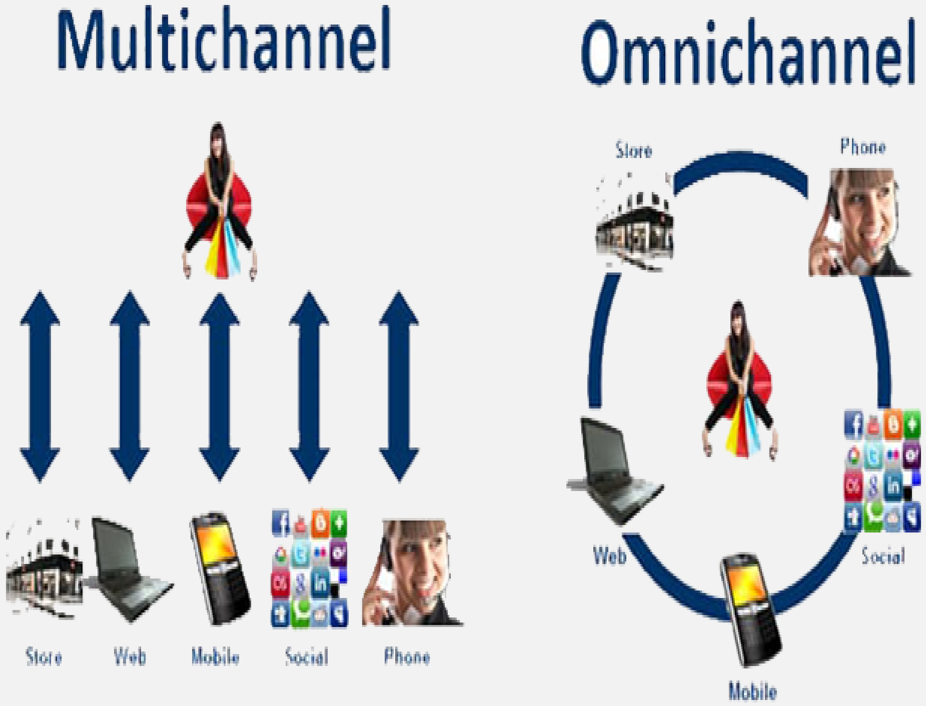
Predictive Churn model based on Customer RFM Segment

By: ANG Kuo Sheng Clement
Email: kuosheng.ang@outlook.com

PROBLEM STATEMENT



- ecommerce market getting more and more competitive with many new entrants globally & locally, no barriers to entry among competitors rivalry

- making it challenging to achieve customer retention via multichannel, omni-channel content and marketing strategies

- In a competitive & market saturation in most industries where the acquisitions of new customers of “brand new" customer is virtually impossible. the solicitation of customers from a competitor is also presents challenges and costly.

Objective

- Used as Customer Relationship Management to identify which marketing campaign or customer rewards scheme works for respective Customer Segment.
- Recency, Frequency, and Monetary Value (RFM) helps inform customer segmentation and identifies customers with propensity to churn
- Based on customer's purchase history enables the establishment of churn thresholds for each customer group and assists in constructing a model to predict future churners.
- To train DecisionTree classifier with optimal tree depth. Model suitability is evaluated via F1 score as well as Accuracy.

Segment	RFM	Description	Marketing
Best Customers	111	Bought most recently and most often, and spend the most	No price incentives, new products, and loyalty programs
Loyal Customers	X1X	Buy most frequently	Use R and M to further segment
Big Spenders	XX1	Spend the most	Market your most expensive products
Almost Lost	311	Haven't purchased for some time, but purchased frequently and spend the most	Aggressive price incentives
Lost Customers	411	Haven't purchased for some time, but purchased frequently and spend the most	Aggressive price incentives
Lost Cheap Customers	444	Last purchased long ago, purchased few, and spent little	Don't spend too much trying to re-acquire

DATASET

Dataset is like any typical sales transactions from retail e-commerce store

Necessary steps to prepare the dataset before performing any descriptive, prescriptive and predictive analysis.

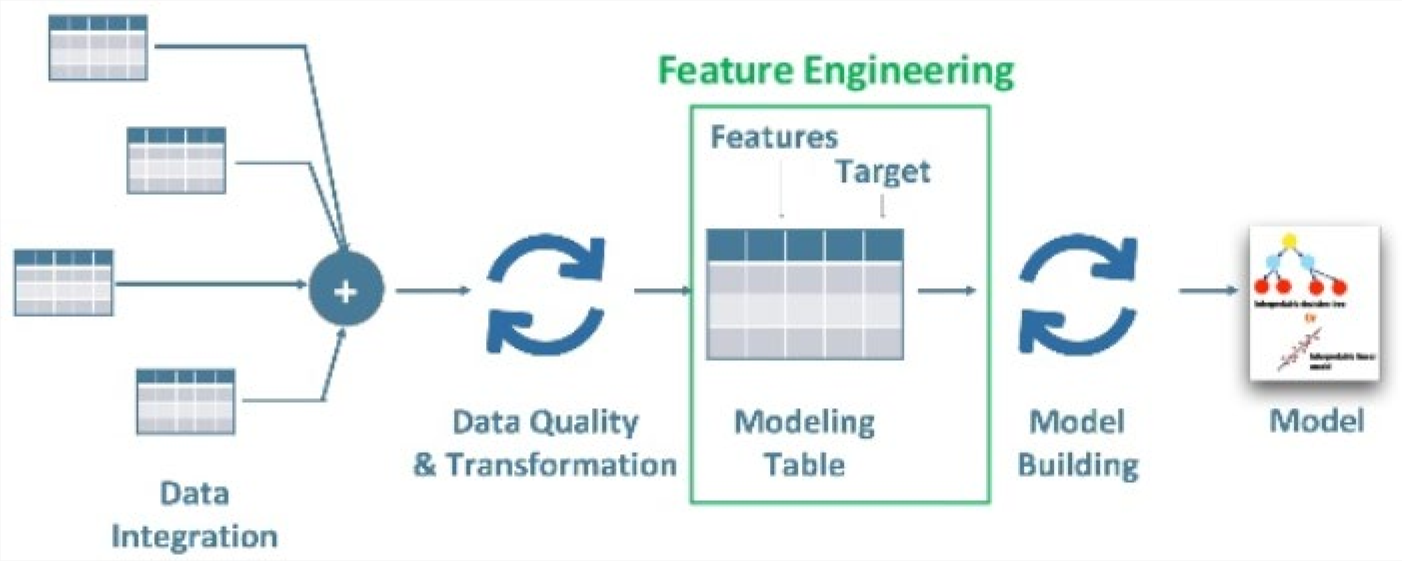
1. To perform data cleaning (ie: Checking for any transactions_id with duplicate entries for removal & removal of data containing null values, negative values, invalid dates).
Hence, any negative values in quantity or transactional figures are disregarded or converted into absolute figure to reduce the complexity for RFM analysis as well as predictive analysis using classification.

2. Perform Standard Normalization / Min-Max Scaling (ie: Data Normalization due to different statistical distribution, skewness, outliers found in certain variables).

Standard scaling was applied for data normalization before dataset is split into training/testing & applied for DecisionTree classification.

Field	Description
customer_Id	Unique identification number of a customer
DOB	Date of Birth
Gender	Gender of Customer
city_code	City code Customer has registered
transaction_id	Transactions Identifier
customer_Id	Customer Identifier
tran_date	Date of transaction performed
prod_subcat_code	Product Sub-Category
prod_cat_code	Product Category
Qty	Quantity purchased by customer
Store_type	Type of Stores
total_amt	Total Sales Amount
prod_cat_code	Product Category Code
prod_cat	Product Category
prod_subcat_code	Product Sub-Category

DATASET FEATURES



Features selection of dataset used for analysis:

1. Consumer's spending behavioral related data, such as spending and consumption habits on category of product/service purchase for all the transactions
2. Store Type is useful for tracking customer engagement in omni-channel distribution or multi-channel

Descriptive Analysis

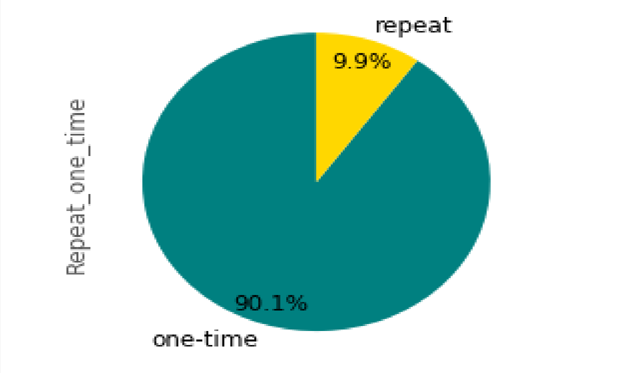
```
cs_freq_summary = cs_df.groupby(['customer_Id', '
transaction_id'])['transaction_id'].aggregate({'freq_transaction_count':'count'})

cs_freq_summary['Repeat_one_time'] = np.where((cs_freq_summary.freq_transaction_count>1),
'repeat', 'one-time')

print(cs_freq_summary['Repeat_one_time'])
plt.style.use('ggplot')

media_df=cs_freq_summary
media_per_user_group=media_df.groupby(['Repeat_one_time'])['Repeat_one_time'].count().nlargest(2)
media_per_user_group.plot(kind='pie', colors = ['teal','gold'], fontsize=12, autopct='%1.1f%%',
startangle=90, pctdistance=0.85)
plt.show()
```

Proportion of transactions with repeat buy (recurring sales)



		freq_transaction_count
customer_Id	**transaction_id**	
266783	**8410316370**	1
	16999552161	1
	25890929042	2
	98477711300	1
266784	**26928161256**	1

- there's a small fraction 10% of customers with repeat-buy & 90% with only 1 single transaction, depicts the opportunity to address on the issue of F (frequency) within RFM analysis to enhance CLV (Customer Lifetime value).

```
#  customer & transaction counts
cs_transac_prod_cat =
cs_df.groupby(['customer_Id','transaction_id'])['prod_cat','Store_type'].
aggregate('count').reset_index().sort_values('customer_Id',
ascending=True).head(20)
```

Customer transaction profile

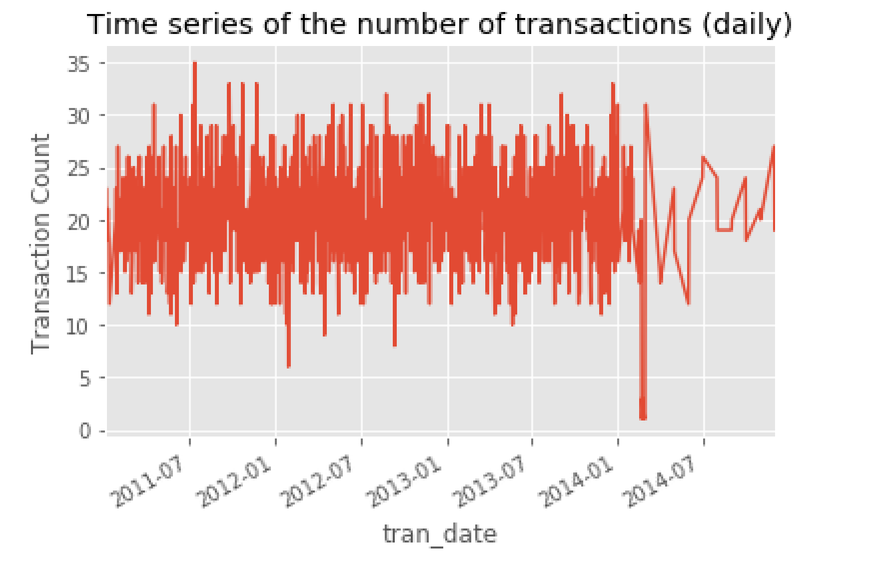
92

	customer_Id	transaction_id	prod_cat	Store_type
0	266783	8410316370	1	1
1	266783	16999552161	1	1
2	266783	25890929042	2	2
3	266783	98477711300	1	1
4	266784	26928161256	1	1
5	266784	36310127403	1	1
6	266784	54234600611	1	1
13	266785	96176911576	2	2
12	266785	94925617839	1	1
10	266785	79527990288	1	1
11	266785	89882144571	1	1
8	266785	62414620900	1	1
7	266785	17960226367	1	1

Descriptive Analysis

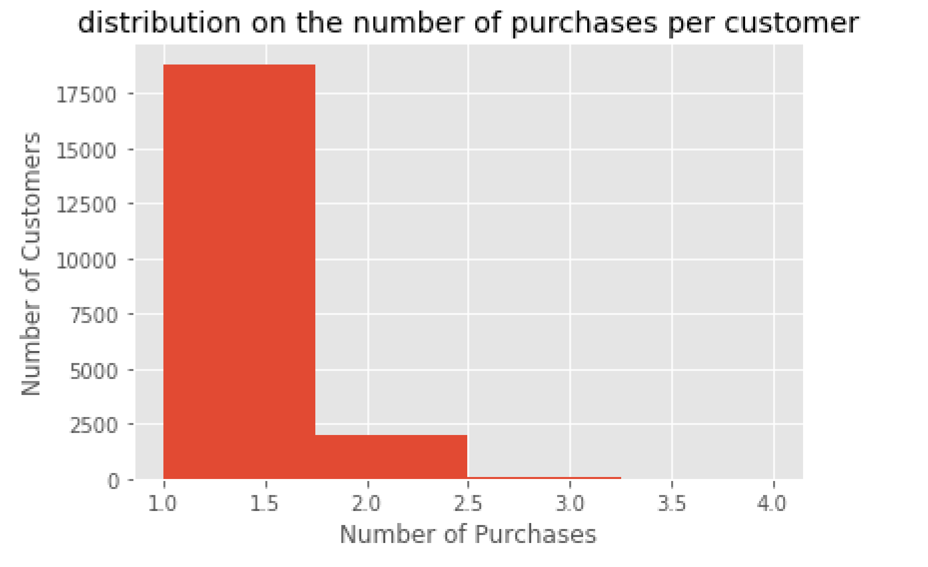
```
# number of transactions count daily in time-series

cs_df['tran_date'] = pd.to_datetime(cs_df['tran_date'])
ts_transactions = cs_df.groupby(['tran_date']).size()
plt.ylabel('Transaction Count')
plt.title('Time series of the number of transactions (daily)')
ts_transactions.plot()
```



- there's a sharp break in the counts at the end of 2014-07 onwards which reflects the decrease in recency and frequency.
- This is a standard practice when modeling CLV. The cohort of customers used to train the models are generally based on their time of first purchase. That way, one can study the evolution of the population parameters over time and pinpoint possible problems in the long run.

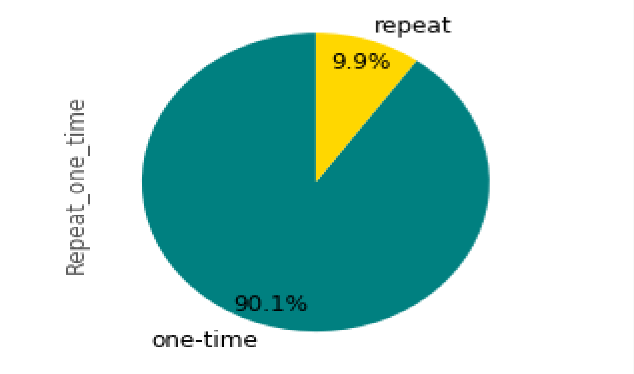
```
# histogram chart - number of purchases per customer
n_purchases = cs_df.groupby(['customer_Id','transaction_id']).size()
print(n_purchases.min(axis=0), n_purchases.max(axis=0))
n_purchases.hist(bins=(n_purchases.max(axis=0) -
n_purchases.min(axis=0)) + 1)
plt.title('distribution on the number of purchases per customer')
plt.xlabel('Number of Purchases')
plt.ylabel('Number of Customers')
```

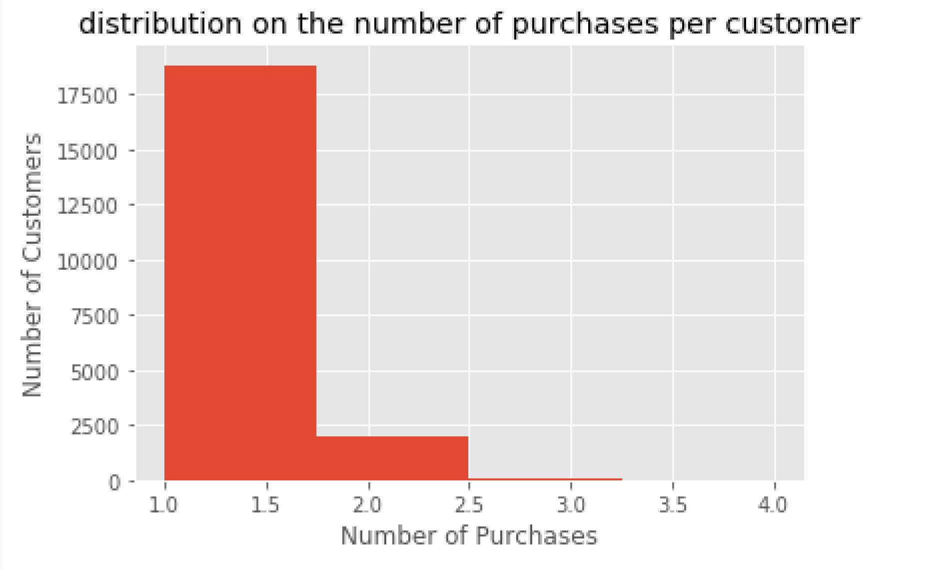


- As we see in the histogram figure above, 90% of the customers made only a single purchase and 10% with more than 1 single purchase which tallies with pie-chart illustrated in the previous slide

Descriptive Analysis

Proportion of transactions with repeat buy (recurring sales)





Possible reasons for large proportion of One-Time Buy phenomenal

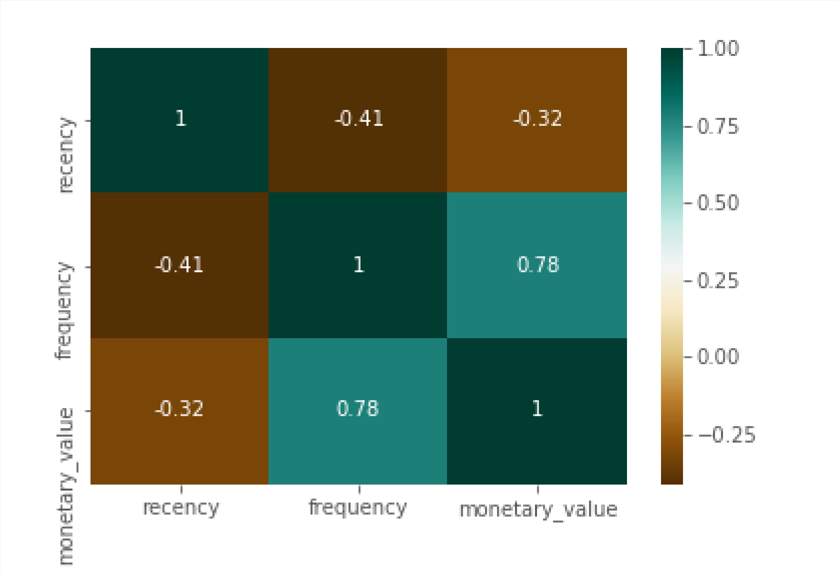
- Marketing perks or discount incentives given to new members ("First-time shopper" or " New Users" offer)

To overcome this issue of having multiple new customer accounts is to develop customer loyalty rewards program.

This will ensure all the customers accounts are unique or no duplicate accounts (ie: 1 person with multiple username or email account)

Descriptive Analysis

```python
# correlation table
c= rfmTable[['recency','frequency','monetary_value']].corr()
sns.heatmap(data=c,cmap="BrBG",annot=True)
plt.show()
```



- Frequency and monetary value are positively correlated with each other implying an increase in frequency implies increase in monetary value

- Frequency and Recency are negatively correlated with each other implying an increase in frequency implies decrease in monetary value

Prescriptive Analysis

RFM Analysis is applied using RFM Score formula based on 4 equal quintiles (25% group), which divide customers into various segments or clusters, also known as Customer Segmentation.

Purpose of RFM Analysis:

- identify customers reaction & respond to promotions and also for future personalization services
- Improve marketing performance by making campaigns relevant to customers, thus increasing response rate and sales revenue
- allow firms marketer to generate different marketing strategies or promotional campaign accordingly to increase customer retention, loyalty and customer lifetime value

Type of Customer Segments/Clusters:
(ie: BEST Customers, High-Spending New Customer, Lowest Spending Active Loyal, Loyal, Potential Churn)

Segment	RFM	Description	Marketing
Best Customers	111	Bought most recently and most often, and spend the most	No price incentives, new products, and loyalty programs
Loyal Customers	X1X	Buy most frequently	Use R and M to further segment
Big Spenders	XX1	Spend the most	Market your most expensive products
Almost Lost	311	Haven't purchased for some time, but purchased frequently and spend the most	Aggressive price incentives
Lost Customers	411	Haven't purchased for some time, but purchased frequently and spend the most	Aggressive price incentives
Lost Cheap Customers	444	Last purchased long ago, purchased few, and spent little	Don't spend too much trying to re-acquire

RFM Segment Table using past purchase transaction history

27

	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile	RFMScore
customer_Id							
266783	457	5	14791.530	2	2	2	222
266784	815	3	5694.065	4	4	3	443
266785	658	8	35271.600	3	1	1	311
266788	366	4	6092.970	1	3	3	133
266794	1	12	28253.745	1	1	1	111
266799	93	4	9958.260	1	3	2	132
266803	1031	1	3984.630	4	4	4	444
266804	484	1	1588.990	3	4	4	344
266805	341	1	4623.320	1	4	4	144
266806	370	6	20229.235	2	2	1	221

Prescriptive Analysis

This treemap also show the class imbalance of RFM segment that needs to be handled differently instead of using predefined partitioning.

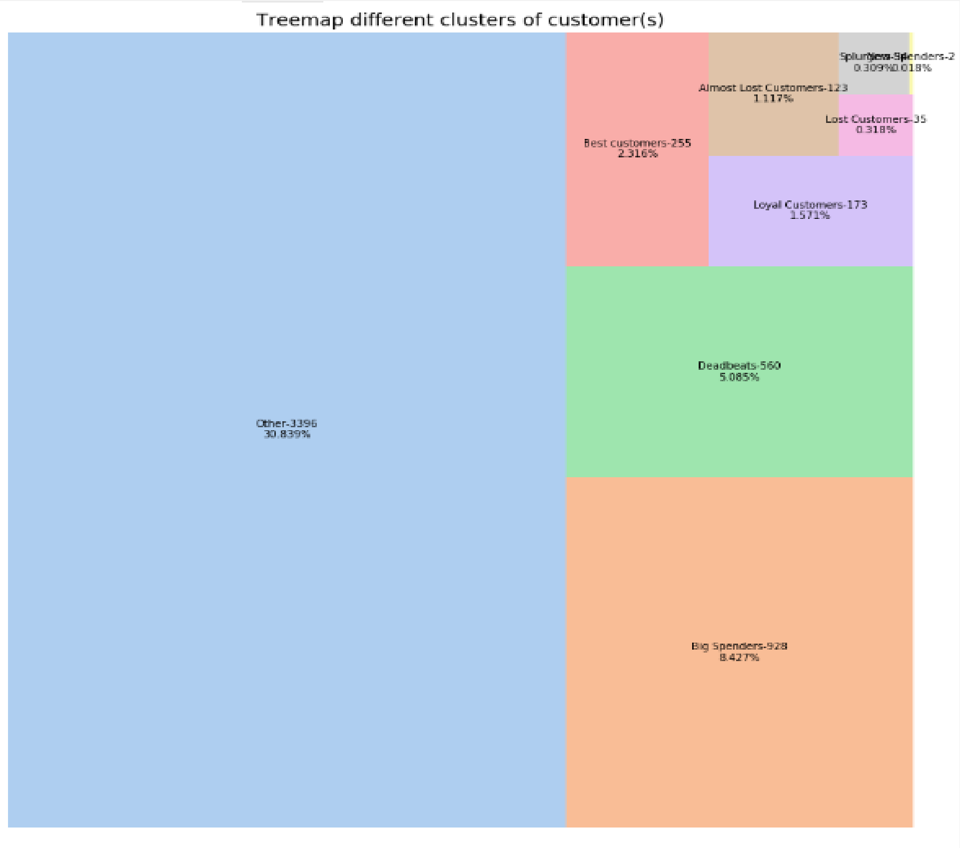
Pre-defined partitioning (80-20%) affects many classifiers model's performance through the lack of opportunity for the algorithm to learn due to sample representative having class imbalance, bias or skewed issue. Class imbalance is known to affect the performance of many classifiers by introducing a bias towards the majority class of target variable such as "Others".

Hence, random sampling is recommended to achieve independence and also a smooth generation of samples without any bias on the training set. Alternatively, splitting on the target (dependent) variable to ensure that we don't train the classifier on imbalanced data.

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,
random_state=123, stratify=y)

skf = StratifiedKFold(n_splits=10,random_state=1).split(X_train, y_train)
```

Treemap Generated from dataset using Python

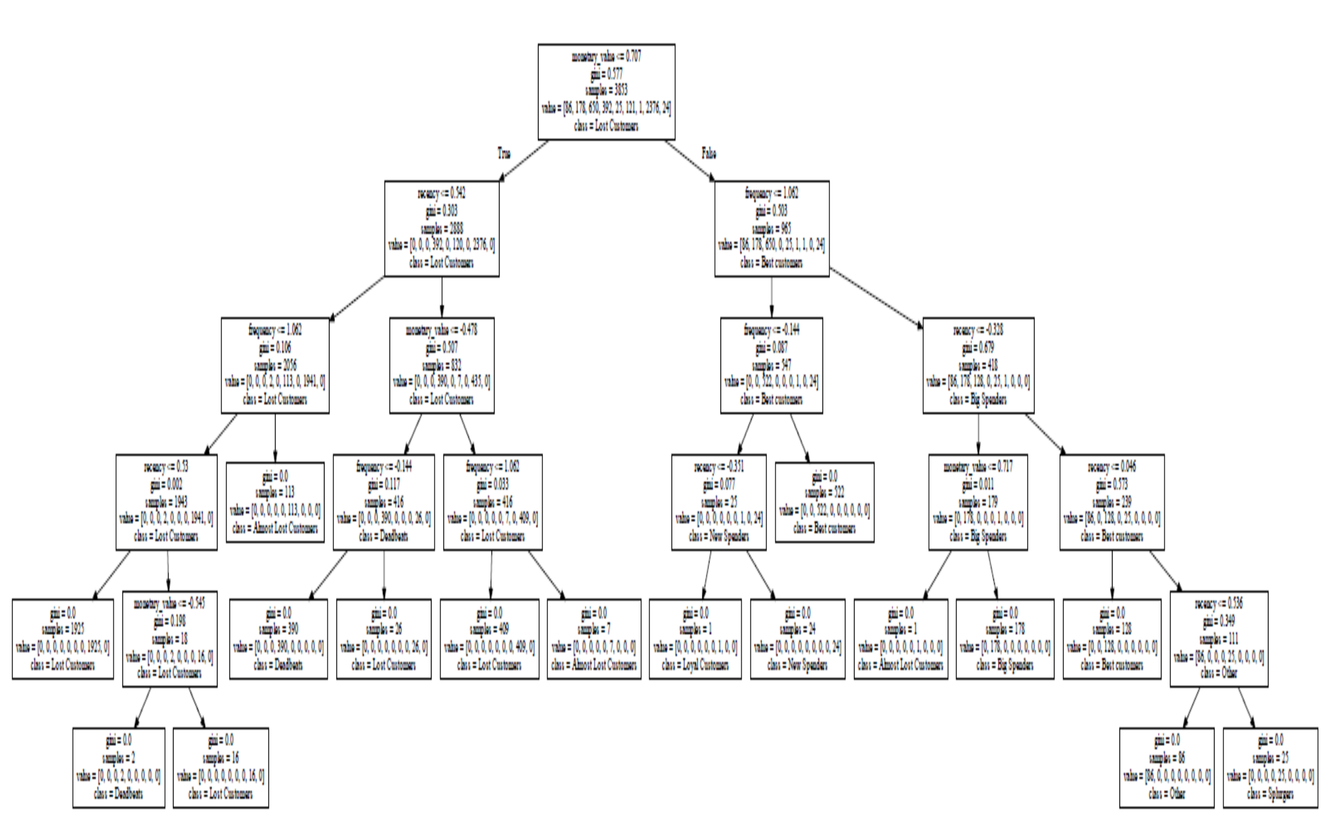


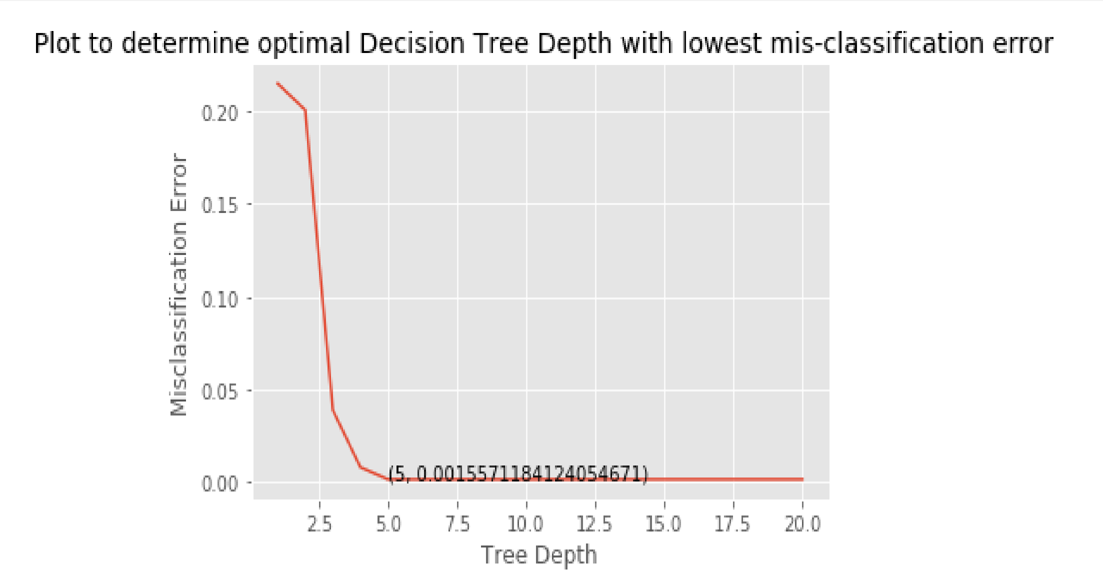
Predictive Analysis - Model

- Various settings/parameters that have been applied in the construction of model

```
clf_gini = DecisionTreeClassifier(criterion='gini', splitter='best',
max_depth=4, random_state=0)

clf_gini = DecisionTreeClassifier(criterion='gini', splitter='best',
max_depth=5, random_state=0)
```





Since tree depth=5 produces the lowest misclassification error, it will be used as the parameter setting for DecisionTree model

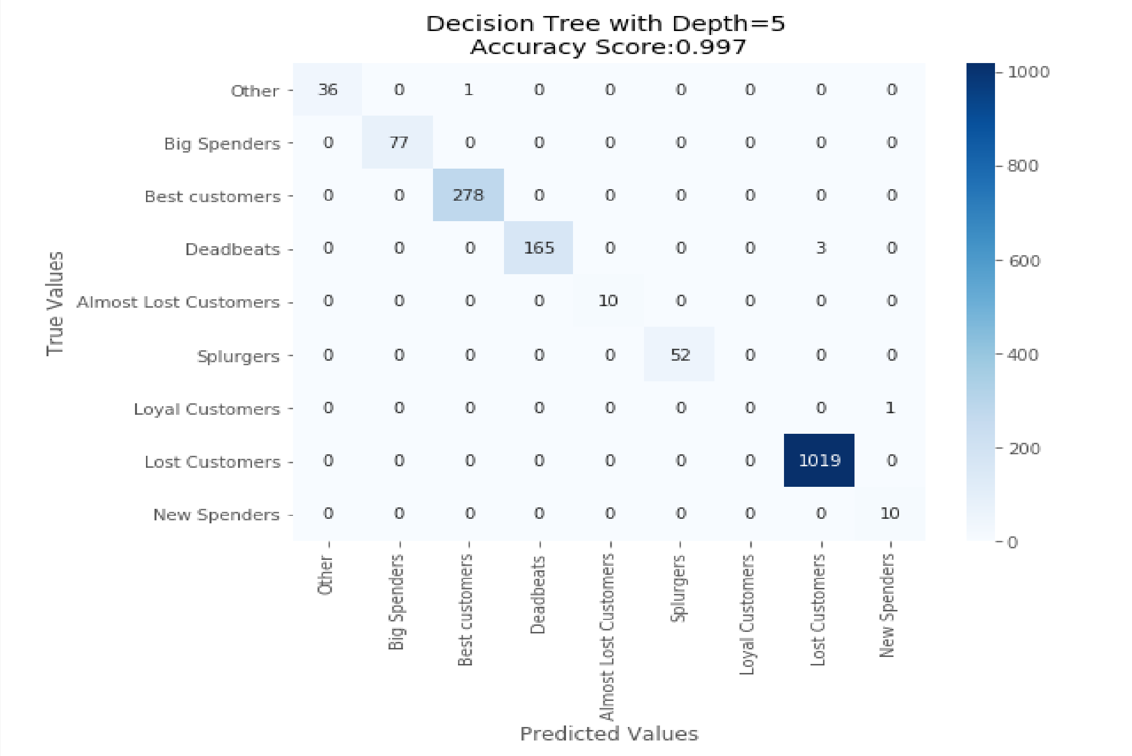
With stratified sampling & when tree depth=5, we can observe there is pure leaf or node with no impurity.

MODEL EVALUATION

Classification Report

	precision	recall	f1-score	support
Almost Lost Customers	1.00	1.00	1.00	37
Best customers	1.00	0.99	0.99	77
Big Spenders	1.00	1.00	1.00	278
Deadbeats	1.00	1.00	1.00	168
Lost Customers	1.00	1.00	1.00	10
Loyal Customers	1.00	1.00	1.00	52
New Spenders	0.00	0.00	0.00	1
Other	1.00	1.00	1.00	1019
Splurgers	0.91	1.00	0.95	10
accuracy			1.00	1652
macro avg	0.88	0.89	0.88	1652
weighted avg	1.00	1.00	1.00	1652

Confusion Matrix of DecisionTree with Depth=5



From this confusion matrix, it shows misclassification error exist even when Tree Depth=5 as being the optimal depth determined earlier

MODEL EVALUATION

```
cross_val_score(model, x_train, segment_type_train, scoring='accuracy', cv=stratified_cv, n_jobs=-1, error_score='raise')
```

	RandomForest	DecisionTree	AdaBoost
Hyper-parameter Settings	max_depth=2, criterion='gini', max_features='auto', random_state=0	criterion='gini', splitter='best', max_depth=5, random_state=0	base_estimator=None, n_estimators=6, learning_rate=1.0, algorithm='SAMME', random_state=None
Scoring Metric = 'Accuracy'	0.786	**0.997**	0.786

- Python cross_val_score Scoring Metric =‘Accuracy’ was used to measure against these 3 supervised learning model.

- Based on highest accuracy, DecisionTree is the preferred choice for predictive model

Lesson Learnt

- If Dataset is large, Regression Tree (CART) requires pruning. Avoid choosing large tree depth to minimize model suffering from over-fitting. Limiting the depth decreasing over-fitting. This leads to decrease accuracy on training set but improvement on the test set.

- The number of hyper-parameters to be tuned is almost very limited, either tree-depth or criterion= gini or entropy. Using ‘gini’ index can have some advantages when dealing with highly skewed data where a large proportion of samples belongs to one class (ie: class imbalance)

- Among the 3 Supervised Classification algorithm used during experiment, DecisionTree produced the best accuracy

Citation

1. Richard Farrow, William Trevino, Vitaly Briker, and Brent Allen, "Identifying Customer Churn-in After-market Operations using Machine Learning Algorithms", Vol. 2 Issue 3