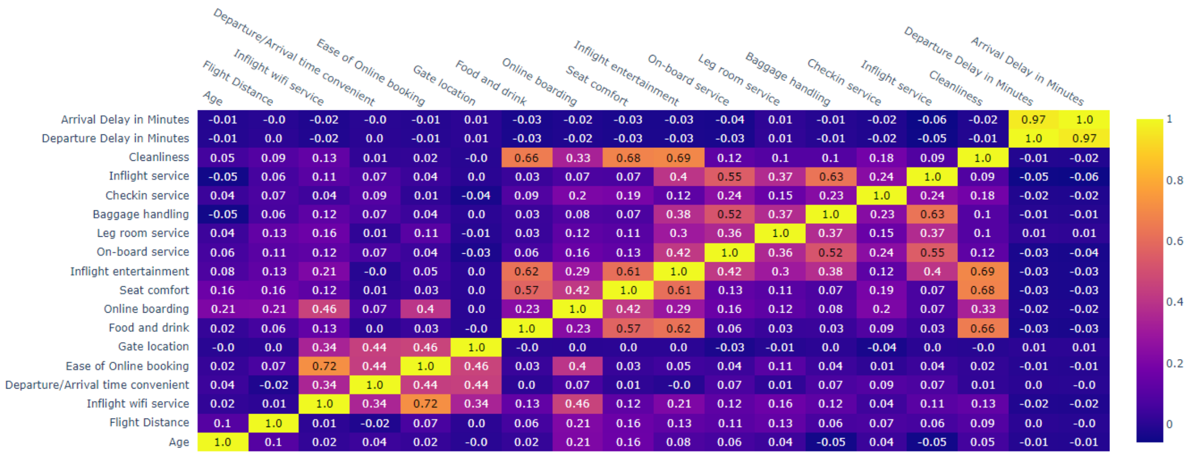
Part 1: Classification of EireJet.csv using Random Forest method, AdaBoost, Gradient Boost methods

1. Data Preparation

1. Imported all the required libraries for the code and then uploaded Eirjet dataset to the environment.
2. Analysed dataset using methods dataset.head(), dataset.info(), dataset.describe() methods.
3. Analysed all the highly correlated variables using heat map method.



- Arrival Delay in Minutes has Null values instead of replacing Null values with Mean or Mode dropping as it was high correlated value
- Dropping columns Arrival Delay in Minutes, Cleanliness
- Column/variable Arrival Delay in Minutes is highly correlated with Departure Delay in Minutes and there is high causation also.
- Column/variable Cleanliness has high correlation with Inflight entertainment, Seat Comfort and Food and Drink and cleanliness has causation with Food and Drink .So dropped columns Arrival Delay in Minutes and Cleanliness

4. Now converting all the categorical variables into numerical values by mapping method as fallows

```
[ ] # Converting Categorical features into Numerical features
    dataset1['Gender'] = dataset1['Gender'].map({'Male':0, 'Female':1})
    dataset1['Frequent Flyer'] = dataset1['Frequent Flyer'].map({'No':0, 'Yes':1})
    dataset1['Type of Travel'] = dataset1['Type of Travel'].map({'Personal Travel':0, 'Business travel':1})
    dataset1['Class'] = dataset1['Class'].map({'Eco':0,'Eco Plus':0,'Business':1})
    dataset1['satisfaction'] = dataset1['satisfaction'].map({'neutral or dissatisfied':0, 'satisfied': 1})
    print(dataset1.head(5))
    print(dataset1.isnull().sum())
```

```
   Gender  Frequent Flyer  ...  Departure Delay in Minutes  satisfaction
0       0               1  ...                          25             0
1       0               0  ...                           1             0
2       1               1  ...                           0             1
3       1               1  ...                          11             0
4       0               1  ...                           0             1
```

5. Now dividing the dataset into feature and label sets

```
[ ] # Dividing dataset into label and feature sets
    X = dataset1.drop(['satisfaction'], axis = 1) # Features
    Y = dataset1['satisfaction'] # Labels
    print(type(X))
    print(type(Y))
    print(X.shape)
    print(Y.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
(103904, 20)
(103904,)
```

6. Normalise feature set feature scaling

7. Divide the dataset into train and test sets as fallows

```
# Normalizing numerical features so that each feature has mean 0 and variance 1
feature_scaler = StandardScaler()
X_scaled = feature_scaler.fit_transform(X)
```

```
[ ] # Dividing dataset into training and test sets
    X_train, X_test, Y_train, Y_test = train_test_split( X_scaled, Y, test_size = 0.3, random_state = 100)
    print(X_train.shape)
    print(X_test.shape)
```

```
(72732, 20)
(31172, 20)
```

8. Balancing the training dataset by smote method

```
[ ] # Implementing Oversampling to balance the dataset; SMOTE stands for Synthetic Minority Oversampling TEchnique
    print("Number of observations in each class before oversampling (training data): \n", pd.Series(Y_train).value_counts())
    smote = SMOTE(random_state = 101)
    X_train,Y_train = smote.fit_sample(X_train,Y_train)
```

```
Number of observations in each class before oversampling (training data):
 0    41204
1    31528
Name: satisfaction, dtype: int64
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:

Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.
```

```
[ ] print("Number of observations in each class after oversampling (training data): \n", pd.Series(Y_train).value_counts())
```

```
Number of observations in each class after oversampling (training data):
 1    41204
0    41204
dtype: int64
```

2. Model Building and Testing (including hyper parameter tuning) and Model Evaluation Strategy

1. Find the optimal number of trees to be grown in random forest by grid search method
2. Here I am are trying to reduce the “False Positives” in the dataset as predicting the unsatisfied passenger as satisfied passenger will be problematic for EireJet Airlines as they end up with some unsatisfied customers and not take right measures for the same .So using “Scoring=Precision” to minimise false positive.

```
rfc = RandomForestClassifier(criterion='entropy', max_features='auto', random_state=1)
grid_param = {'n_estimators': [50, 100, 150, 200, 250, 300]}

gd_sr = GridSearchCV(estimator=rfc, param_grid=grid_param, scoring='precision', cv=5)
```

3. Fit the grid search value to training dataset
4. Find the optimal value of n_estimator to find out the how many optimal trees have to grow and find the best result – optimal value is 150

```
[ ] best_parameters = gd_sr.best_params_
    print(best_parameters)

    {'n_estimators': 150}

[ ] best_result = gd_sr.best_score_ # Mean cross-validated score of the best_estimator
    print(best_result)
    print(type(best_result))

    0.9766959352657733
    <class 'numpy.float64'>
```

5. After finding the optimal value of n_estimator use it in RFC classifier and predict the values

Random Forest Method:

In Random forest method grid search is run to find n_estimator and keeping value cv=5 ,apply the optimal n_estimator value which is 150 in RFC classifier and predict value without removing any features and then we remove features which are insignificant and predict values and repeat this step of removing features and predicting values till we find optimal false positive value.

N_estimator	Features removed	False Positive value/result
150	Arrival Delay in Minutes, Cleanliness(initially dropped columns)	383
150	Arrival Delay in Minutes, Cleanliness ,Gender, Food and Drink, Departure Delay in Minutes ,Gate location, Departure/Arrival time convenient, Inflight service	463

150	Arrival Delay in Minutes, Cleanliness ,Gender, Food and Drink, Departure Delay in Minutes ,Gate location, Departure/Arrival time convenient	404
150	Arrival Delay in Minutes, Cleanliness ,Gender, Food and Drink, Departure Delay in Minutes ,Gate location	378

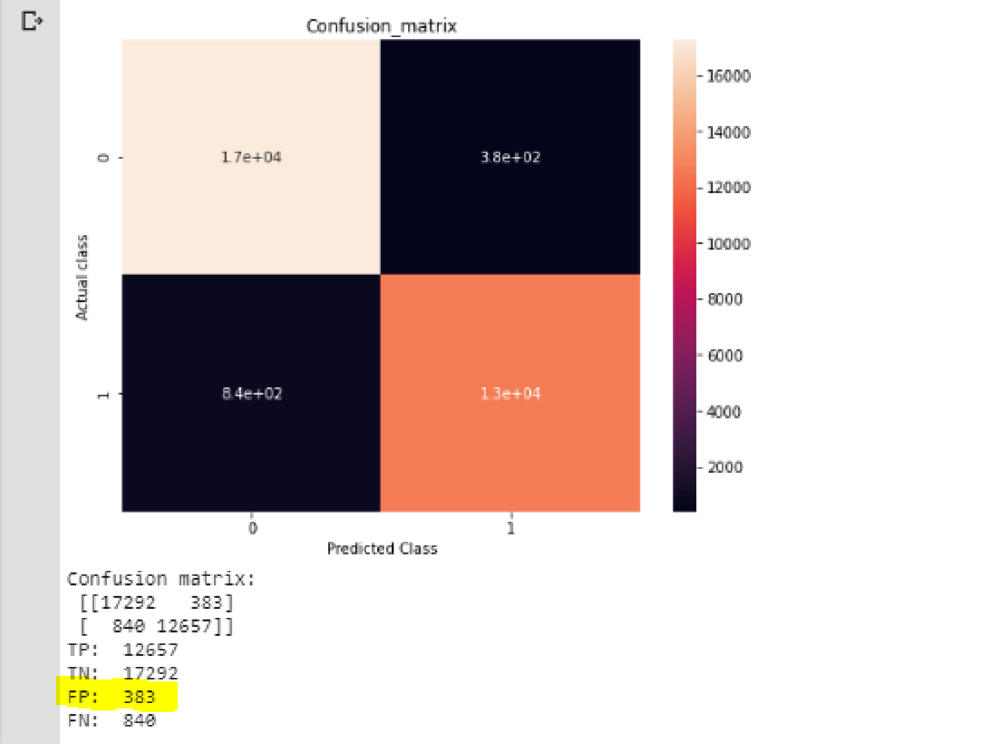


Figure above shows the confusion matrix created without removing any features

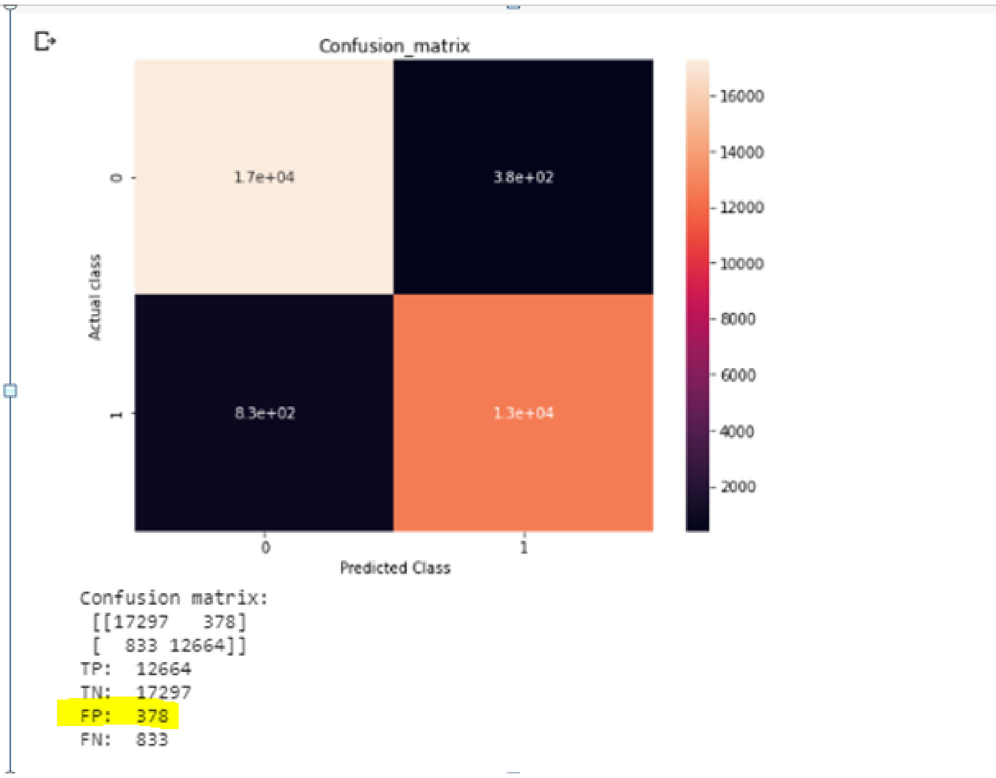
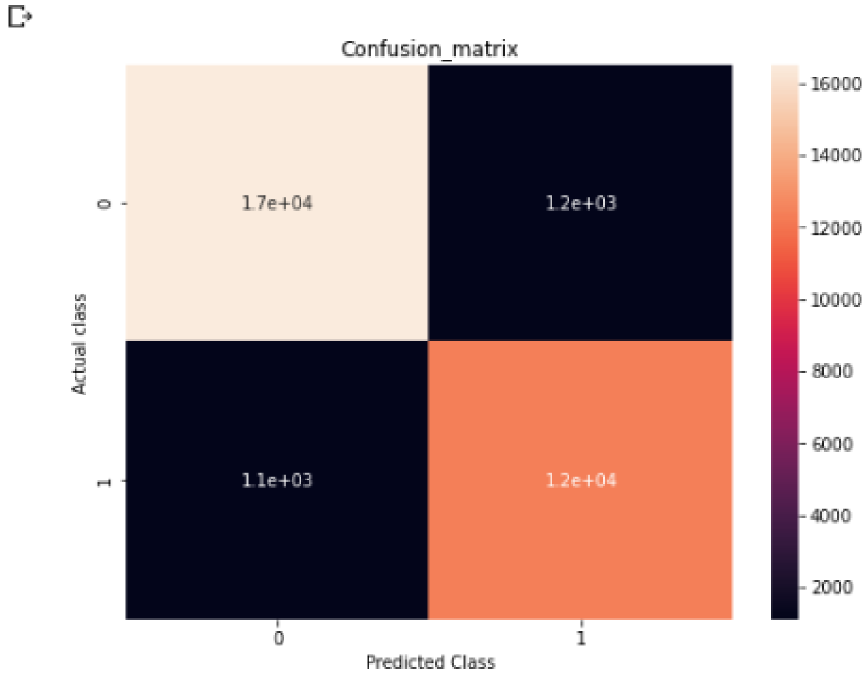


Figure shows the optimal value of false positive 378 after number of trials of removing insignificant features

ADA Boost :

Grid search is performed to find the optimal value of n_estimator values.the values set were CV=5 and `scoring=`'precision'. The Ada boost classifier was run using tuned parameter which is n_estimator =50 and predict the values to find the false positive values.Ada boost value False positives arevery high compared to Random forest method.




```
Confusion matrix:
 [[16513  1162]
 [ 1106 12391]]
TP:  12391
TN:  16513
FP:  1162
FN:  1106
```

Figure above shows the high false positive values

Gradient Boosting:

Perform the grid search provide different parameters to grid_param and max_depth to find optimal value for both to pass to grid search .in Grid search values set are scoring='precision' cv=5.

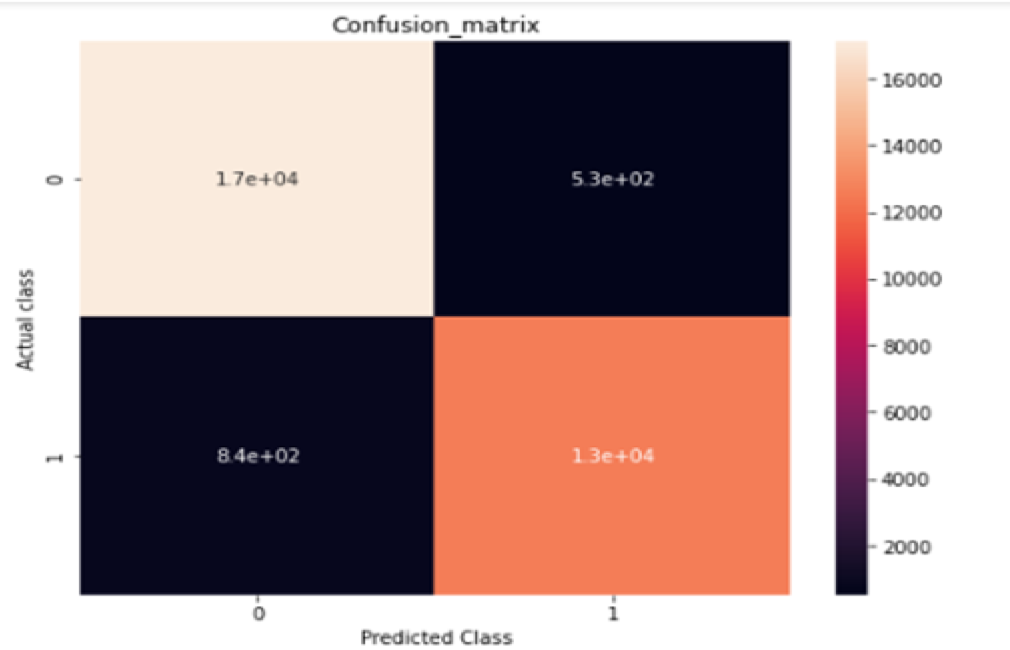
N_estimator	Max_depth	False Positive
50	9	532
60	12	458
80	16	421
80	18	403
90	20	409
90	18	393
100	18	396

Above Table shows different values of false positive results for different values of n_estimator or number of trees and Max_depth of the tree

```
# # Tuning the Gradent Boost parameter 'n_estimators' and implementing cross-validation using Grid Search
gbc = GradientBoostingClassifier(random_state=1)
grid_param = {'n_estimators': [10,20,30,40,50], 'max_depth' : [5,6,7,8,9,10,11,12], 'max_leaf_nodes': [8,12,16,20,24,28,3

gd_sr = GridSearchCV(estimator=gbc, param_grid=grid_param, scoring='recall', cv=5)

# In the above GridSearchCV(), scoring parameter should be set as follows:
# scoring = 'accuracy' when you want to maximize prediction accuracy
# scoring = 'recall' when you want to minimize false negatives
# scoring = 'precision' when you want to minimize false positives
# scoring = 'f1' when you want to balance false positives and false negatives (place equal emphasis on minimizing both)

gd_sr.fit(X_train, Y_train)

best_parameters = gd_sr.best_params_
print(best_parameters)

best_result = gd_sr.best_score_ # Mean cross-validated score of the best_estimator
print(best_result)
```

```
{'max_depth': 9, 'max_leaf_nodes': 32, 'n_estimators': 50}
0.9462921215612677
```

Figure above shows initial grid search performed to find n_estimator and max_depth

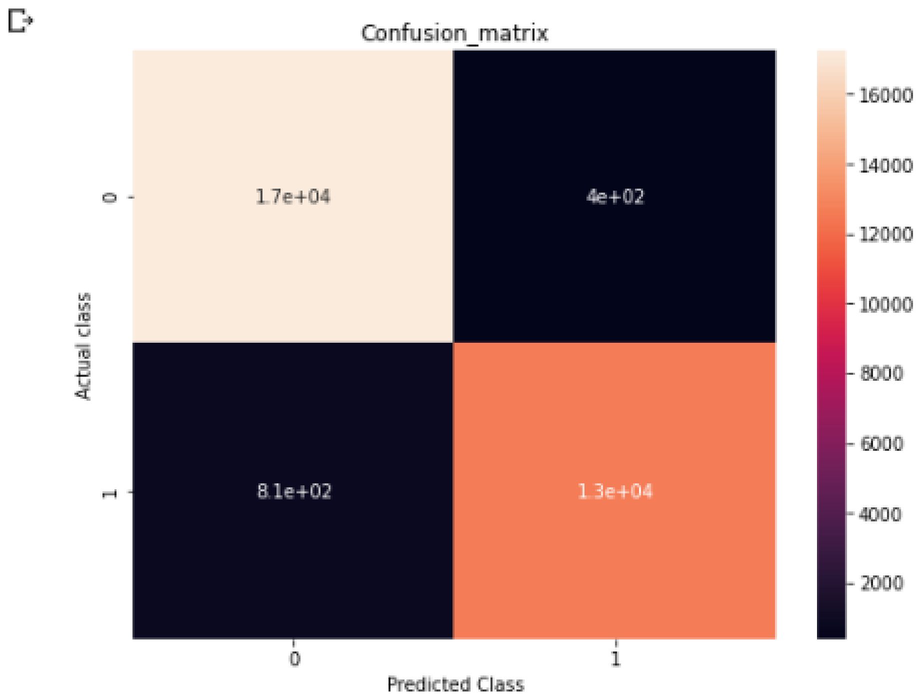



```
Confusion matrix:
 [[17143   532]
 [  840 12657]]
TP:   12657
TN:   17143
FP:   532
FN:   840
```

Figure above show confusion matrix obtained for n_estimator=50 and max_depth =9 which is 532

```
[14] # # Tuning the Gradent Boost parameter 'n_estimators' and implementing cross-validation using Grid Search
     gbc = GradientBoostingClassifier(random_state=1)
     grid_param = {'n_estimators': [80,90,100,110,120], 'max_depth' : [16,17,18], 'max_leaf_nodes': [8,12,16,20,24,28,32]}

     gd_sr = GridSearchCV(estimator=gbc, param_grid=grid_param, scoring='precision', cv=5)

     # In the above GridSearchCV(), scoring parameter should be set as follows:
     # scoring = 'accuracy' when you want to maximize prediction accuracy
     # scoring = 'recall' when you want to minimize false negatives
     # scoring = 'precision' when you want to minimize false positives
     # scoring = 'f1' when you want to balance false positives and false negatives (place equal emphasis on minimizing both)

     gd_sr.fit(X_train, Y_train)

     best_parameters = gd_sr.best_params_
     print(best_parameters)

     best_result = gd_sr.best_score_ # Mean cross-validated score of the best_estimator
     print(best_result)

     {'max_depth': 18, 'max_leaf_nodes': 32, 'n_estimators': 100}
     0.9757148176815063
```

Figure above show optimal n_estimator value as 100 and max_depth as 18




```
Confusion matrix:
 [[17279   396]
 [  807 12690]]
TP:  12690
TN:  17279
FP:  396
FN:  807
```

Figure above show false positive value 396 for n_estimator =100 and max_depth =18

3. Optimal Model:

After performing Random Forest Method, Ada Boost and gradient boosting method with optimal values obtained from grid search (n_estimator,max_depth) Random forest method proved to be best in reducing the false positives and gives significant variables as well.

Below are optimal values obtained from different methods:

Random Forest method:

N_estimator	Features removed	False Positive value/result
150	Arrival Delay in Minutes, Cleanliness ,Gender, Food and Drink, Departure Delay in Minutes ,Gate location	378

Ada Boost:

N_estimator	False Positive value/result
50	1162

Gradient Boost

N_estimator	Max_depth	False Positive
100	18	396

Features selected:

Below are the final significant features selected in Random Forest method to get 396 False Positive

X1 = dataset1[['Frequent Flyer','Checkin service','Age','Type of Travel','Class','Flight Distance','Inflight wifi service','Ease of Online booking','Online boarding','Seat comfort','On-board service','Departure/Arrival time convenient','Inflight service','Leg room service','Baggage handling','Inflight entertainment']]

4. General Recommendations:

Eirjet Airlines Should focus on few factors to improve the satisfaction of its passengers.

- Arrival delay and departure delay are one of the major factors for passenger dissatisfaction. The whole schedule of the passengers might get disturbed due to delayed arrivals some time passengers tend to miss connecting flights due to the same.
- Cleanliness is one more important factor in passenger dissatisfaction. Airline should concentrate on cleanliness especially in terms of quality and service of food and drink.
- Gate location is one more major factor often leads to customer dissatisfaction.
- In addition Airline should try to focus on factors which customers are happy to increase passenger satisfaction especially factors like Inflight Wifi Service, Online checkin options, Inflight entertainment ,seat comfort.

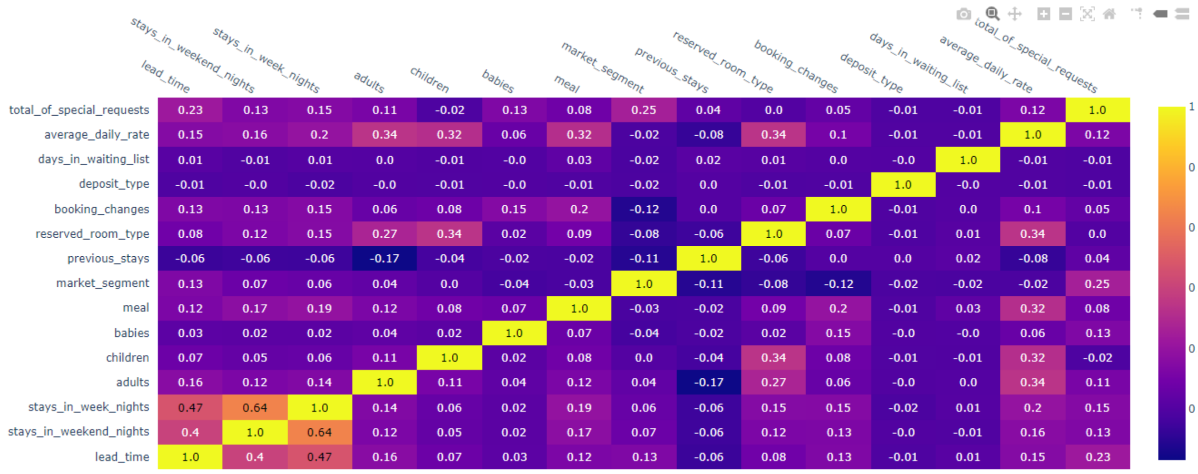
Part 2: Unsupervised Data Mining

1. Data Preparation

1. Imported all the required libraries for the code and then uploaded Eirstay dataset to the environment.

2. Analysed dataset using methods dataset.head(), dataset.info(), dataset.describe() methods.

3. Analysed all the highly correlated variables using heat map method.



- Column/variable stays_in_week_nights is highly correlated with stays_in_weekend_nights however not dropping the column as causation is not clear.

4. Now converting all the categorical variables into numerical values by mapping method as fallows. As mapping method will not increase dimensions used mapping method instead of dummies method

```
# Converting Categorical features into Numerical features
dataset['meal'] = dataset['meal'].map({'BB':0, 'FB':1, 'HB':2, 'SC':3,'Undefined':4})
dataset['market_segment'] = dataset['market_segment'].map({'Direct':0, 'Online TA':1})
dataset['reserved_room_type'] = dataset['reserved_room_type'].map({'A':0, 'B':1, 'C':2, 'D':3, 'E':4, 'F': 5, 'G':6, 'H':7, 'I':8})
dataset['deposit_type'] = dataset['deposit_type'].map({'No Deposit':0, 'Non Refund':1, 'Refundable':2})
print(dataset.head(5))
```

```
   lead_time  ...  total_of_special_requests
0        342  ...                          0
1        737  ...                          0
2          7  ...                          0
3         14  ...                          1
4         14  ...                          1

[5 rows x 15 columns]
```

5. Normalizing numerical features so that each feature has mean 0 and variance 1

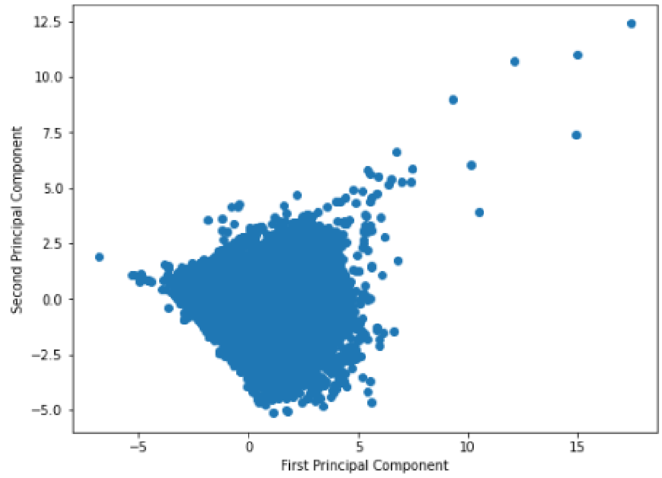
```
# Normalizing numerical features so that each feature has mean 0 and variance 1

feature_scaler = StandardScaler()

X_scaled = feature_scaler.fit_transform(dataset)
```

2. t-SNE Implementation and why t-SNE

1. Tried to recognise the pattern using PCA method as fallows, however clear pattern or cluster is not visible as data points are overlapping as it is non – linear dataset. And Total variance explained is only 28 % .

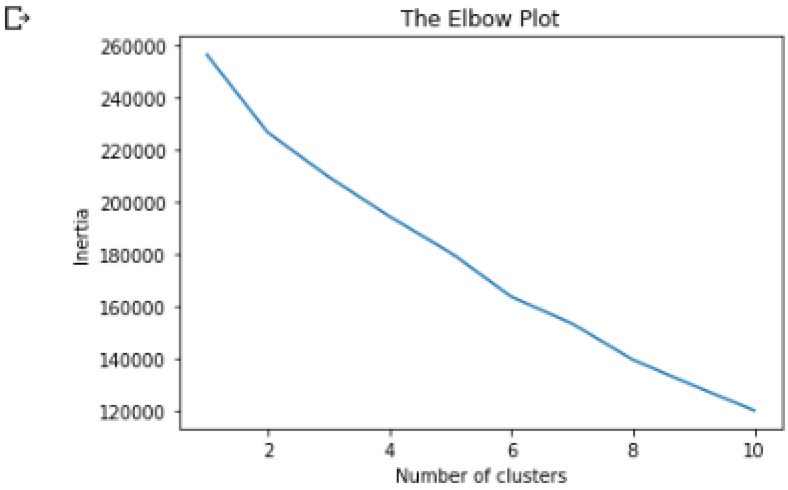
   That's why we used t-SNE method as it can handle non – linear datasets.

```
Variance explained by each of the n_components:  [0.17790746 0.10681466]
Total variance explained by the n_components:  0.28472212341062214
```



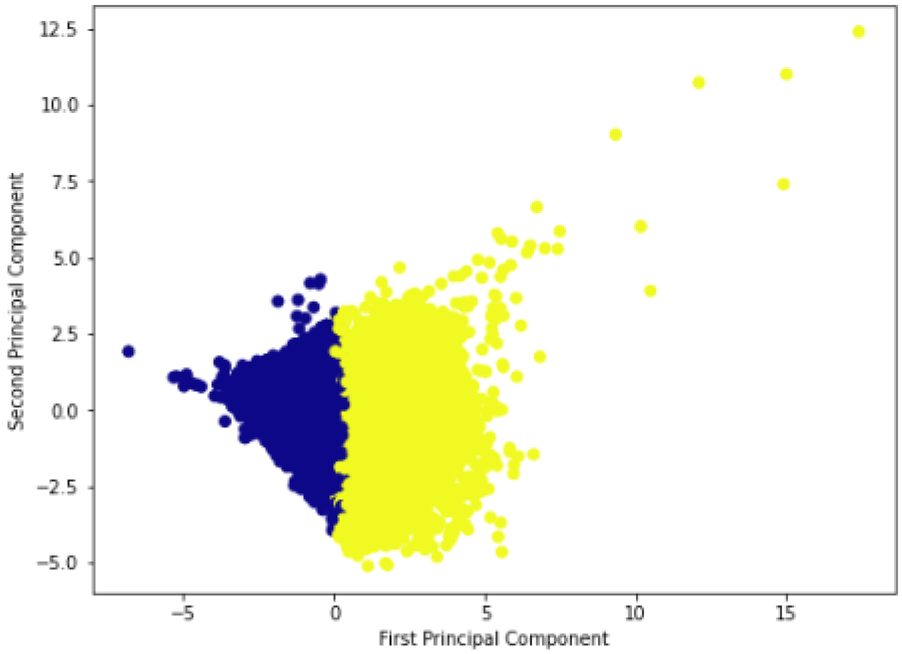
2. Plot elbow graph to see the number of clusters as shown below 2 clusters were formed

```
[18] inertia = []
     for i in range(1,11):
         kmeans = KMeans(n_clusters = i, random_state = 100)
         kmeans.fit(X_scaled)
         inertia.append(kmeans.inertia_)

     plt.plot(range(1, 11), inertia)
     plt.title('The Elbow Plot')
     plt.xlabel('Number of clusters')
     plt.ylabel('Inertia')
     plt.show()
```



3. Tried to label using k means cluster method as fallows

```
Cluster Centers:
 [[-0.4460819  -0.42070158 -0.45338725 -0.25334464 -0.26016651 -0.09296943
  -0.29756931 -0.05683777  0.06147342 -0.3059971  -0.19347669  0.01186187
  -0.01544681 -0.40363474 -0.18453645]
 [ 0.6776925   0.63913445  0.68879088  0.38488395  0.39524782  0.14124017
   0.45207055  0.08634856 -0.09339109  0.46487415  0.29393191 -0.01802068
   0.02346697  0.61320631  0.2803498 ]]
```



As shown in above figures the data points are overlapping totally as data in non-linear. Hence going for t-SNE visualisation to visualise non-linear data.

3. Clustering Implementation

K-Means clustering is used in this problem for clustering. Kmeans clustering method is widely used clustering method which divides n observations into k clusters. It is used to cluster data points/observations into meaningful groups. K-means clustering in used in t-SNE to label the data points.

4. t-SNE tuning

I have tried different subsets ,perplexity and n_itr values to find the clusters

```
subset1 = dataset[['adults','babies','lead_time','stays_in_week_nights'
,'stays_in_weekend_nights','booking_changes','average_daily_rate']]
```

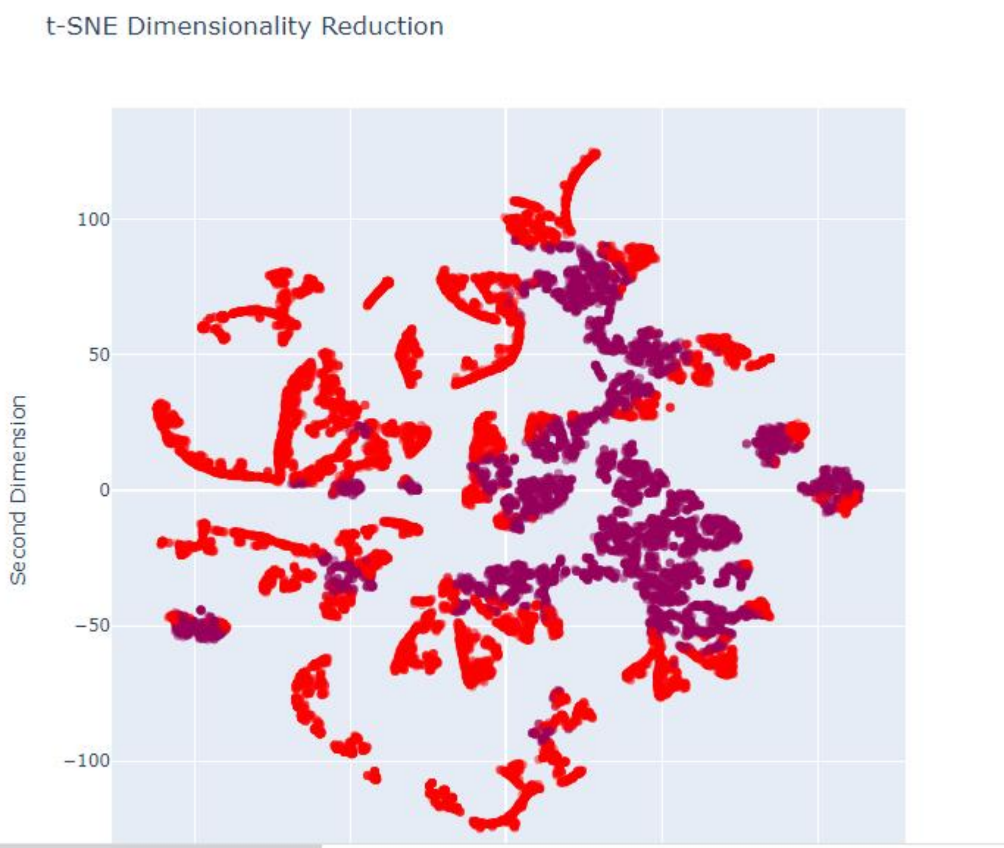
Perplexity and n_iterations tried :

```
tsne = TSNE(n_components = 2, perplexity =50,n_iter=2000)
x_tsne = tsne.fit_transform(X1)
```

```
[42] # Implementing t-SNE to visualize dataset
     tsne = TSNE(n_components = 2, perplexity =50,n_iter=2000)
     x_tsne = tsne.fit_transform(X1)
```

```
(dataset['reserved_room_type'])
list(dataset['average_daily_rate'])
t(dataset['adults'])
t(dataset['babies'])
set['lead_time'])
ist(dataset['stays_in_weekend_nights'])
(dataset['stays_in_week_nights'])
taset['booking_changes'])
st(dataset['average_daily_rate'])
list(dataset['days_in_waiting_list'])

catter(x=x_tsne[:,0], y=x_tsne[:,1], mode='markers',
        marker = dict(color=kmeans.labels_, colorscale='Rainbow', opacity=0.5),
                    text=[f'adults:{a}; babies:{b};lt:{c};sweekend:{d}; sweek:{e}; bc:{f};avgrate:{g}' for a,b,c,d,e,f,g in list(zip(adults,babies,lt,sweekend,sweek,b
                    hoverinfo='text')]

layout(title = 't-SNE Dimensionality Reduction', width = 700, height = 700,
        xaxis = dict(title='First Dimension'),
        yaxis = dict(title='Second Dimension'))
ure(data=data, layout=layout)
(fig,filename='t-SNE1.html')
```

```
't-SNE1.html'
```

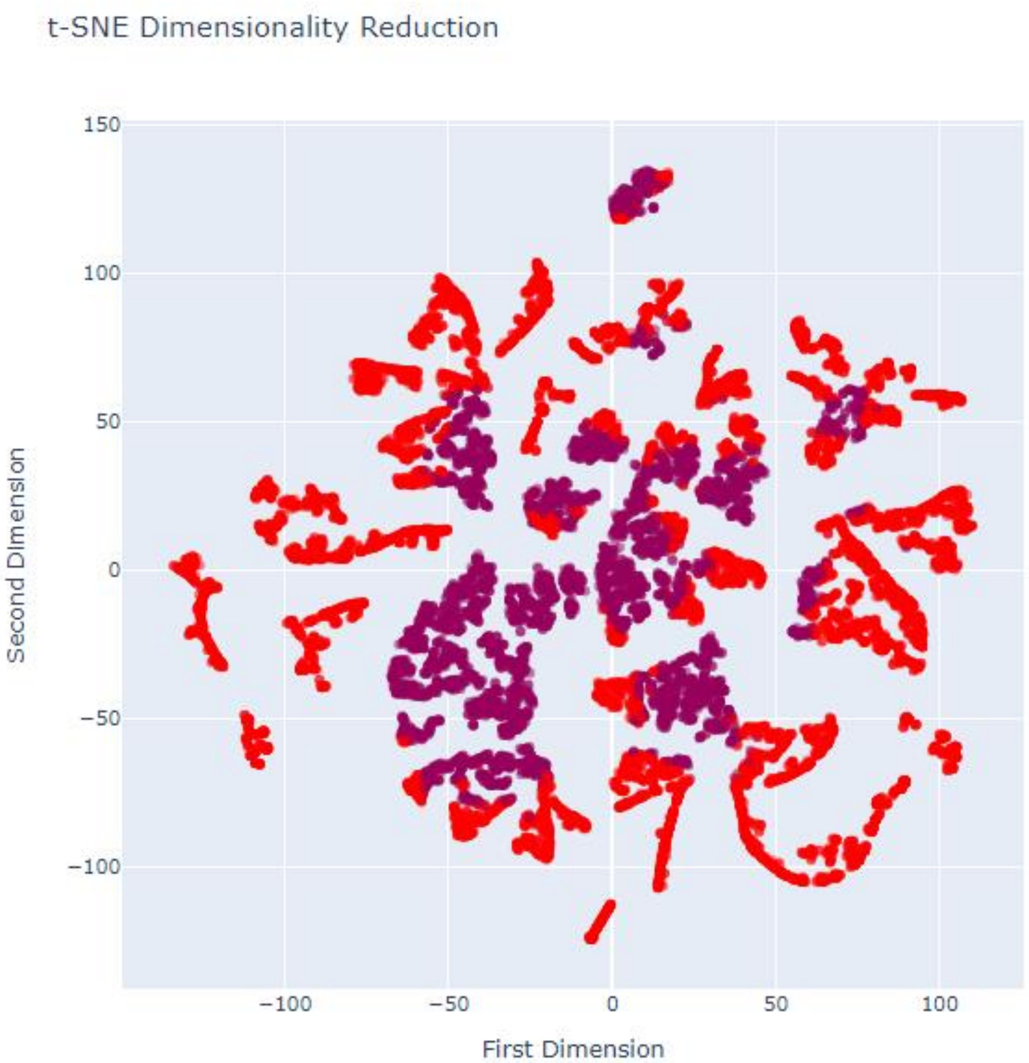


```
tsne = TSNE(n_components = 2, perplexity =50,n_iter=2000)
x_tsne = tsne.fit_transform(X1)
```

```
[45] # Implementing t-SNE to visualize dataset
     tsne = TSNE(n_components = 2, perplexity =40,n_iter=2000)
     x_tsne = tsne.fit_transform(X1)
```

```
(dataset['reserved_room_type'])
list(dataset['average_daily_rate'])
t(dataset['adults'])
t(dataset['babies'])
set['lead_time'])
ist(dataset['stays_in_weekend_nights'])
(dataset['stays_in_week_nights'])
taset['booking_changes'])
st(dataset['average_daily_rate'])
list(dataset['days_in_waiting_list'])

catter(x=x_tsne[:,0], y=x_tsne[:,1], mode='markers',
        marker = dict(color=kmeans.labels_, colorscale='Rainbow', opacity=0.5),
                    text=[f'adults:{a}; babies:{b};lt:{c};sweekend:{d}; sweek:{e}; bc:{f};avgrate:{g}' for a,b,c,d,e,f,g in list(zip(adults,babies,lt,swee
                    hoverinfo='text')]

Layout(title = 't-SNE Dimensionality Reduction', width = 700, height = 700,
        xaxis = dict(title='First Dimension'),
        yaxis = dict(title='Second Dimension'))
ure(data=data, layout=layout)
(fig,filename='t-SNE1.html')
```

```
't-SNE1.html'
```

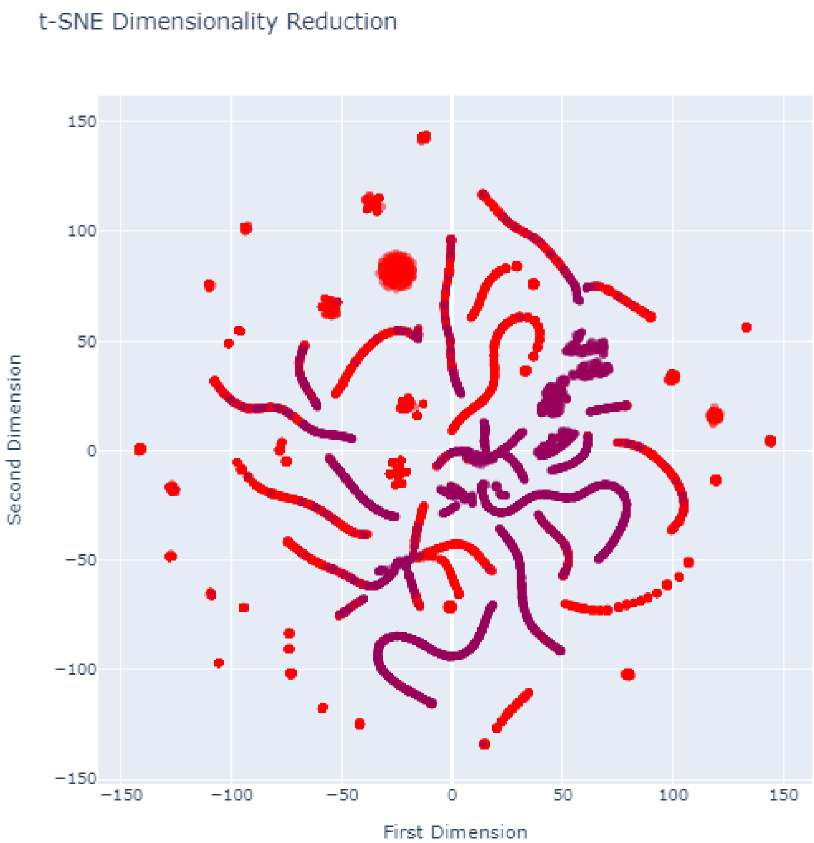


```
[26] # Implementing t-SNE to visualize dataset
tsne = TSNE(n_components = 2, perplexity =300,n_iter=2000)
x_tsne = tsne.fit_transform(X1)
```

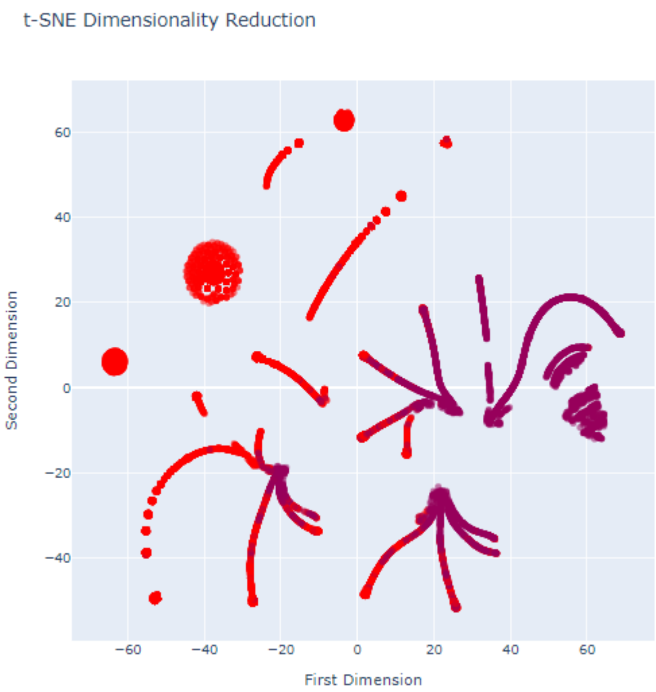
```
# rrt = list(dataset['reserved_room_type'])
# avgrate = list(dataset['average_daily_rate'])
adults = list(dataset['adults'])
children = list(dataset['children'])
lt=list(dataset['lead_time'])
sweekend = list(dataset['stays_in_weekend_nights'])
sweek = list(dataset['stays_in_week_nights'])
ms = list(dataset['market_segment'])
# days_wl = list(dataset['days_in_waiting_list'])

data = [go.Scatter(x=x_tsne[:,0], y=x_tsne[:,1], mode='markers',
                   marker = dict(color=kmeans.labels_, colorscale='Rainbow', opacity=0.5),
                         text=[f'adults:{a};children:{b};lt:{c};sweekend:{d}; sweek:{e};ms{f}' for a,b,c,d,e,f in list(zip(adults,children,lt,sweekend,sweek,ms))],
                         hoverinfo='text')]

layout = go.Layout(title = 't-SNE Dimensionality Reduction', width = 700, height = 700,
                   xaxis = dict(title='First Dimension'),
                   yaxis = dict(title='Second Dimension'))
fig = go.Figure(data=data, layout=layout)
offline.plot(fig,filename='t-SNE1.html')
```

```
't-SNE1.html'
```

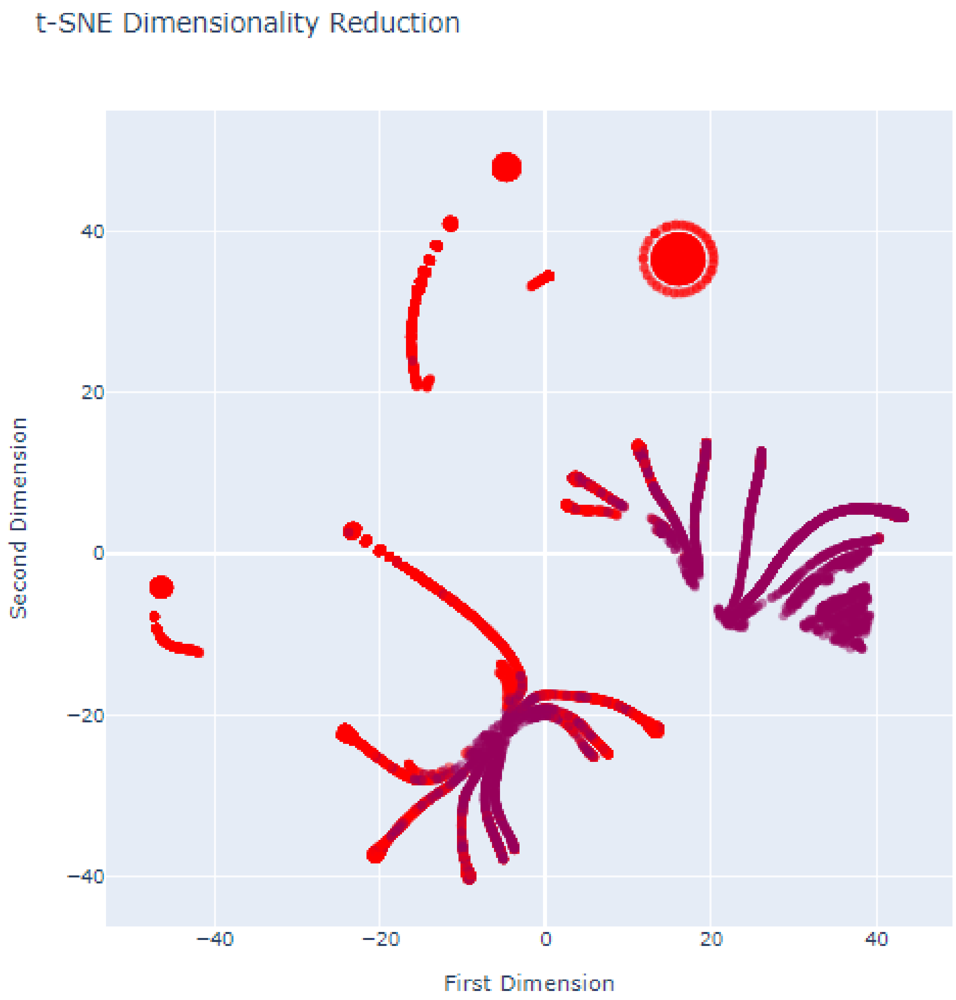
```
#based on corelated data
subset2 = dataset[['lead_time','stays_in_weekend_nights','stays_in_week_nights']]
tsne = TSNE(n_components = 2, perplexity =50,n_iter=2000)
x_tsne = tsne.fit_transform(X2)
```



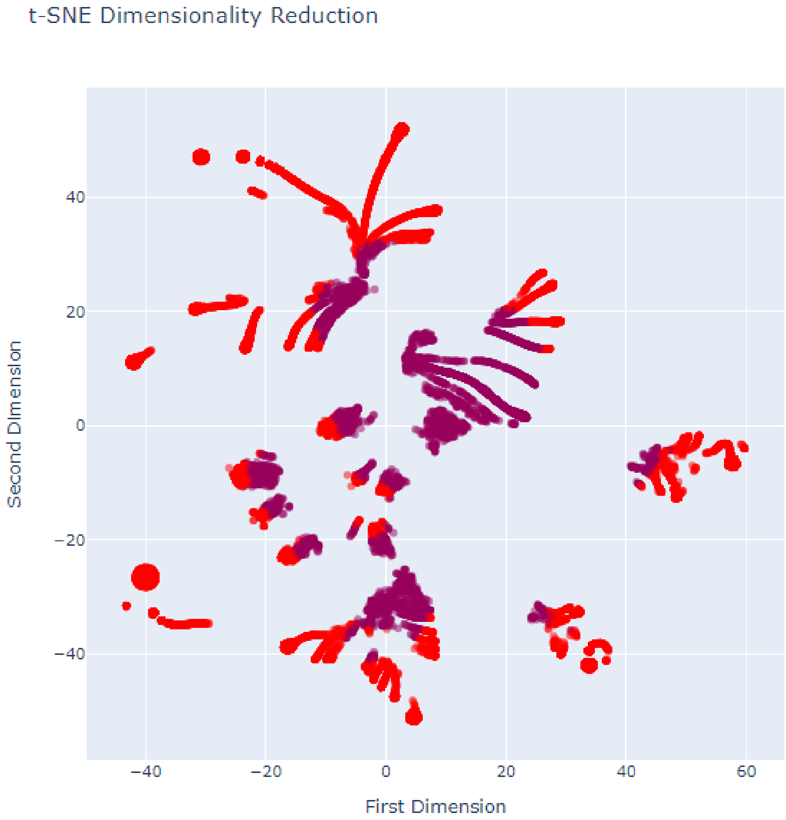

```
tsne = TSNE(n_components = 2, perplexity =200,n_iter=2000)
x_tsne = tsne.fit_transform(X2)
```




```
tsne = TSNE(n_components = 2, perplexity =400,n_iter=2000)
x_tsne = tsne.fit_transform(X2)
```

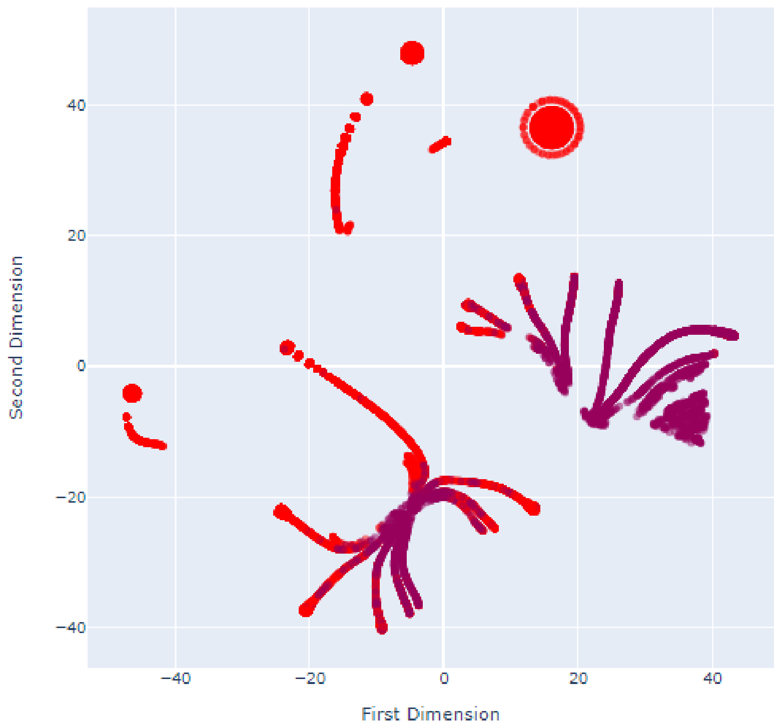

t-SNE Dimensionality Reduction
Second Dimension
First Dimension
40
20
0
−20
−40
−40
−20
0
20
40

```
subset3 = dataset[['adults','children','stays_in_week_nights','stays_in_weekend_nights','lead_time','market_segment']]

tsne = TSNE(n_components = 2, perplexity =300,n_iter=2000)
x_tsne = tsne.fit_transform(X1)
```



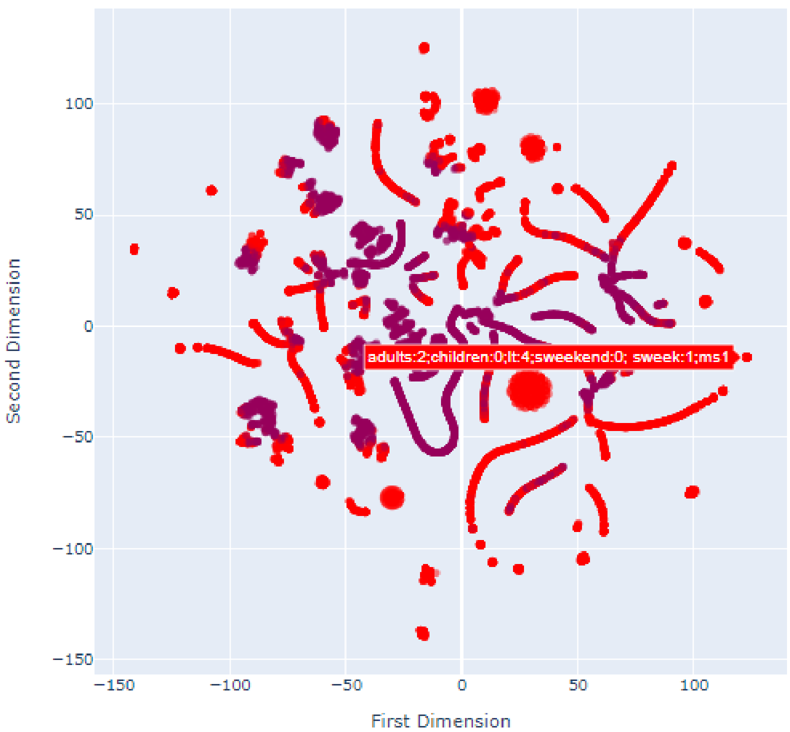

```
tsne = TSNE(n_components = 2, perplexity =400,n_iter=2000)
x_tsne = tsne.fit_transform(X1)
```

t-SNE Dimensionality Reduction




```
tsne = TSNE(n_components = 2, perplexity =300,n_iter=2000)
x_tsne = tsne.fit_transform(X1)
```

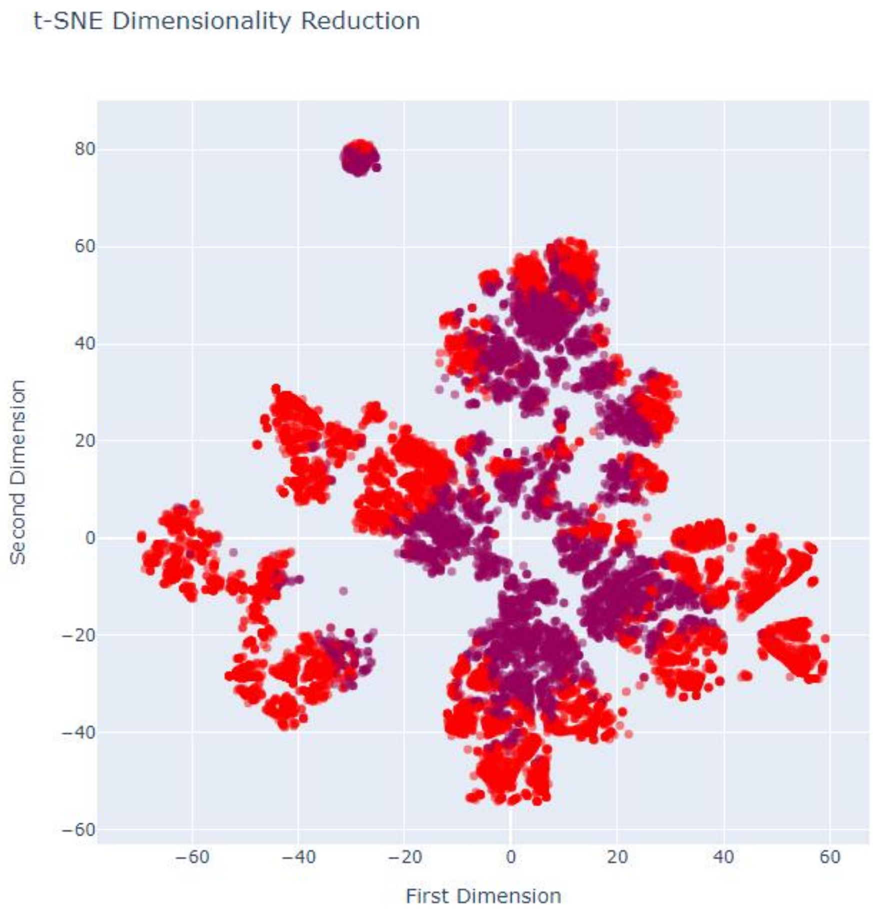
t-SNE Dimensionality Reduction



Subset 4:

#Entire dataset

```
#Entire dataset
subset4 = dataset[['reserved_room_type','average_daily_rate','adults','
children', 'babies','market_segment', 'booking_changes','deposit_type',
 'days_in_waiting_list','previous_stays','stays_in_weekend_nights','lea
d_time','stays_in_week_nights','meal','total_of_special_requests']]

tsne = TSNE(n_components = 2, perplexity =200,n_iter=2000)
x_tsne = tsne.fit_transform(X4)
```



5. Cluster Interpretations :
   After trying three subsets and entire dataset with different perplexity and n_iter values unable to form any clear clusters. The clusters are scattered, overlapped. Hence cannot assign any labels to resultant variables.

   As no clear clusters are formed cannot find any target variable for dataset or subset.