Online Shoppers Purchasing Intention

November 26, 2019

DePaul University

DSC 478 Final Project

Rasheed Hameed

Table of Contents

EXECUTIVE SUMMARY

In our study, of this chosen dataset where the intent of the dataset provider was to study the behavior of real-time users for some virtual shopping environment. The online retailer data had two classes where users resulted in a purchase or with no purchase. The dataset had more customers which didn't resulted in any purchase. We used k-means clustering method an unsupervised machine learning technique. From the supervised learning we used KNN, decision trees and logistic regression. We saw better results with classifying with logistic regression. If we were predicting future customer behavior logistic regression model will be a better choice in predicting if the customer will make a purchase or not with all the other predictor attributes.

METHODS

This dataset didn't require any cleaning as there was not any missing data. We used some modification on some of the attributes to convert to numerical values for the ease of using the different algorithms available for python. The different modules and libraries we used were of NumPy, Pandas, Mathplotlib, and scikit-learn libraries. Using Pandas library was helpful in data exploration where we used plots to explore the different histogram of the attributes. The data was divided between both categorical and numerical variables. The categorical variables were easily converted using the help of pandas create dummy variables. We used cross validation with training and testing subsets of the data with a 80-20 split.

The different sklearn libraries we used were for KMeans, classification report, confusion_matrix as well as decision trees and logistic regression. To create different sets of models. Our reported results were taken from the classification reports and accuracy results.

CONCLUSION

After using multiple machine learning algorithms, we had different results for each model and method. Such as K-means for clustering behavior performed poorly and results were studied using homogenty and completeness scores. The scores showed very low values. For centroids, attributes such as Product related duration have a higher centroid value and we find this attribute to be an anchoring point to clustering. From supervised learning such as KNN, Decision trees and logistic regression. We saw better results with logistic regression as better accuracy, but decision trees algorithm performed well as well. For decision trees we had better true negative rates. Although, looking at the dataset in general there were more instances of customers not making any purchases.

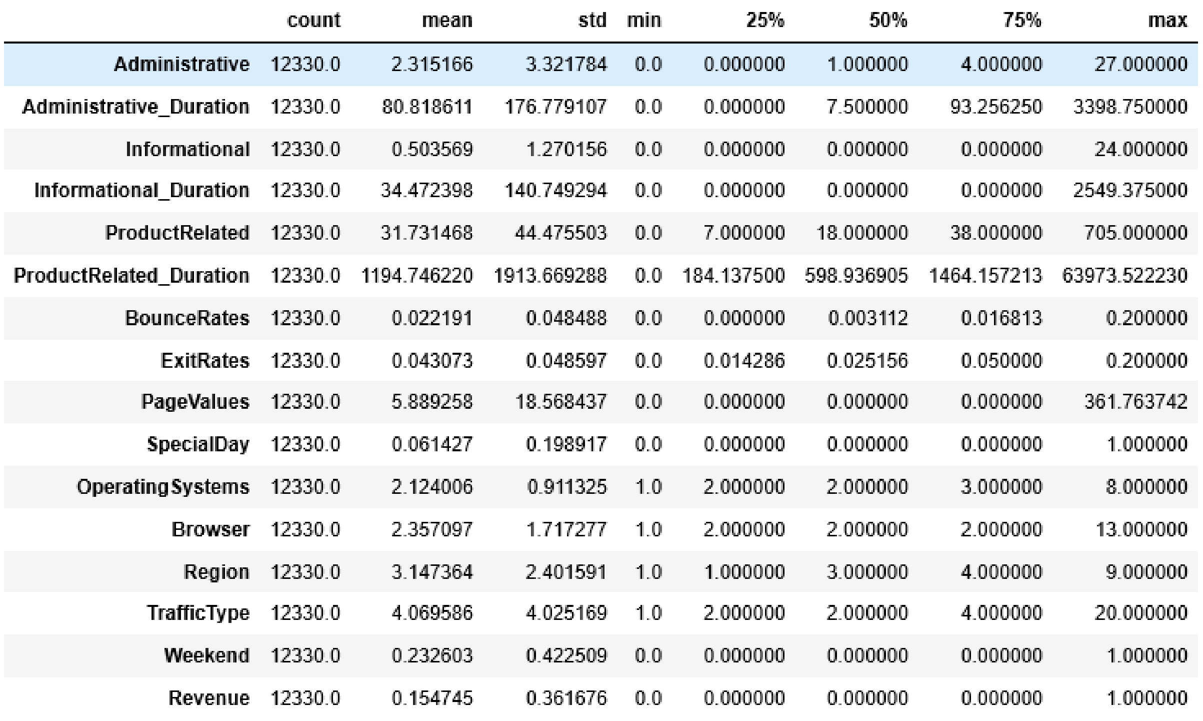
INTRODUCTION

This report uses the dataset from the machine learning repository. Where the original publishers of the data studied the behavior of online shopping using click stream data. They're intention was to study real time how shopping behaviors take place in an e-commerce realm. Within our report we will use attribute, feature and variable interchangeably as these represent the same thing. We aim to predict and classify a chosen target variable such as Revenue. Our goal is to predict and observe at type of accuracy we can predict this behavior. We will employ these machine learning techniques using both unsupervised learning k-means, and supervised learning such as KNN, decision trees, logistic regression. Our goal is to observe our decisions in k-means clustering and to observe we can classify with good accuracy using our dataset.

DATA SET

This dataset is comprised of 12,330 rows which represents online sessions for separate individuals in a period of one year. There are total of 18 attributes both categorical and numerical. For our study the focus will be on Revenue attribute which consists of 10,422 online shoppers that do not make a purchase and 1,988 that make a purchase which is displayed in Figure 5.

Table 1 below shows the numerical values.



	count	mean	std	min	25%	50%	75%	max
Administrative	12330.0	2.315166	3.321784	0.0	0.000000	1.000000	4.000000	27.000000
Administrative_Duration	12330.0	80.818611	176.779107	0.0	0.000000	7.500000	93.256250	3398.750000
Informational	12330.0	0.503569	1.270156	0.0	0.000000	0.000000	0.000000	24.000000
Informational_Duration	12330.0	34.472398	140.749294	0.0	0.000000	0.000000	0.000000	2549.375000
ProductRelated	12330.0	31.731468	44.475503	0.0	7.000000	18.000000	38.000000	705.000000
ProductRelated_Duration	12330.0	1194.746220	1913.669288	0.0	184.137500	598.936905	1464.157213	63973.522230
BounceRates	12330.0	0.022191	0.048488	0.0	0.000000	0.003112	0.016813	0.200000
ExitRates	12330.0	0.043073	0.048597	0.0	0.014286	0.025156	0.050000	0.200000
PageValues	12330.0	5.889258	18.568437	0.0	0.000000	0.000000	0.000000	361.763742
SpecialDay	12330.0	0.061427	0.198917	0.0	0.000000	0.000000	0.000000	1.000000
OperatingSystems	12330.0	2.124006	0.911325	1.0	2.000000	2.000000	3.000000	8.000000
Browser	12330.0	2.357097	1.717277	1.0	2.000000	2.000000	2.000000	13.000000
Region	12330.0	3.147364	2.401591	1.0	1.000000	3.000000	4.000000	9.000000
TrafficType	12330.0	4.069586	4.025169	1.0	2.000000	2.000000	4.000000	20.000000
Weekend	12330.0	0.232603	0.422509	0.0	0.000000	0.000000	0.000000	1.000000
Revenue	12330.0	0.154745	0.361676	0.0	0.000000	0.000000	0.000000	1.000000

Table 1: Numerical Attributes

Description of the different attributes:

Administrative, Administrative Duration, Informational, Informational Duration, Product Related and Product Related Duration represent the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories as shown in Figure 1.

The values of these features as mentioned on the data hosting website are derived from the URL information of the pages visited by the user. This information is updated in real time when a user takes an action, e.g. moving from one page to another. This attributes Bounce Rate, Exit Rate and Page Value shown in figure 2, are the metrics measured using Google Analytics. The value of Bounce Rate attribute refers to the percentage of visitors who enter the site for that page and then leave ("bounce") without triggering any other requests to the analytics server during the same session. The value of Exit Rate attribute is calculated for all pageviews to the page, the percentage that were the last in the session. The Page Value attribute represent the average value for a web page that a user visited before completing an e-commerce transaction.

Attributes shown in figure 3 Special Day indicates the closeness of the site visiting time to a specific special day (e.g. Mother’s Day, Valentine's Day) where it is more likely to be finalized with transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentine’s day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8. The dataset also includes operating system, browser, region, traffic type, visitor type as returning or new visitor, a Boolean value indicating whether the date of the visit is weekend, and month of the year as shown in figure 4. (http://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset)

Analysis of Features

Figure 1 consists of attributes of types of websites that individual shoppers have visited. Observing the plots of these histogram noticing majority of the frequency of the data for these attributes show a skewness to the right.

The distribution for our target variable shows that majority of the revenue is of non-purchasers. We attempted to see the relationship of revenue attribute with weekend shoppers and visitor type shown in figure 6. We noticed the Visitor type “Returning visitors” tend to be high purchasers and during non-weekend time period.

Observing on correlations between the attributes that falls above 0.5. We noticed positive relationship between these attributes Administrative with Administrative Duration, Informational with Information Duration, Product Related with Product Related Duration. A high correlation with bounce rate and exit rates. These are all visible in figure 7. Another item that was noticeable for us was a relationship between month of May and special day. Our conclusion on this is that Mother’s day falls during the month of May and special day will be related to this.

Transformation of attributes

For our study we had to modify Revenue and Weekend attributes. We converted the values from True and False to 0 and 1 for easiness in studying this data.

Another feature transformation was done on categorical variables like month, and visitor type. We transformed these into numerical dummy variables which resulted us in creating these additional attributes:

'Month_Aug', 'Month_Dec', 'Month_Feb', 'Month_Jul', 'Month_June', 'Month_Mar', 'Month_May', 'Month_Nov', 'Month_Oct', 'Month_Sep', 'VisitorType_New_Visitor', 'VisitorType_Other', 'VisitorType_Returning_Visitor'

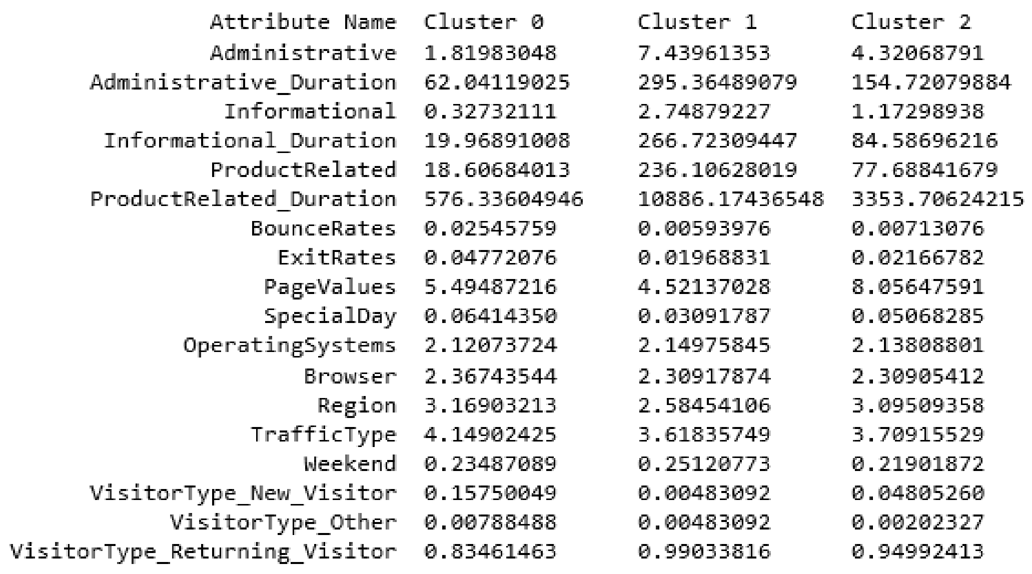
For supervised learning models we normalized the dataset and split our data into 80-20 split for training and testing purposes.

METHODS

We wanted to see some unsupervised learning methods and supervised learning methods. We used these machine learning techniques: K-means, KNN, decision trees and logistic regression.

K-means

For our first method we used this model using the k-means library with “sklearn.clusters”. For the dataset we removed the month dummy variables as we saw this will cause clutter in our reading of data after the output. In our first trial we left the month variable and observed the homogeneity and completeness score and didn’t see much of a significance between keeping these variables for different months. Since, this was our first model we didn’t attempt to transform by normalizing the dataset. We just wanted to see how the data showed after its implementation. We got these results shown below for the centers in table 2. We had our implementation set at 3 clusters and data as raw as possible. From our observation below attributes such as Adminstrative_Duration, Informational_Duration, ProductRelated_Duration, have some large values for centroids between the different clusters. This seems accurate to what the values are in the dataset. There are some largest values with max values being in high ranges. We can also say that these attributes play a higher role in the decision of the customer to make a purchase or no purchase.



```
              Attribute Name  Cluster 0       Cluster 1       Cluster 2
              Administrative  1.81983048      7.43961353      4.32068791
     Administrative_Duration  62.04119025     295.36489079    154.72079884
               Informational  0.32732111      2.74879227      1.17298938
      Informational_Duration  19.96891008     266.72309447    84.58696216
              ProductRelated  18.60684013     236.10628019    77.68841679
     ProductRelated_Duration  576.33604946    10886.17436548  3353.70624215
                 BounceRates  0.02545759      0.00593976      0.00713076
                   ExitRates  0.04772076      0.01968831      0.02166782
                  PageValues  5.49487216      4.52137028      8.05647591
                  SpecialDay  0.06414350      0.03091787      0.05068285
            OperatingSystems  2.12073724      2.14975845      2.13808801
                     Browser  2.36743544      2.30917874      2.30905412
                      Region  3.16903213      2.58454106      3.09509358
                 TrafficType  4.14902425      3.61835749      3.70915529
                     Weekend  0.23487089      0.25120773      0.21901872
     VisitorType_New_Visitor  0.15750049      0.00483092      0.04805260
           VisitorType_Other  0.00788488      0.00483092      0.00202327
VisitorType_Returning_Visitor  0.83461463      0.99033816      0.94992413
```

Table 2 – Centroids with Attributes and clusters

Observing the values, we saw these values to be very low. From this observation we can say that our k-means model performed very poorly with these low scores.

Completeness Score	0.015934804648042894
Homogeneity Score	0.01932075812691264

Model transformation and splitting of dataset

For our second model we transformed our dataset and normalized it. The normalization scaled the values to be between 0 and 1. For these model tests we split our data into 80-20 split were 80% will be used for training and 20% for testing.

KNN

For our second model we used the 80% normalized training data and chose arbitrarily k = 5. KNN is a non-parametric method which helps with classification it is considered a lazy-learning algorithm. For the KNN method we saw better results compared to our previous model. Please see the confusion matrix represented in figure 8. Our accuracy result show that for training we got 100% and for test set got 85% which is pretty good for this model. Getting further into the classification report we see that for classifying no revenue it's precision is at 0.87 and for classifying yes for revenue it is at 0.61

Decision Trees

For the third model we chose to use the decision tree algorithm. Decision tree is a supervised learning algorithm. It is considered an efficient nonparametric method used for both classification and regression. The goal was to predict the individual intent to purchase which is a true case for revenue class. We set the tree with entropy and with setting the levels to grow up to three levels. We got accuracy levels at 99% for the training set and 86.3% for the test set.

Logistic Regression

For our final model we chose logistic regression. Logistic regression is a good binary classification method where two classes.   For our study we tested two models for this method. The first model was with all the variables and the second was with removed correlated variables. The results for both were very similar.   We got accuracy level of 88.1% for training and 88.4% for test set.

KNN	
Training	1.00
Test	0.856
Decision Tree	
Training	0.997
Test	0.863
Logistic Regression	
Training	0.881
Test	0.884

Table 3 – Accuracy Scores per model

Machine Learning Algorithms	Accuracy %	True-positive rate (TPR)	True-Negative rate TNR	F1 Score
KNN	85.6	0.61	0.87	0.33
Decision Trees	86.3	0.56	0.92	0.57
Logistic Regression	88.4	0.89	0.76	0.5

Table 4 – TPR and TNR rates

CONCLUSION

In our study, of this chosen dataset where the intent of the dataset provider was to study the behavior of real-time users for some virtual shopping environment. The online retailer data had two classes where users resulted in a purchase and no purchase was made. Multiple machine learning algorithms were used to see how each behaved. Different methods provided different results. Such as K-means for clustering behavior. Attributes such as Product related duration have a higher centroid value and we find this attribute to be an anchoring point to clustering. The homogeneity and completeness score show very poor results.

From supervised learning such as KNN, Decision trees and logistic regression. We saw better results with logistic regression as better accuracy, but decision trees algorithm performed well as well. For decision trees we had better true negative rates. Although, looking at the dataset in general there were more instances of customers not making any purchases. We performed better with predicting true negatives with decision trees as shown in table 4.

For future studies we will like to dig deeper using neural networks to predict customer purchase behavior in real-time as original intended purpose for the dataset providers.

APPENDIX

Plots

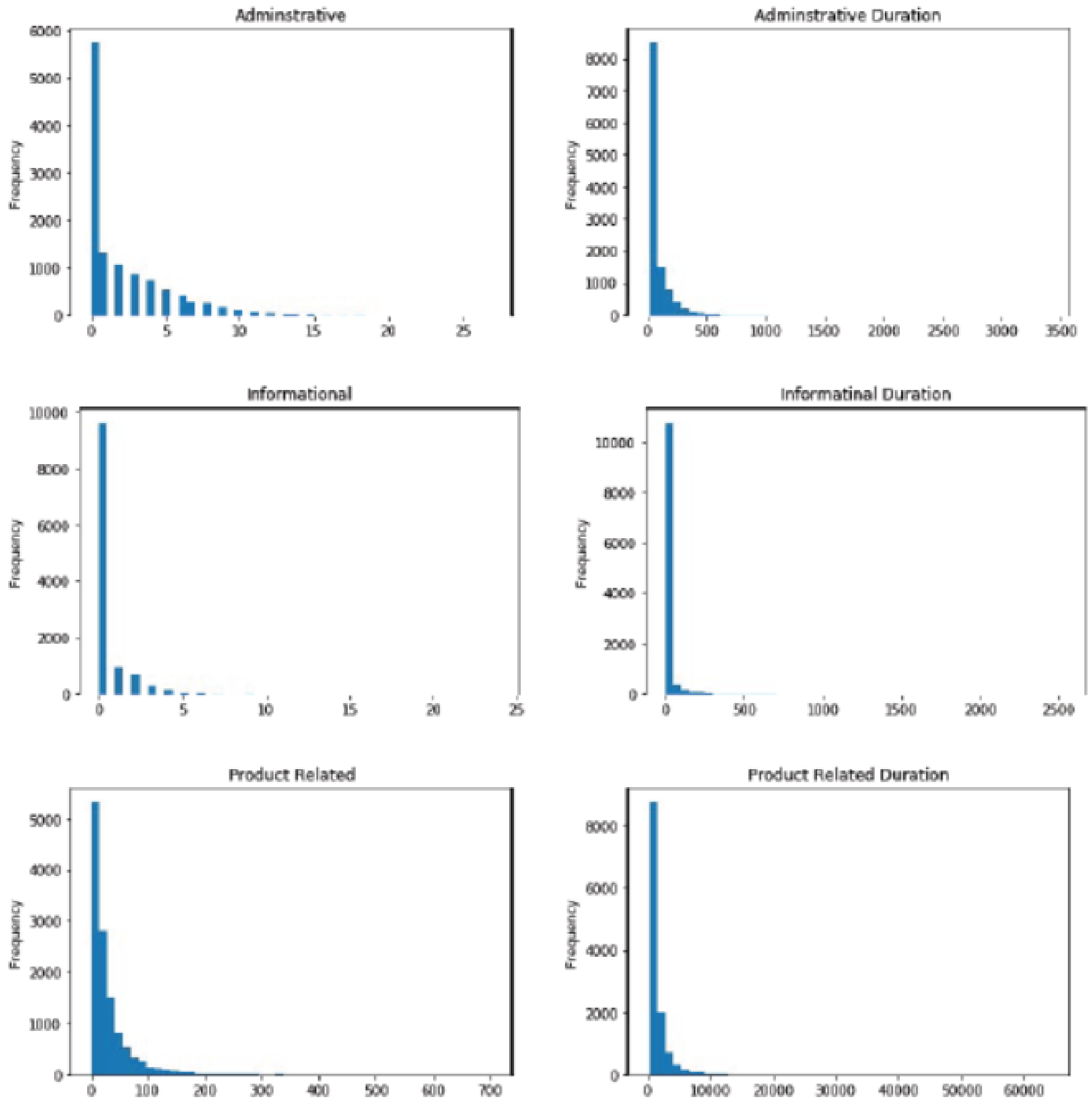


Figure 1: Showing website type visited

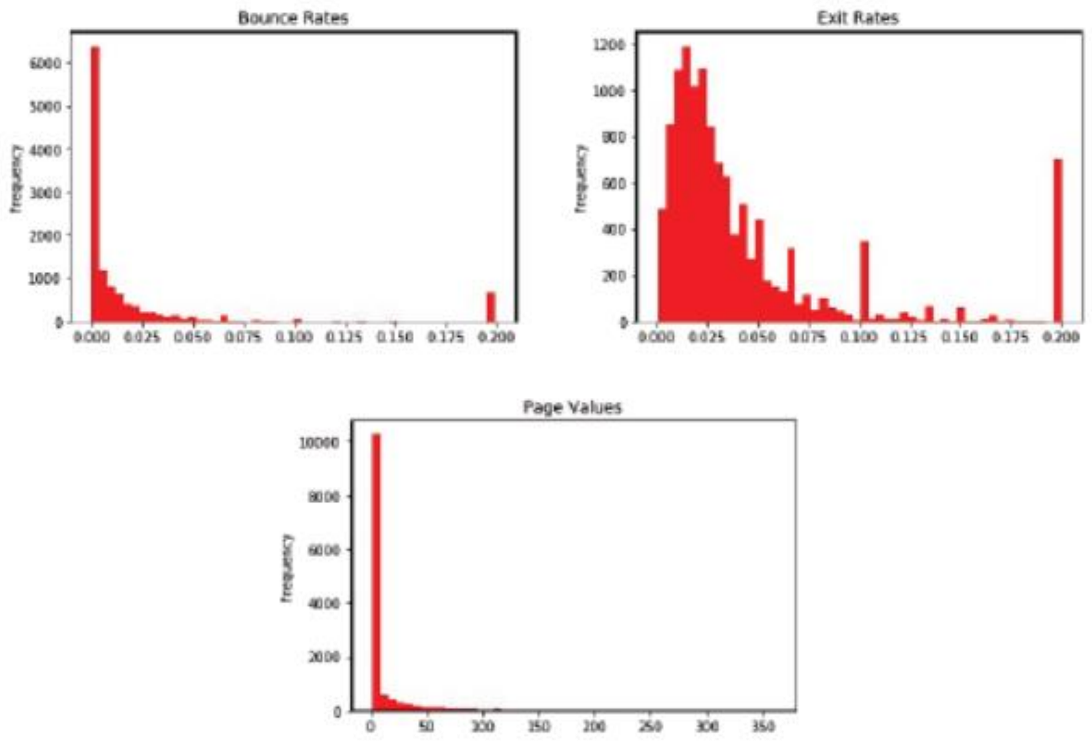


Figure 2 Attributes measured with Google Analytics

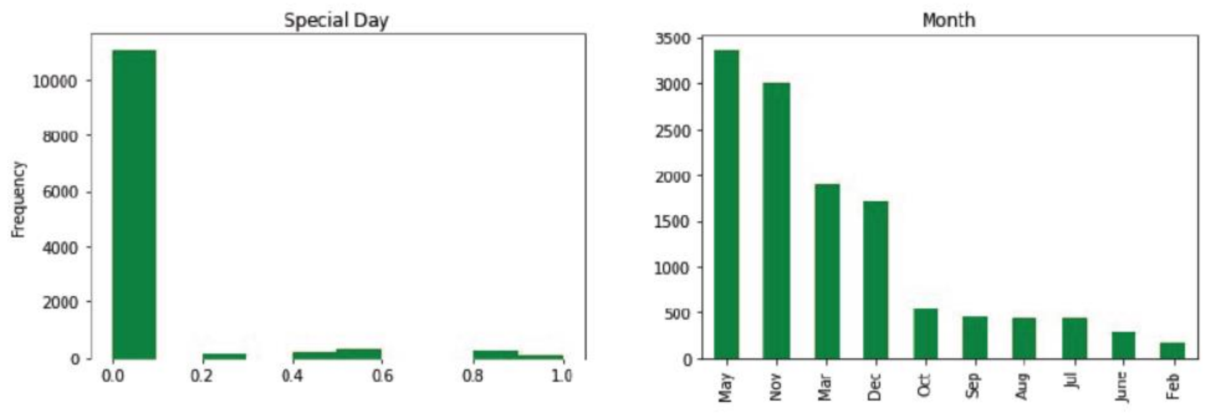


Figure 3 Special Day and Month Attribute Frequency

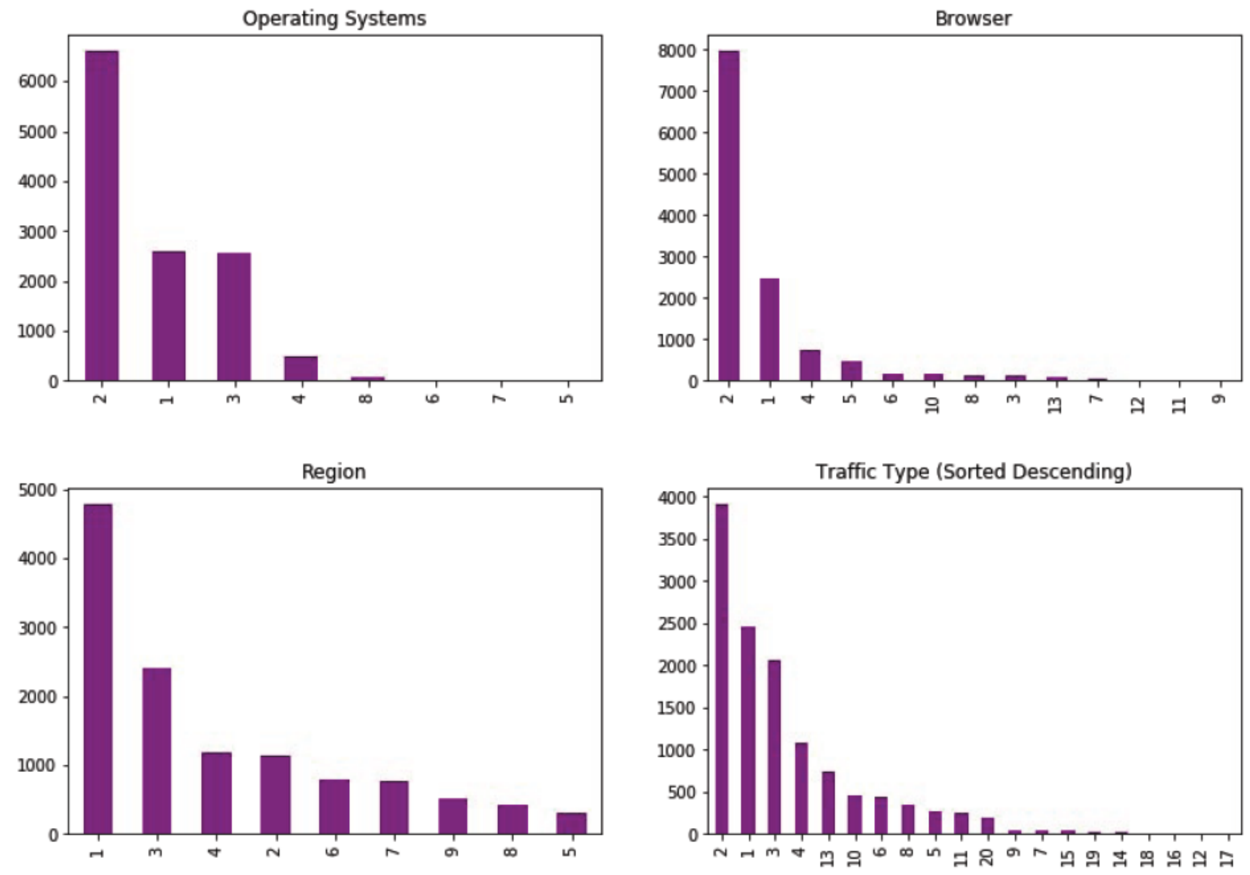


Figure 4 – Operating Systems, Browser, Region and Traffic Type

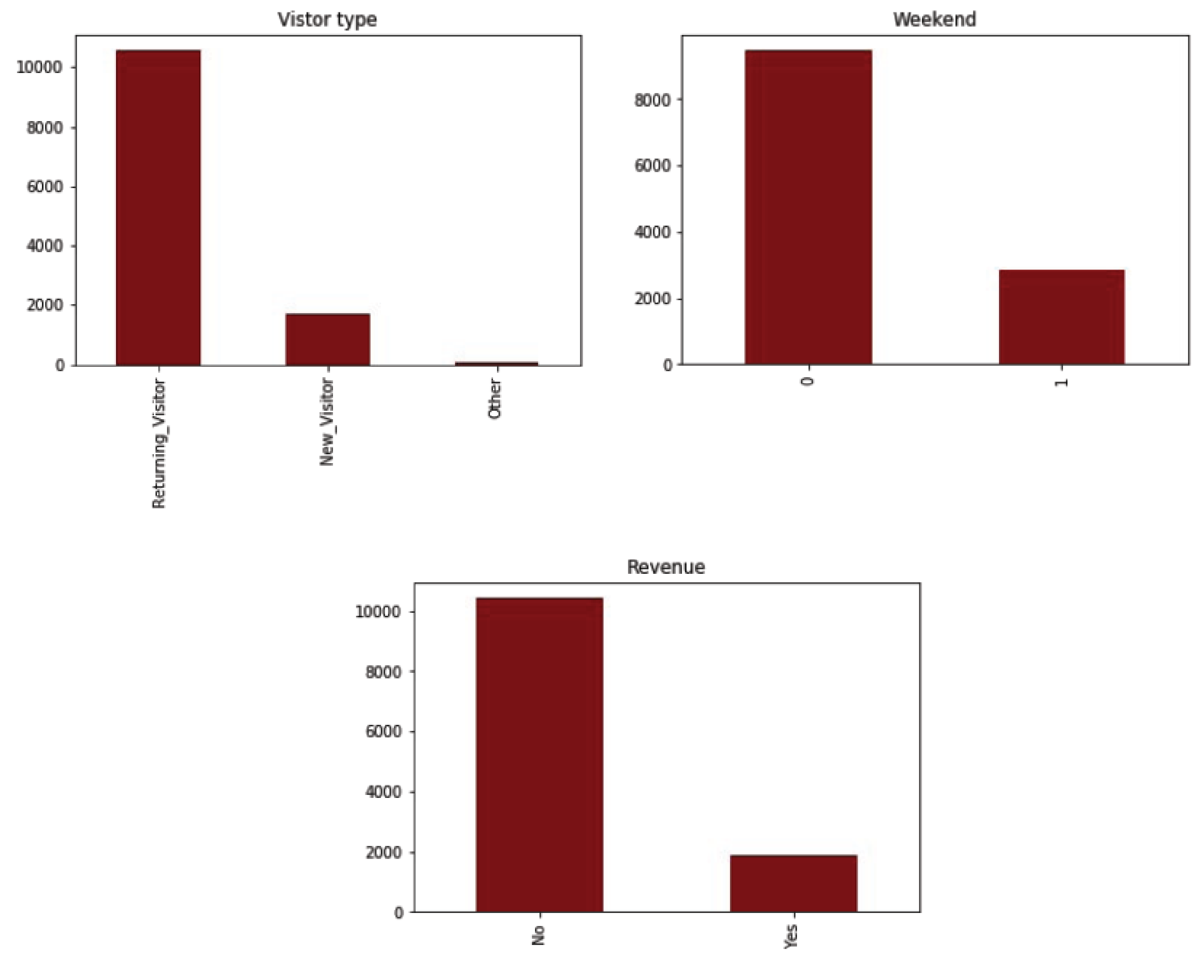


Figure 5 – Visitor type, Weekend and Revenue

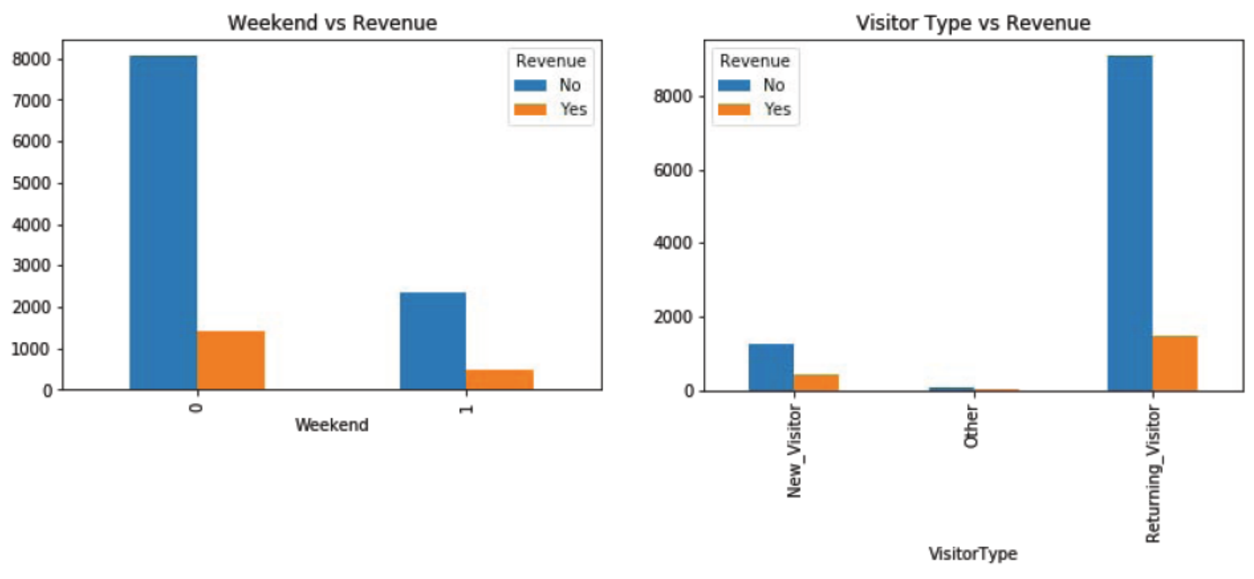
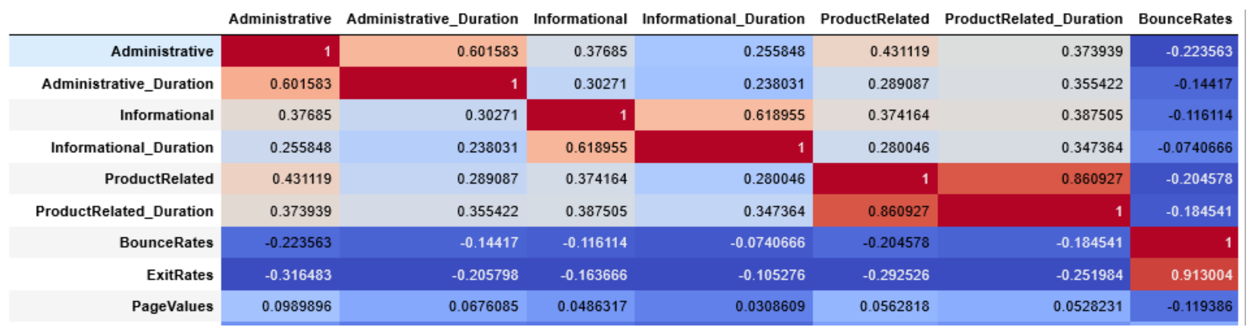


Figure 6 – Relationship of Revenue with Weekend and Visitor Type



	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates
Administrative	1	0.601583	0.37685	0.255848	0.431119	0.373939	-0.223563
Administrative_Duration	0.601583	1	0.30271	0.238031	0.289087	0.355422	-0.14417
Informational	0.37685	0.30271	1	0.618955	0.374164	0.387505	-0.116114
Informational_Duration	0.255848	0.238031	0.618955	1	0.280046	0.347364	-0.0740666
ProductRelated	0.431119	0.289087	0.374164	0.280046	1	0.860927	-0.204578
ProductRelated_Duration	0.373939	0.355422	0.387505	0.347364	0.860927	1	-0.184541
BounceRates	-0.223563	-0.14417	-0.116114	-0.0740666	-0.204578	-0.184541	1
ExitRates	-0.316483	-0.205798	-0.163666	-0.105276	-0.292526	-0.251984	0.913004
PageValues	0.0989896	0.0676085	0.0486317	0.0308609	0.0562818	0.0528231	-0.119386

Figure 7 – Correlation heatmap

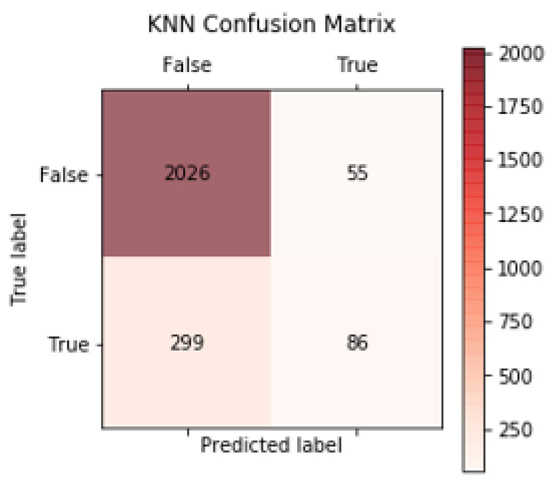


Figure 8 – KNN confusion matrix

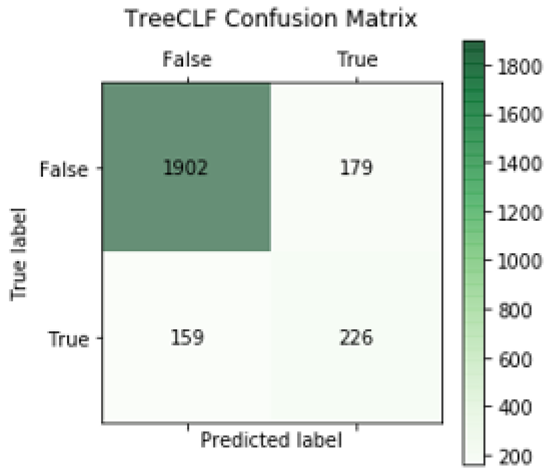


Figure 9 – Decision Tree Confusion Matrix

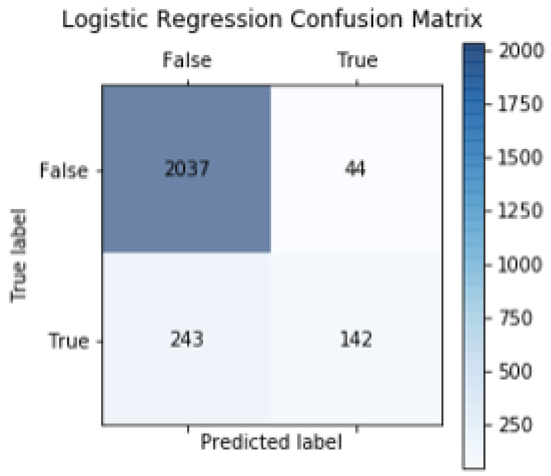


Figure 10 – Logistic Regression Confusion Matrix

Code

```
#Loading libraries


import pandas as pd
import numpy as np
import pylab as pl
import matplotlib.pyplot as plt
#to suppresss the orange warnings
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("online_shoppers_intention.csv")
df.head()
```

Data Exploration¶

```
df.describe(include='all').T
```

Feature engineering, where true and false will be converted to 0 or 1

```
df.Weekend.replace([True, False], [1, 0], inplace=True)
df.Revenue.replace([True, False], [1, 0], inplace=True)
```

Numerical Attributes

```
df.describe().T
```

Columns Month, VisitorType seems to have text values that do not translate well with describe profiler. Dummies will be used to convert these categoricals to numericals

```
df.columns
```

These features below with the histograms in blue represent the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories. The values of these features are derived from the URL information of the pages

visited by the user and updated in real time when a user takes an action, e.g. moving from one page to another.

```
df.Administrative.plot(kind="hist", bins=50, title = 'Adminstrative')
```

majority of the data falls near the between 1 and 10

```
df.Administrative_Duration.plot(kind="hist", bins=50, title = 'Adminstrative Duration'
)

df['Informational'].plot(kind="hist", bins=50, title = 'Informational')

df.Informational_Duration.plot(kind="hist", bins=50, title = 'Informatinal Duration')

df.ProductRelated.plot(kind="hist", bins=50, title = 'Product Related')

df.ProductRelated_Duration.plot(kind="hist", bins=50, title = 'Product Related Duratio
n')
```

The features in red histograms represent the metrics measured by "Google Analytics" for each page in the e-commerce site. 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. 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. The "Page Value" feature represents the average value for a web page that a user visited before completing an e-commerce transaction.

```
df.BounceRates.plot(kind="hist", bins=50,color='red', title = 'Bounce Rates')

df.ExitRates.plot(kind="hist", bins=50,color='red', title = 'Exit Rates')

df.PageValues.plot(kind="hist", bins=50,color='red', title = 'Page Values')
```

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. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentina’s day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8

```
df.SpecialDay.plot(kind="hist", color='green', title ='Special Day')

df['Month'].value_counts().plot(kind='bar', color='green', title ='Month')

df['OperatingSystems'].value_counts().plot(kind='bar', color='purple', title = 'Operat
ing Systems')

df['Browser'].value_counts().plot(kind='bar', color='purple', title ='Browser')
```

```
df['Region'].value_counts().plot(kind='bar', color='purple', title = 'Region')

df['TrafficType'].value_counts().plot(kind='bar', color='purple', title = 'Traffic Typ
e (Sorted Descending)')

df['VisitorType'].value_counts().plot(kind='bar', color='maroon', title = 'Vistor type
')

df['Weekend'].value_counts().plot(kind='bar', color='maroon', title = 'Weekend')

df.Revenue.value_counts()

df.Revenue.value_counts().plot(kind='bar', color='maroon', title = 'Revenue')

df2 = df

df2.Revenue.replace([1, 0], ['Yes','No'], inplace=True)

#df2.Weekend.replace([1, 0], ['Yes','No'], inplace=True)

gg = pd.crosstab(df2.Weekend, df2.Revenue)

gg

plt.show(gg.plot(kind="bar", title = 'Weekend vs Revenue'))
```

From above plot we can say that the Weekend Online browser don't end up complete their shopping by purchasing

```
gg = pd.crosstab(df['VisitorType'], df2["Revenue"])

gg

plt.show(gg.plot(kind="bar", title = 'Visitor Type vs Revenue'))
```

We see above Revenue is more created from returning customers. We can imply on the behavior that individuals will know exactly what they are looking for and would come back and purchase those same items.

```
df_numeric = pd.get_dummies(df[[ 'Month', 'VisitorType']])

df_numeric.head()

df_alternate = pd.concat([df, df_numeric],axis=1)

df_alternate.head()

df_alternate.columns

corr = df_alternate.corr()

corr.style.background_gradient(cmap='coolwarm')
```

In the above correlation plot, some are abvious such as Adminstrative website relates to Adminstrative duration while one shows prominance such as Month_May attribute is correlated with Special day. This is shows that Mothers day fall during month of may as shoppers will make purchase for their mothers on Mother day in the month of may.

Unsupervised Learning¶

```
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
data = df_alternate
#X_train = data[['Administrative', 'Administrative_Duration', 'Informational',
 #      'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
  #     'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'OperatingSystems', 'B
rowser', 'Region', 'TrafficType',
   #     'Weekend', 'Month_Aug', 'Month_Dec', 'Month_Feb',
    #   'Month_Jul', 'Month_June', 'Month_Mar', 'Month_May', 'Month_Nov',
     #  'Month_Oct', 'Month_Sep', 'VisitorType_New_Visitor',
      # 'VisitorType_Other', 'VisitorType_Returning_Visitor'  ]]
X_train = data[['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'OperatingSystems', 'Br
owser', 'Region', 'TrafficType',
        'Weekend',  'VisitorType_New_Visitor',
       'VisitorType_Other', 'VisitorType_Returning_Visitor'  ]]
target = data['Revenue']
X_train.head(2)
X_train.describe(include='all').T
X_train.shape
target.head(3)
kmeans = KMeans(n_clusters=3, max_iter=500, verbose=1) # initialization
kmeans.fit(X_train)
clusters = kmeans.predict(X_train)
print(clusters     )
centroids = kmeans.cluster_centers_
target.shape
centroids.shape
np.set_printoptions(precision=5,suppress=True)
print (centroids)
traincol=[]
for key,value in X_train.iteritems():
    traincol.append(key)
```

```
print ("\t\tAttribute Name\tCluster 0\tCluster 1\tCluster 2")for i in range((X_train.s
hape[1])):

    print ("%30s\t%.8f\t%.8f\t%.8f" %(traincol[i],centroids[0][i],centroids[1][i], cen
troids[2][i]))

from sklearn.metrics import completeness_score, homogeneity_score

print("Completeness Score: ",completeness_score(target,clusters))

print("Homogeneity Score: ",homogeneity_score(target,clusters))
```

Supervised Learning¶

```
from sklearn.model_selection import train_test_split

train, test, target_train, target_test = train_test_split(X_train, target, test_size =
0.2, random_state = 33)

print(train.shape)

print(test.shape)

print(target_train.shape)

print(target_test.shape)

from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler().fit(train)

train_norm = min_max_scaler.transform(train)

test_norm = min_max_scaler.transform(test)

from sklearn import neighbors, tree, naive_bayes

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix
```

KNN¶

```
#KNN

nn = 5

knnclf = neighbors.KNeighborsClassifier(nn, weights = 'distance')

knnclf.fit(train_norm, target_train)

knnpreds_test = knnclf.predict(test_norm)

print(knnpreds_test)

unique_elements, counts_elements = np.unique(knnpreds_test, return_counts=True)

print("Frequency of unique values of the said array:")

print(np.asarray((unique_elements, counts_elements)))

print(classification_report(target_test, knnpreds_test))

knncm = confusion_matrix(target_test, knnpreds_test)

plt.matshow(knncm, cmap = plt.cm.Reds, alpha=0.6)

plt.title("KNN Confusion Matrix\n")
plt.xticks([0,1], ['False', 'True'])
plt.yticks([0,1], ['False', 'True'])
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.colorbar()
for y in range(knncm.shape[0]):
    for x in range(knncm.shape[1]):
        plt.text(x, y, '{}'.format(knncm[y, x]),
                horizontalalignment = 'center',
                verticalalignment = 'center',)
plt.show()
#accuracy score on the training data
print ('Test to see accuracy on training:', knnclf.score(train_norm, target_train))
print ('Accuracy score on test set is: ', knnclf.score(test_norm, target_test))
```

Decision Trees¶

```
treeclf = tree.DecisionTreeClassifier(criterion='entropy', min_samples_split=3)
treeclf = treeclf.fit(train, target_train)
treepreds_test=treeclf.predict(test)
print(classification_report(target_test, treepreds_test))
treecm = confusion_matrix(target_test, treepreds_test)
plt.matshow(treecm, cmap = plt.cm.Greens, alpha=0.6)
plt.title("TreeCLF Confusion Matrix\n")
plt.xticks([0,1], ['False', 'True'])
plt.yticks([0,1], ['False', 'True'])
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.colorbar()
for y in range(treecm.shape[0]):
    for x in range(treecm.shape[1]):
        plt.text(x, y, '{}'.format(treecm[y, x]),
                horizontalalignment = 'center',
                verticalalignment = 'center',)
plt.show()
#accuracy score on the training data
print ('Test to see accuracy on training:', treeclf.score(train, target_train))
print ('Accuracy score is: ', treeclf.score(test, target_test))
```

Logistic Regression¶

```
#LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression(random_state=1)
logistic = logistic.fit(train, target_train)
logpreds_test = logistic.predict(test)
print (classification_report(target_test, logpreds_test))
logisticcm = confusion_matrix(target_test, logpreds_test)
plt.matshow(logisticcm, cmap = plt.cm.Blues, alpha=0.6)
plt.title("Logistic Regression Confusion Matrix\n")
plt.xticks([0,1], ['False', 'True'])
plt.yticks([0,1], ['False', 'True'])
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.colorbar()
for y in range(logisticcm.shape[0]):
    for x in range(logisticcm.shape[1]):
        plt.text(x, y, '{}'.format(logisticcm[y, x]),
                horizontalalignment = 'center',
                verticalalignment = 'center',)
plt.show()
print ('Training Score: ', logistic.score(train, target_train))
print ('Testing Score: ', logistic.score(test, target_test))
```

Model 2 Logistic Regression (removed correlated variables)¶

```
X_train = data[['Administrative', 'Informational', 'ProductRelated', 'ExitRates', 'Pag
eValues', 'SpecialDay', 'OperatingSystems', 'Browser', 'Region', 'TrafficType',
        'Weekend',  'VisitorType_New_Visitor',
       'VisitorType_Other', 'VisitorType_Returning_Visitor'  ]]
from sklearn.model_selection import train_test_split
train, test, target_train, target_test = train_test_split(X_train, target, test_size =
0.2, random_state = 33)
logistic = LogisticRegression(random_state=1)
logistic = logistic.fit(train, target_train)
logpreds_test = logistic.predict(test)
print (classification_report(target_test, logpreds_test))
logisticcm = confusion_matrix(target_test, logpreds_test)
plt.matshow(logisticcm, cmap = plt.cm.Blues, alpha=0.6)
plt.title("Logistic Regression Confusion Matrix\n")
plt.xticks([0,1], ['False', 'True'])
plt.yticks([0,1], ['False', 'True'])
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.colorbar()
for y in range(logisticcm.shape[0]):
    for x in range(logisticcm.shape[1]):
        plt.text(x, y, '{}'.format(logisticcm[y, x]),
                horizontalalignment = 'center',
                verticalalignment = 'center',)
plt.show()
print ('Training Score: ', logistic.score(train, target_train))
print ('Testing Score: ', logistic.score(test, target_test))
```