Predicting Student Performance with Linear Regression

DS & ML Bootcamp by Ai DataYard

Linear Regression

- **Linear Regression** is a simple machine learning method used to predict a value based on the relationship between variables.
- It draws a straight line through data points to show the trend.
- For example, it can predict a student’s score based on how many hours they studied.
- If there's one input, it's called **Simple Linear Regression**, and if there are many inputs, it's **Multiple Linear Regression**.

Introduction

- This project focuses on predicting students' academic performance using **Linear Regression,** a basic yet powerful machine learning technique.

- The aim is to identify the relationship between study-related factors (Hours Studied, Previous Scores, Extracurricular Activities, Sleep Hours, Sample Question Papers Practiced) and students' actual performance, then use that pattern to make predictions.

- By applying linear regression, we can estimate how well a student might perform based on given inputs.

Introduction

- To understand and apply Simple Linear Regression and Multiple Linear Regression techniques.

- To use a real-world dataset (from Kaggle) containing student performance data.
- https://www.kaggle.com/datasets/nikhil7280/student-performance-multiple-linear-regression

- To train a model that can predict the performance of a student based on given features.

- To evaluate model accuracy using metrics like R-squared and compare actual vs predicted performance.

Predicting Student Performance with Linear Regression

- **Importing the necessary library**

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

- **Read the data**

```
#read the data

df = pd.read_csv('/content/Student_Performance.csv')
```

Predicting Student Performance with Linear Regression

- **Preview of the data:**

```
[ ] 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

DataYard
Professional eLearning Partner

Predicting Student Performance with Linear Regression

➢ **See column data type and some info**

```
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
```

Predicting Student Performance with Linear Regression

- **See if there any null values in the dataset**

```
df.isna().sum()
```

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

dtype: int64

There is no null values in this dataset

Predicting Student Performance with Linear Regression

- **See the dimensions of the dataset**

```
[ ]  df.shape

     (10000, 6)
```

Predicting Student Performance with Linear Regression

- See if there any duplicates values in the dataset
- No duplicates found in this dataset

```
df.duplicated()
```

	0
0	False
1	False
2	False
3	False
4	False
...	...
9995	False
9996	False
9997	False
9998	False
9999	False

10000 rows × 1 columns

dtype: bool

Predicting Student Performance with Linear Regression

➢ **See quick info of numeric values**

```
df.describe()
```

	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

Data Preprocessing

- **Import the necessary libraries**

```
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 label encoder**

```
[ ]  encoder = LabelEncoder()

     df["Extracurricular Activities"] =  encoder.fit_transform(df["Extracurricular Activities"])
```

This convert the categorical data in the numeric. “Extracurricular Activities” is the categorical column and it has two values ‘Yes’ and ‘No’ and after label encoding it convert the ‘Yes’ to 1 and ‘No’ to 0

Predicting Student Performance with Linear Regression

➢ **See the sample data after label encoding**

```
df.sample(4)
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
8017	2	94	0	7	3	73.0
6318	7	69	0	9	3	57.0
9555	5	86	1	6	7	72.0
3768	6	83	1	4	6	69.0

Predicting Student Performance with Linear Regression

➢ **Splitting data into Independent and Dependent Variable for the Linear Regression**

```
[ ]  X = df.drop(columns = "Performance Index")
     y = df["Performance Index"]
```

In this, I assign the ‘Performance Index’ to y (dependent variable) and all the remaining columns in the dataset to X(independent variable)

Predicting Student Performance with Linear Regression

➢ **See the sample of X variable**

```
X.sample(4)
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced
1707	2	81	0	8	4
8314	4	44	0	7	5
9746	2	49	0	6	6
712	7	43	0	5	3

Predicting Student Performance with Linear Regression

- **See the sample of y variable**

```
y
```

	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

Predicting Student Performance with Linear Regression

➢ See the sample of y variable

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

X_train: This will hold the independent variables that the model will use for training.
X_test: This will hold the features that the model will use for testing its performance after training.
y_train: This will hold the target variable (dependent variable) that the model will try to predict during training.
y_test: This will hold the actual values of the target variable for the test set, which will be used to evaluate the model's predictions.
train_test_split(): This is a function from the **sklearn.model_selection** module. Its purpose is to split arrays or matrices into random train and test subsets.
test_size = 0.2: This argument specifies the proportion of the dataset to be included in the test split. In this case, 20% of the data will be used for testing, and the remaining 80% will be used for training. [1]
random_state = 42: Setting a specific integer value for **random_state** ensures that you get the same split every time you run the code.

Predicting Student Performance with Linear Regression

➢ **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)
```

```
x_train shape:  (8000, 5)
y_train shape:  (8000,)
x_test shape:  (2000, 5)
y_test shape:  (2000,)
```

Predicting Student Performance with Linear Regression

- **Create an instance of Linear Regression class from the sklearn.linear_model**

```
[ ]  model = LinearRegression()
```

- **Train the LinearRegression model**

```
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

This line is where the machine learning model is trained. The .fit() method is used to train the LinearRegression model using the training data.

Predicting Student Performance with Linear Regression

- **Calculate the score of the model on the training data**

```
[ ] model.score(X_train, y_train)

    0.9886898790682355
```

The .score() method returns the coefficient of determination, also known as the R-squared value.
The R-squared value is a measure of how well the independent variables (X_train) explain the variance in the dependent variable (the Performance Index in y_train).

Predicting Student Performance with Linear Regression

- **See predicted values**

```
predict = model.predict(X_test)
predict
```

```
array([54.71185392, 22.61551294, 47.90314471, ..., 16.79341955,
       63.34327368, 45.94262301])
```

Predicting Student Performance with Linear Regression

➢ Real Values vs Predicted Values

```
pd.DataFrame({"Actual Performance" : y_test, "Predicted Performance" : predict})
```

	Actual Performance	Predicted Performance
6252	51.0	54.711854
4684	20.0	22.615513
1731	46.0	47.903145
4742	28.0	31.289767
4521	41.0	43.004570
...	...	...
6412	45.0	46.886280
8285	66.0	62.698025
7853	16.0	16.793420
1095	65.0	63.343274
6929	47.0	45.942623

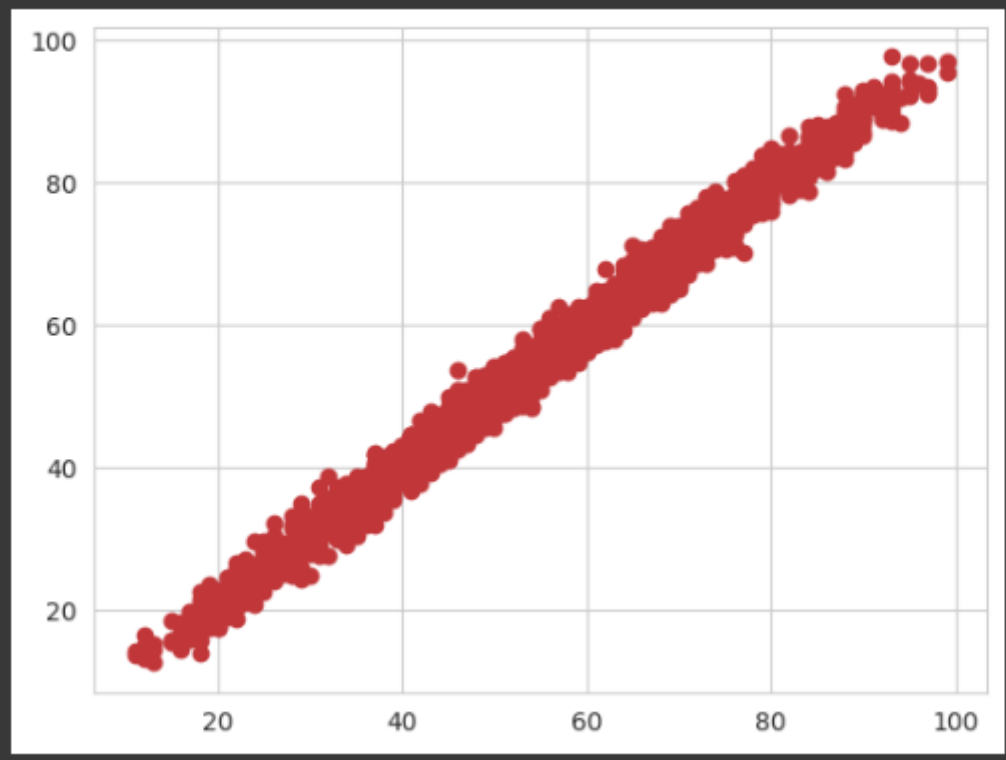
2000 rows × 2 columns

DataYard
Professional eLearning Partner

Predicting Student Performance with Linear Regression

- **Create scatter plot to see distribution**

```
plt.scatter(y_test, predict)
plt.show()
```



Predicting Student Performance with Linear Regression

➢ **See the Mean Absolute Error**

```
mean_absolute_error(y_test,predict)
```

```
1.6111213463123044
```

Mean Absolute Error (MAE) is a way to measure how accurate your predictions are. It tells you **how much your predictions differ from the actual results** — on average.

Predicting Student Performance with Linear Regression

➢ **See the coefficients values**

```
model.coef_
```

```
array([2.85248393, 1.0169882 , 0.60861668, 0.47694148, 0.19183144])
```

These numbers are your model's **coefficients** — they show how much each feature affects the prediction.
For example, the first feature has the biggest impact, while the last one has the smallest.

Predicting Student Performance with Linear Regression

➢ See the intercept value

```
model.intercept_
```

```
np.float64(-33.92194621555638)
```

In general, the **intercept** refers to a value where a function or a line crosses the axis of a graph.

For a linear equation like y=mx+b the **intercept** is represented by b. This is where the line crosses the y-axis, meaning when x=0, y=b.