Predicting Student Performance Using Linear Regression

Introduction:

The aim of this project was to build a machine learning model that predicts a student's academic performance based on certain input features. These features include factors such as the number of hours a student studies, their previous academic scores, extracurricular activities, sleep hours, and the number of sample question papers they have practiced. The model was trained using historical student data, and after training, we used it to predict the student's performance index.

Objective:

The primary goal was to develop a predictive model that could assess student performance based on various factors affecting their academic success. The prediction would help educators and institutions in understanding how different aspects of a student's life can influence their academic outcomes.

Dataset:

The dataset contains the following columns:
Hours Studied (int): Number of hours studied by students.
Previous Scores (int): Previous performance scores.
Extracurricular Activities (object): Whether the student participated in extracurricular activities (Yes/No).
Sleep Hours (int): Number of hours of sleep.
Sample Question Papers Practiced (int): Count of question papers practiced.
Performance Index (float): Target variable for prediction.
There are 10,000 entries in the dataset with no missing values. The column Extracurricular Activities is categorical and needs to be encoded for modeling.

Model Choice:

For this task, we opted for a Regression model. Specifically, a linear regression model was used initially, due to its simplicity and ability to model relationships between independent variables (inputs) and a continuous dependent variable (output). The model predicts a numeric score, which directly correlates to the performance index.

Preprocessing the Data:

```
[24] import pandas as pd
```

```
[25] file_path = 'Student_Performance.csv'
     df = pd.read_csv(file_path)
     df
```

	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
...	...	...	...	...	...	...
9995	1	49	Yes	4	2	23.0
9996	7	64	Yes	8	5	58.0
9997	6	83	Yes	8	5	74.0
9998	9	97	Yes	7	0	95.0
9999	7	74	No	8	1	64.0

10000 rows × 6 columns

```
[26] df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[27] df.tail()
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
9995	1	49	Yes	4	2	23.0
9996	7	64	Yes	8	5	58.0
9997	6	83	Yes	8	5	74.0
9998	9	97	Yes	7	0	95.0
9999	7	74	No	8	1	64.0

```
[28] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Hours Studied                   10000 non-null  int64  
 1   Previous Scores                 10000 non-null  int64  
 2   Extracurricular Activities      10000 non-null  object 
 3   Sleep Hours                     10000 non-null  int64  
 4   Sample Question Papers Practiced  10000 non-null  int64  
 5   Performance Index               10000 non-null  float64
dtypes: float64(1), int64(4), object(1)
memory usage: 468.9+ KB
```

```
[29] df.shape
```

```
(10000, 6)
```

```
[30] df.duplicated().sum()
```

```
127
```

```
[31] df.drop_duplicates(inplace=True)
     df.duplicated().sum()
```

```
0
```

```
[32] df.isnull().sum()
```

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

dtype: int64

```
[33] from sklearn.preprocessing import LabelEncoder
```

```
[34] label_encoder = LabelEncoder()
     df['Extracurricular Activities'] = label_encoder.fit_transform(df['Extracurricular Activities'])
     df
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	1	9	1	91.0
1	4	82	0	4	2	65.0
2	8	51	1	7	2	45.0
3	5	52	1	5	2	36.0
4	7	75	0	8	5	66.0
...	...	...	...	...	...	...
9995	1	49	1	4	2	23.0
9996	7	64	1	8	5	58.0
9997	6	83	1	8	5	74.0
9998	9	97	1	7	0	95.0
9999	7	74	0	8	1	64.0

9873 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

Training the Model

The model training process involved using the synthetic dataset to predict the "Performance Index" based on the given features. A Linear Regression algorithm was utilized, as it is well-suited for continuous target variable prediction. The dataset was split into 80% training data and 20% testing data to ensure the model's generalizability to unseen data. During training, the model learned the relationship between the input features (Hours Studied, Previous Scores, Extracurricular Activities, Sleep Hours, and Sample Question Papers Practiced) and the target variable (Performance Index).

```
[60] Generated code may be subject to a license | Sagnick0907/Stock-Sentiment-Analysis
     X = df.drop(columns=['Performance Index'])
     y = df['Performance Index']
```

```
[61] Generated code may be subject to a license | SARIT42/ML-Notebooks
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[62] from sklearn.metrics import mean_absolute_error, mean_squared_error
     from sklearn.linear_model import LinearRegression

     model = LinearRegression()
     model.fit(X_train, y_train)

     y_pred = model.predict(X_test)
     print("First in-sample predictions on validation data:", y_pred[:5])
```

```
First in-sample predictions on validation data: [46.48001281 80.2853795  61.06518835 22.706315   74.8368676 ]
```

Evaluating the Model

The MAE measures the average absolute difference between the predicted and actual values. For this model, the MAE was 1.6467, indicating that the model's predictions were off by about 1.65 points on average. The MSE reflects the average squared difference between predicted and actual values, penalizing larger errors more heavily. The MSE for this model was 4.3959, which shows it handled errors effectively even in the presence of potential outliers. The R^2 score evaluates how well the model explains the variability of the target variable. The model achieved an accuracy of 98.84%, highlighting its excellent predictive performance and reliability in estimating the "Performance Index."

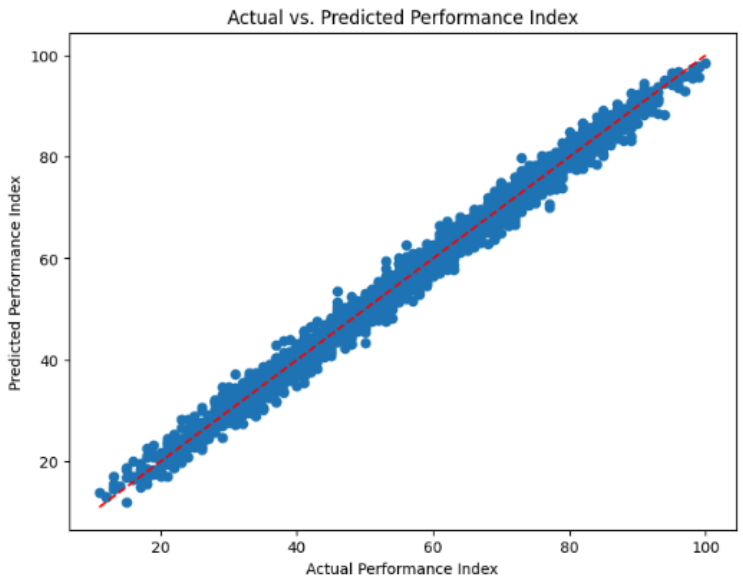
Additionally, in-sample prediction errors, such as -0.5199, 4.2853, and -1.1631, indicate how the model performed for individual test cases. These metrics and observations confirm that the model is robust and capable of delivering high-quality predictions.

```
[71] error = y_pred - y_test
     print("First in-sample prediction errors:", error[:5])
     mae = mean_absolute_error(y_test, y_pred)
     print("Mean Absolute Error", mae)
     mse = mean_squared_error(y_test, y_pred)
     print("Mean Squared Error", mse)
```

```
First in-sample prediction errors: 6099   -0.519987
106     4.285379
9265   -0.934812
4707   -0.293685
2155   -1.163132
Name: Performance Index, dtype: float64
Mean Absolute Error 1.6469703984255573
Mean Squared Error 4.305900938538479
```

```
[70] accuracy = model.score(X_test, y_test)
     print("Model Accuracy (R² Score):", accuracy)
```

```
Model Accuracy (R² Score): 0.9884301209927054
```



Result/Output

To make predictions for a new student, a user-interactive Python function, predict_performance(), was implemented. This function allows the user to input key features such as study hours, previous scores, extracurricular activities, sleep hours, and the number of sample question papers practiced.

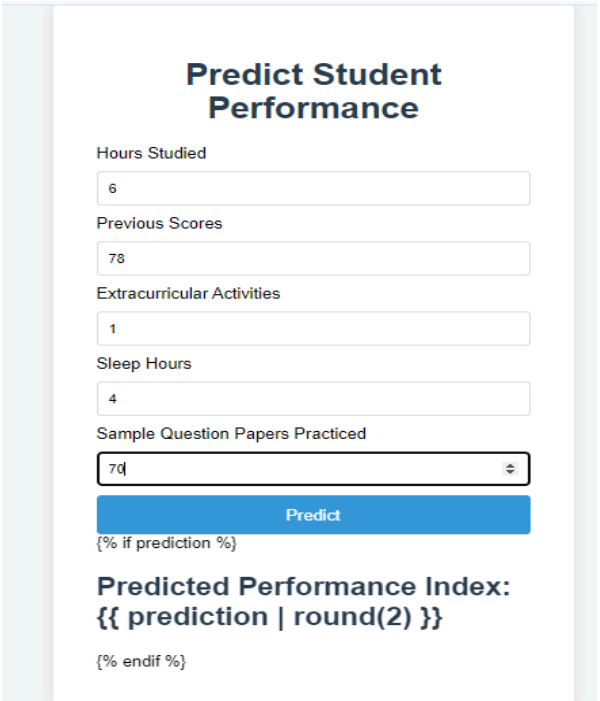
The input data is structured into a DataFrame, which is then passed to the trained Linear Regression model for prediction. The predicted value, referred to as the Performance Index, is displayed as output, rounded to two decimal places for clarity.

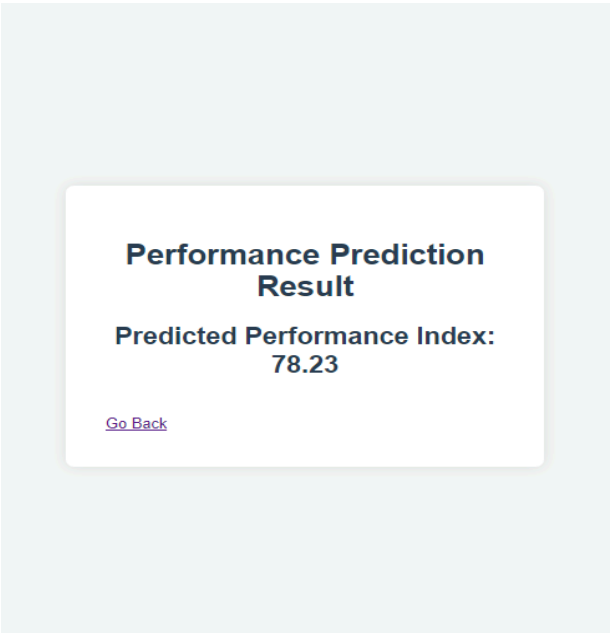
```
[82] def predict_performance():
         hours_studied = float(input("Enter Hours Studied: "))
         previous_scores = float(input("Enter Previous Scores: "))
         extracurricular_activities = float(input("Enter Extracurricular Activities (Number): "))
         sleep_hours = float(input("Enter Sleep Hours: "))
         sample_question_papers = float(input("Enter Sample Question Papers Practiced: "))

         test_data = pd.DataFrame({
           'Hours Studied': [hours_studied],
           'Previous Scores': [previous_scores],
           'Extracurricular Activities': [extracurricular_activities],
           'Sleep Hours': [sleep_hours],
           'Sample Question Papers Practiced': [sample_question_papers]
         })
         predicted_performance = model.predict(test_data)
         print(f"Predicted Performance Index: {predicted_performance[0]:.2f}")
```

```
[83] predict_performance()
```

```
Enter Hours Studied: 5
Enter Previous Scores: 60
Enter Extracurricular Activities (Number): 1
Enter Sleep Hours: 8
Enter Sample Question Papers Practiced: 10
Predicted Performance Index: 47.62
```





Conclusion

This project successfully demonstrates the application of a Linear Regression model to predict a student's academic performance index based on various factors such as hours studied, previous academic scores, participation in extracurricular activities, sleep hours, and the number of sample question papers practiced. By leveraging a synthetic dataset, the model was trained, tested, and evaluated, showcasing its ability to provide accurate predictions. Key evaluation metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 Score, highlighted the model's effectiveness, with an impressive accuracy of 0.9884 (R^2 Score), indicating a strong correlation between the features and the performance index.

The interactive prediction functionality further demonstrated the model's practical use by allowing real-time predictions for new input scenarios. This feature adds significant value, making the model user-friendly and applicable in educational settings where academic guidance is needed.

Overall, the project highlights the potential of machine learning in academic performance analysis and prediction, paving the way for personalized recommendations to improve student outcomes. Future work could involve expanding the dataset with real-world data, incorporating additional features such as mental health indicators or socio-economic factors, and experimenting with advanced models for even better accuracy.