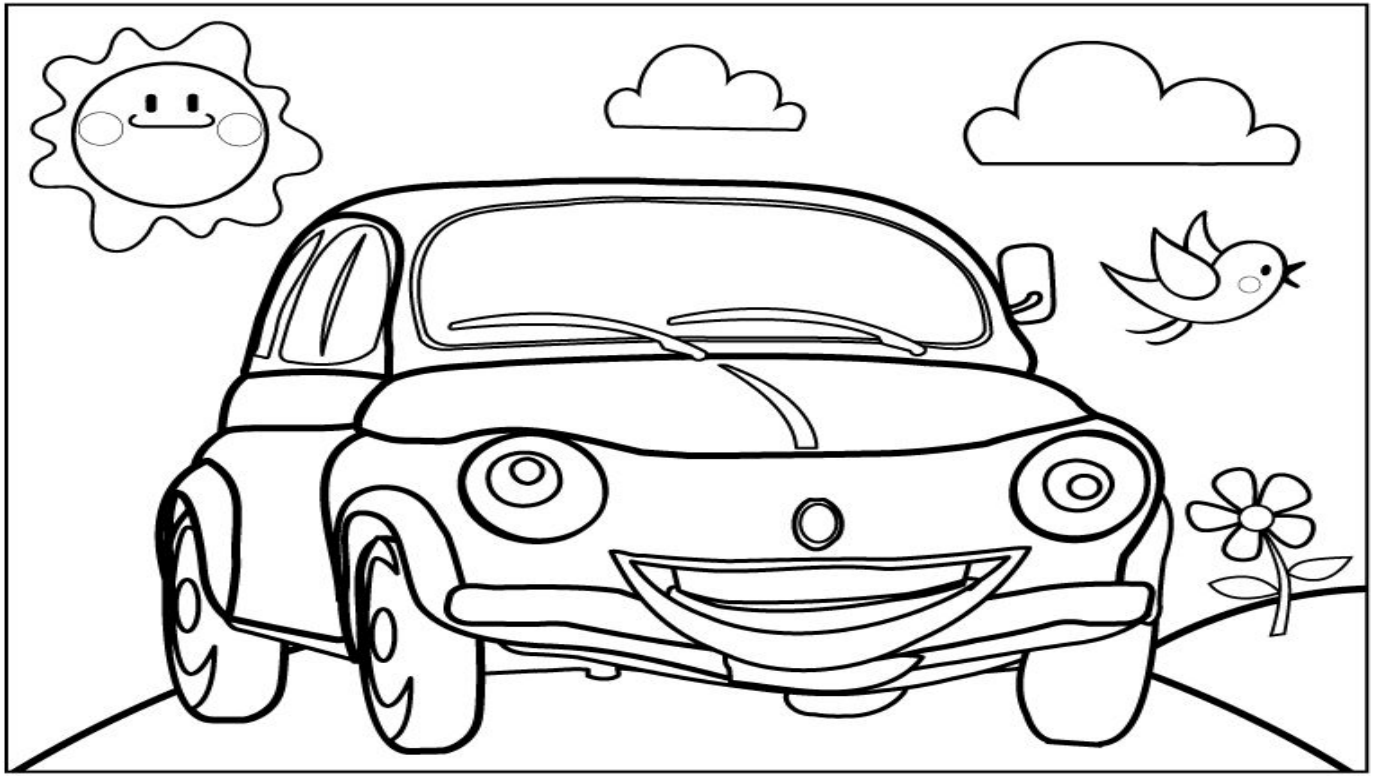
Rideshare Churn Case Study

Keep those customers happy!



Map of Cities in Dataset

1.
TER Cartshare

Let’s start looking into our data

```
categorical= ['city', 'phone' ]
continuous= ['avg_dist', 'avg_rating_by_driver', 'avg_rating_of_driver',
 'surge_pct', 'avg_surge', 'trips_in_first_30_days']
dates=['signup_date','last_trip_date']
all=['avg_dist', 'avg_rating_by_driver', 'avg_rating_of_driver', 'avg_surge',
       'city', 'last_trip_date', 'phone', 'signup_date', 'surge_pct',
       'trips_in_first_30_days', 'luxury_car_user', 'weekday_pct']

train['fill_rating_by_driver'] = train['avg_rating_by_driver'].isnull()
train['fill_rating_of_driver'] = train['avg_rating_of_driver'].isnull()
test['fill_rating_by_driver'] = test['avg_rating_by_driver'].isnull()
test['fill_rating_of_driver'] = test['avg_rating_of_driver'].isnull()
```

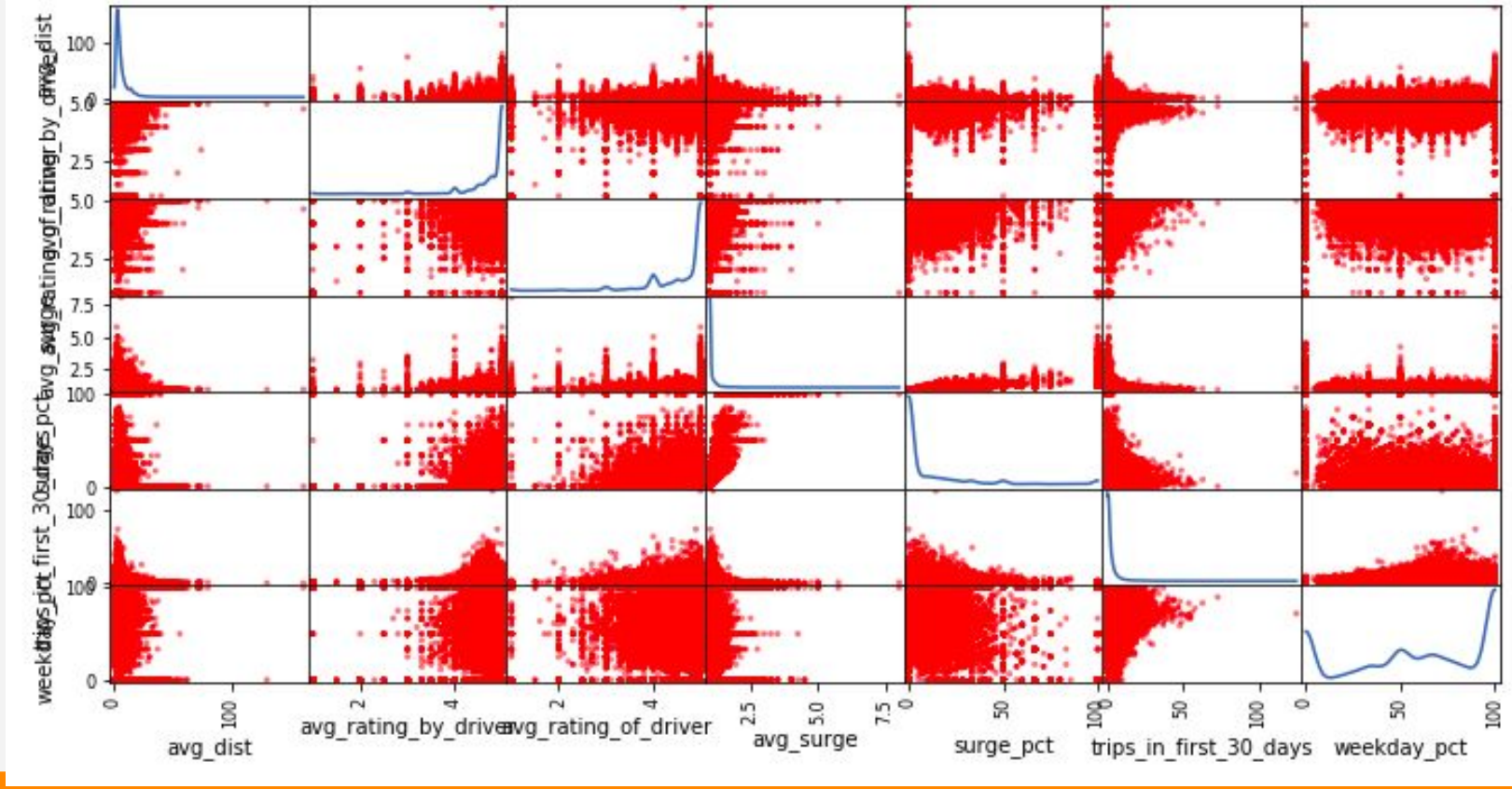
```
train['avg_rating_of_driver'].fillna((train['avg_rating_by_driver']), inplace = True)
# avg diff is .17 between of and by driver
train['avg_rating_by_driver'].fillna((train['avg_rating_by_driver']).mean(), inplace = True)
train['avg_rating_of_driver'].fillna((train['avg_rating_of_driver']).mean(), inplace = True)
test['avg_rating_by_driver'].fillna((test['avg_rating_by_driver']).mean(), inplace = True)
test['avg_rating_of_driver'].fillna((test['avg_rating_of_driver']).mean(), inplace = True)

train['active']=train['last_trip_date']>np.datetime64('2014-06-01')
test['active']=test['last_trip_date']>np.datetime64('2014-06-01')

train.join(pd.get_dummies(train['city'], prefix='city'))
test.join(pd.get_dummies(train['phone'], prefix='phone'))

train.pop('city')
train.pop('phone')
```

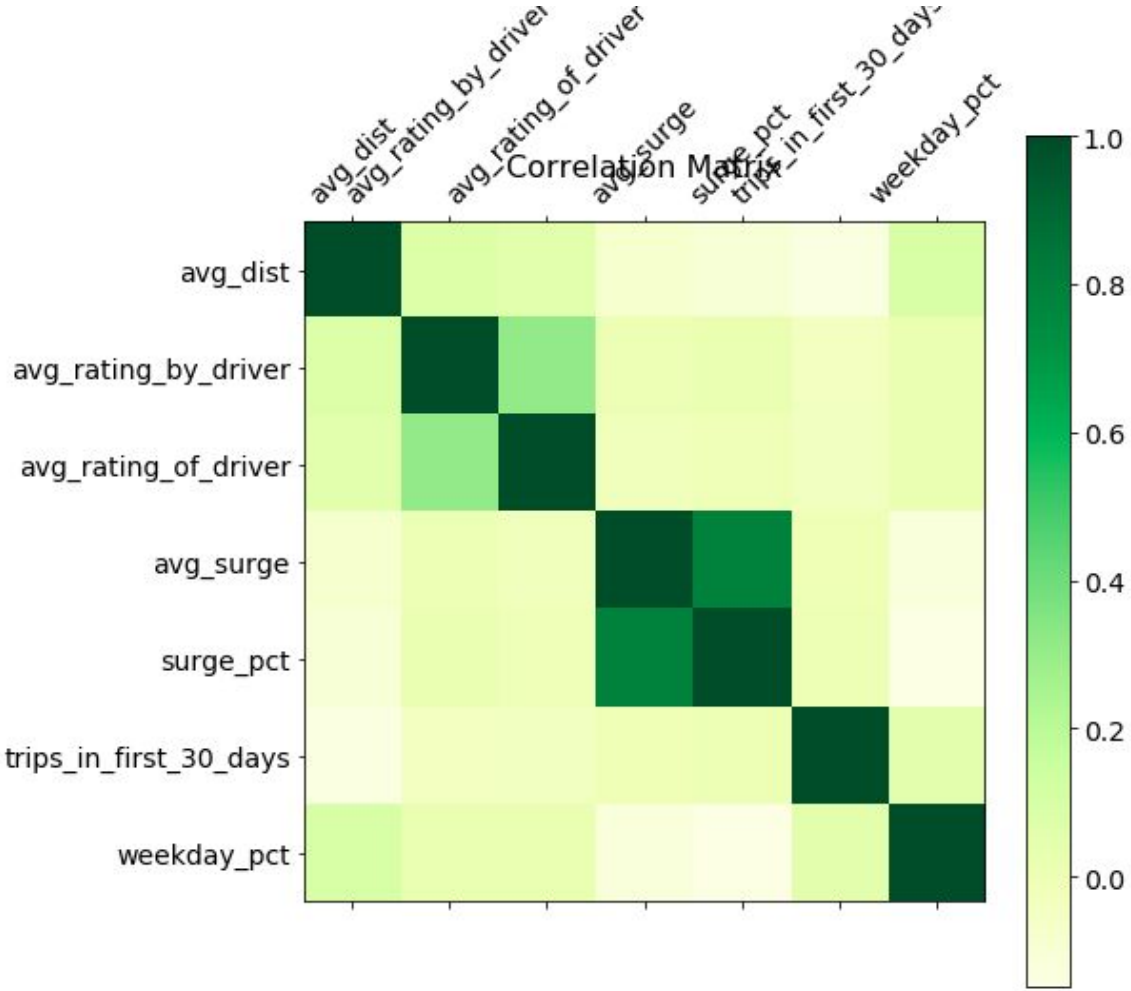
Cleaning the Data

```
train['commuter']=train['weekday_pct'] > 75
test['commuter']=test['weekday_pct']>75

train=train.drop(labels=['last_trip_date','signup_date'],axis=1)
test=test.drop(labels=['last_trip_date','signup_date'],axis=1)


y_train=train.pop('active')
X_train=train
y_test=test.pop('active')
X_test=test
```

Making commuter and dropping dates

Scatter Matrix




Correlation Matrix
avg_dist
avg_rating_by_driver
avg_rating_of_driver
avg_surge
surge_pct
trips_in_first_30_days
weekday_pct
1.0
0.8
0.6
0.4
0.2
0.0

```
dtclass = tree.DecisionTreeClassifier(criterion='gini',
    splitter='best', max_depth=5, min_samples_split=600,
    min_samples_leaf=200, min_weight_fraction_leaf=0.0,
    max_features=None, random_state=None, max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
    class_weight=None, presort=False)

dtclass.fit(X_train,y_train)
dtclass.score(X_test,y_test)
```

```
49] dtclass.fit(X_train,y_train)...

    0.7367588932806324
```

```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier(learning_rate=.1, n_estimators=110,
                                 subsample=1.0,
                                  min_samples_split=600, min_samples_leaf=100,
                                   min_weight_fraction_leaf=0.0,
                                   max_depth=10, min_impurity_decrease=0.0,
                                    min_impurity_split=None, init=None,
                                    random_state=None, max_features=None,
                                    verbose=3, max_leaf_nodes=None,
                                    warm_start=True,
                                     validation_fraction=0.1,
                                     n_iter_no_change=None, tol=0.0001)

gbc.fit(X_train,y_train)
gbc.score(X_test,y_test)
gbc.feature_importances_

from sklearn.linear_model import SGDClassifier
sgdc = SGDClassifier(loss = 'modified_huber',alpha = .0001, max_iter =100
sgdc.fit(X_train,y_train)
sgdc.score(X_test,y_test)
classifiers.append(model2)

from sklearn.svm import LinearSVC
lsvc = LinearSVC(penalty='l2',loss='hinge',verbose=3,max_iter=5000)
lsvc.fit(X_train,y_train)
lsvc.score(X_test,y_test)
```

```
parameters = {'max_depth':range(3,12),
              'min_samples_leaf': range(200,1000,200),
              'min_samples_split': range(600,2400,400)}
clf = GridSearchCV(tree.DecisionTreeClassifier(), parameters,
clf.fit(X=X_train[important], y=y_train)
tree_model = clf.best_estimator_
print (clf.best_score_, clf.best_params_)
```

```
#This is with SMOTE Data
parameters = {'bootstrap': [True],
 'max_depth': [18],
 'max_features': ['auto'],
 'min_samples_leaf': [1],
 'min_samples_split': [2],
 'n_estimators': [65]}
gs = GridSearchCV(clf, param_grid = parameters, cv=5,
 n_jobs=-1, iid=False, scoring='accuracy', verbose=10)
gs.fit(X_smo, y_smo)
```

score = 0.8138

The Top 5 Features :

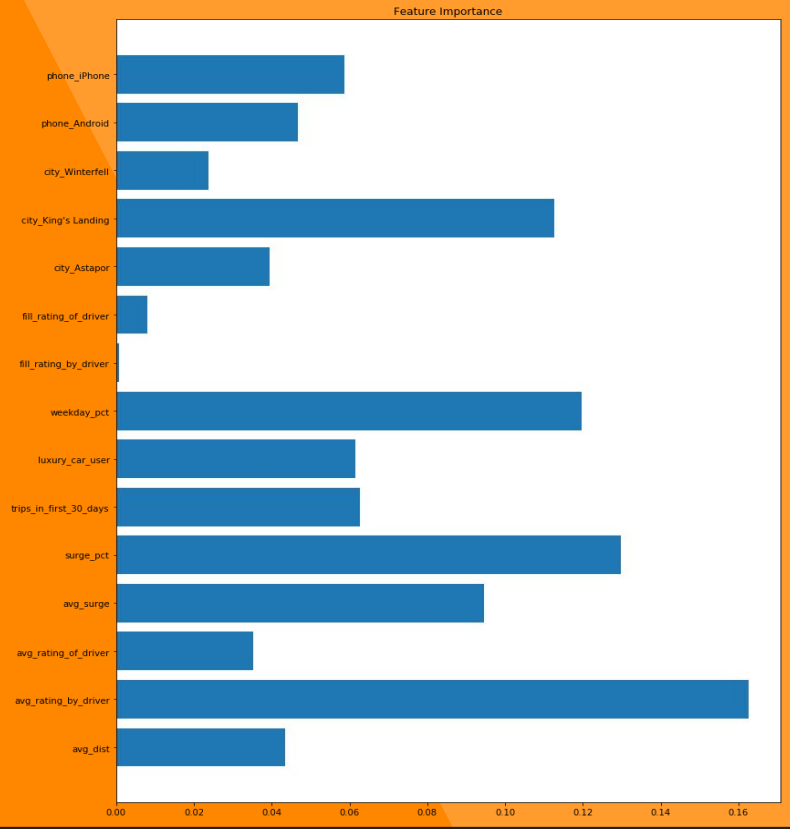
1.Avg rating by driver

2.Surge Pct

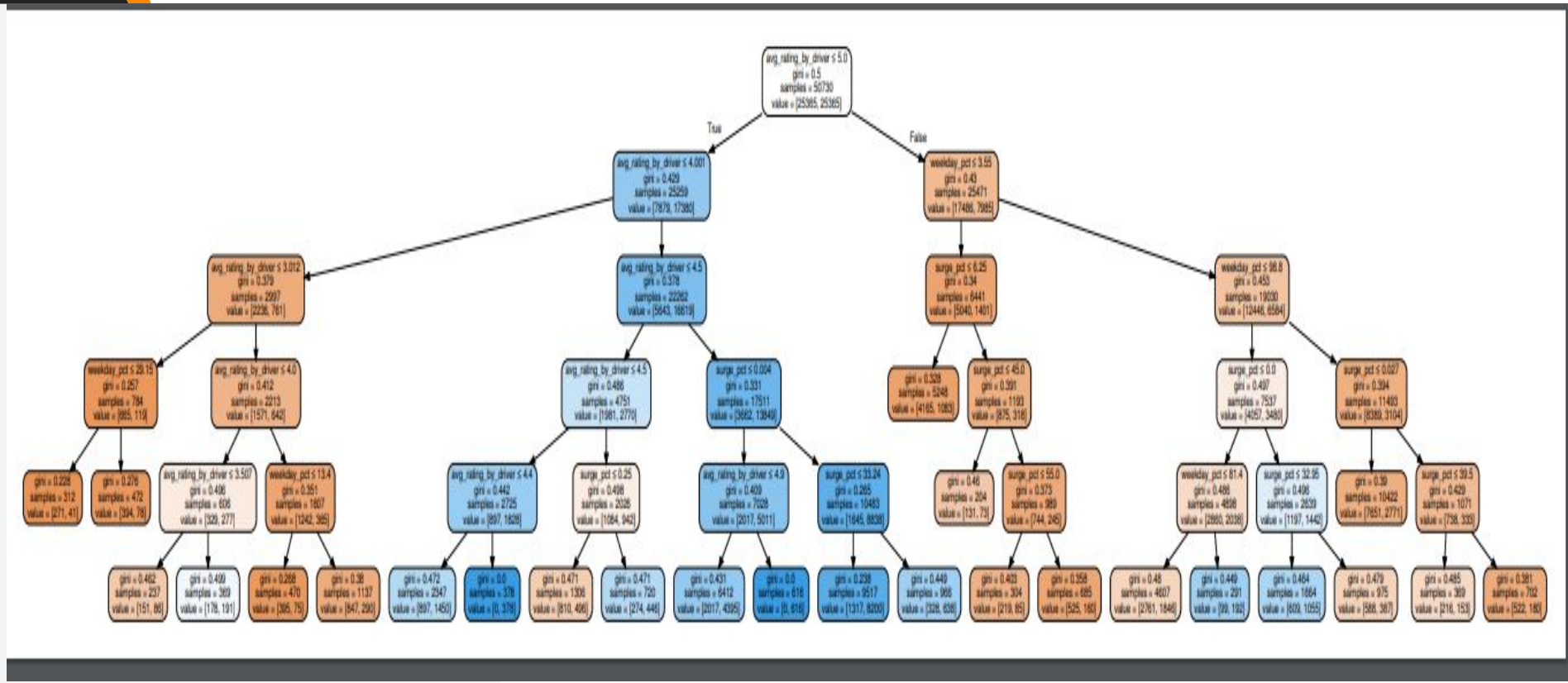
3.Weekday Pct

4.City Kings Landing

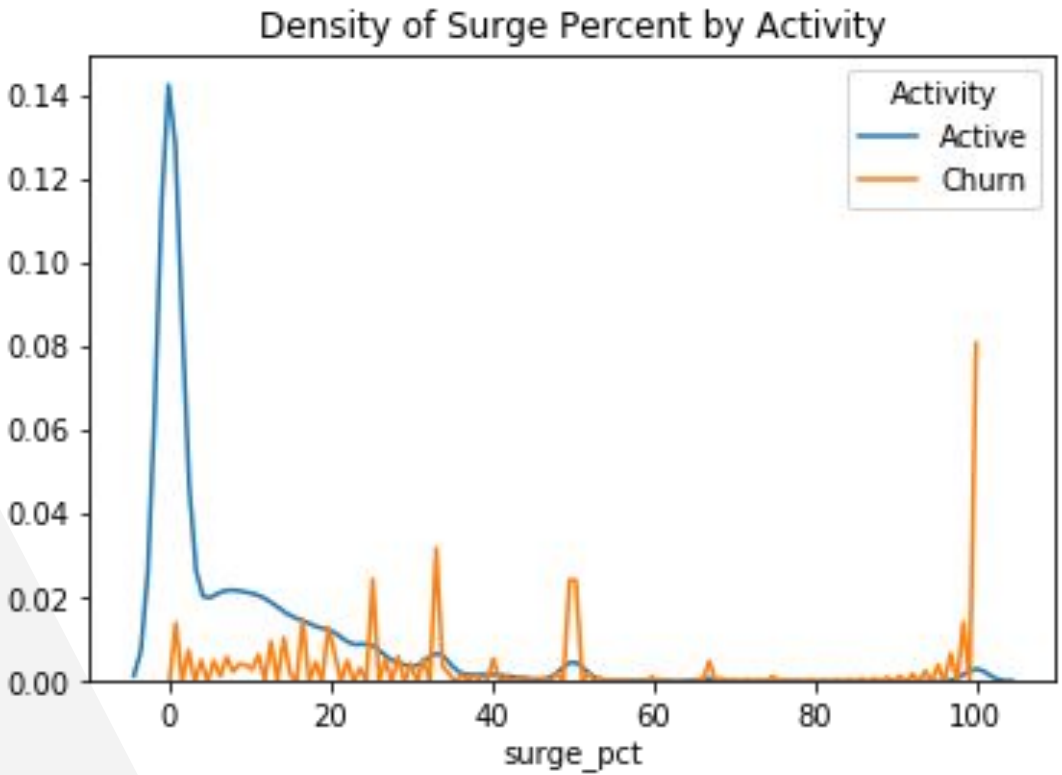
5.Avg Surge



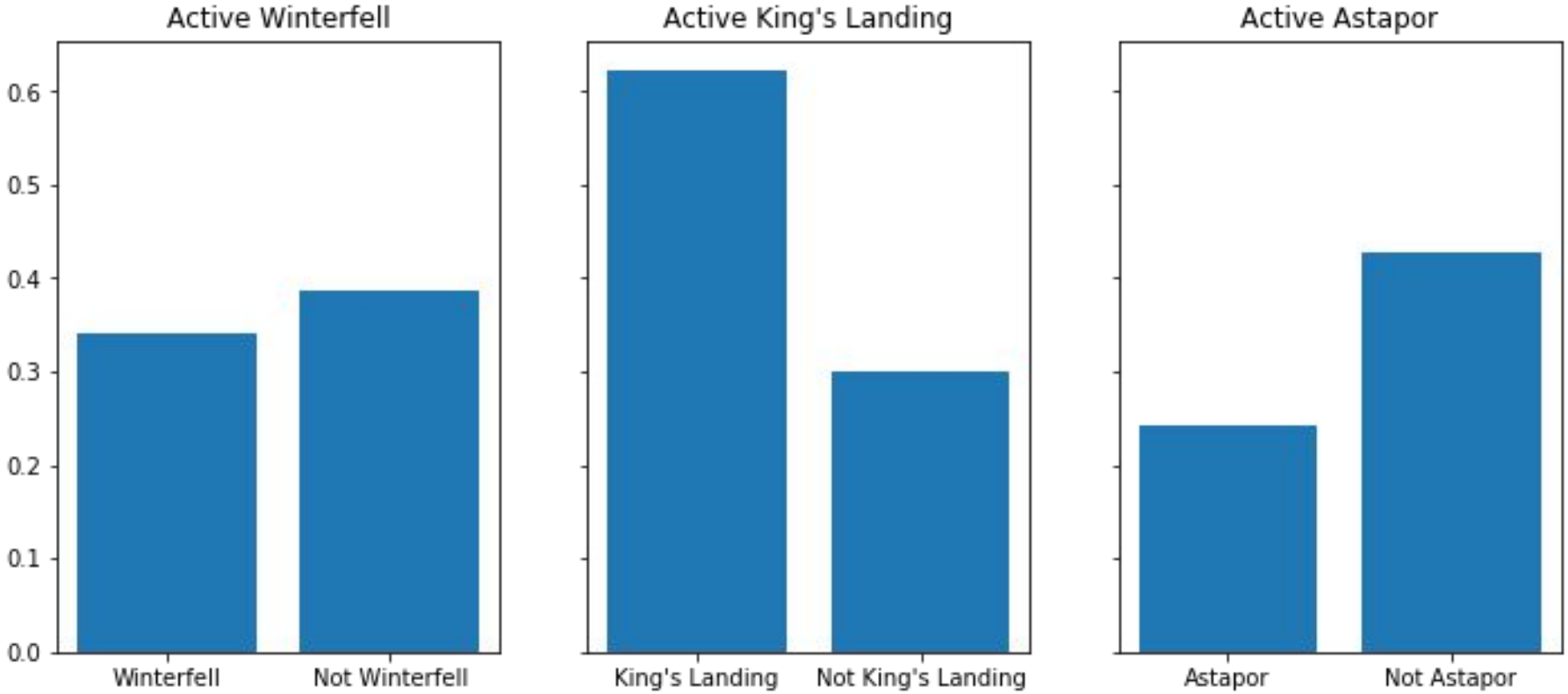
Training a decision tree on the top 3 important features

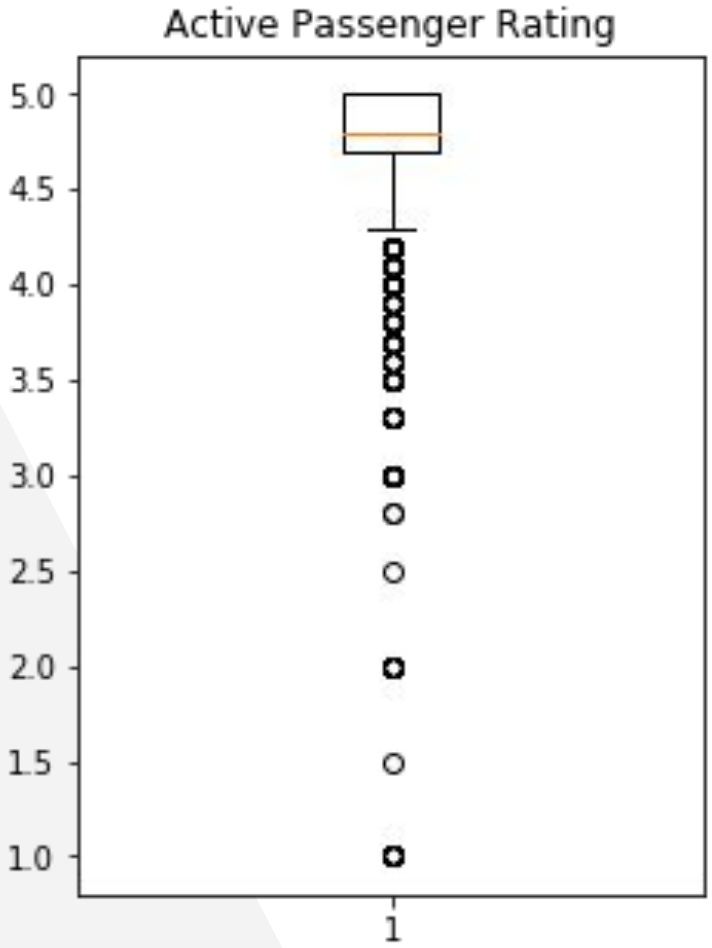


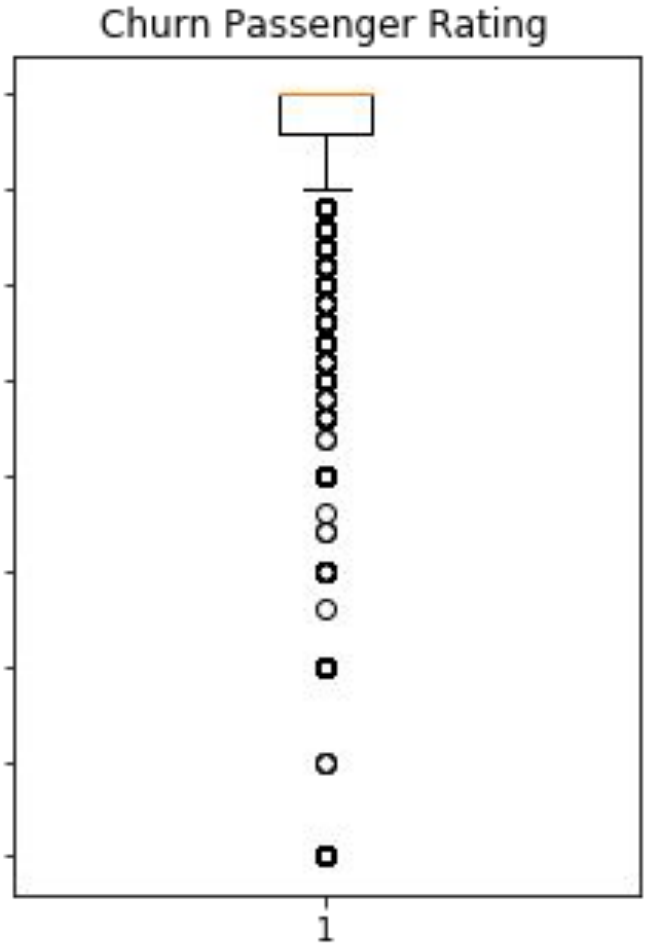
Density of Surge



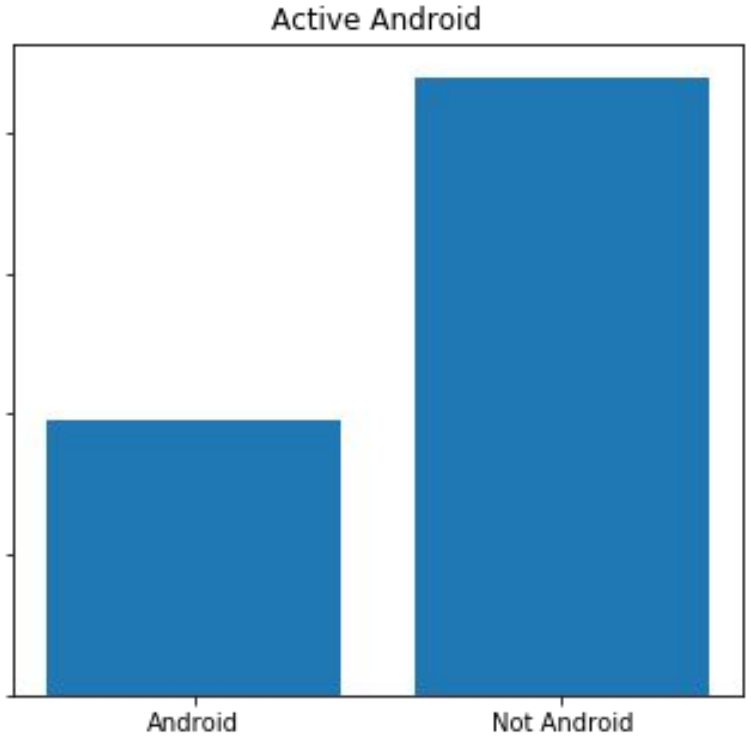
Activity per City




Active Passenger Rating
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1


Churn Passenger Rating
1

Activity per Phone type



Reward your Daily Passengers

These are the users who are going to use the service more often

Make sure they is good loyalty and reward packages.

Pay attention to driver ratings

High feature importance, higher than the avg driver ratings the users experienced.

Low Rating warning signs, targetable for promos

Controlling Surge

Surge has a semi-high feature importance while also being a changeable variable

Test changing surge based on churn risk

THANKS!

Any questions?