Data Science Project
November, 29 2016

Question

Can you predict if a sitter will have churned 180 days after they were approved at the point when their profile is approved?

Defining Churn

At 180 Days after your profile was first approved, you meet the following conditions:

- Your profile is inactive (Explicit)

OR

- You have not had an **Event** in the past 100 days* (Implicit)

*Less than 10% of sitters return who haven't had an event in the past 100 days

Events:

- Booked a Stay
- Sent a Message
- Uploaded a Photo
- Scheduled a Meet and Greet
- Activated a Service
- Set Affirmative Weekend Availability

Feature Set

We can only look at variables known to us at the point at which the sitter's profile is sent for approval. 26 features evaluated in total.

Features Evaluated:

Number of Photos	Zip Code
Services Listed	Time Taken to Complete Profile
Day of Week Availability	Dog owner
Protection Package	Children in Home
Building Type	Number of Testimonials Requested
Dogs Allowed on Furniture	Years of Dog Sitting Experience

Data Set Sizing

All sitters approved > 180 days ago that have data sufficiency due to lack of data availability from very early sitters:

- 96,125

Of that group, number that were churned 180 days after approval:

- 51,742 (53.82%)

Creating the Feature Set

Fixing the earlier issues:

- Instead of attempting to pull all of the data in 1 query, I pulled about 8 columns in each query and then joined them together in Python.

```
import pandas as pd
path = '../Data Science Class/'
url1 = path + 'AllSittersApprovedAtLeast180DaysAgoFile1.csv'
url2 = path + 'AllSittersApprovedAtLeast180DaysAgoFile2.csv'
url3 = path + 'AllSittersApprovedAtLeast180DaysAgoFile3.csv'
rover1 = pd.read_csv(url1, index_col='id')
rover2 = pd.read_csv(url2, index_col='id')
rover3 = pd.read_csv(url3, index_col='id')

frames = [rover1, rover2, rover3]

rover = pd.concat(frames, axis = 1)

rover.head()
```

Evaluating the Feature Set

After initial analysis, the following features were the most predictive:

Listed Boarding as Service

```
rover.groupby(['listed_boarding']).churned.mean()
```

```
listed_boarding
0    0.641888
1    0.480332
Name: churned, dtype: float64
```

Protection Package

```
rover.groupby(['protection_package']).churned.mean()
```

```
protection_package
0    0.560136
1    0.453652
Name: churned, dtype: float64
```

Listed Weekend Availability

```
rover.groupby(['saturday']).churned.mean()
```

```
saturday
0    0.642776
1    0.534311
Name: churned, dtype: float64
```

```
rover.groupby(['sunday']).churned.mean()
```

```
sunday
0    0.649370
1    0.532589
Name: churned, dtype: float64
```

Logistic Regression Model

I used logistic regression and train/test/split to create an initial model.

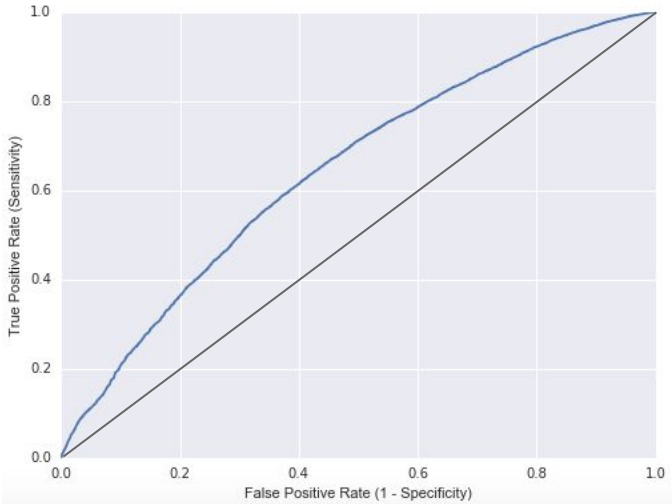
Null model accuracy - .5382

Model accuracy score - .6126

AUC score - .6453

With additional data from merging data instead of limiting data, accuracy improved by .2

ROC Curve



Random Forest Model

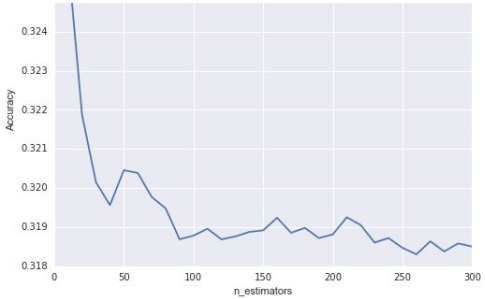
I developed a random forest classification model as well to evaluate the data.

Null model accuracy - .5382

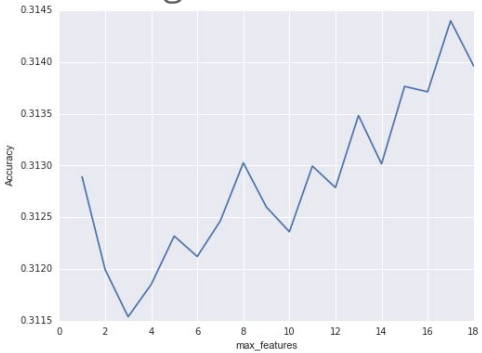
Model oob score (equivalent to accuracy score) - .5979

Random forest model has turned out to be less predictive than logistic regression model.

Choosing Estimator



Choosing Max Features



Next Step

Determine how to make this data usable for people who are evaluating sitter profiles