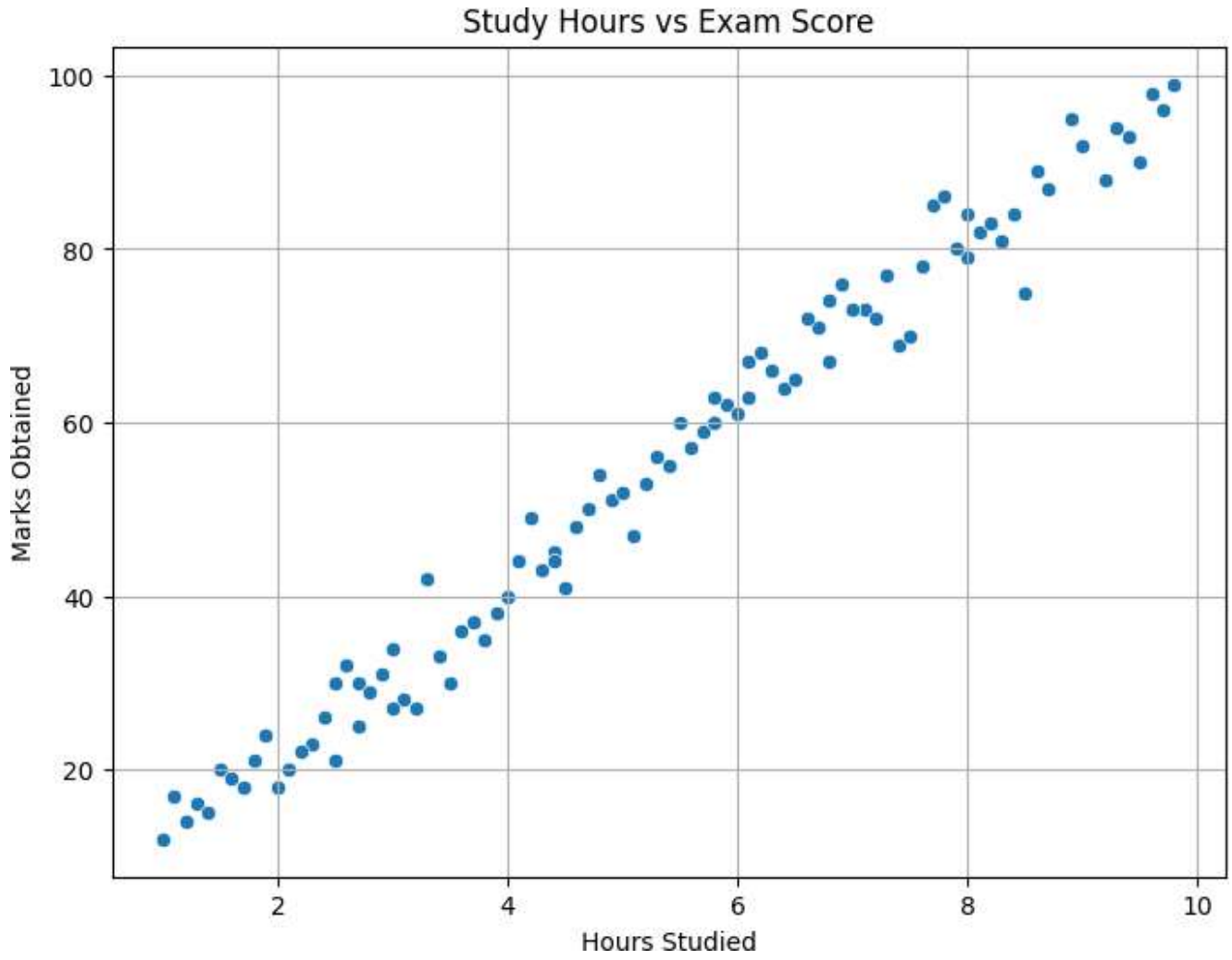
dtype: int64

```
mean = df.mean()
median = df.median()
```

```
mode = df.mode().iloc[0]
std = df.std()
var = df.var()
print("Mean:\n", mean)
print("\nMedian:\n", median)
print("\nMode:\n", mode)
print("\nStandard Deviation:\n", std)
print("\nVariance:\n", var)
```

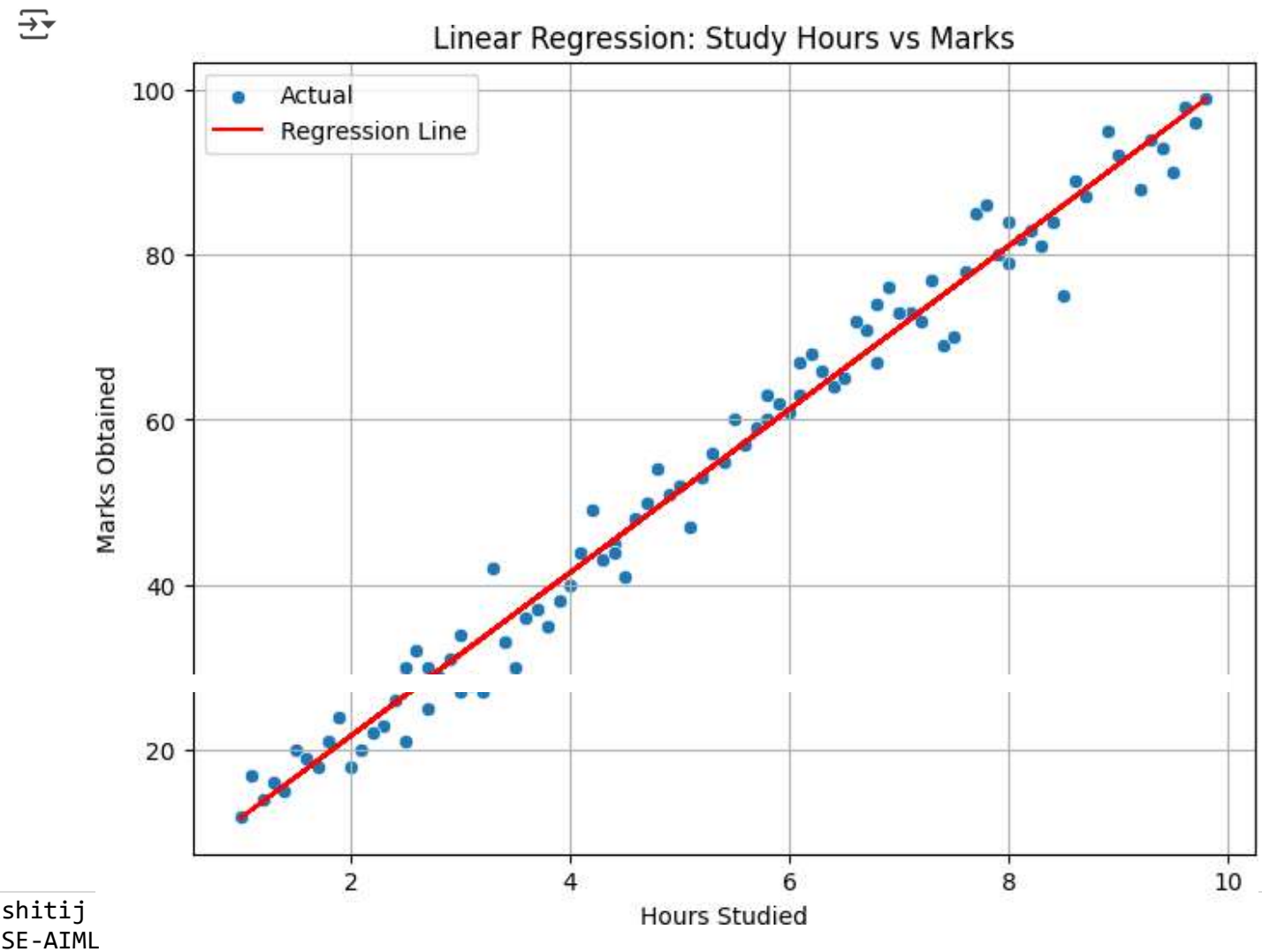
```
Mean:
 Hours       5.267708
Scores     54.020833
dtype: float64

Median:
 Hours       5.25
Scores     54.50
dtype: float64

Mode:
 Hours       2.5
Scores     30.0
Name: 0, dtype: float64

Standard  Deviation:
 Hours       2.503030
Scores     25.017459
dtype: float64

Variance:
 Hours        6.265157
Scores     625.873246
dtype: float64
```

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Hours', y='Scores', data=df)
plt.title('Study Hours vs Exam Score')
plt.xlabel('Hours Studied')
plt.ylabel('Marks Obtained')
plt.grid(True)
plt.show()
```




```
X = df[['Hours']]
y = df['Scores']

model = LinearRegression()
model.fit(X, y)

y_pred = model.predict(X)

plt.figure(figsize=(8, 6))
sns.scatterplot(x='Hours', y='Scores', data=df, label='Actual')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.title('Linear Regression: Study Hours vs Marks')
```

```
plt.xlabel('Hours Studied')
plt.ylabel('Marks Obtained')
plt.legend()
plt.grid(True)
plt.show()
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)
print(f"\nModel Evaluation:\nMean Squared Error: {mse:.2f}\nR² Score: {r2:.2f}

def predict_marks(hours):
    prediction = model.predict(np.array([[hours]]))[0]
    print(f"Predicted marks for {hours} hours of study: {prediction:.2f}")
    return prediction

predict_marks(5)
```



```
Model Evaluation:
Mean Squared Error: 11.54
R² Score: 0.98
Predicted marks for 5 hours of study: 51.37
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarnin
  warnings.warn(
np.float64(51.37016792434766)
```

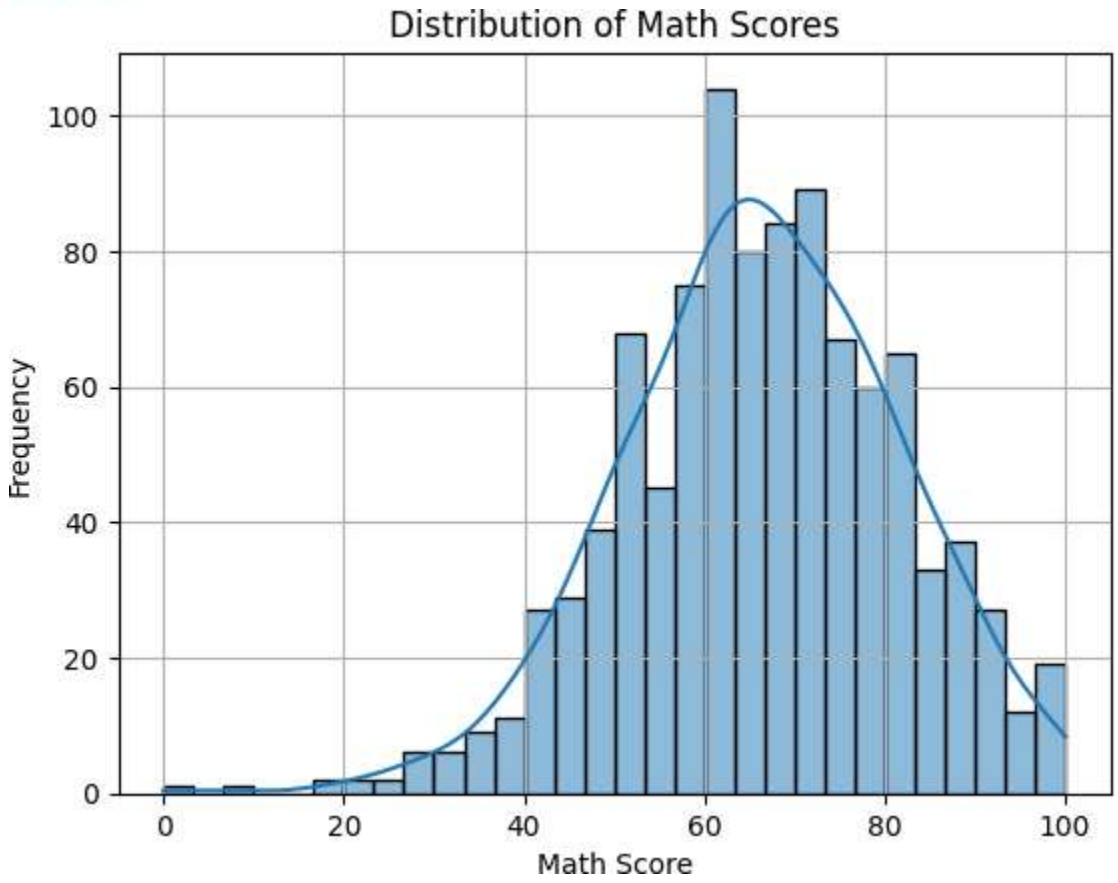
EXPERIMENT: 2 **Date: 27/03/25**

AIM: How can we get Big Data Sets, Learn: Data Distribution, Normal data distribution, Random Data Distribution, Scatter Plot.

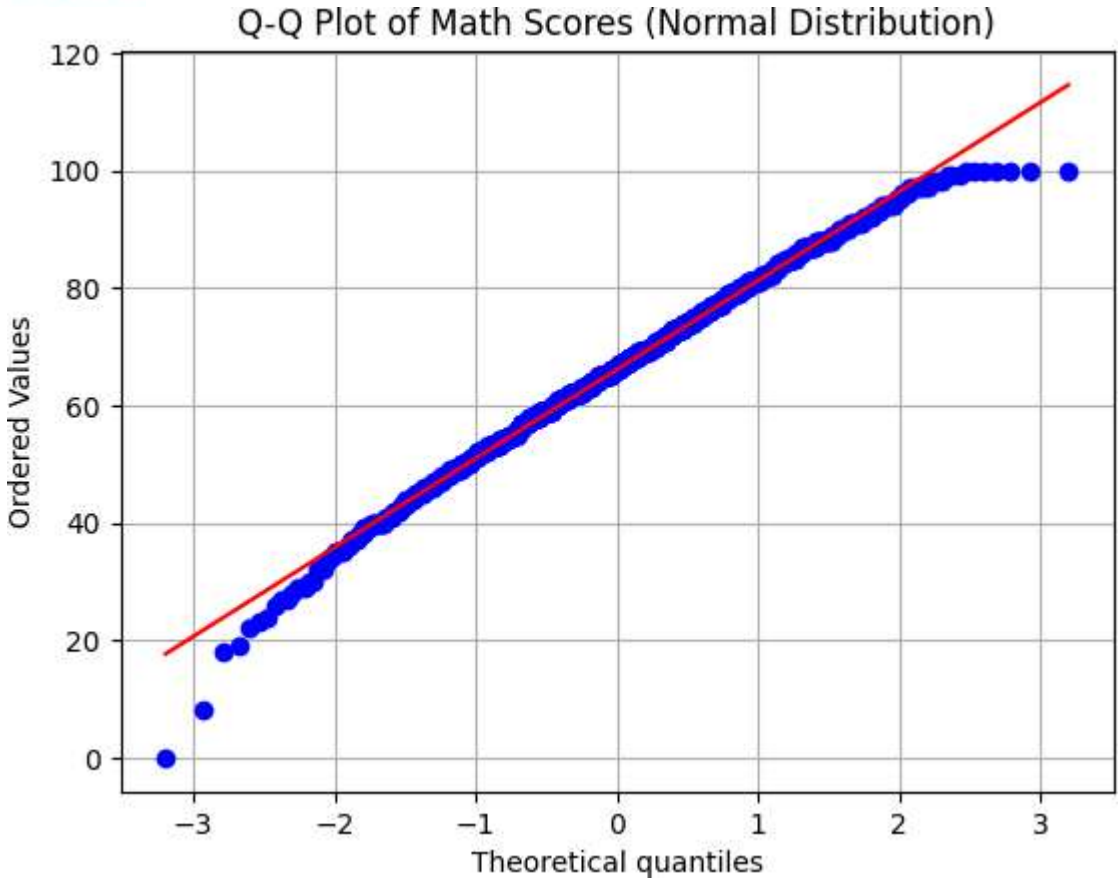
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats


df = pd.read_csv('StudentsPerformance.csv')
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writin scor
0	female	group B	bachelor's degree	standard	none	72	72	7
1	female	group C	some college	standard	completed	69	90	8

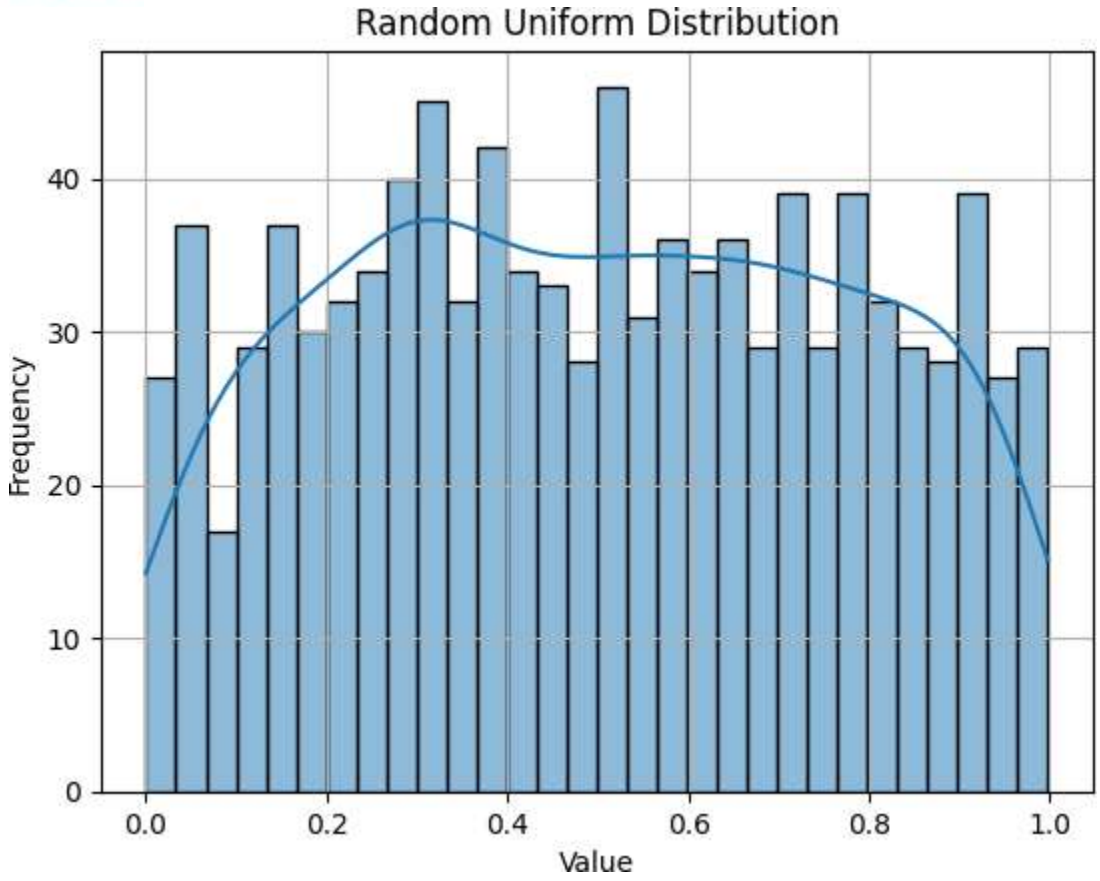
```
sns.histplot(df['math score'], kde=True, bins=30)
plt.title('Distribution of Math Scores')
plt.xlabel('Math Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



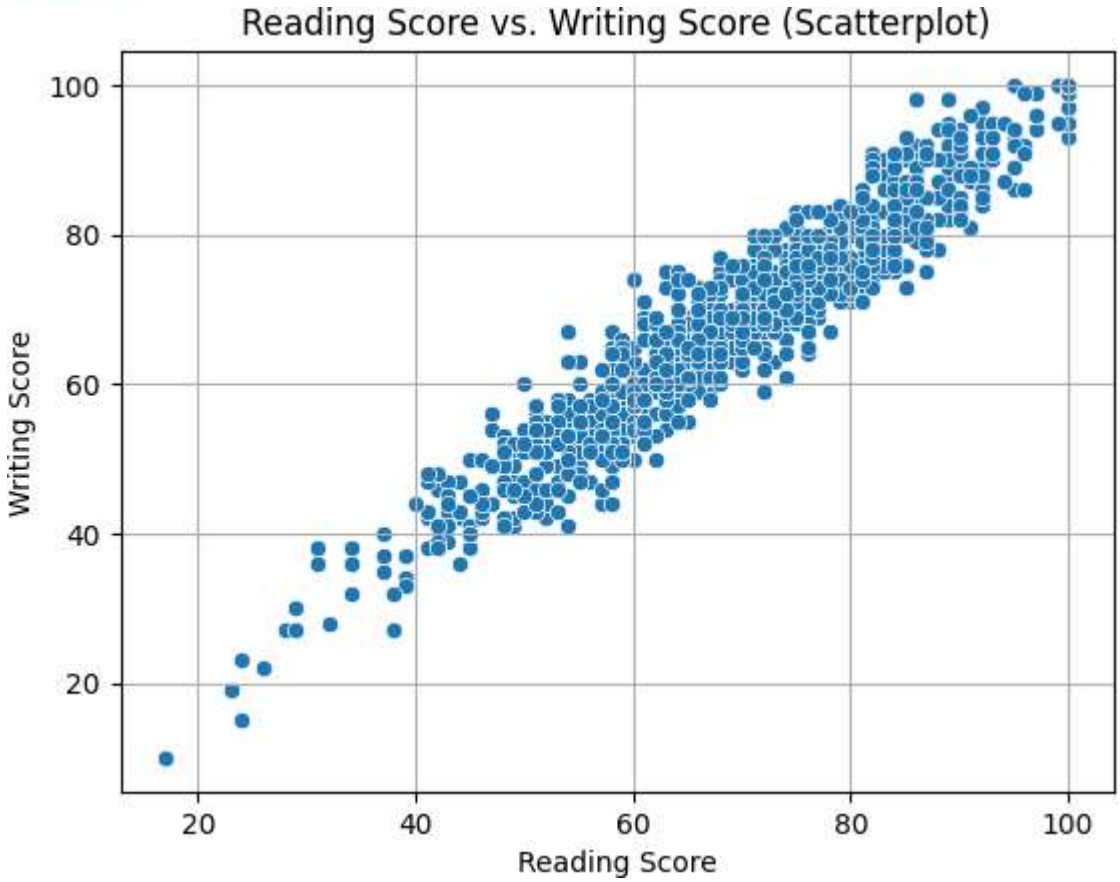

```
stats.probplot(df['math score'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Math Scores (Normal Distribution)')
plt.grid(True)
plt.show()
```




```
random_data = np.random.rand(1000)

sns.histplot(random_data, kde=True, bins=30)
plt.title('Random Uniform Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```




```
sns.scatterplot(x='reading score', y='writing score', data=df)
plt.title('Reading Score vs. Writing Score (Scatterplot)')
plt.xlabel('Reading Score')
plt.ylabel('Writing Score')
plt.grid(True)
plt.show()
```


Reading Score vs. Writing Score (Scatterplot)
Writing Score
Reading Score
100
80
60
40
20
20
40
60
80
100

EXPERIMENT: 3 **Date: 04/04/25**

AIM: Write a program for principal component analysis of iris dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA


iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names

df = pd.DataFrame(X, columns=feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
df1 = pd.DataFrame(X_scaled, columns=feature_names)
df1.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

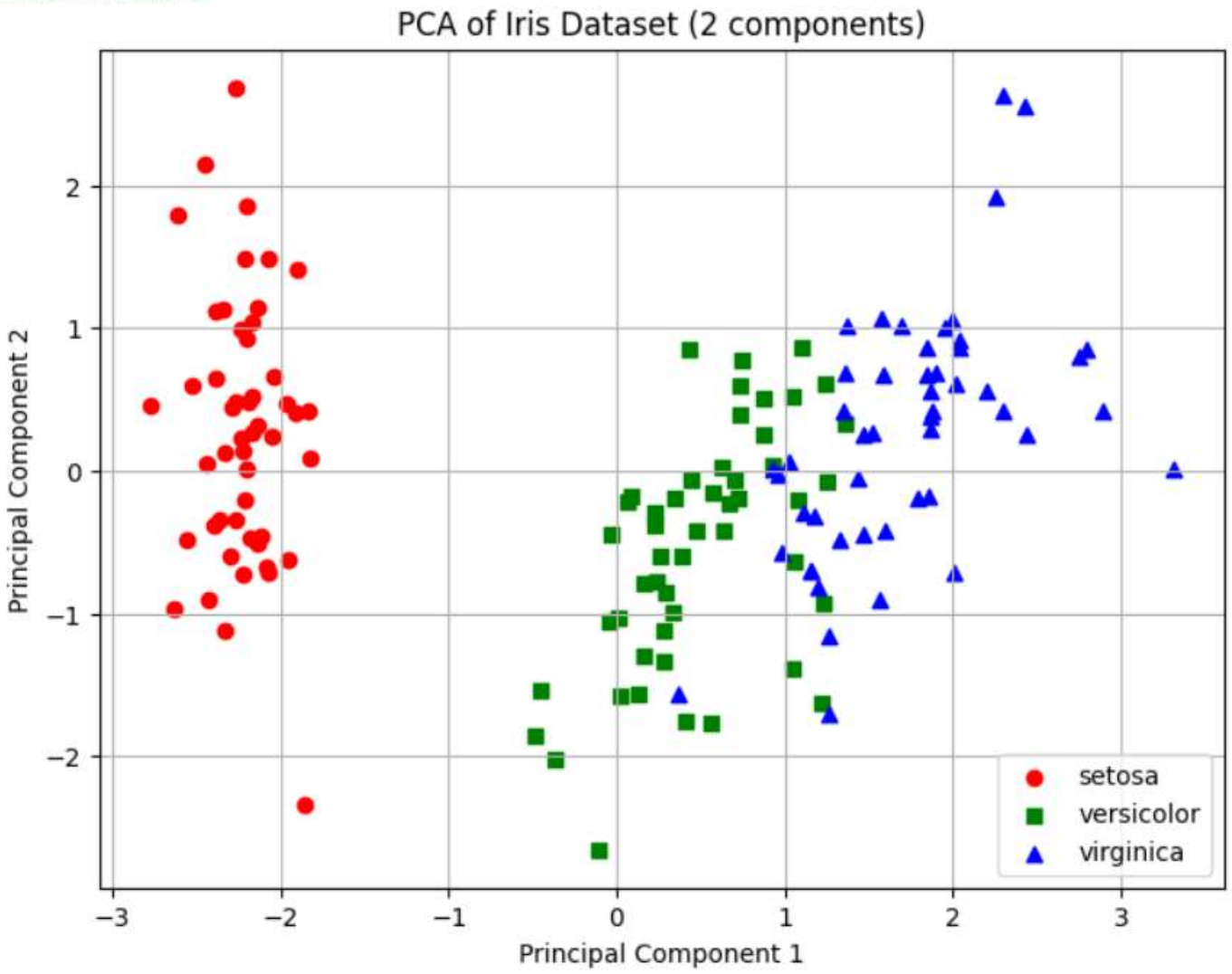
```
n_components = 2
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X_scaled)
df2 = pd.DataFrame(X_pca, columns=[f'PC{i+1}' for i in range(n_components)])
df2.head()
```

	PC1	PC2
0	-2.264703	0.480027
1	-2.080961	-0.674134
2	-2.364229	-0.341908
3	-2.299384	-0.597395
4	-2.389842	0.646835

```
explained_variance_ratio = pca.explained_variance_ratio_
print(f"\nExplained Variance Ratio: {explained_variance_ratio}")
```

```
print(f"Total Explained Variance: {sum(explained_variance_ratio)}")
pca_df = df2.copy()
pca_df['target'] = y

if n_components == 2:
    plt.figure(figsize=(8, 6))
    colors = ['r', 'g', 'b']
    markers = ['o', 's', '^']
    for i, target in enumerate(np.unique(y)):
        subset = pca_df[pca_df['target'] == target]
        plt.scatter(subset['PC1'], subset['PC2'],
                    c=colors[i], marker=markers[i],
                    label=target_names[target])

    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.title('PCA of Iris Dataset (2 components)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

```
Explained Variance Ratio: [0.72962445 0.22850762]
Total Explained Variance: 0.9581320720000166
```


PCA of Iris Dataset (2 components)
Principal Component 2
Principal Component 1
setosa
versicolor
virginica
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

EXPERIMENT: 4 **Date: 11/04/25**

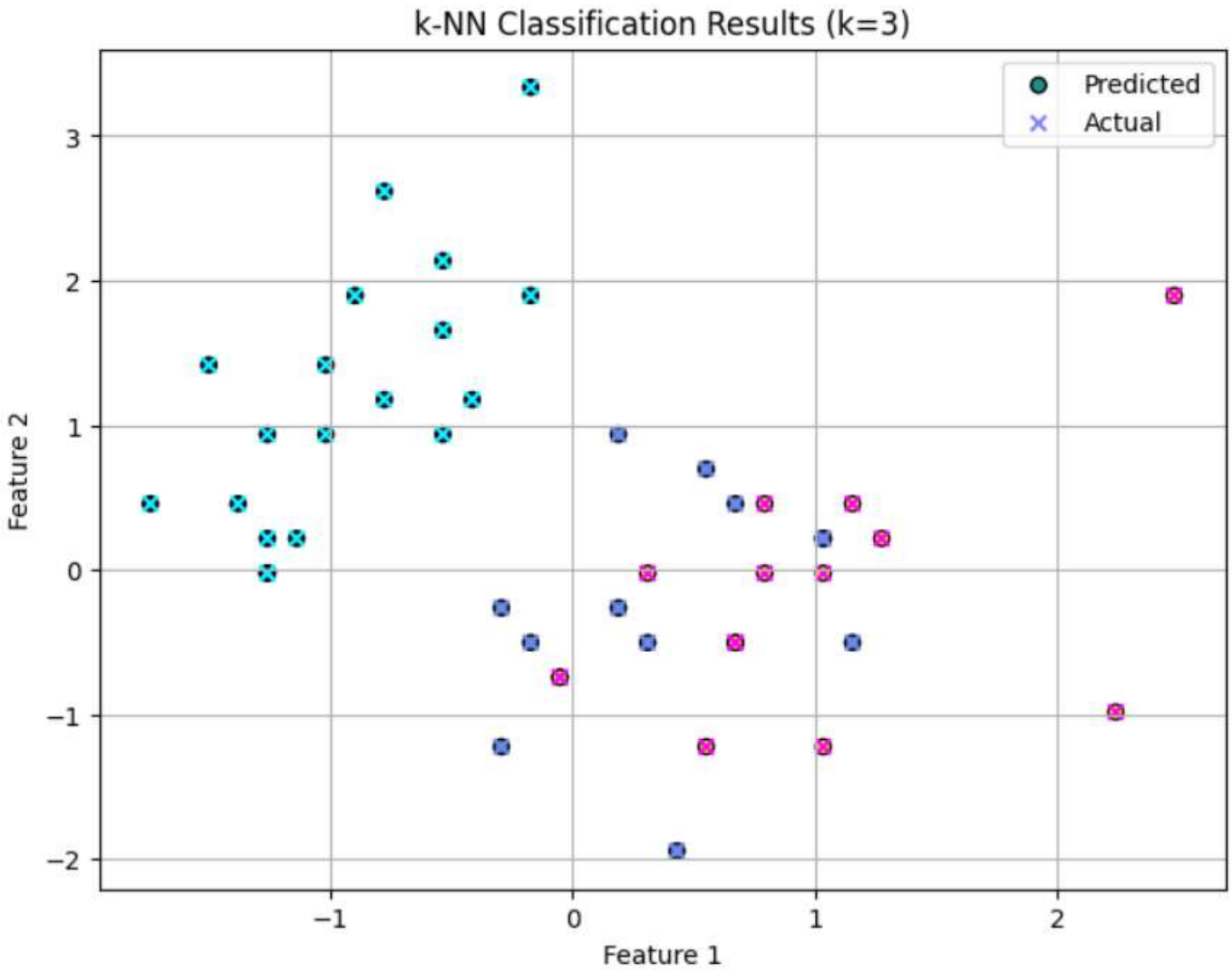
AIM: Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_m
import matplotlib.pyplot as plt


iris = load_iris()
X = iris.data
y = iris.target


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

k = 3
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)


print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13
```

```
              accuracy                           1.00        45
             macro avg       1.00      1.00      1.00        45
          weighted avg       1.00      1.00      1.00        45


                Accuracy Score: 1.0
```

```
plt.figure(figsize=(8, 6))
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='viridis', edgecolor='k',
marker='o', label='Predicted')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='cool', edgecolor='k',
marker='x', label='Actual')
plt.title(f"k-NN Classification Results (k={k})")
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```



EXPERIMENT: 5 **Date: 17/04/25**

AIM: The probability that it is friday and that a student is absent is 3% since there are 5 college days in a week, the probability that it is friday is 20%. What is the probability that a student is absent given that today is friday. Write a python program for it.

```
def probabilityAbsentAndFriday(probFridayAndAbsent, probFriday):
  result = int((probFridayAndAbsent/probFriday)*100)
  return result

probFridayAndAbsent = float(input("Enter the probability of absent and friday : "))
probFriday = float(input("Enter the probability of friday : "))
result = probabilityAbsentAndFriday(probFridayAndAbsent, probFriday)
print(f"The probability that a student is absent given today is friday is
{result}%")
```

```
Enter the probability of absent and friday : 0.03
Enter the probability of friday : 0.2
The probability that a student is absent given today is friday is 15%
```