STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Online Shopper Purchase Intent Analysis

Background

- Online shopping has become one of the shopping channels for consumers with the development of the Internet. In the U.S., total e-commerce sales in 2021 are expected to be $870.8 billion, up 14.2% from 2020 (U.S. Department of Commerce, 2022). In other words, online shopping in the US is still growing in 2021.

- Many online shopping companies have grown into well-known global companies, such as Amazon, Ebay, and Alibaba. Therefore, research on online shopping may bring more benefits to enterprises or companies.

Standard Process: CRISP-DM

Business Understanding

- Goal:
- The purpose of this project is to explore the factors that influence online shopper' purchase intent.

- Question:
- Therefore, the question of this project are following:
- 1) What are the factors that influence online consumers' willingness to buy?
- 2) How to predict consumers' online shopping behavior?

- Accomplishments
- This project try to help enterprises or companies learn about the shopping intentions of online consumers. And companies could improve their strategies for consumers' purchasing intentions.

Standard Process: CRISP-DM

Data Understanding

- Online Shoppers Purchasing Intention Dataset
- https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset
- The dataset consists of feature vectors belonging to 12,330 sessions.The dataset was formed so that each session would belong to a different user in a 1-year period to avoid any tendency to a specific campaign, special day, user profile, or period.

A1 Administrative

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Administr	Administr	Informatic	Informatic	ProductRe	ProductRe	BounceRa	ExitRates	PageValue	SpecialDa	Month	Operating	Browser	Region	TrafficTyp	VisitorTyp	Weekend	Revenue
2	0	0	0	0	1	0	0.2	0.2	0	0	Feb	1	1	1	1	Returning	FALSE	FALSE
3	0	0	0	0	2	64	0	0.1	0	0	Feb	2	2	1	2	Returning	FALSE	FALSE
4	0	0	0	0	1	0	0.2	0.2	0	0	Feb	4	1	9	3	Returning	FALSE	FALSE
5	0	0	0	0	2	2.666667	0.05	0.14	0	0	Feb	3	2	2	4	Returning	FALSE	FALSE
6	0	0	0	0	10	627.5	0.02	0.05	0	0	Feb	3	3	1	4	Returning	TRUE	FALSE
7	0	0	0	0	19	154.2167	0.015789	0.024561	0	0	Feb	2	2	1	3	Returning	FALSE	FALSE
8	0	0	0	0	1	0	0.2	0.2	0	0.4	Feb	2	4	3	3	Returning	FALSE	FALSE
9	1	0	0	0	0	0	0.2	0.2	0	0	Feb	1	2	1	5	Returning	TRUE	FALSE
10	0	0	0	0	2	37	0	0.1	0	0.8	Feb	2	2	2	3	Returning	FALSE	FALSE
11	0	0	0	0	3	738	0	0.022222	0	0.4	Feb	2	4	1	2	Returning	FALSE	FALSE
12	0	0	0	0	3	395	0	0.066667	0	0	Feb	1	1	3	3	Returning	FALSE	FALSE
13	0	0	0	0	16	407.75	0.01875	0.025833	0	0.4	Feb	1	1	4	3	Returning	FALSE	FALSE
14	0	0	0	0	7	280.5	0	0.028571	0	0	Feb	1	1	1	3	Returning	FALSE	FALSE
15	0	0	0	0	6	98	0	0.066667	0	0	Feb	2	5	1	3	Returning	FALSE	FALSE
16	0	0	0	0	2	68	0	0.1	0	0	Feb	3	2	3	3	Returning	FALSE	FALSE
17	2	53	0	0	23	1668.285	0.008333	0.016313	0	0	Feb	1	1	9	3	Returning	FALSE	FALSE
18	0	0	0	0	1	0	0.2	0.2	0	0	Feb	1	1	4	3	Returning	FALSE	FALSE
19	0	0	0	0	13	334.9667	0	0.007692	0	0	Feb	1	1	1	4	Returning	TRUE	FALSE
20	0	0	0	0	2	32	0	0.1	0	0	Feb	2	2	1	3	Returning	FALSE	FALSE
21	0	0	0	0	20	2981.167	0	0.01	0	0	Feb	2	4	4	4	Returning	FALSE	FALSE
22	0	0	0	0	8	136.1667	0	0.008333	0	1	Feb	2	2	5	1	Returning	TRUE	FALSE
23	0	0	0	0	2	0	0.2	0.2	0	0	Feb	3	3	1	3	Returning	FALSE	FALSE
24	0	0	0	0	3	105	0	0.033333	0	0	Feb	3	2	1	5	Returning	FALSE	FALSE
25	0	0	0	0	2	15	0	0.1	0	0.8	Feb	2	4	1	3	Returning	FALSE	FALSE

online shoppers intention

Standard Process: CRISP-DM

Data Understanding

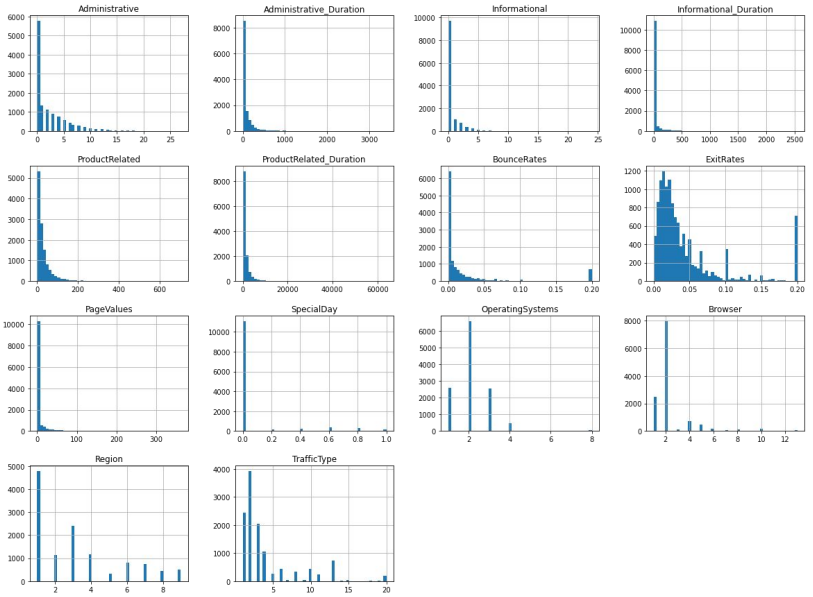
	A	B
1	Feature name	Feature description
2	Administrative	Number of pages visited by the visitor about account management
3	Informational	Number of pages visited by the visitor about Web site, communication and address information of the shopping site
4	Informational duration	Number of pages visited by visitor about product related pages
5	Product related duration	Total amount of time (in seconds) spent by the visitor on product related pages
6	Bounce rate	Average bounce rate value of the pages visited by the visitor
7	Exit rate	Average exit rate value of the pages visited by the visitor
8	Page value	Average page value of the pages visited by the visitor
9	Special day	Closeness of the site visiting time to a special day
10	OperatingSystems	Operating system of the visitor
11	Browser	Browser of the visitor
12	Region	Geographic region from which the session has been started by the visitor
13	TrafficType	Visitor type as “New Visitor,” “Returning Visitor,” and “Other”
14	Weekend	Boolean value indicating whether the date of the visit is weekend
15	Month	Month value of the visit date
16	Revenue	Class label indicating whether the visit has been finalized with a transaction

The feature description(Sakar et al., 2018)

Standard Process: CRISP-DM

Data Preparation

- Choose Python - Jupyter Notebook as the software.
- Using distribution to understand the total data



Distribution of columns

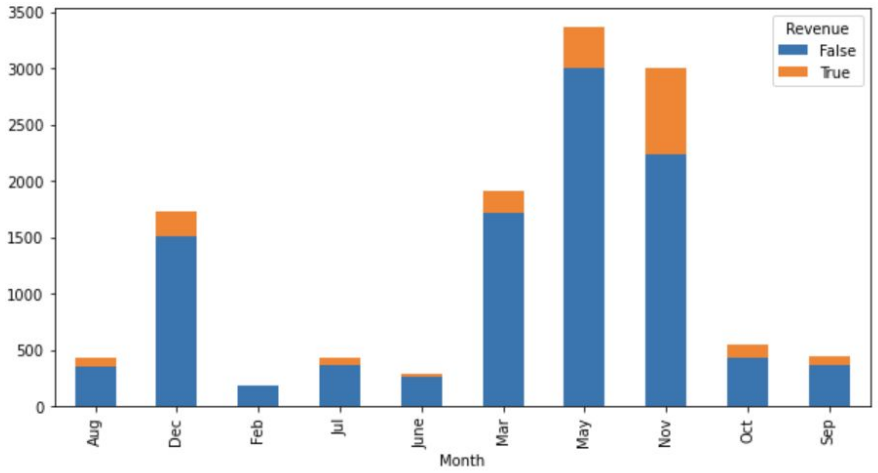
Standard Process: CRISP-DM

Data Preparation

```
memory usage: 1.5+ MB
```

```
In [5]: df.groupby('Month')['Revenue'].value_counts().unstack('Revenue').plot(kind='bar', stacked=True, figsize=(10, 5))
```

```
Out[5]: <AxesSubplot:xlabel='Month'>
```

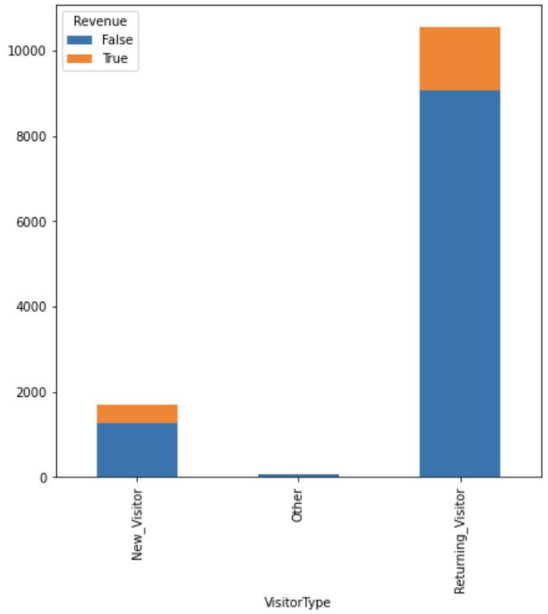


The revenue by month

Standard Process: CRISP-DM

Data Preparation

```
In [6]: df.groupby('VisitorType')['Revenue'].value_counts().unstack('Revenue').plot(kind='bar', stacked=True, figsize=(7, 7))
Out[6]: <AxesSubplot:xlabel='VisitorType'>
```



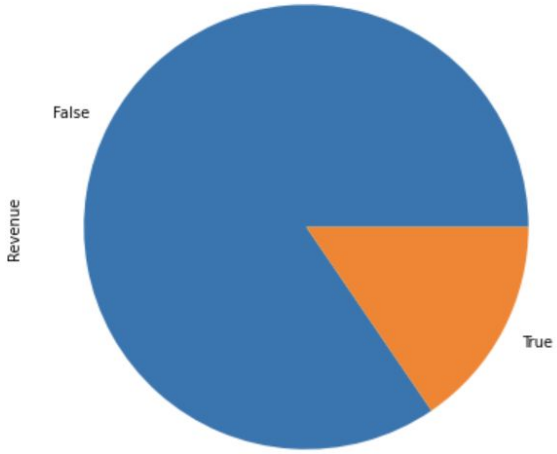
The revenue by visitor type

Standard Process: CRISP-DM

Data Preparation

VisitorType

```
In [7]: df['Revenue'].value_counts().plot.pie(y='Revenue', figsize=(7, 7))
Out[7]: <AxesSubplot:ylabel='Revenue'>
```



The percent of the false and true in revenue

Standard Process: CRISP-DM

Data Preparation

```
: #Transfom the time and the type variables
Month={'Feb':2, 'Mar':3, 'May':5, 'Oct':10, 'June':6, 'Jul':7, 'Aug':8, 'Nov':11, 'Sep':9,'Dec':12}
df['Month']=df['Month'].map(Month)

VisitorType={'Returning_Visitor':3, 'New_Visitor':2, 'Other':1}
df['VisitorType']=df['VisitorType'].map(VisitorType)
d={True:1,False:0}
df['Weekend']=df['Weekend'].map(d)
df['Revenue']=df['Revenue'].map(d)
```

- Transform the data

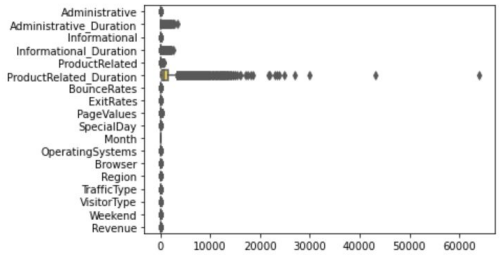
Standard Process: CRISP-DM

Data Preparation

- The data do not have null value.
- However, the boxplot shows that the dataset has outlier.
- The ProductRelated_Duration has significant outlier

```
In [7]: #Checking the null value
        df.isnull().sum()

Out[7]: Administrative             0
        Administrative_Duration    0
        Informational              0
        Informational_Duration     0
        ProductRelated             0
        ProductRelated_Duration    0
        BounceRates                0
        ExitRates                  0
        PageValues                 0
        SpecialDay                 0
        Month                      0
        OperatingSystems           0
        Browser                    0
        Region                     0
        TrafficType                0
        VisitorType                0
        Weekend                    0
        Revenue                    0
        dtype: int64
```

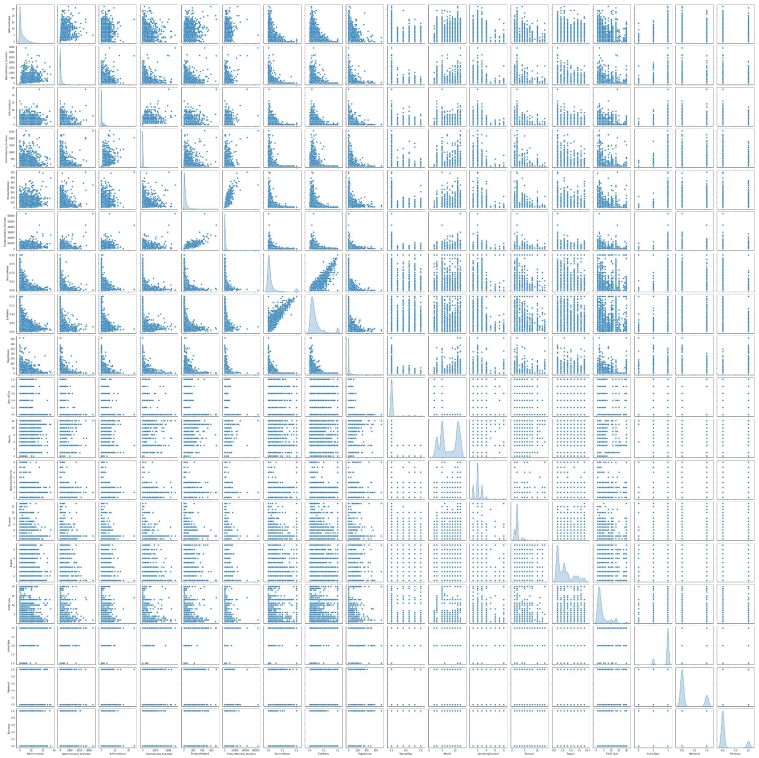
```
In [8]: #Using the boxplot to find outliter
        sns.boxplot(data=df, orient="h", palette="Set2")

Out[8]: <AxesSubplot:>
```

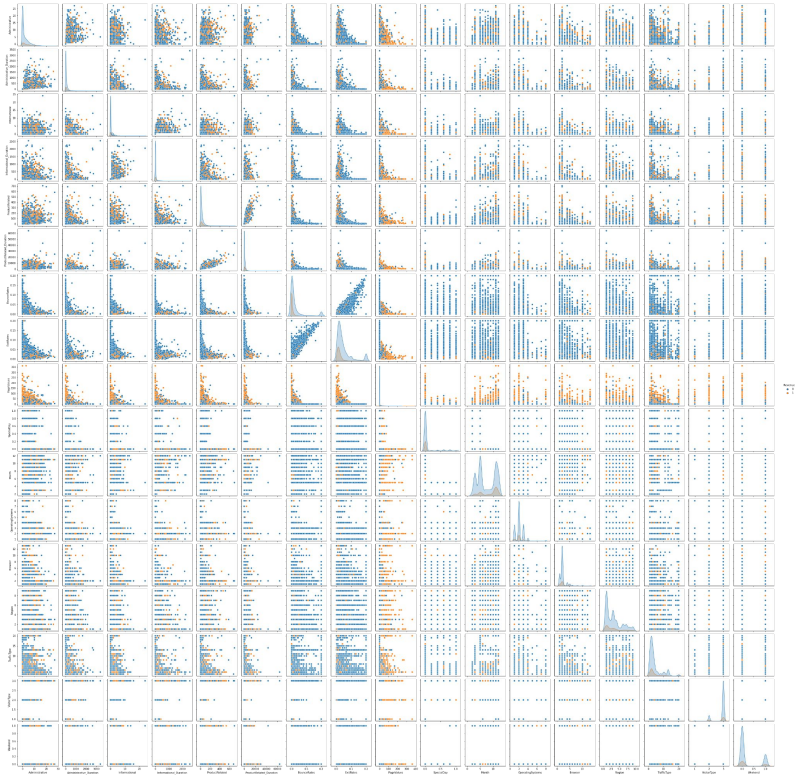


Standard Process: CRISP-DM

Data Preparation

- Using the Bi-Variate Analysis to understand the relationship between variables





Standard Process: CRISP-DM

Data Preparation

Using Interquartile Range (IQR) to check outlier in ProductRelated_Duration

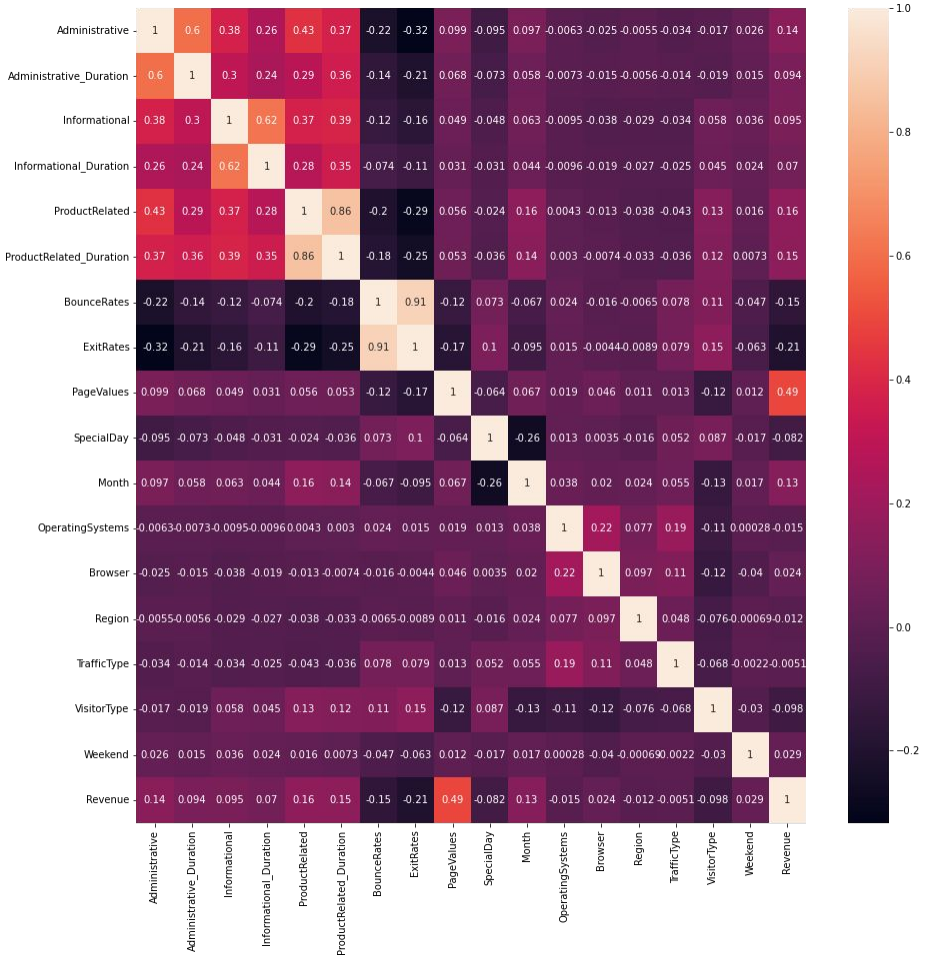
[11]:
```
#Using Interquartile Range (IQR) to check outlier
prd=df['ProductRelated_Duration']
percentile=np.percentile(prd,(25,50,75),interpolation='midpoint')
q1=percentile[0]
q3=percentile[2]
IQR=q3-q1
ulim=q3+1.5*IQR
llim=q1-1.5*IQR

outliter_prd=[]
for i in range(len(prd)):
    if (prd[i] < llim or prd[i] > ulim):
        outliter_prd.append(prd[i])
print(outliter_prd)
```

```
[4084.393939, 6951.972222, 5062.213753, 3556.61241, 5188.5, 5220.083333, 3658.5, 4105.666667, 3510.879903, 5042.458059, 4547.166667, 3608.9,
3410.875, 11301.20416, 4233.4807, 6255.017866, 5958.071931, 5485.105556, 3726.333333, 6682.677557, 12983.78771, 5818.916667, 4699.277778, 39
09.36829, 3960.646154, 6770.274267, 3506.166667, 7221.0, 4566.952839, 8646.0, 3486.690476, 5900.351378, 7669.250794, 3552.702381, 3763.16666
7, 4346.541667, 7533.272727, 3946.008333, 3424.583333, 4099.737392, 4261.842857, 4163.797619, 6737.271825, 8961.407671, 4114.083333, 3641.21
3151, 5764.366667, 4996.666667, 3644.752381, 4056.175369, 4270.141667, 9128.283883, 9694.260714, 3888.166667, 3484.233333, 3451.751515, 405
3.092208, 5186.116667, 3537.991667, 5349.563713, 4891.972222, 4764.429997, 4990.959524, 4164.936905, 6860.632738, 4895.022989, 13158.66667,
5198.533333, 4093.430519, 3882.025758, 4207.666667, 4656.800366, 5378.045851, 4639.410534, 6254.420122, 4945.083333, 5538.916667, 3864.42857
1, 5284.765079, 7009.095996, 8704.372405, 3473.857143, 4107.119658, 3509.130952, 4643.755952, 4415.333333, 8038.325302, 11308.09795, 4115.97
8571, 4139.625, 9487.639162, 4190.75, 8699.407065, 3395.729484, 7968.903361, 4290.833333, 3657.101515, 9487.940434, 4668.052051, 9143.43571
4, 6223.870847, 4346.916667, 4277.0, 9997.728571, 5703.476786, 7428.181584, 5577.826456, 5448.994444, 4977.85, 6842.002513, 3393.903571, 563
3.538274, 5645.176455, 4833.925, 3731.138116, 4509.823016, 3439.733242, 3436.066667, 4324.737073, 5044.75689, 6931.049725, 5459.967105, 456
6.238095, 4051.375, 5604.977381, 5621.599206, 6141.432139, 3827.753571, 4313.139103, 3855.2, 4758.380638, 12065.18135, 5407.788889, 3772.405
525, 3872.50942, 4115.233333, 4086.144897, 3473.733333, 5492.366667, 3583.166667, 6100.527778, 3646.278788, 5478.530087, 4153.276587, 4987.6
88889, 5016.351205, 5664.146032, 7515.583333, 4377.983333, 3731.079365, 5432.836783, 3981.892484, 5233.590476, 4877.257071, 4569.904365, 648
3.775, 3646.966667, 5426.022078, 3386.0, 6657.151362, 4773.335638, 3700.495815, 5343.049577, 3394.130159, 3970.7, 5293.824709, 4651.033333,
7675.14929, 3540.239394, 6266.208586, 13259.29396, 13430.97606, 3909.620363, 4516.588312, 3503.025, 4322.779121, 4555.02381, 5072.395938, 37
27.32381, 5996.025, 3593.12932, 3510.416667, 3837.5, 5731.094444, 5882.174206, 3712.8116, 3900.428571, 3816.366667, 3825.102381, 3639.73730
2, 12634.62089, 4200.597222, 4010.064286, 9951.869139, 3899.001328, 4033.471998, 3733.228571, 4522.409524, 3618.016667, 4079.530461, 4426.65
```

Standard Process: CRISP-DM

Data Preparation

- The features have the correction relationship.



Standard Process: CRISP-DM

Data Preparation

15]:
```
#Standardize the data
scaler = StandardScaler()

scaler.fit(df.drop('Revenue', axis = 1))
scaled_features = scaler.transform(df.drop('Revenue', axis = 1))

df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()
```

15]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues
0	-0.696993	-0.457191	-0.396478	-0.244931	-0.691003	-0.624348	3.667189	3.229316	-0.317178
1	-0.696993	-0.457191	-0.396478	-0.244931	-0.668518	-0.590903	-0.457683	1.171473	-0.317178
2	-0.696993	-0.457191	-0.396478	-0.244931	-0.691003	-0.624348	3.667189	3.229316	-0.317178
3	-0.696993	-0.457191	-0.396478	-0.244931	-0.668518	-0.622954	0.573535	1.994610	-0.317178
4	-0.696993	-0.457191	-0.396478	-0.244931	-0.488636	-0.296430	-0.045196	0.142551	-0.317178

- To avoid outliers affecting the model, normalize the data

Standard Process: CRISP-DM

Modeling

- The project use KNN model and Decision Tree (CART) algorithm to build model.
- The KNN algorithm is to find the distance between the query and all the examples in the data, choose a specified number of examples (K) that are closest to the query, and then vote for the most frequent label or the average label.
- The Decision Tree (CART)construct a binary tree at each node

- Training dataset: 70% data would be used to develop the model.
- Testing dataset: 30% data would be used to evaluate the model.

```
X_train, X_test, y_train, y_test = train_test_split(
      scaled_features, df['Revenue'], test_size = 0.30)
```

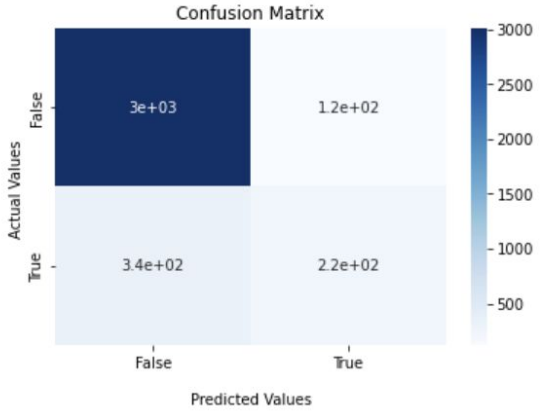
Standard Process: CRISP-DM

Modeling

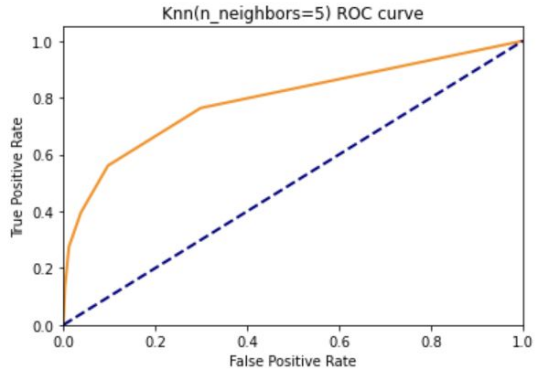
- Build the KNN modeling when k=5
- The accuracy = 0.88

```
# KNN algorithm when K = 5
X_train, X_test, y_train, y_test = train_test_split(
      scaled_features, df['Revenue'], test_size = 0.30)
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
predicted = knn.predict(X_test)
print(classification_report(y_test, predicted))
predict_p=knn.predict_proba(X_test)
y_pred = predict_p[:,1]
auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure();
plt.plot(fpr, tpr, color='darkorange', lw=2);
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--');
plt.xlim([0.0, 1.0]);
plt.ylim([0.0, 1.05]);
plt.title('Knn(n_neighbors=5) ROC curve')
plt.xlabel('False Positive Rate');
plt.ylabel('True Positive Rate');
```

```
ax = sns.heatmap(confusion_matrix(y_test, predicted), annot=True, cmap='Blues')
ax.set_title('Confusion Matrix');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.show()
```



	precision	recall	f1-score	support
0	0.90	0.96	0.93	3132
1	0.65	0.40	0.49	567
accuracy			0.88	3699
macro avg	0.78	0.68	0.71	3699
weighted avg	0.86	0.88	0.86	3699



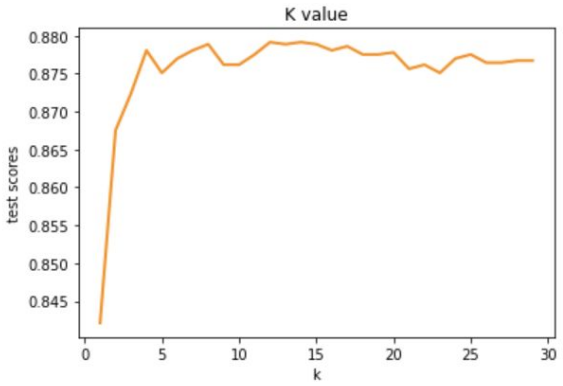
Standard Process: CRISP-DM

Modeling

- Find the best K value for the model
- The best K=12,and K=14

```
# Finding the best K Neighbors value
test_scores = []
train_scores = []
k=[]
for i in range(1,30):
    knn = KNeighborsClassifier(i)
    knn.fit(X_train,y_train)
    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
    k.append(i)
max_train_score = max(train_scores)
m_k1 = [i for i, v in enumerate(train_scores) if v == max_train_score]
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(lambda x: x+1, m_k1))))
max_test_score = max(test_scores)
m_k2 = [i for i, v in enumerate(test_scores) if v == max_test_score]
print('Max test score {} % and k = {}'.format(max_test_score*100,list(map(lambda x: x+1, m_k2))))
plt.figure();
plt.plot(k, test_scores, color='darkorange', lw=2);
plt.plot(color='navy', lw=2, linestyle='--');
plt.title('K value')
plt.xlabel('k');
plt.ylabel('test scores');
```

```
Max train score 100.0 % and k = [1]
Max test score 87.91565287915653 % and k = [12, 14]
```



Standard Process: CRISP-DM

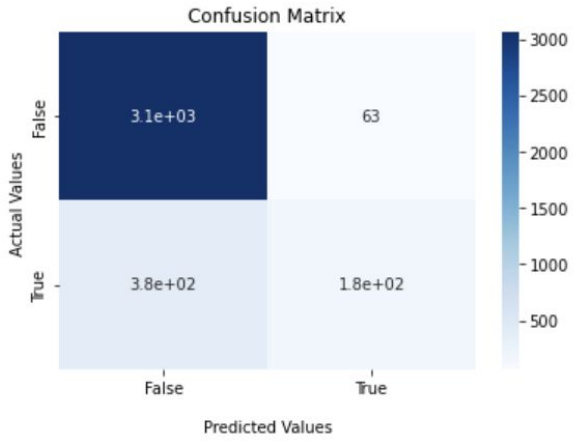
Modeling

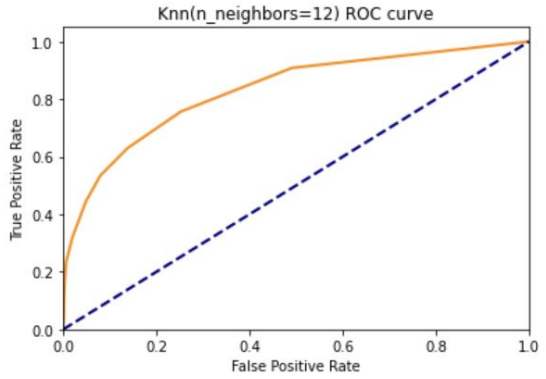
- Build the KNN model with K =12
- The accuracy=0.88

```
#Run the KNN model with the best K
knn = KNeighborsClassifier(n_neighbors =12)
knn.fit(X_train, y_train)
predicted = knn.predict(X_test)
print(classification_report(y_test, predicted))
predict_p=knn.predict_proba(X_test)
y_pred = predict_p[:,1]
auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure();
plt.plot(fpr, tpr, color='darkorange', lw=2);
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--');
plt.xlim([0.0, 1.0]);
plt.ylim([0.0, 1.05]);
plt.title('Knn(n_neighbors=12) ROC curve')
plt.xlabel('False Positive Rate');
plt.ylabel('True Positive Rate');
```

```
ax = sns.heatmap(confusion_matrix(y_test, predicted), annot=True, cmap='Blues
ax.set_title('Confusion Matrix');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.show()
```

	precision	recall	f1-score	support
0	0.89	0.98	0.93	3132
1	0.74	0.32	0.45	567
accuracy			0.88	3699
macro avg	0.82	0.65	0.69	3699
weighted avg	0.87	0.88	0.86	3699





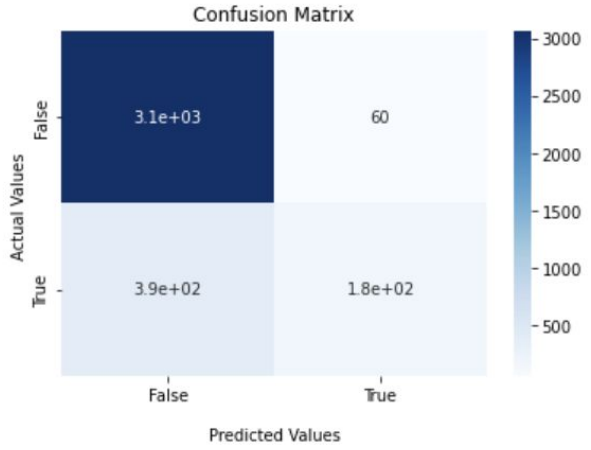
Standard Process: CRISP-DM

Modeling

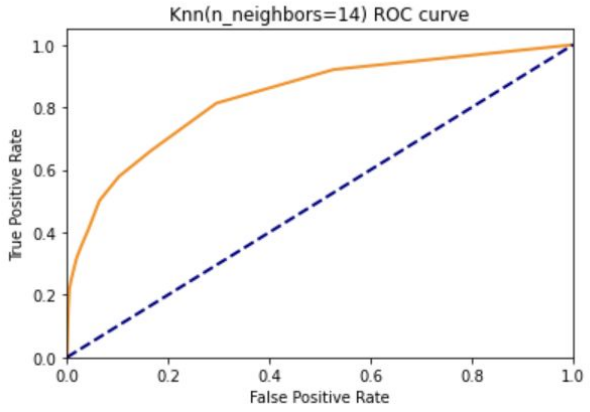
- Build the KNN model with K =14
- The accuracy=0.88

```
knn = KNeighborsClassifier(n_neighbors =14)
knn.fit(X_train, y_train)
predicted = knn.predict(X_test)
print(classification_report(y_test, predicted))
predict_p=knn.predict_proba(X_test)
y_pred = predict_p[:,1]
auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure();
plt.plot(fpr, tpr, color='darkorange', lw=2);
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--'
plt.xlim([0.0, 1.0]);
plt.ylim([0.0, 1.05]);
plt.title('Knn(n_neighbors=14) ROC curve')
plt.xlabel('False Positive Rate');
plt.ylabel('True Positive Rate');
```

```
ax = sns.heatmap(confusion_matrix(y_test, predicted), annot=True, cmap='Blues')
ax.set_title('Confusion Matrix');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.show()
```



	precision	recall	f1-score	support
0	0.89	0.98	0.93	3132
1	0.75	0.32	0.45	567
accuracy			0.88	3699
macro avg	0.82	0.65	0.69	3699
weighted avg	0.87	0.88	0.86	3699

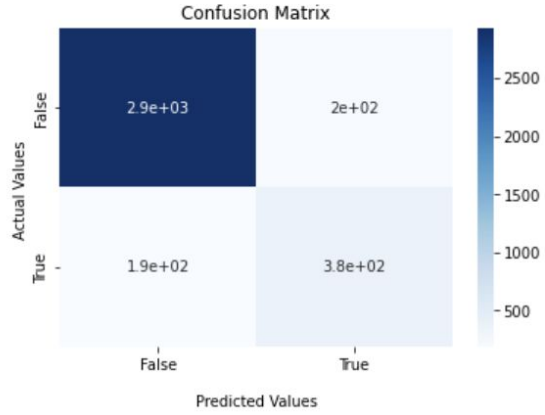


Standard Process: CRISP-DM

Modeling

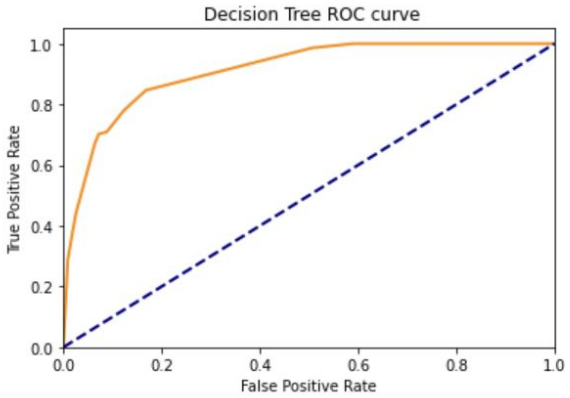
- Build the decision tree model
- The accuracy = 0.90

```
ax = sns.heatmap(confusion_matrix(y_test, predicted), annot=True, cmap='Blues')
ax.set_title('Confusion Matrix');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.show()
```




```
# Creating the Decision Tree model (Cart model)
dt = DecisionTreeClassifier(max_depth = 4)
dt.fit(X_train, y_train)
predicted = dt.predict(X_test)
print(classification_report(y_test, predicted))
predict_p=dt.predict_proba(X_test)
y_pred = predict_p[:,1]
auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure();
plt.plot(fpr, tpr, color='darkorange', lw=2);
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--');
plt.xlim([0.0, 1.0]);
plt.ylim([0.0, 1.05]);
plt.title(' Decision Tree ROC curve')
plt.xlabel('False Positive Rate');
plt.ylabel('True Positive Rate');
```

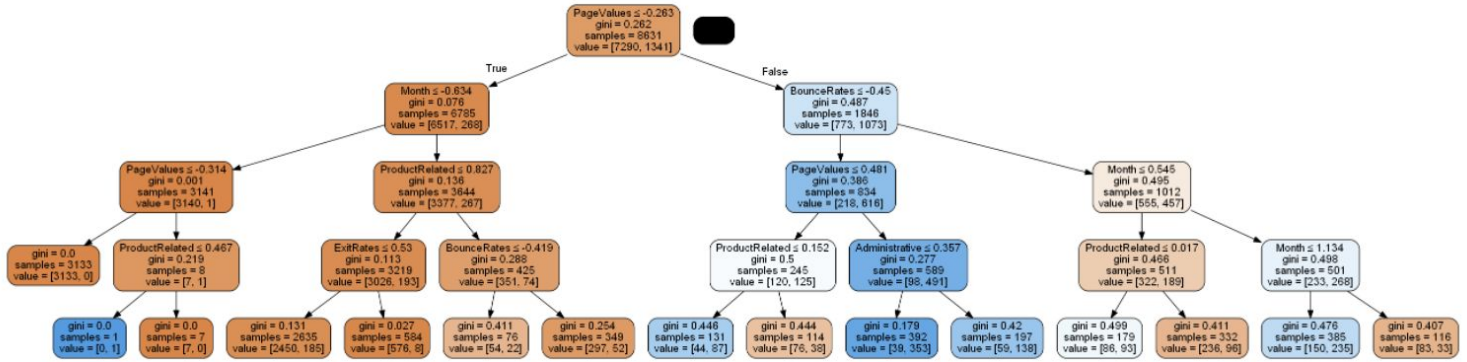
	precision	recall	f1-score	support
0	0.94	0.94	0.94	3132
1	0.66	0.67	0.66	567
accuracy			0.90	3699
macro avg	0.80	0.80	0.80	3699
weighted avg	0.90	0.90	0.90	3699



Standard Process: CRISP-DM

Modeling

```
# Creat the tree graph about the Decision Tree model
import pydotplus
from six import StringIO
from sklearn import tree
from IPython.display import Image
features = df.drop('Revenue',axis=1).columns.tolist()
dot_data = StringIO()
tree.export_graphviz(dt, out_file=dot_data, feature_names=features, filled=True, rounded=True,  special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



The tree graph of decision tree model

Standard Process: CRISP-DM

Evaluation

- This project uses the most accurate model to predict factors that affect online shopper purchase intent
- Both models in the project have high accuracy and are close
- The decision tree(CART) has higher accuracy (0.90) than KNN (0.88).
- Therefore, the decision tree model could be use in the project

Model	Accuracy
KNN	0.88
Decision tree (CART)	0.90

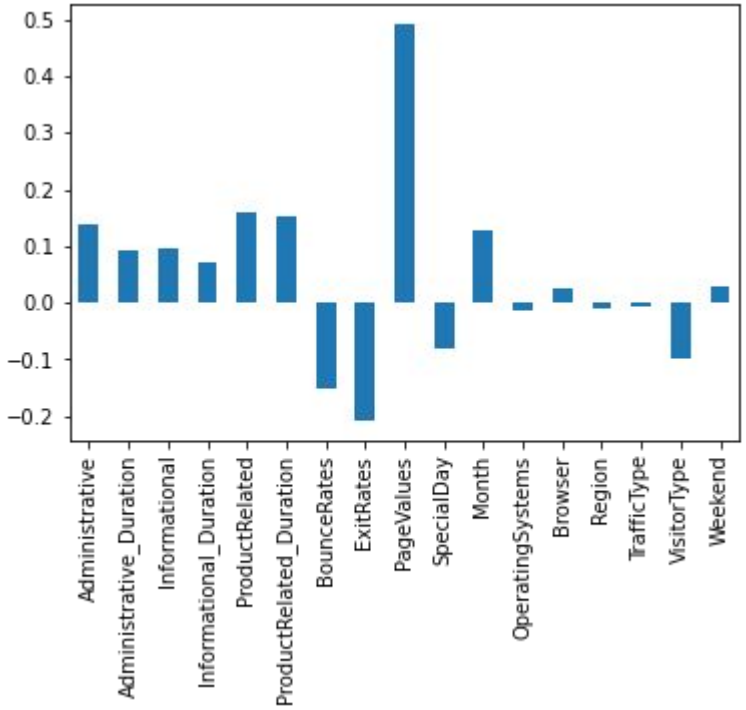
Standard Process: CRISP-DM

Deployment

- The established model can be handed over to the online shopping company to complete the deployment.
- Online shopping companies predict the factors that affect consumers' purchase intention by inputting the consumer information they have.
- The predicted results should be compared with the real situation to evaluate the accuracy of the model in actual use.
- After refining the model with real data from the company, the company can use the results of the model to make decisions that attract consumers to shop online.

Conclusion

- The page value has the strongest positive correlation with Revenue(purchase intent)
- The exit rates has the strongest negative correlation with Revenue(purchase intent)
- Therefore, When consumers browse more web pages, their purchase intention is stronger.
- The higher the average exit rate value of the pages visited by the visitor, the lower the purchase intention of consumers



Reference

Authors & websites used for reference

- Sakar, C. Okan, et al. “Real-Time Prediction of Online Shoppers’ Purchasing Intention Using Multilayer Perceptron and LSTM Recurrent Neural Networks.” *Neural Computing and Applications*, vol. 31, no. 10, 2018, pp. 6893–6908., https://doi.org/10.1007/s00521-018-3523-0.

- Sakar, C. Okan, and Yomi Kastro. “Online Shoppers Purchasing Intention Dataset Data Set.” UCI Machine Learning Repository: Online Shoppers Purchasing Intention Dataset Data Set, 31 Aug. 2018, https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset.

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

stevens.edu

THANK YOU