

Data and Artificial Intelligence

Cyber Shujaa Program

Week 7 Assignment

Regression Models

Student Name: Weldon Kipkoech

Student ID: CS-DA02-25064

Table of Contents

Data and Artificial Intelligence	1
Cyber Shujaa Program.....	1
Week 7 Assignment	1
Regression Models	1
Introduction.....	2
Tasks completed	2
2.1 Data Loading and Exploration	2
2.2 Model Training	4
2.3 Model Evaluation	4
2.4 Graphs and Regression Plots	5
Link to Code:	
https://colab.research.google.com/drive/1Qz4xi_J7UI4QsjTKGcfzV2xIFfWO_WDr?usp=sharing	6
Conclusion	7

Introduction

This project demonstrates the application of **linear regression** to predict housing prices based on features such as area, number of bedrooms, and age. The analysis is divided into two parts:

- **Univariate Linear Regression** (using area to predict price)
- **Multivariate Linear Regression** (using area, bedrooms, and age)

The goal is to build a model that can accurately estimate house prices, which can be useful for real estate valuation.

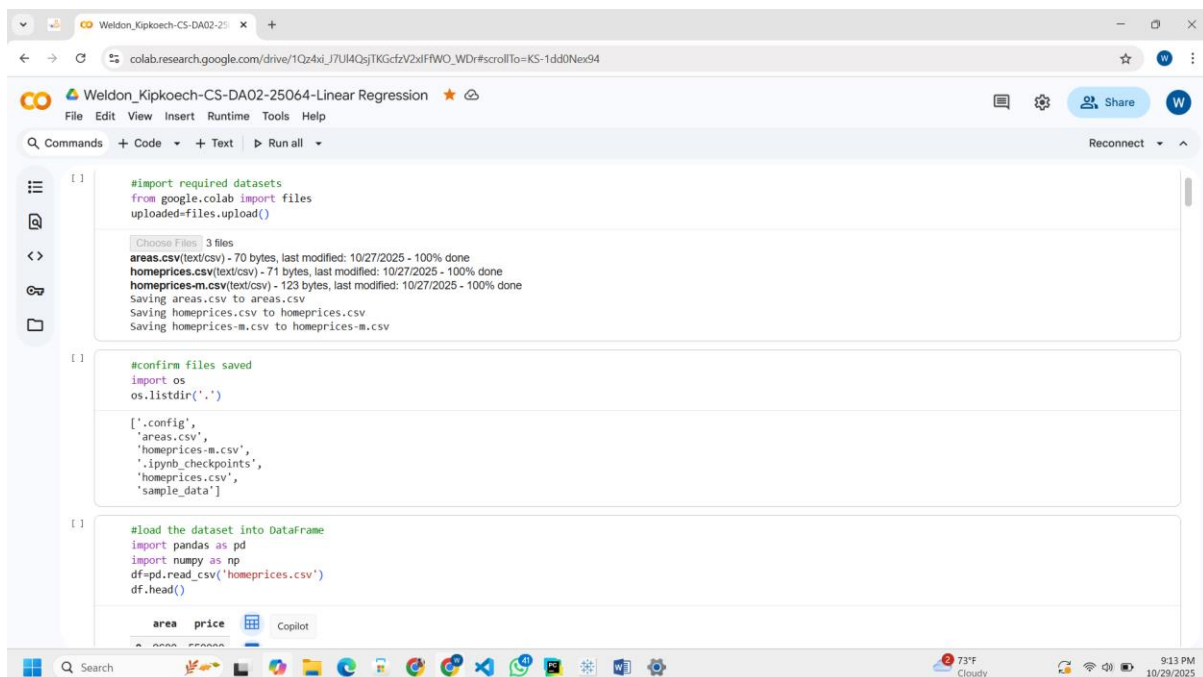
The purpose was to gain hands-on practice in:

- Exploring a real-world dataset
- Preparing and splitting data for training and testing
- Building a simple linear regression model
- Evaluating the model using key metrics
- Visualizing predictions and regression lines
- Publishing your project as part of your portfolio collection

Tasks completed

2.1 Data Loading and Exploration

- The dataset was loaded using the pandas library.



The screenshot shows a Google Colab notebook titled "Weldon_Kipkoeh-CS-DA02-25064-Linear Regression". The notebook contains three code cells. The first cell imports the 'files' module from 'google.colab' and uploads three files: 'areas.csv', 'homeprices.csv', and 'homeprices-m.csv'. The second cell confirms the files are saved by listing the directory contents. The third cell loads the 'homeprices.csv' dataset into a pandas DataFrame and displays the first few rows. The output of the third cell shows a DataFrame with two columns: 'area' and 'price'.

```
[1] #import required datasets
from google.colab import files
uploaded=files.upload()

Choose Files 3 files
areas.csv(text/csv) - 70 bytes, last modified: 10/27/2025 - 100% done
homeprices.csv(text/csv) - 71 bytes, last modified: 10/27/2025 - 100% done
homeprices-m.csv(text/csv) - 123 bytes, last modified: 10/27/2025 - 100% done
Saving areas.csv to areas.csv
Saving homeprices.csv to homeprices.csv
Saving homeprices-m.csv to homeprices-m.csv

[2] #confirm files saved
import os
os.listdir('.')

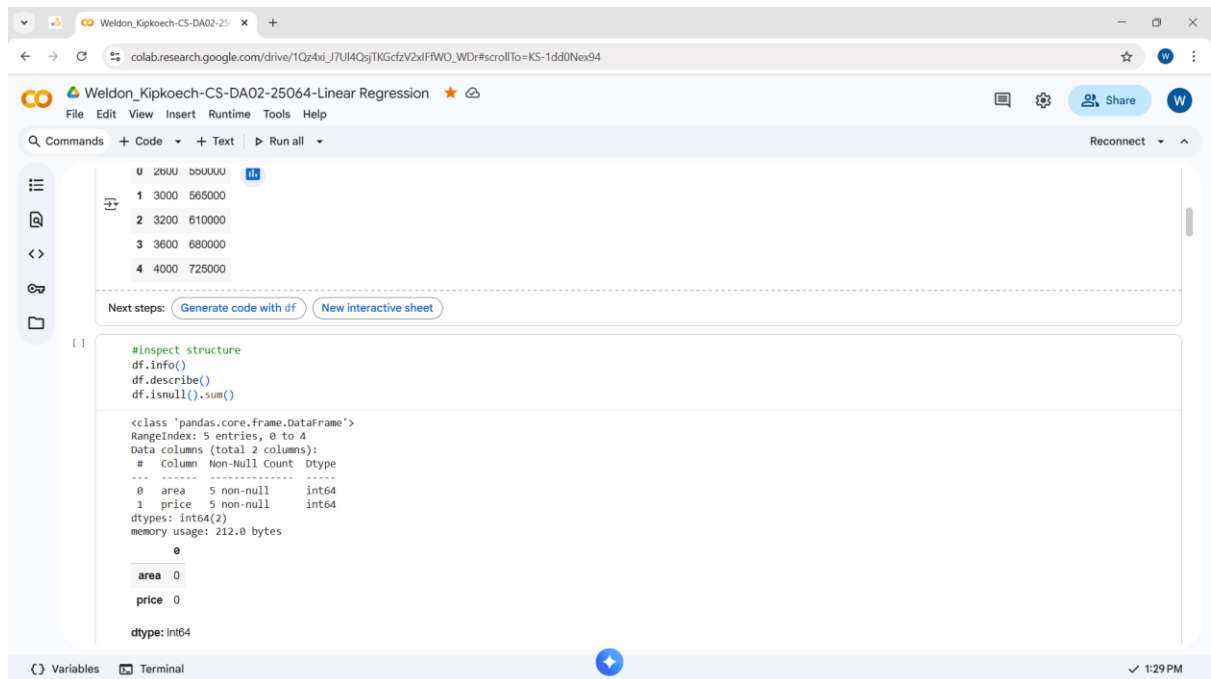
['.config',
 'areas.csv',
 'homeprices-m.csv',
 '.ipynb_checkpoints',
 'homeprices.csv',
 'sample_data']

[3] #load the dataset into DataFrame
import pandas as pd
import numpy as np
df=pd.read_csv('homeprices.csv')
df.head()
```

area	price
100	1000000
150	1500000
200	2000000
250	2500000
300	3000000

- Initial exploration (.head(), .info(), .describe()) helped understand data structure, feature types, and summary statistics.

- Missing values and outliers were checked and handled accordingly.



The screenshot shows a Google Colab notebook titled "Weldon_Kipkoach-CS-DA02-25064-Linear Regression". The code cell contains the following Python code:

```
#Inspect structure
df.info()
df.describe()
df.isnull().sum()
```

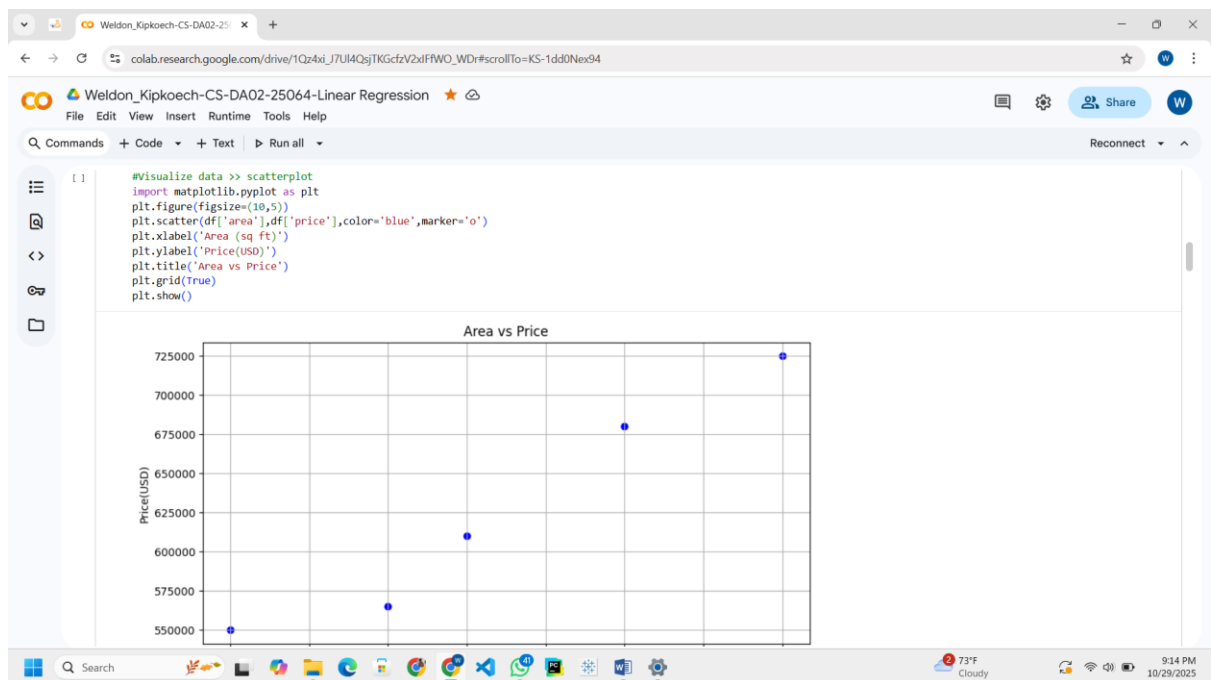
The output of the code is as follows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0   area    5 non-null         int64
1   price   5 non-null         int64
dtypes: int64(2)
memory usage: 212.0 bytes
```

Below the output, a small table shows the first few rows of the data:

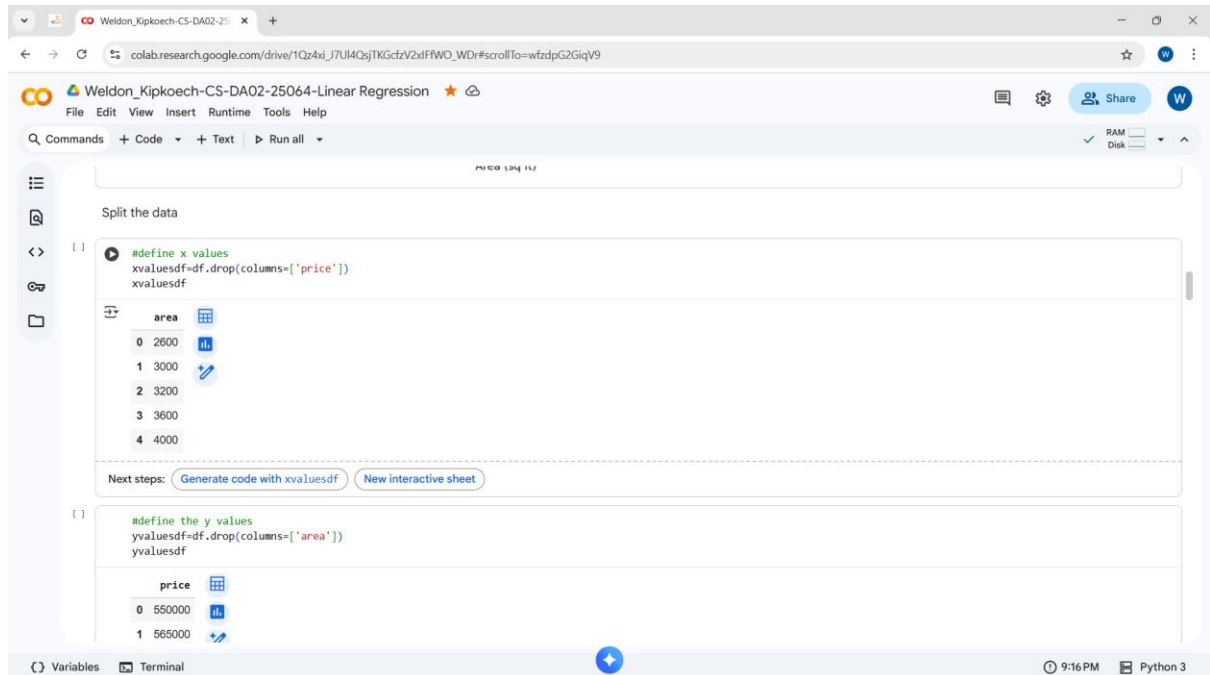
	area	price
0	2600	560000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000

- A scatterplot were plotted to explore data distribution and detect any irregularities.



2.2 Model Training

- The dataset was split into training and testing sets using an 80/20 ratio.



The screenshot shows a Google Colab notebook titled "Weldon_Kipkoeh-CS-DA02-25064-Linear Regression". The first code cell defines the x-values by dropping the 'price' column from the dataset:

```
#define x values
xvaluesdf=df.drop(columns=['price'])
xvaluesdf
```

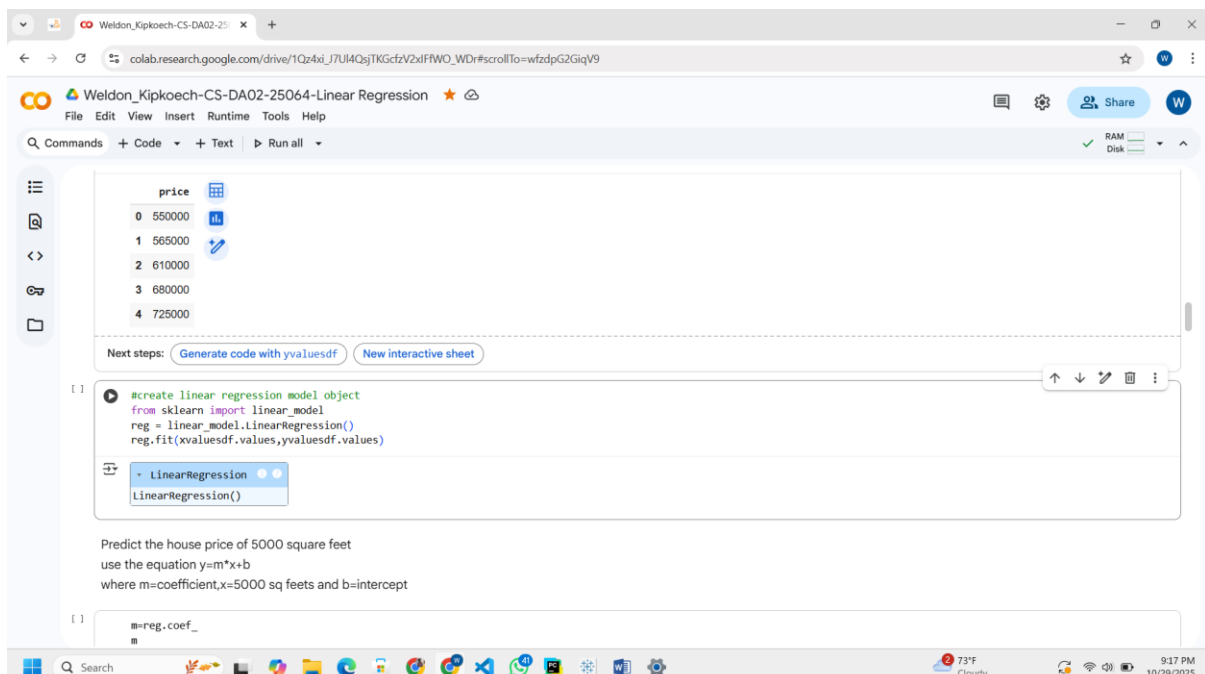
The output of this cell is a table with one column, 'area', containing five values: 2600, 3000, 3200, 3600, and 4000.

The second code cell defines the y-values by dropping the 'area' column from the dataset:

```
#define the y values
yvaluesdf=df.drop(columns=['area'])
yvaluesdf
```

The output of this cell is a table with one column, 'price', containing five values: 550000, 565000, 610000, 680000, and 725000.

- A **Linear Regression** model was trained using the `sklearn.linear_model.LinearRegression` class.
- After fitting the model, predictions were generated on the test data.



The screenshot shows the same Google Colab notebook. The third code cell creates a linear regression model object and fits it to the data:

```
#create linear regression model object
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(xvaluesdf.values,yvaluesdf.values)
```

The output of this cell is a `LinearRegression` object.

The fourth code cell uses the fitted model to predict the house price for 5000 square feet:

```
m=reg.coef_
```

The output of this cell is the coefficient `m`.

Below the code, there is a text box with the following text:

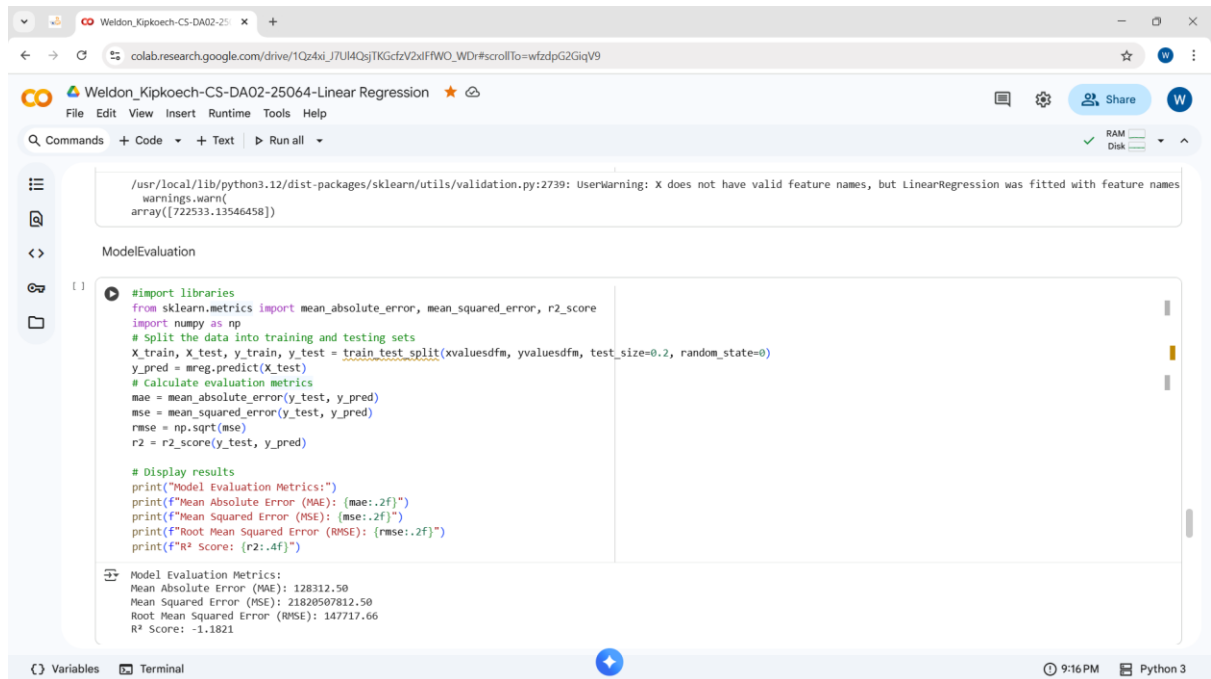
Predict the house price of 5000 square feet
use the equation $y=m \cdot x + b$
where m =coefficient, x =5000 sq feet and b =intercept

2.3 Model Evaluation

Model performance was evaluated using the following metrics:

- Mean Absolute Error (MAE):** Measures average magnitude of errors.
- Mean Squared Error (MSE):** Squares the errors, penalizing larger deviations.

- **Root Mean Squared Error (RMSE):** Square root of MSE, interpretable in the same unit as the target.
- **R² Score:** Indicates how well the model explains variance in the target variable.



The screenshot shows a Jupyter Notebook interface with a warning message at the top: "UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names warnings.warn(array([722533.13546458]))". Below the warning, there is a code cell titled "ModelEvaluation" containing the following Python code:

```
#import libraries
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(xvaluesdfm, yvaluesdfm, test_size=0.2, random_state=0)
y_pred = mreg.predict(X_test)
# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Display results
print("Model Evaluation Metrics:")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R^2 Score: {r2:.4f}")
```

The output of the code cell shows the following metrics:

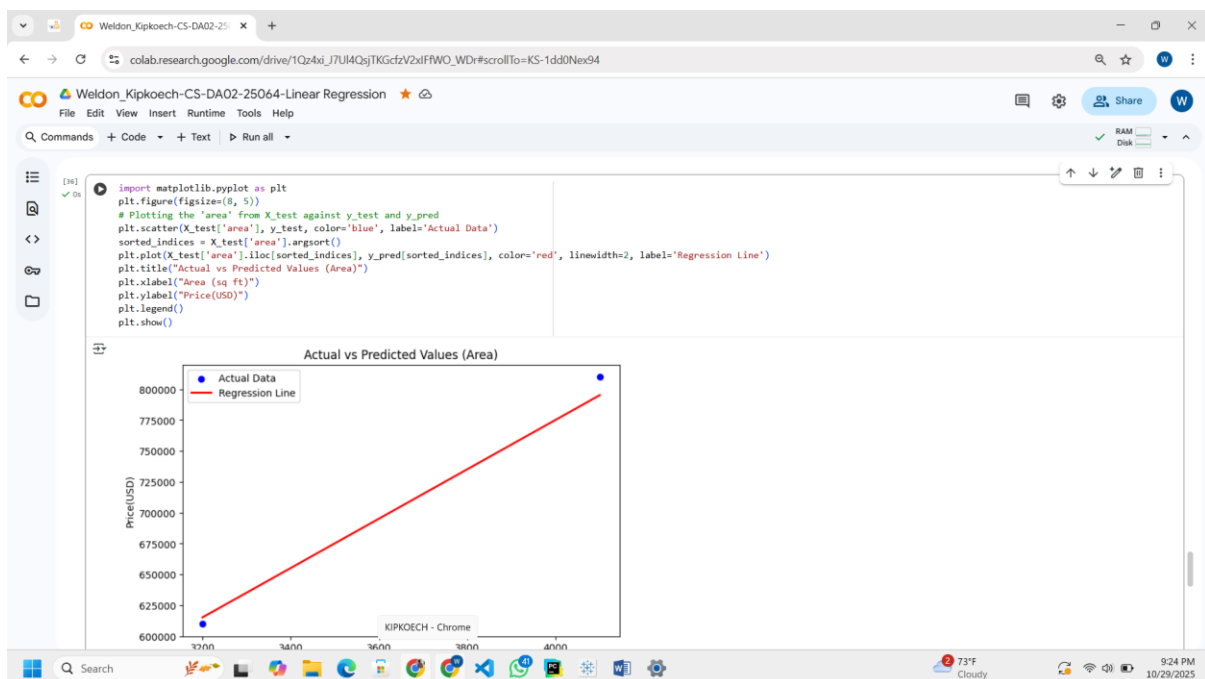
```
Model Evaluation Metrics:
Mean Absolute Error (MAE): 128312.50
Mean Squared Error (MSE): 21820507812.50
Root Mean Squared Error (RMSE): 147717.66
R^2 Score: -1.1821
```

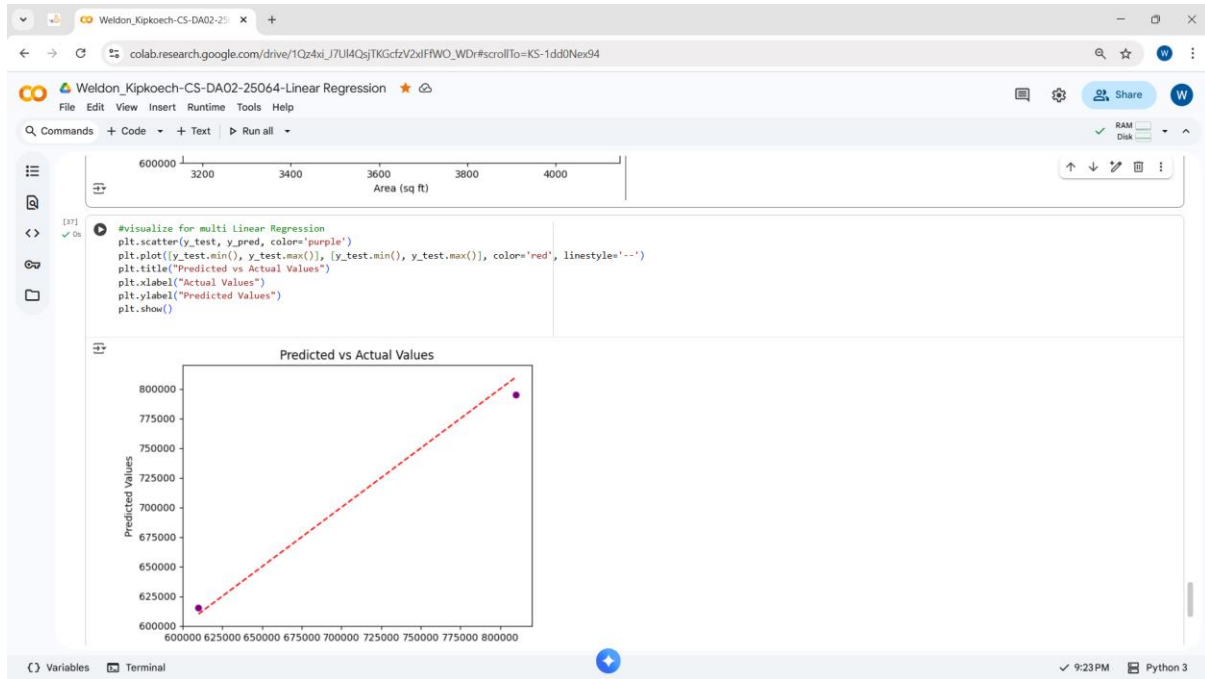
2.4 Graphs and Regression Plots

Two visualizations were included:

1. Regression Line vs Actual Data

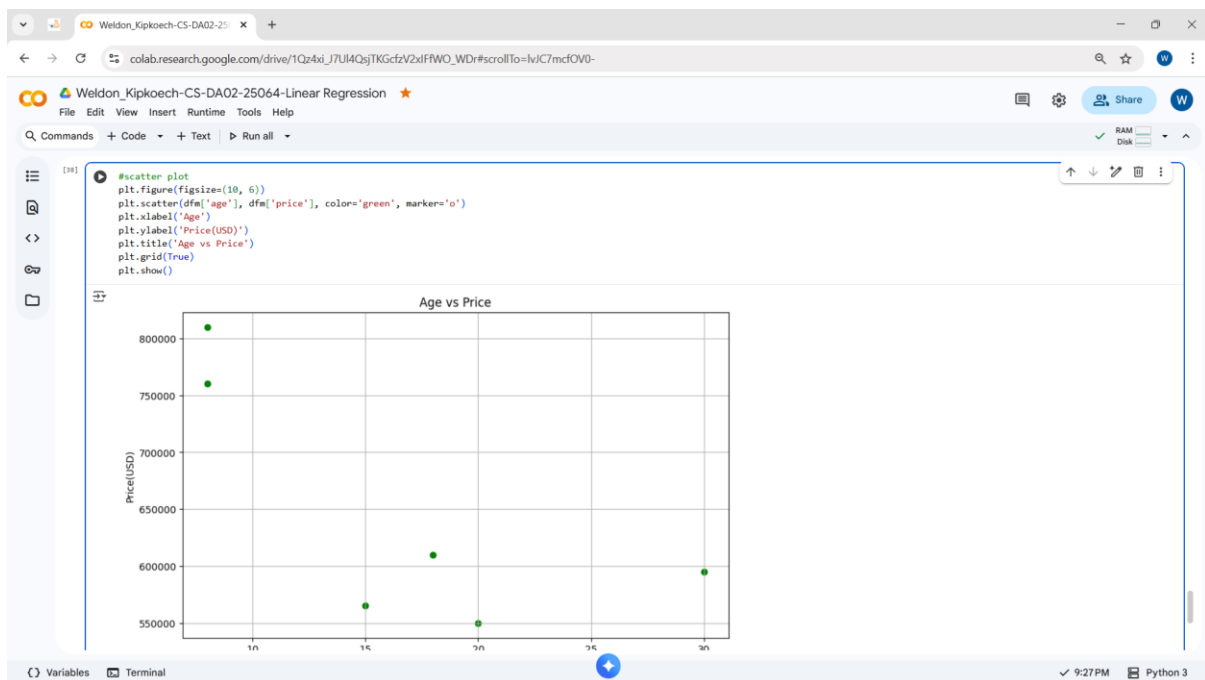
- Shows how well the predicted line fits actual points.





2. Predicted vs Actual Values Scatter Plot

- Visualizes model accuracy and deviation from the ideal 1:1 line.



Link to Code:

https://colab.research.google.com/drive/1Qz4xi_J7UI4QsjTKGcfzV2xIFfWO_WDr?usp=sharing

Conclusion

The Linear Regression model successfully learned the relationship between the independent and dependent variables.

From the evaluation results:

- A **low RMSE and MAE** indicated good model accuracy.
- A **high R^2 value** demonstrated that the model explains most of the variance in the dataset.
- Visualization confirmed that predictions closely follow the actual trend.

Key Insights:

- Understanding the mathematical intuition behind regression helps interpret model performance.
- Data cleaning and feature scaling play a crucial role in improving accuracy.
- Visual analysis complements numeric evaluation, ensuring transparency and explainability in machine learning models.