NBA 3 POINT REVOLUTION USING LOGISTICS REGRESSION

Mod 5 Presentation

Lim Zheng Wei

Intro

- 3 PT was first introduced in 1979.
- Only 1 3 PT attempt(3PTA) during the 1979 NBA Finals
- 2017/18 Houston Rockets have more 3PTA than all the teams combined in the 1980s
- Also the 1st NBA team in history to attempt more 3s than 2s
- Players are more comfortable shooting from 3PT line and beyond

When all your friends are drunk and you're the designated driver.

Like 0 Pin It Via hiphopmemesdaily

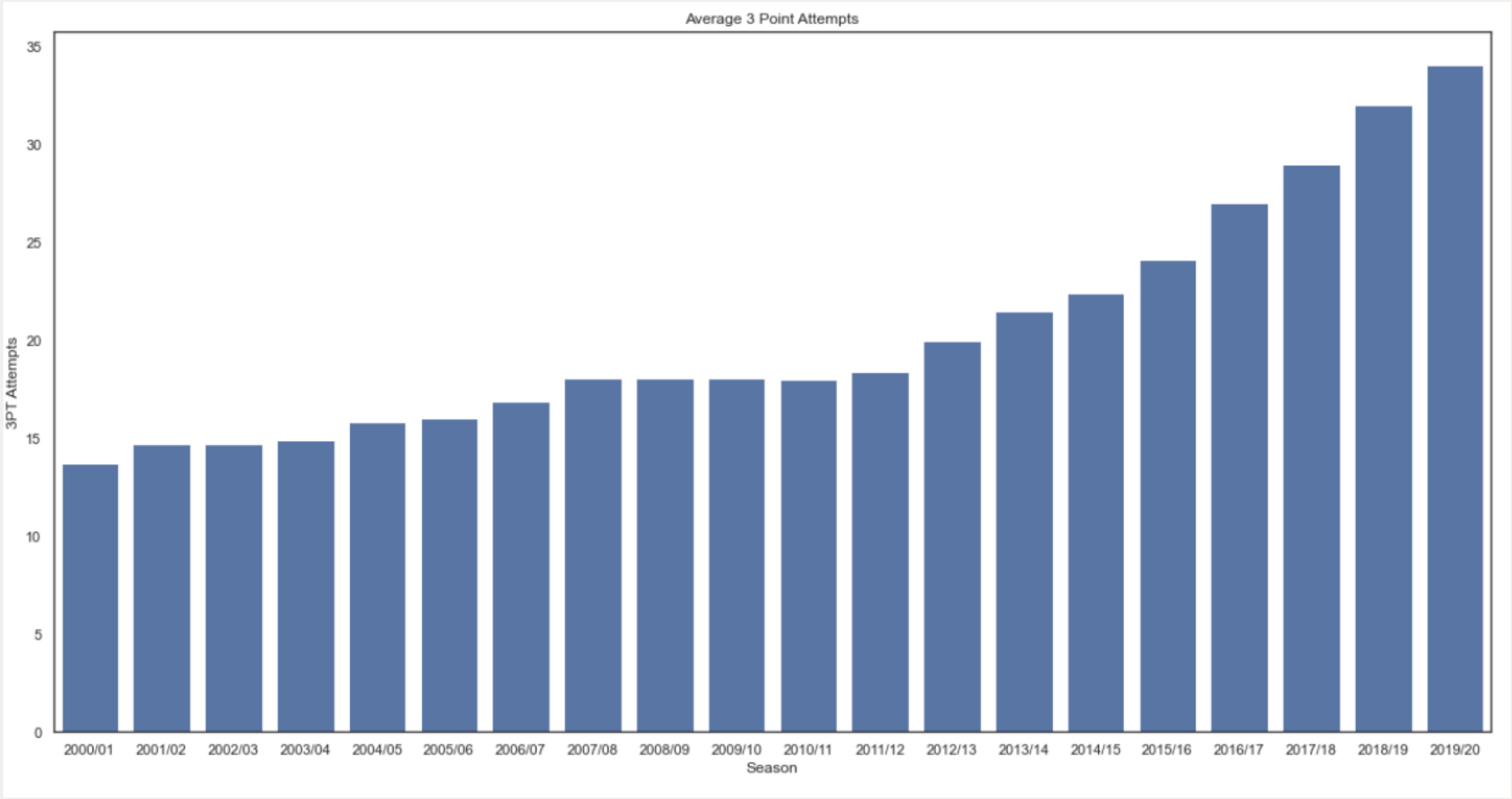
When she ask to go through your phone

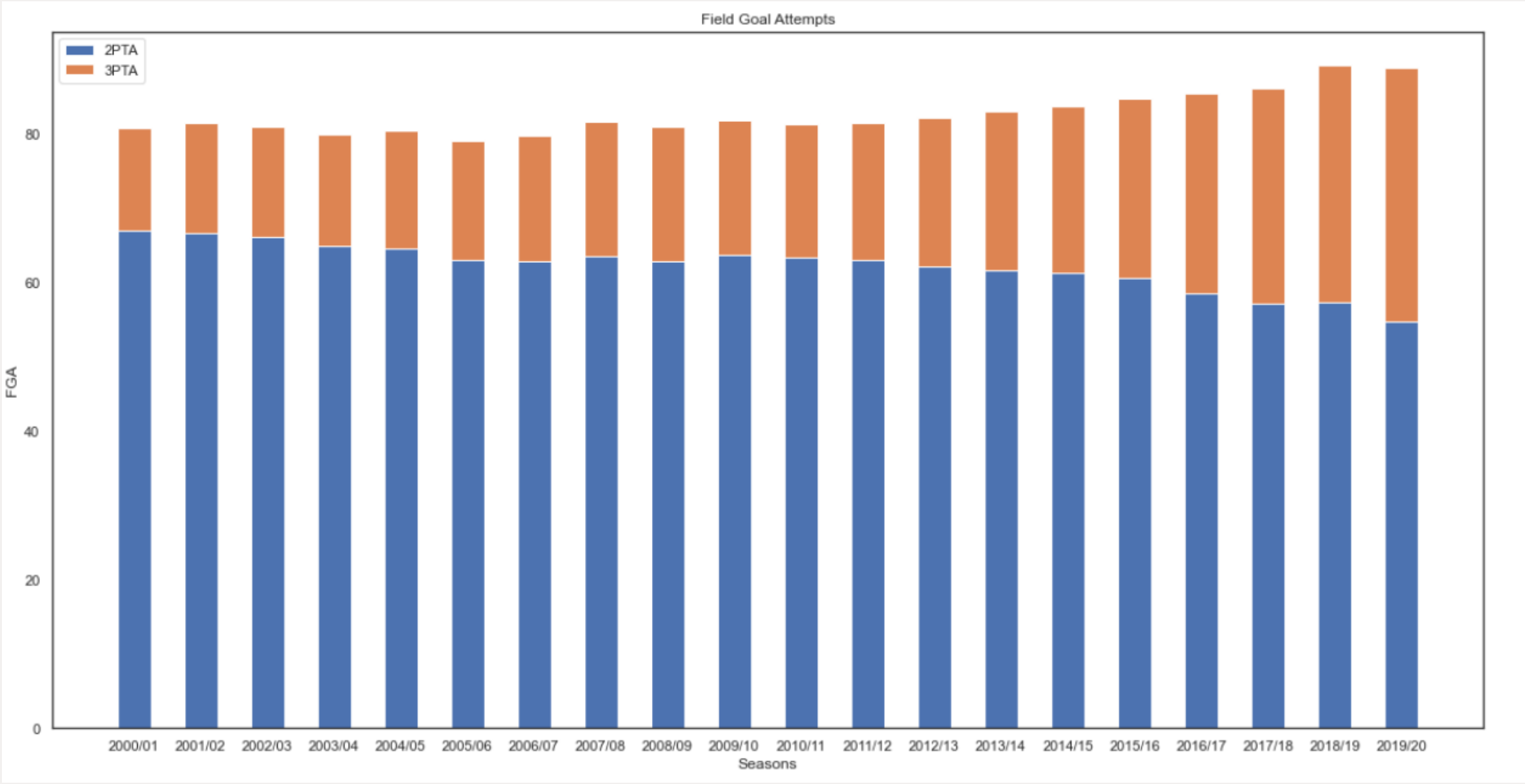
Like 0 Pin It Via thepearlsession

When you bring in donuts for the office

Like 0 Pin It Via revparblems

Like 0 Pin It Via knowyourmeme


Average 3 Point Attempts
3PT Attempts
0
5
10
15
20
25
30
35
2000/01
2001/02
2002/03
2003/04
2004/05
2005/06
2006/07
2007/08
2008/09
2009/10
2010/11
2011/12
2012/13
2013/14
2014/15
2015/16
2016/17
2017/18
2018/19
2019/20
Season


Field Goal Attempts
2PTA
3PTA
FGA
0
20
40
60
80
2000/01
2001/02
2002/03
2003/04
2004/05
2005/06
2006/07
2007/08
2008/09
2009/10
2010/11
2011/12
2012/13
2013/14
2014/15
2015/16
2016/17
2017/18
2018/19
2019/20
Seasons

							Per Game															Shooting			
Rk	Season	Lg	Age	Ht	Wt	G	MP	FG	FGA	3P	3PA	FT	FTA	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	FG%	3P%	T%	Pace
1	2019-20	NBA	26.1	6-6	216	1059	241.8	40.9	88.8	12.2	34.1	7.9	23.1	10.1	34.8	44.8	24.4	7.6	4.9	14.5	20.8	111.8	.460	.358	.773	100.3
2	2018-19	NBA	26.3	6-6	217	1230	241.6	41.1	89.2	11.4	32.0	7.7	23.1	10.3	34.8	45.2	24.6	7.6	5.0	14.1	20.9	111.2	.461	.355	.766	100.0
3	2017-18	NBA	26.4	6-7	219	1230	241.4	39.6	86.1	10.5	29.0	6.6	21.7	9.7	33.8	43.5	23.2	7.7	4.8	14.3	19.9	106.3	.460	.362	.767	97.3
4	2016-17	NBA	26.6	6-7	220	1230	241.6	39.0	85.4	9.7	27.0	7.8	23.1	10.1	33.4	43.5	22.6	7.7	4.7	14.0	19.9	105.6	.457	.358	.772	96.4
5	2015-16	NBA	26.7	6-7	221	1230	241.8	38.2	84.6	8.5	24.1	7.7	23.4	10.4	33.3	43.8	22.3	7.8	5.0	14.4	20.3	102.7	.452	.354	.757	95.8
6	2014-15	NBA	26.7	6-7	222	1230	242.0	37.5	83.6	7.8	22.4	7.1	22.8	10.9	32.4	43.3	22.0	7.7	4.8	14.4	20.2	100.0	.449	.350	.750	93.9
7	2013-14	NBA	26.5	6-7	223	1230	242.0	37.7	83.0	7.7	21.5	7.8	23.6	10.9	31.8	42.7	22.0	7.7	4.7	14.6	20.7	101.0	.454	.360	.756	93.9
8	2012-13	NBA	26.7	6-7	223	1229	241.9	37.1	82.0	7.2	20.0	6.7	22.2	11.2	31.0	42.1	22.1	7.8	5.1	14.6	19.8	98.1	.453	.359	.753	92.0
9	2011-12	NBA	26.6	6-7	223	990	241.9	36.5	81.4	6.4	18.4	6.9	22.5	11.4	30.8	42.2	21.0	7.7	5.1	14.6	19.6	96.3	.448	.349	.752	91.3
10	2010-11	NBA	26.6	6-7	223	1230	241.9	37.2	81.2	6.5	18.0	8.6	24.4	10.9	30.5	41.4	21.5	7.3	4.9	14.3	20.7	99.6	.459	.358	.763	92.1
11	2009-10	NBA	26.6	6-7	222	1230	241.7	37.7	81.7	6.4	18.1	8.6	24.5	11.0	30.8	41.7	21.2	7.2	4.9	14.2	20.9	100.4	.461	.355	.759	92.7
12	2008-09	NBA	26.6	6-7	221	1230	241.7	37.1	80.9	6.6	18.1	9.1	24.7	11.0	30.3	41.3	21.0	7.3	4.8	14.0	21.0	100.0	.459	.367	.771	91.7
13	2007-08	NBA	26.8	6-7	220	1230	241.5	37.3	81.5	6.6	18.1	8.8	24.9	11.2	30.8	42.0	21.8	7.3	4.7	14.1	21.0	99.9	.457	.362	.755	92.4
14	2006-07	NBA	26.6	6-7	219	1230	242.2	36.5	79.7	6.1	16.9	9.6	26.1	11.1	29.9	41.1	21.3	7.2	4.6	15.1	22.2	98.7	.458	.358	.752	91.9
15	2005-06	NBA	26.5	6-7	220	1230	242.1	35.8	79.0	5.7	16.0	9.6	26.3	11.2	29.8	41.0	20.6	7.2	4.7	14.4	22.8	97.0	.454	.358	.745	90.5
16	2004-05	NBA	26.9	6-7	220	1230	241.9	35.9	80.3	5.6	15.8	9.7	26.1	12.0	29.8	41.9	21.3	7.5	4.9	14.5	22.6	97.2	.447	.356	.756	90.9
17	2003-04	NBA	27.0	6-7	219	1189	241.7	35.0	79.8	5.2	14.9	8.2	24.2	12.1	30.1	42.2	21.3	7.9	5.1	15.0	21.4	93.4	.439	.347	.752	90.1
18	2002-03	NBA	27.2	6-7	219	1189	242.0	35.7	80.8	5.1	14.7	8.5	24.4	12.0	30.3	42.3	21.5	7.9	5.0	14.9	21.8	95.1	.442	.349	.758	91.0
19	2001-02	NBA	27.4	6-7	218	1189	241.7	36.2	81.3	5.2	14.7	7.9	23.8	12.2	30.2	42.4	21.9	7.8	5.2	14.5	21.2	95.5	.445	.354	.752	90.7
20	2000-01	NBA	27.7	6-7	216	1189	242.0	35.7	80.6	4.8	13.7	8.6	24.9	12.0	30.5	42.5	21.8	7.8	5.3	15.0	22.3	94.8	.443	.354	.748	91.3

Why shoot more 3s?

- Floor spacing
- Allows non-shooters more room to operate/anyone to attack the paint
- 3 > 2

18 Apr 2009


TD Banknorth GARDEN
HERE WILL AMAZING
PLAYOFFS
ESPN
CHI 63
BOS 60
3RD
7:08
:15
PLAYOFFS
EAST 1st ROUND - GAME 1

18 Apr 2009


TD Banknorth GARDEN
ESPN
CHI 63
BOS 60
3RD
7:03
:10
PLAYOFFS
EAST 1st ROUND - GAME 1

6 Feb 2020

Austin Rivers: 35.6%
Ben McLemore: 40%
James Harden: 35.5%
PJ Tucker: 35.8%
Robert Covington: 33.5%

29 Feb 2020

Austin Rivers: 35.6%
Ben McLemore: 40%
Eric Gordon: 31.7%
Jeff Green: 35.4%
Russell Westbrook: 25.5%

Questions

1. Does outshooting your opponent = win?
2. Does making more 3s than your opponent = win?
3. Does having a higher 3PT % than your opponent = win?
4. Does shooting the 3PT well(>=50%) = win?
5. Does having more assists than your opponent = win?

Notice they are all Naïve Bayes Theorem.

5 questions, I shall use 1 different technique for each question for illustration purposes.

- Import relevant libraries and data

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime, date
from sklearn import svm
from sklearn.datasets import make_blobs
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB


%matplotlib inline
```

```
nba_csv = 'C:/Users/zheng/Desktop/Data Science/Presentations/Mod 5//nba.games.stats.csv'

nba = pd.read_csv(nba_csv)

nba.head()
```

	Unnamed: 0	Team	Game	Date	Home	Opponent	WINorLOSS	TeamPoints	OpponentPoints	FieldGoals	...	Opp.FreeThrows	Opp.FreeThrowsAttempted	O
0	1	ATL	1	2014-10-29	Away	TOR	L	102	109	40	...	27	33	
1	2	ATL	2	2014-11-01	Home	IND	W	102	92	35	...	18	21	
2	3	ATL	3	2014-11-05	Away	SAS	L	92	94	38	...	27	38	
3	4	ATL	4	2014-11-07	Away	CHO	L	119	122	43	...	20	27	
4	5	ATL	5	2014-11-08	Home	NYK	W	103	96	33	...	8	11	

5 rows × 41 columns

Perform EDA

- nba.isnull().sum()
- nba.shape
- (9840, 41)

```
Unnamed: 0                0
Team                      0
Game                      0
Date                      0
Home                      0
Opponent                  0
WINorLOSS                 0
TeamPoints                0
OpponentPoints            0
FieldGoals                0
FieldGoalsAttempted       0
FieldGoals.               0
X3PointShots              0
X3PointShotsAttempted     0
X3PointShots.             0
FreeThrows                0
FreeThrowsAttempted       0
FreeThrows.               0
OffRebounds               0
TotalRebounds             0
Assists                   0
Steals                    0
Blocks                    0
Turnovers                 0
TotalFouls                0
Opp.FieldGoals            0
Opp.FieldGoalsAttempted   0
Opp.FieldGoals.           0
Opp.3PointShots           0
Opp.3PointShotsAttempted  0
Opp.3PointShots.          0
Opp.FreeThrows            0
Opp.FreeThrowsAttempted   0
Opp.FreeThrows.           0
Opp.OffRebounds           0
Opp.TotalRebounds         0
Opp.Assists               0
Opp.Steals                0
Opp.Blocks                0
Opp.Turnovers             0
Opp.TotalFouls            0
dtype: int64
```

Make date as index

```
# convert date to datetime, make them as index as well
```

```
nba = nba[['Date', 'WINorLOSS', 'X3PointShots', 'X3PointShotsAttempted', 'X3PointShots.', 'Opp.3PointShots', 'Opp.3PointShotsAtte
```

```
nba.head()
```

	Date	WINorLOSS	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.
0	2014-10-29	L	13	22	0.591	8	26	0.308
1	2014-11-01	W	7	20	0.350	12	32	0.375
2	2014-11-05	L	8	25	0.320	5	17	0.294
3	2014-11-07	L	13	33	0.394	6	21	0.286
4	2014-11-08	W	9	22	0.409	8	21	0.381

```
nba['Date'] = pd.to_datetime(nba['Date'], format = '%Y-%m-%d')
```

```
nba = nba.set_index('Date')
```

```
nba.head()
```

Date	WINorLOSS	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.
2014-10-29	L	13	22	0.591	8	26	0.308
2014-11-01	W	7	20	0.350	12	32	0.375
2014-11-05	L	8	25	0.320	5	17	0.294
2014-11-07	L	13	33	0.394	6	21	0.286
2014-11-08	W	9	22	0.409	8	21	0.381

Set dummy variable for Win / Loss

```
nba = pd.get_dummies(nba, columns=['WINorLOSS'], prefix=['WINorLOSS'])
```

```
nba.head()
```

Date	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.	WINorLOSS_L	WINorLOSS_W
2014-10-29	13	22	0.591	8	26	0.308	1	0
2014-11-01	7	20	0.350	12	32	0.375	0	1
2014-11-05	8	25	0.320	5	17	0.294	1	0
2014-11-07	13	33	0.394	6	21	0.286	1	0
2014-11-08	9	22	0.409	8	21	0.381	0	1

```
nba.shape
```

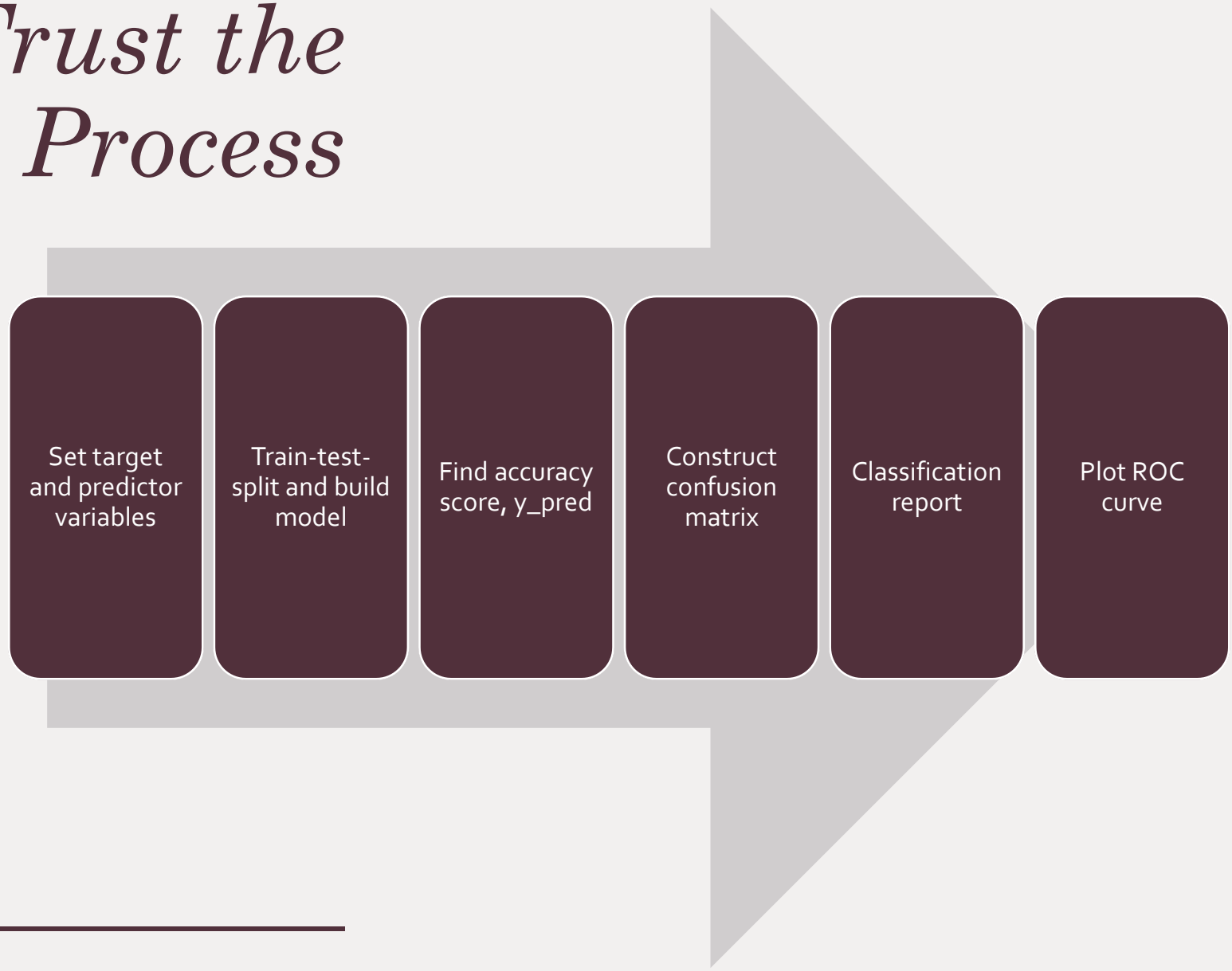
```
(9840, 8)
```

nba.corr()

	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.	WINorLOSS_L	WINorLOSS_W
X3PointShots	1.000000	0.747013	0.685878	0.086748	0.115195	0.004273	-0.240486	0.240486
X3PointShotsAttempted	0.747013	1.000000	0.068394	0.115195	0.142953	0.021016	-0.032582	0.032582
X3PointShots.	0.685878	0.068394	1.000000	0.004273	0.021016	-0.020231	-0.317418	0.317418
Opp.3PointShots	0.086748	0.115195	0.004273	1.000000	0.747013	0.685878	0.240486	-0.240486
Opp.3PointShotsAttempted	0.115195	0.142953	0.021016	0.747013	1.000000	0.068394	0.032582	-0.032582
Opp.3PointShots.	0.004273	0.021016	-0.020231	0.685878	0.068394	1.000000	0.317418	-0.317418
WINorLOSS_L	-0.240486	-0.032582	-0.317418	0.240486	0.032582	0.317418	1.000000	-1.000000
WINorLOSS_W	0.240486	0.032582	0.317418	-0.240486	-0.032582	-0.317418	-1.000000	1.000000

Predict: This data is going to be bad

Trust the Process



Q1 Does outshooting your opponent in 3s result = win?

- We shall use Logistic Regression for this question

```
outshoot = nba['X3PointShotsAttempted'] > nba['Opp.3PointShotsAttempted']
```

```
nba_Q1 = nba[outshoot]

nba_Q1.head()
```

Date	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.	WINorLOSS_L	WINorLOSS_W
2014-11-05	8	25	0.320	5	17	0.294	1	0
2014-11-07	13	33	0.394	6	21	0.286	1	0
2014-11-08	9	22	0.409	8	21	0.381	0	1
2014-11-10	10	27	0.370	6	26	0.231	0	1
2014-11-14	11	28	0.393	10	21	0.476	0	1

Step It Up like 1,2,3

```
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size = 0.2, random_state = 1)

logreg = LogisticRegression()
logreg.fit(X1_train, Y1_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
logreg.intercept_
```

```
array([-0.04598634])
```

```
logreg.coef_
```

```
array([[0.00575189]])
```

```
y_pred1 = logreg.predict(X1_train)

accuracy_score_train = accuracy_score(Y1_train, y_pred1)

accuracy_score_train
```

```
0.5308674827035658
```

```
y_pred_test1 = logreg.predict(X1_test)
accuracy_score_test = accuracy_score(Y1_test, y_pred_test1)

accuracy_score_test
```

```
0.5085106382978724
```

Confusing?

```
cm = metrics.confusion_matrix(Y1_test, y_pred_test1)

print(cm)
```

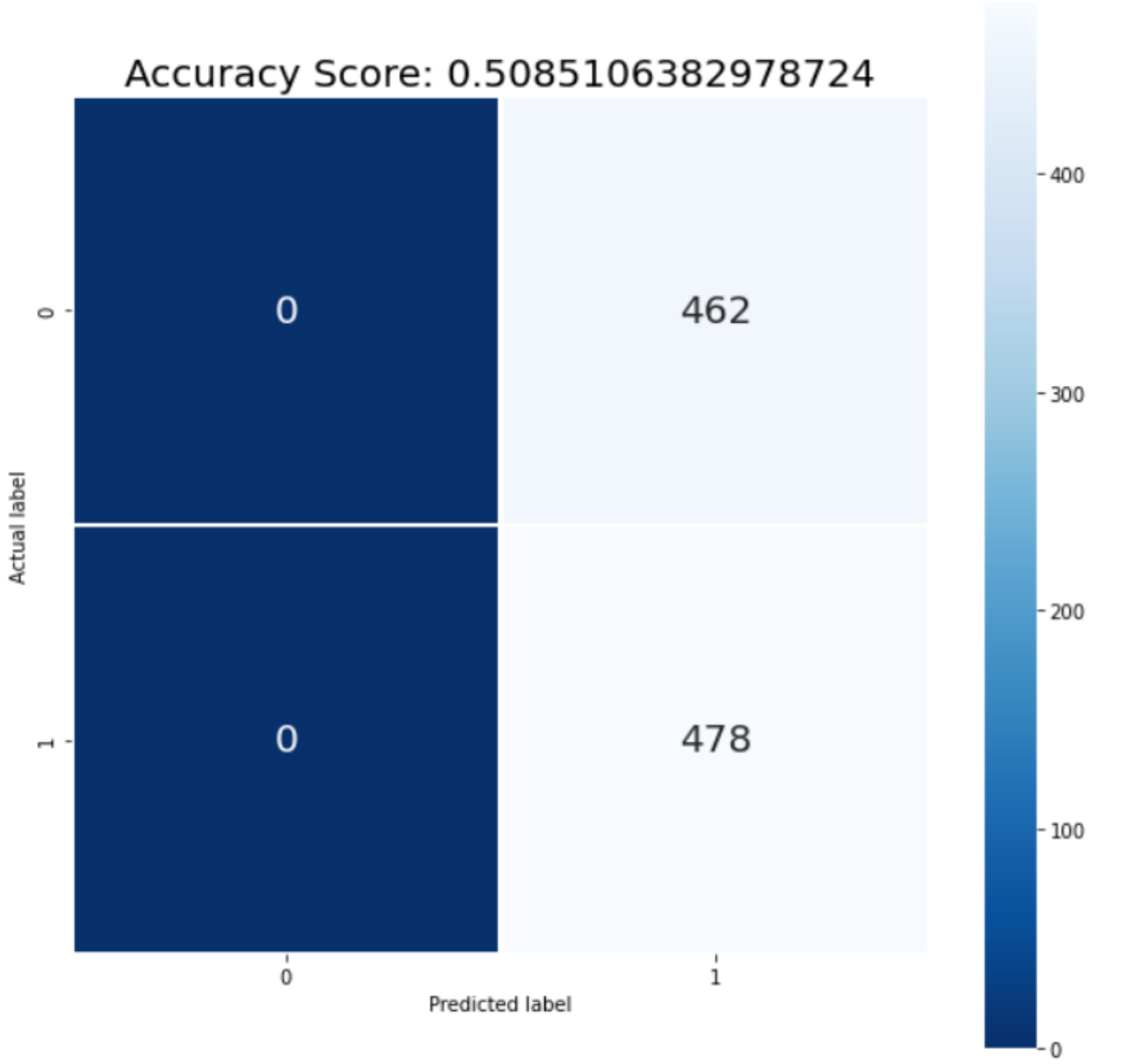
```
[[  0 462]
 [  0 478]]
```

```
# https://seaborn.pydata.org/generated/seaborn.heatmap.html
```

```
_ = plt.figure(figsize=(10,10))
_ = sns.heatmap(cm, annot=True, fmt='d', linewidths=2, square = True, cmap = 'Blues_r', annot_kws={"fontsize":20});
_ = plt.ylabel('Actual label');
_ = plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(accuracy_score_test)
_ = plt.title(all_sample_title, size = 20);
```

True Negative

False Positive



False Negative

True Positive

Report please

```
print(classification_report(Y1_test, y_pred_test1))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	462
1	0.51	1.00	0.67	478
accuracy			0.51	940
macro avg	0.25	0.50	0.34	940
weighted avg	0.26	0.51	0.34	940

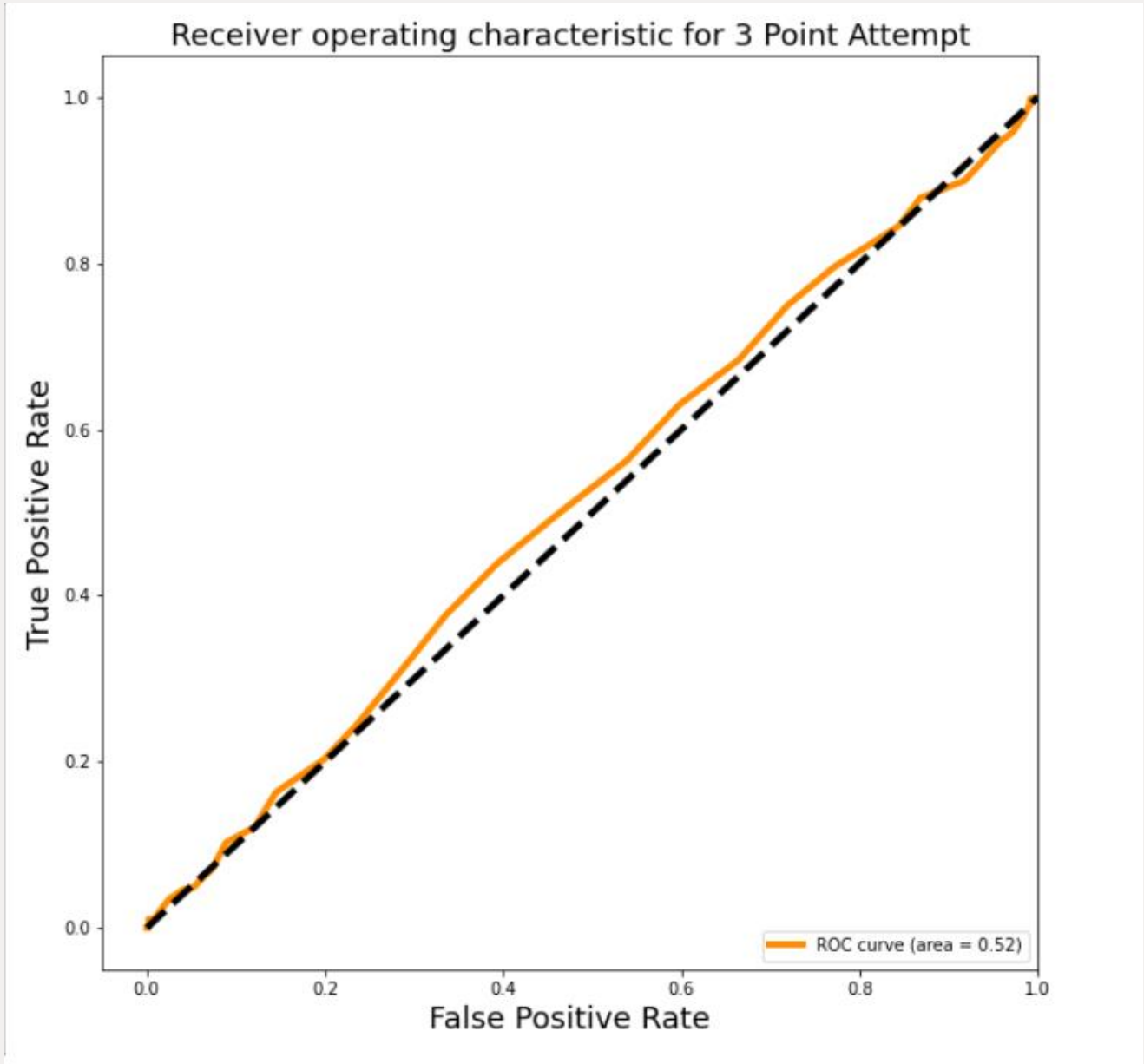
Recall(no pun intended):

Precision : How good at predicting positive outcomes when it's actually positive - TP/(TP+FP)

Recall : How good at predicting actual positive outcomes - TP/(TP+FN)

ROC that curve

```
# For class 1, find the area under the curve
preds1 = logreg.predict_proba(X1_test)[:,1]
# Find fpr, tpr
fpr, tpr, _ = metrics.roc_curve(Y1_test, preds1)

# Find auc
roc_auc = metrics.auc(fpr, tpr)

# Plot of a ROC curve for class 1 (won the game)
plt.figure(figsize=[10,10])

# Plot fpr, tpr
plt.plot(fpr, tpr, color='darkorange', linewidth=4, label = 'ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--', linewidth=4)
plt.xlim([-0.05, 1.0])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate', fontsize=18)
plt.ylabel('True Positive Rate', fontsize=18)
plt.title('Receiver operating characteristic for 3 Point Attempt', fontsize=18)
plt.legend(loc="lower right")
plt.show()
```



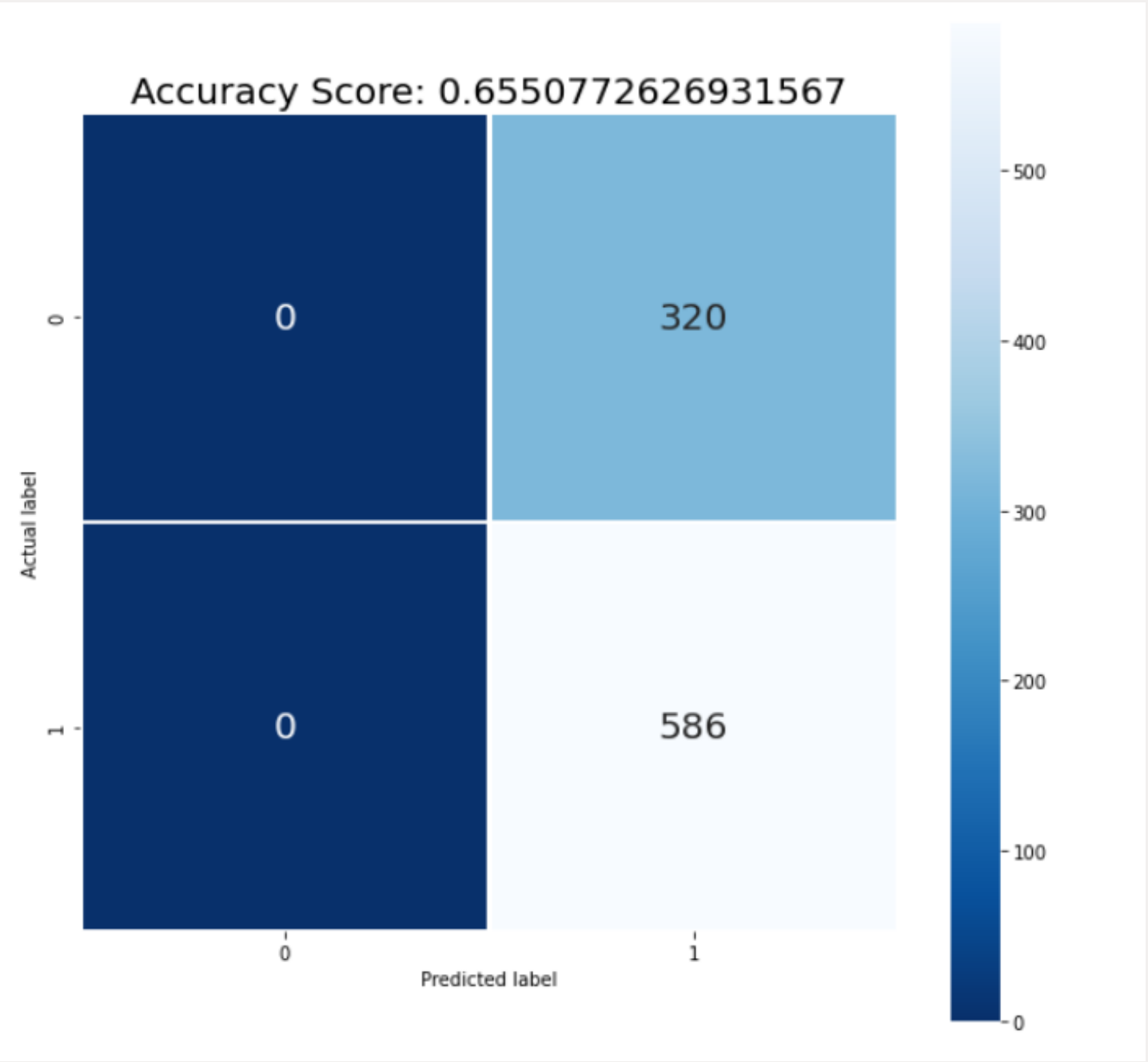
Q2 Does making more 3s than your opponent = win?

- We shall use SVM Linear for this question.

```
makemore = nba['X3PointShots'] > nba['Opp.3PointShots']
```

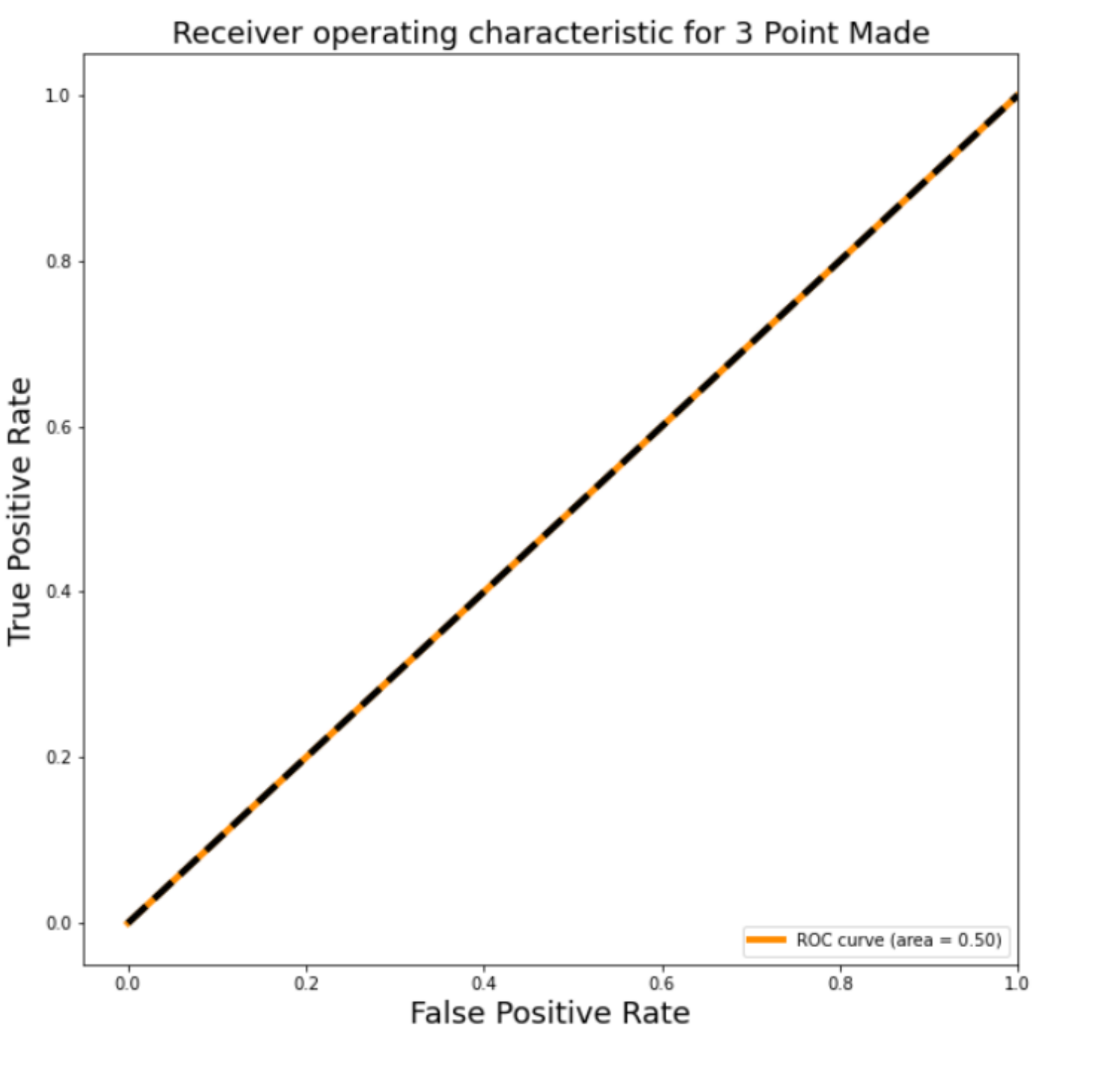
```
nba_Q2 = nba[makemore]

nba_Q2.head()
```

Date	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.	WlNorLOSS_L	WlNorLOSS_W
2014-10-29	13	22	0.591	8	26	0.308	1	0
2014-11-05	8	25	0.320	5	17	0.294	1	0
2014-11-07	13	33	0.394	6	21	0.286	1	0
2014-11-08	9	22	0.409	8	21	0.381	0	1
2014-11-10	10	27	0.370	6	26	0.231	0	1




```
print(classification_report(Y2_test, y_pred2))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	320
1	0.65	1.00	0.79	586
accuracy			0.65	906
macro avg	0.32	0.50	0.39	906
weighted avg	0.42	0.65	0.51	906


Receiver operating characteristic for 3 Point Made
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.50)

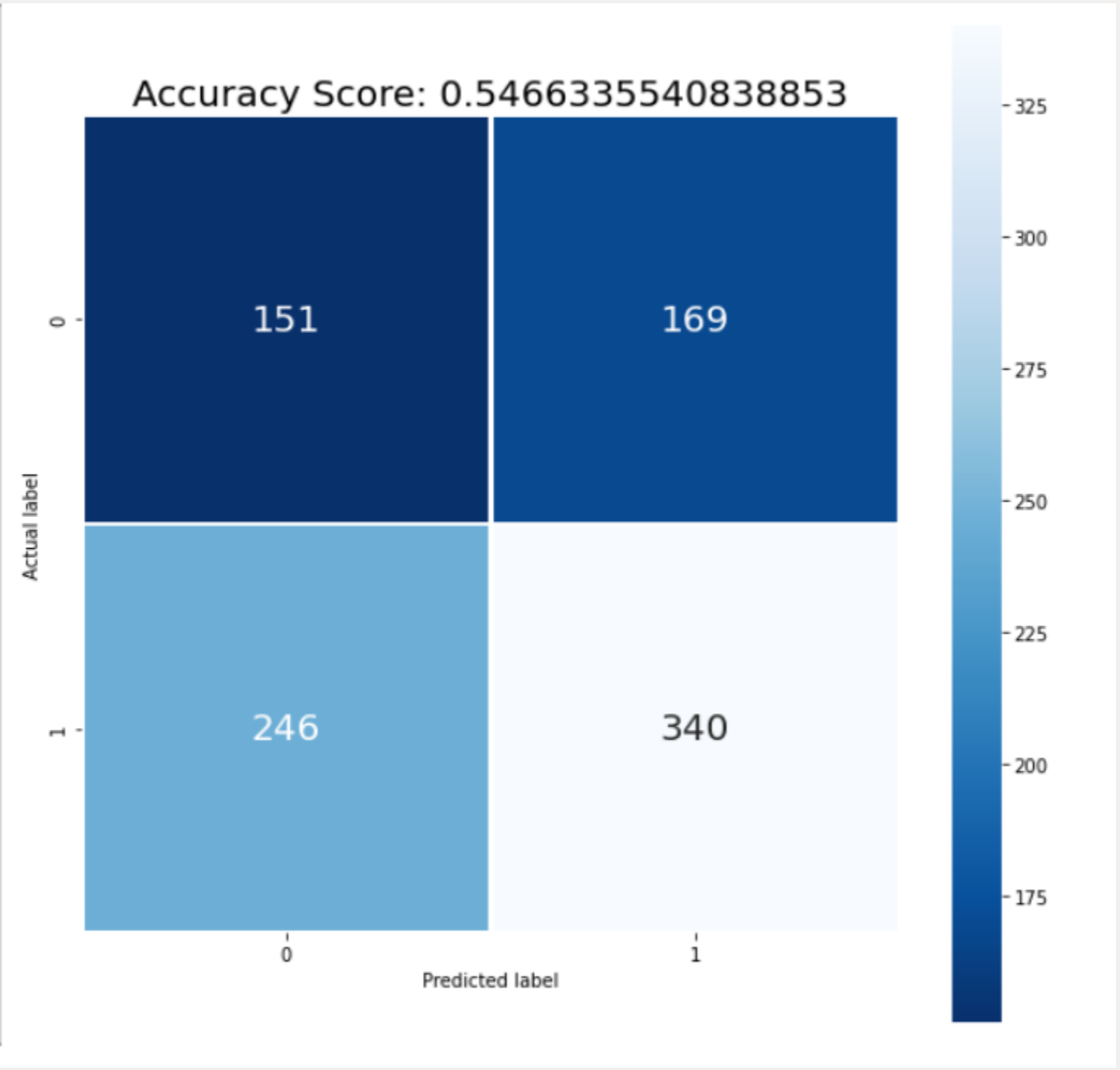
Q3 Does having higher 3PT% than your opponent = win?

- We shall use non-linear SVM for this question.

```
shootbetter = nba['X3PointShots.'] > nba['Opp.3PointShots.']
```

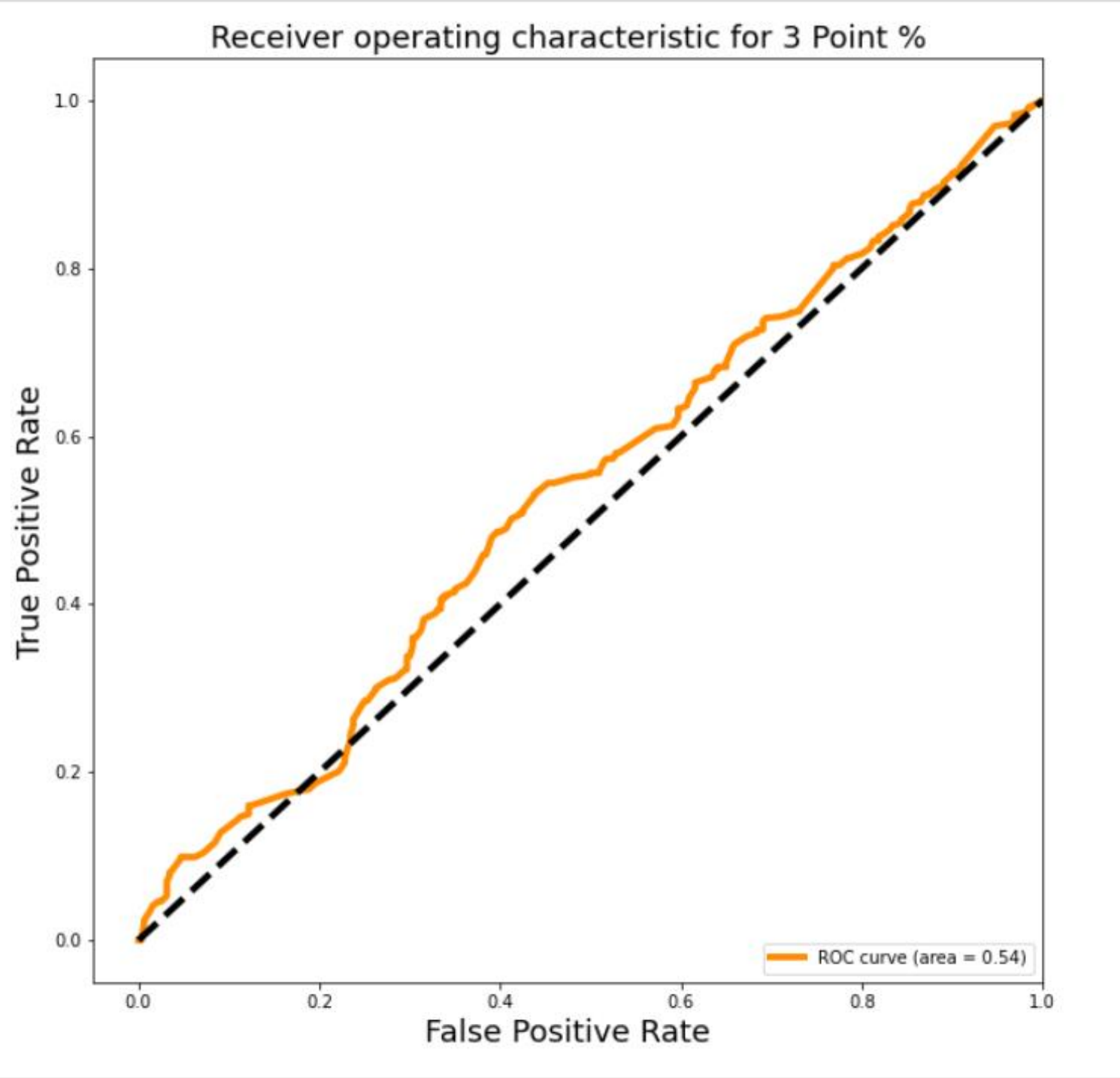
```
nba_Q3 = nba[shootbetter]

nba_Q3.head()
```

Date	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.	WlNorLOSS_L	WlNorLOSS_W
2014-10-29	13	22	0.591	8	26	0.308	1	0
2014-11-05	8	25	0.320	5	17	0.294	1	0
2014-11-07	13	33	0.394	6	21	0.286	1	0
2014-11-08	9	22	0.409	8	21	0.381	0	1
2014-11-10	10	27	0.370	6	26	0.231	0	1




```
print(classification_report(Y3_test, y_pred3))
```

	precision	recall	f1-score	support
0	0.38	0.47	0.42	320
1	0.67	0.58	0.62	586
accuracy			0.54	906
macro avg	0.52	0.53	0.52	906
weighted avg	0.57	0.54	0.55	906


Receiver operating characteristic for 3 Point %
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.54)

Q4
3PT% >= 50% = win?

- We shall use Naive Bayes to tackle this question.

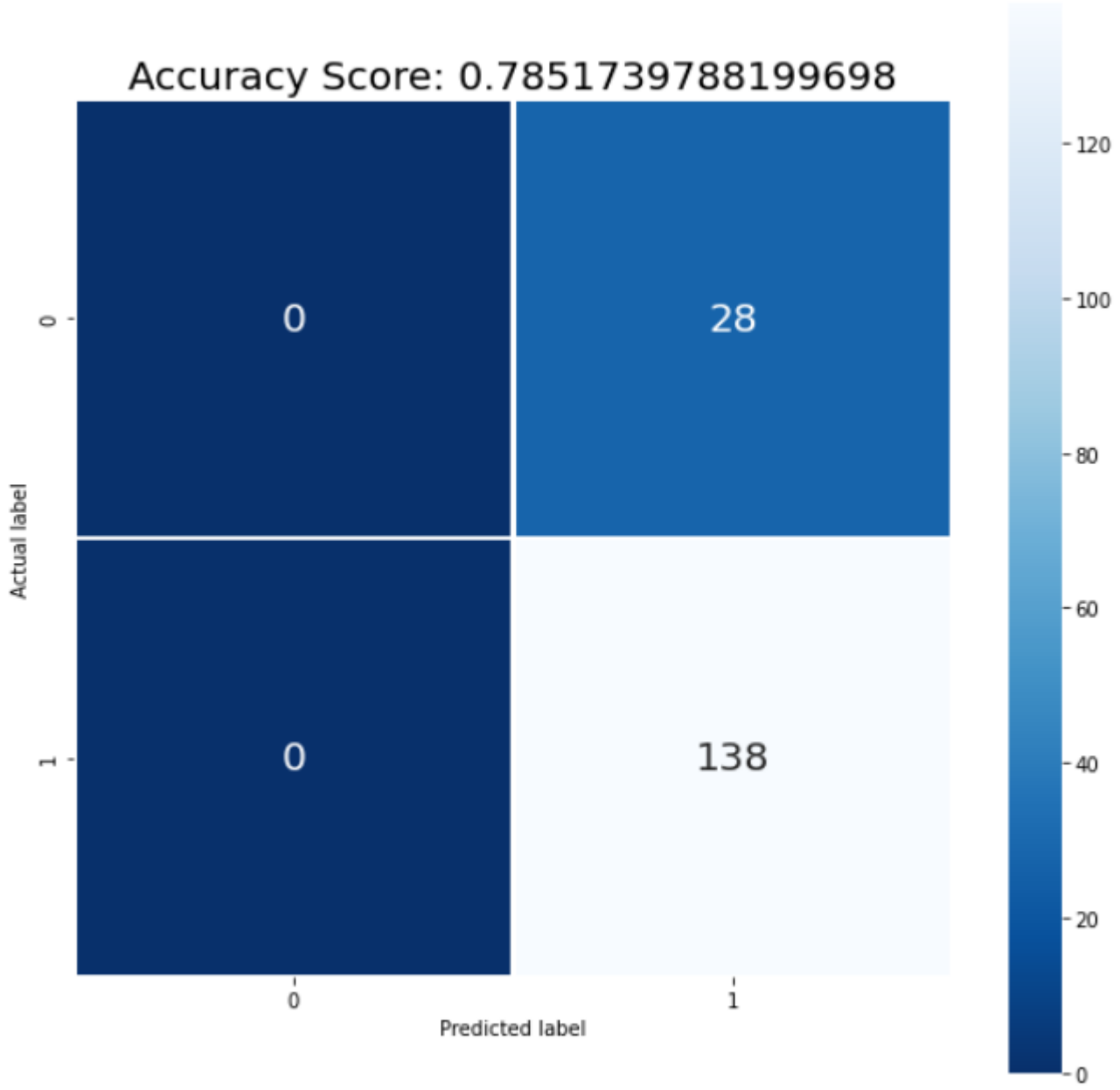
```
shoot_well = nba['X3PointShots.'] >= 0.500
```

```
nba_Q4 = nba[shoot_well]

nba_Q4.head()
```

Date	X3PointShots	X3PointShotsAttempted	X3PointShots.	Opp.3PointShots	Opp.3PointShotsAttempted	Opp.3PointShots.	WINorLOSS_L	WINorLOSS_W
2014-10-29	13	22	0.591	8	26	0.308	1	0
2014-11-29	12	23	0.522	5	23	0.217	0	1
2014-12-17	16	28	0.571	8	30	0.267	0	1
2014-12-23	12	23	0.522	11	29	0.379	0	1
2015-01-07	13	25	0.520	5	17	0.294	0	1

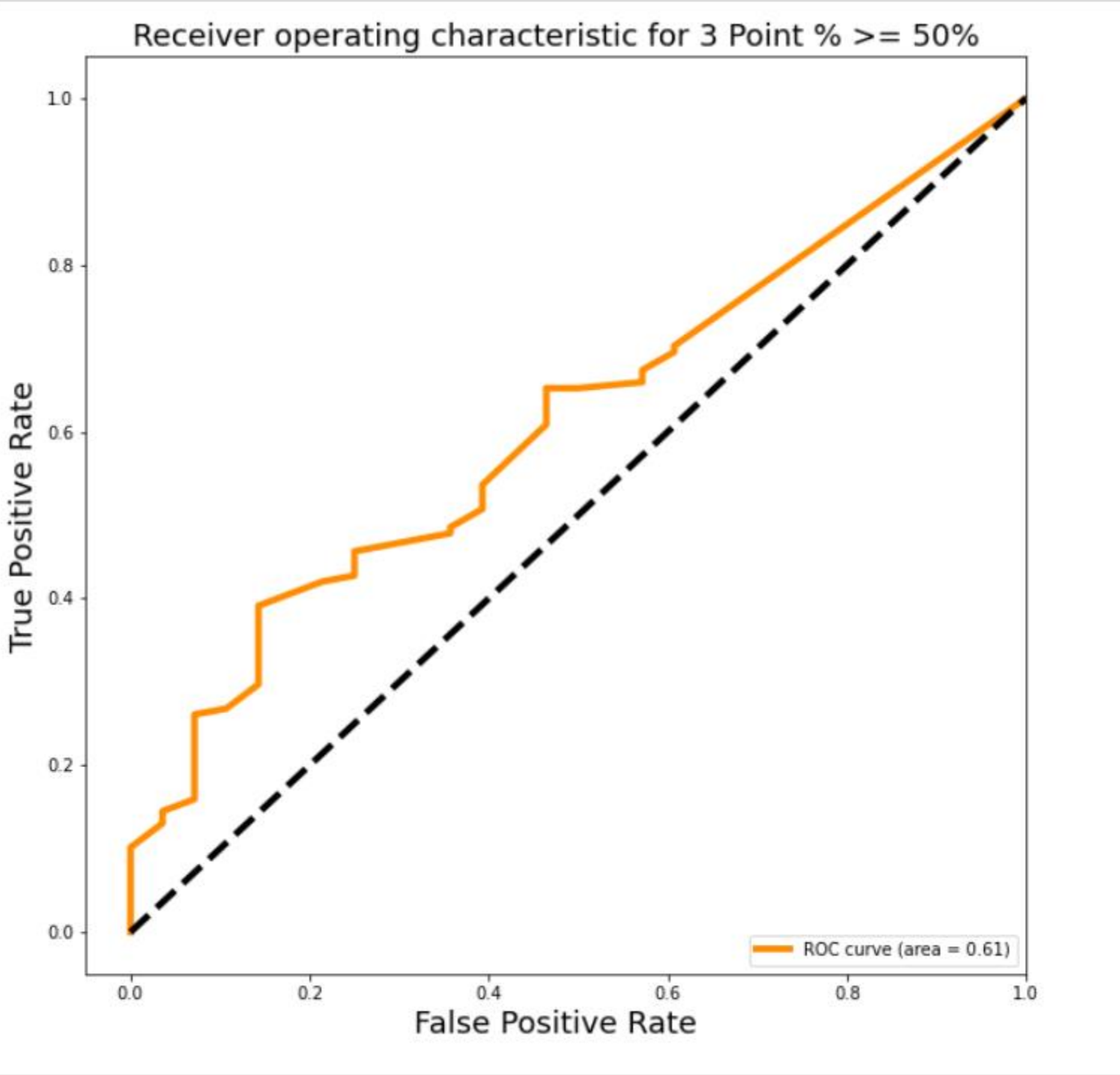
```
gnb = GaussianNB()
y_pred4 = gnb.fit(X4_train, Y4_train).predict(X4_test)
print("Number of mislabeled points out of a total %d points : %d" % (X4_test.shape[0], (Y4_test != y_pred4).sum()))
```

```
Number of mislabeled points out of a total 166 points : 28
```




```
print(classification_report(Y4_test, y_pred4))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	28
1	0.83	1.00	0.91	138
accuracy			0.83	166
macro avg	0.42	0.50	0.45	166
weighted avg	0.69	0.83	0.75	166


Receiver operating characteristic for 3 Point % >= 50%
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.61)

Q5 More assists than your opponent = win?

- I shall use the GridSearch Method

```
from sklearn.model_selection import GridSearchCV
```

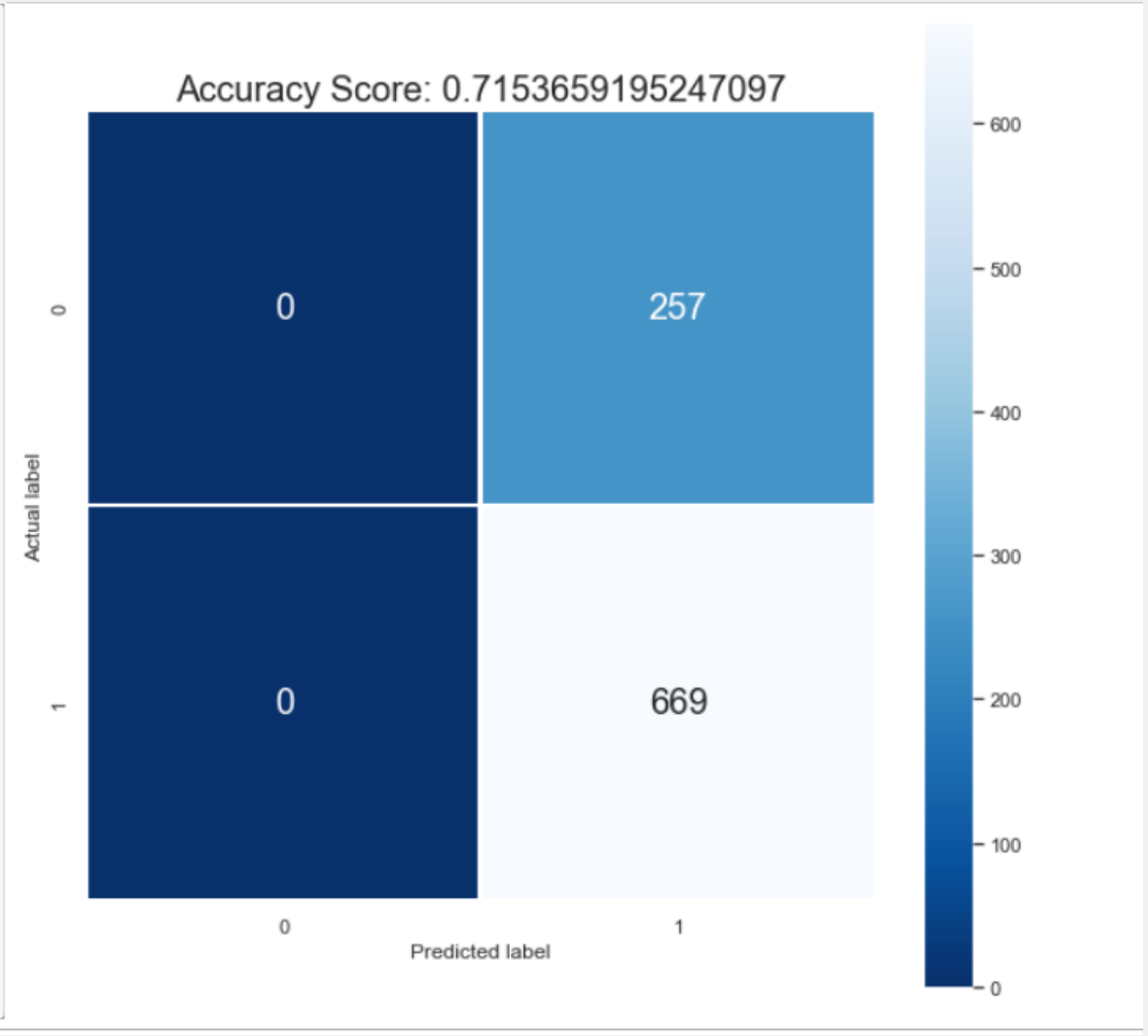
```
search_svm_nl = GridSearchCV(clf, svc_params, cv=5)
search_svm_nl #your optimised model
```

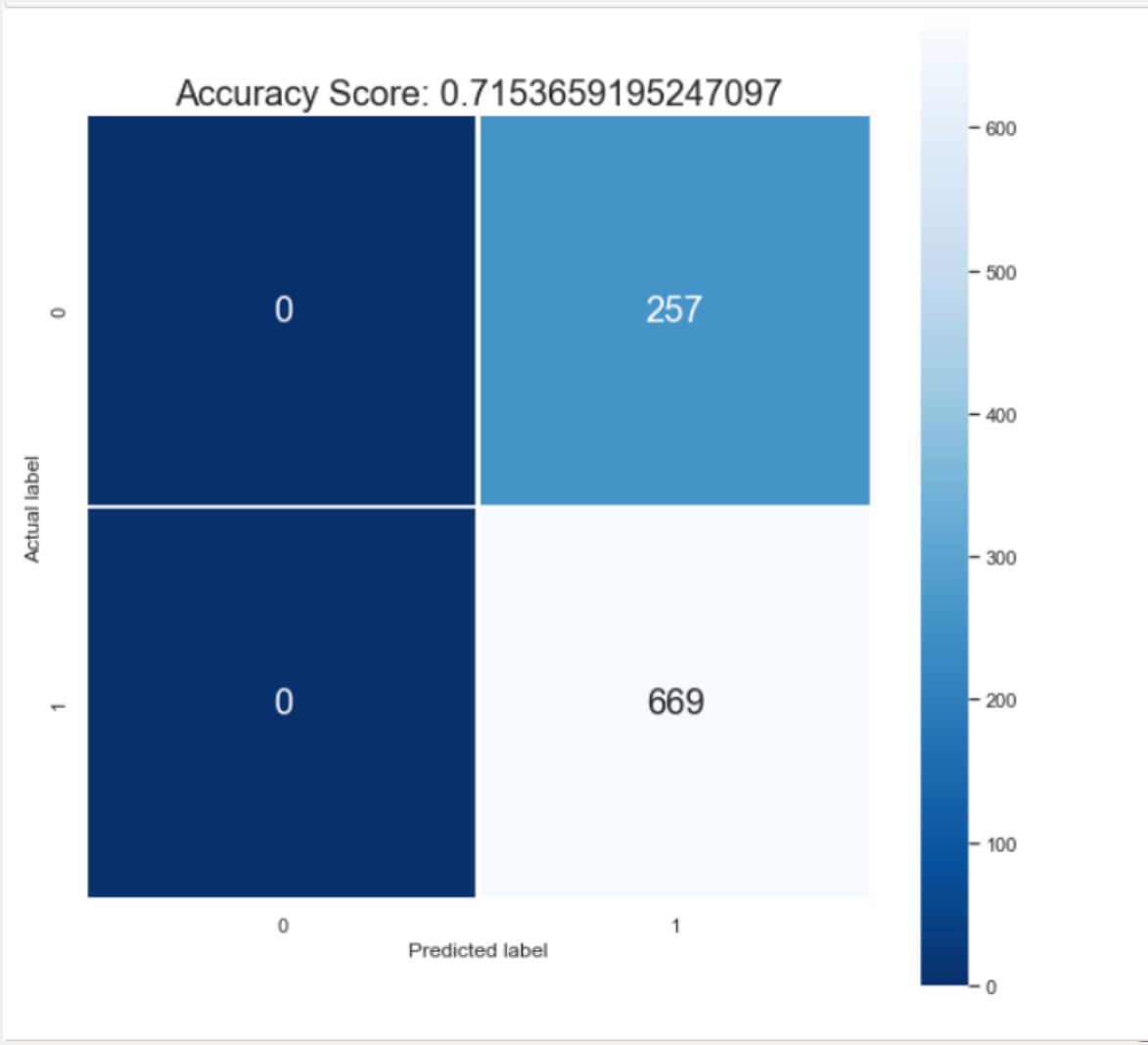
```
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1000, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='linear', max_iter=-1,
                           probability=True, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [1, 10, 100], 'gamma': [0.001, 0.0001],
                         'kernel': ['linear', 'rbf']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
search_svm_nl.fit(X5_train, Y5_train)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1000, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='linear', max_iter=-1,
                           probability=True, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [1, 10, 100], 'gamma': [0.001, 0.0001],
                         'kernel': ['linear', 'rbf']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

GridSearch(Linear) vs Naïve Bayes



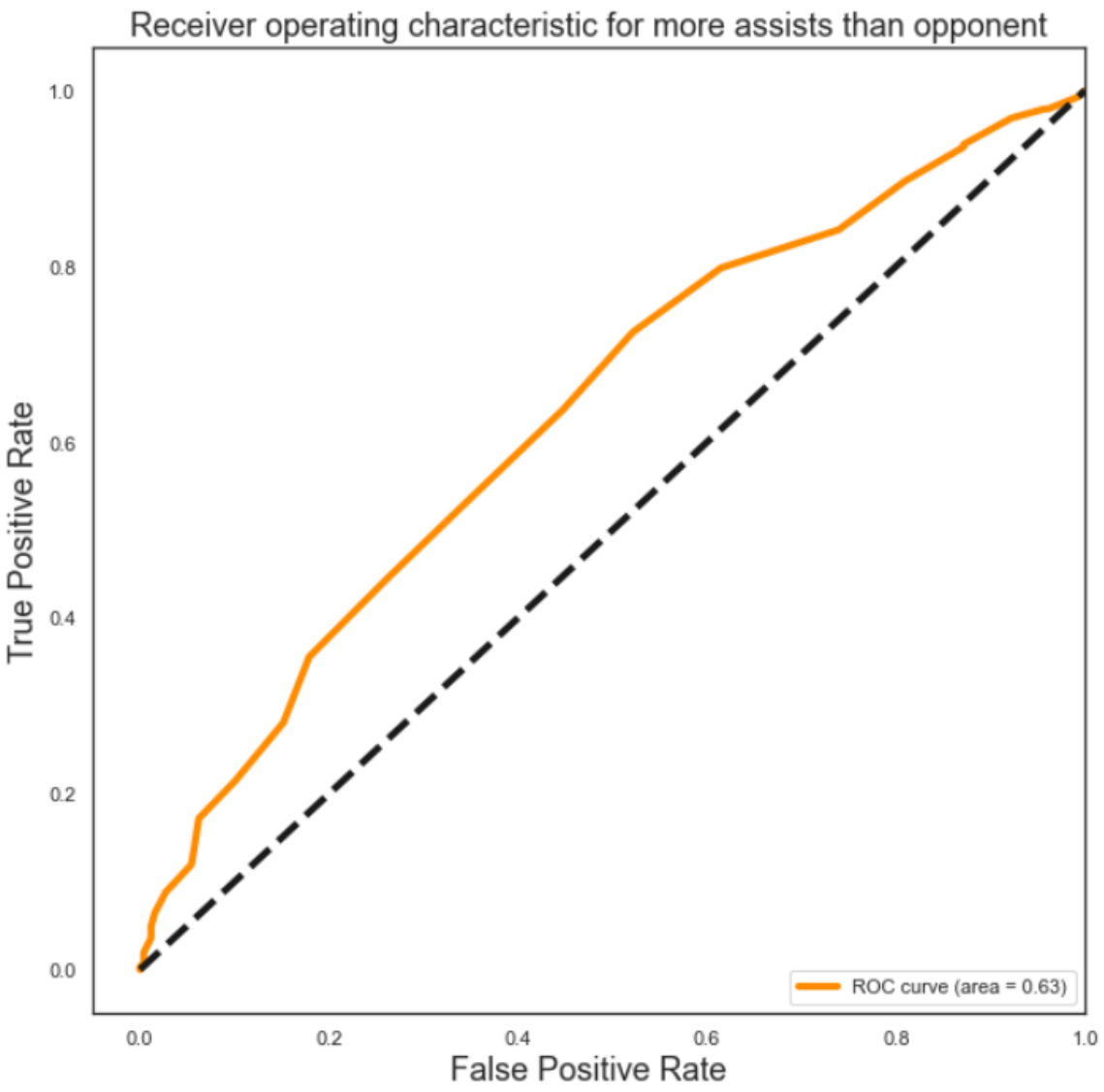


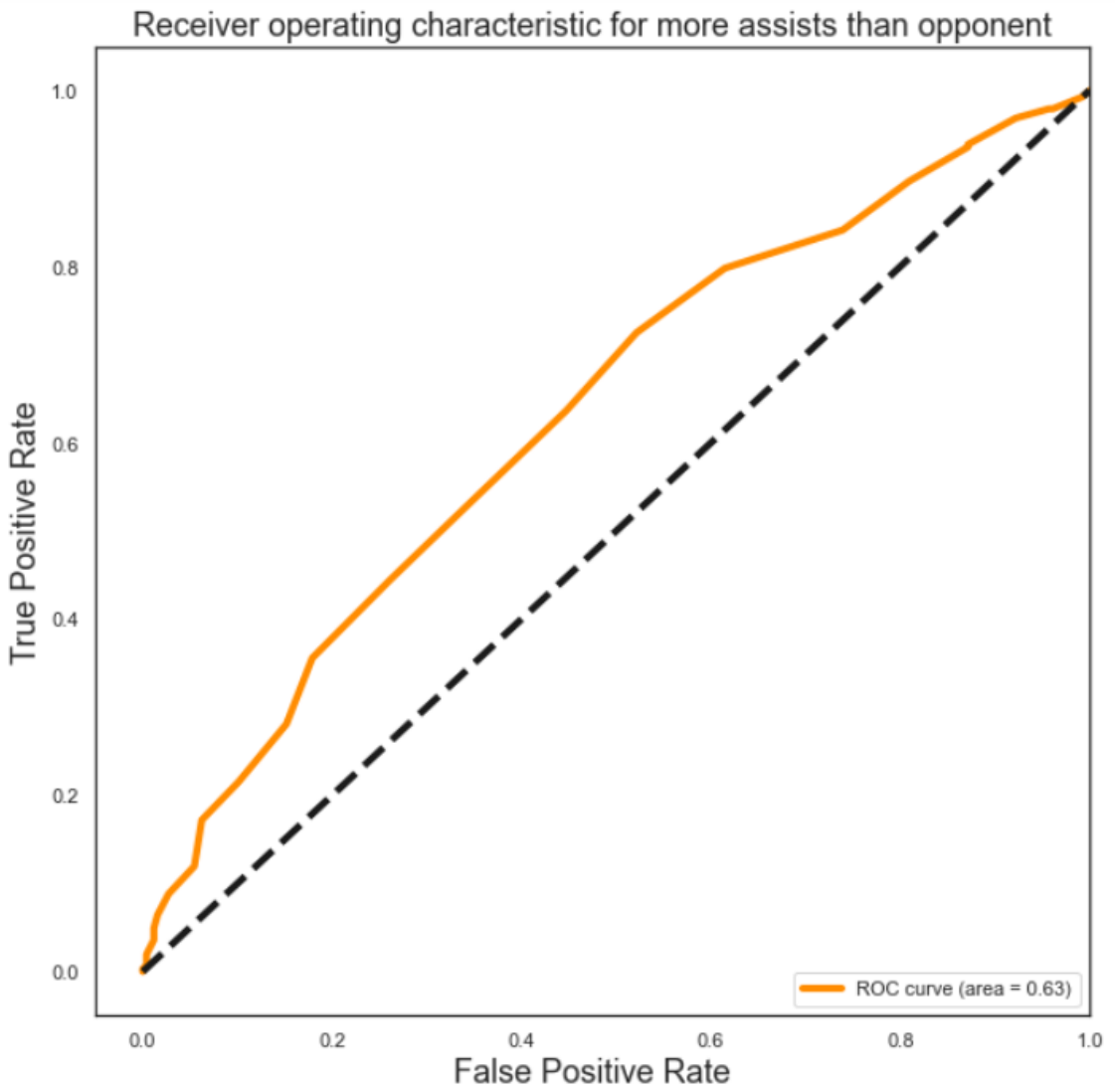
```
print(classification_report(Y5_test, y_pred5a))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	257
1	0.72	1.00	0.84	669
accuracy			0.72	926
macro avg	0.36	0.50	0.42	926
weighted avg	0.52	0.72	0.61	926

```
print(classification_report(Y5_test, y_pred5))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	257
1	0.72	1.00	0.84	669
accuracy			0.72	926
macro avg	0.36	0.50	0.42	926
weighted avg	0.52	0.72	0.61	926


Receiver operating characteristic for more assists than opponent
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.63)


Receiver operating characteristic for more assists than opponent
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.63)

Analysis

- Given that it's true when P>= 0.5,
- For Q1,2,4,5 – it is expected that the team would go on to win when the criteria is fulfilled
- Using logistic regression is probably not the preferred way for analysis

```
logreg.predict_proba(X1_test)
```

```
array([[0.47700146, 0.52299854],
       [0.47413239, 0.52586761],
       [0.46267504, 0.53732496],
       ...,
       [0.47413239, 0.52586761],
       [0.48130785, 0.51869215],
       [0.47987205, 0.52012795]])
```

Conclusion

- Naive Bayes appears to be the best method for classification
- It appears shooting more 3s does not translate to wins
- Then again ROC value is relatively low
- Lack of advanced data e.g quality of shots, individual stats
- Future Work: Look into other factors affecting the outcome
- If you want to shoot your way to Ws, “live by the 3, die by the 3”

If you wanna find out more…

- https://www.espn.com.sg/nba/story/_/id/28312678/how-deep-audacious-3-pointers-taking-nba
- https://www.nbcsports.com/bayarea/warriors/odds-rockets-missing-27-straight-3-pointers-vs-warriors-was-insane#:~:text=On%20this%20day%20two%20years,in%20the%20Western%20Conference%20finals.

References

- https://www.kaggle.com/ionaskel/nba-games-stats-from-2014-to-2018
- https://www.basketball-reference.com/leagues/NBA_stats_per_game.html
- https://cheezburger.com/8244485/damian-lillards-apathetic-victory-stare-is-getting-rightfully-memed
- https://www.theringer.com/nba/2019/2/27/18240583/3-point-boom-nba-daryl-morey
- https://www.youtube.com/watch?v=583U6pE6ovQ
- https://www.youtube.com/watch?v=uqNI4WZGSgU
- https://www.youtube.com/watch?v=bI1eM2NeVyA
- https://www.basketball-reference.com/teams/HOU/2020.html