DS-SF-27 Final Project

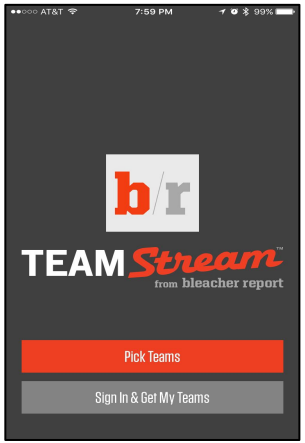
Andrew Burke

Project Problem & Hypothesis

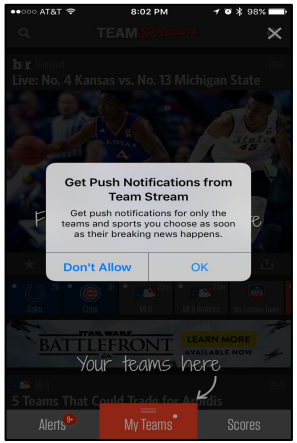
- Problem
  - I want to predict if a user will be retained after using our mobile app for the first time. If we are able to reliably predict this, we can then design features to improve retention.
- Hypothesis
  - As users add more streams/view more articles/enable push notifications in their first session, the higher the probability they will be retained.

- Machine Learning Model
  - This is a classification problem, and the outcome of the machine learning model will be the probability that a user will be retained after their first session.

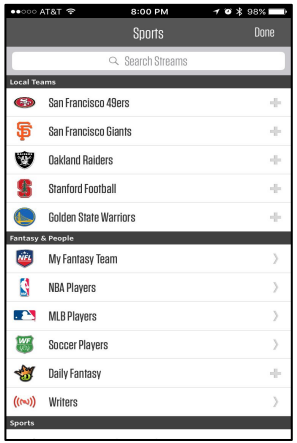
New User Flow



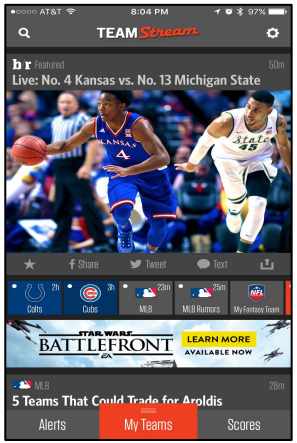
Launch App



Enable Notifications



Add Streams



View Articles

The Data

```
df = pd.read_csv(os.path.join('..', 'Datasets', 'BR_data.csv'))
df = df.set_index('user_identities.identity')
df
```

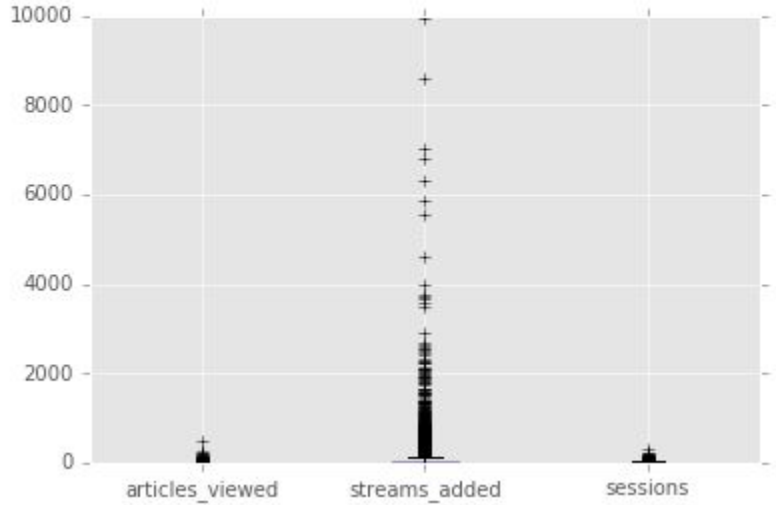
	articles_viewed	streams_added	sessions	greater_than_5_sessions	push_enabled
user_identities.identity					
4.32E+17	1	1	1	0	0
2.39E+18	1	5	1	0	0
5.27E+18	2	5	1	0	0
0000000000000a419698266553970235	3	3	1	0	0
000000000000a6409315927365159953	1	8	1	0	0
...	...	...	...	...	...
ffeeaa76195841efab257d819dac7bbb	2	27	12	1	0
fff0a4ec1e7548b2a8b3245defbeabc1	15	51	9	1	0
fff16bbe71f74b06a2e0b58186be2db9	1	5	2	0	0
fff8033f52ac4d67ae61f2e70559e681	2	3	3	0	0
fffc6535ff9441ee81e321e95374ed77	2	28	6	1	1

25854 rows × 5 columns

EDA

```
df[['articles_viewed', 'streams_added', 'sessions']].plot(kind = 'box')
```

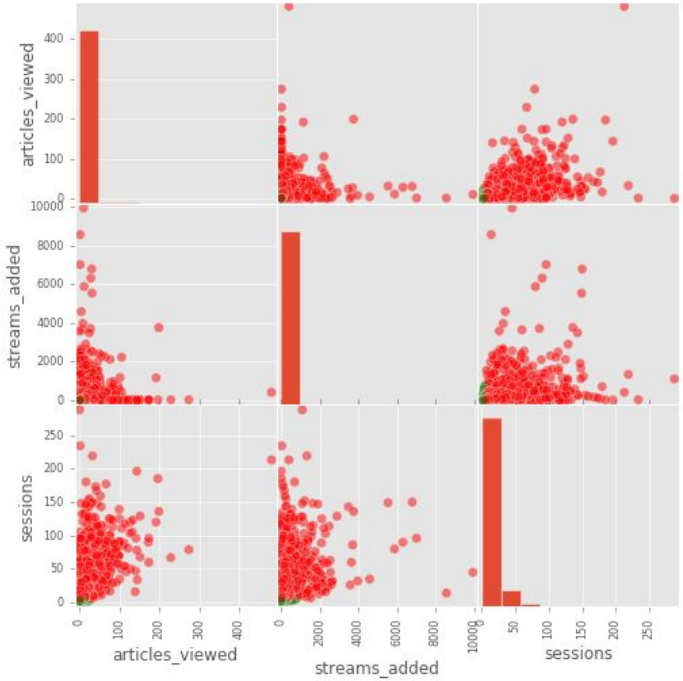
```
<matplotlib.axes._subplots.AxesSubplot at 0x11c470090>
```



EDA

```
pd.tools.plotting.scatter_matrix(df[ ['articles_viewed', 'streams_added', 'sessions'] ], s = 200,
figsize = (8, 8), c = color)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1250d7d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1258a6dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1258344d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x125602610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12541f690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x125484450>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x123b2c4d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x125978690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x125c5d490>]], dtype=object)
```



EDA

```
Q1_sa = df.streams_added.quantile(0.25)
Q3_sa = df.streams_added.quantile(0.75)

IQR_sa = Q3_sa - Q1_sa

IQR_sa
```

```
50.0
```

```
df.drop(df[df.streams_added > Q3_sa + 1.5 * IQR_sa].index, inplace = True)
df.shape[0]
```

```
23025
```

```
Q1_av = df.articles_viewed.quantile(0.25)
Q3_av = df.articles_viewed.quantile(0.75)

IQR_av = Q3_av - Q1_av

IQR_av
```

```
5.0
```

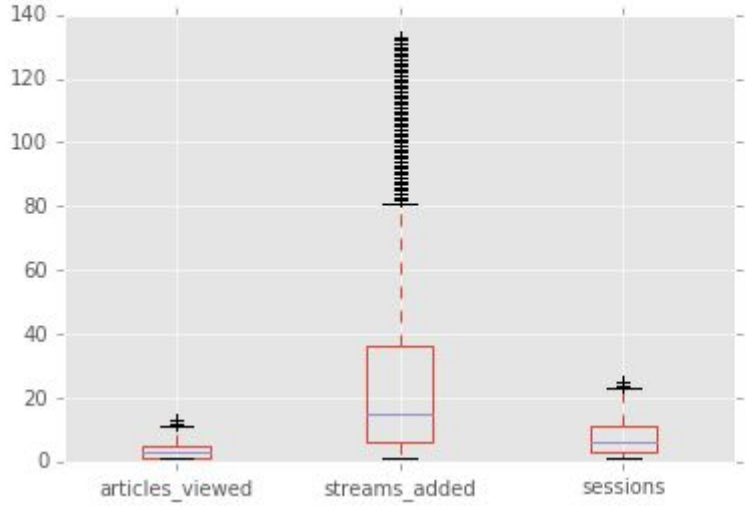
```
df.drop(df[df.articles_viewed > Q3_av + 1.5 * IQR_av].index, inplace = True)
df.shape[0]
```

```
21242
```

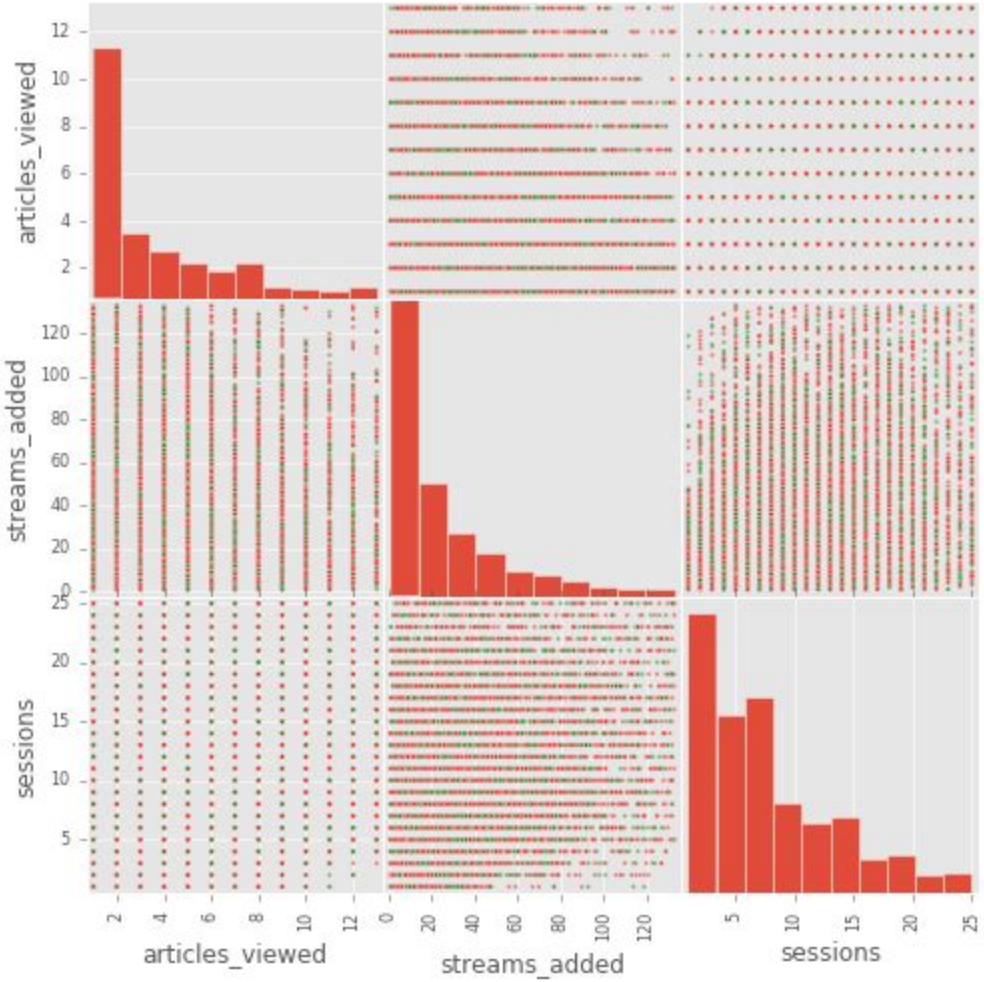
EDA

```
df[['articles_viewed', 'streams_added', 'sessions']].plot(kind = 'box')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11dea2650>
```

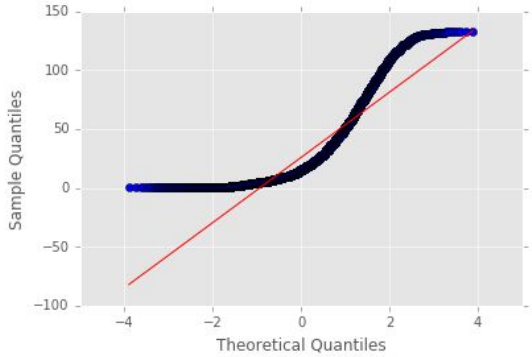


EDA

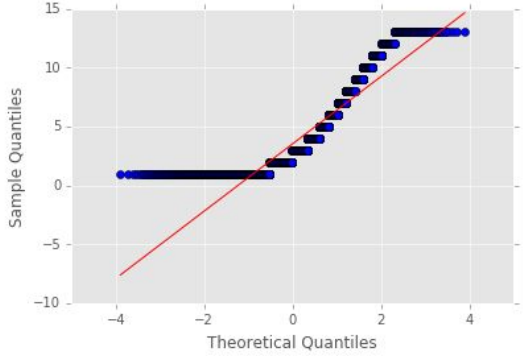

articles_viewed
streams_added
sessions
articles_viewed
streams_added
sessions

EDA

```
sm.qqplot(df.streams_added, line = 's')

pass
```




```
sm.qqplot(df.articles_viewed, line = 's')

pass
```

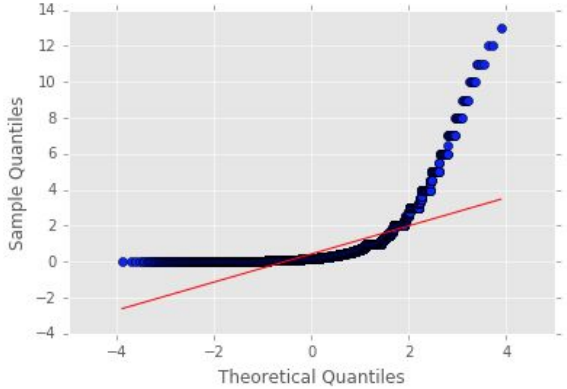


EDA

```
articles_per_stream = df.articles_viewed / df.streams_added
articles_per_stream
```

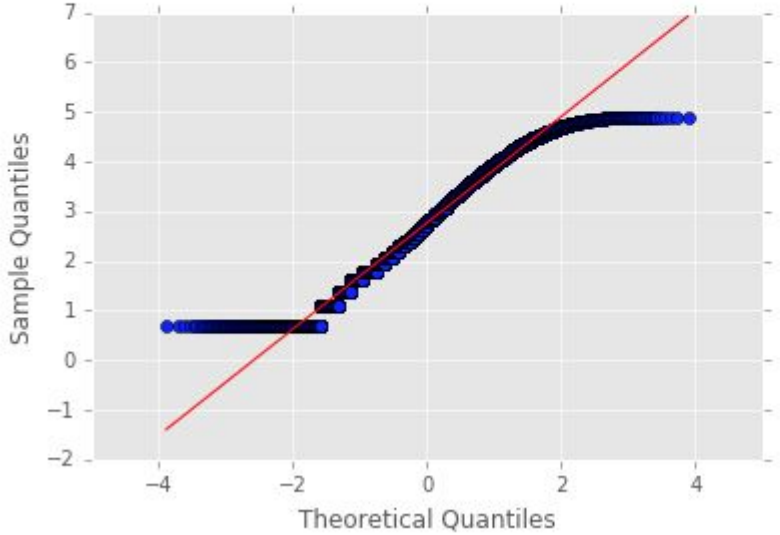
```
user_identities.identity
4.32E+17                                1.000000
2.39E+18                                0.200000
5.27E+18                                0.400000
0000000000000a419698266553970235        1.000000
000000000000a6409315927365159953        0.125000
                                           ...
ffebf65d7954494cabf9c7d932f5c916        0.500000
ffeeaa76195841efab257d819dac7bbb        0.074074
fff16bbe71f74b06a2e0b58186be2db9        0.200000
fff8033f52ac4d67ae61f2e70559e681        0.666667
fffc6535ff9441ee81e321e95374ed77        0.071429
dtype: float64
```

```
sm.qqplot(articles_per_stream, line = 's')

pass
```



EDA

```
sm.qqplot(df.streams_added.apply(lambda x: np.log(1 + x)), line = 's')
#sm.qqplot(df.streams_added, line = 's')


pass
```



Next Steps

- Set up training and test sets
- Random forest and feature importance
- Run cross validation on training set for logistic regression model
- Evaluate model
- Run final model on test set

Setting up training and test sets

```
X = df[df.columns.values]
X.drop('greater_than_5_sessions', axis = 1, inplace = True)

y = df.greater_than_5_sessions
```

```
train_X, test_X, train_y, test_y = cross_validation.train_test_split(X, y, train_size = .6, random_state = 0)
```

60/40 split

Logistic Regression Model #1

```
model = linear_model.LogisticRegression().\
    fit(train_X, train_y)
```

```
print np.exp(model.intercept_)
print np.exp(model.coef_)
```

```
[ 0.18468435]
[[ 1.3486749   1.04257393]]
```

```
model.score(train_X, train_y)
```

```
0.7370733621027854
```

```
y_hat = model.predict(train_X)

pd.crosstab(y_hat,
    train_y,
    rownames = ['Hypothesized Class'],
    colnames = ['True Class'])
```

True Class	0	1
Hypothesized Class		
0	4203	1960
1	1391	5191

Decision Tree and Feature Importance

```
model = tree.DecisionTreeRegressor(random_state = 0).\
    fit(train_X, train_y)
```

```
train_y_hat = model.predict(train_X)
print np.sqrt(metrics.mean_squared_error(train_y, train_y_hat))
```

```
0.401564754718
```

```
sorted(zip(model.feature_importances_, X.columns.values), reverse = True)
```

```
[(0.68213475116320976, 'streams_added'),
 (0.31786524883679018, 'articles_viewed')]
```

```
cross_validation.cross_val_score(model, train_X, train_y, cv = 10).mean()
```

```
0.23622385776101601
```

Logistic Regression Model #2

```
model = linear_model.LogisticRegression().\
    fit(train_X, train_y)
```

```
print np.exp(model.intercept_)
print np.exp(model.coef_)
```

```
[ 0.46020328]
[[ 1.04642719]]
```

```
model.score(train_X, train_y)
```

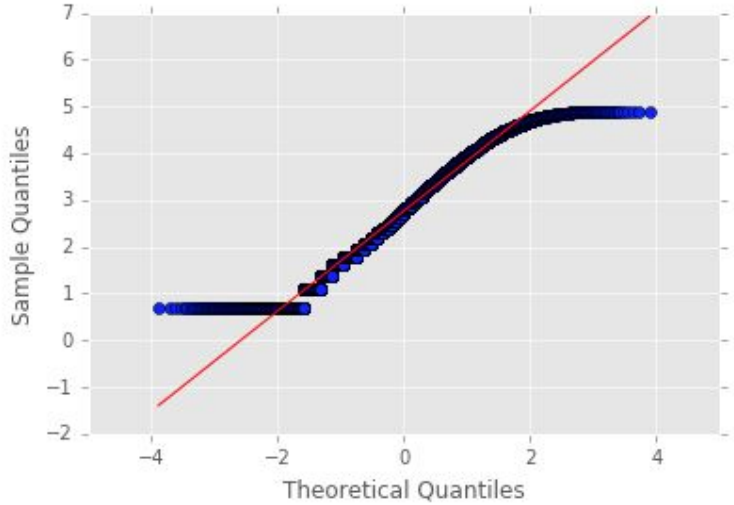
```
0.69776382895253042
```

```
y_hat = model.predict(train_X)

pd.crosstab(y_hat,
    train_y,
    rownames = ['Hypothesized Class'],
    colnames = ['True Class'])
```

True Class	0	1
Hypothesized Class		
0	4255	2513
1	1339	4638

Logistic Regression Model #3

```
sm.qqplot(df.streams_added.apply(lambda x: np.log(1 + x)), line = 's')
#sm.qqplot(df.streams_added, line = 's')


pass
```



Logistic Regression Model #3

```
model = linear_model.LogisticRegression().\
    fit(train_X, train_y)
```

```
print np.exp(model.intercept_)
print np.exp(model.coef_)
```

```
[ 0.01530738]
[[ 3.95730089  2.5831387 ]]
```

```
model.score(train_X, train_y)
```

```
0.74225186347587291
```

```
y_hat = model.predict(train_X)

pd.crosstab(y_hat,
    train_y,
    rownames = ['Hypothesized Class'],
    colnames = ['True Class'])
```

True Class	0	1
Hypothesized Class		
0	3783	1474
1	1811	5677

Running on the Testing Set

```
model.score(test_X, test_y)
```

```
0.73331764152053669
```

```
print 'training misclassification =', 1 - model.score(train_X, train_y)
print 'testing  misclassification =', 1 - model.score(test_X, test_y)
```

```
training misclassification = 0.257748136524
testing  misclassification = 0.266682358479
```

Playing with Predictions and Probabilities

```
predict_1 = [ [0,1] ]

print model.predict(predict_1)
print model.predict_proba(predict_1)
```

```
[0]
[[ 0.96196294  0.03803706]]
```

```
predict_2 = [ [0,2] ]

print model.predict(predict_2)
print model.predict_proba(predict_2)
```

```
[0]
[[ 0.90732565  0.09267435]]
```

```
predict_3 = [ [0,3] ]

print model.predict(predict_3)
print model.predict_proba(predict_3)
```

```
[0]
[[ 0.79123809  0.20876191]]
```

```
predict_4 = [ [0,4] ]

print model.predict(predict_4)
print model.predict_proba(predict_4)
```

```
[0]
[[ 0.5946927  0.4053073]]
```

```
y_hat = model.predict(X)
```

```
p_hat = model.predict_proba(X)[:,1]
```

```
new_df = X.join(pd.DataFrame({'y': y, 'y_hat': y_hat, 'p_hat': p_hat}))
```

```
new_df['c1'] = new_df.y.map({0: 'black', 1: 'red'})
new_df['c2'] = new_df.y_hat.map({0: 'black', 1: 'red'})
```

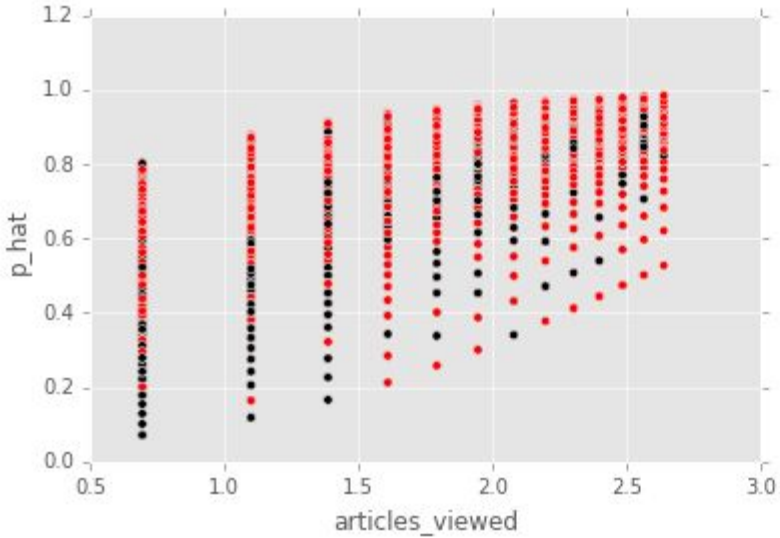
```
new_df
```

	articles_viewed	streams_added	p_hat	y	y_hat	c1	c2
user_identities.identity							
4.32E+17	0.693147	0.693147	0.071216	0	0	black	black
2.39E+18	0.693147	1.791759	0.178644	0	0	black	black
5.27E+18	1.098612	1.791759	0.275315	0	0	black	black
0000000000000a419698266553970235	1.386294	1.386294	0.277502	0	0	black	black
000000000000a64093159273651599 53	0.693147	2.197225	0.242178	0	0	black	black
...	...	...	...	...	...	...	...
ffedae157fae4875b0325db40c81ec81	1.791759	3.850148	0.874250	1	1	red	red
ffeeaa76195841efab257d819dac7bbb	1.098612	3.332205	0.621064	1	1	red	red
fff16bbe71f74b06a2e0b58186be2db9	0.693147	1.791759	0.178644	0	0	black	black
fff8033f52ac4d67ae61f2e70559e681	1.098612	1.386294	0.205444	0	0	black	black
fffc6535ff9441ee81e321e95374ed77	1.098612	3.367296	0.628870	1	1	red	red

21242 rows × 7 columns

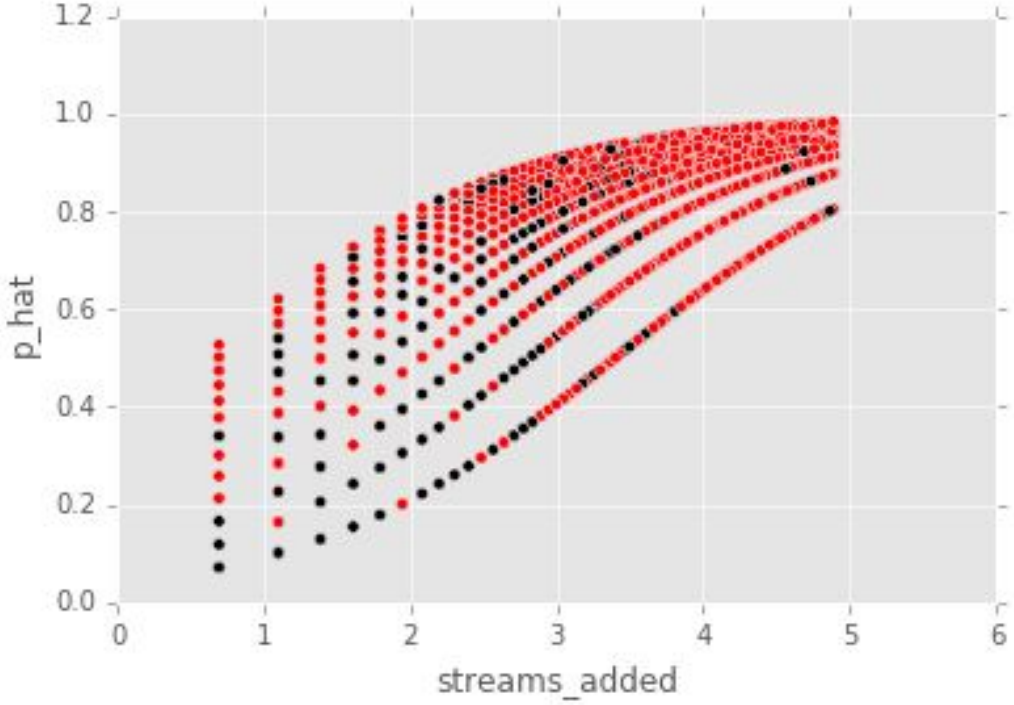
```
new_df.plot(x = 'articles_viewed', y = 'p_hat', kind = 'scatter', c = new_df.c1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24ee4860>
```



```
new_df.plot(x = 'streams_added', y = 'p_hat', kind = 'scatter', c = new_df.c1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11cbf710>
```



Conclusions

- Streams added explain more of the variance in retained users, but articles viewed is somewhat important too
- Probability that user will be retained goes up as a user adds more streams / views more articles
- The model gets more accurate as a user adds > ~8 streams
- Assuming these are reliable conclusions, we can design features to encourage new users to add at least 8 streams

- Need to acquire and clean more data -> push_enabled issues
- Can’t screw up training and test sets in future analyses

Next Steps

- Acquire more data
- More cleaning ➜ test accounts
- Expand list of features (time per session, time between sessions, push notification counts, etc.)
- Re-run models
- Design in-app experiments see if model holds true