Stats 141 Final Project: DataFest 2015

Jiahao Huo, Minjie Xia

March 12, 2018

Abstract

The goal of this project is to answer a key question that the website *Edmunds.com* is facing: "Will this customer buy or not after he/she express interests on website by leaving contact information?" This project utilized five datasets provided by *Edmunds*: Visitor, Shopping, Configuration, Leads and Transactions. After applying data cleaning, features engineering, and feature selections on the datasets, we split the data into training and testing, and performed a binary logistic regression with XGBoost on the training data. To optimize the performance of the model, we applied random search to tune the hyper-parameters, including eta, max depth and nrounds. Then, the tuned hyper-parameters were used on the testing data, and the accuracy of the model was calculated to be 86.8%. Correspondingly, relevant graphs were plotted to better visualize the results of the model.

1 Introduction

For a car dealer like *Edmunds.com*, one of the most pertinent question is that "if a customer leaves information on the website for a particular car, is he/she going to buy the car?" Therefore, the main aim of this project is to answer this question by building a binary logistic model based on the datasets provided by *Edmunds*. In this way, *Edmunds* will be able to know how likely a certain customer will buy cars by referring to our model, and thus decide if to pursue this customer or not. The datasets manipulated in this project includes Visitor, Shopping, Configuration, Leads, and Transactions. The methodology of data manipulation and detailed analysis of the data are in the next sections.

2 Methodology

2.1 Data Cleaning and Feature Engineering

The first step of the data cleaning procedures is to create a response variable for the predictive model. The y variable was binary such that: customers who both left contact information and bought cars are 1s, ones who only left contact information but did not bought cars are 0s.

Then, we selected key features from the Leads dataset based on our prior understanding of the variables and its relationship with car-buying. The selected features include “dealer_distance”, “model_year”, “make”, “model”, and “style”, among which the last four are transformed to be binary to indicate if there is a value for each observation.

Next, 12 new features were engineered from the Leads, Shopping, Configuration and Transactions datasets. The first feature is called “contactinfo_n”. This variable represents the number of times a certain customer left information on the website. The second feature is “shoppingdate_n”, which shows how many days a customer viewed the website. The third new feature is also a “counting” one, called “diffcar_n”. This feature counts how many different cars a customer viewed.

The next two features are related: “bcarview_n” and “bmakeview_n”. One represents the number of times a customer viewed a car which he/she eventually bought, the other indicates the number of time a customer viewed a make which he/she eventually bought.

The sixth new feature, “singleday_max” is the maximum number of webpages a customer has gone through on a single day.

The next two new features discuss the role of time in the datasets. “year_diff” shows the difference in terms of year between the year when customers left their contact information and the year of the model they were interested in. “leads_month” indicates which month customers left their information.

The features,”impinfo_n” and “lessinfo_n”, show how specific a customer’s requirements for a car were when leaving information. The requirements are model year, make, model, style, body type, trim, interior color, exterior color, interior fabric color, fuel, engine, and transmission. We divided these requirements into two categories: important and less important. We thought that if a customer is determined to buy a car, he/she usually knows what model year, make, model and style of cars he/she wants. Therefore, we grouped these features to be important requirements, and the rest to be less important ones. Then, for each category, we counted the number of requirements requested by a certain customer.

The next features are “new”, “old”, and “cpo”. As the names indicates, these features are binary and show whether a certain customer wants new or old or cpo cars. Similarly, the next features, “ppfY” and “ppfN”, are also binary and show that whether a particular customer wants cars with price promise or not.

For the next batch of features, “top10”, “top20”, “top30”, “top40”, and “top50”, we imported external data to help us creating new variables. The external data used is the state ranking of vehicle per thousand people from Wikipedia[1]. A map representation of the ranking is shown to be Figure 1. These variables are again binary, and indicate a certain customer is from a state that is in the top 10 of the ranking or not, etc.

Then, we examined the new features engineered from the 4 datasets, and eliminated the ones that has more than 80% of missing values. The remaining new features are "year_diff", "leads_month", "impinfo_n", "lessinfo_n", "new", "old", "cpo", "ppfY", "ppfN", "top10", "top20", "top30", "top40", and "top50".

For the fifth dataset, Visitor, it has the most detailed information for each visitor. The data has 203 variables and 771382 unique observations. Before cleaning it, we first observed the data and studied all the features, and picked those which did not have a lot of missing values and may have some connections with the response variable. After that, we separated them into four groups.

Group one contains the features that have great influence on the response variable by checking the odds ratio between each feature and the response variable, if the odds ratio is greater than one, which means that the odds for buying a vehicle after leaving contact information at *Edmunds.com* compared to not buying has a higher probability of having this feature.

Group two contains the features that have more influence on the response variable after we turned them into binary by checking the odds ratio between each feature and the response variable, which means if a group-two feature has more than one record for a visitor, this feature will be transferred into 1, otherwise, it will be transferred into 0.

Group three contains categorical features. After data observation and consideration of code running efficiency, we decided to make each category as a new binary variable, which means that if one visitor falls into a category of one categorical feature, then the new binary feature of that category will be marked as 1, otherwise, marked as 0.

Group four contains continuous features. For the modeling purpose and code running efficiency, we removed some rare cases for the training dataset by omitting some observations that 99% population had less records, but for the testing dataset, to keep everything real, we skipped this process. After that, we quantified them into four quantiles by population (0-25%, 25%-50%, 50%-75%, 75%-100%) and made each quantile as a new binary variable, which means that if one visitor falls into a quantile of one continuous feature, then the new binary feature of that quantile will be marked as 1, otherwise, marked as 0. A pie chart is used to illustrate the quantified age groups in the dataset (Figure 2).

Excluding visitor key, we subtracted and created 113 features. After merging with other features, the final dataset contains totally 137 features.

2.2 Modeling

After consideration and our former projects' experience, we believed tree based model such as XGBoost would have a better performance than other simple models. For tuning efficiency and laptop loading limitation, we first used chi-square statistic to filter features because a chi-square statistic is a measurement of how expectations compare to results, and we selected those have scores greater than 0, which reduced the dimension of our datasets to number of observations by 47 features (excluding visitor key and the response variable).

First, we created a task, specified its data source and target column, and excluded the variable visitor key from all further model fitting and evaluation. In the binary classification, it by default selects the first factor level of the target variable as the positive class. Then we standardized all the numeric features in the data. Data standardization is the critical process of bringing data into a common format that allows large-scale analytics like our data. Some features like

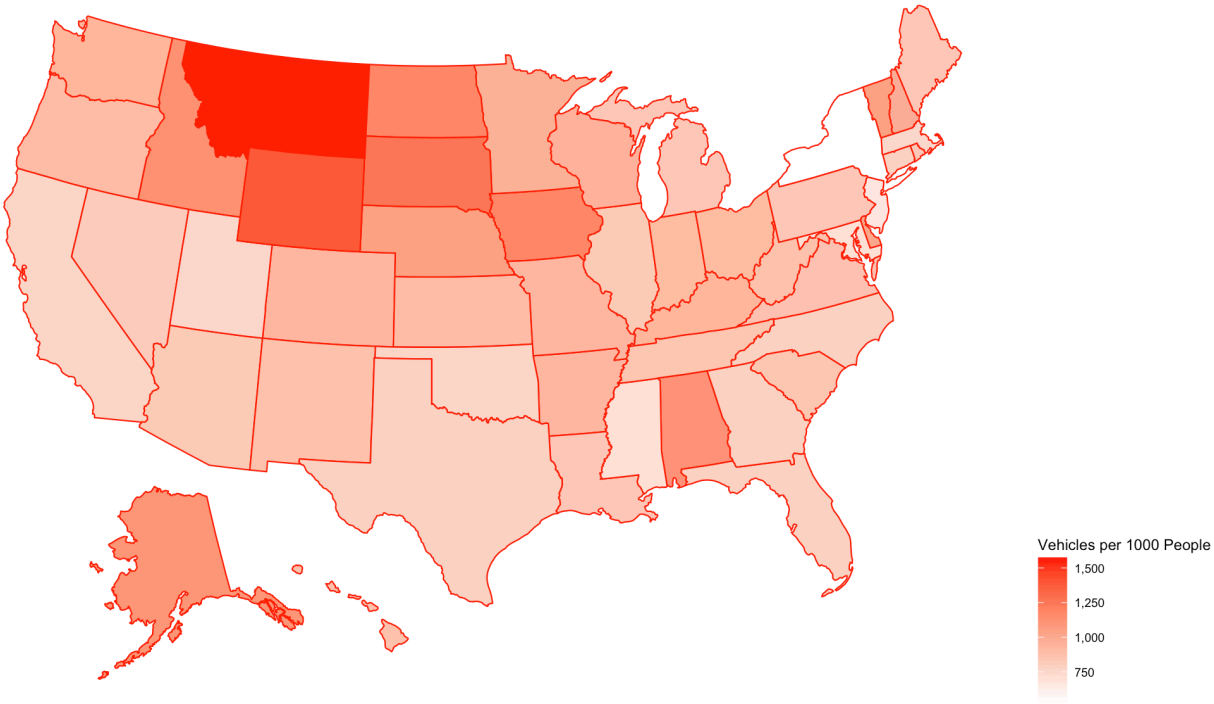

Figure 1: Map of Each US State's Vehicles per 1000 People

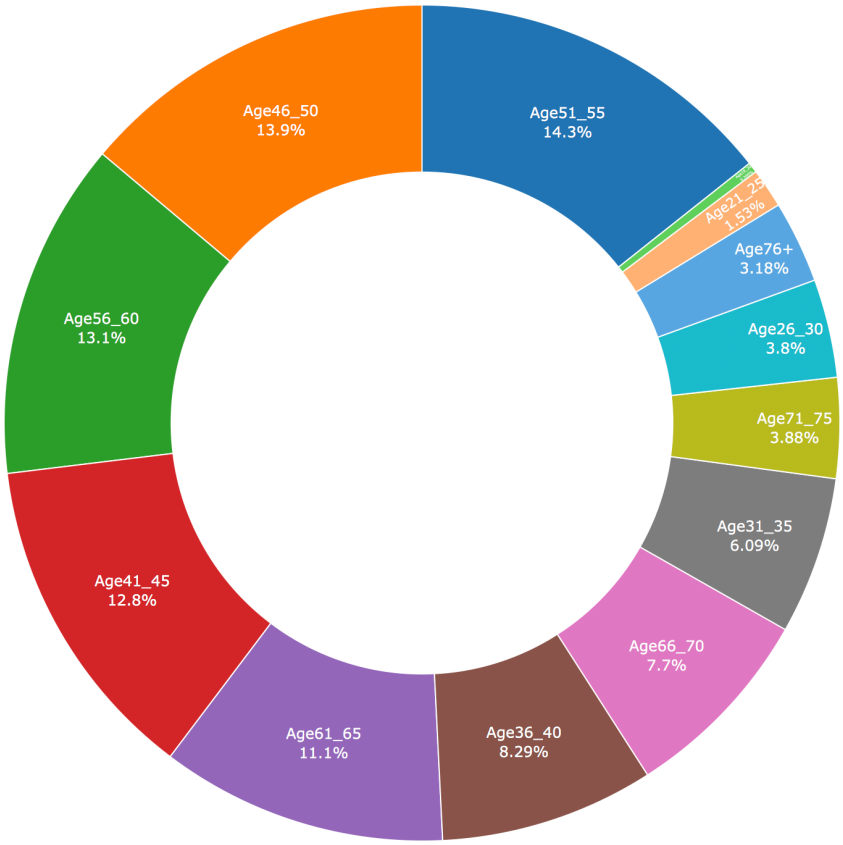

Figure 2: Pie Graph of Age Distribution of Edmunds Customers

“dealer_distance” is highly skewed, so standardizing the data will help improve the model. After that we created XGBoost learner and set the predict type to probabilities because logistic regression was used to find the best conditional probability fit of the data.

During hyper-parameter tuning process, we used random search for parameter combinations. To maximize working efficiency, we activated the parallel computing capabilities for local multicore execution. Among all the parameters, we selected three of them to do hyper-parameter tuning, and the others stay the default setting. The first one is “eta”, which shrinks the feature weights to make the boosting process more conservative. We choose it because we have 47 features, which may cause overfitting problem. The second one is “nrounds”, which is the number of rounds for boosting. As “nrounds” increases, the model will be enhanced by further reducing the difference between ground truth and the prediction. However, because the way boosting works, there is a time when having too many rounds lead to an overfitting. The third one is “max_depth”, which is the maximum delta step we allow each tree’s weight estimation to be. We choose it because it might help in logistic regression when class is imbalanced.

Then we set the seed to 1 for replicating the results and used 3-fold cross validation to measure improvements. It will randomly partition the training dataset into 3 equal sized subsamples. The advantage of this method over repeated random subsampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. We used “AUC” as the evaluation matrix because it works better when facing a probabilistic, binary classifier such as logistic regression. After 100 times of tuning, we got the optimal hyper-parameters with the highest AUC value (eta = 0.0608, max_depth = 9, nrounds = 376).

3 Analysis and Interpretation

After we trained the model on the training dataset, we tested the optimal hyper-parameters out on the testing data. Since the output of binary logistic model are probabilities, we assigned those with probabilities larger than 0.5 to be 1, that is customers that will buy cars, and those with probabilities lower than 0.5 to be 0, that is customers who will not buy. Then, we compared the predicted results to the original response values and constructed the confusion matrix.

Actual/ Predicted	**0**	**1**
0	1579183	241
1	239916	157

Table 1: The Confusion Matrix

According to the confusion matrix, the accuracy of the model is 86.8%, and correspondingly the misclassification rate is less that 13%. More specifically, the false positive rate of our model is very low: 0.015%, and the specificity rate is extremely high, meaning that the model is highly good at detecting customers that do not buy cars.

Although the true positive rate for the model is not high, it is largely because the prevalence in the data is low: there is only approximately 13% of people bought cars. This low percentage of

buying population in the sample may skew the model, and consequently resulted in a high specificity rate and a low true positive rate.

After finishing modeling, XGBoost package has the built-in function to measure the feature importance, which returns a sheet contains Feature name, Gain, Cover and Frequency. The column Gain provides the information we are looking for because features are classified by it. Gain is the improvement in accurate brought by a feature to the branches it is on. The idea is that before adding a new split on a feature X to the branch there was some wrongly classified elements, after adding the split on this feature, there are two new branches, and each of these branch is more accurate.

Feature	Gain	Cover	Frequency	Importance
dealer_distance	0.419	0.560	0.469	0.419
year_diff	0.070	0.075	0.087	0.070
lessinfo_n	0.058	0.033	0.068	0.058
credit_Excellent	0.031	0.017	0.027	0.031
auto_lease_calc	0.025	0.018	0.014	0.025
buy_guide_carrev	0.024	0.007	0.021	0.024

Table 2: Importance Table

The plot of feature importance provides a better visualization (Figure 3). The graph represents each feature as a horizontal bar of length proportional to the importance of a feature. Features are shown ranked in a decreasing importance order. As you can see from the graph, “dealer_distance” is extremely important to the model, which makes sense for us because distance to a dealer store has a strong influence on whether buying car online or directly through a dealer. Next one is “year_diff”, which also makes sense because some make and model will only be sold online, while some others will only be found in a dealer store. The difference in terms of year between the year when customers left their contact information and the year of the model they were interested in may be a determinant of whether buying car online or directly through a dealer. “lessinfo_n” is placed on the third place. The higher value “lessinfo_n” has, the more specific less important information a customer leaves during his car searching. Most of the customers will not fill out the less important information like color, fuel, engine or transmission, so those who specify what kind of car they want may have a higher chance of getting a car.

Due to the highest importance “dealer distance ” has, we observed it again through a histogram (Figure 4). As we can see from the histogram, it is highly skewed. 93.1% of the customers who leave their information on Edmunds.com lives within 50 miles from their interested dealers. But customers who lives within 5 miles from a dealer have less information records than customers live 5 miles away, which makes this an interesting observation. Customers live nearby dealers will prefer shop directly in a dealer store to shopping online. A graphic visualization of this buying pattern is shown as Figure 5 to illustrate our points. This graph is an example showing the dealer distance from one dealer to customers. As can be seen, the red dots are more tense when close to the dealer.

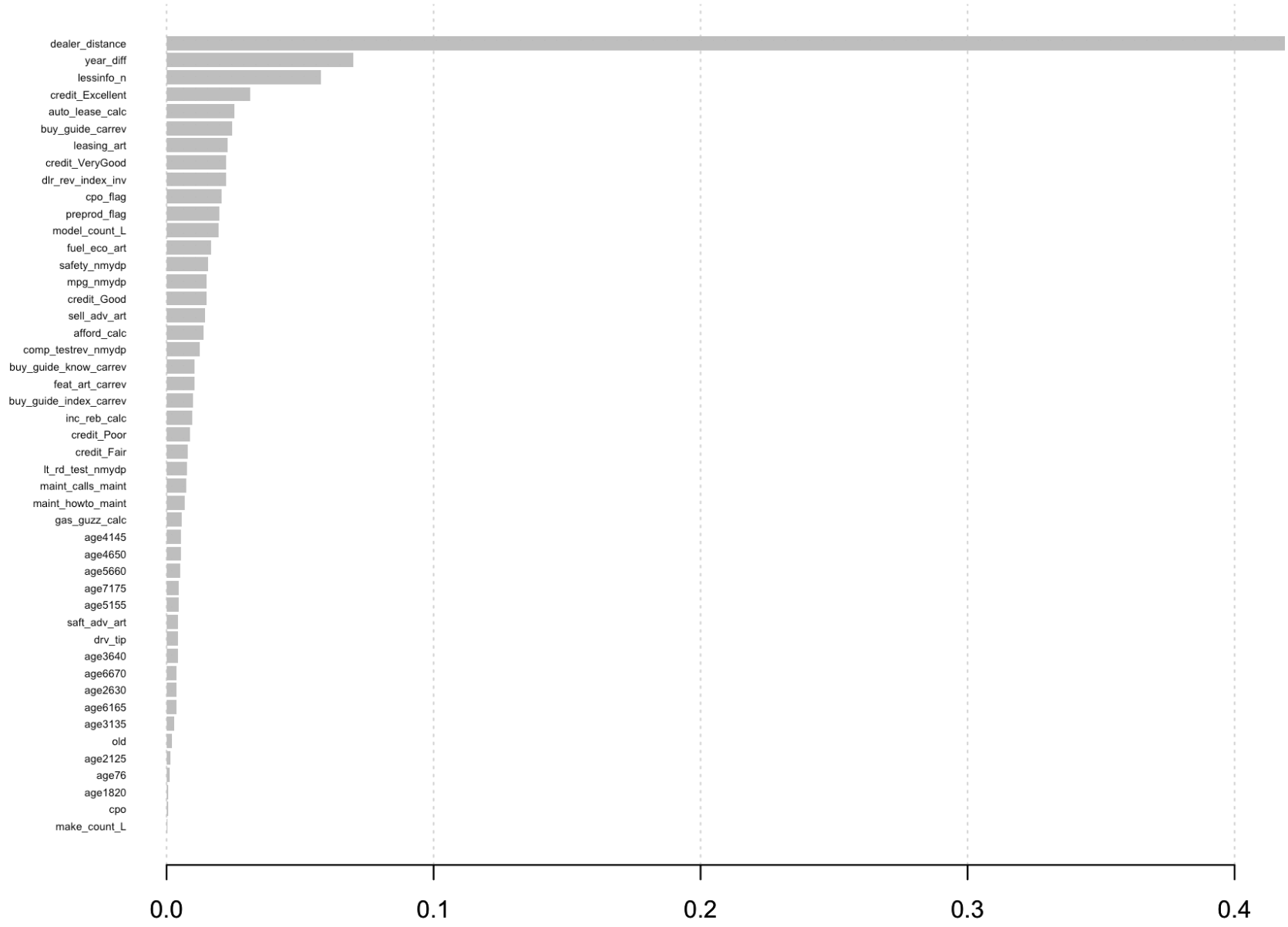

Figure 3: Importance Plot

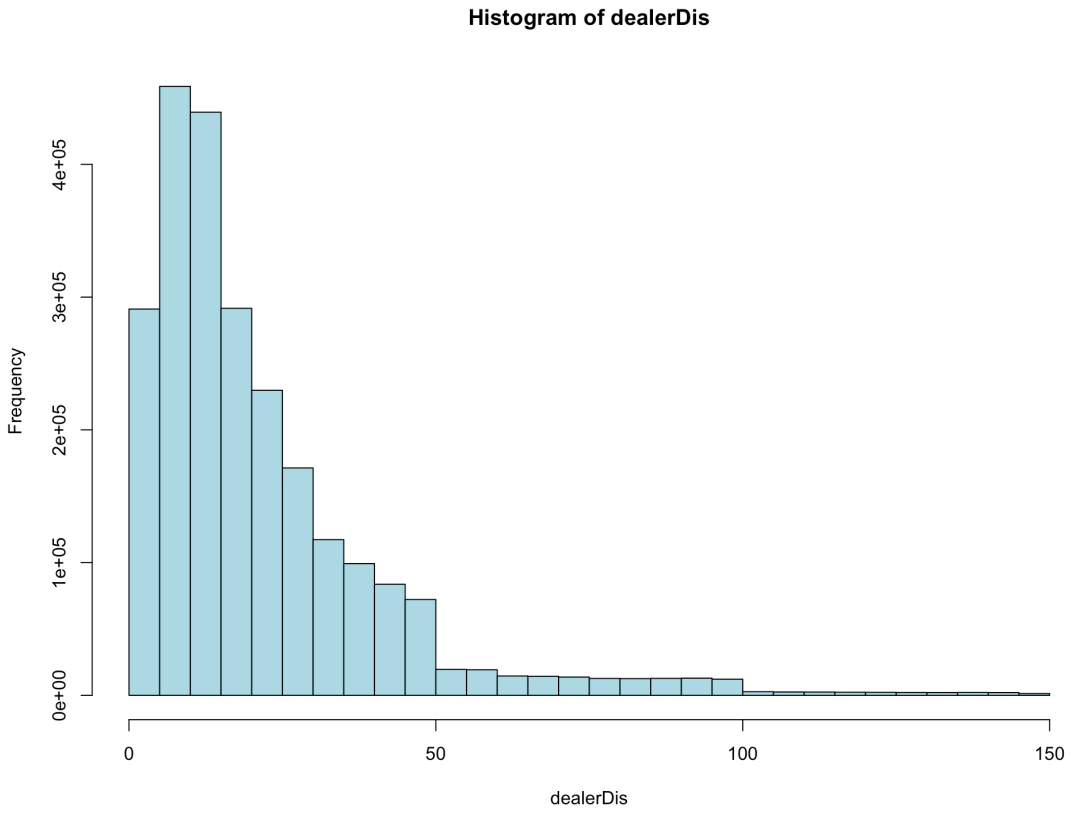

Figure 4: Dealer Distance Histogram

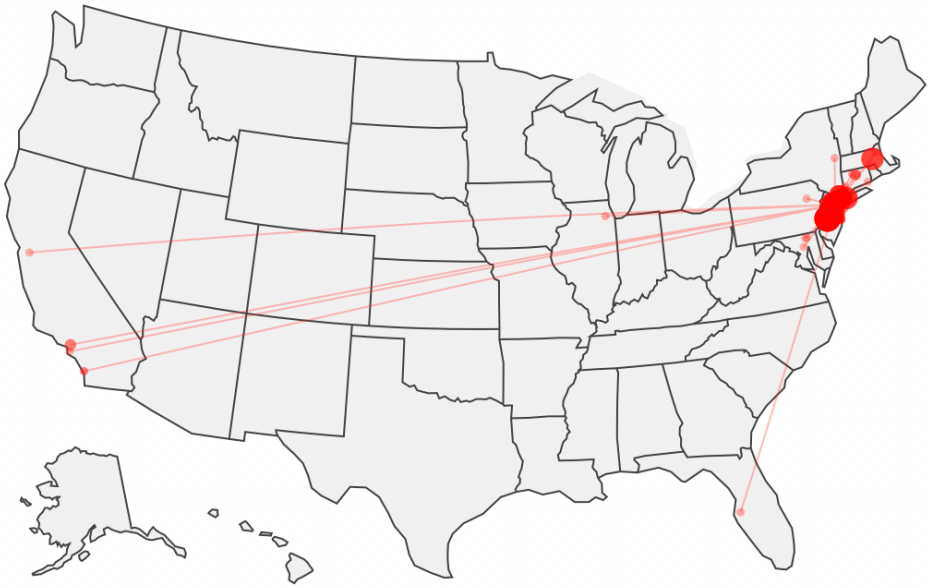

Figure 5: Map example of Dealer distance

4 Conclusion

For this project, we used the datasets provided by *Edmunds.com* and performed a binary logistic regression with XGBoost to answer a pertinent question that: "if a customer leaves information of website for a particular car, is he/she going to buy the car?" For the data cleaning part, we engineered 113 features from the original datasets. Majority of them were made to be binary variables to empower the machine learning algorithm. Then, we selected other 24 variables from the original datasets. In total, we have 137 features for machine learning. Then, we removed some of the features when they have 80% missing values or 0 scores for Chi-Square test. There are 47 features remained.

With these 47 features, we performed binary logistic regression with XGBoost. The most crucial part of the modeling process is hyper-parameter tuning. To avoid overfitting and under-fitting of the data, we tuned eta, max-depth and nrounds with random search. Then, the optimal parameters were used to test the testing data. The accuracy of our model is as high as 86.8%.

Then, we explored the feature importance of the model. We found out that dealer distance has a huge impact on the outcome that if customer will buy cars or not. Through graphs, we learned that when the dealers are close to customers, they are more likely to buy cars.

Reference:

[1] "Wikipedia" "*List of U.S. states by vehicles per capita.*" N.p., n.d. Web. Feb. 4 2018. https://en.wikipedia.org/wiki/List_of_U.S._states_by_vehicles_per_capita