CCT College Dublin

Assessment Cover Page

Module Title:	Strategic Thinking
Assessment Title:	CA 3 - Capstone Project Final Submission
Lecturer Name:	James Garza
Student Full Name:	Kavi Patak
Student Number:	sba22391
Assessment Due Date:	12 May 2024
Date of Submission:	12 May 2024

Kavi Patak

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Leveraging Machine Learning and Data Science for Competitive Advantage: Estimating Football Player Market Values

by Kavi Patak

sba22391

Strategic Thinking (M1)

Lecturer: James Garza

CCT College, Dublin

May 12, 2024

Table of Contents

Introduction

Organisations have historically implemented enterprise systems such as enterprise resource planning in attempts to gain a competitive advantage (Goundar, 2021). The advent of the internet and proceeding developments in technologies like the Internet of Things(IoT), cloud computing, block chain, big data, Machine Learning (ML) and Artificial Intelligence (AI), have heavily influenced enterprises systems of today, and opened new avenues for conducting business.

Organisations have realised the potential value that resides in data and thus search for ways of utilising this valuable asset, it is here that Business Intelligence (BI) has become an important concept (Agarwal & Dhar 2014, cited in Persson and Sjöö, 2017). BI is an organisation's ability to effectively use the information it collects from daily enterprise (Vidal-García et al., cited in Niu et al., 2021). By identifying emerging opportunities, highlighting potential risks, providing useful insights and supporting decision making, ensure BI plays a significant role in optimising organisational effectiveness (Zhao et al., cited in Niu et al., 2021).

The role of analytics in football has evolved over the past decade, and will continue to do so. Technological developments continue to improve the volume and quality of data available to the world's leading clubs, as well as the ability to derive insight from them. The opportunity exists for clubs of all sizes to use analytics to build a sustainable competitive advantage, something which will be most evident in the area where they invest most: the transfer market.

Having previously assessed the potential for a club to gain a competitive advantage through AI by means of Porter's Five Force Framework (Patak, 2023), this research will focus on leveraging machine learning and data analytics in player scouting and recruitment. More precisely, this report will outline the proposed steps in developing a football player value assessment model using machine learning techniques, and in doing so, aid a football club in making more objective and data driven transfer decisions.

This research project will follow the Cross-Industry Standard Process for Data Mining (CRISP-DM) approach in iterating through the stages of Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation and Deployment.

Problem Domain and Objectives

The first phase of the CRISP-DM lifecycle calls for an understanding of the domain and business objectives, and extracting from this the requirements and goals for this project (Wirth and Hipp, 2000).

Football is a highly lucrative sport that depends heavily on the services of football players, the main suppliers to a football club (Patak, 2023). It is therefore no surprise that the majority of a club's expenditure is on player transfers and wages, with smaller clubs unable to compete financially with their larger counterparts (Metelski, 2021). Furthermore, the consequences of poor scouting and recruitment can be devastating to a club, regardless of size, and can have adverse affects not only from a business perspective (Depken and Globan, 2020), but to squad harmony, on pitch result, and even to a clubs reputation.

This project aims to level the playing field, at least financially in the transfer market, by developing a machine learning model to estimate the market value of football players based on various features such as age, goals, assists and other contributing factors.

Hypothesis: A fair and accurate player market value can be estimated through ML from the available features within the selected dataset.

In addition to this core objective, this project aims to provide insights into promising players who may be undervalued in the market. In-depth analyses shall be conducted on player performance data in specific positions, based on certain physical attributes, as well as exploring the relationship between a players age, value and statistics.
Finally, this project will compare the market value within different domestic leagues and investigate any changes in player market values over time.

Scope and Methodology

Having developed a business understanding of the task at hand, and following the CRISP-DM methodology, the remaining steps in the development lifecycle call for Data Understanding, Data Preparation, Modelling, Evaluation and Deployment.
The CRISP_DM methodology is "Agile" in nature, and unlike the traditional linear Waterfall lifecycle, the sequence of phases is not strict (Wirth and Hipp, 2000). In identifying a viable use case for this capstone project, both the Business Understanding and Data Understanding stages have been initiated as one is intrinsically linked to the other. A brief exploration of the datasets is conducted to confirm that it will satisfy its need.

Continuing on from that and adhering to the CRISP_DM lifecycle, the following processes will be carried out and discussed in this report:

- Consolidating and Characterising the Data,
- Exploratory Data Analysis (EDA),
- Data Preprocessing,
- Feature Engineering,
- Model Selection,
- Training and Validation,
- Defining Evaluation Metrics,
- Tuning Hyperparameters,
- Producing Visualisations,
- Boundaries and Limitations,
- Results
- Model Evaluation,
- Predictive Analytics and Interpretability,
- Deployment,

Consolidating and Characterising the Data

The selected dataset is called "Football Data from Transfermarkt" provided by Kaggle and available at:
https://www.kaggle.com/datasets/davidcariboo/player-scores/versions/284

This dataset consists of nine CSV files with information on football competitions, games, clubs, players and player appearances. Each dataset contains a vast amount of observations (entries or rows) and numerous features (columns, attributes or variables). So too for missing and duplicate values. A more detailed overview of each file's feature names, data types, unique value counts and statistical makeup of numerical features, and missing values can be found in appendix A. A schema of the database design with attributes of the entity and the ID's that are used to join them can also be found in the appendix. EDA and initial preprocessing is performed on individual datasets before merging the required data for further processing and modelling. The

appearances dataset contains valuable performance statistics which needed to be collated along with game and player attributes. Following failed attempts at merging the data, the researcher is fortunate to find a coding solution which is modified for use. The original code is provided by Luis Gasper Cordeiro and found on David Coxon's kaggle webpage (Cordeiro, n.d., in Coxen, n.d.). The consolidated datasets general information can be seen in Figure 1 below.

```
<class 'pandas.core.frame.DataFrame'>
Index: 30130 entries, 0 to 30301
Data columns (total 79 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   player_id                             30130 non-null  int64
 1   first_name                            28173 non-null  object
 2   last_name                             30130 non-null  object
 3   name                                  30130 non-null  object
 4   last_season                           30130 non-null  int64
 5   current_club_id                       30130 non-null  int64
 6   player_code                           30130 non-null  object
 7   country_of_birth                      27528 non-null  object
 8   city_of_birth                         28013 non-null  object
 9   country_of_citizenship                29587 non-null  object
 10  date_of_birth                         30130 non-null  datetime64[ns]
 11  sub_position                          30130 non-null  object
 12  position                              30130 non-null  object
 13  foot                                  27844 non-null  object
 14  height_in_cm                          28099 non-null  float64
 15  market_value_in_eur                   30130 non-null  float64
 16  highest_market_value_in_eur           30130 non-null  float64
 17  contract_expiration_date              18799 non-null  datetime64[ns]
 18  image_url                             30130 non-null  object
 19  url                                   30130 non-null  object
 20  current_club_domestic_competition_id  30130 non-null  object
 21  current_club_name                     30130 non-null  object
 22  age                                   30130 non-null  int32
 23  remaining_contract_days               18799 non-null  float64
 24  age_group                             30130 non-null  object
 25  games_total                           30130 non-null  int64
 26  goals_total                           30130 non-null  int64
 27  assists_total                         30130 non-null  int64
 28  minutes_played_total                  30130 non-null  int64
 29  goals_for_total                       30130 non-null  int64
 30  goals_against_total                   30130 non-null  int64
 31  clean_sheet_total                     30130 non-null  int64
 32  yellow_cards_total                    30130 non-null  int64
 33  red_cards_total                       30130 non-null  int64
 34  games_2019                            30130 non-null  float64
 35  goals_2019                            30130 non-null  float64
 36  assists_2019                          30130 non-null  float64
 37  minutes_played_2019                   30130 non-null  float64
 38  goals_for_2019                        30130 non-null  float64
 39  goals_against_2019                    30130 non-null  float64
 40  clean_sheet_2019                      30130 non-null  float64
 41  yellow_cards_2019                     30130 non-null  float64
 42  red_cards_2019                        30130 non-null  float64
 43  games_2020                            30130 non-null  float64
 44  goals_2020                            30130 non-null  float64
 45  assists_2020                          30130 non-null  float64
 46  minutes_played_2020                   30130 non-null  float64
 47  goals_for_2020                        30130 non-null  float64
 48  goals_against_2020                    30130 non-null  float64
 49  clean_sheet_2020                      30130 non-null  float64
 50  yellow_cards_2020                     30130 non-null  float64
 51  red_cards_2020                        30130 non-null  float64
 52  games_2021                            30130 non-null  float64
 53  goals_2021                            30130 non-null  float64
 54  assists_2021                          30130 non-null  float64
 55  minutes_played_2021                   30130 non-null  float64
 56  goals_for_2021                        30130 non-null  float64
 57  goals_against_2021                    30130 non-null  float64
 58  clean_sheet_2021                      30130 non-null  float64
 59  yellow_cards_2021                     30130 non-null  float64
 60  red_cards_2021                        30130 non-null  float64
 61  games_2022                            30130 non-null  float64
 62  goals_2022                            30130 non-null  int32
 63  assists_2022                          30130 non-null  int32
 64  minutes_played_2022                   30130 non-null  float64
 65  goals_for_2022                        30130 non-null  float64
 66  goals_against_2022                    30130 non-null  float64
 67  clean_sheet_2022                      30130 non-null  int32
 68  yellow_cards_2022                     30130 non-null  int32
 69  red_cards_2022                        30130 non-null  int32
 70  games_2023                            30130 non-null  float64
 71  goals_2023                            30130 non-null  float64
 72  assists_2023                          30130 non-null  float64
 73  minutes_played_2023                   30130 non-null  float64
 74  goals_for_2023                        30130 non-null  float64
 75  goals_against_2023                    30130 non-null  float64
 76  clean_sheet_2023                      30130 non-null  float64
 77  yellow_cards_2023                     30130 non-null  float64
 78  red_cards_2023                        30130 non-null  float64
dtypes: datetime64[ns](2), float64(44), int32(6), int64(12), object(15)
memory usage: 18.7+ MB
```

Figure 1. Merged data general information

Exploratory Data Analysis (EDA)

EDA is a process of examining the available data to discover patterns of correlation, identify trends, and gain insights of interest. It aids in spotting anomalies, testing hypotheses and clarifying any assumptions (Suresh Kumar Mukhiya and Ahmed, 2020). Initial EDA is performed by utilising Pandas Profiling which generates a comprehensive profile report on the data. The report consists of an overview of the datasets statistics, missing values, duplicates, attributes, as well as the correlation between variables. It is a quick and easy way to assess the data, especially with large datasets. This process aids in identifying outliers within the players dataset, with two player height values considerably lower than the mean for that attribute. These values were thus replaced with the correct values sourced from the transfermarkt website. On this second iteration, and on the consolidated data, we analyse the frequency distribution of 'market_value_in_eur', the target variable more closely. Plotting the distribution using a Seaborn histogram (Figure 2) we find the data to be right skewed. Three transformations in Box Cox, square root and log are applied in attempts to achieve a Gaussian distribution, with the latter being the most successful (Figure 2). Univariate analysis continues by investigating the distribution of all numerical variables. Apart from a player's height, they too do not follow a normal distribution. Boxplots are created which highlight the existence of outliers, while also displaying the variance and spread of quantitative variables (Figure 2). The findings of this additional analysis proves invaluable in selecting the most suitable preprocessing methods for optimal ML modelling results. With the ultimate objective of evaluating a players market value, thorough EDA using numerous graphs, including scatterplots, barcharts, histograms, pie charts and boxplots are leveraged in plotting different categorical variables against player market value. The transfer market itself is first analysed for changes in market values over time and to establish the players market distribution by position, footedness and age group.

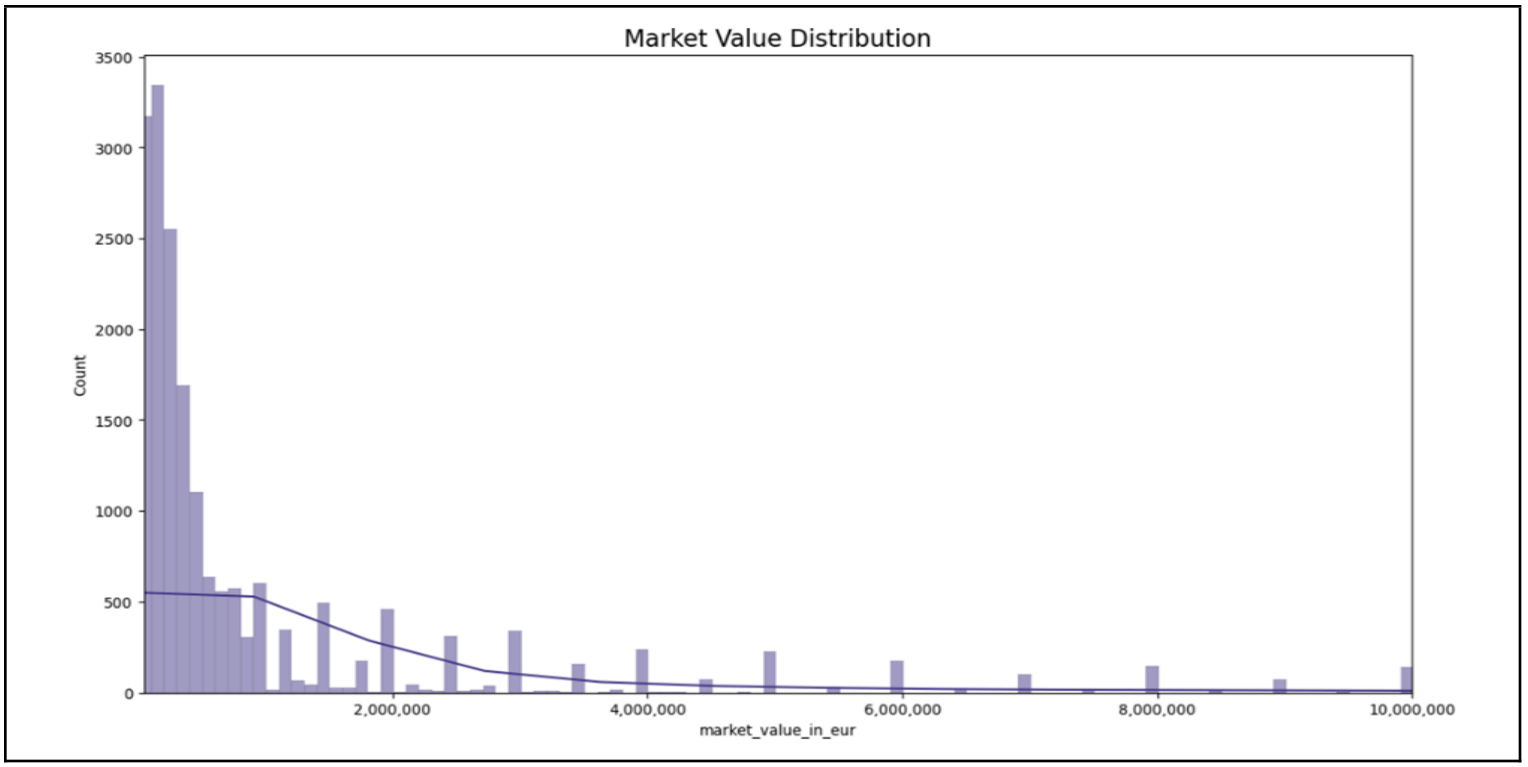


Figure 1. Frequency distribution of Market Value (target variable)

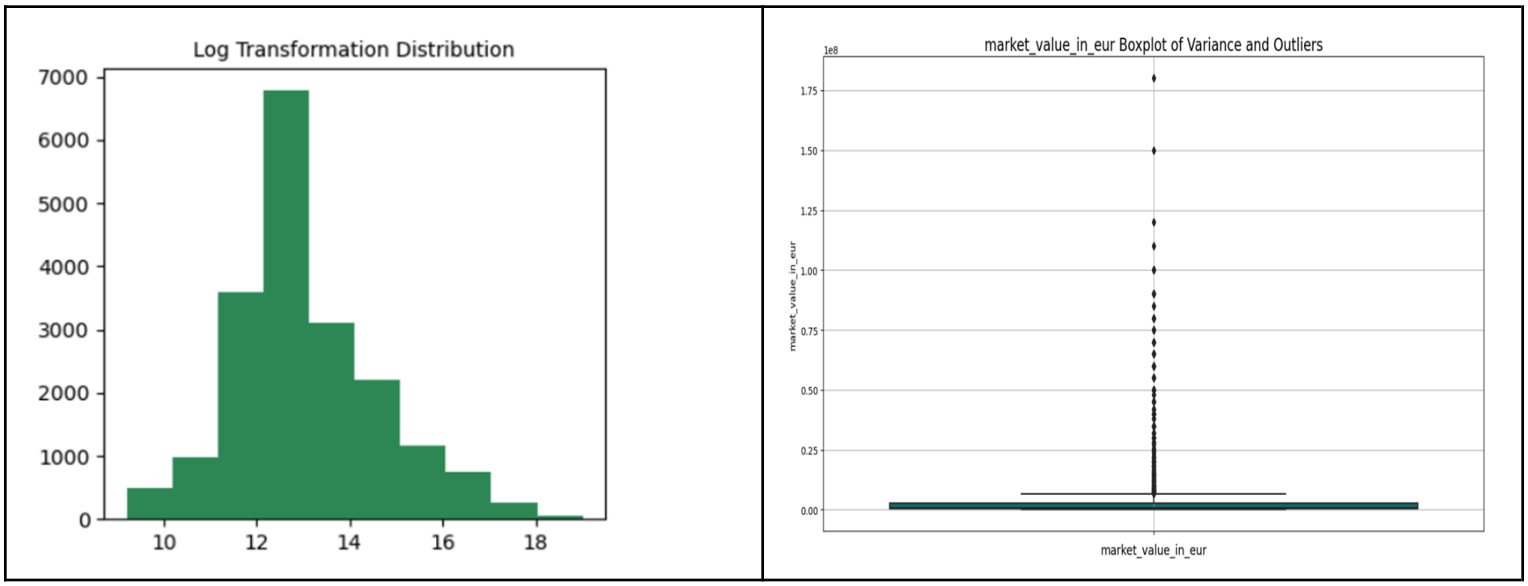


Figure 2. Log Transformed distribution of Market Value (left) abd Box plot of variance and outliers (right)

Player Market Values Over Time

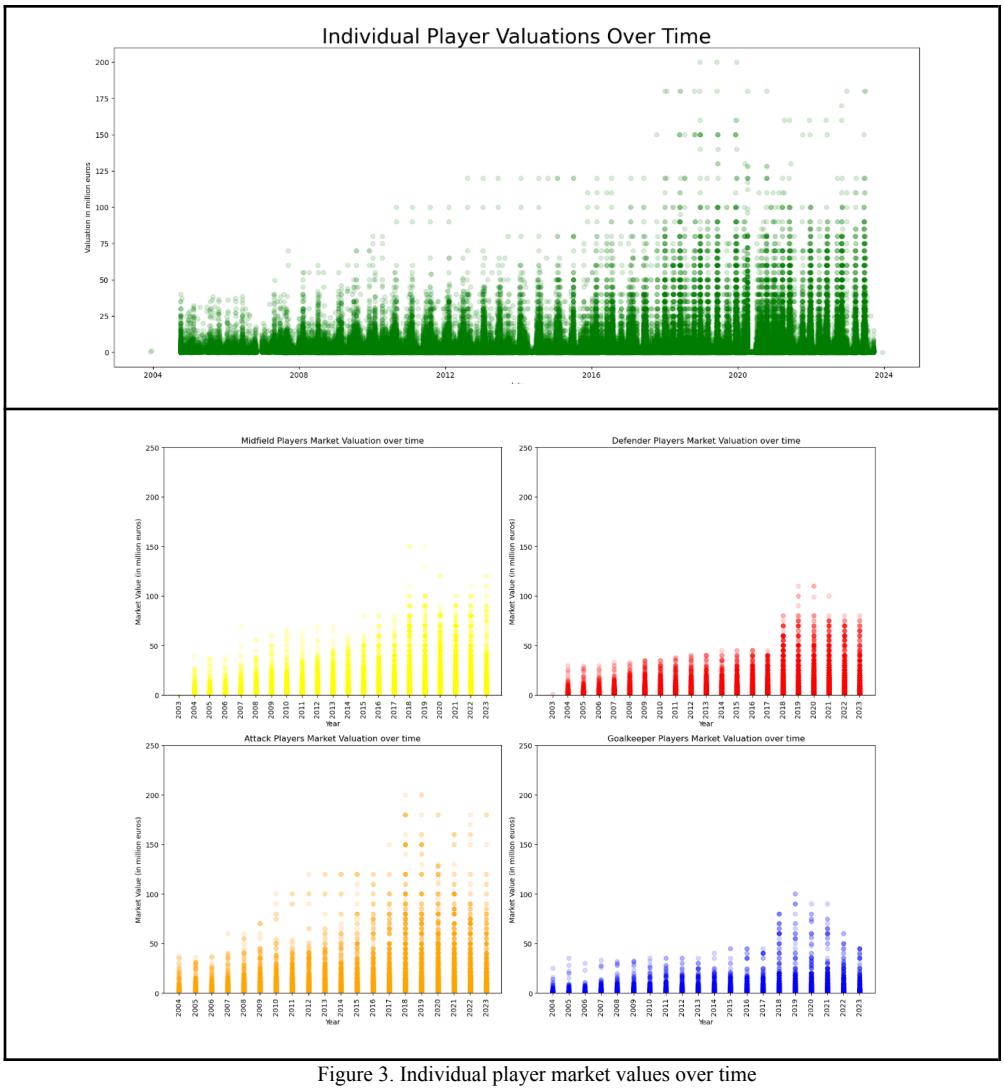


Figure 3. Individual player market values over time

For the individual player transfer market values above (Figure 3), we see a strong positive relationship between the market value and year from 2004 to 2019 with a steady increase until falling off in the last three years.

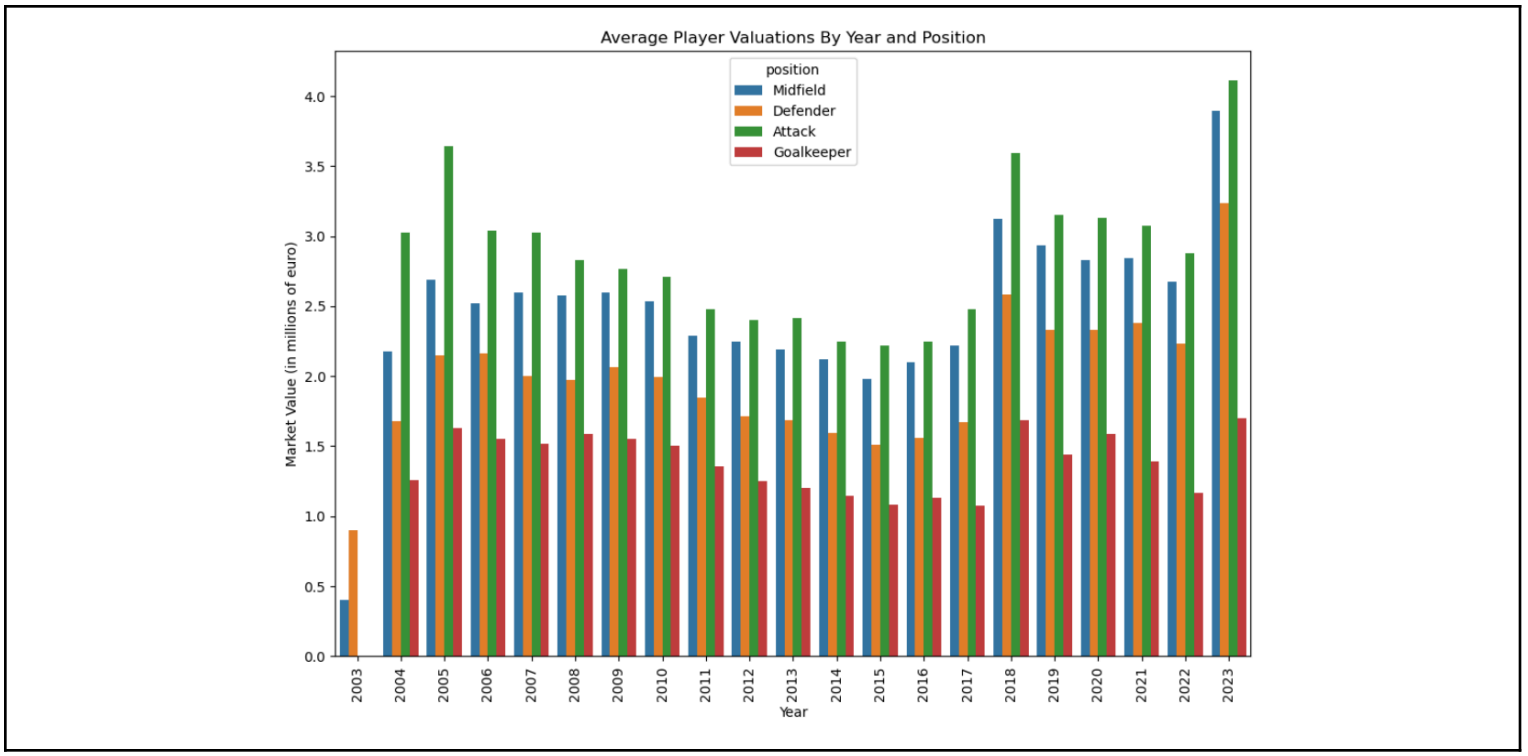


Figure 4. Averaged player market value changes over time

The averaged player market values over the same time is nonlinear with peaks in 2005, 2018 and 2023 and lows in 2003 and 2015 (Figure 4).

Current Player Market Distribution

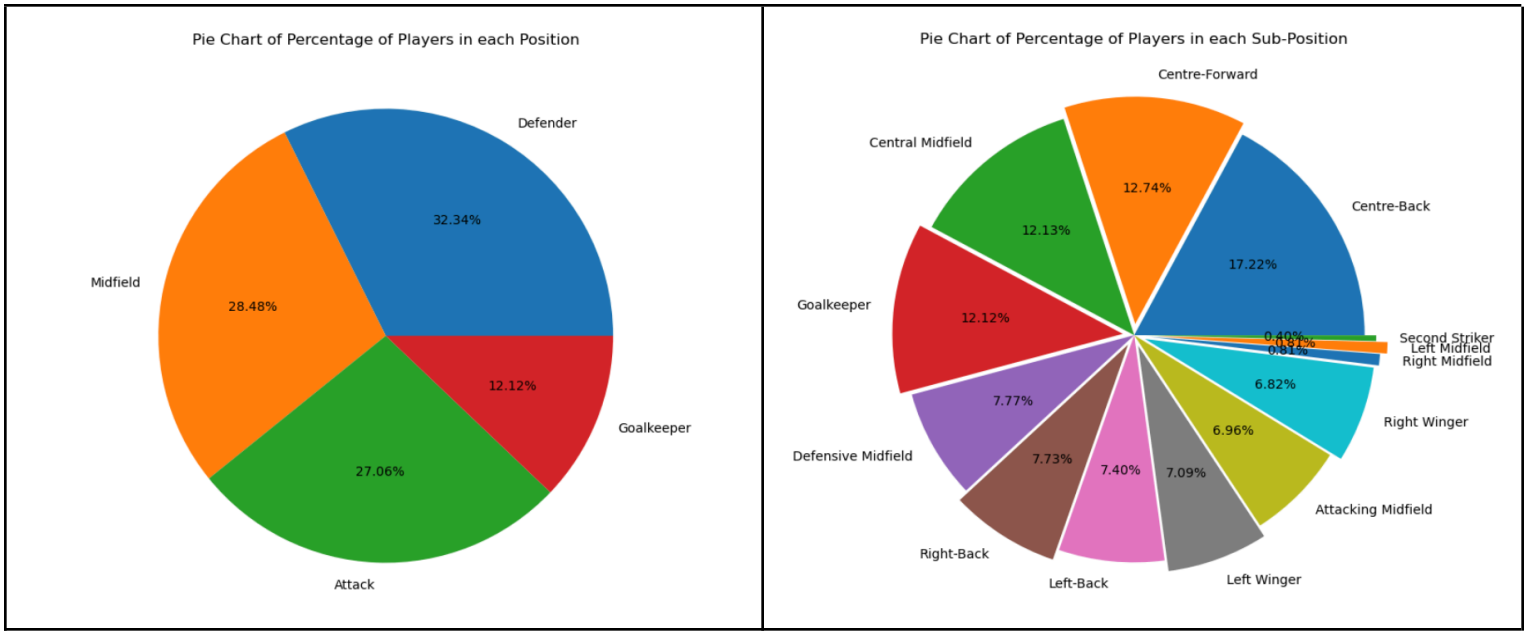


Figure 5. Current market density by position (left) and sub position (right)

The pie chart above (Figure5 left) illustrates that the market of currently playing players is evenly distributed between Attack, Midfield and Defence. There are fewer players in the goalkeeper position which is to be expected as only one goalkeeper is ever selected in a starting eleven.

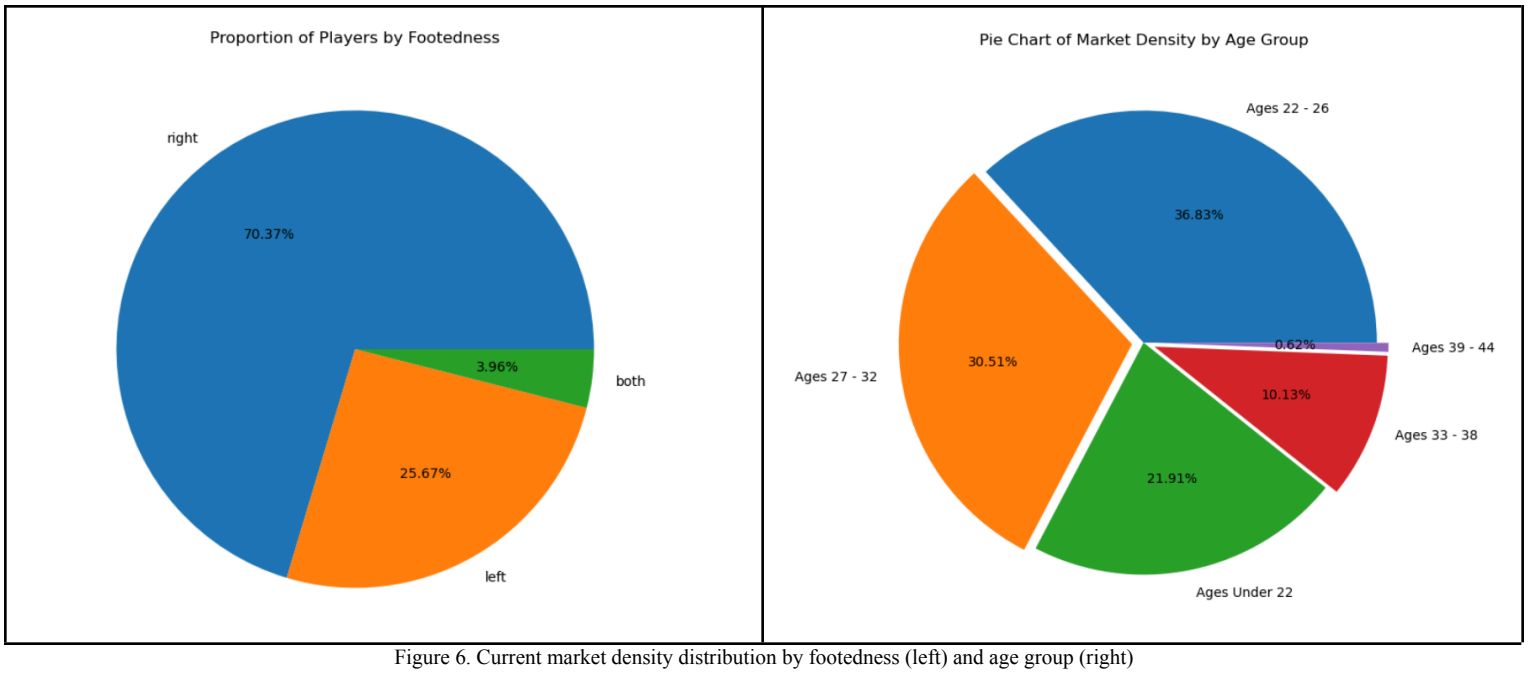


Figure 6. Current market density distribution by footedness (left) and age group (right)

We gather from the pie chart above left that the large majority of players are right footed. Only 25.67% are left footed which, along with the 3.96% for both footed players, may contribute to a higher market value due to scarcity.
As one would expect, approximately 67% of the player market is of players aged between 22 and 32 which is generally considered a players prime years. 21.91% consists of younger, up and coming players, with roughly 10% of players in the twilight of their professional careers.

Player Market Values by Domestic Competition

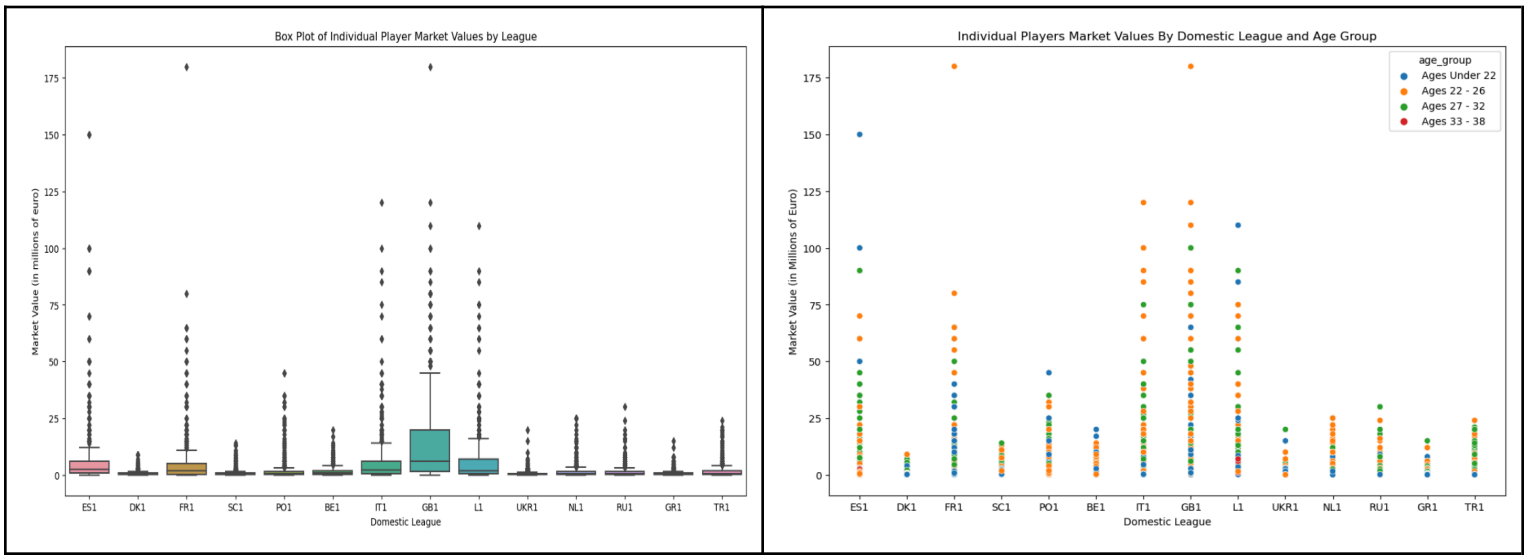


Figure 7. Boxplot of individual market values by domestic competition on the left and scatterplot of market value by age group on the right

The highest valued players currently play in LaLiga (Spain - ES1), Ligue 1 (France - FR1), Serie A (Italy - IT1), Premier League (England - GB1) and Bundesliga (Germany - L1). All five leagues also contain players whose market values far exceed the average within those leagues. This box plot also highlights the variance and outliers with regards to market value within each league (Figure 7 left). From the scatterplot on the top right we see that the highest valued players are generally under 27 years of age across all domestic competitions (Figure 7 right).

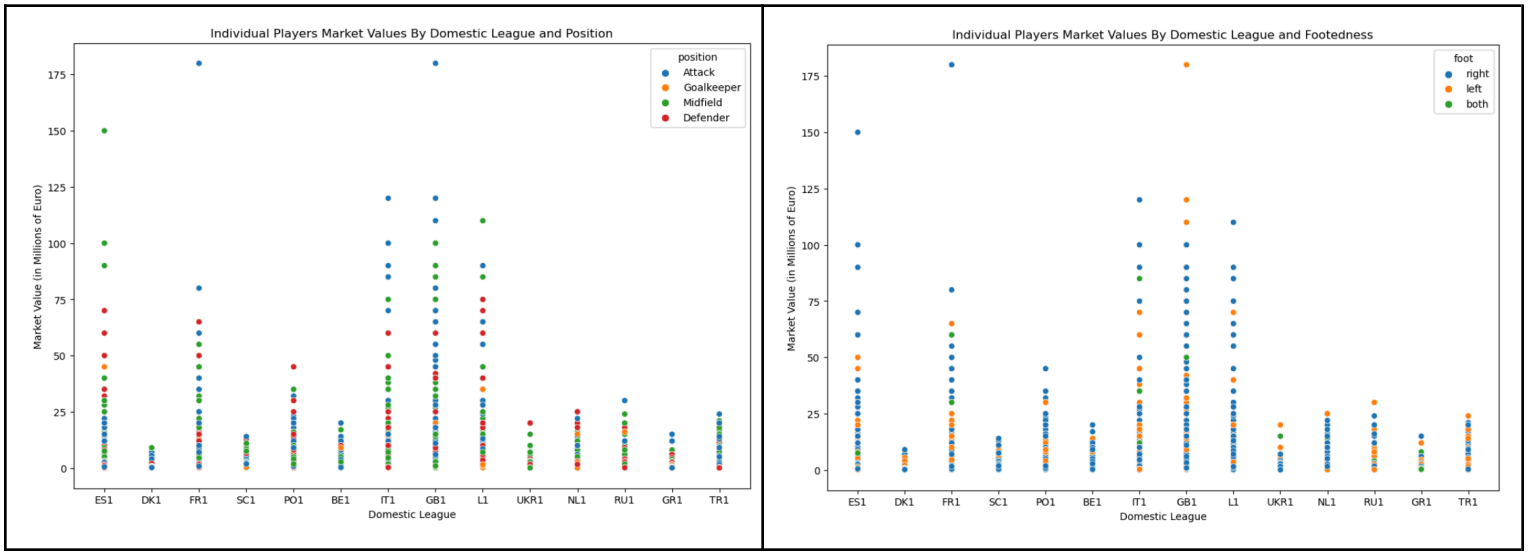


Figure 1.8.Individual player market values for each domestic competition

Attackers and Midfielders are valued the highest with Goalkeepers generally valued the lowest. Left footed players are highly valued relative to their market density. Three of the top seven highest valued players are left footed. Additionally they all apply their trade in the English Premier League (GB1) (Figure 8 right).

Player Market Values by Position

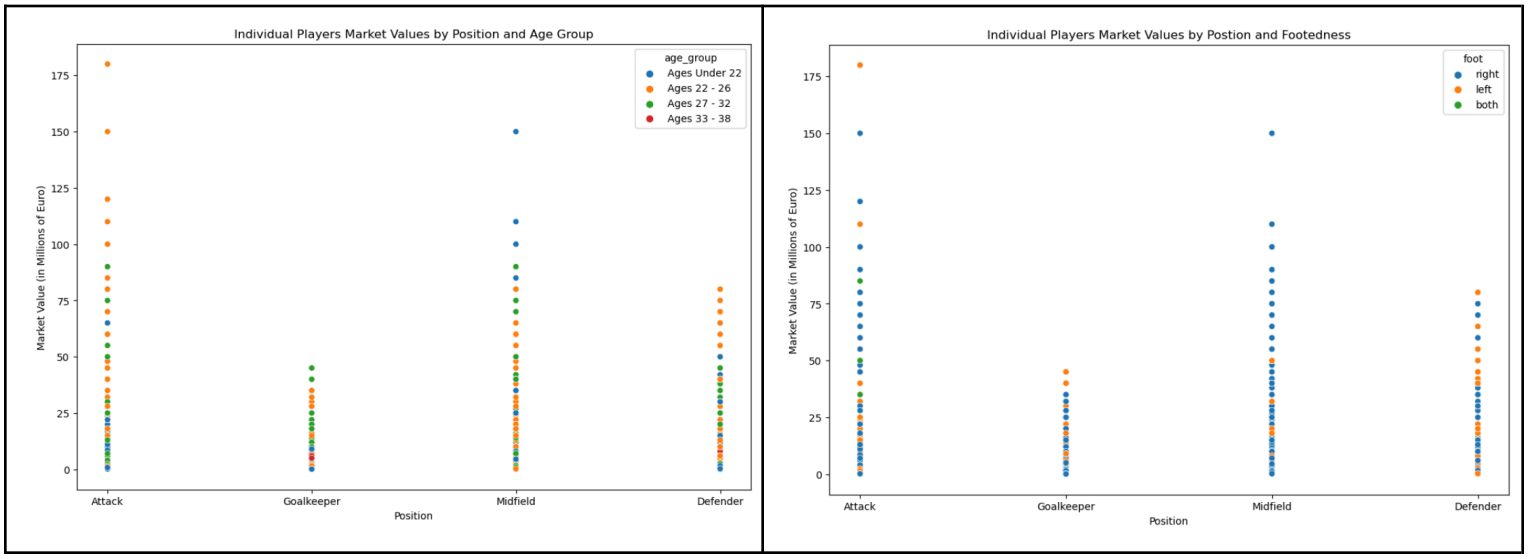


Figure 9. Individual player market values by position

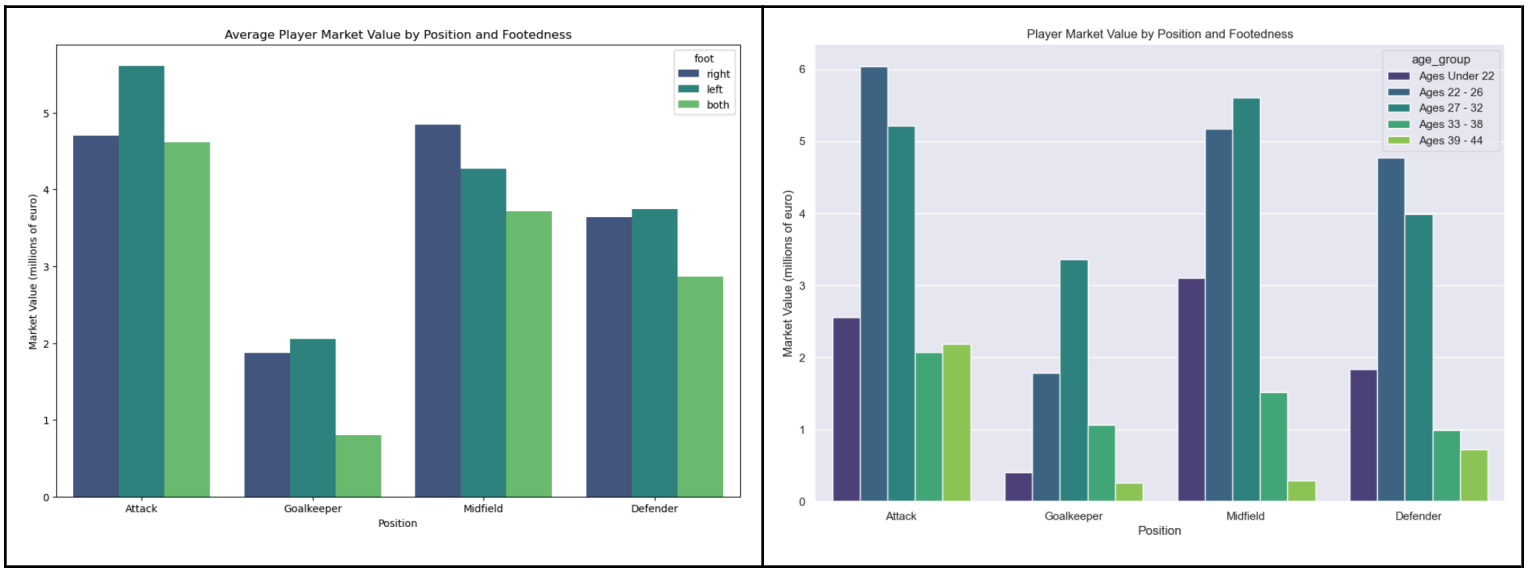


Figure 10. Averaged player market values, by position and foot (left) and position and age group (right).

Left footed attackers seem to be valued the highest on average (Figure 10 left). This is little surprise following earlier analysis of current players and their footing which showed that only 25.67% of current players are left footed. There is clearly a shortage and demand for such players. The highest valued age category appears to be between the ages of 22 and 26. However, in the goalkeeping department, age and experience is slightly more valued (Figure 1 right).

Player Market Values by Age

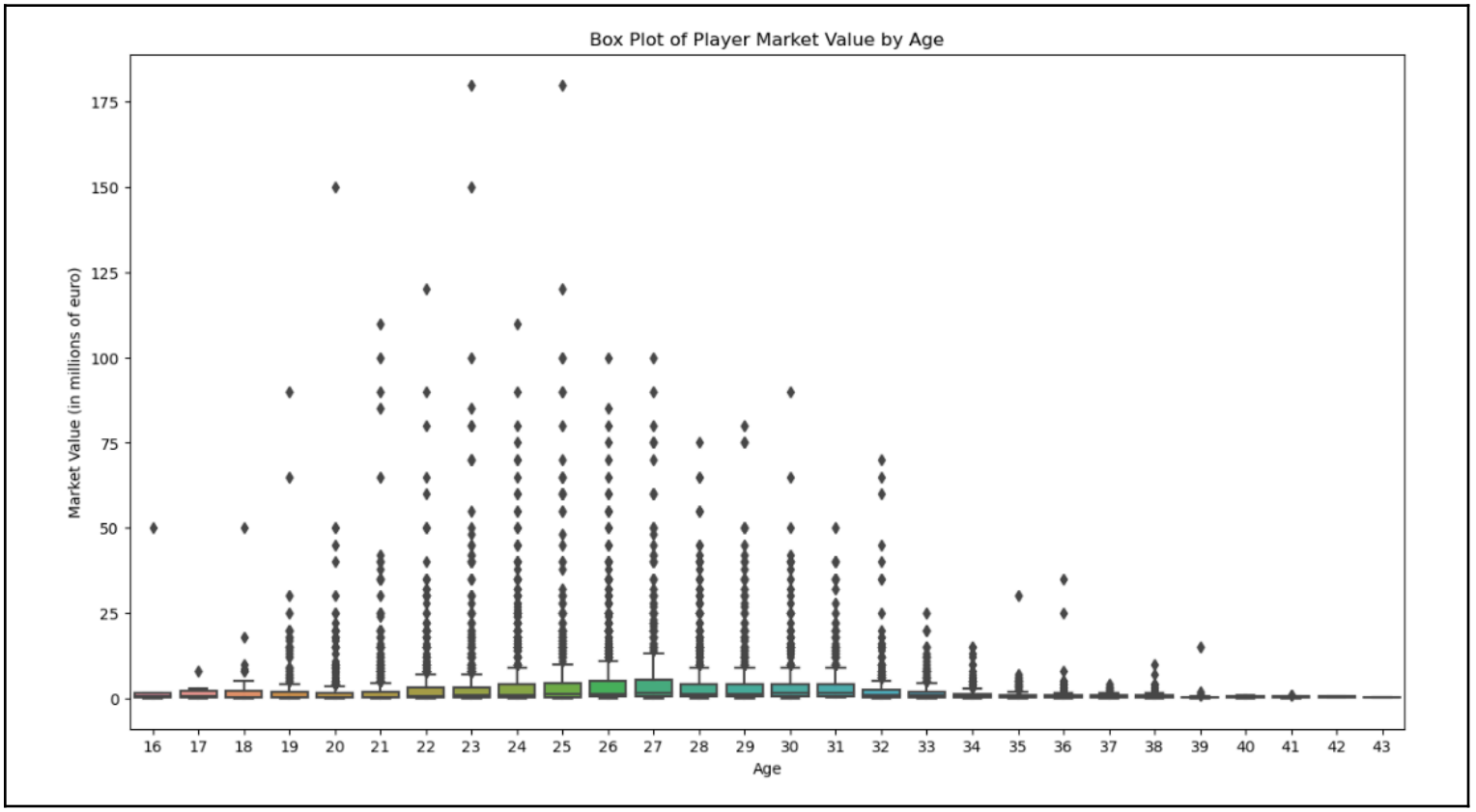


Figure 11. Individual player market values by age

We garner that the highest valued players at around 100 million euro or more are aged between 20 and 27 years (Figure 11).

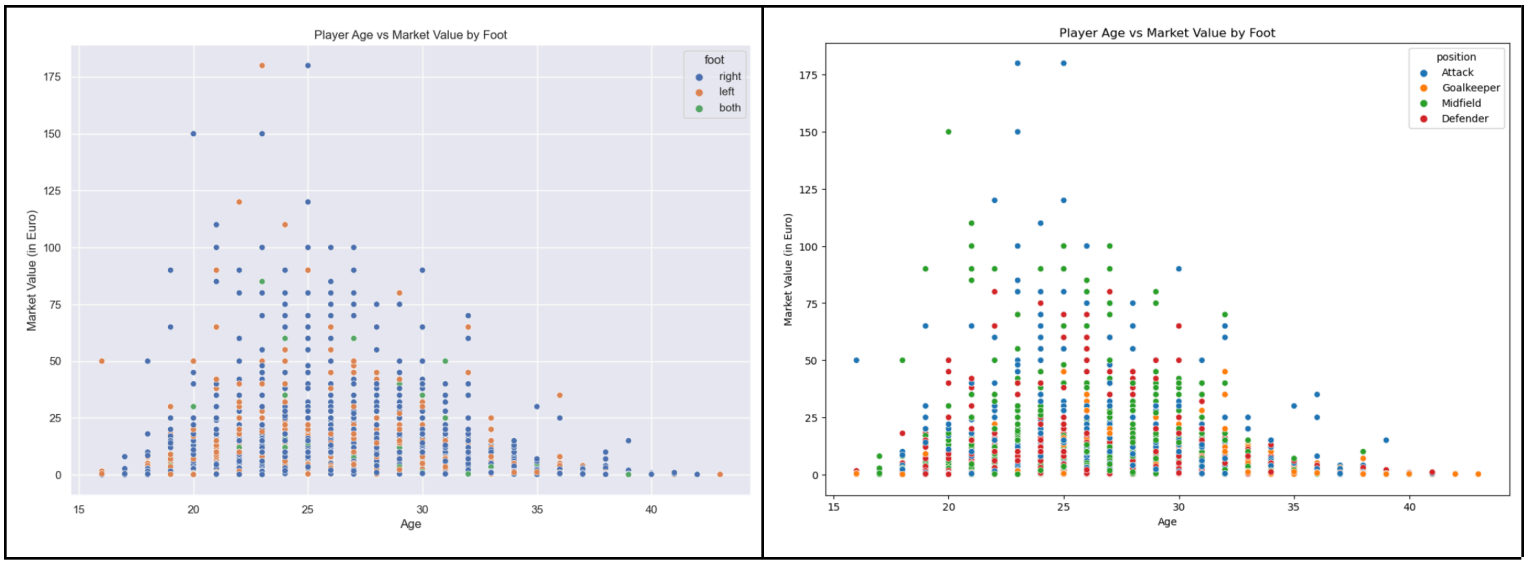


Figure 12. Individual player market values by age, footedness (left) and position (right)

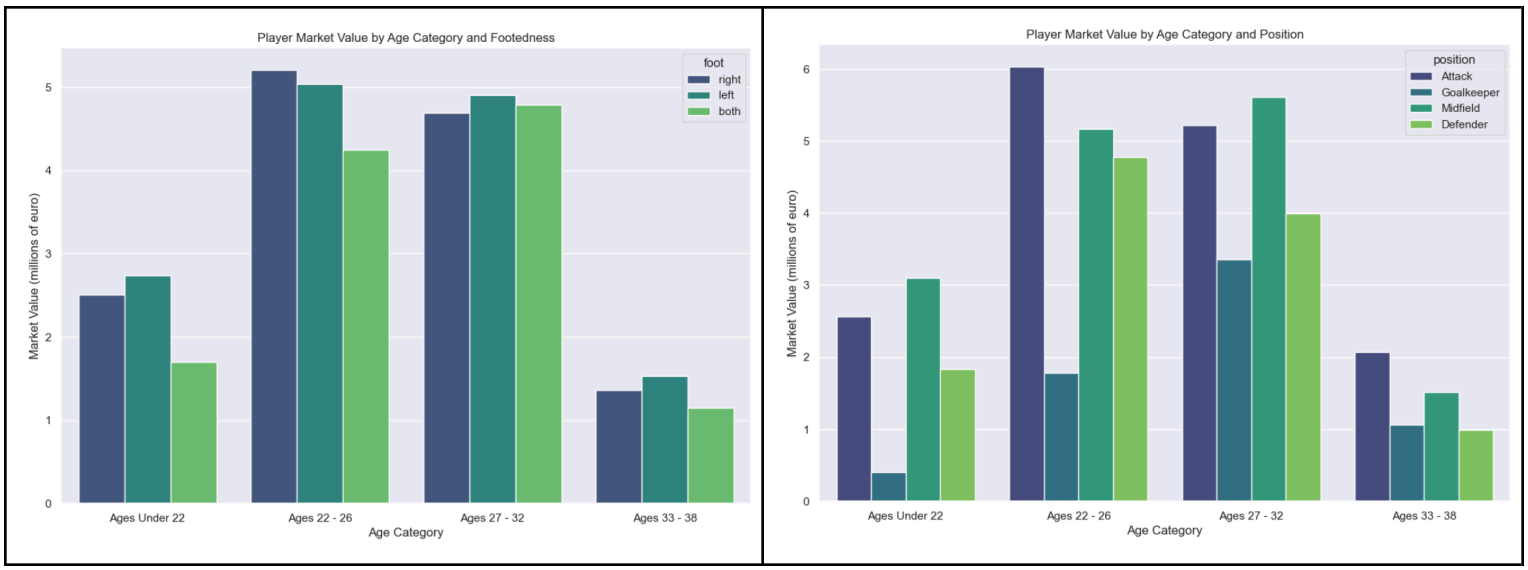


Figure 13. Averaged player market values by age, footedness (left) and position (right)

Across all domestic leagues, players of all footedness average a value between the ages of 22 and 32. The average market value of players is around 5 million or under across all leagues (Figure 13 left). The highest average market value across all leagues is for attackers aged between 22 and 26 with an average value of around 6 million euro. Midfielders are valued higher on average between the ages of 27 and 32 years old (Figure 13 right).

Data Preprocessing

Data preprocessing is a crucial step in any Data Science projects lifecycle. It involves a number of operations and transformations being applied to raw data to make it suitable for analysis and modelling. The aim is to enhance the quality of the data by addressing missing or inconsistent values, handling various data types, including type casting, number formatting and label encoding categorical variables. Transforming the data using scaling, standardisation or normalisation techniques, and conducting feature engineering. Preprocessing is applied in an iterative manner throughout this project. As previously discussed, inconsistencies within the player height attribute are identified and replaced. The player's dataset contains 10919 missing market values. The player valuations dataset does not contain any. Original thoughts were to replace the missing values from the players dataset with those from the player valuations. On further investigation, this proved futile as the players' valuation market values are inconsistent, having been evaluated at different time periods. It was thus decided to impute missing values with the mean. A more refined and robust mean market value was aggregated by grouping players by position and age group, using a custom defined python function of conditional statements to replace the missing values. Improved EDA through univariate and bivariate analysis of variables highlighted the flaws in this process, due to the variance in the data and existence of multiple outliers. Two new imputation methods are applied on this second iteration. The first using the median values which are not influenced by extreme outliers, and the second using a KNNImputer which is a more robust imputation method based on using the mean values of k-nearest neighbours, making it less sensitive to outliers.

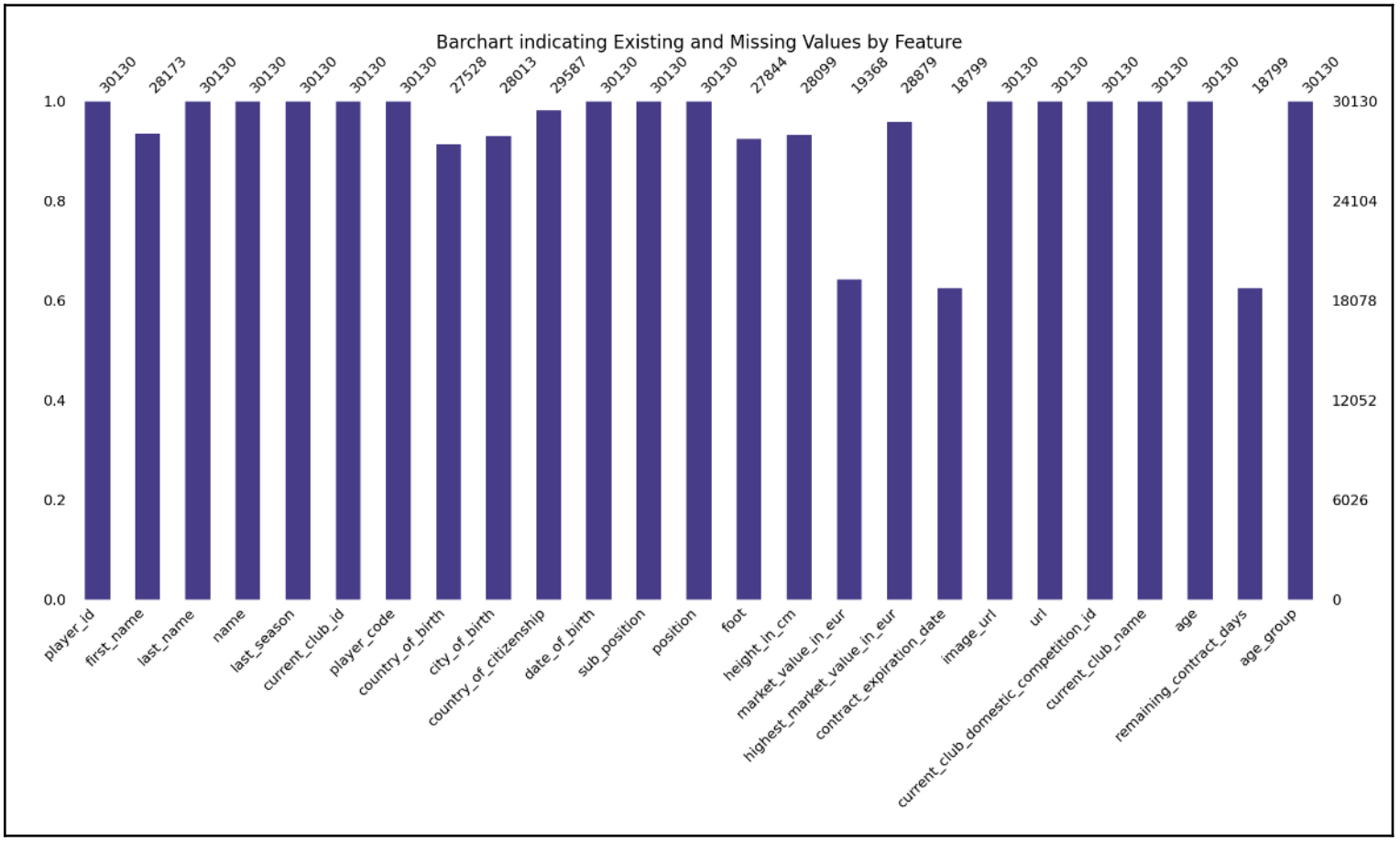


Figure 14. Barchart of missing values for the players dataset

The same logic and steps are followed in imputing missing values for highest market value, player heights and remaining contract days attributes (Appendix B). As the foot feature is categorical, the mode is used to impute missing values based on a player's position and sub-position. EDA highlights the inclusion of retired players within the data, who's attributes, mainly

market values, are inaccurate based on their current age, performance and the current market. Retaining them in the dataset would only introduce noise. They are therefore removed along with any features that would not aid in the Machine Learning (ML) process. Unique identifiers such as player names, player id's, date of births, demographics and unrelated features like images and urls are all removed. Duplicate entries too, are identified and removed. At this stage, the dataset still contains valuable categorical features including, a player position and sub-position, preferred foot, age-group, current club name and competition id. As the order or rank of these categories has no significance, one-hot label encoding is applied to create a new binary feature for each category, and assign a value of 0 or 1 depending on whether the player belongs to that category.
On the first run through, and for comparative purposes, multiple datasets are created based on the features, and the preprocessing applied to them. Following one-hot label encoding, a smaller dataset of a select few categorical features (to avoid the curse of dimensionality) is created, a dataset of all categorical features, and a dataset with features based on Pearson's Correlation. As the data contains outliers, a Robust Scaler is the first choice for scaling as it removes the median, and scales the data according to the Interquartile Range (IQR) (Nalcin, 2022). MinMax Scaler is also applied for comparative purposes, with new datasets created accordingly. It is worth noting that scaling is only applied to nonbinary numerical features so as to retain the value and impact of categorical features on the ML models.
However, not all machine learning models require this processing, with many producing exceptional results on categorical type data. With new insight and knowledge gained on models such as CatBoost Regressor and LightGBM, additional datasets are created on the second iteration, with categorical variables type cast from object data types to categorical. It is also important to mention that preprocessing is only applied following the split of data for training and testing. A step that was originally overlooked. By isolating the training from test data, we reduce the risk of data leakage, thereby allowing for more reliable and reproducible ML model results.

Feature Engineering

Feature engineering is an invaluable process in developing and enriching ML models. It includes feature generation, feature extraction and feature selection. Feature generation entails creating new features from the existing features within the dataset. These new features are engineered to provide additional information that may be relevant and useful to the problem at hand.
A feature for a player's age is created by casting the date of birth feature to datetime and subtracting it from an instantiated object of the current datetime, before dividing by 365.25 for years. A feature for remaining contact days is created in a similar manner, by calculating the difference between the current time, and the contract expiration date feature. A feature to categorise players into age groups is created using a python function of conditional statements.
Further feature engineering includes creating features for goals, assists, yellow and red cards, minutes played, goals for, goals against and clean sheets for each of the last five years, as well as an accumulated total feature for each of these attributes. This was performed whilst merging the appearance, games and players datasets. Finally, features are engineered from the above features for minutes per a goal and minutes per an assist with the resulting NaN (not a number) and inf (infinity) values replaced with 1e400 to represent infinity. Unfortunately, these features are ultimately removed as they are too large for processing.

Feature extraction is the process of reducing the size or dimensionality of a large dataset while feature selection involves choosing a subset of the most relevant features, by removing redundant or irrelevant rows that may introduce noise or lead to overfitting (Rahul Kumar, 2019). To aid in feature selection, statistical testing is applied. Pearson's Correlation coefficients are generated to measure the correlations between independent variables to the target variable, 'market_value_in_eur'. Correlation coefficient values, and heatmap visualisations, can be found in appendix C. It was surprising to find that many of the categorical features for player position and age group do not rank high for correlation to the target variable. Playing in the Premier League (GB1), or for one of the bigger clubs in Europe, does however correlate high to market value. There is no universal correlation coefficient threshold value for retaining features, this is rather determined through domain knowledge and per use case. A refined dataset of the highest correlated feature was created on the first iteration. We found no improvement in model results, and

chose to proceed with all numerical features on this latest iteration. Feature selection to combat overfitting can always be performed at a later stage, if required , additionally, many models accommodate for this, such as Ridge and Lasso regression through regularisation (Anwar, 2021). For feature selection of the categorical attributes, Analysis of Variance (ANOVA), or two

way factorial Analysis of Variance to be precise, is tested (qualtrics, n.d.). ANOVA is an extension of the t-test for independent samples, for more than two groups. It measures the difference in mean between the different categories of the independent variable with respect to the dependent variable (qualtrics, n.d.). The null hypothesis (H0) of ANOVA is that there is no statistical difference among the group means. If a difference exists, it has occurred “by chance”. On the other hand, the alternative hypothesis (Ha) is that at least one group differs significantly from the overall mean of the target variable (Kalyvas, 2024). ANOVA assumes a normal distribution, as such, is tested against the log transformation of the target variable. With all resulting p-values below a 0.05 (5%) significance level, we conclude that they all influence the target variable, and are thus retained for modelling.

Model Selection

Model selection is choosing the appropriate machine learning algorithm based on the objective at hand. The objective is to predict the market value of a player, which is continuous data, making this a regression problem. As we are measuring the relationships between multiple independent variables to the target variable, it is a multiple regression problem (Indeed Career Guide, n.d.). Furthermore, we are using labelled data making this a supervised ML application. Many supervised ML algorithms offer classification and regression variants, with some excelling in one over the other. It is important to note that when developing a ML model, it is extremely unlikely to produce an optimal model on a first attempt. It is about trial and error, and continuous development and improvement. A linear regression model is implemented as a baseline model, followed by Kernel Ridge (KR), with different kernels applied for multiple linear regression on non linear data. Random Forest Regressor (RFR) and Gradient Boosting Regressor (GBR) are also selected, based on positive results found through literature reviews of similar applications (www.kaggle.com, n.d.).

For modelling on the latest datasets, we proceed in a more intuitive and efficient manner. PyCaret is an open-source, low-code, machine learning library in Python, that can automate machine learning workflows (Gitbook.io, 2023). We leverage this tool on all of the new and improved datasets to establish which are the most promising models for further optimisation. As many as 25 baseline models are tested. We find that the CatBoost Regression (CBR), indicated by CBR in Figure 15, returns the most promising and consistent scores overall, for all but two datasets which contain the categorical data type attributes. Although this model, along with LightGBM, excel at handling such data types, they need to be specified when initiating the model. In truth, and in comparison to results from the first iteration, the baseline results below are surprisingly good, across all datasets. LightGBM Regresor requires manual setup, and will be optimised along with CatBoost Regressor.

CatBoost or Categorical Boost in full, is a gradient boosting model that is designed to handle categorical features and even missing values, thereby removing the need for extensive preprocessing. This is done using an innovative method called ordered boosting. CatBoost is an ensemble model of decision trees, with each subsequent tree seeking to eliminate the errors of the previous one (GeeksforGeeks, 2023). LightGBM or Light Gradient Boosting Machine in full, is as the name suggests, another gradient boosting algorithm. It too incorporates tree based learning techniques and introduces the concept of leaf-wise tree growth, which chooses the leaf with the maximum gain to grow the tree. This reduces the memory requirements while enhancing model efficiency. LighGBM leverages two innovative techniques, Gradient-based One Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), to overcome the limitations of traditional models (Technology, 2023).

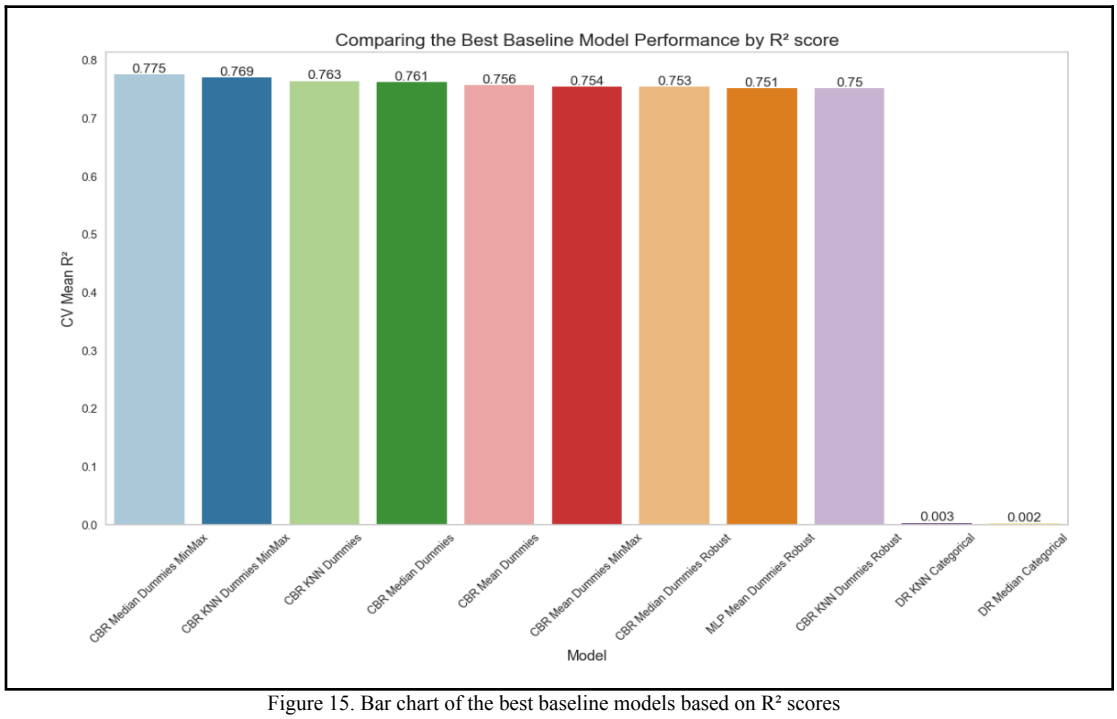


Figure 15. Bar chart of the best baseline models based on R² scores

Training and Validation

This step requires splitting the dataset into a training and testing set. The train set which allows the model to learn, and the test set on which we test how well the model has learned. It is not uncommon for ML practitioners to split the dataset into three subsets, one each for training, validation/development and testing. The development set can be used to assess different models performance, tune parameters and minimise overfitting (Rahul Kumar, 2019). Multiple instances of different training to testing set ratios are measured on the first application, including 70:30, 80:20 and 90:10, while studying the training error, validation error and evaluation metrics. We find the best results occurring on an 80:20, train:test ratio, and as such, apply this split before preprocessing the updated datasets.

In selecting and training a model, it is important to be aware of overfitting, underfitting, bias and variance. Overfitting occurs when you fit a model too precisely to the particularities of the training set, thereby obtaining a model that works well on the training set but is not able to generalise to new data. Essentially, the model is memorising the training data instead of learning the relationships within it (docs.aws.amazon.com, n.d.). This can occur when a model is too complex and learns the noise in the data instead of the underlying pattern. Having a complex model and too many features can also lead to high variance and low bias. Underfitting on the other hand occurs when a model is too simple and does not capture the underlying pattern in the data (Simplilearn.com, n.d.). An underfitting model with too few features will have low variance and high bias. Bias refers to the difference between the expected value of a model's prediction and the actual value. When a model makes test predictions, bias leads it to make inaccurate estimates (Hali, 2019). Variance refers to how much a model is dependent on the training data. It refers to the amount by which a model's prediction may vary for different training sets (Hali, 2019). Finding the right balance between bias and variance is essential in developing a model that generalises well. Generalisation refers to the ability of a model to perform well on new data. This means that the model is not underfitting nor overfitting and can be achieved by balancing the complexity of a model with the amount of training data available. Cross validation is used to estimate the ability of a model to generalise to new data. K-fold cross-validation is the process of splitting the data into k-many folds. A series of training and testing is then applied, holding one split of the data as the test set and the remainder as the training set. The test and training split are alternated k times, each time using a different fold for validation. An average score is calculated based on each fold to determine the overall performance of a given model. Cross validation is more stable and reliable than using a split in training and test set (Müller and Guido, 2017).

Evaluation Metrics

Here we will define metrics to evaluate the performance of each model. For regression problems, Sklean provides Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared (R^2) for more advanced metrics, with R^2 being more intuitive, and the preferred choice in evaluating regression models (Müller and Guido, 2017). Without delving too deep into the mathematics behind each metric, MAE measures the average absolute differences between predicted and actual values. MSE calculates the average of the squared differences between predicted and actual values. RMSE is the square root of MSE while R^2 measures the proportion of variance in the target variable that is predictable from the independent variables. R^2 ranges from 0 to 1 with a score of 1 indicating a perfect fit. For, MAE. MSE and RMSE, lower values indicate better performing models. As models are trained on different dataset, some with scaling or transformations applied, comparing their MAE, MSE, and RMSE becomes difficult. Cross Validated Mean R^2 (CV Mean R^2) will be the determining metric for this project.

Tune Hyperparameters

Hyper parameter tuning is the process of selecting the optimal set of hyperparameters for a machine learning model. This is an important step in the development of a model, as the choice of parameters can have a significant impact on performance. Manual, Random and Grid Search are all utilised in optimising models. For GBR applying the suggested values from Random Search did not contribute to a better model. Hyperparameters such as n_estimators, learning_rate, max_features, max_depth, min_samples_split, min_samples_leaf, n_iter_no_change, subsample and validation_fraction are all adjusted for GBR in attempts to counter overfitting while optimising CV Mean R^2. For KR, alpha, gamma, degree, coef0 and the kernel are all adjusted in optimising the model with positive results. For CBR, hyperparameter tuning through GridSearchCV was unsuccessful overall. Multiple values for depth, iterations, 12_leaf_reg, learning_rate and early_stopping_rounds are specified in attempts to optimise the model and counter overfitting. Although improvements are made, the model performs comparably well with default settings. Parameters tuned for LightGBM include n_estimators, learning_rate, num_leaves and max_depth.

Producing Visualisations

Visualisations provide a clear and intuitive way to present data, making complex patterns, trends, outliers and relationships easier to understand, without the need for attentive processing (McQuaid, n.d.).
Graphics and plots are leveraged throughout the development process, in EDA, feature engineering, modelling and finally to compare and communicate the research findings effectively. Open source libraries like Matplotlib, Seaborn and Yellowbricks are utilised using Python code in an interactive Jupyter Notebook environment, in developing and documenting the project.

Boundaries and Limitations

It is important to note that each club is unique and so to their structure, ambitions and vision. What is right for one club may not be for another. Some aspire to win the league while others to avoid relegation. Other business models may depend solely on selling players for profit (Sloane, cited in Van den Berg, 2011). Hence aligning and framing scouting requirements with the vision of a club is crucial, as is the recognition that talent id is one small part of squad evolution and building. Another challenge may reside in harnessing the data and bringing together different data sources. There are multiple factors that contribute to evaluating a player's value, some of which may not be available within the selected dataset. Aspects such as exchange rates and social or economical factors like war, recession or a worldwide pandemic may prove challenging to account for (Metelski, 2021).

Results

	Name	Model	Data	CV Mean R^2	CV STD	CV MAE	CV MSE	CV RMSE	R^2 training set score	R^2 testing set score
0	Baseline Linear Regression	Linear Regression	Original Mean	0.461554	0.04788	3.742086e+06	4.809032e+13	6.934718e+06	0.487314	0.497589
1	Best Kernal Ridge	Kernal Ridge	Mean Dummies Robust	0.722191	0.02619	8.154943e-01	2.324137e+00	1.524512e+00	0.847303	0.778362
2	Best Random Forest	Random Forest Regressor	Mean Dummies MinMax	0.706437	0.042542	4.005719e-03	1.165881e-04	1.093838e-02	0.959733	0.757246
3	Best Gradient Booster	Gradient Boosting Regressor	Mean Dummies Robust	0.721352	0.042758	9.480687e-03	3.596097e-04	1.896338e-02	0.8758	0.80096
4	catboost_modelA	CatBoost Regressor	Median Dummies MinMax	0.773912	0.042547	4.842887e-03	6.884513e-05	8.297296e-03	0.8758	0.80096
5	lgbmA	LightGBM Regressor	Median Categorical	0.823245	0.045841	7.461111e+05	2.350053e+12	1.532988e+06	0.976574	0.807402
6	lgbmA_log	LightGBM Regressor	Median Categorical	0.851580	0.045841	4.794483e-01	4.051202e-01	6.362758e-01	0.948533	0.821507
7	catboost_median	CatBoost Regressor	Median Categorical	0.791185	0.05395	7.122953e+05	1.599051e+12	1.264536e+06	0.98406	0.805298
8	best_lgbmB	LightGBM Regressor	KNN Categorical	0.811371	Not tested	1.671173e+06	1.749355e+13	4.170307e+06	0.964407	0.846511
9	Extra Trees Baseline	Extra Trees	Median Dummies	0.740200	Not tested	1.828058e+06	2.658267e+13	5.076259e+06	Not tested	Not tested
10	MLP Baseline	Multi Layer Perceptron	Mean Dummies Robust	0.750900	Not tested	1.055300e+00	3.951900e+00	1.980900e+00	Not tested	Not tested

Figure 16. Table of Model Results

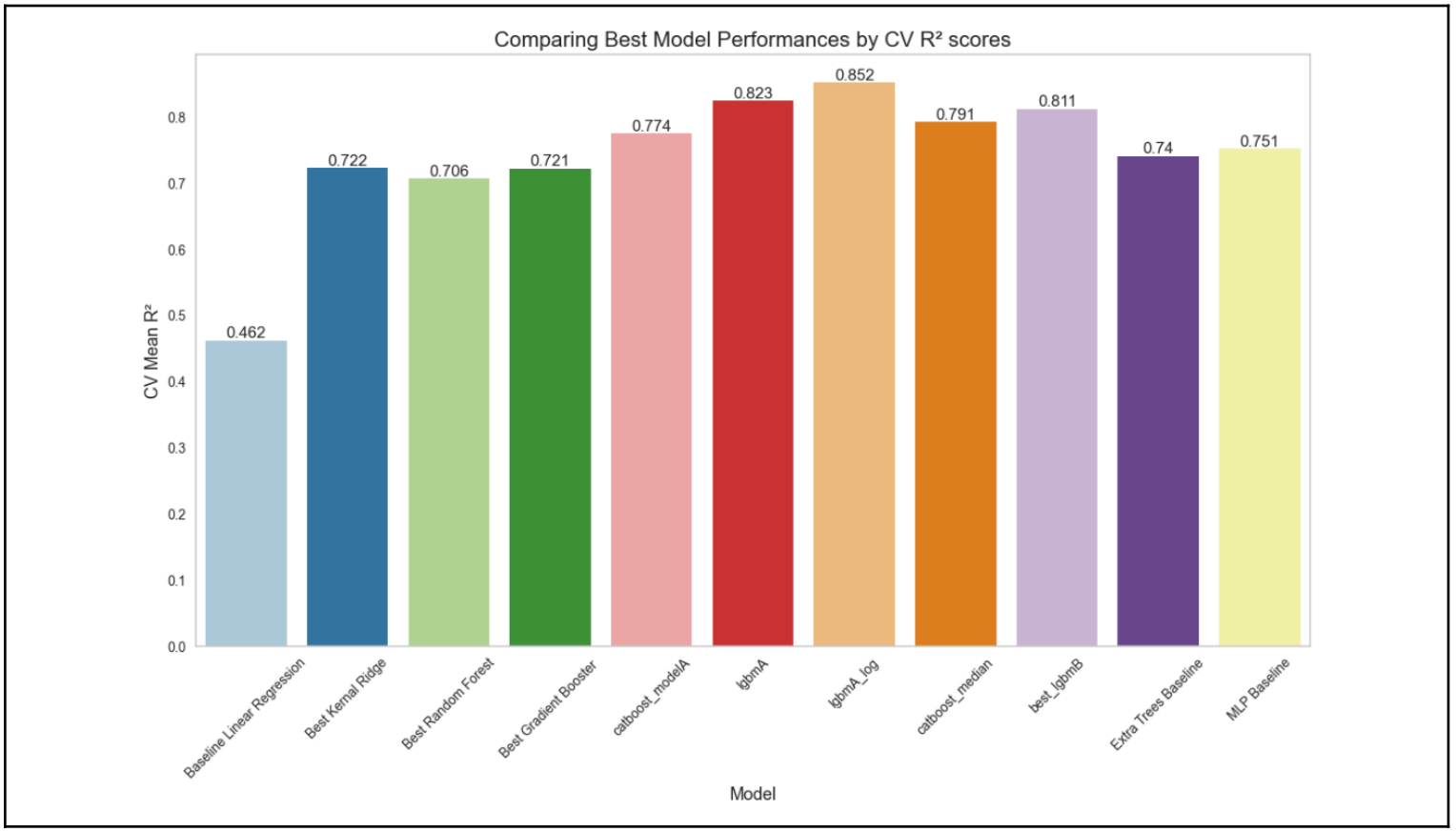


Figure 17. Barchart of the Best Model Results for R^2 scores

The best performing models on the new datasets, based on R² scores, can be seen above (Figure 17), and all metrics can be easily compared to the first iteration results, and LR baseline model. The first LR model's results are poor, which is expected, given that it is a linear model being applied for multiple linear regression on nonlinear data. The model is underfitting with poor training and testing results. RFR produced an overfitting model with a high training score of 0.9597334032950553 and a low test score of 0.7572457452441076, relative to training. Kernel Ridge on a polynomial kernel exhibited surprisingly good results on the first iteration, with a training R² of 0.944062, and test R² score of 0.943771. On further investigation, the results were indeed too good to be true, having mistakenly trained the model on the complete dataset. Its CV Mean R² score has however marginally increased from 0.704091 to 0.7221912831991801 on better processed data. These results are still on the mean imputed datasets. GBR provided a slightly better CV Mean R² of 0.7213522752382111. However the model is also clearly overfitting to the training data with a considerable difference in training to testing results.

From the barchart in Figure 17, we appreciate the improvements made on Cross Validated R² scores from the earlier models. Both CBR and LightGBM algorithms have provided consistently higher results, predominantly on the median imputed datasets, with categorical data types attributes which have not been normalised or label encoded.
LightGBM has produced the best results. The model achieves a CV Mean R² score of 0.823 which increases to 0.852 when optimised and trained on the log transformation of the target variable. In comparison to the baseline model, we have achieved an 84.5% improvement.

Model Evaluation

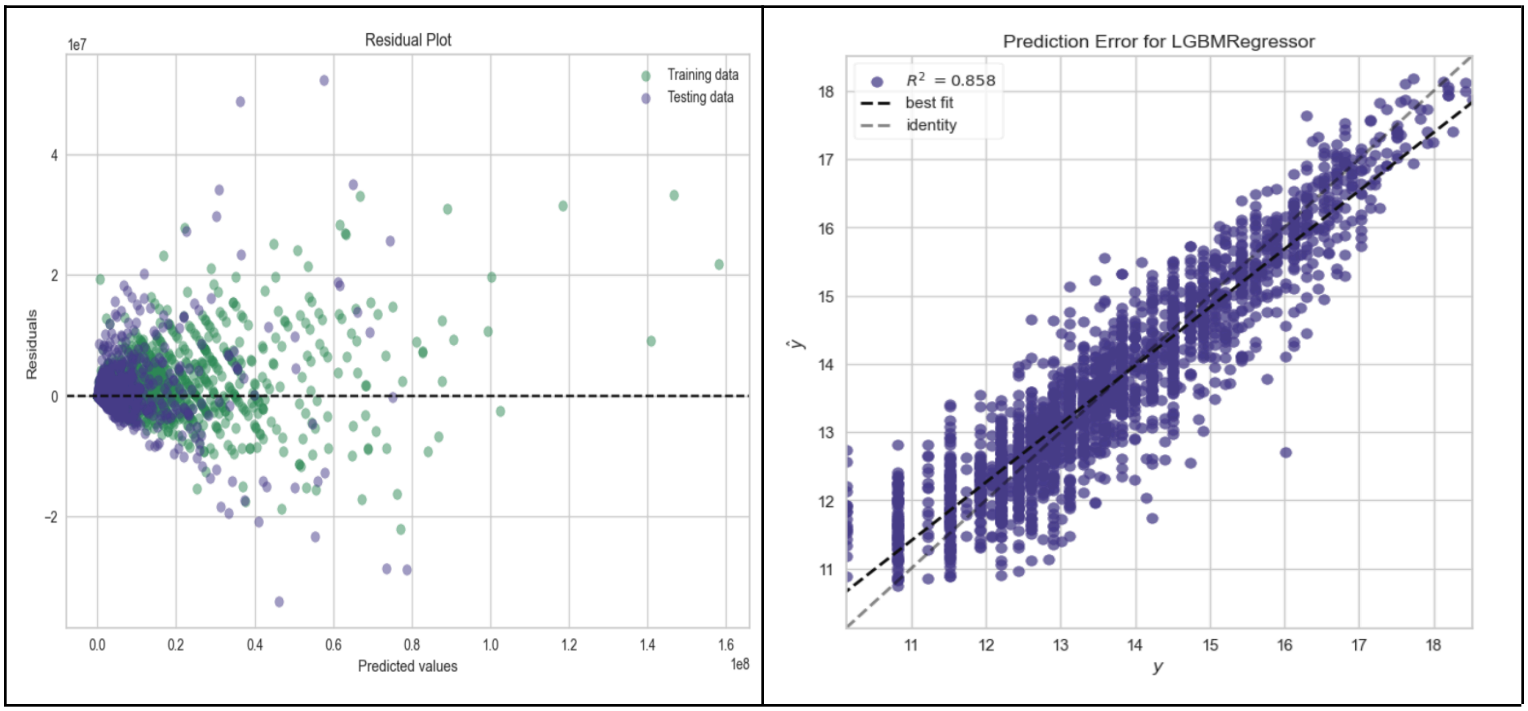


Figure 18. Residuals Plot (left) and Prediction Error graph (right) for the Best LightGBM Regressor Model

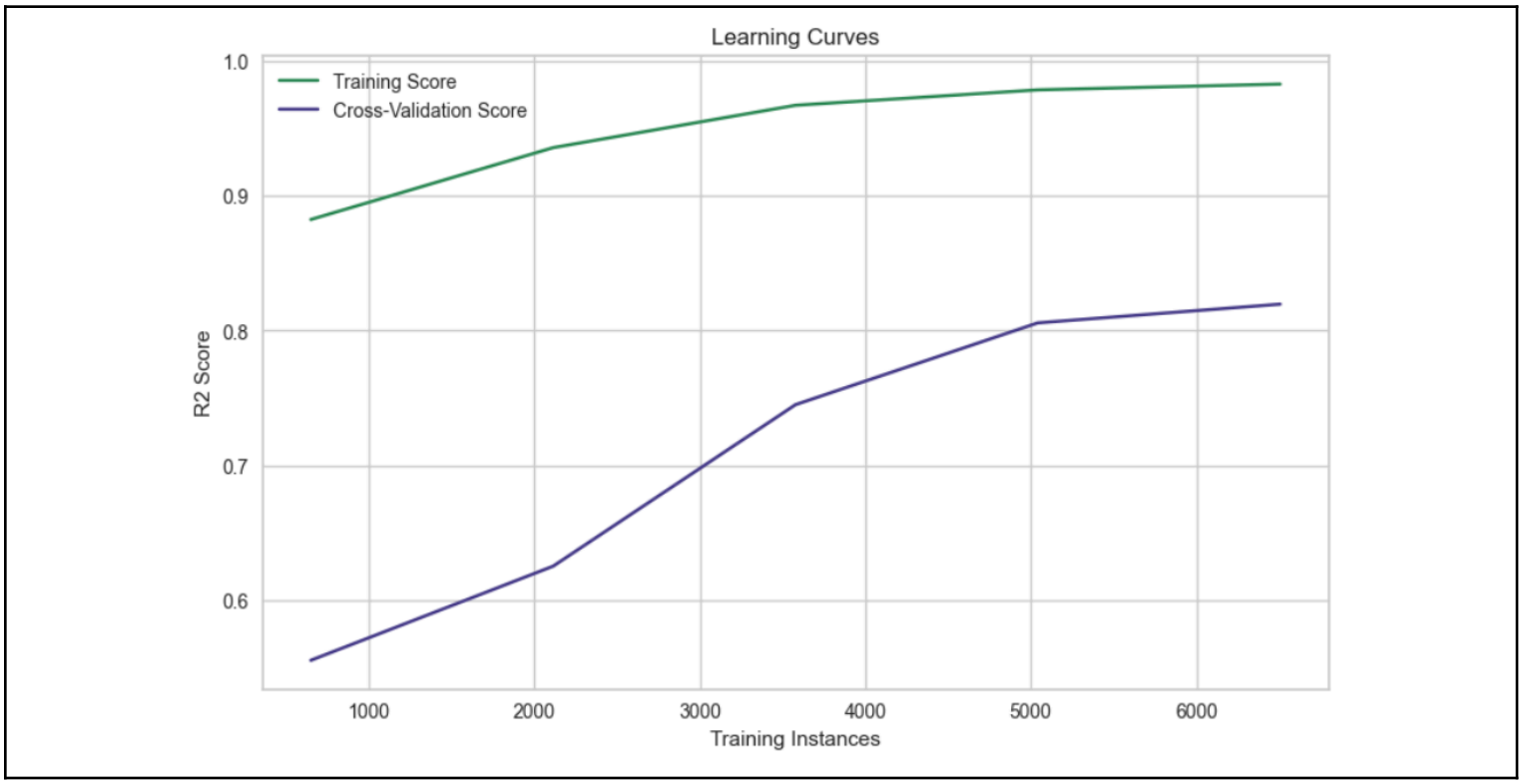


Figure 19. Learning curve for the Best LightGBM Regressor Model

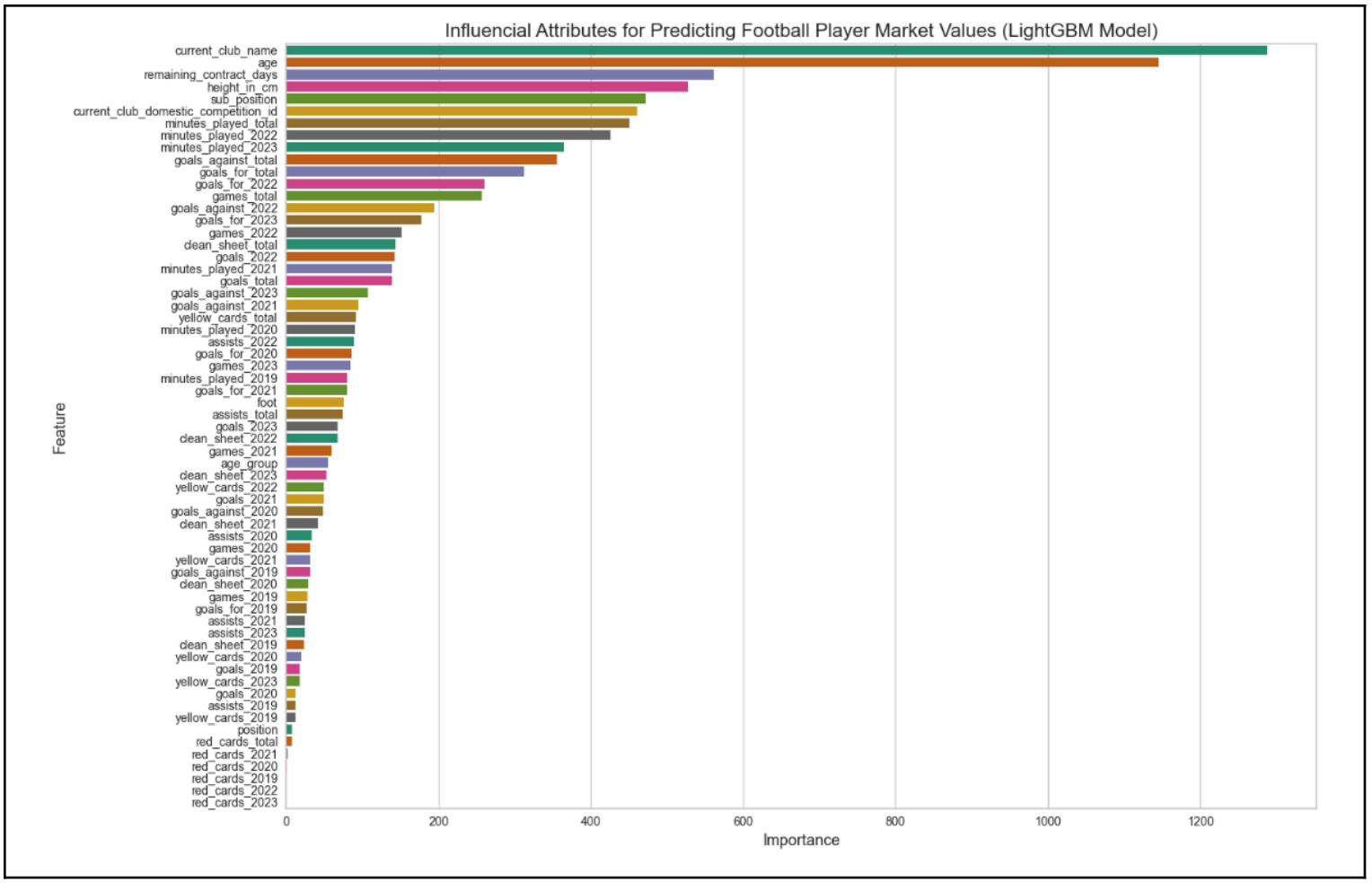


Figure 20. Feature Importance for the Best LightGBM Regressor Model

In evaluating regression models, a residual is a measure of how far away a point is vertically from the regression line (Gohar, 2020). They are the difference between our predicted player market values and their actual market values (i.e. predicted — actual values), and a residuals plot displays the relationship of errors, with predictions, as well as the distribution of these errors.
From the residuals plot in Figure 18, we find little noticeable difference or pattern in error for lower market values on the left. The model appears to predict well for lower market values, densely located around the origin. However, as predicted values increase, we see a steady increase in prediction errors for the test data displayed in purple. As our data is transformed, we are unable to utilise Yellowbricks library for creating the plot, which would also provide its frequency distribution. We can nevertheless establish a normal distribution by eye, based on the even or symmetrical spread of purple dots about the origin line (Gohar, 2020).
Through the prediction error plot in Figure 18, we visualise the variance in the model. The prediction error plot displays the actual targets from the data against the predicted values generated by the model. The closer the broken lines align, the better the model is performing. We find the model performs best when predicting around 14.
Although we could have determined an overfitting model from the training and testing R^2 scores in Figure 16, the learning curves in Figure 19 do offer hope for future improvement, as they indicate an increase in scores with an increase in training instances (Brownlee, 2019).

To conclude the evaluation of the model, we leverage an inbuilt LightGBM function for feature importance. The horizontal barchart in Figure 20 provides direct insight into the contributing features in descending order of importance. We find that the model considers a players club name, age, remaining contract days, height and sub position the most, when making a prediction on their market value. This is valuable information for presentation to stakeholders, for model explainability and informed decision making.

Predictive Analytics and Interpretability

Once satisfied with the training and performance of a model, it can be used to make predictions on new data. In interpreting the results both globally and locally, we utilise the SHAP library, designed for explainable AI. SHAP Produces shapley values which help us understand how a model's features impact its predictions. It uses a game theoretic approach that measures each player's contribution to the final outcome.
The SHAP summary plots in Figure 21,provide a visual representation of the absolute SHAP values for each feature, showing their importance relative to the model's output. The x-axis of the summary plot typically displays the average absolute SHAP value for each feature across all instances in the dataset. Therefore, the values on the x-axis represent the magnitude of the impact of each feature on the model's predictions. The beauty and advantage of this SHAP plot is that values on the x-axis are usually displayed in the same metric as the target variable. However, as this model is developed on the log transformation of the target variable, 'market_value_in_eur', it may not be as insightful. Nonetheless, in the plot on the left, we understand that the model is most influenced by a player's current club domestic competition, current club name, their total goals scored, goals scored in the year 2022, age and total games played. Both graphics display features in order of importance. The visual on the right provided additional insight. Each point represents an entry or player from the dataset, while the colour of each point represents the value of the corresponding feature, with red indicating higher value and blue indicating lower value to the model's prediction. Taking age and remaining contract days as examples, through the visual we garner that the lower a player's age, the higher the predicted market value, while the higher a player's remaining contract days, the higher the predicted market value.

In Figure 22 and 23, SHAP values add up to the difference between the expected model output and the actual output for a given input. By doing so, SHAP values provide an accurate and local interpretation of the model's prediction for a given input (Readthedocs.io, 2024). For a random player, in Figure 22 we find a measurable impact of features to the model's prediction. As with the previous plots, ideally this value would be in euros, the same metric as the target variable. We see that the age feature value at 36 is bringing down the predicted market value while the current club name being Real Madrid is contributing to a higher predicted market value.

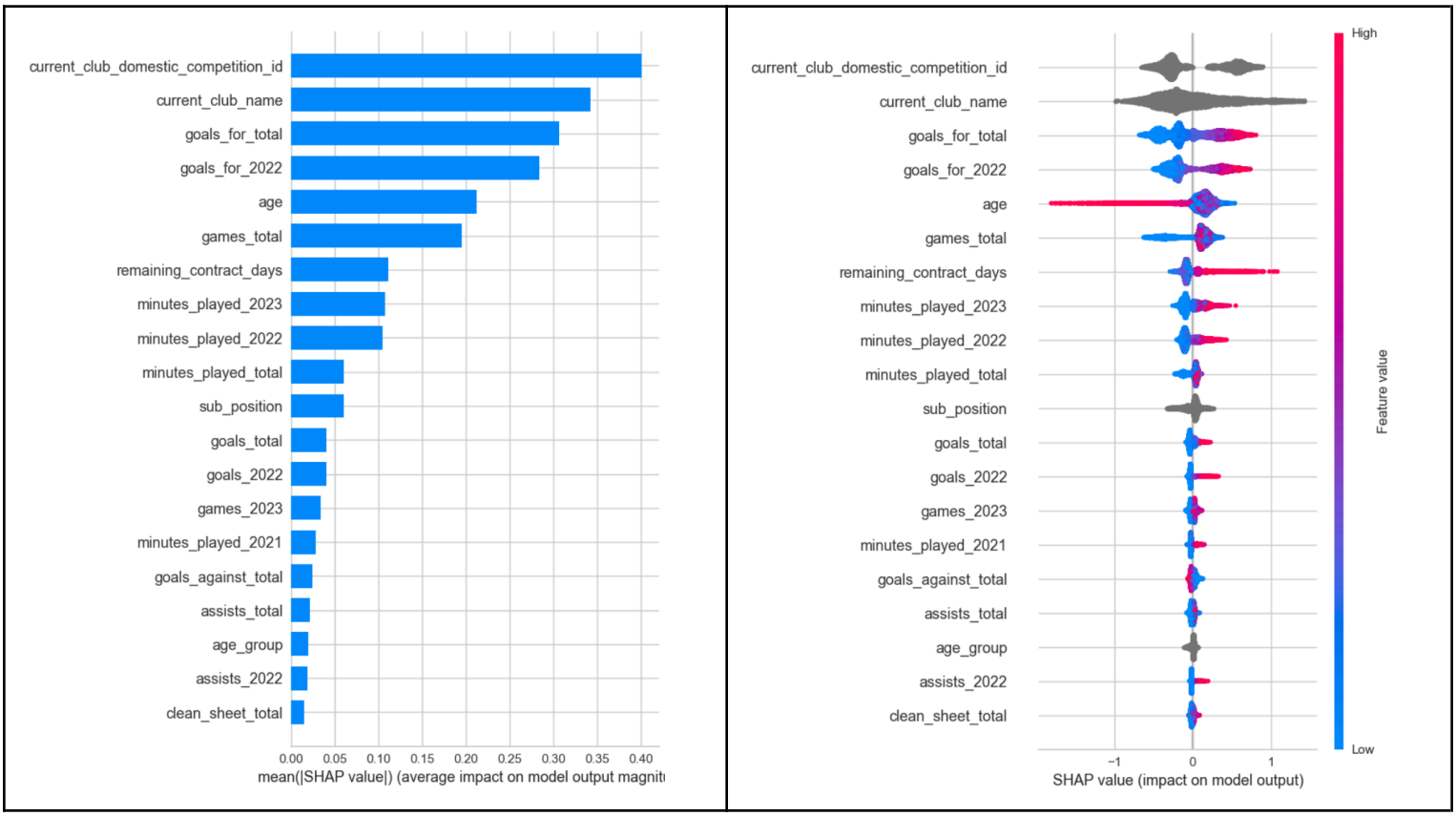


Figure 21. SHAP Feature Importance for the Best LightGBM Regressor Model

Figure 22. SHAP force for local interpretation

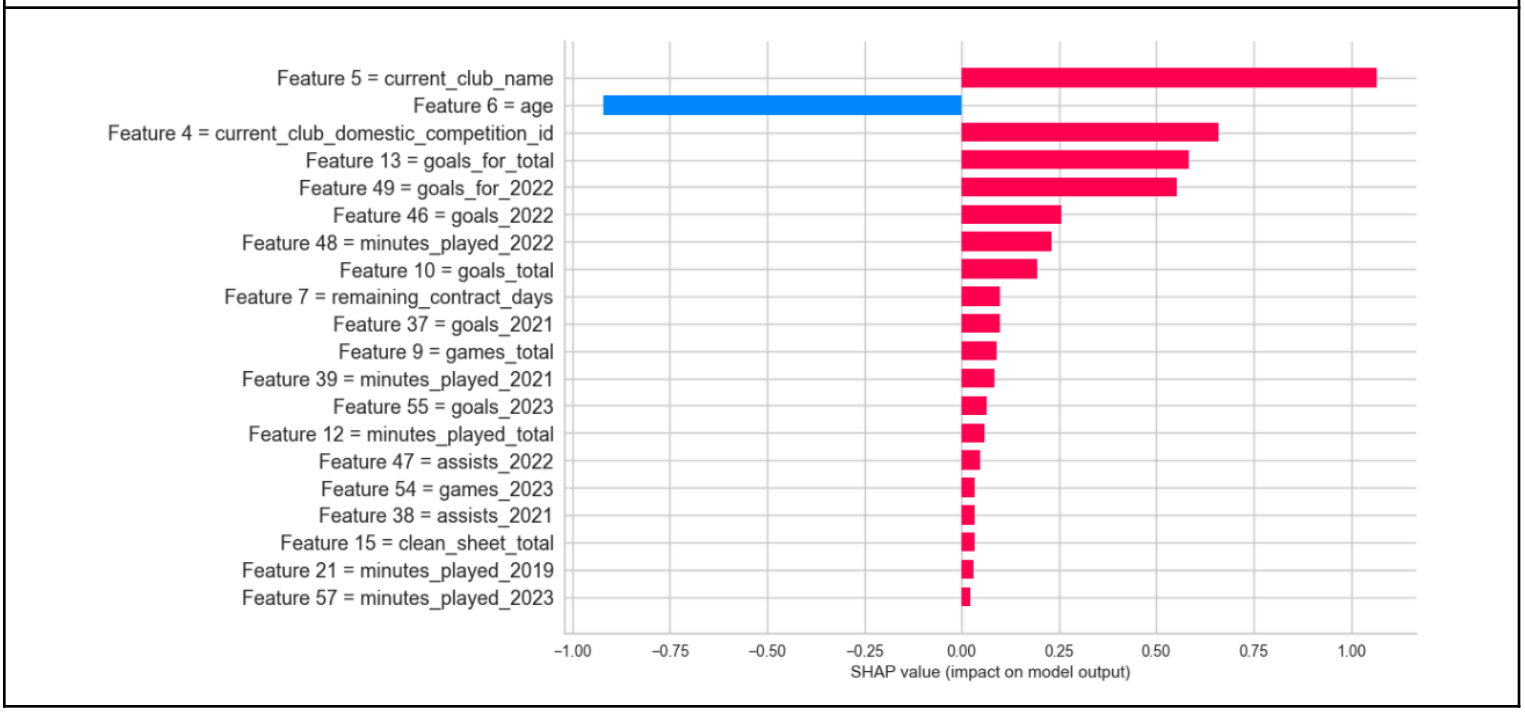


Figure 22. SHAP force for local interpretation on the Best LightGBM Regressor Model

Deployment

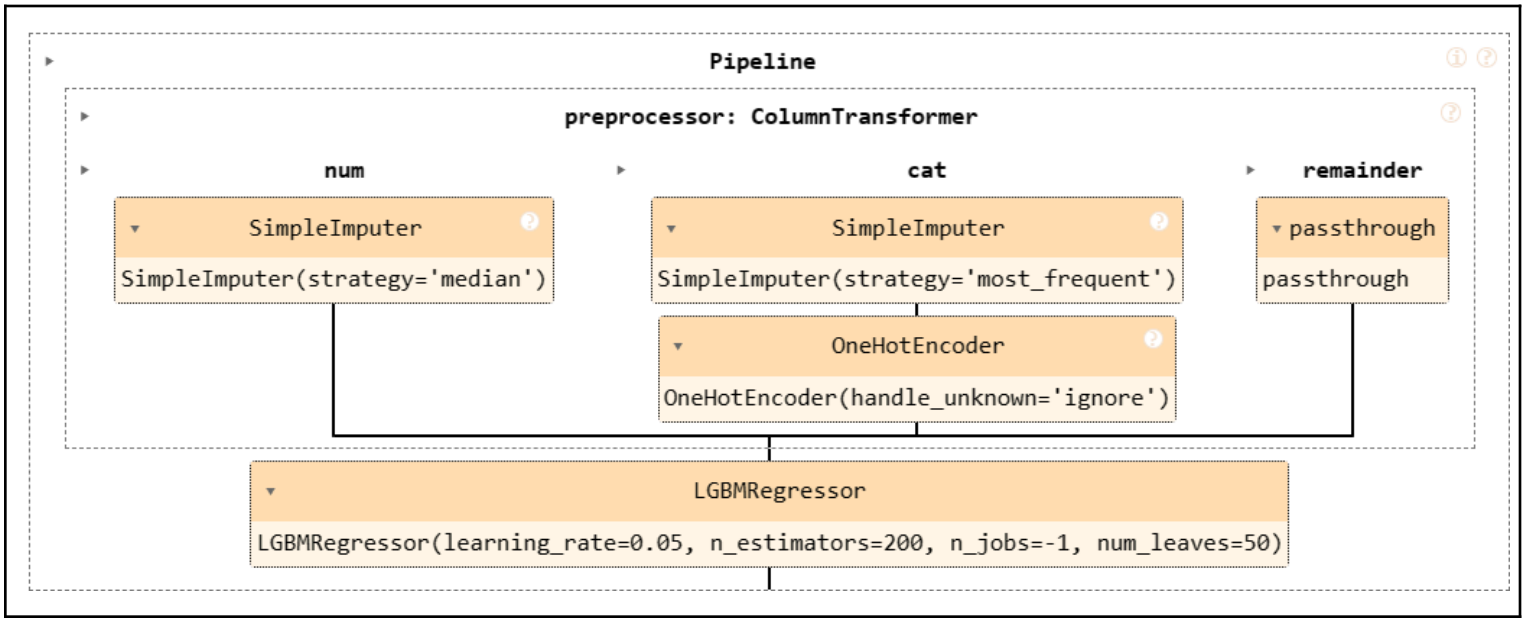


Figure 23. LightGBM Regressor Model for PLayer Market Value Prediction Pipeline

The Deployment phase within the CRISP-DM methodology involves implementing the solutions derived from modelling into practical use. It marks the culmination of the data mining process by applying these solutions and models into operational settings. This includes integrating them with existing systems, providing documentation, and offering support to end-users to derive business value. Although perhaps premature, a pipeline of the model has been generated.

Conclusion

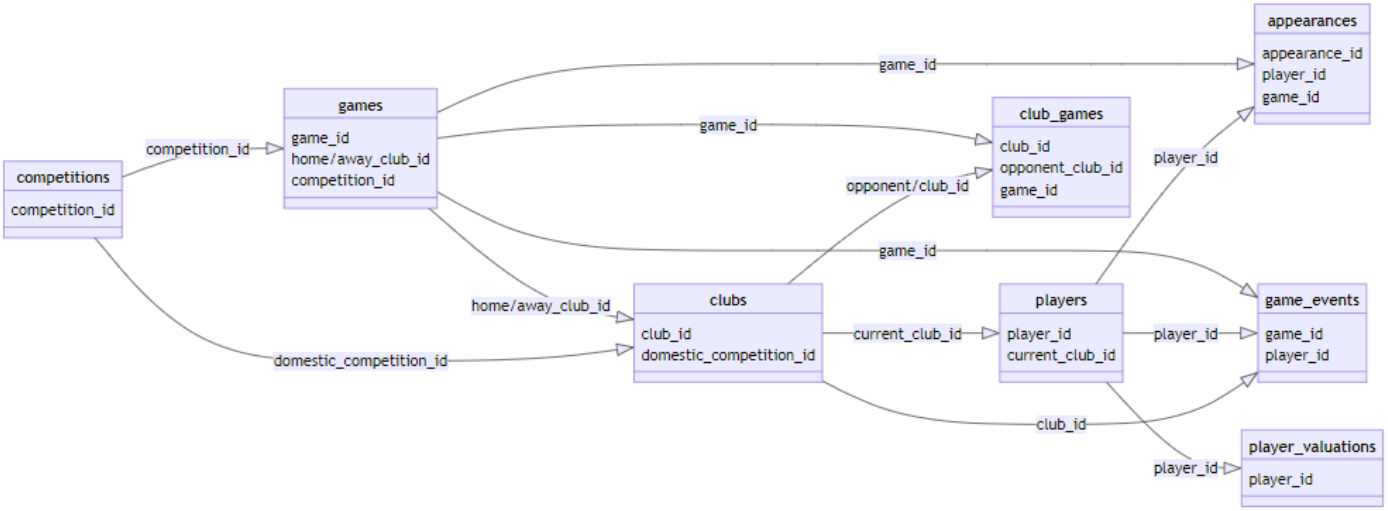
Player scouting and market value analysis are key areas where ML can provide valuable knowledge and insights to the football industry. This capstone project aims to contribute to this by empowering clubs and stakeholders with data driven decision making capabilities for a competitive advantage. By leveraging machine learning on the “Football Data from Transfermarkt" dataset, the project aligns with the evolving landscape of sports analytics, providing practical solutions to real world challenges in the football industry. In adhering to the CRISP_DM life cycle, the application of this project has proven challenging, due to data complexities, time restraints, and computational limitations. The results however, are quite promising, with as much as an 84.5% improvement on select baseline metrics. The project still provides plenty of scope for improvement, at each step of the development process. Further statistical analysis can be conducted for additional insight into the characteristics of the datasets. Different techniques can be applied in preprocessing, including alternative imputation and scaling methods. Additional feature engineering, including dimensionality reduction through PCA, and feature generation, by resolving the infinity problem can be achieved. If anything, whilst modelling, this iteration has highlighted the importance of individually garnering, tuning and optimising a model. The most successful model is produced with the LightGBM algorithm. A model that does not even rank in the top ten, when comparing baseline models for further optimisation using Pycarot. This highlights the hidden potential for improvement within each model, and emphasises the need for continual research and improvement.

Four Jupyter notebooks documenting the process and python code are created, one each for EDA, Preprocessing and Modelling Iterations 1 and 2. Unfortunately the Preprocessing notebook is too large to be pushed to the Github repository. A pdf of the notebook has been uploaded instead.

Github: https://github.com/KaviCCT/Strategic_Thinking_CA2

Appendix A

```
appearances_df: (1485697, 13)
club_games_df: (128586, 11)
clubs_df: (426, 16)
competitions_df: (43, 10)
game_events_df: (652010, 10)
game_lineups_df: (86822, 9)
games_df: (64293, 23)
player_valuations_df: (440663, 9)
players_df: (30302, 23)
```



```
********************************************************************************
appearances (1485697, 13)

The total number of missing values in appearances dataframe are:  324 out of 19314061.

That is equal to  0.0 percent.

********************************************************************************
club_games (128586, 11)

The total number of missing values in club_games dataframe are:  79412 out of 1414446.

That is equal to  5.61 percent.

********************************************************************************
clubs (426, 16)

The total number of missing values in clubs dataframe are:  937 out of 6816.

That is equal to  13.75 percent.

********************************************************************************
competitions (43, 10)

The total number of missing values in competitions dataframe are:  14 out of 430.

That is equal to  3.26 percent.

********************************************************************************
game_events (652010, 10)

The total number of missing values in game_events dataframe are:  1195930 out of 6520100.

That is equal to  18.34 percent.

********************************************************************************
game_lineups (86822, 9)

The total number of missing values in game_lineups dataframe are:  0 out of 781398.

That is equal to  0.0 percent.

********************************************************************************
games (64293, 23)

The total number of missing values in games dataframe are:  195309 out of 1478739.

That is equal to  13.21 percent.

********************************************************************************
player_valuations (440663, 9)

The total number of missing values in player_valuations dataframe are:  0 out of 3965967.

That is equal to  0.0 percent.

********************************************************************************
players (30302, 23)

The total number of missing values in players dataframe are:  51174 out of 696946.

That is equal to  7.34 percent.
```

Figure 1.1. Dataset names, shapes and missing values

```
********************************************************************************
appearances (1485697, 13)
********************************************************************************

head:

    appearance_id  game_id  player_id  player_club_id  player_current_club_id  \
0   2231978_38004  2231978      38004             853                     235
1   2233748_79232  2233748      79232            8841                    2698
2   2234413_42792  2234413      42792            6251                     465
3   2234418_73333  2234418      73333            1274                    6646
4  2234421_122011  2234421     122011             195                    3008

         date       player_name competition_id  yellow_cards  red_cards  \
0  2012-07-03  Aurélien Joachim            CLQ             0          0
1  2012-07-05    Ruslan Abyshov            ELQ             0          0
2  2012-07-05       Sander Puri            ELQ             0          0
3  2012-07-05   Vegar Hedenstad            ELQ             0          0
4  2012-07-05  Markus Henriksen            ELQ             0          0

   goals  assists  minutes_played
0      2        0              90
1      0        0              90
2      0        0              45
3      0        0              90
4      0        1              90

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1485697 entries, 0 to 1485696
Data columns (total 13 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   appearance_id           1485697 non-null  object
 1   game_id                 1485697 non-null  int64
 2   player_id               1485697 non-null  int64
 3   player_club_id          1485697 non-null  int64
 4   player_current_club_id  1485697 non-null  int64
 5   date                    1485697 non-null  object
 6   player_name             1485373 non-null  object
 7   competition_id          1485697 non-null  object
 8   yellow_cards            1485697 non-null  int64
 9   red_cards               1485697 non-null  int64
 10  goals                   1485697 non-null  int64
 11  assists                 1485697 non-null  int64
 12  minutes_played          1485697 non-null  int64
dtypes: int64(9), object(4)
memory usage: 147.4+ MB
None

Number of uniques values:

appearance_id             1485697
game_id                     58417
player_id                   23740
player_club_id               1043
player_current_club_id        425
date                         3291
player_name                 23223
competition_id                 43
yellow_cards                    3
red_cards                       2
goals                           7
assists                         7
minutes_played                120
dtype: int64

Statistical description:

            game_id     player_id  player_club_id  player_current_club_id  \
count  1.485697e+06  1.485697e+06    1.485697e+06            1.485697e+06
mean   2.998498e+06  1.769102e+05    2.858144e+03            3.626989e+03
std    5.445426e+05  1.623144e+05    7.024775e+03            9.271304e+03
min    2.211607e+06  1.000000e+01    1.000000e+00           -1.000000e+00
25%    2.512848e+06  5.297900e+04    2.890000e+02            3.360000e+02
50%    2.899907e+06  1.264220e+05    8.550000e+02            9.310000e+02
75%    3.433282e+06  2.574480e+05    2.441000e+03            2.687000e+03
max    4.196249e+06  1.166093e+06    8.367800e+04            8.367800e+04

       yellow_cards     red_cards         goals       assists  minutes_played
count  1.485697e+06  1.485697e+06  1.485697e+06  1.485697e+06    1.485697e+06
mean   1.491495e-01  3.837929e-03  9.638843e-02  7.434019e-02    6.972077e+01
std    3.677364e-01  6.183205e-02  3.319122e-01  2.834616e-01    2.973195e+01
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00    1.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00    5.300000e+01
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00    9.000000e+01
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00    9.000000e+01
max    2.000000e+00  1.000000e+00  6.000000e+00  6.000000e+00    1.200000e+02
```

```
********************************************************************************
club_games (128586, 11)
********************************************************************************

head:

   game_id  club_id  own_goals  own_position      own_manager_name  \
0  2221751      431        1.0           NaN         Lutz Göttling
1  2221755       83        3.0           NaN      Ralph Hasenhüttl
2  2222597     3725        2.0           2.0  Stanislav Cherchesov
3  2222627     2696        0.0          11.0        Andrey Kobelev
4  2222658     2410        0.0           2.0        Leonid Slutski

   opponent_id  opponent_goals  opponent_position opponent_manager_name  \
0           60             2.0                NaN     Christian Streich
1         4795             0.0                NaN            Tomas Oral
2          232             1.0                5.0           Unai Emery
3         4128             2.0               10.0         Rustem Khuzin
4          121             2.0               13.0          Dan Petrescu

  hosting  is_win
0    Home       0
1    Home       1
2    Home       1
3    Home       0
4    Home       0

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128586 entries, 0 to 128585
Data columns (total 11 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   game_id                128586 non-null  int64
 1   club_id                128586 non-null  int64
 2   own_goals              128582 non-null  float64
 3   own_position           90362 non-null   float64
 4   own_manager_name       127108 non-null  object
 5   opponent_id            128586 non-null  int64
 6   opponent_goals         128582 non-null  float64
 7   opponent_position      90362 non-null   float64
 8   opponent_manager_name  127108 non-null  object
 9   hosting                128586 non-null  object
 10  is_win                 128586 non-null  int64
dtypes: float64(4), int64(4), object(3)
memory usage: 10.8+ MB
None

Number of uniques values:

game_id                  64293
club_id                   2669
own_goals                   18
own_position                21
own_manager_name          5473
opponent_id               2669
opponent_goals              18
opponent_position           21
opponent_manager_name     5473
hosting                      2
is_win                       2
dtype: int64

Statistical description:

            game_id        club_id      own_goals  own_position  \
count  1.285860e+05  128586.000000  128582.000000  90362.000000
mean   3.001949e+06    4655.724223       1.456191      9.366991
std    5.478810e+05   10806.479172       1.401471      5.312555
min    2.211607e+06       1.000000       0.000000      1.000000
25%    2.512848e+06     352.000000       0.000000      5.000000
50%    2.903799e+06     995.000000       1.000000      9.000000
75%    3.433394e+06    3057.000000       2.000000     14.000000
max    4.196249e+06  112755.000000      19.000000     21.000000

         opponent_id  opponent_goals  opponent_position         is_win
count  128586.000000   128582.000000       90362.000000  128586.000000
mean     4655.724223        1.456191           9.366991       0.392539
std     10806.479172        1.401471           5.312555       0.488317
min         1.000000        0.000000           1.000000       0.000000
25%       352.000000        0.000000           5.000000       0.000000
50%       995.000000        1.000000           9.000000       0.000000
75%      3057.000000        2.000000          14.000000       1.000000
max    112755.000000       19.000000          21.000000       1.000000
```

```
********************************************************************************
clubs (426, 16)
********************************************************************************

head:

   club_id         club_code              name domestic_competition_id  \
0      105    sv-darmstadt-98     SV Darmstadt 98                    L1
1    11127  ural-ekaterinburg  Ural Yekaterinburg                   RU1
2      114  besiktas-istanbul         Besiktas JK                   TR1
3       12            as-rom             AS Roma                   IT1
4      148  tottenham-hotspur   Tottenham Hotspur                   GB1

   total_market_value  squad_size  average_age  foreigners_number  \
0                 NaN          29         26.1                 11
1                 NaN          25         27.6                 13
2                 NaN          32         26.7                 16
3                 NaN          26         26.7                 17
4                 NaN          29         25.6                 21

   foreigners_percentage  national_team_players  \
0                   37.9                      1
1                   52.0                      4
2                   50.0                     14
3                   65.4                     17
4                   72.4                     21

                    stadium_name  stadium_seats net_transfer_record  \
0  Merck-Stadion am Böllenfalltor          17500             €-1.60m
1             Yekaterinburg Arena          23000              €-770k
2                  Beşiktaş Park          42590            €-14.50m
3                Olimpico di Roma          73261            +€65.20m
4      Tottenham Hotspur Stadium          62062           €-126.40m

   coach_name  last_season                                                url
0         NaN         2023  https://www.transfermarkt.co.uk/sv-darmstadt-9...
1         NaN         2023  https://www.transfermarkt.co.uk/ural-ekaterinb...
2         NaN         2023  https://www.transfermarkt.co.uk/besiktas-istan...
3         NaN         2023  https://www.transfermarkt.co.uk/as-rom/startse...
4         NaN         2023  https://www.transfermarkt.co.uk/tottenham-hots...

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426 entries, 0 to 425
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   club_id                 426 non-null    int64
 1   club_code               426 non-null    object
 2   name                    426 non-null    object
 3   domestic_competition_id  426 non-null    object
 4   total_market_value      0 non-null      float64
 5   squad_size              426 non-null    int64
 6   average_age             388 non-null    float64
 7   foreigners_number       426 non-null    int64
 8   foreigners_percentage   379 non-null    float64
 9   national_team_players   426 non-null    int64
 10  stadium_name            426 non-null    object
 11  stadium_seats           426 non-null    int64
 12  net_transfer_record     426 non-null    object
 13  coach_name              0 non-null      float64
 14  last_season             426 non-null    int64
 15  url                     426 non-null    object
dtypes: float64(4), int64(6), object(6)
memory usage: 53.4+ KB
None

Number of uniques values:

club_id                   426
club_code                 426
name                      426
domestic_competition_id    14
total_market_value          0
squad_size                 31
average_age                68
foreigners_number          28
foreigners_percentage     173
national_team_players      23
stadium_name              409
stadium_seats             379
net_transfer_record       273
coach_name                  0
last_season                12
url                       426
dtype: int64

Statistical description:

            club_id  total_market_value  squad_size  average_age  \
count    426.000000                 0.0  426.000000   388.000000
mean    5314.525822                 NaN   24.352113    25.320619
std    11752.013818                 NaN    8.796949     1.525676
min        3.000000                 NaN    0.000000    18.300000
25%      421.000000                 NaN   24.000000    24.300000
50%     1139.500000                 NaN   27.000000    25.300000
75%     3415.750000                 NaN   29.000000    26.300000
max    83678.000000                 NaN   41.000000    29.000000

       foreigners_number  foreigners_percentage  national_team_players  \
count         426.000000             379.000000             426.000000
mean           10.946009              45.152507               4.793427
std             6.714998              19.944304               5.036297
min             0.000000               2.400000               0.000000
25%             6.000000              30.800000               1.000000
50%            12.000000              46.700000               3.500000
75%            16.000000              58.200000               7.000000
max            31.000000             100.000000              22.000000

       stadium_seats  coach_name  last_season
count     426.000000         0.0   426.000000
mean    24300.420188         NaN  2020.830986
std     17146.038353         NaN     3.207283
min         0.000000         NaN  2012.000000
25%     11006.000000         NaN  2019.000000
50%     20046.000000         NaN  2023.000000
75%     32798.000000         NaN  2023.000000
max     81365.000000         NaN  2023.000000
```

```
********************************************************************************
competitions (43, 10)
********************************************************************************

head:

  competition_id              competition_code  \
0            CIT                    italy-cup
1           NLSC          johan-cruijff-schaal
2            GRP               kypello-elladas
3           POSU  supertaca-candido-de-oliveira
4           RUSS             russian-super-cup

                          name           sub_type          type  \
0                    italy-cup       domestic_cup  domestic_cup
1          johan-cruijff-schaal  domestic_super_cup         other
2               kypello-elladas       domestic_cup  domestic_cup
3  supertaca-candido-de-oliveira  domestic_super_cup         other
4              russian-super-cup  domestic_super_cup         other

   country_id country_name domestic_league_code confederation  \
0          75        Italy                  IT1        europa
1         122  Netherlands                  NL1        europa
2          56       Greece                  GR1        europa
3         136     Portugal                  PO1        europa
4         141       Russia                  RU1        europa

                                              url
0  https://www.transfermarkt.co.uk/italy-cup/star...
1  https://www.transfermarkt.co.uk/johan-cruijff-...
2  https://www.transfermarkt.co.uk/kypello-ellada...
3  https://www.transfermarkt.co.uk/supertaca-cand...
4  https://www.transfermarkt.co.uk/russian-super-...

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43 entries, 0 to 42
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   competition_id        43 non-null     object
 1   competition_code      43 non-null     object
 2   name                  43 non-null     object
 3   sub_type              43 non-null     object
 4   type                  43 non-null     object
 5   country_id            43 non-null     int64
 6   country_name          36 non-null     object
 7   domestic_league_code  36 non-null     object
 8   confederation         43 non-null     object
 9   url                   43 non-null     object
dtypes: int64(1), object(9)
memory usage: 3.5+ KB
None

Number of uniques values:

competition_id          43
competition_code        42
name                    42
sub_type                11
type                     4
country_id              15
country_name            14
domestic_league_code    14
confederation            1
url                     43
dtype: int64

Statistical description:

       country_id
count   43.000000
mean    97.093023
std     69.766896
min     -1.000000
25%     39.500000
50%    122.000000
75%    157.000000
max    190.000000
```

```
********************************************************************************
game_events (652010, 10)
********************************************************************************

head:

                    game_event_id        date  game_id  minute  \
0  2f41da30c471492e7d4a984951671677  05/08/2012  2211607      77
1  a72f7186d132775f234d3e2f7bc0ed5b  05/08/2012  2211607      77
2  b2d721eaed4692a5c59a92323689ef18  05/08/2012  2211607       3
3  aef768899cedac0c9a650980219075a2  05/08/2012  2211607      53
4  5d6d9533023057b6619ecd145a038bbe  05/08/2012  2211607      74

           type  club_id  player_id  \
0         Cards      610       4425
1         Cards      383      33210
2         Goals      383      36500
3         Goals      383      36500
4  Substitutions      383      36500

                                      description  player_in_id  \
0              1. Yellow card  , Mass confrontation           NaN
1              1. Yellow card  , Mass confrontation           NaN
2  , Header, 1. Tournament Goal Assist: , Corner,...           NaN
3  , Right-footed shot, 2. Tournament Goal Assist...           NaN
4                                  , Not reported       49499.0

   player_assist_id
0               NaN
1               NaN
2           56416.0
3          146258.0
4               NaN

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 652010 entries, 0 to 652009
Data columns (total 10 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   game_event_id     652010 non-null  object
 1   date              652010 non-null  object
 2   game_id           652010 non-null  int64
 3   minute            652010 non-null  int64
 4   type              652010 non-null  object
 5   club_id           652010 non-null  int64
 6   player_id         652010 non-null  int64
 7   description       317778 non-null  object
 8   player_in_id      413164 non-null  float64
 9   player_assist_id  29158 non-null   float64
dtypes: float64(2), int64(4), object(4)
memory usage: 49.7+ MB
None

Number of uniques values:


game_event_id       652010
date                  3358
game_id              64149
minute                 121
type                     4
club_id               2574
player_id            51478
description           4570
player_in_id         50149
player_assist_id      9439
dtype: int64

Statistical description:

            game_id         minute        club_id     player_id  player_in_id  \
count  6.520100e+05  652010.000000  652010.000000  6.520100e+05  4.131640e+05
mean   3.027107e+06      63.491683    4548.496678  1.962196e+05  2.386084e+05
std    5.918583e+05      21.908195   10796.108584  1.834003e+05  2.043951e+05
min    2.211607e+06      -1.000000       1.000000  1.000000e+01  1.000000e+01
25%    2.481046e+06      51.000000     336.000000  5.641600e+04  7.309700e+04
50%    2.942767e+06      68.000000     984.000000  1.352290e+05  1.863680e+05
75%    3.553565e+06      80.000000    3015.000000  2.862970e+05  3.441070e+05
max    4.196249e+06     120.000000  112755.000000  1.195049e+06  1.195054e+06

       player_assist_id
count      2.915800e+04
mean       1.142005e+05
std        1.575255e+05
min        1.000000e+01
25%        2.943400e+04
50%        5.728750e+04
75%        1.252508e+05
max        1.189238e+06
```

```
**********************************************************************************************
game_lineups (86822, 9)
**********************************************************************************************

head:

                    game_lineups_id  game_id  club_id             type number  \
0  baa0d6827dab4fab07d8dc1604e720a7  3606208      338  starting_lineup     15
1  1715ec342bf902522b69322a036a3f29  3606208      338  starting_lineup     71
2  d7c22834cb4c20efeddc527511feabad  3606208      338  starting_lineup     34
3  113289469cede9376049b78521f7b382  3606208      338  starting_lineup     25
4  e6abe553801b09bc623c3deb96acba17  3606208      338  starting_lineup     16

   player_id        player_name  team_captain      position
0     264372   Viktor Tsygankov             0  Right Winger
1      91931        Denys Boyko             0    Goalkeeper
2     505463   Oleksandr Syrota             0   Centre-Back
3     659089      Ilya Zabarnyi             0   Centre-Back
4     404842  Vitaliy Mykolenko             0     Left-Back

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86822 entries, 0 to 86821
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   game_lineups_id  86822 non-null  object
 1   game_id          86822 non-null  int64
 2   club_id          86822 non-null  int64
 3   type             86822 non-null  object
 4   number           86822 non-null  object
 5   player_id        86822 non-null  int64
 6   player_name      86822 non-null  object
 7   team_captain     86822 non-null  int64
 8   position         86822 non-null  object
dtypes: int64(4), object(5)
memory usage: 6.0+ MB
None

Number of uniques values:

game_lineups_id    86822
game_id             2144
club_id              972
type                   2
number               100
player_id          21942
player_name        21660
team_captain           2
position              17
dtype: int64

Statistical description:

            game_id        club_id     player_id  team_captain
count  8.682200e+04   86822.000000  8.682200e+04  86822.000000
mean   4.113804e+06    9505.120476  4.509931e+05      0.045611
std    4.844309e+04   19245.549845  2.733798e+05      0.208640
min    3.606208e+06       1.000000  3.159000e+03      0.000000
25%    4.094631e+06     418.000000  2.376585e+05      0.000000
50%    4.113131e+06    1148.000000  4.066400e+05      0.000000
75%    4.146925e+06    6673.000000  6.269540e+05      0.000000
max    4.196249e+06  112755.000000  1.195054e+06      1.000000
```

```
********************************************************************************
games (64293, 23)
********************************************************************************

head:

   game_id competition_id  season         round        date  home_club_id  \
0  2222597            RU1    2012   6. Matchday  2012-08-25          3725
1  2222627            RU1    2012   5. Matchday  2012-08-20          2696
2  2222658            RU1    2012  10. Matchday  2012-09-30          2410
3  2222664            RU1    2012   8. Matchday  2012-09-15           932
4  2222683            RU1    2012  12. Matchday  2012-10-22          2696

   away_club_id  home_club_goals  away_club_goals  home_club_position  ...  \
0           232              2.0              1.0                 2.0  ...
1          4128              0.0              2.0                11.0  ...
2           121              0.0              2.0                 2.0  ...
3          2698              1.0              0.0                 5.0  ...
4         12438              0.0              1.0                11.0  ...

        stadium attendance               referee  \
0  Akhmat-Arena    21700.0  Vladislav Bezborodov
1     Metallurg    11400.0         Sergey Ivanov
2  Arena Khimki    12000.0        Sergey Karasev
3     RZD Arena    11408.0        Sergey Karasev
4     Metallurg     7534.0      Timur Arslanbekov

                                                 url  home_club_formation  \
0  https://www.transfermarkt.co.uk/terek-grozny_s...                  NaN
1  https://www.transfermarkt.co.uk/krylya-sovetov...                  NaN
2  https://www.transfermarkt.co.uk/cska-moscow_di...                  NaN
3  https://www.transfermarkt.co.uk/lokomotiv-mosc...                  NaN
4  https://www.transfermarkt.co.uk/krylya-sovetov...                  NaN

  away_club_formation         home_club_name                  away_club_name  \
0                 NaN          Akhmat Grozny                  Spartak Moscow
1                 NaN  Krylya Sovetov Samara                      Amkar Perm
2                 NaN             CSKA Moscow                  Dynamo Moscow
3                 NaN       Lokomotiv Moscow                    Rubin Kazan
4                 NaN  Krylya Sovetov Samara  Volga Nizhniy Novgorod (- 2016)

  aggregate competition_type
0       2:1  domestic_league
1       0:2  domestic_league
2       0:2  domestic_league
3       1:0  domestic_league
4       0:1  domestic_league

[5 rows x 23 columns]

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64293 entries, 0 to 64292
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   game_id                 64293 non-null  int64
 1   competition_id          64293 non-null  object
 2   season                  64293 non-null  int64
 3   round                   64293 non-null  object
 4   date                    64293 non-null  object
 5   home_club_id            64293 non-null  int64
 6   away_club_id            64293 non-null  int64
 7   home_club_goals         64291 non-null  float64
 8   away_club_goals         64291 non-null  float64
 9   home_club_position      45181 non-null  float64
 10  away_club_position      45181 non-null  float64
 11  home_club_manager_name  63554 non-null  object
 12  away_club_manager_name  63554 non-null  object
 13  stadium                 64078 non-null  object
 14  attendance              54810 non-null  float64
 15  referee                 63697 non-null  object
 16  url                     64293 non-null  object
 17  home_club_formation     2091 non-null   object
 18  away_club_formation     2111 non-null   object
 19  home_club_name          53256 non-null  object
 20  away_club_name          54407 non-null  object
 21  aggregate               64291 non-null  object
 22  competition_type        64293 non-null  object
dtypes: float64(5), int64(4), object(14)
memory usage: 11.3+ MB
None

Number of uniques values:

game_id                  64293
competition_id              43
season                      12
round                      116
date                      3362
home_club_id              2384
away_club_id              2122
home_club_goals             16
away_club_goals             18
home_club_position          21
away_club_position          21
home_club_manager_name    4731
away_club_manager_name    4491
stadium                   2253
attendance               26875
referee                   2316
url                      64293
home_club_formation         39
away_club_formation         37
home_club_name             426
away_club_name             426
aggregate                  112
competition_type             4
dtype: int64

Statistical description:

            game_id        season   home_club_id   away_club_id  \
count  6.429300e+04  64293.000000   64293.000000   64293.000000
mean   3.001949e+06   2017.172849    4887.516028    4423.932419
std    5.478832e+05      3.304216   11355.003131   10223.416646
min    2.211607e+06   2012.000000       1.000000       2.000000
25%    2.512848e+06   2014.000000     354.000000     347.000000
50%    2.903799e+06   2017.000000     995.000000     989.000000
75%    3.433394e+06   2020.000000    3205.000000    3026.000000
max    4.196249e+06   2023.000000  112751.000000  112755.000000

       home_club_goals  away_club_goals  home_club_position  \
count     64291.000000     64291.000000        45181.000000
mean          1.592276         1.320107            9.275691
std           1.427064         1.361882            5.300881
min           0.000000         0.000000            1.000000
25%           1.000000         0.000000            5.000000
50%           1.000000         1.000000            9.000000
75%           2.000000         2.000000           14.000000
max          15.000000        19.000000           21.000000

       away_club_position    attendance
count        45181.000000  54810.000000
mean             9.458290  18030.075369
std              5.322696  17734.316215
min              1.000000      1.000000
25%              5.000000   4301.000000
50%              9.000000  12135.000000
75%             14.000000  26107.500000
max             21.000000  99354.000000
```

```
**********************************************************************************************
player_valuations (440663, 9)
**********************************************************************************************

head:

   player_id  last_season             datetime        date    dateweek  \
0       3132         2013  2003-12-09 00:00:00  2003-12-09  2003-12-08
1       6893         2012  2003-12-15 00:00:00  2003-12-15  2003-12-15
2         10         2015  2004-10-04 00:00:00  2004-10-04  2004-10-04
3         26         2017  2004-10-04 00:00:00  2004-10-04  2004-10-04
4         65         2015  2004-10-04 00:00:00  2004-10-04  2004-10-04

   market_value_in_eur  n  current_club_id player_club_domestic_competition_id
0               400000  1              126                                TR1
1               900000  1              984                                GB1
2              7000000  1              398                                IT1
3              1500000  1               16                                 L1
4              8000000  1             1091                                GR1

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440663 entries, 0 to 440662
Data columns (total 9 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   player_id                            440663 non-null  int64
 1   last_season                          440663 non-null  int64
 2   datetime                             440663 non-null  object
 3   date                                 440663 non-null  object
 4   dateweek                             440663 non-null  object
 5   market_value_in_eur                  440663 non-null  int64
 6   n                                    440663 non-null  int64
 7   current_club_id                      440663 non-null  int64
 8   player_club_domestic_competition_id  440663 non-null  object
dtypes: int64(5), object(4)
memory usage: 30.3+ MB
None

Number of uniques values:


player_id                             28794
last_season                              12
datetime                               4817
date                                   4817
dateweek                                972
market_value_in_eur                     423
n                                         1
current_club_id                         424
player_club_domestic_competition_id      14
dtype: int64

Statistical description:

          player_id    last_season  market_value_in_eur         n  \
count  4.406630e+05  440663.000000         4.406630e+05  440663.0
mean   1.964113e+05    2018.762887         2.357557e+06       1.0
std    1.793622e+05       3.624305         6.603356e+06       0.0
min    1.000000e+01    2012.000000         1.000000e+04       1.0
25%    5.532200e+04    2016.000000         2.000000e+05       1.0
50%    1.407480e+05    2019.000000         5.000000e+05       1.0
75%    2.896450e+05    2022.000000         1.600000e+06       1.0
max    1.166093e+06    2023.000000         2.000000e+08       1.0

       current_club_id
count    440663.000000
mean       4041.891491
std        9508.375247
min           3.000000
25%         368.000000
50%        1010.000000
75%        2944.000000
max       83678.000000
```

```
********************************************************************************
players (30302, 23)
********************************************************************************

head:

   player_id first_name   last_name             name  last_season  \
0        598       Timo  Hildebrand   Timo Hildebrand         2014
1        670     Martin      Petrov     Martin Petrov         2012
2       1323     Martin     Amedick    Martin Amedick         2012
3       3195   Jermaine     Pennant  Jermaine Pennant         2013
4       3259     Damien        Duff       Damien Duff         2013

   current_club_id       player_code country_of_birth city_of_birth  \
0               24   timo-hildebrand          Germany         Worms
1              714      martin-petrov         Bulgaria        Vratsa
2               24    martin-amedick          Germany     Paderborn
3              512  jermaine-pennant          England    Nottingham
4              931       damien-duff          Ireland    Ballyboden

  country_of_citizenship  ...   foot height_in_cm market_value_in_eur  \
0                Germany  ...    NaN          NaN                 NaN
1               Bulgaria  ...    NaN          NaN                 NaN
2                Germany  ...    NaN          NaN                 NaN
3                England  ...  right        173.0                 NaN
4                Ireland  ...   left        177.0                 NaN

  highest_market_value_in_eur  contract_expiration_date  agent_name  \
0                  10000000.0                       NaN         NaN
1                  12000000.0                       NaN         IFM
2                   2750000.0                       NaN         NaN
3                  10500000.0                       NaN  Andrew Sky
4                  17000000.0                       NaN         NaN

                                   image_url  \
0  https://img.a.transfermarkt.technology/portrai...
1  https://img.a.transfermarkt.technology/portrai...
2  https://img.a.transfermarkt.technology/portrai...
3  https://img.a.transfermarkt.technology/portrai...
4  https://img.a.transfermarkt.technology/portrai...

                                                 url  \
0  https://www.transfermarkt.co.uk/timo-hildebran...
1  https://www.transfermarkt.co.uk/martin-petrov/...
2  https://www.transfermarkt.co.uk/martin-amedick...
3  https://www.transfermarkt.co.uk/jermaine-penna...
4  https://www.transfermarkt.co.uk/damien-duff/pr...

  current_club_domestic_competition_id       current_club_name
0                                   L1     Eintracht Frankfurt
1                                  ES1  RCD Espanyol Barcelona
2                                   L1     Eintracht Frankfurt
3                                  GB1            Stoke City
4                                  GB1             Fulham FC

[5 rows x 23 columns]

General information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30302 entries, 0 to 30301
Data columns (total 23 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   player_id                             30302 non-null  int64
 1   first_name                            28337 non-null  object
 2   last_name                             30302 non-null  object
 3   name                                  30302 non-null  object
 4   last_season                           30302 non-null  int64
 5   current_club_id                       30302 non-null  int64
 6   player_code                           30302 non-null  object
 7   country_of_birth                      27613 non-null  object
 8   city_of_birth                         28099 non-null  object
 9   country_of_citizenship                29759 non-null  object
 10  date_of_birth                         30255 non-null  object
 11  sub_position                          30130 non-null  object
 12  position                              30302 non-null  object
 13  foot                                  27913 non-null  object
 14  height_in_cm                          28204 non-null  float64
 15  market_value_in_eur                   19383 non-null  float64
 16  highest_market_value_in_eur           28981 non-null  float64
 17  contract_expiration_date              18835 non-null  object
 18  agent_name                            14941 non-null  object
 19  image_url                             30302 non-null  object
 20  url                                   30302 non-null  object
 21  current_club_domestic_competition_id  30302 non-null  object
 22  current_club_name                     30302 non-null  object
dtypes: float64(3), int64(3), object(17)
memory usage: 5.3+ MB
None

Number of uniques values:

player_id                               30302
first_name                               6551
last_name                               22310
name                                    29662
last_season                                12
current_club_id                           424
player_code                             29628
country_of_birth                          184
city_of_birth                            8184
country_of_citizenship                    180
date_of_birth                            8892
sub_position                               13
position                                    5
foot                                        3
height_in_cm                               50
market_value_in_eur                       125
highest_market_value_in_eur               204
contract_expiration_date                   98
agent_name                               2622
image_url                               24683
url                                     30302
current_club_domestic_competition_id       14
current_club_name                         424
dtype: int64

Statistical description:

          player_id   last_season  current_club_id  height_in_cm  \
count  3.030200e+04  30302.000000     30302.000000  28204.000000
mean   3.112814e+05   2018.768926      4366.055574    182.234577
std    2.502577e+05      3.654540     10056.373140      6.833916
min    1.000000e+01   2012.000000         3.000000     18.000000
25%    9.527400e+04   2016.000000       403.000000    178.000000
50%    2.578345e+05   2019.000000      1071.000000    182.000000
75%    4.655942e+05   2022.000000      3008.000000    187.000000
max    1.186012e+06   2023.000000     83678.000000    207.000000

       market_value_in_eur  highest_market_value_in_eur
count         1.938300e+04                 2.898100e+04
mean          2.234721e+06                 3.571396e+06
std           7.340683e+06                 9.352255e+06
min           1.000000e+04                 1.000000e+04
25%           1.750000e+05                 2.500000e+05
50%           3.500000e+05                 7.500000e+05
75%           1.000000e+06                 2.700000e+06
max           1.800000e+08                 2.000000e+08
```

Appendix B

	position	age_group	market_value_in_eur
0	Attack	Ages 22 - 26	3.750507e+06
1	Attack	Ages 27 - 32	2.532845e+06
2	Attack	Ages 33 - 38	7.983708e+05
3	Attack	Ages 39 - 44	4.390385e+05
4	Attack	Ages 45 - 50	NaN
5	Attack	Ages Over 50	NaN
6	Attack	Ages Under 22	2.187830e+06
7	Defender	Ages 22 - 26	3.013702e+06
8	Defender	Ages 27 - 32	2.109870e+06
9	Defender	Ages 33 - 38	5.403169e+05
10	Defender	Ages 39 - 44	2.053261e+05
11	Defender	Ages 45 - 50	NaN
12	Defender	Ages Under 22	1.495578e+06
13	Goalkeeper	Ages 22 - 26	1.093900e+06
14	Goalkeeper	Ages 27 - 32	1.824842e+06
15	Goalkeeper	Ages 33 - 38	6.415633e+05
16	Goalkeeper	Ages 39 - 44	1.860000e+05
17	Goalkeeper	Ages 45 - 50	1.000000e+06
18	Goalkeeper	Ages Over 50	NaN
19	Goalkeeper	Ages Under 22	3.498621e+05
20	Midfield	Ages 22 - 26	3.164518e+06
21	Midfield	Ages 27 - 32	2.646681e+06
22	Midfield	Ages 33 - 38	7.164245e+05
23	Midfield	Ages 39 - 44	1.773148e+05
24	Midfield	Ages 45 - 50	NaN
25	Midfield	Ages Under 22	2.725868e+06

```
def impute_market_value(cols):
    market_value_in_eur = cols[0]
    position = cols[1]
    age_group = cols[2]

    if pd.isnull(market_value_in_eur):

        if position == 'Attack':
            if age_group == 'Ages 22 - 26':
                return 3.761589e+06
            elif age_group == 'Ages 27 - 32':
                return 2.519384e+06
            elif age_group == 'Ages 33 - 38':
                return 7.900107e+05
            elif age_group == 'Ages 39 - 44':
                return 4.437255e+05
            elif age_group == 'Ages Under 22':
                return 2.209577e+06
            else:
                return 2.236328e+06

        elif position == 'Defender':
            if age_group == 'Ages 22 - 26':
                return 2.989714e+06
            elif age_group == 'Ages 27 - 32':
                return 2.109582e+06
            elif age_group == 'Ages 33 - 38':
                return 5.224117e+05
            elif age_group == 'Ages 39 - 44':
                return 2.053261e+05
            elif age_group == 'Ages Under 22':
                return 1.579725e+06
            else:
                return 2.236328e+06

        elif position == 'Goalkeeper':
            if age_group == 'Ages 22 - 26':
                return 1.093724e+06
            elif age_group == 'Ages 27 - 32':
                return 1.836525e+06
            elif age_group == 'Ages 33 - 38':
                return 6.415633e+05
            elif age_group == 'Ages 39 - 44':
                return 1.860000e+05
            elif age_group == 'Ages 45 - 50':
                return 1.000000e+06
            elif age_group == 'Ages Under 22':
                return 3.466327e+05
            else:
                return 2.236328e+06

        elif position == 'Midfield':
            if age_group == 'Ages 22 - 26':
                return 3.168564e+06
            elif age_group == 'Ages 27 - 32':
                return 2.641277e+06
            elif age_group == 'Ages 33 - 38':
                return 7.158922e+05
            elif age_group == 'Ages 39 - 44':
                return 1.768868e+05
            elif age_group == 'Ages Under 22':
                return 2.715097e+06
            else:
                return 2.236328e+06

        else:
            return 2.236328e+06

    else:
        return market_value_in_eur
```

Figure 3.1. Mean values grouped by position and age group (left), Function for imputing mean values (right)

	position	age_group	highest_market_value_in_eur
0	Attack	Ages 22 - 26	4.212937e+06
1	Attack	Ages 27 - 32	4.303414e+06
2	Attack	Ages 33 - 38	4.724629e+06
3	Attack	Ages 39 - 44	5.751636e+06
4	Attack	Ages 45 - 50	8.016406e+06
5	Attack	Ages Over 50	NaN
6	Attack	Ages Under 22	2.352500e+06
7	Defender	Ages 22 - 26	3.203632e+06
8	Defender	Ages 27 - 32	3.291563e+06
9	Defender	Ages 33 - 38	3.308145e+06
10	Defender	Ages 39 - 44	3.818917e+06
11	Defender	Ages 45 - 50	4.756200e+06
12	Defender	Ages Under 22	1.508693e+06
13	Goalkeeper	Ages 22 - 26	1.130562e+06
14	Goalkeeper	Ages 27 - 32	2.276045e+06
15	Goalkeeper	Ages 33 - 38	2.481094e+06
16	Goalkeeper	Ages 39 - 44	2.836383e+06
17	Goalkeeper	Ages 45 - 50	2.665569e+06
18	Goalkeeper	Ages Over 50	2.066667e+06
19	Goalkeeper	Ages Under 22	3.498675e+05
20	Midfield	Ages 22 - 26	3.334781e+06
21	Midfield	Ages 27 - 32	4.029915e+06
22	Midfield	Ages 33 - 38	4.292481e+06
23	Midfield	Ages 39 - 44	4.881241e+06
24	Midfield	Ages 45 - 50	6.363918e+06
25	Midfield	Ages Under 22	2.726000e+06

```python
def impute_highest_market_value(cols):
    highest_market_value_in_eur = cols[0]
    position = cols[1]
    age_group = cols[2]

    if pd.isnull(highest_market_value_in_eur):

        if position == 'Attack':
            if age_group == 'Ages 22 - 26':
                return 4.223364e+06
            elif age_group == 'Ages 27 - 32':
                return 4.297336e+06
            elif age_group == 'Ages 33 - 38':
                return 4.713043e+06
            elif age_group == 'Ages 39 - 44':
                return 5.758007e+06
            elif age_group == 'Ages 45 - 50':
                return 8.016406e+06
            elif age_group == 'Ages Under 22':
                return 2.392311e+06
            else:
                return 3.582088e+06

        elif position == 'Defender':
            if age_group == 'Ages 22 - 26':
                return 3.183876e+06
            elif age_group == 'Ages 27 - 32':
                return 3.314468e+06
            elif age_group == 'Ages 33 - 38':
                return 3.277274e+06
            elif age_group == 'Ages 39 - 44':
                return 3.810223e+06
            elif age_group == 'Ages 45 - 50':
                return 4.756200e+06
            elif age_group == 'Ages Under 22':
                return 1.590434e+06
            else:
                return 3.582088e+06

        elif position == 'Goalkeeper':
            if age_group == 'Ages 22 - 26':
                return 1.129222e+06
            elif age_group == 'Ages 27 - 32':
                return 2.288247e+06
            elif age_group == 'Ages 33 - 38':
                return 2.477691e+06
            elif age_group == 'Ages 39 - 44':
                return 2.843714e+06
            elif age_group == 'Ages 45 - 50':
                return 2.665569e+06
            elif age_group == 'Ages Over 50':
                return 2.066667e+06
            elif age_group == 'Ages Under 22':
                return 3.469281e+05
            else:
                return 3.582088e+06

        elif position == 'Midfield':
            if age_group == 'Ages 22 - 26':
                return 3.344822e+06
            elif age_group == 'Ages 27 - 32':
                return 4.022066e+06
            elif age_group == 'Ages 33 - 38':
                return 4.301016e+06
            elif age_group == 'Ages 39 - 44':
                return 4.890127e+06
            elif age_group == 'Ages 45 - 50':
                return 6.341667e+06
            elif age_group == 'Ages Under 22':
                return 2.716215e+06
            else:
                return 3.582088e+06

        else:
            return 3.582088e+06

    else:
        return highest_market_value_in_eur
```

```python
# Calculating the mean market values based on position and age category
mean_height_values = df3.groupby(['position', 'sub_position'])['height_in_cm'].mean().reset_index()

# Displaying the mean values
print(mean_height_values)
```

```
      position        sub_position  height_in_cm
0       Attack      Centre-Forward    184.554187
1       Attack         Left Winger    176.814056
2       Attack        Right Winger    177.224924
3       Attack      Second Striker    178.676471
4     Defender         Centre-Back    187.966463
5     Defender           Left-Back    179.100575
6     Defender          Right-Back    179.516129
7   Goalkeeper          Goalkeeper    190.461938
8     Midfield  Attacking Midfield    177.996979
9     Midfield    Central Midfield    179.848057
10    Midfield  Defensive Midfield    182.069987
11    Midfield       Left Midfield    178.350649
12    Midfield      Right Midfield    177.253333
```

```python
def impute_height(cols):
    height_in_cm = cols[0]
    position = cols[1]
    sub_position = cols[2]

    if pd.isnull(height_in_cm):

        if position == 'Attack':
            if sub_position == 'Centre-Forward':
                return 184.554187
            elif sub_position == 'Left Winger':
                return 176.814056
            elif sub_position == 'Right Winger':
                return 177.224924
            elif sub_position == 'Second Striker':
                return 178.676471
            else:
                return 182.666527

        elif position == 'Defender':
            if sub_position == 'Centre-Back':
                return 187.966463
            elif sub_position == 'Left-Back':
                return 179.100575
            elif sub_position == 'Right-Back':
                return 179.516129
            else:
                return 182.666527

        elif position == 'Goalkeeper':
            if sub_position == 'Goalkeeper':
                return 190.461938
            else:
                return 182.666527

        elif position == 'Midfield':
            if sub_position == 'Attacking Midfield':
                return 177.996979
            elif sub_position == 'Central Midfield':
                return 179.848057
            elif sub_position == 'Defensive Midfield':
                return 182.069987
            elif sub_position == 'Left Midfield':
                return 178.350649
            elif sub_position == 'Right Midfield':
                return 177.253333
            else:
                return 182.666527

        else:
            return 182.666527

    else:
        return height_in_cm
```

```
# Calculating the mode for foot values based on position and sub_position
mode_foot_values = df4.groupby(['position', 'sub_position'])['foot'].agg(lambda x: x.mode().iloc[0] if not x.mode().empty
# Displaying the mode values
print(mode_foot_values)
```

```
      position        sub_position   foot
0       Attack      Centre-Forward  right
1       Attack         Left Winger  right
2       Attack        Right Winger  right
3       Attack      Second Striker  right
4     Defender         Centre-Back  right
5     Defender           Left-Back   left
6     Defender          Right-Back  right
7   Goalkeeper          Goalkeeper  right
8     Midfield  Attacking Midfield  right
9     Midfield    Central Midfield  right
10    Midfield  Defensive Midfield  right
11    Midfield       Left Midfield   left
12    Midfield      Right Midfield  right
```

In [150]:

```
# Writing a function of conditional statements for imputing the mode for foot based on positions
def impute_foot(cols):
    foot = cols[0]
    position = cols[1]
    sub_position = cols[2]

    if pd.isnull(foot):
        if position == 'Attack':
            return 'right'

        elif position == 'Defender':
            if sub_position == 'Left-Back':
                return 'left'
            else:
                return 'right'

        elif position == 'Goalkeeper':
            return 'right'

        elif position == 'Midfield':
            if sub_position == 'Left Midfield':
                return 'left'
            else:
                return 'right'

        else:
            return 'right'

    else:
        return foot
```

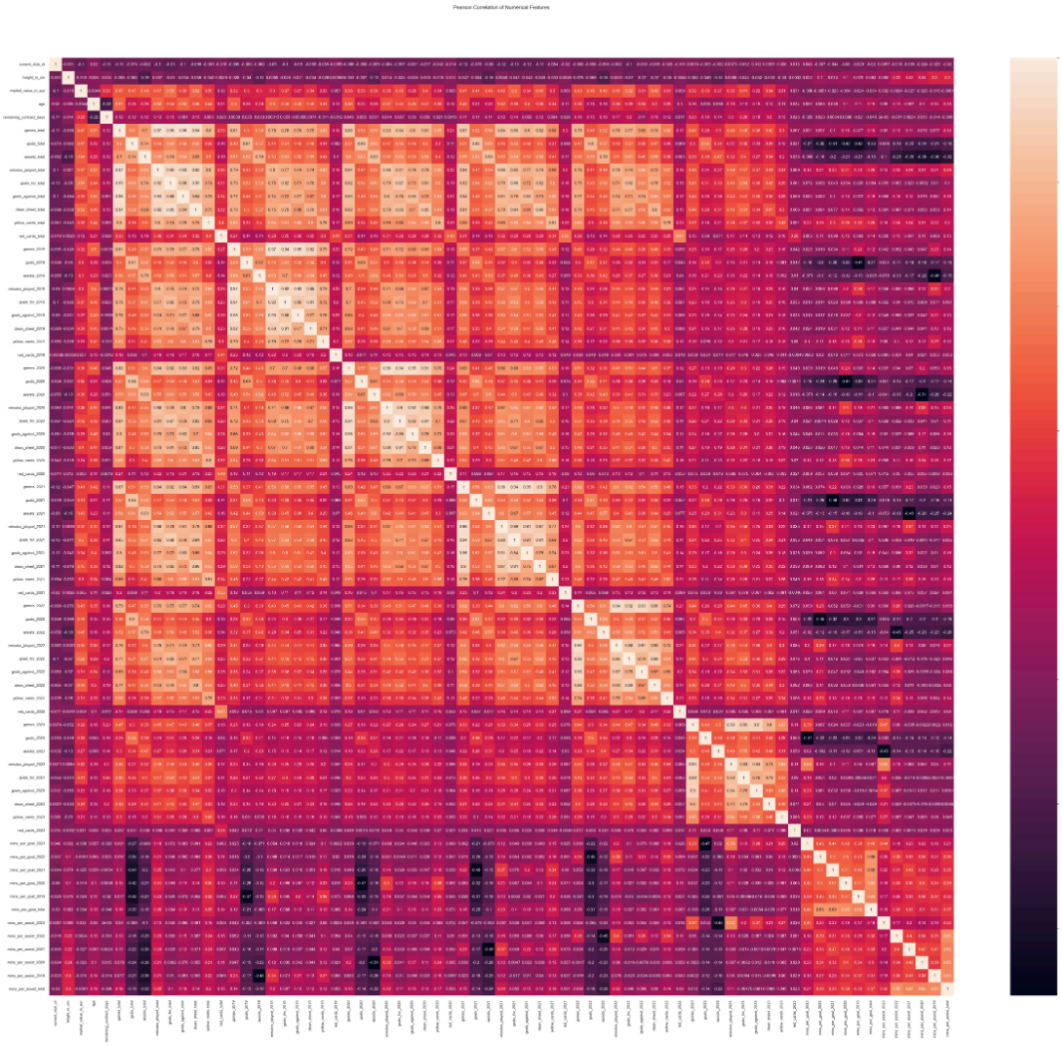
In [151]:

```
# Applying the impute_height function to the dataframe
df4['foot'] = df4[['foot', 'position', 'sub_position']].apply(impute_foot, axis=1)
```

Appendix C

market_value_in_eur	1.000000
games_total	0.470027
goals_total	0.467667
assists_total	0.486171
minutes_played_total	0.471368
goals_for_total	0.545848
clean_sheet_total	0.519063
goals_for_2020	0.422572
games_2021	0.412321
goals_2021	0.413660
assists_2021	0.427364
minutes_played_2021	0.416458
goals_for_2021	0.495140
clean_sheet_2021	0.447354
games_2022	0.434351
goals_2022	0.457946
assists_2022	0.452212
minutes_played_2022	0.446138
goals_for_2022	0.518047
clean_sheet_2022	0.488459
goals_for_2023	0.406962

Figure 3.2. Results for Pearson Correlation on Numerical Features



Creating a feature for Age

In [53]:
```
# Calculating the age of each player
players_df['date_of_birth'] = pd.to_datetime(players_df['date_of_birth'])
# Instantiating an object now for the current datetime
now = datetime.now()
# Creating a feature for age based on the difference between the current datetime and their birth datetime
players_df['age'] = (now - players_df['date_of_birth']).apply(lambda x: x.days) / 365.25
# rounding up their age and casting to integer type
players_df['age'] = players_df['age'].round().astype(int)
```

Pearsons Correlation between All Features

In [255]:
```
cor = df8_dummies_all.corr()
```

In [256]:
```
# Correlation with target variable
cor_target = abs(cor['market_value_in_eur'])

# Selecting highly correlated features
relevant_features = cor_target[cor_target>0.1]
relevant_features
```

Out[256]:
```
current_club_id                          0.104776
market_value_in_eur                      1.000000
remaining_contract_days                  0.334212
games_total                              0.470027
goals_total                              0.467667
assists_total                            0.486171
minutes_played_total                     0.471368
goals_for_total                          0.545848
goals_against_total                      0.392081
clean_sheet_total                        0.519063
yellow_cards_total                       0.352771
red_cards_total                          0.115185
games_2019                               0.319025
goals_2019                               0.297224
assists_2019                             0.303601
minutes_played_2019                      0.304588
goals_for_2019                           0.365819
goals_against_2019                       0.268493
clean_sheet_2019                         0.335958
yellow_cards_2019                        0.225521
games_2020                               0.361509
goals_2020                               0.349266
assists_2020                             0.359543
minutes_played_2020                      0.357576
goals_for_2020                           0.422572
goals_against_2020                       0.291681
clean_sheet_2020                         0.395995
yellow_cards_2020                        0.238545
games_2021                               0.412321
goals_2021                               0.413660
assists_2021                             0.427364
minutes_played_2021                      0.416458
goals_for_2021                           0.495140
goals_against_2021                       0.338094
clean_sheet_2021                         0.447354
yellow_cards_2021                        0.296076
games_2022                               0.434351
goals_2022                               0.457946
assists_2022                             0.452212
minutes_played_2022                      0.446138
goals_for_2022                           0.518047
goals_against_2022                       0.312198
clean_sheet_2022                         0.488459
yellow_cards_2022                        0.314594
games_2023                               0.319920
goals_2023                               0.329007
assists_2023                             0.273764
minutes_played_2023                      0.324534
goals_for_2023                           0.406962
goals_against_2023                       0.223611
clean_sheet_2023                         0.338500
yellow_cards_2023                        0.206615
last_season_2022                         0.195083
last_season_2023                         0.195083
current_club_domestic_competition_id_GB1 0.320959
current_club_name_Arsenal FC             0.160583
current_club_name_Bayern Munich          0.141541
current_club_name_Chelsea FC             0.148239
current_club_name_FC Barcelona           0.128979
current_club_name_Liverpool FC           0.138185
current_club_name_Manchester City        0.210800
current_club_name_Manchester United      0.118698
current_club_name_Paris Saint-Germain    0.172725
current_club_name_Real Madrid            0.159904
current_club_name_Tottenham Hotspur      0.106932
Name: market_value_in_eur, dtype: float64
```

Again, most categorical variables do not seem to have a high correlation with the target variable. The current domestic competition and playing for one of the larger clubs does however influence market value.

Appendix D

Hyperparameters Tuning

In [157]:

```python
# define the hyperparameter space
param_distributions = {
    'alpha': [0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'gamma': [0.01, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0],
    'degree': [1, 2, 3, 4, 5],
    'coef0': [1, 2, 3, 4, 5]
}

# define the number of iterations and cross-validation folds
n_iter = 50
cv = 5

# create the random search object
random_search = RandomizedSearchCV(
    kr,
    param_distributions=param_distributions,
    n_iter=n_iter,
    scoring='neg_mean_absolute_error',
    n_jobs=-1,
    cv=cv,
    random_state=42
)


# fit the random search on the data
random_search.fit(X, y)

# print the best parameters and score
print('Best parameters:', random_search.best_params_)
print('\nBest score:', random_search.best_score_)
```

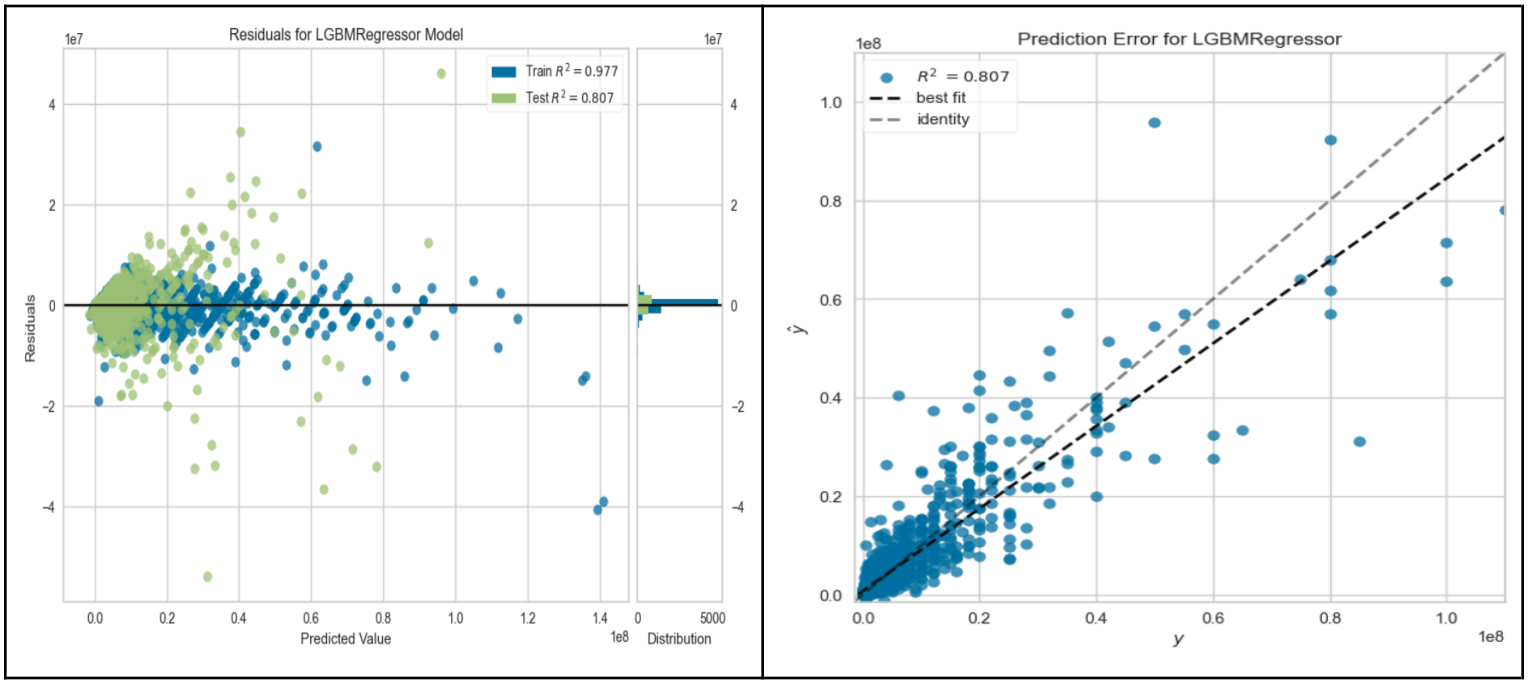
```
Best parameters: {'kernel': 'poly', 'gamma': 0.01, 'degree': 3, 'coef0': 5, 'alpha': 2.0}

Best score: -0.9375677025819453
```

Hyperparameter Tuning

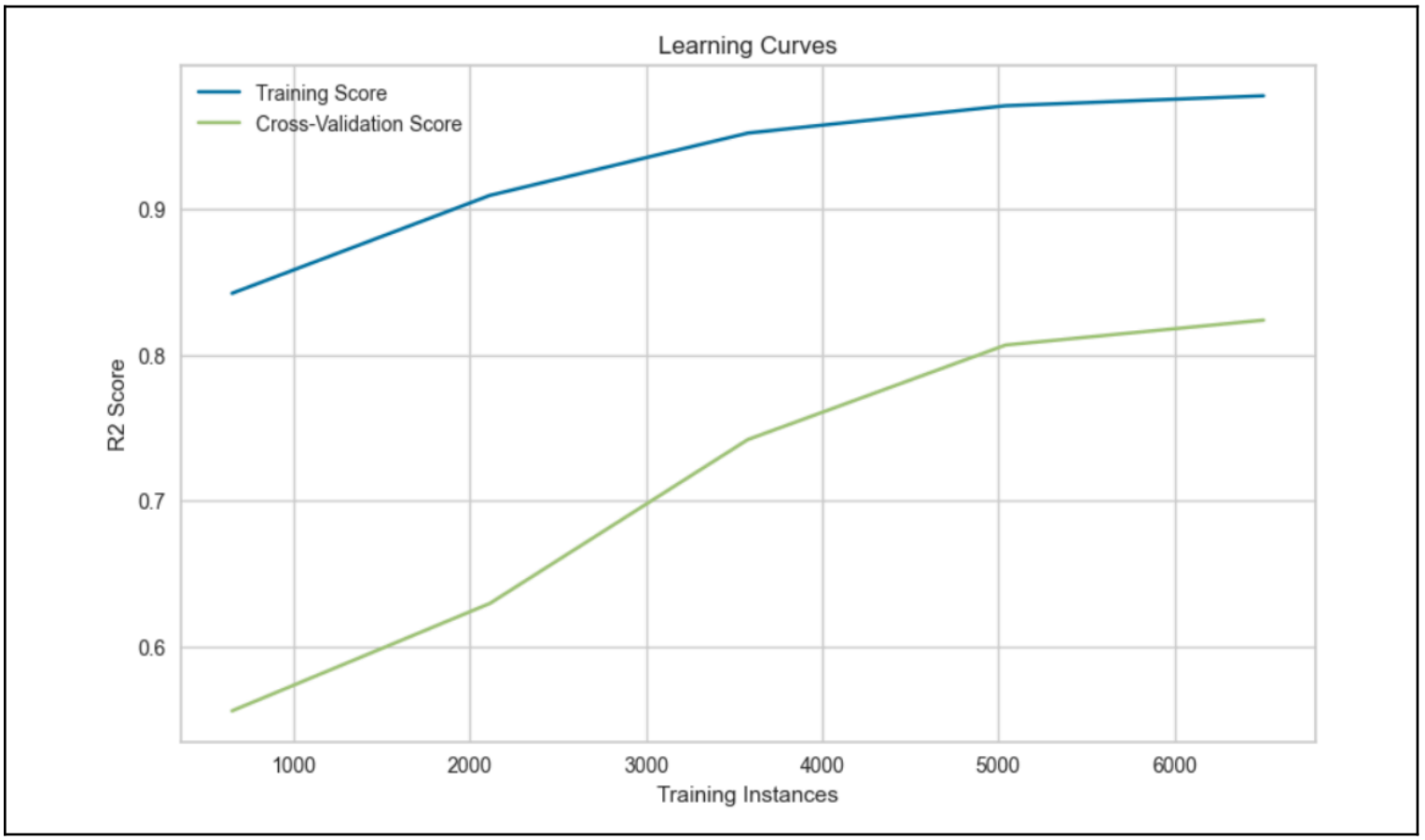
In [352]:

```python
# define the hyperparameter space
param_distributions = {
    'n_estimators': [20, 50, 70, 100, 200, 400, 600],
    'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3, 0.4],
    'max_depth': [1, 2, 3, 4, 5, 6, 7, 8],
    'min_samples_split': [2, 4, 6, 8, 10, 12, 15],
    'min_samples_leaf': [1, 2, 3, 4, 5, 6],
    'n_iter_no_change': [1, 2, 3, 4, 5, 6],
    'validation_fraction': [0.1, 0.2],
    'max_features': ['sqrt', 'log2'],
    'subsample': [0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
}

# define the number of iterations and cross-validation folds
n_iter = 50
cv = 5

# create the random search object
random_search = RandomizedSearchCV(
    gbr,
    param_distributions=param_distributions,
    n_iter=n_iter,
    scoring='neg_mean_absolute_error',
    n_jobs=-1,
    cv=cv,
    random_state=42
)


# fit the random search on the data
random_search.fit(X, y)

# print the best parameters and score
print('Best parameters:', random_search.best_params_)
print('\nBest score:', random_search.best_score_)
```

```
Best parameters: {'validation_fraction': 0.2, 'subsample': 1.0, 'n_iter_no_change': 6, 'n_estimators': 20, 'min_samples_spli
t': 8, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'max_depth': 6, 'learning_rate': 0.2}

Best score: -0.01303734965804074
```



Residuals and Prediction Error plots for LightGBM(prior to log transformation)



Learning curve for LightGBM Model (prior to log transformation)

References

Agnellutti, C. (2014) Big Data: an Exploration of Opportunities, Values, and Privacy Issues. New York: Nova Science Publishers, Inc (Internet Theory, Technology and Applications).
Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e020mww&AN=811106&site=eds-live

Ahmed, T. (n.d.). CCT College Dublin: Log in to the site. [online] moodle.cct.ie.
Available at: https://moodle.cct.ie/course/view.php?id=2637 [Accessed 20 Dec. 2023].

Analytics Vidhya. (2020). KNNImputer | Way To Impute Missing Values. [online]
Available at:
https://www.analyticsvidhya.com/blog/2020/07/knnimputer-a-robust-way-to-impute-missing-values-using-scikit-learn/.

Anwar, A. (2021). A Beginner's Guide to Regression Analysis in Machine Learning. [online] Medium.
Available at: https://towardsdatascience.com/a-beginners-guide-to-regression-analysis-in-machine-learning-8a828b491bbf.

ASQ (2019). What are Histograms? Analysis & Frequency Distribution | ASQ. [online] Asq.org.
Available at: https://asq.org/quality-resources/histogram.

Brownlee, J. (2019a). A Gentle Introduction to Learning Curves for Diagnosing Machine Learning Model Performance. [online] Machine Learning Mastery.
Available at: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

Brownlee, J. (2019b). Overfitting and Underfitting With Machine Learning Algorithms. [online] Machine Learning Mastery.
Available at: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

Campbell, F. (2019). Data Scraping – Considering the Privacy Issues. [online] Fieldfisher.
Available at:
https://www.fieldfisher.com/en/services/privacy-security-and-information/privacy-security-and-information-law-blog/data-scraping-considering-the-privacy-issues.

Codecademy. (n.d.). Seaborn Styling, Part 2: Color. [online]
Available at: https://www.codecademy.com/article/seaborn-design-ii.

Cordeiro, L. (n.d.). Market Value EDA 📈⚽💸. [online] kaggle.com.
Available at: https://www.kaggle.com/code/luisgasparcordeiro/market-value-eda/notebook [Accessed 22 Dec. 2023].

Coxen, D. (n.d.). Football Transfer Market EDA & Basic Modelling. [online] kaggle.com.
Available at: https://www.kaggle.com/code/davidcoxon/football-transfer-market-eda-basic-modelling [Accessed 19 Dec. 2023].

Depken, C.A. and Globan, T. (2020). Football Transfer-Fee Premiums and Europe's Big Five: Online Appendix. SSRN Electronic Journal. doi:https://doi.org/10.2139/ssrn.3617260.

docs.aws.amazon.com. (n.d.). Model Fit: Underfitting vs. Overfitting - Amazon Machine Learning. [online] Available at: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html.

Garza, J. (2024). Class Lecture: Cars Project. [online] accounts.google.com. Available at: https://drive.google.com/drive/folders/1JHFnYE0FJ-OAAJZIJbmH2uezZBjCERgC.

GeeksforGeeks. (2023). Regression using CatBoost. [online] Available at: https://www.geeksforgeeks.org/regression-using-catboost/.

Geurts, P., Ernst, D. and Wehenkel, L. (2006). Extremely randomized trees. Machine Learning, [online] 63(1), pp.3–42. doi:https://doi.org/10.1007/s10994-006-6226-1.

Gitbook.io. (2023). PyCaret 3.0 | Docs. [online] Available at: https://pycaret.gitbook.io/docs.

Gohar, U. (2020). How to use Residual Plots for regression model validation? [online] Medium. Available at: https://towardsdatascience.com/how-to-use-residual-plots-for-regression-model-validation-c3c70e8ab378#:~:text=Residuals.

Goundar, S. (2021). Enterprise systems and technological convergence : research and practice. [online] Charlotte, NC: https://eds.p.ebscohost.com/eds/ebookviewer/ebook/ZTAyMG13d19fMjczMDYwNF9fQU41?sid=997bf805-38e3-47f1-bfb3-813e10171b3f@redis&vid=3&format=EB

Hali, B. (2019). Understanding Bias-Variance Trade-Off in 3 Minutes. [online] Medium. Available at: https://towardsdatascience.com/understanding-bias-variance-trade-off-in-3-minutes-c516cb013513.

Indeed Career Guide. (n.d.). 13 Types of Regression Analysis (Plus When To Use Them). [online] Available at: https://www.indeed.com/career-advice/career-development/regression-types.

Kalyvas, V. (2024). Predicting Car Prices with Machine Learning (Step 1 — Analysis). [online] Medium. Available at: https://python.plainenglish.io/predicting-car-prices-with-machine-learning-step-1-the-analysis-76024196f92b [Accessed 3 May 2024].

Marsja, E. (2020). How to use Square Root, log, & Box-Cox Transformation in Python. [online] Erik Marsja. Available at: https://www.marsja.se/transform-skewed-data-using-square-root-log-box-cox-methods-in-python/.

Matplotlib (2012). Matplotlib: Python plotting — Matplotlib 3.1.1 documentation. [online] Matplotlib.org. Available at: https://matplotlib.org/.

McQuaid, D. (n.d.). Week 1 Overview. [online] moodle.cct.ie. Available at: https://moodle.cct.ie/course/view.php?id=80.

Metelski, A. (2021). Factors affecting the value of football players in the transfer market. Journal of Physical Education and

Sport, 21(2). doi:https://doi.org/10.7752/jpes.2021.s2145.

Müller, A.C. and Guido, S. (2017). Introduction to machine learning with Python : a guide for data scientists. Beijing: O'reilly.

Nalcin, S. (2022). StandardScaler vs. MinMaxScaler vs. RobustScaler: Which one to use for your next ML project? [online]

OpenAi (2022). ChatGPT . [online] chatgpt.com. Available at: https://chatgpt.com.

Medium. Available at: https://medium.com/@onersarpnalcin/standardscaler-vs-minmaxscaler-vs-robustscaler-which-one-to-use-for-your-next-ml-project-ae5b44f571b9.

Niu, Y., Ying, L., Yang, J., Bao, M. and Sivaparthipan, C.B. (2021). Organisational business intelligence and decisionmaking using big data analytics. Information Processing & Management, [online] 58(6), p.102725. doi:https://doi.org/10.1016/j.ipm.2021.102725.

Pannucci, C.J. and Wilkins, E.G. (2011). Identifying and Avoiding Bias in Research. Plastic and Reconstructive Surgery, [online] 126(2), pp.619–625. doi:https://doi.org/10.1097/prs.0b013e3181de24bc.

Patak, K. (2023). Strategic Analysis of Emerging Technology for Competitive Advantage: Artificial Intelligence in the Football Industry. [online] Available at: https://drive.google.com/file/d/1g7y_U2cSYDq771WJZoka4AvgIkdOvIW7/view?usp=sharing [Accessed 27 Oct. 2023].

Persson, J. and Sjöö, E. (2017). Business Intelligence - its impact on the decision making process at higher education institutions

qualtrics (n.d.). What is ANOVA (Analysis Of Variance). [online] Qualtrics. Available at: https://www.qualtrics.com/uk/experience-management/research/anova/#:~:text=ANOVA%20stands%20for%20Analysis%20of.

Rahul Kumar (2019). Machine learning quick reference : quick and essential machine learning hacks for training smart data models. Packt Uuuu-Uuuu.

Readthedocs.io. (2024). Catboost tutorial — SHAP latest documentation. [online] Available at: https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/tree_based_models/Catboost%20tutorial.html

Saxena, S. (2020). Everything you Should Know about p-value from Scratch for Data Science. [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/everything-you-should-know-about-p-value-from-scratch-for-data-science-f3c0bfa3c4cc.

Scikit-learn.org. (2018). Compare the effect of different scalers on data with outliers — scikit-learn 0.20.3 documentation. [online] Available at: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html .

Scikit-learn.org. (2019). 1. Supervised learning — scikit-learn 0.21.3 documentation. [online] Available at: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning.

Simplilearn.com. (n.d.). The Complete Guide on Overfitting and Underfitting in Machine Learning. [online] Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting.

Solon Barocas and Selbst, A.D. (2015). Big data's disparate impact. Ssrn Elibrary.

Statistics Solutions. (n.d.). ANOVA. [online] Available at: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/anova/#:~:text=ANOVA%20assumes%20that%20the%20data.

Suresh Kumar Mukhiya and Ahmed, U. (2020). Hands-on exploratory data analysis with Python : perform EDA techniques to understand, summarize, and investigate your data smartly. Birmingham: Packt Publishing.

Technology, T. (2023). Light GBM Light and Powerful Gradient Boost Algorithm. [online] Medium. Available at: https://medium.com/@turkishtechnology/light-gbm-light-and-powerful-gradient-boost-algorithm-eaa1e804eca8.

Van den Berg, E. (2011). The Valuation of Human Capital in the Football Player Transfer Market. [online] Available at: Link

wiki.creativecommons.org. (n.d.). CC0 FAQ - Creative Commons. [online] Available at: https://wiki.creativecommons.org/wiki/CC0_FAQ.

Wirth, R. and Hipp, J. (2000). CRISP-DM: Towards a Standard Process Model for Data Mining . Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining, (Vol. 1, pp. 29-39).

World Population Review (2022). Most Popular Sport by Country 2020. [online] worldpopulationreview.com. Available at: https://worldpopulationreview.com/country-rankings/most-popular-sport-by-country

www.footballbenchmark.com. (n.d.). Football Benchmark - Methodology and limitations of published information. [online] Available at: https://www.footballbenchmark.com/methodology/player_valuation.

www.kaggle.com. (n.d.). Football Data from Transfermarkt. [online] Available at: https://www.kaggle.com/datasets/davidcariboo/player-scores/versions/284/data [Accessed 29 Oct. 2023].

www.scikit-yb.org. (n.d.). Residuals Plot — Yellowbrick v1.5 documentation. [online] Available at: https://www.scikit-yb.org/en/latest/api/regressor/residuals.html#:~:text=The%20residuals%20plot%20shows%20the.

www.transfermarkt.co.uk. (n.d.). Aïssa Boudechicha - Player profile 23/24. [online] Available at: https://www.transfermarkt.co.uk/aissa-boudechicha/profil/spieler/592398 [Accessed 16 Dec. 2023].

www.transfermarkt.co.uk. (n.d.). Genar Fornés - Player profile 23/24. [online] Available at: https://www.transfermarkt.co.uk/genar-fornes/profil/spieler/628490 [Accessed 5 Dec. 2023].