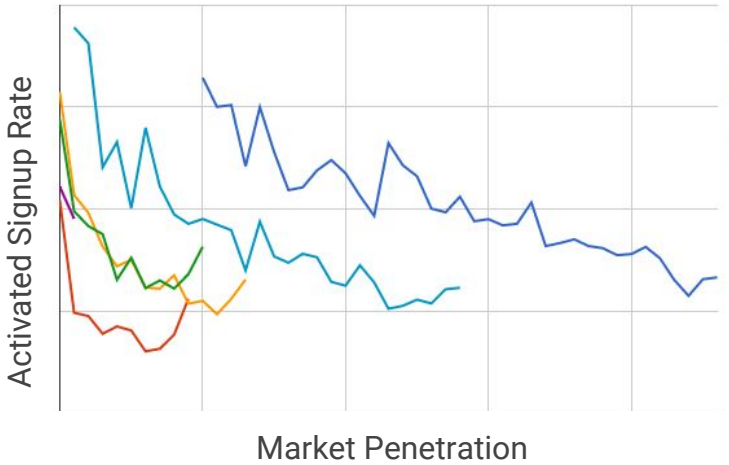
Activated Signups Prediction

Goal

Identify the model that predicts signup or activated signup*



* Activated Signup : users who was active in their 4th week after signup.

* Activated Signup Rate = Activated Signups / Signups

Data

(102295, 7)

	refer_type	page_type	search_type	country	category	duration	activated_signup
0	GOOGLE		unknown	GB	HAIR_BEAUTY	over_5min	1
2	GOOGLE		unknown	TH	ART	1min_to_3min	0
7	GOOGLE		unknown	US	FOOD_DRINK	1min_to_3min	0
8	GOOGLE		unknown	US	DIY_CRAFTS	over_5min	0
9	DIRECT		unknown	US	WOMENS_FASHION	over_5min	0

Signup sessions data on 7/1/2016

activated_signup = 1, if the use has actions between 7/22 and 7/28

Data

	activated_signup
count	102295.000000
mean	0.132255
std	0.338769
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Conversion Rate = 0.13

```
print SignupData.shape
```

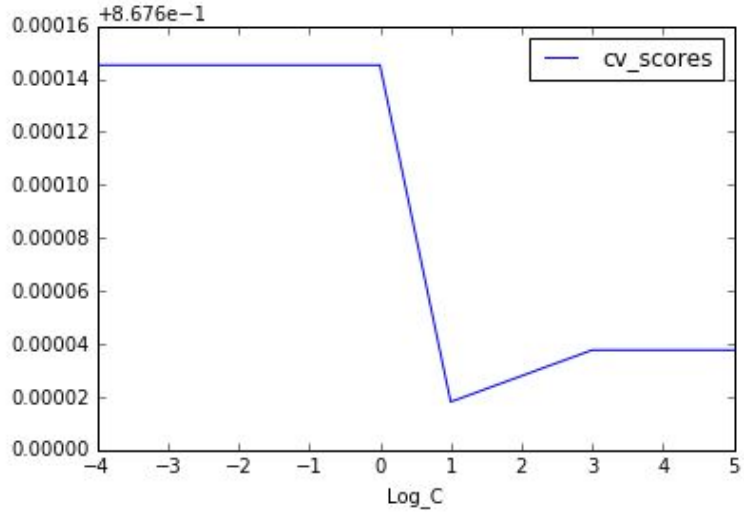
```
(102295, 268)
```

Logistic Regression

CV score by changing C

```
0.867745247456 5.60613107681
0.867745247456 12.0417211056
0.867745247456 27.1350998878
0.867745247456 94.7220349312
0.867745248411 197.150823116
0.867618167365 310.803071976
0.867627943491 430.512274027
0.867637718662 541.536948919
0.867637718662 653.438175917
0.867637718662 758.160063028
```

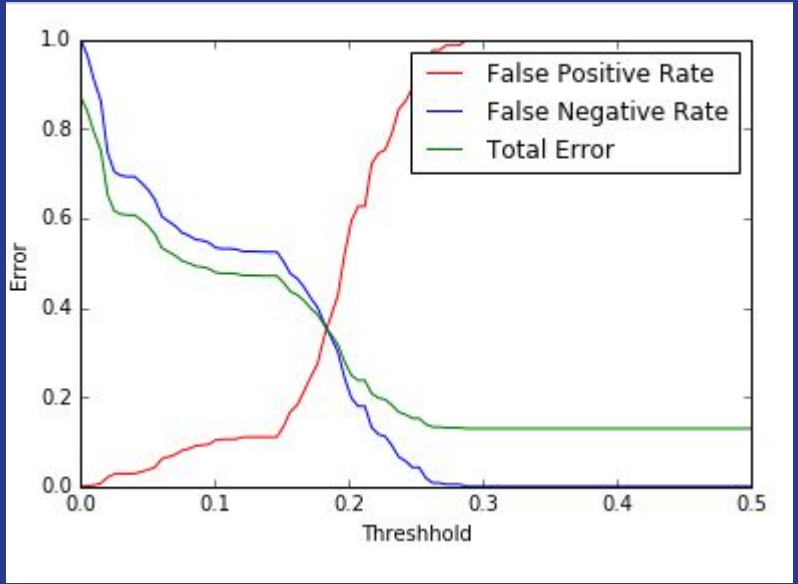
```
<matplotlib.axes._subplots.AxesSubplot at 0x11817a710>
```



Threshold

Confusion Matrix

```
array([[88766,     0],
       [13527,     2]])
```



Threshold = 0.25

```
[[86219  2547]
 [12551   978]]
```

Threshold = 0.15

```
[[46133 42633]
 [ 2479 11050]]
```

Logistic Regression

lm = LogisticRegression(C = 1, penalty = "l1")

```
(0.35284489159093213, 'category_EDUCATION'),
(0.35751096435273061, 'category_DESIGN'),
(0.36389969511080822, 'country_RE'),
(0.36502488626054069, 'country_BO'),
(0.37006237653409335, 'country_VE'),
(0.38974819836277375, 'country_AO'),
(0.40844718442187061, 'country_SD'),
(0.40904374080139561, 'country_CV'),
(0.46445886518871965, 'country_OM'),
(0.52056906602024611, 'country_GY'),
(0.63416031543126816, 'country_BA'),
(0.64140467799845124, 'country_AF'),
(0.68384037365509087, 'country_FJ'),
(0.77829772847371148, 'country_ZW'),
(0.96141049796668021, 'country_NC'),
(1.0819972609960393, 'country_KG'),
(1.0846816689285714, 'country_BI'),
(1.1321740215854936, 'duration_3min_to_5min'),
(1.1984547499346265, 'country_BJ'),
(2.3207343281713468, 'duration_over_5min')]
```

```
(-2.0155859736136073, 'duration_30s_to_1min'),
(-1.1738456781808744, 'duration_10s_to_30s'),
(-1.1390789879885599, 'country_MO'),
(-1.0138877999755382, 'country_JO'),
(-0.72538182679051399, 'country_KW')]
```

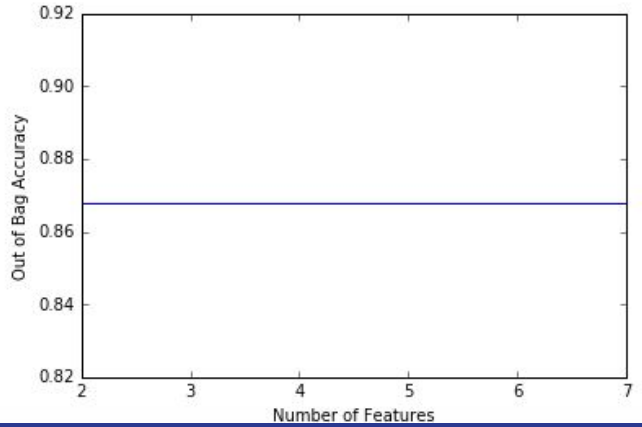
Random Forest

```
###  Model 2 : Random Forest ###

from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import cross_val_score

RFClass = RandomForestClassifier(n_estimators = 1000,
                                 max_features = 4, # You
                                 min_samples_leaf = 10,
                                 oob_score = True,
                                 random_state = 1,
                                 n_jobs = -1)
RFClass.fit(X, y)
print("Out of Bag Accuracy  = %f" %RFClass.oob_score_)
scores = cross_val_score(RFClass, X, y, cv = 10)
print("Cross-validation Accuracy = %f" %scores.mean())
```

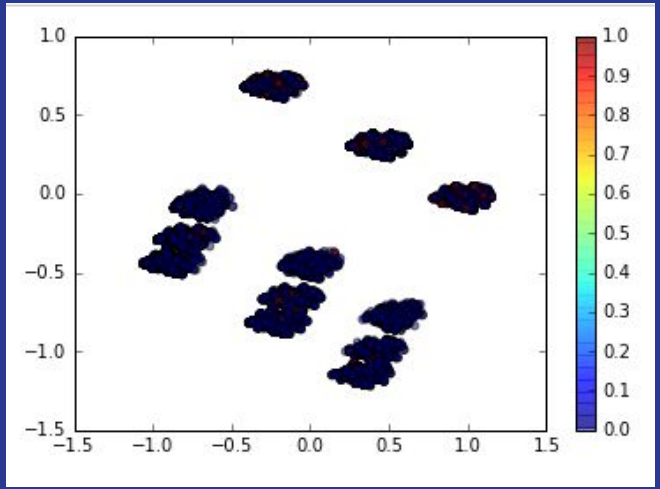
```
Out of Bag Accuracy  = 0.867745
Cross-validation Accuracy = 0.867745
```

Random Forest

```
Features = range(2,8)
oob_score_RF = []
for i in Features:
        RFClass = RandomForestClassifier(n_estimators = 1000,
                                max_features = i,      #How many fea
                                min_samples_leaf = 5, #Minimum numb
                                oob_score = True,
                                random_state = 1,
                                n_jobs = -1)
        RFClass.fit(X,y)
        oob_score_RF.append(RFClass.oob_score_)

plt.plot(Features, oob_score_RF)
plt.xlabel("Number of Features")
plt.ylabel("Out of Bag Accuracy")
plt.show()
```



PCA & KNN




```
:
from sklearn import neighbors, metrics
misclass = []

for i in range(1, 41, 10):
    clf = PCA(i)
    X_trans = clf.fit_transform(X)
    knn = neighbors.KNeighborsClassifier(n_neighbors = 3, weights='uniform')
    knn.fit(X_trans,y)
    MisClassificationError = 1 - (cross_val_score(knn, X_trans, y, cv=10).mean())
    misclass.append(MisClassificationError)

print(min(misclass))
```

```
0.168942693584
```