Presented By: Amit Kukreja

Updated: 28-June-2022

Insurance Purchase Prediction

for

AllState Car Insurance online customers

Table of Contents

1.0 Introduction

As customers shop for a car insurance policy, they visit websites of multiple insurance companies to generate a number of quotes with different coverage options before purchasing a plan. Once the customer arrives at choosing their quote, they may choose many different options before arriving at their ultimate purchase option. Each revision that’s required, increases the probability of losing the customer. Allstate Insurance would like to develop a **predictive model** that can predict the coverage plan that the customer would eventually buy. If the eventual purchase can be predicted sooner in the shopping window, the quoting process is shortened and the company is less likely to lose the customer's business.

1.1 Problem Statement

How can Allstate Insurance improve **conversion rates** for prospective customers who visit its website and generate multiple quotes for car insurance.

1.2 Criteria for success

A high prediction accuracy for the seven coverage options of a policy. Allstate has the quote histories and some personal data of customers who eventually purchased a plan from it. The task is to predict the purchased coverage options using a limited subset of the total interaction history.

1.3 Scope of solution space

The predictive model shall be based on data available for approx. 97,000 customers who eventually purchased a car insurance policy from Allstate. This data primarily includes customer’s information and features of the coverage options for which the customer generated quotes on Allstate’s website.

1.4 Constraints within solution space

The following factors would limit the predictive accuracy of the model:

1) Non-availability of data on external factors such as competitive products & pricing, market shares of various insurers in a region/geography.
2) Customer data availability on limited parameters. Important parameters that are not available include household income, car model and driving history.

1.5 Data Sources

Two datasets have been provided by Allstate:

1) Training dataset: Entire purchase history of approx. 97,000 customers who finally bought a car insurance policy from Allstate. The data includes parameters such as:
    a) Customer data: State, Pin code, No. of persons driving car, home ownership, Marital status, age of oldest and youngest driver.
    b) Customer's car data: Age, value and risk factor
    c) Purchase History: Date, Coverage features (anonymized), cost
2) Test dataset: Partial purchase history of approx. 56,000 customers.

As the final vector choices are not available for the test dataset, the training dataset of 97,009 customers would be split into a training set and a test set.

2.0 Data Wrangling

The objectives of this stage are as follows:

1) Exploring and understanding the data

   a) What do various fields represent ?

   b) Which variables are numerical, categorical, boolean etc.

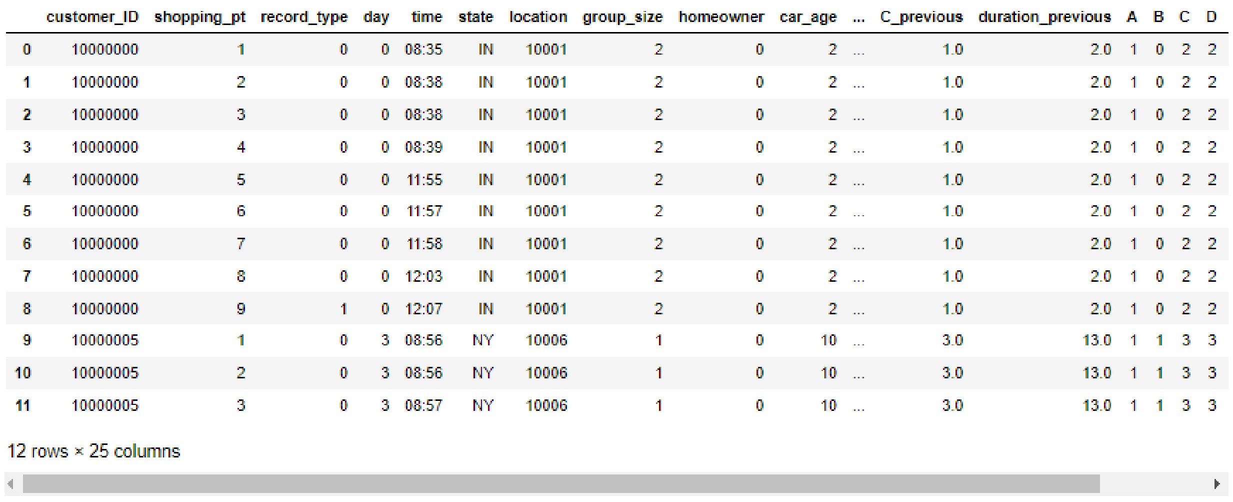
   c) Looking at summary statistics

2) Cleaning the data and imputing missing values where possible

3) Organizing the data and making it ready for the EDA stage

2.1 Exploring the data and profiling the variables

The training dataset has **665249 observations** across **25 features**. There are **97,009 unique customer_ID's.** Each observation is a quote taken by a customer, where the customer can change his/her information, product vector (A-G) choices and will get a resultant policy cost quote. As such, the dataset is organized in the **long form**, as shown below:



	customer_ID	shopping_pt	record_type	day	time	state	location	group_size	homeowner	car_age	...	C_previous	duration_previous	A	B	C	D
0	10000000	1	0	0	08:35	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
1	10000000	2	0	0	08:38	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
2	10000000	3	0	0	08:38	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
3	10000000	4	0	0	08:39	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
4	10000000	5	0	0	11:55	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
5	10000000	6	0	0	11:57	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
6	10000000	7	0	0	11:58	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
7	10000000	8	0	0	12:03	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
8	10000000	9	1	0	12:07	IN	10001	2	0	2	...	1.0	2.0	1	0	2	2
9	10000005	1	0	3	08:56	NY	10006	1	0	10	...	3.0	13.0	1	1	3	3
10	10000005	2	0	3	08:56	NY	10006	1	0	10	...	3.0	13.0	1	1	3	3
11	10000005	3	0	3	08:57	NY	10006	1	0	10	...	3.0	13.0	1	1	3	3

12 rows × 25 columns

customer_ID 10000000 took **9 quotes** before buying the policy, each quote contained in a separate row.

The variables are as follows:

1) customer_ID: The ID is an eight digit number with 97009 distinct values, one for each customer. We shall treat it as the **index**.

2) shopping_pt : The column represents the quote number the customer is taking. It has values ranging from 1 to 13. In chart 2.1, the distribution of purchase shopping_pt(the shopping_pt at which customer finally buys the policy) shows that the min. value is 3, so it means customers take at least 3 quotes before buying a policy. We shall consider this variable as **numeric** as it has a 'distance' property, representing how far a customer travels in his/her shopping journey. **More than 50%** of the customers take between 6 to 8 quotes before buying the policy, with the average no. of quotes being **6.86.** The distribution has some resemblance to a **normal distribution.**

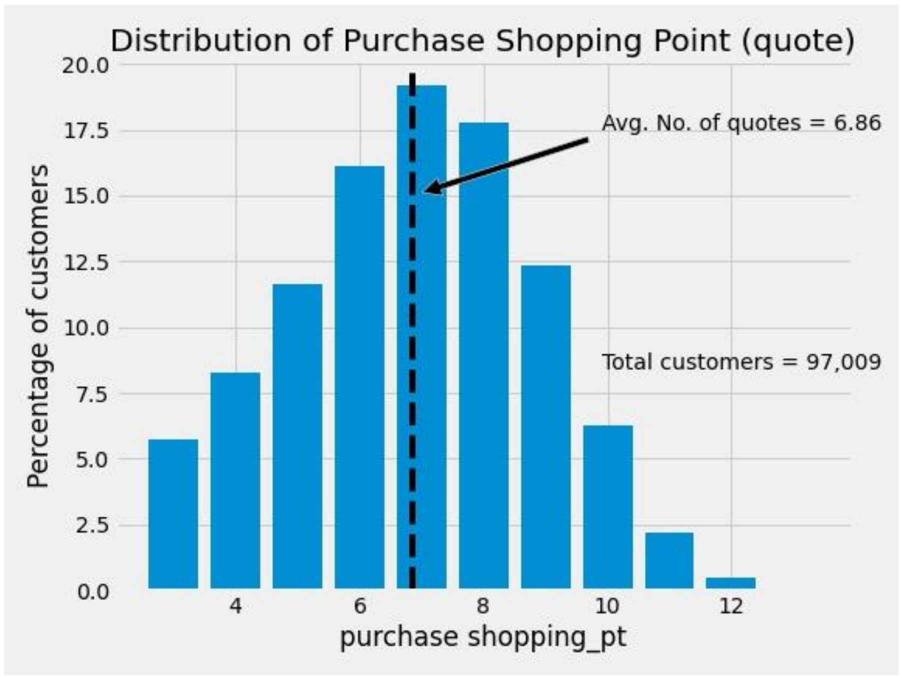


Chart 2.1

3) record_type: Signals the difference between a 'shopping' quote and the 'final purchase' quote. **0**=shopping point, **1**=purchase point. Shall treat it as **boolean**.

4) day: The day values (0-6) are used to represent Monday-Sunday, so are not real numbers. They have **7** distinct values, so we will treat them as **categorical**.

5) time: Time of day (HH:MM). **1204** distinct values, pretty close to 1440, the total minutes in a day! We shall treat this as **datetime**.

6) state: State where a shopping point occurred. There are **36** unique values. This is a **categorical** variable.

7) location: These are **6248** masked zip codes where a shopping point occurred, so we will treat them as **categorical**.

8) group_size: This represents the total persons for whom the policy is being taken. It can only take distinct values and ranges from 1 to 4. We shall treat it as **categorical**.

9) homeowner: Indicates if a person owns a home or not. We shall treat it as **categorical**.

10) car_age: This represents how old the car is and is clearly **numerical**.

11) car_value: This is supposed to be the price of the car when newly bought, but has values 'a' to 'g', so it has been categorized into 7 classes. We shall treat it as **categorical**.

12) risk_factor: Each customer is categorized into one of 4 types, from 1 to 4. As of now, no way of knowing if 4 is riskier than 1. So keep it **categorical** for now.

13) age_oldest: Age of the oldest member in the group. It is **numerical**.

14) age_youngest: Age of the youngest member in the group. This too is **numerica**l.

15) married_couple: Has values 0 and 1, we shall treat it as **categorical**.

16) C_previous: What the customer formerly had or currently has for product option C (0=nothing, 1, 2, 3,4). So **categorical** feature.

17) duration_previous: The no. of years previous policy was in force, so **numerical**.

18) cost: Cost of the quoted policy, is a continuous number, so **numerical**.

19) to 25) Product Vectors: These are the 7 product features from **A to G**. They are the **target variables**, but take distinct values. So we will treat them as categorical.

In summary, there are:

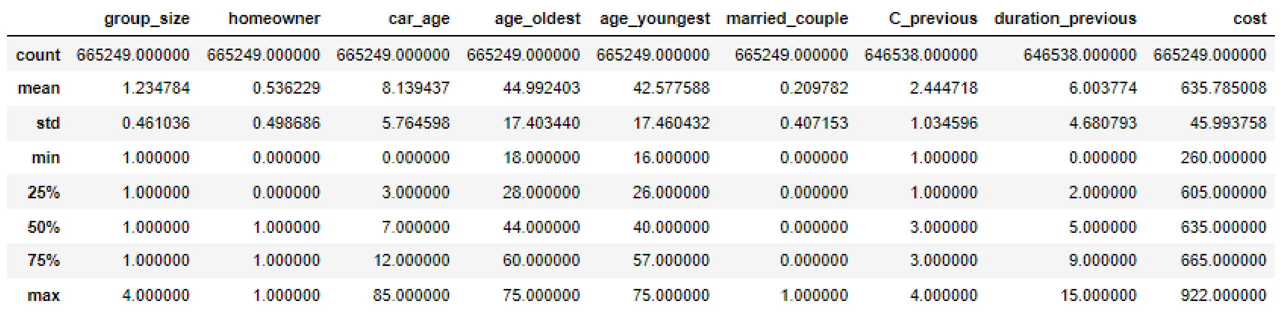
6 numerical variables: shopping_pt, car_age, age_oldest and age_youngest, duration_previous & cost.

1 datetime variable: time

10 categorical variables: day, state, location, group_size, car_value, risk_factor, C_previous, record_type, homeowner, married_couple

7 Response Variables or Target : The All State problem is about predicting these final product vectors the consumers would end up choosing. All the 7 components of the product vector (A-G) take discrete values. As these 7 targets take discrete values, **this is a classification problem**.

Now let's look at some summary statistics of the numerical variables.



	group_size	homeowner	car_age	age_oldest	age_youngest	married_couple	C_previous	duration_previous	cost
count	665249.000000	665249.000000	665249.000000	665249.000000	665249.000000	665249.000000	646538.000000	646538.000000	665249.000000
mean	1.234784	0.536229	8.139437	44.992403	42.577588	0.209782	2.444718	6.003774	635.785008
std	0.461036	0.498686	5.764598	17.403440	17.460432	0.407153	1.034596	4.680793	45.993758
min	1.000000	0.000000	0.000000	18.000000	16.000000	0.000000	1.000000	0.000000	260.000000
25%	1.000000	0.000000	3.000000	28.000000	26.000000	0.000000	1.000000	2.000000	605.000000
50%	1.000000	1.000000	7.000000	44.000000	40.000000	0.000000	3.000000	5.000000	635.000000
75%	1.000000	1.000000	12.000000	60.000000	57.000000	0.000000	3.000000	9.000000	665.000000
max	4.000000	1.000000	85.000000	75.000000	75.000000	1.000000	4.000000	15.000000	922.000000

- Average car_age is 8 yrs, ranges from 0 to a staggering 85 years!
- age_oldest: ranges from 18 to 75, with a mean of ~45 yrs.
- age_youngest: ranges from 16 to 75, with a mean of ~43 yrs.
- duration_previous: ranges from 0 to 15, with a mean of 6 yrs.
- cost: ranges from $260 to $922, with a mean of $636. As seen in Chart 2.2, the distribution looks very much like a ***gaussian distribution***.

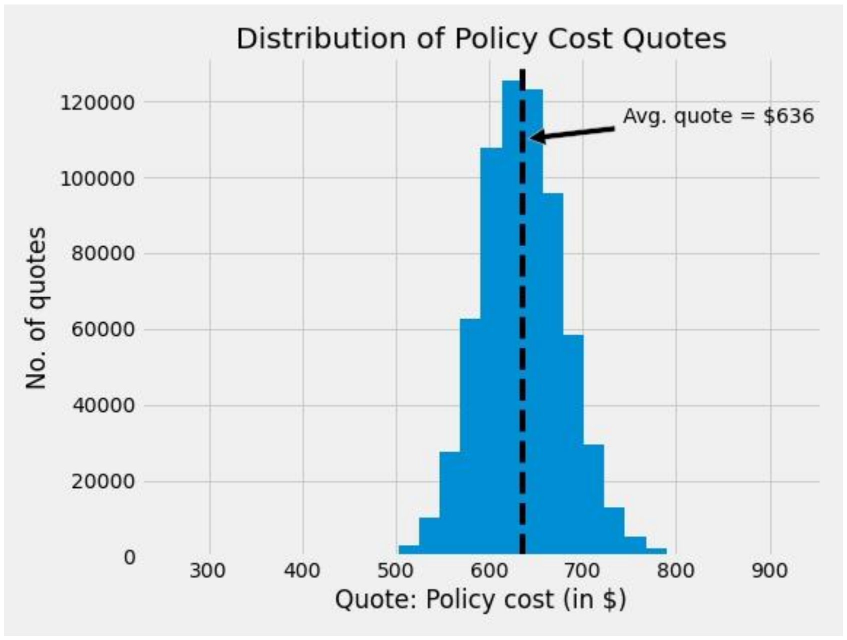


Chart 2.2

2.2 Missing Values & Imputing them

There are missing value only in four of the 25 features, these are:

1) car_value: 1531 missing values i.e. 0.2%
2) risk_factor: 240,418 missing values, that's **36.1%**!
3) C_previous: 18711 missing values at 2.8%.
4) duration_previous: 18711 missing values at 2.8%.

Out of these, there are 3 fields (1,3,4 above) that are customer provided. The percentage missing values are relatively low. As we know that customers take multiple quotes, it is possible that some customers might have provided this information at some point in their purchase process. So for e.g, if car_value is present for one quote, it would be filled using that for other quotes of the customer. After doing this process, the missing value left are:

1) car_value: 0 missing values. All missing values found at some stage of quoting.
2) risk_factor: 230,643 missing values. That's still **34.7%** missing values.
3) C_previous: 5079 missing values, only 0.76%
4) duration_previous: 5079 missing values, only 0.76%

Missing risk_factor values

The risk_factor is not a customer provided information, it is an independent assessment by Allstate. Some models were generated to predict risk_factor based on customer provided information, but these were only **30% accurate**. As almost 35% of risk_factor values are missing, they were **imputed with a new value 0** - implying missing/others.

Missing values for C_previous were imputed with the **most frequent** value of 3 that occurs 42% of the time.

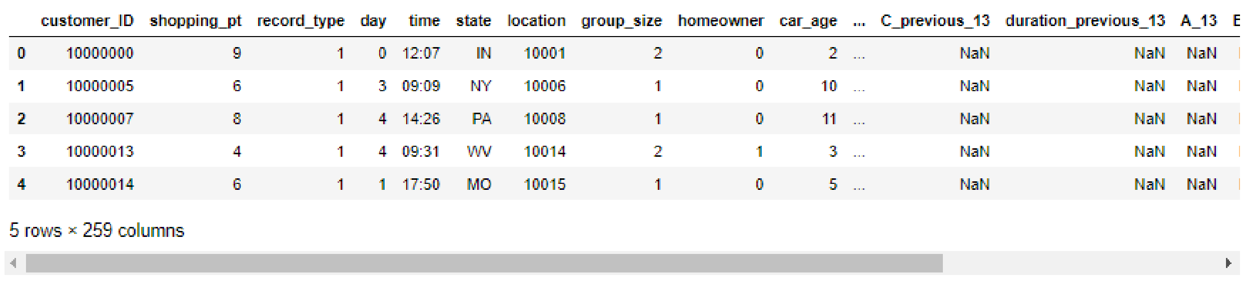
Missing values for duration_previous were imputed with the **average** value of 5.97.

2.3 Organizing the data

We saw that the data is organized in the **long form**, with a customer's purchase history spread across multiple rows, each row representing a quote taken(shopping_pt). As we need to predict the final product vectors the customer is going to choose, each customer's purchase history needs to be contained in a single row, so that the customer's entire purchase history becomes different features for a model.

As such, we shall organize the data into a **wide form**. Also, we split the training dataset itself into training and test sets. This is since we do not know the final targets for the test set given by All State, so we need to carve out a test set from the training set itself.

The dataset was converted into a wide form with **97,009 observations** across **259 features**, with each observation referring to a customer. Each row represents the entire purchase history of a single customer.



	customer_ID	shopping_pt	record_type	day	time	state	location	group_size	homeowner	car_age	...	C_previous_13	duration_previous_13	A_13	E
0	10000000	9	1	0	12:07	IN	10001	2	0	2	...	NaN	NaN	NaN	
1	10000005	6	1	3	09:09	NY	10006	1	0	10	...	NaN	NaN	NaN	
2	10000007	8	1	4	14:26	PA	10008	1	0	11	...	NaN	NaN	NaN	
3	10000013	4	1	4	09:31	WV	10014	2	1	3	...	NaN	NaN	NaN	
4	10000014	6	1	1	17:50	MO	10015	1	0	5	...	NaN	NaN	NaN	

5 rows × 259 columns

We take out **20%** of the observations for the test set, stratifying them based on shopping_pt, so that both training and test sets have similar distribution of shopping_pts.

Finally, the **training set has 77,607 observations** and the **test set has 19,402 observations**. We store these datasets in separate csv files.

3.0 Exploratory Data Analysis

The objectives of this stage are as follows:

1) Look at characteristics of various variables
    a) Understand the correlations between customer data features (e.g. state, day, car_age, homeowner etc.) and product vectors. Based on this, decide which features should be retained for model development.
2) Understand the dynamics of product vectors:
    a) How are product vector choices distributed, at the start of a shopping journey and at the final purchase.
    b) Understand how and when customers change product vectors, specifically
        i) Distribution of changes by quote
        ii) Distribution of changes by product vector
3) For people with a long shopping history (e.g. more than 10 codes), the Cramer's V correlation between each code and the final code.
4) Modelling Approach : Given varying levels of shopping history for different customers, think through how much history should be used for model development. There are different approaches possible:
    a) **Multiple models** developed for varying levels of customer history e.g. A model for 2 shopping quotes, another one for 3 shopping quotes and so on.
    b) **A generic model** that perhaps considers the last 2 quotes i.e. quote n and quote n-1 or the first 2 quotes.

Generic models would have more observations to learn from, as for all customers 2 quotes are available. On the other hand, specific models would have the ability to analyze more information for each customer e.g. all possible values for a certain product vector that the customer has experimented with. This may help improve the final prediction. At this stage, let's keep all options open and find an optimal approach during the EDA phase.

3.1.1 state

There are customers from 36 states with the **top 5** accounting for **50%** of the customers.

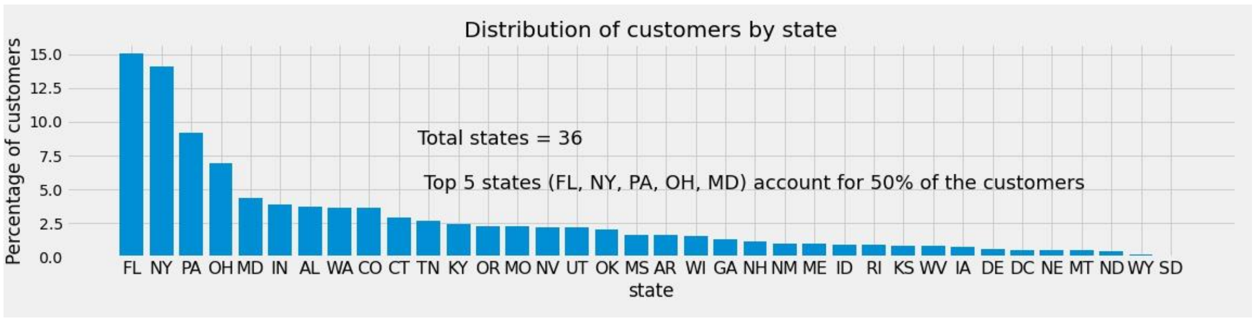


Chart 3.1

To test the independence of state with product vectors, we use the **chi2 test of independence** and check the p_values for a few vectors.

```
from scipy.stats import chi2_contingency
vectors = ['A', 'B', 'G']
for vector in vectors:
    stat, p, dof, expected = chi2_contingency(pd.crosstab(df_train['state'], df_train[vector]))
    print(f"The p_value for chi_squared test between state & {vector} vector = {p}")
```

```
The p_value for chi_squared test between state & A vector = 0.0
The p_value for chi_squared test between state & B vector = 0.0
The p_value for chi_squared test between state & G vector = 0.0
```

The **p_value** of 0.0 for all 3 vectors signifies that state and product vector selection are not independent. As such, we shall retain state as a feature.

3.1.2 location (zip codes)

There are 6248 locations. This is a large number. Given that location would have to be treated as a categorical variable and one-hot coded for classification models, this feature would add a lot of complexity. Even if we were to look at only zip codes that deviated significantly from the mean, given the large number of zip codes we might find variation solely due to randomness. Let’s check how customers are distributed across zip codes and whether it is worth retaining this feature or not. Chart 3.2 shows that **90% of the zip codes have less than 30 customers**. The largest had just over 100. It seems unlikely that the model would learn much of use from such a large no. of zip codes.. As such, we shall not consider location in modeling.

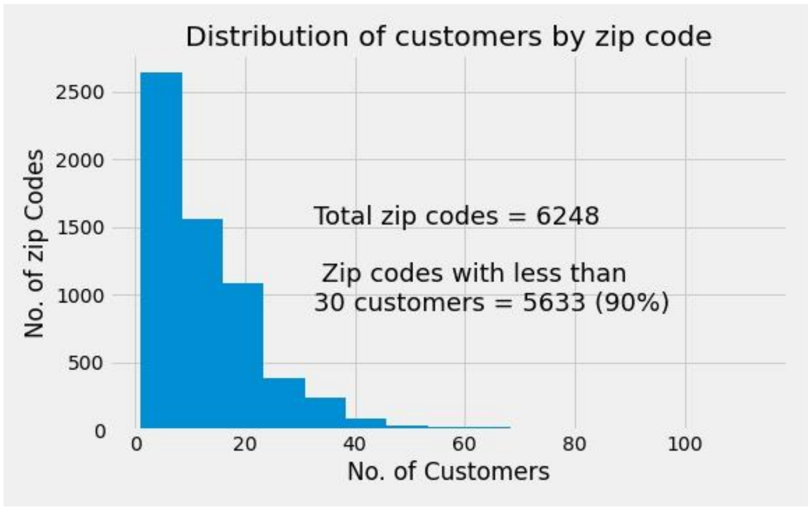
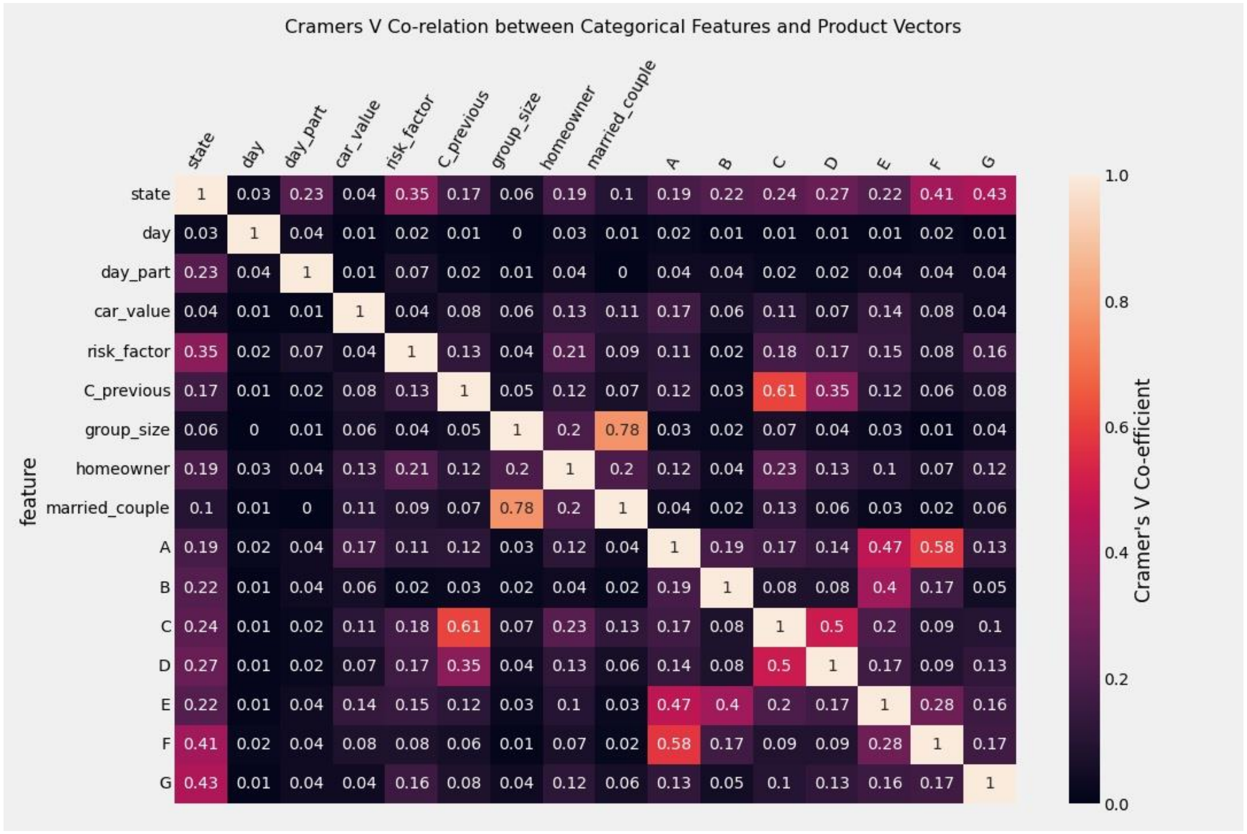


Chart 3.2

3.1.3 Other categorical variables

To check how strongly other categorical variables are connected with the final product vectors, we shall use **Cramer’s V correlation** that is based on the chi2 contingency test of independence.

But before doing that, we need to decide what to do with ‘time’. This has 1204 unique values, almost one for every minute of the day! To simplify, we shall convert this into ‘day_part’ - morning, day, evening and night and then use the ‘day_part’ variable for **Cramer’s V correlation.** Chart 3.3 shows the heatmap of the Cramer’s V correlation with all categorical variables.



	state	day	day_part	car_value	risk_factor	C_previous	group_size	homeowner	married_couple	A	B	C	D	E	F	G
state	1	0.03	0.23	0.04	0.35	0.17	0.06	0.19	0.1	0.19	0.22	0.24	0.27	0.22	0.41	0.43
day	0.03	1	0.04	0.01	0.02	0.01	0	0.03	0.01	0.02	0.01	0.01	0.01	0.01	0.02	0.01
day_part	0.23	0.04	1	0.01	0.07	0.02	0.01	0.04	0	0.04	0.04	0.02	0.02	0.04	0.04	0.04
car_value	0.04	0.01	0.01	1	0.04	0.08	0.06	0.13	0.11	0.17	0.06	0.11	0.07	0.14	0.08	0.04
risk_factor	0.35	0.02	0.07	0.04	1	0.13	0.04	0.21	0.09	0.11	0.02	0.18	0.17	0.15	0.08	0.16
C_previous	0.17	0.01	0.02	0.08	0.13	1	0.05	0.12	0.07	0.12	0.03	0.61	0.35	0.12	0.06	0.08
group_size	0.06	0	0.01	0.06	0.04	0.05	1	0.2	0.78	0.03	0.02	0.07	0.04	0.03	0.01	0.04
homeowner	0.19	0.03	0.04	0.13	0.21	0.12	0.2	1	0.2	0.12	0.04	0.23	0.13	0.1	0.07	0.12
married_couple	0.1	0.01	0	0.11	0.09	0.07	0.78	0.2	1	0.04	0.02	0.13	0.06	0.03	0.02	0.06
A	0.19	0.02	0.04	0.17	0.11	0.12	0.03	0.12	0.04	1	0.19	0.17	0.14	0.47	0.58	0.13
B	0.22	0.01	0.04	0.06	0.02	0.03	0.02	0.04	0.02	0.19	1	0.08	0.08	0.4	0.17	0.05
C	0.24	0.01	0.02	0.11	0.18	0.61	0.07	0.23	0.13	0.17	0.08	1	0.5	0.2	0.09	0.1
D	0.27	0.01	0.02	0.07	0.17	0.35	0.04	0.13	0.06	0.14	0.08	0.5	1	0.17	0.09	0.13
E	0.22	0.01	0.04	0.14	0.15	0.12	0.03	0.1	0.03	0.47	0.4	0.2	0.17	1	0.28	0.16
F	0.41	0.02	0.04	0.08	0.08	0.06	0.01	0.07	0.02	0.58	0.17	0.09	0.09	0.28	1	0.17
G	0.43	0.01	0.04	0.04	0.16	0.08	0.04	0.12	0.06	0.13	0.05	0.1	0.13	0.16	0.17	1

Chart 3.3

The above heatmap based on Cramer's V correlation between different categorical features and product vectors shows some noticeable correlations between:

a) state and vectors F and G
b) C_previous and Vectors C and D
c) Vector A with Vectors E and F
d) Vectors C and D

e) Vectors B and E

We can also notice that certain categorical variables have extremely low correlations with product vectors. These are day, day_part and group_size. These variables may be considered to be dropped to avoid adding unnecessary complexity in the modeling.

3.1.4 Categorical variables - conclusion

After analysing all the categorical and day/time variables, the following variables would be kept in the model:

- State, car_value, C_previous and risk_factor

5 features that are being dropped from modeling stage are:

- Location, Day, Time, day_part and group_size

3.2 Numerical Features

There are 6 numerical variables identified during Data wrangling:

- shopping_pt, group_size, car_age, age_oldest, age_youngest, duration_previous & cost

We have already seen some summary statistics of these variables in the data wrangling stage. Let's check the correlations between these variables and the 7 product vectors.

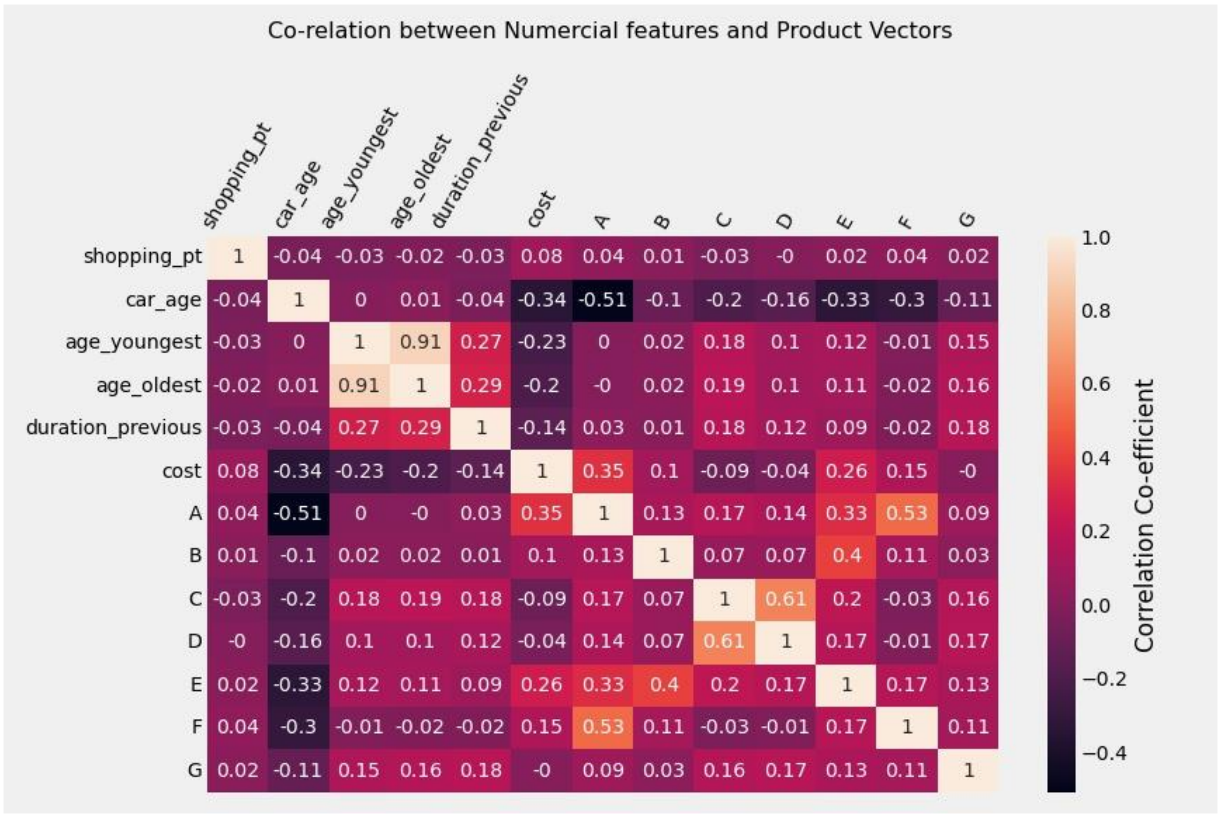


Chart 3.4

From the heatmap in Chart 3.4, some key observations are:

a) shopping_pt has very negligible correlation with all 7 product vectors
b) car_age has a noticeable -ve correlation with vectors A, E and F
c) cost has a noticeable +ve correlation with vectors A and E
d) Vector A has a noticeable +ve correlation with Vectors E and F

e) Vectors C has a noticeable +ve correlation with Vector D
f) Vectors B has a noticeable +ve correlation with Vector E

Based on the above analysis, we can drop shopping_pt from the modeling stage.

3.3 Product Vectors (Target variables)

3.3.1 Vector A

Chart 3.5 shows the choice of vector A at the start, 2nd quote and final quote.

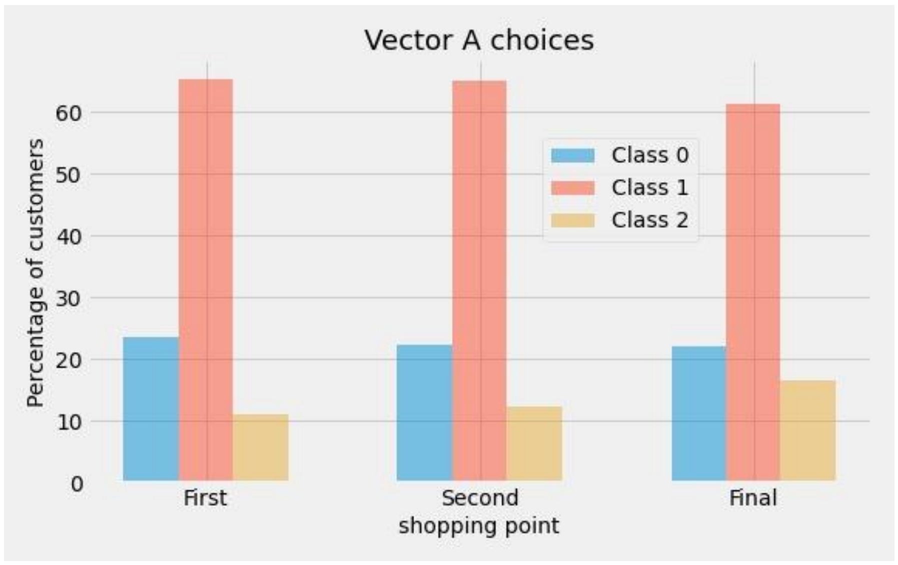


Chart 3.5

Clearly, option 1 is the most preferred choice for this vector with more than 60% consumers opting for it, followed by option 0 at about ~21%. However, we see option 1 and 0 see small declines in preference from quote 1 to the final purchase quote, while option 2 sees an increase.

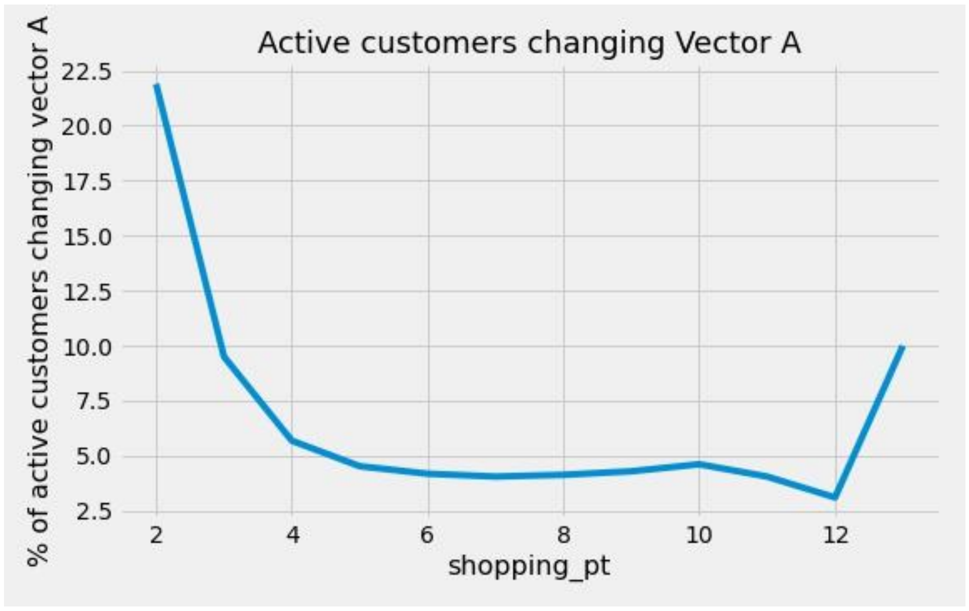


Chart 3.6

Chart 3.6 shows the proportion of active customers who make changes to vector A at different shopping points. **Active customers** are those who are still shopping and haven't made a purchase yet. **Changes in vector A happen early on**, with ~22% customers changing it at shopping_pt 2 and this drops sharply after that. From shopping_pt 5 onwards, less than 5% of active customers change vector A.

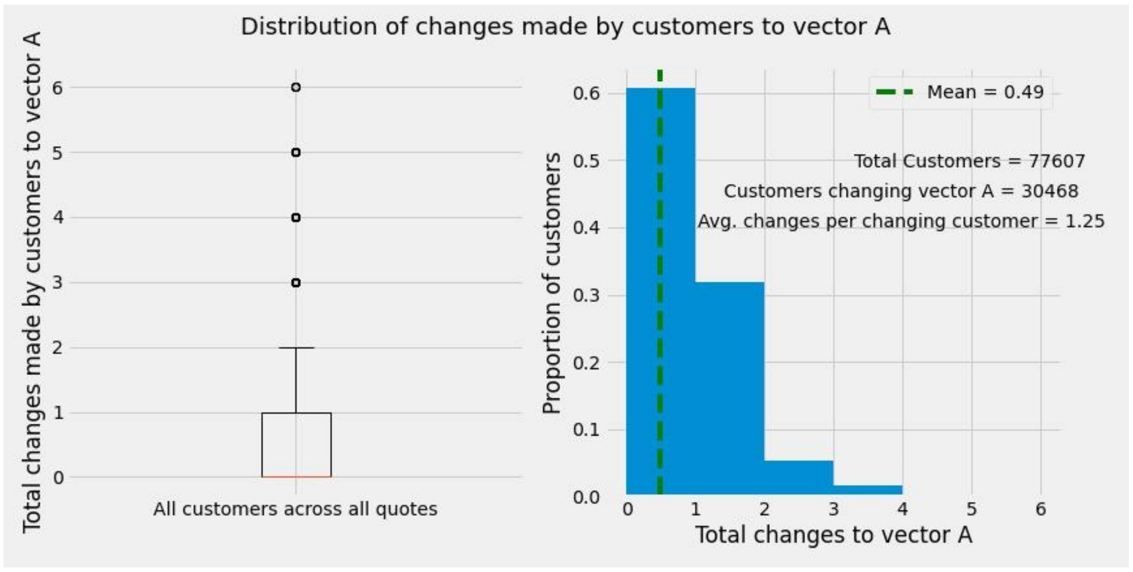


Chart 3.7

Chart 3.7 shows the distribution of changes to vector A. **Only about 40%** of total customers make changes to vector A during their shopping journey. Amongst customers who change vector A, the **vast majority (~81%) change it only once** and the average no. of changes for customers who do change A is 1.25.

3.3.2 Customers making changes

The chart on changes in Vector A shows that **a very small proportion** of active customers change it **after shopping_pt 3**. This raises some important questions - what proportion of active customers are making changes? What's the distribution of changes across shopping_pts? What's the distribution of total changes across the entire shopping journey? Let's answer these with some visualizations.

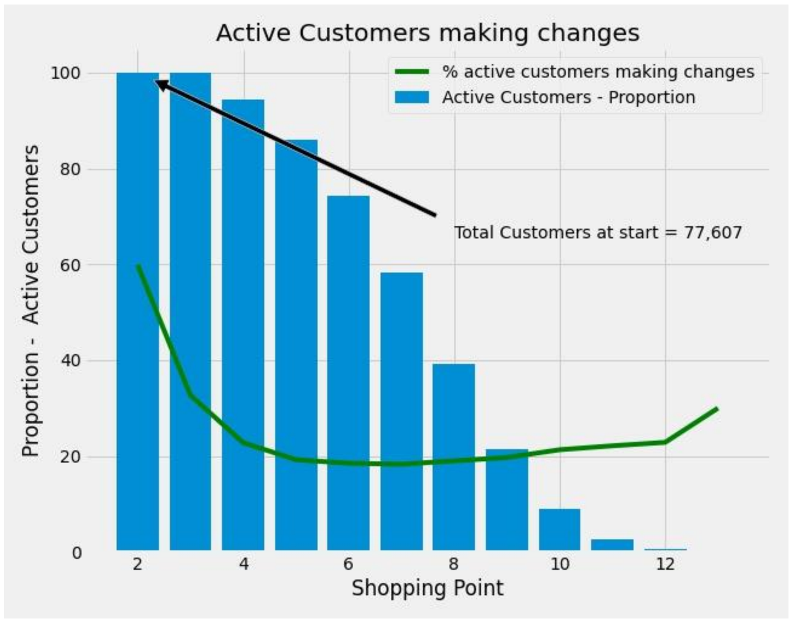


Chart 3.8

Chart 3.8 shows the proportion of active customers making changes at various shopping points. The trend is very similar to vector A. At shopping point 2, **almost 60% of the customers** make some changes and this declines rapidly after. After shopping pt 4, **only about 20%** of the active customers make changes. So 80% of the active customers are not making any changes!

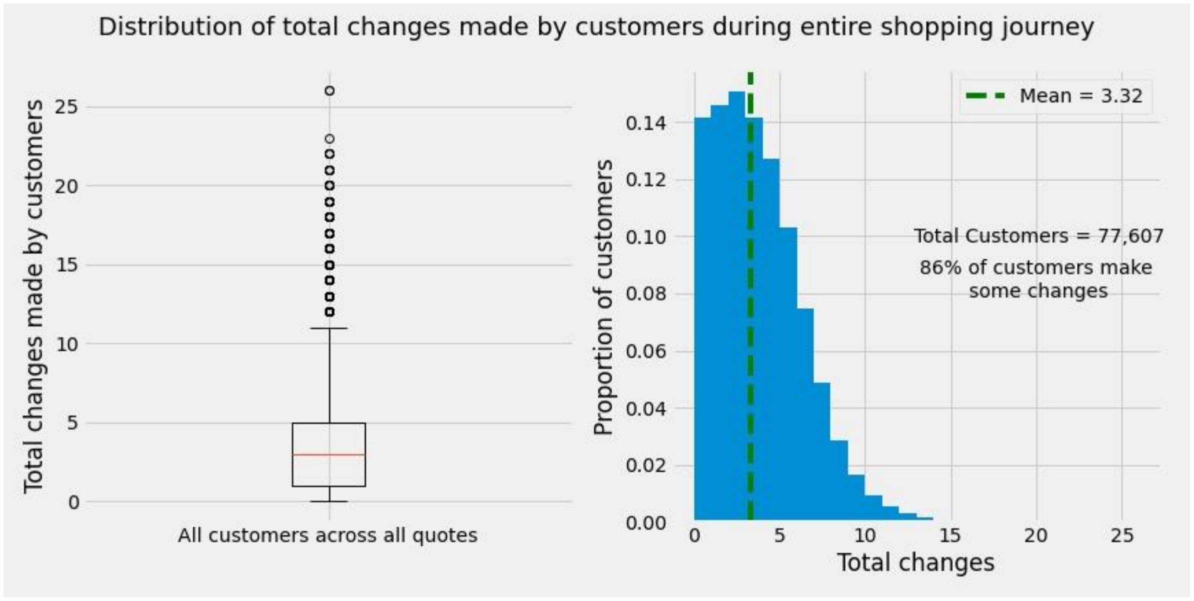


Chart 3.9

Chart 3.9 shows the distribution of total changes made by the customer during the entire shopping journey. As seen in the histogram, about 14% of the customers make no changes whatsoever! And the distribution is right-tailed, with no. of customers who make higher changes continuously dropping after 2 changes. The **mean** changes per customer is **3.32**. The boxplot shows that the middle 50% customers make **between 1 to 5 changes**, with the median being 3. Another interesting observation is the **large no. of outliers** that push up the mean. There are many customers who make more than 10 changes while they are shopping, the highest no. of changes by a customer being 26!

This raises another question - do consumers with longer shopping journeys make more changes i.e. say a customer who purchased at shopping point 8 is more likely to have made higher changes to the vectors? As chart 3.10 shows, the average no. of changes does increase gradually. Customers who purchase at shopping point 3 make approx. 2.5 changes on average, while customers who purchase at shopping point 8 make 3.5 changes. Customers who purchase at shopping point 13 make 5 changes on average.

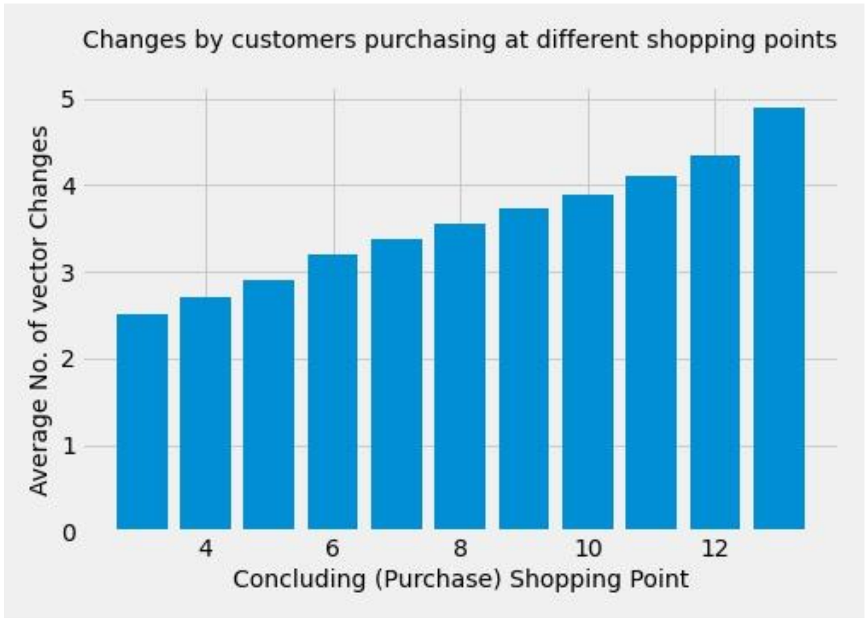


Chart 3.10

The change process in all other vectors is very similar to vector A. A large no. of changes are made early on on shopping_pts 2 and 3 and after that the rate of change slows down.

Key Learnings

- Most customers make changes early on, at shopping points 2 & 3.
- The average changes made by a customer are around 3.
- There are many outlier customers who make lots of changes, some as many at 20+!
- After shopping_pt 3, only 18- 20% of active customers make changes at a given shopping_pt.

3.4 Modeling Considerations

Based on the analysis above for product vectors and the pattern of changes by customers, it would make sense to create a model based on the early shopping journey of the customer. **60% of the customers make some changes at shopping_pt no.2**. So a model could be based on customer and vector information at shopping_pt 1 and 2 to predict the final vector choices. Such a model would be able to learn the pattern of changes made by 60% of the customers at shopping_pt 2 and linking this information to final vector choices would allow the model to learn how customers make further changes.

4.0 Modelling

The key objective of the model building are as follows:

1) Predict the 7 product vector's final value based on shopping quotes 1 and 2
2) Test and evaluate the performance of different classifiers on validation set.
3) Tune hyperparameters to optimize model performance.
4) Finally, apply the models to the test set to predict the vectors.

4.1 Nomenclature and metrics

Before diving into modeling and performance testing, a quick word on some nomenclature and metrics

4.1.1 Some Nomenclature

'baseline_model' : This refers to predicting the final vector choice based on the customer's selection of the vector at the most recent shopping pt, say 2.

'A': Customer's final choice for vector A i.e. the target we are predicting for vector A.

'A_2': Customer's choice for vector A at shopping pt 2.

'A_2_2': One-hot encoded feature for vector A, at shopping pt 2, for class 2.

4.1.2 Metrics for model evaluation:

The no. of classes for vectors range from 2 to 4. The key objective we have is to predict each customer's correct class for each of the 7 vectors. As such, the key metrics for model evaluation would be **'micro accuracy'** i.e. the overall correct class predictions across all customers.

4.2 Model building approach

The key steps followed to build a model for each vector were:

1) Training a basic model based on 3 different classifiers - Random Forest, Gradient Boost and XG Boost. For binary vectors (B and E), a Logistic Regression classifier was also evaluated.
2) Applying 60 iterations of Randomized Search CV to determine performance of different hyperparameter combinations on the training dataset.
3) Applying each of the 60 sets of hyperparameters on the validation set and measuring micro accuracy.
4) Choose the best set of hyperparameters as the one that performs most consistently across the training and validation set.
5) Based on these hyperparameters, compare the performance of different classifiers on the test set to then choose the best classifier for each vector.

4.3 Building a model for Vector A

We start by understanding the linkage between customer choice for vector A at shopping_pt 2 and their final choice for vector A i.e. a **'baseline_model'** classification of predicting vector A based on its status at shopping_pt 2. The confusion matrix where A's final vector is predicted using **'A_2' (A's choice at shopping pt 2)** shows that **recall** based on simply predicting vector A = A_2 is **81.91%**. A **baseline_model** works best for class 1, where **90.46%** customers retain their choice at shopping_pt 2. However, a lot of those with class 1 at shopping_pt 2 also move to class 0 or 2. The baseline_model doesn't work as well for class 0 and particularly poorly for class 2.

A_2	0	1	2	total	A0_%	A1_%	A2_%
A							
0	3264	856	148	4268	76.48	20.06	3.47
1	769	10785	369	11923	6.45	90.46	3.09
2	258	1110	1843	3211	8.03	34.57	57.40

4.3.1 Vector A: Models and performance

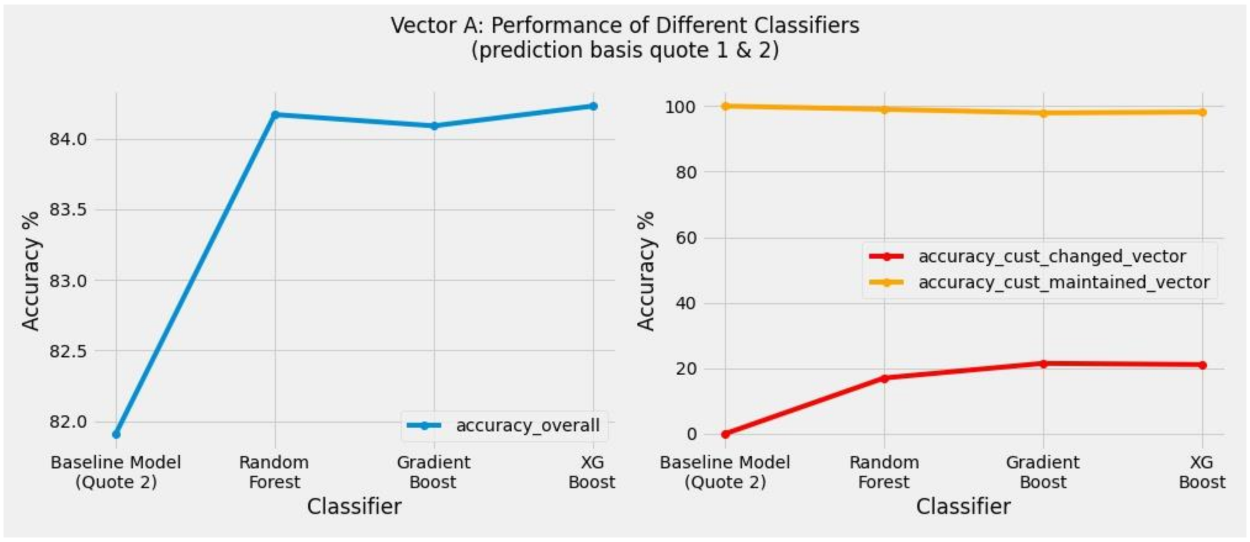


Chart 4.1

In Chart 4.1, the blue line shows that the 3 classifiers had a **2.2 to 2.3% improvement** over 'baseline_model' performance, with XGBoost having a slight edge over the other two classifiers. The yellow line in chart 4.1 shows that all 3 models were **very close to 100%** in predicting the **customers that wouldn't change** their vector choice from shopping_pt 2 to the final purchase. As shown by the red line, the classifiers did reasonably well in predicting **customers who would change** their vector choice, with XG Boost predicting almost **21%** of such customers correctly.

Next we look at the **confusion matrix** for the best classifier. We see that there is an improvement

```
XGBoost Classifier results for Vector A
Parameters: No. of Estimators = 300,  max_depth = 3,     learning_rate = 0.3, colsample_bytree = 0.5
Accuracy=0.8423, f1-score=0.837
Confusion Matrix - normalized
 [[0.799  0.1717 0.0293]
 [0.0396 0.9288 0.0316]
 [0.0492 0.3722 0.5786]]
Confusion Matrix
 [[ 3410   733   125]
 [  472 11074   377]
 [  158  1195  1858]]
```

over ‘baseline_model’ for all classes. For **class 0**, XGBoost gets 79.9% predictions right (vs 76.48% baseline). For **class 1**, the classifier gets 92.88% recall (vs 90.46% baseline) and for **class 2**, it is 57.86% XGBoost vs 57.40% for baseline.

Chart 4.2 shows how the classifiers perform for customers making the final purchase at different

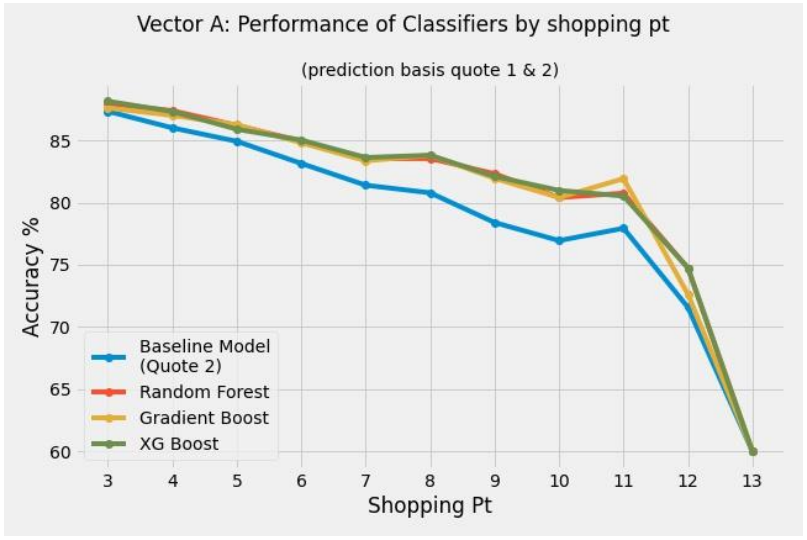


Chart 4.2

shopping points. We can see that the lead of classifiers over ‘baseline_model’ is narrow for shopping points 3 and 4, but they do much better across shopping points 5 to 11. This shows that **the classifiers are able to predict more accurately than baseline_model for customers further out** from shopping point 2.

4.3.2 Vector A: Models and feature importances

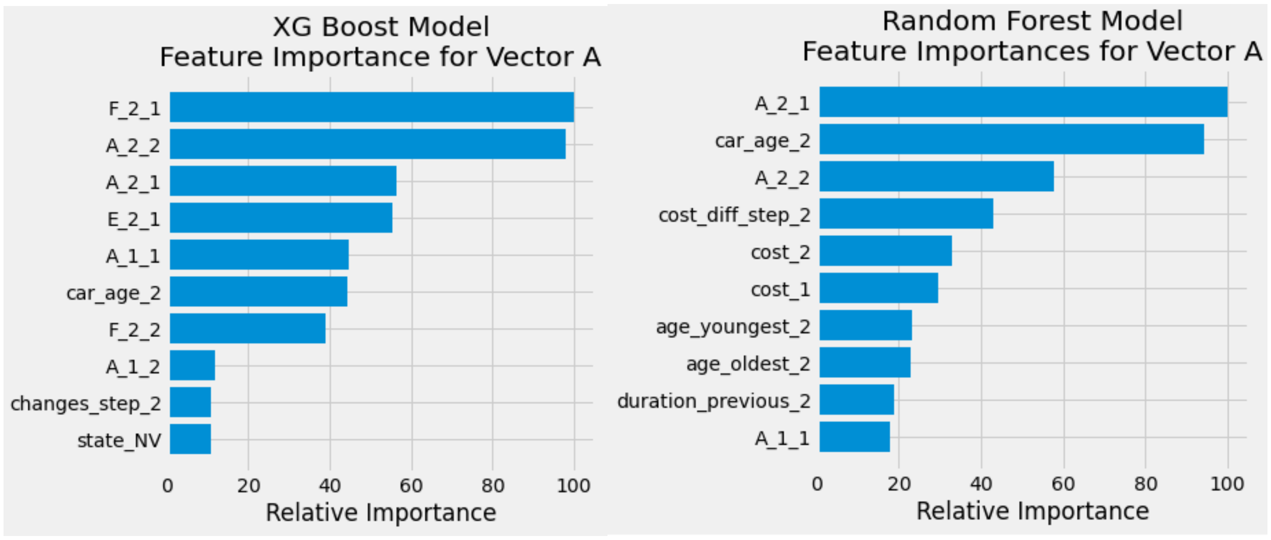


Chart 4.3

Chart 4.3 shows the top 10 features considered by XGBoost and Random Forest models. While ‘A_2_1’ , ‘A_2_2’ (the vector A choices at shopping_pt two) and car_age_2 **are common** to both classifiers, there are **noticeable differences**. The best classifier - XGBoost considers vector E, F and changes_step_2 as important. On the other hand, for Random Forest the cost, cost_difference (cost_diff_step_2), customer age and duration_previous are important features. Random Forest doesn’t have any representation of vectors E and F in the top 10. It should be noted that changes_step_2 (no. of vector changes made by customer at shopping_pt 2) and cost_diff_step_2 are **engineered features** that appear among top 10 features for these classifiers.

4.4 Model for Vector B - a binary vector

For vector B**,** the recall of baseline_model is **83.96%**. The baseline_model works best for class 0, with a recall of **88.19%**, while for class 1 it is lower at 79.25%. We can also see that more customers move from class 0 to class 1 (1904) than vice versa (1208).

B_2	0	1	total	B0_%	B1_%
B					
0	9018	1208	10226	88.19	11.81
1	1904	7272	9176	20.75	79.25

4.4.1 Vector B: Models and performance

In Chart 4.4, the blue line shows the performance of the 4 classifiers used to predict vector B. There was just a **0.04 to 0.11% improvement** over 'baseline_model' performance, with Random Forest having a slight edge over the other classifiers. It turns out that **vector B is the hardest to predict**. All classifiers are close to 100% in predicting customers who don't change. The best model - Random Forest with a recall of **84.07%,** is able to correctly predict **only about 3%** of customers who change. Chart 4.5 shows that very little separates the classifiers from the baseline_model for most shopping_pts. Random Forest does marginally better from quotes 10 to 12.

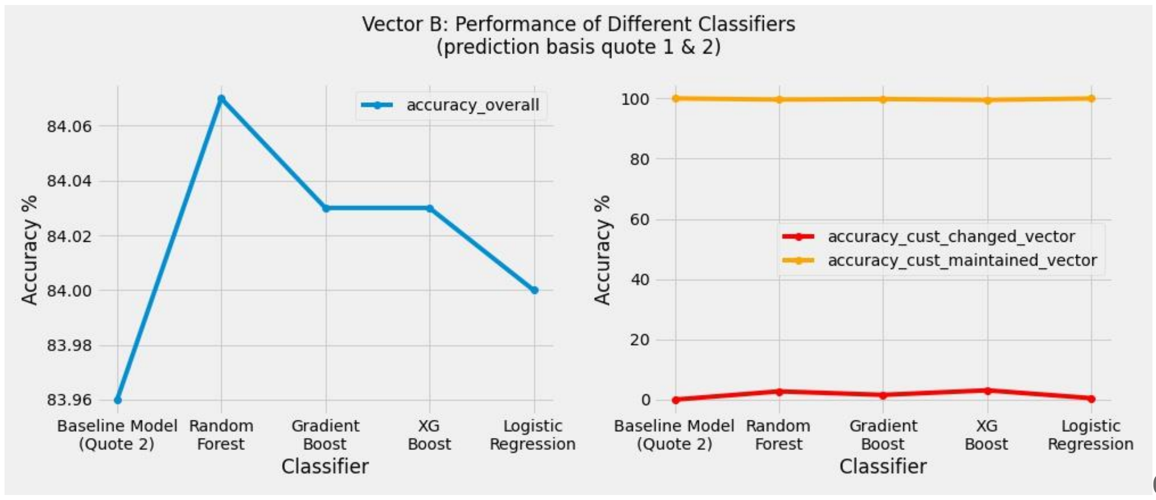


Chart 4.4

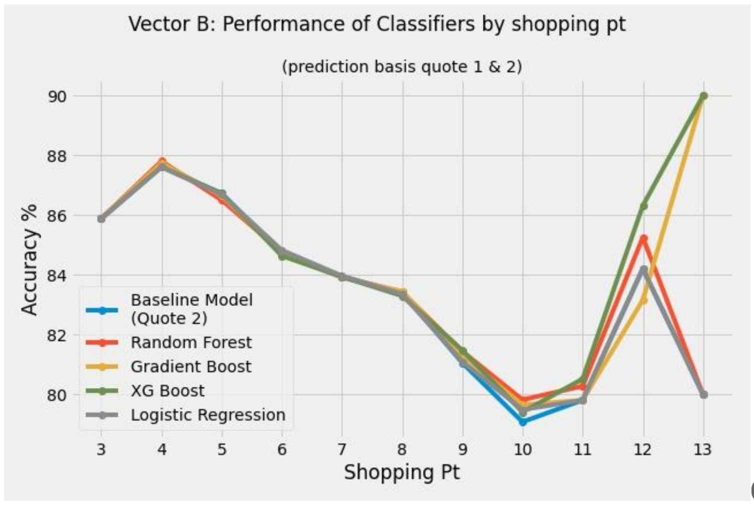
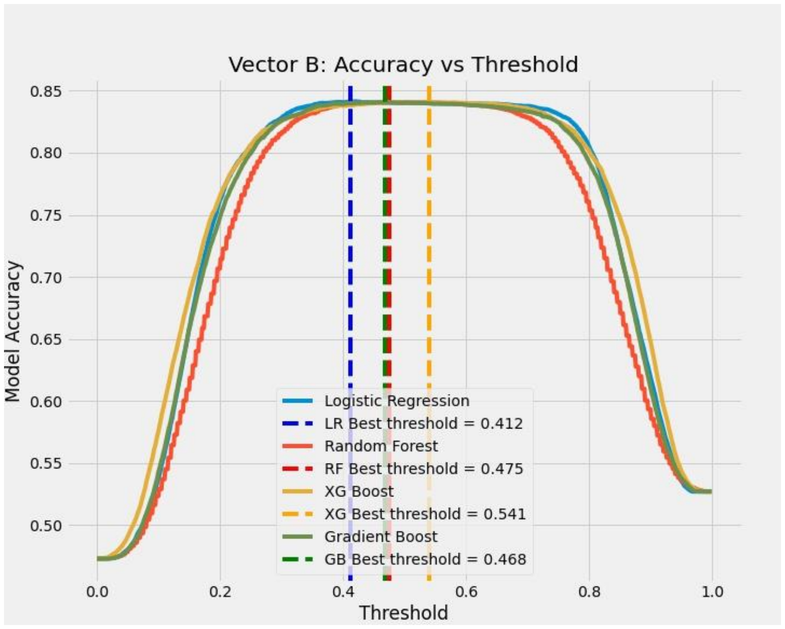


Chart 4.5

As B is a binary vector, we can do thresholding to see if that improves the performance of the classifiers.



For Logistic Regression Model, the best recall of **0.8414** occurs at threshold of 0.412.

For Random Forest Model, the best recall of **0.8407** occurs at threshold of 0.475.

For XG Boost Model, the best recall of **0.8408** occurs at threshold of 0.541.

For Gradient Boost Model, the best recall of **0.8409** occurs at threshold of 0.468.

So it turns out that with thresholding, **Logistic Regression** becomes the best model, with a **0.18%** improvement over the baseline_model. Small but nevertheless useful gain!

Chart 4.6

4.4.2 Vector B: Models and feature importances

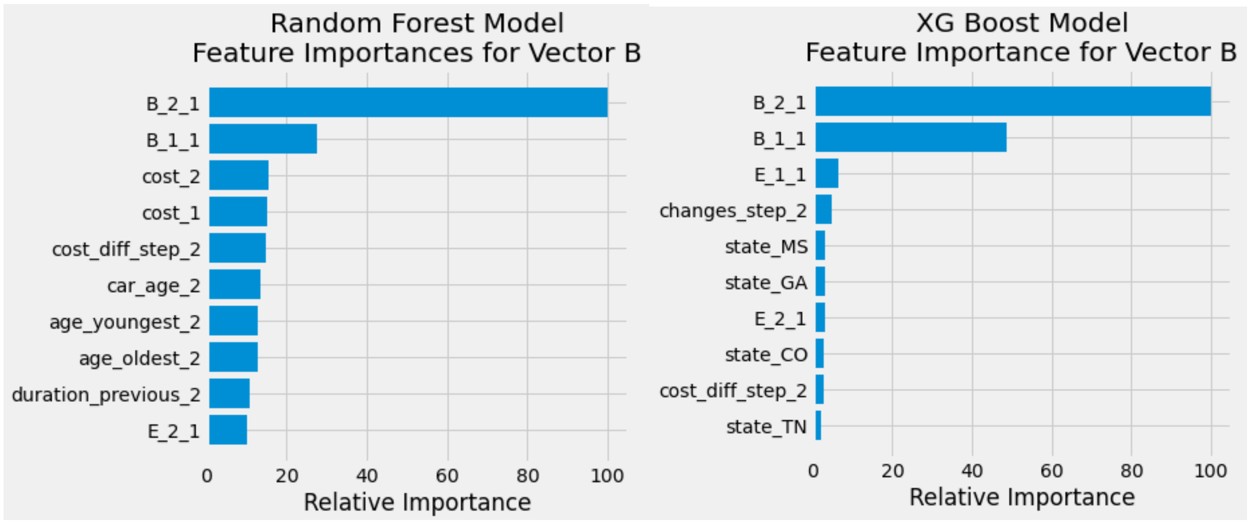


Chart 4.6

As expected, vector B_2 choices (B_2_1 & B_2_1) at shopping points 1 and 2 are the most important features for both the top classifiers. The other important features in Random Forest are primarily cost, cost_differnece, car_age and age of the customers. Vector E also has representation amongst top 10 features. It is again noteworthy that changes_step_2 (no. of vector changes made by customer at shopping_pt 2) and cost_diff_step_2 are **engineered features** that appear among top 10 features for these classifiers.

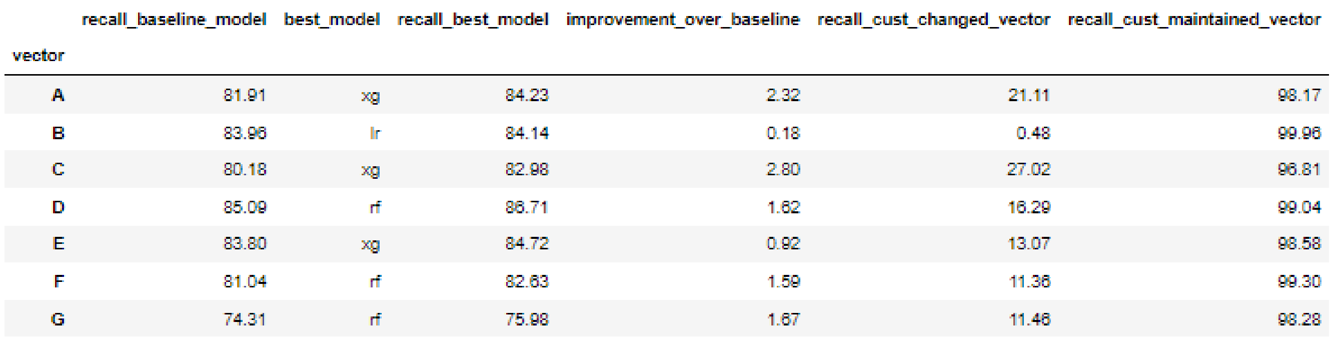
A similar process was followed to build and tune model parameters for the other vector. The results are summarized in the Chapter 5.

5.0 Findings and Recommendations

5.1 Findings

To predict the customer's final choice for each of the 7 product vectors, models were built using different classifiers based on the customer's quote 1 and 2 information. Each model's hyperparameters were tuned and thresholding was done for binary vectors B and E. The performance of different models was compared to then choose the best model for each vector.

Table 5.1 summarizes the performance of the best models across each of the 7 vectors.



vector	recall_baseline_model	best_model	recall_best_model	improvement_over_baseline	recall_cust_changed_vector	recall_cust_maintained_vector
A	81.91	xg	84.23	2.32	21.11	98.17
B	83.96	lr	84.14	0.18	0.48	99.96
C	80.18	xg	82.98	2.80	27.02	96.81
D	85.09	rf	86.71	1.62	16.29	99.04
E	83.80	xg	84.72	0.92	13.07	98.58
F	81.04	rf	82.63	1.59	11.36	99.30
G	74.31	rf	75.98	1.67	11.46	98.28

Table 5.1

Key Highlights

1) Random Forest and XGBoost came up as the best classifiers, with each of these classifiers giving the best performance for 3 of the 7 vectors.
2) The models gave a performance improvement ranging from **0.18% to 2.8%** over the baseline model (final vector = quote2).
    a) The average improvement was **1.59%**.
    b) The best performance was for vector C at **2.8%**.
    c) **Vector B** was the hardest to predict with only 0.18% improvement.

3) Another way to analyze the model performance is by looking at predictive prowess of various models for customers who changed their vectors (from quote 2 to the final purchase) and those who maintained their vector choices (classes).

    a) All models did a very good job in predicting **customers who didn't change** their vector choice. The recall of the models ranged from **96.81%** for vector C to **99.61%** for vector B. On average, the models predicted **98.54%** of such customers correctly.

    b) For customers **that changed vector choice**, the task of prediction was harder. The models had to predict which customers would change vector choice and what the new choice would be. Here the model performance ranged from **0.48%** improvement for vector B over baseline model to **27.02%** for vector C. On average, the models predicted **14.72%%** of such customers correctly.

5.2 Recommendations

The key recommendations on using the work done in this project are:

1) Follow the below 4 steps to get predictions for new data:
    a) Step 1 - make the data wide: Use the 'expand_horizontal' function in pre-processing notebook (04_Pre_processing_and_training_data_development.ipynb) to convert the data from long form to a wide form.
    b) Step 2 - extract quote 1 and 2: In the pre-processing notebook, create an instance of the QuoteHistory() class and pass the wide form dataframe to it using the 'pass_data' method. Then use the 'get_history' method, setting argument how='first2' to get columns related to the first two quotes.
    c) Step 3 - Pre-process and Transform data: In the pre-processing notebook, create an instance of the PreProcess() class and pass the dataframe from step 2 to it using the 'dataset' method. Then use the 'transform' method, setting argument quote_nos='[1,2]' to get X and y dataframes to be used for modeling. X contains the preprocessed and transformed

feature columns and y contains the 7 target vectors under columns 'A' through 'G'. Save these dataframes in separate csv files.

   d) Step 4 - Make Predictions: In the modeling notebook (05_Modelling_ver2.0.ipynb) , follow the steps outlined in the section 'How to make Predictions' to make predictions for the target vectors.

2) In order to make predictions using quotes other than quote 1 and 2, follow the above 4 steps for a different set of quotes.

6.0 Ideas for further research

Some ideas that could be worked on in order to further enhance predictive prowess of the models are:

1) **Creating an ensemble of different classifiers based on certain voting rules or by combining the probabilities of different models:**: The comparison of feature importance of different classifiers showed that as many as 5-6 of the top 10 features were different, even though the final performance of the models was very close to each other. This shows that different models were able to extract the customer's final vector choice based on different features. It was also found that certain models predicted certain vector classes better than others. This then raises the possibility that performance enhancements could be made by combining individual models in certain ways.

2) **More engineered features**: The two features that were engineered, namely 'cost_difference_step2' and 'changes_step_2' were found amongst top 10 features for most of the vectors and classifiers. Another feature that could be engineered is based on time. While time per se was not found to have any correlation with vector choice, the 'time difference' between two successive quotes might have some bearing on whether a customer makes vector

changes or not. The hypothesis here is that ‘time difference’ could be negatively correlated with no. of changes that a customer might make.

3) **More data to get the models to learn more about features like location**: The location (zip code) feature could not be used in model building as for more than 90% of the 6248 zip codes, the no. of customers were less than 30. There were just a handful of zip codes with 100 customers or more. So data on more customers would certainly help add location into the mix.