

Machine Learning Project

In this project, I have used **Linear Regression** Machine Learning model for the PC parts prices Dataset from kaggle website.

Problem Statement

Compare CPU prices with raising demand for memory chips in smart phone industry and older parts tend to stop being made. Prices don't really go down. Prices stay static. High Prices of certain components may hikes the actual price of the PCs parts. By Checking the Product id, Merchant Id and Region Id we can predict the Original Price of the CPU.

Machine Learning (Methodology)

The methodology used for training and testing the dataset is **Linear Regression**.

Linear Regression is a method to explain the relationship between a dependent variable and one or more explanatory variables using a straight line. It is a special case of regression analysis. This method is mostly used for forecasting and finding out cause and effect relationship between variables. Models depend linearly on their unknown parameters are easier to fit the models which are non-linearly related to their parameters.

Dataset Description

Some relevant columns in the dataset

- 1664-CPU Products
- ProdId -Id of a CPU Product
- TimeId -Id of a Time Dimension
- RegionId -Id of a Region Dimension
- Merchant -Id of a Merchant Dimension
- Price_USD -Price in USD
- Price_Original -Price in Regional Currency

Pre-Processing:

Pre-processing involves transforming raw data into an understandable format. For example: Extracting data from a larger dataset.

1. %matplotlib inline

Import matplotlib.pyplot as plt

Import numpy as np

import pandas as pd

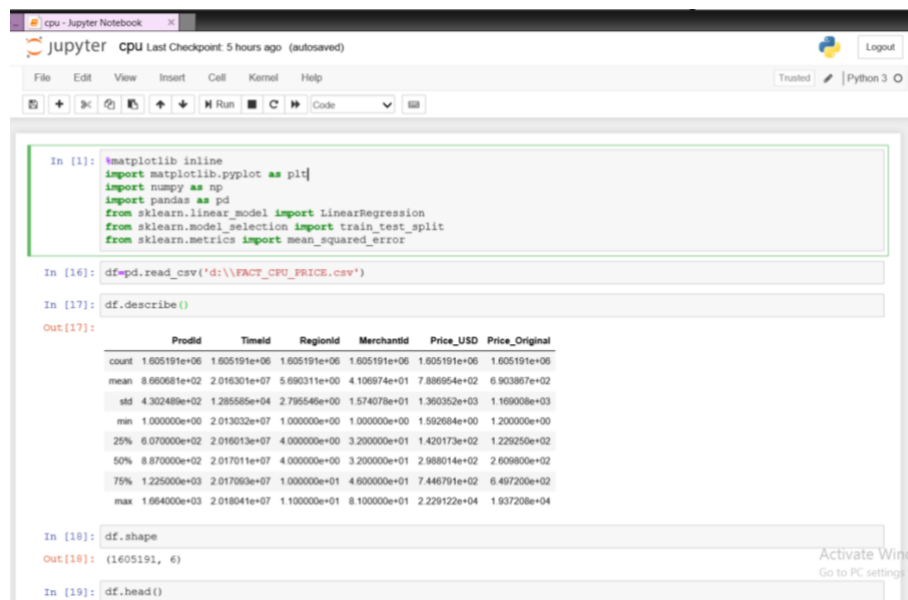
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

2. df=pd.read_csv('d:\\FACT_CPU_PRICE.csv')

df.describe()



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

```
In [1]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

In [16]: df=pd.read_csv('d:\\FACT_CPU_PRICE.csv')

In [17]: df.describe()

Out[17]:
```

	ProdId	TimeId	RegionId	MerchantId	Price_USD	Price_Original
count	1.605191e+06	1.605191e+06	1.605191e+06	1.605191e+06	1.605191e+06	1.605191e+06
mean	8.660681e+02	2.016301e+07	5.690311e+00	4.106974e+01	7.886954e+02	6.903867e+02
std	4.302489e+02	1.285585e+04	2.795549e+00	1.574078e+01	1.360352e+03	1.169008e+03
min	1.000000e+00	2.013032e+07	1.000000e+00	1.000000e+00	1.592684e+00	1.200000e+00
25%	6.070000e+02	2.016013e+07	4.000000e+00	3.200000e+01	1.420173e+02	1.229250e+02
50%	8.870000e+02	2.017011e+07	4.000000e+00	3.200000e+01	2.988014e+02	2.609800e+02
75%	1.225000e+03	2.017093e+07	1.000000e+01	4.600000e+01	7.446791e+02	6.497200e+02
max	1.664000e+03	2.018041e+07	1.100000e+01	8.100000e+01	2.229122e+04	1.937208e+04

```
In [18]: df.shape
Out[18]: (1605191, 6)

In [19]: df.head()
```

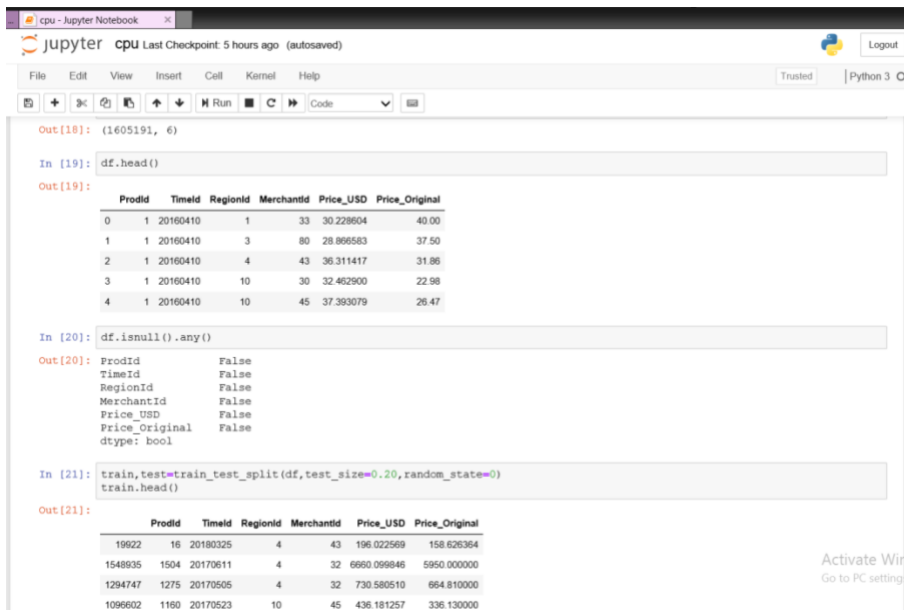
In[3]:df.shape

In[4]:df.head()

```
In[5]:df.isnull().any()
```

```
In[6]:train,test=train_test_split(df,test_size=0.20,random_state=0)
```

```
train.head()
```



The screenshot shows a Jupyter Notebook window titled 'cpu - Jupyter Notebook'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help), a toolbar with icons for file operations and code execution, and a status bar indicating 'Trusted' and 'Python 3'. The notebook content shows the following code and output:

```
Out[18]: (1605191, 6)
```

```
In [19]: df.head()
```

```
Out[19]:
```

	ProdId	TimeId	RegionId	MerchantId	Price_USD	Price_Original
0	1	20160410	1	33	30.229604	40.00
1	1	20160410	3	80	28.866583	37.50
2	1	20160410	4	43	36.311417	31.86
3	1	20160410	10	30	32.462900	22.98
4	1	20160410	10	45	37.393079	26.47

```
In [20]: df.isnull().any()
```

```
Out[20]: ProdId      False
TimeId      False
RegionId     False
MerchantId   False
Price_USD    False
Price_Original False
dtype: bool
```

```
In [21]: train,test=train_test_split(df,test_size=0.20,random_state=0)
train.head()
```

```
Out[21]:
```

	ProdId	TimeId	RegionId	MerchantId	Price_USD	Price_Original
19922	16	20180325	4	43	196.022569	158.626364
1548935	1504	20170611	4	32	6660.099846	5950.000000
1294747	1275	20170505	4	32	730.580510	664.810000
1096602	1160	20170523	10	45	436.181257	336.130000

```
In[7]:X_train=train[['RegionId','MerchantId','Price_USD']]
```

```
In[8]:y_train=train.Price_Original
```

```
In[9]:X_test=test[['RegionId','MerchantId','Price_USD']]
```

```
In[10]:y_test=test.Price_Original
```

```
In[11]:print(X_train.shape)
```

```
In[12]:print(X_test.shape)
```

```
In[13]:print(y_train.shape)
```

```
In[14]:print(y_test.shape)
```

```
In[15]:10.model=LinearRegression()
```

```
In[16]:model.fit(X_train,y_train)
```

```
In[17]:y_pred=model.predict(X_test)
```

The screenshot shows a Jupyter Notebook window titled 'cpu - Jupyter Notebook'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help), a toolbar with icons for file operations and execution, and a status bar indicating 'Trusted' and 'Python 3 C'. A data table is displayed at the top, followed by several code cells. The first code cell (In [42]) defines training data. The second (In [43]) defines test data. The third (In [45]) prints the shapes of the data arrays. The fourth (In [46]) fits a LinearRegression model. The output (Out[46]) shows the model's parameters. The fifth (In [47]) uses the model to predict values. The sixth (In [48]) prints the model's coefficients. The output (Out[48]) shows the coefficients array. The seventh (In [49]) prints the model's intercept.

1548935	1504	20170611	4	32	6660.096846	5950.000000
1294747	1275	20170505	4	32	730.580510	664.810000
1096602	1160	20170523	10	45	436.181257	336.130000
1321353	1288	20160515	4	32	1078.570056	953.855000

```
In [42]: X_train=train[['RegionId','MerchantId','Price_USD']]
         y_train=train.Price_Original

In [43]: X_test=test[['RegionId','MerchantId','Price_USD']]
         y_test=test.Price_Original

In [45]: print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)

(1284152, 3)
(321039, 3)
(1284152,)
(321039,)

In [46]: model=LinearRegression()
         model.fit(X_train,y_train)

Out[46]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [47]: y_pred=model.predict(X_test)

In [48]: model.coef_

Out[48]: array([-4.65956477, -1.6495825 ,  0.84698215])

In [49]: model.intercept_
```

In[18]:model.coef_

In[19]:model.intercept_

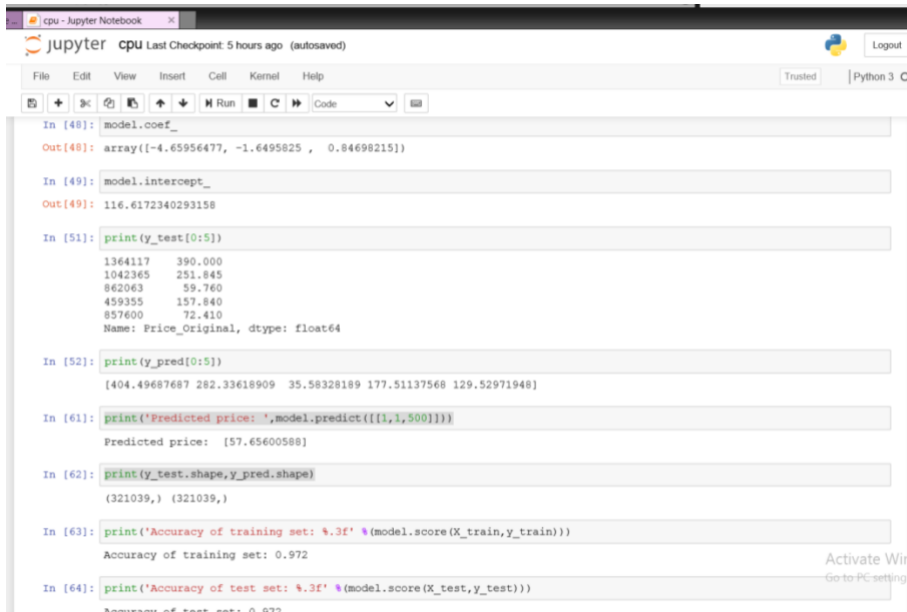
In[20]:print(y_test[0:5])

In[21]:print(y_pred[0:5])

In[22]:print('Predicted price: ',model.predict([[1,1,500]]))

In[23]:print(y_test.shape,y_pred.shape)

In[24]:print('Accuracy of training set: %.3f' %(model.score(X_train,y_train)))



```
In [48]: model.coef_  
Out[48]: array([-4.65956477, -1.6495825 ,  0.84698215])  
  
In [49]: model.intercept_  
Out[49]: 116.6172340293158  
  
In [51]: print(y_test[0:5])  
1364117    390.000  
1042365    251.845  
862063     59.760  
459355     157.840  
857600     72.410  
Name: Price_Original, dtype: float64  
  
In [52]: print(y_pred[0:5])  
[404.49687687 282.33618909 35.58328189 177.51137568 129.52971948]  
  
In [61]: print('Predicted price: ',model.predict([[1,1,500]]))  
Predicted price:  [57.65600588]  
  
In [62]: print(y_test.shape,y_pred.shape)  
(321039,) (321039,)  
  
In [63]: print('Accuracy of training set: %.3f' %(model.score(X_train,y_train)))  
Accuracy of training set: 0.972  
  
In [64]: print('Accuracy of test set: %.3f' %(model.score(X_test,y_test)))  
Accuracy of test set: 0.972
```

In[25]:print('Accuracy of test set: %.3f' %(model.score(X_test,y_test)))

In[26]:mse=mean_squared_error(y_test,y_pred)

In[27]:print('Mean Squared Error: %.3f' %mse)

In[28]:y_pred=y_pred.round()

In[29]:y_test=y_test.astype(float).round()

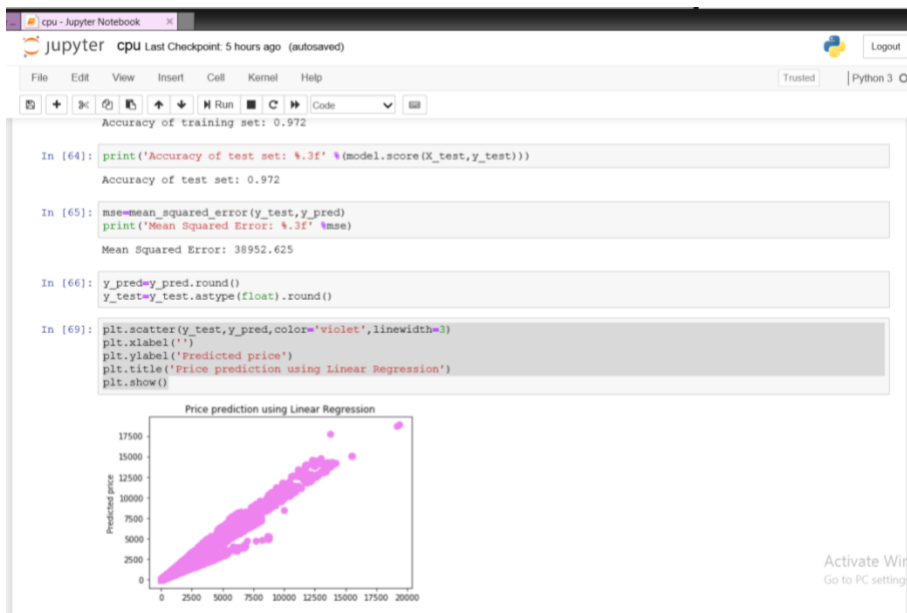
In[30]:plt.scatter(y_test,y_pred,color='violet',linewidth=3)

In[31]:plt.xlabel("")

In[32]:plt.ylabel('Predicted price')

In[33]:plt.title('Price prediction using Linear Regression')

In[34]:plt.show()



Conclusion:

Here the CPU price is predicted with merchant Id, Region Id and Price_USD using Linear Regression. So that we could find the cheapest PC's parts .