

## 1. What you learned in the ICP:

Linear Regression is the most statistical technique that is used for prediction. Its applied on a causal relationships between two or more variables. In this ICP, I learned how to implement a linear regression model on the Houses Dataset in order to show the predicting price of the houses. Also, I found the best fit line in the data and calculating the Mean Squared Error, Y intercept, and Slope. I'll explain what I did in details in the upcoming part of the report.

## 2. Screen shots that shows the successful execution of each required step of your code:

First of all, I have import all the required libraries and read the data using pandas function `read_csv`.

The screenshot shows a Jupyter Notebook interface with the following code cells:

```
[38] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
[39] train = pd.read_csv('houses_dataset.csv')
train.shape
```

The output of the second cell is:

```
(1460, 81)
```

The third cell displays the first few rows of the 'train' dataset:

```
train
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	↑
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	↑
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	↑
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	↑
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	↑
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	↑
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	↑
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Crawfor	Norm	↑
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	↑

The notebook status bar at the bottom indicates: 0s completed at 4:43 PM.

In this step, I built the linear model by creating a new variable called (y = SalePrice), and all the rest of columns to X (except SalePrice, ID). Then, I split the data using the Sklearn library and fit the linear regression.

#### ▼ Build a linear model

```
✓ [42] y = np.log(train.SalePrice)
0s    X = data.drop(['SalePrice', 'Id'], axis=1)
```

#### ▼ Train Test Split

```
✓ [43] from sklearn.model_selection import train_test_split
0s    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=.30)
```

#### ▼ Fit Linear Model

```
✓ [44] from sklearn import linear_model
0s    lr = linear_model.LinearRegression()
    model = lr.fit(X_train, y_train)
```

After I saved the model predict on the prediction variable, I calculate and print the following:

- Mean Squared Error: which tells us how close a regression line is to a set of points
- Mean Absolute Error: to know the amount of error in our measurements
- Root Mean Square Error: to tells us the standard deviation of the residuals (prediction errors)
- R Squared ( $R^2$ ): to show us how close the data are to the fitted regression line

*Please note that: Although the instructor has only asked for the **Mean Squared Error**, I did the rest functions in order to learn more about the linear regression.*

### ▼ Predict in testing data

```
[45] predictions = model.predict(X_test)
```

```
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
import math

mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
rmse = math.sqrt(mse)
r2 = r2_score(y_test, predictions)

print('Mean Squared Error is: \n', mse)
print('\n')
print('Mwan Absulate Error is: \n', mae)
print('\n')
print('Root Mean Square Error (RMSE) is: \n', rmse)
print('\n')
print("R Squared (R²) Score is: \n", r2)
```

```
Mean Squared Error is:
0.022179533395466976

Mwan Absulate Error is:
0.10907817789132872

Root Mean Square Error (RMSE) is:
0.14892794699272188

R Squared (R²) Score is:
0.8692600366399272
```

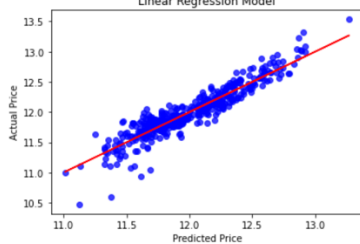
In the last screenshot, plot the data in a scatter plot and print out the **intercept** and **slope**.

- Intercept: to indicates the location where it intersects an axis
- Slope: to indicates the steepness of a line.

### ▼ visualize

```
actual_values = y_test
plt.scatter(predictions, actual_values, alpha=.75, color='b') #alpha helps to show overlapping data
plt.plot(predictions, model.predict(X_test), color='red')
plt.xlabel('Predicted Price')
plt.ylabel('Actual Price')
plt.title('Linear Regression Model')
plt.show()
```

```
#Y intercept, and Slope for the relationship in data
print('\n')
print("intercept = ", model.intercept_, "solpe = ", model.coef_[0])
```



```
intercept = 10.343091966865943 solpe = -0.0008492436316884773
```

3. Conclusion:

I have Successfully executed the code and built a Linear Regression model. As shown in the screenshot above, the MSE result has less than 0 which means the model is perfect and the prediction value is close to the actual one. On the other side,  $R^2$  should reach more than 80% to indicate a better fit for the model. However, I got a positive Y intercept and negative slope, which means the line crosses the y-axis above the origin.

4. Video link:

[https://drive.google.com/file/d/1dkhOZIIFufMKw2FIEbrnhIVxf\\_dT6VF-/view?usp=sharing](https://drive.google.com/file/d/1dkhOZIIFufMKw2FIEbrnhIVxf_dT6VF-/view?usp=sharing)