Data Analysis of E-Commerce Data from Big Basket using linear regression

Date : 07-16-2023

Project Start Date - End Date	● Start Date – 07 -06 -2023 ● End Date – 07 -16 2023
Objectives	● General descriptive analyses ● General exploratory analyses ● To analyze revenue using the features in the dataset and predict near future revenue using linear regression machine learning model
Milestones accomplished the week of Start Date - End Date:	● Calculate accuracy metrics for performance evaluation of the model. ● Conduct regression analysis to explore relationships between variables. ● Detect and handle outliers in the dataset. ● Predict future values using the developed data analysis model.

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

Student Name

Name: MARJIBA

Mobile: +91 8415946250

Email: marjibajamir9@gmail.com

Project Abstract

In this project, we aim to perform advanced data analysis on the shopping data obtained from the e-commerce platform Big Basket. Our main objective is to develop an accurate and efficient data analysis model that can provide valuable insights into customer behavior and other relevant aspects. The tasks to be accomplished include:

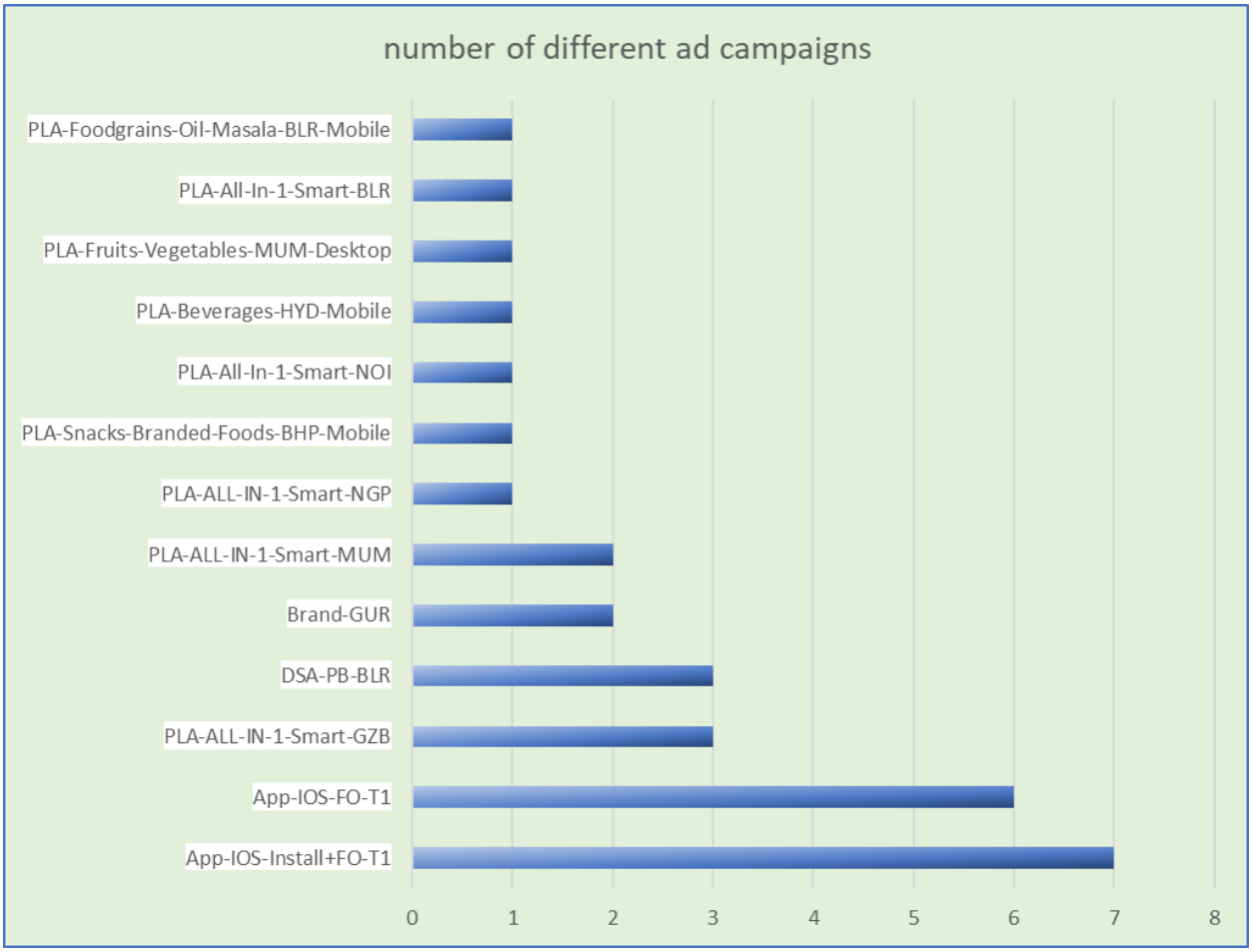
Calculate Accuracy: Develop a code to calculate the accuracy of the data analysis model. This accuracy metric will be used to evaluate the performance of the model in predicting customer behavior or any other relevant aspect.

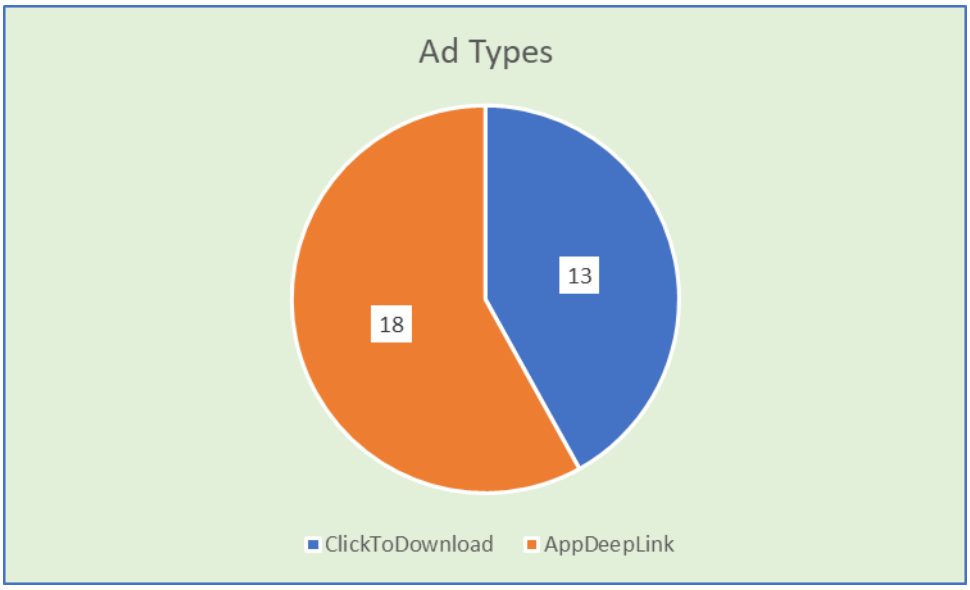
Regression Analysis: Explore other variables within the dataset as independent variables and apply regression analysis.

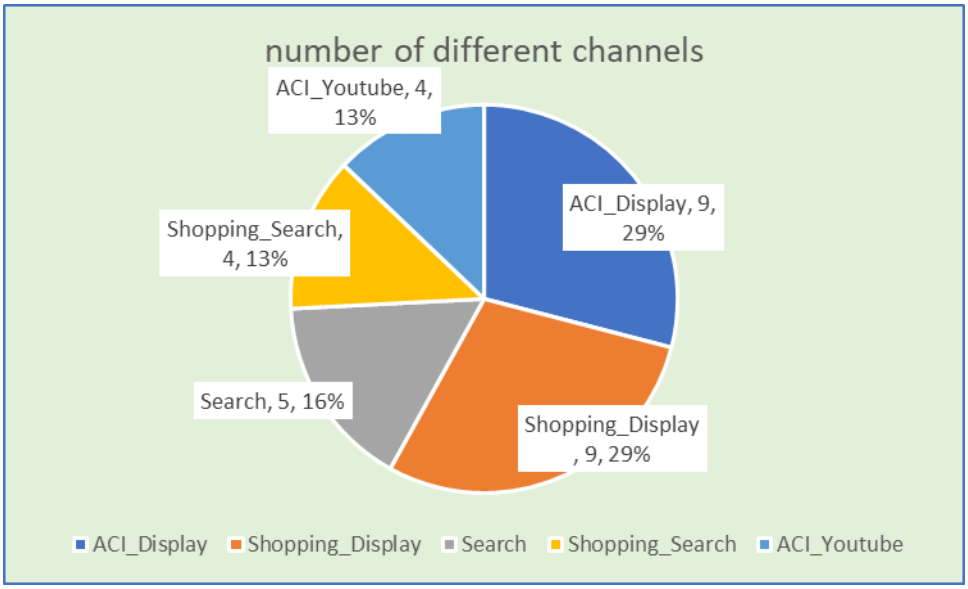
Outlier Detection: Identify and remove outliers from the dataset.

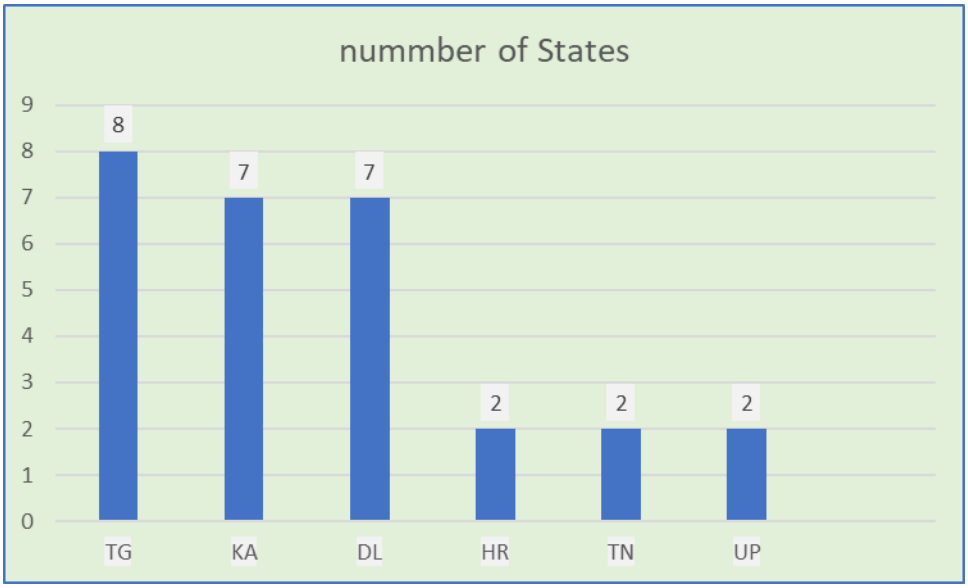
Prediction of Future Values: Utilize the developed data analysis model to predict the next 10 values in the given dataset.

Some General Descriptive Data:






number of different channels
ACI_Youtube, 4, 13%
ACI_Display, 9, 29%
Shopping_Search, 4, 13%
Search, 5, 16%
Shopping_Display, 9, 29%
ACI_Display
Shopping_Display
Search
Shopping_Search
ACI_Youtube


nummber of States
9
8
7
6
5
4
3
2
1
0
8
7
7
2
2
2
TG
KA
DL
HR
TN
UP


number of cities
Whitefield
Mumbai
Sriperumbudur
Chennai
Nandi
Thane
Secunderabad
Gautam Buddha Nagar
Gurgaon
Hyderabad
Kukatpally
Bangalore
Delhi
1
1
1
1
1
2
2
2
2
2
4
5
7
0
1
2
3
4
5
6
7
8

Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# save the path of csv file in a variable called path
path="C:/Users/rooki/OneDrive/Desktop/9 july in app ios.csv"

# Load the dataset into a pandas dataframe
bbdata = pd.read_csv(path)

# Print the first few rows of the dataframe to verify the data has been loade
d correctly
bbdata.head()
```

```
  Attributed Touch Type  Event Name  \
0                 click  placeorder
1                 click  placeorder
2                 click  placeorder
3                 click  placeorder
4                 click  placeorder

                                        Event Value  Event Revenue  \
0  {"af_content_type":"product","order id":"21135...         702.00
1  {"af_content_type":"product","order id":"21134...        1595.00
2  {"af_content_type":"product","order id":"21133...         713.51
3  {"af_content_type":"product","order id":"21133...        1886.27
4  {"af_content_type":"product","order id":"21132...         468.45

  Event Revenue Currency  Event Revenue USD  Cost Model  Cost Value  \
0                    INR           9.320797         NaN         NaN
1                    INR          21.184909         NaN         NaN
2                    INR           9.476893         NaN         NaN
3                    INR          25.048669         NaN         NaN
4                    INR           6.220768         NaN         NaN

   Cost Currency Event Source  ...  Is Retargeting  \
0            NaN          SDK  ...           False
1            NaN          SDK  ...           False
2            NaN          SDK  ...           False
3            NaN          SDK  ...           False
4            NaN          SDK  ...           False

  Retargeting Conversion Type Is Primary Attribution Attribution Lookback  \
0                         NaN                    True                  30d
```

```
1                        NaN                  False                   30d
2                        NaN                   True                   30d
3                        NaN                   True                   30d
4                        NaN                   True                   30d

   Reengagement Window Match Type  \
0                     NaN        srn
1                     NaN        srn
2                     NaN        srn
3                     NaN        srn
4                     NaN        srn

                                        User Agent  HTTP Referrer  \
0  bigbasket/6.2.2 CFNetwork/1240.0.4 Darwin/20.6.0            NaN
1      bigbasket/6.2.2 CFNetwork/1197 Darwin/20.0.0            NaN
2      bigbasket/6.2.2 CFNetwork/1209 Darwin/20.2.0            NaN
3      bigbasket/6.2.2 CFNetwork/1209 Darwin/20.2.0            NaN
4  bigbasket/6.2.2 CFNetwork/1240.0.4 Darwin/20.6.0            NaN

   Original URL Store Product Page
0           NaN                NaN
1           NaN                NaN
2           NaN                NaN
3           NaN                NaN
4           NaN                NaN

[5 rows x 56 columns]
```

```
bbdata.shape
```

```
(31, 56)
```

the dataset contains 31 rows and 56 columns (features) the first step is to filter out the irrelevent features and Pre-process the data

```
# we'll use the first 4 columns which include: Attributed touch type, Event N
ame, Event Value and Event Revenue
data_1 = bbdata.iloc[:,:4]
```

```
data_1.head()
```

```
  Attributed Touch Type  Event Name  \
0                 click  placeorder
1                 click  placeorder
2                 click  placeorder
3                 click  placeorder
4                 click  placeorder

                                        Event Value  Event Revenue
0  {"af_content_type":"product","order id":"21135...         702.00
1  {"af_content_type":"product","order id":"21134...        1595.00
```

```
2  {"af_content_type":"product","order id":"21133...         713.51
3  {"af_content_type":"product","order id":"21133...        1886.27
4  {"af_content_type":"product","order id":"21132...         468.45
```

```
data_1.describe()
```

```
       Event Revenue
count      31.000000
mean     1115.810968
std      1339.215906
min        75.000000
25%       389.000000
50%       713.510000
75%      1334.000000
max      6990.000000
```

```
# although not obvious immediately, we can notice that the maximum revenue is
69900 where as the 75th percentile revenue is
# only 1334, this indicates that entry number 22 (Event Revenue is an outlier
)
# the idea way to handle this would be to remove this since, 1 datapoint is r
oughly 3% of the whole dataset.

# Drop the row using drop function
data_1 = data_1.drop(22)

# Resetting the index
data_1 = data_1.reset_index(drop=True)

data_1.head()
```

```
  Attributed Touch Type  Event Name  \
0                 click  placeorder
1                 click  placeorder
2                 click  placeorder
3                 click  placeorder
4                 click  placeorder

                                        Event Value  Event Revenue
0  {"af_content_type":"product","order id":"21135...         702.00
1  {"af_content_type":"product","order id":"21134...        1595.00
2  {"af_content_type":"product","order id":"21133...         713.51
3  {"af_content_type":"product","order id":"21133...        1886.27
4  {"af_content_type":"product","order id":"21132...         468.45
```

```
data_1.describe()
```

```
       Event Revenue
count      30.000000
mean      920.004667
std       791.085620
```

```
min        75.000000
25%       381.500000
50%       707.755000
75%      1224.250000
max      3530.000000
```

Let's convert the features into numerical values and see which features can help predict our dependent variable (Event Revenue)

```
# Mapping for "Attributed Touch Type"
touch_type_mapping = {'click': 1}  # Add more mappings as needed

# Mapping for "Event Name"
event_name_mapping = {'placeorder': 1}  # Add more mappings as needed

# Convert "Attributed Touch Type" column
data_1['Attributed Touch Type'] = data_1['Attributed Touch Type'].map(touch_t
ype_mapping)

# Convert "Event Name" column
data_1['Event Name'] = data_1['Event Name'].map(event_name_mapping)
```

```
data_1.head()
```

```
   Attributed Touch Type  Event Name  \
0                      1           1
1                      1           1
2                      1           1
3                      1           1
4                      1           1

                                        Event Value  Event Revenue
0  {"af_content_type":"product","order id":"21135...         702.00
1  {"af_content_type":"product","order id":"21134...        1595.00
2  {"af_content_type":"product","order id":"21133...         713.51
3  {"af_content_type":"product","order id":"21133...        1886.27
4  {"af_content_type":"product","order id":"21132...         468.45
```

Now that we have everything in numerical form, we can split the data into train - test uisng available library and then, build a linear Regression model and finally check our model's accuracy.

```
# convert the DataFrame to values
X = data_1.iloc[:, 0:1].values
y = data_1.iloc[:, -1].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rand
om_state=42)
```

```
X_train
```

```
array([[1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1]], dtype=int64)
```

```
X_test
```

```
array([[1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1]], dtype=int64)
```

```
y_train
```

```
array([ 702.  ,  468.45, 1154.52,  715.71,  693.  ,  663.  , 1174.  ,
       1595.  ,  713.51,  407.  , 1886.27, 3530.  , 2565.59,  100.  ,
        896.6 , 1804.11, 1241.  , 1124.95, 1549.91,  246.6 ,  442.84])
```

```
y_test
```

```
array([  75.  ,  920.42,  149.  ,  404.  , 1427.  ,  125.  ,  223.  ,
        374.  ,  228.66])
```

```
# Train a linear regression model on the data
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

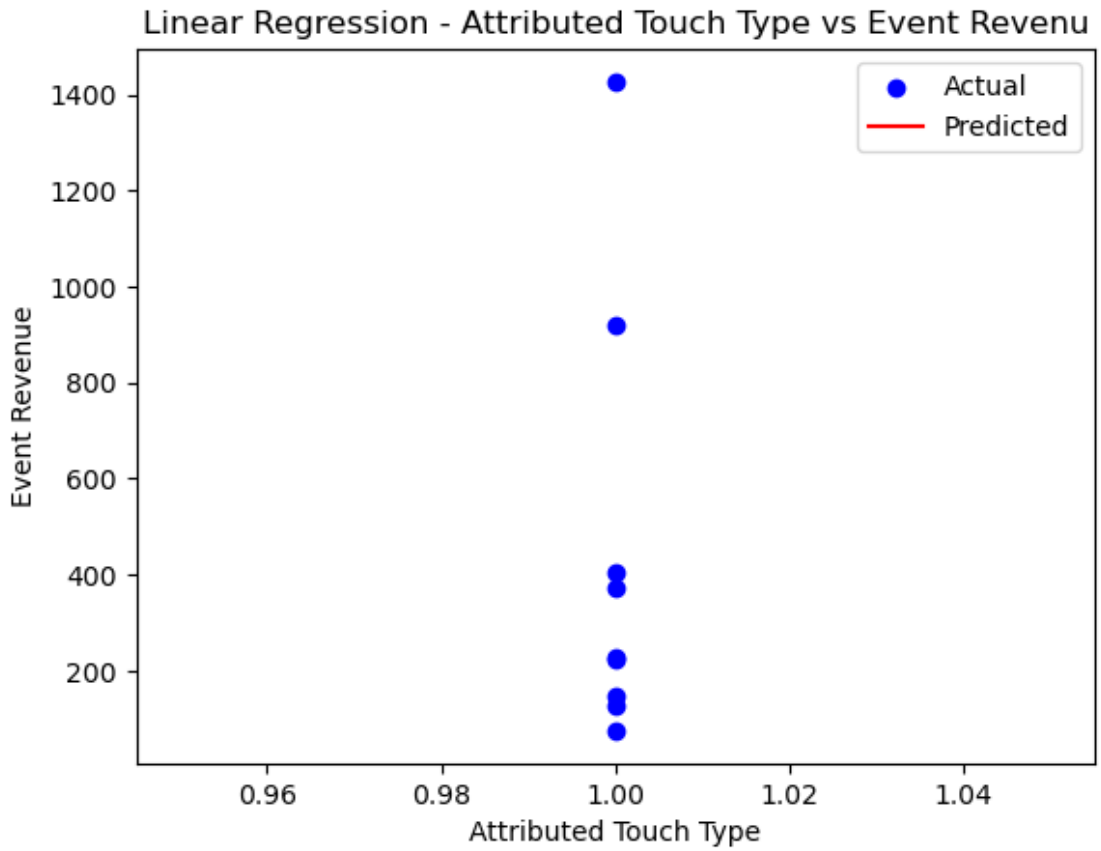
```
# Make predictions
y_pred = model.predict(X_test)

y_pred
```

```
array([1127.33619048, 1127.33619048, 1127.33619048, 1127.33619048,
       1127.33619048, 1127.33619048, 1127.33619048, 1127.33619048,
       1127.33619048])
```

```
# Calculate the accuracy (R^2 score)
accuracy = r2_score(y_test, y_pred)
accuracy
```

```
-2.6544931229972804
```

```
# Plot the results
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Predicted')
plt.xlabel('Attributed Touch Type')
plt.ylabel('Event Revenue')
plt.title('Linear Regression - Attributed Touch Type vs Event Revenu')
plt.legend()
plt.show()
```




```
data_2=bbdata.iloc[:,3:14]
```

```
data_2.head()
```

```
   Event Revenue Event Revenue Currency  Event Revenue USD  Cost Model  \
0         702.00                    INR           9.320797         NaN
1        1595.00                    INR          21.184909         NaN
2         713.51                    INR           9.476893         NaN
3        1886.27                    INR          25.048669         NaN
4         468.45                    INR           6.220768         NaN

   Cost Value  Cost Currency Event Source  Partner       Media Source  \
0         NaN            NaN          SDK      NaN  googleadwords_int
1         NaN            NaN          SDK      NaN  googleadwords_int
2         NaN            NaN          SDK      NaN  googleadwords_int
3         NaN            NaN          SDK      NaN  googleadwords_int
4         NaN            NaN          SDK      NaN  googleadwords_int

            Channel                Campaign
0       ACI_Display           App-IOS-FO-T1
1  Shopping_Display  PLA-ALL-IN-1-Smart-NGP
2            Search               Brand-GUR
3            Search               Brand-GUR
4       ACI_Display           App-IOS-FO-T1
```

```
from sklearn.preprocessing import LabelEncoder

# Convert the "Channel" column to numerical format
label_encoder = LabelEncoder()
data_2['Channel'] = label_encoder.fit_transform(data_2['Channel'])
data_2['Campaign'] = label_encoder.fit_transform(data_2['Campaign'])

# Print the updated DataFrame
data_2.head()
```

```
   Event Revenue Event Revenue Currency  Event Revenue USD  Cost Model  \
0         702.00                    INR           9.320797         NaN
1        1595.00                    INR          21.184909         NaN
2         713.51                    INR           9.476893         NaN
3        1886.27                    INR          25.048669         NaN
4         468.45                    INR           6.220768         NaN

   Cost Value  Cost Currency Event Source  Partner       Media Source  \
0         NaN            NaN          SDK      NaN  googleadwords_int
1         NaN            NaN          SDK      NaN  googleadwords_int
2         NaN            NaN          SDK      NaN  googleadwords_int
3         NaN            NaN          SDK      NaN  googleadwords_int
4         NaN            NaN          SDK      NaN  googleadwords_int

   Channel  Campaign
0        0         0
1        3         6
```

```
2         2         2
3         2         2
4         0         0
```

```
# Drop the row using drop function
data_2 = data_2.drop(22)
data_2 = data_2.drop(21)
```

```
# Resetting the index
data_2 = data_2.reset_index(drop=True)
```

```
data_2.head()
```

```
   Event Revenue Event Revenue Currency  Event Revenue USD  Cost Model  \
0         702.00                     INR           9.320797         NaN
1        1595.00                     INR          21.184909         NaN
2         713.51                     INR           9.476893         NaN
3        1886.27                     INR          25.048669         NaN
4         468.45                     INR           6.220768         NaN

   Cost Value  Cost Currency Event Source  Partner       Media Source  \
0         NaN            NaN          SDK      NaN  googleadwords_int
1         NaN            NaN          SDK      NaN  googleadwords_int
2         NaN            NaN          SDK      NaN  googleadwords_int
3         NaN            NaN          SDK      NaN  googleadwords_int
4         NaN            NaN          SDK      NaN  googleadwords_int

   Channel  Campaign
0        0         0
1        3         6
2        2         2
3        2         2
4        0         0
```

```
# convert the DataFrame to values
X2 = data_2.iloc[:, 0].values.reshape(-1, 1)
y2 = data_2.iloc[:, -2].values.reshape(-1, 1) # channel
```

```
X2
```

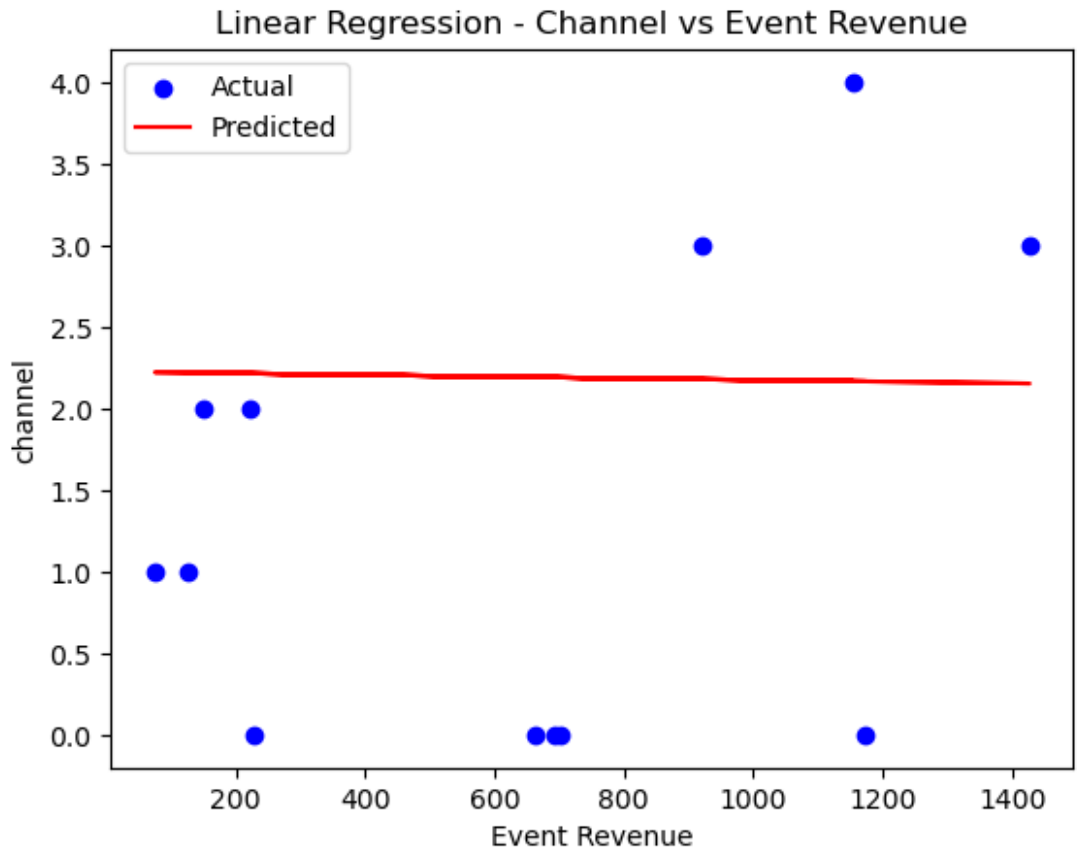
```
array([[ 702.  ],
       [1595.  ],
       [ 713.51],
       [1886.27],
       [ 468.45],
       [ 715.71],
       [ 442.84],
       [1241.  ],
       [1427.  ],
```

```
       [ 125.  ],
       [1124.95],
       [ 663.  ],
       [ 228.66],
       [ 693.  ],
       [1549.91],
       [ 920.42],
       [1154.52],
       [ 404.  ],
       [ 100.  ],
       [ 246.6 ],
       [1804.11],
       [1174.  ],
       [ 149.  ],
       [ 374.  ],
       [ 407.  ],
       [2565.59],
       [  75.  ],
       [ 223.  ],
       [ 896.6 ]])
```

y2

```
array([[0],
       [3],
       [2],
       [2],
       [0],
       [3],
       [0],
       [4],
       [3],
       [1],
       [1],
       [0],
       [0],
       [0],
       [3],
       [3],
       [4],
       [3],
       [2],
       [4],
       [3],
       [0],
       [2],
       [3],
       [3],
       [1],
       [1],
```

```
       [2],
       [0]])

# Split the data into training and testing sets
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.4
, random_state=42)

# Train a linear regression model on the data
model2 = LinearRegression()
model2.fit(X2_train, y2_train)

LinearRegression()

# Make predictions
y2_pred = model2.predict(X2_test)

# Plot the results
plt.scatter(X2_test, y2_test, color='blue', label='Actual')
plt.plot(X2_test, y2_pred, color='red', label='Predicted')
plt.xlabel('Event Revenue')
plt.ylabel('channel')
plt.title('Linear Regression - Channel vs Event Revenue')
plt.legend()
plt.show()
```



```
# Calculate the accuracy (R^2 score)
accuracy2 = r2_score(y2_test, y2_pred)
accuracy2
```

```
-0.40395883048994174
```

```
# convert the DataFrame to values
y3 = data_2.iloc[:, 0].values.reshape(-1, 1)
X3 = data_2.iloc[:, -1].values.reshape(-1, 1) # channel
```

```
X3
```

```
array([[ 0],
       [ 6],
       [ 2],
       [ 2],
       [ 0],
       [ 8],
       [ 0],
       [13],
       [ 9],
       [ 1],
       [ 1],
       [ 1],
       [ 1],
       [ 1],
       [ 5],
       [ 5],
       [10],
       [ 4],
       [ 3],
       [12],
       [ 7],
       [ 0],
       [ 3],
       [ 4],
       [ 4],
       [ 1],
       [ 1],
       [ 3],
       [ 0]])
```

```
y3
```

```
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],
       [1174.  ],
       [ 149.  ],
       [ 374.  ],
       [ 407.  ],
       [2565.59],
       [  75.  ],
       [ 223.  ],
       [ 896.6 ]])
```

```
# Split the data into training and testing sets
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.3
, random_state=42)

# Train a linear regression model on the data
model3 = LinearRegression()
model3.fit(X3_train, y3_train)
```

```
LinearRegression()
```

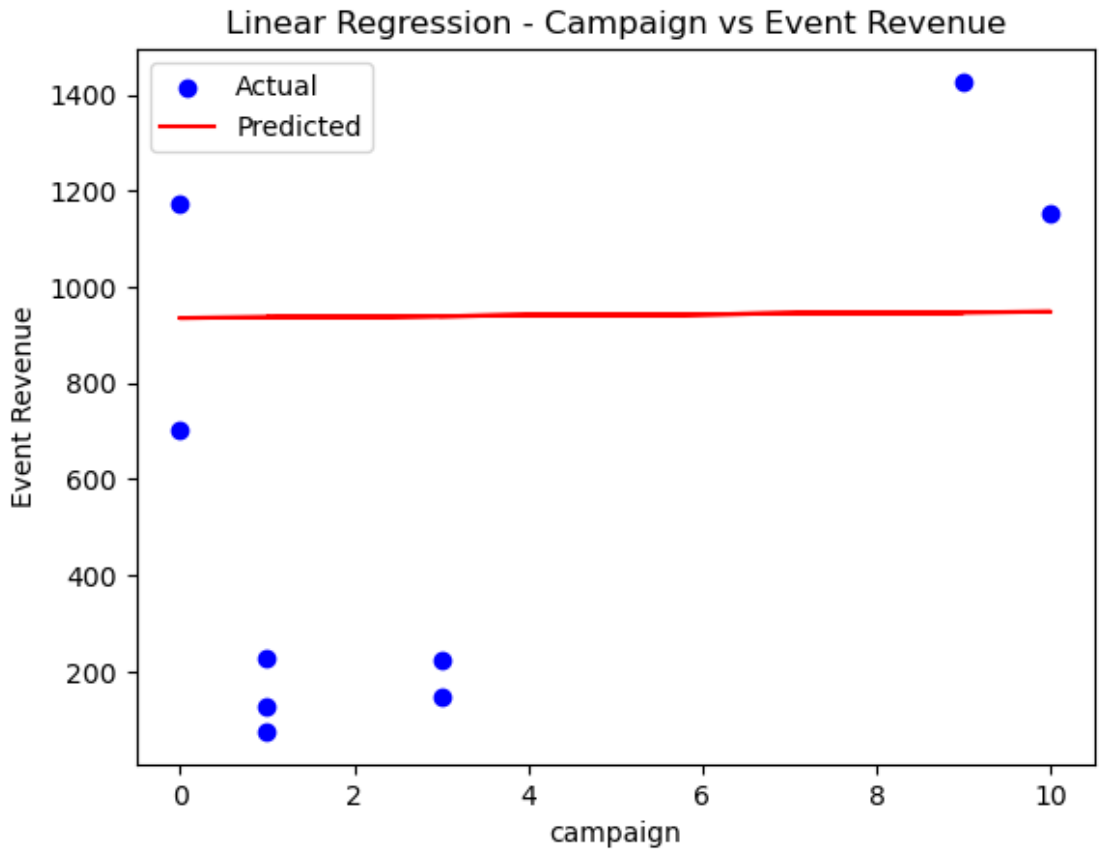
```
# Make predictions
y3_pred = model3.predict(X3_test)
```

```
y3_pred
```

```
array([[939.3717494 ],
       [948.40728016],
       [936.79016918],
       [939.3717494 ],
       [947.11649005],
       [936.79016918],
       [935.49937907],
       [935.49937907],
       [936.79016918]])
```

```
# Plot the results
plt.scatter(X3_test, y3_test, color='blue', label='Actual')
plt.plot(X3_test, y3_pred, color='red', label='Predicted')
```

```
plt.xlabel('campaign')
plt.ylabel('Event Revenue')
plt.title('Linear Regression - Campaign vs Event Revenue')
plt.legend()
plt.show()
```




```
# Calculate the accuracy (R^2 score)
accuracy3 = r2_score(y3_test, y3_pred)
accuracy3
```

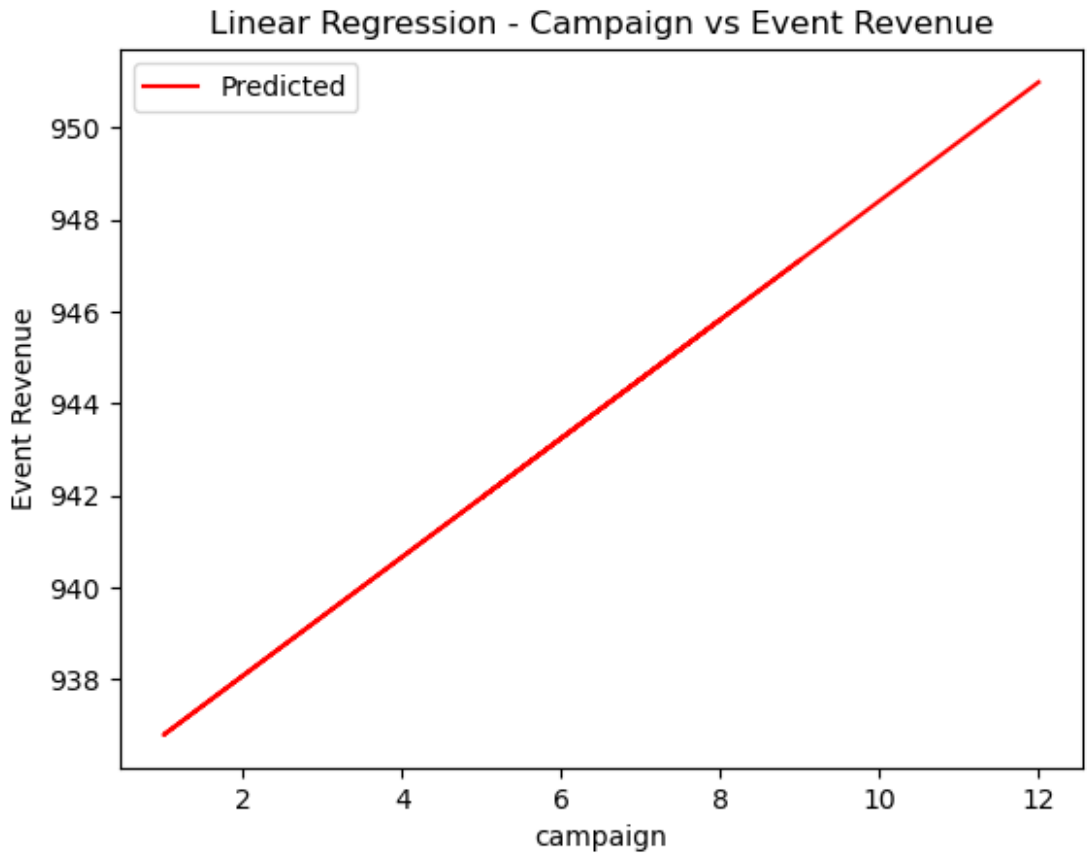
```
-0.48036726397303164
```

```
# try for next 10 values
# Generate an array of 10 random numbers between 0 and 13
campaign_array = np.random.randint(0, 14, size=10).reshape(-1, 1)
```

```
campaign_array
```

```
array([[ 9],
       [ 1],
       [ 6],
       [ 7],
       [ 6],
       [ 5],
       [ 8],
```

```
       [ 7],
       [10],
       [12]])

campaign_predictions = model3.predict(campaign_array)
# Make predictions
y3_pred = model3.predict(X3_test)

campaign_predictions

array([[947.11649005],
       [936.79016918],
       [943.24411972],
       [944.53490983],
       [943.24411972],
       [941.95332962],
       [945.82569994],
       [944.53490983],
       [948.40728016],
       [950.98886038]])

# Plot the results
# plt.scatter(X3_test, y3_test, color='blue', label='Actual')
plt.plot(campaign_array, campaign_predictions, color='red', label='Predicted'
)
plt.xlabel('campaign')
plt.ylabel('Event Revenue')
plt.title('Linear Regression - Campaign vs Event Revenue')
plt.legend()
plt.show()
```




```
## predicted revenue for next 10 campaigns --
campaign_predictions

array([[947.11649005],
       [936.79016918],
       [943.24411972],
       [944.53490983],
       [943.24411972],
       [941.95332962],
       [945.82569994],
       [944.53490983],
       [948.40728016],
       [950.98886038]])
```

Summary and Insights

Geographical User Distribution: Telangana state leads with the highest number of interaction at 8, clos ely followed by Karnataka and Delhi, each with 7 users. Delhi city stands out with 7 interactions, making it the city with the highest interaction count. Bangalore and Kukatpally rank second and third with 5 and 4 in teractions, respectively.

Ad Performance Analysis: The two primary ad types, "Click to Download" and "App Deep Link," have b een analyzed for their performance. "Click to Download" ads constitute 58% of the ad types, while "App D eep Link" accounts for the remaining 42%. This information can assist in optimizing ad campaigns to targ et specific user preferences.

Popular Channels for User Interaction: The data highlights "Shopping Display" and "ACI Display" as th e most popular channels for user engagement. Leveraging these channels can help improve customer int eractivity and boost overall revenue.

Revenue Prediction Models: Linear regression models have been developed to predict event revenue b ased on channel and campaign data. The model accuracy, measured by the R2_SCORE metric, indicate s negative values for both channel and campaign-based predictions.

This suggests that the current models may not be the best fit for revenue prediction, and further refineme nt may be necessary.

Campaign-Based Revenue Prediction: Using the campaign-based model to predict the next 10 revenue s for randomly selected campaigns shows a linear regression line ranging approximately from 938 to 950 in revenue.

This prediction range can aid in setting revenue expectations and refining future marketing strategies.