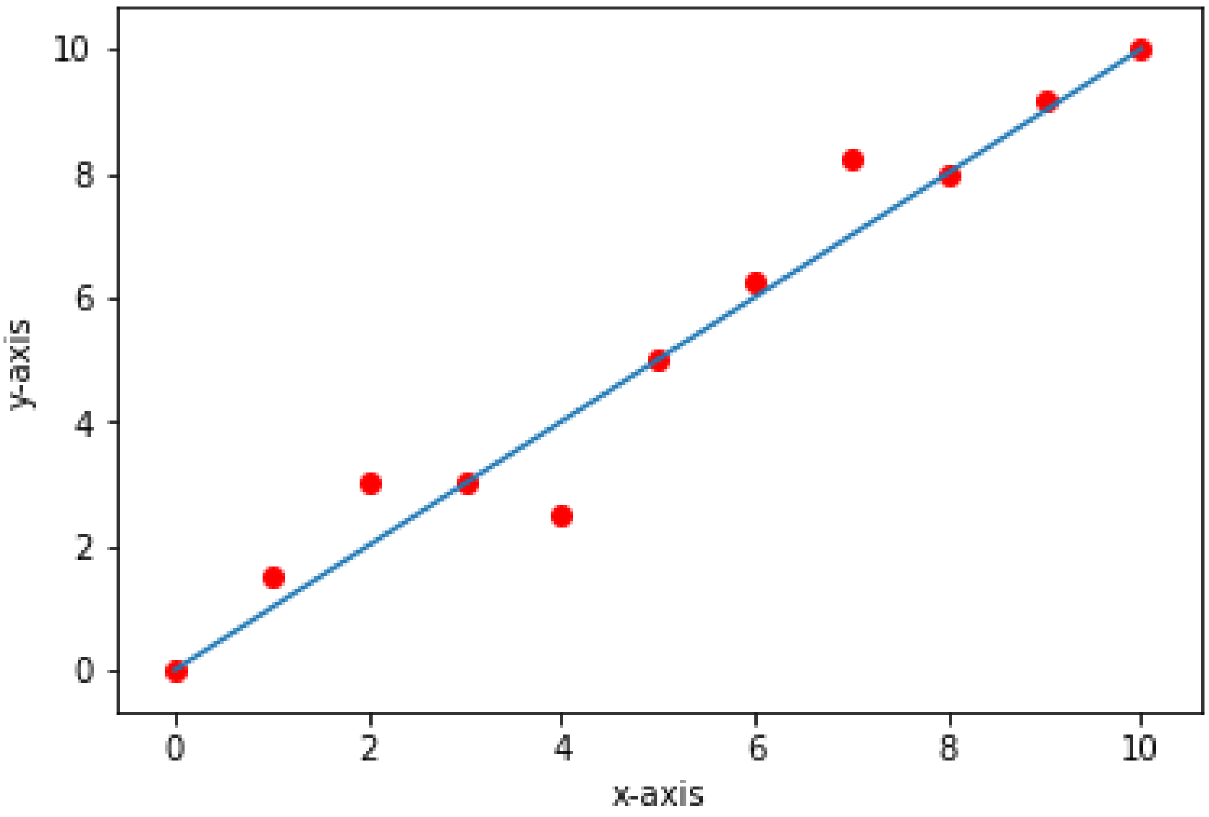
Linear Regression for Scikit Learn



The Theory

Linear Regression is the process of fitting a line to the dataset.

Single Variable Linear Regression

The Mathematics

The equation of Line is

$$y = m * x + c$$

Where,
 y = dependent variable
 X = independent variable
C = intercept

The algorithm is trying to fit a line to the data by adjusting the values of m and c. Its Objective is to attain to a value of m such that for any given value of x it would be properly predicting the value of y.

There are various ways in which we can attain the values of m and c

1. Statistical approach
2. Iterative approach

Here we are using a scikit learn framework which internally uses iterative approach to attain the linear regression

The Dataset

Dataset consists of two columns namely X and y
Where

For List Price Vs. Best Price for a New GMC Pickup dataset
X = List price (in $1000) for a GMC pickup truck
Y = Best price (in $1000) for a GMC pickup truck
The data is taken from *Consumer's Digest.*

For Fire and Theft in Chicago
X = fires per 100 housing units
Y = thefts per 1000 population within the same Zip code in the Chicago metro area
The data is taken from U.S Commission of Civil Rights.

For Auto Insurance in Sweden dataset
X = number of claims
Y = total payment for all the claims in thousands of Swedish Kronor
The data is taken from Swedish Committee on Analysis of Risk Premium in Motor Insurance.

For Gray Kangaroos dataset
X = nasal length (mm ¥10)
Y = nasal width (mm ¥ 10)
for a male gray kangaroo from a random sample of such animals
The data is taken from Australian *Journal of Zoology*, Vol. 28, p607-613.

Link to All Datasets

The Code

The Code was written in three phases

1. Data preprocessing phase
2. Training

3. Prediction and plotting

The data preprocessing phase

Imports

Numpy import for array processing, python doesn’t have built in array support. The feature of working with native arrays can be used in python with the help of numpy library.

Pandas is a library of python used for working with tables, on importing the data, mostly data will be of table format, for ease manipulation of tables pandas library is imported

Matplotlib is a library of python used to plot graphs, for the purpose of visualizing the results we would be plotting the results with the help of matplotlib library.

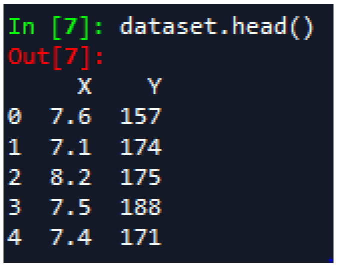
```python
# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Reading the dataset from data

In this line of code using the read_excel method of pandas library, the dataset has been imported from data folder and stored in dataset variable.

```python
# Reading the dataset from data
dataset = pd.read_csv(r'..\\data\\prices.csv')
```

On viewing the dataset, it contains of two columns X and Y where X is dependent variable and Y is Independent Variable.



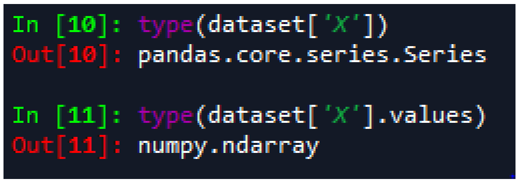
```
In [7]: dataset.head()
Out[7]:
     X    Y
0  7.6  157
1  7.1  174
2  8.2  175
3  7.5  188
4  7.4  171
```

X is an independent variable
Y is dependent variable Inference
For x-value of 7.6 ,157 y-value
for   x-value of 7.1 ,174 y-value
And goes on

Creating Dependent and Independent variables

The X Column from the dataset is extracted into an X variable of type numpy, similarly the y variable
X is an independent variable
Y is dependent variable Inference

```
# Creating Dependent and Independent variables
X = dataset['X'].values
y = dataset['Y'].values
```



```
In [10]: type(dataset['X'])
Out[10]: pandas.core.series.Series

In [11]: type(dataset['X'].values)
Out[11]: numpy.ndarray
```

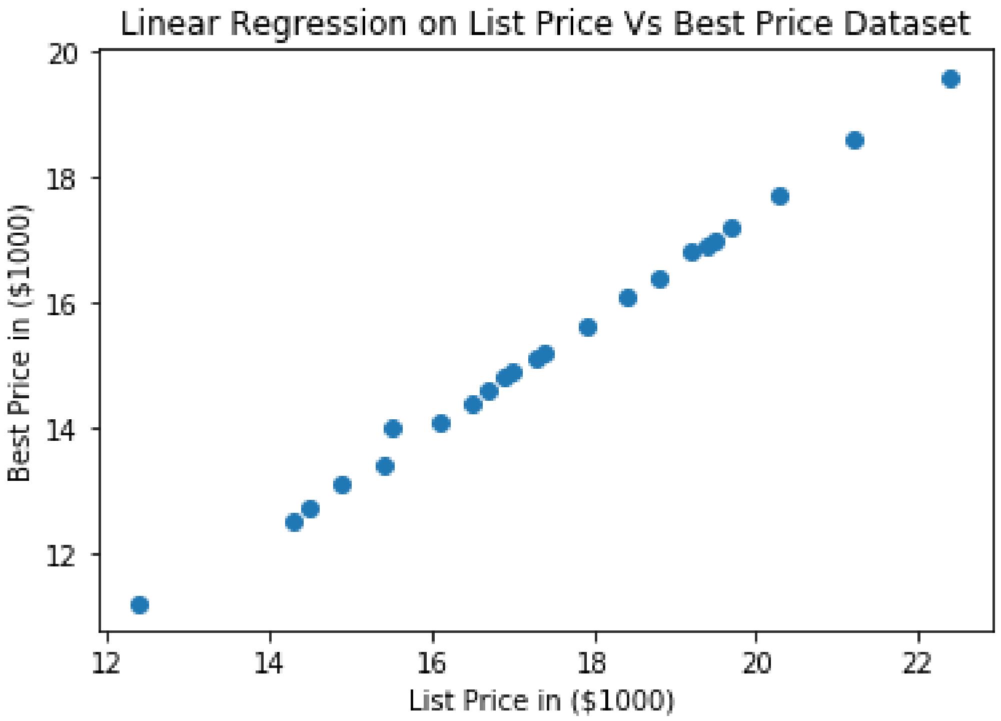
On input 10 it would result in a pandas Series object
So, values attribute is used to attain an numpy array

Visualizing the data

The step is to just see how the dataset is

```
# Visualizing the data
title='Linear Regression on <Dataset>'
x_axis_label = 'X-value < The corresponding attribute of X in dataset >'
y_axis_label = 'y-value < The corresponding attribute of X in dataset >'
plt.scatter(X,y)
plt.title(title)
plt.xlabel(x_axis_label)
plt.ylabel(y_axis_label)
plt.show()
```

On visualization the data would appear something like this



Each point on the plot is a data point showing the respective list price on x-axis and Best Price on y-axis

The X and Y attributes would vary based on dataset.

Splitting the data into training set and test set

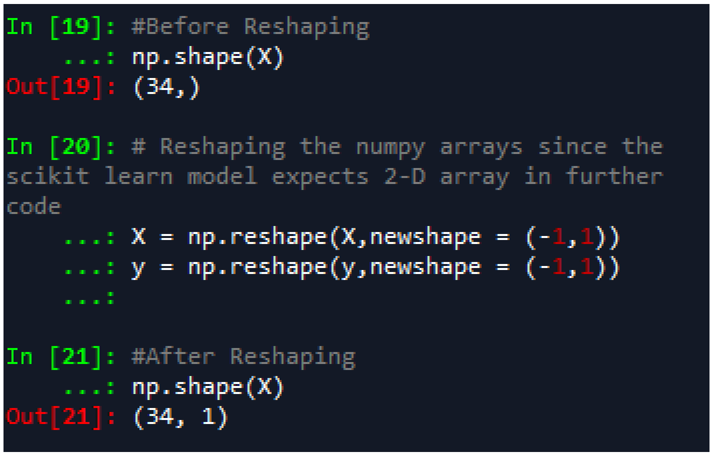
We are splitting the whole dataset into training and test set where training set is used for fitting the line to data and test set is used to check how good the line if for the data.

```
# Splitting the data into training set and test set
from sklearn.model_selection import train_test_split
X_test,X_train,y_test,y_train = train_test_split(X,y, test_size = 0.8)
```

Reshaping the numpy arrays since the scikit learn model expects 2-D array in further code

In further the scikit learn model would be expecting a 2-D array of shape (length,1).

```
# Reshaping the numpy arrays since the scikit learn model expects 2-D array
in further code
X_train = np.reshape(X_train,newshape = (-1,1))
y_train = np.reshape(y_train,newshape = (-1,1))
X_test = np.reshape(X_test,newshape = (-1,1))
y_test = np.reshape(y_test,newshape = (-1,1))
```



```
In [19]: #Before Reshaping
    ...: np.shape(X)
Out[19]: (34,)

In [20]: # Reshaping the numpy arrays since the
scikit learn model expects 2-D array in further
code
    ...: X = np.reshape(X,newshape = (-1,1))
    ...: y = np.reshape(y,newshape = (-1,1))
    ...:

In [21]: #After Reshaping
    ...: np.shape(X)
Out[21]: (34, 1)
```

The code was just to convert a single dimensional array into a 2-D array where each element is an array

The Training phase

Importing the linear model from sklearn framework

From scikit learn Library LinearRegression is imported. Lr is an object of LinearRegression. The process of training is done in the fit method, our dependent and independent variable are fed into to the fit method in which it would try to fit a line to the data provided.

```
# Importing the linear model from sklearn framework
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X = X_train, y = y_train)
```

The Prediction phase

Predicting the Results

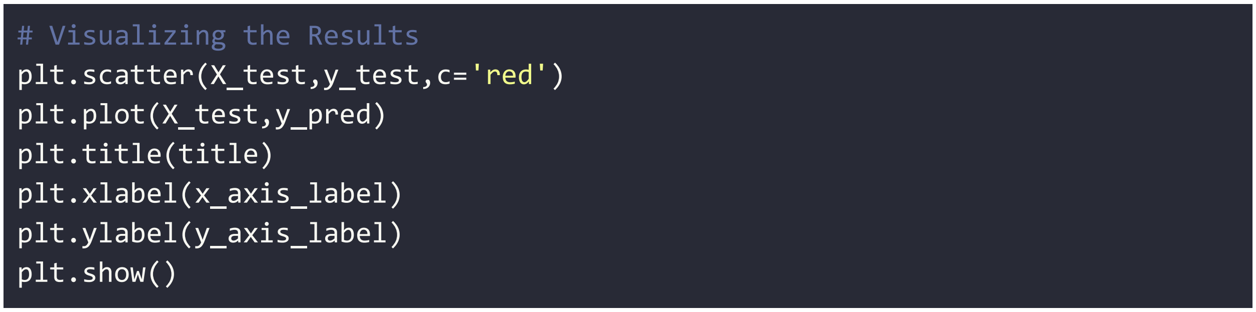
By the trained linear regression model we are trying to predict the values of test data. Y_pred variable contains all the predicted y-values of the test x-values.

```
# Predicting the Results
y_pred = lr.predict(X_test)
```

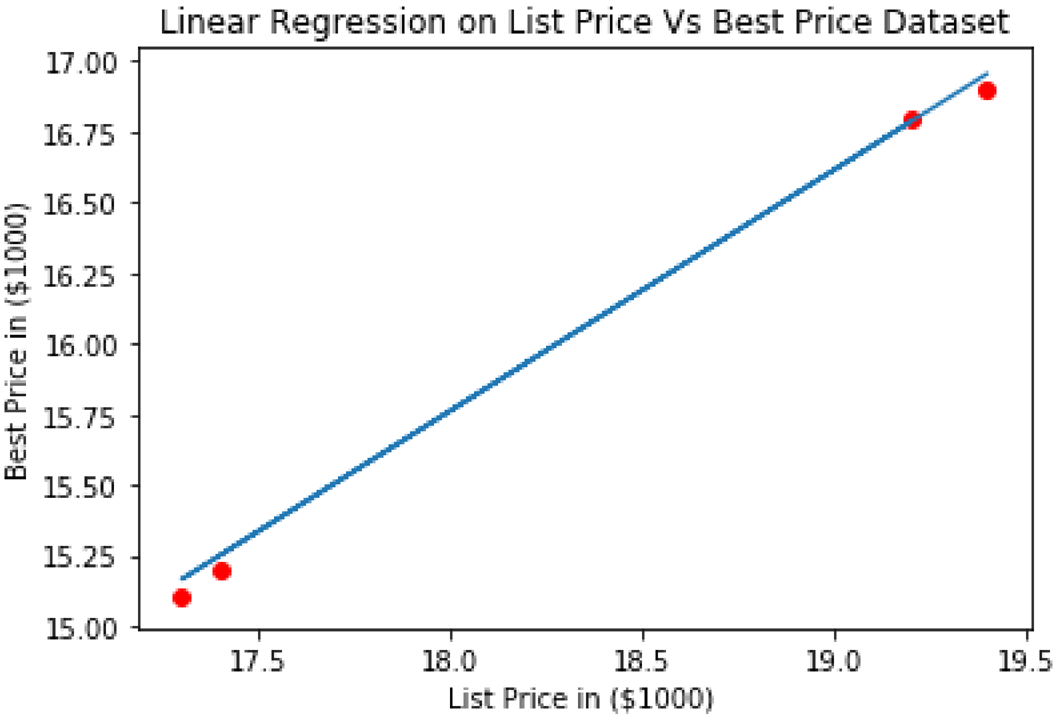
Visualizing the Results

As we have predicted the y-values for a set of x-values we are visualizing the results to check how good did our line fit for our predictions.

The plot shows the red points are the data points are actual values where the blue line is the predictions.

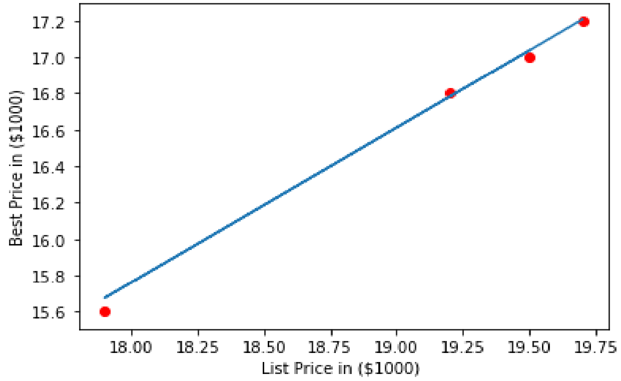


```
# Visualizing the Results
plt.scatter(X_test,y_test,c='red')
plt.plot(X_test,y_pred)
plt.title(title)
plt.xlabel(x_axis_label)
plt.ylabel(y_axis_label)
plt.show()
```

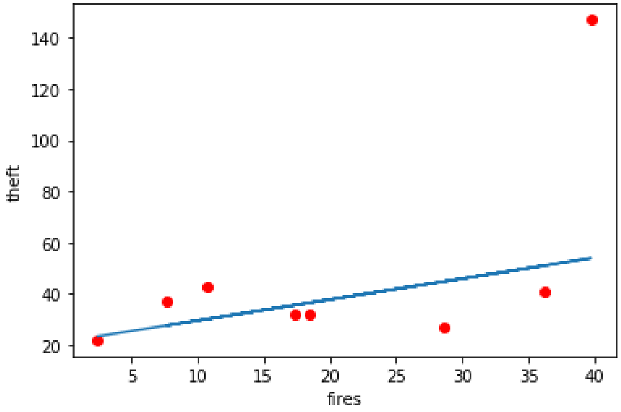


Similarly,
Graph results for all the datasets

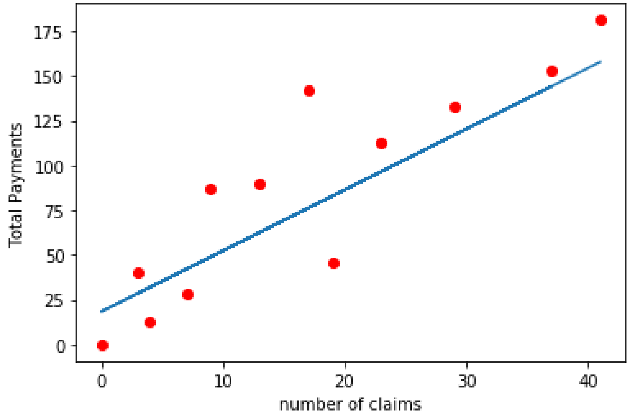
Linear Regression on List Price Vs Best Price Dataset



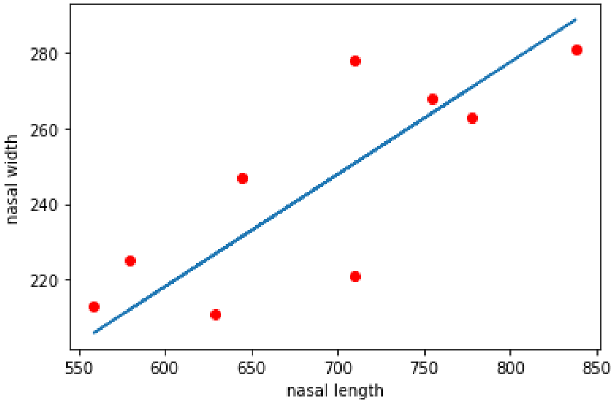
Linear Regression on Fire and Theft in Chicago Dataset



Linear Regression on Auto Insurance in Sweden Dataset



Linear Regression on Gray Kangaroos Dataset



There are certain outliers in Fire Theft graphs. Since, there is random parameter exits the results might not be the same for every execution of code.