Uber's customer churn prediction and analysis

EE 660 Project Type: Individual

Haoyu Lan, haoyulan@usc.edu
12/1/2018

1. Abstract

Customers' churn is the loss of customers for some service provider companies. Through analysis of the customers churn, people could find existed problems and solve them. In this classification project, churn and active are two outputs I focused on. I used 'Logistic Regression Classifier', 'KNN Classifier', ''Random Forest Classifier', 'Gradient Boosting Classifier' to build the classification models; used roc_auc_score to choose the best model for each classifier; used auc, recall, f1 score, accuracy to choose the best classifier. I analysis the pros and cons for each classifier and choose the best one for this problem. For the classifier which could output the feature importance like logistic regression, random forest and gradient boosting, I analyzed the feature importance and combine some dataset previews to understand the feature influence for customers' churn.

2. Introduction

2.1. Problem Type, Statement and Goals

This project uses the dataset containing users' information and activities. My goal is to find the relation between these users' information and activities, then to predict whether the specific user will leave the platform in the future. This is a classification project, 'churn' and 'active' are two classes I want to predict.

Customer churn is the loss of customers or clients, which is an important business metric in the service provider companies. Since the cost of retaining an existed customer is much less than attracting a new customer. If the prediction model is useful than the companies could use some business methods to retain the users who are likely to churn in the future. That is good for each other. On the other hand, customer churn prediction model is useful when the churn could be controlled by the customers, some features like location change, health situation cannot be used to build the model. Customer churn is relevant to many features that have non-linear behaviors. Sometimes, features in these problems cannot be used directly and need some cleaning and preprocessing. Thus, customer churn is a valuable and useful information which has to be predicted by modeling.

2.2. Literature Review (optional)

2.3. Prior and Related Work - None

2.4. Overview of Approach

I've used 'Logistic Regression Classifier', 'KNN Classifier', ''Random Forest Classifier', 'Gradient Boosting Classifier' to build the models, and used performance metrics like 'accuracy score', 'roc-auc score', 'precision score', 'recall score', 'f1 score' to compare between different classifier. Finally, I used model with the best performance. I also visualized the feature importance for different methods to find some useful features.

3. Implementation

3.1. Data Set

This dataset contains the information about the users who signed up between 2014-01-01 and 2014-01-31. Information is recorded from 2014-01-01 to 2014-07-01, half a year, without churn information. I have to create an output variable 'churn' using the feature 'last_trip_date'. I supposed the users whose last_trip_date is before 2014-06-01 are regarded as churned customers. Since users don't use uber in one month could be regarded as inactive users which cannot make profits for uber, we can mark these users as churned customers.

In the original dataset, there are 12 features, followings are some feature explanations:
'avg_dist': the average distance (in miles) per trip taken.
'avg_surge': the average surge multiplier over all of this user's trips. High surge multiplier means more extra money has to be paied for a trip.
'surge_pct': the percent of trips taken with surge multiplier > 1. The higher surge multiplier, the more cost has to be paid for the trip.
'weekday_pct': the percent of a user's trips in weekday by all this user's trips.
'city': the city a user signed up in.
'avg_rating_by_driver': the average rating of a rider by drivers.
'trips_in_first_30_days': the number of trips per user took in the first 30 days after signing up

Following is the original dataset information:

Number of data points	***Number of input variables***	***Number of output variable (extracted from an input variable)***
50000	12	1

Following is the features information in the original dataset:

Feature names	***Feature type***	***Range/cardinality***	***Number of input variables***
avg_dist	numeric	0 - 160.96	numerical variable (7)
avg_rating_by_driver	numeric	1 - 5	
avg_rating_of_driver	numeric	1 - 5	
avg_surge	numeric	1 - 8	

weekday_pct	numeric	0 - 100	
surge_pct	numeric	0 - 100	
trips_in_first_30_days	numeric	0 - 125	
signup_date	datatime	2014-01-01 2014-1-31	timestamp (2)
Last_trip_date	datetime	2014-01-01 2014-07-01	
luxury_car_user	boolean	True, False	categorical variable (3)
phone	string	iPhone, Android	
city	string	King's Landing, Astapor, Winterfell	

Following is the output variable information:

Output variable name	***Output variable type***	***Range/cardinality***	***Number of output variable***
churn	binary	1(churn), 0(active)	Categorical variable (1)

3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

a) missing values

The original dataset has missing values at three features:

Feature names	avg_rating_by_driver	avg_rating_of_driver	phone
Number of missing values	201	8122	396

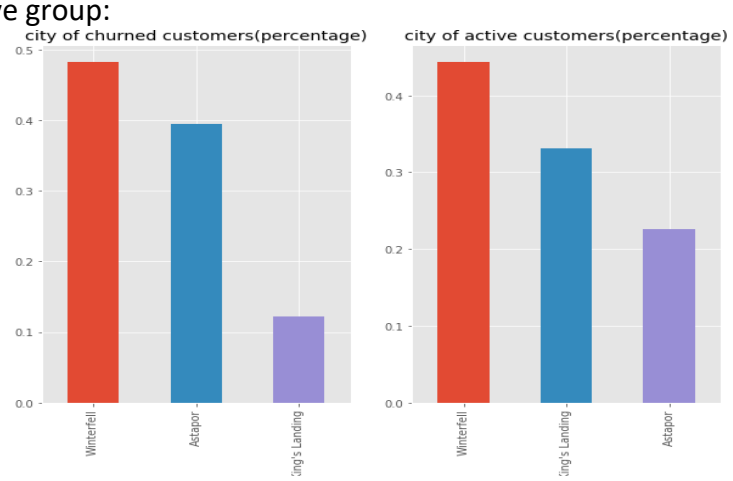
The first two features 'avg_rating_by_driver' and 'avg_rating_of_driver' are numerical variables. Thus, I used median of the training set to fill the missing values in training set, validation set and test set. Since mean of features could be influenced by some extreme values, median is robust to the extreme values; As for the third feature 'phone', this is a categorical variable. I filled the missing values with another category 'no_phone'.

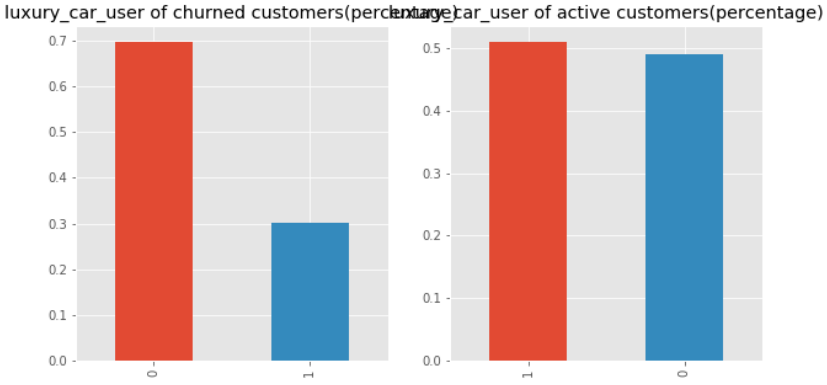
b) data manipulation

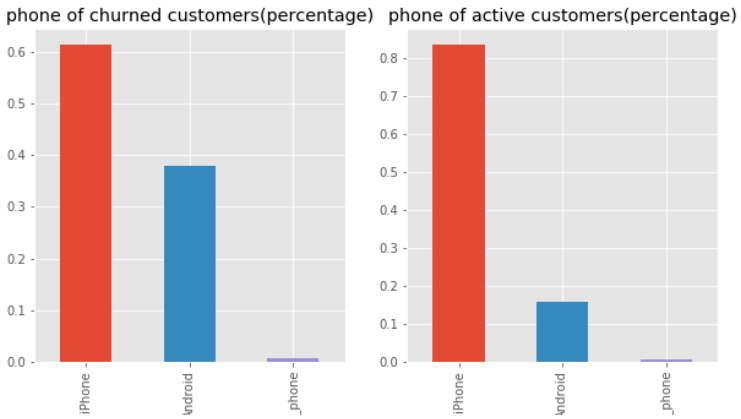
1) I converted the timestamp feature 'signup_date' to categorical features as 'day of week' to plot the bar chart.
2) I plotted histograms for the numerical features and bar charts for the categorical features to visualize the distributions and categories ratio according to the churn and active groups. I compared the distributions and bar charts to find some features which are

distributed differently in the churned group and active group to find some useful features.

I found features ‘city’, ‘phone’ and ‘luxury_car_user’ have obvious difference in the churn and active group. Followings are some features that have obvious different behaviors in churn and active group:

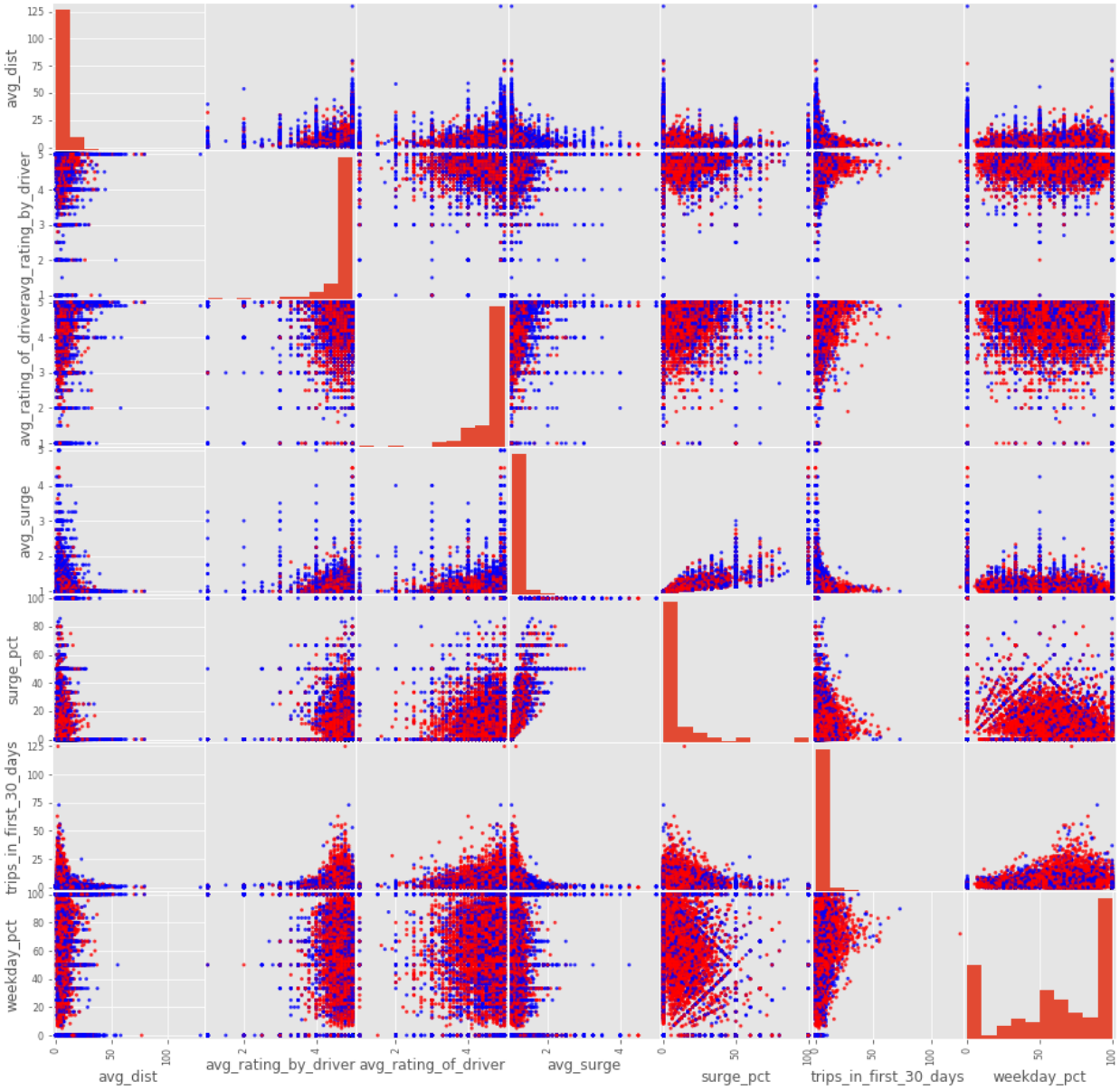




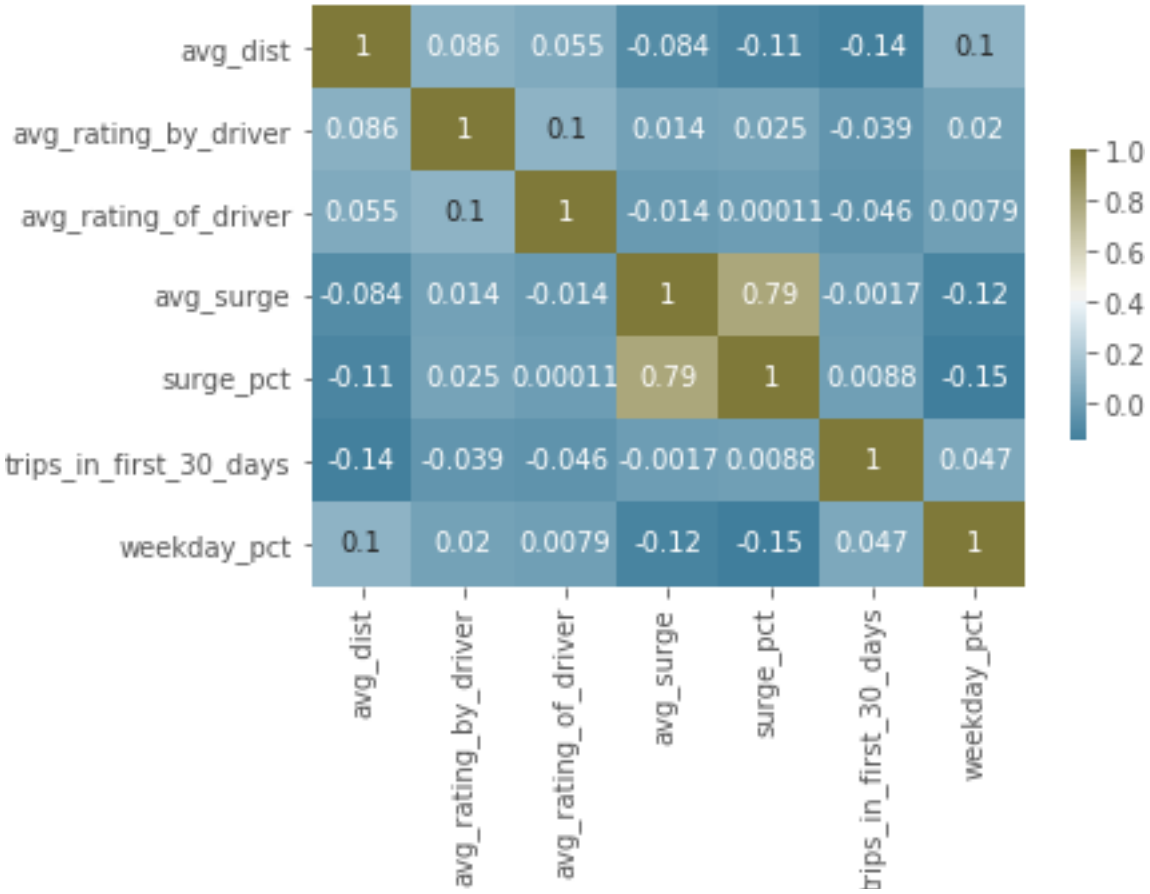


3) I plotted covariance matrix and scatter matrix for the continuous features to see if there are some correlations between features. In this project, correlations could be cured by regularizations in the logistic regression and tree-based methods are robust to correlations.

Following is the scatter matrix for continuous variables, I found ‘avg_dist’ vs ‘trips_in_first_30_days’ plot is the one could find the decision boundary easily:



Following is the correlation matrix, I found features ‘surge_pct’ and ‘avg_surge’ have strong correlation, which could be cured by regularization:



4) I recast categorical variables by ‘one-hot-encoding’.

After data manipulation:

The number of features	The number of outputs
15	1

c) standardization

Since building Logistic Regression Classifier and KNN Classifier needs to scale the features, I standardized the features before building these two models using StandardScaler in the library sklearn.preprocessing. For the tree-based classifier, there is no need to scale features. When watch the feature importance, I still need to standardize the features for the tree-based classifiers.

3.3. Dataset Methodology

a) dataset usage procedure

1) Split the original dataset into training set, validation set and test set.
2) Do some visualization and overview on the training set to check if there are some outliers or noise.
3) Do feature imputation and preprocessing including feature selection through the visualization and intuition.
4) Build machine learning models on training set and use 10-fold cross validation to tune the hyperparameters for each supervised learning model and choose the model with best roc-auc score.
5) Use several performance metrics: 'accuracy score', 'roc-auc score', 'precision score', 'recall score', 'f1 score' to do the model selection on the validation set to get the best model between different supervised learning models.
6) Build the best supervised learning model on the test set to get the final result.

b) dataset information

Dataset	Training set	10-fold cross validation set	Validation set	Test set
Number of points	27000	3000	10000	10000

c) 10-fold cross validation

Use 10-fold cross validation to tune the hyperparameters for each model, which is step 4. Since I used GridSearchCV in sklearn.model_selection which finds all the possible combinations of parameter values. For each combination of parameters, splits the training set into 10 folds and iterates the following process 10 times to get average score, every time let a fold as validation set:

- trains the model on 9 folds datasets.
- predicts on the left one validation set to get the score.

The model with highest average score is retained. Thus, cross validation results are used to find the best hyperparameters for each supervised learning classifier and cure the overfitting.

d) usage of test set

Test set was just used once at step 6, after I got the best model which has the highest performance scores. Test set is used to get the final result.

3.4. TrainingProcess

In this project, I used four classifiers to train and predict: Logistic Regression Classifier, KNN Classifier, Random Forest Classifier, Gradient Boosting Classifier.

a) Logistic Regression Classifier

1) Library used

sklearn.preprocessing.StandardScaler
sklearn.linear_model.LogisticRegression
sklearn.model_selection.GridSearchCV

2) Algorithm
   - Logistic regression maps the feature space onto probability using sigmoid function:
   $$h_\theta(\underline{x}) = sigm(\underline{\theta}^T \underline{x}) = \frac{1}{1 + e^{-\underline{\theta}^T \underline{x}}}$$
   - In this project, there are two classes: churn$(y = 1)$, active$(y = 0)$. Using MLE gets the likelihood function:
   $$p\left(\underline{y} \middle| \underline{x}, \underline{\theta}\right) = \prod_{i=1}^{27000} h_\theta\left(\underline{x_i}\right)^{y_i} \left(1 - h_\theta\left(\underline{x_i}\right)\right)^{1-y_i}$$
   - Get the objective function (NLL): $-\log p\left(\underline{y} \middle| \underline{x}, \underline{\theta}\right)$
   - Minimize the objective function through Gradient Descent to get the best parameter $\underline{\theta}: \theta_j = \theta_j - \frac{\alpha}{27000} \sum_{i=1}^{27000} \left(h_\theta(\underline{x_i}) - y_i\right) * x_{i(j)} \qquad j = 1, \dots, 15$
   - Adding the regulation could reduce overfitting and solve the features collinearity. L1 regulation could make feature selection automatically.

3) Justification
   Logistic Regression is the supervised learning model which has the good interpretability and easy to train. I could get feature importance after training, this could be a reference to understand the features influence. Since it doesn't have hyperparameters to tune (no regularization situation), the training speed could be very fast. However, it cannot handle non-linear features, which means the performance cannot as good as other models.

4) Procedure
   - First, use the features gotten by the preprocessing.
   - Second, use StandardScaler to standardize the train set, since logistic regression and regularization are sensitive to the feature with larger scale.
   - Third, use LogisticRegression and GridSearchCV to tune the hyperparameters to get the best parameters (the model has best roc-auc score).
   - Get the feature importance and analysis.

5) Parameters
   Since I use the regularization in the logistic regression, there are two parameters could be tuned by GridSearchCV:

Parameter name	***Range***	***Number of parameters***	***Meaning***	***Best parameters***
C	0 - 1	10	Inverse of regularization strength	0.9
penalty	L1, L2	2	the norm used in the penalization	L1

6) Complexity of hypothesis set
   Since there are two hyperparameters, the complexity of hypothesis set is 10*2 = 20:

Complexity of hypothesis set	*Training set data points*	*Dimension of feature space*
20	27000	15

Using cross validation and adding regularization are the ways to prevent the overfitting. Through changing parameter 'C' could control the strength of regularization. Since logistic regression is a linear classifier and not complex enough to hand the non-linear dataset, thus, there could be a little bit underfitting. I used the other complex model to solve this problem.

7) Results
Since the good interpretability of logistic regression, I get the feature importance score after building the model. Then I compare the first three highest score features: 'city', 'phone', 'luxury_car_user' with the corresponding bar charts in the preprocessing step. The comparation results made sense, since these three features have very different distributions between churned customers group and active customers group.

b) KNN Classifier

1) Library used
sklearn.preprocessing.StandardScaler
sklearn.neighbors.KNeighborsClassifier
sklearn.model_selection.GridSearchCV
2) Algorithm
   - Knn classifier is a distance-based classifier. Euclidean distance is the default measurement.
   - Set the number n of neighbors used to make the majority poll.
   - Iterate for each data point in the cross validation set to make majority poll by n closest data points in the training set.
3) Justification
Knn classifier is an easy model which could predict directly on data, no need of training, and could handle the non-linear features. One disadvantage is that prediction speed depends on the size of dataset.
4) Procedure
   - First, use the features gotten by the preprocessing.
   - Second, use StandardScaler to standardize the train set.
   - Third, use KNeighborsClassifier and GridSearchCV to tune the hyperparameyers to get the best parameters (the model has best roc-auc score).

5) Parameters

Parameter name	***Range***	***Number of parameters***	***Meaning***	***Best parameters***
n_neighbors	{5,10,15,20,25,30,35,40}	8	Number of neighbors to use	25
leaf_size	{5,10,15,20,25,30}	6	affect the speed of the construction and query	5

6) Complexity of hypothesis set

Complexity of hypothesis set	***Training set data points***	***Dimension of feature space***
48	27000	15

When n_neighbors is small, the model could overfitting easily, thus, I used GridSearchCV and more n_neighbors to prevent overfitting.

c) Random Forest Clasifier

1) Library used
sklearn.preprocessing.StandardScaler
sklearn.ensemble.RandomForestClassifier
sklearn.model_selection.GridSearchCV
2) Algorithm
   - Set the number n of trees will be grown.
   - Use bootstrap to create n datasets which will be used to grow the trees.
   - Grow the tree on one bootstrap dataset each time. At each split node, choose the best pair of cut off and feature in m features which are chosen by a random subsample of the entire feature space p (in classification problem, usually choose $m = \sqrt{p}$).
   - Use the majority poll of n trees for a data point to be the final result of one data point.
   - Calculate the out-of-bag error for every single tree, then get the average out-of-bag error among n trees.
   - Since random forest grows n deep trees, then calculate the average for the regression problem and majority for the classification problem, it can effectively cure the overfitting caused by the deep tree.
3) Justification
Random forest is one of the most efficient and accurate classifiers to handle the non-linear features and could get better performance than other classifiers. Random forest could reduce overfitting effectively.

4) Procedure
   - First, use the features gotten by the preprocessing.
   - Second, use RandomForestClassifier and GridSearchCV to tune the hyperparameyers to get the best parameters (the model has best roc-auc score).
   - Train on the standardized training data to get the feature importance.
5) Parameters

Parameter name	***Range***	***Number of parameters***	***Meaning***	***Best parameters***
n_estimators	300, 400, 500, 600	4	Number of trees to grow	500
max_depth	20, 30, 40, 50	4	The depth of each single tree	20
min_samples_split	10, 15, 20	3	The minimum number of samples required to split an internal node	15
min_samples_leaf	10, 25, 40, 55	4	The minimum number of samples required to be at a leaf node	15

6) Complexity of hypothesis set

Complexity of hypothesis set	***Training set data points***	***Dimension of feature space***
192	27000	15

Adjust the hypermeter and use GridSearchCV could reduce the overfitting.

d) Gradient Boosting Classifier

1) Library used
   sklearn.preprocessing.StandardScaler
   sklearn.ensemble.GradientBoostingClassifier
   sklearn.model_selection.GridSearchCV
2) Algorithm
   - Use exponential loss function: $L\big(y, f(x)\big) = \exp\big(-y * f(x)\big)$.
   - Initialize model with a constant value r: $F_0(x) = argmin_r \sum_{i=1}^{n} L(y_i, r)$.
   - For m=1 to M:
     i. Compute so-called pseudo-residuals:

$$r_{im} = \left[\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1(x_i)}} \quad for\ i = 1 \ldots\ldots 27000$$

     ii. Fit a base weak learner $h_m(x)$ to pseudo-residuals.

iii. Compute the multiplier γ_m by solving one-dimensional-optimization:

$$\gamma_m = argmin_\gamma \sum_{i=1}^{27000} L(y_i, F_{m-1}(x_i) + \gamma h_m(x))$$

iv. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma h_m(x)$$

3) Justification
Gradient boosting is one of the most efficient and accurate classifiers to handle the non-linear features. Since it learns the pseudo-residuals using the shadow tree (weak learner), it can reduce the bias and is hard to overfitting. This model should have the best performance among all four models in this project.

4) Procedure
- First, use the features gotten by the preprocessing.
- Second, use GradientBoostingClassifier and GridSearchCV to tune the hyperparameyers to get the best parameters (the model has best roc-auc score).
- Train on the standardized training data to get the feature importance.

5) Parameters

Parameter name	***Range***	***Number of parameters***	***Meaning***	***Best parameters***
n_estimators	50 - 500(per 50)	10	The number of boosting stages to perform	360
max_depth	2, 4, 6, 8, 10, 20	6	The depth of base learner	3
learning_rate	0.1 – 1(per 0.1)	10	learning rate shrinks the contribution of each tree by learning rate.	0.1

6) Complexity of hypothesis set

Complexity of hypothesis set	***Training set data points***	***Dimension of feature space***
600	27000	15

Adjust the hypermeter and use GridSearchCV could reduce the overfitting.

3.5. Model Selection and Comparison of Results

Since churn customer is the output I care about, and False Negative makes more profit loss to the company then False Positive. Thus, recall, F1-score, auc and accuracy are the metrics I used to compare:

Model	*AUC (validation set)*	*Recall (validation set)*	*F1-score (validation set)*	*Accuracy (validation set)*
Logistic Regression	0.757	0.846	0.788	0.713
KNN	0.826	0.841	0.817	0.765
Random Forest	0.848	0.862	0.833	0.781
Gradient Boosting	0.856	0.863	0.835	0.786

Through these metrics on the validation set, I found Gradient Boosting classifier has the best performance at almost every score. Thus, I choose Gradient Boosting classifier as the final classifier.

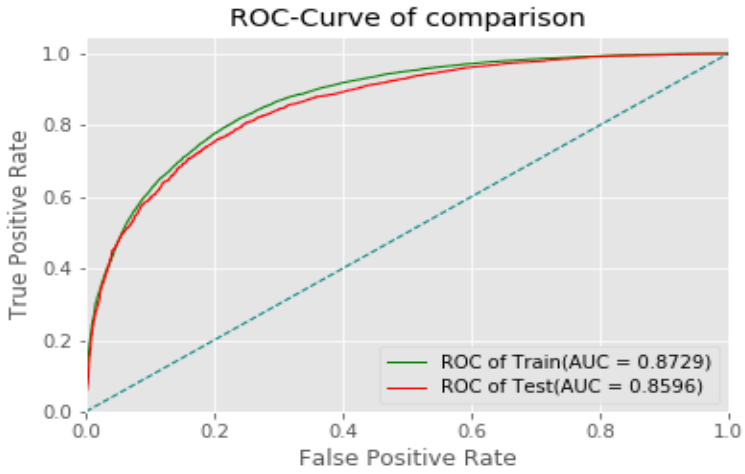
As analysis before, Logistic Regression is a simple classifier with good interpretability and cannot handle non-linear features, thus, its performance is not as good as other classifiers. KNN could handle non-linear features, however, its accuracy is not as good as tree-based classifiers and high dimension of feature space will reduce accuracy. Random Forest is focusing on decreasing the variance, since it uses the deep tree, thus, the bias is very small and has good accuracy. Gradient Boosting has the best performance, since it could reduce bias and variance at the same time using weak learner and addictive model.

4. Final Results and Interpretation

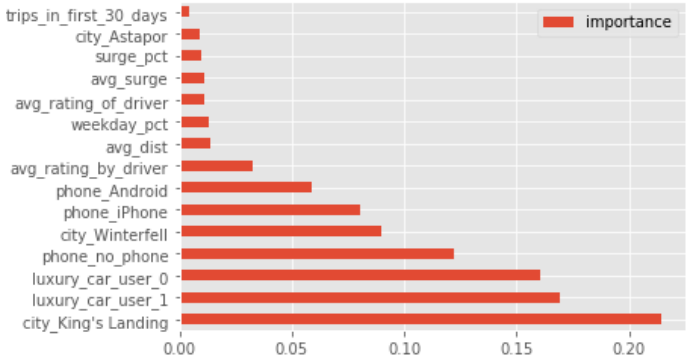
Final result:

Model	*AUC (test set)*	*parameters*	*Out of sample performance (AUC)*
Gradient Boosting	0.860	learning_rate = 0.1 max_depth = 3 n_estimators = 360	0.852 – 0.868

ROC curve:



Feature importance:



Top three important features
'city_king's Landing'
'luxury_car_user_1'
'luxury_car_user_0'

Interpretation:

Since Gradient Boosting could handle non-linear features and has the way to reduce bias and variance at the same time, thus, it should have best performance in this classification problem. Through getting the feature importance after fitting the gradient boosting classifier, I found the three most important features for the gradient boosting classifier are 'city_king's Landing', 'luxury_car_user_1' and 'luxury_car_user_0', which consists with my expectation at the preprocessing step.

5. Contributions of each team member - None

6. Summary and conclusions

Tree-based classifier can handle non-linear features better with higher accuracy, in which gradient boosting do the best. In the gradient boosting, three features are used to make decision the most, which are 'city_king's Landing', 'luxury_car_user_1' and 'luxury_car_user_0'. We could assume that these features play the important roles in the customer churn prediction.

Since the feature 'city' has the highest importance among the feature space, and city 'King's Landing' has much lower percent in the churned group than that in the active group. We could assume that the uber's service in King's Landing are better than the other two cities or this city is a good place to get profits, since people in this area are less likely to churn in the future. The second important feature is the 'luxury_car_user'. I found that in the active group people are more likely to use the luxury car than the people in the churned group, maybe the active users can afford more expensive cars and they prefer the better service.

Next, I could use more preprocessing and feature extraction technologies to extract more features and find the features could strongly influence the customer churn. Some business decisions could be made based on the customers' churn analysis.

7. References

Kevin P. Murphy, Machine Learning: *A Probabilistic Perspective* (MIT Press, Cambridge, 2012).

Yasir S. Abu-Monstafa, Malik Magdon-Ismail, and Hsuan-Tien Lin, *Learning from Data* (AMLbook.com, 2012).

T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Second Edition (Spring, 2009).