Big Data and Analytics Spring 2017

PROJECT TITLE: ALLSTATE POLICY PREDICTION

TEAM MEMBERS:
Gagandeep Singh
Janhvi Parekh
Srishti Govil

1. EXECUTIVE SUMMARY

The aim of this project is to predict the car insurance policy quotation a customer might buy along with its total cost. There are 7 policy options, each with 2 to 4 version values. A quote is a combination of the 7 options. The customers review multiple quotes before making the final purchase. Using the customer's shopping history and characteristics, a prediction of the exact options they will buy has to be made.

2. MOTIVATION

As an Insurance Agent approaches a customer with policy options, he is usually clueless about what policy combination the customer might buy. Thus, the process of policy quoting, and ultimately selection is through hit and trial method, which may take a long time and effort on both the agent's and the customer's side. The agent might offer the customer plenty of options, amongst which the customer might end up choosing one. Also, there might be a scenario wherein the customer might get confused and frustrated and hence might not buy any policy combination. The need of the hour is to reduce the number of these hits and trials and shorten the policy quotation process.
Thus, this model predicts the policy combination a customer might buy as well as the cost of the entire combination.
This would help in building customer relationships and improve customer satisfaction. The insurance company would also save valuable time and effort. It would also be a good start to understanding which policy options are popular among customers, so that Allstate can market those more vigorously. Also, options can be increasingly customized so as to achieve a higher conversion rate with the quoting process.

3. DATA DESCRIPTION

The training and test datasets contain transaction history for customers that ended up purchasing a policy for their car. There are 665,250 rows in the training data and about 200,000 in the test dataset. For each customer_ID, the quote history is available. The

training dataset contains the complete quote history with all the policy combinations quoted to the customer, with the last row containing the combination he/she actually purchased. The test dataset doesn't contain the entire quote history, nor the purchased options. The aim is to predict the 7 options that customer will end up purchasing.
Customer_ID:
A customer_ID can represent a group of people for the policy combination and the cost is dependent on the policies purchased as well as the characteristics of the customer.

Product Options:
There are 7 customizable options that a customer can purchase. Each option has 2,3 or 4 ordinal values.

Option Name	**Possible Values**
A	0,1,2
B	0,1
C	1,2,3,4
D	1,2,3
E	0,1
F	0,1,2,3
G	1,2,3,4

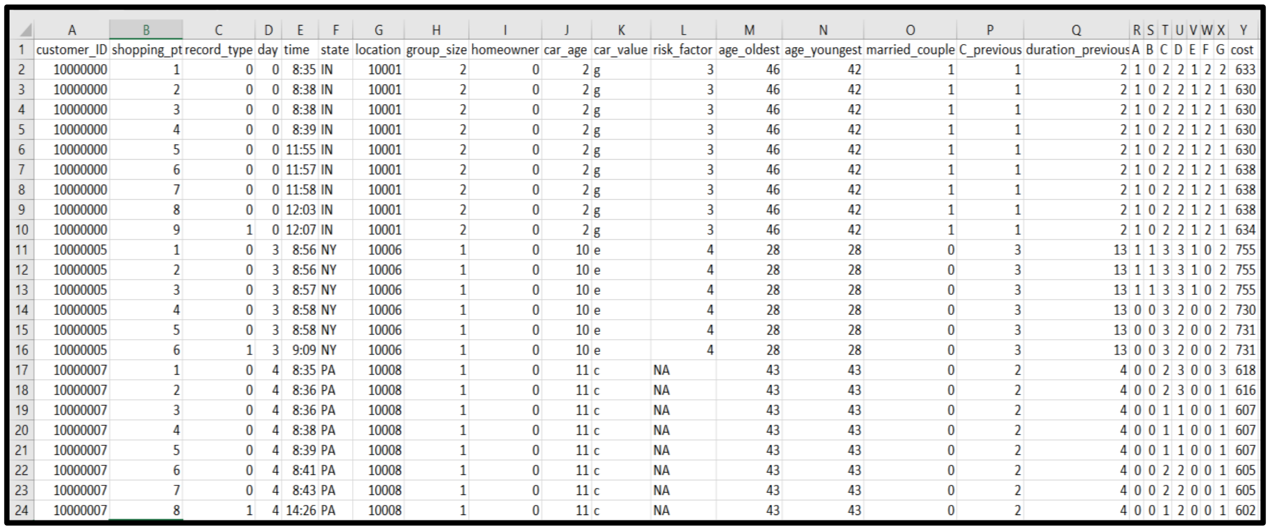
A product is a combination of A-G whose values are chosen from each of the options listed above. For e.g.: [A-0, B-1, C-3, D-1, E-0, F-2, G-3] can be a product option.

Variable Descriptions

customer_ID -	A unique identifier for the customer
shopping_pt -	Unique identifier for the shopping point of a given customer
record_type -	0=shopping point, 1=purchase point
day -	Day of the week (0-6, 0=Monday)
time -	Time of day (HH:MM)
state -	State where shopping point occurred
location -	Location ID where shopping point occurred
group_size -	How many people will be covered under the policy (1,2,3,4)
homeowner -	Whether the customer owns a home or not (0=no, 1=yes)
car_age -	Age of the customer's car
car_value -	How valuable was the customer's car when new. From a-i (1-9)

risk_factor -	An ordinal assessment of how risky the customer is (1,2,3,4)
age_oldest -	Age of the oldest person in customer's group
age_youngest -	Age of the youngest person in customer’s group
married_couple -	Does the customer group contain a married couple (0=no, 1=yes)
C_previous -	What the customer formerly had or currently has for product option C (0=nothing,1,2,3,4)
duration_previous -	How long the customer was covered by their previous issuer(years)
A,B,C,D,E,F,G -	the coverage options
cost -	cost of the quoted coverage options

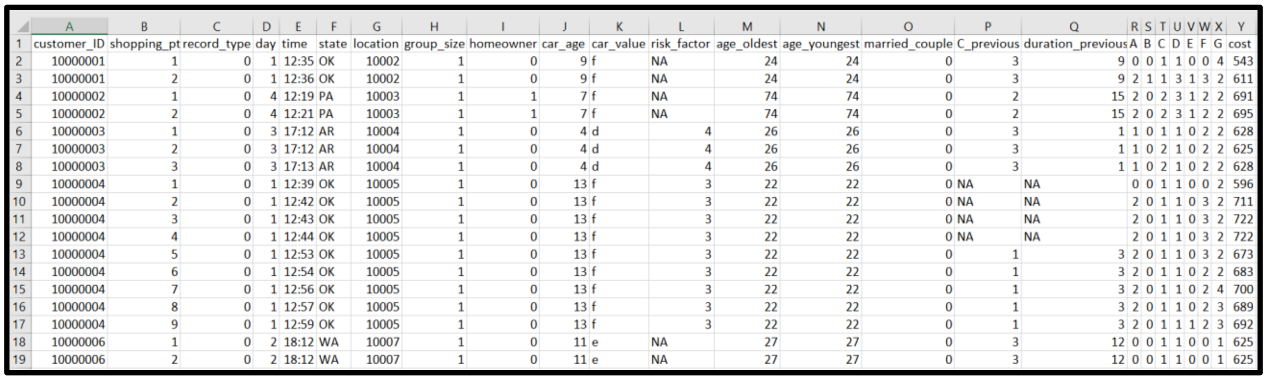
A small sample of the training data is shown below:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	customer_ID	shopping_pt	record_type	day	time	state	location	group_size	homeowner	car_age	car_value	risk_factor	age_oldest	age_youngest	married_couple	C_previous	duration_previous	A	B	C	D	E	F	G	cost
2	10000000	1	0	0	8:35	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	2	633
3	10000000	2	0	0	8:38	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	630
4	10000000	3	0	0	8:38	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	630
5	10000000	4	0	0	8:39	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	630
6	10000000	5	0	0	11:55	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	630
7	10000000	6	0	0	11:57	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	638
8	10000000	7	0	0	11:58	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	638
9	10000000	8	0	0	12:03	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	638
10	10000000	9	1	0	12:07	IN	10001	2	0	2	g	3	46	42	1	1	2	1	0	2	2	1	2	1	634
11	10000005	1	0	3	8:56	NY	10006	1	0	10	e	4	28	28	0	3	13	1	1	3	3	1	0	2	755
12	10000005	2	0	3	8:56	NY	10006	1	0	10	e	4	28	28	0	3	13	1	1	3	3	1	0	2	755
13	10000005	3	0	3	8:57	NY	10006	1	0	10	e	4	28	28	0	3	13	1	1	3	3	1	0	2	755
14	10000005	4	0	3	8:58	NY	10006	1	0	10	e	4	28	28	0	3	13	0	0	3	2	0	0	2	730
15	10000005	5	0	3	8:58	NY	10006	1	0	10	e	4	28	28	0	3	13	0	0	3	2	0	0	2	731
16	10000005	6	1	3	9:09	NY	10006	1	0	10	e	4	28	28	0	3	13	0	0	3	2	0	0	2	731
17	10000007	1	0	4	8:35	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	2	3	0	0	3	618
18	10000007	2	0	4	8:36	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	2	3	0	0	1	616
19	10000007	3	0	4	8:36	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	1	1	0	0	1	607
20	10000007	4	0	4	8:38	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	1	1	0	0	1	607
21	10000007	5	0	4	8:39	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	1	1	0	0	1	607
22	10000007	6	0	4	8:41	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	2	2	0	0	1	605
23	10000007	7	0	4	8:43	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	2	2	0	0	1	605
24	10000007	8	1	4	14:26	PA	10008	1	0	11	c	NA	43	43	0	2	4	0	0	1	2	0	0	1	602

Fig 3.1 Training data sample

As seen in the above data, the quoting history for customer IDs 10000000, 10000005 and 10000007 is available. Its seen that customer_ID 10000000 is offered a combination of policies 9 times before he makes a purchase. A record type=1 indicates that the combination has been purchased, while 0 indicates that its not. There are also NA values for the risk_factor column of customer_ID 10000007 which will have to be handled later.
A small sample of the test data is shown below:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	customer_ID	shopping_pt	record_type	day	time	state	location	group_size	homeowner	car_age	car_value	risk_factor	age_oldest	age_youngest	married_couple	C_previous	duration_previous	A	B	C	D	E	F	G	cost
2	10000001	1	0	1	12:35	OK	10002	1	0	9	f	NA	24	24	0	3	9	0	0	1	1	0	0	4	543
3	10000001	2	0	1	12:36	OK	10002	1	0	9	f	NA	24	24	0	3	9	2	1	1	3	1	3	2	611
4	10000002	1	0	4	12:19	PA	10003	1	1	7	f	NA	74	74	0	2	15	2	0	2	3	1	2	2	691
5	10000002	2	0	4	12:21	PA	10003	1	1	7	f	NA	74	74	0	2	15	2	0	2	3	1	2	2	695
6	10000003	1	0	3	17:12	AR	10004	1	0	4	d	4	26	26	0	3	1	1	0	1	1	0	2	2	628
7	10000003	2	0	3	17:12	AR	10004	1	0	4	d	4	26	26	0	3	1	1	0	2	1	0	2	2	625
8	10000003	3	0	3	17:13	AR	10004	1	0	4	d	4	26	26	0	3	1	1	0	2	1	0	2	2	628
9	10000004	1	0	1	12:39	OK	10005	1	0	13	f	3	22	22	0	NA	NA	0	0	1	1	0	0	2	596
10	10000004	2	0	1	12:42	OK	10005	1	0	13	f	3	22	22	0	NA	NA	2	0	1	1	0	3	2	711
11	10000004	3	0	1	12:43	OK	10005	1	0	13	f	3	22	22	0	NA	NA	2	0	1	1	0	3	2	722
12	10000004	4	0	1	12:44	OK	10005	1	0	13	f	3	22	22	0	NA	NA	2	0	1	1	0	3	2	722
13	10000004	5	0	1	12:53	OK	10005	1	0	13	f	3	22	22	0	1	3	2	0	1	1	0	3	2	673
14	10000004	6	0	1	12:54	OK	10005	1	0	13	f	3	22	22	0	1	3	2	0	1	1	0	2	2	683
15	10000004	7	0	1	12:56	OK	10005	1	0	13	f	3	22	22	0	1	3	2	0	1	1	0	2	4	700
16	10000004	8	0	1	12:57	OK	10005	1	0	13	f	3	22	22	0	1	3	2	0	1	1	0	2	3	689
17	10000004	9	0	1	12:59	OK	10005	1	0	13	f	3	22	22	0	1	3	2	0	1	1	1	2	3	692
18	10000006	1	0	2	18:12	WA	10007	1	0	11	e	NA	27	27	0	3	12	0	0	1	1	0	0	1	625
19	10000006	2	0	2	18:12	WA	10007	1	0	11	e	NA	27	27	0	3	12	0	0	1	1	0	0	1	625

Fig 3.2 Test data sample

As seen in the above data, only the partial quoting history for customer IDs 10000001, 10000002, 10000003, 10000004, 10000006 is available.

4. METHODOLOGY

i. Data Exploration

To get a good sense of the dataset, exploratory data analysis and visualization was conducted in Tableau. A histogram of the Customers with 'state' in the US was visualized. It was found that most customers were from the states of Florida and New York.
A histogram of Customers with their 'car_value' (how valuable was their car when new) showed that most customers had a car value = "e" (5). For a detailed view refer Appendix *Fig 9.2*. Car_value was assigned a numerical value from 1-9 since the original data had string values from a-i, that are hard to interpret. Imputation of missing values of car_value with the mode (= "e" or 5) was clear through this data exploration.
A distribution of policy options offered to customers was also visualized. This showed the count of customers to which different versions of policies were offered.

ii. Data Preprocessing

The complex nature of the dataset involved quite a lot of preprocessing to ensure that all the features we needed for making the prediction models were complete and in the right format. In terms of missing values, columns like C_previous, duration_previous, risk_factor, car_values had to be handled.

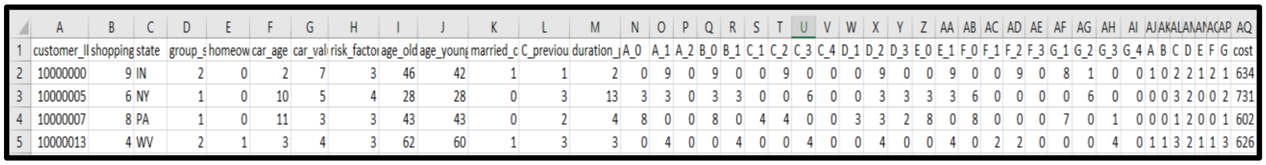
C_previous, duration_previous:
For C_previous and duration_previous, missing values (NAs) were replaced with 0. NA values probably indicated new customers, which meant that C_previous and duration_previous wouldn't apply to them, hence 0 would be an apt replacement.

Car_value:
This was a string value in the dataset from a-i that indicated the value of the customer's car when new. A value of "a" was the lowest on the scale, "i" being the highest. These string values were replaced by numerical values from 1-9. Thus, a customer with car_value = b, would now have car_value = 2. This replacement was inevitable, since a string value would not be of much help while generating predictive models and this feature could not be ignored either, since car_value is a major factor when it comes to car insurance policy prediction. Next, missing values were replaced by the mode of the remaining observations which was "e" or 5 in our case as seen during data exploration in Tableau.

Risk_factor:
A large chunk of values were missing for risk_factor in the training and test dataset-approximately 35% in both. Imputing the median/mode would not be prudent for such a large number of observations. Risk_factor was dependent on various customer attributes like group_size, homeowner, married_couple, age_oldest and age_youngest. Thus, running various regression models to predict risk_factor based on the customer attributes above seemed to be a good way to go forward. Linear regression was successful, but yielded an accuracy of about 28% only. K-means clustering to cluster similar customers together and then impute the average value of the risk_factor in that cluster to observations with missing values was another approach. However, assigning observations to a cluster was not easy and the clusters were not distinctly formed which would lead to sub-par accuracy again. Techniques like Neural networks and MICE package for imputation of missing values could not handle the large volume of data and processing and thus, even they had to be eliminated. Finally, Multinomial Regression was used to build a predictive model for risk_factor. The training data for this model was all the observations for customer_IDs for which risk_factor was available. All the missing value rows were part of the test data. A fit on the training data was used to predict test data values for risk_factor which were then combined and replaced in the original dataset.

Transaction history of each customer provides us with the valuable information about customer's inclination towards any particular quotation. Data aggregation was done in python such that there will be 1 row for each customer and 22 new features were introduced representing each version of different policy options. These columns contain the number of interactions each customer had with that particular version. This was done by using groupby and sum function in python.

This resulted in a set of new features and hence we could consolidate our dataset to have only 1 row per customer ID, with record type =1 indicating the customer purchased the policy options. The figure below shows the new dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ
1	customer_II	shopping	state	group_s	homeow	car_age	car_val	risk_facto	age_old	age_young	married_c	C_previou	duration_	A_0	A_1	A_2	B_0	B_1	C_1	C_2	C_3	C_4	D_1	D_2	D_3	E_0	E_1	F_0	F_1	F_2	F_3	G_1	G_2	G_3	G_4	A	B	C	D	E	F	G	cost
2	10000000	9	IN	2	0	2	7	3	46	42	1	1	2	0	9	0	9	0	0	9	0	0	0	9	0	0	9	0	0	9	0	8	1	0	0	1	0	2	2	1	2	1	634
3	10000005	6	NY	1	0	10	5	4	28	28	0	3	13	3	3	0	3	3	0	0	6	0	0	3	3	3	3	6	0	0	0	0	6	0	0	0	0	3	2	0	0	2	731
4	10000007	8	PA	1	0	11	3	3	43	43	0	2	4	8	0	0	8	0	4	4	0	0	3	3	2	8	0	8	0	0	0	7	0	1	0	0	0	1	2	0	0	1	602
5	10000013	4	WV	2	1	3	4	3	62	60	1	3	3	0	4	0	0	4	0	0	4	0	0	4	0	0	4	0	2	2	0	0	0	4	0	1	1	3	2	1	1	3	626

Fig 4.1 Consolidated customer information in 1 row/customer

iii. Predictive Modelling

After Data Preprocessing, the cleaned training and test datasets were uploaded to Amazon S3, to the bucket created for the project. The dataset was read using PySpark on Databricks in order to do further processing and build models.

Cleaned training dataset was read on Databricks using PySpark and was divided into training and validation sets in 70:30 ratio. This was done to train the model on training dataset and check the accuracy on validation dataset.

Required libraries were imported and categorical variables in the dataset such as 'state' were converted to a numerical vector form to provide for different models. This was done using StringIndexer and OneHotEncoder which converts categorical variable to binary sparse vector. All input features including 'StateVector' created previously were combined into a vector using VectorAssembler. This was done to pass all input parameters in a vector form to different models.
We planned to build three different classifiers which were all based on supervised learning algorithms.

1. Decision Trees
2. Random Forest
3. Gradient Boosting

Gradient boosting only supports binary classification, so it could not be used to solve this multiclass problem.

Decision Tree and Random Forest models were built and run to predict the policy combination A-G. The Pipeline function was used to give the execution order in which all the above processing occurred.
Since we had a multiclass multilabel classification problem with different number of classes in each label, we didn't use multiclass library function of sklearn library. Instead, we used *for* loop to repeat the whole process for each label. Finally, results were appended to list at the end of each iteration of for loop. So, this list contained the dataframes of predicted values. Trained model was fitted on validation dataset to make the predictions and check the accuracy.

Given the high prediction accuracy of Random Forest and its tendency to avoid overfitting, the predictions made by Random Forest model on test dataset were used to build the web application and fed into regression model for predicting the cost of policy quotation.

Linear Regression was used to predict the cost of the combination. Cost of the policy combination is a function of both customer characteristics and quoted policy. A similar kind of procedure was followed to predict the cost. Train data was loaded on AWS which was divided into training and validation set. A linear regression model was built on training dataset and its accuracy was measured on validation set. Finally, predictions were made for test data using the trained linear regression model.

5. WEB APPLICATION

Caspio, a cloud platform was used to build a web application to display the results. Once the predictions were made for test data, they were exported to a csv file which was connected to Caspio. HTML and CSS were used to enhance the look of the webpage. The Tableau dashboard URLs that were created during Data Exploration phase were also added to the webpage. Navigation between different pages was also a functionality - the different tabs were: Home Page, Customer Information, Customer Attributes, Policy Distributions and Policy Quotations.
Customer Information contains an entry field for customer_ID and on submitting one can view customer details. For more details, refer Appendix *Fig 9.1*
Customer Attributes and Policy Distributions redirect to the dashboards on Tableau Public. For more details, refer Appendix *Fig 9.2 , 9.3*
Policy Quotations outputs the predicted combination of policies done through Random Forest on entering a customer_ID. For more details, refer Appendix *Fig 9.4*

6. RESULTS

With an aim to reduce the test error and improve prediction accuracy, different models were forayed to better predict the policy combination for each customer given their characteristics and transaction history. The accuracy achieved by both Decision Tree and Random Forest was pretty good for all the labels. The figures below show a detailed view of each policy's prediction accuracy on the validation dataset.

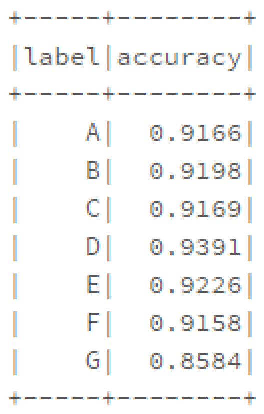
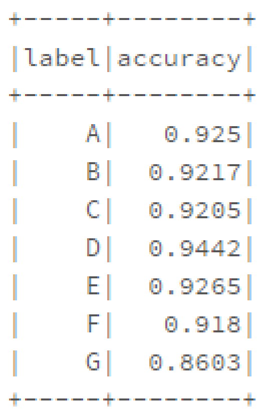

```
+-----+--------+
|label|accuracy|
+-----+--------+
|    A|  0.9166|
|    B|  0.9198|
|    C|  0.9169|
|    D|  0.9391|
|    E|  0.9226|
|    F|  0.9158|
|    G|  0.8584|
+-----+--------+
```

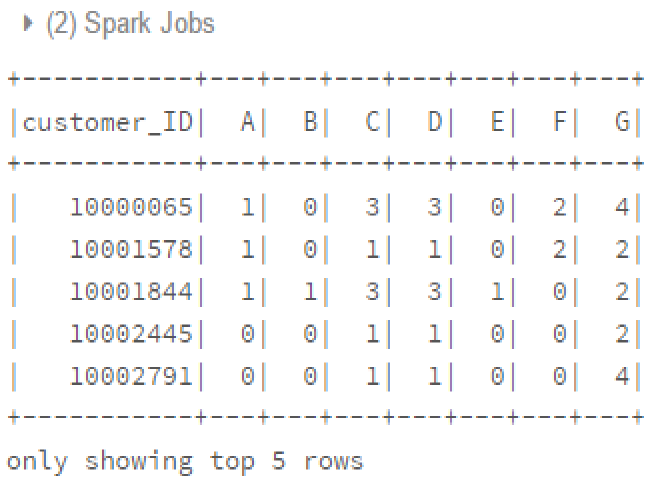
Fig 6.1 A-G accuracy Random Forest



```
+-----+--------+
|label|accuracy|
+-----+--------+
|    A|   0.925|
|    B|  0.9217|
|    C|  0.9205|
|    D|  0.9442|
|    E|  0.9265|
|    F|   0.918|
|    G|  0.8603|
+-----+--------+
```

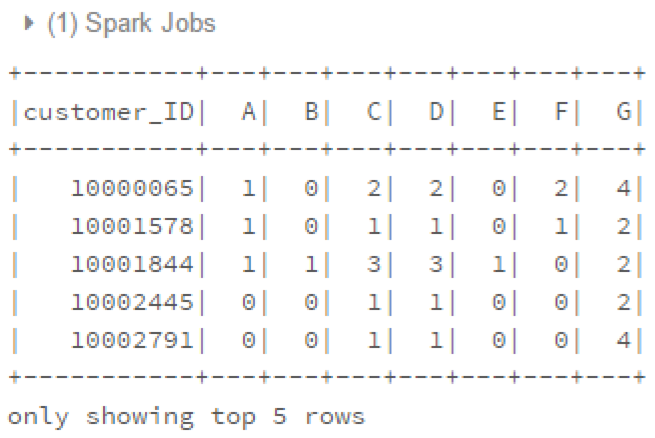
Fig 6.2 A-G accuracy Decision Tree

Next, prediction on the test data customer_IDs was done. The figures below show a detailed view of each policy's prediction on the test dataset.



```
▸ (2) Spark Jobs

+-----------+---+---+---+---+---+---+---+
|customer_ID|  A|  B|  C|  D|  E|  F|  G|
+-----------+---+---+---+---+---+---+---+
|   10000065|  1|  0|  3|  3|  0|  2|  4|
|   10001578|  1|  0|  1|  1|  0|  2|  2|
|   10001844|  1|  1|  3|  3|  1|  0|  2|
|   10002445|  0|  0|  1|  1|  0|  0|  2|
|   10002791|  0|  0|  1|  1|  0|  0|  4|
+-----------+---+---+---+---+---+---+---+
only showing top 5 rows
```

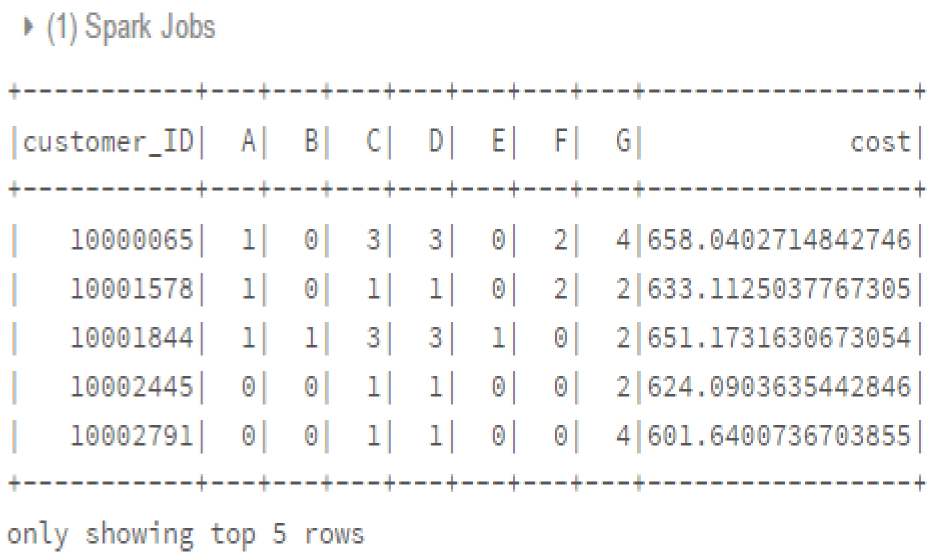
Fig 6.3 A-G prediction Random Forest



```
▸ (1) Spark Jobs

+-----------+---+---+---+---+---+---+---+
|customer_ID|  A|  B|  C|  D|  E|  F|  G|
+-----------+---+---+---+---+---+---+---+
|   10000065|  1|  0|  2|  2|  0|  2|  4|
|   10001578|  1|  0|  1|  1|  0|  1|  2|
|   10001844|  1|  1|  3|  3|  1|  0|  2|
|   10002445|  0|  0|  1|  1|  0|  0|  2|
|   10002791|  0|  0|  1|  1|  0|  0|  4|
+-----------+---+---+---+---+---+---+---+
only showing top 5 rows
```

Fig 6.4 A-G prediction Decision Tree

Since Random Forest generally provides high prediction accuracy and avoids overfitting, the predictions done by Random Forest model on test dataset were used as input to regression model for predicting the cost of the policy quotation. Cost of the policy combination was predicted using Linear Regression. Model was trained on training data and test errors were computed using validation dataset. RMSE(Root mean square error) value came out to be 35.6 and MAE(Mean Average Error) was 28.03. The error rates could be further reduced by using advanced data mining techniques and regression algorithms. Results from test dataset are as follows:



```
▸ (1) Spark Jobs

+-----------+---+---+---+---+---+---+---+-----------------+
|customer_ID|  A|  B|  C|  D|  E|  F|  G|             cost|
+-----------+---+---+---+---+---+---+---+-----------------+
|   10000065|  1|  0|  3|  3|  0|  2|  4|658.0402714842746|
|   10001578|  1|  0|  1|  1|  0|  2|  2|633.1125037767305|
|   10001844|  1|  1|  3|  3|  1|  0|  2|651.1731630673054|
|   10002445|  0|  0|  1|  1|  0|  0|  2|624.0903635442846|
|   10002791|  0|  0|  1|  1|  0|  0|  4|601.6400736703855|
+-----------+---+---+---+---+---+---+---+-----------------+
only showing top 5 rows
```

Fig 6.5 Prediction of cost

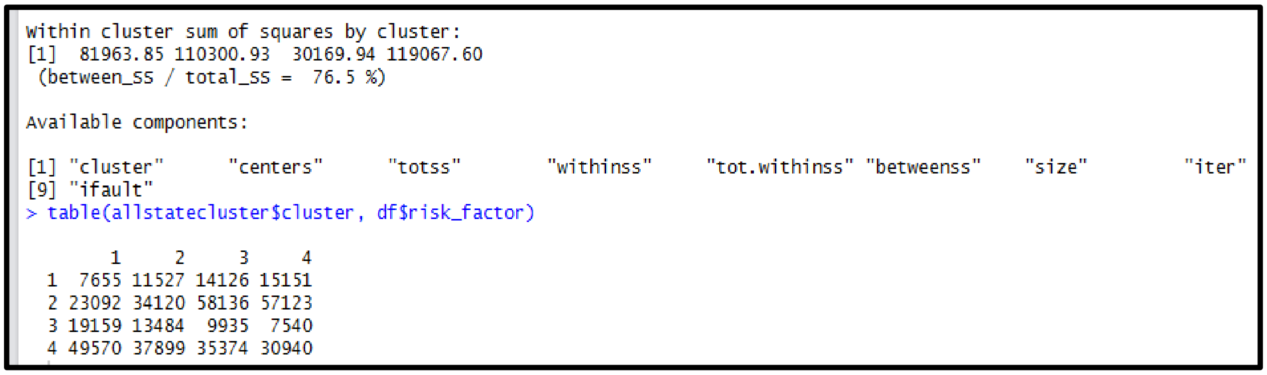
7. CHALLENGES

i. Data Cleaning

The data set had a large number of missing values which were hard to impute in some cases. Some missing values like C_Previous and Duration_previous were simply replaced by 0. Missing values in the car_value column were replaced by its mode. However, values like risk_factor depended on the characteristics of each customer. Thus it was not possible to simply impute the median or the most commonly occurring risk_factor observation for the missing values. Thus, after exploration of many methods, the ideal way was to run Multinomial Regression in R to predict NA values for risk_factor during the data preprocessing phase. Numerous other methods were assessed prior to fixing Multinomial regression as our model, for eg; K-means clustering, MICE package for imputation, Neural Networks.

However, each of these methods failed due to different reasons. K-means clustering was successful in clustering similar customers together based on attributes, but there was no way to assign new Customer_IDs to a cluster. If risk_factor value for Customer_ID 10000007 was missing, there was no way to assign this customer to one of the clusters and then impute that cluster's risk_factor value to him. Also, the clusters were not very distinct and assigning missing observations with the average of cluster risk_factor values was a difficult task, and not accurate.

The figure below shows a table containing cluster IDs against risk_factor values. Cluster ID = 1 has 7655 observations with risk_factor=1, 11527 with risk_factor=2, 14126 with risk_factor=3 and 15151 with risk_factor=4.

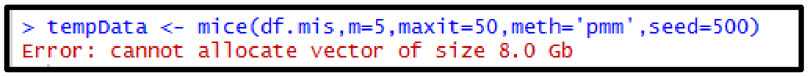


```
Within cluster sum of squares by cluster:
[1]  81963.85 110300.93  30169.94 119067.60
 (between_SS / total_SS =  76.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> table(allstatecluster$cluster, df$risk_factor)

        1     2     3     4
  1  7655 11527 14126 15151
  2 23092 34120 58136 57123
  3 19159 13484  9935  7540
  4 49570 37899 35374 30940
```

Fig 7.1 K-means clustering for risk_factor imputation

Neural Networks would have been an ideal approach to this problem, however, it couldn't handle the sheer volume of data. The computation power needed was very high and neural networks are time-consuming even with small data sizes due to multiple hidden layers and weight computation iterations.

Similarly, the MICE package did not work, again, due to the volume of data as seen in the figure below.



```
> tempData <- mice(df.mis,m=5,maxit=50,meth='pmm',seed=500)
Error: cannot allocate vector of size 8.0 Gb
```

Fig 7.2 MICE imputation failure

Finally, Multinomial Regression was chosen to predict the missing values for the risk_factor. Though the accuracy was not as good in this case, as could have been achieved through a model like Neural Networks or advanced regression techniques, it was more accurate than simple replacement of missing values by Mean/Median.
Thus, a lot of time and effort was spent in the data preprocessing phase to ensure we have the best possible version of data, that was both complete and accurate.

ii. Understanding the dataset

Due to lack of knowledge of the Insurance industry, understanding the dataset and a lot of the variables in the dataset was initially a difficulty that had to be overcome by researching about how insurance policies work, and what each variable might mean. It was important to understand which features depended on others in the data. For eg; risk_factor was derived from customer attributes. Extensive knowledge about the dataset was a necessity to understand what variables are important for predicting the policy quotations and the cost of each quotation.

iii. Learning various new technologies

This project had multiple phases and each phase required the use of different tools and technologies depending on need. The data exploration phase was done in Tableau, which is simple to use, but publishing our dashboards to Tableau Public and embedding a url to those in our web application was something that the team never had previous experience with.
Data Preprocessing and Cleaning was done in R and Jupyter notebook.
Storing our dataset on AWS S3 was also a first time effort, but it proved to be extremely beneficial for storing large amounts of data and processing it. Also, setting up a Spark cluster and running spark scripts was initially a daunting task since PySpark on Databricks had never been used before.
Learning had to be done from scratch, including how to read files from AWS, process the data, run predictive models on it, store the output in a readable format in PySpark. Handling and manipulating dataframes in PySpark was often quite different from the way it's done in R and Python, which took some time and effort.
Also, regression models like Random Forest and Decision Trees needed the inputs to be numerical and in a particular format. Features like 'state' had to be converted to numerical indexes and all the input features needed to be combined into one vector for the Random

Forest and Decision Tree model input parameter. These proved to take up some time and effort and a lot of research from various sources.

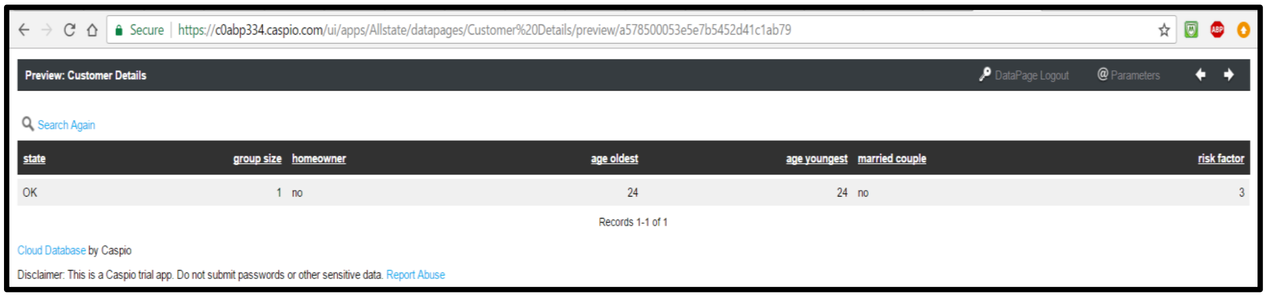
iv. Web Application

Having no prior front-end development experience, developing the web application and connecting it to the database was also a challenge. The first attempt was to build the front-end using jsp and ajax on HTML, however even after repeated efforts, the connection to database failed. Therefore, Caspio, a web development application was used to build the front-end. It had a simple UI to assist in the web form development. Also, connection to the database was simple. Further, HTML and CSS was used in Caspio pages to enhance the application appearance and navigation functionality.

8. CONCLUSION

Thus, prediction of policy options can reduce the quoting process significantly. Early offering of correct policy combination leads to stronger customer relationships and prevents losses on the insurance company’s side.
Built model can be used to improve pricing accuracy and offer a customized quotation.
Big data and Analytics can be used to find similarity between customers. Similar customers can then be offered similar policies.

9. APPENDIX



https://c0abp334.caspio.com/ui/apps/Allstate/datapages/Customer%20Details/preview/a578500053e5e7b5452d41c1ab79

Preview: Customer Details

DataPage Logout   Parameters

Search Again

state	group size	homeowner	age oldest	age youngest	married couple	risk factor
OK	1	no	24	24	no	3

Records 1-1 of 1

Cloud Database by Caspio

Disclaimer: This is a Caspio trial app. Do not submit passwords or other sensitive data. Report Abuse

Fig 9.1 Customer attributes for 10000001

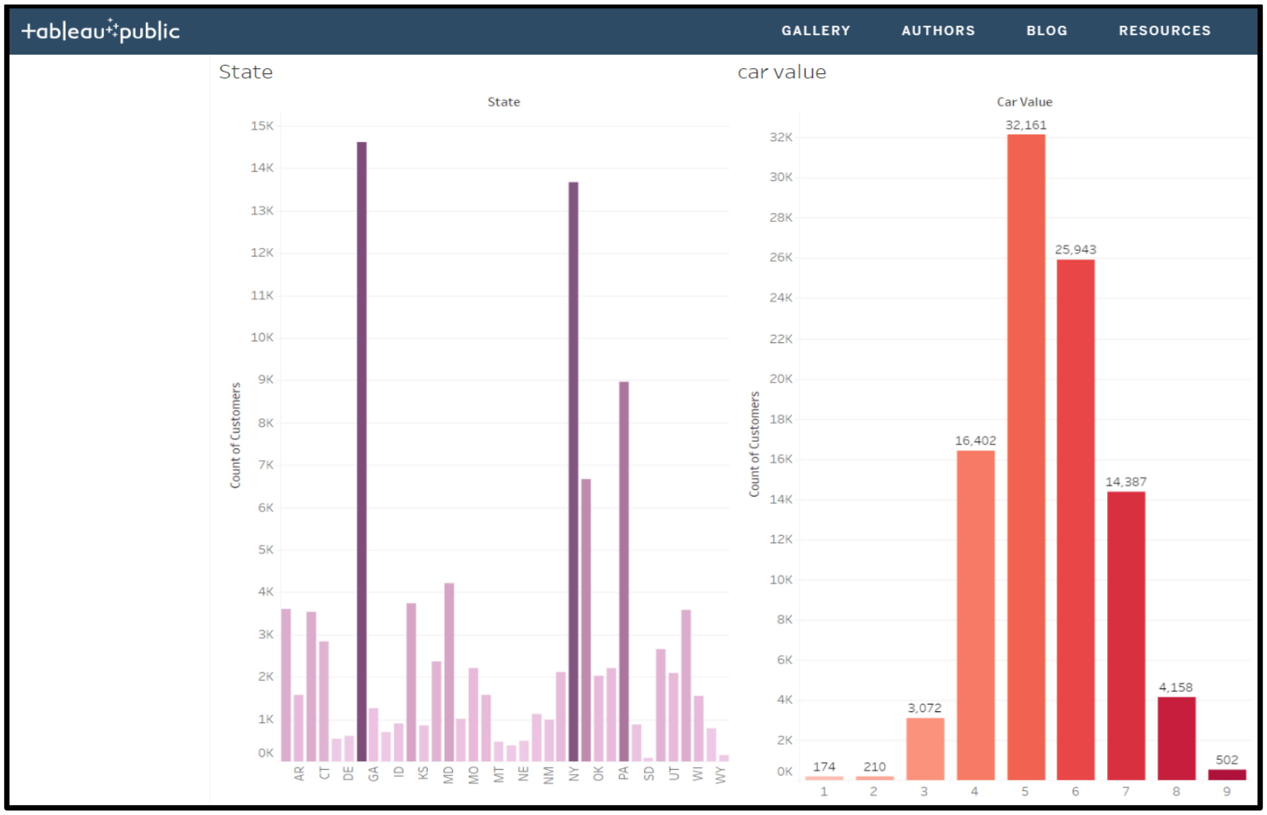


Fig 9.2 Customer attributes: state and car_value distribution

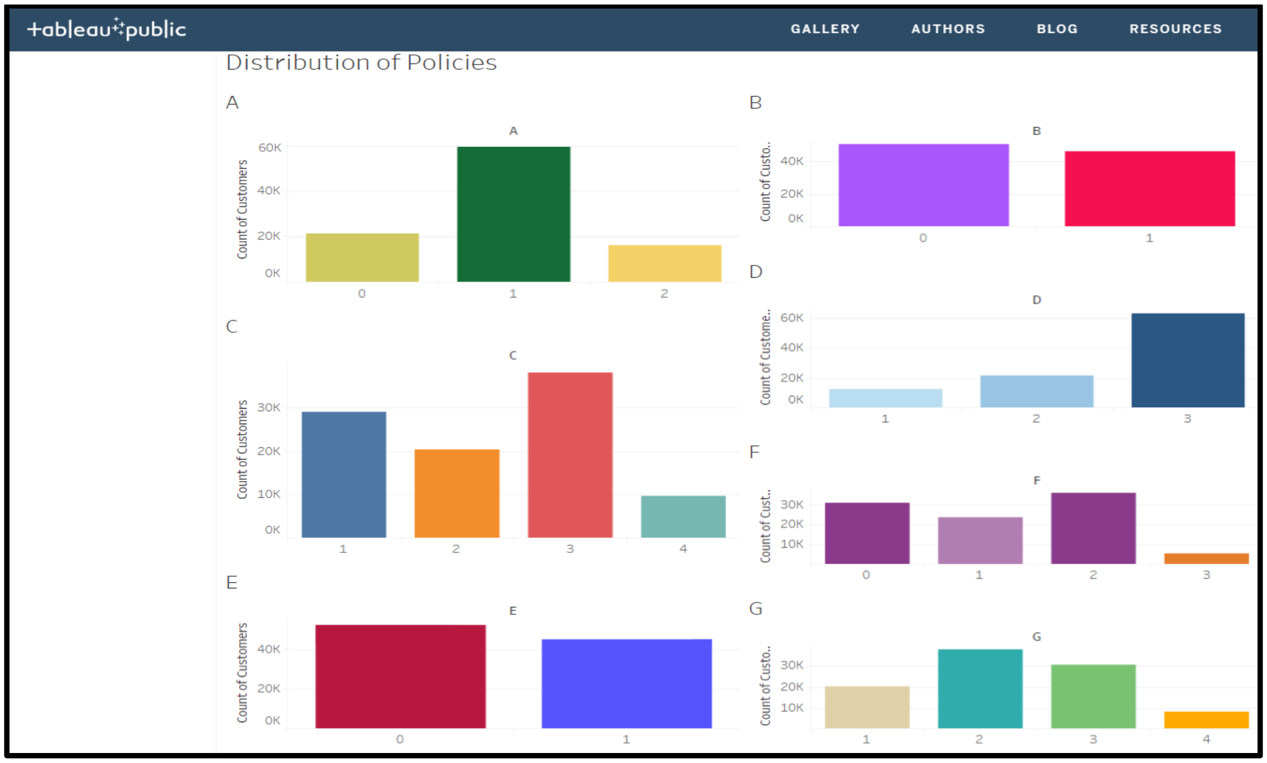
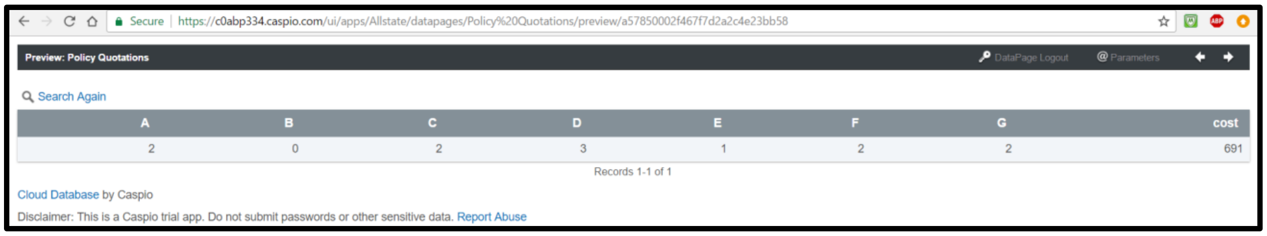


Fig 9.3 Distribution of Policies


Secure | https://c0abp334.caspio.com/ui/apps/Allstate/datapages/Policy%20Quotations/preview/a57850002f467f7d2a2c4e23bb58
Preview: Policy Quotations
DataPage Logout
Parameters
Search Again

A	B	C	D	E	F	G	cost
2	0	2	3	1	2	2	691

Records 1-1 of 1

Cloud Database by Caspio

Disclaimer: This is a Caspio trial app. Do not submit passwords or other sensitive data. Report Abuse

Fig 9.4 Policy Quotations for Test data, Customer_ID 10000002

10. REFERENCES

https://spark.apache.org/docs/2.1.0/ml-classification-regression.html#decision-tree-classifier
https://spark.apache.org/docs/2.1.0/ml-classification-regression.html#random-forest-classifier
https://spark.apache.org/docs/2.1.0/mllib-ensembles.html#classification
https://spark.apache.org/docs/1.6.2/api/python/pyspark.sql.html
http://stackoverflow.com/questions/36132322/join-two-data-frames-select-all-columns-from-one-and-some-columns-from-the-othe
http://stackoverflow.com/questions/40467449/how-to-select-and-order-multiple-columns-in-a-pyspark-dataframe-after-a-join
https://spark.apache.org/docs/0.9.1/python-programming-guide.html
https://www.kaggle.com/c/allstate-purchase-prediction-challenge/data
http://www.terpconnect.umd.edu/~kpzhang/teaching/budt758b_s17/index.html
https://pandas.pydata.org/pandas-docs/stable/missing_data.html
http://pandas.pydata.org/pandas-docs/stable/groupby.html
https://databricks.com/blog/2015/08/12/from-pandas-to-apache-sparks-dataframe.html
https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html
https://forums.databricks.com/questions/7894/random-forest-regression-with-categorical-variable.html
https://github.com/justmarkham/kaggle-allstate/blob/master/allstate-paper.md