Pacmann Research Scientist Take Home Test

Iqbal Ahmad Dahlan (dahlanelka@gmail.com)

1.You'll be asked to create a **fully functional** Python module **from scratch** of **one of these** Models:
Linear Regression from scratch:

- You are only allowed to use the basic **NumPy function**.
- The module components have to be in the form of functions. **Plus point** if you can make the whole module OOP.
- At least, the model must have "fit" and "predict" functions as an interface to the user.
- The Linear Regression model must have several options for the cost function. (Minimum "Mean Squared Error" and "Mean Absolute Error")
- For the benchmark, you can use the scikit-learn model as a benchmark. Here's the gist: https://gist.github.com/fazaghifari/5c16db847c9217b2a603733055ec7eb5

2.Please keep in mind that **fully-functional** means:

- The module is in the form **Python Script** not **Jupyter Notebook**.
- Users can seamlessly use your module for their needs.
- Has clear documentation.
- Please upload your code to **Github/GitLab** then send the link to us.

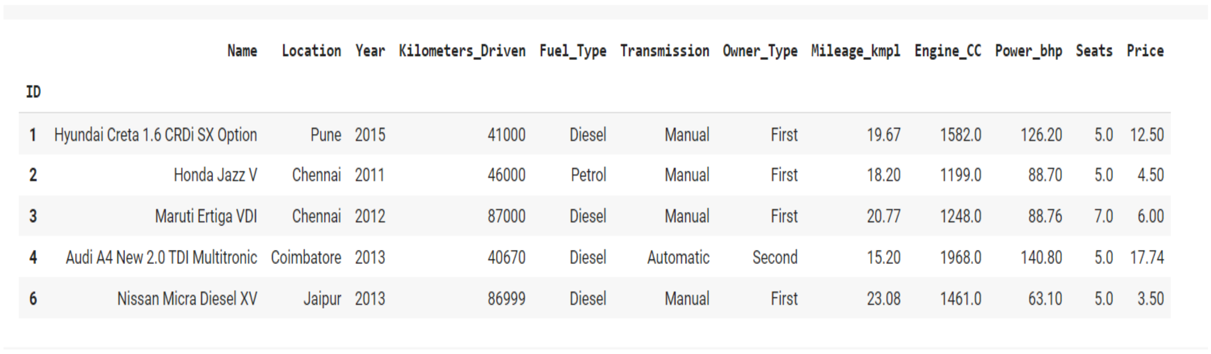
3.Your module has to satisfy these requirements:
Having a clear code structure and documentation of each class / function / method
Having a nice interface (e.g. your code may composed of many functions/modules. But user can easily use your code from a simple function such as):

- model = LinearReg(data_x, data_y, *args)
- model.fit()
- Y = model.predict(data_x2, *args)
- Provide us a demo script/demo notebook.
- For the demo you may use your own / external dataset.

Background

Figure 1.Illustration second car (https://www.liputan6.com/)

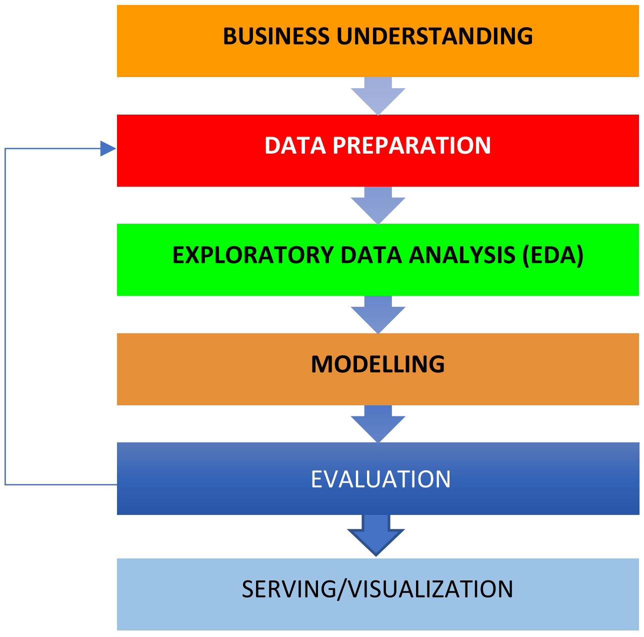
So called Second hand's car have a huge market base.One other hand,There are also many frauds in the market who not only sale wrong but also they could mislead to wrong price.So, here I used this following dataset to Predict the price of any used car.



ID	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage_kmpl	Engine_CC	Power_bhp	Seats	Price
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67	1582.0	126.20	5.0	12.50
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.20	1199.0	88.70	5.0	4.50
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77	1248.0	88.76	7.0	6.00
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.20	1968.0	140.80	5.0	17.74
6	Nissan Micra Diesel XV	Jaipur	2013	86999	Diesel	Manual	First	23.08	1461.0	63.10	5.0	3.50

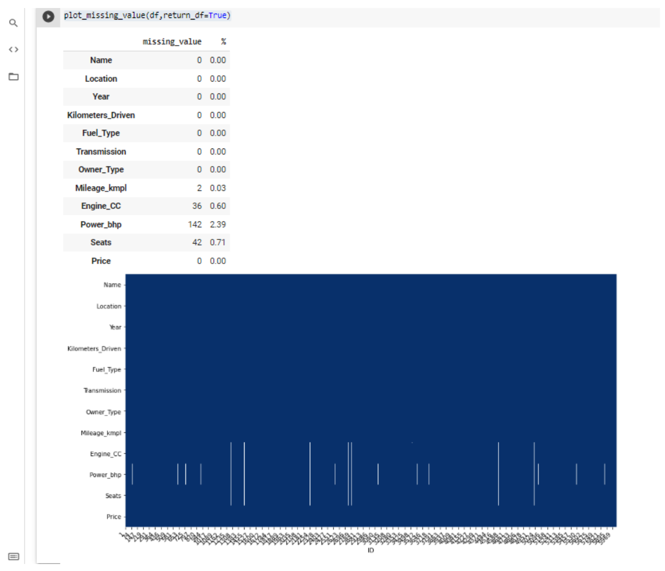
Method

This method is designing to predict the car's price value with machine learning. This will be elaborated into some steps.



Target this method is predicting price based on all parameters.

Plot missing value



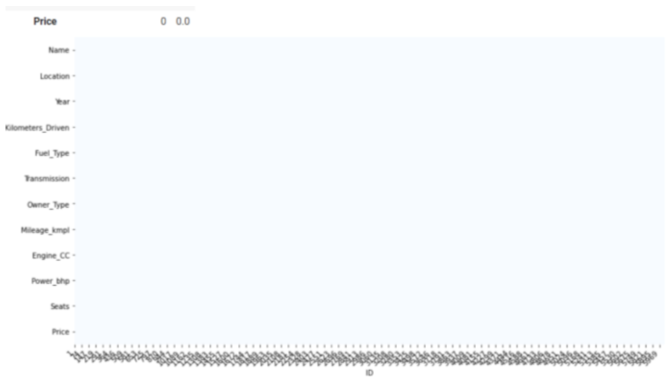
```
plot_missing_value(df,return_df=True)
```

	missing_value	%
Name	0	0.00
Location	0	0.00
Year	0	0.00
Kilometers_Driven	0	0.00
Fuel_Type	0	0.00
Transmission	0	0.00
Owner_Type	0	0.00
Mileage_kmpl	2	0.03
Engine_CC	36	0.60
Power_bhp	142	2.39
Seats	42	0.71
Price	0	0.00

```
df = df.fillna(df.mean())
df.isnull().sum()
```

Because the parameters are numeric.The missing valu will be changed into mean value.

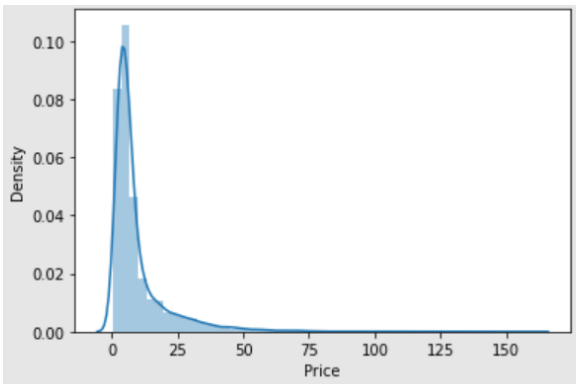
Figure below shows that there are no missing values again.



Price	0	0.0

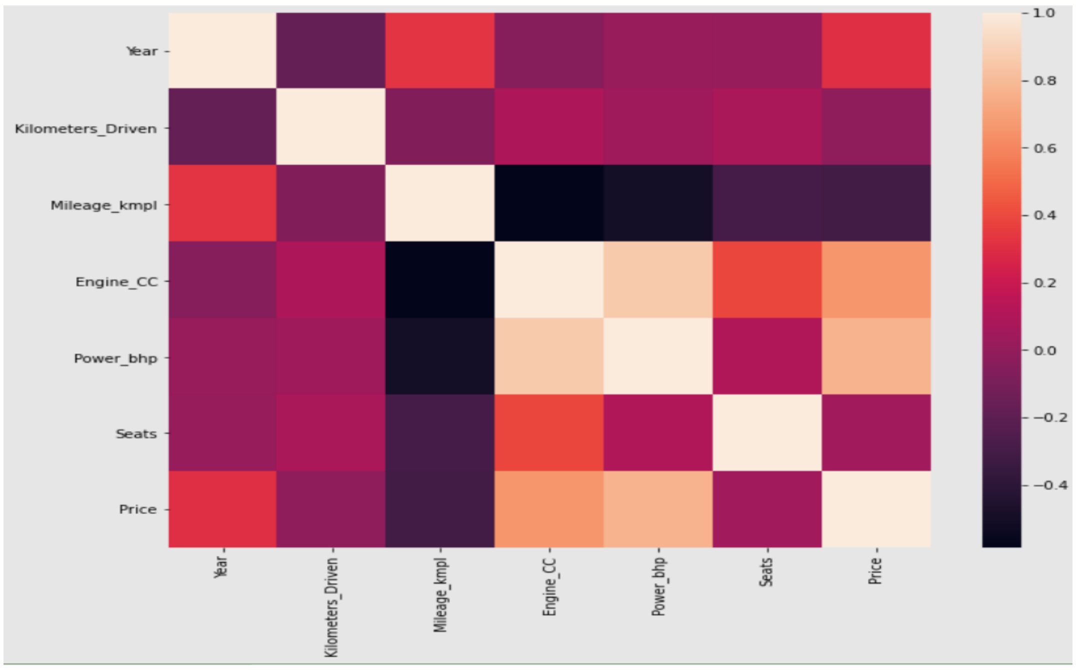
Exploratory Data Analysis

Firstly, we want to plot the target distribution

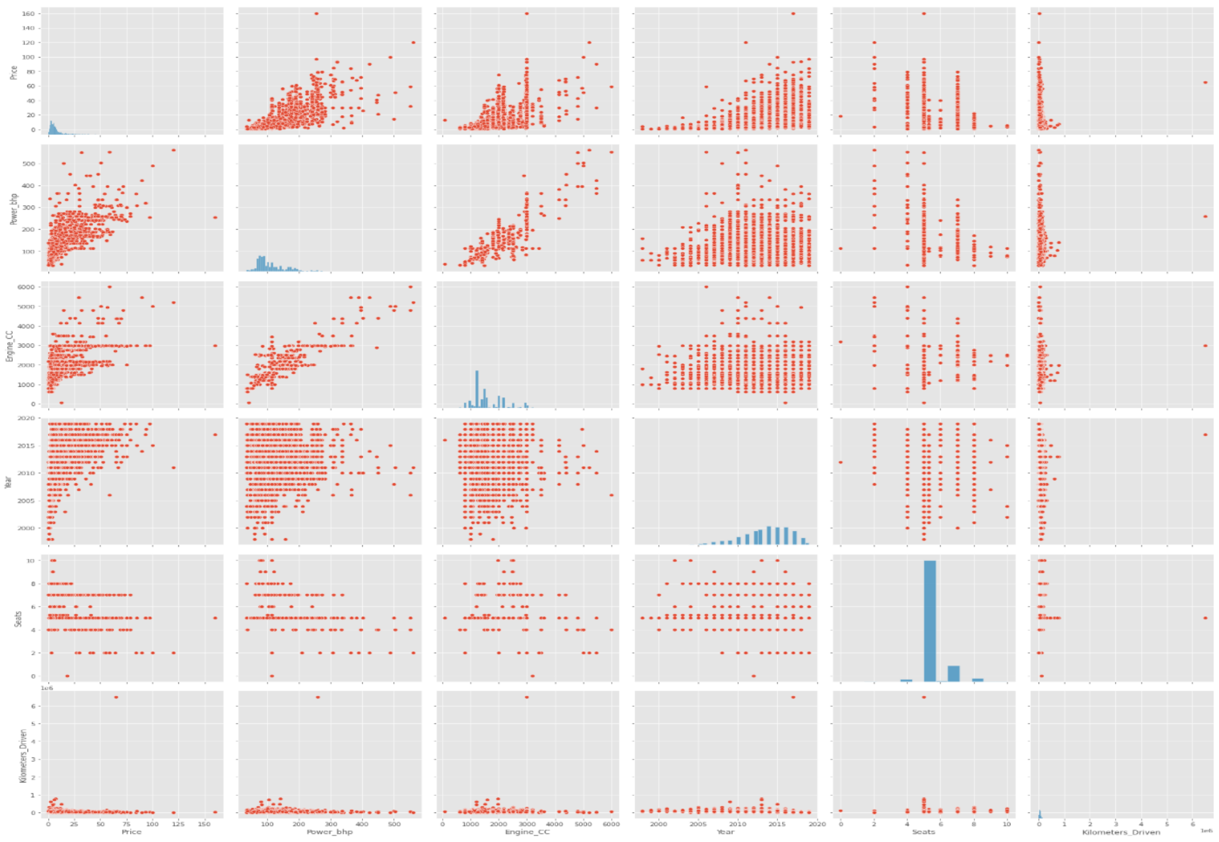


Secondly we want to check the correlation between parameters and the target

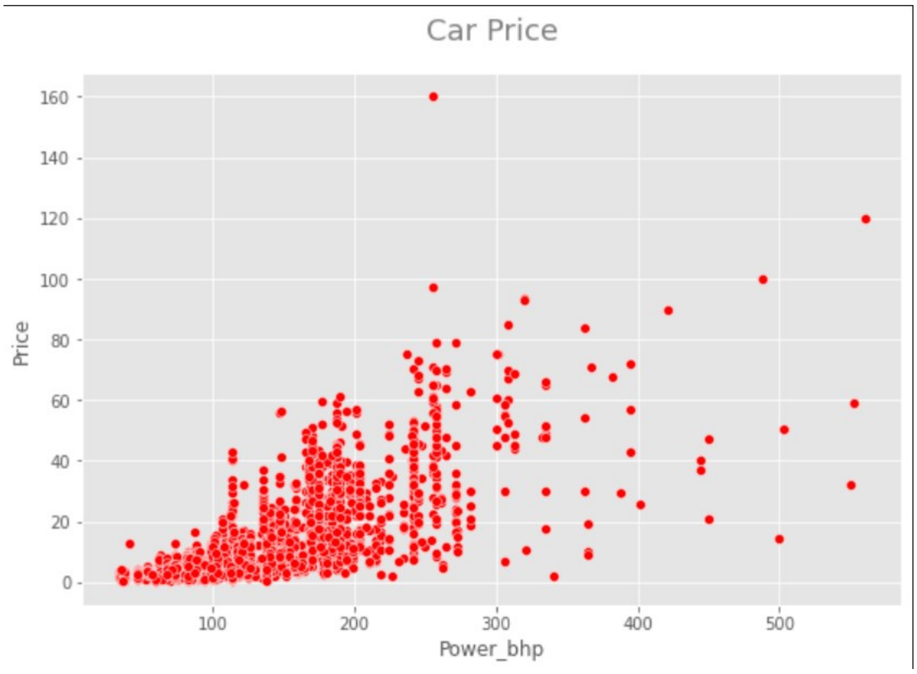
```
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=1, square=True);
```



In this first assignment, this will be predicted with Numpy from scratch (linear regression). This method must be visualized the distribution from parameters.



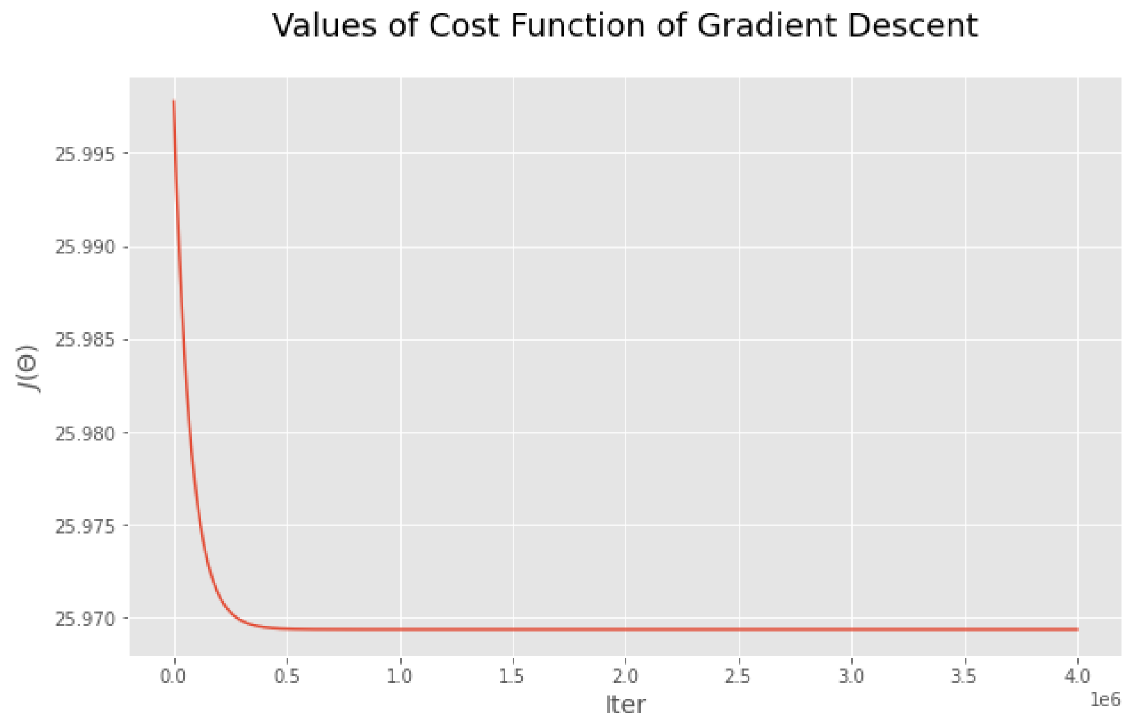
Based on two figures above, we pick Power_bhp as a parameter and Price as a target.



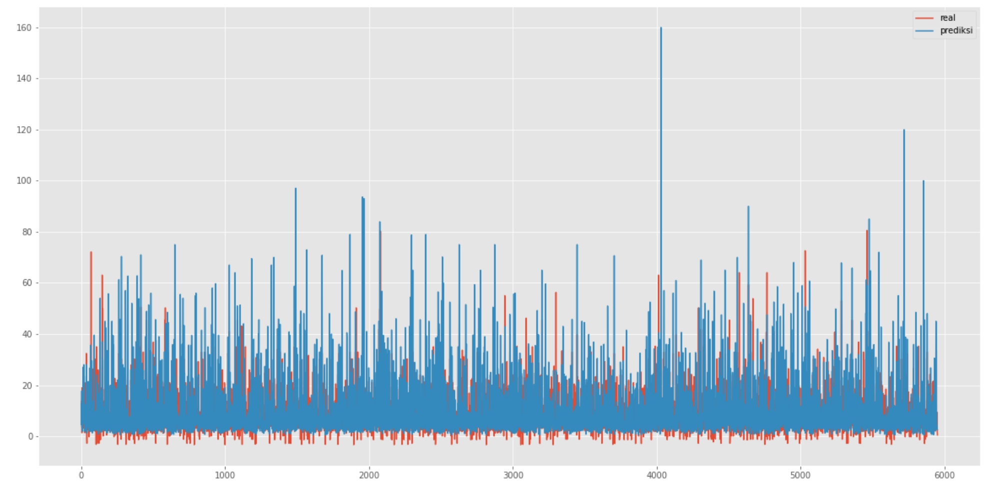
Based on this program .

(https://github.com/dahlan77/Pacman-Test-Dahlan/blob/main/Pacman_Test_Dahlan_Jupyter(Car_Price).ipynb.

Gradient Descent Plot (cost = `25.969369881500555)`



Plot prediction (real vs prediction)



Plot prediction with theta1 and theta 2

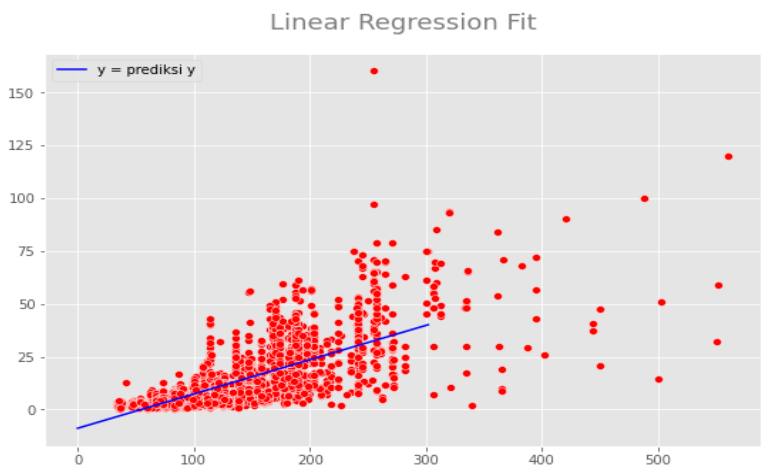
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(9, 6))
sns.scatterplot(x='Power_bhp', y='Price', data=df, ax=ax, color='red')

# Create random data based on theta and x value
a = np.round(theta, 3)
x_value = np.array(range(0, 303))
# theta + theta * x
y_value = a[0, 0] + (a[1, 0] * x_value)


sns.lineplot(x_value, y_value, ax=ax, label='y = prediksi y', color='bl
ue')

ax.set_title('Linear Regression Fit', pad=20, size=18, color='gray')
ax.set_xlabel('TV advertising spending in $')
ax.set_ylabel('Sales in $1000s')
ax.legend(loc='upper left')
plt.savefig('manual_linear_regression_fit.png')
```



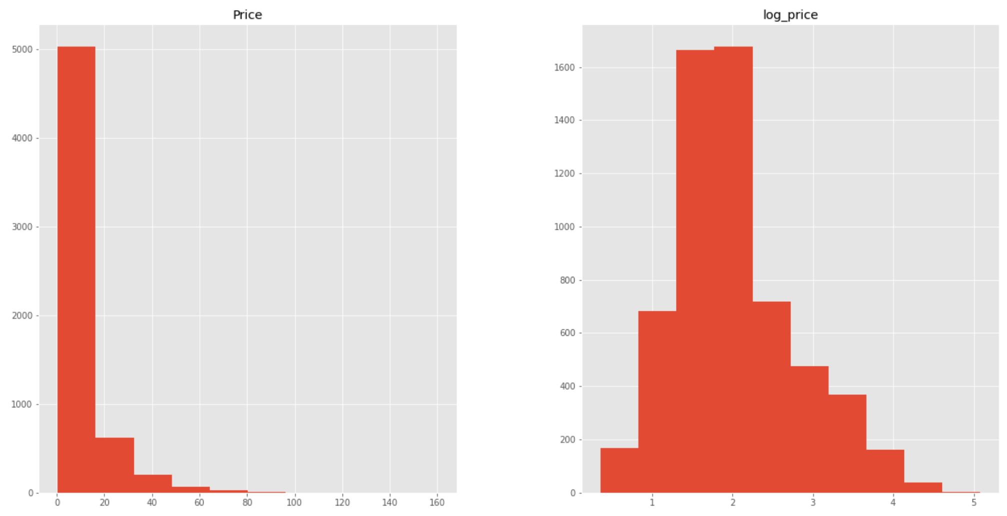
```
R^2: 0.81 RMSE: 7.21
```

Version 2 – Target Engineering

Logarithm transformation (or log transform) is one of the most commonly used mathematical transformations in feature engineering. What are the benefits of log transform:

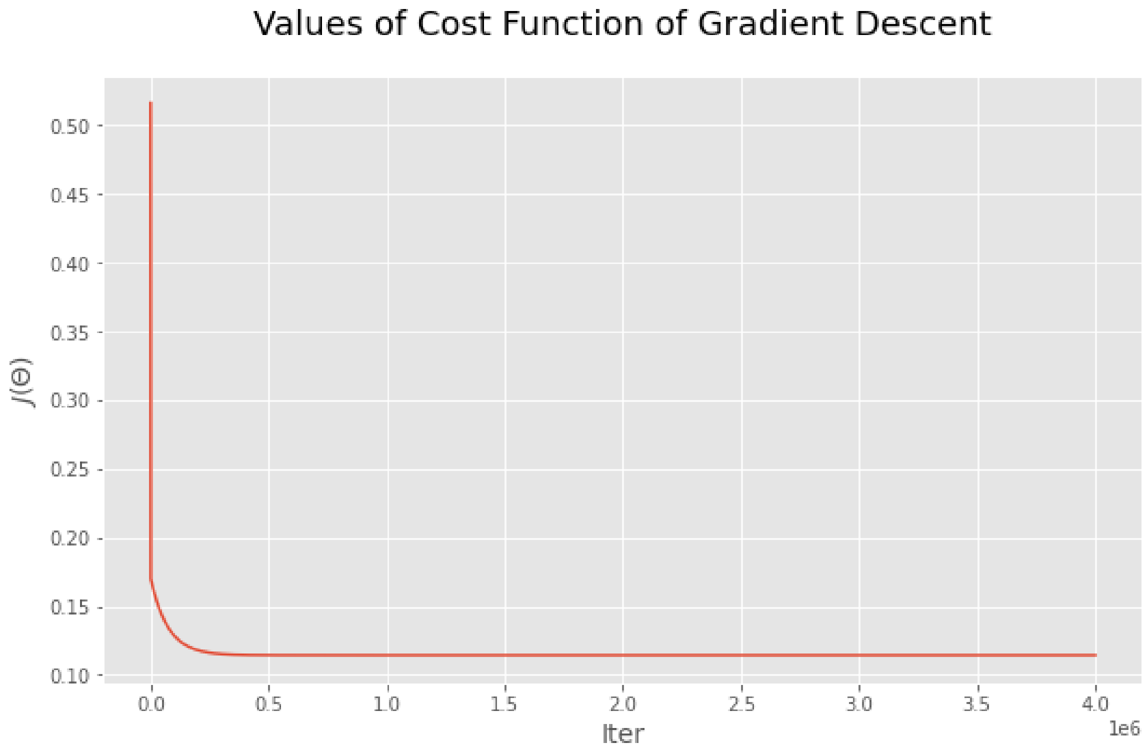
It helps to handle skewed data and after transformation, the distribution becomes more approximate to normal.

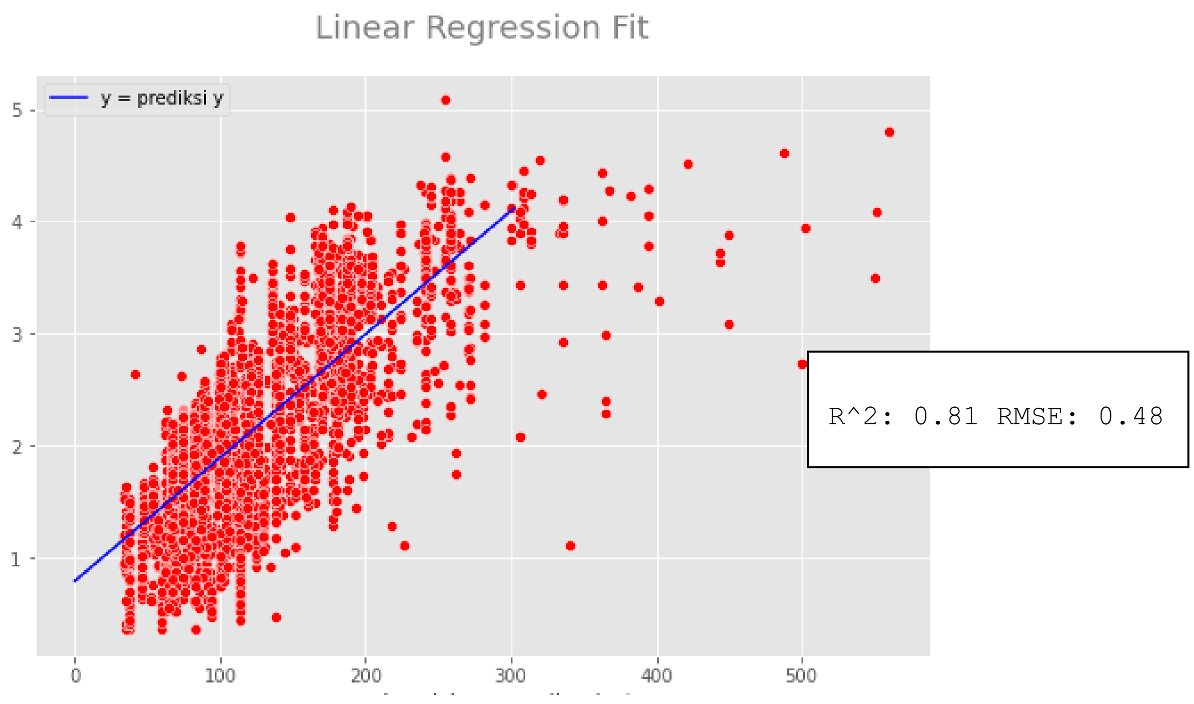
It will be check ,the prediction is better or no because the data's target is skew.



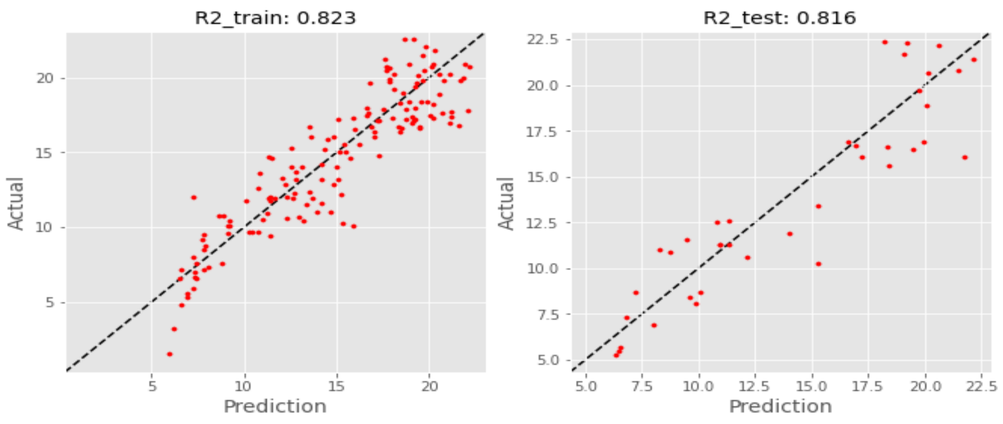
It will be analyzed with the same process and the output is will be elaborated below:

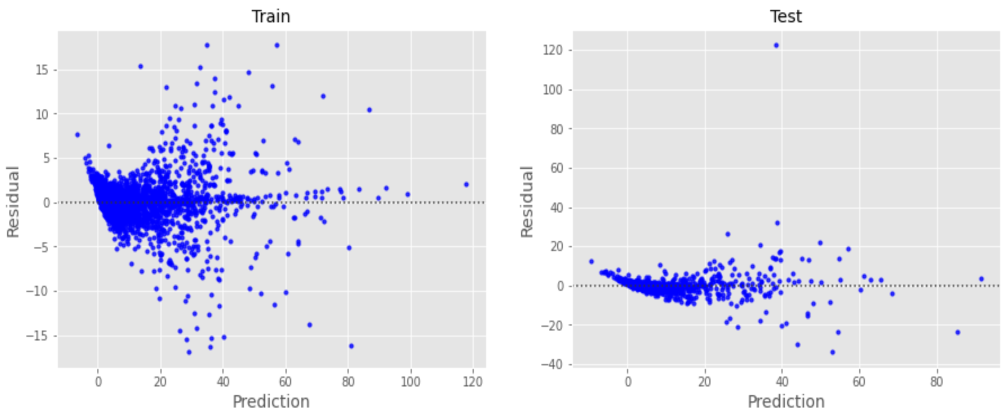
Suprisingly,the cost is better with the value at (`0.11410315119667343`)





The other version is using (scikit-learn) also in the same jupyter notebook.The result shows:


R2_train: 0.823
Actual
Prediction
R2_test: 0.816
Actual
Prediction


Train
Residual
Prediction
Test
Residual
Prediction