CAPSTONE PROJECT

PREDICTING CUSTOMER EVENT RESPONSE USING CLASSIFICATION MODELS FOR A CUSTOMER-TRANSACTIONS PROJECT

Contents

Introduction

The Customer-Transactions project is a data visualization and analytics initiative that utilizes a sample dataset that simulates customer transaction data for a fictional office supplies store, covering sales, profit, orders, and customers.

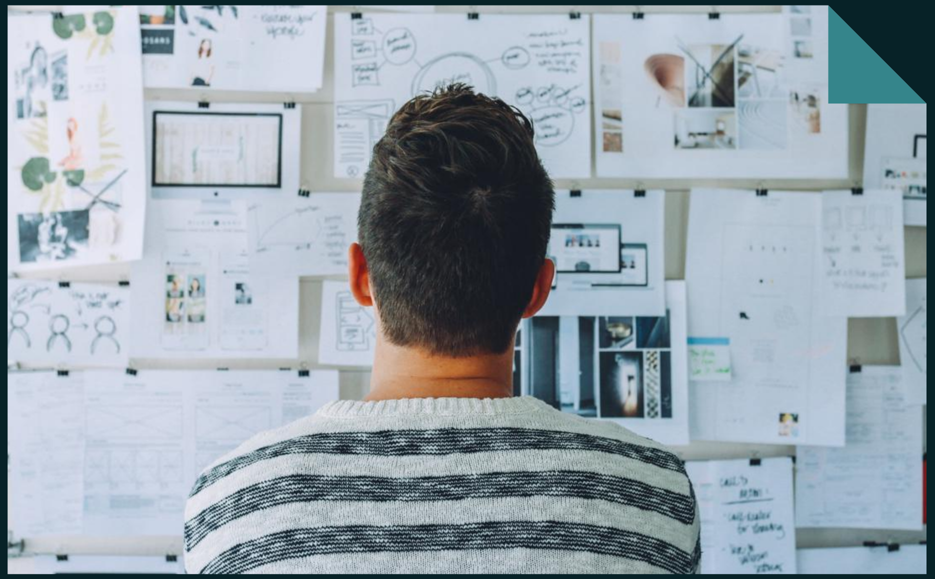
Dataset

The given dataset consists of two Excel files named 'Customer by Store' and 'Transactions'

2851 Rows in Customer by Store
9426 Rows in Transactions

Customer by Store	Transactions
Customer ID	Customer ID
Store Number	Order_ID
Customer Segment	Order_Priority
Responder	Discount
Name	Unit_Price
Address	Quantity_ordered_new
Postcode	Order_Date
CustomerType	Ship_Date
AverageTransaction	Shipping_Cost
LastTransactionAmount	Ship_Mode
TransactionsPerYear	Product_Category
Income	Product_Sub-Category
HomeValue	Product_Container
Lat	Product_Base_Margin
Lon	Profit

Insights



The manager from the sales department would like to gain insights from the data. Some of the questions are as follows:

- What is the profit by customer segment and store no.?
- Which store has the highest customer?
- How many transactions do not match with customer info?
- Estimate (predict) the responder (Yes/No) based on the customer segment and another appropriate input
- What is the summarised value (Total) for Average Transaction, Last Transaction and Transaction Per Year by customer type?

Data Preprocessing

Load Data

```python
# read the excel file and put it into CustomerStore dataframe
CustomerStore = pd.ExcelFile("Customers by Store.xlsx")

# List all sheet names
sheet_names = CustomerStore.sheet_names

# Read all sheets into a list of DataFrames
CustomerByStore = [pd.read_excel(CustomerStore, sheet_name=sheet) for sheet in sheet_names]

# Concatenate all DataFrames into one
combined_df = pd.concat(CustomerByStore, ignore_index=True)

# Save the combined DataFrame to a new Excel file (in Google Colab environment)
combined_df.to_excel('combined_output.xlsx', index=False)
```

```python
Transactions = pd.read_excel("transactions.xlsx")
Transactions.head()
```

Merge Data

```python
# Load the two Excel files
file1 = 'transactions.xlsx'
file2 = 'combined_output.xlsx'

# Read the specific sheets or the first sheet of each file
Transactions = pd.read_excel(file1, sheet_name='sheet1')
combined_df = pd.read_excel(file2, sheet_name='Sheet1')

# Merge the two DataFrames based on 'Customer ID' using inner join. This is to make sure that only the data with a valid Customer ID will be merged.
mergedfile_df = pd.merge(Transactions, combined_df, left_on='Customer_ID', right_on='Customer ID', how='inner')
```

Data Preprocessing

Clean Data

```python
# Check missing values, are all values 0?
merged_df.isna().sum()
```

```python
# Remove rows with missing values.
merged_df.dropna(inplace=True)
```

```python
CustTransaction.info()
```

```python
# Convert Order_Date to datetime (assuming the format is mmddyyyy)
CustTransaction['Order_Date'] = pd.to_datetime(CustTransaction['Order_Date'])
CustTransaction['Ship_Date'] = pd.to_datetime(CustTransaction['Ship_Date'])
```

```python
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# Select all the columns with object datatype that need to be encoded
col = ['Order_Priority', 'Ship_Mode', 'Product_Category',
'Product_Sub-Category', 'Product_Container', 'Customer Segment', 'Responder',
'Postcode', 'CustomerType']

# Transform the data
for i in col:
    CustTransactionClassification[i] = encoder.fit_transform
    (CustTransactionClassification[i])
```

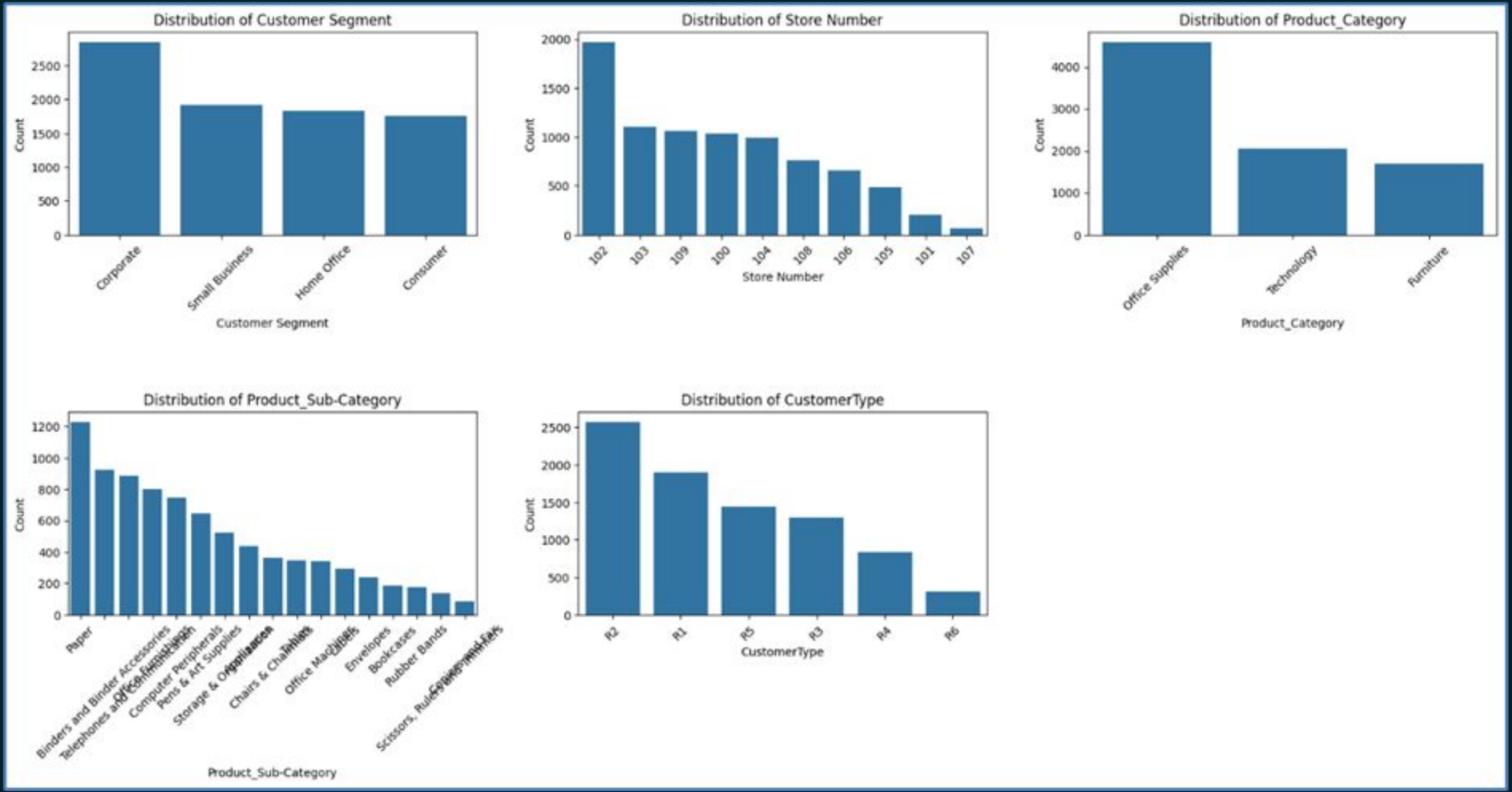
Export Data

```python
CustTransaction.to_excel('CustTransaction.xlsx', index=False)
```

```python
from google.colab import files
files.download('CustTransaction.xlsx')
```

Exploratory Data Analysis

Data Distribution



Exploratory Data Analysis

Data Correlation - Spearman rank's correlation coefficient

	Discount	Unit_Price	Quantity_ordered_new	Shipping_Cost	Product_Base_Margin	Profit	AverageTransaction	LastTransactionAmount	TransactionsPerYear	Income	HomeValue
Discount	1	-0.006273707	-0.011591588	-0.00340045	-0.000576748	-0.064887861	0.005281703	0.005851087	0.004238613	-0.004911294	-0.006424278
Unit_Price	-0.006273707	1	-0.039479181	0.649984896	0.399350072	0.208505328	-0.009248684	-0.010134597	-0.015947754	-0.015425719	-0.005722735
Quantity_ordered_new	-0.011591588	-0.039479181	1	-0.03043601	-0.003447643	0.207969625	-0.002555293	-0.001511665	0.041255511	-0.020158532	-0.042787842
Shipping_Cost	-0.00340045	0.649984896	-0.03043601	1	0.29663404	-0.124863538	-0.006314271	-0.005851825	-0.006638293	-0.014284819	-0.009893534
Product_Base_Margin	-0.000576748	0.399350072	-0.003447643	0.29663404	1	-0.131848372	-0.000822717	-0.000370578	0.000445982	0.003993226	0.005169546
Profit	-0.064887861	0.208505328	0.207969625	-0.124863538	-0.131848372	1	-0.011219557	-0.010446186	-0.012317973	-0.007648621	-0.018788029
AverageTransaction	0.005281703	-0.009248684	-0.002555293	-0.006314271	-0.000822717	-0.011219557	1	0.925767913	0.153715444	0.04521585	0.171308516
LastTransactionAmount	0.005851087	-0.010134597	-0.001511665	-0.005851825	-0.000370578	-0.010446186	0.925767913	1	0.137948673	0.03808749	0.160062008
TransactionsPerYear	0.004238613	-0.015947754	0.041255511	-0.006638293	0.000445982	-0.012317973	0.153715444	0.137948673	1	-0.014380358	-0.001589061
Income	-0.004911294	-0.015425719	-0.020158532	-0.014284819	0.003993226	-0.007648621	0.04521585	0.03808749	-0.014380358	1	0.57905103
HomeValue	-0.006424278	-0.005722735	-0.042787842	-0.009893534	0.005169546	-0.018788029	0.171308516	0.160062008	-0.001589061	0.57905103	1

SIGNIFICANT CORRELATION: Shipping_Cost and Unit_Price
AverageTransaction and LastTransactionAmount
Income and HomeValue

Feature Selection

The variables that are more likely not suitable for ML are dropped out.

```
CustTransactionClassification = pd.DataFrame(CustTransaction.drop(columns=
['Customer_ID', 'Order_ID', 'Order_Date', 'Ship_Date', 'Name', 'Address']))
```

Another proposed feature selection is to remove Lat and Lon. Both of these feature sets will be tested through ML modelling to see which feature set will give better result in terms of model accuracy.

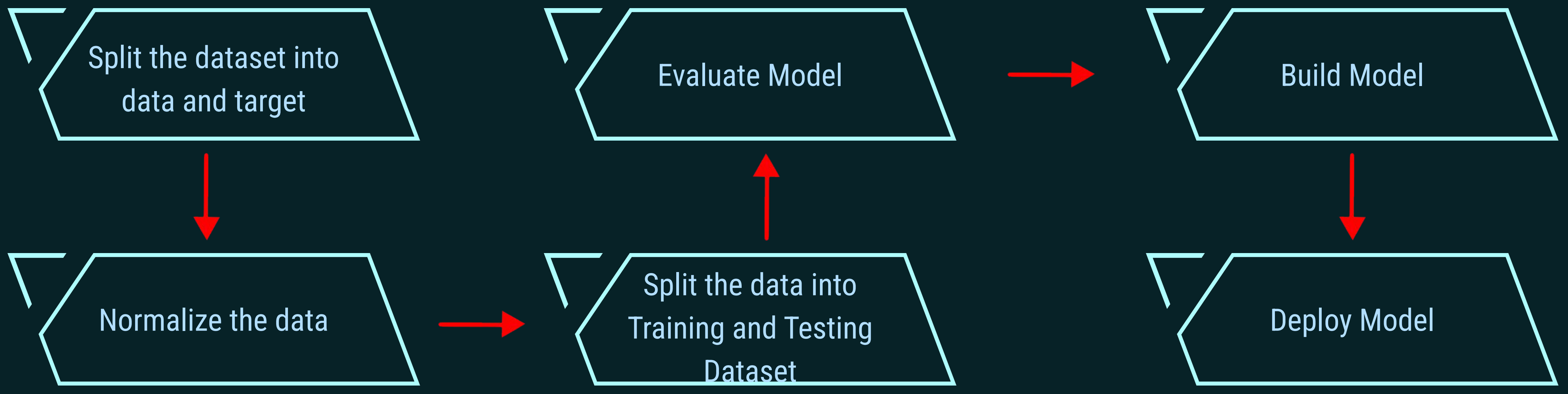
Selected Features

#	Column	Non-Null Count	Dtype
0	Order_Priority	8352 non-null	object
1	Discount	8352 non-null	float64
2	Unit_Price	8352 non-null	float64
3	Quantity_ordered_new	8352 non-null	int64
4	Shipping_Cost	8352 non-null	float64
5	Ship_Mode	8352 non-null	object
6	Product_Category	8352 non-null	object
7	Product_Sub-Category	8352 non-null	object
8	Product_Container	8352 non-null	object
9	Product_Base_Margin	8352 non-null	float64
10	Profit	8352 non-null	float64
11	Store Number	8352 non-null	int64
12	Customer Segment	8352 non-null	object
13	Responder	8352 non-null	object
14	Postcode	8352 non-null	object
15	CustomerType	8352 non-null	object
16	AverageTransaction	8352 non-null	float64
17	LastTransactionAmount	8352 non-null	float64
18	TransactionsPerYear	8352 non-null	float64
19	Income	8352 non-null	int64
20	HomeValue	8352 non-null	int64
21	Lat	8352 non-null	float64
22	Lon	8352 non-null	float64

Modelling

- The ML model focused in this capstone project is **CLASSIFICATION**, a type of prediction modeling used to categorize data into predefined classes or labels.
- This is to cater one of the insights to estimate (predict) the responder (Yes/No) based on the customer segment and another appropriate input
- To achieve accurate classification, three models were selected: K-Nearest Neighbors (KNN), Decision Tree, and Random Forest.

Modelling Steps



Modelling

1. Split the dataset into data and target

```
X = CustTransactionClassification.drop('Responder', axis=1)
Y = CustTransactionClassification['Responder']
```

2. Normalize the data

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Normalize all the data in dataframe X using scaler and put it in dataframe Xscaled
Xscaled = scaler.fit_transform(X)
```

```
Xscaled
```

Modelling

3. Split into Training and Testing dataset

```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(Xscaled, Y, test_size=0.3,
random_state=123)
```

K-Nearest Neighbours

```python
# K-NEAREST NEIGHBOUR MODEL

from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=2)
knn_model.fit(xtrain, ytrain)
knn_prediction = knn_model.predict(xtest)
```

```python
#EVALUATE K-NEAREST NEIGHBOUR MODEL

from sklearn import metrics
knn_CM = metrics.confusion_matrix(ytest, knn_prediction)
print('Confusion Matrix: \n', knn_CM)
print('\n')
knn_Acc = metrics.accuracy_score(ytest, knn_prediction)
print('Model accuracy is: ', knn_Acc)
```

```
Confusion Matrix: 
 [[1814   84]
 [ 431  177]]


Model accuracy is:  0.7944932162809257
```

Decision Tree

```
# DECISION TREE MODEL

from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(criterion='entropy', random_state=123)
dt_model.fit(xtrain, ytrain)
dt_prediction = dt_model.predict(xtest)
```

```
#EVALUATE DECISION TREE MODEL

from sklearn import metrics
dt_CM = metrics.confusion_matrix(ytest, dt_prediction)
print('Confusion Matrix: \n', dt_CM)
print('\n')
dt_Acc = metrics.accuracy_score(ytest, dt_prediction)
print('Model accuracy is: ', dt_Acc)
```

```
Confusion Matrix:
 [[1880   18]
 [  12  596]]


Model accuracy is:  0.9880287310454908
```

Random Forest

```
# RANDOM FOREST MODEL

from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=300)
rf_model.fit(xtrain, ytrain) #TRAIN MODEL
rf_prediction = rf_model.predict(xtest) #TEST MODEL
```

```
#EVALUATE RANDOM FOREST MODEL

from sklearn import metrics
rf_CM = metrics.confusion_matrix(ytest, rf_prediction)
print('Confusion Matrix: \n', rf_CM)
print('\n')
rf_Acc = metrics.accuracy_score(ytest, rf_prediction)
print('Model accuracy is: ', rf_Acc)
```

```
Confusion Matrix:
 [[1893    5]
 [  19  589]]


Model accuracy is:  0.9904229848363927
```

Modelling

Feature Set 1
(All the suitable features)

Classification Model	Accuracy
K-Nearest Neighbors	0.79449
Decision Tree	0.98803
Random Forest	0.99042
SVM (linear kernel)	0.78093
SVM (RBF kernel)	0.79888
SVM (sigmoid kernel)	0.70072
SVM (polynomial kernel)	0.79848
Neural Network Tensorflow	0.79409

Best Model: Random Forest (99%)

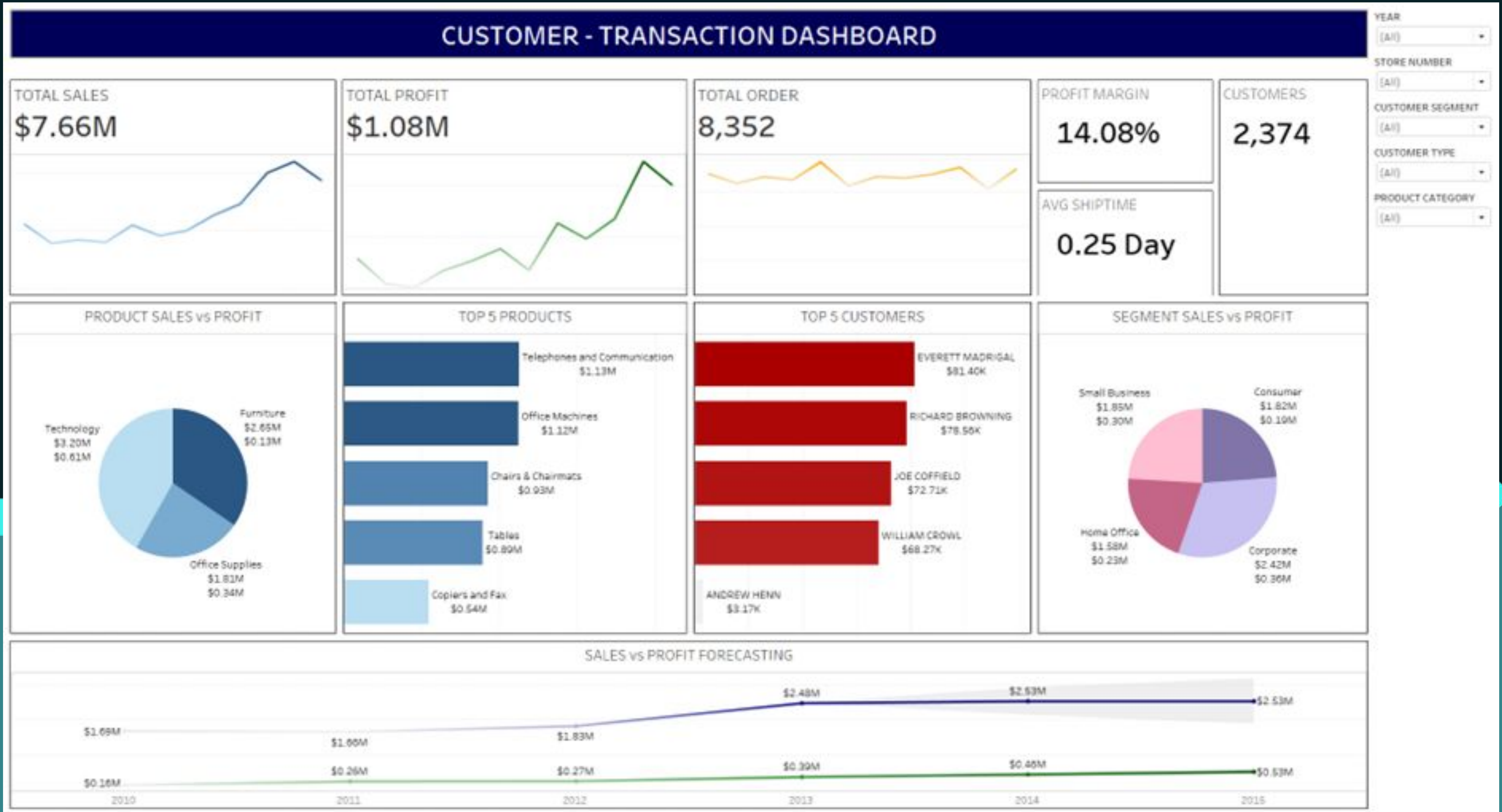
Feature Set 2
(All suitable features except Lat and Lon)

Classification Model	Accuracy
K-Nearest Neighbors	0.78372
Decision Tree	0.9573
Random Forest	0.91141
SVM (linear kernel)	0.78053
SVM (RBF kernel)	0.79489
SVM (sigmoid kernel)	0.69553
SVM (polynomial kernel)	0.7921
Neural Network Tensorflow	0.79409

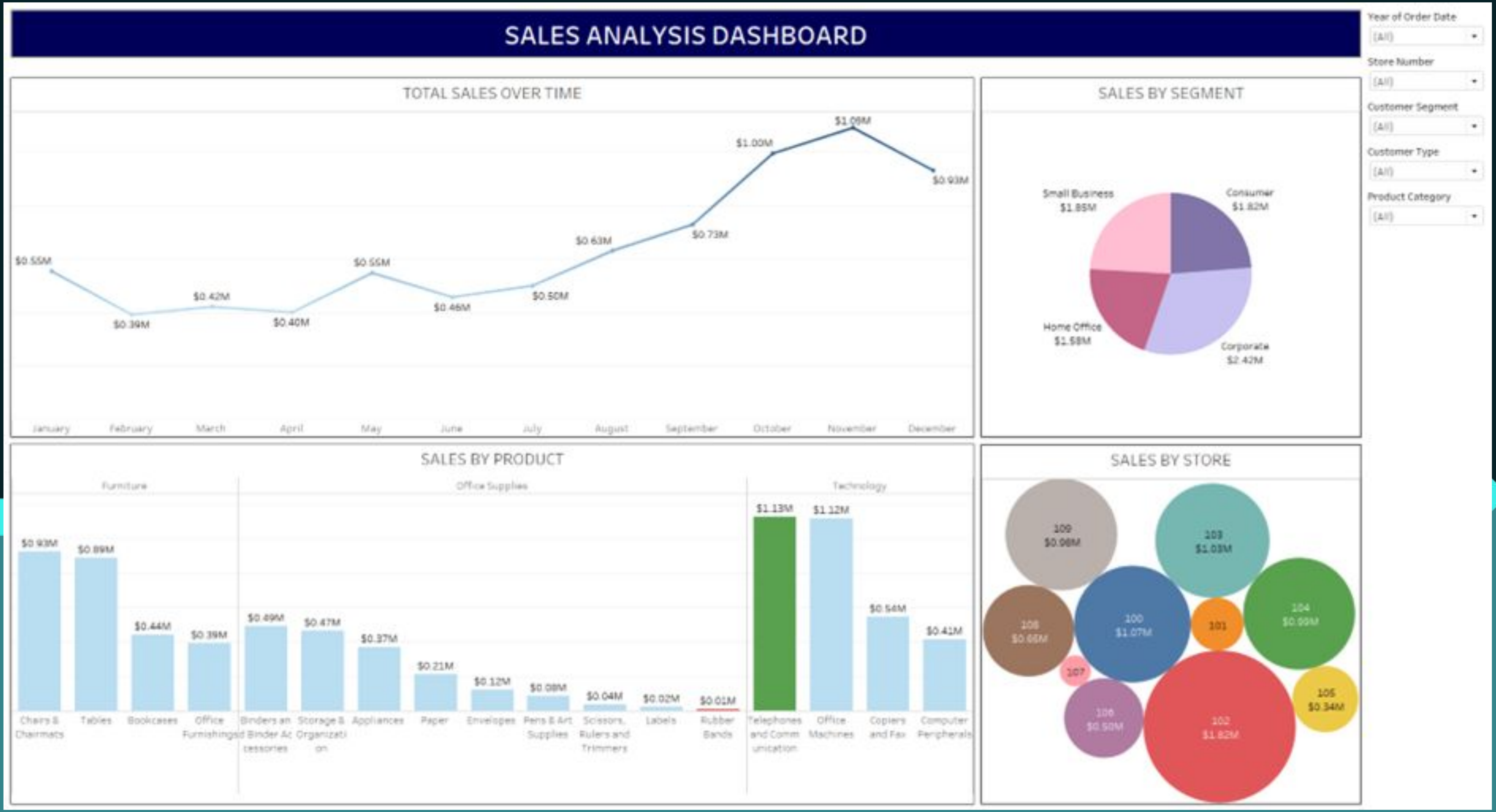
Best Model: Decision Tree (95%)

CONCLUSION: **RANDOM FOREST MODEL with FEATURE SET 1**

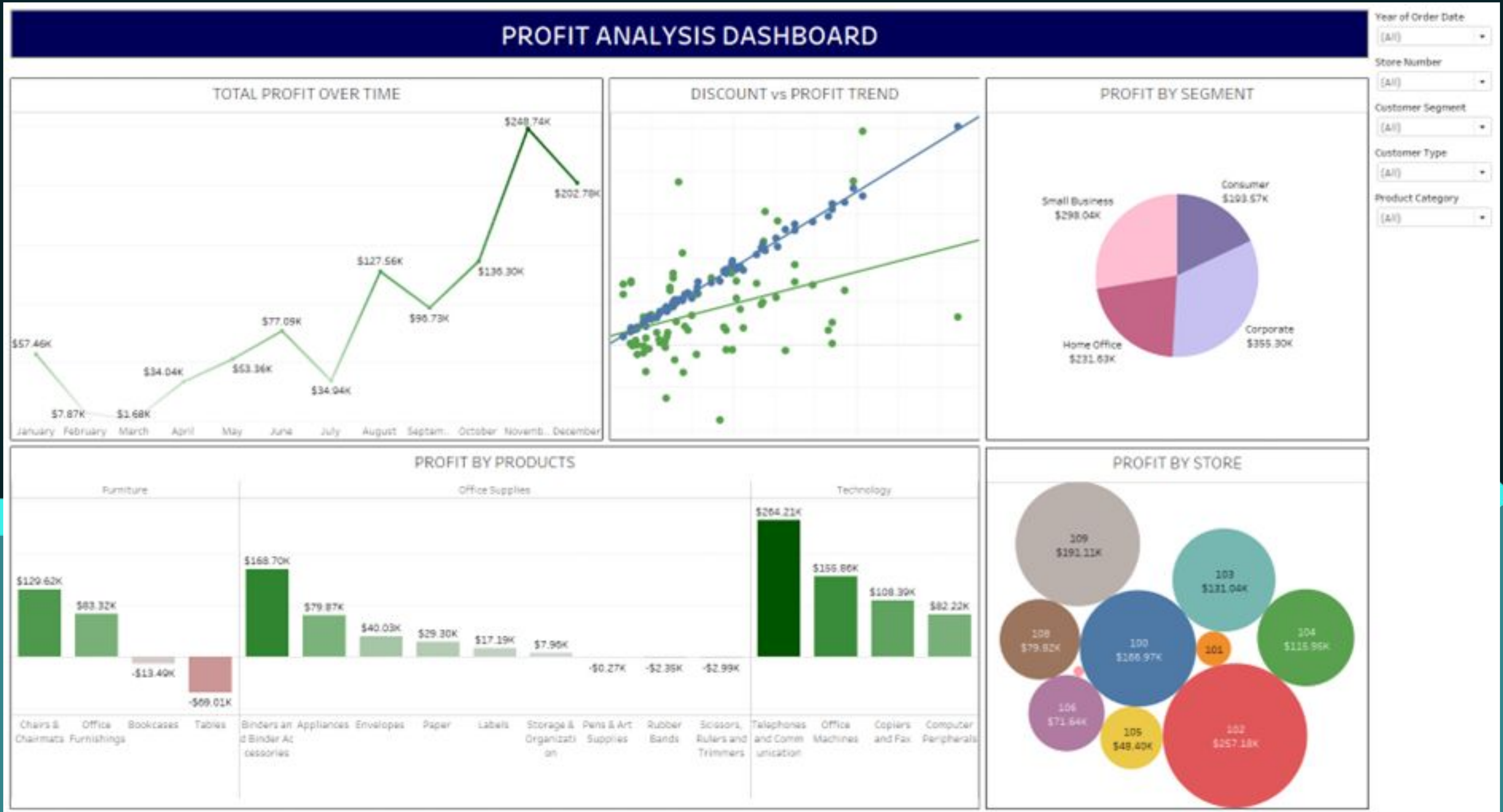
Visualization



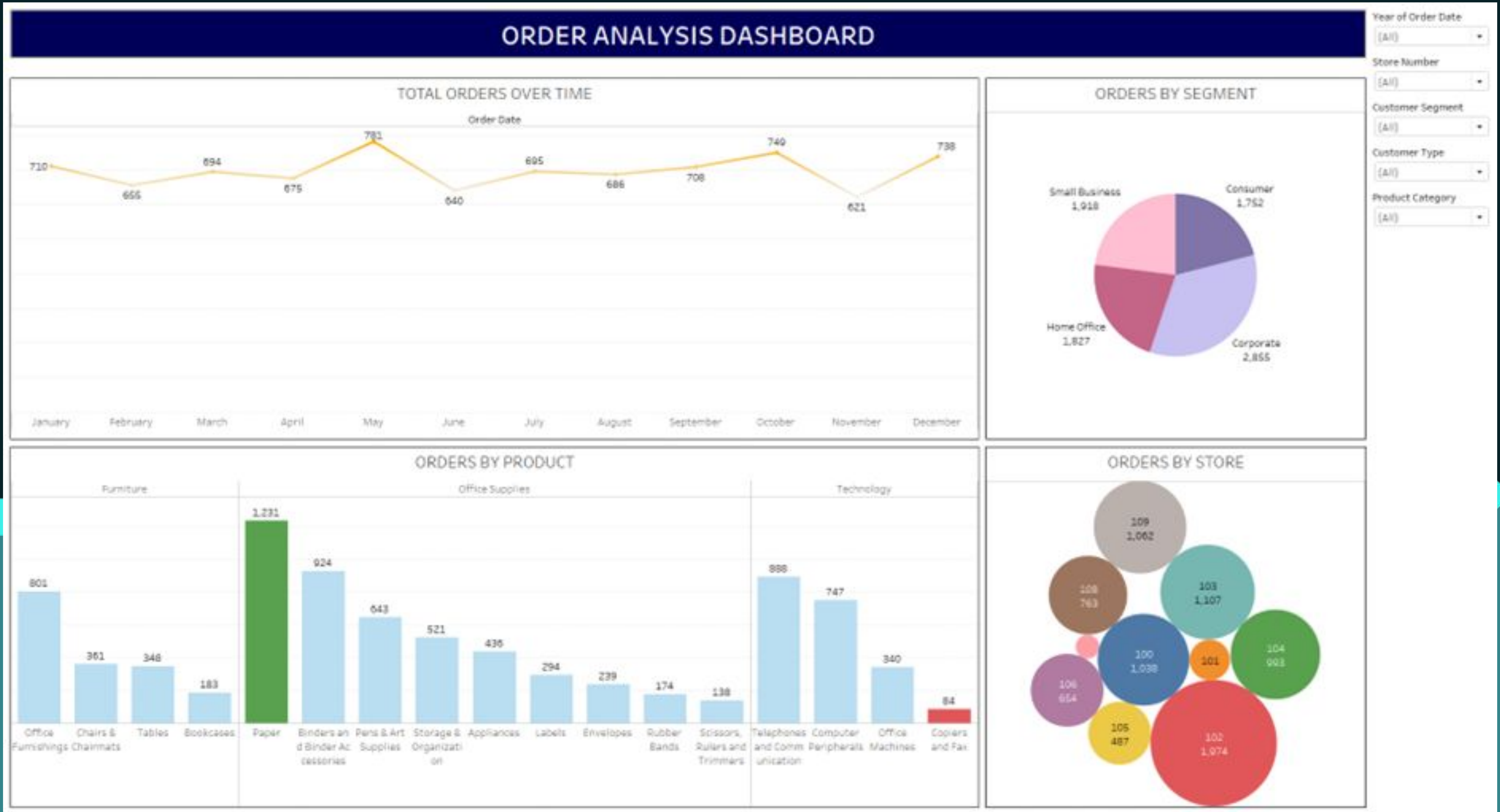
Visualization



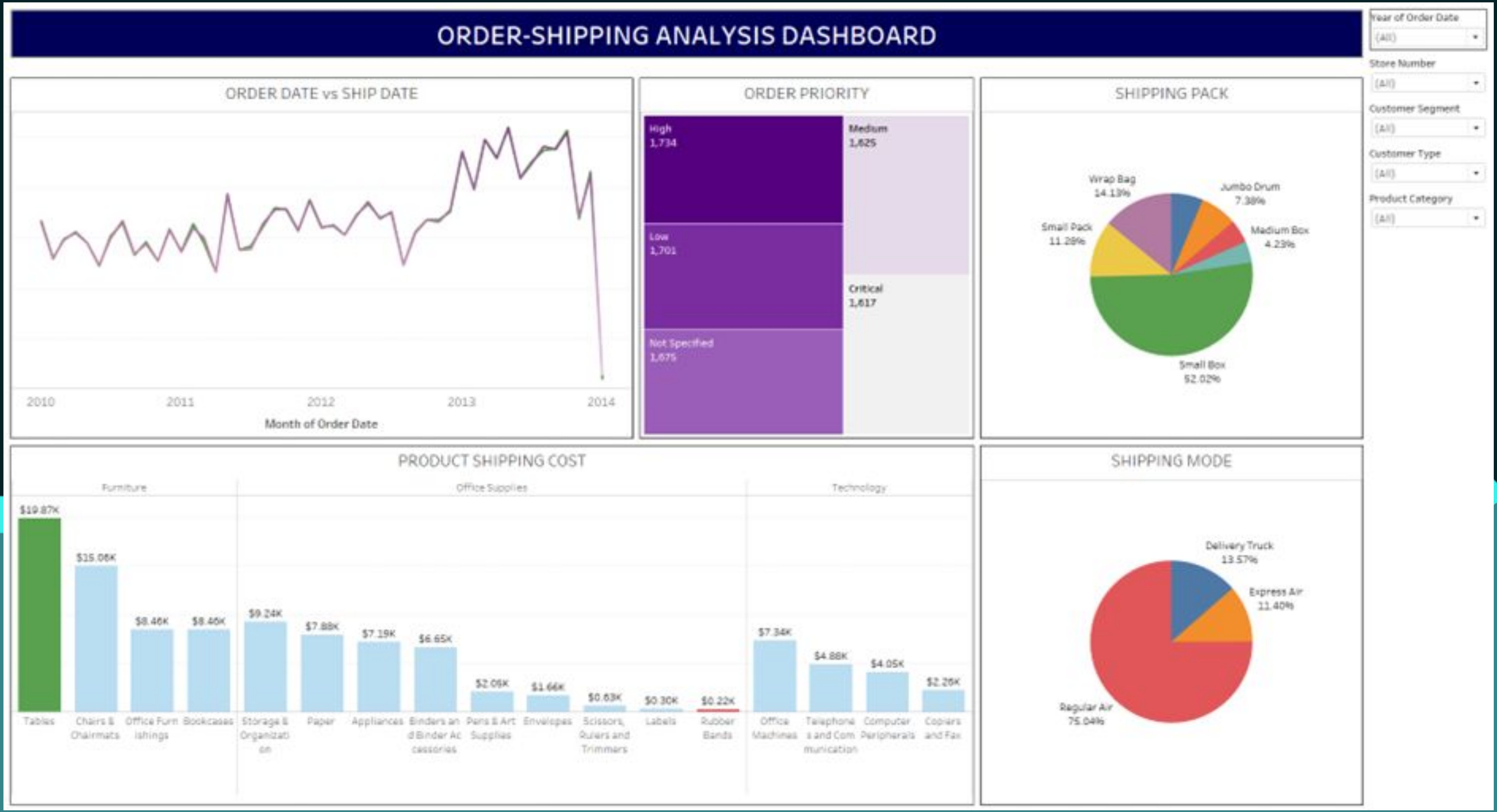
Visualization



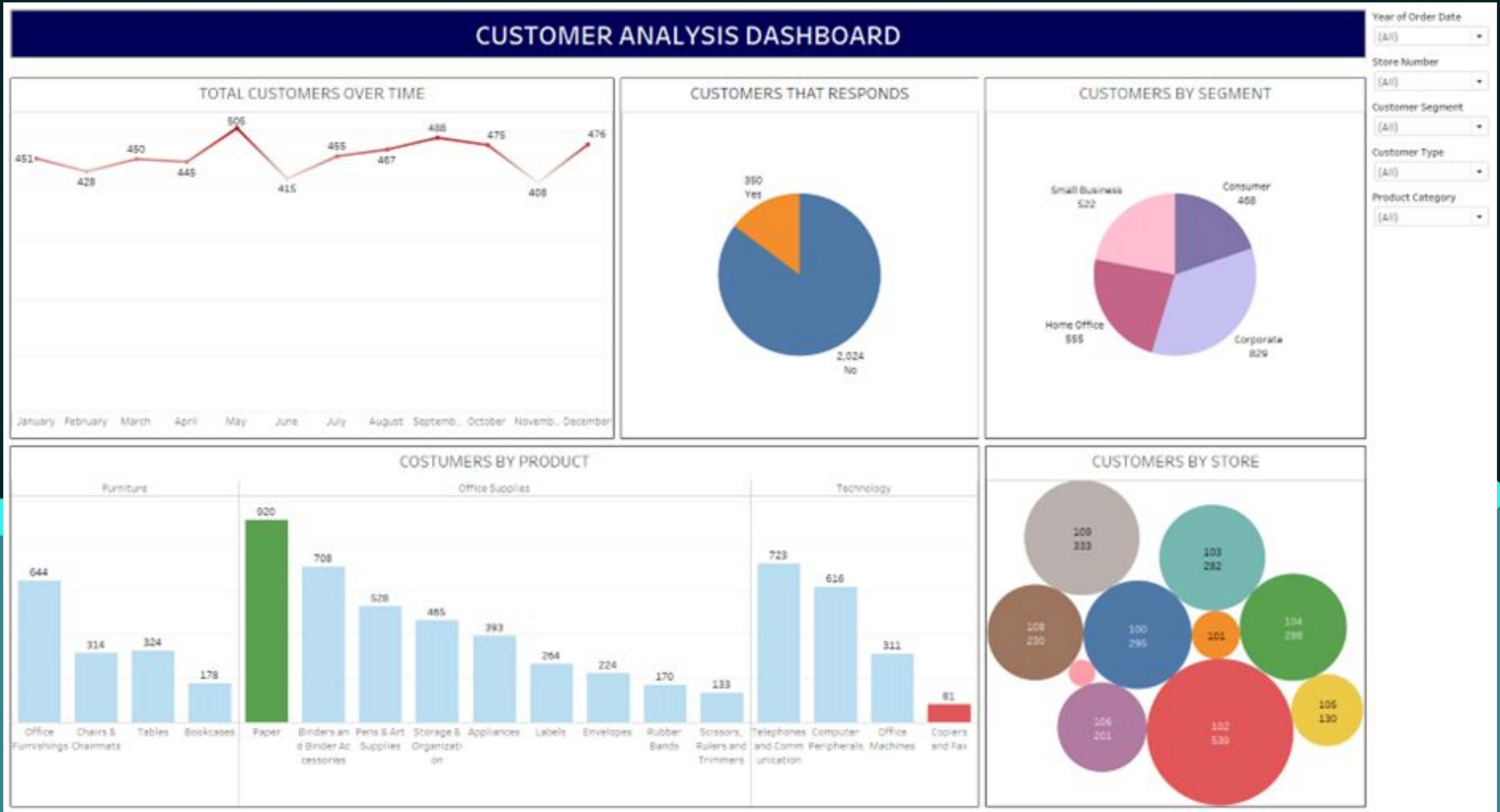
Visualization



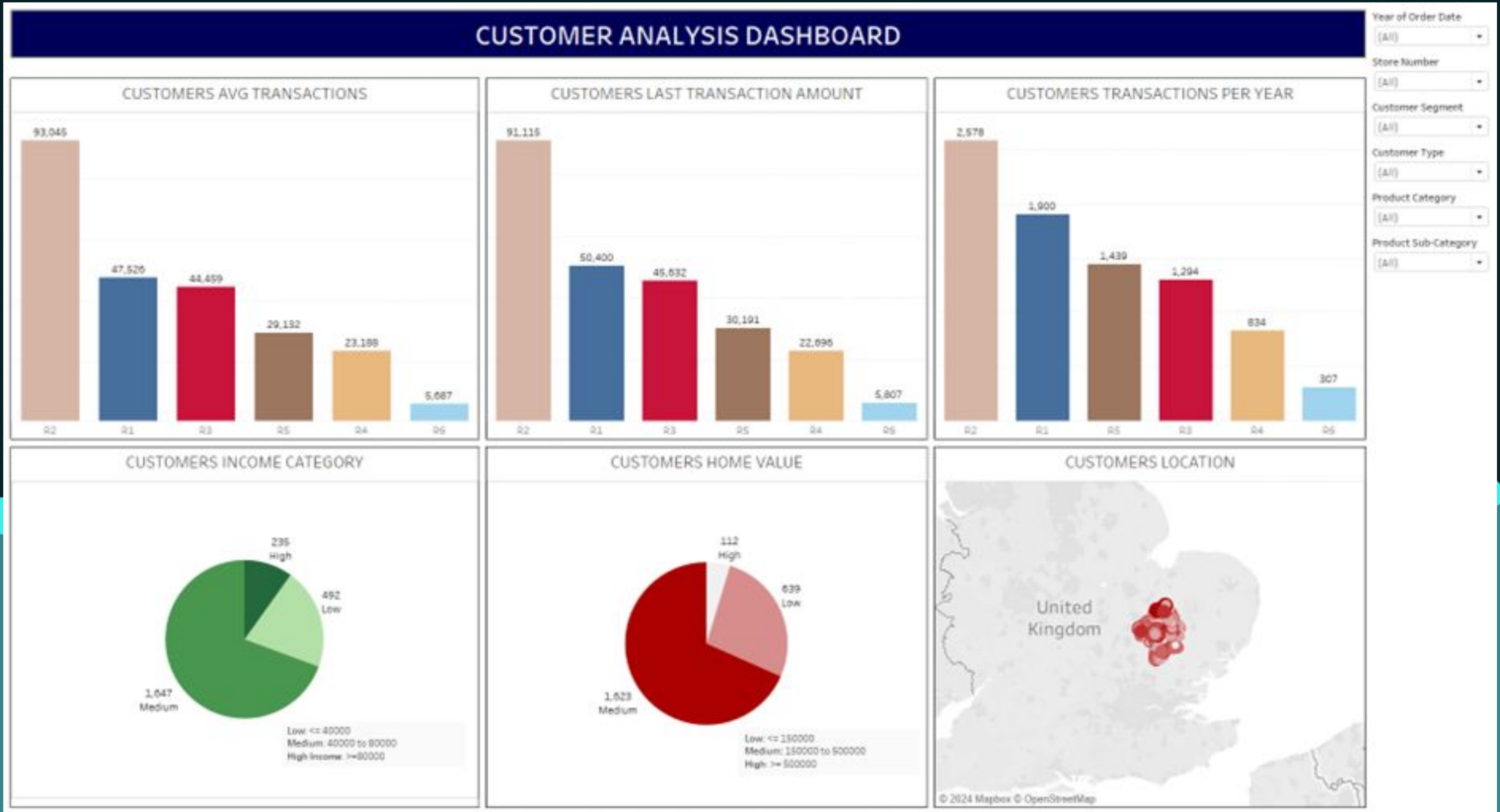
Visualization



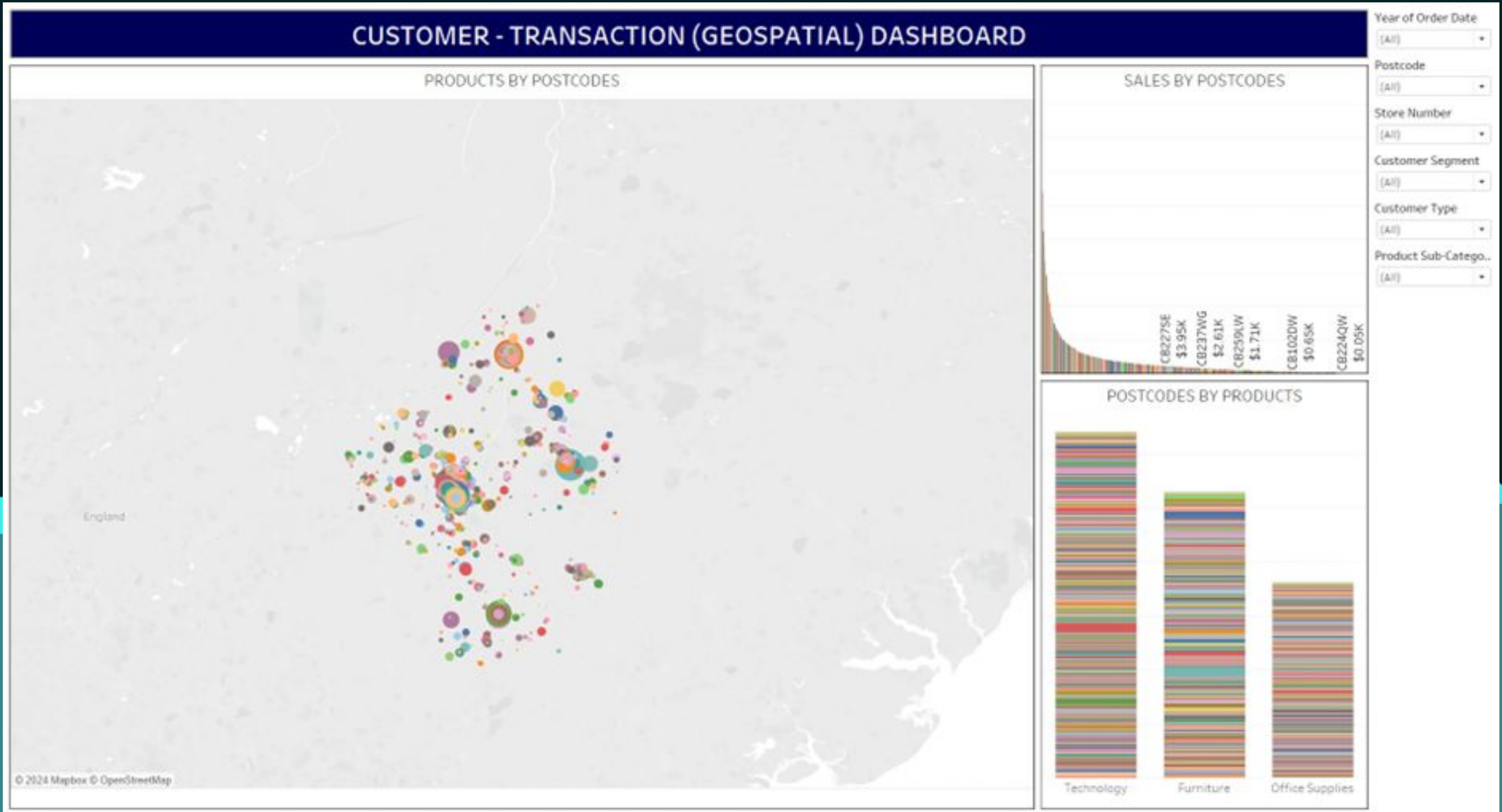
Visualization



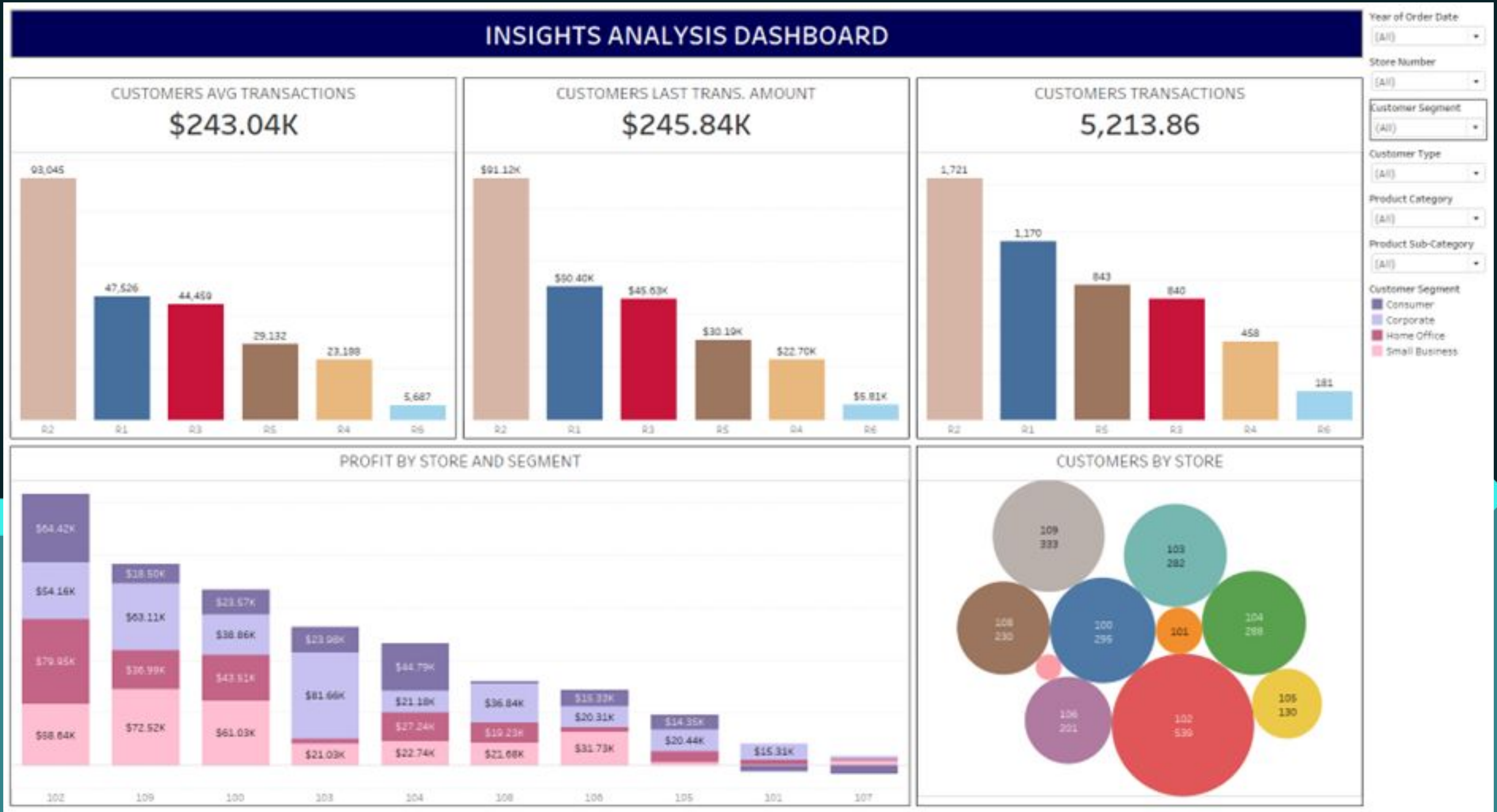
Visualization



Visualization



Visualization



Conclusion

This capstone project successfully leveraged the customer-transactions dataset by cleaning and preprocessing it, and using it to predict classification outcomes to provide valuable insights into patterns and trends that inform strategic decisions.

By developing dashboards in Tableau, the data was transformed into accessible, interactive visualizations that communicate vital findings effectively to stakeholders.

For the future, it is highly recommended to use a real dataset from a real industry and collect it in real-time instead of using dataset samples from the Internet, as well as explore additional visualization features in Tableau to engage users further

THANK YOU