
# Project name

**Date :**

| | |
|---|---|
| **Project Start Date - End Date** | ● Start Date – 07 -06 -2023<br>● End Date – 07 -06 2023 |
| **Objectives** | ● To analyses how many people who clicked on the advertisement enrolled in our course<br>● General exploratory analyses<br>● General descriptive analyses |
| **Milestones accomplished the week of Start Date - End Date:** | ● Descriptive analyses<br>● Exploratory analyses<br>● Classification of data with respect to term |
| | |
| | |
| | |
| | |

## Contact Information

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

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

Signed by:
siddhivinayak P
C40578280437463...

**Student Name**
Name : Vishnu Prabhu C
Mobile: 7397627320
Email: vishnuprabhu633@gmail.com

## Project Abstract

The dataset is about showing the advertisement to customers to enrol in Big Basket. Our main objective was to understand on which time customers are clicked on our ads and enrolled our page in iOS. Problem statement is classify as we are looking for preferred timing in a day where we can do marketing and we will get definitely sales. For this dataset we have applied Linear Regression and performed accuracy ,remove the outliers then check the accuracy ,check other variables as independent variables and try to apply regression and predict the next 10 values of given data.

# Linear Regression Analysis

## Importing libraries

### Importing libararies

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as py
```

In [2]:
```
data = pd.read_csv("9 july in app ios.csv")
data
```

Out[2]:

| | 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 Primar Attributio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | click | placeorder | {"af_content_type":"product","order id":"21135... | 702.00 | INR | 9.320797 | NaN | NaN | NaN | ... | False | NaN | Tru |
| 1 | 2 | click | placeorder | {"af_content_type":"product","order id":"21134... | 1595.00 | INR | 21.184909 | NaN | NaN | NaN | ... | False | NaN | Fals |
| 2 | 3 | click | placeorder | {"af_content_type":"product","order id":"21133... | 713.51 | INR | 9.476893 | NaN | NaN | NaN | ... | False | NaN | Tru |
| 3 | 4 | click | placeorder | {"af_content_type":"product","order id":"21133... | 1886.27 | INR | 25.048669 | NaN | NaN | NaN | ... | False | NaN | Tru |
| 4 | 5 | click | placeorder | {"af_content_type":"product","order id":"21132... | 468.45 | INR | 6.220768 | NaN | NaN | NaN | ... | False | NaN | Tru |

## Importing Dataset

In [3]:
```
dataset = data.iloc[:,[0,4]]
dataset
```

Out[3]:

| | 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 |
| **12** | 13 | 228.66 |
| **13** | 14 | 693.00 |
| **14** | 15 | 1549.91 |
| **15** | 16 | 920.42 |
| **16** | 17 | 1154.52 |
| **17** | 18 | 404.00 |
| **18** | 19 | 100.00 |
| **19** | 20 | 246.60 |
| **20** | 21 | 1804.11 |

# Preprocessing the Dataset

# Defining the Independent and Dependent Variables

In [4]:
```
dataset.shape
```

Out[4]: (31, 2)

In [5]:
```
x=dataset.iloc[:,:-1].values #independent variable
y=dataset.iloc[:,-1].values #dependent variable
```

```
In [6]: x
```

```
Out[6]: array([[ 1],
               [ 2],
               [ 3],
               [ 4],
               [ 5],
               [ 6],
               [ 7],
               [ 8],
               [ 9],
               [10],
               [11],
               [12],
               [13],
               [14],
               [15],
               [16],
               [17],
               [18],
               [19],
               [20],
               [21],
               [22],
               [23],
               [24],
               [25],
               [26],
               [27],
               [28],
               [29],
               [30],
               [31]])
```

```
In [7]: y
```

```
Out[7]: array([ 702.  , 1595.  ,  713.51, 1886.27,  468.45,  715.71,  442.84,
               1241.  , 1427.  ,  125.  , 1124.95,  663.  ,  228.66,  693.  ,
               1549.91,  920.42, 1154.52,  404.  ,  100.  ,  246.6 , 1804.11,
               3530.  , 6990.  , 1174.  ,  149.  ,  374.  ,  407.  , 2565.59,
                 75.  ,  223.  ,  896.6 ])
```

# Linear Regression

## LINEAR REGRESSION

In [8]:
```
import sklearn#library used for ML
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)
```

In [9]:
```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
```

In [10]:
```
LR.fit(x_train,y_train)
```

Out[10]:
```
▾ LinearRegression
LinearRegression()
```

In [11]:
```
y_pred =LR.predict(x_test)
```

In [12]:
```
y_pred
```

Out[12]:
```
array([1073.58715013,  790.99876395,  958.45854835,  989.85725793,
        811.931237  ,  832.86371005,  864.26241962,  979.3910214 ,
        916.59360225,  853.7961831 , 1042.18844055])
```

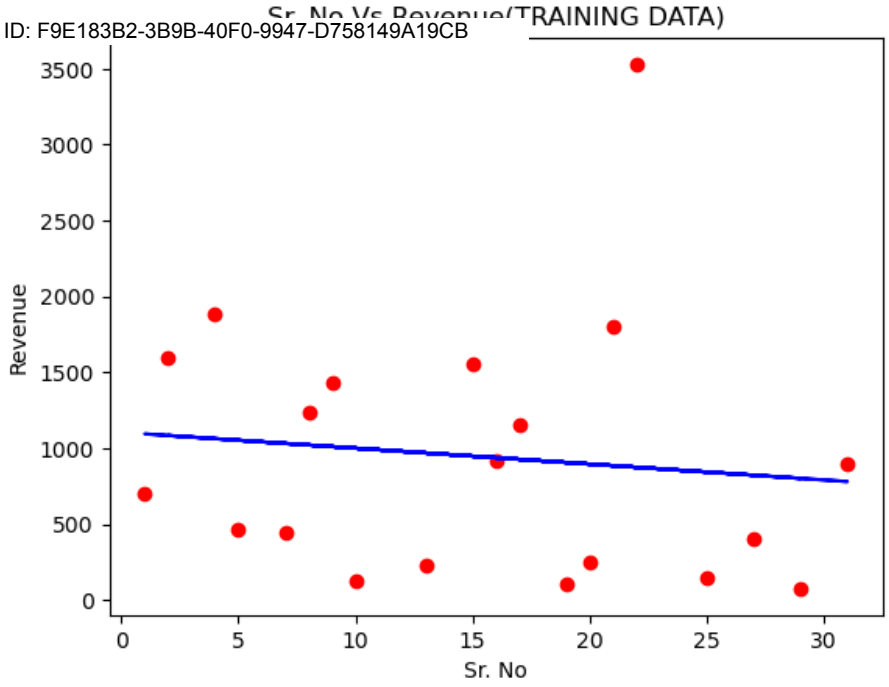
In [13]:
```
y_test
```

Out[13]:
```
array([ 713.51,  223.  ,  693.  , 1124.95, 2565.59,  374.  , 6990.  ,
        663.  ,  404.  , 1174.  ,  715.71])
```

# Plotting the training set

## PLOTTING TRAINING SET

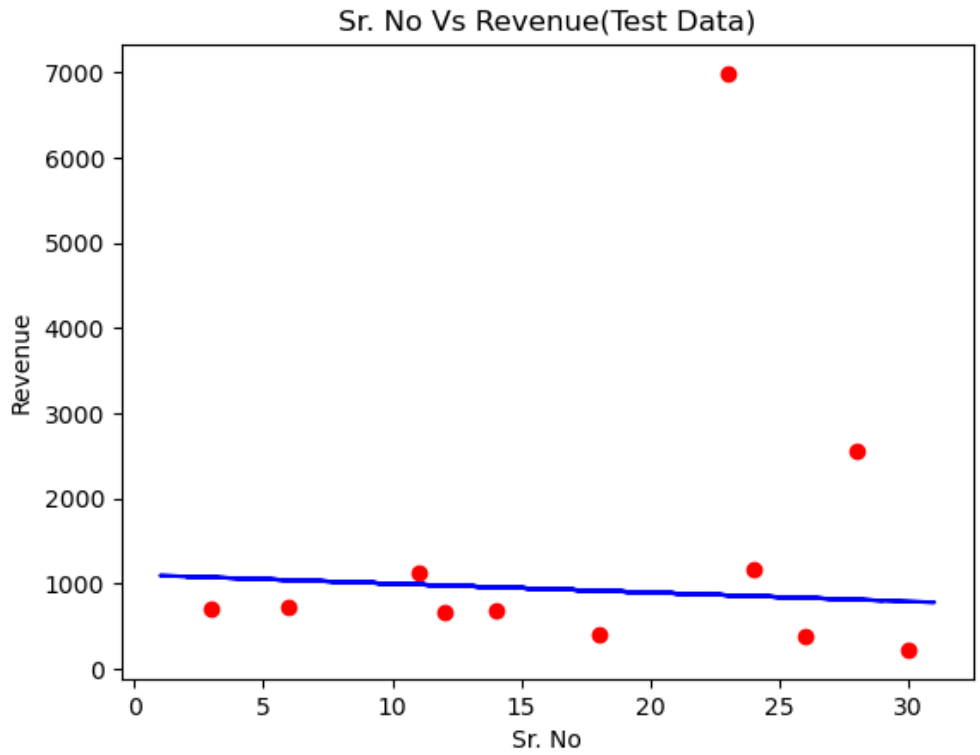
In [14]:
```
plt.scatter(x_train,y_train, color='red')
plt.plot(x_train,LR.predict(x_train), color ='blue' )
plt.title("Sr. No Vs Revenue(TRAINING DATA)")
plt.xlabel("Sr. No")
plt.ylabel("Revenue")
plt.show()
```




## Plotting the test set

## PLOTTING TEST SET

In [15]:
```
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 Data)")
plt.xlabel("Sr. No")
plt.ylabel("Revenue")
plt.show()
```



# Checking the Accuracy

## calculating accuracy

In [21]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [22]:
```
# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared (R2):", r2)
```

```
Mean Squared Error (MSE): 3811227.858257694
Root Mean Squared Error (RMSE): 1952.2366296783016
Mean Absolute Error (MAE): 1012.9594671851294
R-squared (R2): -0.09917975370513132
```

# Check other variables as independent variables and try to apply

In [26]:
```
dataset1 = data.iloc[:,[0,6]]
dataset1
```

Out[26]:

| | S.No | Event Revenue USD |
|---|---|---|
| 0 | 1 | 9.320797 |
| 1 | 2 | 21.184909 |
| 2 | 3 | 9.476893 |
| 3 | 4 | 25.048669 |
| 4 | 5 | 6.220768 |
| 5 | 6 | 9.499690 |
| 6 | 7 | 5.877149 |
| 7 | 8 | 16.451533 |
| 8 | 9 | 18.946547 |
| 9 | 10 | 1.656824 |
| 10 | 11 | 14.910755 |
| 11 | 12 | 8.790976 |
| 12 | 13 | 3.033256 |
| 13 | 14 | 9.198985 |
| 14 | 15 | 20.573736 |
| 15 | 16 | 12.217792 |
| 16 | 17 | 15.315411 |
| 17 | 18 | 5.358844 |
| 18 | 19 | 1.326601 |
| 19 | 20 | 3.270917 |
| 20 | 21 | 23.901670 |

# regression

In [29]:
```
y1
```

Out[29]:
```
array([ 702.  , 1595.  ,  713.51, 1886.27,  468.45,  715.71,  442.84,
       1241.  , 1427.  ,  125.  , 1124.95,  663.  ,  228.66,  693.  ,
       1549.91,  920.42, 1154.52,  404.  ,  100.  ,  246.6 , 1804.11,
       3530.  , 6990.  , 1174.  ,  149.  ,  374.  ,  407.  , 2565.59,
         75.  ,  223.  ,  896.6 ])
```

In [30]:
```
import sklearn#library used for ML
from sklearn.model_selection import train_test_split
x_train,x_test,y1_train,y1_test= train_test_split(x,y1,test_size=0.25, random_state=0)
```

In [31]:
```
from sklearn.linear_model import LinearRegression
LR1=LinearRegression()
```

In [32]:
```
LR1.fit(x_train,y1_train)
```

Out[32]:
```
▾ LinearRegression
LinearRegression()
```

In [33]:
```
y1_pred =LR1.predict(x_test)
```

In [34]:
```
y1_pred
```

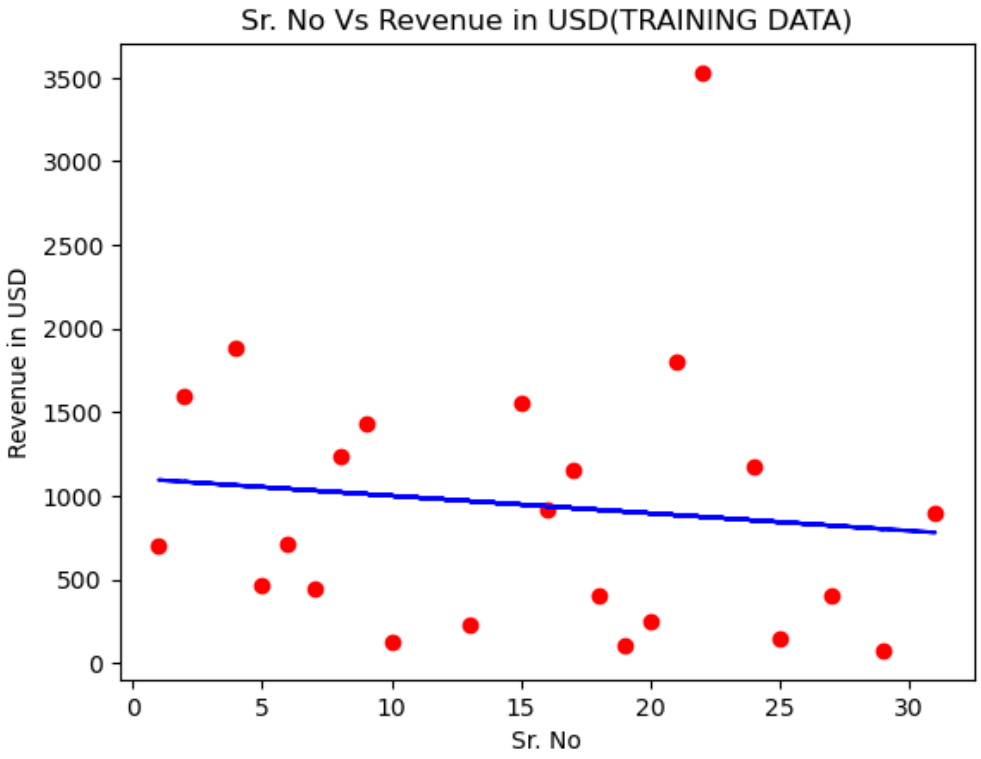
Out[34]:
```
array([1020.80482103,  805.24637867,  932.98471489,  956.93565293,
        821.2136707 ,  837.18096273,  861.13190077,  948.95200691])
```

## Applying the Linear Regression

## Plotting the training set

### PLOTTING TRAINING SET

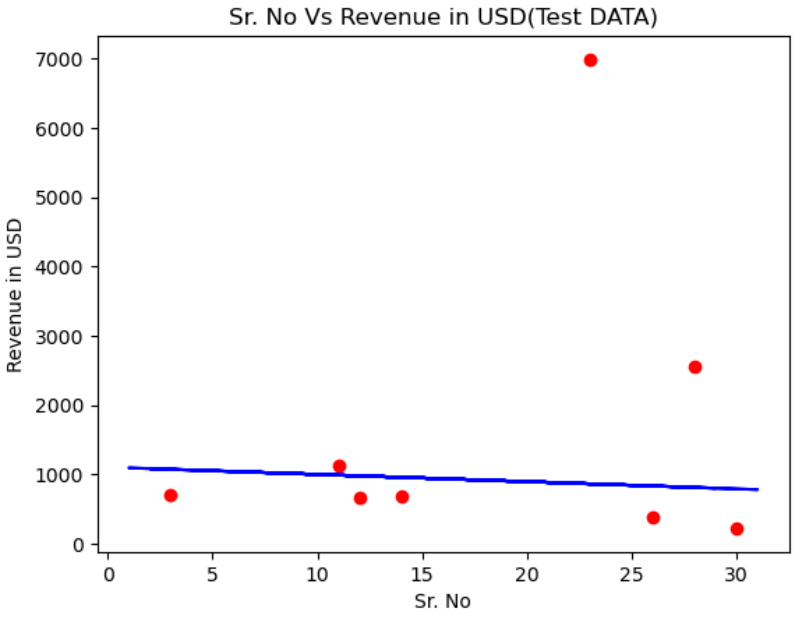
In [35]:
```
plt.scatter(x_train,y1_train, color='red')
plt.plot(x_train,LR.predict(x_train), color ='blue' )
plt.title("Sr. No Vs Revenue in USD(TRAINING DATA)")
plt.xlabel("Sr. No")
plt.ylabel("Revenue in USD")
plt.show()
```



## Plotting the test set

# PLOTTING TEST SET

[36]:
```
plt.scatter(x_test,y1_test, color='red')
plt.plot(x_train,LR.predict(x_train), color ='blue' )
plt.title("Sr. No Vs Revenue in USD(Test DATA)")
plt.xlabel("Sr. No")
plt.ylabel("Revenue in USD")
plt.show()
```



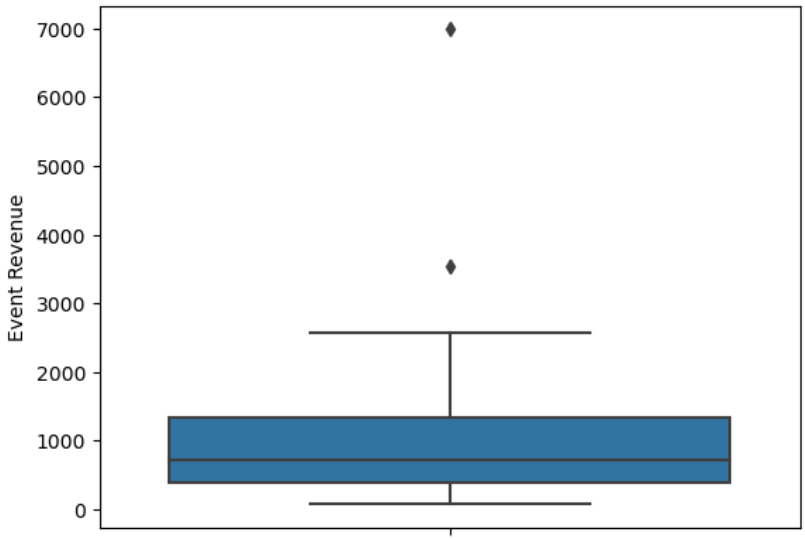
# Removing Outliers

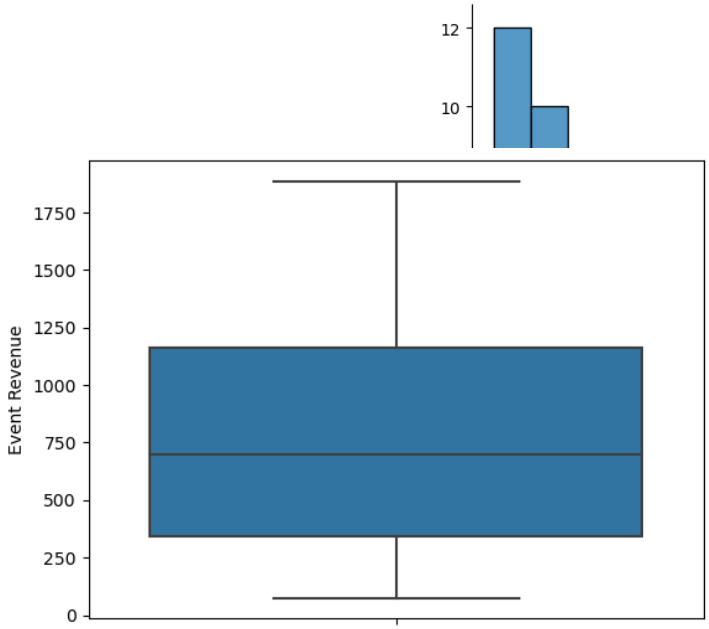
## Remove outliers

In [37]:
```
import seabor
```

In [38]:
```
sns.displot(da
```

Out[38]: <seaborn.axis




In [40]:
```
dataset['Event Reve
```

Out[40]: 1115.8109677419352

In [41]:
```
data1=dataset[datas
```

In [42]:
```
sns.boxplot(y='Even
```

Out[42]: <Axes: ylabel='Even

In [39]:
```
sns.boxplot(y='Event Revenue',data=dataset)
```

## check accuracy

In [49]:
```
# Calculate metrics
mse = mean_squared_error(y2_test, y2_pred)
rmse = mean_squared_error(y2_test, y2_pred, squared=False)
mae = mean_absolute_error(y2_test, y2_pred)
r2 = r2_score(y2_test, y2_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared (R2):", r2)
```

```
Mean Squared Error (MSE): 5181454.447907679
Root Mean Squared Error (RMSE): 2276.2808367834755
Mean Absolute Error (MAE): 1247.909784916245
R-squared (R2): -0.149568951094172
```

## Checking accuracy after removing outliers

## Predicting the next 10 values

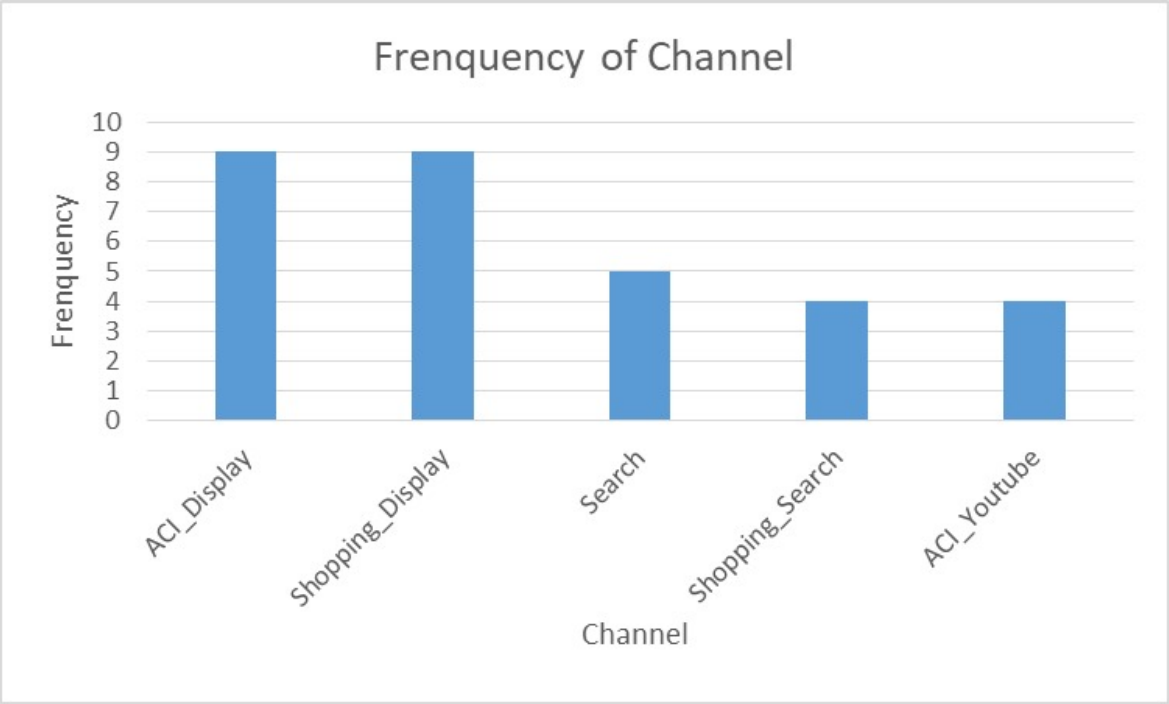
### Predict the next 10 values of given data

In [50]:
```
LR.fit(x, y)

next_values = py.linspace(101, 110, 10).reshape(-1, 1)
predictions = LR.predict(next_values)

print(predictions)
```

```
[2350.72489919 2365.25329839 2379.78169758 2394.31009677 2408.83849597
 2423.36689516 2437.89529435 2452.42369355 2466.95209274 2481.48049194]
```

## Data visualisation

- In the data visualisation, we need to find insights on the given dataset as given above.




## Insights of Channels

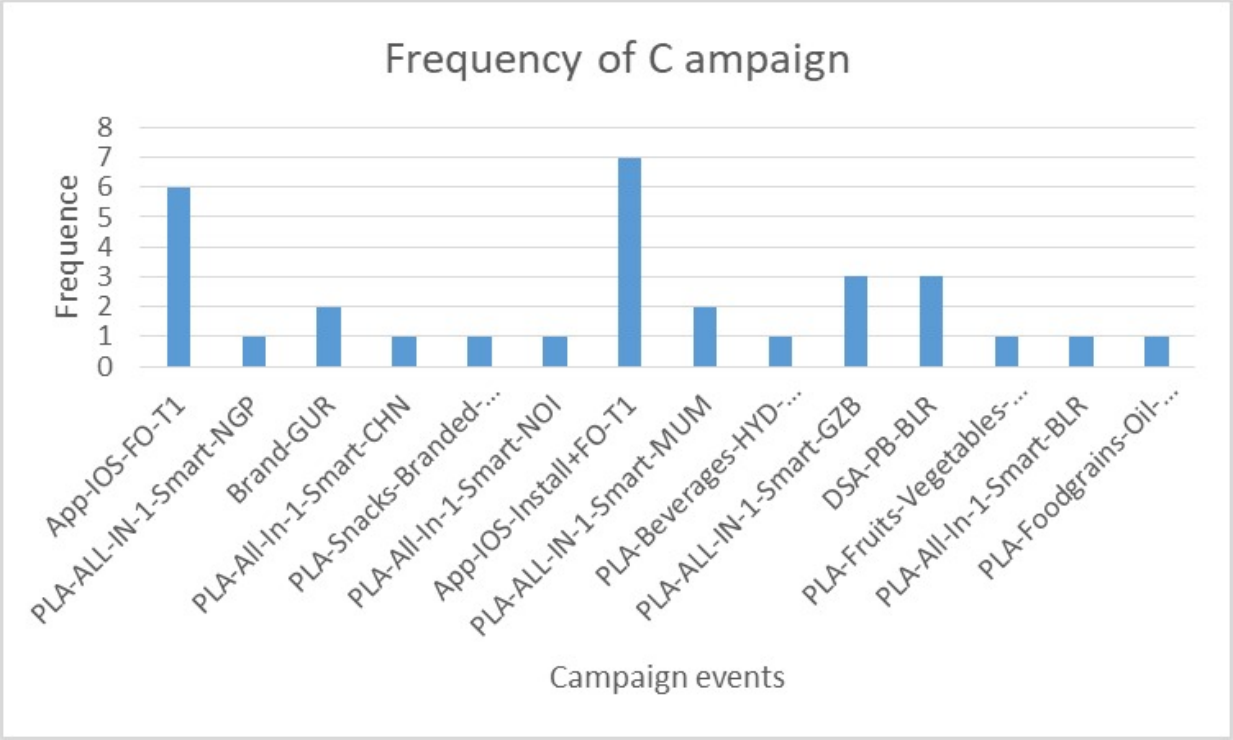
- In the above insight, it represents the channel where the customers of clicks in Big basket site through the following ways.

- ACI_Display and Shopping_Display are the most clicks which the ads that are displays are more engagement to the customers.

- Shopping_Search and ACI_Youtube are the least clicks which the ads makes customers to engage less so we can take steps to engage the customers more.

## Insights of Campaign

- In the above insight, it represents the campaign where the customers of clicks in Big basket site through the following ways.




- The most of the clicks from App-IOS-Install+FO-T1 which has more potential to engage the customers in this campaign .

- The least are many campaigns that need to impose in the upcoming days.

## Insights of State

- In the above insight, it represents the state where the customers of clicks in Big basket site through the following ways.

- The most clicks are the Telangana in Hyderabad where the customers are more engagement in the ads.

- The least clicks are get from Tamil Nadu, Haryana and Uttar Pradesh that makes the ads should be more engagement to the customers.

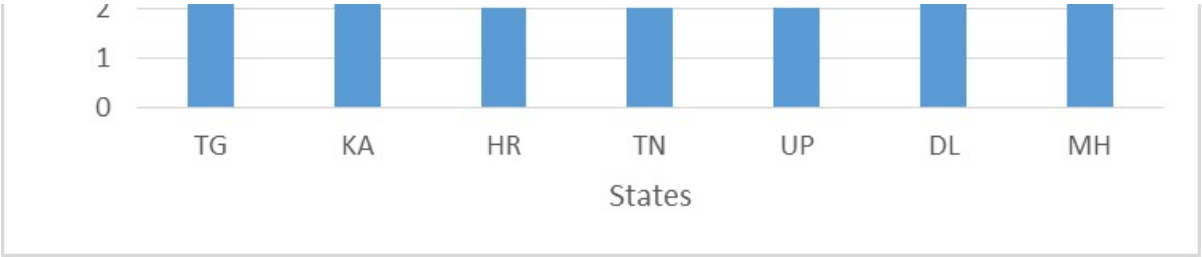
# Conclusion

In [1]:
```
# There are 31 clicks in the given dataset that we made the linear regression in it
```

In [2]:
```
# As the ads are more engagement in the following insights
# Channels : ACI_Display and Shopping_Display
# Campaign : App-IOS-Install+FO-T1
# State : Telangana
```

In [3]:
```
# In the above code are clearly displays the following tasks
# calculating accuracy
# Check other variables as independent variables and try to apply regression
# Remove outliers and check accuracy
# Predict the next 10 values of given data
```

In [ ]:
```
# In this way we have used Linear Regression to analyse which are preferable for conversion campaign
# This will help marketing to take decisions about the remarketing Campaign
```




Conclusion