Big Basket Ads in App iOS

Date : 9 July 20233

Project Start Date - End Date	● Start Date – 09 -07 -2023 ● End Date – 09 -07- 2023
Objectives	● Write code for calculating accuracy ● Check other variables as independent variables and try to apply regression ● Remove outliers and check the accuracy ● Predict the next 10 values of the given data
Milestones accomplished the week of Start Date - End Date:	● Exploratory analysis ● Linear Regression ● Classification of data with respect to term

Contact Information

This project is performed for educational purposes under the guidance of Siddhivinayak Sir.

Project Manager
Name : Siddhivinayak Phulwadkar
Mobile: 9028965955
Email:
siddhivinayakphulwadkar@gmail.com

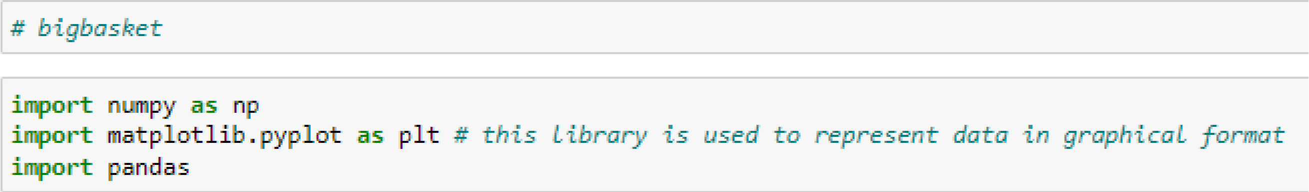
Student Name
Name : Sugumar
Mobile:9342613454
Email: sugumarvsb@gmail.com

Project Abstract

The dataset is about showing advertisements to App iOS users for product purchases. Our main objectives are to find the accuracy with and without outliers and to predict the next ten values. Also, we need to analyze the regression with different independent Variable. Applying Linear Regression and performed exploratory analysis for this dataset.

Big Basket Ads in App iOS

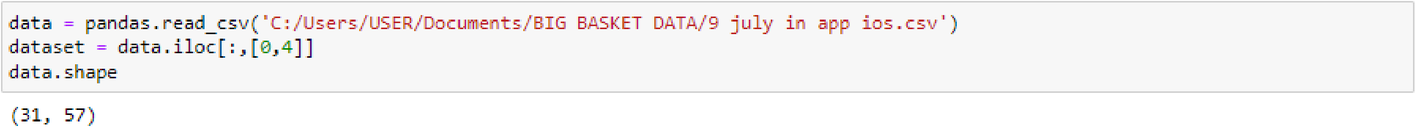
Importing the libraries



```
# bigbasket
```

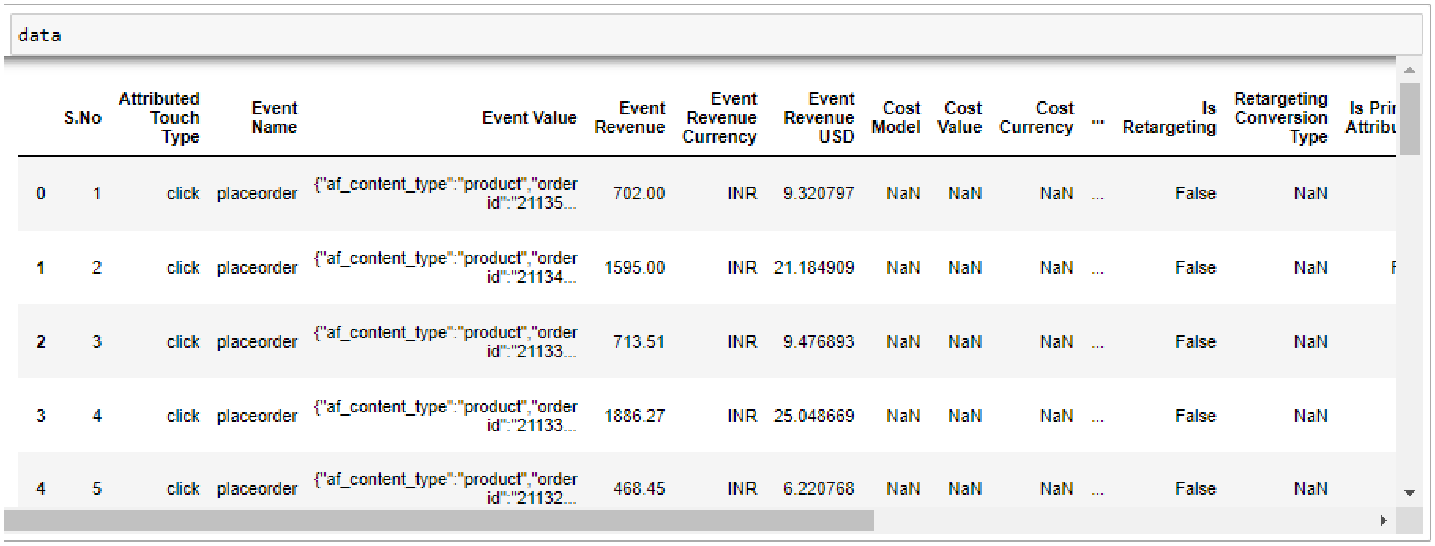
```
import numpy as np
import matplotlib.pyplot as plt # this library is used to represent data in graphical format
import pandas
```

Importing the dataset



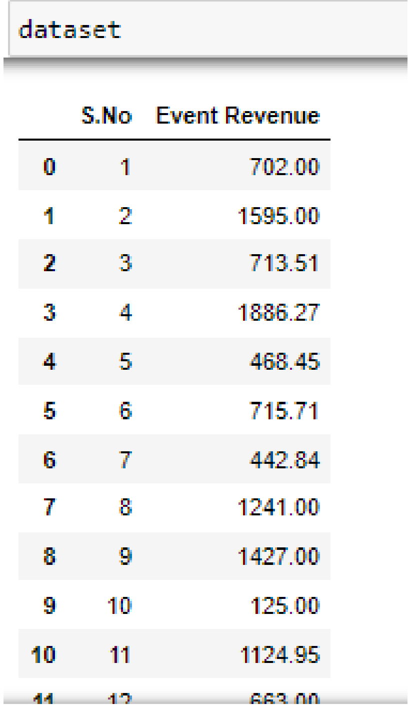
```
data = pandas.read_csv('C:/Users/USER/Documents/BIG BASKET DATA/9 july in app ios.csv')
dataset = data.iloc[:,[0,4]]
data.shape
```

```
(31, 57)
```



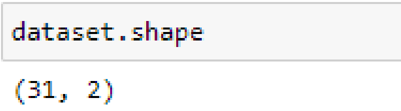
```
data
```

	S.No	Attributed Touch Type	Event Name	Event Value	Event Revenue	Event Revenue Currency	Event Revenue USD	Cost Model	Cost Value	Cost Currency	...	Is Retargeting	Retargeting Conversion Type	Is Pri Attribu
0	1	click	placeorder	{"af_content_type":"product","order id":"21135...	702.00	INR	9.320797	NaN	NaN	NaN	...	False	NaN	
1	2	click	placeorder	{"af_content_type":"product","order id":"21134...	1595.00	INR	21.184909	NaN	NaN	NaN	...	False	NaN	F
2	3	click	placeorder	{"af_content_type":"product","order id":"21133...	713.51	INR	9.476893	NaN	NaN	NaN	...	False	NaN	
3	4	click	placeorder	{"af_content_type":"product","order id":"21133...	1886.27	INR	25.048669	NaN	NaN	NaN	...	False	NaN	
4	5	click	placeorder	{"af_content_type":"product","order id":"21132...	468.45	INR	6.220768	NaN	NaN	NaN	...	False	NaN	



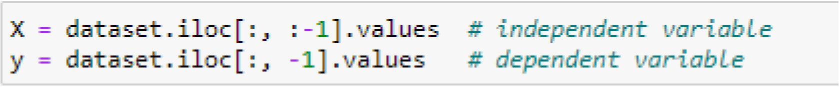
```
dataset
```

	S.No	Event Revenue
0	1	702.00
1	2	1595.00
2	3	713.51
3	4	1886.27
4	5	468.45
5	6	715.71
6	7	442.84
7	8	1241.00
8	9	1427.00
9	10	125.00
10	11	1124.95
11	12	663.00

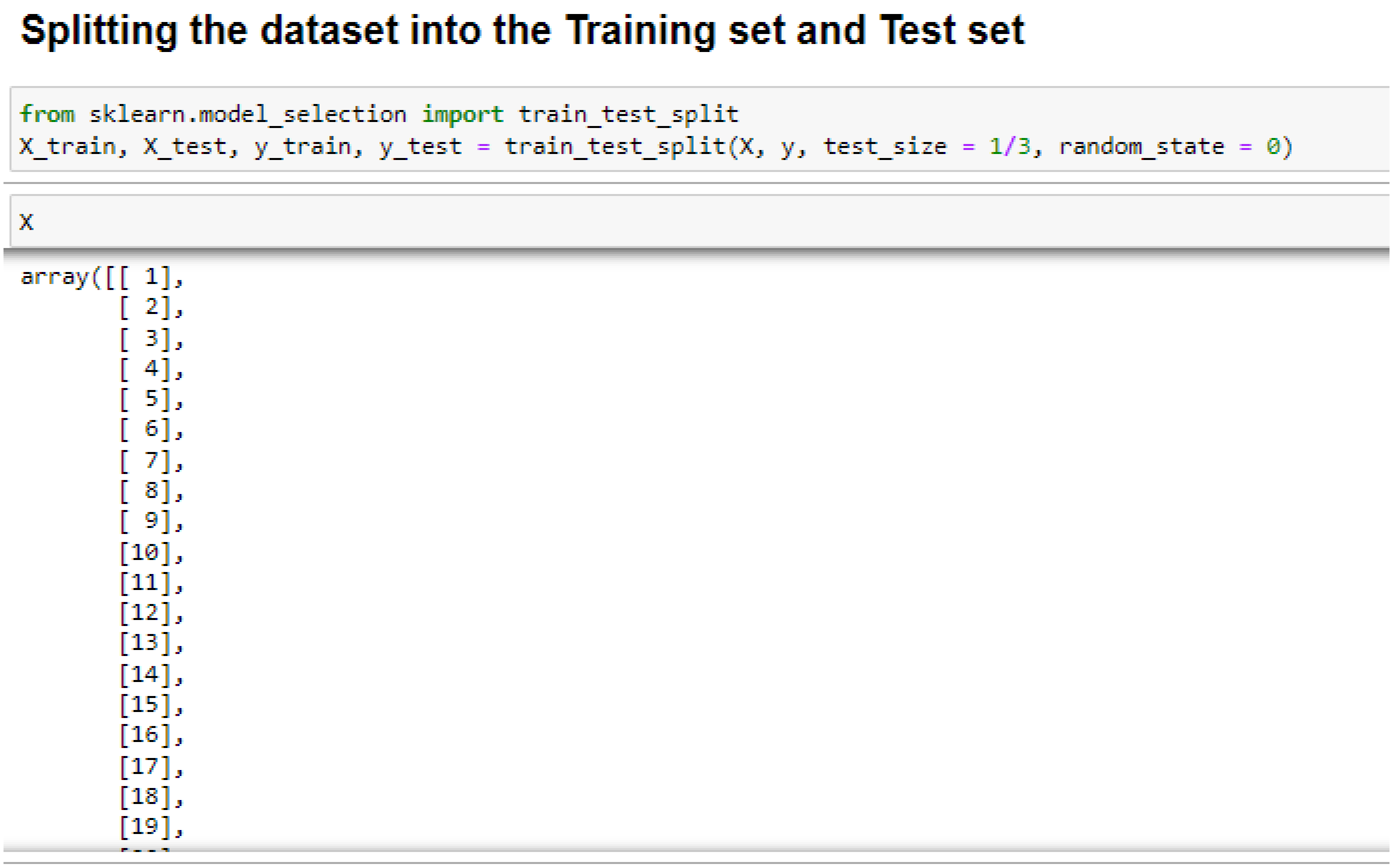


```
dataset.shape
```

```
(31, 2)
```



```
X = dataset.iloc[:, :-1].values  # independent variable
y = dataset.iloc[:, -1].values   # dependent variable
```

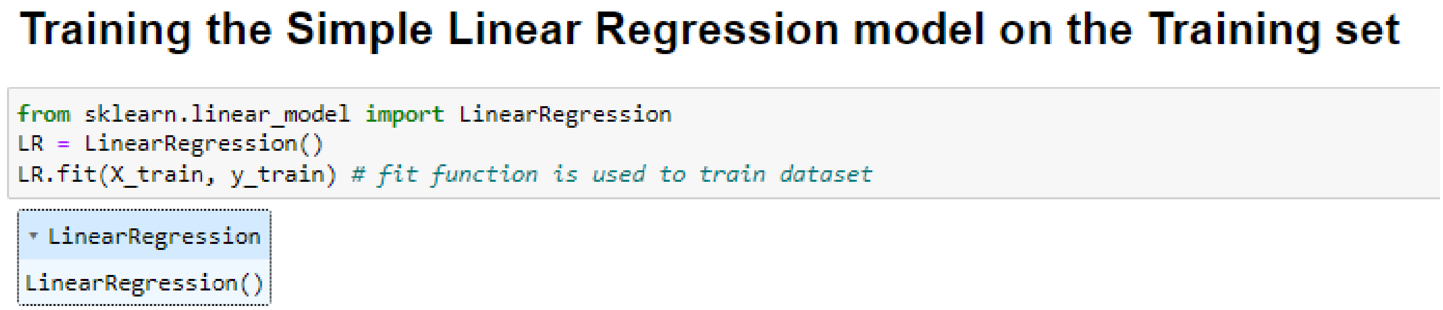


Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

```
X
```

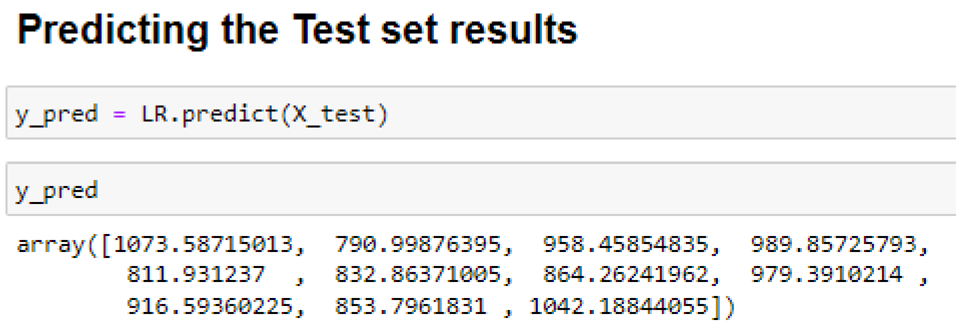
```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12],
       [13],
       [14],
       [15],
       [16],
       [17],
       [18],
       [19],
```



Training the Simple Linear Regression model on the Training set

```
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
LR.fit(X_train, y_train) # fit function is used to train dataset
```

```
LinearRegression()
```



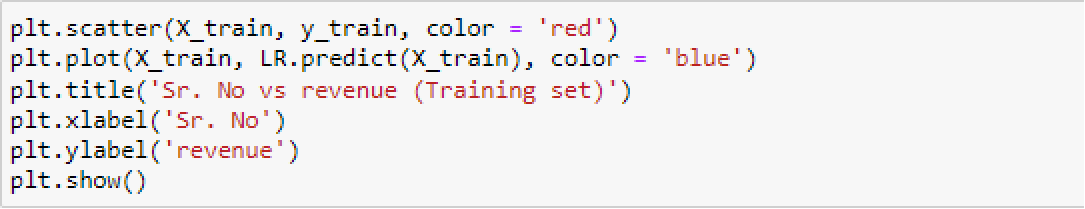
Predicting the Test set results

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

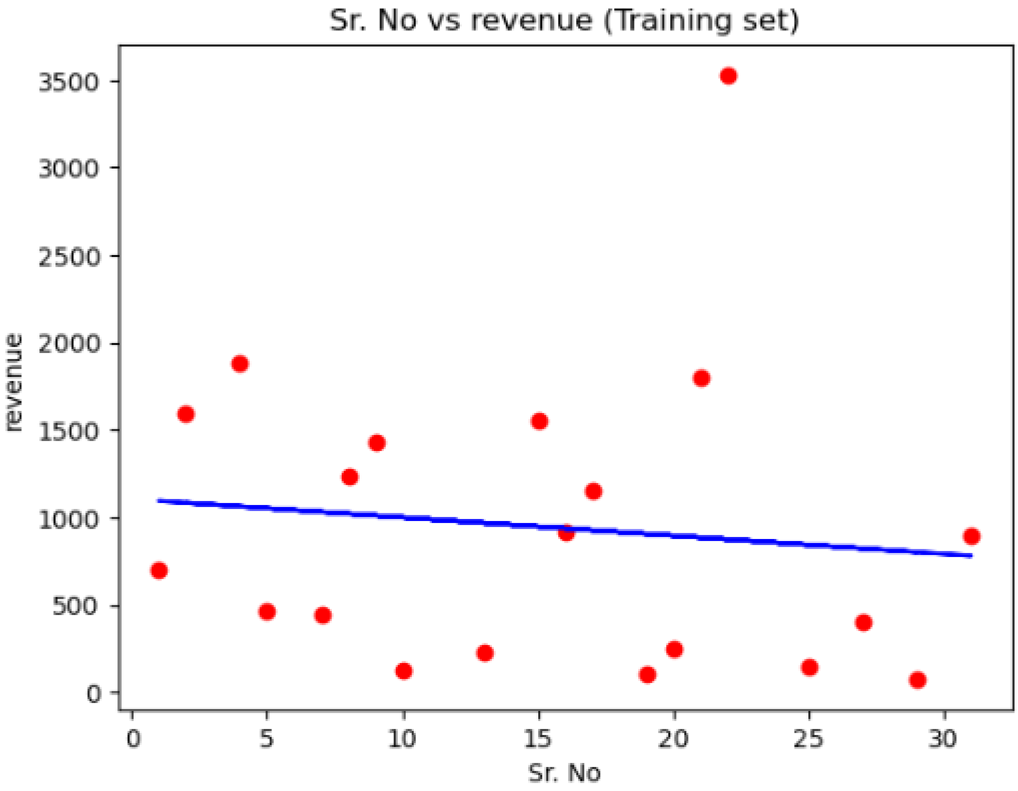
```
y_pred
```

```
array([1073.58715013,  790.99876395,  958.45854835,  989.85725793,
        811.931237  ,  832.86371005,  864.26241962,  979.3910214 ,
        916.59360225,  853.7961831 , 1042.18844055])
```

Visualising the Training set results



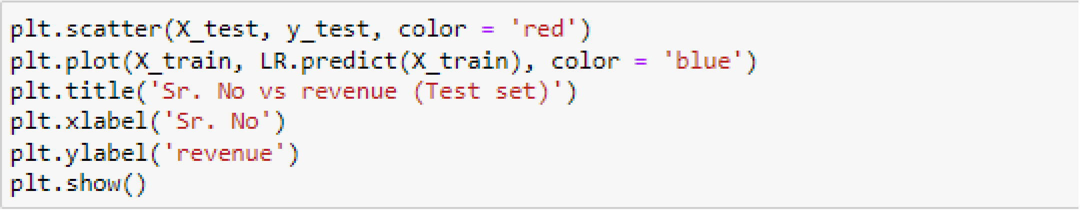
```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, LR.predict(X_train), color = 'blue')
plt.title('Sr. No vs revenue (Training set)')
plt.xlabel('Sr. No')
plt.ylabel('revenue')
plt.show()
```



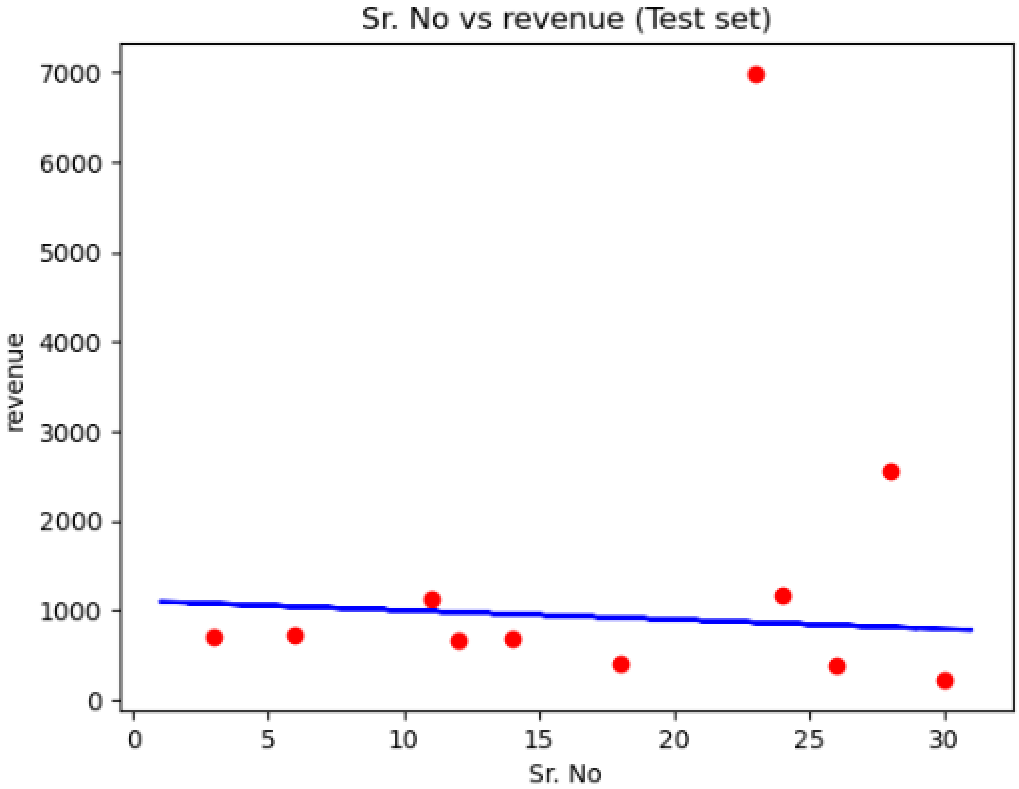
Insights for the Training dataset

- **Some points** are closer and **more** points are away from the best fit line.
- This arises used the **error** in the **regression**. As a result of this regression, the **accuracy level** became **very poor** to find the expected value for the next values.
- we **can't find** the expected revenue value of the **next values**.
- The accuracy level is low because of **imbalanced data**.

Visualising the Test set results

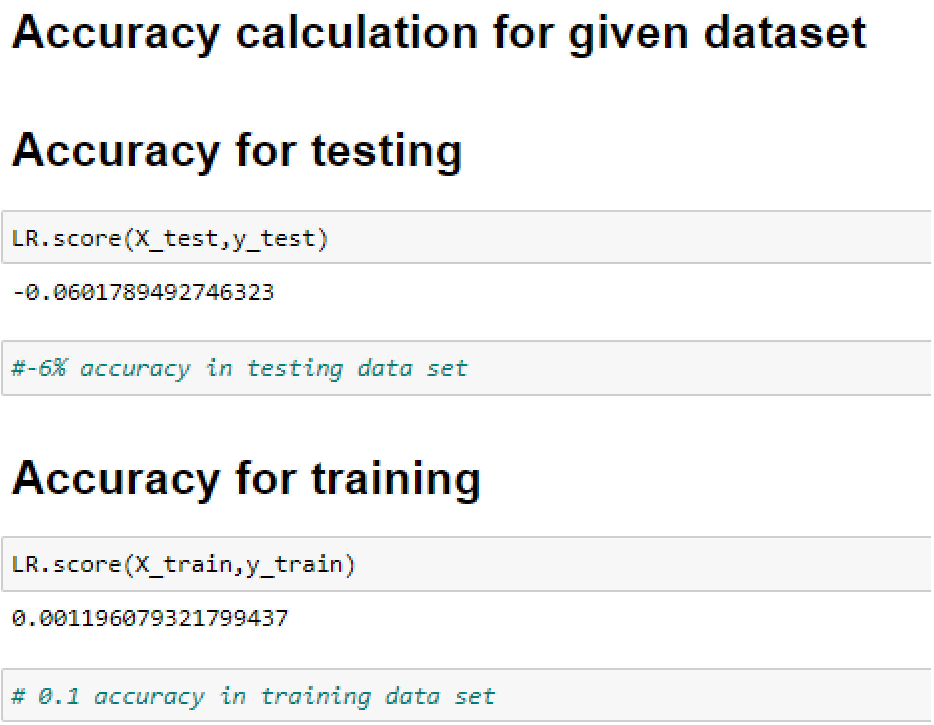


```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, LR.predict(X_train), color = 'blue')
plt.title('Sr. No vs revenue (Test set)')
plt.xlabel('Sr. No')
plt.ylabel('revenue')
plt.show()
```



Insights for the Testing dataset

- The Testing dataset is **better** in **accuracy** by comparing testing and training dataset. But not good for this regression.
- This arises the **error** in the **regression**. As a result of this regression, the accuracy level is very poor to find the expected value for the next values.
- At this level of accuracy, we can't find the expected revenue value of the **next values**.
- The accuracy level is **low** because of **imbalanced data**.



Accuracy calculation for given dataset

Accuracy for testing

```
LR.score(X_test,y_test)
```

```
-0.0601789492746323
```

```
#-6% accuracy in testing data set
```

Accuracy for training

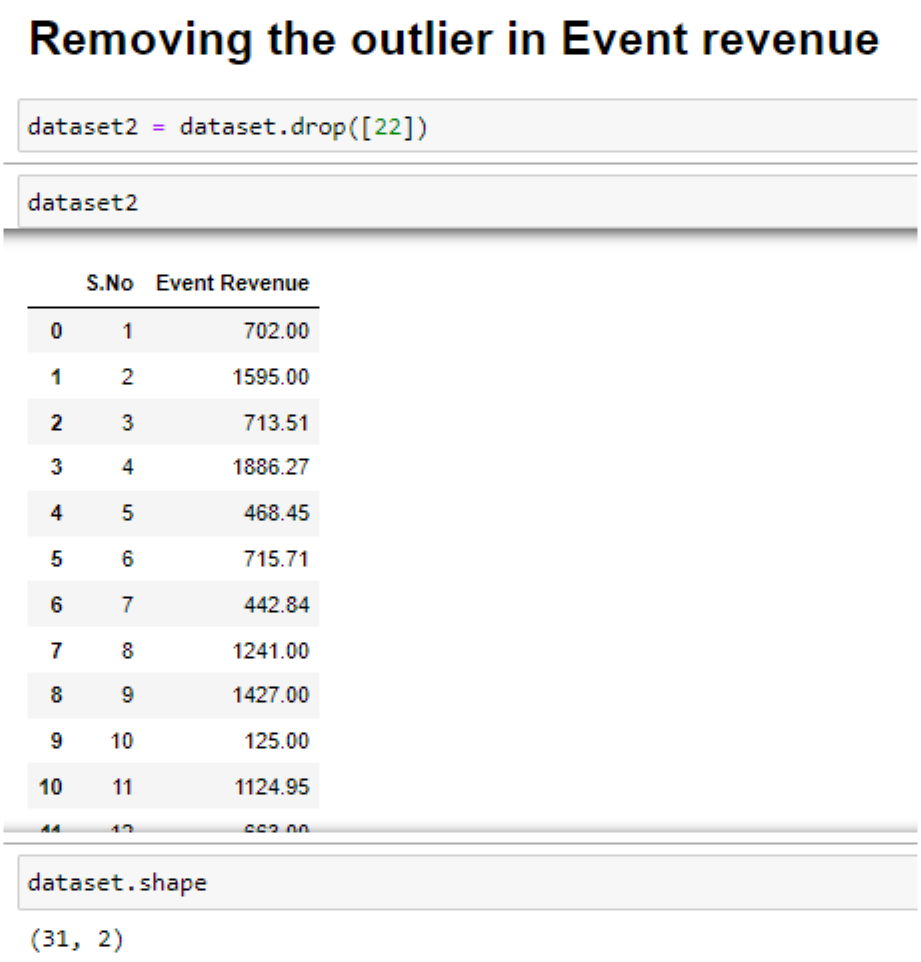
```
LR.score(X_train,y_train)
```

```
0.001196079321799437
```

```
# 0.1 accuracy in training data set
```

Insights

- At the time of the **accuracy calculation**, training is **better** than testing but both the accuracies are **not** good for this regression.
- It can be overcome by sufficient datasets and **unbiased** featured datasets
- As a result of this regression, we can't find the next values with **better accuracy**.



Removing the outlier in Event revenue

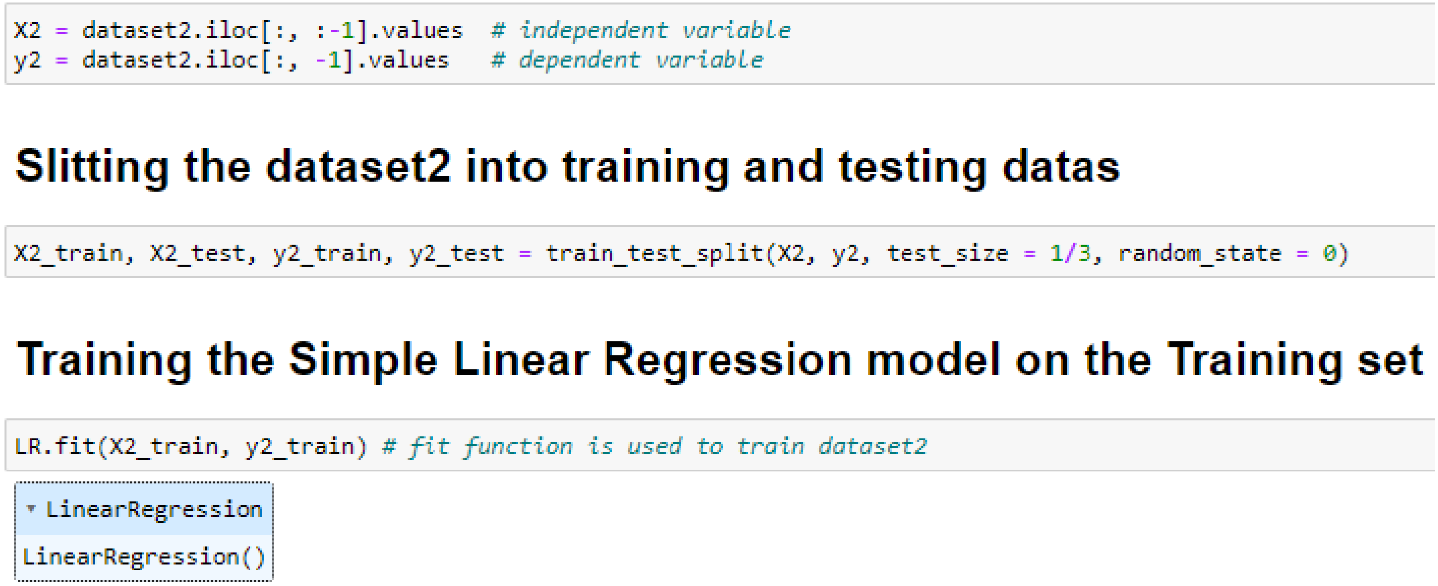
```
dataset2 = dataset.drop([22])
```

```
dataset2
```

	S.No	Event Revenue
0	1	702.00
1	2	1595.00
2	3	713.51
3	4	1886.27
4	5	468.45
5	6	715.71
6	7	442.84
7	8	1241.00
8	9	1427.00
9	10	125.00
10	11	1124.95
11	12	663.00

```
dataset.shape
```

```
(31, 2)
```



```
X2 = dataset2.iloc[:, :-1].values  # independent variable
y2 = dataset2.iloc[:, -1].values   # dependent variable
```

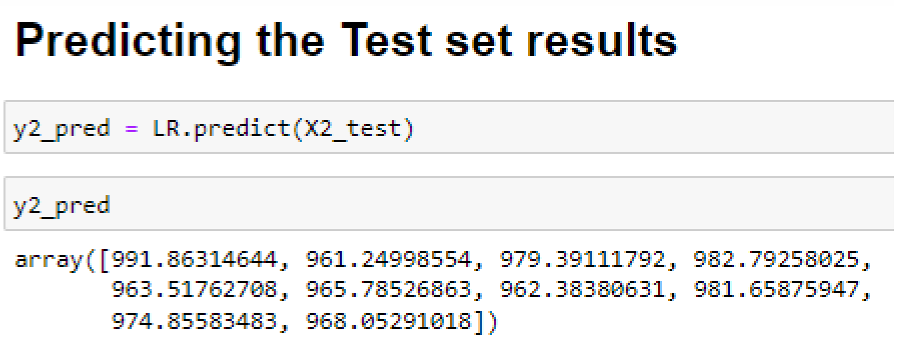
Slitting the dataset2 into training and testing datas

```
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size = 1/3, random_state = 0)
```

Training the Simple Linear Regression model on the Training set

```
LR.fit(X2_train, y2_train) # fit function is used to train dataset2
```

```
▾ LinearRegression
LinearRegression()
```



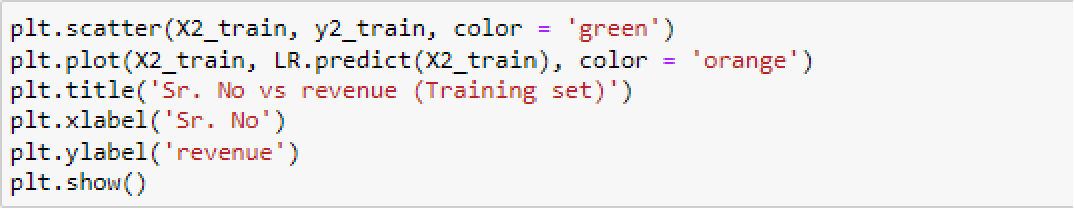
Predicting the Test set results

```
y2_pred = LR.predict(X2_test)
```

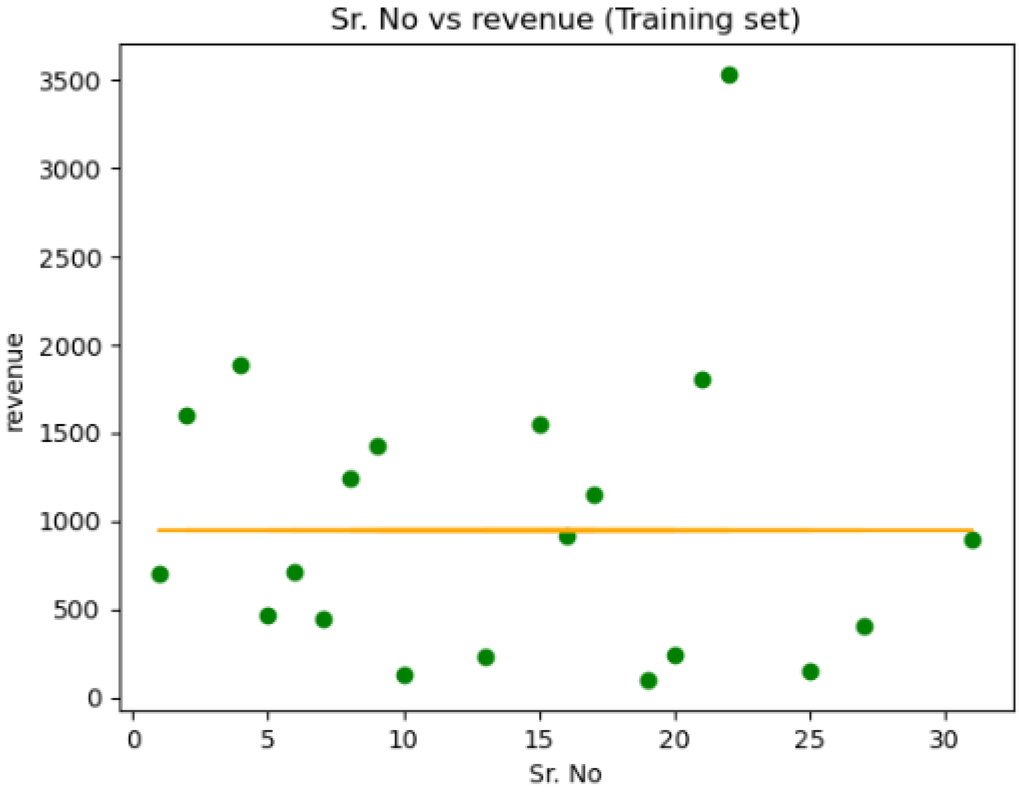
```
y2_pred
```

```
array([991.86314644, 961.24998554, 979.39111792, 982.79258025,
       963.51762708, 965.78526863, 962.38380631, 981.65875947,
       974.85583483, 968.05291018])
```

Visualising the Training set results



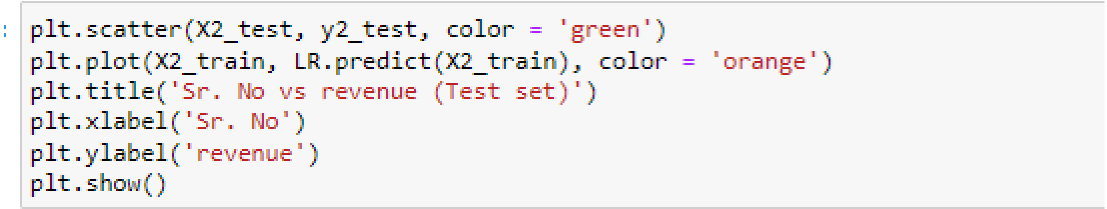
```
plt.scatter(X2_train, y2_train, color = 'green')
plt.plot(X2_train, LR.predict(X2_train), color = 'orange')
plt.title('Sr. No vs revenue (Training set)')
plt.xlabel('Sr. No')
plt.ylabel('revenue')
plt.show()
```



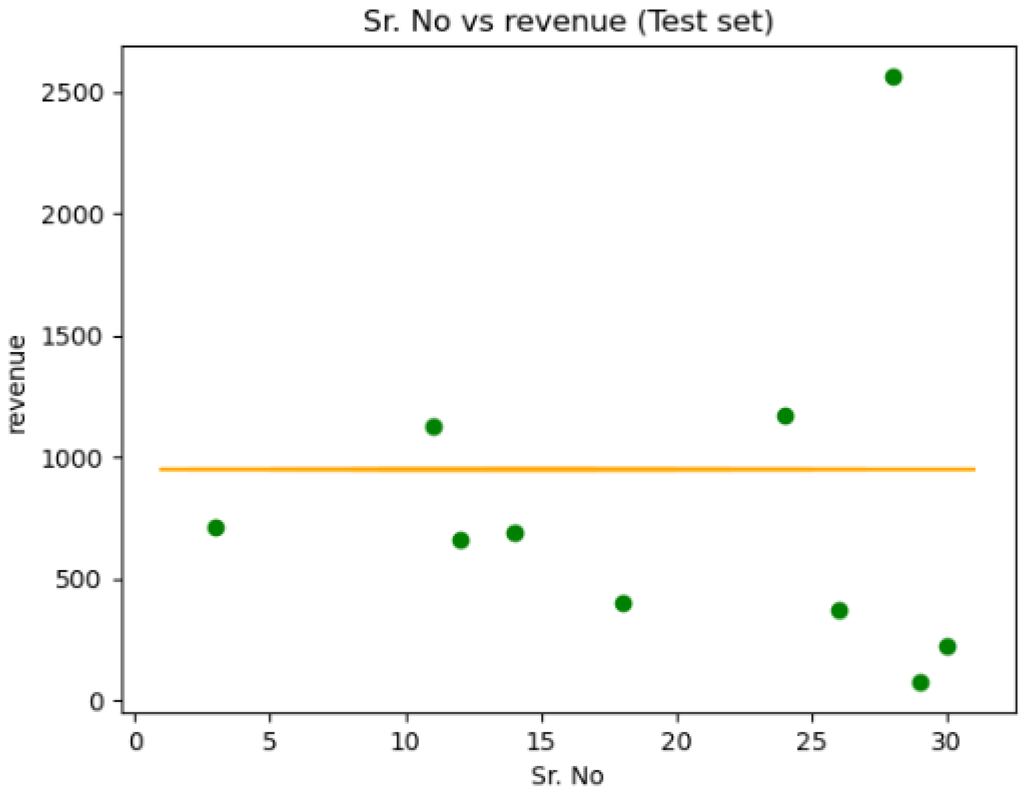
Insights for the Training dataset

- **Some** points are closer and **more** points are away from the **Best-fit line.**
- This arises the error in the regression. As a result of this regression, the accuracy level became very poor to find the expected value for the next values.
- At this level of accuracy, we can’t find the expected revenue value of the next value.
- It is because of insufficient datasets and biased data features

Visualising the Test set results

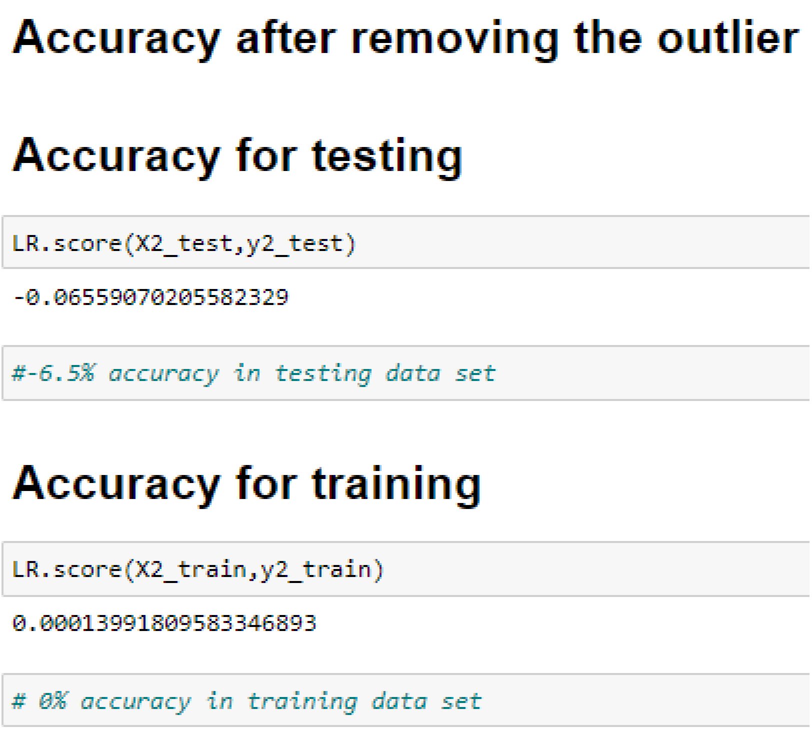


```
plt.scatter(X2_test, y2_test, color = 'green')
plt.plot(X2_train, LR.predict(X2_train), color = 'orange')
plt.title('Sr. No vs revenue (Test set)')
plt.xlabel('Sr. No')
plt.ylabel('revenue')
plt.show()
```



Insights for the Testing dataset

- The accuracy level is low because of insufficient data.
- The testing dataset is better by comparing training dataset and testing dataset, but **not good** for this regression.
- It arises the error in the regression. As a result of this regression, the accuracy level
- It became very poor to find the expected value for the next values.
- At this level of accuracy, we can't find the expected revenue value of the next values.



Accuracy after removing the outlier

Accuracy for testing

```
LR.score(X2_test,y2_test)
```

```
-0.06559070205582329
```

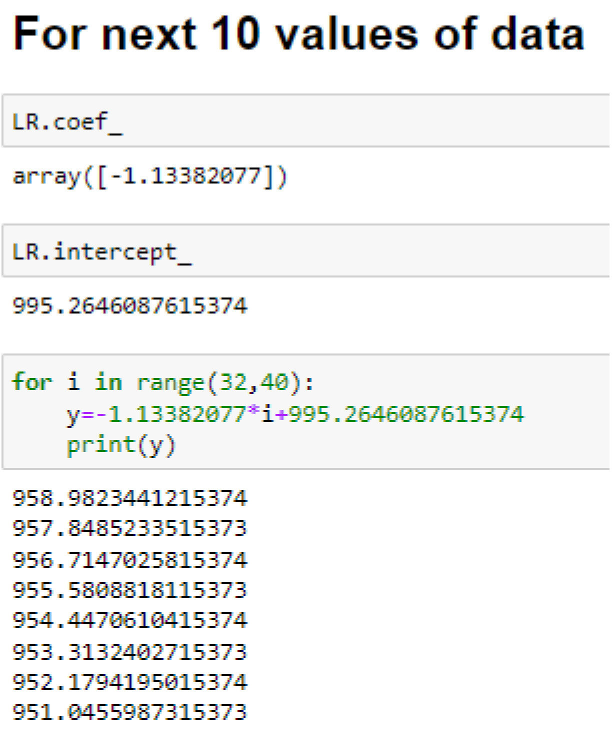
```
#-6.5% accuracy in testing data set
```

Accuracy for training

```
LR.score(X2_train,y2_train)
```

```
0.00013991809583346893
```

```
# 0% accuracy in training data set
```



For next 10 values of data

```
LR.coef_
```

```
array([-1.13382077])
```

```
LR.intercept_
```

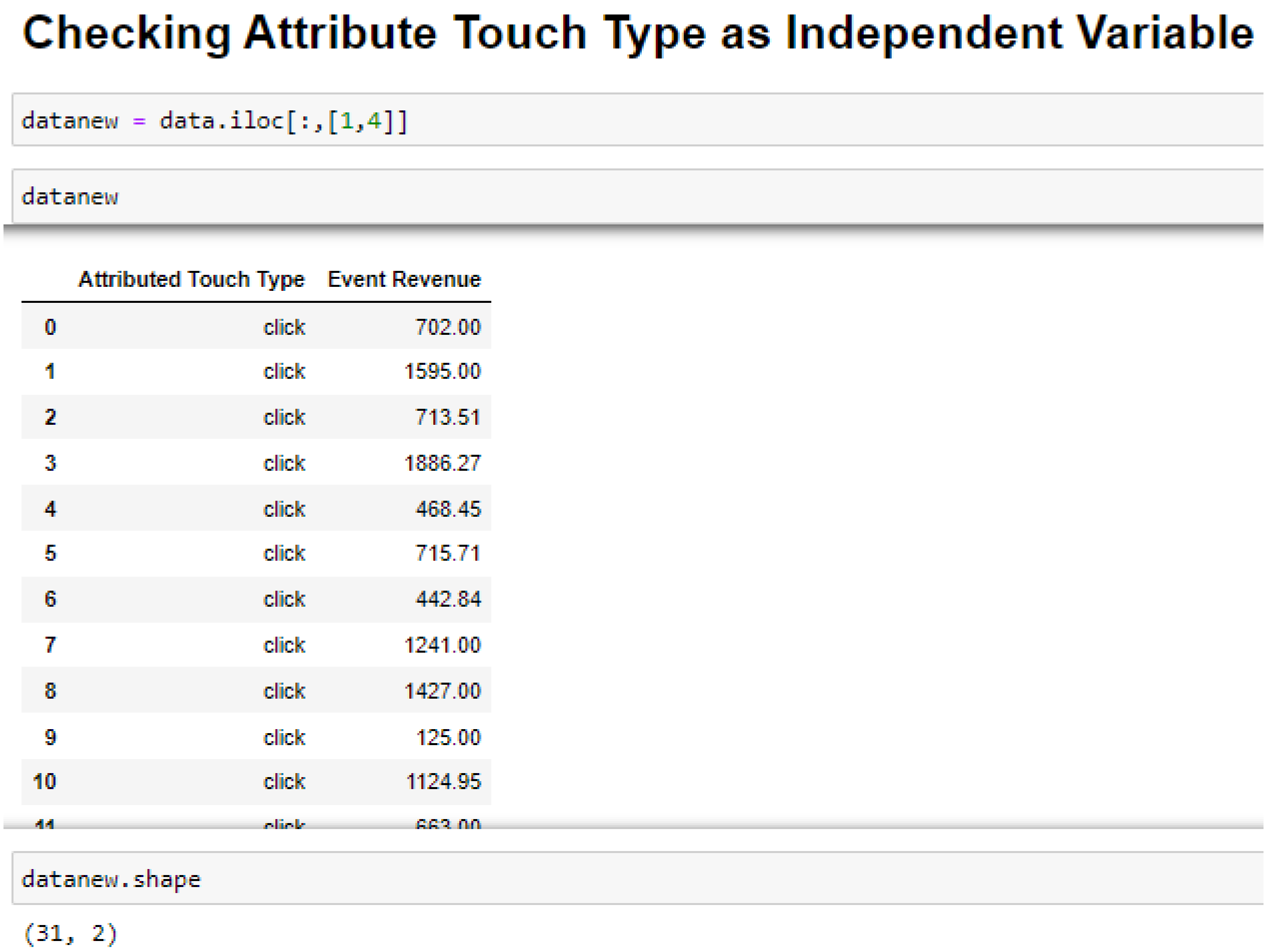
```
995.2646087615374
```

```
for i in range(32,40):
    y=-1.13382077*i+995.2646087615374
    print(y)
```

```
958.9823441215374
957.8485233515373
956.7147025815374
955.5808818115373
954.4470610415374
953.3132402715373
952.1794195015374
951.0455987315373
```

Insights

- At the time of accuracy calculation, training is better than testing but both the accuracies are not good for this regression. As a result, inaccuracy datasets are produced



Checking Attribute Touch Type as Independent Variable

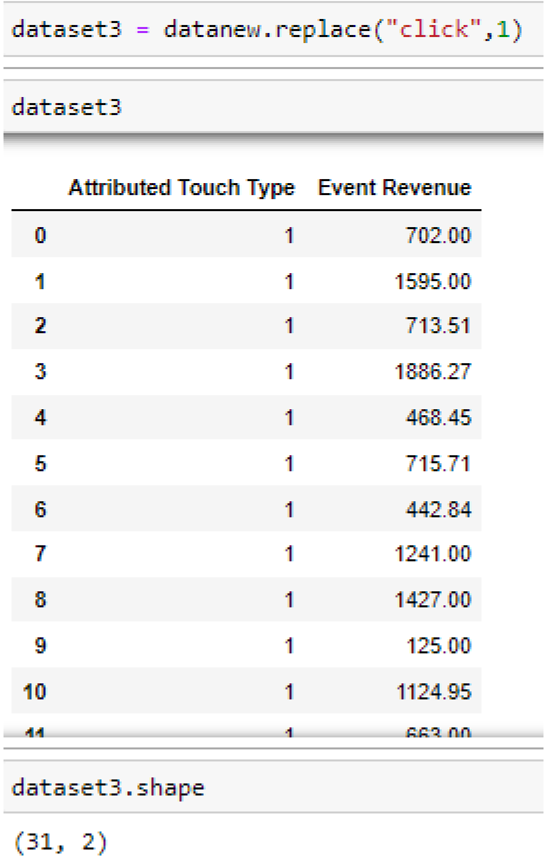
```
datanew = data.iloc[:,[1,4]]
```

```
datanew
```

	Attributed Touch Type	Event Revenue
0	click	702.00
1	click	1595.00
2	click	713.51
3	click	1886.27
4	click	468.45
5	click	715.71
6	click	442.84
7	click	1241.00
8	click	1427.00
9	click	125.00
10	click	1124.95
11	click	663.00

```
datanew.shape
```

```
(31, 2)
```



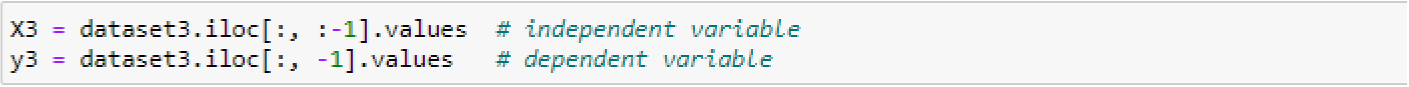
```
dataset3 = datanew.replace("click",1)
```

```
dataset3
```

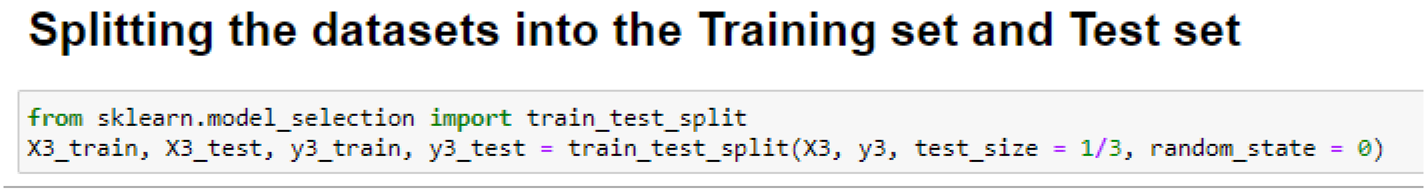
	Attributed Touch Type	Event Revenue
0	1	702.00
1	1	1595.00
2	1	713.51
3	1	1886.27
4	1	468.45
5	1	715.71
6	1	442.84
7	1	1241.00
8	1	1427.00
9	1	125.00
10	1	1124.95
11	1	663.00

```
dataset3.shape
```

```
(31, 2)
```

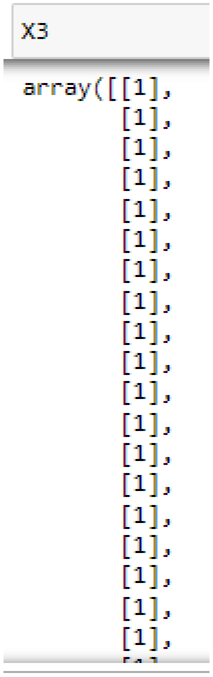


```
X3 = dataset3.iloc[:, :-1].values  # independent variable
y3 = dataset3.iloc[:, -1].values   # dependent variable
```



Splitting the datasets into the Training set and Test set

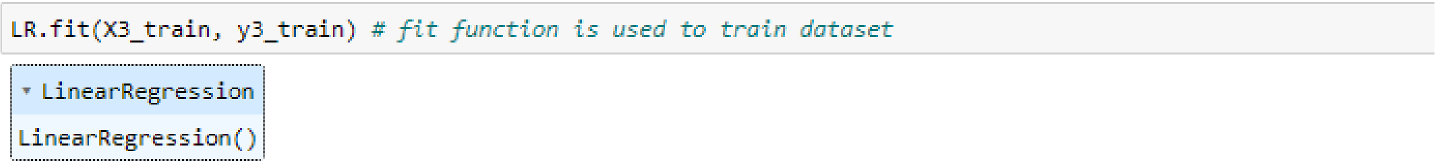
```
from sklearn.model_selection import train_test_split
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size = 1/3, random_state = 0)
```



```
X3
```

```
array([[1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
```

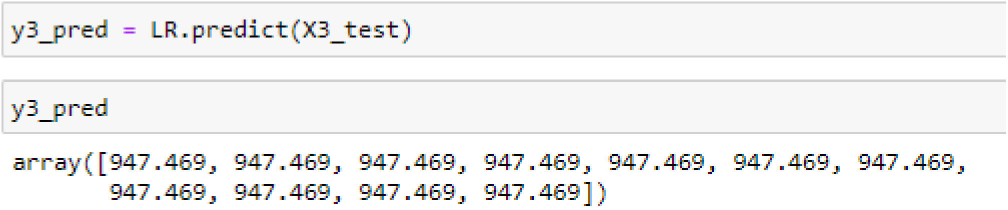
Training the Simple Linear Regression model on the Training set



```
LR.fit(X3_train, y3_train) # fit function is used to train dataset
```

```
▾ LinearRegression
LinearRegression()
```

Predicting the Test set results

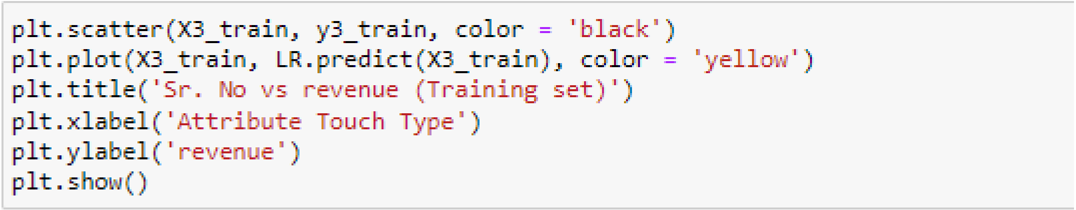


```
y3_pred = LR.predict(X3_test)
```

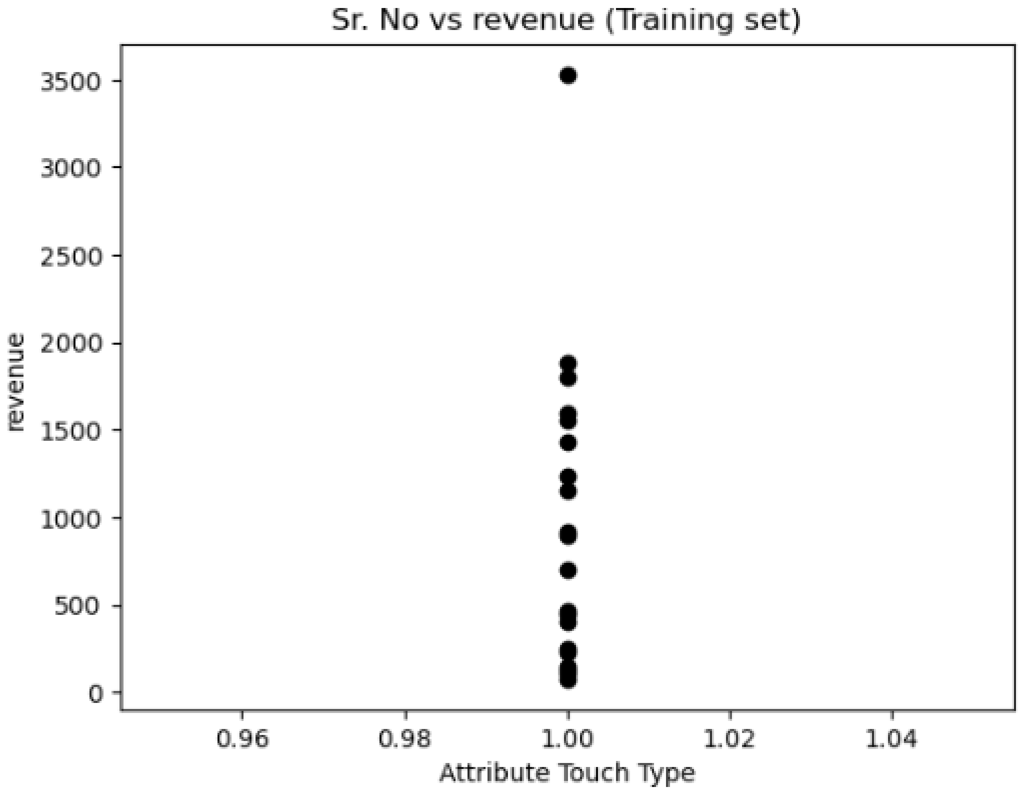
```
y3_pred
```

```
array([947.469, 947.469, 947.469, 947.469, 947.469, 947.469, 947.469,
       947.469, 947.469, 947.469, 947.469])
```

Visualising the Training set results



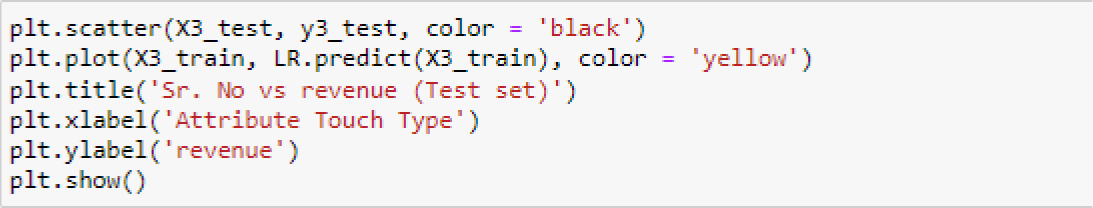
```
plt.scatter(X3_train, y3_train, color = 'black')
plt.plot(X3_train, LR.predict(X3_train), color = 'yellow')
plt.title('Sr. No vs revenue (Training set)')
plt.xlabel('Attribute Touch Type')
plt.ylabel('revenue')
plt.show()
```



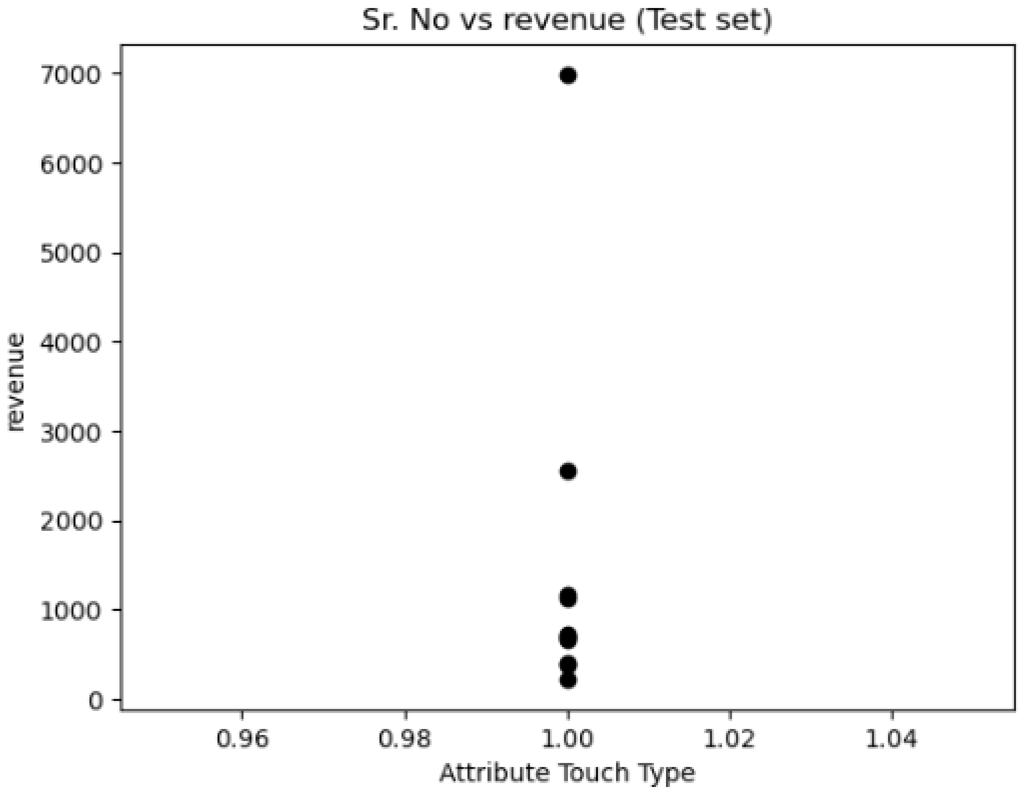
Insights

- There is no use for this graph because the **independent variable** is a biased dataset.
- So, the results are with zero accuracy.
- It can be overcome by unbiased featured datasets.

Visualising the Test set results



```
plt.scatter(X3_test, y3_test, color = 'black')
plt.plot(X3_train, LR.predict(X3_train), color = 'yellow')
plt.title('Sr. No vs revenue (Test set)')
plt.xlabel('Attribute Touch Type')
plt.ylabel('revenue')
plt.show()
```

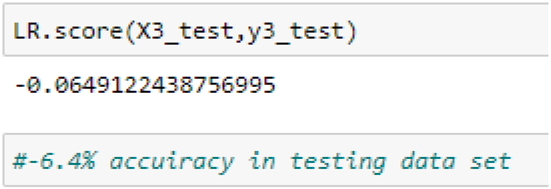


Insights

- There is no use for this graph because the independent variable is a biased dataset.
- So, the results are with zero accuracy.
- This can be overcome by unbiased featured datasets.

Accuracy calculation

For testing datasets

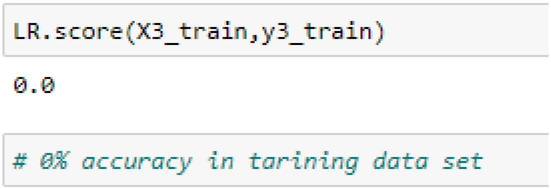


```
LR.score(X3_test,y3_test)
```

```
-0.0649122438756995
```

```
#-6.4% accuiracy in testing data set
```

For training data set



```
LR.score(X3_train,y3_train)
```

```
0.0
```

```
# 0% accuracy in taring data set
```

summary

- Our main objectives are **successfully done** by **Simple Linear Regression.**
- Values are highly deviating from the best-fit line. So, there is highly error on the regression part
- Simple Linear Regression is not given the best accuracy for the given dataset.
- These are because of **insufficient data or biased data features**.
- So, we can't find the near future values with these data.
- **Regression accuracy** is **better** with large amounts of data unlike the given data and **unbiased** featured data.

TheSiddhivinayak