UNIVERSITY OF WATERLOO

MSCI 446

Unscouted Potential

Looking for value in the pool of undrafted NCAA players

William Li	(20519497)
Arun Issac	(20526302)
Alexander Jakel	(20524376)
Thabishion Yogananda	(20532748)

Submission Date: December 4th, 2017

Table of contents

Abstract

The purpose of this study is to detail the construction of a model that can evaluate thousands of undrafted NCAA players and evaluate which of these players would be able to provide above replacement level value for NBA teams from which NBA teams can further explore. A model that can identify talent in players left undrafted was created and it can provide massive value to teams and scouts.

Data was retrieved from basketball-reference.com and cleaned manually. The data was placed into four models and the results were compared. Out of the KNN, Decision Tree, Naive Bayes and the Logistical Regression models the one deemed to provide the most desirable results was the Naive Bayes model. It was able to identify the highest number of drafted players and also a significant number of undrafted players who ended up playing in the NBA one day.

Unsupervised learning was done on the data set by clustering the player data and clusters were found and further insight was provided through the analysis of these clusters.

Introduction

Every year, the NBA draft acts as the major pipeline for college players to turn professional by entering the NBA. A total of 60 players are selected; 30 being drafted during the first and second round respectively with their player rights being given to the team that made the selection. However, with there being several hundred players entering their name into the NBA draft, there still remains a massive pool of undrafted players available for any team to sign as a rookie free agent. In the past, NBA teams would be satisfied with just their draft picks because a team had a limited number of roster spots available. They also had a limited amount of scouting resources which ultimately were used on the top prospects. Lastly would not be able to find enough playing time to develop draft picks and rookie signings.

The hypothesis is that there exists more than 60 players every year that can potentially provide above replacement[1] value for an NBA team. Using a statistical model using real-world player data, we propose a model that can be created to help identify these players of value.

The recent affiliation of the G-League (the official NBA development league) with the NBA gives us the opportunity to test this hypothesis. The G-League acts as a minor league for NBA teams with each major nba team having a development team. For example, the Toronto Raptors (NBA team) corresponding G-League development team is the Raptors 905. Essentially, an NBA teams corresponding G-League team doubles the available roster spots an NBA team has. Teams now have the ability to stash prospects in their development system (G-League team) and give these players time to develop. The NBA draft provides an average of two prospects to each team which is not enough to satisfy the new demand since teams can take on more prospects now. It is in a team's best interest to sign as many low end prospects as possible because the cost versus potential upside favors teams in addition to failed players being easily cut from the team.

Currently, teams fill out their G-League roster spots by holding tryouts where unsigned players are tested and if they perform well enough, can make the G-League team. This method of finding players does not maximize the value of these draft spots because teams using their limited scouting resources are inviting certain players to show up at tryouts, when in reality there could be several dozen players that would be worth an evaluation.

A model was created to provide data on which players can provide value above replacement level from which teams can then specifically target if they are still available after the draft to see if it is worthwhile to introduce said players into their system.

[1] Replacement Level is a concept in sports analytics that refers to the level of production a team can get from a player that would cost nothing but the league minimum salary to acquire.

To be clear, the model does not provide projections on players so it should not be used as a tool to decide who to draft. Draft picks are extremely valuable and scouts should be tasked on making such large decisions. Human evaluation of talent by scouts will always be better than a statistical model at projecting players and every player that may be drafted has been so heavily scouted that a statistical model cannot provide further insight.

However, the model will look at the lower end players who are not scouted and act as a supplementary tool used to bring attention to worthwhile players. The beauty of the model and machine learning algorithms in general is the massive amounts of data it can quickly analyze and make predictions on. Teams do not have the resources to scout every college player in detail so this model will act as a tool to help scouts identify which players outside of the top 60 projected draftees are worth taking a look.

Related Works

Models projecting the value of prospects using college data exist online and provide a very detailed analysis of the players projected to be drafted. FiveThirtyEight has a projection tool for every player in the NBA and rookies[1][5]. However these projections do not provide much insight because these players have been heavily scouted and teams will always take the advice of professional scouts over a statistical model.

The model differs because it is meant as an aid for scouts to use and identify which players to focus on. It aims to find players overlooked by other teams. It does not compare players or rank them.

Outside of basketball, there are many tools used in the MLB to project minor league players. This is similar to the model in the sense that it projects player performance in the MLB based on statistics from a different league. Fangraphs has its KATOH projections which project a minor league player's total wins above replacement[2] for his first 6 years in the majors [3].

Fangraphs also uses mahalanobis distances to project prospects [4]. It looks for the closest major league comparison for a prospect and projects the prospect's performance based on his major league comparison.

Baseball does some pretty amazing things with statistics but it is unlikely the same results can be replicated with basketball because of the nature of the sport. Basketball is a much more free flowing sport and statistics are not able to fully capture the game like it can in baseball.

---

[2] Wins above replacement (WAR) is a method of quantifying a player's total contributions in one statistics. Like its name it refers to the number of wins a player produced over a replacement level player. Quantifying a player's contributions into wins is the holy grail of sports analytics and the NBA does not currently have a widely accepted version of this statistic.

Data

Data was scraped from sports-reference.com. The top 1000 players in terms of minutes played were selected. Players from the 2011, 2012, 2013, 2014, 2015, 2016 seasons were chosen, resulting in a total of 5000 rows of player season data. Data included in the analysis consists of 32 columns of on court and off court data which deemed to be important in judging the potential of college players. See the appendix for a list of the data used.

Non-numeric data such as position and class had to be changed into numerical values. Data from sports-reference.com have positions recorded as “G”,”F” and “C” for guard, forward and center respectively. The position data was then encoded to ‘1’,’2’ and ‘3’ respectively. A similar process had to be done with class data which which was changed from “FR,” “SO,” “JR,” and “SR” (Freshman, Sophomore, Junior, and Senior) to ‘1’,’2’,’3’ and ‘4’ respectively.

Certain percentage data on basketball-reference.com are not decimal numbers. For example, rate statistics such as rebounding percentage and assist percentage are recorded as “36%” and not “0.36”. Such data had to be transformed to the correct numeric value.

Two desired columns of data were missing from the data that was obtained from basketball-reference.com. Firstly the class variable for whether a player was drafted was needed. This data was obtained by cross referencing a the player’s names with a list of players drafted for a certain year. Each year of player data had to be individually cross referenced because many drafted players spend multiple years in college and it doesn’t make sense to mark a certain season as drafted if the player wasn’t actually drafted that year.

There is a very large variance in conference strength in college basketball as good players tend to go to the best teams and conferences. The conference of each player season had to be converted to a numerical value which specifies how strong the quality of play was for that season.

Data Statistics

Position	Count
Guard	1968
Forward	1080
Center	134

Figure 1: Distribution by position

There are much fewer centers than other positions because teams typically have one center, two guards and two forwards on the court. In addition, there are very few humans in the world tall enough to play the center position for this reason there is usually lower turnover for the position.

The number of players increases as the class increases because older players tend to be better and therefore get more playing time and more chances to make a team.

Class	Count
Freshmen	481
Sophmore	1026
Junior	1613
Senior	1805
Graduate	13

Figure 2: Distribution of players by Class

In the data 3.6% of all players end up being drafted to an NBA team. The average strength of schedule of the players included in the data is 0.358, showing that the data set is slightly skewed towards players in higher end conferences. Players from these conferences are more likely to be drafted anyways so it makes sense to skew the data is this direction.

There have been 4,952,315 minutes played overall across all players in the data. This puts into perspective the sheer magnitude of the data analyzed by the model and how it would be impossible for a team to get scouts to do the analysis by purely watching games.

Results

The purpose of this project was to predict whether or not a player in the NCAA had the abilities and necessary skills to provide value in the NBA. As such, the class variable was whether or not the NCAA player was drafted or not. The binary class variable, Drafted, was used to determine this classification.

Supervised Learning

The supervised learning portion of the project required the usage of classification machine learning algorithms such as: KNN, Logistic Regression, Decision Trees, and Naive Bayes. Through running the data through these algorithms a dashboard of statistics were determined, these statistics were then compared to find the ideal algorithm which produced the most accurate results.

False Positive vs. False Negative

All models are inherently incorrect; through this understanding, the model must be evaluated even when it produces false positive and negative results. When the algorithm provides a false positive result it is essentially finding athletes who the algorithm classifies as being drafted even though they were not. False negative represents the athletes which the algorithm classified as not making the draft, even though they were drafted. As this tool is one which is meant to help scouts in their job to find potential candidates to draft, having a higher false positive rate is more beneficial and less costly than a higher false negative. This is due to the fact that when provided athletes who would not be drafted a scout can easily discern this and discount the athlete.

However, is a possible candidate is not identified, it may result in loss of profits for the NBA and loss of commission to the scout. Therefore to reduce cost algorithms with higher false positive results are more economical than algorithms with higher false negative results.

K-Folds Validation

K-Folds validation was done on each supervised learning method with the k value set to 10. Due to the nature of the class variable, k-folds validation yielded a high accuracy for every one of the methods. The reason for this is because the class variable is overwhelming skewed towards one value and any model will classify the large majority of players to be undrafted leading to a high accuracy. Accuracy is not a good statistic to use to judge the goodness of each model making K- Folds validation not a good way to evaluate the model. An accuracy of 100% will actually not be ideal because that will not provide insight to scouts on who to target after the draft is over.

KNN

K-Nearest Neighbour is a non-parametric means of classifying data which uses distance to determine the value of the output. KNN is one of the most simple algorithms which can be used for classification. The dataset was passed through a KNN algorithm and the following dashboard and confusion matrix was discovered.

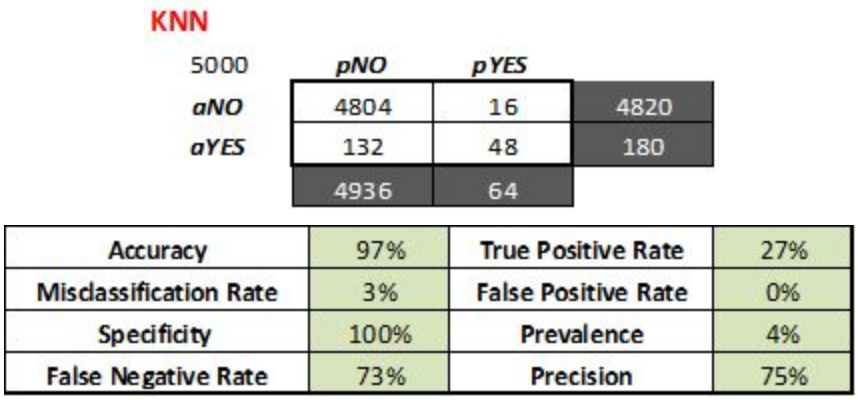


Figure 3: Confusion matrix and Dashboard for KNN

Solely observing the accuracy of the KNN algorithm provides a very misleading representation of the real accuracy of the classifier. Due to the fact that there are many more “NO” classifications than the “YES” classification the accuracy is skewed. However, looking at the true positive rates, false positive rates, and misclassification rate give a more well rounded and accurate representation of the model. Comparing all the values, KNN is very clearly seen to be a less than desirable model as it has a low true positive rate and a low false positive rate with a very high false negative rate.

It only predicts a total of 64 drafted players over the five year period when in reality 180 were drafted. This means that the model only selected the top end players to be drafted and scouts already know who the best players are without the help with a model. Such a low pYES number is unlikely to be able to provide any insight to scouts on who to sign after the draft which is the purpose of the model.

Decision Trees

Decision trees build classification models which uses a tree-like graph. At the nodes of the tree it maps decisions while giving it's possible consequences. The objective of a decision tree is to develop rules and decisions which take data and parse it to determine a classification. The dataset of NCAA players were passed through a decision tree model and the following dashboard and confusion matrix was determined:

DT

5000	*pNO*	*pYES*	
aNO	4815	5	4820
aYES	22	158	180
	4837	163	

Accuracy	99%	True Positive Rate	88%
Misclassification Rate	1%	False Positive Rate	0%
Specificity	100%	Prevalence	4%
False Negative Rate	12%	Precision	97%

Figure 4: Confusion matrix and Dashboard for Decision Trees

At first glance the decision tree model is almost entirely accurate. However, through further inspection is was clear that the decision tree was severely overfitting the model to the data. In fact, the extent of the overfitting was so severe that the decision tree created contained specific rules for every single instance of the positive instance in the binary class variable (See Appendix A for the decision tree visualization). Although pruning and other methods exist to reduce overfitting it was found through iterations of testing that the level of overfitting in decision trees were too high to be useful in providing useful conclusions. As such the decision tree model was disregarded as a means to classify NCAA players accurately.

Logistic Regression

Logistic regression is a categorical regression model; it is a means of defining the likelihood that a the class variable will be either a one or zero. Using a logit function the logistic regression model issues weights to explanatory variables and maps this to the logit function. Using a linear discriminator the logistic regression model classifies an input as either a one or a zero. Passing the NCAA data through the logistic regression model the following dashboard, and confusion matrix was determined:

LR

5000	pNO	pYES	
aNO	4799	21	4820
aYES	127	53	180
	4926	74	

Accuracy	97%	True Positive Rate	29%
Misclassification Rate	3%	False Positive Rate	0%
Specificity	100%	Prevalence	4%
False Negative Rate	71%	Precision	72%

Figure 5: Confusion matrix and Dashboard for Logistic Regression

Although better than the previous two models the logistic regression model has a very high false negative rate. This means that players who were drafted would not have been predicted to be drafted. This oversight of the model does not provide insight to scouts. As mentioned previously a successful model is one which somewhat accurately predicts the athletes who will get drafted and also more who should've been drafted. The model should have more false positives than negatives if the case arises to reduce the possibility of overlooking potential NBA talent . As this model does not minimize false negatives, it is not fit to be used as the model of choice to determine future NBA draft picks.

Naive Bayes

The naive bayes model is a classification model which uses likelihood estimates of the dataset to determine the classification of an input. As the data provided in the dataset is primarily numeric, and continuous data points the naive bayes algorithm creates normal distributions on each column when learning. Applying these distributions to the input, the algorithm determines how to classify the input. The NCAA dataset was passed through the naive bayes model and the following dashboard and confusion matrix was determined:

NB

5000	pNO	pYES	
aNO	4270	550	4820
aYES	43	137	180
	4313	687	

Accuracy	88%	True Positive Rate	76%
Misclassification Rate	12%	False Positive Rate	11%
Specificity	89%	Prevalence	4%
False Negative Rate	24%	Precision	20%

Figure 6: Confusion matrix and Dashboard for Naive Bayes

Barring the decision tree model the naive bayes model has the highest number of correctly predicted athletes who get drafted. The false positive value of the model is higher than the false

negative; this creates value in terms of accounting for value which would have been lost with a higher false negative rate. The high true positive rate also depicts the usefulness of this model compared to the previous three. Aggregating all the positives of this model, it can be determined that this model is one which most accurately and economically predicts the athletes who have the potential to get drafted for the NBA.

Naive bayes yields a larger “YES” value than the actual. This great because it is able to find useful NBA players outside of the draft. An analysis of the results of Naive bayes found that out of the 687 predicted yes values 41 were undrafted players who eventually made it to the NBA. Naive bayes was able to identify useful undrafted players such as Robert Covington, Tim Frazier, Matthew Dellavedova, and Kent Bazemore. This ability makes it a valuable tool. Robert Covington is projected to produce over 27 million dollars of value this year[2] yet will only be paid a bit more that 1.5 million [6]. The fact that the naive bayes model is able to find such a player makes it extremely valuable.

The reason naive bayes model produces so many more false positives than other models is because it utilizes the naive assumption that features in the model are normally distributed. However, at the highest competitive level of NCAA the differences between drafted players and good players are miniscule.. Consider the fact that the difference in FG% between a great and average player would be around 5%, the marginal difference between a good and great player is even smaller. Since the naive bayes model is unable to make such small distinctions it ends up making predictions liberally.

Compare and Contrast

As stated previously a higher number of false positives is desirable as it is the data that will result in finding useful undrafted players from the NCAA. Looking at the four models detailed above it is clear the only one with a desirable number of false positives is the naive bayes model. The other models are essentially useless because it cannot provide that same insight. The decision tree model overfit the data to a level where it was not translatable to other models.

To further understand the effect of the models on data it has not seen, testing was done on a set of test data the models had never seen (see Appendix D). From this test it is clear that the naive bayes is superior other models created. It predicts the highest number correctly, while providing a higher false positive through which the scouts can comb through and find draftable players.

Clustering

Clustering was done on the entirety of the dataset with FG, TRB% and AST% on the axises. These three data items were chosen because scoring, rebounds and assists are the three most looks at statistics in the NBA and in college basketball. It is evident by the popularity of the triple

double[3] as a talking point in the media. Clustering resulted in 3 clusters with different skillsets (Refer to Appendix B).

Looking at the clusters in Appendix B; the teal cluster represents players who score a high number of field goals but do not contribute heavily in terms of rebounds and assists. The purple cluster contains specialists who do not score much but contribute by having a high number of rebounds and/ or assists. Finally there is a yellow cluster which consists of unskilled players who do not contribute much in any aspect. In the NBA nearly every player is able to contribute in some way but in college there is a much more diluted talent pool and there will be a large number of players who are not skilled enough to contribute.

Clustering was also done with the same axises on only the drafted players and it provided an interesting insight. There ended up being three clusters with distinctly different skillsets. The teal cluster contained high scoring players with high rebound rates. These are likely very tall centres or power forwards who at the college level could score with ease. The variance of height is much higher in college basketball and large players have much more of an advantage than they do in the NBA. Due to their physical gifts, they are very strong at scoring and rebounding. The next cluster is the purple cluster which contains low scoring players with high rebound rates or high assist rates but not both. These players are specialists who are either great at rebounding or playmaking and they are drafted by nba teams who have a need for that one skillset. Lastly the yellow cluster consists of pure scorers. These players make a lot of field goals but do not rebound or assist much. Scoring is inherently valuable in basketball and teams will always value scoring. These are also likely to be average or shorter players evident by the low rebound rates. They rely more heavily on skill to score unlike their teal cluster counterparts who rely more on their physical stature to score.

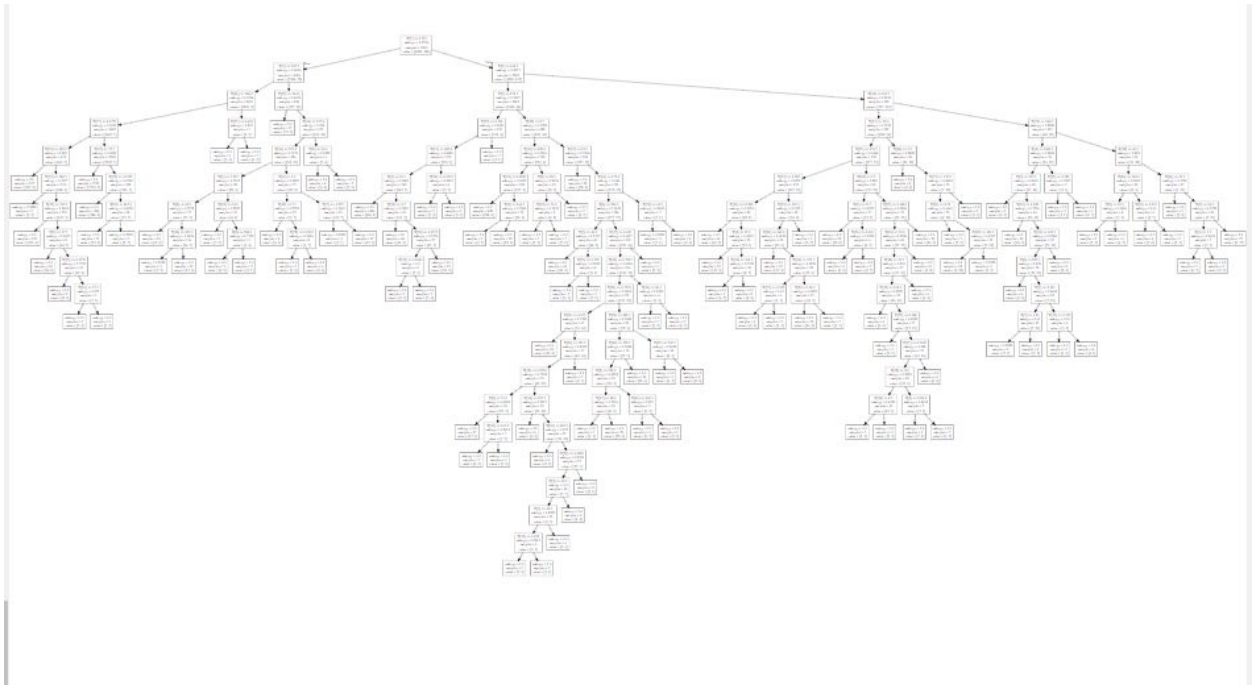
Conclusions

Through comparing and contrasting different methods of classification through supervised learning, it is abundantly clear that naive bayes provides the best results for this instance of analysis. By introducing NCAA data to the model, it has successfully determined players who were drafted as well as identified players who were not drafted but eventually got into the NBA and had successful. The model was designed to help NBA scouts and NBA teams identify lower end player available after the draft to look into and its ability to identify cheap talent will help NBA teams greatly.

[3] The triple double is a feat where a player's stat line reaches double digits in points, rebounds and assists in a single game.

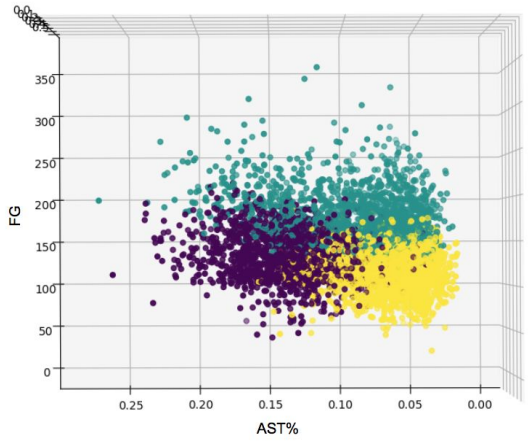
Appendices

Appendix A: Decision Tree Visualization

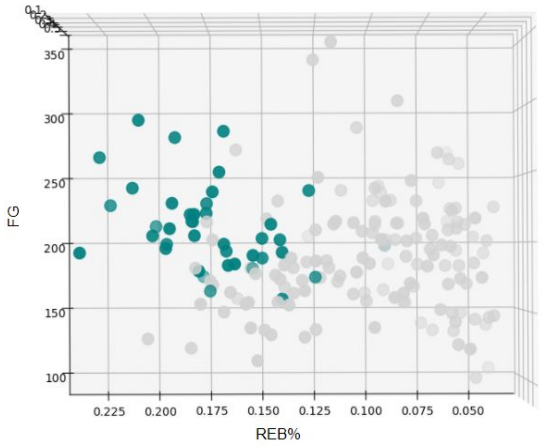


Appendix B: Clusters Graphs

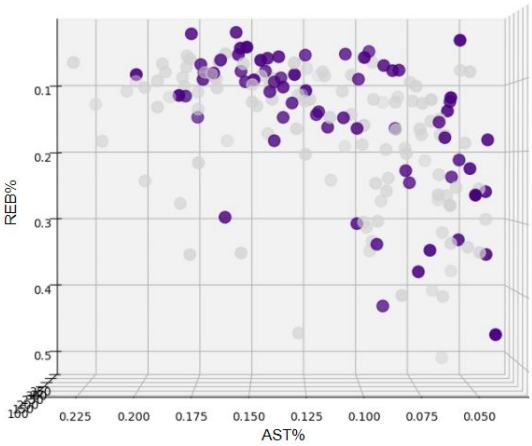
B-1: All Data



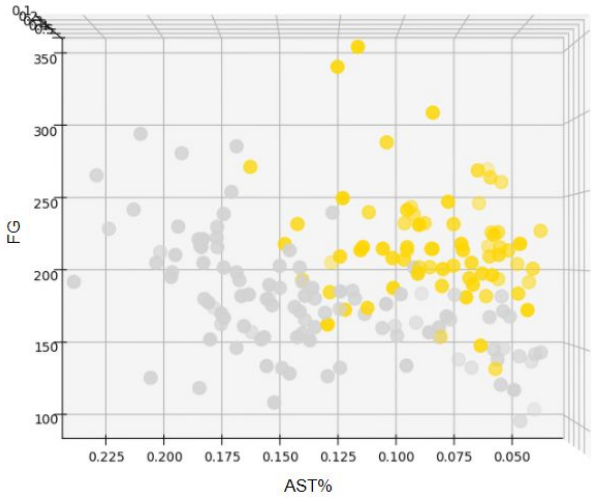
B-2: Cluster representing Massive Centers



B-3:Cluster representing Specialists (Role Players)



B-4: Cluster representing Pure Scorers



Appendix C: Screenshots of Data

Player	SOS	FG	FT	FTA	ORB
Doug McDermott	3.48	307	133	167	84
Reggie Hamilton	-1.94	281	262	299	26
Thomas Robinson	7.49	261	163	239	113
Kevin Jones	6.81	260	103	132	141
C.J. McCollum	-4.75	254	198	244	50
Shane Gibson	-4.99	253	112	130	25
Nate Wolters	-1.94	250	188	240	31
Antwan Carter	-4.13	245	88	123	78
Deshaun Thomas	8.18	243	83	110	100
Michael Glover	-2.85	239	126	194	106
Kevin Murphy	-4.05	239	124	172	30
Rakim Sanders	-2.85	235	101	159	102
Wendell McKines	-0.4	233	136	179	118
Damian Lillard	-3.69	231	228	257	17
Rian Pearson	-0.91	231	119	175	106
Will Barton	2.48	230	134	179	70
Harrison Barnes	5.76	229	141	195	74
John Shurna	8.18	228	110	146	48
Jared Sullinger	8.18	228	175	228	116

Player	Drafted	Class	Pos	SOS	G
Doug McDermott	0	SO	F	3.48	35
Reggie Hamilton	0	SR	G	-1.94	36
Thomas Robinson	1	JR	F	7.49	39
Kevin Jones	0	SR	F	6.81	33
C.J. McCollum	0	JR	G	-4.75	35
Shane Gibson	0	JR	G	-4.99	32
Nate Wolters	0	JR	G	-1.94	34
Antwan Carter	0	SR	C	-4.13	31
Deshaun Thomas	0	SO	F	8.18	39
Michael Glover	0	SR	F	-2.85	33
Kevin Murphy	1	SR	G	-4.05	33
Rakim Sanders	0	SR	F	-2.85	37
Wendell McKines	0	SR	F	-0.4	35
Damian Lillard	1	JR	G	-3.69	32
Rian Pearson	0	SO	G	-0.91	36
Will Barton	1	SO	G	2.48	35
Harrison Barnes	1	SO	F	5.76	38
John Shurna	0	SR	F	8.18	33
Jared Sullinger	1	SO	F	8.18	37

Appendix D: Model Statistics with Test Data

D-1: KNN confusion matrix and dashboard

KNN

1000	*pNO*	*pYES*	
aNO	937	21	958
aYES	38	4	42
	975	25	

Accuracy	94%	True Positive Rate	10%
Misclassification Rate	6%	False Positive Rate	2%
Specificity	98%	Prevalence	4%
False Negative Rate	90%	Precision	16%

D-2: Decision Tree confusion matrix and dashboard

DT

1000	*pNO*	*pYES*	
aNO	930	28	958
aYES	30	12	42
	960	40	

Accuracy	94%	True Positive Rate	29%
Misclassification Rate	6%	False Positive Rate	3%
Specificity	97%	Prevalence	4%
False Negative Rate	71%	Precision	30%

D-3: Naive Bayes confusion matrix and dashboard

NB

1000	*pNO*	*pYES*	
aNO	833	125	958
aYES	16	26	42
	849	151	

Accuracy	86%	True Positive Rate	62%
Misclassification Rate	14%	False Positive Rate	13%
Specificity	87%	Prevalence	4%
False Negative Rate	38%	Precision	17%

D-4: Logistic Regression confusion matrix and dashboard

LR

1000	*pNO*	*pYES*	
aNO	941	17	958
aYES	24	18	42
	965	35	

Accuracy	96%	True Positive Rate	43%
Misclassification Rate	4%	False Positive Rate	2%
Specificity	98%	Prevalence	4%
False Negative Rate	57%	Precision	51%

Appendix E: Source Code

E1 - KNN

```
import csv
import pandas as pd
import numpy as np
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import KFold, cross_val_score
import pickle

dataset = pd.read_csv('Workbook5Clean.csv')
print(dataset)

# prepare datasets to be fed in the regression model
#predict attend class given extra hours and grade
CV =  dataset.Drafted.reshape((len(dataset.Drafted), 1))
data = (dataset.ix[:,'Class':'USGp'].values).reshape((len(dataset.Drafted), 32))

# Create a KNN object
KNN = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
KNN.fit(data, CV)

#predict the class for each data point
predicted = KNN.predict(data)
print("Predictions: \n", np.array([predicted]).T)

df = pd.DataFrame()
df = np.array([predicted]).T

with open('KNNPrediction.csv', 'w') as fp:
   writer = (csv.writer(fp)).writerows(df)

# predict the probability/likelihood of the prediction
print("Probability of prediction: \n",KNN.predict_proba(data))

print("Neighbors and their Distance: \n",KNN.kneighbors(data, return_distance=True))

print("Accuracy score for the model: \n", KNN.score(data,CV))

#metrics.confusion_matrix(CV, predicted, labels=["Yes","No"]))

# Calculating 5 fold cross validation results
model = KNeighborsClassifier()
kf = KFold(len(CV), n_folds=10)
scores = cross_val_score(model, data, CV, cv=kf)
```

```
print("Accuracy of every fold in 10 fold cross validation: ", abs(scores))
print("Mean of the 5 fold cross-validation: %0.2f" % abs(scores.mean()))
```

E2 - Decision Tree

```
import csv
import pandas as pd
import numpy as np
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn import preprocessing
from sklearn.cross_validation import KFold, cross_val_score

# prepare datasets to be fed in the regression model
dataset = pd.read_csv('Workbook5Clean.csv')

DS_Scorers = pd.read_csv('Group_DataScorers.csv')
DS_Specialists = pd.read_csv('Group_DataSpecialists.csv')
DS_Centers = pd.read_csv('Group_DataCenters.csv')
#predict attend class given extra hours and grade
CV =  dataset.Drafted.reshape((len(dataset.Drafted), 1))
data = (dataset.ix[:,'Class':'USGp'].values).reshape((len(dataset.Drafted), 32))

#print(dataset)
#print(data)
#print(CV)

# Create linear regression object
DT = DecisionTreeClassifier(criterion="entropy", min_samples_leaf = 2)

# Train the model using the training sets
DT.fit(data, CV)

tree.export_graphviz(DT, out_file='tree.dot')

#predict the class for each data point
predicted = DT.predict(data)

print("Predictions: \n", np.array([predicted]).T)

df = pd.DataFrame()
df = np.array([predicted]).T

with open('DTPredictions_TEST.csv', 'w') as fp:
   writer = (csv.writer(fp)).writerows(df)

# predict the probability/likelihood of the prediction
print("Probability of prediction: \n",DT.predict_proba(data))

print("Feature importance: ", DT.feature_importances_)
```

```
print("Accuracy score for the model: \n", DT.score(data,CV))

# Calculating 5 fold cross validation results
model = DecisionTreeClassifier()
kf = KFold(len(CV), n_folds=10)
scores = cross_val_score(model, data, CV, cv=kf)
print("Accuracy of every fold in 10 fold cross validation: ", abs(scores))
print("Mean of the 10 fold cross-validation: %0.2f" % abs(scores.mean()))
```

E3 - Naive Bayes

```
import csv
import pandas as pd
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
import numpy as np
from sklearn.cross_validation import KFold, cross_val_score
from sklearn.naive_bayes import MultinomialNB

dataset = pd.read_csv('Workbook5Clean.csv')
print(dataset)

# prepare datasets to be fed into the naive bayes model
#predict attend class given extra hours and grade
CV =  dataset.Drafted.reshape((len(dataset.Drafted), 1))
data = (dataset.ix[:,'Class':'USGp'].values).reshape((len(dataset.Drafted), 32))

# Create model object
NB = GaussianNB()

# Train the model using the training sets
NB.fit(data, CV)

#Model
print("Probability of the classes: ", NB.class_prior_)
print("Mean of each feature per class:\n", NB.theta_)
print("Variance of each feature per class:\n", NB.sigma_)

#predict the class for each data point
predicted = NB.predict(data)
print("Predictions:\n",np.array([predicted]).T)

df = pd.DataFrame()
df = np.array([predicted]).T

with open('NBPredictions.csv', 'w') as fp:
   writer = (csv.writer(fp)).writerows(df)

# predict the probability/likelihood of the prediction
prob_of_pred = NB.predict_proba(data)
```

```
print("Probability of each class for the prediction: \n",prob_of_pred)

print("Accuracy of the model: ",NB.score(data,CV))

#print("The confusion matrix:\n", metrics.confusion_matrix(CV, predicted,
['No','Yes']))

# Calculating 5 fold cross validation results
model = GaussianNB()
kf = KFold(len(CV), n_folds=10)
scores = cross_val_score(model, data, CV, cv=kf)
print("Accuracy of every fold in 10 fold cross validation: ", abs(scores))
print("Mean of the 5 fold cross-validation: %0.2f" % abs(scores.mean()))
```

E4 - Logistic Regression

```
import csv
import pandas as pd
import numpy as np
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold, cross_val_score

# prepare datasets to be fed in the regression model
dataset = pd.read_csv('Workbook5Clean.csv')
DS_Scorers = pd.read_csv('Group_DataScorers.csv')
DS_Specialists = pd.read_csv('Group_DataSpecialists.csv')
DS_Centers = pd.read_csv('Group_DataCenters.csv')
print(dataset)
#predict attend class given extra hours and grade
CV =  dataset.Drafted.reshape((len(dataset.Drafted), 1))
data = (dataset.ix[:,'Class':'USGp'].values).reshape((len(dataset.Drafted), 32))

# Create a KNN object
LogReg = LogisticRegression()

# Train the model using the training sets
LogReg.fit(data, CV)

# the model
print('Coefficients (m): \n', LogReg.coef_)
print('Intercept (b): \n', LogReg.intercept_)

#predict the class for each data point
predicted = LogReg.predict(data)
print("Predictions: \n", np.array([predicted]).T)

df = pd.DataFrame()
df = np.array([predicted]).T
```

```
with open('LRPredictions.csv', 'w') as fp:
   writer = (csv.writer(fp)).writerows(df)

# predict the probability/likelihood of the prediction
print("Probability of prediction: \n",LogReg.predict_proba(data))

print("Accuracy score for the model: \n", LogReg.score(data,CV))

#print(metrics.confusion_matrix(CV, predicted, labels=["Yes","No"]))

# Calculating 5 fold cross validation results
model = LogisticRegression()
kf = KFold(len(CV), n_folds=10)
scores = cross_val_score(model, data, CV, cv=kf)
print("Accuracy of every fold in 10 fold cross validation: ", abs(scores))
print("Mean of the 5 fold cross-validation: %0.2f" % abs(scores.mean()))
```

E5 - Clustering

```
from sklearn.cluster import KMeans
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

dataset = pd.read_csv('Workbook5CleanClustering.csv')
#dataset_without_drafted = dataset
dataset_without_drafted = dataset[dataset.Drafted == 1]

KM = KMeans(n_clusters=3, init='k-means++', random_state=170)

KM = KM.fit(dataset_without_drafted)

print("The cluster centroids are: \n", KM.cluster_centers_)
print("Cluster", KM.labels_)
print("Sum of distances of samples to their closest cluster center: ", KM.inertia_)

#colors = ['black','red']
#plt.scatter(dataset_with_drafted.ASTp, dataset_with_drafted.BLKp, c=KM.labels_,
cmap=matplotlib.colors.ListedColormap(colors), s=75)
#plt.show()

fig = plt.figure(1, figsize=(10, 8))
ax = Axes3D(fig, elev=-150, azim=110)

colormap = np.array(['indigo', 'teal', 'gold'])
ax.scatter(dataset_without_drafted.TRBp, dataset_without_drafted.FG,
dataset_without_drafted.ASTp,  c=colormap[KM.labels_], s=100)
plt.show()
```

References

[1] N. (2017, October 17). CARMELO NBA Player Projections. Retrieved December 02, 2017, from https://projects.fivethirtyeight.com/carmelo/

[2]N. (2017, October 17). CARMELO NBA Player Projections. Retrieved December 02, 2017, from https://projects.fivethirtyeight.com/carmelo/robert-covington/

[3]KATOH's 2017 Top-100 Prospects. (n.d.). Retrieved December 02, 2017, from https://www.fangraphs.com/blogs/katohs-2017-top-100-prospects/

[4]Projecting Clint Frazier. (n.d.). Retrieved December 02, 2017, from https://www.fangraphs.com/blogs/projecting-clint-frazier/

[5]N. (2017, April 21). Projecting The Top 50 Players In The 2015 NBA Draft Class. Retrieved December 02, 2017, from https://fivethirtyeight.com/features/projecting-the-top-50-players-in-the-2015-nba-draft-class/

[6]Robert Covington NBA Salary. (n.d.). Retrieved December 02, 2017, from http://hoopshype.com/player/robert-covington/salary/