Shot Prediction in NBA

Chinmay Sathe

19MT0119

—

NEURAL NETWORKS & DEEP LEARNING LAB
(MCC542)

—

Prof. Subhashis Chatterjee

Abstract

The National Basketball Association (NBA) is a men's professional basketball league in North America, composed of 30 teams (29 in the United States and 1 in Canada). It is one of the four major professional sports leagues in the United States and Canada, and is widely considered to be the premier men's professional basketball league in the world.

Objective here is to develop a Machine Learning algorithm that predicts if the shot will be made or not.

1 INTRODUCTION

The main reason why sports is in its own an area of entertainment and has so many channels dedicated to this one single genre, is because of the unpredictability of results.

The NBA's regular season runs from October to April, with each team playing 82 games. Its playoffs extend into June. As of 2015, NBA players are the world's best paid athletes by average annual salary per player.

Here, while doing predictions, we are being a spoilsport (pun intended). But this may also result in increasing interest of the viewers, and the unpredictability of the shots outcome will play a major role in that.

The workflow is as follows:

1. First step is Exploratory Data Analysis on various features.
2. Second step is to using EDA results to find most predictive features.
3. In the last step, we apply our Model (XGBOOST).

Model tried is:
- XGBOOST

Performance Metrics used are:
- Precision

In one line the objective can be stated as:

"Given all the basic information of a certain game, already being played, with all the information of the players (defenders, etc.) we are predicting if the shot is being made or not ".

Since we have to predict yes or no output, this is a "**Binary-Class Classification Problem**".

2 DATA AND FEATURES

2.1 Data

Data on shots taken during the 2014-2015 season, who took the shot, where on the floor was the shot taken from, who was the nearest defender, how far away was the nearest defender, time on the shot clock, and much more. We are considering the team “Charlotte Hornets” vs. any other teams matches only.

The column titles are generally self-explanatory.

Useful for evaluating who the best shooter is, who the best defender is, the hot-hand hypothesis, etc.

Dataset is scrapped from NBA’s REST API and in this project, it is taken from:

https://www.kaggle.com/dansbecker/nba-shot-logs

2.2 Features

In total there are 21 features, we shall define all the features here:

1. GAME_ID – The unique id representing game being played.

2. MATCHUP – Details of when and where match is being played.

3. LOCATION – A for away & H for home

4. W – Match outcome W for Win & L for loss

5. FINAL_MARGIN – The final margin by which Charlotte Hornets won.

6. SHOT_NUMBER – the number of shot that was made

7. PERIOD – A block of time in which the game is divided

8. GAME_CLOCK – time of the clock

9. SHOT_CLOCK - time at which shot was made

10. DRIBBLES -number of dribbles the shot took

11. TOUCH_TIME – the time from begging till end

12. SHOT_DIST – the distance from which shot was made

13. PTS_TYPE - what type of points could be earned 2 or 3

14. SHOT_RESULT – was the shot missed or made.

15. CLOSEST_DEFENDER – name of the closest defender

16. CLOSEST_DEFENDER_PLAYER_ID – ID of the closest defender

17. CLOSE_DEF_DIST - distance of the closest defender

18. FGM – count of Field Goal Made

19. PTS – Points earned

20. player_name – name of the player making the shot

21. player_id – ID of the player making the shot

3 EXPLORATORY DATA ANALYSIS

3.1 Most Predictive Feature

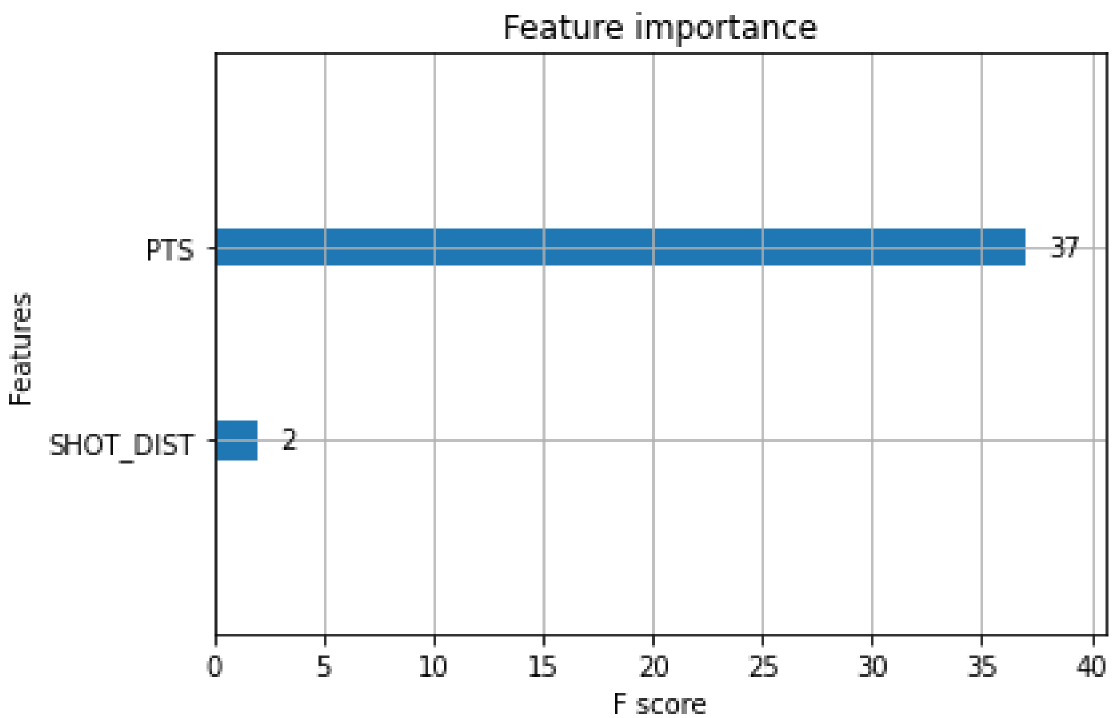


Figure 1: using XGBOOST's plot_importance function to plot the above figure

Observation: PTS looks like it is totally correlated with FGM. Thus, it was removed.

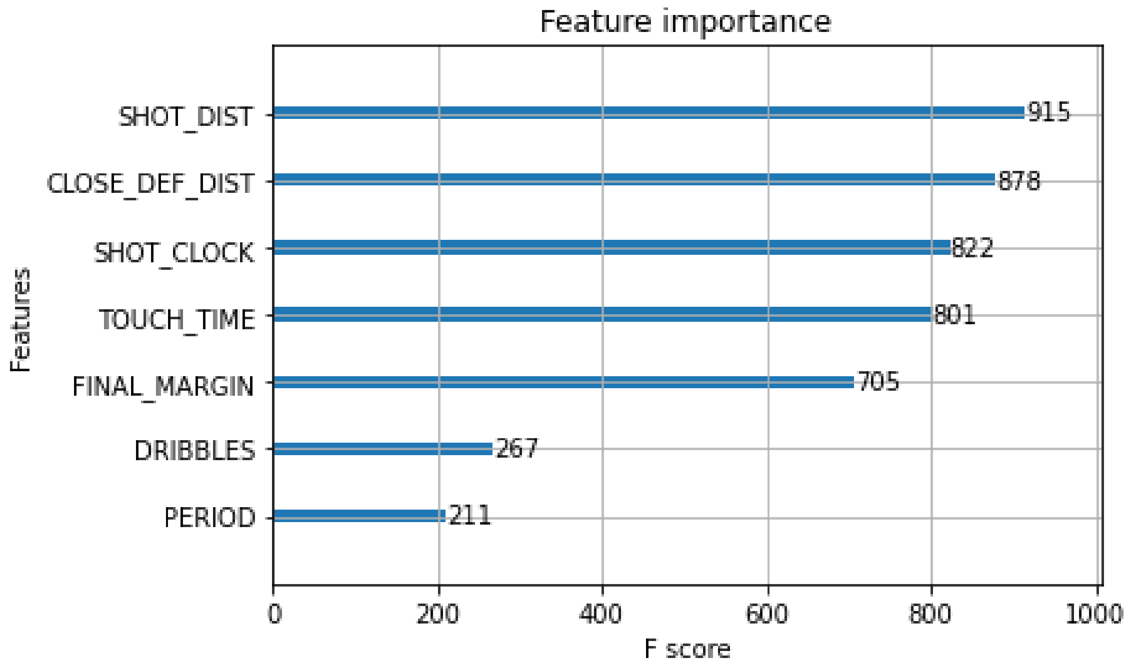


Figure 2: using XGBOOST's plot_importance function to plot the above figure

Observation: this looks better, we have the features that are correlated but not directly.

3.2 Shot Distance

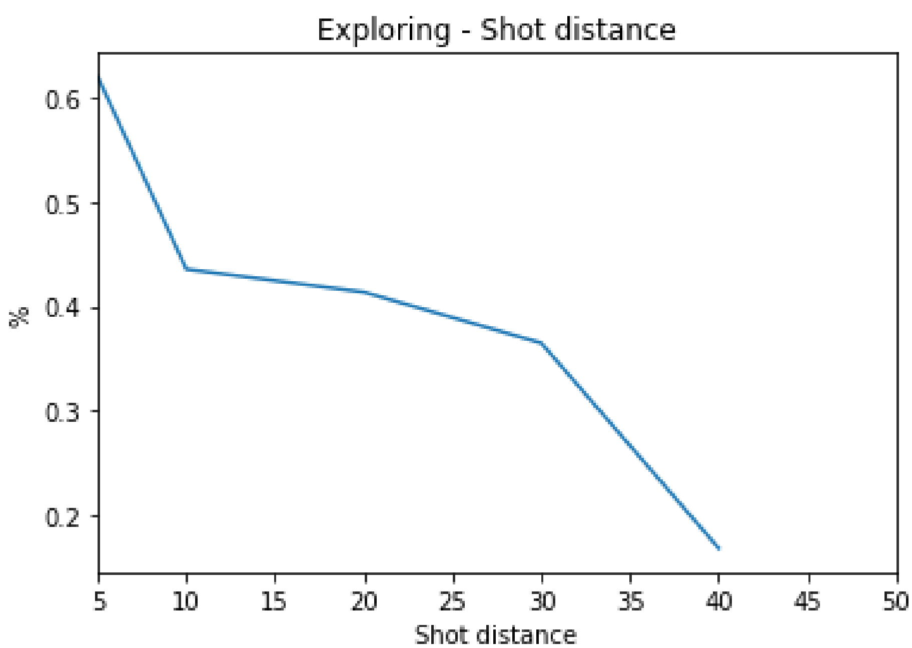


Figure 3: Relation between Shot Distance and Percentage Shots Made.

3.3 Close defence distance

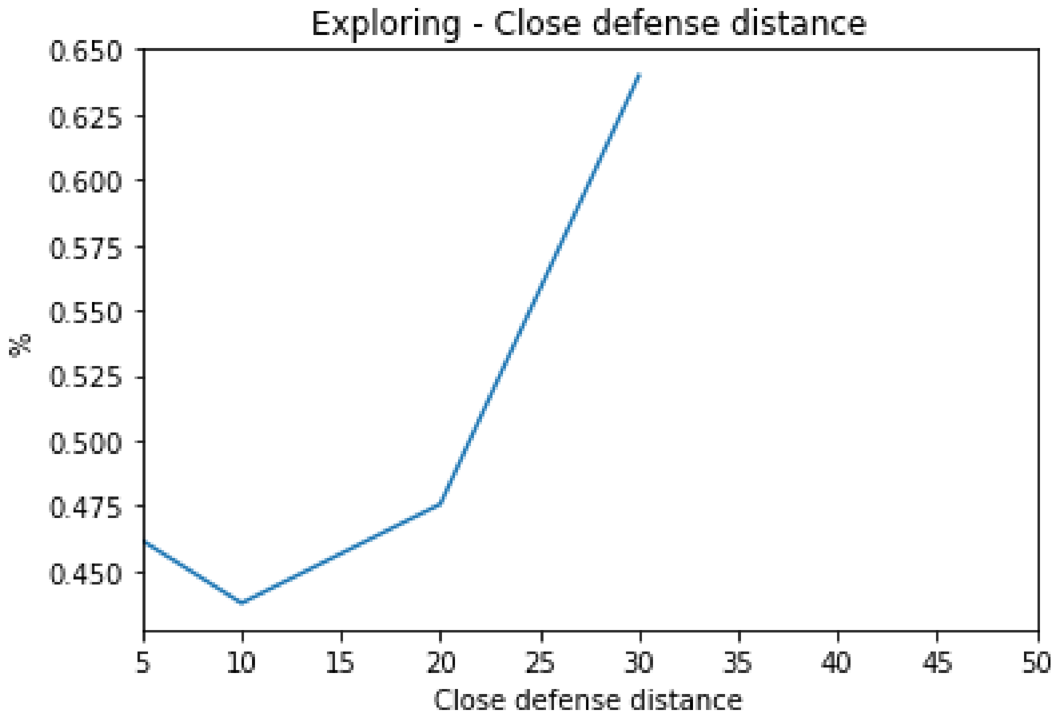


Figure 4: Relation between Close Defence Distance and Percentage Shots Made.

3.4 Working on Target Variable

```
shots made 55743
shots missed 66409
total shots 122152
we must at least have better precision than  0.5436587202829262
```

Figure 5: Understanding our target variable FGM

4 MODEL

4.1 Predicting with every feature

Applying XGBOOST algorithm on each and every feature.

```
every feature
(122152,)
[[26431  6885]
 [16962 10798]]
0.6106429904427981
```

Figure 6: Data count, Confusion matrix and precision score

4.2 Predicting with only 4 important features

```
[[26700  6616]
 [17308 10452]]
0.6123740332786501
```

Figure 7: Confusion matrix and precision score

4.3 Using Grid search to find best suitable alpha and lambda

```
best params
{'learning_rate': 1e-05, 'max_depth': 3, 'min_child_weight': 0.0001, 'n_estimators': 1}
best score
0.6786442659965666
```

Figure 8: Result of GridCV

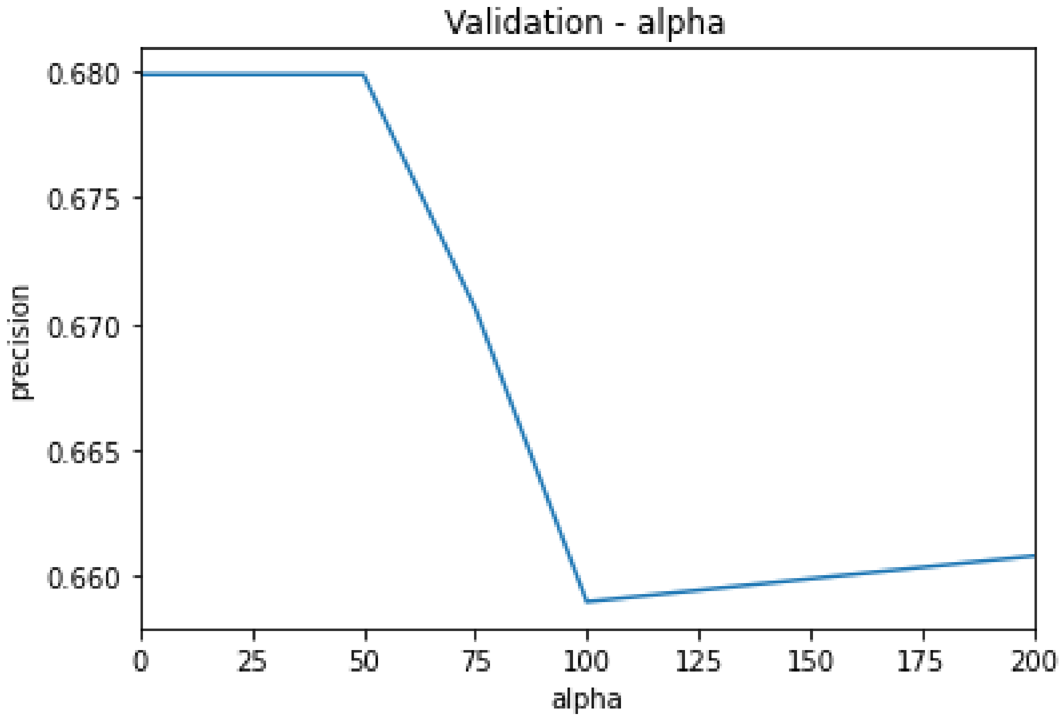


Figure 9: Relation between Alpha and precision score

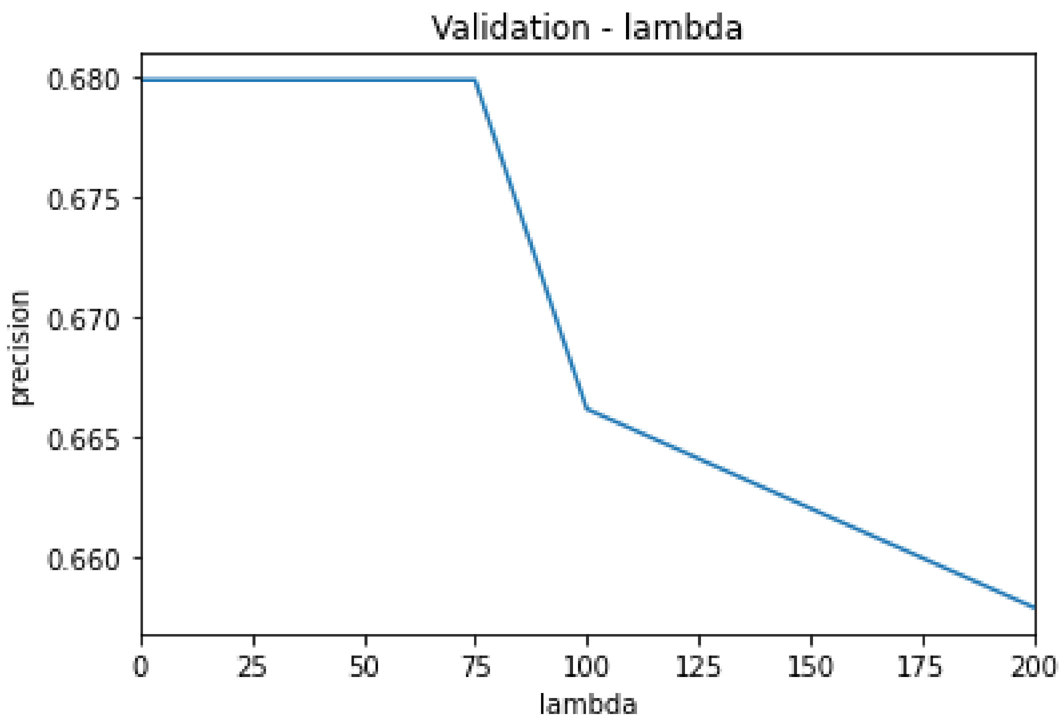


Figure 10: Relation between lambda and precision score

5 RESULTS AND CONCLUSIONS

5.1 Using Best Hyperparameters found earlier XGBOOST was applied

```
0.679740913997841
```

Figure 11: Precision Score after applying XGBOOST

5.2 Conclusions

- Using XGBOOST with every feature, precision score calculated was: **0.6106**
- Using XGBOOST with only 4 features, precision score calculated was: **0.6123**
- Finally, Using XGBOOST with best parameters, precision score calculated was: **0.6797**

Thus, we shall use XGBOOST model for future predictions. But a better model can be thought of to increase precision score by doing further feature reductions.

6 CODE

Shots Prediction

by Chinmay Sathe

Dataset from : https://www.kaggle.com/dansbecker/nba-shot-logs (https://www.kaggle.com/dansbecker/nba-shot-logs)

1. Reading Data

+ Code + Markdown

[3]:

```
import numpy as np
import pandas as pd
from xgboost import XGBClassifier
from xgboost import plot_importance
from xgboost import plot_tree
import matplotlib.pyplot as plt
from matplotlib import pyplot
from statsmodels.nonparametric.smoothers_lowess import lowess
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, mean_squared_error,precision_score
from sklearn.model_selection import KFold, train_test_split,GridSearchCV
import math

%config InlineBackend.figure_format = 'png'
%matplotlib inline
```

[4]:

```
dataset =  pd.read_csv('../input/shot_logs.csv', header=0)
```

[7]:

```
dataset.head()
```

Out[7]:

	GAME_ID	MATCHUP	LOCATION	W	FINAL_MARGIN	SHOT_NUMBER	PERIOD	GAME_CLOCK	SHOT_CLOCK	DRIBBLES	...	SHOT_DIST	PTS_TYPE
0	21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	1	1	1:09	10.8	2	...	7.7	2
1	21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	2	1	0:14	3.4	0	...	28.2	3
2	21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	3	1	0:00	NaN	3	...	10.1	2
3	21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	4	2	11:47	10.3	2	...	17.2	2
4	21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	5	2	10:34	10.9	2	...	3.7	2

5 rows × 21 columns

[9]:

```
dataset.columns
```

Out[9]:

```
Index(['GAME_ID', 'MATCHUP', 'LOCATION', 'W', 'FINAL_MARGIN', 'SHOT_NUMBER',
       'PERIOD', 'GAME_CLOCK', 'SHOT_CLOCK', 'DRIBBLES', 'TOUCH_TIME',
       'SHOT_DIST', 'PTS_TYPE', 'SHOT_RESULT', 'CLOSEST_DEFENDER',
       'CLOSEST_DEFENDER_PLAYER_ID', 'CLOSE_DEF_DIST', 'FGM', 'PTS',
       'player_name', 'player_id'],
      dtype='object')
```

2. Exploratory Data Analaysis

2.1 Basic Feature Processing

+ Code + Markdown

[10]:

```
#Removing Negative touch times
dataset=dataset[dataset['TOUCH_TIME']>=0]

#Removing Shorts Greater than 40 metre
dataset=dataset[dataset['SHOT_DIST']<40]

#Removing Nan values

nan=float('nan')
dataset=dataset[~np.isnan(dataset['SHOT_CLOCK'])]
dataset=dataset[~np.isnan(dataset['FGM'])]

#Removing Closest defender distace by 40
dataset=dataset[dataset['CLOSE_DEF_DIST']<30]
```

[12]:

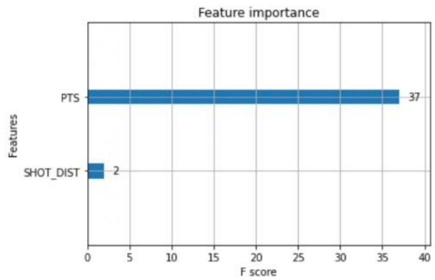
```
#Feature Dataset
datasetwithouttarget = dataset[['SHOT_DIST','TOUCH_TIME','FINAL_MARGIN','PERIOD','SHOT_CLOCK','DRIBBLES','CLOSE_DEF_DIST','

#Target Dataset
datasettarget = dataset['FGM']
```

2.2 Most Predictive Features

+ Code + Markdown

[14]:

```
model = XGBClassifier()
model.fit(datasetwithouttarget,datasettarget)
# plot feature importance
plot_importance(model)
pyplot.show()
```

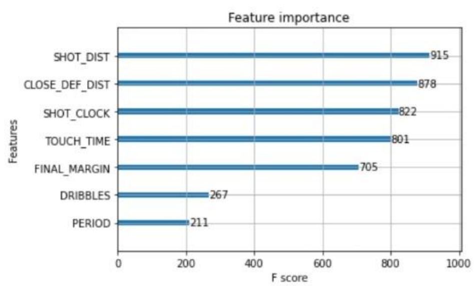


[15]:

```
datasetwithouttarget = dataset[['SHOT_DIST','TOUCH_TIME','FINAL_MARGIN','PERIOD','SHOT_CLOCK','DRIBBLES','CLOSE_DEF_DIST']]
```

[16]:

```
model = XGBClassifier()
model.fit(datasetwithouttarget,datasettarget)
# plot feature importance
plot_importance(model, importance_type ='weight')
pyplot.show()
```



2.3 Most Important Features

+ Code
+ Markdown

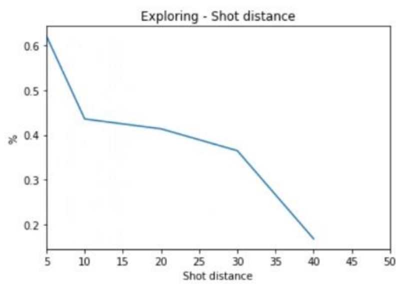
2.3.1 Shot distance

[17]:

```
distances = [0,5,10,20,30,40,50]

shot_made = [(dataset[np.logical_and(np.logical_and(dataset['SHOT_DIST']>distances[i-1],dataset['SHOT_DIST']<distances[i] )

lambda_results = pd.Series(shot_made, index =  distances[1:len(distances)])
lambda_results.plot(title = "Exploring - Shot distance")
plt.xlabel("Shot distance")
plt.ylabel("%")
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: RuntimeWarning: invalid value encountered in long_scalars
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[17]: Text(0, 0.5, '%')

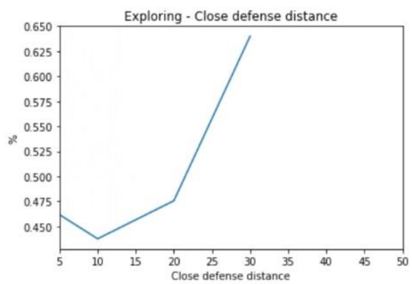


2.3.2 Close defense distance

```
[18]:
distances = [0,5,10,20,30,40,50]

shot_made = [(dataset[np.logical_and(np.logical_and(dataset['CLOSE_DEF_DIST']>distances[i-1],dataset['CLOSE_DEF_DIST']<dist

lambda_results = pd.Series(shot_made, index = distances[1:len(distances)])
lambda_results.plot(title = "Exploring - Close defense distance")
plt.xlabel("Close defense distance")
plt.ylabel("%")
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: RuntimeWarning: invalid value encountered in long_scalars
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Out[18]: Text(0, 0.5, '%')
```



2.4 Working on Target Variable

```
[19]:
print ('shots made',np.count_nonzero(datasettarget))
print ('shots missed',datasettarget.size-np.count_nonzero(datasettarget))
print ('total shots',datasettarget.size)
print ('we must at least have better precision than ',(datasettarget.size-np.count_nonzero(datasettarget))/datasettarget.si
```

```
shots made 55743
shots missed 66409
total shots 122152
we must at least have better precision than  0.5436587202829262
```

3 Model

3.1 Predict with every feature

+ Code + Markdown

[20]:
```python
print("every feature")
print(datasettarget.shape)


X_train, X_test, y_train, y_test = train_test_split( datasetwithouttarget[['SHOT_DIST','TOUCH_TIME','FINAL_MARGIN','PERIOD'

xgb_model = XGBClassifier().fit(X_train,y_train)

predictions = xgb_model.predict(X_test)

actuals = y_test

print(confusion_matrix(actuals, predictions))
print(precision_score(actuals, predictions) )
```

```
every feature
(122152,)
[[26431  6885]
 [16962 10798]]
0.6106429904427981
```

3.2 Predict with the 4 important features

+ Code + Markdown

[24]:
```python
print("4 most important features")

X_train, X_test, y_train, y_test = train_test_split( datasetwithouttarget[['SHOT_DIST','TOUCH_TIME','CLOSE_DEF_DIST','SHOT_
y = np.array(datasettarget)
X = datasetwithouttarget[['SHOT_DIST','TOUCH_TIME','CLOSE_DEF_DIST','SHOT_CLOCK']].as_matrix()


xgb_model = XGBClassifier().fit(X_train,y_train)

predictions = xgb_model.predict(X_test)

actuals = y_test

print(confusion_matrix(actuals, predictions))
print(precision_score(actuals, predictions) )
```

```
4 most important features


/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:5: FutureWarning: Method .as_matrix will be removed in a futu
re version. Use .values instead.
  """


[[26700  6616]
 [17308 10452]]
0.6123740332786501
```

3.3 Finding better configuration for XGB

[25]:
```python
X_train, X_test, y_train, y_test = train_test_split( datasetwithouttarget[['SHOT_DIST','TOUCH_TIME','CLOSE_DEF_DIST','SHOT_
X_validation, X_test, y_validation, y_test = train_test_split( X_test, y_test, test_size=0.50, random_state=42)
```

3.3.1 Using GridCV

[26]:
```python
parameters_for_testing = {
    'min_child_weight':[0.0001,0.001,0.01],
    'learning_rate':[0.00001,0.0001,0.001],
    'n_estimators':[1,3,5,10],
    'max_depth':[3,4]
}

xgb_model = XGBClassifier()

gsearch1 = GridSearchCV(estimator = xgb_model, param_grid = parameters_for_testing, scoring='precision')
gsearch1.fit(X_train[['SHOT_DIST','TOUCH_TIME','CLOSE_DEF_DIST','SHOT_CLOCK']],y_train)

print('best params')
print (gsearch1.best_params_)
print('best score')
print (gsearch1.best_score_)
```

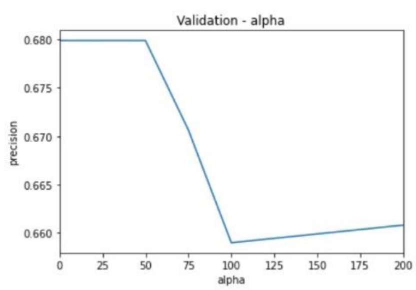
```
best params
{'learning_rate': 1e-05, 'max_depth': 3, 'min_child_weight': 0.0001, 'n_estimators': 1}
best score
0.6786442659965666
```

3.3.2 Analyze alpha

+ Code + Markdown

[27]:
```python
def ExecuteWithAlpha(x_validation,y_validation, alpha):
    xgb_model = XGBClassifier(reg_alpha=alpha,min_child_weight=0.0001,learning_rate=1e-05,
                              n_estimators=1,max_depth=3).fit(X_train,y_train)
    predictions = xgb_model.predict(x_validation)
    actuals = y_validation
    precision=precision_score(actuals, predictions)
    return precision

alphas = [0, 1, 5, 10, 15, 30, 50, 75,100,200]
cv_xgb = [ExecuteWithAlpha(X_validation,y_validation,alpha)
            for alpha in alphas]

alpha_results = pd.Series(cv_xgb, index = alphas)
alpha_results.plot(title = "Validation - alpha")
plt.xlabel("alpha")
plt.ylabel("precision")
```

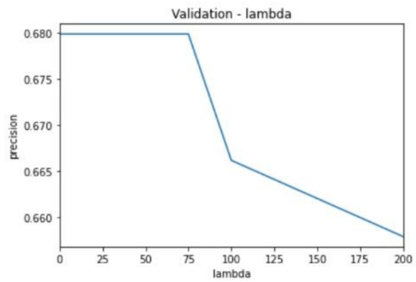
Out[27]: Text(0, 0.5, 'precision')



3.3.2 Analyze lambda

[28]:
```
def ExecuteWithLambda(x_validation,y_validation, lamb):
    xgb_model = XGBClassifier(reg_lambda=lamb,min_child_weight=0.0001,learning_rate=1e-05,
                              n_estimators=1,max_depth=3).fit(X_train,y_train)
    predictions = xgb_model.predict(x_validation)
    actuals = y_validation
    precision=precision_score(actuals, predictions)
    return precision

lambs = [0, 1, 5, 10, 15, 30, 50, 75,100,200]
cv_xgb = [ExecuteWithLambda(X_validation,y_validation,lamb)
            for lamb in lambs]

lambda_results = pd.Series(cv_xgb, index = lambs)
lambda_results.plot(title = "Validation - lambda")
plt.xlabel("lambda")
plt.ylabel("precision")
```

Out[28]: Text(0, 0.5, 'precision')



3.4 Using Best Parameters

[29]:
```
xgb_model = XGBClassifier(min_child_weight=0.0001,learning_rate=1e-05,
                              n_estimators=1,max_depth=3).fit(X_train,y_train)
predictions = xgb_model.predict(X_test)
actuals = y_test
precision=precision_score(actuals, predictions)
print(precision)
```

0.679740913997841

7 REFERENCES

https://en.wikipedia.org/wiki/National_Basketball_Association

https://www.cbssports.com/nba/news/10-biggest-nba-shots-of-the-decade-ray-allen-or-kyrie-irving-at-no-1-lebron-james-damian-lillard-show-up-twice/

https://fansided.com/2017/04/17/30-best-shots-nba-playoffs-history/