New Mexico SuperComputing Challenge

Final Report

April 2021

Project Title: Machine Learning in Sports

Team Number: 22

School Name(s): Justice Code

Team Members: Isaac Rankin & McLight Emma-Asonye

Sponsor(s): Caia Brown

Project Mentor: Wayne Witzel

Problem Statement

Imagine an NBA player who has the potential to become the next Michael Jordan, but he has a coach who doesn't know just how good he is. This player sits the bench the whole season and gets one minute on the court when he has to guard a guy half a foot taller. If the coach had known where to put this player in the lineup to make him the best he can be, the player would show his true potential. This is a situation that could realistically happen and ruin a player's career.

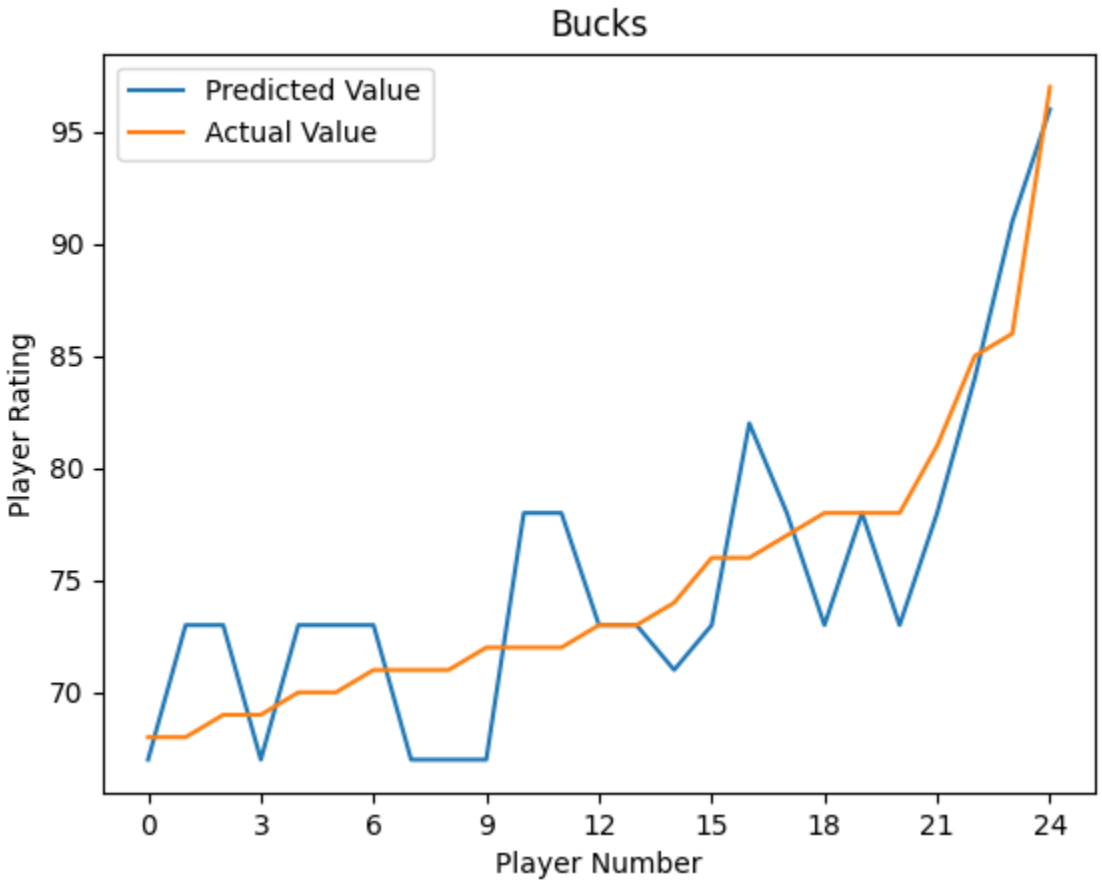
Description Problem Solving Method

Our method to solving this problem is by using machine learning. By using a method that finds patterns in number relations, we can give the computer a set of a player's stats and it will tell us what that player's rating is. We gave the machine 25 different players and their respective NBA 2k ratings, and it found the relationship between their stats and their rating. This would help a coach know how efficient certain players are by looking at their rating.

Discussion of how we Verified and Validated our Model

After running the code, we chose a random player from the displayed list and compared their actual NBA 2k rating to our prediction software rating. This method was too long,

and we weren't able to get the full picture of the accuracy of the machine learning algorithm. We then added a function using matplotlib to graph lines of our prediction and of the actual player rating.



Code to Date (Used Python, 175 lines)

```
# Import the machine learning "classifier" we want to tes
from decimal import Decimal
from tokenize import Double
from sklearn.ensemble import RandomForestClassifier
# Import the numpy package with "np" as an alias.
import numpy as np
import csv

# Create an "classifier" object which will try to "learn"
clf = RandomForestClassifier()

playerRoster = []
rankedRoster = []
```

```
class Player:
    def __init__(self, NAME, TEAM, POS, AGE, GP, MPG, FTA
        self.name = NAME
        self.team = TEAM
        self.position = POS
        self.age = AGE
        self.gamesPlayed = GP
        self.minPerGame = MPG

        self.stats = []
        self.stats.append( float(FTP) )
        self.stats.append( float(TWPP) )
        self.stats.append( float(THPP) )
        self.stats.append( float(PPG) )
        self.stats.append( float(BPG) )
        self.stats.append( float(APG) )
        self.stats.append( float(SPG) )
        self.stats.append( float(BPG) )
        self.stats.append( float(TOPG) )
        self.stats.append( float(VI) )
        self.stats.append( float(ORTG) )
        self.stats.append( float(DRTG) )
```

```
55  # Example players that the computer will take stats from
56  guardExPlayer = ["Stephen Curry", "Kyrie Irving", "Zach L
57  guardExRtg = [96, 91, 88, 84, 78, 73, 67, 82]
58
59  forwardExPlayer = ["Giannis Antetokounmpo", "Jayson Tatum
60  forwardExRtg = [96, 90, 87, 84, 78, 73, 68, 91, 96]
61
62  centerExPlayer = ["Nikola Jokic", "Karl-Anthony Towns", "
63  centerExRtg = [95, 89, 86, 83, 78, 72, 71, 74]
64
65  allPlayerRtg = []
66  allExStats = []
67
68  tempList = []
```

```
70  # Go through all the example players and add their stats
71  # of all the stats the computer will read
72  for row in rows:
73      for index, g in enumerate(guardExPlayer):
74          if g == row[0]:
75              tempList.append(float(row[7]))
76              tempList.append(float(row[9]))
77              tempList.append(float(row[11]))
78              tempList.append(float(row[14]))
79              tempList.append(float(row[15]))
80              tempList.append(float(row[16]))
81              tempList.append(float(row[17]))
82              tempList.append(float(row[18]))
83              tempList.append(float(row[19]))
84              tempList.append(float(row[20]))
85              tempList.append(float(row[21]))
86              tempList.append(float(row[22]))
87
88              allPlayerRtg.append(guardExRtg[index])
89              allExStats.append(tempList)
90
```

```
    for index, f in enumerate(forwardExPlayer):
        if f == row[0]:
            tempList.append(float(row[7]))
            tempList.append(float(row[9]))
            tempList.append(float(row[11]))
            tempList.append(float(row[14]))
            tempList.append(float(row[15]))
            tempList.append(float(row[16]))
            tempList.append(float(row[17]))
            tempList.append(float(row[18]))
            tempList.append(float(row[19]))
            tempList.append(float(row[20]))
            tempList.append(float(row[21]))
            tempList.append(float(row[22]))

            allPlayerRtg.append(forwardExRtg[index])
            allExStats.append(tempList)

```

```
    for index, c in enumerate(centerExPlayer):
        if c == row[0]:
            tempList.append(float(row[7]))
            tempList.append(float(row[9]))
            tempList.append(float(row[11]))
            tempList.append(float(row[14]))
            tempList.append(float(row[15]))
            tempList.append(float(row[16]))
            tempList.append(float(row[17]))
            tempList.append(float(row[18]))
            tempList.append(float(row[19]))
            tempList.append(float(row[20]))
            tempList.append(float(row[21]))
            tempList.append(float(row[22]))

            allPlayerRtg.append(centerExRtg[index])
            allExStats.append(tempList)
    tempList = []

# Let the machine do its magic
clf.fit(allExStats, allPlayerRtg)
```

```
137  # create a player object for every player on the selected team
138  for row in rows:
139      if row[1] == team or row[0] == team:
140          if len(row) == 23:
141              playerRoster.append( Player(row[0], row[1], row[2],
142          if len(row) == 24:
143              playerRoster.append( Player(row[0], row[1], row[2],
144  print('')
145
146  predictions = []
147  actual = []
148
149  # Display all the players and their rating
150  for players in playerRoster:
151      prediction = clf.predict( [players.stats] )[0]
152      print(players.name, prediction, " - Actual Rating:", players
153      predictions.append(prediction)
154      actual.append(players.rating)
155
156  predAndActual = list(zip(predictions, actual))
157  predAndActual = sorted(predAndActual, key=lambda data : data[1])
158
159  for index, smallzip in enumerate(predAndActual):
160      predictions[index] = smallzip[0]
161      actual[index] = smallzip[1]
162
```

```
163  # create and save a plot of the results
164  ax = plt.figure().gca()
165  plt.plot(predictions,label="Predicted Value")
166  plt.plot(actual,label="Actual Value")
167  plt.title(team)
168  plt.xlabel("Player Number")
169  plt.ylabel("Player Rating")
170  ax.xaxis.set_major_locator(MaxNLocator(integer=True))
171  plt.legend()
172
173  plt.show()
174  plt.savefig("predictionsVsActual.png")
175
```

The Conclusions We Reached by Analyzing Our Results

Although we didn't achieve what we expected, our results are still amazing. We were able to give each player in the NBA a rating and have decent accuracy. If we wanted to increase the accuracy of our model, I think we would need more

examples to help the machine learn. In the future, I think we could show the rating for each player in all positions to show the importance of how lineups are set up.

Most Significant Achievement on the Project

“I think the most significant achievement on the project was the predictions, they were a lot closer than expected, and plotting the accuracy was also pretty cool” -Isaac

Acknowledgment of the People and Organizations that Helped Us

Caia Brown
Wayne Witzel
David Cournapeau (scikit-learn inventor)
2k Entertainment