Machine Learning Techniques to Predict NBA Players’ Salary Category

Group 6

November 29, 2021

Samiksha Deshmukh, Ben Hickman, Johan Jacob, Yucheng Xie

Abstract

This paper discusses the efficacy of five different machine learning algorithms in predicting the salary class of NBA athletes. Our group is acting as an analytics team working for an NBA organization that needs to sign one additional player without spending more than the average salary for all players throughout the league. We used a neural network, decision tree, k-nearest neighbor, naïve bayes, and logistic regression to predict whether a player would be in a high-salary category (ie, above league average) or a low-salary category (ie, below league average). The independent variables used to predict this are in-game performance metrics, for example a player’s average points scored per game or average minutes played per game. Our findings are that the k-nearest neighbor model has the highest accuracy of all the models we ran (84%). However, the logistic model performed almost as well (84% accurate) and can tell us more information about which independent variables have a significant impact on a player falling into a high or low salary category.

**Problem Statement**

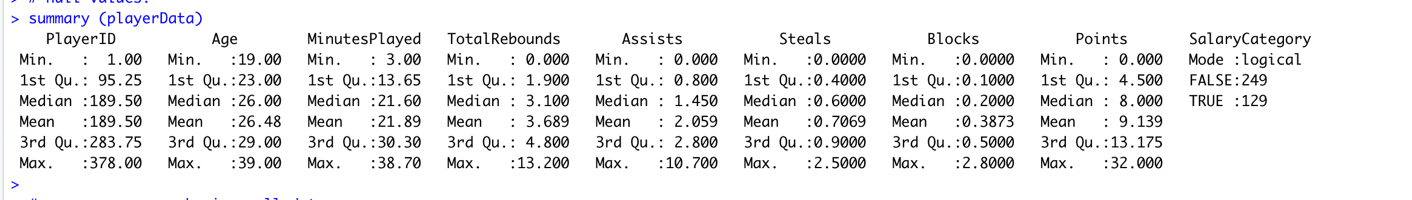
Our scenario is that we are an analytics group working for an NBA team and we are trying to predict the salary range of all players in the league. An important rule in the NBA is that each basketball organization only has a certain amount of money they are permitted to spend on the salary of their team each year (this is called the "Salary Cap"). Each year, most teams only have a small amount of money available to spend on acquiring a new player. This year, our team only has $4,651,201.05 to spend before we reach the salary cap. Due to this constraint, our manager has asked us to create an analysis that will help them predict which players are likely to earn a “low salary” so we don’t waste time or effort on trying to sign a player who will be too expensive.

This is extremely important to our organization because NBA teams do not have unlimited resources and they need to prioritize negotiating with players who we have a chance at signing. The NBA is extremely competitive, and any time spent on negotiating with a player who isn’t likely to sign with us would be much better spent negotiating with a player who is more likely to earn less than or equal to $4,651,201.05.

The data mining algorithms that we have used potentially solve our problem by identifying whether a player is likely to require a salary higher or lower than $4,651,201.05, so our organization doesn’t waste time on players who aren’t likely to play on our team.

Our dependent variable is Salary, and our independent variables are player ID, age, average minutes played per game, total rebounds per game, assists per game, steals per game, blocks per game, and points scored per game. In total, there are 9 features (including our dependent variable) and 378 instances.

Summary Statistics:



One of the interesting statistics we found in our summary table is that the age range for all players in the league for this season was from 19 to 39 years old, while the average age was 26. Another interesting thing we found was that for total rebounds, assists, steals, blocks, and points, there seem to be some outliers in each category. We know this because the max value for each category is much higher than the mean value (i.e., average total rebounds is 3.689 while the max is more than four times that, at 13.2).

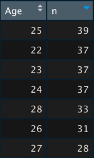
**Data Types and Description**

|  |  |  |
| --- | --- | --- |
| Column | R Tibble Data Type | Description |
| PlayerID | Integer | ID of player |
| Age | Integer | Player's age |
| MinutesPlayed | Number | This is the average minutes the player is on the court per game. A game is 48 minutes. |
| TotalRebounds | Number | Amount of rebounds a player has per game, on average over the course of the 2013-2014 season. |
| Assists | Number | Amount of assists a player has per game, on average over the course of the 2013-2014 season. |
| Steals | Number | Amount of steals a player has per game, on average over the course of the 2013-2014 season. |
| Blocks | Number | Amount of blocks a player has per game, on average over the course of the 2013-2014 season. |
| Points | Number | Amount of points a player has per game, on average over the course of the 2013-2014 season. |
| SalaryCategory | Logical | 1 = High Salary, 0 = Low Salary. A player has a High/Low Salary if he earned more than/less than the league average. |

Our dataset was scraped from two websites: [www.hoopshype.com](http://www.hoopshype.com) and [www.basketballreference.com](http://www.basketballreference.com). [www.hoopshype.com](http://www.hoopshype.com) has the salary data for every player for every season, dating back to 1990. [www.basketball-reference.com](http://www.basketballreference.com) has the in-game performance statistics for every player dating back to the 1950’s.

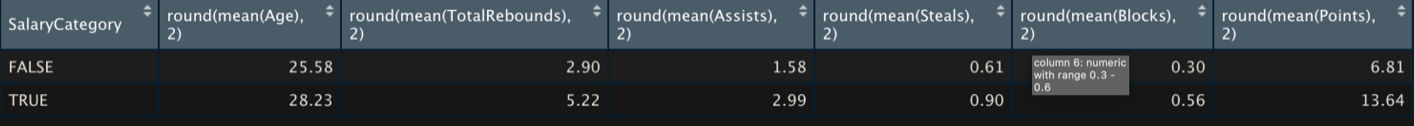
Our hypothesis is that all of our independent variables will have a significant impact on the dependent variable. The reason for this is that it is a common thought in the sports world that those players who have better statistics will get paid more money. Whether a player is an offensive powerhouse with a lot of points and assists or is a defensive centerpiece who racks up blocks, steals, and rebounds, it is commonly thought of that players who have large stat lines will be more valuable to a team and, therefore, will get offered more money.

To elaborate a bit further, we hypothesize that all independent variables will have a positive coefficient, signaling that if a player can add one more unit of any stat (whether that is one more point, one more assist, one more rebound, etc.), that player’s salary will increase to some degree. The statistics we believe will have the largest coefficients (and therefore the largest effect on salary) will be MinutesPlayed, Age, and Points. MinutesPlayed should have a large effect because it would seem reasonable to assume that a player who plays more is thought to have a greater impact on their team winning, and would also be valued more by a team, resulting in a higher salary. Age should have a large effect because as a player gains experience, their skill level will increase and they will be more impactful in terms of winning. We believe Points will have a large effect on salary because, using anecdotal evidence, the superstars of the league who get paid the most tend to outscore their teammates and opponents who get paid less.

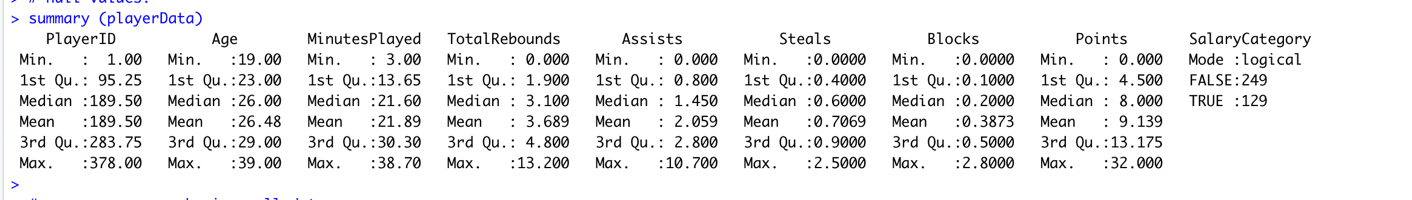
**Queries**

1. By using count() function (shown in the image to the right) on the players’ age during 2013-14 NBA regular season, we could see that the most age range for NBA players is 22-27 (Table I), which is also reasonable because in most scenarios, young players will have better body conditions. Those players who are too old will probably consider retiring from the league.

2. By using group\_by() function (shown in the below image) on the salary category to display the average point, rebound, assist, steal and block (Table II), we could have a brief understanding the average ability of different salary level players. Also, it is a useful way to help NBA franchises to decide whether to pay a player a high salary based on his average stats in this season.

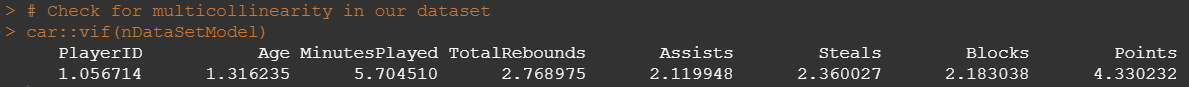


3. One of the interesting statistics we found in our summary table is that the age range for all players in the league for this season was from 19 to 39 years old, while the average age was 26. Another interesting thing we found was that for total rebounds, assists, steals, blocks, and points, there seem to be some outliers in each category. We know this because the max value for each category is much higher than the mean value (i.e., average total rebounds is 3.689 while the max is more than four times that, at 13.2).

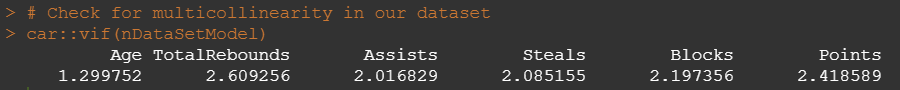


Describe what steps you did to preprocess your data

Initially, the data was joined from two different tables (the salary data that was scraped from [www.hoopshype.com](http://www.hoopshype.com) and the statistics data that was downloaded from [www.basketball-reference.com](http://www.basketball-reference.com)). Many features which seemed to be irrelevant to our business case were removed from the dataset. Player ID's were given to each player's name to uniquely identify them. For the next step, the data had multiple rows with NA or null or bad values. The reason a row would consist only of NA values is if a player was paid that year but did not actually play in any games. For example, a player could be under contract for this year but be injured at the beginning of the season. This player would still have a value under Salary, but they wouldn’t have any values in their statistics. Since there were so few instances of this occurring, our group decided it would be best to remove these records entirely so they didn’t skew the data. We then used a mean value of the salaries of the players to classify them into “High” or “Low” salary categories. After checking for multicollinearity, we found that MinutesPlayed was collinear with Points. Because of this, we omitted this variable from our models.



Once MinutesPlayed was removed, we again checked for multicollinearity and didn’t find any. We then ran our models with the remaining 6 independent variables, age, total rebounds, assists, steals, blocks, and points.



**Neural Network**

The neural network algorithm is used to predict the outcome with similar processes to a real neural network. A neural network has multiple layers of which the bottom layer consists of the predictor variables. These predictor variables are the independent variables using which the dependent variable needs to be predicted. There are then multiple hidden layers that perform complex tasks to predict the final forecast. Each layer provides the input to the layer after that and this process continues till the model reaches the final layer.

Neural network model is used to predict the salaryCategory of a player given that we have all the required independent variable information. We have used three hidden layers and act.fct = "logistic" used for smoothing the result. The accuracy attained by this algorithm is 79.78%.

**K Nearest Neighbor**

This algorithm is mainly used to assign a class to an unlabeled data point based upon the most common class of existing similar data points. In this scenario, the column salary category will be picked up as the label and deleted from the original dataset. And for the rest of the dataset, we split it into training dataset which is used to build the prediction model, and test dataset to test the accuracy of the model. In this model, we choose the square root of the total rows of the training set as the default K-value and find that the accuracy is 77% by building a confusion matrix between the actual test label dataset and our prediction outcome. To figure out the best one, we build a k-value matrix and find the best accuracy is 84% when k-value is 71.

**Decision Tree**

This algorithm has a tree-like structure to identify the relation between the predictor and the required outcome. The decision tree algorithm is made up of multiple nodes and each node requires the model to compare the given data with a variable to make a decision. The result of the comparison helps to navigate to either the left side of the tree or the right. We have used the rpart function to generate the decision tree and in the function the cp value used is 0.001. The final decision tree creates the first decision based on the points earned, then age and so on. The accuracy attained is 73.4%.

**Naïve Bayes**

This algorithm is a supervised model and can be used to classify both binary as well as muti class data based on the probability of events to estimate likelihood that a record belongs to a certain class. Based on the independent input variables, Naïve Bayes assigns a class to the record which has greatest probability. This algorithm doesn’t work well with continuous variables and hence we used the Binning technique on the data to convert it into a set of bins which was done using functions in excel. In this context, we use the algorithm to predict if the salary of a player will fall in the high or low category based on the other independent variables. We first generate the model and calculate probabilities using the model and predict a class for each record.

**Logistic Regression**

A logistic regression model is useful when trying to predict the likelihood of a binary outcome, such as trying to predict how likely a player is to earn a high salary or a low salary. To do this, we start by importing our dataset into a tibble in R. We then randomly take 75% of our dataset to train our logistic regression model.Once our model is trained, we apply it to the 25% of our original dataset that we reserved for testing. We come to find that, using the independent variables our group selected, our logistic regression model is able to predict whether a player will fall into the high salary/low salary category with a 78.7% accuracy.

**Results**

The results from each model are as follows: K-nearest Neighbor – 84%, Logistic Regression – 83%, Neural Networks – 78.78%, Decision Tree – 73.4%, and Naïve Bayes – 73%. The only model we ran that can tell us information about the independent variables is the Logistic Regression model.

Text, calendar

Description automatically generated

To the right, the summary results of our logistic regression show that age, total rebounds, and points are the only significant variables when predicting salary category.

Graphical user interface, text

Description automatically generated with medium confidenceSince the coefficients for our model are in log-odds units, we created odds ratios for each coefficient, because odds ratios are easier to interpret. The log-odds for age is 1.27, indicating that, on average and all else constant, for every year older that a player gets, the odds of that player moving to the “high” salary category increases by a factor of 1.27. The log-odds for total rebounds are 1.33, indicating that, on average and all else constant, as a player gets one more rebound per game the odds of that player moving to the “high” salary category increases by a factor of 1.33. The log-odds for points is 1.25, indicating that, on average and all else constant, as a player scores one more point per game, the odds of that player moving to the “high” salary category increases by a factor of 1.25.

After doing predictive accuracy comparison among the five models that we applied on this scenario, the two best models which have the same predictive accuracy 0.82, are the K-Nearest neighbor and the Neural Networks. However, when thinking about the feasibility, we think that the logistic regression model should also be considered as one of the best options. Because only the logistic model can display the correlated coefficients between every independent variable and the output (Salary category). And the correlation coefficient that each independent variable has on the output will be a good factor for franchises to consider.

And when doing the multicollinearity check in the logistic regression model, we found that MinutesPlayed needs to be removed. After re-doing the analysis with existing independent variables on all the 5 models, we found that the new predictive accuracy of the K-Nearest Neighbor Model is 0.84, which is the highest. In conclusion, the best model for predicting salary category is K –Nearest Neighbor.

**Recommendations**

Based on the results of our models, we recommend using the Logistic Regression model when predicting which salary category, a player will fall in. Additionally, based on the results from our logistic regression, we recommend targeting players who are below average in age, rebounds, and points. This allows our team to find players who excel in other statistics such as assists, steals, and blocks. Players who excel in these categories will be able to help our team while also being likely to be in the lower salary category. While steals, assists, and blocks aren’t as impactful in terms of increasing a player’s salary, they are statistics that can contribute to building a well-rounded team.

**Data Mining Ethics**

Ethics play a significant role in today’s world as data is so easily accessible. It is crucial that the party whose data is being used has consented to the use of their data for a specific purpose. In our case, the data has been scraped from an official public website which had a disclaimer saying that the data could be publicly accessed. Our model is used to predict the salary category for a player based on other statistics so the scope for ethical misconduct is limited. Having said that, it is the responsibility of the party using the data to act as stewards of the data and take full accountability for any backlash or repercussions that could result from the mining or prediction on the data.

**Group Contribution**

Deshmukh, Samiksha (Team Leader): Preprocessed the data with Ben, wrote code for neural network and decision tree models, contributed to writing the paper and making the slide presentation. Her contribution was 25%.

Hickman, Benjamin: Found the dataset, preprocessed the data with Samiksha, wrote code for logistic regression model, contributed to writing the paper and making the slide presentation. His contribution was 25%.

Jacob, Johan: Wrote code for Naive Bayes model, contributed to writing the paper and making the slide presentation. His contribution was 25%.

Xie, Yucheng: Wrote code for k-Nearest Neighbor model, contributed to writing the paper and making the slide presentation. His contribution was 25%.

**Appendix**

**Histogram of continuous variables**

Chart, histogram

Description automatically generated

**Correlation Plot for dataset**

Chart

Description automatically generated with medium confidence

**Data Collection and Preprocessing**

# Set the working directory to the Group Project folder

setwd("~/U of A/Year 2 - Semester 1/MIS 545/Group Project")

# Read csv file of player statistics into player\_info\_stats13\_14 # tibble (this data was not scraped - it was downloaded from # basketball-reference.com as a .csv file)

player\_info\_stats13\_14 <- read\_csv(file = "player\_info\_stats13\_14.csv",

col\_types = "cinnnnnnnnn",

col\_names = TRUE)

# url for salary data

link\_13\_14 = "https://hoopshype.com/salaries/players/2013-2014/"

# Scrape salary data and save into tibble titled 'salary\_13\_14'

salary\_13\_14 <- link\_13\_14 %>%

read\_html() %>%

html\_table() %>%

.[[1]] %>%

setNames(.[1, ]) %>% #Since column names are in 1st row

slice(-1) %>% #Remove 1st row

select(-1) #Remove 1st column

# From salary dataset, change players' names to omit special # characters, remove spaces, and transform all characters to # lowercase.

for (i in (1:nrow(salary\_13\_14))) {

salary\_13\_14[i, 1] = str\_replace\_all(salary\_13\_14[i, 1],

"[[:punct:]]", "")

salary\_13\_14[i, 1] = str\_replace\_all(salary\_13\_14[i, 1], " ", "")

salary\_13\_14[i, 1] = lapply(salary\_13\_14[i, 1], tolower)

}

# Remove '$' and commas in salary data

salary\_13\_14$'2013/14' = gsub("[[:punct:]]", "", salary\_13\_14$'2013/14')

# Convert salary from string to integer

salary\_13\_14$'2013/14' = strtoi(salary\_13\_14$'2013/14')

# Remove the third column (third column is salary figure in current US # dollars adjusted for inflation, which isn't relevant here)

salary\_13\_14 <- select(salary\_13\_14, -c('2013/14(\*)'))

# Rename column name from year to "Salary"

colnames(salary\_13\_14)[colnames(salary\_13\_14) == "2013/14"] <- "Salary"

# Merge player\_info\_stats13\_14 and salary to one tibble called

# nDataSet

nDataSet <- merge(player\_info\_stats13\_14, salary\_13\_14, by =

c("Player"), all.x = TRUE, all.y = TRUE)

# Delete rows that contain NA

nDataSet <- na.omit(nDataSet)

# Convert salary to binary categories (1 if above league average, 0 if # below)

nDataSet$SalaryCategory <- ifelse(nDataSet$Salary >= mean(nDataSet$Salary)

, TRUE, FALSE)

# Remove continuous "Salary" column since we now have "SalaryCategory"

nDataSet <- select(nDataSet, -c(Salary))

**K-Nearest Neighbor**

# install.packages("tidyverse")

library(tidyverse)

library(class)

setwd("/Users/xieyucheng/Desktop/2021FALL/MIS545/FInalproject")

# read the csv file

playerStats <- read\_csv(file = "NDataset.csv",

col\_types = "iinnnnnnl",

col\_names = TRUE)

# display the structure as well as the summary

str(playerStats)

summary(playerStats)

# remove the playerID and MinutesPlayed column

playerStats <- playerStats %>% select(-PlayerID)

playerStats <- playerStats %>% select(-MinutesPlayed)

# what stats do those high-salary players have

playerStatsHighSalary <- playerStats %>%

filter(SalaryCategory == 1)

# the age range of all the NBA player during 2013-14 season

playerCountAge <- playerStats %>%

count(Age)

# the average stats that players have, grouping by salary level 0/1

playerGroupBy <- playerStats %>%

group\_by(SalaryCategory) %>%

summarize(round(mean(Age), 2), round(mean(TotalRebounds), 2), round(mean(Assists), 2),

round(mean(Steals), 2), round(mean(Blocks), 2), round(mean(Points), 2))

# set the salary category as the label which is used to be predicted

playerSalaryLabels <- playerStats %>% select(SalaryCategory)

# remove the label column

playerStats <- playerStats %>% select(-SalaryCategory)

displayAllHistograms <- function(tibbleDataset) {

tibbleDataset %>%

keep(is.numeric) %>%

gather() %>%

ggplot() + geom\_histogram(mapping = aes(x=value, fill=key),

color = "black") +

facet\_wrap(~ key, scales = "free") +

theme\_minimal()

}

displayAllHistograms(playerStats)

set.seed(545)

sampleSet <- sample(nrow(playerStats),

round(nrow(playerStats) \* 0.75),

replace = FALSE)

# use the 75% sampleset to create the training and testing table

playerStatsTraining <- playerStats[sampleSet, ]

playerStatsTesting <- playerStats[-sampleSet, ]

# same as last step that create the training and testing table for # label

playerStatsTrainingLabels <- playerSalaryLabels[sampleSet, ]

playerStatsTestingLabels <- playerSalaryLabels[-sampleSet, ]

# create the knn model

playerStatsPrediction <- knn(train = playerStatsTraining,

test = playerStatsTesting,

cl = playerStatsTrainingLabels$SalaryCategory,

k = 17)

print(playerStatsPrediction)

print(summary(playerStatsPrediction))

# createa a confusion matrix, making up of the original testing and # prediciion result

playerStatsMatrix <- table(playerStatsTestingLabels$SalaryCategory,

playerStatsPrediction)

print(playerStatsMatrix)

# compute the prediction accuracy

predictionAccuracy <- sum(diag(playerStatsMatrix)) /

nrow(playerStatsTesting)

print(predictionAccuracy)

# create a matrix to store k value and corresponding prediction # accuracy

kMatrix <- matrix(data = NA,

nrow = 0,

ncol = 2)

colnames(kMatrix) <- c("k value", "Predictive accuracy")

# create a for loop to compute the accuracy for each k value

for(kValue in 1:nrow(playerStatsTraining)) {

if(kValue %% 2 != 0){

playerStatsPrediction <- knn(train = playerStatsTraining,

test = playerStatsTesting,

cl = playerStatsTrainingLabels$SalaryCategory,

k = kValue)

playerStatsMatrix <- table(playerStatsTestingLabels$SalaryCategory,

playerStatsPrediction)

predictionAccuracy <- sum(diag(playerStatsMatrix)) /

nrow(playerStatsTesting)

kMatrix <- rbind(kMatrix, c(kValue, predictionAccuracy))

}

}

print(kMatrix)

# by creating the k value matrix, we could observe that the best k # value for this prediction model is k=71

KNN Results:

Text

Description automatically generated with medium confidence

KNN Confusion Matrix:

Text

Description automatically generated with low confidence

**Neural Network**

# Install tidyverse package and neuralnet package

# install.packages ("tidyverse")

# install.packages ("neuralnet")

# install.packages("dummies")

# Load the tidyverse, corrplot packages

library (tidyverse)

library(neuralnet)

library (corrplot)

library (olsrr)

# setting folder to Lab4

setwd ("/Users/samiksha/Desktop/Sem1/Data Mining/Project")

# read the contents of csv in a tibble called playerData

playerData <- read\_csv(file = "NDataset.csv",

col\_types = "iinnnnnnl",

col\_names = TRUE)

# print the tibble playerData

print (playerData)

# Display the structure of fishingCharter in the console

str (playerData)

# Display the summary of fishingCharter in the console

summary (playerData)

# craete a correlation plot

round (cor (playerData), 2 )

# removed MinutesPlayed

playerData <- playerData %>% select(-MinutesPlayed)

#scale the data to normalize it

playerData <- playerData %>%

mutate(AgeScaled = (Age - min(Age)) /

(max(Age) - min(Age)),

TotalReboundsScaled = (TotalRebounds - min(TotalRebounds)) /

(max(TotalRebounds) - min(TotalRebounds)),

AssistsScaled = (Assists - min(Assists)) /

(max(Assists) - min(Assists)),

StealsScaled = (Steals - min(Steals)) /

(max(Steals) - min(Steals)),

BlocksScaled = (Blocks - min(Blocks)) /

(max(Blocks) - min(Blocks)),

PointsScaled = (Points - min(Points)) /

(max(Points) - min(Points)))

print (playerData)

# set random seed

set.seed(591)

# Randomly split the dataset into playerDataTraining (75% of records) # and playerDataTesting (25% of records) using 591 as the random seed

sample\_set <- sample (nrow (playerData),

round (nrow (playerData) \* 0.75),

replace = FALSE)

# riceFarmsTesting (25% of records)

playerDataTraining <- playerData[sample\_set, ]

playerDataTesting <- playerData[-sample\_set, ]

# generating the neural network

playerDataNeuralNet <- neuralnet(

formula = SalaryCategory ~ Age + TotalRebounds + Steals

+ Assists + Blocks + Points ,

data = playerDataTraining,

hidden = 3,

act.fct = "logistic",

linear.output = FALSE,

stepmax = 1e10)

# display the numeric result

print (playerDataNeuralNet$result.matrix)

# visualize the neural network

plot(playerDataNeuralNet)

# Use playerDataNeuralNet to generate probabilities on the

# playerDataTesting data set and store this in playerDataProbability

playerDataProbability <- compute(playerDataNeuralNet,

playerDataTesting)

# display the predictions from the testing dataset on consolde

print(playerDataProbability)

# Convert probability predictions into 0/1 predictions and store this # into playerDataPrediction

playerDataPrediction <-

ifelse(playerDataProbability$net.result > 0.5, 1, 0)

# print the predictions

print(playerDataPrediction)

# Evaluate the model by forming a confusion matrix

playerDataConfusionMatrix <- table(playerDataTesting$SalaryCategory,

playerDataPrediction)

# print it on console

print(playerDataConfusionMatrix)

# calculate the model predictive accuracy

predictiveAccuracy <- sum(diag(playerDataConfusionMatrix)) /

nrow(playerDataTesting)

# display the predictive accuracy

print(predictiveAccuracy)

Neural Network Result:

Diagram

Description automatically generated

Neural Network Confusion Matrix

Text

Description automatically generated with medium confidence

Predictive Accuracy:

Graphical user interface, text, application

Description automatically generated

**Decision Tree**

# Install tidyverse package and neuralnet package

# install.packages ("tidyverse")

# install.packages ("neuralnet")

# Load the tidyverse, corrplot packages

library (tidyverse)

library (rpart)

library (rpart.plot)

# setting folder to Lab4

setwd ("/Users/samiksha/Desktop/Sem1/Data Mining/Project")

# read the contents of csv in a tibble called playerData

playerData <- read\_csv(file = "NDataset.csv",

col\_types = "iinnnnnnl",

col\_names = TRUE)

# print the tibble playerData

print (playerData)

# Display the structure of playerData in the console

str (playerData)

# Display the summary of playerData in the console and remove data # with null values.

summary (playerData)

# remove any rows having null data

playerData %>% drop\_na()

# display only relavant information

playerData <- playerData %>% select(-MinutesPlayed)

print(playerData)

# set random seed

set.seed(591)

# Randomly split the dataset into playerDataTraining (75% of records) # and playerDataTesting (25% of records) using 591 as the random seed

sample\_set <- sample (nrow (playerData),

round (nrow (playerData) \* 0.75),

replace = FALSE)

playerDataTraining <- playerData[sample\_set, ]

playerDataTesting <- playerData[-sample\_set, ]

# Generate the decision tree model to predict SalaryCategory based on # the other variables in the dataset. Use 0.01 as the complexity # parameter.

playerDataDecisionTreeModel <- rpart(formula = SalaryCategory ~ Age +

TotalRebounds +

Assists +

Steals +

Blocks +

Points,

method = "class",

cp= 0.001,

data = playerDataTraining)

# Display the decision tree visualization in R

rpart.plot(playerDataDecisionTreeModel)

# Predict classes for each record in the testing dataset and store # them in playerDataPrediction

playerDataPrediction <- predict (playerDataDecisionTreeModel,

playerDataTesting,

type = "class")

# Display playerDataPrediction on the console

print (playerDataPrediction)

# Evaluate the model by forming a confusion matrix

playerDataConfusionMatrix <- table(playerDataTesting$SalaryCategory,

playerDataPrediction)

# Display the confusion matrix on the console

print (playerDataConfusionMatrix)

# Calculate the model predictive accuracy and store it into a variable # called predictiveAccuracy

predictiveAccuracy <- sum(diag(playerDataConfusionMatrix)) /

nrow(playerDataTesting)

# Display the predictive accuracy on the console

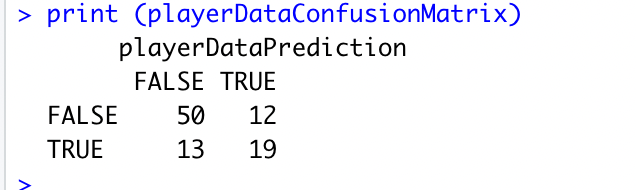
print (predictiveAccuracy)

Decision Tree Plot

Diagram

Description automatically generated

Decision Tree Confusion Matrix:



Predictive Accuracy:

Text, application

Description automatically generated

**Logistic Regression**

# Load the tidyverse, corrplot, and olsrr libraries

library(tidyverse)

library(corrplot)

library(olsrr)

library(smotefamily)

# Set the working directory to the Group Project folder

setwd("~/U of A/Year 2 - Semester 1/MIS 545/Group Project")

# Read NDataSet.csv into a tibble called nDataSet

nDataSet <- read\_csv(file = "NDataSet.csv",

col\_types = "iinnnnnnl",

col\_names = TRUE)

# Display nDataSet in the console

print(nDataSet)

# Display the structure of nDataSet in the console

print(str(nDataSet))

# Display the summary of nDataSet in the console

print(summary(nDataSet))

# Remove PlayerID and MinutesPlayed (We remove MinutesPlayed because, # at the end of our code, we check for multicollinearity, which we # find with MinutesPlayed and Points, so we decide to eliminate # MinutesPlayed)

nDataSet <- select(nDataSet, -c(PlayerID, MinutesPlayed))

# Recreate the displayAllHistograms() function

displayAllHistograms <- function(tibbleDataset) {

tibbleDataset %>%

keep(is.numeric) %>%

gather() %>%

ggplot() + geom\_histogram(mapping= aes(x=value, fill=key),

color= "black") +

facet\_wrap (~ key, scales = "free") +

theme\_minimal ()

}

# Call the displayAllHistograms() function, passing in nDataSet as an

# argument

displayAllHistograms(nDataSet)

# Display a correlation matrix of nDataSet rounded to two decimal # places

round(cor(nDataSet), 2)

# Display a correlation plot using the "number" method and limit # output to the bottom left

corrplot(cor(nDataSet),

method = "number",

type = "lower")

# Set random seed to 198

set.seed(198)

# Create a vector of 75% randomly sampled rows from the dataset

sampleSet <- sample(nrow(nDataSet),

round(nrow(nDataSet) \* 0.75),

replace = FALSE)

# Randomly split the dataset into nDataSetTraining (75% of records)

nDataSetTraining <- nDataSet[sampleSet, ]

# Randomly split the dataset into nDataSetTesting (25% of records)

nDataSetTesting <- nDataSet[-sampleSet, ]

# Check if we have a class imbalance issue in SalaryCategory

summary(nDataSetTraining$SalaryCategory)

# Generate the logistic regression model (using SalaryCategory as the # binary dependent variable) and save it in an object called # nDataSetModel

nDataSetModel <- glm(data = nDataSetTraining,

family = binomial,

formula = SalaryCategory ~ .)

# Display the logistic regression model results using the summary() # function

summary(nDataSetModel)

# Calculate the odds ratios for each of the 3 significant independent # variable coefficients

exp(coef(nDataSetModel)["Age"])

exp(coef(nDataSetModel)["TotalRebounds"])

exp(coef(nDataSetModel)["Points"])

# Use the model to predict outcomes in the testing dataset

nDataSetPrediction <- predict(nDataSetModel,

nDataSetTesting,

type = "response")

# Treat anything below or equal to 0.5 as a 0, anything above 0.5 as a # 1.

nDataSetPrediction <-

ifelse(nDataSetPrediction >= 0.5, 1, 0)

# Generate a confusion matrix of predictions

print(nDataSetConfusionMatrix <- table(nDataSetTesting$SalaryCategory,

nDataSetPrediction))

# Calculate the false positive rate

nDataSetConfusionMatrix[1, 2] /

(nDataSetConfusionMatrix[1, 2] +

nDataSetConfusionMatrix[1, 1])

# Calculate the false negative rate

nDataSetConfusionMatrix[2, 1] /

(nDataSetConfusionMatrix[2, 1] +

nDataSetConfusionMatrix[2, 2])

# Calculate the model prediction accuracy

sum(diag(nDataSetConfusionMatrix)) / nrow(nDataSetTesting)

# Check for multicollinearity in our dataset

car::vif(nDataSetModel)

Logistic Regression Result:



Logistic Regression Confusion Matrix:

Text

Description automatically generated

**Naïve Bayes**

# Write R code to generate a naive bayes model and testing the fit of # the model by calculating the accuracy based on the results of the # confusion matrix

#install the required packages

#install.packages("tidyverse")

#install.packages("e1071")

#load the required libraries

library(tidyverse)

library(e1071)

#Read the csv file and store it into a tibble by specifying the column # types

playerStatistics <- read\_csv(file = "NDataset.csv",

col\_types = "iinnnnnnl",

col\_names = TRUE)

#Print the newly generated tibble onto the console

print(playerStatistics)

#Display the summary and structure of the newly created tibble

summary(playerStatistics)

structure(playerStatistics)

#Drop the NA values from the tibble if any

playerStatistics %>% drop\_na()

#Remove playerId from the tibble

playerStatistics <- playerStatistics %>% select(-PlayerID, -MinutesPlayed)

#Set a random seed to get consistent results on every run

set.seed(456)

#Create a sample set with 75% of the records

sampleSet <- sample(nrow(playerStatistics),

round(nrow(playerStatistics) \* 0.75),

replace = FALSE)

#Create the training and testing datasets

playerStatTraining <- playerStatistics[sampleSet,]

playerStatTesting <- playerStatistics[-sampleSet,]

#Generate the naive bayes model to predict the salary category based # on the other independent variables

playerModel <- naiveBayes(formula = SalaryCategory ~.,

data = playerStatTraining,

laplace = 1)

print(playerModel)

#Calculate the probability of the the salary category being either 0 # or 1

playerStatProbability <- predict(playerModel,playerStatTesting, type = "raw")

print(playerStatProbability)

#Predict the value of 0 or 1 for salary category

playerStatPrediction <- predict(playerModel, playerStatTesting, type = "class")

print(playerStatPrediction)

#Generate a confusion matrix using the testing dataset

playerConfusionMatrix <- table(playerStatTesting$SalaryCategory ,

playerStatPrediction)

print(playerConfusionMatrix)

#Calculate and display the predictive accuracy of the generated model

predictiveAccuracy <- sum(diag(playerConfusionMatrix)) / nrow(playerStatTesting)

print(predictiveAccuracy)

Naïve Bayes Confusion Matrix:

Text

Description automatically generated

Predictive Accuracy:

