Project on Online Shopping Intention

Aim - To Predict and to identify whether the customers will purchase the product or not provide winning excellent services in the E-commerce industry that can gain competitive advantages. Competitiveness is being faced by this industry Worldwide. Customers compare the price of products within many E-commerce apps. Companies thus need to provide an excellent product. A satisfied customer will spread positive word of mouth about the product when they have a positive experience at every stage of their purchase. Thus, we will visualise the given dataset to find meaningful insights from them and use Machine Learning Models to predict the purchase behaviour of customers based on the features given in the dataset.

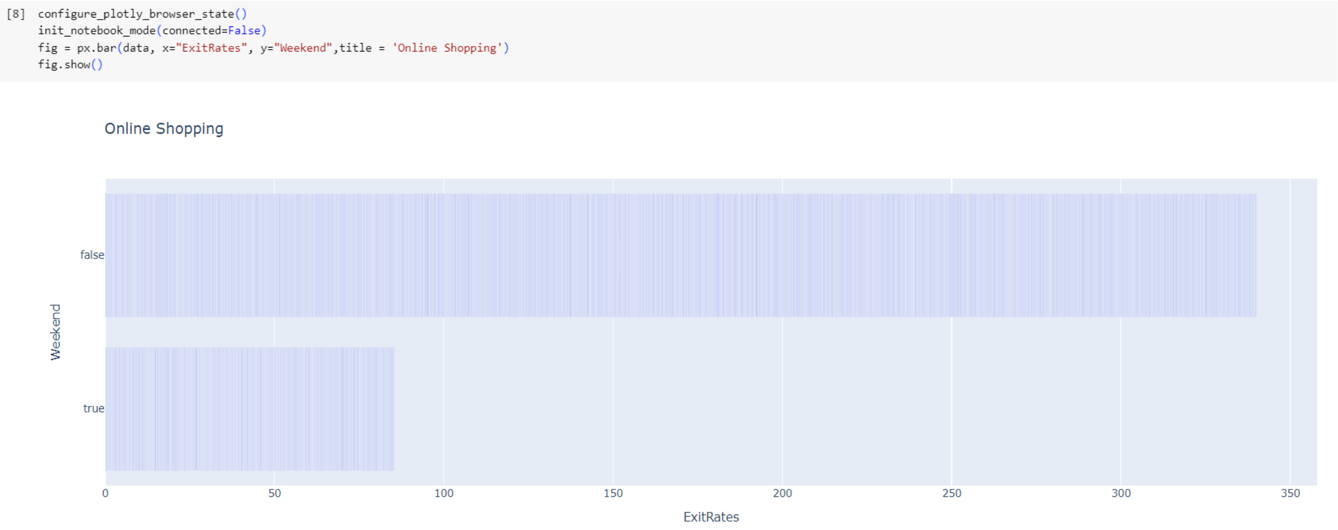
Problem Statement – The recent surge in online shopping has given the business world a new dimension. People frequently search the internet for the products they require and purchase them through online transactions. It has made their lives more convenient and comfortable. Simultaneously, the sellers must understand the patterns and intentions of the customers.

Data-
Train Dataset - It consists of a total of 9864 datapoints.
Test Dataset - You must predict the stage of cirrhosis of 2466 datapoints.

The project is sub-divided following sections. These are:

- Loading necessary libraries
- Loading Dataset from a CSV file or from a Table.
- Summarization of Data to understand Dataset.
- Visualization of Data to understand Dataset (Plots, Graphs etc.) using Cufflinks and plotly libraries of Python and using Power BI.
- Data pre-processing and Data transformation.
- Splitting the data set into independent & dependent sets.
- Importing the train_test_split model from sklearn model for splitting data into train & test sets.
- Importing various ML model & then training those models with the help of fit ().
- Predicting the trained models & then checking their accuracy of the model.
- Then, trained the test dataset with Tain dataset with the help of better accuracy model.
- Finally, predicting whether the customer will purchase the product or not based on the new data frame using dictionary.

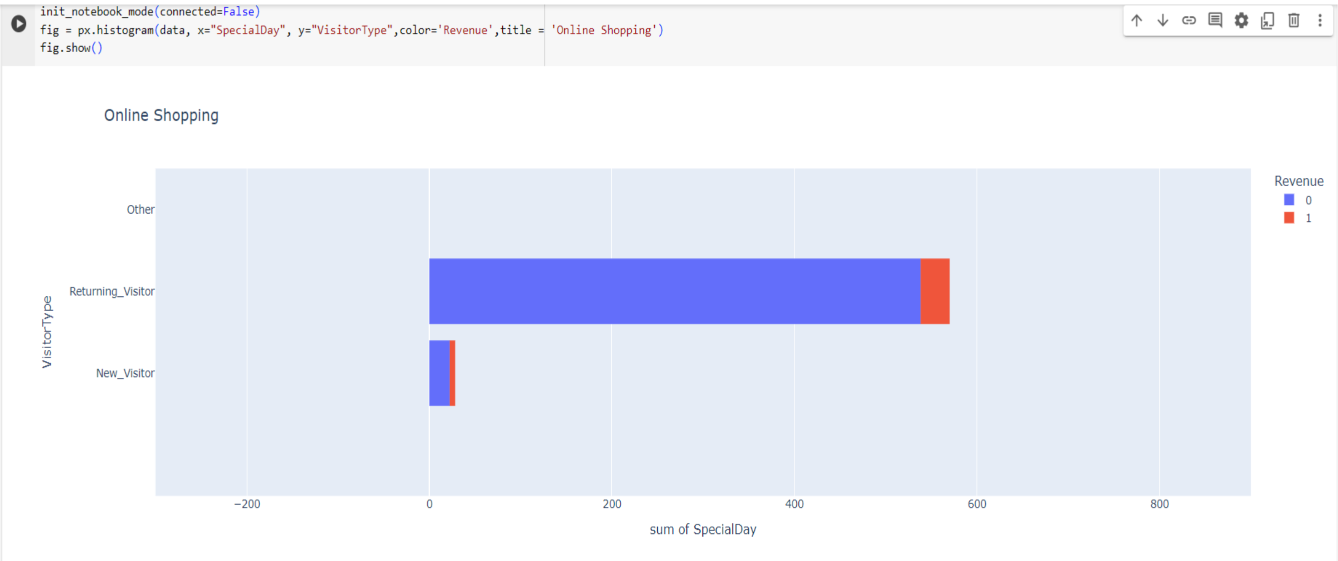
Data Visualization:



```
[8] configure_plotly_browser_state()
    init_notebook_mode(connected=False)
    fig = px.bar(data, x="ExitRates", y="Weekend",title = 'Online Shopping')
    fig.show()
```

Fig 1: Bar Graph showing Exit Rate with respect to Weekend.

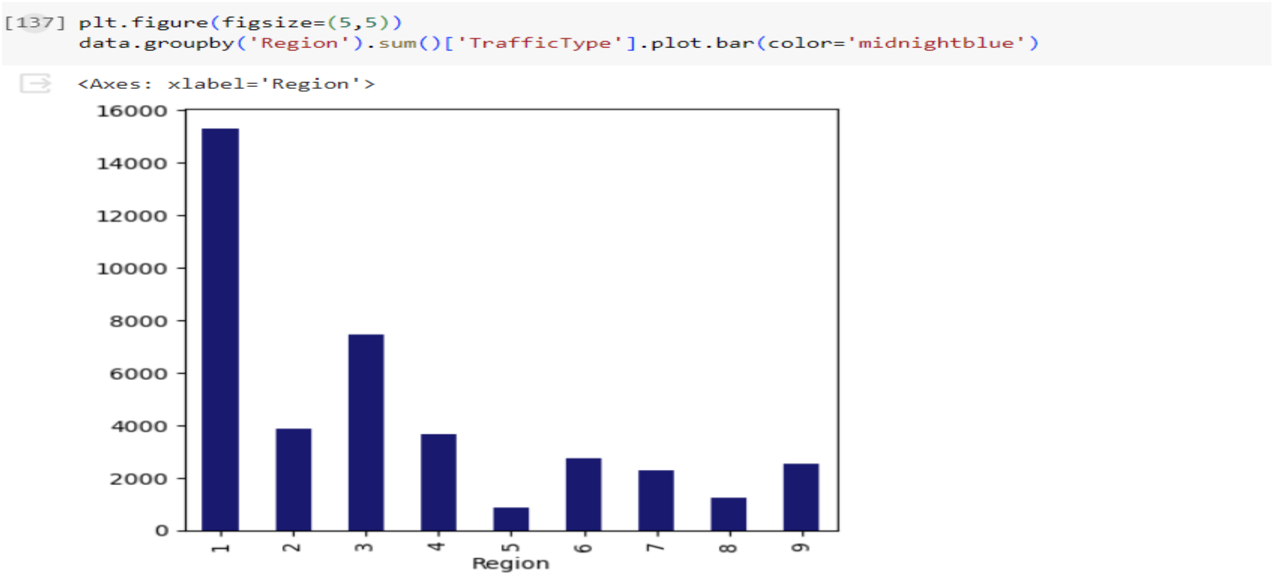
Inference: This graph shows that Exit Rate is more during the weekdays, it can be said that customers prefer to do shopping during weekends.



```
init_notebook_mode(connected=False)
fig = px.histogram(data, x="SpecialDay", y="VisitorType",color='Revenue',title = 'Online Shopping')
fig.show()
```

Fig 2: Histogram showing relation between Special Day and Visitor type with respect to Revenue.

Inference: This graph shows that the most visitors are the returning ones, and they prefer to do shopping during the special days such as mother’s day, valentines day etc.



```
[137] plt.figure(figsize=(5,5))
      data.groupby('Region').sum()['TrafficType'].plot.bar(color='midnightblue')
```

```
<Axes: xlabel='Region'>
```

Fig 3: Bar Graph showing type of traffic in specific Region.

Inference: This graph shows that the traffic is highest in Region 1and lowest in Region 5.

Visualization through Power BI:

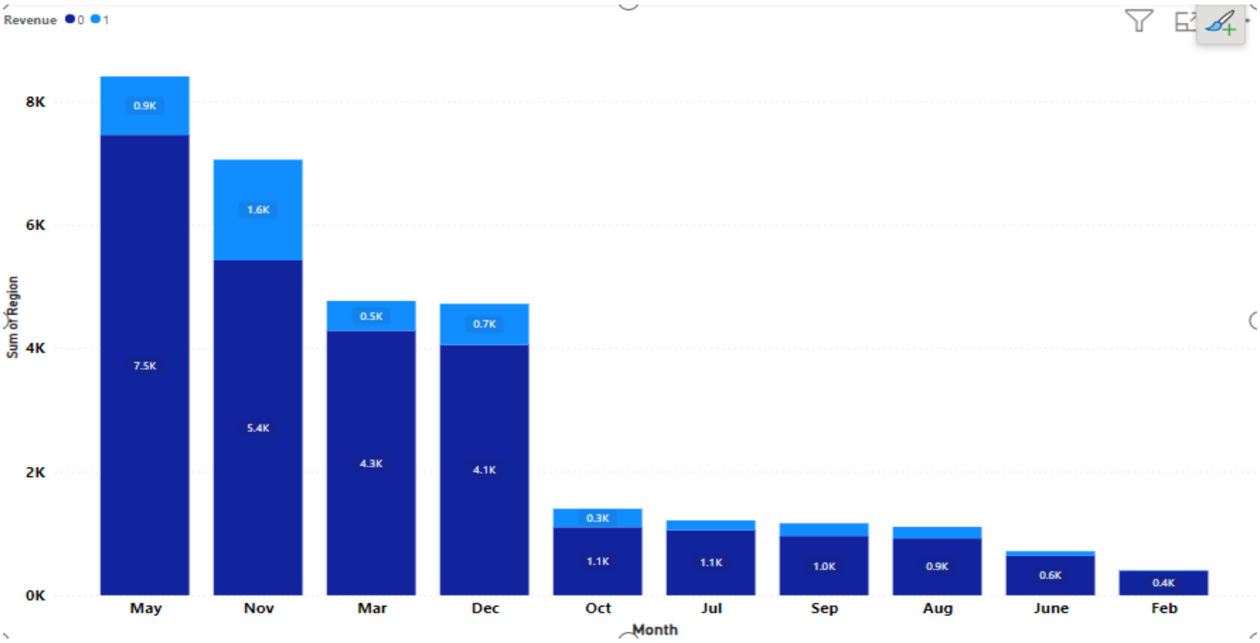


Fig 4: Stacked Column chart showing month wise revenue in regions.

Inference: This chart shows that the revenue generated was highest in the month of November and lowest was in the month of June whereas there is no revenue in the month of February (shown in Fig 4.1 below).

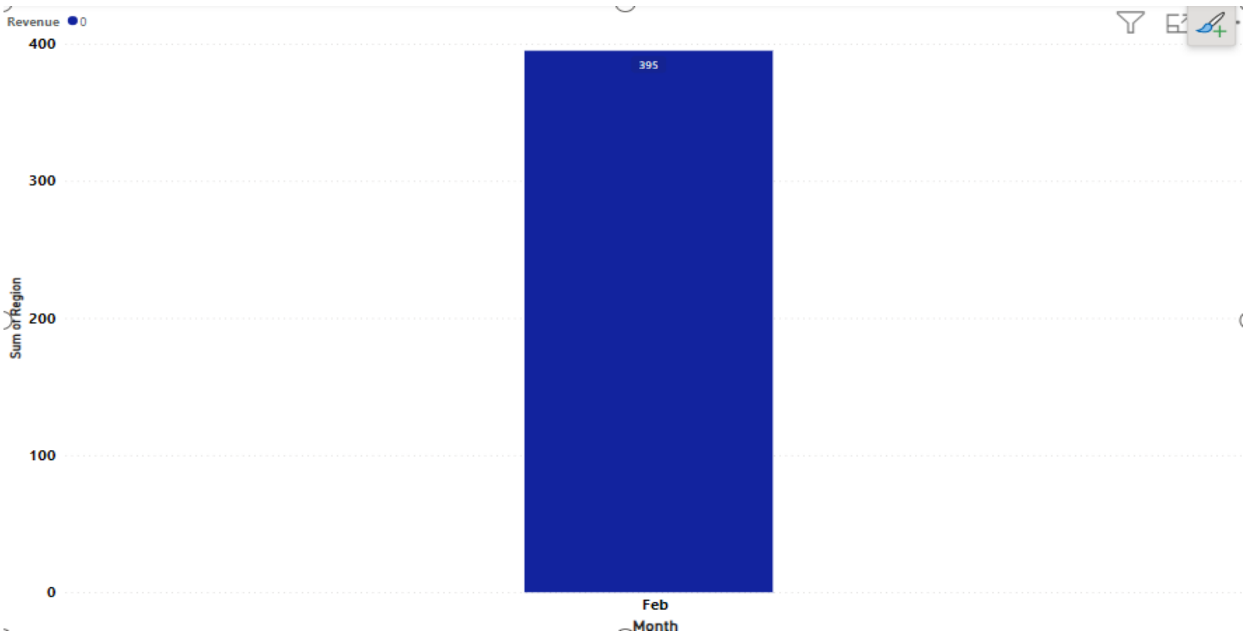


Fig 4.1: No Revenue in the month of February.

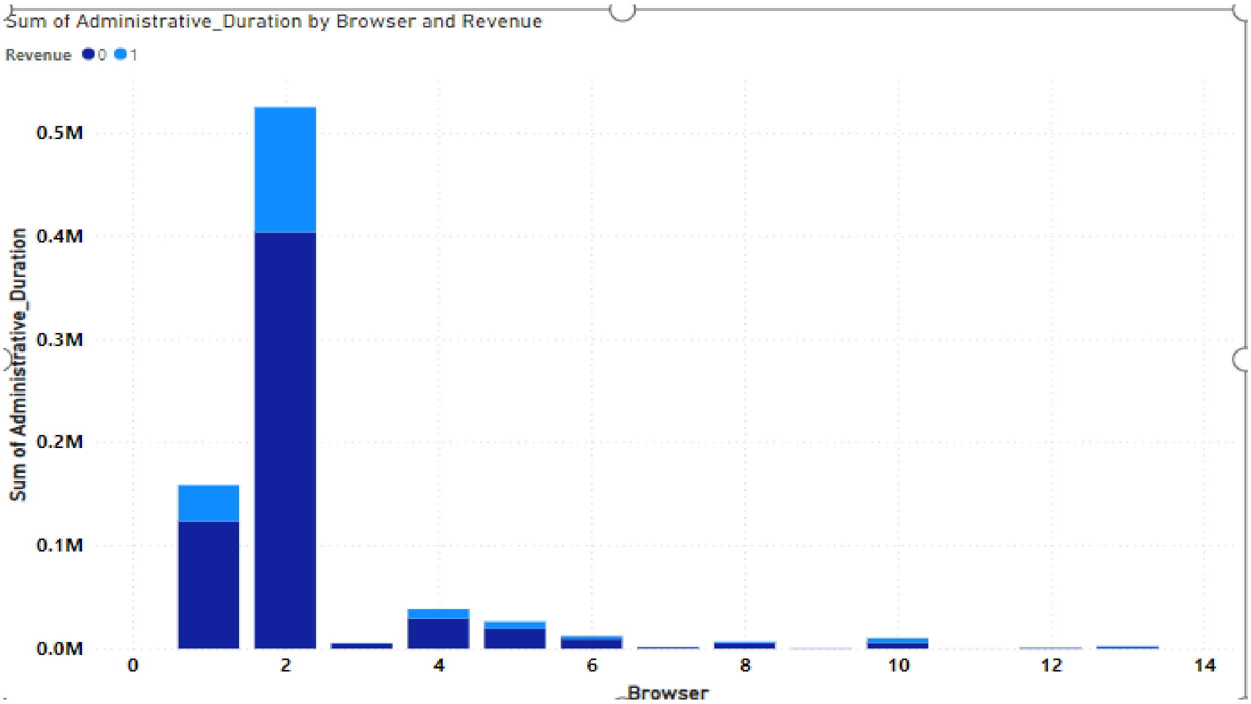


Fig 5: Stacked Column chart showing Browser used time spent on Administrative page with respect to Revenue.

Inference: This chart shows that the revenue generated was highest with the use of browser no. 2, followed by browser no. 1.

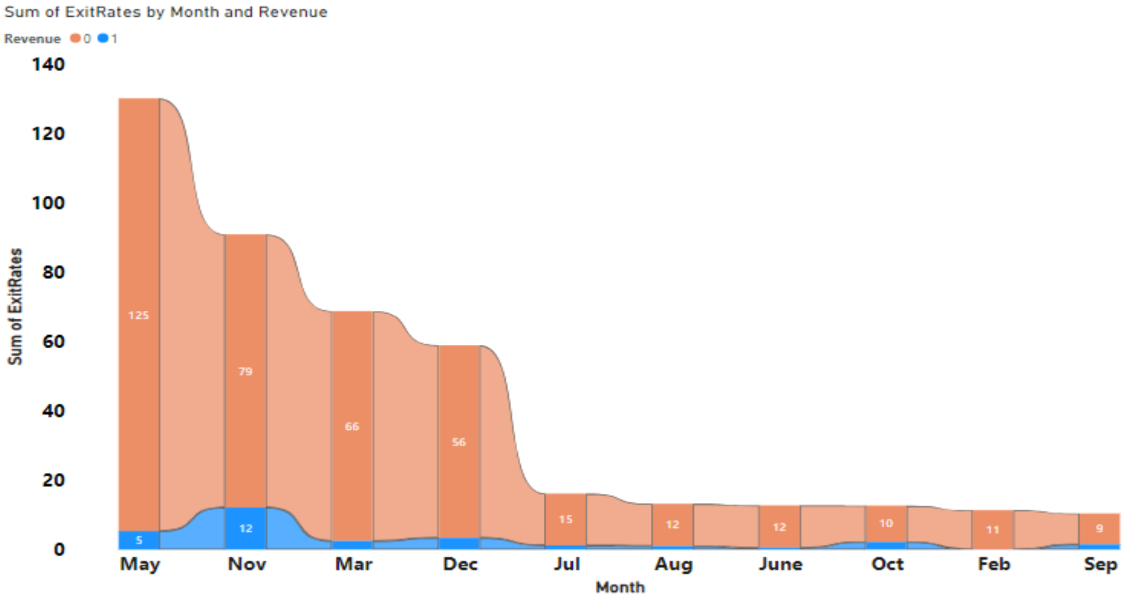


Fig 6: Ribbon chart showing Exit Rate month wise with respect to Revenue.

Inference: This chart shows that the revenue generated was highest in the month of November whereas the Exit rate was greater in May month and lowest in September month.

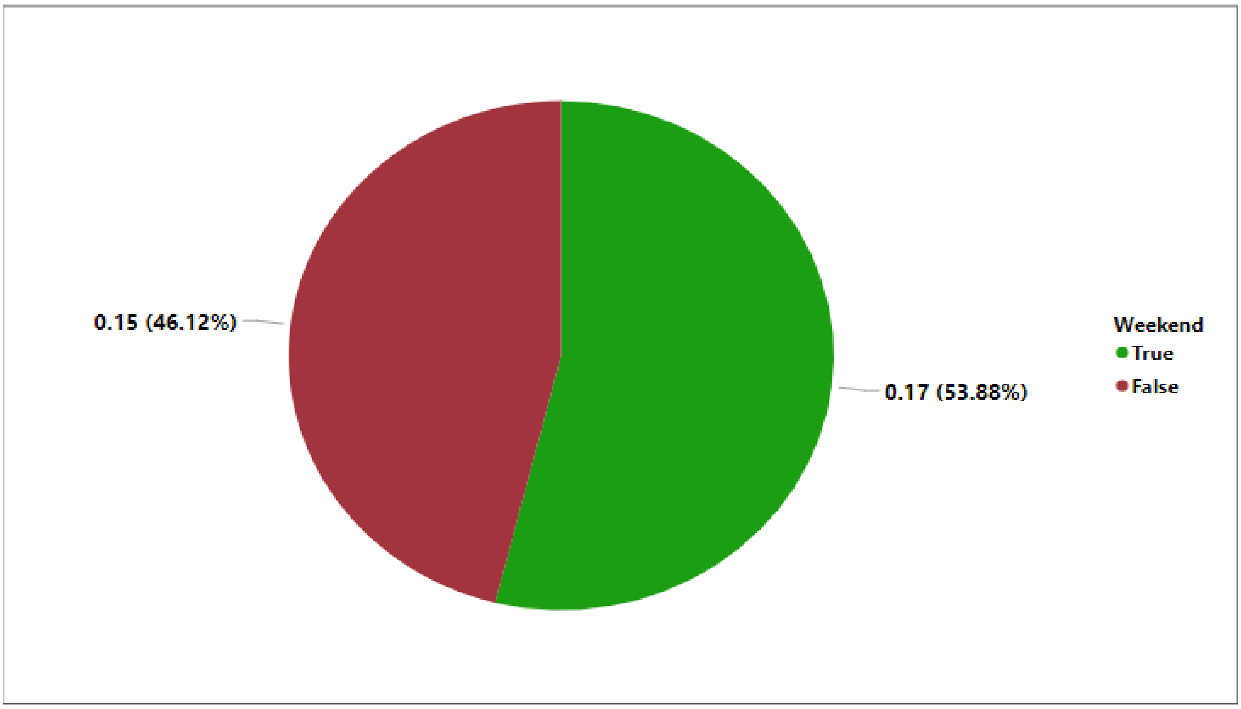


Fig 7: Pie chart showing Revenue with respect to weekend.

Inference: This chart shows that the revenue generated is more in weekends (53.88%) as compared to weekdays (46.12%).

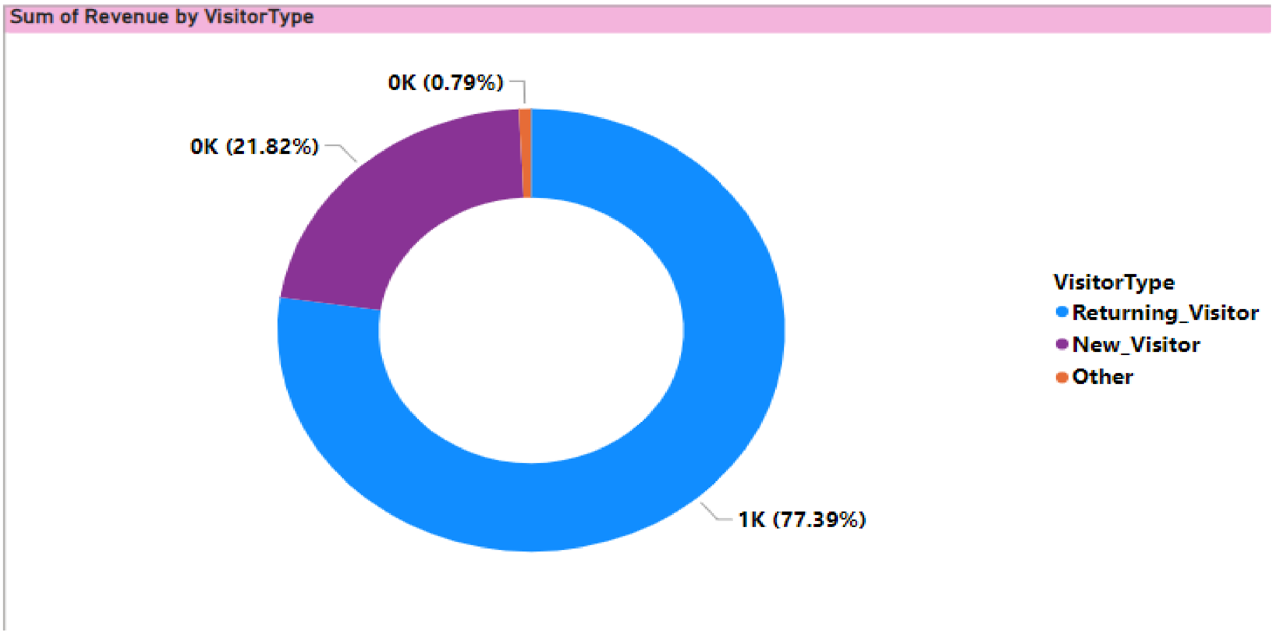


Fig 8: Donut chart showing Revenue with respect to visitor type.

Inference: This chart shows that the revenue is generated more from the returning visitors, following new visitor and is very less by other types of visitors.

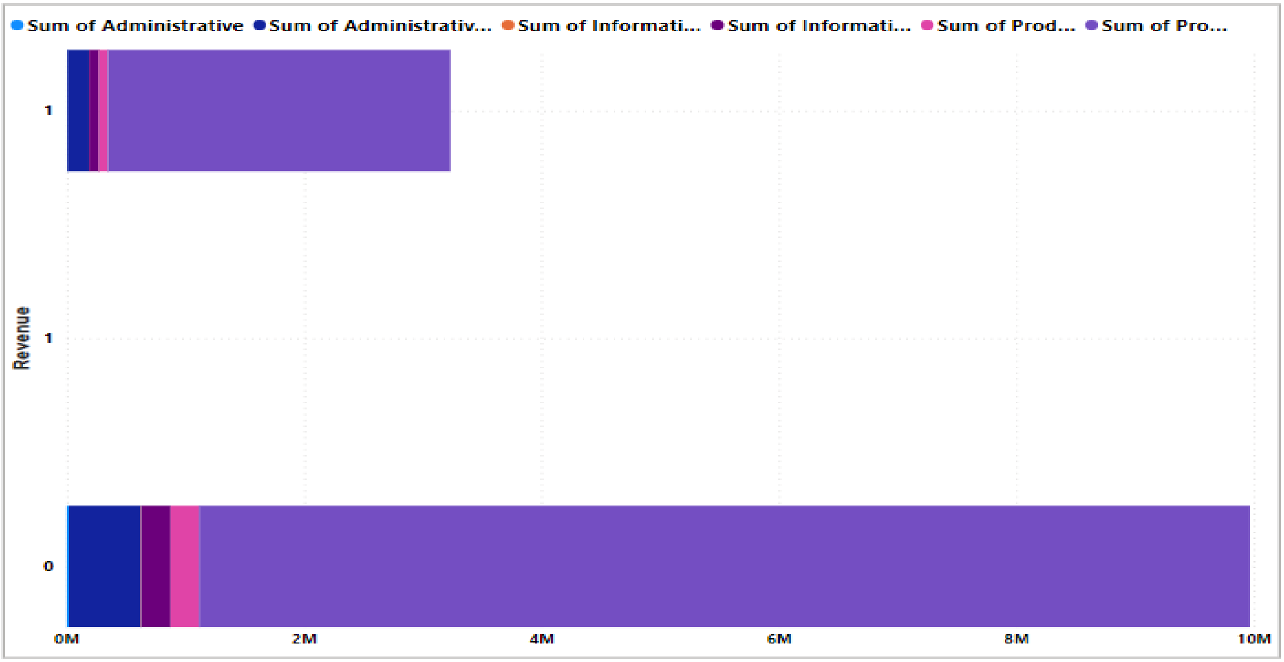
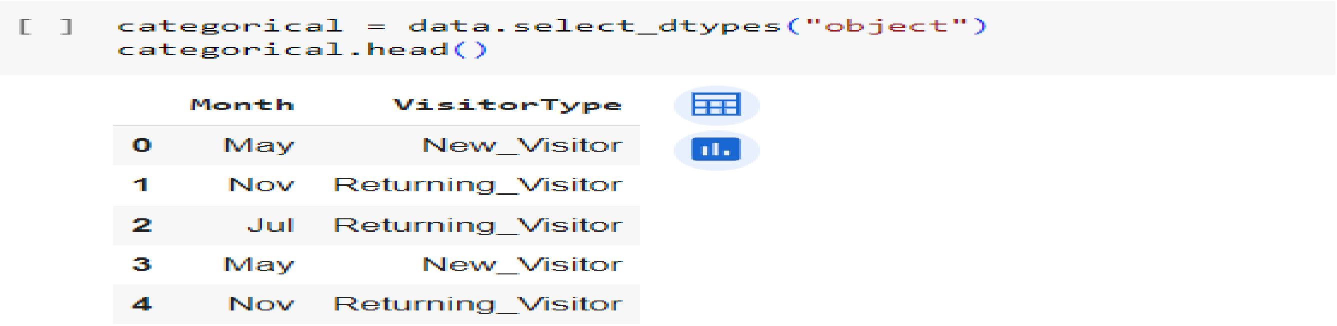


Fig 9: Stacked Bar chart showing Revenue with respect to types of pages visited by the visitor in that session and total time spent in each of these page categories.

Inference: This chart shows that user spends more time in the pages related to products they want to purchase and more revenue is generated from those pages only.

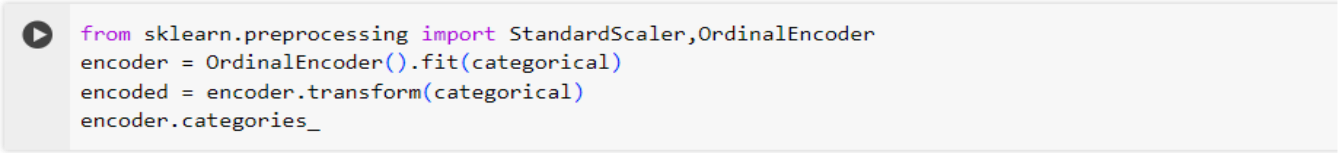
One-hot encoding:

Using Ordinal Encoder:

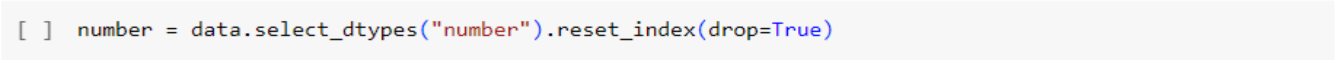


```
[ ]  categorical = data.select_dtypes("object")
     categorical.head()
```

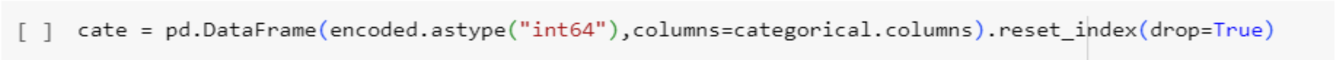
	Month	VisitorType
0	May	New_Visitor
1	Nov	Returning_Visitor
2	Jul	Returning_Visitor
3	May	New_Visitor
4	Nov	Returning_Visitor



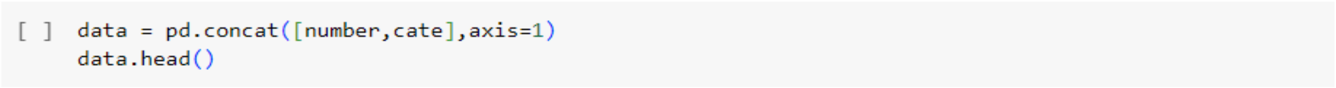
```
from sklearn.preprocessing import StandardScaler,OrdinalEncoder
encoder = OrdinalEncoder().fit(categorical)
encoded = encoder.transform(categorical)
encoder.categories_
```



```
[ ]  number = data.select_dtypes("number").reset_index(drop=True)
```

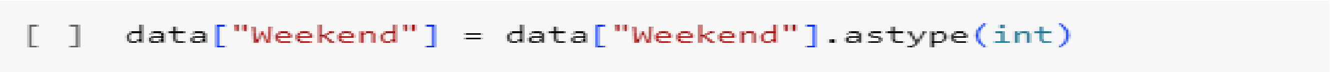


```
[ ]  cate = pd.DataFrame(encoded.astype("int64"),columns=categorical.columns).reset_index(drop=True)
```



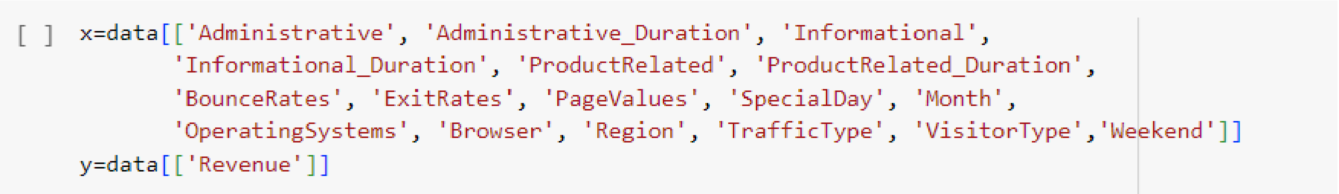
```
[ ]  data = pd.concat([number,cate],axis=1)
     data.head()
```

Dealing with Boolean Datatype in the Dataset:



```
[ ]  data["Weekend"] = data["Weekend"].astype(int)
```

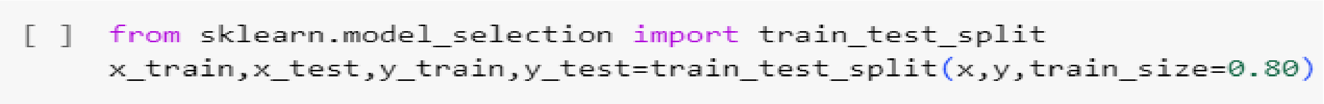
Splitting Data in to Independent and Dependent Columns.



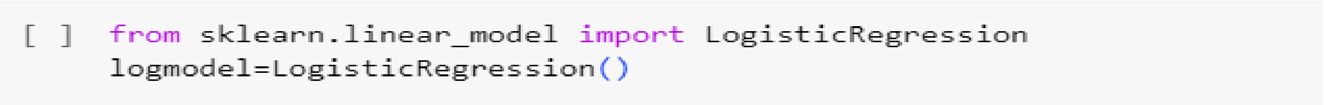
```
[ ]  x=data[['Administrative', 'Administrative_Duration', 'Informational',
            'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
            'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
            'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType','Weekend']]
     y=data[['Revenue']]
```

Creating Machine Learning Model.

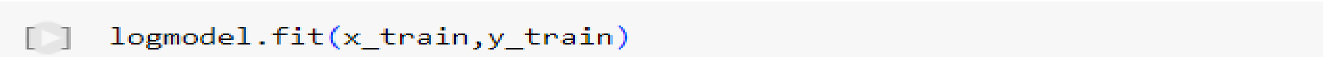
1.Using Logistic Regression Classifier:



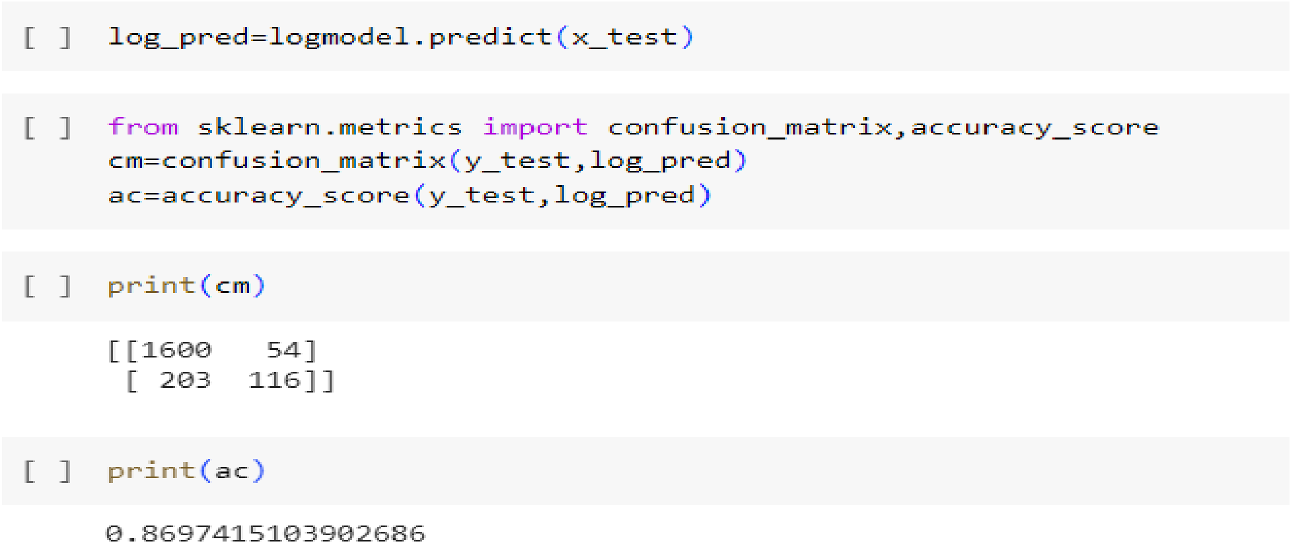
```
[ ]  from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.80)
```



```
[ ]  from sklearn.linear_model import LogisticRegression
     logmodel=LogisticRegression()
```



```
[ ]  logmodel.fit(x_train,y_train)
```



```
log_pred=logmodel.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,log_pred)
ac=accuracy_score(y_test,log_pred)
```

```
print(cm)
```

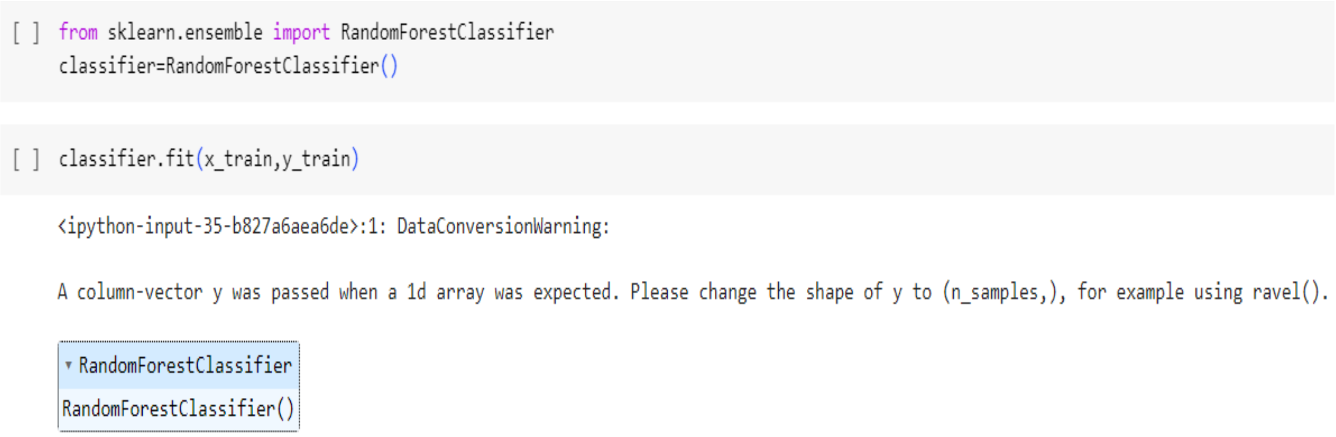
```
[[1600   54]
 [ 203  116]]
```

```
print(ac)
```

```
0.8697415103902686
```

Accuracy Score is 86%.

1.Using Random Forest Classifier:

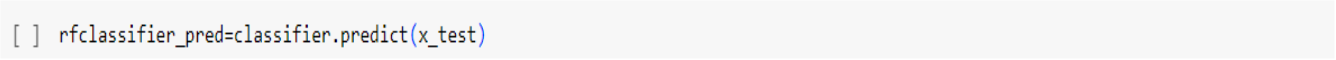


```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier()
```

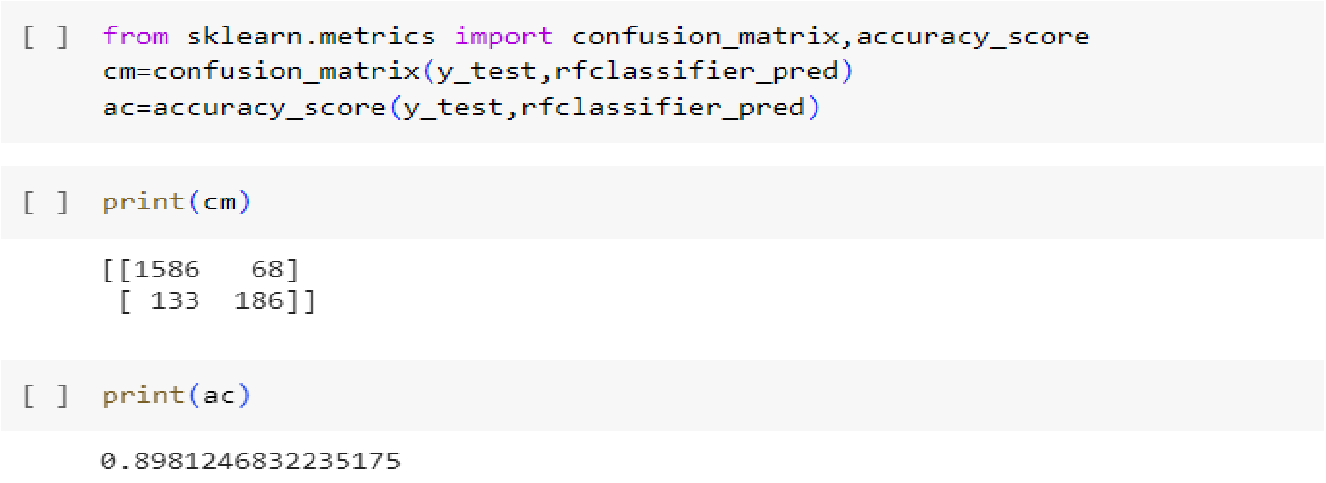
```
classifier.fit(x_train,y_train)
```

```
<ipython-input-35-b827a6aea6de>:1: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```



```
rfclassifier_pred=classifier.predict(x_test)
```



```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,rfclassifier_pred)
ac=accuracy_score(y_test,rfclassifier_pred)
```

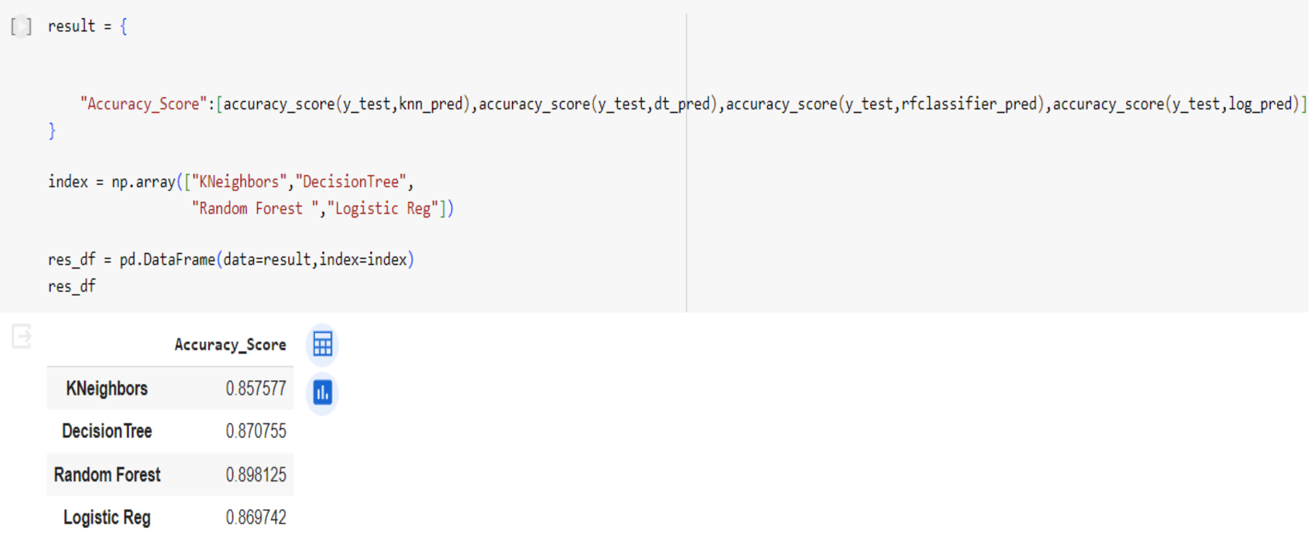
```
print(cm)
```

```
[[1586   68]
 [ 133  186]]
```

```
print(ac)
```

```
0.8981246832235175
```

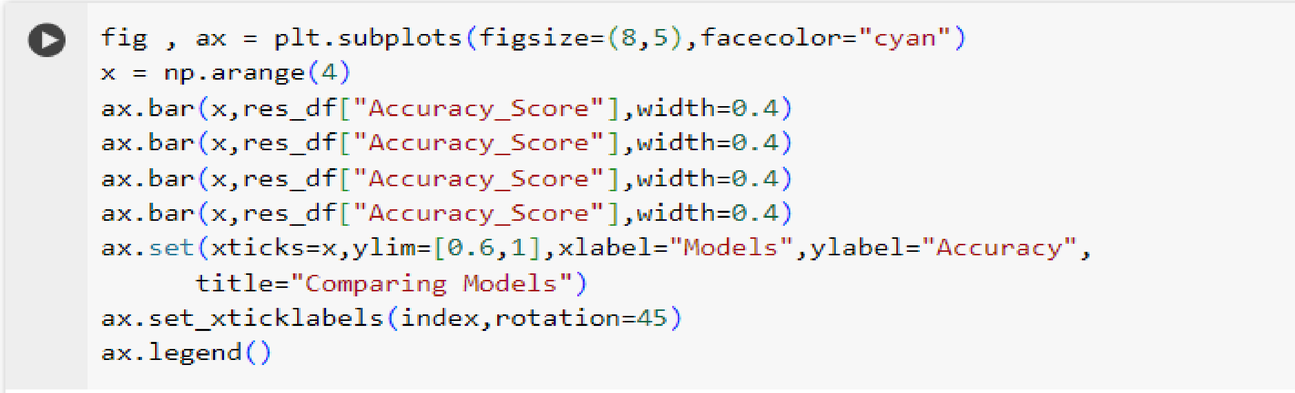
Calculated Accuracy for different models and assembling them in a Data frame:



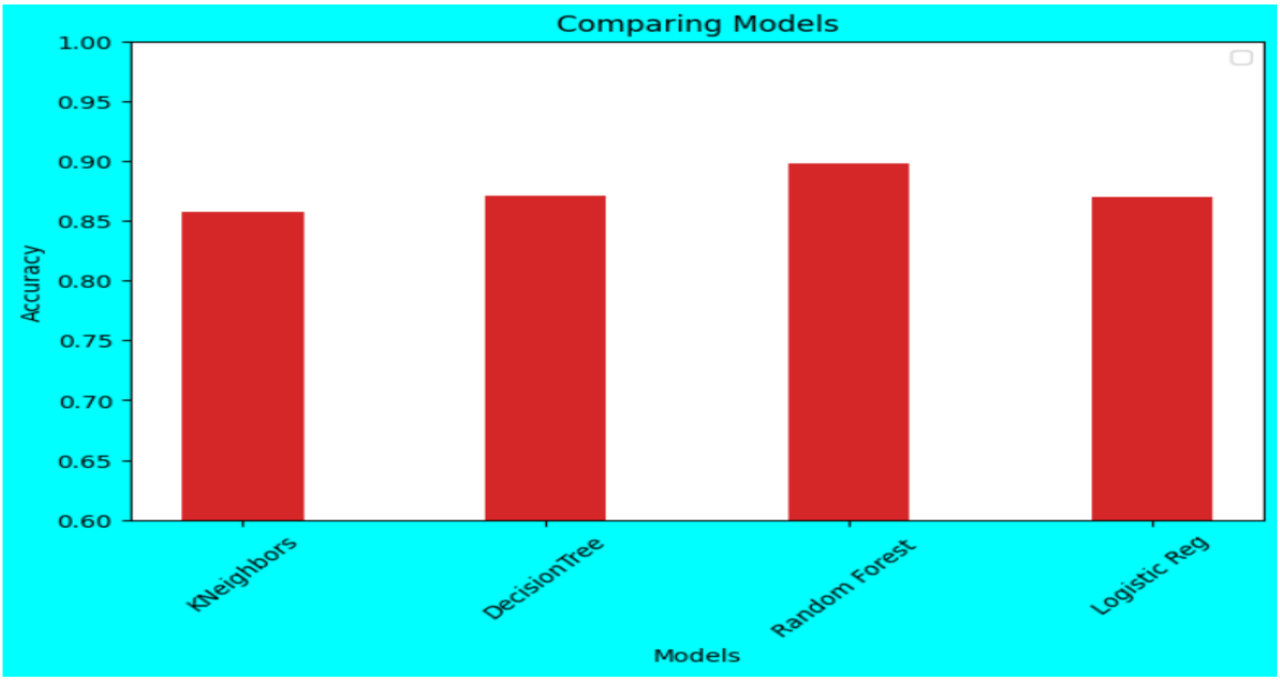
```
[ ] result = {

        "Accuracy_Score":[accuracy_score(y_test,knn_pred),accuracy_score(y_test,dt_pred),accuracy_score(y_test,rfclassifier_pred),accuracy_score(y_test,log_pred)]
    }

    index = np.array(["KNeighbors","DecisionTree",
                      "Random Forest ","Logistic Reg"])

    res_df = pd.DataFrame(data=result,index=index)
    res_df
```

	Accuracy_Score
KNeighbors	0.857577
DecisionTree	0.870755
Random Forest	0.898125
Logistic Reg	0.869742

Graphical Representation of Accuracy scores of different ML Models:

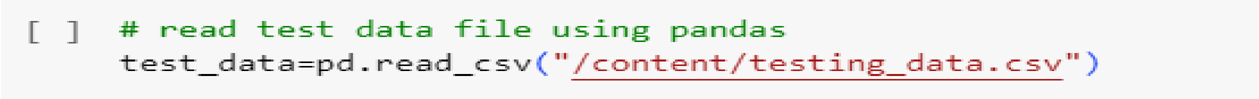


```
fig , ax = plt.subplots(figsize=(8,5),facecolor="cyan")
x = np.arange(4)
ax.bar(x,res_df["Accuracy_Score"],width=0.4)
ax.bar(x,res_df["Accuracy_Score"],width=0.4)
ax.bar(x,res_df["Accuracy_Score"],width=0.4)
ax.bar(x,res_df["Accuracy_Score"],width=0.4)
ax.set(xticks=x,ylim=[0.6,1],xlabel="Models",ylabel="Accuracy",
      title="Comparing Models")
ax.set_xticklabels(index,rotation=45)
ax.legend()
```

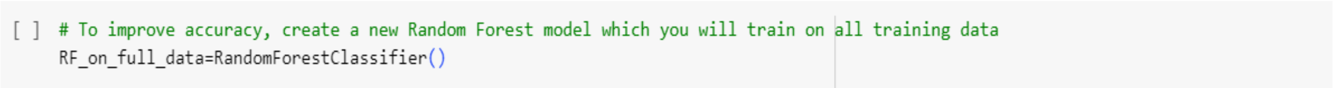


Using Test Dataset:

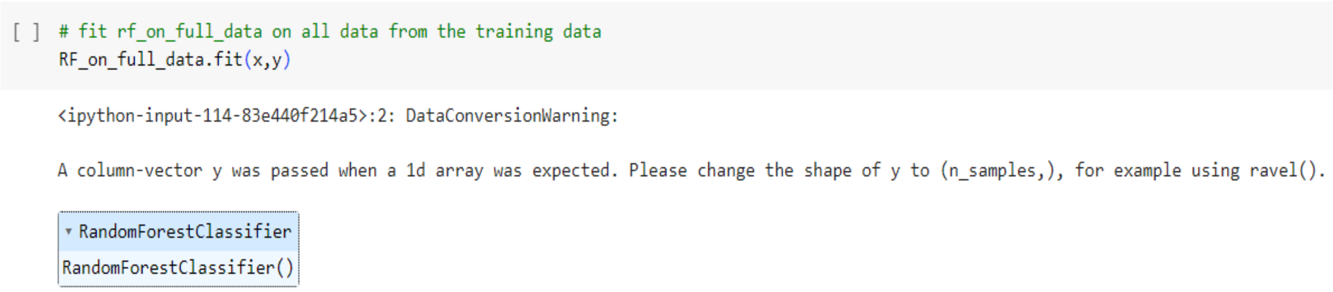
We have done prediction on test data using Random Forest Classifier as it is giving highest accuracy 89%.



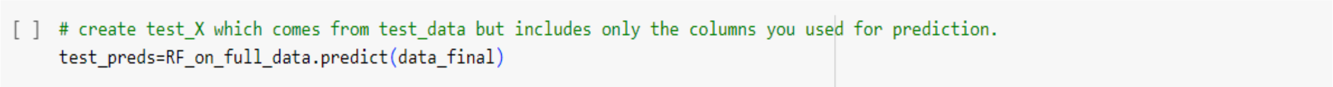
```
[ ]  # read test data file using pandas
     test_data=pd.read_csv("/content/testing_data.csv")
```



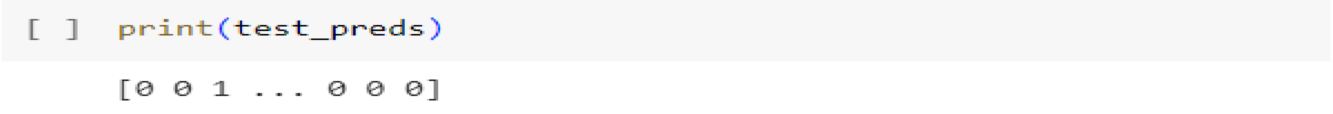
```
[ ]  # To improve accuracy, create a new Random Forest model which you will train on all training data
     RF_on_full_data=RandomForestClassifier()
```



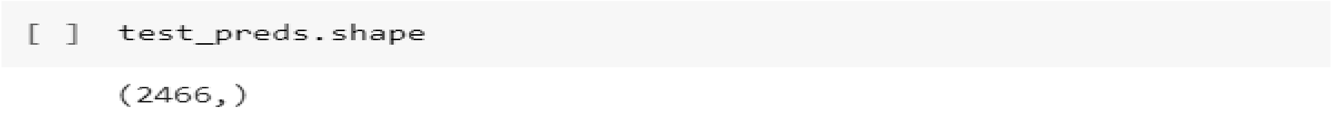
```
[ ]  # fit rf_on_full_data on all data from the training data
     RF_on_full_data.fit(x,y)

     <ipython-input-114-83e440f214a5>:2: DataConversionWarning:

     A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

     ▾ RandomForestClassifier
     RandomForestClassifier()
```



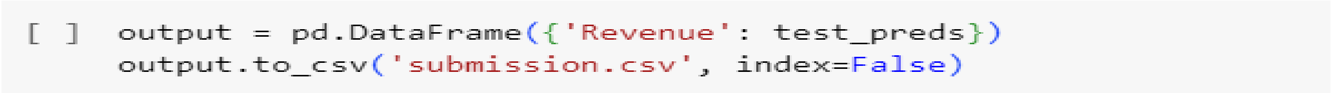
```
[ ]  # create test_X which comes from test_data but includes only the columns you used for prediction.
     test_preds=RF_on_full_data.predict(data_final)
```



```
[ ]  print(test_preds)

     [0 0 1 ... 0 0 0]
```

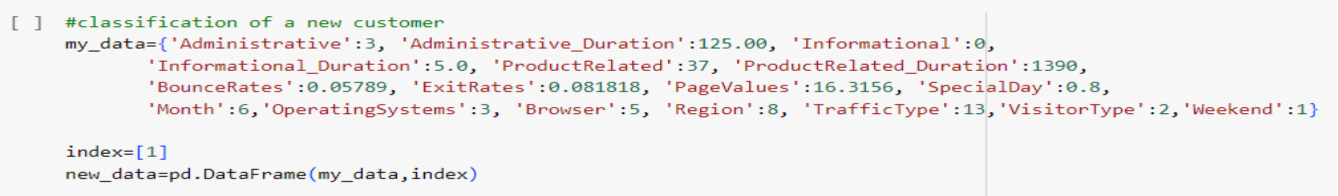


```
[ ]  test_preds.shape

     (2466,)
```

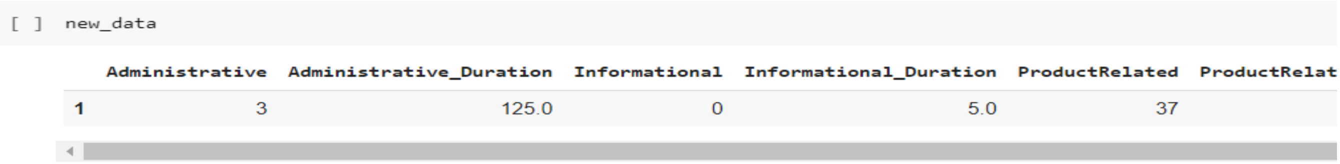


```
[ ]  output = pd.DataFrame({'Revenue': test_preds})
     output.to_csv('submission.csv', index=False)
```

Classification of a new Customer and making predictions using Logistic Regression-

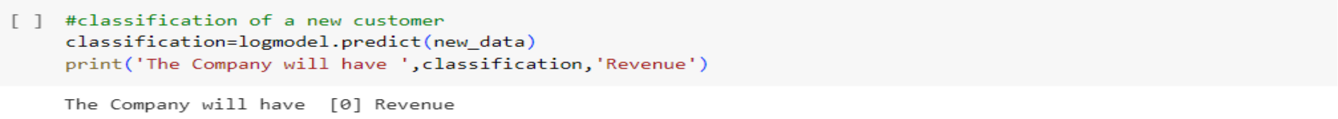


```
[ ]  #classification of a new customer
     my_data={'Administrative':3, 'Administrative_Duration':125.00, 'Informational':0,
             'Informational_Duration':5.0, 'ProductRelated':37, 'ProductRelated_Duration':1390,
             'BounceRates':0.05789, 'ExitRates':0.081818, 'PageValues':16.3156, 'SpecialDay':0.8,
             'Month':6,'OperatingSystems':3, 'Browser':5, 'Region':8, 'TrafficType':13,'VisitorType':2,'Weekend':1}

     index=[1]
     new_data=pd.DataFrame(my_data,index)
```



```
[ ]  new_data
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelat
1	3	125.0	0	5.0	37	



```
[ ]  #classification of a new customer
     classification=logmodel.predict(new_data)
     print('The Company will have ',classification,'Revenue')

     The Company will have  [0] Revenue
```

CONCLUSION: The online shopping data set was used to implement prediction and classification algorithms. We also visualised different features present in the dataset. Through online shopping customer experience faster buying process, flexibility, there are no reach limitations, they can explore and can-do product and price comparison.

Here, we have predicted whether the customer will purchase the product or not looking at the features present in the given dataset. Revenue is generated specially in weekends or on special days and mostly by the returning visitors. Also, time spent by the customers on the pages related to product related information, Informational duration or administrative duration has also been observed to be useful in predicting the future purchase by customers.