INTERNSHIP PROJECT REPORT
Submitted to
INLIGHNX GLOBAL PVT. LTD.
Bangalore, India

Date of Submission
15/01/26

Prediction Model
Student Mark Prediction
Model: Simple Linear Regression

Prepared by
Nadar Deepthi Chenthil
AI & ML Intern
ITID5802

Internship Duration
15/11/25 – 15/01/26

INDEX

A snapshot of the dataset opened in Excel

	A	B
1	Hours_Studied	Marks
2	4.76	46.27
3	3	34.3
4	2.08	33.63
5	4.04	47.81
6	9.49	66.26
7	3.91	38.37
8	5.67	56.24
9	7.33	60.47
10	4.27	39.02

The dataset we have chosen contains students' information with two columns the number of Hours Studied and the corresponding Marks obtained. We intend to create a Simple Linear Regression model to analyze the relationship between study hours and marks, and to predict the marks that a student is likely to score based on the number of hours studied.

We have used Jupyter Notebook to create our model. Here are the steps as follows:

Step 1: Import the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from pandas.core.common import random_state
```

- NumPy is used for numerical operations.
- Pandas helps in loading, reading, and analyzing the dataset.
- Matplotlib is used to visualize data using graphs.
- Scikit-learn is used to build and evaluate the machine learning model.
- train_test_split is used to divide the dataset into training and testing sets.
- LinearRegression is used to create the simple linear regression model.
- r2_score is used to evaluate the accuracy of the model.

These libraries are essential for data handling, model building, visualization, and performance evaluation.

Step 2: Loading the data

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

	Hours_Studied	Marks
0	4.76	46.27
1	3.00	34.30
2	2.08	33.63
3	4.04	47.81
4	9.49	66.26

- The dataset is loaded from a CSV file using Pandas.
- head() displays the first few rows of the dataset to understand its structure.
- The dataset contains two columns: Hours Studied and Marks.

Step 3: Data analysis

```
df.describe()
```

	Hours_Studied	Marks
count	100.000000	100.000000
mean	5.376300	46.360200
std	2.593013	14.284805
min	1.050000	20.190000
25%	3.175000	33.675000
50%	5.565000	47.485000
75%	7.255000	57.377500
max	9.870000	73.260000

This function provides statistical details such as:

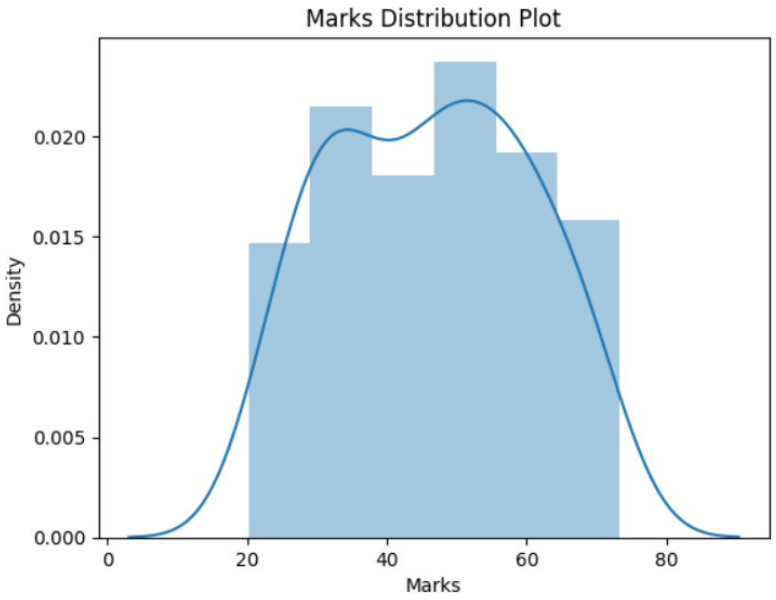
- Minimum and maximum values
- Mean and median
- Standard deviation

It helps us understand the overall distribution of marks

we can see that Marks range from 20.19 to 73.26, with a median of about 47.49

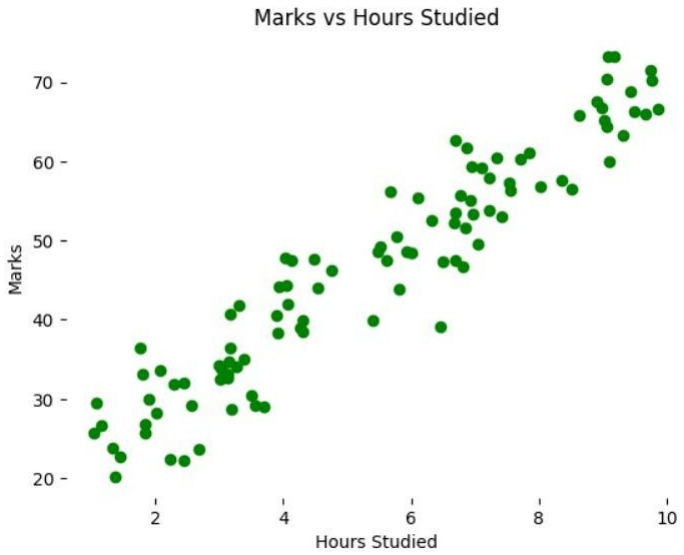
Data distribution

```
plt.title('Marks Distribution Plot')
sns.distplot(df['Marks'])
plt.show()
```



Relationship between Hours studied and Mark

```
plt.scatter(df['Hours_Studied'], df['Marks'], color='green')
plt.title('Marks vs Hours Studied')
plt.xlabel('Hours Studied')
plt.ylabel('Marks')
plt.box(False)
plt.show()
```



A scatter plot is created to visualize the relationship between hours studied and marks.

The graph shows a positive trend, indicating that marks increase as study hours increase.

Step 4: Split the dataset into dependent/independent variables

Splitting variables

***Hours_Studied (X)* is the independent variable**
***Mark (y)* is dependent variable**

```
X = df.iloc[:, :1]  # Hours_Studied
y = df.iloc[:, 1:]  # Marks
```

The model learns how marks depend on study hours

Split data into Train/Test sets

```
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

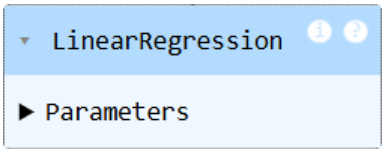
The dataset is split into:

- 80% training data
- 20% testing data

Training data is used to build the model.
Testing data is used to evaluate the model's performance.

Step 5: Train the regression model

```
# Step 5: Train model
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```



- A Simple Linear Regression model is created.
- The model learns the relationship between hours studied and marks.
- The best-fit line is calculated during training.

Step 6: Predict the result

```
# Step 6: Predictions
y_pred_train = regressor.predict(X_train)
y_pred_test = regressor.predict(X_test)
```

The trained model predicts marks for both:

- Training data
- Testing data

These predictions are later compared with actual values.

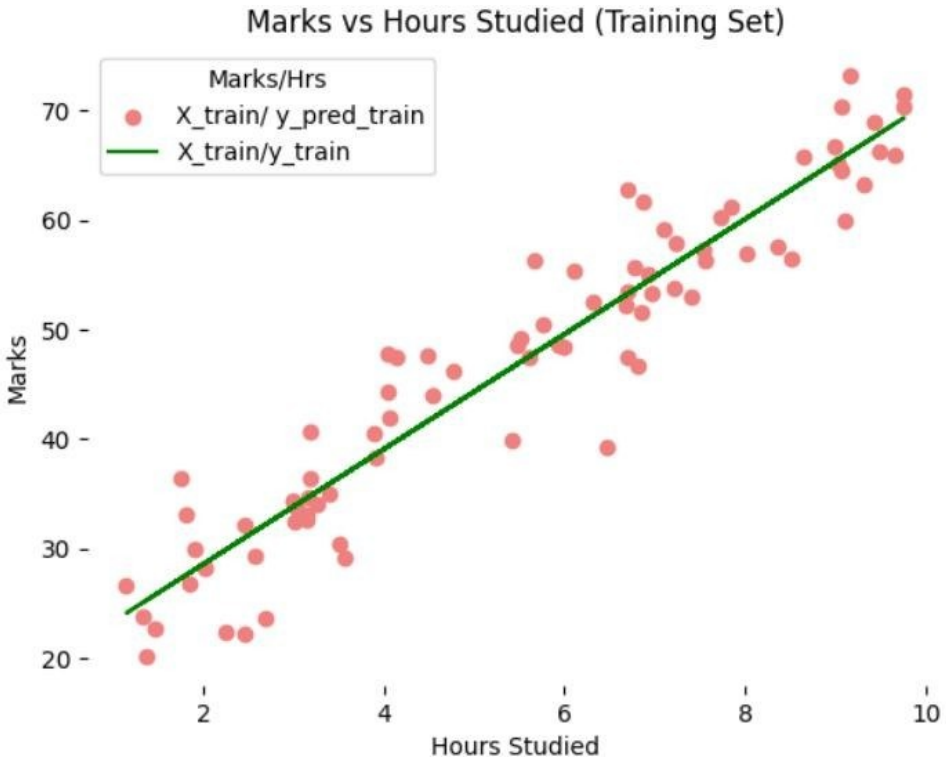
Step 7: Plot the training and test results

Next test predicted results by plotting graphs

Plot training set data vs predictions

First we plot the result of training sets (X_train, y_train) with X_train and predicted value of y_train (regressor.predict(X_train))

```
plt.scatter(X_train, y_train, color='lightcoral')
plt.plot(X_train, y_pred_train, color='green')
plt.title('Marks vs Hours Studied (Training Set)')
plt.xlabel('Hours Studied')
plt.ylabel('Marks')
plt.legend(['X_train/ y_pred_train', 'X_train/y_train'], title='Marks/Hrs', loc='best', facecolor='white')
plt.box(False)
plt.show()
```

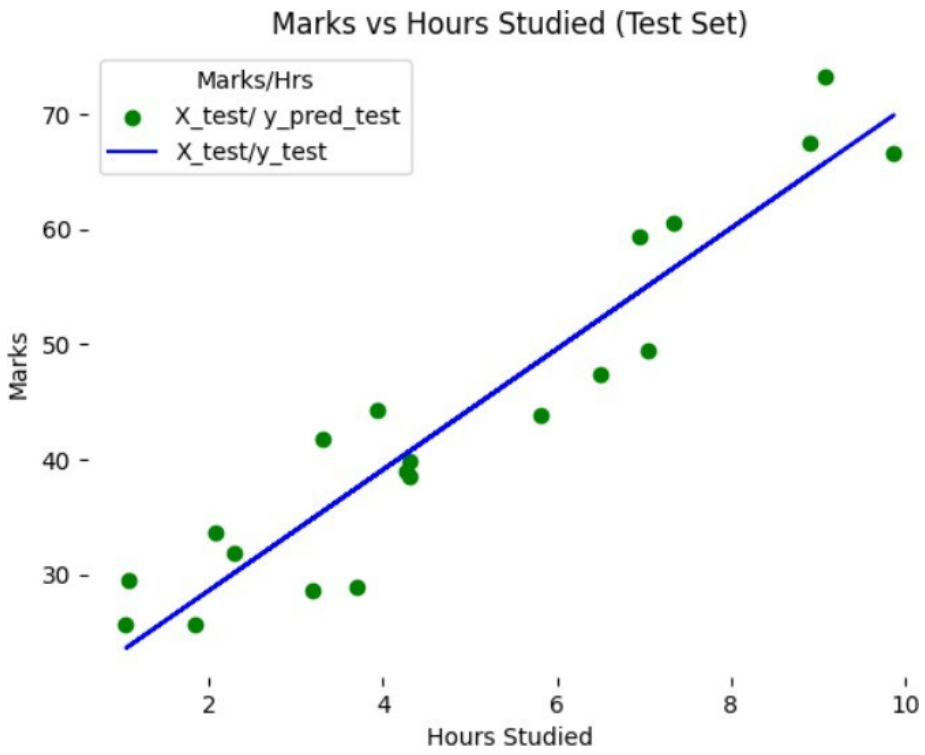


- The scatter plot shows actual training data.
- The straight line represents the regression line.
- It indicates how well the model fits the training data.

Plot test set data vs predictions

Secondly, we plot the result of test sets (X_test, y_test) with X_test and predicted value of y_test (regressor.predict(X_test))

```
plt.scatter(X_test, y_test, color='green')
plt.plot(X_test, y_pred_test, color='blue')
plt.title('Marks vs Hours Studied (Test Set)')
plt.xlabel('Hours Studied')
plt.ylabel('Marks')
plt.legend(['X_test/ y_pred_test', 'X_test/y_test'], title='Marks/Hrs', loc='best', facecolor='white')
plt.box(False)
plt.show()
```



We can see, in both plots, the regressor line covers train and test data, that is, the regressor line fits well across both train and test data.

Regressor coefficient and intercept

```
# Regressor coefficient and intercept
print(f'Coefficient: {regressor.coef_}')
print(f'Intercept: {regressor.intercept_}')
```

```
Coefficient: [[5.24917023]]
Intercept: [18.08838389]
```

- Coefficient (Slope) shows how much marks increase per additional hour studied.
- Intercept is the predicted mark when study hours are zero.

Testing Accuracy of the Model

```
train_r2 = r2_score(y_train, y_pred_train)
```

```
print("The accuracy of the model is {}%".format(round(train_r2, 2) *100))
```

```
The accuracy of the model is 90.0%
```

The value of R Square is 90.0, which indicates that 94.0% of the data fit the regression model.

Predict the Mark for a new Hours of Studied data

```
new_hours = np.array([[7.5]])
predicted_marks = regressor.predict(new_hours)[0][0]
print(predicted_marks)
```

```
57.457160642939925
```

- The model predicts marks for a student who studies 7 hours.
- This shows the practical use of the regression model.

Conclusion

In this project, we successfully implemented a Simple Linear Regression model to analyze the relationship between hours studied and marks obtained by students. The dataset was explored, visualized, and split into training and testing sets to ensure accurate model evaluation.

The regression model demonstrated a strong positive relationship between study hours and marks, with a high R-Square value indicating good accuracy. The visualization of both training and testing results confirmed that the regression line fits the data well.

Finally, the model was used to predict marks for new study hours, proving its effectiveness in real-world scenarios. This study shows that Simple Linear Regression is a powerful and easy-to-use technique for prediction and analysis in data science applications.