Predicting Career Longevity of NBA Rookies

Abstract

With the competition among players for positions on NBA teams rising, predicting the success of a rookie player is critical. In this project, we examine the success of a rookie by predicting whether a rookie player will last five or more years in the NBA. Using 16 continuous, quantitative, predictor variables, we use support vector machines to fit a separating hyperplane with a soft margin. We fit a linear, polynomial, and radial hyperplane before determining which model performs the best. Ultimately, we find that the hyperplane found using the radial support vector machine performs the best with an accuracy of 73.82%, sensitivity of 81.43%, specificity of 61.54, and precision of 77.38%.

Introduction

The National Basketball Association contains much more talented players now than it did in the 20th century. The arisen talent can be attributed to greater competition. There are more basketball players now than ever before which allows the NBA to be more selective in choosing its incoming talent. Sports science has also played an important role. More modern training equipment and techniques have allowed players to better develop their bodies and skill sets to unprecedented levels. The availability of technology has given all youth basketball players easy access to learn and better develop their skills. The younger generations of basketball players keep getting better and better which means the NBA coaches should put more emphasis on the importance of the rookies. Coaches are going to want to ensure that these rookie players will be valuable assets to the team, especially for the teams that are looking to rebuild. To be successful in the future, NBA coaches should have an increased concentration on the longevity of these rookies' careers. In our analysis, we will use a support vector classifier and support vector machines to create models that make the most accurate predictions on whether or not a rookie NBA player will last at least five years.

Methods

Data Collection - The NBA dataset contains information on game statistics for NBA rookies who were categorized as "seasonal leaders" and the longevity of their NBA careers. The data was collected by the NBA between 1980 and 2016. There are 1350 observations, 19 quantitative predictors, and one qualitative response. [1]

Variable Creation - The response variable is qualitative and classifies the longevity of an NBA player's career based off of performance in rookie year. The response was originally classified as 1 for rookies with a career length greater than or equal to 5 years, and 0 if their career length is less than 5 years. To implement a support vector classifier, we change the classification from 1 to +1 and 0 to -1, respectively. The null classification of a rookie's career as longer than 5 years is 62.01%.

There are 19 quantitative predictor variables that we will use to predict the response variable. The specific definitions for each variable can be found in the Appendix. Because some of the variables were discrete numerical variables, we scale all quantitative predictors so that every variable is a continuous, numerical variable. We do this by subtracting the mean value of the variable from each observation and then divide that difference by the standard deviation. Notably, there are instances where a variable is calculated using other variables. For example, the percentage of free throws made is calculated using the number of free throw attempts as well as the number of free throws made. Therefore, we would expect this percentage to be correlated with the other two variables. There are three variables in the data that are percentages calculated using two other variables: free throw percentage, field goal percentage, and 3-point percentage.

We removed these three predictor variables from the dataset before beginning any machine learning. Additionally, the variable of rebounds is calculated by combining the variables, offensive rebounds and defensive rebounds. In the Appendix, we look at the specific correlations between the variables in these instances.

Analytic Methods - To predict whether a rookie's career will be 5 years or longer, we construct both a support vector classifier and support vector machine. Before using the observations to create models, we divided the data into a train-test split with 1000 training observations and 340 testing observations. For all modeling we use the training data only. First, we implement the support vector classifier which is an extension of the maximal margin classifier. Unlike the maximal margin classifier, the support vector classifier has some tolerance for misclassifications. Because of the use of a support vector classifier, our data does not need to be linearly separable. Additionally, the support vector classifier increases the robustness of individual observations and often better classifies most of the training observations. Like a maximal margin classifier, the support vector classifier fits a separating linear hyperplane that allows us to classify our data. Because there are an infinite number of potential separating hyperlanes, the support vector classifier helps us find the hyperplane that maximizes the distance between the plane and the closest data point(s), known as the support vector(s). Because we have 16 predictor variables, our hyperplane will have 16 dimensions. For this reason we are not able to graphically represent our entire model, but only project the hyperplane onto a two dimensional feature-space containing two variables.

Using the 'e1071' library in R, we create a support vector classifier that minimizes misclassification through selecting a cost hyperparameter through cross-validation. Cost indicates the size of the margin. As cost increases, the margin becomes tighter and tolerance from misclassification decreases. To determine the best value for cost, we experimented with a range of cost values and looked at the number of support vectors in the model as well as the number of correct predictions. In this experimentation, we noted that once the cost became greater than 10^4 the number of support vectors significantly increased and the number of correct predictions significantly decreased. We noted a similar change to support vectors and number of correct predictions once the cost became smaller than 10^{-1}. Because of this, we tested the values of 0.01, 0.1, 1, 10, 100, 10^3 using the tune function to determine which cost minimized the test error. We found a cost of 1 as the best, with a test error of 0.306. (See Appendix for full results). We examine decision boundaries between different sets of two predictor variables (i.e. points and games played) in order to make predictions on whether a player will last at least five years.

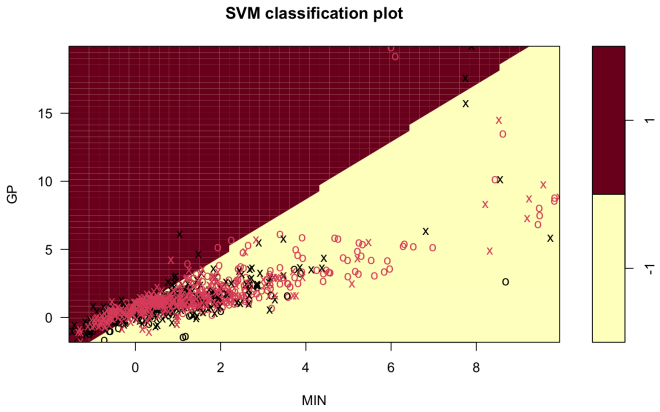
After implementing the support vector classifier to obtain a linear separating hyperplane, we implement two support vector machines. The support vector machines allow us to find a non-linear separating hyperplane. One will allow for a polynomial transformation using the polynomial kernel and the other will allow for a radial transformation using the radial kernel. For both support vector machines, we need to conduct cross-validation for both cost as well as a new

hyperparameter. For the polynomial kernel, the new hyperparameter is degree and for the radial kernel, the new hyperparameter is gamma. Like our process for support vector classification, we experimented with different values for degree and gamma before creating a range of values to test using the tune function. For degree, we decided to test the values of 1, 2, 3, and 4, and for gamma, we decided to test the values of 0.05, 0.01, 0.1, and 1. We used the test values for cost of .01, .1, 1, 10, 100, and 10^3 in cross-validation in determining the polynomial and radial kernel. We obtained a cost of 10 and degree of 1 with a test error of .312 for the polynomial kernel. We obtained a cost of 100 and gamma of .01 with a test error of .299 for the radial kernel.

Once we obtain the support vector classifier and the two support vector machines, we construct train and test confusion matrices. Using the values of true positive, false positive, true negative, and false negative from the test confusion matrices, we are able to calculate the measures of accuracy, sensitivity, specificity and precision. We then use these measures to evaluate which support vector machine performs the best.

Results

Support Vector Classifier (kernel = linear, cost = 1)



Here is a plot of MIN versus GP where the linear separating hyperplane is projected onto the two dimensional feature space. We can see that observations in the red area are classified at +1. See the equation for the hyperplane in the appendix.

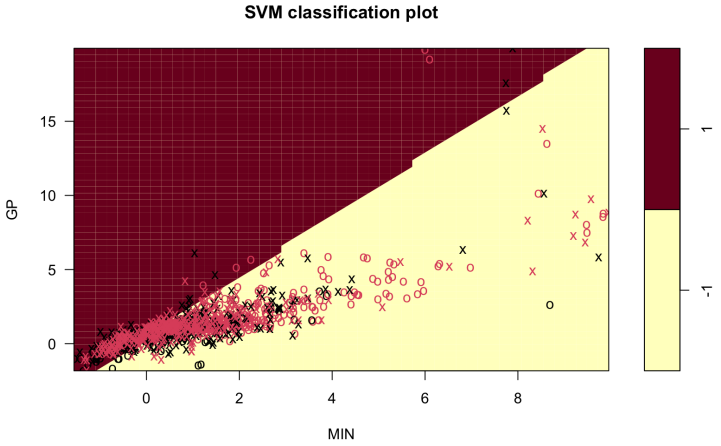
Train Confusion Matrix

	Truly did not last 5 years	Truly last 5 years	**Accuracy: 70.60%** **Sensitivity: 78.74%** **Specificity: 57.26%** **Precision: 75.12%**
Predicted not to last 5 years	217	132	
Predicted to last 5 years	162	489	

Test Confusion Matrix

	Truly did not last 5 years	Truly last 5 years	**Accuracy: 71.18%** **Sensitivity: 79.05%** **Specificity: 58.46%** **Precision: 75.45%**
Predicted not to last 5 years	76	44	
Predicted to last 5 years	54	166	

Support Vector Machine (kernel = polynomial, cost = 10, degree = 1)



Here is a plot of MIN versus GP where the polynomial separating hyperplane is projected onto the two dimensional feature space. We can see that observations in the red area are classified at +1.

Train Confusion Matrix

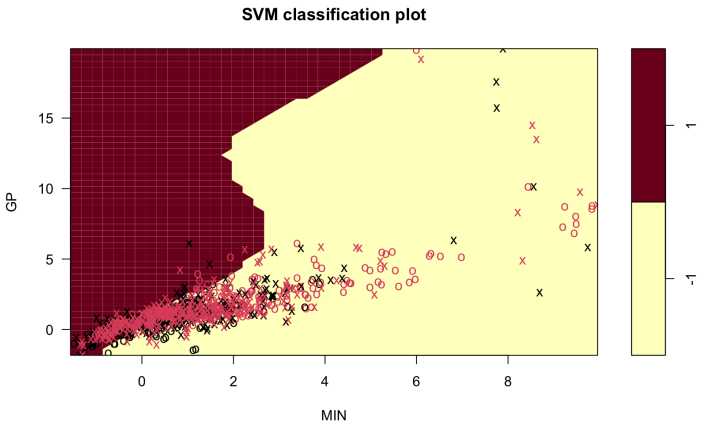
	Truly did not last 5 years	Truly last 5 years	**Accuracy: 70.80%** **Sensitivity: 79.07%** **Specificity: 57.26%** **Precision: 75.12%**
Predicted not to last 5 years	217	130	
Predicted to last 5 years	162	491	

Test Confusion Matrix

	Truly did not last 5 years	Truly last 5 years	**Accuracy: 71.47%** **Sensitivity: 79.52%** **Specificity: 58.46%** **Precision: 75.57%**
Predicted not to last 5	76	43	

years			
Predicted to last 5 years	54	167	

Support Vector Machine (kernel = radial, cost = 100, gamma = 0.01)



Here is a plot of MIN versus GP where the radial separating hyperplane is projected onto the two dimensional feature space. We can see that observations in the red area are classified at +1.

Train Confusion Matrix

	Truly did not last 5 years	Truly last 5 years	**Accuracy: 75.10%** **Sensitivity: 82.29%** **Specificity: 63.32%** **Precision: 78.62%**
Predicted not to last 5 years	240	110	
Predicted to last 5 years	139	511	

Test Confusion Matrix

	Truly did not last 5 years	Truly last 5 years	**Accuracy: 73.82%** **Sensitivity: 81.43%** **Specificity: 61.54%** **Precision: 77.38%**
Predicted not to last 5 years	80	39	
Predicted to last 5 years	50	171	

Discussion/Conclusion

We have created three different models in order to determine how to most accurately predict rookie longevity. The three models consist of a support vector classifier, a support vector machine with a polynomial kernel, and a support vector machine with a radial kernel. In examining the test confusion matrices and their associated metrics, it is quite clear that the support vector machine with the radial kernel is the best model as it contains the highest accuracy, sensitivity, specificity, and precision of all three models, 73.82%, 81.43%, 61.54%, and 77.38%, respectively. The accuracy of 73.82% was a notable increase as compared to our null model of 62.01%. Our primary concern is accuracy, so this model has constructed the hyperplane that best minimizes the number of misclassifications. However, NBA rookies are becoming increasingly more talented than ever before, so coaches should want to ensure that the rookies drafted to their teams are going to be able to last for at least five years. A better approach in creating a model may encompass prioritizing the minimization of false positives (predicting a rookie will last five years when in actuality he does not). This corresponds to maximizing the specificity of the model at a small cost for other metrics and is a basis for future work.

There are several limitations to our model. In the effort to increase accuracy by decreasing the variance of our support vector machine with a radial kernel, the coefficients were regularized. The model became much less interpretable in the process. The greatest limitation concerns the necessary structure of our dataset in creating the support vector machine. All predictor variables must be numeric while the response variable must be qualitative. We may only predict whether or not a NBA rookie will last at least five years, yet it is much more beneficial to create a model that predicts the number of years the rookie will last. In the instance that we wanted to predict a variable such as number of points scored, we would need to convert the numerical observations to categorical by assessing whether the rookie scored at least a certain number of points which is far less efficient. Also we would never be able to use rookie longevity as a predictor without knowing the actual number of years each rookie has lasted. Support vector machines are very restrictive on the type of observations and response that may be used which limits the possible analysis. Lastly, many of our predictors are highly correlated with each other. The sets of predictors that involve 3-pointers, field goals, free throws, and rebounds exhibit multicollinearity which means that the coefficients of our model may be misrepresented. The coefficient of FTM is negative which implies that more free throws made predicts a rookie will not last as long in the league. FTM is extremely correlated with FTA ($r = .98$), so it is likely the case that FTA has some influence on the true impact of FTM. The coefficients of our support vector classifier are not as meaningful due to this colinearity and the coefficients for the support vector machines were not calculated as it would require more extensive, complex calculations.

References

[1] “ML Classification: Career Longevity for NBA Players - Project by Ssaudz.” *Data.world*, 7 Sept. 2017, data.world/ssaudz/ml-classification-predicting-5-year-career-longevity-for-nb/workspace/file?agentid=exercises&datasetid=logistic-regression-exercise-1&filename=nba_logreg.csv.

[2] Honkasalo, Mika. “Sorry, Old-School Guys: Modern-Day NBA Players Are Better than Ever.” *HoopsHype*, HoopsHype, 12 Mar. 2016, hoopshype.com/2016/03/11/why-nba-players-are-better-than-ever/.

Appendix

Variable Description

The response variable is categorical.

TARGET_5yrs	–1 = career < 5 years
	+1 = career ≥ 5 years

All explanatory variables are numeric.

GP	Games played
MIN	Minutes played
PTS	Points per game
FGM	Field goals made
FGA	Field goal attempts
FG%	Field goal percent
3P Made	3-Points made
3PA	3-Point attempts
3P%	3-Point percent
FTM	Free throws made
FTA	Free throw attempts
FT%	Free throw percent
OREB	Offensive rebound
DREB	Defensive rebound
REB	Rebounds
AST	Assists
STL	Steals
BLK	Blocks
TOV	Turnovers

Cross-Validation for Hyperparameter Selection

Determining Cost:

```
- Detailed performance results:
   cost error dispersion
1 1e-02 0.370 0.05228129
2 1e+00 0.306 0.04299871
3 1e+01 0.313 0.04715224
4 1e+02 0.312 0.05223877
5 1e+03 0.319 0.04863698
```

Determining Cost and Degree:

```
> tuning_cost_poly$performances
    cost degree error dispersion
1  1e-02      1 0.379 0.03071373
2  1e+00      1 0.313 0.04831609
3  1e+01      1 0.312 0.06160808
4  1e+02      1 0.312 0.06142746
5  1e-02      2 0.381 0.03142893
6  1e+00      2 0.382 0.03326660
7  1e+01      2 0.365 0.04326918
8  1e+02      2 0.391 0.05820462
9  1e-02      3 0.384 0.03204164
10 1e+00      3 0.394 0.03134042
11 1e+01      3 0.371 0.05989806
12 1e+02      3 0.364 0.05211099
13 1e-02      4 0.384 0.03405877
14 1e+00      4 0.403 0.02710064
15 1e+01      4 0.398 0.03392803
16 1e+02      4 0.401 0.03754997
```

Determining Cost and Gamma:

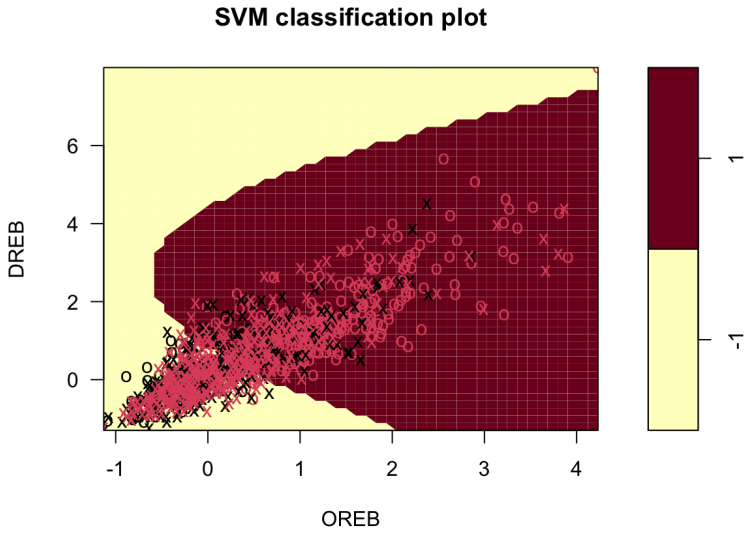
```
> tuning_cost_radial$performances
    cost gamma error dispersion
1  1e-02  0.05 0.379 0.05152130
2  1e+00  0.05 0.319 0.05043147
3  1e+01  0.05 0.304 0.05738757
4  1e+02  0.05 0.335 0.05816643
5  1e-02  0.01 0.379 0.05152130
6  1e+00  0.01 0.331 0.04408325
7  1e+01  0.01 0.317 0.04217688
8  1e+02  0.01 0.299 0.05425250
9  1e-02  0.10 0.379 0.05152130
10 1e+00  0.10 0.317 0.04498148
11 1e+01  0.10 0.335 0.04813176
12 1e+02  0.10 0.371 0.05646041
13 1e-02  1.00 0.379 0.05152130
14 1e+00  1.00 0.341 0.04280446
15 1e+01  1.00 0.359 0.05384133
16 1e+02  1.00 0.362 0.04779586
```

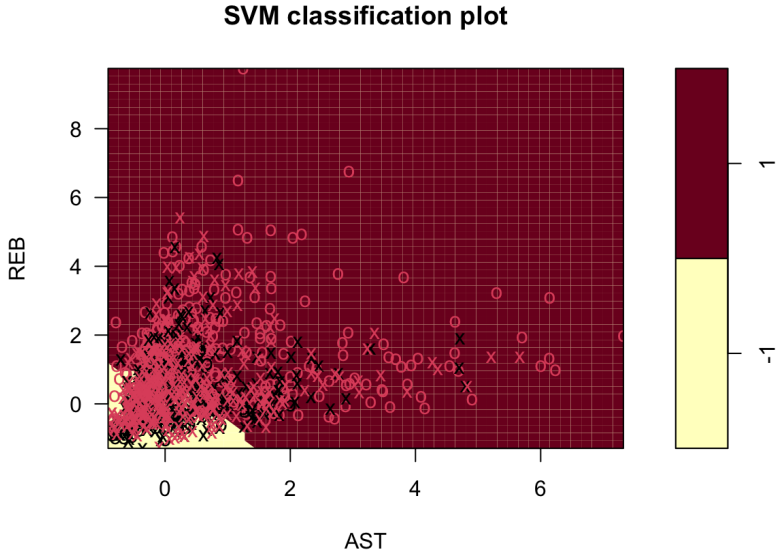
Support Vector Classifier - Hyperplane

This is the hyperplane we obtained using the support vector classifier with a cost of 1. If Target_5yrs is larger than zero (positive) we would classify the observation as +1 and if the Target_5yrs is smaller than zero (negative) we would classify the observation as -1.

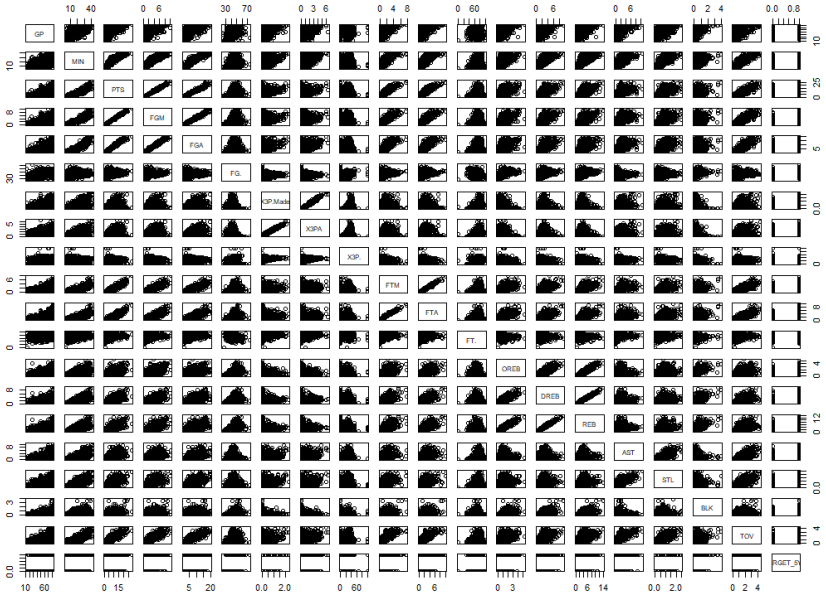
Target_5yrs = .0889 -.4617(GP) + .9675(MIN) - .8307(PTS) - .4656(FGM) + .3199(FGA) - .6348(3P Made) - .5730(3PA) - .8826(FTM) + .8330(FTA) - .9553(OREB) - .3912(DREB) + .3748(REB) - .5107(AST) - .1346(STL) - .1995(BLK) + .4162(TOV)

Two Variable Scatter Plots with Radial Hyperplane Projection

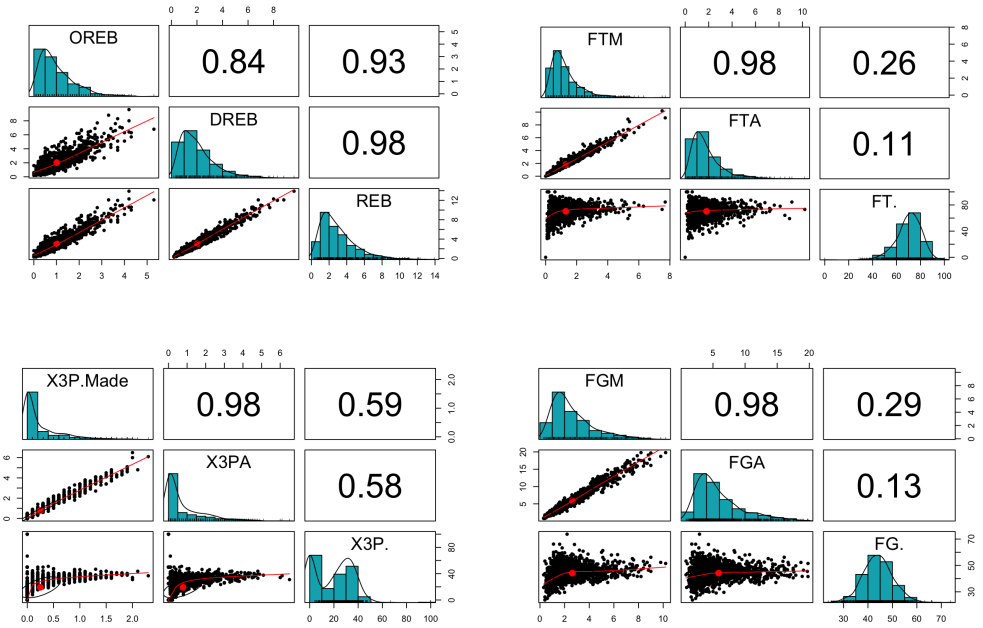




Predictor Variable Correlations

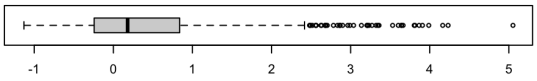


Correlation Matrices

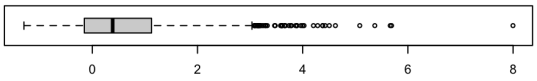


Boxplots of the Variables

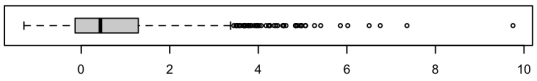
Free Throw Attempts (FTA)



Offensive Rebounds (OREB)

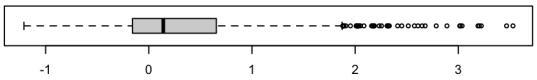


Defensive Rebounds (DREB)

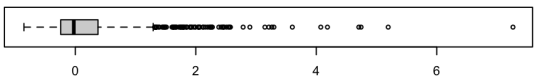


Rebounds (REB)

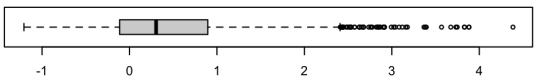
Assists (AST)



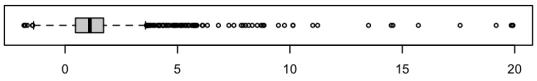
Steals (STL)



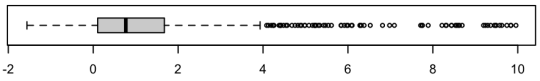
Blocks (BLK)



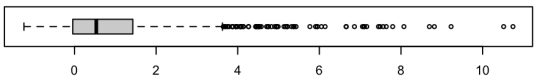
Turnovers (TOV)



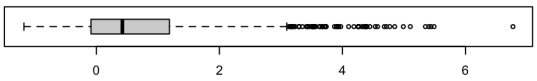
Games Played (GP)



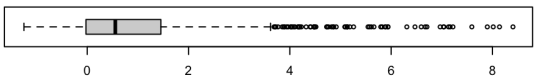
Minutes Played (MP)



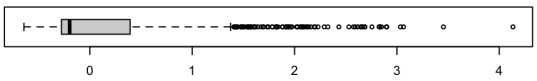
Points Per Game (PTS)



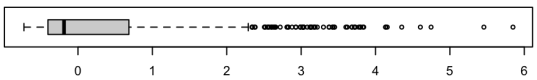
Field Goals Made (FGM)



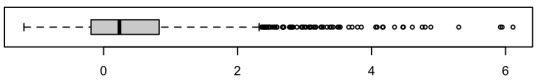
Field Goal Attempts (FGA)



3 Point Made (3P Made)

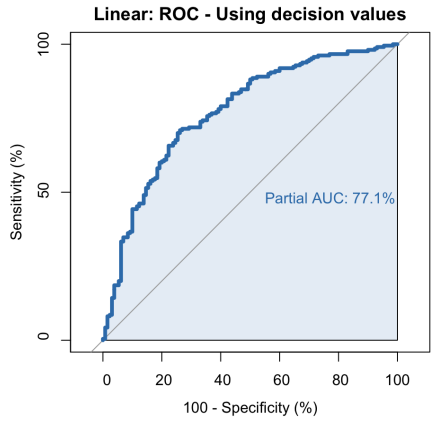


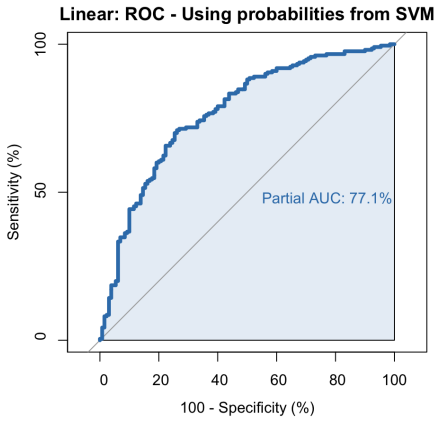
3 Point Attempts (3PA)

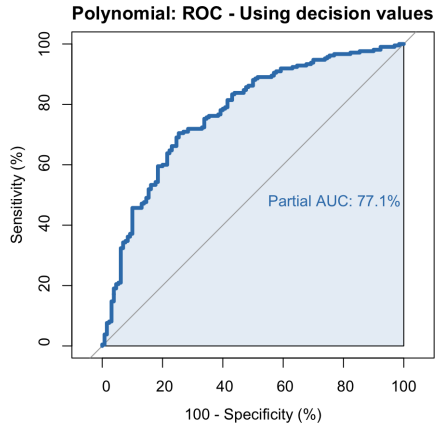


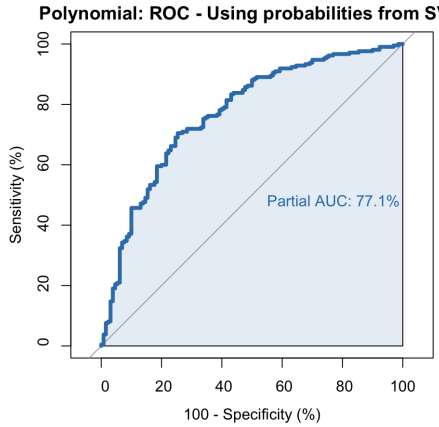
Free Throws Made (FTM)

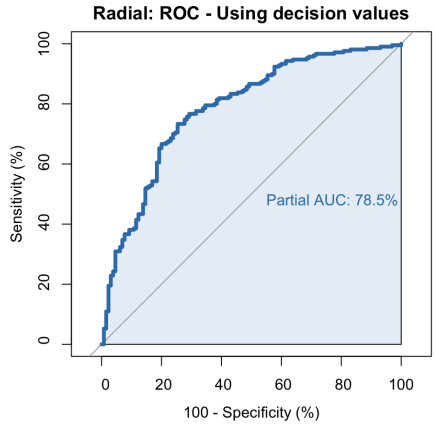
ROC Curves

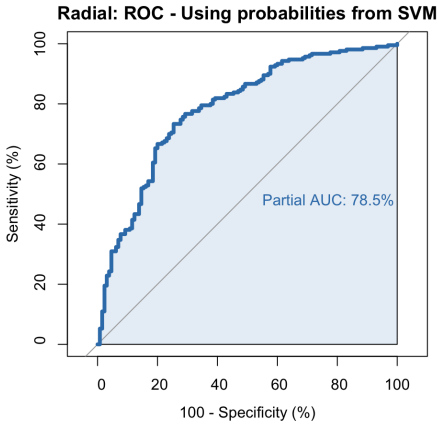

Linear: ROC - Using decision values
Sensitivity (%)
100 - Specificity (%)
Partial AUC: 77.1%


Linear: ROC - Using probabilities from SVM
Sensitivity (%)
100 - Specificity (%)
Partial AUC: 77.1%


Polynomial: ROC - Using decision values
Sensitivity (%)
100 - Specificity (%)
Partial AUC: 77.1%


Polynomial: ROC - Using probabilities from SVM
Sensitivity (%)
100 - Specificity (%)
Partial AUC: 77.1%


Radial: ROC - Using decision values
Sensitivity (%)
100 - Specificity (%)
Partial AUC: 78.5%


Radial: ROC - Using probabilities from SVM
Sensitivity (%)
100 - Specificity (%)
Partial AUC: 78.5%

R Code -

```
# ================= DATA ======

nba = read.csv(file = "nba1.csv")

# removing names from data
nba = nba[,-c(1)]

null_probability = sum(nba$TARGET_5Yrs)/length(nba$TARGET_5Yrs)

# responce: (1, >= 5 years, -1 < 5 years)
nba$TARGET_5Yrs = ifelse(nba$TARGET_5Yrs == 1, 1, -1)

nba_scale = nba
R = length(nba_scale$GP)
V = 19
for(j in 1:V){
  for(i in 1:R) {
    average = mean(nba_scale[,j])
    standev = sd(nba_scale[,j])
    nba_scale[i,j] = (nba_scale[i,j]-average)/standev
  }
}

nba_scale = nba_scale[, -c(6, 9, 12)]
View(nba_scale)

response = nba_scale$TARGET_5Yrs
nba_scale$TARGET_5Yrs = as.factor(response)

### CREATING TRAIN-TEST SPLITS

set.seed(2021)
train_indexes = sample(1340, 1000)
test_indexes = -train_indexes


#================== SUPPORT VECTOR MACHINE: LINEAR ===========

install.packages('e1071')
library(e1071)
```

```
svm_nba = svm(TARGET_5Yrs~.,
              data = nba_scale, ### eventually this should be training data
              subset = train_indexes,
              kernel = 'linear',
              scale = F,
              cost = 1)

summary(svm_nba)
svm_nba$SV
svm_nba$index
svm_nba$decision.values
svm_nba$coefs

yhat_test = predict(object = svm_nba,
                    newdata = nba_scale[test_indexes, ],
                    decision.values = F)

ytrue_test = nba$TARGET_5Yrs[test_indexes]

svm_nba$coefs

table(yhat_test, ytrue_test)

# cross validation to find the best cost value
set.seed(2021)
tuning_cost = tune(method = svm,
                   TARGET_5Yrs~.,
                    data = nba_scale[train_indexes,],
                    kernel = 'linear',
                    scale = F,
                    ranges = list(cost = c(.01, 1)),
                    probability = T)

summary(tuning_cost)

tuning_cost$best.parameters
tuning_cost$best.performance
tuning_cost$best.model
tuning_cost$performances

best_svm_nba = tuning_cost$best.model
summary(best_svm_nba)

#================== LINEAR: TRAIN - TEST CONFUSION MATRICIES ===========
```

```
# training confusion matrix
yhat_train = predict(object = best_svm_nba,
          newdata = nba_scale[train_indexes,],
          descision.values = T)
yhat_train

ytrue_train = nba_scale$TARGET_5Yrs[train_indexes]
table(yhat_train, ytrue_train)

# testing confusion matrix
yhat_test = predict(object = best_svm_nba,
                newdata = nba_scale[test_indexes,],
                decision.values = T)

ytrue_test = nba_scale$TARGET_5Yrs[test_indexes]


table(yhat_test, ytrue_test)

#================== LINEAR: ROC CURVES ==================
library(pROC)

yhat_test_linear_roc = predict(object = best_svm_nba,
                        newdata = nba_scale[test_indexes,],
                        decision.values = T,
                        probability = T)

linear_roc_probs = attributes(yhat_test_linear_roc)$probabilities
linear_roc_preds = attributes(yhat_test_linear_roc)$decision.values

par(mfrow = c(1,2))
roc(ytrue_test, linear_roc_preds,
   plot = T,
   main = 'Linear: ROC - Using decision values',
   legacy.axes = T,
   lwd = 4,
   col="#377eb8",
   print.auc = T,
   percent = T,
   print.auc.x = 45,
   partial.auc=c(100, 0),
   auc.polygon = TRUE,
```

```
    auc.polygon.col = "#377eb822")
roc(ytrue_test, linear_roc_probs[,2],
    plot = T,
    main = 'Linear: ROC - Using probabilities from SVM',
    legacy.axes = T,
    lwd = 4,
    col="#377eb8",
    print.auc = T,
    percent = T,
    print.auc.x = 45,
    partial.auc=c(100, 0),
    auc.polygon = TRUE,
    auc.polygon.col = "#377eb822")


#================== LINEAR: FITTING PARAMETERS ======================

# trying to determine fitted parameters

# obtain nonzero Lagrange multipliers
lambdas = best_svm_nba$coefs
View(lambdas)
lambdas_row = t(lambdas) # transpose
View(lambdas_row) # as row

# obtain support vectors
support_vectors = best_svm_nba$SV

# beta = sum lambdas*support vectors
beta = lambdas_row %*% support_vectors
View(beta)
beta
beta0 = - best_svm_nba$rho
beta0

#================== LINEAR: PLOTTING (BY HAND) ============================
#plot decision boundaries for two selected variables
n = 4
m = 5
x1 = nba_scale[,n]
x2 = nba_scale[,m]

par(mfrow = c(1,1))
plot(x2~x1, pch = 20, cex = 1, main = "Games Played vs. Minutes Played",
```

```
    xlab = "Games Played (GP)", ylab = "Minutes Played (MP)" )
points(x2[nba_scale$TARGET_5Yrs == 1]~x1[nba_scale$TARGET_5Yrs == 1],
       pch = 20, cex = 1, col = 'darkgreen')
points(x2[nba_scale$TARGET_5Yrs == -1]~x1[nba_scale$TARGET_5Yrs == -1],
       pch = 20, cex = 1, col = 'darkred')

abline(-beta0 / beta[m], -beta[n] / beta[m])
abline((-beta0 - 1) / beta[m], -beta[n] / beta[m], lty = 2)
abline((-beta0 + 1) / beta[m], -beta[n] / beta[m], lty = 2)

#================== LINEAR: PLOTTING ==================
plot(best_svm_nba, data = nba_scale[train_indexes,], GP~MIN)


#================== SUPPORT VECTOR MACHINE: POLYNOMIAL ===========

svm_nba_poly = svm(TARGET_5Yrs~.,
            data = nba_scale, ### eventually this should be training data
            subset = train_indexes,
            kernel = 'polynomial',
            scale = F,
            cost = 1,
            degree = 13)

yhat_test = predict(object = svm_nba_poly,
                    newdata = nba_scale[test_indexes, ],
                    decision.values = F)

ytrue_test = nba_scale$TARGET_5Yrs[test_indexes]

summary(svm_nba_poly)
table(yhat_test, ytrue_test)

#degree, neg correct preds, pos correct preds
#1, 72, 175
#2, 0, 208
#3, 1, 208
#4, 2, 204
#5, 2, 198
#10, 4, 187
#13, 4, 185


tuning_cost_poly = tune(method = svm,
```

```
                TARGET_5Yrs~.,
                data = nba_scale[train_indexes,],
                kernel = 'polynomial',
                scale = F,
                ranges = list(cost = c(1, 10),
                              degree = c(1, 2)),
                probability = T)


summary(tuning_cost_poly)

tuning_cost_poly$best.parameters
tuning_cost_poly$best.performance
tuning_cost_poly$best.model
tuning_cost_poly$performances

best_svm_nba_poly = tuning_cost_poly$best.model
summary(best_svm_nba_poly)

#================== POLYNOMIAL: TRAIN - TEST CONFUSION MATRICIES
===========


# training confusion matrix
yhat_train_poly = predict(object = best_svm_nba_poly,
                          newdata = nba_scale[train_indexes,],
                          descision.values = T)
ytrue_train_poly = nba_scale$TARGET_5Yrs[train_indexes]

table(yhat_train_poly, ytrue_train_poly)

# testing confusion matrix
yhat_test_poly = predict(object = best_svm_nba_poly,
                         newdata = nba_scale[test_indexes,],
                         decision.values = T)
ytrue_test_poly = nba_scale$TARGET_5Yrs[test_indexes]

table(yhat_test_poly, ytrue_test_poly)

#================== POLYNOMIAL: ROC CURVES ==================
library(pROC)

yhat_test_poly_roc = predict(object = best_svm_nba_poly,
                             newdata = nba_scale[test_indexes,],
```

```
                        decision.values = T,
                        probability = T)

poly_roc_probs = attributes(yhat_test_poly_roc)$probabilities
poly_roc_preds = attributes(yhat_test_poly_roc)$decision.values

par(mfrow = c(1,2))
roc(ytrue_test_poly, poly_roc_preds,
    plot = T,
    main = 'Polynomial: ROC - Using decision values',
    legacy.axes = T,
    lwd = 4,
    col="#377eb8",
    print.auc = T,
    percent = T,
    print.auc.x = 45,
    partial.auc=c(100, 0),
    auc.polygon = TRUE,
    auc.polygon.col = "#377eb822")
roc(ytrue_test_poly, poly_roc_probs[,1],
    plot = T,
    main = 'Polynomial: ROC - Using probabilities from SVM',
    legacy.axes = T,
    lwd = 4,
    col="#377eb8",
    print.auc = T,
    percent = T,
    print.auc.x = 45,
    partial.auc=c(100, 0),
    auc.polygon = TRUE,
    auc.polygon.col = "#377eb822")

#================== POLYNOMIAL: FITTING PARAMETERS ======================

# trying to determine fitted parameters

# obtain nonzero Lagrange multipliers
lambdas_poly = best_svm_nba_poly$coefs
View(lambdas_poly)
lambdas_row_poly = t(lambdas_poly) # transpose
View(lambdas_row_poly) # as row

# obtain support vectors
support_vectors_poly = best_svm_nba_poly$SV
```

```
# beta = sum lambdas*support vectors
beta_poly = lambdas_row_poly %*% support_vectors_poly
View(beta_poly)
beta_poly
beta0_poly = - best_svm_nba_poly$rho
beta0_poly



#================== POLYNOMIAL: PLOTTING ==================
plot(best_svm_nba_poly, data = nba_scale[train_indexes,], GP~MIN)



#================== SUPPORT VECTOR MACHINE: RADIAL ===========

svm_nba_radial = svm(TARGET_5Yrs~.,
                     data = nba_scale, ### eventually this should be training data
                     subset = train_indexes,
                     kernel = 'radial',
                     scale = F,
                     cost = 1,
                     gamma = 0.05,
                     probability = T)

yhat_test = predict(object = svm_nba_radial,
                    newdata = nba_scale[test_indexes, ],
                    decision.values = F)

ytrue_test = nba_scale$TARGET_5Yrs[test_indexes]

summary(svm_nba_radial)
table(yhat_test, ytrue_test)

#gamma, neg correct preds, pos correct preds
#10, 1, 206
#5, 8, 205
#2, 37, 191
#1, 54, 180
#.1, 79, 172
#.01, 64, 184
#.05, 78, 173
#.001, 0, 210
```

```
#4, 2, 204
#5, 2, 198
#10, 4, 187
#13, 4, 185

set.seed(2021)
tuning_cost_radial = tune(method = svm,
                          TARGET_5Yrs~.,
                          data = nba_scale[train_indexes,],
                          kernel = 'radial',
                          scale = F,
                          ranges = list(cost = c(.01, 1, 10, 100),
                                        gamma = c(0.05, 0.01, 0.1, 1)),
                          probability = T)

summary(tuning_cost_radial)

tuning_cost_radial$best.parameters
tuning_cost_radial$best.performance
tuning_cost_radial$best.model
tuning_cost_radial$performances

best_svm_nba_radial = tuning_cost_radial$best.model
summary(best_svm_nba_radial)

#================== RADIAL: TRAIN - TEST CONFUSION MATRICIES ===========


# training confusion matrix
yhat_train_radial = predict(object = best_svm_nba_radial,
                            newdata = nba_scale[train_indexes,],
                            descision.values = T)
ytrue_train_radial = nba_scale$TARGET_5Yrs[train_indexes]

table(yhat_train_radial, ytrue_train_radial)

# testing confusion matrix
yhat_test_radial = predict(object = best_svm_nba_radial,
                           newdata = nba_scale[test_indexes,],
                           decision.values = T)
ytrue_test_radial = nba_scale$TARGET_5Yrs[test_indexes]

table(yhat_test_radial, ytrue_test_radial)
```

```
#================== RADIAL: ROC CURVES ==================
library(pROC)

yhat_test_radial_roc = predict(object = best_svm_nba_radial,
                        newdata = nba_scale[test_indexes,],
                        decision.values = T,
                        probability = T)

radial_roc_probs = attributes(yhat_test_radial_roc)$probabilities
radial_roc_preds = attributes(yhat_test_radial_roc)$decision.values

par(mfrow = c(1,2))
roc(ytrue_test_radial, radial_roc_preds,
    plot = T,
    main = 'Radial: ROC - Using decision values',
    legacy.axes = T,
    lwd = 4,
    col="#377eb8",
    print.auc = T,
    percent = T,
    print.auc.x = 45,
    partial.auc=c(100, 0),
    auc.polygon = TRUE,
    auc.polygon.col = "#377eb822")
roc(ytrue_test_radial, radial_roc_probs[,2],
    plot = T,
    main = 'Radial: ROC - Using probabilities from SVM',
    legacy.axes = T,
    lwd = 4,
    col="#377eb8",
    print.auc = T,
    percent = T,
    print.auc.x = 45,
    partial.auc=c(100, 0),
    auc.polygon = TRUE,
    auc.polygon.col = "#377eb822")

#================== RADIAL: PLOTTING ==================
plot(best_svm_nba_radial, data = nba_scale[train_indexes,], REB~AST)

#================== CREATING BOX PLOTS ================
par(mfrow = c(4,2))

boxplot(nba_scale[,1], horizontal = TRUE, xlab = "Games Played (GP)")
```

```
boxplot(nba_scale[,2], horizontal = TRUE, xlab = "Minutes Played (MP)")
boxplot(nba_scale[,3], horizontal = TRUE, xlab = "Points Per Game (PTS)")
boxplot(nba_scale[,4], horizontal = TRUE, xlab = "Field Goals Made (FGM)")
boxplot(nba_scale[,5], horizontal = TRUE, xlab = "Field Goal Attempts (FGA)")
boxplot(nba_scale[,6], horizontal = TRUE, xlab = "3 Point Made (3P Made)")
boxplot(nba_scale[,7], horizontal = TRUE, xlab = "3 Point Attempts (3PA)")
boxplot(nba_scale[,8], horizontal = TRUE, xlab = "Free Throws Made (FTM)")

par(mfrow = c(4,2))

boxplot(nba_scale[,9], horizontal = TRUE, xlab = "Free Throw Attempts (FTA)")
boxplot(nba_scale[,10], horizontal = TRUE, xlab = "Offensive Rebounds (OREB)")
boxplot(nba_scale[,11], horizontal = TRUE, xlab = "Defensive Rebounds (DREB)")
boxplot(nba_scale[,12], horizontal = TRUE, xlab = "Rebounds (REB)")
boxplot(nba_scale[,13], horizontal = TRUE, xlab = "Assists (AST)")
boxplot(nba_scale[,14], horizontal = TRUE, xlab = "Steals (STL)")
boxplot(nba_scale[,15], horizontal = TRUE, xlab = "Blocks (BLK)")
boxplot(nba_scale[,16], horizontal = TRUE, xlab = "Turnovers (TOV)")


#================== LOOKING AT CORRELATIONS ====
# cor between 3-point
install.packages('psych')
library(psych)

pairs.panels(nba[,c(5,6,7)],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = T,  # show density plots
             ellipses = T # show correlation ellipses
)
pairs.panels(nba[,c(8,9,10)],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = T,  # show density plots
             ellipses = T # show correlation ellipses
)
pairs.panels(nba[,c(11,12,13)],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = T,  # show density plots
             ellipses = T # show correlation ellipses
)
pairs.panels(nba[,c(14,15,16)],
```

```
        method = "pearson", # correlation method
        hist.col = "#00AFBB",
        density = T,  # show density plots
        ellipses = T # show correlation ellipses
)

plot(nba_scale[c(7, 8, 9)])
```