**Predicting high-quality players in FIFA21 with k Nearest Neighbor**

Clayton Catanzarite, [ccatanzarite@bellarmine.edu](mailto:ccatanzarite@bellarmine.edu)

Haziq Zed, [hzed@bellarmine.edu](mailto:student@bellarmine.edu)

**ABSTRACT**

The k Nearest Neighbor is a powerful tool that can be used for analytical prediction that is applied on data sets. Unique data sets can be scrutinized through exploratory analysis, and in this experiment, a FIFA 21 data set was used that contained all the qualities and statistics of the players in the FIFA 21 video game. The data set was not only interesting to use for analysis because of its content, but also because of its diversity in data types from continuous to discrete. This allows for miscellaneous exploration of patterns and relationships to understand the dynamic of the data set. Missing data were found in some columns of the data, and so a normalization process was conducted to prevent any skewed data. Summary statistics and heat map graphs were also produced to depict the unusual nature of parts of the data set, especially those with missing or mismatched data. During the prediction methodology, different parameters of the k Nearest Neighbor model were implemented to deduce an outcome with the greatest prediction scores. The results showed that the parameter with the greatest outcome was using All four (4) normalized features of Ball Control, Stamina, Reactions, and Potential with 80/10/10 split for train, validate, and test. This parameter gave the most consistent prediction, with every score being greater than 0.80. If the experiment were to be carried further, perhaps the use of various combination parameters may have enhance prediction scores even more.

1. **INTRODUCTION**

We are running a machine learning model in this project on a chosen data set of ours. The machine learning algorithm that we decided to use for our project was k Nearest Neighbor. We found a dataset for the game FIFA21 that had a lot of stats and information for 17108 different players. We are using the k Nearest Neighbor algorithm to predict if a player, based on their stats, would be considered a high-quality player. A high-quality player would be a player that has an overall rating greater than 75.

1. **BACKGROUND**
   1. *Data Set Description*

The FIFA21 dataset that we chose to use came from Kaggle at <https://www.kaggle.com/bryanb/fifa-player-stats-database?select=FIFA21_official_data.csv> The user that created this Kaggle dataset used a crawler to scrap the data from [www.sofifa.com](http://www.sofifa.com) . They chose to collect this data because of the rise of FIFA in the esports scene and they thought it's important that the data for the games is easily available. We chose to use this dataset because it has a good number of columns and plenty of samples to run our models with. We also chose this dataset because we both enjoy FIFA and thought it would be an interesting dataset to run the machine learning models on.

* 1. *Machine Learning Model*

We considered a few different ML models for our dataset. We considering k Nearest Neighbor (knn) and Decision Tree as our ML models. We were considering these two algorithms because they were the algorithms that we understood the best. Ultimately, we chose to use k Nearest Neighbor as our ML model. We decided that knn would be the easiest for us to implement and understand the results for in our Jupyter Notebook.

K Nearest Neighbor is a machine learning algorithm that is used to classify labels for new instances based on the distance from the other points. The algorithm starts with a set of data points where we know what the correct label is for the data points already. Before you add a new instance, you set the amount of data points that you want to compare the new instance to. This needs to be an odd number, so it is not an even split vote. For example, you can compare the new instance to 3, 5, or 7 of the closest data points to it. The distance to find these closest data points can either use the Euclidean distance or the Manhattan distance. We will be using the Euclidean distance for our ML model. When you add the new instance, the algorithms compare it to the closest points and figure out what those closest points are classified as. It than takes a vote of what the most common classified label was out of the nearest points and assigns that label to the new point. This algorithm also required us to normalize our data which is talked about more in the Data Preparation section.

1. **EXPLORATORY ANALYSIS**

This dataset contains 17,108 samples with 62 columns with various data types. 44 columns are continuous ratio data (Integer, Decimal), and 17 columns are discrete nominal data (String, Binary). A complete listing of the variable names, data types and proportion of missing data is shown in **Table 1**.

**Table 1: Data Types**

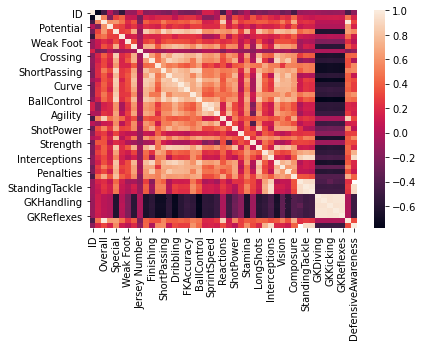
|  |  |  |
| --- | --- | --- |
| *Variable Name* | *Data Type* | *Missing Data (%)* |
| ID | Ratio (Integer) | 0% |
| Name | Nominal (String) | 0% |
| Age | Ratio (Integer) | 0% |
| Nationality | Nominal (String) | 0% |
| Overall | Ratio (Integer) | 0% |
| Potential | Ratio (Integer) | 0% |
| Club | Nominal (String) | 2% |
| Value | Nominal (String) | 0% |
| Wage | Nominal (String) | 0% |
| Special | Ratio (Integer) | 0% |
| Preferred Foot | Nominal (Binary) | 0% |
| International Reputation | Ratio (Decimal) | 0% |
| Weak Foot | Ratio (Decimal) | 0% |
| Skill Moves | Ratio (Decimal) | 0% |
| Work Rate | Nominal (Binary) | 0% |
| Body Type | Nominal (Binary) | 0% |
| Real Face | Nominal (Binary) | 0% |
| Jersey Number | Ratio (Decimal) | 0% |
| Joined | Nominal (String) | 0% |
| Loaned From | Nominal (String) | 0% |
| Contract Valid Until | Nominal (String) | 0% |
| Height | Nominal (String) | 0% |
| Weight | Nominal (String) | 0% |
| Crossing | Ratio (Decimal) | 0% |
| Finishing | Ratio (Decimal) | 0% |
| HeadingAccuracy | Ratio (Decimal) | 0% |
| ShortPassing | Ratio (Decimal) | 0% |
| Volleys | Ratio (Decimal) | 0% |
| Dribbling | Ratio (Decimal) | 0% |
| Curve | Ratio (Decimal) | 0% |
| FKAccuracy | Ratio (Decimal) | 0% |
| LongPassing | Ratio (Decimal) | 0% |
| BallControl | Ratio (Decimal) | 0% |
| Acceleration | Ratio (Decimal) | 0% |
| SprintSpeed | Ratio (Decimal) | 0% |
| Agility | Ratio (Decimal) | 0% |
| Reactions | Ratio (Decimal) | 0% |
| Balance | Ratio (Decimal) | 0% |
| ShotPower | Ratio (Decimal) | 0% |
| Jumping | Ratio (Decimal) | 0% |
| Stamina | Ratio (Decimal) | 0% |
| Strength | Ratio (Decimal) | 0% |
| LongShots | Ratio (Decimal) | 0% |
| Aggression | Ratio (Decimal) | 0% |
| Interceptions | Ratio (Decimal) | 0% |
| Positioning | Ratio (Decimal) | 0% |
| Vision | Ratio (Decimal) | 0% |
| Penalties | Ratio (Decimal) | 0% |
| Composure | Ratio (Decimal) | 2% |
| Marking | Ratio (Decimal) | 94% |
| StandingTackle | Ratio (Decimal) | 0% |
| SlidingTackle | Ratio (Decimal) | 0% |
| GKDiving | Ratio (Decimal) | 0% |
| GKHandling | Ratio (Decimal) | 0% |
| GKKicking | Ratio (Decimal) | 0% |
| GKPositioning | Ratio (Decimal) | 0% |
| GKReflexes | Ratio (Decimal) | 0% |
| Best Position | Nominal (String) | 0% |
| Best Overall Rating | Ratio (Decimal) | 0% |
| Release Clause | Nominal (String) | 0% |
| DefensiveAwareness | Ratio (Decimal) | 6% |

**Table 2: Summary Statistics for Variables with missing data**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **ID** | 17108.0 | 221421.276187 | 36028.786065 | 2.0 | 205451.75 | 230441.0 | 245402.5 | 259105.0 |
| **Composure** | 16821.0 | 59.634088 | 11.912131 | 12.0 | 52.00 | 60.0 | 68.0 | 96.0 |
| **Marking** | 942.0 | 46.403397 | 21.803653 | 4.0 | 25.25 | 48.0 | 66.0 | 94.0 |
| **DefensiveAwareness** | 16166.0 | 47.014475 | 20.152543 | 3.0 | 29.00 | 52.0 | 64.0 | 94.0 |

The summary statistics table shown above (Table 2) includes the three variables in the dataset that exhibited significant missing data. The variables contained missing data that ranged from 2%-94%. The first variable in the table has no missing data but is there as a reference point to see the different in statistics with the variables that did exhibit missing data. Showing the summary statistics for these particular variables is important to point out, in order to see the difference in the count and how some of the statistics can be skewed.

**Correlation Map**



**Figure 1: Heatmap of continuous variables**

The heat map seemed like an appropriate figure to include in the analysis as it would provide a brief insight on the dynamic and the relationships that the different continuous variables in the dataset have. Since most of the continuous variables in the dataset include ratings of player’s abilities, a lot of variables would form significant correlation with one another. This is seen in the correlation map where it shows significant areas of lighter shades, which depict stronger correlations between variables as shown by the legend. However, the map also exhibits a particular pattern of a lack of correlation between variables, which may be due to the nature of these variables and its specificity to certain players. Overall, the heat map is unique and effective in showing the nature of the dataset in terms of its correlation.

1. **METHODS**
   1. *Data Preparation*

We made a couple changes to our parameters as we went through our experiments. The first change that we made to our parameters was that we decided to change a couple of the columns that we used. We decided that stamina, reaction, and potential were better features to include to help predict overall rating. This made a big improvement to all 4 of our classification measures. After that we changed the split from 70/15/15 to 80/10/10. This was also a good change for our model. Lastly, we decided to normalize the data using a Min-Max function that we created. This yielded the best results for our classification measure and used these parameters to test with.

* 1. *Experimental Design*

Table 3: Experiment Parameters

|  |  |
| --- | --- |
| **Experiment Number** | **Parameters** |
| 1 | All four (4) raw features of Ball Control, Shot Power, Finishing, and Sprint Speed with 70/15/15 split for train, validate, and test. |
| 2 | All four (4) raw features of Ball Control, Stamina, Reactions, and Potential with 70/15/15 split for train, validate, and test. |
| 3 | All four (4) raw features of Ball Control, Stamina, Reactions, and Potential with 80/10/10 split for train, validate, and test. |
| 4 | All four (4) normalized features of Ball Control, Stamina, Reactions, and Potential with 80/10/10 split for train, validate, and test. |
| Test | All four (4) normalized features of Ball Control, Stamina, Reactions, and Potential with 80/10/10 split for train, validate, and test. |

* 1. *Tools Used*

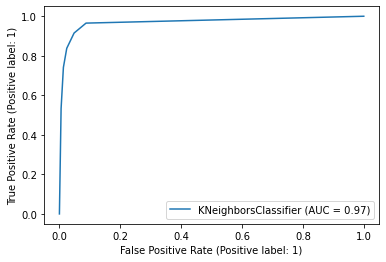
The following tools were used for this analysis: Python v3.8.5 running the Anaconda 4.9.2 environment on Apple Macintosh computer was used for all analysis and implementation. In addition to base Python, the following libraries were also used: Panda 1.1.3, NumPy 1.19.2, Matplotlib 3.3.2, Seaborn 0.11.0, and SKLearn 0.23.2. We chose these tools because they were what we were most familiar with and these libraries are ones we have been using all year for our projects.

1. **RESULTS**
   1. *Classification Measures*

Provide the classification measures for each experiment. For example, you could provide a contingency table for each model to measure how well it classifies data. You could also do an ROC curve (using SciKit Learn).

**Table 4: Contingency Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment** | **Accuracy** | **Precision** | **Recall** | **F1** |
| 1 | 0.9127045985970382 | 0.792332268370607 | 0.6093366093366094 | 0.6888888888888889 |
| 2 | 0.9469992205767732 | 0.8537859007832899 | 0.8034398034398035 | 0.8278481012658229 |
| 3 | 0.9503214494447692 | 0.8775510204081632 | 0.7962962962962963 | 0.8349514563106796 |
| 4 | 0.9491525423728814 | 0.8674698795180723 | 0.8 | 0.8323699421965319 |
| Test | 0.9549970777323202 | 0.8611111111111112 | 0.8378378378378378 | 0.8493150684931507 |



**Figure 2: ROC Curve from KNN Model**

* 1. *Discussion of Results*

Our best model was the last one, Experiment 4, that we used for our test. It provided the best measure for almost all of the classifications. The only classification measure that was a little bit lower was the precision test measure and that was only .006 lower. We think this was the best model because of the features that we used. The first model that we used gave us results that were way below what we were looking for. We changed the features and splits until we had the results that we wanted to get. After we normalized our data, our validation numbers got to where we wanted them to be. Our worst model was our first and that was because we didn’t use the right features to help predict the overall rating. It also could have been lower because we didn’t normalize those features.

* 1. *Problems Encountered*

There were a couple of different problems that we encountered while working on this project. The first problem that we faced was finding a good dataset to use for this project. Neither of us have used a dataset for a machine learning model so we weren’t sure what to look for in our dataset. The next problem that we faced was finding out how to implement our model with our data set. We weren’t very sure how to choose which columns we wanted to use with our model. Our last problem that we faced was figuring out what our results meant. We needed to review what the different classifications measures meant and how we need to improve them.

* 1. *Limitations of Implementation*

Discuss the limitations of your model. Is there is reason it might not be the best way to model the data? What other models might work better?

There were a couple of limitations of our model. One limitation of our model was that it can become computationally expensive relatively quickly. If there is enough data and we have many comparisons and measuring distances, this can slow down the algorithm. Our algorithm overall though worked very well. We believe we chose the correct algorithm for what we were looking for though. Our measurements came out very well when we slowly optimized which features to include and how to split the data.

* 1. *Improvements/Future Work*

What would you like to do to improve your model in future work? Do more experiments, use a different model, add/remove variables, find a different data set, etc?

One way that we can improve our model for future work is to do more experiments with our data. If we did more experiments, it could’ve yielded different results each time and they could've become more accurate in its predictions. Another advantage of doing more experiments is that we could change the variables that we used. We were happy with the results of our experiments, but in the future, we would add maybe more variables or add more comparisons to the knn algorithm.

1. **CONCLUSION**

Finish up with a paragraph or two of summarizing your problem, the results and your conclusions (good model, bad model, needs more work, etc.).

Our problem coming into this project was how we can take this FIFA21 dataset and implement an effective model to help predict if a player is going to be a high-quality player or not. We decided that we can implement each players different stats as the features for the model and use that model to predict the overall skill rating of the player. The most effective model that we found to use was the k Nearest Neighbor model. Using this model, we were able to come up with quality results for our classification measures and our contingency table. Overall, we found this to be a successful model and the correct model to implement with our dataset.

**REFERENCES**

Shah, C. (2020). *A hands-on introduction to data science*. Cambridge, United Kingdom: Cambridge University Press.