#Introduction

Linear Regression, a foundational algorithm in data science, plays a pivotal role in predicting continuous outcomes. This project provides an in-depth exploration of Linear Regression, covering its principles, applications, and implementation in Python on a real-world dataset. From understanding simple and multiple linear regression to unveiling its significance, limitations, and practical use cases, this note serves as a comprehensive resource for both beginners and practitioners. Join us on this journey through the intricacies of linear regression, offering insights into its workings and hands-on application. This article is part of the Data Science Blogathon, delivering valuable knowledge for data enthusiasts.

Linear regression is a type of machine-learning algorithm more specifically a supervised machine-learning algorithm that learns from the labelled datasets and maps the data points to the most optimized linear functions. which can be used for prediction on new datasets.

#Classification

Linear regression can be further divided into two types of the algorithm:

1. Simple Linear Regression: If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

2. Multiple Linear regression: If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

#Why Linear Regression is Important?

The interpretability of linear regression is a notable strength. The model's equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

Linear regression is not merely a predictive tool; it forms the basis for various advanced models. Techniques like regularization and support vector machines draw inspiration from linear regression, expanding its utility. Additionally, linear regression is a cornerstone in assumption testing, enabling researchers to validate key assumptions about the data.

#Import necessary libraries.

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

#Load the dataset.

```
df = pd.read_csv("../input/house-rent-dataset/Dhaka Rent.csv")
df.head()
```

```
   area   rent
0  2000  31500
1  2100  35000
2  2500  41050
3  2250  36100
4  3000  52100
```

#Dividing Dependent and Independent.

-An independent variable is a variable that represents a quantity that is being manipulated in an experiment. x is often the variable used to represent the independent variable in an equation.

-A dependent variable represents a quantity whose value depends on how the independent variable is manipulated. y is often the variable used to represent the dependent variable in an equation.

```
#Dependent rent, y
#Independent area, x
x = df.drop('rent', axis=1)

x.head()
```

```
   area
0  2000
1  2100
2  2500
3  2250
4  3000
```

```
x.shape
```

```
(60, 1)
```

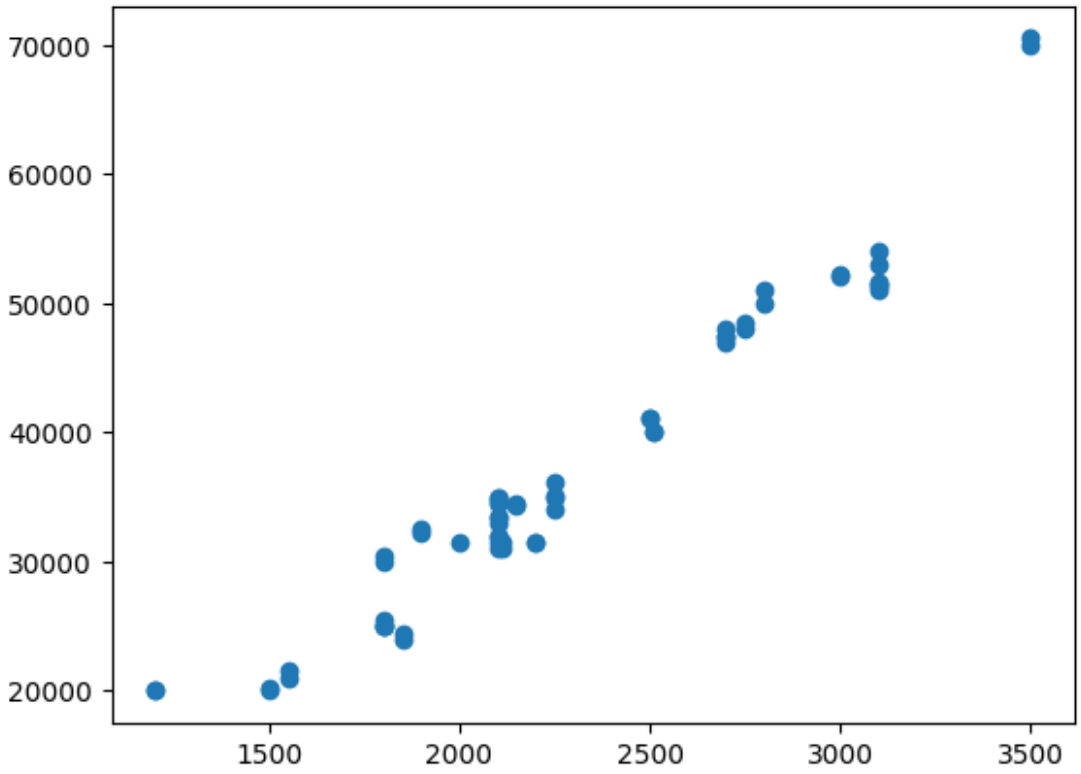
```
y = df['rent']
y.head()
```

```
0    31500
1    35000
2    41050
3    36100
4    52100
Name: rent, dtype: int64
```

#Visualization.

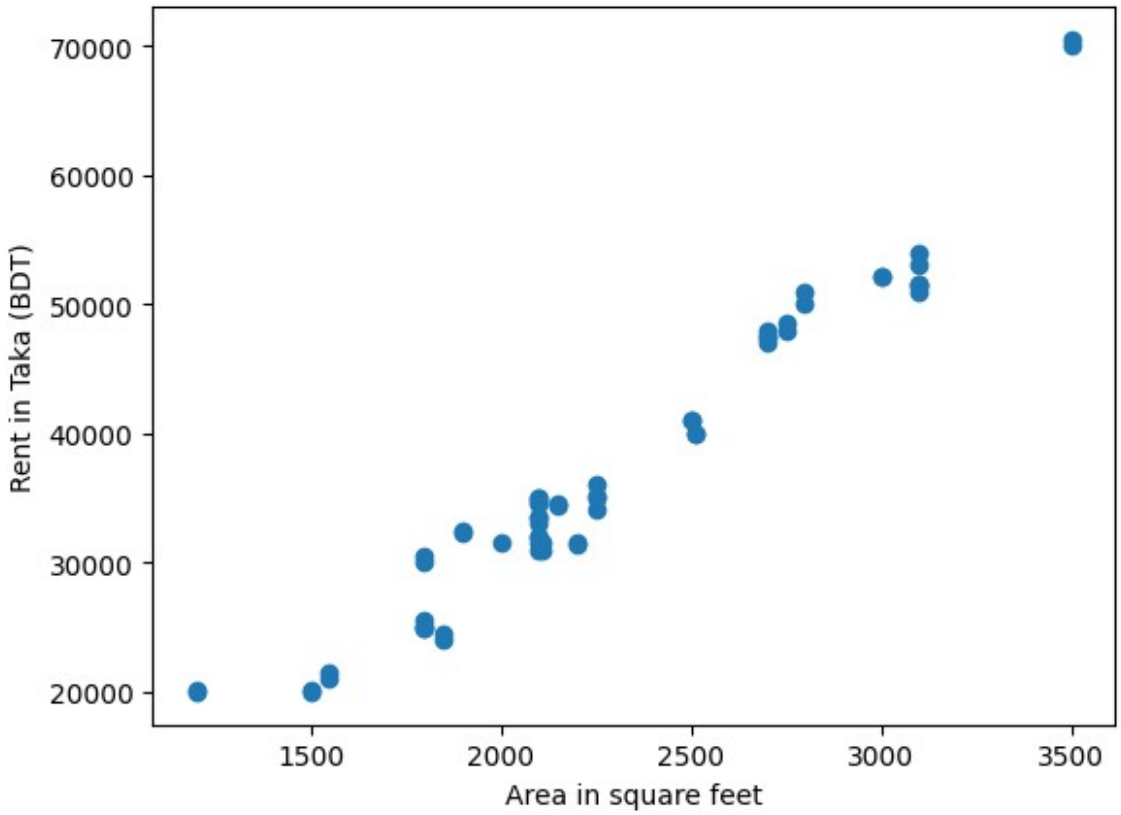
```
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x79a6f981beb0>
```




```
plt.scatter(x, y)
plt.xlabel('Area in square feet')
plt.ylabel('Rent in Taka (BDT)')

Text(0, 0.5, 'Rent in Taka (BDT)')
```



#Mean and Median.

-Mean: The "average" number; found by adding all data points and dividing by the number of data points. Example: The mean of 4,1,and 7 is (4+1+7)/3 = 12/3 = 4

-Median: The middle number; found by ordering all data points and picking out the one in the middle (or if there are two middle numbers, taking the mean of those two numbers). Example: The median of 4,1 and 7 is 4 because when the numbers are put in order (1,4,7), the number 4 is in the middle.

```
x.mean()
```

```
area    2289.0
dtype: float64
```

```
y.mean()
```

```
37269.166666666664
```

#Data Splitting.

#What is data splitting?

In data science or machine learning, data splitting comes into the picture when the given data is divided into two or more subsets so that a model can get trained, tested and evaluated.

In practice or in real-life projects, data splitting is an important aspect, and it becomes a must when models are based on the data as it ensures the making of machine learning models. Usually, we create two or three parts of the main dataset.

-If two splits are there, it means one will be utilised for training and another one will be used for testing, or, -If three splits are there will mean there are training, testing and validation sets.

```
from sklearn.model_selection import train_test_split
train, test=train_test_split(df, test_size=.30)
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

```
train.head()
```

```
    area   rent
11  2100  32000
56  2750  48500
15  3100  51500
28  2200  31460
26  2750  48000
```

```
train.shape
```

```
(42, 2)
```

```
test.shape
```

```
(18, 2)
```

```
df.shape
```

```
(60, 2)
```

#Seperate train and test for xtrain, xtest, ytrain, ytest.

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=.30,
random_state=42)
```

```
xtrain.head()
```

```
    area
40  2150
4   3000
43  2510
19  2700
34  3000
```

```
ytrain.head()
```

```
40    34400
4     52100
43    40000
19    47000
34    52200
Name: rent, dtype: int64
```

#Linear Regression Algorithm.

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()

reg.fit(xtrain, ytrain)

LinearRegression()
```

#Finding important values.

```
reg.coef_

array([20.68636687])

reg.intercept_

-10252.90474033673
```

#Predict a random value.

```
reg.predict([[2510]])

/opt/conda/lib/python3.10/site-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(

array([41669.87610729])
```

#Residual

Residual formula: Observed Value - Predicted Value

```
residual = 41669.87610729-40000
residual

1669.8761072900015
```

#Linear Regression Formula.

```
#Y = (M*X) + C
Y = ((reg.coef_)*2510)+(reg.intercept_)
Y
```

```
array([41669.87610729])
```

```
xtest.head()
```

```
    area
0   2000
5   1900
36  1500
45  3100
13  2510
```

```
reg.predict(xtest) # corrosponding to ytest
```

```
array([31119.82900279, 29051.19231563, 20776.64556701, 53874.83256151,
       41669.87610729, 47668.92250004, 36291.42072068, 26982.55562848,
       33188.46568994, 26982.55562848, 26982.55562848, 33395.32935866,
       33188.46568994, 36291.42072068, 62149.37931013, 21810.96391059,
       45600.28581288, 20776.64556701])
```

```
ytest
```

```
0     31500
5     32500
36    20200
45    51000
13    40050
54    51000
33    35100
48    25500
12    34500
57    25100
46    30000
50    31000
31    35010
3     36100
52    70500
17    21000
8     48000
6     20000
Name: rent, dtype: int64
```

```
pred = reg.predict(xtest)
pred
```

```
array([31119.82900279, 29051.19231563, 20776.64556701, 53874.83256151,
       41669.87610729, 47668.92250004, 36291.42072068, 26982.55562848,
       33188.46568994, 26982.55562848, 26982.55562848, 33395.32935866,
```

```
       33188.46568994, 36291.42072068, 62149.37931013, 21810.96391059,
       45600.28581288, 20776.64556701])
```

#MAE and MSE Formula.

MAE and MSE have similar names and the same goal, to measure the error of regression models, but they are not the same. They actually have quite different approaches to measuring the prediction error.

1. MAE (Mean Absolute Error) is the average absolute error between actual and predicted values.

Absolute error, also known as L1 loss, is a row level error calculation where the non-negative difference between the prediction and the actual is calculated. MAE is the aggregated mean of these errors, which helps us understand the model performance over the whole dataset.

MAE is a popular metric to use as the error value is easily interpreted. This is because the value is in the same scale as the target you are predicting for

1. MSE (Mean Squared Error) is the average squared error between actual and predicted values.

Squared error, also known as L2 loss, is a row level error calculation where the difference between the prediction and the actual is squared. MSE is the aggregated mean of these errors, which helps us understand the model performance over the whole dataset.

The main draw for using MSE is that it squares the error, which results in large errors being punished or clearly highlighted. It's therefore useful when working on models where occasional large errors must be minimised.

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

mae = mean_absolute_error(ytest, pred)
mae

2103.50832336242

mse = mean_squared_error(ytest,pred)
mse

7677066.248378809
```

#Model Score.

```
reg.score(xtest, ytest)

0.9515542918540623
```

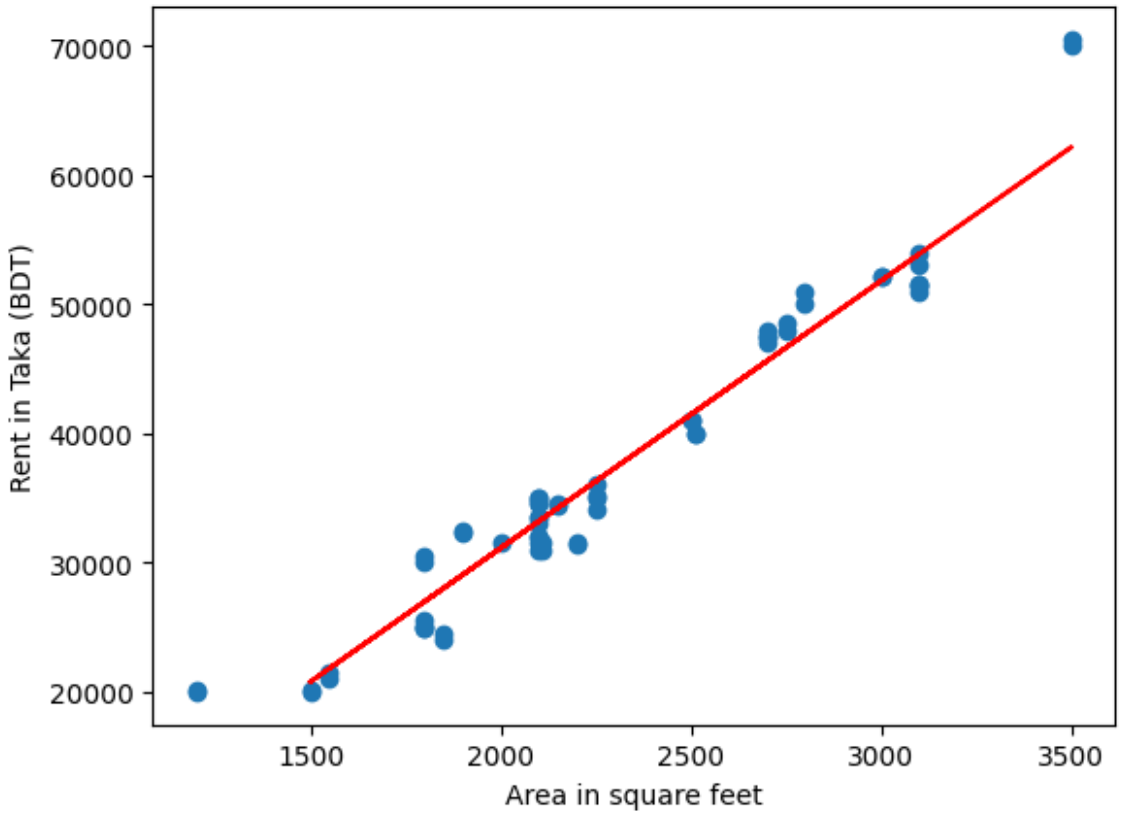
#Best Fit Line.

1. A line of best fit is a straight line that minimizes the distance between it and some data.

2. The line of best fit is used to express a relationship in a scatter plot of different data points.
3. It is an output of regression analysis and can be used as a prediction tool for indicators and price movements.
4. In finance, the line of best fit is used to identify trends or correlations in market returns between assets or over time.

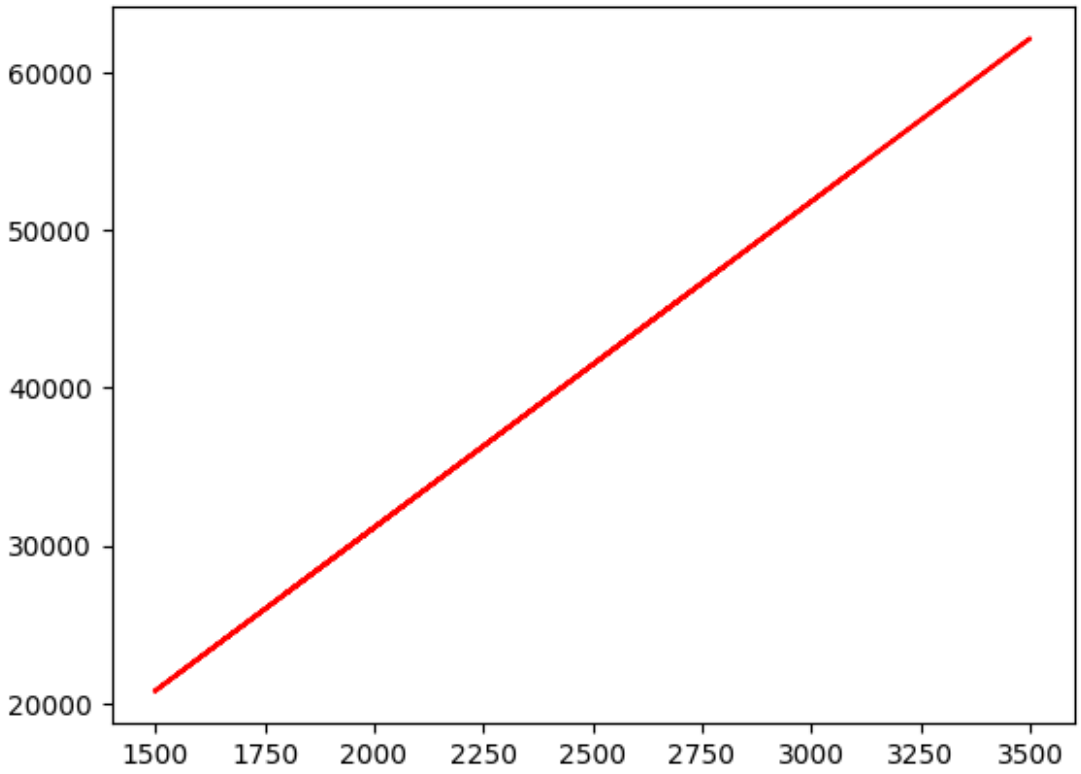
```
plt.plot(xtest, pred, color='red')
plt.scatter(x, y)
plt.xlabel('Area in square feet')
plt.ylabel('Rent in Taka (BDT)')

Text(0, 0.5, 'Rent in Taka (BDT)')
```




```
plt.plot(xtest.squeeze(), pred, color='red')

[<matplotlib.lines.Line2D at 0x79a6ea99a3e0>]
```



#Conclusion

In conclusion, the linear regression model serves as a powerful tool for analyzing and modeling relationships between variables in a linear fashion. By fitting a straight line to the data, it provides a clear representation of the underlying patterns and trends. However, it is crucial to acknowledge the assumptions of linearity, independence, homoscedasticity, and normality for the model to be reliable.

The effectiveness of the linear regression model depends on the nature of the data and the appropriateness of the assumptions. It is essential to critically evaluate the results, considering factors such as outliers, multicollinearity, and the potential impact of influential data points. Regular validation techniques, such as cross-validation, help ensure the model's generalization to new data.

Moreover, continuous improvement and refinement of the model can be achieved through feature engineering, incorporating domain knowledge, or exploring more advanced regression techniques when faced with complex relationships.

In practical applications, the linear regression model provides valuable insights, aiding in prediction, decision-making, and understanding the underlying dynamics of the data. As with any statistical model, it is imperative to interpret the results within the context of the specific problem domain and be cautious of overfitting or oversimplification.

In summary, while linear regression is a fundamental and widely used technique, its success hinges on careful consideration of assumptions, continuous model evaluation, and a thoughtful approach to handling real-world complexities.