University of Texas at Arlington
INSY 5377
Final Project Report

Online Shoppers Purchasing Intention

['Pradeep Medagiri]'

Overview:

Introduction:

In the research paper, Dataset is taken from UCI repository "*Online Shoppers Purchasing Intention*" consists of web data with 10 numerical features and 8 Categorical features having 12330 sessions. Each session belongs to a different user in 1-year period. From the dataset: 10422 sessions are False (did not complete shopping) and 1908 sessions are True (completed shopping). Each feature has unique importance and the research paper extracts all information from web data. And this information further helps to choose a model to determine shoppers purchasing intention. Data visualization techniques are used to recommend changes or new implementations. Best classification models such as logistics regression, random forest classifier, decision tree classifier and gradient boosting classifier are selected that suits the dataset to predict visitors intention of making a purchase.

Source:https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset#

Dataset Description:

Numerical Features(10):

Administrative -
Number of pages visited by visitor to know or look into administration.[1]

Administrative duration:
Total time or duration on administrative page.[1]

Informational:
Number of pages visited by visitor to get information[1]

Informational duration:
Total time or duration on information page.[1]

Product related:
Number of pages visited by visitor to get product information.[1]

Product related duration:
Total time or duration on Product related page.[1]

Bounce rate:
The value of "Bounce Rate" feature for a web page refers to the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session [1]

Exit rate:
The value of "Exit Rate" feature for a specific web page is calculated as for all pageviews to the page, the percentage that were the last in the session [1]

Page value:
The "Page Value" feature represents the average value for a web page that a user visited before completing an e-commerce transaction [1]

Special day:
The "Special Day" feature indicates the closeness of the site visiting time to a specific special day (e.g. Mother's Day, Valentine's Day) in which the sessions are more likely to be finalized with transaction. [1]

Categorical Features(8):

Operating Systems:
Type of operating system used by user for the session.[1]

Browser:
Browser used by user for the session.[1]

Region:
Region from where session recorded.[1]

Traffic Type:
Different sources used by using to land on website.[1]

Visitor Type:
Type of visitor [new visitor, returning visitor, others].[1]

Weekend:
Boolean value indicating whether the date of the visit is weekend or not. [1]

Month:
Month of the year

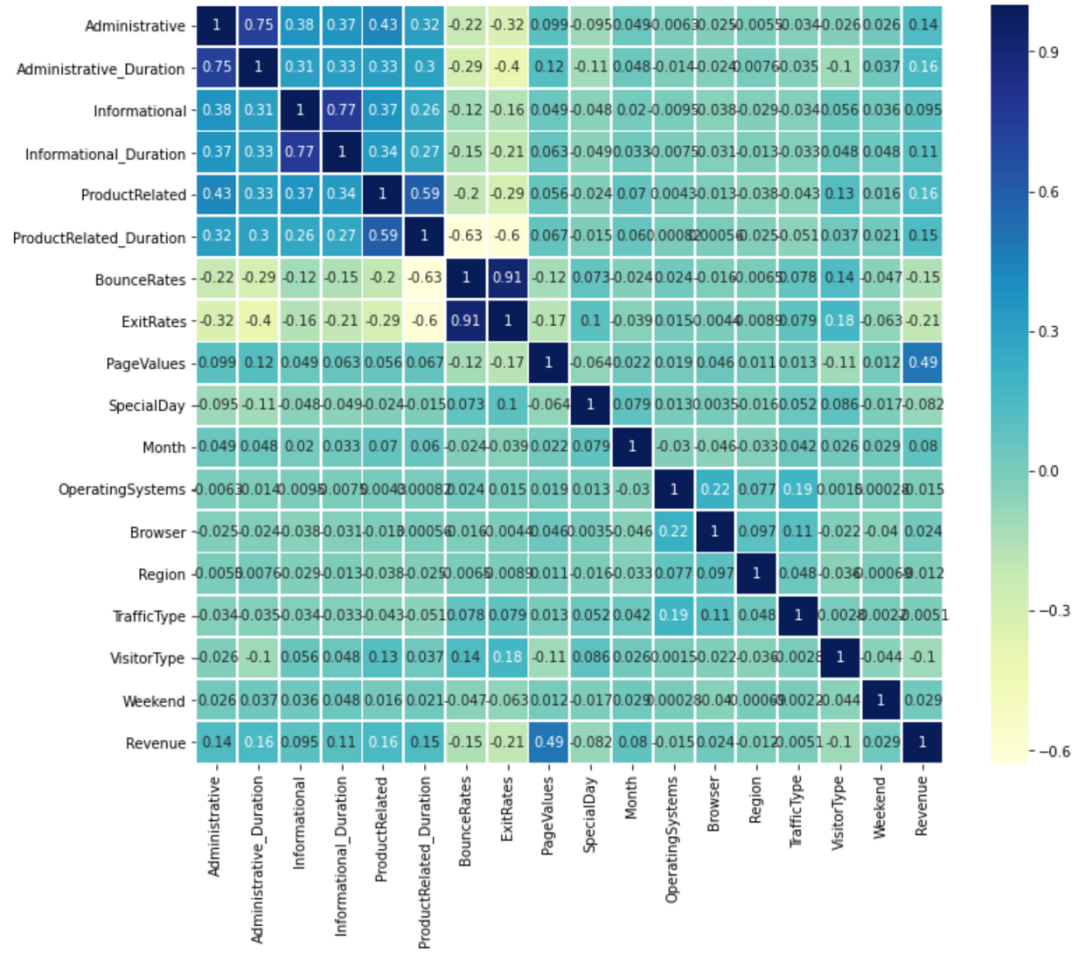
Revenue:
Boolean value indicating whether visitor purchased or not. [1]

Questions:
1. To plot web data and see how all attributes are varying.
2. To check factors effecting important attributes.
3. Build a model to predict - either customer will complete a purchase or not (revenue).
4. Propose a model best suits to the dataset and recommend necessary changes if required (that increases revenue).

Data visualization:

Correlation matrix:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc371bf9c88>
```

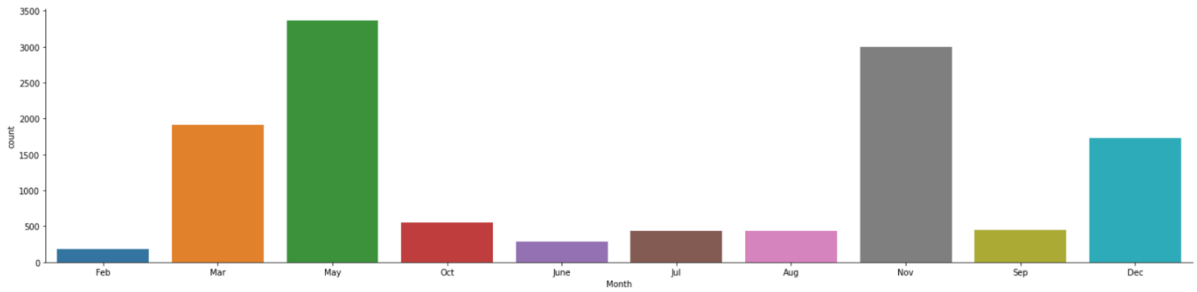


- From the above graph: Administrative_Duration, Informational_Duration, ProductRelated_Duration are highly correlated with relative columns.
- Other is PageValues and Revenue, Pagevalues represents values of last page before making a transaction. So undoubtedly Pagevalues are doing good in terms of revenue.

Univariate plots:

1. Month

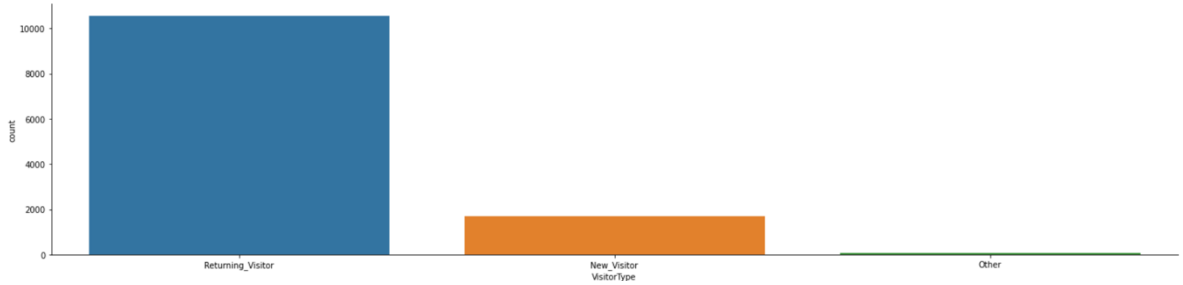
```
<seaborn.axisgrid.FacetGrid at 0x7fc3a3904d68>
```



- May, November, March, December are considered as peak months that has to be noted.

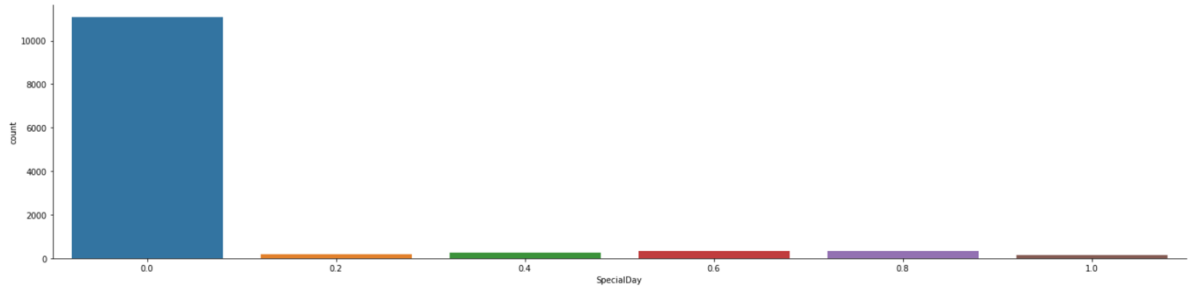
2. Visitor Type

```
<seaborn.axisgrid.FacetGrid at 0x7fc3a619b2b0>
```



3. Special Day

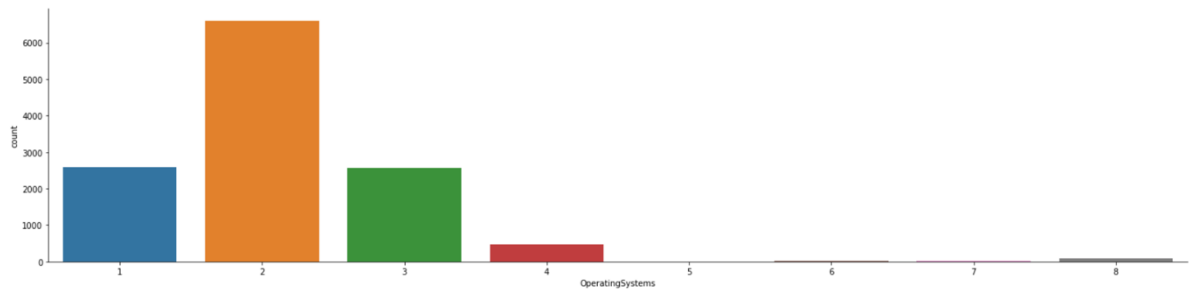
```
<seaborn.axisgrid.FacetGrid at 0x7fc371efa208>
```



- Usually on special days, flow of visitors has to be huge but not too many visitors are seen (has to be noted).

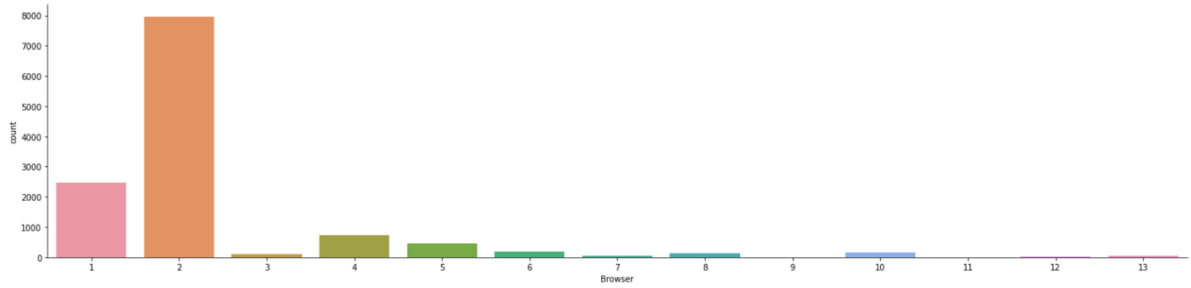
4. Operating Systems:

```
<seaborn.axisgrid.FacetGrid at 0x7fc37293fcc0>
```



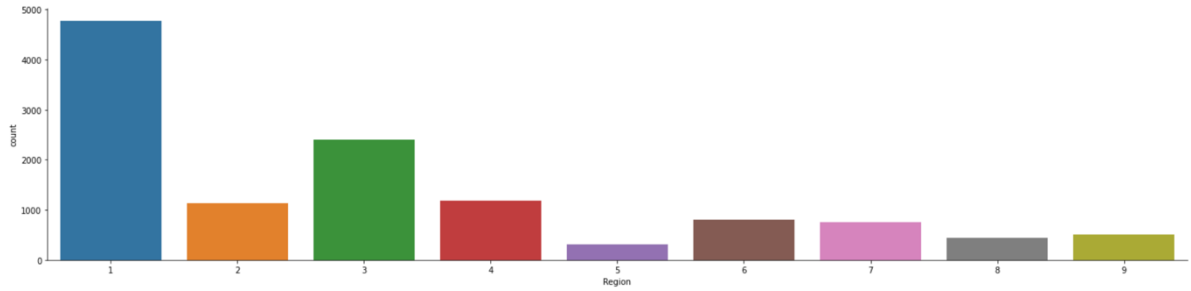
5. Browser:

```
<seaborn.axisgrid.FacetGrid at 0x7fc372bd4c50>
```



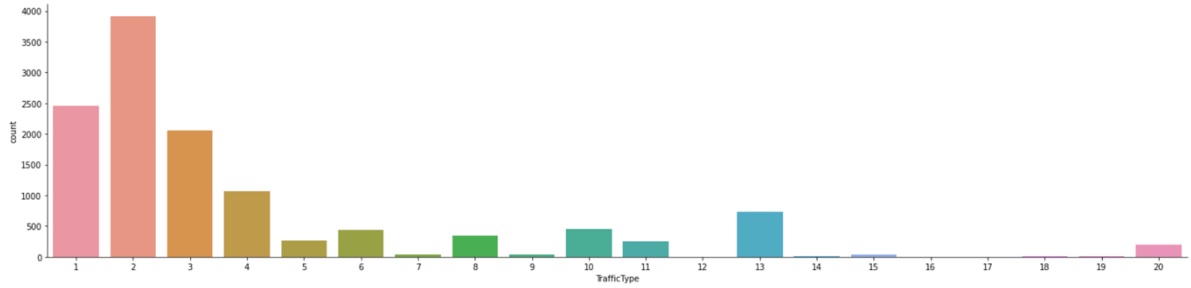
6. Region:

```
<seaborn.axisgrid.FacetGrid at 0x7fc372c62518>
```



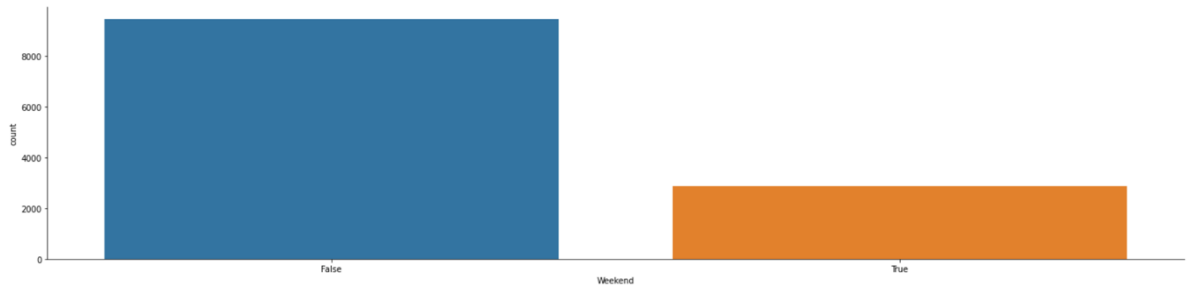
7. Traffic Type:

```
<seaborn.axisgrid.FacetGrid at 0x7fc3729a0ac8>
```



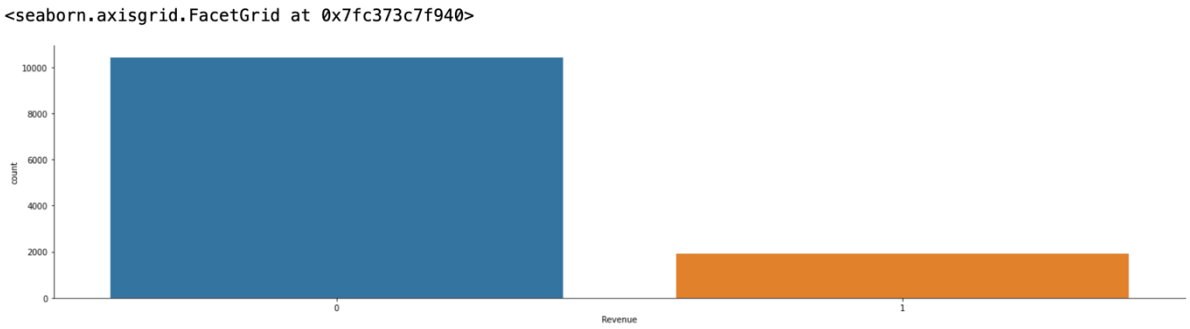
8. Weekend:

```
<seaborn.axisgrid.FacetGrid at 0x7ff2e840abe0>
```



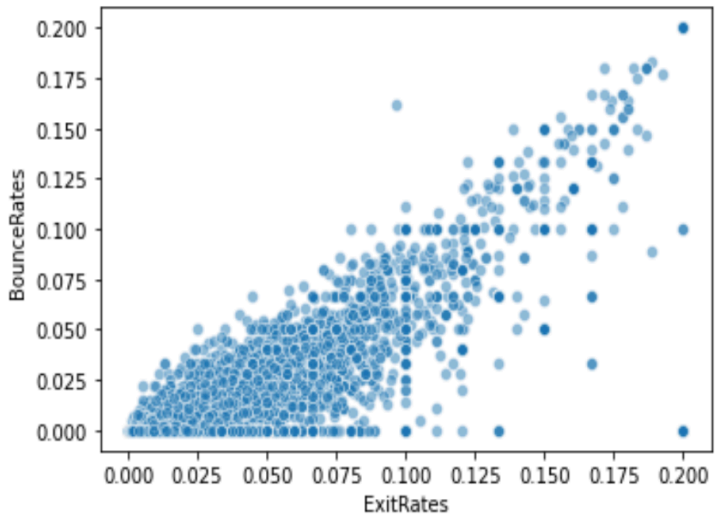
- Visiting rate is less on weekend's

9. Revenue:



Bivariate Plots:

1. ExitRates vs BounceRates

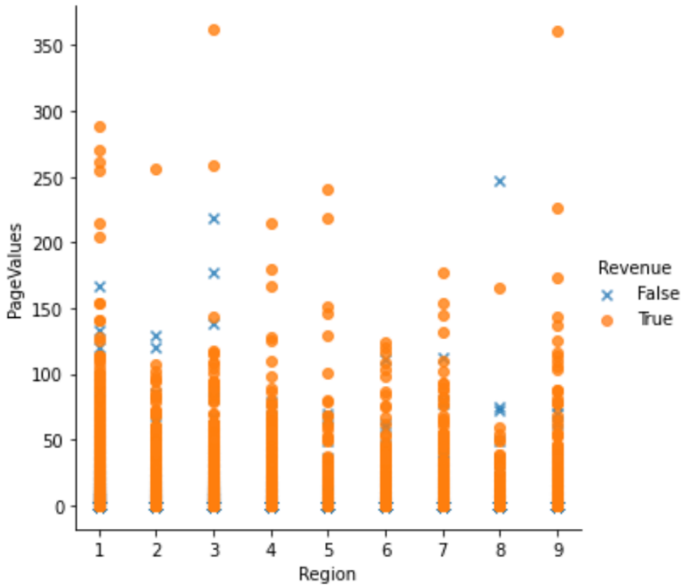


- **Recommendations:**
  1. To convert visitors (BounceRates and ExitRates), it is recommended to create user friendly platform displaying all information required to make a purchase such as discounts, coupons, shipping charges.
  2. Every user who shop online may not be willing to create online account. So, for those customers if an option "continue as guest" will be helpful to make them purchase. Later, company will have opportunity to contact customer via email with discounts, offers and referrals.

Multivariate Plots:

1. Region vs Page values

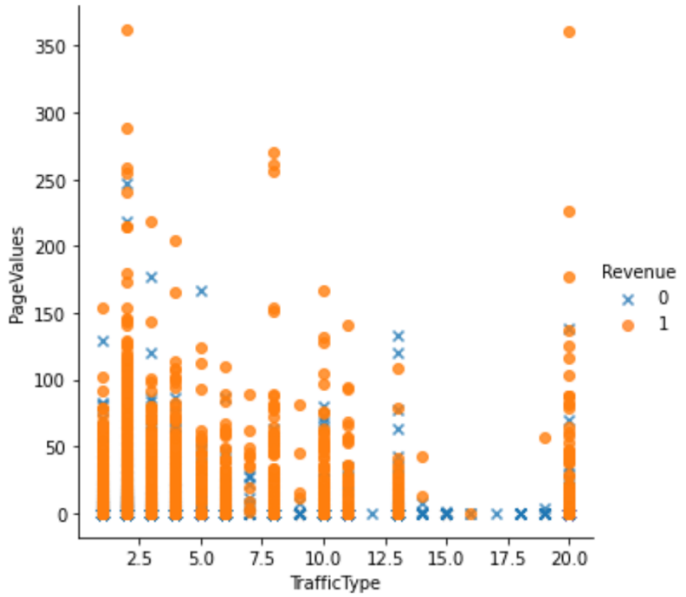
```
<seaborn.axisgrid.FacetGrid at 0x7ff2ccf6d5f8>
```



- **Recommendations:**
  3. Region wise Revenue is good in terms of PageValues. There as false revenue's in increase in pagevalues and that has to be look into it. On the page: introducing information of the products in cart, options to check availability of product, estimated shipping charges, tax's and estimated date of arrival helps visitor to make decision (allowing them not to bounce back or exit).

2. TrafficType vs PageValues

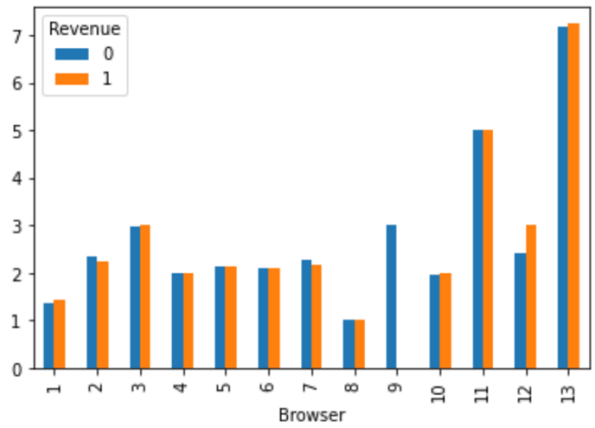
```
<seaborn.axisgrid.FacetGrid at 0x7fc36cae37b8>
```



- **Recommendations:**
  4. Pagevalues are doing good in almost all traffic types 15 out of 20. There are couples of types that team has to look into. These users comes from different sources, so Highlighting or directing them to right pages in website is first step to Convert them to make purchase.

3. Browser vs OperatingSystems

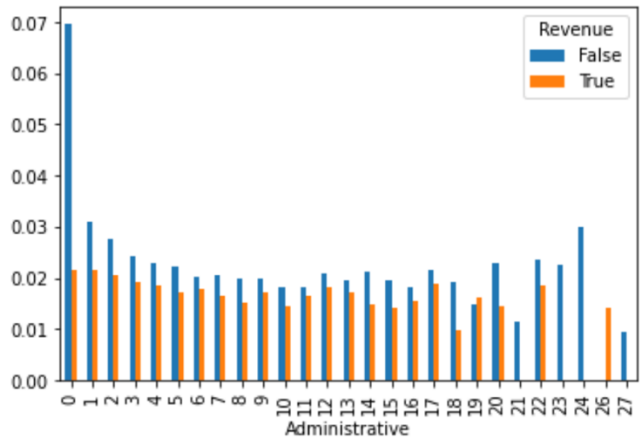
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc36d0c6f98>
```



- **Recommendations:**
  5. Now looking at browsers used by all users, there are only 50% of users making purchase. So, changes or modifications on webpages to support all types of browser by making them user friendly helps better getting them to last page of placing order.

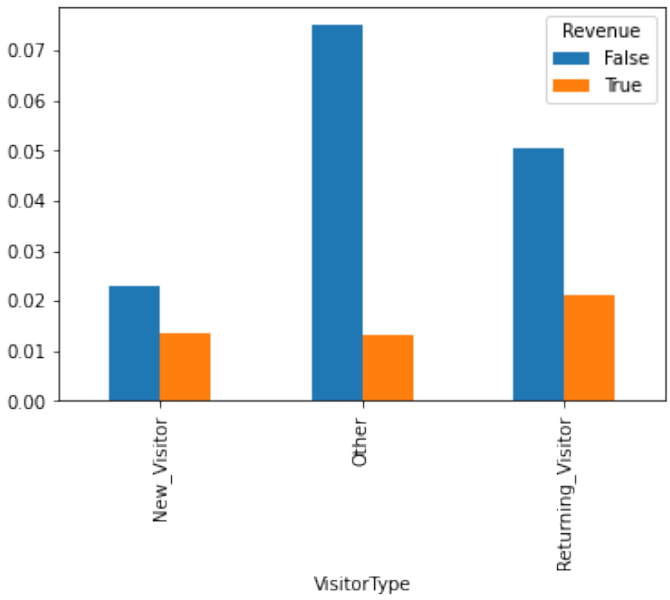
4. Administrative vs ExitRates

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc38e41b5f8>
```



- False rate 'decreased' when different pages visited to know about administration that means administrative service is doing good.

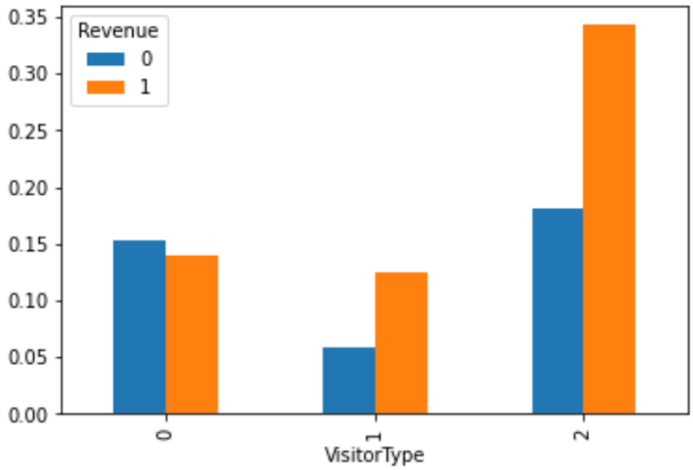
5. VisitorType vs ExitRates



- **Recommendations:**
  6. 0 is Returning users, 1 is new users, 2 is other users. Since there are less percent of new users ending up with purchase, team has to focus on new users by offering first time user discounts (by pop-ups, ad's). on the hand, improving recommendation system for returning users makes them easy to shop ending up with purchase.

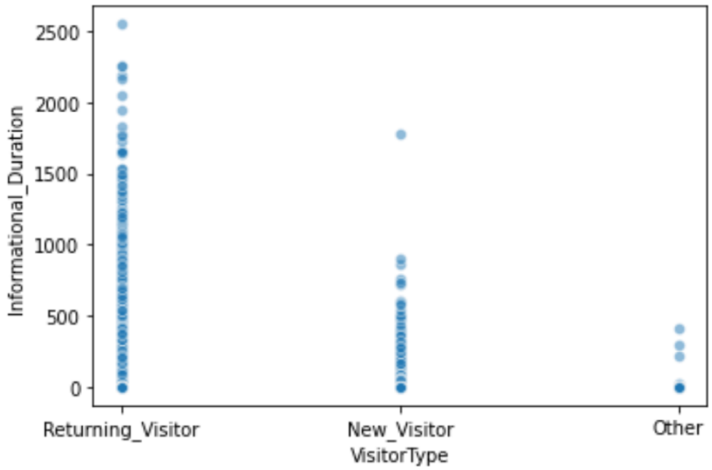
6. VisitorType vs Informational_Duration

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc36d6dfeb8>
```



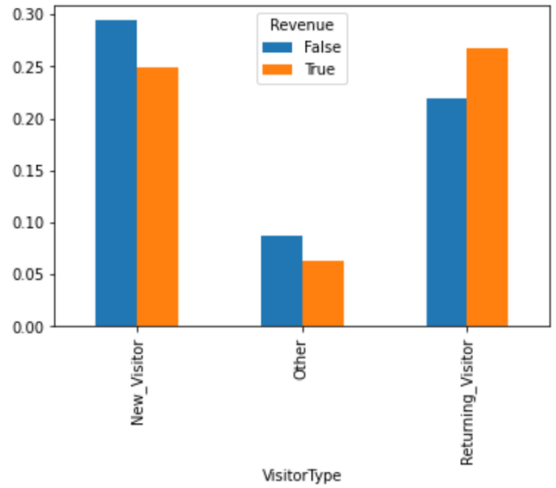
- **Recommendations:**
  7. Providing detailed information to new users is required. Expected to have precise content allows new visitor to make decision faster. For example by adding an option to compare related products.

7(i). VisitorType vs Informational_Duration



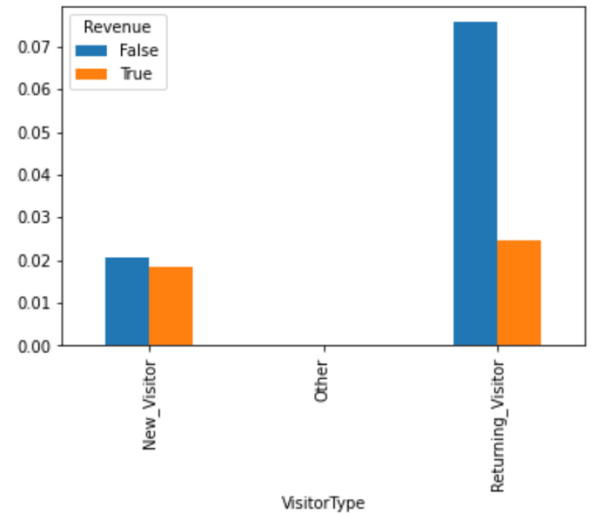
7(ii). VisitorType vs Weekend

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc39dad92b0>
```



7(iii). VisitorType vs SpecialDay

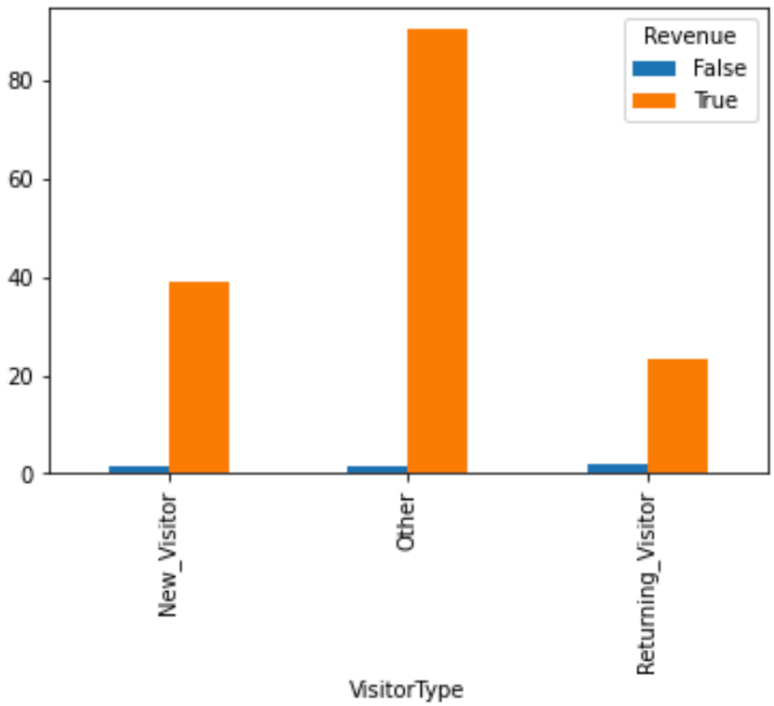
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3a3e097b8>
```



- **Insights:**
  From graph 1: Returning users and other users duration is more when compared with new users, that means new users may not showing interest to know about the website (company).
  From graph 2: Percentage of Returning users and other users Revenue is high in weekends.
  From graph 3: Website (company) has no revenue from other users on special days compared to Returning users and new users.
- **Recommendations:**
  8. Team has to focus on New Users entering into company's website to get information. Duration spent by new user is less so, limiting website's Information in detailed format with precise content on the same page such as offers on weekends, on special days by posting them on home page highlighting dates and days helps new user to return on both weekends and special days. Additionally asking for email or phone number of new user to receive messages of any discounts or special offers will be next step to make new user purchase.
  9. Though returning visitors are high in number, all most 50% are not making purchase. So, offering them discounts, referral bonus, shipping free charges will allow them to make a purchase.

8. VisitorType vs PageValues

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd152147ac8>
```



- **Recommendations:**
  10. It is surprising that, new users revenue is higher in terms of page value compared to Returning users. So, above recommendations could be true for a company's team to focus on new users making their visits as purchase and converting them as returning user for next terms.

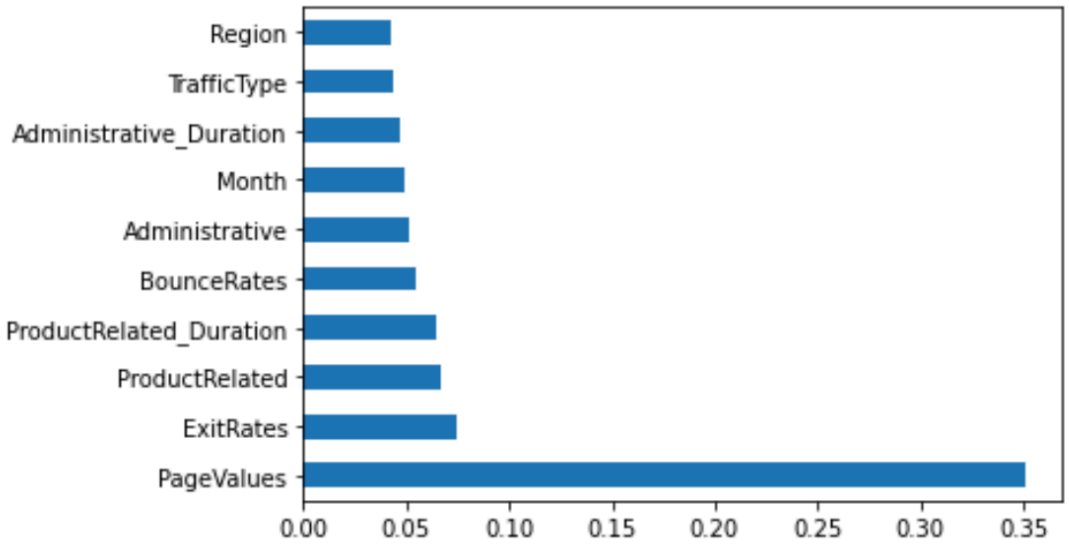
Data Transformation:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df=data
df['Month']= le.fit_transform(df['Month'])
df['VisitorType']= le.fit_transform(df['VisitorType'])
df['Weekend']= le.fit_transform(df['Weekend'])
df['Revenue']= le.fit_transform(df['Revenue'])
```

- Using label encoder column's Month, VisitorType, Weekend, Revenue are converted to numerical columns.

Feature Selection:

```
from sklearn.ensemble import ExtraTreesClassifier
x=df.iloc[:,:17]
y=df.iloc[:,17:18]
model = ExtraTreesClassifier()
model.fit(x,y)
Cols_FS = pd.Series(model.feature_importances_, index=x.columns)
Cols_FS.nlargest(10).plot(kind='barh')
plt.show()
```



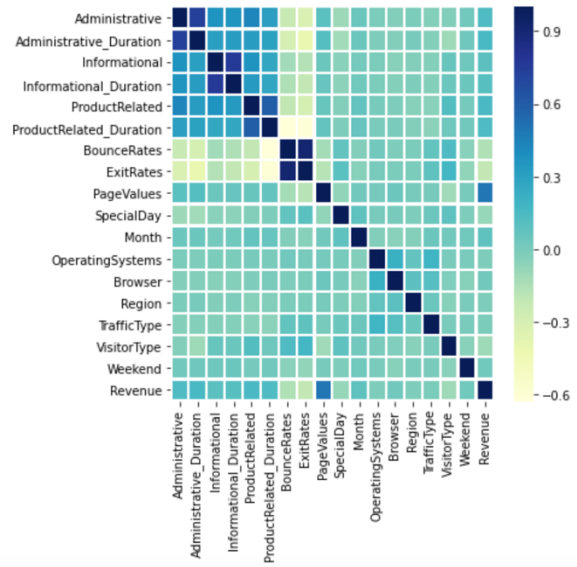
- Out of 17 independent Attributes, top 10 are: PageValues, ExitRates, ProductRelated, BounceRates, Month, TrafficeType, Region, Administrative, OperatingSystems, Broswer.
- PageValues being the highest number among all others. It is the page before a transaction is done. Team must focus on converting all users bring to this page. Offering user benefits allows them to make a purchase.

Data Preprocessing:

Correlation matrix:

```
corr_m = df.corr()
f, ax = plt.subplots(figsize =(6,6))
sns.heatmap(corr_m,annot=False, cmap ="YlGnBu", linewidths = 1.2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1591b40b8>
```



- Since, columns Administrative_Duration, Informational_Duration, ProductRelated_Duration are more correlated with other three columns it is considered to work on them by replacing values with interquartile range (iqr) as mentioned below.

```
#This column has 3335 unique values --> divided as 0,1,2
AD_q1=df['Administrative_Duration'].quantile(0.25)
AD_q3=df['Administrative_Duration'].quantile(0.75)
AD_iqr=AD_q3-AD_q1
ul1=AD_q3+1.5*AD_iqr
df['Administrative_Duration']=df['Administrative_Duration'].map(lambda x:0 if x==0 else (1 if x>0 and x<ul1 else 2))
df['Administrative_Duration'].value_counts()
```

```
0    5903
1    5255
2    1172
Name: Administrative_Duration, dtype: int64
```

```
#This column has 1258 unique values --> divided as 0,1
df['Informational_Duration']=df['Informational_Duration'].map(lambda x:0 if x==0 else 1)
df['Informational_Duration'].value_counts()
```

```
0    9925
1    2405
Name: Informational_Duration, dtype: int64
```

```
#This column has 9551 unique values -- divided as 0,1,2
PRD_q1=df['ProductRelated_Duration'].quantile(0.25)
PRD_q3=df['ProductRelated_Duration'].quantile(0.75)
PRD_iqr=PRD_q3-PRD_q1
ul3=PRD_q3+1.5*PRD_iqr
df['ProductRelated_Duration']=df['ProductRelated_Duration'].map(lambda x:0 if x==0 else (1 if x>0 and x<ul3 else 2))
df['ProductRelated_Duration'].value_counts()
```

```
1    10614
2      961
0      755
Name: ProductRelated_Duration, dtype: int64
```

Model Building:

1. Logistic Regression

```
from sklearn.metrics import f1_score,classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
X=df.iloc[:,:17]
y=df.iloc[:,17:18]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1)
```

```
from sklearn.linear_model import LogisticRegression
log=LogisticRegression(C=100,solver='liblinear',random_state=0)
log.fit(X_train,y_train)
print('Train score:',log.score(X_train,y_train), '&&' ,'Test score:',log.score(X_test,y_test))
```

```
Train score: 0.8850654617077974 && Test score: 0.8867261422005948
```

```
/Users/pradee/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:760: 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().
  y = column_or_1d(y, warn=True)
```

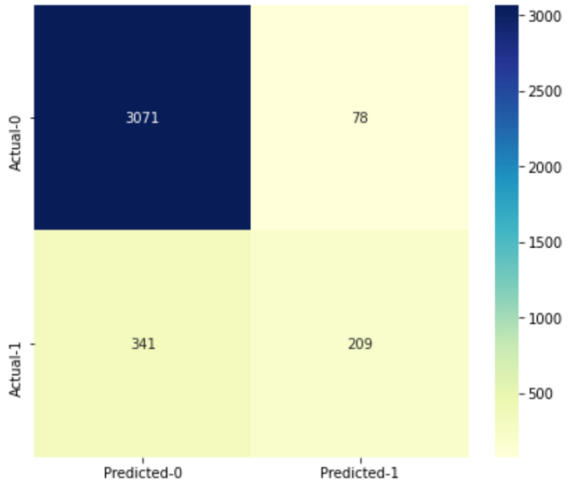
```
y_pred1=log.predict(X_test)
print('F1 Score:',f1_score(y_test,y_pred1))
print('Classification report:',classification_report(y_test,y_pred1))
```

```
F1 Score: 0.4994026284348866
Classification report:               precision    recall  f1-score   support

           0       0.90      0.98      0.94      3149
           1       0.73      0.38      0.50       550

    accuracy                           0.89      3699
   macro avg       0.81      0.68      0.72      3699
weighted avg       0.87      0.89      0.87      3699
```

```
c_matrix1=confusion_matrix(y_test,y_pred1)
c_matrix_fig1=pd.DataFrame(data=c_matrix1,columns=['Predicted-0','Predicted-1'],index=['Actual-0','Actual-1'])
plt.figure(figsize = (7,6))
sns.heatmap(c_matrix_fig1, annot=True,fmt='d',cmap="YlGnBu")
```

- Confusion Matrix

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd156567390>
```



2. Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
print('Train score:',rf.score(X_train,y_train), '&&' ,'Test score:',rf.score(X_test,y_test))
```

```
/Users/pradee/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: 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().
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Train score: 1.0 && Test score: 0.9053798323871317
```

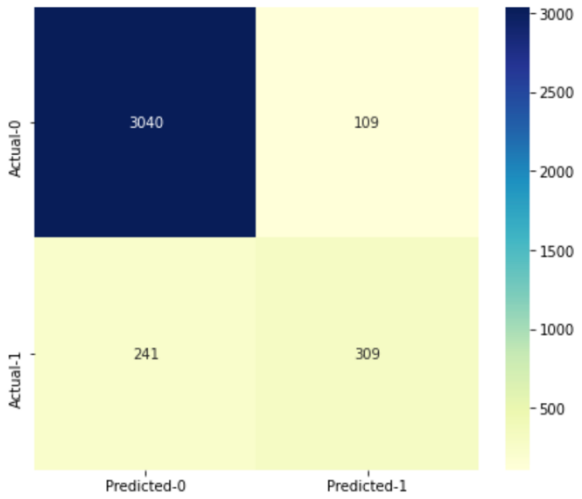
```
y_pred2=rf.predict(X_test)
print('F1 Score:',f1_score(y_test,y_pred2))
print('Classification report:',classification_report(y_test,y_pred2))
```

```
F1 Score: 0.6384297520661157
Classification report:               precision    recall  f1-score   support

           0       0.93      0.97      0.95      3149
           1       0.74      0.56      0.64       550

    accuracy                           0.91      3699
   macro avg       0.83      0.76      0.79      3699
weighted avg       0.90      0.91      0.90      3699
```

```
c_matrix2=confusion_matrix(y_test,y_pred2)
c_matrix_fig2=pd.DataFrame(data=c_matrix2,columns=['Predicted-0','Predicted-1'],index=['Actual-0','Actual-1'])
plt.figure(figsize = (7,6))
sns.heatmap(c_matrix_fig2, annot=True,fmt='d',cmap="YlGnBu")
```

- Confusion Matrix

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb507eaac8>
```



3. Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
DTC=DecisionTreeClassifier()
DTC.fit(X_train,y_train)
print('Train score:',DTC.score(X_train,y_train), '&&' ,'Test score:',DTC.score(X_test,y_test))
```

```
Train score: 1.0 && Test score: 0.8718572587185726
```

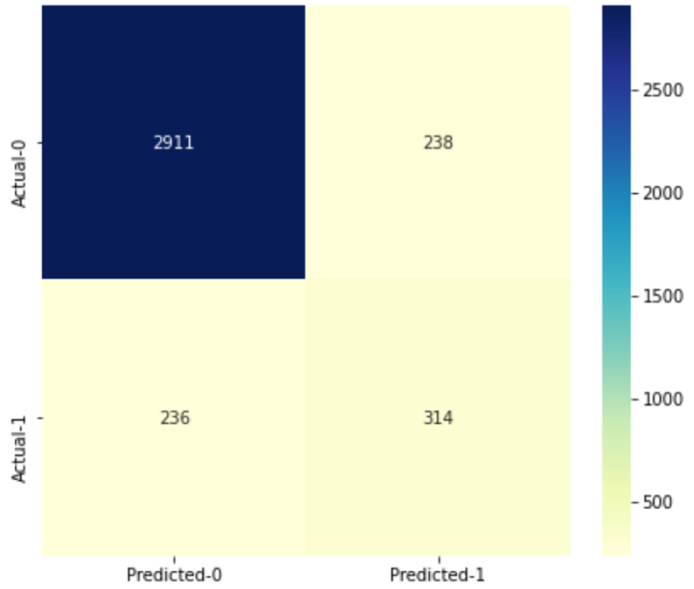
```
y_pred3=DTC.predict(X_test)
print('F1 Score:',f1_score(y_test,y_pred3))
print('Classification report:',classification_report(y_test,y_pred3))
```

```
F1 Score: 0.5698729582577132
Classification report:               precision    recall  f1-score   support

           0       0.93      0.92      0.92      3149
           1       0.57      0.57      0.57       550

    accuracy                           0.87      3699
   macro avg       0.75      0.75      0.75      3699
weighted avg       0.87      0.87      0.87      3699
```

```
c_matrix3=confusion_matrix(y_test,y_pred3)
c_matrix_fig3=pd.DataFrame(data=c_matrix3,columns=['Predicted-0','Predicted-1'],index=['Actual-0','Actual-1'])
plt.figure(figsize = (7,6))
sns.heatmap(c_matrix_fig3, annot=True,fmt='d',cmap="YlGnBu")
```

- Confusion Matrix

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb68f22a58>
```



4. Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
GBC=GradientBoostingClassifier()
GBC.fit(X_train,y_train)
print('Train score:',GBC.score(X_train,y_train), '&&' ,'Test score:',GBC.score(X_test,y_test))
```

```
/Users/pradee/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/_gb.py:1454: DataConversionWarning: A column-ve
ctor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  y = column_or_1d(y, warn=True)
```

```
Train score: 0.9204031977754605 && Test score: 0.9010543390105434
```

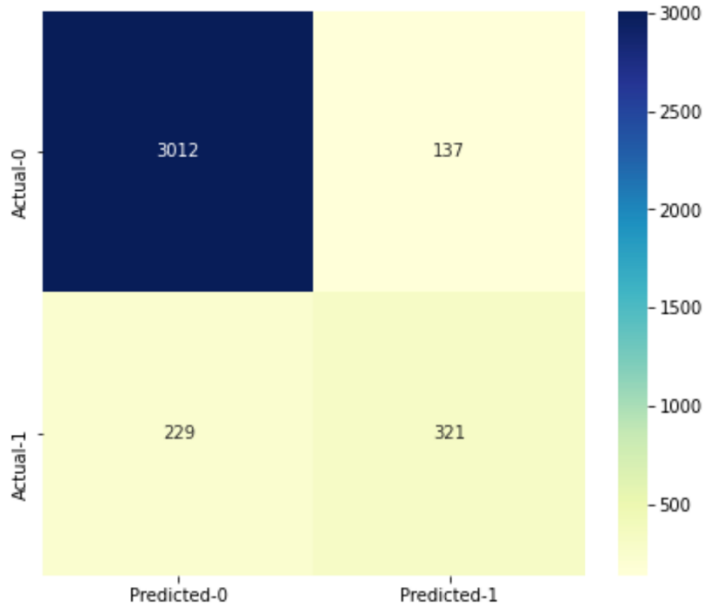
```
y_pred4=GBC.predict(X_test)
print('F1 Score:',f1_score(y_test,y_pred4))
print('Classification report:',classification_report(y_test,y_pred4))
```

```
F1 Score: 0.6369047619047619
Classification report:               precision    recall  f1-score   support

           0       0.93      0.96      0.94      3149
           1       0.70      0.58      0.64       550

    accuracy                           0.90      3699
   macro avg       0.82      0.77      0.79      3699
weighted avg       0.90      0.90      0.90      3699
```

```
c_matrix4=confusion_matrix(y_test,y_pred4)
c_matrix_fig4=pd.DataFrame(data=c_matrix4,columns=['Predicted-0','Predicted-1'],index=['Actual-0','Actual-1'])
plt.figure(figsize = (7,6))
sns.heatmap(c_matrix_fig4, annot=True,fmt='d',cmap="YlGnBu")
```

- Confusion Matrix

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb51740748>
```



Comparing Model Metrics:

```
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, accuracy_score
```

```
#precision_scores
precision1=round(precision_score(y_test,y_pred1),2)
precision2=round(precision_score(y_test,y_pred2),2)
precision3=round(precision_score(y_test,y_pred3),2)
precision4=round(precision_score(y_test,y_pred4),2)
#recall_scores
recall1=round(recall_score(y_test,y_pred1),2)
recall2=round(recall_score(y_test,y_pred2),2)
recall3=round(recall_score(y_test,y_pred3),2)
recall4=round(recall_score(y_test,y_pred4),2)
#f1_scores
f1=round(f1_score(y_test,y_pred1),2)
f2=round(f1_score(y_test,y_pred2),2)
f3=round(f1_score(y_test,y_pred3),2)
f4=round(f1_score(y_test,y_pred4),2)
#roc_auc_score
roc1=round(roc_auc_score(y_test,y_pred1),2)
roc2=round(roc_auc_score(y_test,y_pred2),2)
roc3=round(roc_auc_score(y_test,y_pred3),2)
roc4=round(roc_auc_score(y_test,y_pred4),2)
#accuracy_score
acc1=round(roc_auc_score(y_test,y_pred1),2)
acc2=round(roc_auc_score(y_test,y_pred2),2)
acc3=round(roc_auc_score(y_test,y_pred3),2)
acc4=round(roc_auc_score(y_test,y_pred4),2)
```

```
scores = {
          'Model':  ['Logistic Regression', 'Random Forest','Decision Tree','Gradient Boosting'],
         'Precision': [precision1,precision2,precision3,precision4],
          'Recall': [recall1,recall2,recall3,recall4],
          'f1 score':[f1,f2,f3,f4],
          'ROC':[roc1,roc2,roc3,roc4],
          'Accuracy':[acc1,acc2,acc3,acc4],
          'Misclassification':[a,b,c,d]  #a,b,c,d values are calculated from confusion matrix
         }
```

```
df_scores = pd.DataFrame (scores, columns = ['Model','Precision','Recall','f1 score','ROC','Accuracy','Misclassificat
df_scores
```

Results:

Model scores:

	Model	Precision	Recall	f1 score	ROC	Accuracy	Misclassification
0	Logistic Regression	0.73	0.38	0.50	0.68	0.68	419
1	Random Forest	0.73	0.57	0.64	0.77	0.77	350
2	Decision Tree	0.56	0.55	0.55	0.74	0.74	474
3	Gradient Boosting	0.70	0.58	0.64	0.77	0.77	376

- From above metrics, Random Forest and Gradient Boosting performs best. But we can still improve model by implementing again preprocessing technique using dummies on month column.
- Below are the changes made to data set, stored in df1. Now, same steps are repeated to build algorithms.

```
df1 = pd.get_dummies(df, columns=['Month'])
```

```
X2=df1.drop(['Revenue'],axis=1)
y2=df1.iloc[:,16:17]
X_train,X_test,y_train,y_test=train_test_split(X2,y2,test_size=0.3,random_state=1)
```

Conclusion:

	Model	Precision	Recall	f1 score	ROC	Accuracy	Misclassification
0	Random Forest	0.72	0.56	0.63	0.76	0.76	362
1	Gradient Boosting	0.70	0.61	0.65	0.78	0.78	365

- By comparing both the tables, Gradient Boosting Classifier improved by decreasing misclassification number and on the other hand Random Forest performed poor by increasing count of misclassification.
- All other metrics of Random forest is decreased. Gradient Boosting on the other hand performed 100% well increasing in all metrics other than precision that remained constant.
- Considering all these factors, we conclude that Gradient Boosting Classifier suits best to the data set in predicting online shoppers purchasing intention.

References:

1. Sakar, C.O., Polat, S.O., Katircioglu, M. et al. Neural Comput & Applic (2018). [Web Link]