Churn Case Study

Team Bar Charts for Days

(formerly team dropna)

The problem

Ride-sharing company X wants to predict rider retention.

What factors are the best predictors for retention??

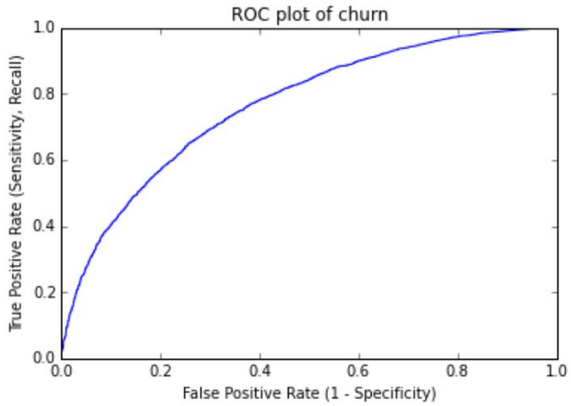
Context

- sample dataset of a cohort of users who signed up for an account in January 2014
- data was pulled on July 1, 2014;

Logistic regression

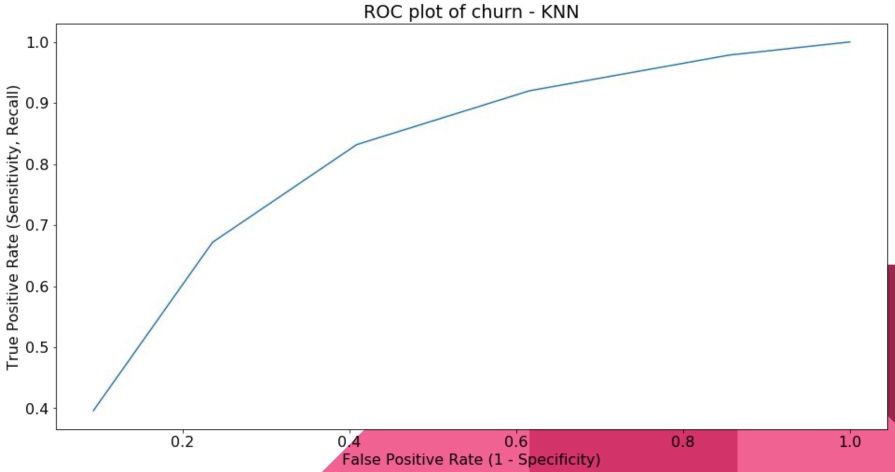
1. Fit logistic regression to training data
2. Get cross-validated y - predictions (cross_val_predict)
3. Score:
    a. X_train vs. cross_val_pred_y's **0.722**
    b. X_tst vs. y_test **0.718**

Accuracy: 0.728
Recall: 0.499
Precision: 0.675



K-Nearest Neighbors Model

1. Fit KNeighborsClassifier to training data
2. Get cross-validated y - predictions (cross_val_predict)
3. Score:
   a. X_train vs. cross_val_pred_y's **0.808**
   b. X_tst vs. y_test **0.744**



Take-aways:
- This model appears to be slightly overfit
- Out-of-set performance worse than in-set x-val performance

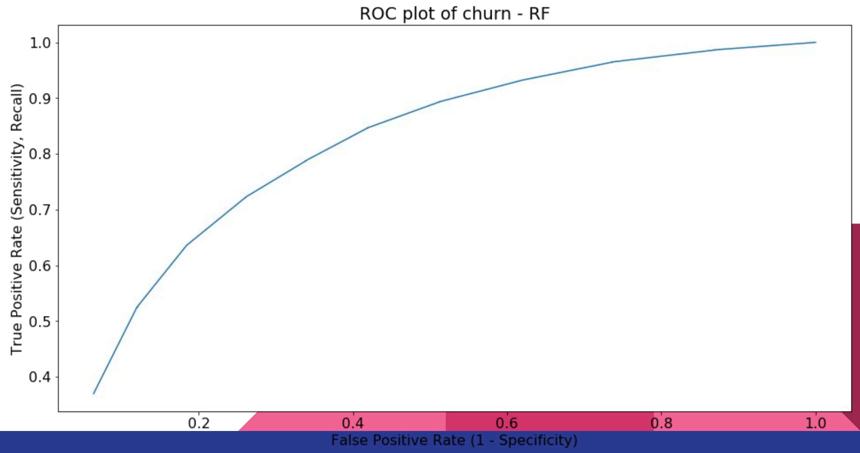
Random Forest Model

1. Fit RandomForestClassifier to training data
2. Get cross-validated y - predictions (cross_val_predict)
3. Score:
    a. X_train vs. cross_val_pred_y's **0.772**
    b. X_test vs. y_test **0.749**

Accuracy: 0.769
Recall: 0.626
Precision: 0.714

Take-aways:

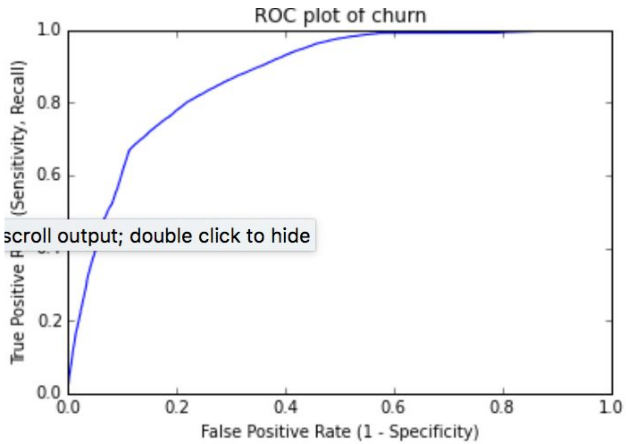
- Not overfit (like KNN)
- Virtually identical out-of-set performance



SVM

1. Fit SVC to training data
2. Get cross-validated y - predictions (cross_val_predict)
3. Score:
    a. X_train vs. cross_val_pred_y's **0.806**
    b. X_test vs. y_test **0.762**

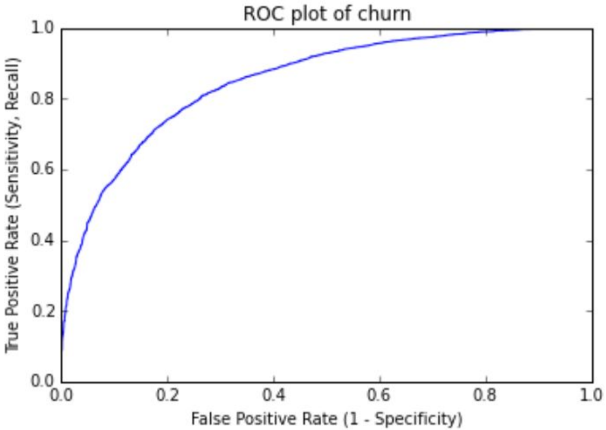
Accuracy: 0.745
Recall: 0.55904
precision: 0.689



Gradient Boosted Trees Classification Model

1. Split Training data into training and validation set
2. Fit GradientBoostingClassifier to training data
3. Got cross-validated y - predictions (cross_val_predict) on training set
4. Score:
    a. X_train vs. cross_val_pred_y's **0.788**
    b. X_val vs. y_val **0.792**

Accuracy: 0.792
Recall: 0.665
Precision: 0.749



EDA

5]:

	feature name	feature description
0	last_trip_date	the last time this user completed a trip; in the form `YYYYMMDD`
1	phone	primary device for this user
2	weekday_pct	the percent of the user's trips occurring during a weekday
3	avg_rating_by_driver	the rider's average rating over all of their trips
4	city	city this user signed up in
5	trips_in_first_30_days	the number of trips this user took in the first 30 days after signing up
6	signup_date	date of account registration; in the form `YYYYMMDD`
7	avg_rating_of_driver	the rider's average rating of their drivers over all of their trips
8	avg_surge	The average surge multiplier over all of this user's trips
9	surge_pct	the percent of trips taken with surge multiplier > 1
10	avg_dist	the average distance (in miles) per trip taken in the first 30 days after signup
11	luxury_car_user	TRUE if the user took a luxury car in their first 30 days; FALSE otherwise

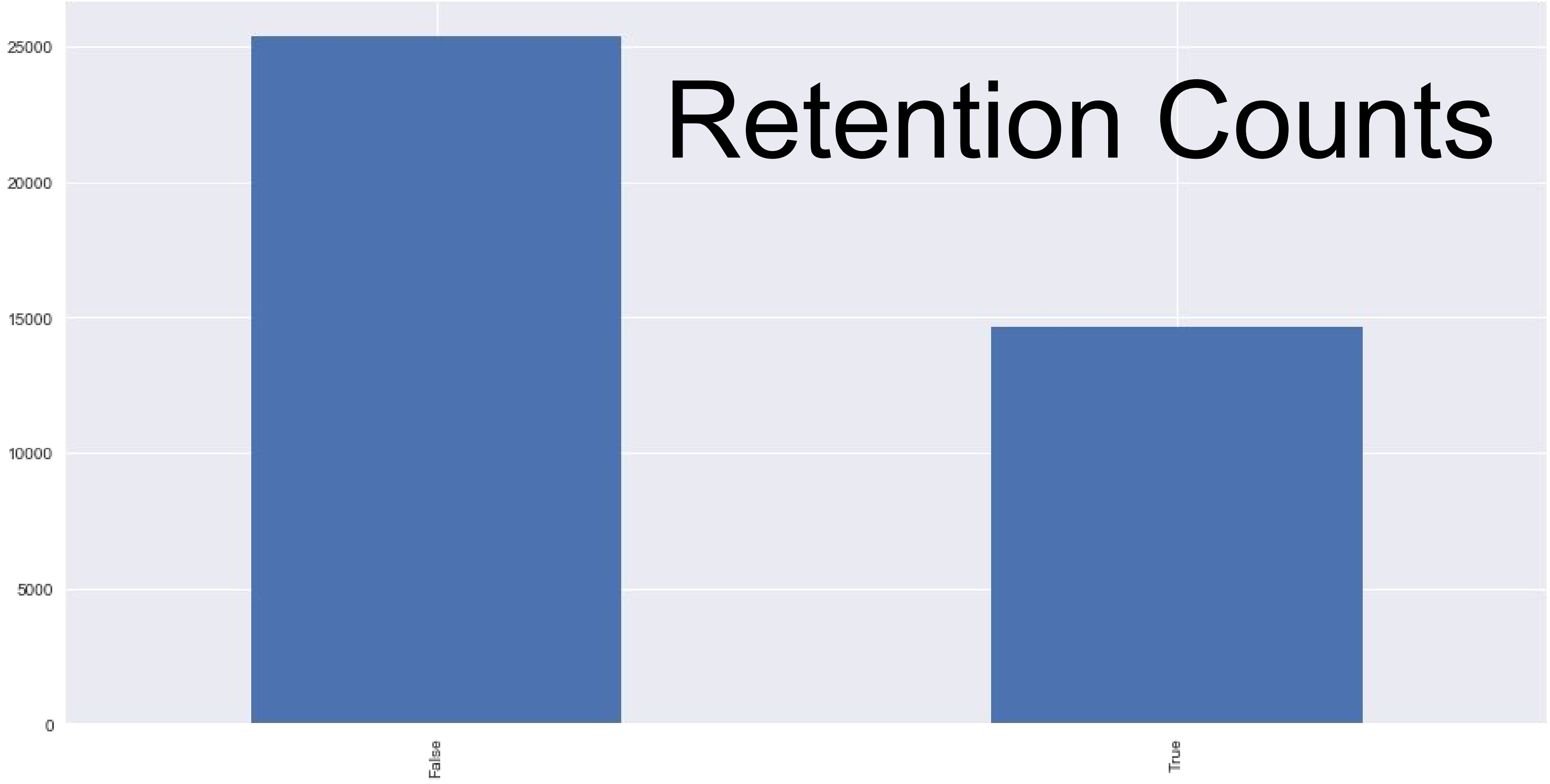
Computing the Target

We computed retention, rather than churn

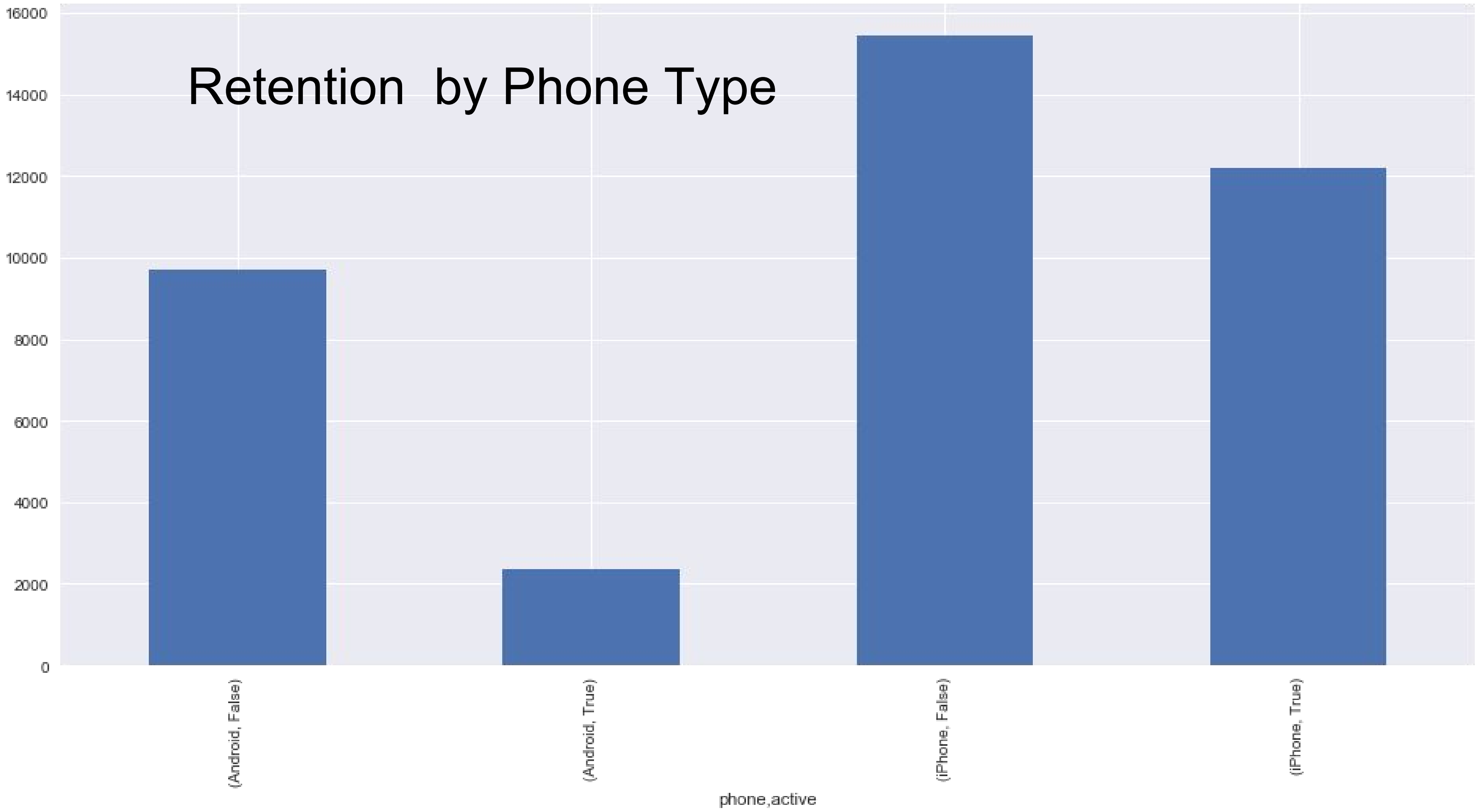
We consider a user retained if they were “active” (i.e. took a trip) in the preceding 30 days (from the day the data was pulled). In other words, a user is "active" if they have taken a trip since June 1, 2014

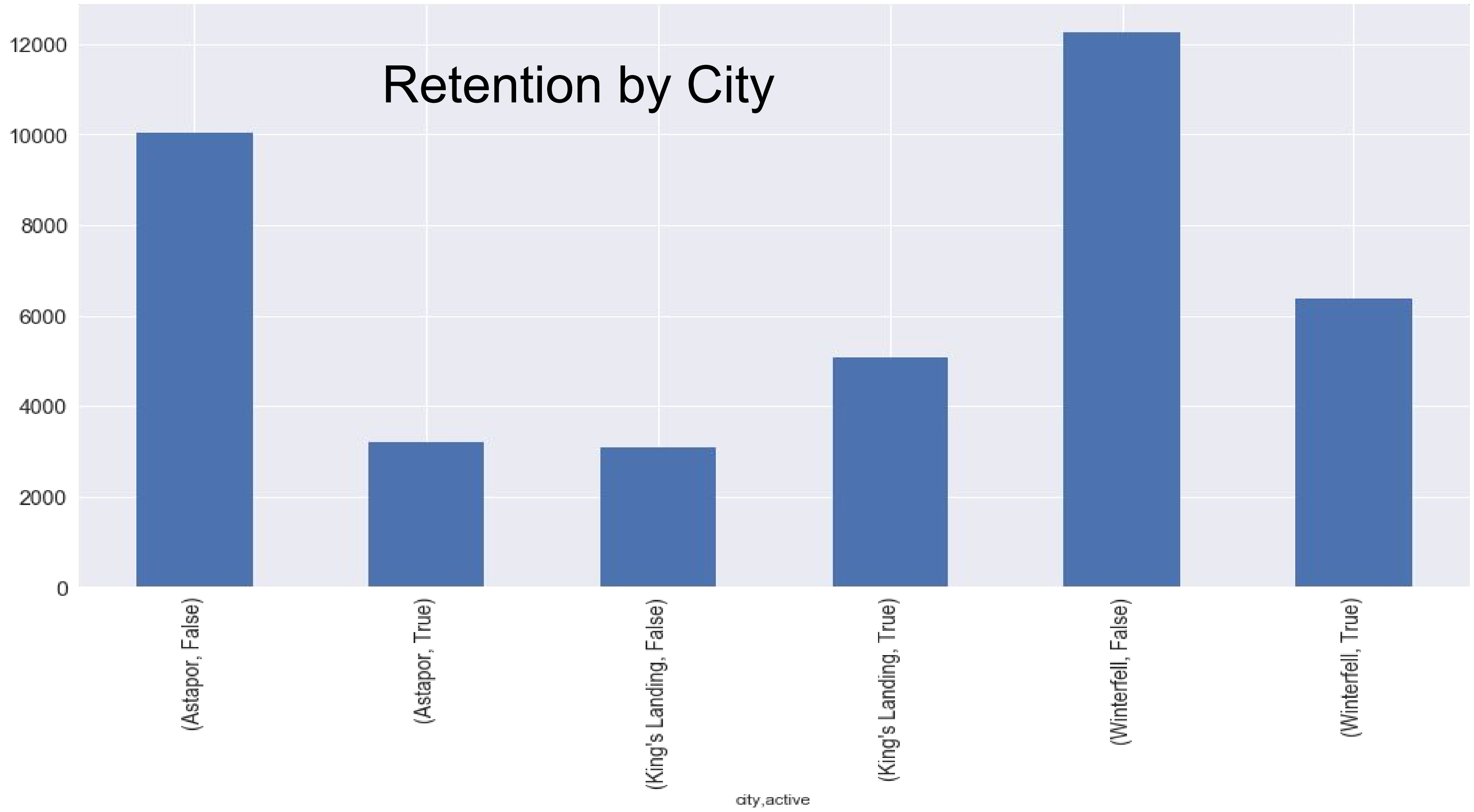
```
df.last_trip_date = pd.to_datetime(df.last_trip_date)

df["active"] = df['last_trip_date'] > "2014-06-01"
```

Retention Counts

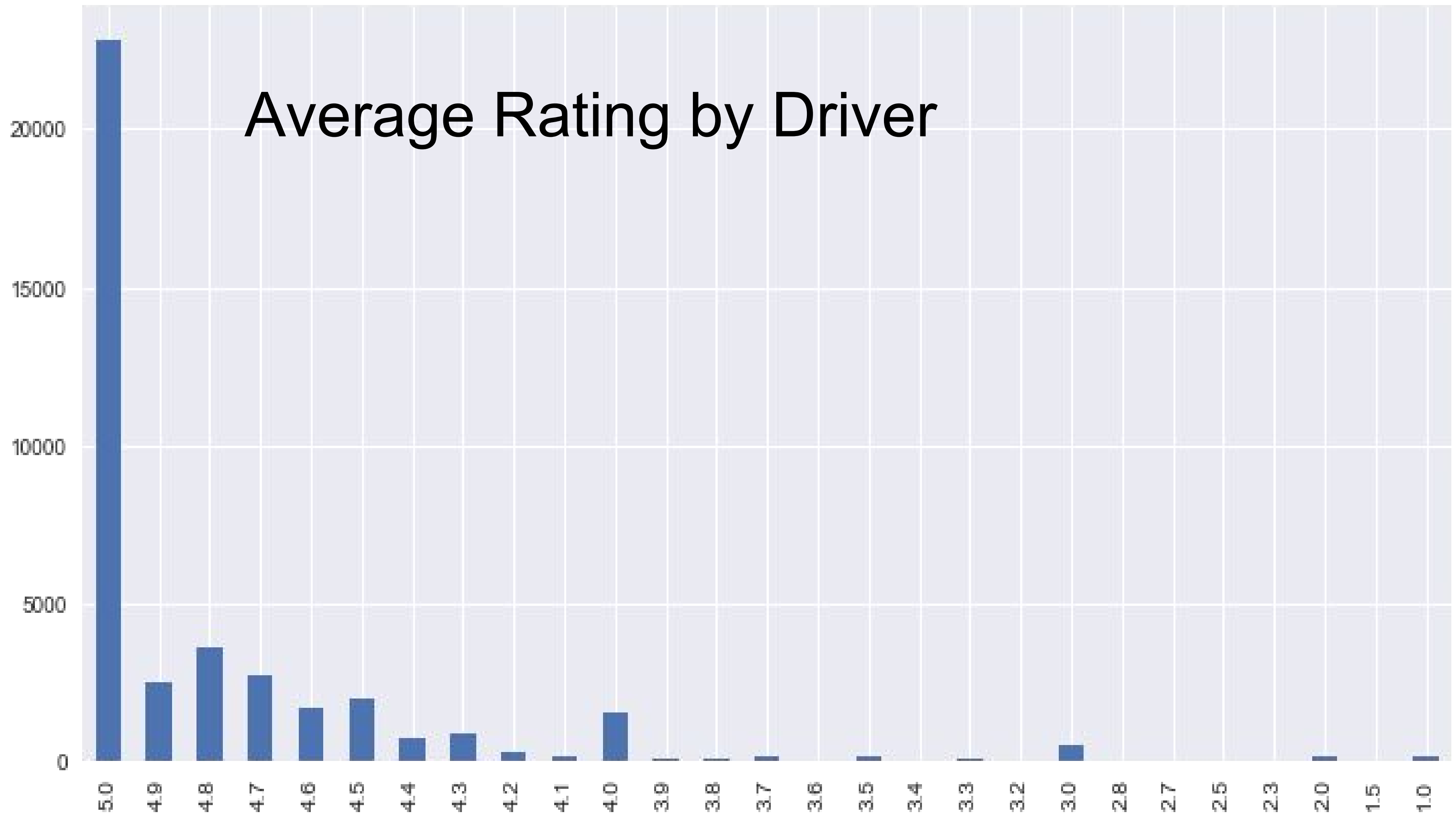


Retention by Phone Type





Retention by City
0
2000
4000
6000
8000
10000
12000
(Astapor, False)
(Astapor, True)
(King's Landing, False)
(King's Landing, True)
(Winterfell, False)
(Winterfell, True)
city,active


Average Rating by Driver
20000
15000
10000
5000
0
5.0
4.9
4.8
4.7
4.6
4.5
4.4
4.3
4.2
4.1
4.0
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.0
2.8
2.7
2.5
2.3
2.0
1.5
1.0

Data Munging

Find missing values

Impute Missing values (used mean for user and driver ratings)

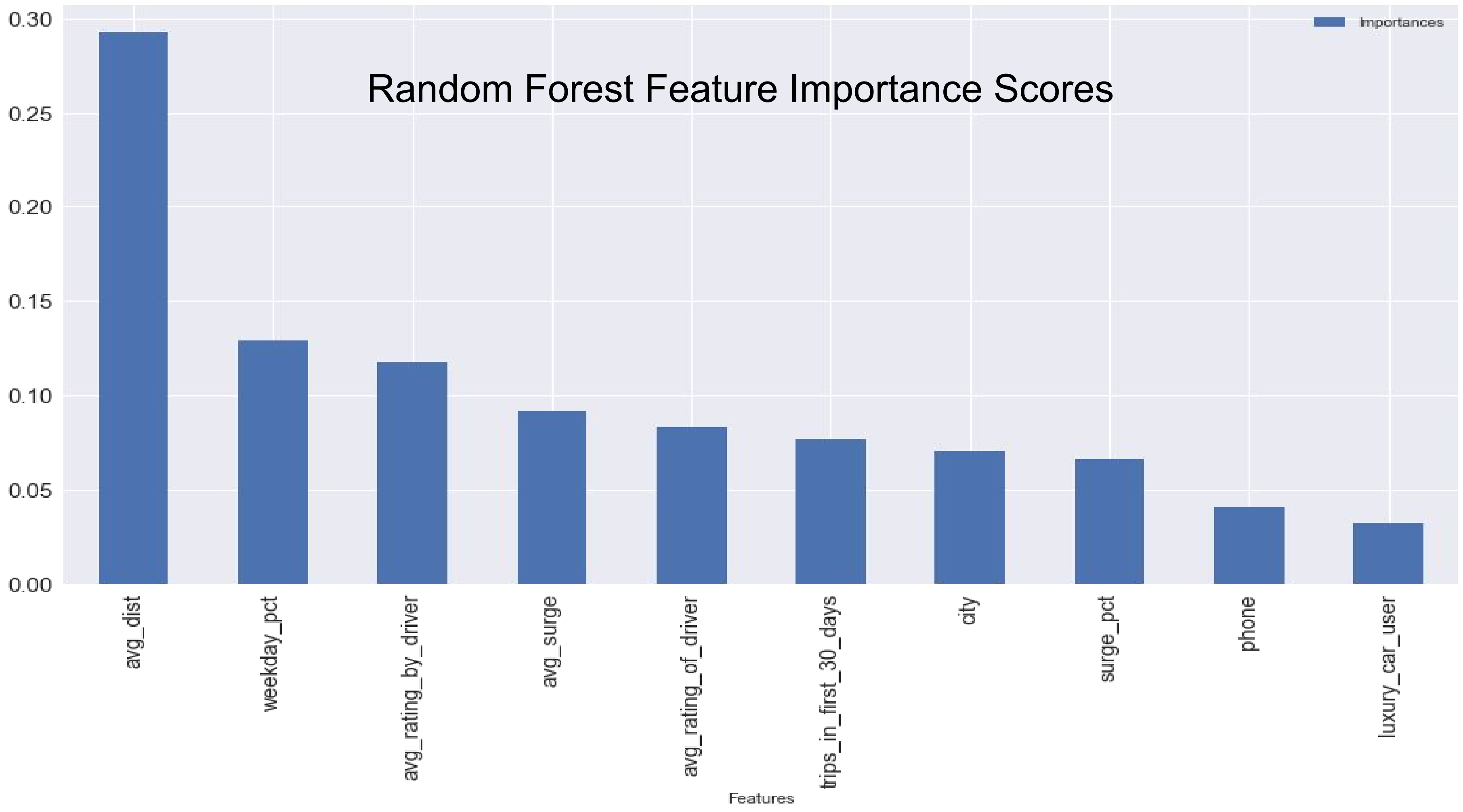
Drop remaining NAs (from phones)

Drop date columns

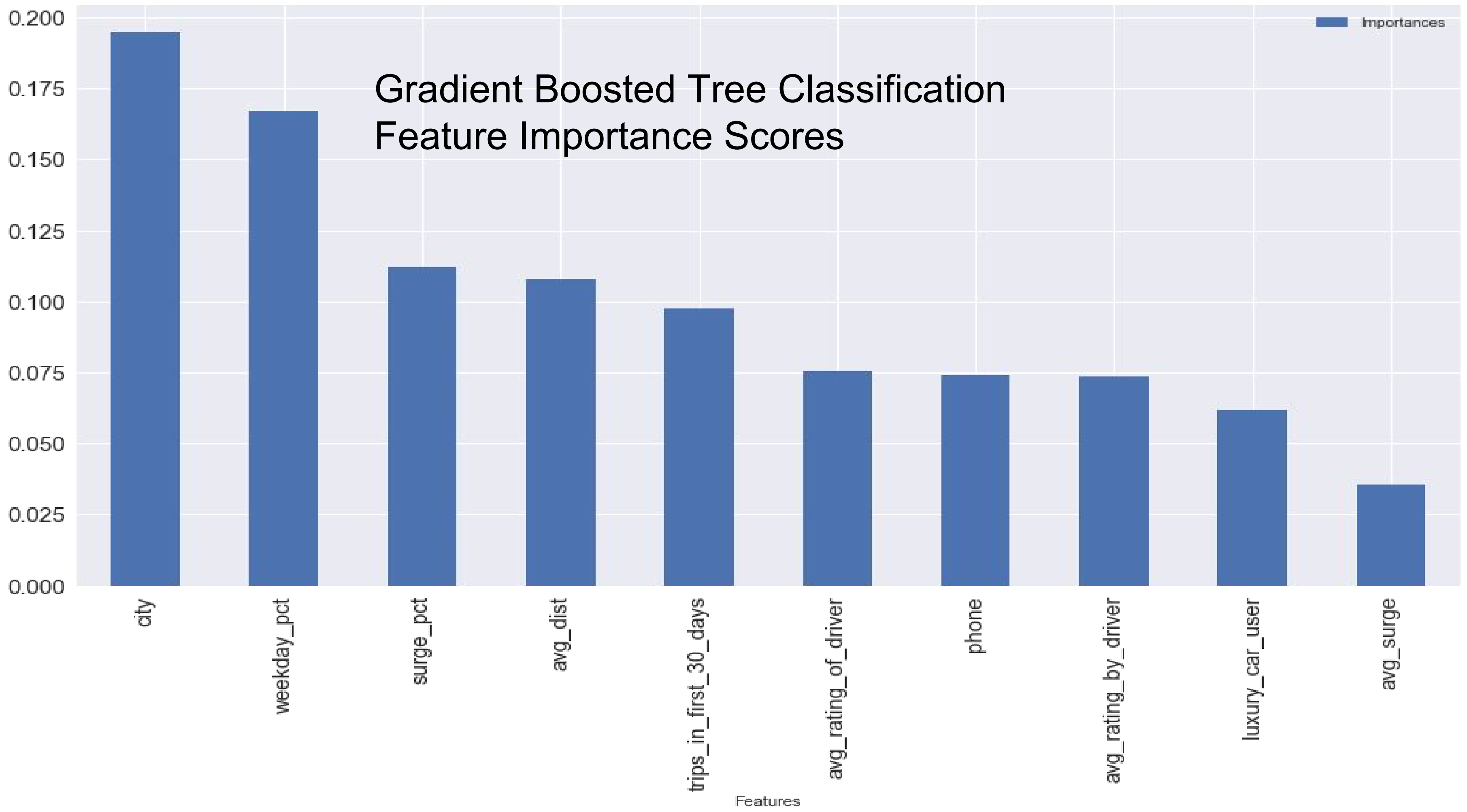
6]:

	NaNs as prop of total
avg_dist	0.000000
avg_rating_by_driver	0.004050
avg_rating_of_driver	0.163200
avg_surge	0.000000
city	0.000000
last_trip_date	0.000000
phone	0.007975
signup_date	0.000000
surge_pct	0.000000
trips_in_first_30_days	0.000000
luxury_car_user	0.000000
weekday_pct	0.000000
active	0.000000

Model Fitting



Random Forest Feature Importance Scores
Importances
0.30
0.25
0.20
0.15
0.10
0.05
0.00
avg_dist
weekday_pct
avg_rating_by_driver
avg_surge
avg_rating_of_driver
trips_in_first_30_days
city
surge_pct
phone
luxury_car_user
Features


Gradient Boosted Tree Classification
Feature Importance Scores
Importances
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
city
weekday_pct
surge_pct
avg_dist
trips_in_first_30_days
avg_rating_of_driver
phone
avg_rating_by_driver
luxury_car_user
avg_surge
Features

Recommendations