Student Performance Predictor

AI Project Report

For

Session (2024-2025)

Submitted by

Imran Ahmad

202410116100092

Harshit Singh

202410116100089

Submitted in partial fulfilment of the

Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Mr. Apoorv Jain Sir

Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206

1.Introduction **:-** Student performance is a key metric in educational research, as it helps educators and policymakers design better learning strategies. Various factors influence student success, including study habits, past academic performance, extracurricular involvement, sleep patterns, and exam preparation methods.

This study aims to analyze the impact of these variables on students' performance, measured through a Performance Index. By leveraging data analytics and machine learning techniques, we seek to uncover patterns and relationships that can provide actionable insights for academic improvement.

2.Methodology

Data Collection

The dataset used in this study consists of 10,000 student records containing six key variables:

- Hours Studied (Numerical)
- Previous Scores (Numerical)
- Extracurricular Activities (Categorical: Yes/No)
- Sleep Hours (Numerical)
- Sample Question Papers Practiced (Numerical)
- Performance Index (Target variable, Numerical)

The dataset was imported into a Python-based environment for analysis. Data preprocessing, visualization, and modeling were conducted using libraries such as pandas, NumPy, seaborn, and scikit-learn.

Data Preprocessing

- Checking for Missing and Duplicate Values: No missing values were detected, but duplicate entries were identified and handled.
- Data Type Verification: All columns were validated to match their expected data types.
- Categorical Variable Encoding: The "Extracurricular Activities" column was label-encoded (Yes = 1, No = 0).
- Feature Scaling: Features were normalized using MinMaxScaler for improved model performance.

Exploratory Data Analysis (EDA)

- Descriptive Statistics were computed to understand the distribution of numerical values.
- Box Plots and Count Plots were used to analyze variable distributions.
- Correlation Heatmaps were generated to identify relationships between features.

Model Development

The dataset was split into training (80%) and testing (20%) subsets. A Linear Regression Model was trained using the independent variables to predict student performance. Model evaluation metrics included:

- R^2 Score: 0.99 on training data, indicating a strong fit.
- Mean Absolute Error (MAE): Used to assess prediction accuracy.
- Scatter Plots: Used to visualize actual vs. predicted values.

Project Implementation:

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Set visualization style
sns.set_style("whitegrid")
sns.set_palette("RdBu")


# Load dataset
```

```
data = pd.read_csv("/content/Student_Performance.csv")

# Display basic information
data.info()

# Check for missing values
data.isna().sum() / data.shape[0]

# Check for duplicates

data.duplicated().any()

# Encode categorical variables
data.describe(include = object)

# create function to visualized categorical column using count plot

def count_plot(column_name, hue = None, rotation = 0):
    """
    1) input : column name, column data type must be object or categorical
    3) output : cout plot using seaborn modules, unique values in x-axis and frequency in y-axis

    4) i use bar_label to show frequency of each unique values above each column in graph

    """

    graph = sns.countplot(x = column_name, data = data, hue = hue, order = data[column_name].value_counts().index)

    for container in graph.containers:

        graph.bar_label(container)



    plt.xticks(rotation = rotation)

    plt.show()


# create function that visualized numeric columns using box plot


def box_plot(x_axis = None, y_axis = None, hue = None, col = None):

    """

    input : x_axis, y_axis and hue column, column data type must be numeric in y_axis

    output : box plot to see distribution of column values such as min,max,mean,medien,std
    """
    sns.catplot(x = x_axis, y = y_axis, data = data, hue = hue, kind = "box", col = col)
    plt.xlabel(x_axis)
    plt.ylabel("FRQ")
    plt.show()

# number of unique values is relatively large, count plot more suitable for it


# first set figure size
plt.figure(figsize = (15,6))


# call function
count_plot(column_name = "Hours Studied")


# see distribution


box_plot(y_axis = "Previous Scores") # call function i create it in cell 11
# see unique values


data["Extracurricular Activities"].unique()
```

```
# output number of values count


plt.pie(data["Extracurricular Activities"].value_counts(), labels
= data["Extracurricular Activities"].value_counts().index,

        shadow = True, autopct = "%1.1f%%")

plt.show()

# output number of values count


plt.pie(data["Extracurricular Activities"].value_counts(), labels
= data["Extracurricular Activities"].value_counts().index,

        shadow = True, autopct = "%1.1f%%")

plt.show()

# number of unique values is relatively large, count plot
more suitable for it


# first set figure size

plt.figure(figsize = (15,6))


# call function


count_plot(column_name = "Sleep Hours")
```

```
# number of unique values is relatively large, count plot more suitable for it

# first set figure size
plt.figure(figsize = (15,6))

# call function
count_plot(column_name = "Sample Question Papers Practiced")
# What is "Hours Studied" and "Performance Index" distribution
box_plot(x_axis = "Hours Studied", y_axis = "Performance Index") # call function i create it in cell 11
# What is " Extracurricular Activities" and "Performance Index" distribution

box_plot(x_axis = "Extracurricular Activities", y_axis = "Performance Index") # call function i create it in cell 11

# What is "Extracurricular Activities" and "Performance Index" distribution

avg_performance_by_hours = data.groupby('Hours Studied')['Performance Index'].mean()
```

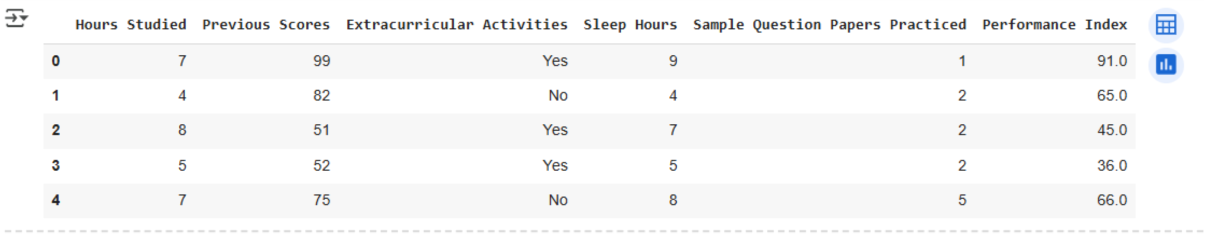
```
plt.plot(avg_performance_by_hours.index, avg_performance_by_hours.values)
plt.show()

# first visualize correlation matrix between numerical columns
plt.figure(figsize = (10,6))
sns.heatmap(data.select_dtypes(exclude = object).corr(), annot = True, fmt = ".2f", linewidths = 0.2)
plt.show()
# import libraries to model
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,r2_score
# create object from labelencoder
encoder = LabelEncoder()
data["Extracurricular Activities"] =  encoder.fit_transform(data["Extracurricular Activities"])
# Splitting data into Indipendent and Dependent Variable
Train = data.drop(columns = "Performance Index")
Target = data["Performance Index"]
```

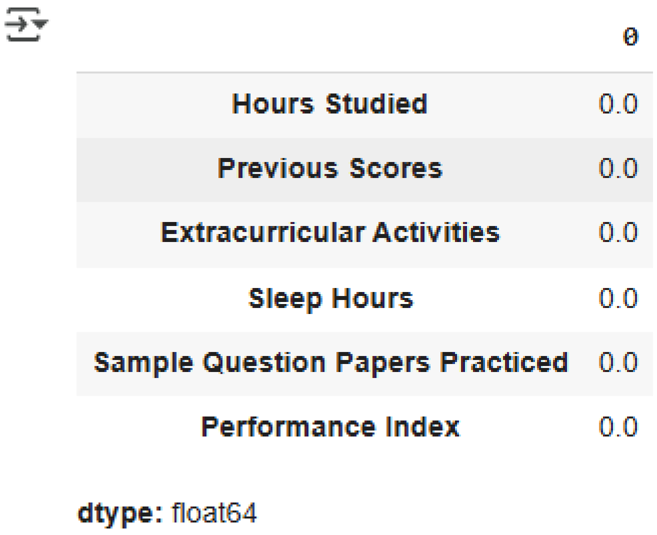
```
X_train, X_test, y_train, y_test = train_test_split(Train, Target,
test_size = 0.2, random_state = 42)
# see shape of splited data
print("x_train shape: ", X_train.shape)
print("y_train shape: ", y_train.shape)
print("x_test shape: ", X_test.shape)
print("y_test shape: ", y_test.shape)
# create object from RandomForestRegressor
model = LinearRegression()
# fit model
model.fit(X_train,y_train)
# Calculate the score of the model on the training data
model.score(X_train, y_train)
# see predicted values
predict = np.round(model.predict(X_test), decimals = 1)
# Real Values vs Predicted Values
pd.DataFrame({"Actual Performance" : y_test, "Predicted
Performance" : predict})
# Create scatter plot to see distribution
plt.scatter(y_test, predict)
plt.show()
# see mean absolute error
```

```
mean_absolute_error(y_test,predict)
# see score
r2_score(y_test,predict)
# see coefficients values
model.coef_
# see y intercept
model.intercept_
```

Output:-

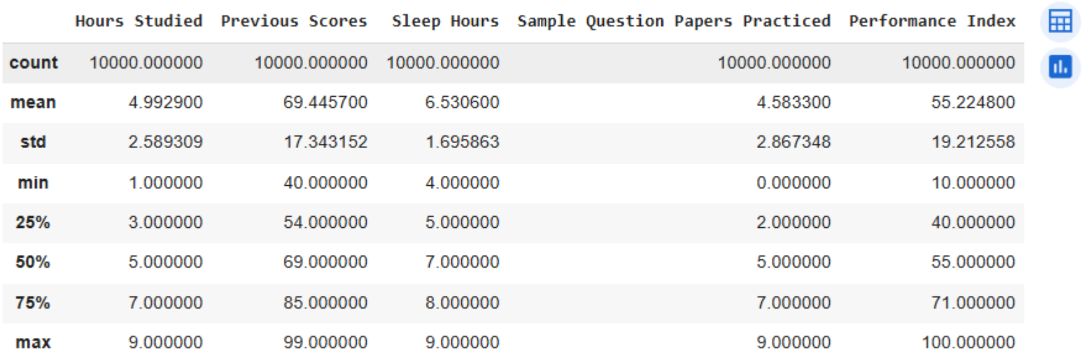


	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0

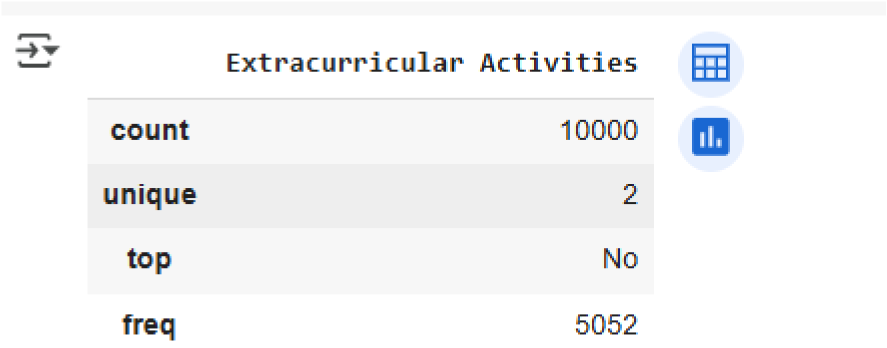


	0
Hours Studied	0.0
Previous Scores	0.0
Extracurricular Activities	0.0
Sleep Hours	0.0
Sample Question Papers Practiced	0.0
Performance Index	0.0

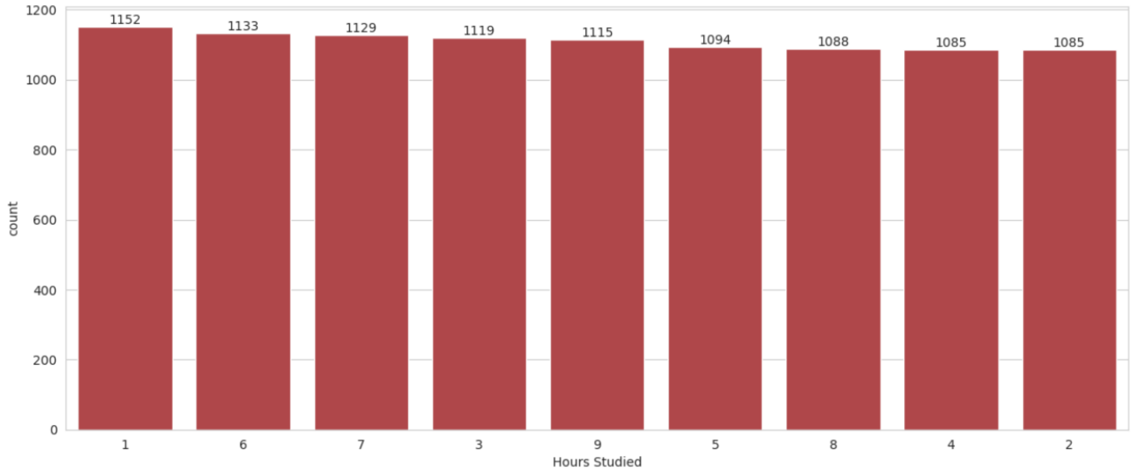
dtype: float64

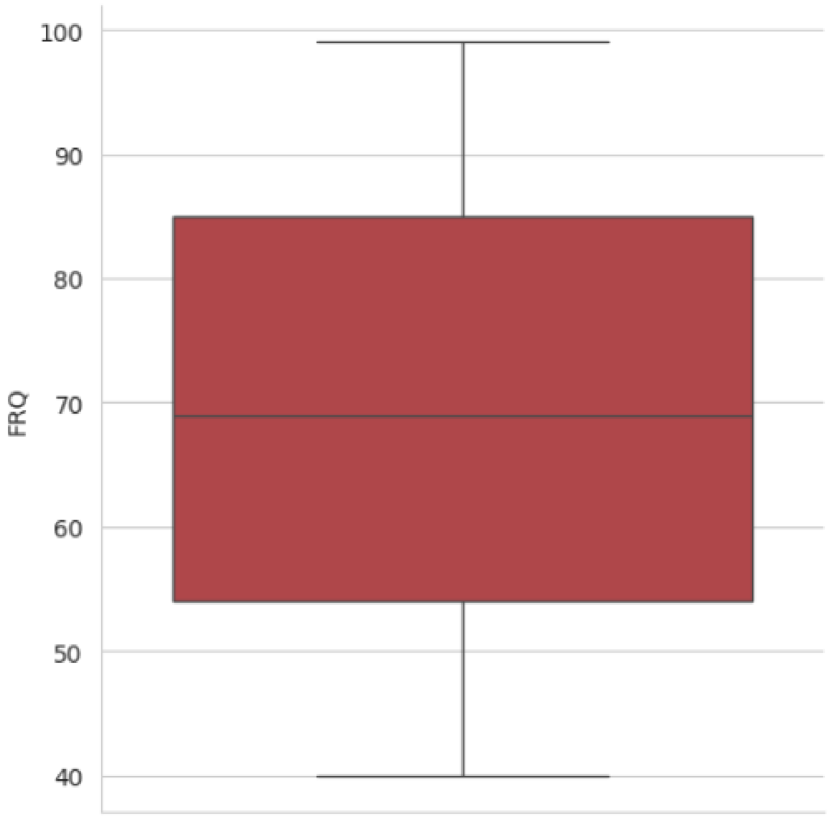


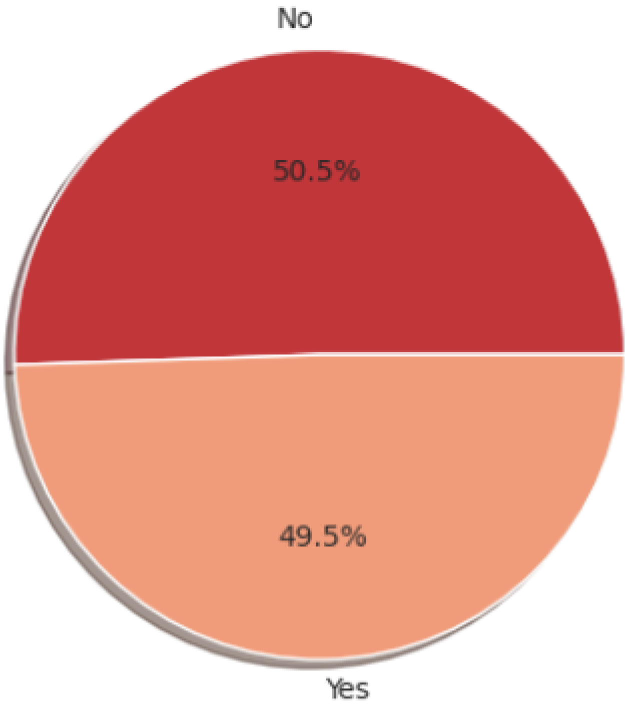
	Hours Studied	Previous Scores	Sleep Hours	Sample Question Papers Practiced	Performance Index
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	4.992900	69.445700	6.530600	4.583300	55.224800
std	2.589309	17.343152	1.695863	2.867348	19.212558
min	1.000000	40.000000	4.000000	0.000000	10.000000
25%	3.000000	54.000000	5.000000	2.000000	40.000000
50%	5.000000	69.000000	7.000000	5.000000	55.000000
75%	7.000000	85.000000	8.000000	7.000000	71.000000
max	9.000000	99.000000	9.000000	9.000000	100.000000

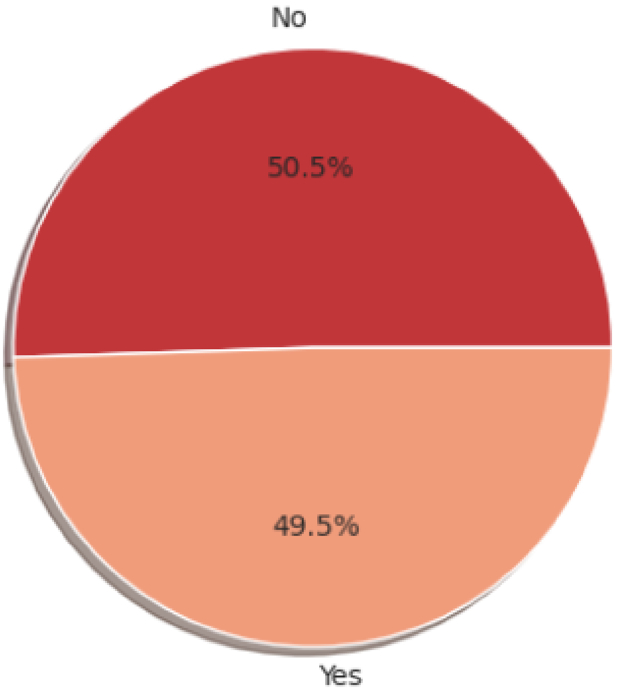


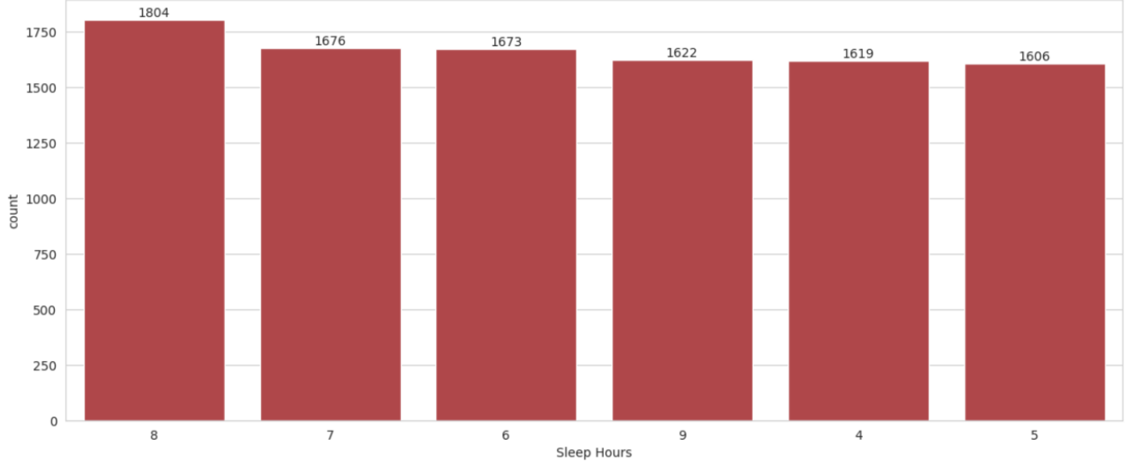
	Extracurricular Activities
count	10000
unique	2
top	No
freq	5052

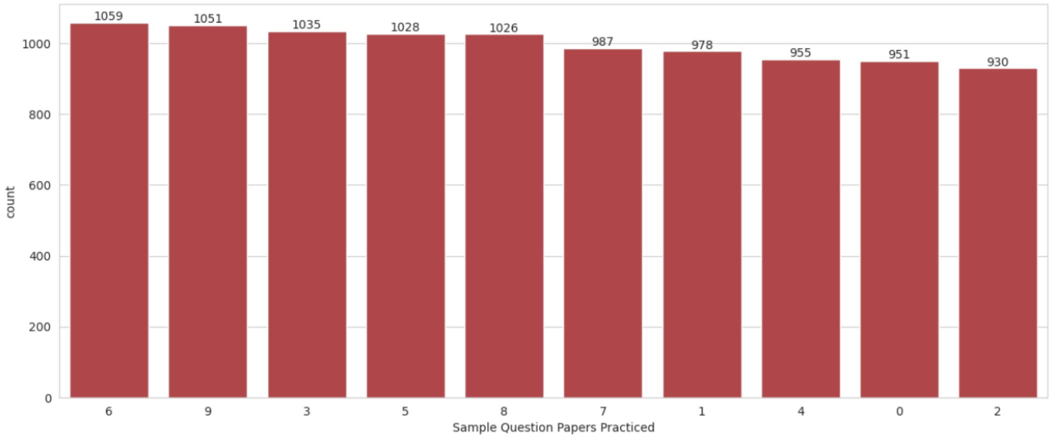



100
90
80
70
60
50
40
FRQ

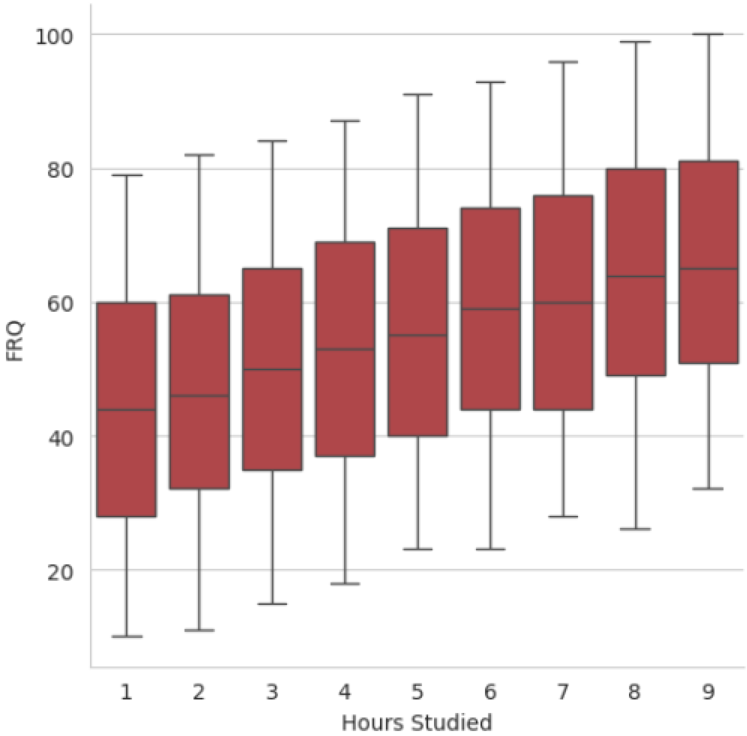

No
50.5%
49.5%
Yes


No
50.5%
49.5%
Yes

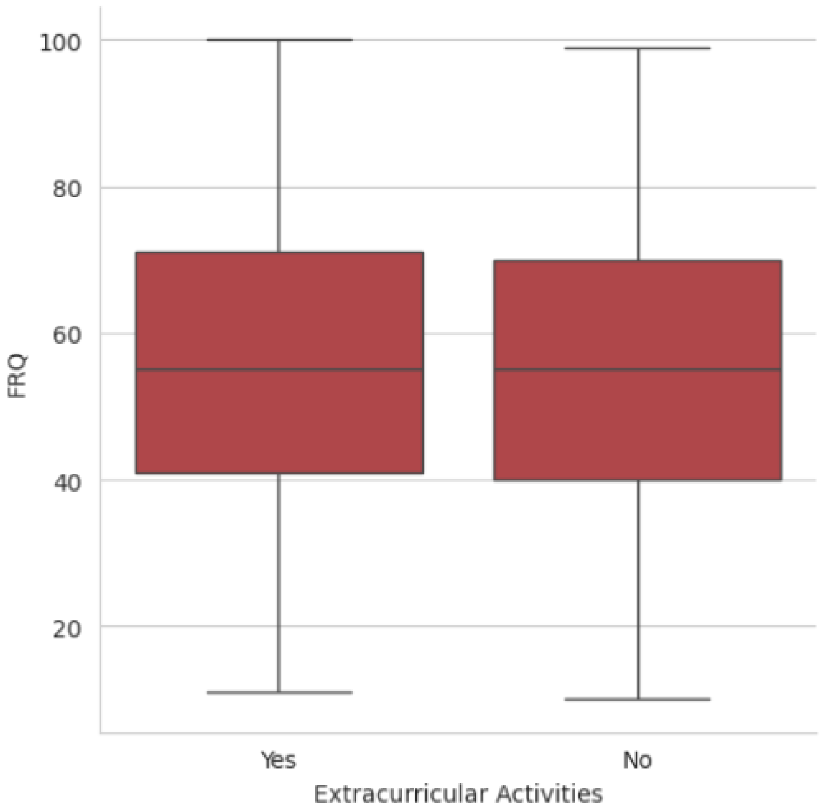

1804
1676
1673
1622
1619
1606
1750
1500
1250
1000
750
500
250
0
count
8
7
6
9
4
5
Sleep Hours

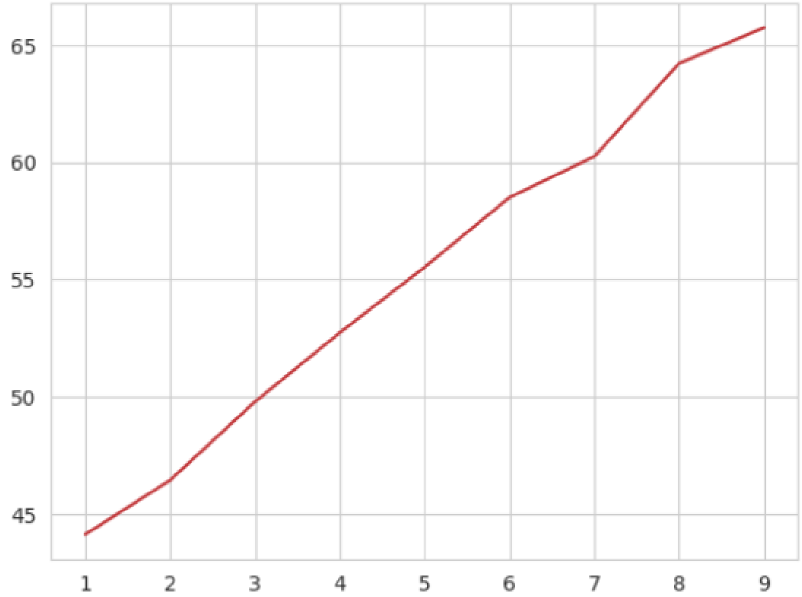


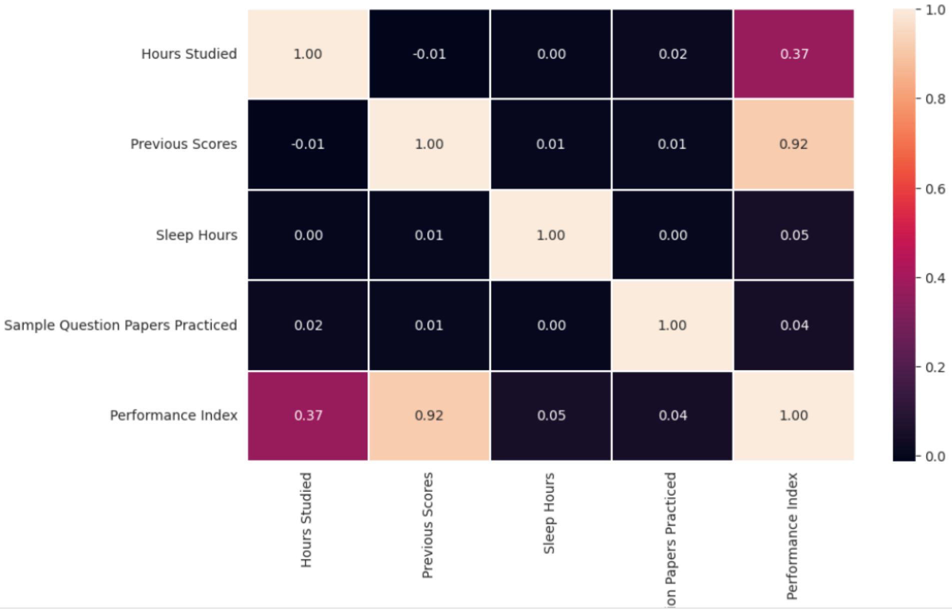
1059
1051
1035
1028
1026
987
978
955
951
930
1000
800
600
400
200
0
count
6
9
3
5
8
7
1
4
0
2
Sample Question Papers Practiced



100
80
60
40
20
FRQ
1
2
3
4
5
6
7
8
9
Hours Studied

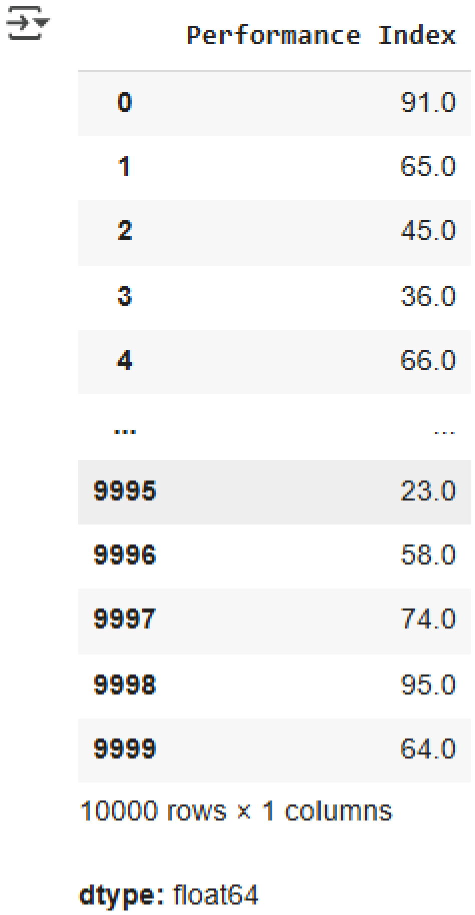

100
80
60
40
20
FRQ
Yes
No
Extracurricular Activities


65
60
55
50
45
1
2
3
4
5
6
7
8
9



	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
292	3	52	1	5	7	37.0
1533	3	83	1	7	5	62.0

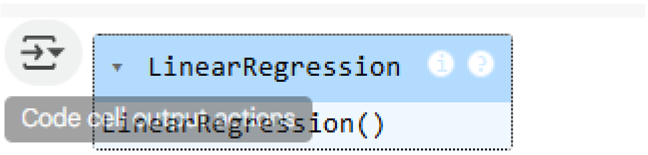
	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced
7311	2	98	1	8	5
4177	5	88	0	7	1
7926	9	43	1	8	6



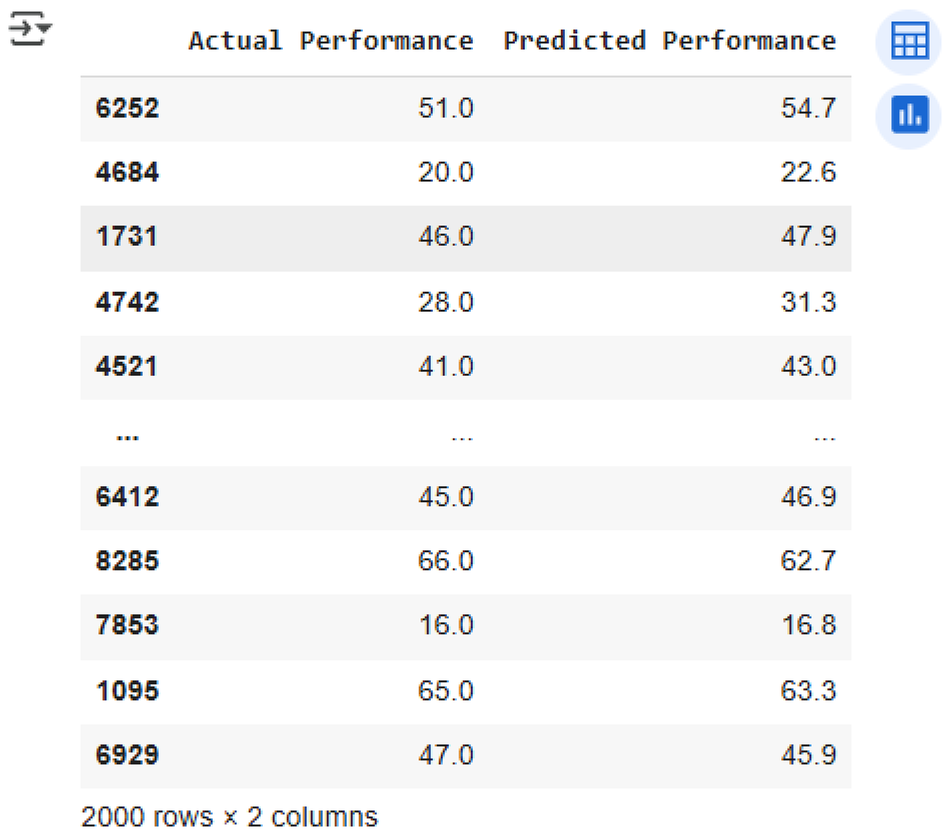
```
      Performance Index
   0               91.0
   1               65.0
   2               45.0
   3               36.0
   4               66.0
 ...                ...
9995               23.0
9996               58.0
9997               74.0
9998               95.0
9999               64.0

10000 rows × 1 columns

dtype: float64
```

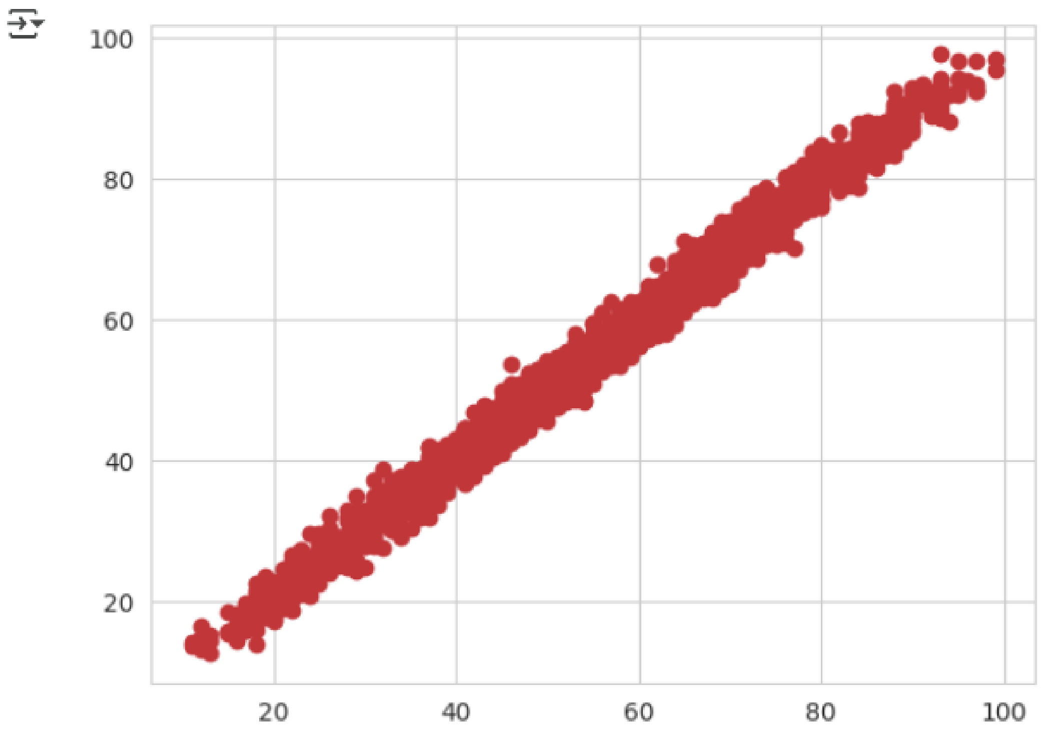


```
▾ LinearRegression
LinearRegression()
```



	Actual Performance	Predicted Performance
6252	51.0	54.7
4684	20.0	22.6
1731	46.0	47.9
4742	28.0	31.3
4521	41.0	43.0
...	...	...
6412	45.0	46.9
8285	66.0	62.7
7853	16.0	16.8
1095	65.0	63.3
6929	47.0	45.9

2000 rows × 2 columns


100
80
60
40
20
20
40
60
80
100