

Project Final Report

Customer Attrition Classification

ISOM3360 - Data Mining for Business Analytics

Table of Contents

Introduction	3
Data Understanding	4
Descriptive Statistics	5
Model Building	7
Data Cleansing	7
Missing Values	7
Z-Score Normalization	10
One-Hot Encoding	11
Classification	14
Decision Tree	14
Logistic Regression	18
Naive Bayes	24
K Nearest Neighbor	26
Clustering	29
K Means Clustering	29
Performance Evaluation	30
Metric Selection	30
Evaluation	32
Cost Sensitive Analysis & Decision Threshold	35
Conclusion	36
References	40

Introduction

Under the COVID-19 pandemic, the amount of people deemed safe to gather in a single place has dwindled from the thousands, to the hundreds, and to the tens. Restaurants, bars, movie theaters, and gyms in many major cities are shutting down. People are adjusting to this period of isolation and uncertainty through drastic changes to their shopping behaviors. As a result, COVID-19 has massively accelerated the growth of ecommerce. According to an article by Forbes written in June 2020, *COVID-19 Accelerated e-Commerce Growth '4 to 6 years'*, it states that total online spending in May hit 82.5 billion - jumping 77% year over year.^[1] COVID-19 has pushed online retail to new heights; meanwhile, online retailers should observe customer behavior and tailor the shopping experience for customers to reduce churn rate. Under the conditions of a pandemic, it becomes ever so important to adapt.

Based on the attached dataset, the key questions below were formulated to address ecommerce business issues and problems:

- 1) Which feature(s) influence the customer churn rate most?
- 2) How to utilize the appropriate promotions to minimize the customers with propensity to churn and turnover?
- 3) Possibility to segment the customers to better cater to their needs?

These questions serve to guide ecommerce businesses. The overarching goal is to reduce the customer churn rate, which inevitably will increase revenue for ecommerce businesses. Businesses will find customer information very useful when it comes to expanding their product range, offering incentives for customer retention, and maintaining customer-to-business relationships. However, it would be very difficult to answer these questions through traditional programming or simple heuristics. Businesses will have to survey customers, look at descriptive statistics and generalize, or attempt to manually formulate rules to distinguish between churn and non-churn customers. Fortunately, the sample dataset has only 20 features and 5000 samples, which relatively isn't impossible to answer the questions through traditional programming or simple heuristics, but in the larger context, ecommerce businesses will have a lot more features and samples to consider. Take Amazon for example, a global ecommerce retailer has around 200 million Amazon Prime (the premium version of Amazon) users worldwide in 2020.^[2] The amount of customer data that Amazon has overwhelms the outdated techniques of traditional programming and simple heuristics. Hence, machine learning is utilized to solve these types of questions.

Data Understanding

Dataset is retrieved from Kaggle, a website that has public datasets and notebooks open sourced for users to do data science work. The dataset taken under consideration is:

Kaggle - Ecommerce Customer Churn Analysis and Prediction^[3]

The dataset contains 20 features and 5630 samples, which notably includes CustomerID (unique identifier of customers) and Churn (label that the machine learning model will predict). The data dictionary below generally describes the dataset:

<i>Feature</i>	<i>Description</i>	<i>Attribute Type</i>
CustomerID	Unique customer ID	string
Churn	(Label) Churn Flag. Churn Customers = 1 / Non-Churn Customers = 0	integer
Tenure	Tenure of customers in organization. Length of time the user has been a customer	float
PreferredLoginDevice	Preferred login device of customer	string
CityTier	City tier	integer
WarehouseToHome	Distance in between warehouse to home of customer	float
PreferredPaymentMode	Preferred payment method of customer	string
Gender	Gender of customer	string
HourSpendOnApp	Number of hours spend on mobile application or website	float
NumberOfDeviceRegistered	Total number of deceives is registered on particular customer	integer
PreferedOrderCat	Preferred order category of customer in last month	string
SatisfactionScore	Satisfactory score of customer on service	integer
MaritalStatus	Marital status of customer	float
NumberOfAddress	Total number of added added on particular	integer

	customer	
Complain	Any complaint has been raised in last month	integer
OrderAmountHikeFromlast Year	Percentage increases in order from last year	float
CouponUsed	Total number of coupon has been used in last month	float
OrderCount	Total number of orders has been places in last month	float
DaySinceLastOrder	Day Since last order by customer	float
CashbackAmount	Average cashback in last month	float

Descriptive Statistics

Overview of the dataset features (sample size, attribute, mean, range) - excluding CustomerID:

```

Churn                5630 non-null int64
Tenure               5366 non-null float64
PreferredLoginDevice 5630 non-null object
CityTier             5630 non-null int64
WarehouseToHome      5379 non-null float64
PreferredPaymentMode  5630 non-null object
Gender               5630 non-null object
HourSpendOnApp        5375 non-null float64
NumberOfDeviceRegistered 5630 non-null int64
PreferedOrderCat      5630 non-null object
SatisfactionScore     5630 non-null int64
MaritalStatus         5630 non-null object
NumberOfAddress       5630 non-null int64
Complain             5630 non-null int64
OrderAmountHikeFromlastYear 5365 non-null float64
CouponUsed           5374 non-null float64
OrderCount           5372 non-null float64
DaySinceLastOrder     5323 non-null float64
CashbackAmount        5630 non-null float64
dtypes: float64(8), int64(6), object(5)

```

dataset.info()

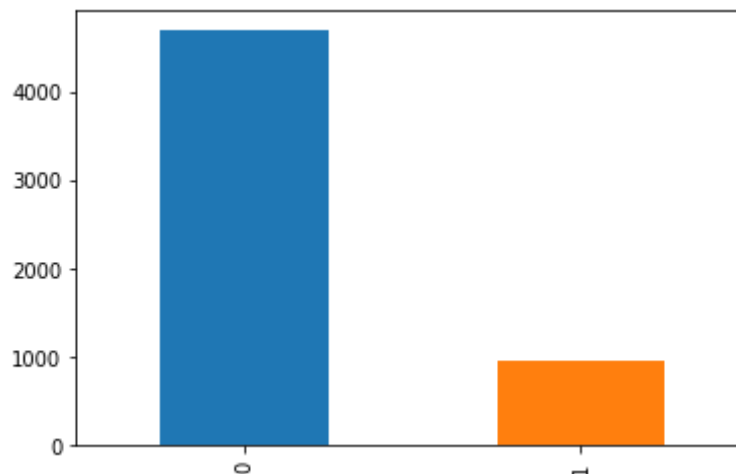
	Churn	Tenure	CityTier	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	SatisfactionScore	NumberOfAddress
count	5630.000000	5366.000000	5630.000000	5379.000000	5375.000000	5630.000000	5630.000000	5630.000000
mean	0.168384	10.189899	1.654707	15.639896	2.931535	3.688988	3.066785	4.214032
std	0.374240	8.557241	0.915389	8.531475	0.721926	1.023999	1.380194	2.583586
min	0.000000	0.000000	1.000000	5.000000	0.000000	1.000000	1.000000	1.000000
25%	0.000000	2.000000	1.000000	9.000000	2.000000	3.000000	2.000000	2.000000
50%	0.000000	9.000000	1.000000	14.000000	3.000000	4.000000	3.000000	3.000000
75%	0.000000	16.000000	3.000000	20.000000	3.000000	4.000000	4.000000	6.000000
max	1.000000	61.000000	3.000000	127.000000	5.000000	6.000000	5.000000	22.000000

Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
5630.000000	5365.000000	5374.000000	5372.000000	5323.000000	5630.000000
0.284902	15.707922	1.751023	3.008004	4.543491	177.223030
0.451408	3.675485	1.894621	2.939680	3.654433	49.207036
0.000000	11.000000	0.000000	1.000000	0.000000	0.000000
0.000000	13.000000	1.000000	1.000000	2.000000	145.770000
0.000000	15.000000	1.000000	2.000000	3.000000	163.280000
1.000000	18.000000	2.000000	3.000000	7.000000	196.392500
1.000000	26.000000	16.000000	16.000000	46.000000	324.990000

dataset.describe()

There's a total of **seven variables that have missing values** - HourSpendOnApp (5375) / WarehouseToHome (5379) / Tenure (5366) / OrderAmountHikeFromlastYear (5365) / DaySinceLastOrder (5323) / CouponUsed (5374) / OrderCount (5372).

Overview of the “Churn” prediction label. Non-Churn (labeled: 0) and Churn (labeled: 1) visualization:



ax=dataset['Churn'].value_counts().plot(kind='bar')

Clear **class imbalance** between the Label, with Non-Churn having **4.5x time** more entries than Churn customers.

Model Building

It is apparent that the raw dataset needs to be cleansed before any machine learning techniques can be employed. Data cleansing process includes filling in missing values, z-score normalization, and one-hot encoding.

Data Cleansing

The missing values of most features are handled by filling in the grouped mean for each group's missing values.

Missing Values

HourSpendOnApp: the values are filled by grouping the PreferredLoginDevice and finding the grouped mean and filling it in.

HourSpendOnApp	
PreferredLoginDevice	
Computer	2.908916
Mobile Phone	3.034720
Phone	2.693625

```
dataset.groupby(['PreferredLoginDevice',]).mean()
```

```
dataset['HourSpendOnApp'].fillna(dataset.groupby(['PreferredLoginDevice'])['HourSpendOnApp'].transform('mean'), inplace=True)
```

Though the differences seem marginal, it seems logical to handle the missing values with the grouped mean because users. In ecommerce, users tend to order and browse from their mobile devices. Also they seem to make more purchases when they sit down and use their computers.^[4] These behaviors warrant the handling of the missing values for HourSpendOnApp by using the PreferredLoginDevice.

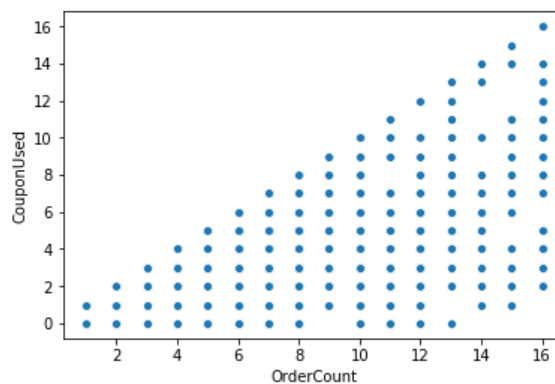
The same techniques for filling in missing values are also done with four other features:

Features	Grouped By	Reasoning
WarehouseToHome	CityTier	Distance to a warehouse is often affected by the location the customer

		is in. CityTier distinguishes which city customers are located
Tenure	Gender	There seems to be some correlation between Tenure and Gender, with females having a higher tenure than males. However, this could be a confounder and a spurious association caused by socioeconomic factors.
OrderAmountHikeFromlastYear	SatisfactionScore	Customers with a higher satisfaction have a tendency to order and purchase more, which leads to an increased order amount.
DaySinceLastOrder	SatisfactionScore	Customers with a higher satisfaction have a tendency to order and purchase more, which leads to less gaps between orders.

The missing values of the last two features filled through linear regression. CouponUsed and OrderCount have a positive linear relationship. As the OrderCount increases, CouponUsed will also increase, vice versa.

CouponUsed as dependent variable (y) and OrderCount as independent variable (x):



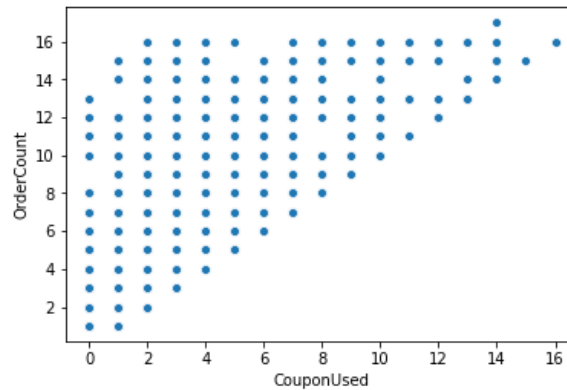
model = LinearRegression() was used to fit the x and y variables, which calculates the linear equation:

$$y = 0.49105096x + 0.28518233347145006$$

Then simply input the OrderCount as x and fill the missing count of CouponUsed with the output y.

```
full['CouponUsed'].fillna(round(full.OrderCount * 0.49105096 + 0.28518233347145006,0), inplace = True)
```

OrderCount as dependent variable (y) and CouponUsed as independent variable (x):



model = LinearRegression() was used to fit the x and y variables, which calculates the the linear equation:

$$y = 1.13102243x + 0.9738270014063504$$

Then simply input the CouponUsed as x and fill the missing count of OrderCount with the output y.

Checking dataset information after handling the missing values of seven features:

Churn	5630 non-null int64
Tenure	5630 non-null float64
PreferredLoginDevice	5630 non-null object
CityTier	5630 non-null int64
WarehouseToHome	5630 non-null float64
PreferredPaymentMode	5630 non-null object
Gender	5630 non-null object
HourSpendOnApp	5630 non-null float64
NumberOfDeviceRegistered	5630 non-null int64
PreferedOrderCat	5630 non-null object
SatisfactionScore	5630 non-null int64
MaritalStatus	5630 non-null object
NumberOfAddress	5630 non-null int64
Complain	5630 non-null int64
OrderAmountHikeFromlastYear	5630 non-null float64
CouponUsed	5630 non-null float64
OrderCount	5630 non-null float64
DaySinceLastOrder	5630 non-null float64
CashbackAmount	5630 non-null float64

dataset.info()

Z-Score Normalization

Z-Score normalization was chosen to normalize the numerical features from the dataset because it normalizes the outliers and makes the dataset more consistent.^[5] There are a total of six features to be normalized - Tenure / HourSpendOnApp / WarehouseToHome / DaySinceLastOrder / OrderAmountHikeFromlastYear / CashbackAmount.

```

# Tenure
|
zscore_scaler = preprocessing.StandardScaler().fit(full_final[['Tenure']])
full_final['Tenure_zscore'] = zscore_scaler.transform(full_final[['Tenure']])
full_final = full_final.drop(['Tenure'], axis='columns')

# HourSpendOnApp

zscore_scaler = preprocessing.StandardScaler().fit(full_final[['HourSpendOnApp']])
full_final['HourSpendOnApp_zscore'] = zscore_scaler.transform(full_final[['HourSpendOnApp']])
full_final = full_final.drop(['HourSpendOnApp'], axis='columns')

# WarehouseToHome

zscore_scaler = preprocessing.StandardScaler().fit(full_final[['WarehouseToHome']])
full_final['WarehouseToHome_zscore'] = zscore_scaler.transform(full_final[['WarehouseToHome']])
full_final = full_final.drop(['WarehouseToHome'], axis='columns')

# DaySinceLastOrder

zscore_scaler = preprocessing.StandardScaler().fit(full_final[['DaySinceLastOrder']])
full_final['DaySinceLastOrder_zscore'] = zscore_scaler.transform(full_final[['DaySinceLastOrder']])
full_final = full_final.drop(['DaySinceLastOrder'], axis='columns')

# OrderAmountHikeFromLastYear

zscore_scaler = preprocessing.StandardScaler().fit(full_final[['OrderAmountHikeFromLastYear']])
full_final['OrderAmountHikeFromLastYear_zscore'] = zscore_scaler.transform(full_final[['OrderAmountHikeFromLastYear']])
full_final = full_final.drop(['OrderAmountHikeFromLastYear'], axis='columns')

# CashbackAmount

zscore_scaler = preprocessing.StandardScaler().fit(full_final[['CashbackAmount']])
full_final['CashbackAmount_zscore'] = zscore_scaler.transform(full_final[['CashbackAmount']])
full_final = full_final.drop(['CashbackAmount'], axis='columns')

```

One-Hot Encoding

One-Hot Encoding is used to turn the categorical features into numerical ones represented by numbers. Since the categorical features are all nominal, the order of the values doesn't matter, so they were given arbitrary values. There are a total of five features to be encoded - PreferredLoginDevice / PreferredPaymentMode / Gender / PreferredOrderCat / MaritalStatus.

```

# PreferredLoginDevice (baseline - Phone)

temp_dummy = pd.get_dummies(full_final['PreferredLoginDevice'])
full_final = pd.concat([full_final,temp_dummy],axis=1,sort=True)
full_final = full_final.drop(['PreferredLoginDevice','Phone'],axis='columns')

# PreferredPaymentMode (baseline - UPI)

temp_dummy = pd.get_dummies(full_final['PreferredPaymentMode'])
full_final = pd.concat([full_final,temp_dummy],axis=1,sort=True)
full_final = full_final.drop(['PreferredPaymentMode','UPI'],axis='columns')

# Gender (baseline - female)

temp_dummy = pd.get_dummies(full_final['Gender'])
full_final = pd.concat([full_final,temp_dummy],axis=1,sort=True)
full_final = full_final.drop(['Gender','Female'],axis='columns')

# PreferredOrderCat (baseline - others)

temp_dummy = pd.get_dummies(full_final['PreferredOrderCat'])
full_final = pd.concat([full_final,temp_dummy],axis=1,sort=True)
full_final = full_final.drop(['PreferredOrderCat','Others'],axis='columns')

# MaritalStatus (baseline - single)

temp_dummy = pd.get_dummies(full_final['MaritalStatus'])
full_final = pd.concat([full_final,temp_dummy],axis=1,sort=True)
full_final = full_final.drop(['MaritalStatus','Single'],axis='columns')

```

Final check for dataset information after handling the z-score normalization and one-hot encoding. There are now a total of 30 features after one-hot encoding and 5630 complete samples:

Churn	5630 non-null int64
CityTier	5630 non-null int64
NumberOfDeviceRegistered	5630 non-null int64
SatisfactionScore	5630 non-null int64
NumberOfAddress	5630 non-null int64
Complain	5630 non-null int64
CouponUsed	5630 non-null float64
OrderCount	5630 non-null float64
Tenure_zscore	5630 non-null float64
HourSpendOnApp_zscore	5630 non-null float64
WarehouseToHome_zscore	5630 non-null float64
DaySinceLastOrder_zscore	5630 non-null float64
OrderAmountHikeFromlastYear_zscore	5630 non-null float64
CashbackAmount_zscore	5630 non-null float64
Computer	5630 non-null uint8
Mobile Phone	5630 non-null uint8
CC	5630 non-null uint8
COD	5630 non-null uint8
Cash on Delivery	5630 non-null uint8
Credit Card	5630 non-null uint8
Debit Card	5630 non-null uint8
E wallet	5630 non-null uint8
Male	5630 non-null uint8
Fashion	5630 non-null uint8
Grocery	5630 non-null uint8
Laptop & Accessory	5630 non-null uint8
Mobile	5630 non-null uint8
Mobile Phone	5630 non-null uint8
Divorced	5630 non-null uint8
Married	5630 non-null uint8

dataset.info()

Since the machine learning problem is a classification problem, predicting the class of customers (either Churn or Non-Churn), we will be using three machine learning models - [Decision Tree](#), [Logistic Regression](#), [Naive Bayes](#), and [K Nearest Neighbor](#). The [K Means Cluster](#), would be used for customer segmentation and clustering tasks. Also for the models, the dataset will be split with `random_state = 42` with the `test_size = 0.3` and `train_size = 0.7`.

```
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size=0.3, random_state=42)
```

The performance of the model will be optimized based on increasing the precision and recall percentage for Churn (labelled: 1) classification, while metrics such as accuracy is ignored due to having a skewed dataset. See [Metric Selection](#) section for more information. Each of the intermediary revised models will be evaluated against the Precision-Recall curve and ROC / AUC curve. If the revised model and the baseline model are marginally similar, the revised model will be picked because it would be less overfitted than the baseline.

Classification

Decision Tree

For the decision tree model, first we establish a baseline / benchmark, which will be compared to improved models.

Baseline Model Specification:

```
model = tree.DecisionTreeClassifier()
```

Baseline Classification Report:

```
Classification Report:
              precision    recall  f1-score   support

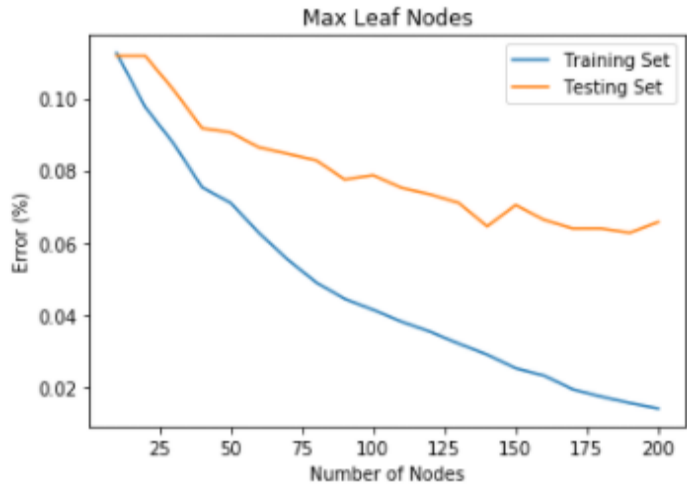
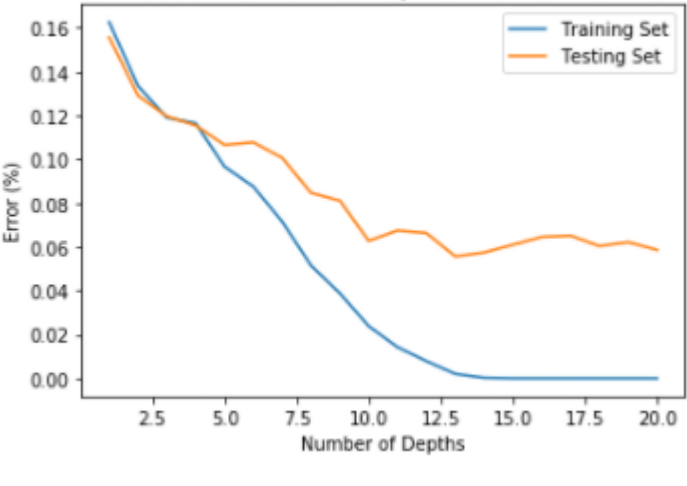
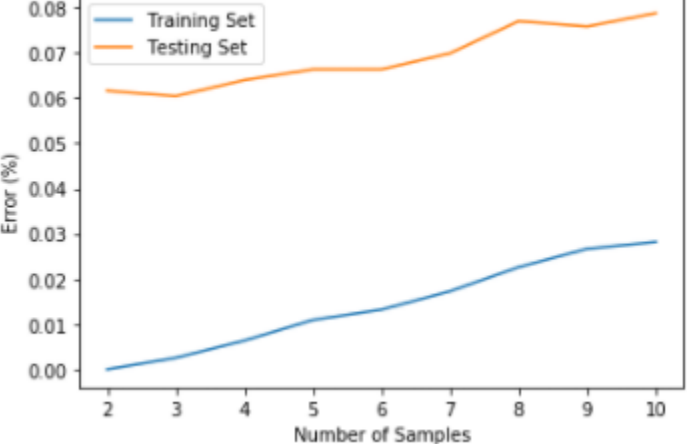
     0       0.96      0.96      0.96      1414
     1       0.81      0.80      0.81       275

 avg / total       0.94      0.94      0.94      1689

Confusion Matrix:
[[1361   53]
 [  54  221]]
```

Before running the parameters in the cross validation, we will run an overfitting check of the error rate between the train and test set based on the parameters: 'max_depth', 'max_leaf_nodes', and 'min_samples_split', which will determine the range of parameters to use based on the minimizing error rate.

<i>Parameter</i>	<i>Error Rate</i>	<i>Range</i>
------------------	-------------------	--------------

max_depth	 <p>Max Leaf Nodes</p> <p>Training Set</p> <p>Testing Set</p> <p>Error (%)</p> <p>Number of Nodes</p>	150 - 170
max_leaf_nodes	 <p>Max Depths</p> <p>Training Set</p> <p>Testing Set</p> <p>Error (%)</p> <p>Number of Depths</p>	13 - 18
min_samples_split	 <p>Min Sample Split</p> <p>Training Set</p> <p>Testing Set</p> <p>Error (%)</p> <p>Number of Samples</p>	2 - 4

Then parameters for cross validation will be initialized with the determined range.

```

# define criterion
crit = ['gini','entropy']

# define max depths
depths = np.arange(13, 19)

# define max leaf nodes
num_leafs = [150,155,160,165,170]

# define the min sample split
split = [2,3,4]

# initiate grid
try_grid = [{'criterion':crit,
              'max_depth':depths,
              'max_leaf_nodes':num_leafs,
              'min_samples_split':split,
              }]

try_grid

```

initial values before cross validation

These values are determined by running a cross validation of $k = 10$ folds.

<i>Parameter</i>	<i>Values</i>	<i>Reason</i>
criterion	'gini'	Use the gini index to split the tree nodes.
max_depth	17	Limit the depth of the tree by pruning.
max_leaf_nodes	170	Limit the leaf nodes of the tree by pruning.
min_samples_split	2	Limit the minimum samples required for a leaf node.

Revised Model Classification Report:


```

Classification Report:
              precision    recall  f1-score   support

     0       0.95         0.98         0.96       1414
     1       0.88         0.72         0.79        275

 avg / total       0.94         0.94         0.94       1689

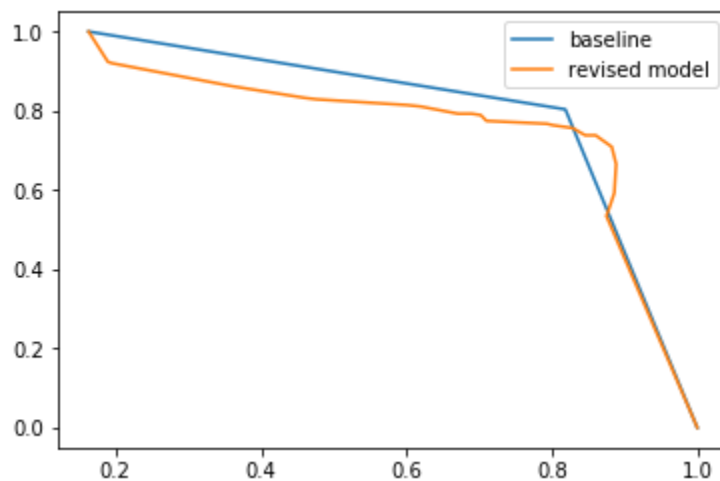
Confusion Matrix:
[[1386   28]
 [  77  198]]

```

non-churn (0): marginal gain for precision and recall
churn (1): precision increased by 7% and recall decreased by 8%

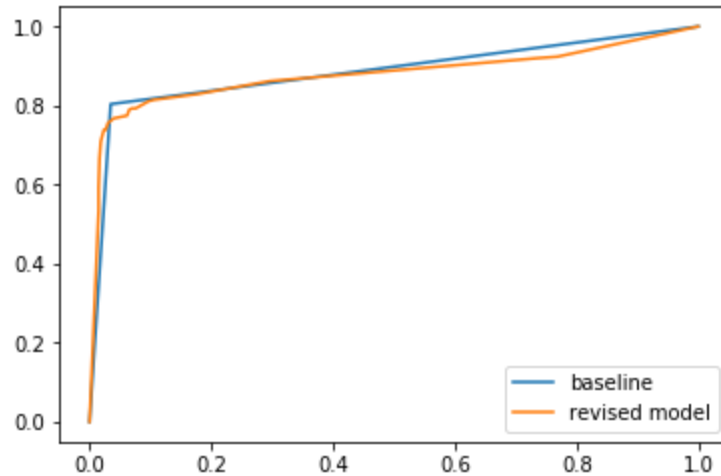
Revised Model Compared to Baseline:

Precision-Recall Curve:



The baseline model has a slightly better performing precision-recall curve.

ROC / AUC Curve:



<i>AUC - Baseline</i>	<i>AUC - Revised Model</i>
~88.449%	~87.818%

The baseline model has a better performing ROC / AUC Curve.

Based on the results from Precision-Recall Curve and ROC / AUC Curve, the baseline model performed marginally better than the revised model; however due to the baseline having overfitted parameters, the revised model would be the better choice. Overall, the **revised model** with hypertuned parameters from cross validation performs better than the baseline model.

Logistic Regression

For the logistic regression model, first we establish a baseline / benchmark, which will be compared to improved models generated from L1 and L2 regressions.

Baseline Model Specification:

```
lr = LogisticRegression()
```

Baseline Classification Report:

```

Classification Report:
              precision    recall  f1-score   support

     0       0.91         0.98         0.95        1414
     1       0.83         0.52         0.64         275

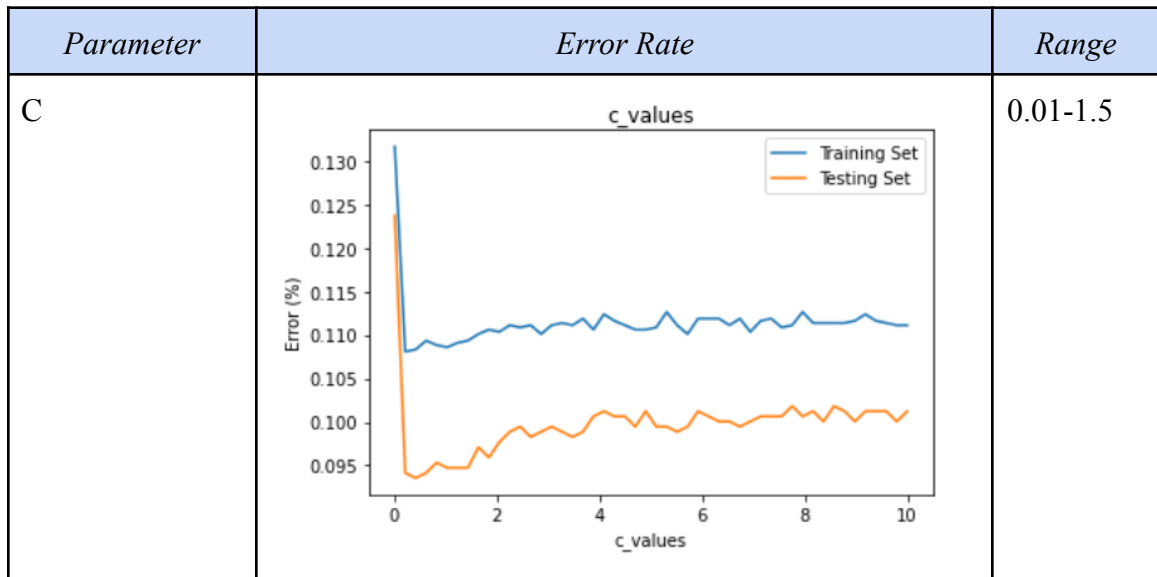
 accuracy          0.91        1689
 macro avg       0.87         0.75         0.79        1689
 weighted avg    0.90         0.91         0.90        1689

Confusion Matrix:
[[1385   29]
 [ 131  144]]

```

The parameters that the logistic regression model will be initialized with are ‘solver’, ‘penalty’ and ‘C’. Since different solvers are used for L1 and L2 regression^[6], hyperparameter tunings are carried out twice to find the best models that use L1 and L2 regression as the penalty term respectively.

Before running the parameters in the cross validation, we will run an overfitting check of the error rate between the train and test set based on the parameter: ‘C’, which will determine the range of ‘C’ values to use based on the minimizing error rate.



```

# define different algorithms to use in the optimization problem of L1 regression
solvers1 = ['saga', 'liblinear']

# define the norm used in the penalization as L1 regression
penalty1 = ['l1']

# define an array of C values (inverse of regularization strength)
c_values1 = np.linspace(0.01,1.5,50)

# initiate grid
grid1 = dict(solver=solvers1, penalty=penalty1, C=c_values1)
cv1 = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

```

initial values before cross validation of L1 regression

<i>Parameter</i>	<i>Values</i>	<i>Reason</i>
solver	'liblinear'	Use 'liblinear' algorithm, which can support L1 penalty, in the optimization problem
penalty	'l1'	Specify the norm used in the penalization
C	0.7093877551020409	Use a smaller C value to specify stronger regularization

Revised Model (using L1 regression) Classification Report:

```

Classification Report:
              precision    recall  f1-score   support

     0       0.91      0.98      0.95      1414
     1       0.83      0.53      0.65       275

 accuracy          0.91      1689
 macro avg       0.87      0.76      0.80      1689
 weighted avg    0.90      0.91      0.90      1689

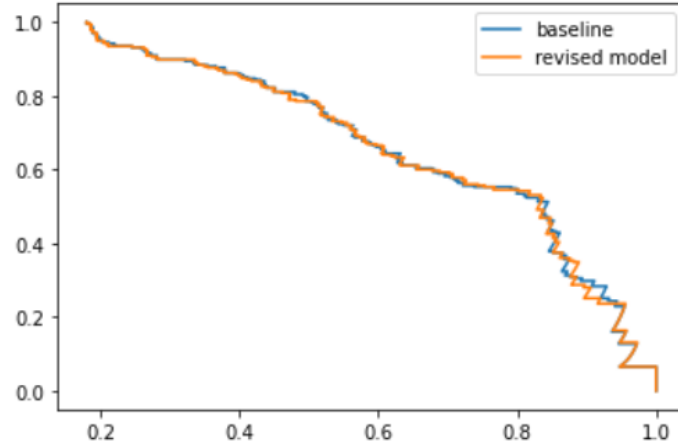
Confusion Matrix:
[[1385   29]
 [ 129  146]]

```

non-churn (0): no gain for precision and recall
churn (1): marginal gain in precision and recall

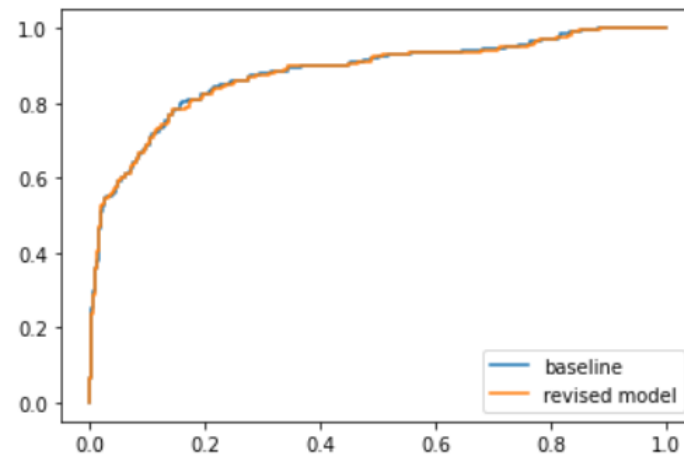
Revised Model (using L1 regression) Compared to Baseline:

Precision-Recall Curve:



The revised model has a similar precision-recall curve.

ROC / AUC Curve:



<i>AUC - Baseline</i>	<i>AUC - Revised Model (L1)</i>
~87.953%	~87.840%

The revised model (using L1 regression) has almost the same performance in ROC / AUC Curve.

```

# define different algorithms to use in the optimization problem of L1 reg
solvers2 = ['newton-cg', 'lbfgs', 'liblinear']

# define the norm used in the penalization as L1 regression
penalty2 = ['l2']

# define an array of C values (inverse of regularization strength)
c_values2 = np.linspace(0.01,1.5,50)

# initiate grid

grid2 = dict(solver=solvers2, penalty=penalty2, C=c_values2)
cv2 = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

```

initial values before cross validation of L2 regression

<i>Parameter</i>	<i>Values</i>	<i>Reason</i>
solver	'newton-cg'	Use 'newton-cg' algorithm, which can support L2 penalty, in the optimization problem
penalty	'l2'	Specify the norm used in the penalization
C	0.07081632653061225	Use a smaller C value to specify stronger regularization

Revised Model (using L2 regression) Classification Report:

```

Classification Report:
              precision    recall  f1-score   support

      0       0.91      0.98      0.94      1414
      1       0.86      0.49      0.63       275

 accuracy          0.90      0.90      0.89      1689
 macro avg          0.88      0.74      0.79      1689
 weighted avg          0.90      0.90      0.89      1689

```

```

Confusion Matrix:
[[1391  23]
 [ 139 136]]

```

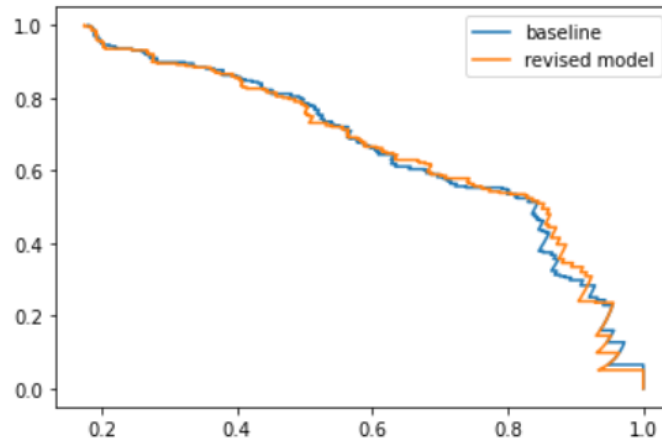
non-churn (0): no gain for precision and recall

churn (1): 3% gain in precision and 3% loss in recall

Revised Model (using L2 regression) Compared to Baseline:

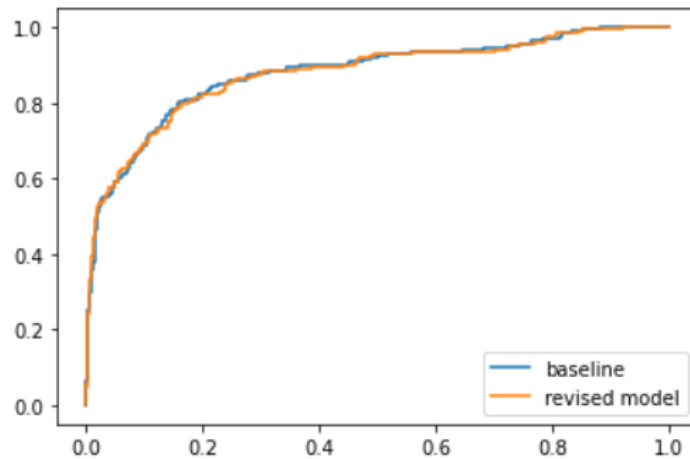
The revised model will have to be evaluated against the baseline based on the Precision-Recall curve and ROC / AUC curve.

Precision-Recall Curve:



The revised model has a similar precision-recall curve.

ROC / AUC Curve:



<i>AUC - Baseline</i>	<i>AUC - Revised Model</i>
~87.953%	~87.789%

The revised model (using L2 regression) has similar performance in terms of AUC.

Based on the results from ROC / AUC Curve, there are no significant discrepancies in the performances of the three models. The revised model generated by L1 regression is chosen instead of the other two because of its slight improvement in precision and recall. **The best performing model is L1.**

Naive Bayes

For the Naive Bayes models, we establish two benchmark models which predict the probability of all y to be zeros and ones. Then compare it with the Naive Bayes model.

Naive Bayes Model Specification:

```
gnb = GaussianNB()
```

Benchmark Model 0 (pred_val_maj = 0):

```
Accuracy of Benchmark Model:
0.837181764357608

Confusion Matrix of Benchmark Model:
[[1414   0]
 [ 275   0]]

Classification Report of Benchmark Model:
              precision    recall  f1-score   support

     0       0.84         1.00         0.91         1414
     1       0.00         0.00         0.00          275

   accuracy          0.84          1689
  macro avg          0.42          1689
 weighted avg          0.70          1689
```

In Benchmark Model 0, we assume that all the y values are 0. As such there is no false positive case (*predicted churn customers as not churn customers*) as no customers are predicted as churn.

Benchmark Model 1 (pred_val_maj = 1):

Accuracy of Benchmark Model:
0.16281823564239195

Confusion Matrix of Benchmark Model:
[[0 1414]
[0 275]]

Classification Report of Benchmark Model:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1414
1	0.16	1.00	0.28	275
accuracy			0.16	1689
macro avg	0.08	0.50	0.14	1689
weighted avg	0.03	0.16	0.05	1689

In Benchmark Model 1, we assume that all the y values are 1. As such there is no false negative cases (*predicted not churn customers are churn customers*) as no customers are predicted as not churn.

Naive Bayes Model:

Accuracy of Naive Bayes Model:
0.7572528123149793

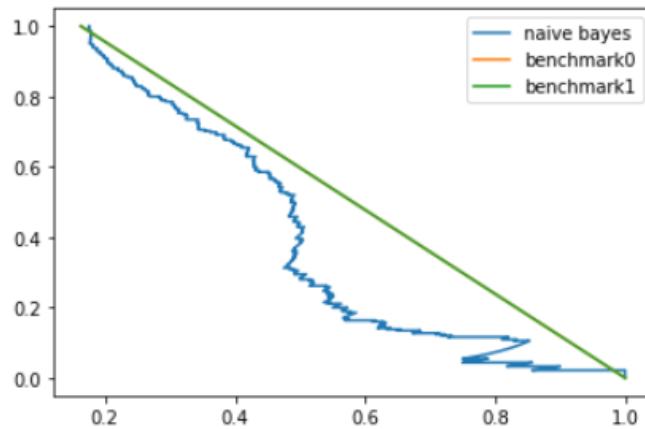
Confusion Matrix of Naive Bayes Model:
[[1106 308]
[102 173]]

Classification Report of Naive Bayes Model:

	precision	recall	f1-score	support
0	0.92	0.78	0.84	1414
1	0.36	0.63	0.46	275
accuracy			0.76	1689
macro avg	0.64	0.71	0.65	1689
weighted avg	0.83	0.76	0.78	1689

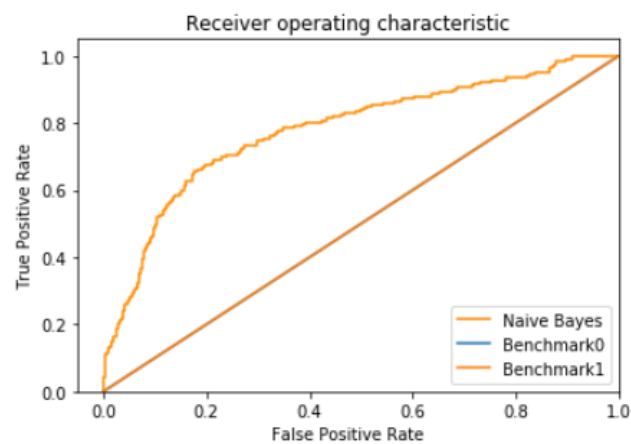
In this model, use 10-fold cross validation to predict y and the accuracy of the model predicted as 0.76. However, the result indicated that the false positive rate (the customers are churn but we predicted as no churn) is relatively high, 1001 instances.

Precision-Recall Curve:



The revised model performed a lot worse than the benchmark models.

ROC / AUC Curve:



<i>AUC - Naive Bayes</i>	<i>AUC - Benchmark 0</i>	<i>AUC - Benchmark 1</i>
0.78%	0.5%	0.5%

Based on the AUC of the three models, Naive Bayes model perform slightly better than the other two models as the **Naive Bayes model** has the highest AUC among the three models.

K Nearest Neighbor

For the K Nearest Neighbour, we first establish a baseline / benchmark, which will be compared to the improved model.

Baseline Specification:

```
baseline = KNeighborsClassifier()
```

Baseline Classification Report:

```
Confusion Matrix:
[[1373  41]
 [ 154 121]]
Classification Report:
              precision    recall  f1-score   support

     0       0.90       0.97       0.93       1414
     1       0.75       0.44       0.55        275

 accuracy          0.88       1689
 macro avg       0.82       0.71       0.74       1689
 weighted avg    0.87       0.88       0.87       1689
```

The parameters that the K Nearest Neighbours model will be initialized with are 'n_neighbors' and 'weights'. The parameters for cross validation will be initialized with a certain range.

```
# define weight
weights = ['uniform', 'distance']

#define n_neighbours
neighbors = np.arange(5, 65, 5)

#initiate grid
param_grid = dict(weights=weights, n_neighbors=neighbors)
param_grid
```

initial value before cross validation

Parameters	Values	Reason
'weights'	'distance'	Weight points by the inverse of their distance
'n_neighbors'	2	Limit the number of neighbours(k) required for each sample

Revised Model Classification Report:

Confusion Matrix:

```
[[1364  50]
```

```
[ 68 207]]
```

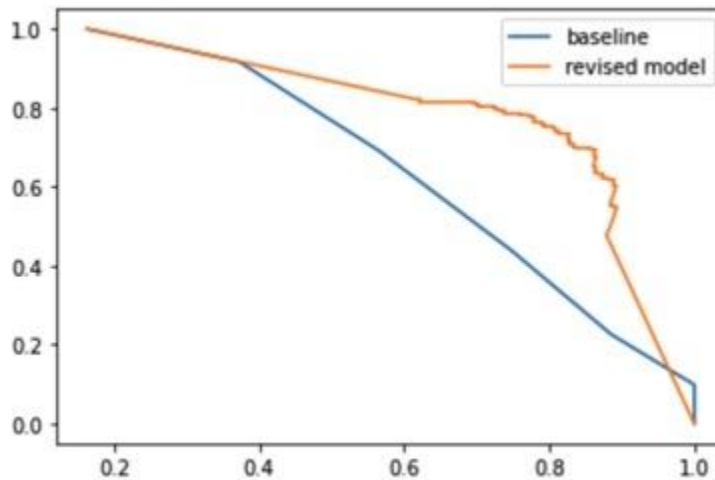
Classification Report

	precision	recall	f1-score	support
0	0.95	0.96	0.96	1414
1	0.81	0.75	0.78	275
accuracy			0.93	1689
macro avg	0.88	0.86	0.87	1689
weighted avg	0.93	0.93	0.93	1689

Revised Model Compared to Baseline:

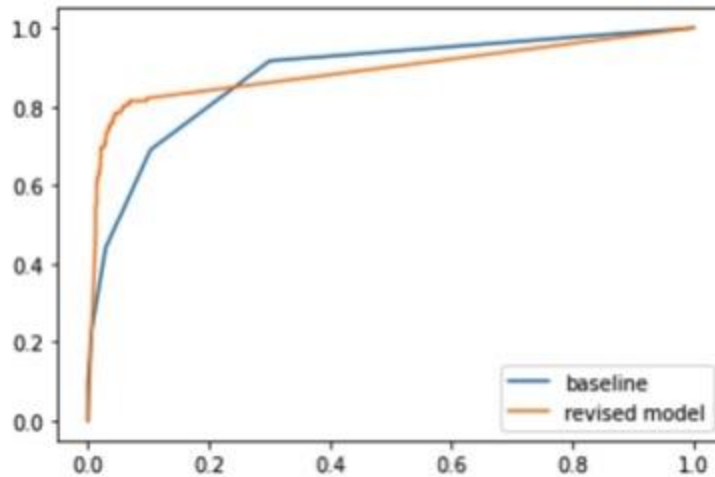
The revised model will have to be evaluated against the baseline based on the Precision-Recall curve and ROC / AUC curve.

Precision-Recall Curve:



The revised model is generally better than the baseline model.

ROC / AUC Curve:



<i>AUC - Baseline model</i>	<i>AUC - Revised model</i>
~87.953%	~88.996%

Based on the AUC of the two models, the revised model performs slightly better than the baseline model. However, there is an intersection point between the two ROC curves. Therefore, the **revised model is better** than the baseline model when the decision threshold is high enough.

Clustering

K Means Clustering

To determine the clusters and segments of the customers, K Means Clustering would be used. It is not part of the classification question and would not be used to predict churn and non churn customers. The model with the parameters are set as below:

```
kmeansmodel = KMeans(n_clusters=3, n_init=10, max_iter=100)
```

<i>Parameter</i>	<i>Values</i>	<i>Reason</i>
'n_clusters'	3	A total of three clusters is set.
'n_init'	10	10 runs to initialize the centroid seeds.
'max_iter'	100	iterate the calculations 100 times to ensure convergence.

The model is then ran with 13 features - CityTier, NumberOfDeviceRegistered, SatisfactionScore, NumberOfAddress, Complain, CouponUsed, OrderCount, Tenure, HourSpendOnApp, WarehouseToHome, DaySinceLastOrder, OrderAmountHikeFromlastYear, and CashbackAmount.

Then each of the clusters are grouped and the mean values for each feature is compared. It seems like there are only four features with significant difference in values - NumberOfAddress, CouponUsed, OrderCount, and DaySinceLastOrder.

	NumberOfAddress	CouponUsed	OrderCount	DaySinceLastOrder
Cluster				
0	2.923077	1.161254	1.940171	0.088432
1	7.979102	1.375387	2.138545	0.081690
2	3.811594	5.121981	9.070048	0.169252

cluster significant features

These clusters can be interpreted differently, which will be elaborated in the [Conclusion](#) section in question 4.

Performance Evaluation

Metric Selection

The models are evaluated with specific metrics. Firstly, the dataset is imbalanced, because the prediction label (Churn) does not have an equal distribution of samples. In the [Descriptive Statistics](#) section, Non-Churn customers are 4.5x more frequent than Churn customers. This means that **accuracy cannot be used** to evaluate the models. Therefore, there should be a **focus on Precision and Recall**.

After more expansive research, we have concluded that it is more expensive to acquire new customers than to maintain existing customers. The more customer churn, the more money the business must spend on recouping the loss of business by finding new ones.^[2] This means it is critical to **reduce the churn customers**, which can be done by firstly correctly identifying the customers who are likely to churn and spending efforts to target those customers.

False Negatives are more costly to the business because it is a situation where we incorrectly identify Churn customers are Non-Churn customers, which will consequently

cause the customers to Churn. To **reduce the False Negative**, it means we want the **Recall percentage** for Churn customers to be higher.

		Predicted class	
		+	-
Actual class	+	True + (TP)	False - (FN)
	-	False + (FP)	True - (TN)

False Negative: predicted churn customers as not churn customers

False Positive: predicted not churn customers are churn customers

To find a balance between Recall Percentage and Precision Percentage, we will be **comparing Precision-Recall curves**, by looking at True Positive, False Negative, and False Positive.

It is also equally important to **consider the False Positive** because there are also costs associated with it. Acquiring a new customer is about 25 times more expensive than retaining an existing one.^[8] This means that the ratio of 25:1 [False Negative:False Positive] will be used to calculate the costs associated with the predictions. We will also **compare ROC / AUC curves** to take a machine learning, heuristic approach to evaluating the prediction power of each of the models because all four elements True Positive, False Negative, False Positive, and True Negative are important.

Another approach from a business shareholders perspective is to look at the cost-sensitive classification. Since there is a ratio of 25:1 in terms of cost associated with False Negative and False Positive, there should be an **equilibrium between True Positive Rate and False Positive Rate where the Cost is the lowest**. The Cost Curve will also be employed to decide the optimal decision threshold. The equation for costs will be used for calculation:

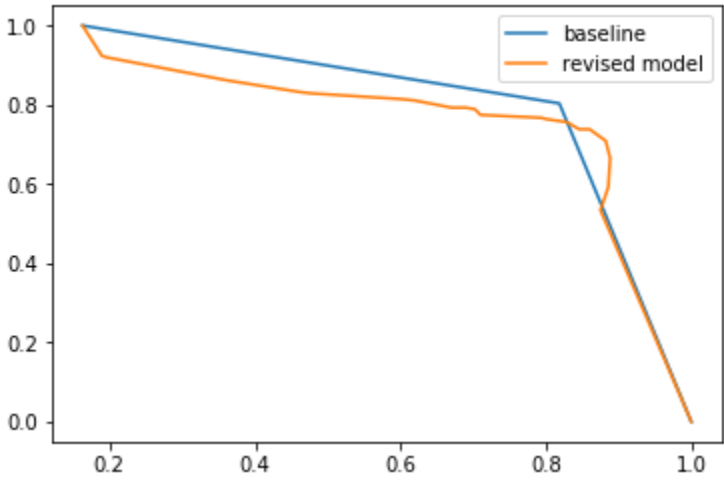
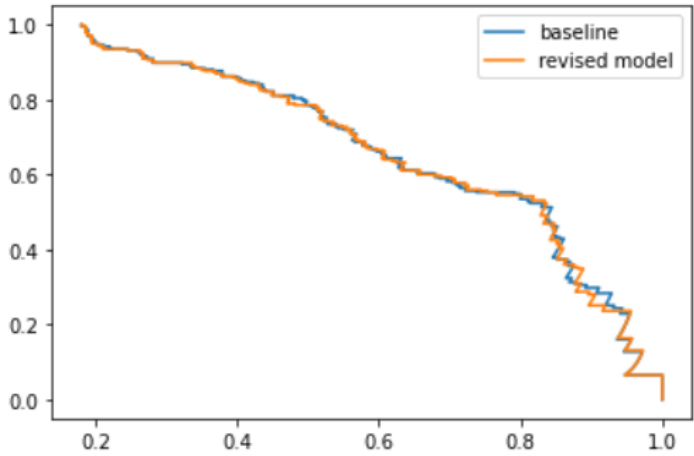
$$\text{Cost} = \text{FP} \times (\$1) + \text{FN} \times (\$25)$$

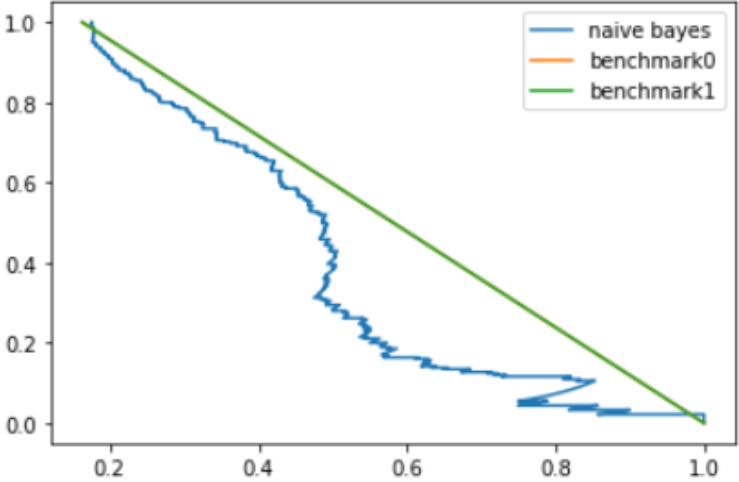
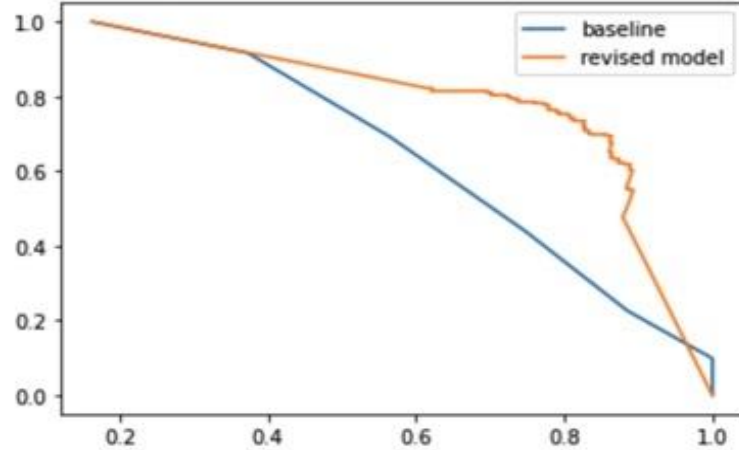
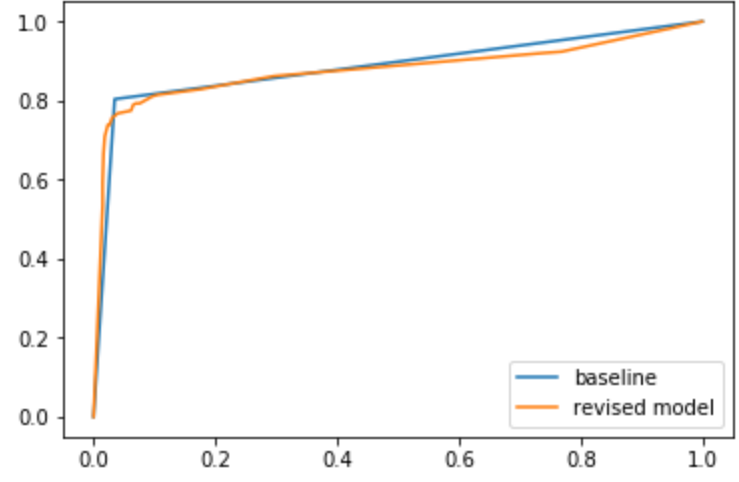
FP: False Positive

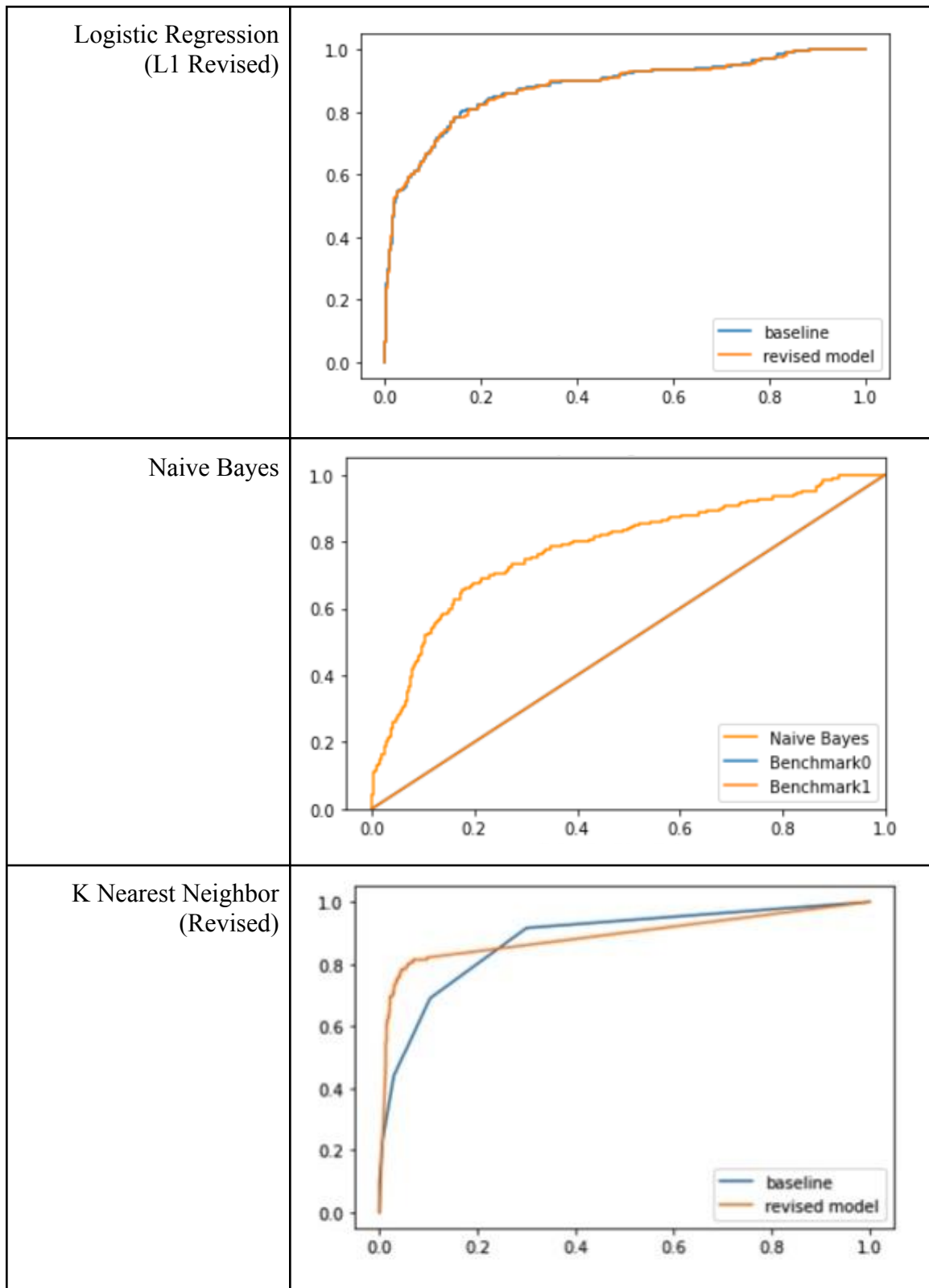
FN: False Negative

Evaluation

Selection criteria of the model includes: Precision-Recall Curve and ROC / AUC Curve.

<i>Model</i>	<i>Precision-Recall Curve</i>
Decision Tree (Revised)	 <p>The Precision-Recall Curve for the Decision Tree (Revised) model compares a baseline (blue line) and a revised model (orange line). The x-axis represents Recall from 0.2 to 1.0, and the y-axis represents Precision from 0.0 to 1.0. The baseline starts at (0.2, 1.0) and decreases to (0.8, 0.8), then drops sharply to (1.0, 0.0). The revised model starts at (0.2, 0.95), maintains higher precision than the baseline until recall reaches 0.8, and then drops to (1.0, 0.0).</p>
Logistic Regression (L1 Revised)	 <p>The Precision-Recall Curve for the Logistic Regression (L1 Revised) model compares a baseline (blue line) and a revised model (orange line). The x-axis represents Recall from 0.2 to 1.0, and the y-axis represents Precision from 0.0 to 1.0. Both curves start at (0.2, 1.0) and follow a similar downward path until recall reaches 0.8, where precision is approximately 0.55. Beyond this point, the revised model (orange line) maintains higher precision than the baseline (blue line) as recall increases towards 1.0.</p>

Naive Bayes	 <p>This ROC curve compares the performance of a Naive Bayes model (blue line) against two benchmarks (orange and green lines). The x-axis represents the False Positive Rate (FPR) and the y-axis represents the True Positive Rate (TPR), both ranging from 0.0 to 1.0. The blue line starts at (0,1) and follows a jagged path towards (1,0), indicating moderate performance. The orange line (benchmark0) and green line (benchmark1) are diagonal lines from (0,1) to (1,0), representing random performance.</p>
K Nearest Neighbor (Revised)	 <p>This ROC curve compares the performance of a baseline K-Nearest Neighbor model (blue line) with a revised model (orange line). The x-axis is FPR and the y-axis is TPR, both from 0.0 to 1.0. The blue line starts at (0,1) and curves down to (1,0). The orange line starts at (0,1), follows the blue line closely until FPR ≈ 0.4, then stays higher (better performance) until FPR ≈ 0.8, before dropping to (1,0). This indicates the revised model outperforms the baseline.</p>
Model	ROC / AUC Curve
Decision Tree (Revised)	 <p>This ROC curve compares the performance of a baseline Decision Tree model (blue line) with a revised model (orange line). The x-axis is FPR and the y-axis is TPR, both from 0.0 to 1.0. Both lines start at (0,0) and rise sharply to a TPR of approximately 0.8 at a very low FPR. The orange line (revised model) is slightly above the blue line (baseline) for the remainder of the curve, indicating superior performance.</p>



Based on these metrics the best performing tentative models are **Decision Tree** and **Logistic Regression (L1)**. According to the Precision-Recall Curve and ROC / AUC

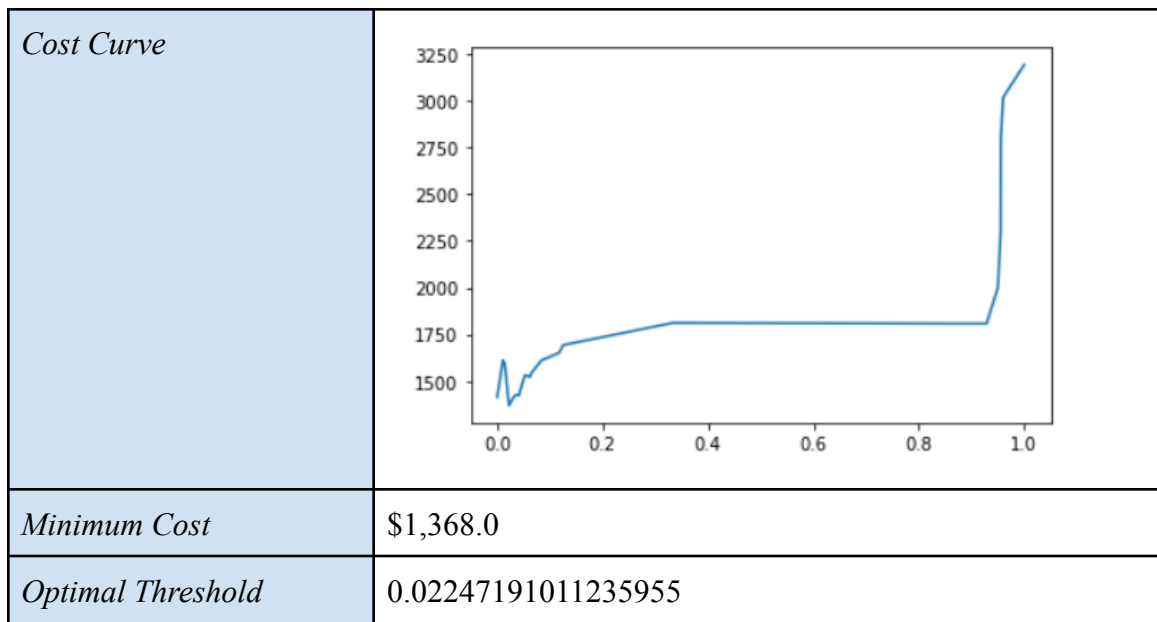
curves the differences are only marginal and it is inconclusive which model performs the best; therefore to determine the best model, we will have to optimize the models according to the costs and decision threshold.

Cost Sensitive Analysis & Decision Threshold

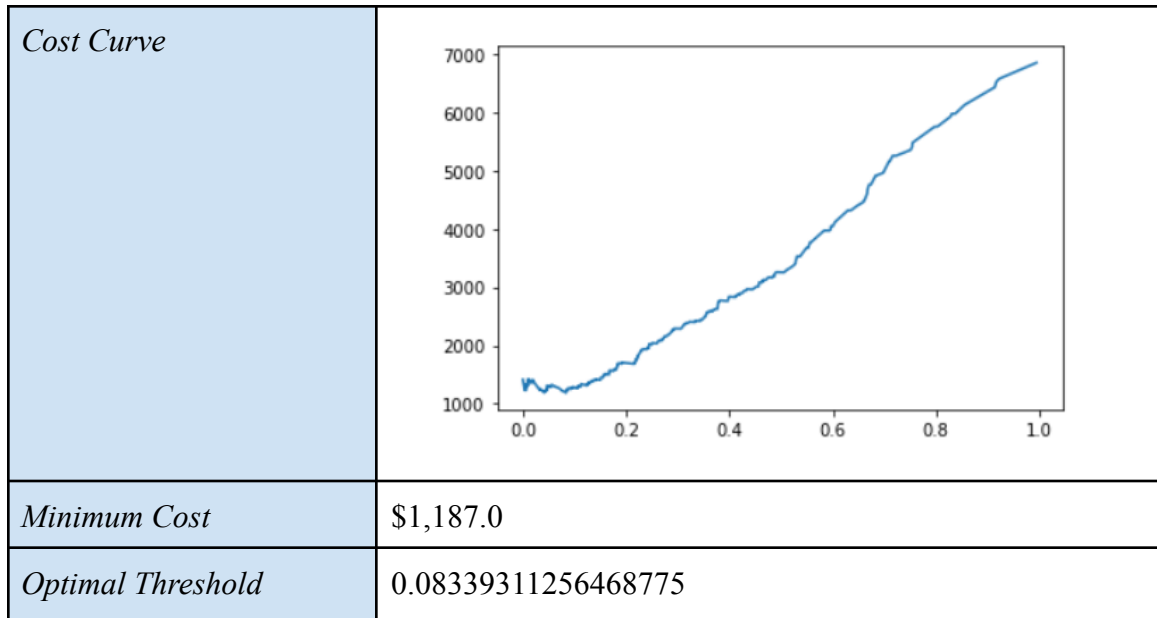
Based on the models, we would want to calculate the Decision Threshold versus Cost curve to determine the optimal Decision Threshold that minimizes costs. The Cost Curves below are generated using the function:

$$Cost = FP \times (\$1) + FN \times (\$25)$$

Decision Tree



Logistic Regression (L1)



The generated Cost Curves are not shaped like a typical U-shaped curve, with the Decision Tree plateauing around the decision threshold 0.3 - 0.9 and exponentially increasing at 0.95. On the other hand, the Logistic Regression (L1) exhibits positive linear behavior after the decision threshold 0.1. The reason why these curves are not completely U-shaped because the magnitude of the False Negative (\$25) outweighs the False Positive (\$1). At higher decision thresholds, the misclassification to False Negative is higher and causes the cost to increase drastically; therefore, the optimal decision thresholds for both the models are relatively low. This would mean that the cost of losing a customer is so high that the business cannot risk that happening. Consequently, it would be more cost efficient to classify most customers as Churn customers even if they display some characteristics of having a propensity to Churn. Customers would receive customer retention promotions even if they do not have high propensity to churn.

After determining the optimal decision threshold for each model, the **Logistic Regression** model generated from **L1 regression** performs better than the Decision Tree model in minimizing the total costs of False Negative predictions.

Conclusion

After modelling the dataset with multiple machine learning models, insights were derived to answer the business questions formulated in the beginning. The models, Logistic L1 Regression and Decision Tree, will be used as referenced for the questions below:

- 1) Which feature(s) influence the customer churn rate most?

According to the text representation of the revised decision tree model, the top three nodes are Tenure_zscore, Complain, and CashbackAmount respectively. It can be concluded that customer tenure is the most informative feature, followed by whether the customer had complained before, and cashback.

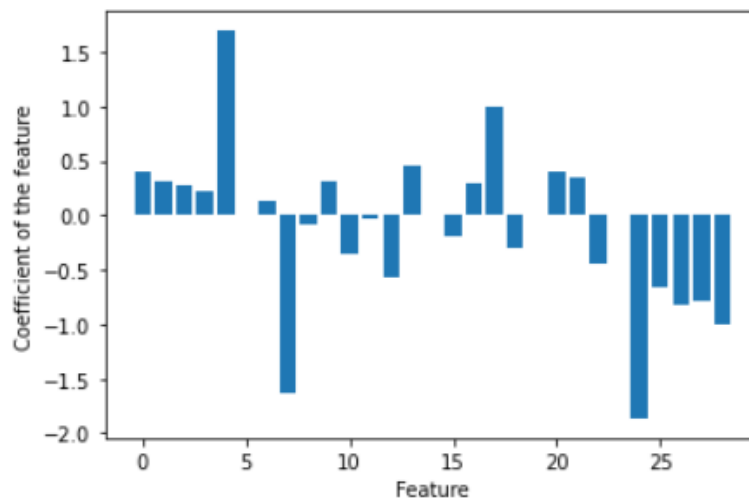
```

|--- feature_7 <= -1.04
|   |--- feature_4 <= 0.50
|       |--- feature_3 <= 4.50
|           |--- feature_10 <= -0.86
|       |--- feature_12 > -1.14
|           |--- feature_4 <= 0.50
|               |--- feature_5 <= 6.50
|                   |--- feature_22 <= 0.50

```

It is reasonable that the customer tenure is the most influential feature, as tenure refers to the number of months that a customer has subscribed for. It is a key indicator to review the customer loyalty towards the company. Customers with higher tenure score are less likely to churn

From the Logistic Regression model generated from L1 regression, the coefficient values against features are plotted in the graph below.



The more informative the feature, the larger the absolute value of its coefficient. The most informative features from the Logistic Regression model were found to be whether the customers' preferred order category is laptop and accessory, subsequently whether the customer had complained before and tenure.

The influential features are then decided from the coinciding results from the best performing models. Therefore, it can be concluded that whether the **customer had complained before** and the number of months that the **customer spent as a subscriber (tenure)** are the most predictive features of whether a customer would

churn.

- 2) How to utilize the appropriate promotions to minimize the customers with propensity to churn and turnover?

There are two notable features that influence customer churn rate the most - Tenure_zscore and Complain. To reduce the customer churn rate and turnover strategies will have to be device:

<i>Feature</i>	<i>Strategy</i>
Tenure_zscore	‘Tenure’ describes the length of the time the customer has been a subscriber and has been using the services of the organization. The reason why this feature serves as the best churn rate determinant is because longer tenure customers tend to be Non-Churn customers, while lower tenure customers tend to Churn customers. There needs to be a focus on retaining the customer and converting them into long-term customers. To do so, there will be a rewards system implemented that will incentive customers to become recurrent customers. Such promotions will include discounts and exclusive offers when customers are utilizing the services for a longer duration. These rewards will be transparent and customers will know how long they will have to stay with the organization to receive these rewards. Also a “Check In” system will be implemented, which is a reward system that distributes rewards based on the consecutive check-in weekly.
Complain	‘Complain’ describes the amount of complaints that the customer has raised. It seems like the more complaints that customers have the higher tendency the customer is to Churn. Therefore, there needs to be a focus on reducing the number of complaints but more importantly the unresolved complains. To do so, there will be frequent surveys and machine learning models run on customer complaints data to determine which are the most problematic issues. Afterwards there will be a focus on fixing these issues. Additionally, there will be a machine learning chatbot that will attempt to resolve customer complaints immediately without the handling of customer representatives.

- 3) Possibility to segment the customers to better cater to their needs?

To segment the customers, [K Means Clustering](#) was utilized to determine the

three distinct clusters. The characteristics of the clusters are summarized below:

<i>Cluster</i>	<i>Description</i>
0	Customers with LOW number of addresses, LOW coupons used and order count, MORE recent orders.
1	Customers with MEDIUM number of addresses, HIGH coupons used and order count, LESS recent order.
2	Customers with HIGH number of addresses, LOW coupons used and order count, MORE recent orders.

According to the table, it seems **Cluster 0** customers seem to have a low order count and coupons used with more recent orders. These customers may be customers that only order once or twice in a month, but with a large basket per order. To incentivize these customers to increase their order count, we could target these customers by giving better discounts and more coupons, which may lead to increased order count. As for **Cluster 1**, these are customers with high order count and coupons used but less recent orders. These customers seem to be compulsive shoppers that order a lot of items sporadically and supposedly a low basket per order. To target these customers, we could have more personalized recommendations, which will be sent via marketing communication channels, to incentive the customer to purchase more. Lastly, **Cluster 2** customers are those with low order count and coupons used but more recent orders. These may be customers that were recently attracted by the personalized recommendations made through marketing communication channels. To target these customers, we would also give better discounts and more coupons that are targeted towards the recommended items. The needs of the cluster customers will be fulfilled according to these methods.

Overall, the machine learning models derived many useful insights. Though the modelling was quite expansive, it can be continually expanded to derive future insights. In the future the cost-sensitive analysis can be built upon. Currently, there are costs related to False Positive and False Negative; furthermore, there should be profits associated with True Positive and True Negative. This will give a more accurate measure on the performance of the model. Also the use of unsupervised learning can be utilized in the downstream pipeline to prepare the dataset through clustering before the supervised learning, meanwhile ensemble learning can be used to get a more aggregated and accurate result. Additionally, because our dataset has only 19 features preprocessed, there will be no need for dimension reduction; however, if in the future the collection of the

data features increases to over 100, then dimension reduction would be used. Lastly, to add on, two additional methods (machine learning chatbot and recommendation system based on collaborative and content-based filtering) were determined to be effective in the Question 2 in the Strategy section for minimizing customer churn rate, which will require more data as well.

References

- [1] <https://www.forbes.com/sites/johnkoetsier/2020/06/12/covid-19-accelerated-e-commerce-growth-4-to-6-years/>
- [2] <https://backlinko.com/amazon-prime-users#amazon-prime-usage-statistics>
- [3] <https://www.kaggle.com/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction>
- [4] <https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics>
- [5] <https://www.codecademy.com/articles/normalization>
- [6] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- [7] <https://www.profitwell.com/customer-churn/analysis>
- [8] <https://hbr.org/2014/10/the-value-of-keeping-the-right-customers>