

***Faculty of Science and Technology***

**Assignment Coversheet**

|  |  |
| --- | --- |
| **Student ID number &**  **Student Name** | U3243759  Thomas Sharwood |
| **Unit name** | Software Technology |
| **Unit number** | 4483 |
| **Unit Tutor** | Gurpreet Kaur |
| **Assignment name** | ST1 Capstone Project |
| **Due date** | 28/04/2024 |
| **Date submitted** | 28/04/2024 |

**You must keep a photocopy or electronic copy of your assignment.**

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source, and other bibliographic details.

**Date: 28/04/2024**

This report details ST1 Capstone Project. For this assignment I decided to work alone. This was primarily due to me working full time and not being able to attend university in person.

The Dataset I was assigned can be found on Kaggle here: <https://www.kaggle.com/datasets/thedevastator/footballpriceprediction>

It is a massive dataset that contains a large amount of data about almost all active football players. This data was originally sourced from: <https://sofifa.com/> a website that actively tracks football players and official ratings.

As an example, Lionel Messi’s page on SoFifa looks like: A screenshot of a computer

Description automatically generated

The problem I decided to focus on was that despite all this data there was no clear link between the value of a player and everything else. While there may be obvious features which would increase/decrease a player’s value, it is never something simple enough that it can be simply predicted with something like y=mx+b. That’s where SkLearn’s machine learning models come in.

**Stage 1 – Preparing the Data**

The first step when designing this value predictor system was to take a look at what the data actually contained. This dataset is quite large containing over 1.5 million cells containing data.

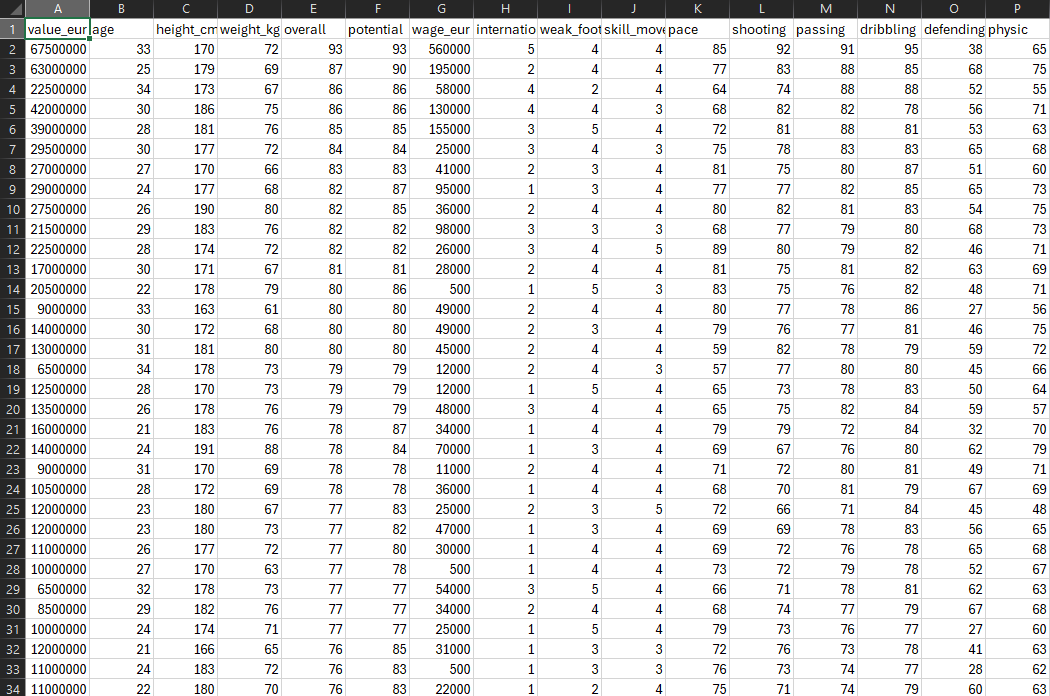
The first thing I did was simplify some of this, many features are simply the product of many other features. Examples of this are “overall” or “goalkeeping”. Because of this I simplified much of the data by removing some of the more specific columns.

Next there was data which was not quantifiable or just straight up irrelevant. This including the SoFifa ID of players, they’re names, jersey number, etc…

Removal of any blank columns was next, there was exactly one which was quickly deleted.

Originally, I had intended to split the data into multiple smaller subcategories by each players team position. I later reverted this however as the amount of data each subcategory had varied so drastically that it become hard to find any meaningful connection using SkLearn models.

After playing around a bit with the dataset using models (There was a lot of back and forth between steps when working on this project, I ended up making dataset changes quite late into development,) I decided the best course of action was to delete the individual position (GoalKeeper) and keep all other positions.

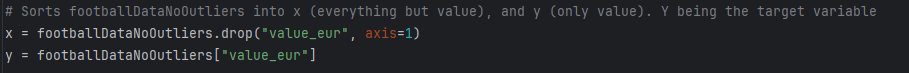


**Stage 2 – Processing the Data**

With the dataset ready to be placed into python the next step was processing it within Python so it would be ready for SkLearn models to use. The first step of this is removing any duplicates, outliers, or rows with any empty cells.

A screen shot of a computer

Description automatically generated

The next step was preprocessing, which I did try to use but it created too many issues way further down the line so I decided on a solution that used the data straight from the Pandas DataFrame. This was quite easy in this situation as all the remaining columns of data only contained integers (besides the top row but pandas reads CSV’s as DataFrames so it gets saved as the column names anyways.) This meant that sorting the data into Y, the target variable, and X, everything else, was as simple as grabbing our making 1 DataFrame that was only the “value\_eur” column and another which was everything but.

We can then print the shape of the DataFrames out at each step to see how much each step is changing the shape:

A black background with white text

Description automatically generated

From this we can see that the Data doesn’t contain any duplicates but when we drop all outliers and rows with empty cells the lose a total of 2522 rows of data for a total of 40352 cells of data.

We have trimmed approximately 13.809% of our total data in just empty cells, (which would mostly be reserve/substitute goalkeepers who have don’t have stats recorded for most features), and outliers.

A screenshot of a computer program

Description automatically generatedFinally with the data ready to be processed we use SkLearn’s Train\_Test\_Split module to create the splits of data the Models will use. We also specify the sample size (in this case 20%), the Seed (so that each time running the program it uses the same samples) and the number of test samples for KFold Validation to use later (in this case 10).

**Stage 3 – EDA**

Technically speaking readying the data for SkLearn application wasn’t necessary for beginning EDA and as such this can be done before creating X and Y.

Now beginning EDA the biggest things to analyze when chopping to create a predictive system is how important each feature is to the target variable and how each variable relates to eachother.

**A graph of blue bars

Description automatically generated with medium confidence**Using Random Forest to sort and graph every feature of X by it’s importance to Y.

Using this graph in order of importance the features are:

1. Overall
2. Potential
3. Age
4. Shooting
5. Defending
6. Pace
7. Physical
8. Wage
9. Passing
10. Dribbling
11. Height
12. Weight
13. Weak Foot
14. Skill Moves
15. Reputation

A group of blue and white graphs

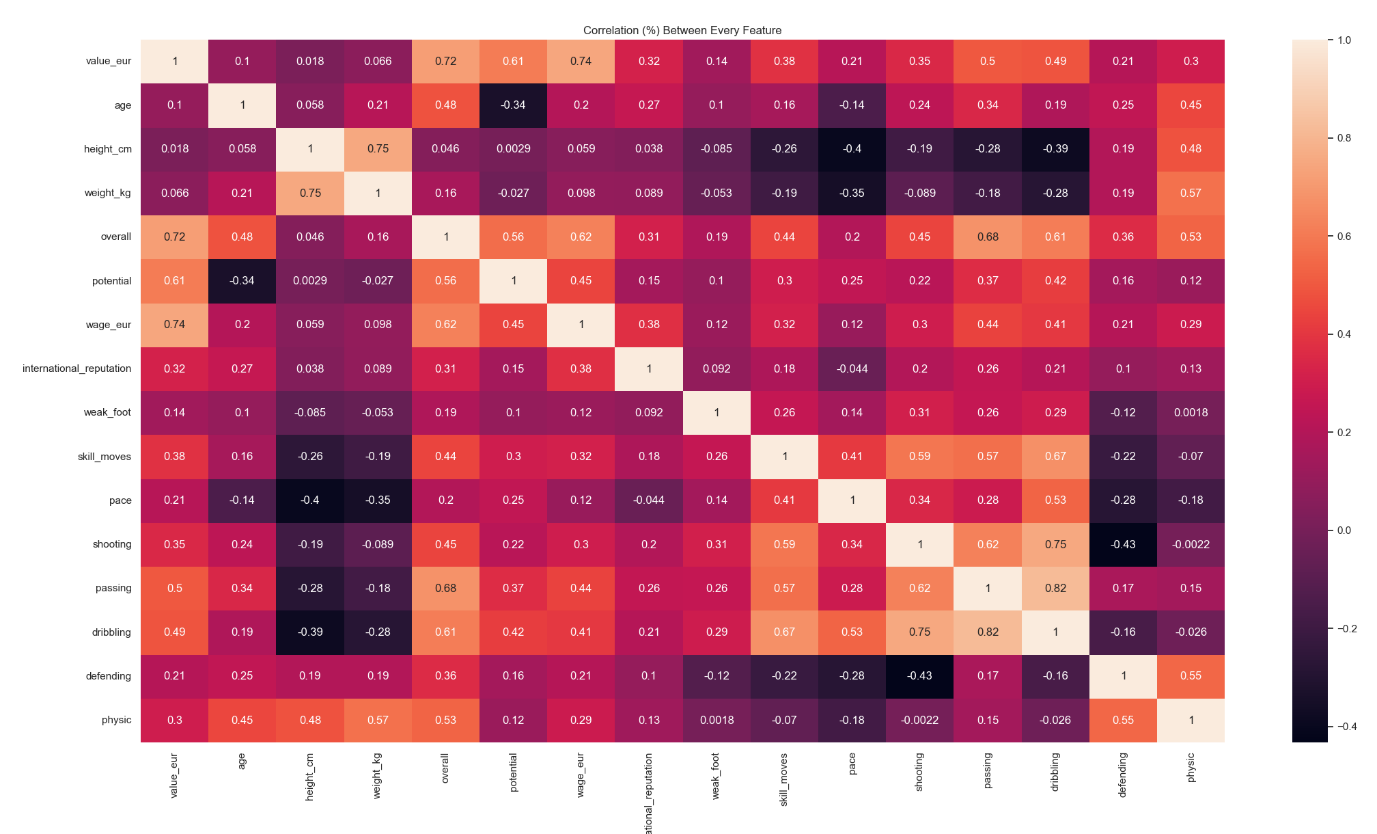
Description automatically generatedNext are the histogram charts which we can use to visualize the distribution of data for each feature individually.

A group of blue and white graphs

Description automatically generated

This second collection of histogram charts is the same but with a larger amount of bins (50 to 100) allowing a better visualization of more heavily skewed features such as value and wage.

These histograms allow an easy way to visualize the distribution and how the distribution is skewed for each feature individually. E.g. value and wage are of right skewed distribution while overall and potential are of normal distribution.

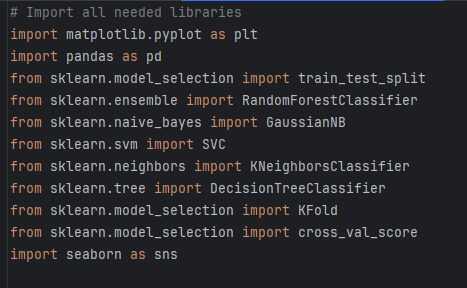
Next is the correlation chart. This chart puts every single feature along both axis and shows what % each correlate.

This allows a deep insight into how some features relate or don’t relate to one another. Some of the stronger correlations are things such as height and weight at 0.75 (75%), or skill moves and shooting at 0.59 (59%).

While some weaker connections would be age and height at only 0.058 (5.8%) therefore having almost no correlation.

There are also negative correlations such as potential and age at -0.34 (-34%) implying that a player’s potential decreases with age.

**Stage 4 – Predictive Data Analysis**

Stage 4 is all about analyzing the models within SkLearn and comparing them.

As seen in the libraries that are imported into this program there are a total of 5 Classification Models that will be compared, them being: Random Forest Classifier, Naïve Bayes GaussianNB, SVM, K Neighbors, and Decision Tree Classifier.

A computer screen shot of a program

Description automatically generatedTo Compare them we create a list containing every model and 1 by 1 run them on the same set of data to compare their accuracy score.

This is also where Test Train Split and KFold from earlier finally come into play.

A screenshot of a computer screen

Description automatically generatedAfter running the code we can see that in this particular case Decision Tree Classifier is the most consistently Accurate.

We can also then plot this data onto a boxplot to visualize the distribution in accuracy for each model:

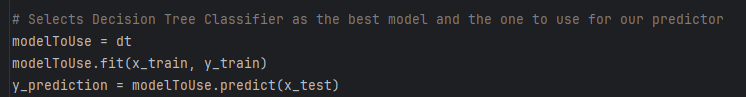
**A white background with black and orange lines

Description automatically generated**

**A white paper with a rectangular object in the middle

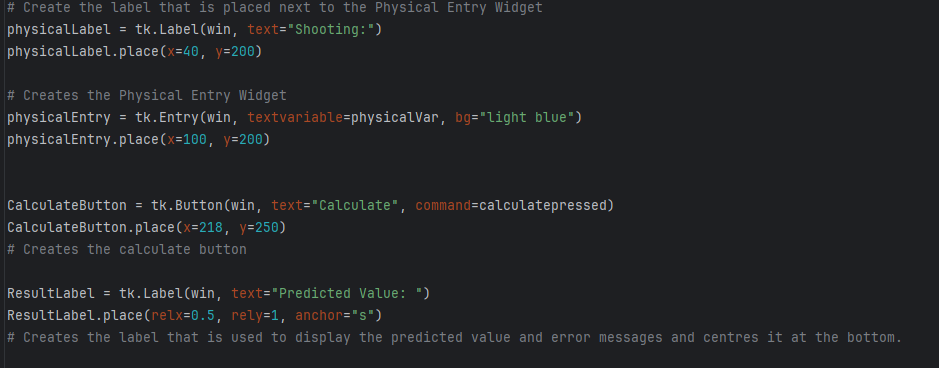
Description automatically generated**And again with only Decision Tree Classifier so as to better visualize it’s distribution.

Now knowing that a Decision Tree Classifier has the best accuracy for our set of data we can set it as the model to use for our implementation / GUI.



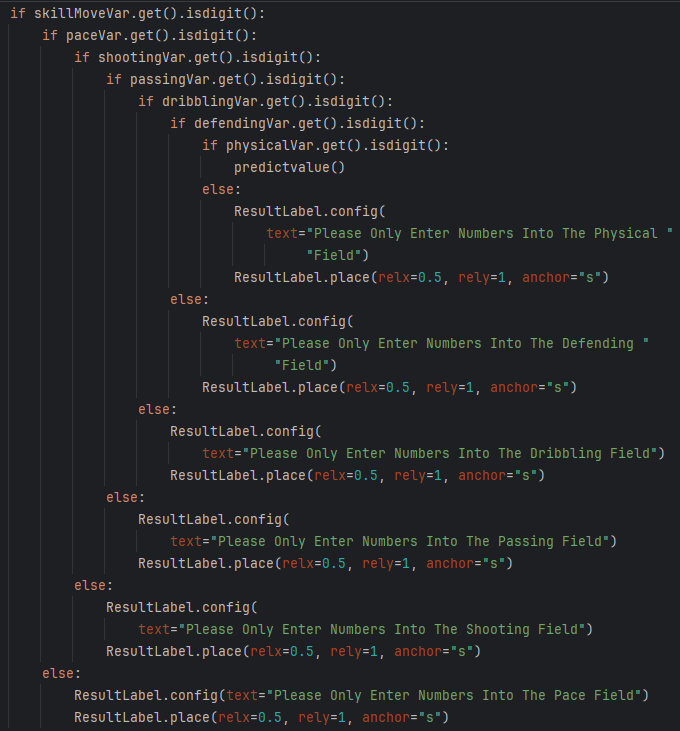
**Stage 5 – Implementation/GUI**

The idea behind implementing a Value predicting GUI is relatively simple using TkInter: Create 15 Entry Widgets for each of the 15 non target variables, Label each of those widgets, accordingly, add a calculate button and a text box to be updated upon pressing the calculate button. Once this is all setup it’s simply a case of feeding the data inside the widgets back to our Decision Tree Classifier and telling it to make a prediction. Despite the relative simplicity of behind it, GUI programming takes ages, so this took quite some time to get right.

The code consists of 14 other labels and entries like the Physical Label and Entry visible above. Below that the Calculate button is visible which runs the calculatepressed function when pressed. And below that is the result label anchored to the bottom of the window which will be used to place text when the calculate button is pressed.

A computer screen shot of a calculator

Description automatically generatedWhenever the Calculate button is pressed it runs a check on every entry widget that 1, it contains a value within it, and 2, that that value is a number. It does this through the use of .isDigit(). We then use a bunch of nested if statements for every single entry widget, if at any time an entry widget doesn’t meet both criteria the Result Label will display a message specifying which box is having the issue.



If every single entry widget meets the criteria it then runs the predictvalue function.

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generatedThe predictvalue function packs every single entry widget variable into one variable (in order) and gives that data to our Decision Tree Classifier which then makes a prediction on the value. This predicted value is then displayed at the bottom of the window.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The above screenshots demonstrate 3 random attempts at comparing to data within the original CSV file.

**Conclusions**

My biggest personal takeaways on the data were the lack of any perfect, hard connections to value. Even the overall skill of a player at best contributed to 72% of the value. At first, I thought this may have been a similar case to other sports such as Formula One, where the value of a competitor isn’t necessarily just their skill but also the fame and marketability that they bring to their teams. However, this is quickly disproved as reputation only corelates 32% to the value of a player.

Football isn’t an exact science and I think this Data analysis proves that, despite there being clear connections between features present there are too many external factors that aren’t tracked also in play.

I still believe the GUI program I developed to be useful as, despite the relatively low accuracy scoring of under 65%, it is very consistent at getting the value of a player right (despite a few far-off predictions).

Given the chance to redo this entire project my biggest thing would be understanding how Pandas DataFrames work early on and therefore being able to build a system that can handle number of columns that were present in the original data.

**Sources:**

H Bonthu, “How to Read and Write With CSV Files in Python?” <https://www.analyticsvidhya.com/blog/>. <https://www.analyticsvidhya.com/blog/2021/08/python-tutorial-working-with-csv-file-for-data-science/>. (accessed April 18, 2024).

“Pandas Read CSV.” <https://www.w3schools.com/>. <https://www.w3schools.com/python/pandas/pandas_csv.asp>. (accessed April 18, 2024).

“Feature importances with a forest of trees.” <https://scikit-learn.org/stable/index.html>. <https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html>. (accessed April 21, 2024).

“sklearn.preprocessing.LabelEncoder.” <https://scikit-learn.org/stable/index.html>. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>. (accessed April 21, 2024).

“Remove outlier from multiple lists in python.” <https://stackoverflow.com/>. <https://stackoverflow.com/questions/73355051/remove-outlier-from-multiple-lists-in-python>. (accessed April 21, 2024).

“Classifier Comparison in Scikit Learn.” <https://www.geeksforgeeks.org/>. <https://www.geeksforgeeks.org/classifier-comparison-in-scikit-learn/>. (accessed April 25, 2024).

Pieren Training, “Understanding Cross Validation in Scikit-Learn with cross\_validate.” <https://pieriantraining.com/>. <https://pieriantraining.com/understanding-cross-validation-in-scikit-learn-with-cross_validate/>. (accessed April 25, 2024)

“sklearn.model\_selection.KFold.” <https://scikit-learn.org/stable/index.html>. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html>. (accessed April 25, 2024).

P Płońsk, “Random Forest Feature Importance Computed in 3 Ways With Python.” <https://mljar.com/>. <https://mljar.com/blog/feature-importance-in-random-forest/>. (accessed April 25, 2024).

J Brownless, “Feature Importance and Feature Selection with XGBoost in Python.” <https://machinelearningmastery.com/>. <https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/>. (accessed April 25, 2024).

J Brownlee, “How To Compare Machine Learning Algorithms in Python with scikit-learn.” <https://machinelearningmastery.com/>. <https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/>. (accessed April 25, 2024).

“1. Supervised Learning.” <https://scikit-learn.org/stable/index.html>. <https://scikit-learn.org/stable/supervised_learning.html>. (accessed April 25, 2024).

“Delete a column from a Pandas DataFrame.” <https://stackoverflow.com/>. <https://stackoverflow.com/questions/13411544/delete-a-column-from-a-pandas-dataframe>. (accessed (April 28, 2024).

“Visualisations with Display Objects.” <https://scikit-learn.org/stable/index.html>. <https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_display_object_visualization.html#sphx-glr-auto-examples-miscellaneous-plot-display-object-visualization-py>. (accessed April 28, 2024).