Part 3:

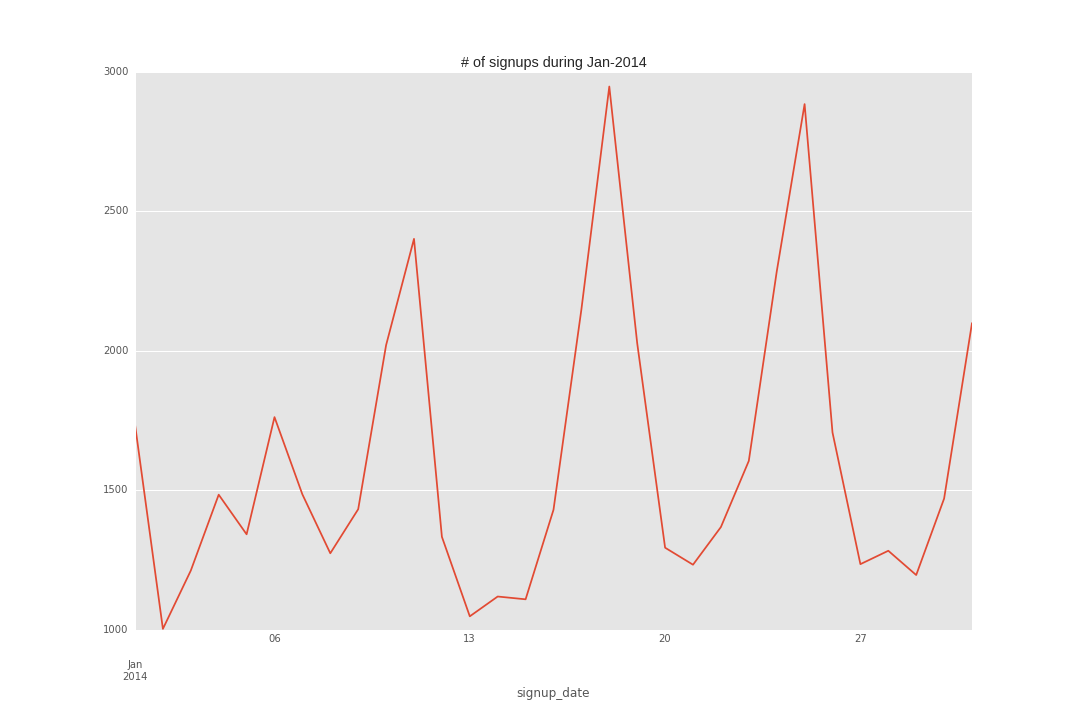
1. Exploratory Data Analysis:

In uber\_data\_challenge.json, there are 12 features and 50,000 records of users.

Except two features signup\_date and last\_trip\_date, the other 10 features are listed below:

|  |  |
| --- | --- |
| **Feature Types** | **Feature Names** |
| **Numerical** | avg\_dist |
| avg\_rating\_of\_driver |
| avg\_surge |
| surge\_pct |
| trips\_in\_first\_30\_days |
| avg\_rating\_by\_driver |
| weekday\_pct |
| **Categorical** | phone |
| city |
| uber\_black\_user |

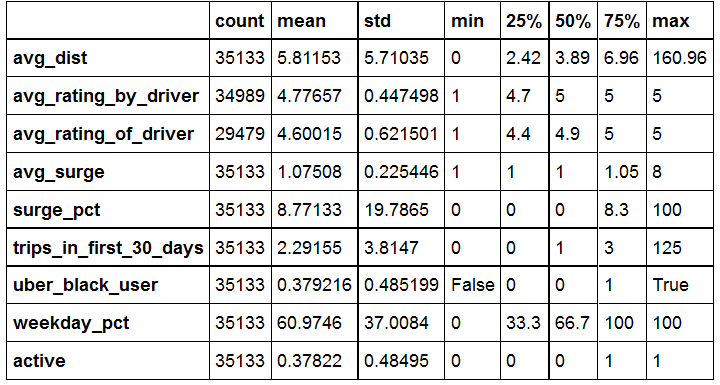
All customers signed up during 2014-01, users tend to sign up later in the week. So it might suggest that we could target ads for signups more heavily during Fridays and Weekends.

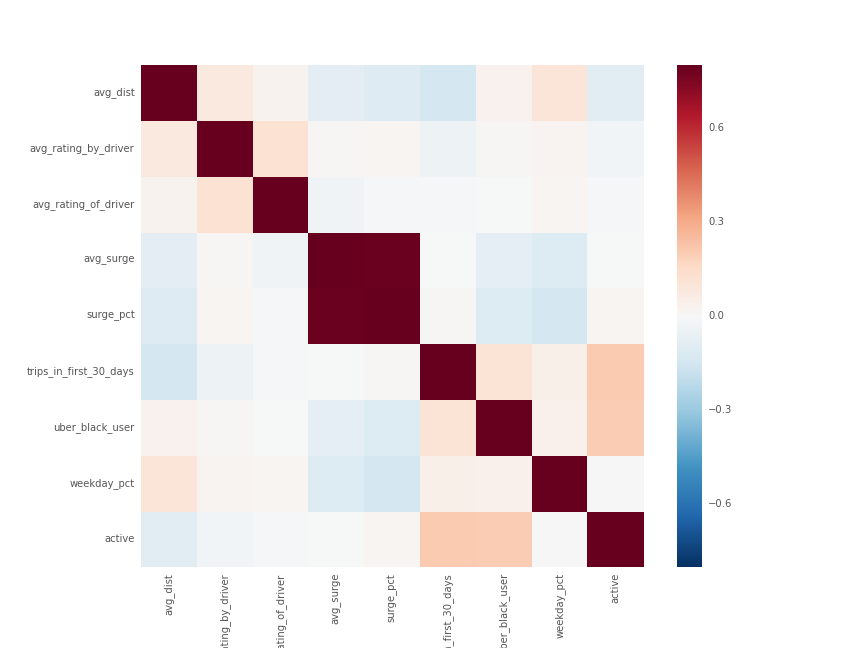


A user retained if they were “active” (i.e. took a trip) in the preceding 30 days. Within 50,000 users, **37.61%** users are active in the preceding 30 days.

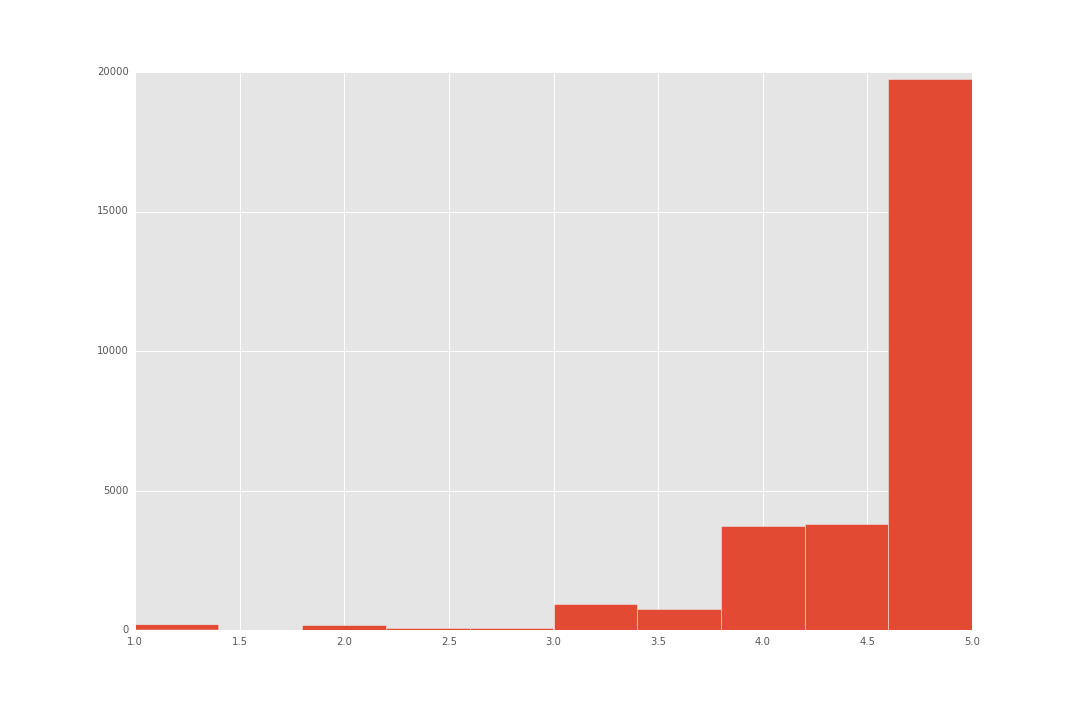
Below is a quick summary statistic table for the numerical features, first thing to note is that there is some missing values among features, but not that much, mainly for avg\_rating\_of\_driver, which might suggest we need to improve our rating experience for users.

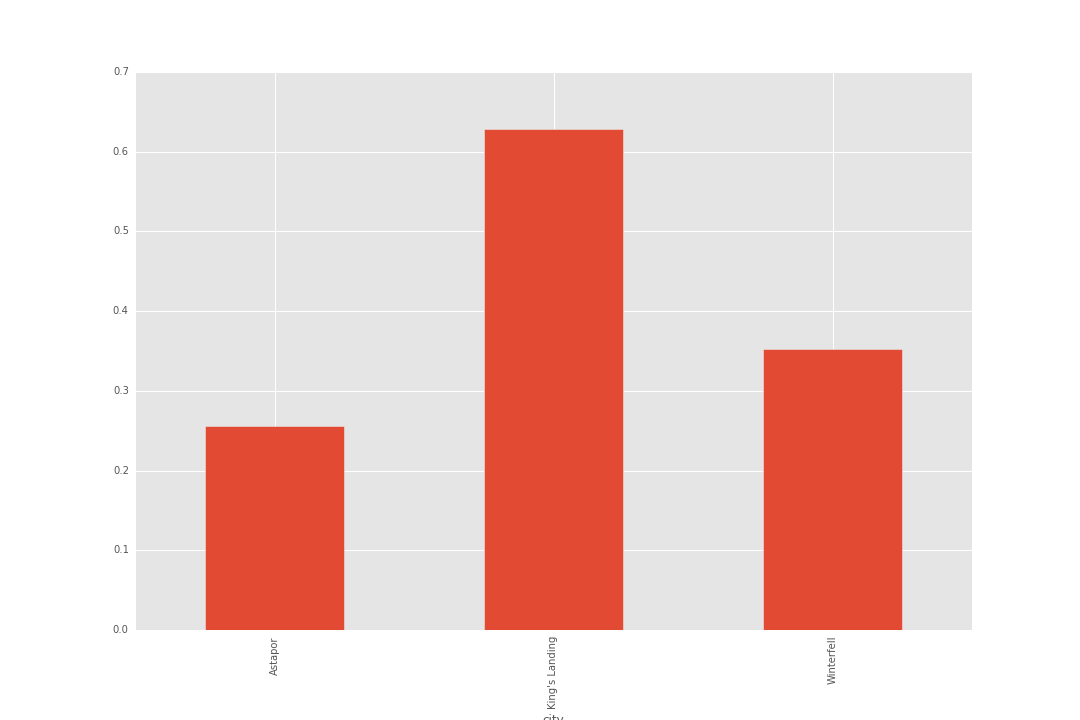
We will use median value to fill in the missing values in this analysis



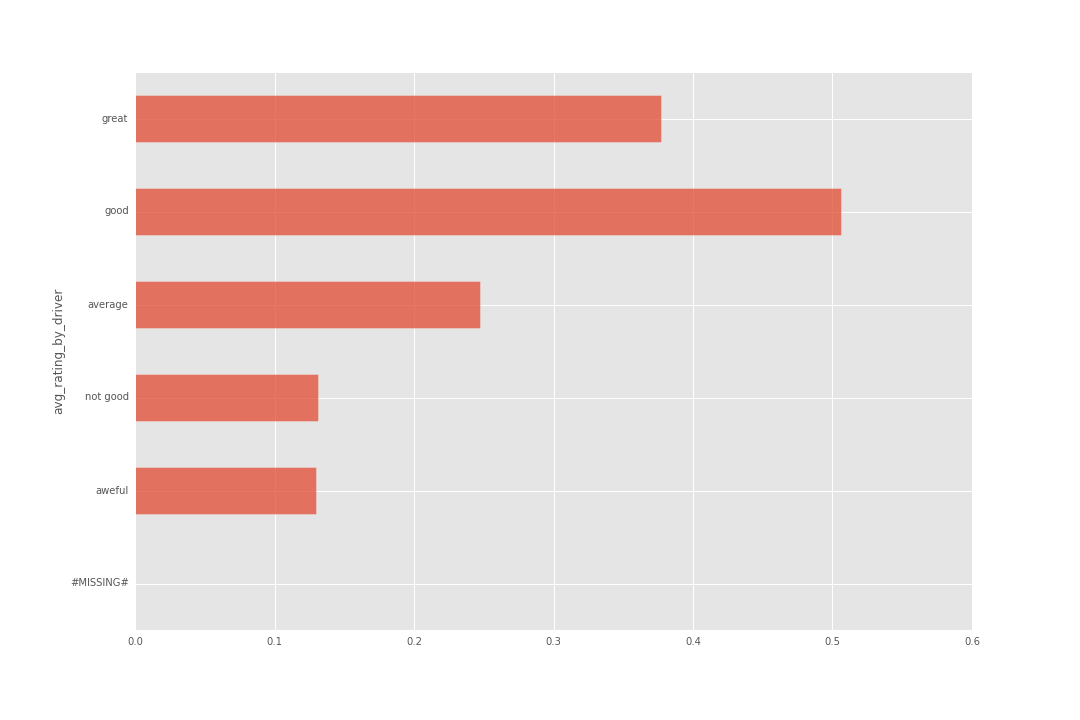
Let us examine the correlation between the features and target variable ‘active’. Avg\_surge and surge\_pct has high correlation, the reason could be that, if surge\_pct is high, means the user use service when surge multiple is > 1 quite often, and thus his avg\_surge should be high also. In order to avoid multicorrlinearity, we will drop avg\_surge in building our final model.

We could also plot the histogram for all the numerical features, for example, below is the histogram for avg\_rating\_of\_driver, most of the rating are between 4 and 5, which makes rating like 1 and 2 outliers, we decide to transform following features to categorical values,

avg\_rating\_of\_driver, avg\_rating\_by\_driver, surge\_pct, avg\_dist and trips\_in\_first\_30\_days. For example, for avg\_rating\_of\_driver, we transform it into five categories: aweful, not good, average, good, great using intervals [1,2,3,4,4.5,5]. We could have done these programmatically using maximum entropy, but given that there are handful of features, we could set the bins manually, the advantage of this is that, the bins makes more sense to others, rather than purely from mathematical calculations.



After we transform these features into categorical values, we could extract more meaningful information from them. For avg\_rating\_by\_driver, we could see from below that, if a user has a high rating by driver, he has a better chance to stay active compared to a user with really low rating, which makes sense since if a user behaves really bad, there is a low chance for him to continue to use uber service.



1. Build predictive model

We will first try out some simple models like logistic regression and Naïve Bayes. 70% data will be used in training (including train and cross validation), 30% data will be used in final test, note that we will only be using the test data once after we finish all the analysis to avoid any bias.

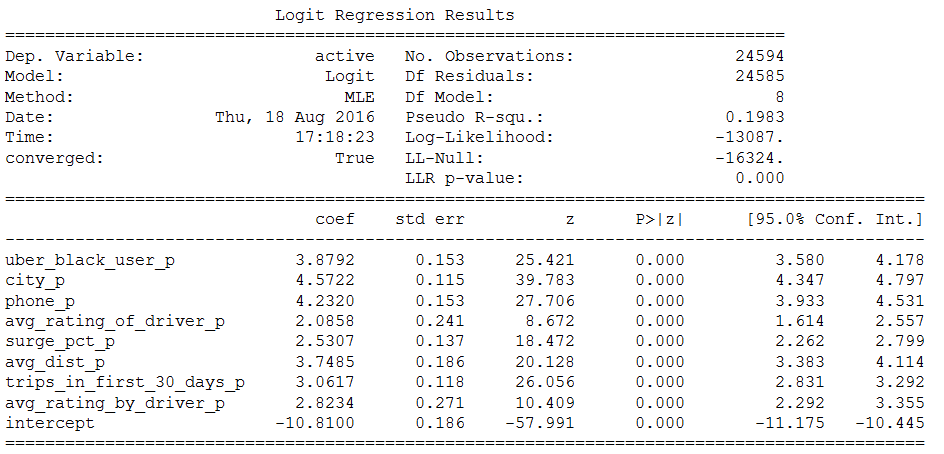
One thing to note is that, after we transform all the features to categorical values, we would then transform them into probabilities. For example, feature uber\_black\_user will be transformed into following probabilities based upon their stay active prob.

Astapor 0.255715

King's Landing 0.628134

Winterfell 0.351945

Now we are ready to build our models. From logistic regression result below, we could see that all features have significant confidents,

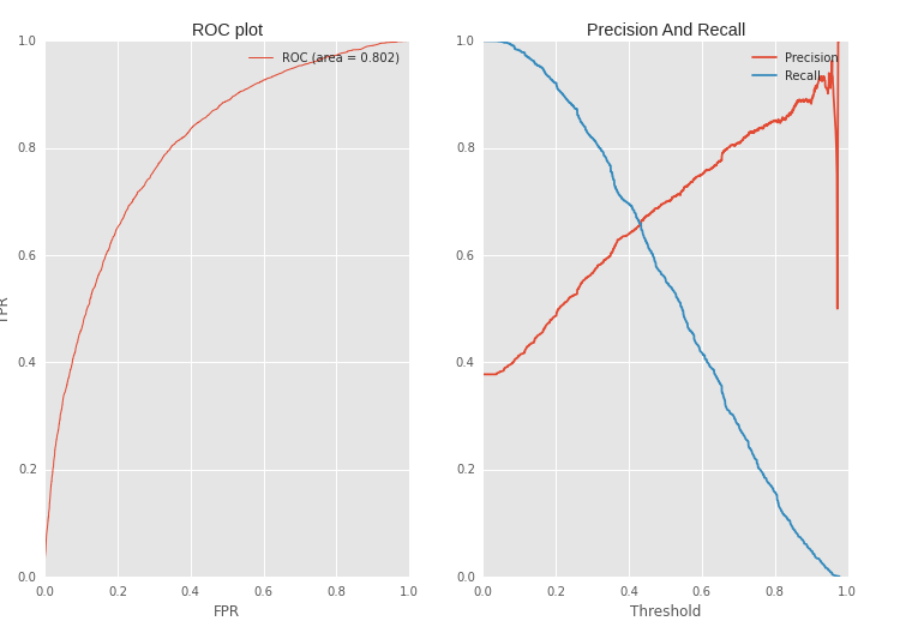


Since we are building classification models, we will use Logloss and AUC ROC, precision, recall accuracy and confusion matrix to evaluate model performance. For logistics regression, the stats and plots are below:

logloss: 0.52, auc roc: 0.802

recall: 0.556 , precision: 0.697, accuracy: 0.743

|  |  |  |
| --- | --- | --- |
| True label/Prediction | 0 | 1 |
| 0 | 5625 | 956 |
| 1 | 1756 | 2203 |

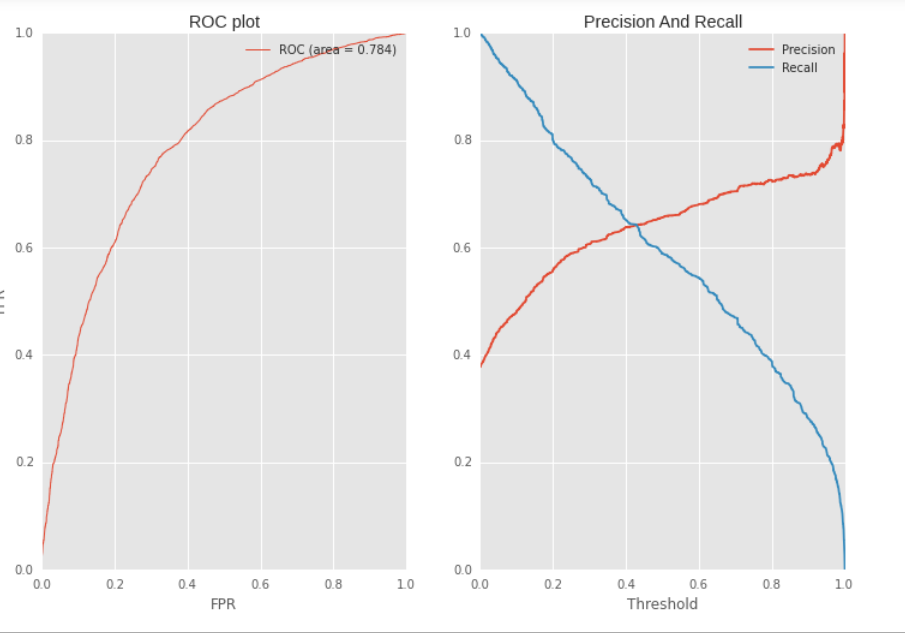


Next we will try out Naïve Bayes model, the stats and result plot are below:

logloss: 0.645, auc roc: 0.784

recall: 0.589, precision: 0.656, accuracy: 0.73

|  |  |  |
| --- | --- | --- |
| True label/Prediction | 0 | 1 |
| 0 | 5359 | 1222 |
| 1 | 1629 | 2330 |



Overall, from the stats above, we could conclude logistic regression outperforms Naïve Bayes.

Let us try more complex model like Gradient Boosting Machine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Result** | **Logistic Regression** | **Naïve Bayes** | **GBM** |
| AUC ROC | 0.784 | 0.781 | 0.799 |
| LOGLOSS | 0.543 | 0.641 | 0.521 |
| ACCURACY | 0.731 | 0.73 | 0.741 |
| PRECISION | 0.689 | 0.663 | 0.712 |
| RECALL | 0.517 | 0.551 | 0.506 |

