Linear Regression with Artificial Generated Dataset

Import Library

In [1]:

```
import pandas as pd
import numpy as np
```

Generate Dataset

In [2]:

```
from sklearn.datasets import make_regression
```

In [4]:

```
# without coefficient of underline model
X,y =  make_regression(n_samples=500,n_features=5,coef=False,bias=12,noise=10,random_state=
```

In [5]:

```
coeficient of underline model return
= make_regression(n_samples=500,n_features=5,coef=True,bias=12,noise=10,random_state=2529)
```

In [6]:

```
X.shape,y.shape
```

Out[6]:

```
((500, 5), (500,))
```

In [7]:

```
w #coeficient of x
```

Out[7]:

```
array([29.45661718, 60.14529878, 61.7409438 , 13.32437893, 99.08122896])
```

Get the first five rows of target variable(y) and features(X)

In [8]:

```
X[0:5]
```

Out[8]:

```
array([[ 0.77913208, -1.09701784, -0.14239962,  1.02427891, -1.0708024 ],
       [-0.6925009 ,  0.45535977,  0.34707569, -0.32456746,  0.21970203],
       [-0.03901601, -0.3265115 ,  0.59793721,  0.61686653, -0.6237489 ],
       [-0.61566117, -0.11782129, -0.98234619, -0.78292727,  0.42713048],
       [ 1.30822207, -0.72541559,  0.60187975,  0.33285998,  1.48506184]])
```

In [9]:

```
y[0:5]
```

Out[9]:

```
array([-136.21858395,   49.83118244,  -29.81097858,  -31.74001475,
        193.0687778 ])
```

get shape of Dataframe

In [10]:

```
X.shape, y.shape
```

Out[10]:

```
((500, 5), (500,))
```

Get Train_Test_Split

In [11]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3,random_state=2528)
```

In [15]:

```
X_train.shape,X_test.shape, y_train.shape, y_test.shape
```

Out[15]:

```
((350, 5), (150, 5), (350,), (150,))
```

Get Linear Regression Model Train

Linear Regression (*,fit intercept =True, copy X=True,n jobs=None,positive=False)

fit_intercept: bool, default=True whether to calculate the intercept for this model.if set to False, no intercept will be used in calculations(i.e. data is expected to be centered)

copy_X:bool,default=True if True, X will be copied; else, it may be overwritten.

n_jobs: int,default=None The Number of jobs use for the computation. this will only provide speedup in case of sufficiently large problems, that is if firstly n_targets>1 and secondly X is sparse or if positive is set to true.None means 1 unless in a joblib.parallel_backend context.-1 means using all processors. see Glossary for more details.

Positive:bool,default=False when set to True, forces the coefficients o be positive. This option is only supported for dense arrays

In [17]:

```
from sklearn.linear_model import LinearRegression
```

In [18]:

```
model = LinearRegression()
```

In [19]:

```
model.fit(X_train,y_train)
```

Out[19]:

```
LinearRegression()
```

Get Intercept and Coefficents

In [20]:

```
model.intercept_
```

Out[20]:

```
12.935789982359488
```

Bias introduced at the time of generating dataset is 12 and predicted intercept is very close to 12. The difference is due to noise=10 introduced at the time of data generation

In [21]:

```
model.coef_
```

Out[21]:

```
array([30.92125443, 60.0960533 , 60.51449229, 13.47444154, 98.86261499])
```

Respective weights or coefficient at the time to data generation are[29,45661718, 60.14529878, 61.7409438, 13.32437893, 99.08122896] and coeficient calculated(model_coef) are close to acual.Difference due to noise introduced during data generation process

Get Model Prediction

In [22]:

```
y_pred = model.predict(X_test)
```

In [23]:

```
y_pred.shape
```

Out[23]:

```
(150,)
```

In [24]:

```
y_pred
```

Out[24]:

```
array([-6.31061545e+01, -1.53122743e+02, -1.49960838e+02,  8.95188450e+01,
        2.45092106e+02, -1.40468826e+01, -4.73082131e+01,  1.76686378e+02,
       -1.01696730e+02,  4.91935456e+01,  7.37482185e+01, -6.78057643e+01,
       -1.98034449e+02,  5.08009762e+01,  4.43842560e+01,  1.83042477e+01,
       -8.65049852e+01,  6.72088274e+01,  1.09911652e+02,  6.17675180e+01,
       -1.03616297e+02, -7.13328502e+00, -1.11016255e+02,  1.50749792e+02,
       -2.33071900e+02,  2.48680011e+02,  2.52055816e+02, -8.60233041e+01,
       -4.84154459e+01,  3.66090530e+02,  1.57621409e+02,  5.58630758e+01,
        3.36456351e-02,  4.92680006e+01,  8.11944040e+01,  3.29853142e+02,
        1.32782904e+02, -2.52299841e+01,  1.62292318e+02, -2.25886830e+01,
        2.80960362e+02, -1.11309510e+02,  3.31370880e+02,  9.00193090e+01,
       -7.99192278e+01,  1.01418678e+02,  1.48381149e+01,  2.58456785e+01,
       -7.35221125e+01, -2.18672015e+02, -7.44588530e+01, -7.00973141e+00,
       -5.96431184e+01, -9.12223227e+01,  1.19823542e+01,  3.94949544e+01,
        4.06586220e+01, -8.65681424e+01,  3.04170322e+02, -2.37143987e+01,
       -3.18925204e+02,  5.77401611e+01,  2.19138786e+02, -3.90574560e+01,
        8.53095090e+01,  4.09897197e+00,  2.36509741e+02, -1.61070099e+01,
        2.28836070e+01,  2.44696068e+02, -1.14489508e+02, -1.02708398e+02,
        8.74793477e+00,  4.39176134e+01,  1.00478470e+02,  2.40458367e+01,
       -9.87597159e+01, -1.78534176e+02,  8.96035740e+01,  7.48394620e+00,
       -1.33159091e+01,  4.47463467e+01,  8.76324184e+01,  7.02833528e+01,
        9.91376406e+01, -1.65239582e+02, -3.50498388e+01, -2.44828722e+02,
       -5.76441635e+01,  1.73977450e+02, -5.45679941e+00, -1.50941537e+02,
        9.20448257e+01, -1.62202739e+01, -1.14994466e+00,  3.67111771e+01,
       -2.06860881e+02, -3.43449358e+01,  2.20223567e+02,  6.30510011e+01,
        1.37093146e+01,  3.02354074e+02,  2.43470167e+01,  6.74642134e+01,
        2.18273733e+01,  6.35506249e+01,  4.18787757e+01, -5.31115225e+01,
        2.29518775e+02, -2.69146622e+00, -2.36238871e+01,  6.49483837e+01,
        7.65403704e+01,  8.99849830e+01, -1.68986341e+00,  3.03680161e+01,
        5.63153790e+01,  5.39519879e+01, -2.35308524e+02, -6.38811086e+01,
        1.54932909e+02,  1.21909017e+02, -2.71352962e+02,  5.44087562e+01,
        4.02377490e+01, -1.44045052e+02, -2.31927324e+01,  1.87958264e+01,
        1.99065103e+02,  1.43940452e+02, -1.79034343e+02, -1.20398953e+02,
        2.19980542e+02, -3.27178625e+01, -1.45169498e+02, -2.80199851e+02,
        1.34081042e+02, -1.74274320e+01,  4.04977017e+01,  7.46754452e+01,
        4.54471909e+02,  1.56628260e+02, -1.39936889e+01, -7.83929873e+01,
       -8.97809825e+01, -3.31816114e+01, -5.57821435e+01,  2.45816636e+02,
       -2.24391472e+02,  1.36729652e+02])
```

Get Model Evaluation

In [25]:

```
ics import mean_squared_error, mean_absolute_error,mean_absolute_percentage_error,r2_score
```

In [26]:

```
mean_squared_error(y_test,y_pred)
```

Out[26]:

```
116.21079848749177
```

In [27]:

```
mean_absolute_error(y_test,y_pred)
```

Out[27]:

```
8.971292315023243
```

In [28]:

```
mean_absolute_percentage_error(y_test,y_pred)
```

Out[28]:

```
0.5261384999461314
```

In [29]:

```
r2_score(y_test,y_pred)
```

Out[29]:

```
0.9938588749436735
```

In [ ]:

```

```