**HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**
**SCHOOL OF INFORMATION TECHNOLOGY AND COMMUNICATION**

---



# MACHINE LEARNING PROJECT REPORT

## Project name: Prediction of football player's value

***Lecturer: Assoc. Prof . Than Quang Khoat***

***Students:***
***Nguyen Nho Trung – 20204894***
***Ho Minh Khoi – 20204917***
***Truong Quang Binh – 20200068***

***Group: 14***
***Class: IT3190E - Machine Learning - 131679***

# Table of contents

**1. Introduction** **2**

**2. Datasets** **2**

2.1. Integration 2

2.2. Data cleaning process 3

2.3. Exploratory data analysis 4

2.3.1. Players' value 4

2.3.2. Position 6

2.3.2. Age 7

2.3.4. Correlation between independent variables and target variables (Dataset 2) 8

**3. Metric for evaluation of models** **9**

**4. Choosing data models & result** **9**

4.1. For dataset 1 9

4.4.1. Preprocessing 9

4.1.2. Applied models 10

4.1.3. Experimental results 10

4.2. For dataset 2 10

4.2.1. Lasso - Ridge - KNN 10

4.2.1.1. Preprocessing 10

4.2.1.2. Experimental results 11

4.2.2. Decision Tree – Random Forest – AdaBoost 11

4.2.2.1. Preprocessing 11

4.2.2.2. Experimental results 11

4.2.3. Artificial Neural Network 11

4.2.3.1. Layers of Artificial Neural-network 11

4.2.3.2. Algorithm for ANN 12

4.2.3.3. Experimental results 12

4.2.4. Gradient Boosting: XGBoost and LightGBM 13

4.2.4.1. Theory of gradient boosting and its variations 13

4.2.4.1.1. XGBoost 14

4.2.4.1.2. LightGBM 14

4.2.4.2. Preprocessing 15

4.2.4.3.Experimental results 15

4.2.5. Stacking 16

4.2.5.1. Preprocessing 16

4.2.5.2.Experimental results 16

**5. Summary of results and analysis with best model** **16**

5.1. Summary of results 16

5.2. Analysis with best model 18

**6. Limitations and Future extensions** **18**

**7. References** **19**

# 1. Introduction

Football, or soccer, is one of the most well-known sports in the world. There is no doubt that football is regarded as the "king of sports". According to The Irish Times, English Premier League broadcasting is valued at £1.6 billion per season, and many more football leagues are making billions of dollars of money annually. The high numbers were also reported by other leagues. However, for clubs, the game isn't completely fair: large clubs always get the larger piece of cake, thus, being able to reinvest and remain their throne, while small clubs often struggle to do so, and in the worst case, get relegated and get into deeper debt. Therefore, instead of waiting for miracles to happen, like the story of Leicester City in 2016, many small to average clubs have been trying to make use of the transfer market. One of the most popular stories is EPL club Brentford's Moneyball strategy, where they try to buy undervalued talents from a market that is less inflated than England.

So how did Brentford find out those players? Having the same owner as a betting company, they were provided statistical research that helps them analyze players that fit the club's philosophy. Moreover, the transfer market in Europe takes place every year, and while there are successful cases such as Brentford, many transfers also can be considered failures. Harry Maguire, Eden Hazard, Jack Grealish, and many other players contribute to their clubs fewer than what they have spent to sign them. Therefore, evaluation of player values is very important for team managers, especially for target players that they want to sign. This project aims to help managers estimate player prices based on player statistics and their performance in the past, and particularly the 2021-2022 season. Our project is applying Machine Learning methods to predict the price of football players who are playing in the top 5 leagues in Europe: Premier League, La Liga, Bundesliga, Serie A and Ligue 1.

# 2. Datasets

## 2.1. Integration

We are going to work on 2 datasets for this project, each contains sub-datasets from different sources, and those sub-datasets are integrated:

- First dataset: Value of players that are currently playing this season throughout their careers. It is the combination of 2 sub-datasets
    + General data of players' transfer value from past seasons, crawled from Transfermarkt, source: Kaggle at https://www.kaggle.com/datasets/davidcariboo/player-scores
    + Players' performance history, crawled using Selenium on Whoscored at https://www.whoscored.com/
    + The final dataset contains 13891 indices with 20 columns, other than players' basic information such as name, current club, age, position and preferred foot, etc. it also contains their seasonal performance: appearances, minutes played, number of goals, assists, cards, man of the match (MotM) awards, and our target variable: market value.
- Second dataset: Values of players in 2021-2022 season. It is also combined by 2 sub-datasets
    + Football statistics from FIFA 22, one of the most famous football video games, crawled using Scrapy on SoFIFA at https://sofifa.com/
    + Performance of players in 2021-2022 season, available on Kaggle at https://www.kaggle.com/datasets/vivovinco/20212022-football-player-stats
    + The dataset contains 2002 records with 52 columns, other than the indices with similar meaning to the first dataset, it also contains more attributes of the player, such as volleys, vision, strength, standing_tackle, stamina, Sprint_Speed, Sliding_Tackle, Skill_Moves, Shot_Power, Short_Passing, Reactions, Positioning, Penalties, Marking, Long_Shots, Long_Passing, Jumping, International_Reputation, Interceptions,

Heading_Accuracy, GK_Reflexes, GK_Positioning, GK_Kicking, GK_Handling, GK_Diving, Finishing, FK_Accuracy, Dribbling, Curve, Crossing, Composure, Ball_Control, Balance, Agility, Aggression, Acceleration. These attributes are FIFA 22 in-game attributes of the players.

## 2.2. Data cleaning process

For the first dataset:

- First sub-dataset from Kaggle:
  - + Firstly, we need to remove irrelevant data, keeping only players currently playing in the top 5 leagues.
  - + Since the Kaggle dataset is splitted into 7 tables, and our necessary data consists of 4 of them, we need to concatenate those 4.
  - + The value of each player is evaluated multiple times per year, thus we need to filter the value measured at the end of each season or summer break (usually from May to September).
- Second sub-dataset crawled on Whoscored:
  - + Since the players' statistics we crawled are divided into competitions, we need to sum them up into annual data.
  - + Format of the columns are string, so firstly they need to be numerically converted and replace null values by 0.
  - + Rating of the players are recalculated as average rating per match. Rating value is based on in-match performance of the player. Starting at 6.0, if a player performs a "good" action, for example scoring a goal, the rating increases and vice versa.
- Finally the sub-datasets above are merged into 'data_table.csv', and we delete duplicate values and null values. Since the name of the players are denoted differently on those sub-datasets, we linked them by simplified form of name. For example José Sá, or his formatted name Jose Sa are referenced as jose-sa.

For the second dataset:

- Firstly, we removed duplicate records in 2 sub-datasets: 136 were removed from Kaggle sub-dataset and 2 from the SoFIFA one.
- Name of the players on 2 sub-datasets are connected by similar tokens (2 similar words per name)
- Then, we merge the two tables and remove missing values (0 were removed)
- Each player will be assigned to their best position, so that 'SUB': substitution players and 'RES': reserved players will have a proper position.
- Sub-positions are also converted into main positions. For example, a LWB or Left Wing-back is converted into a "defender". 4 main positions are available: "striker", "midfielder", "defender" and "goalkeeper".
- For value and wage columns: because these columns contain special characters, we need to convert it into numerical.

| value | wage |
|---|---|
| €3.6M | €23K |

=> Value: 3600000
Wage: 23000

- Finally, we removed redundant columns that are not relevant in the predictive model and imported the dataframe into 'processed data.csv'. This data set has 2002 records

## 2.3. Exploratory data analysis

### 2.3.1. Players' value

Figure 1 and 2 describe the distribution and boxplot of market value column:

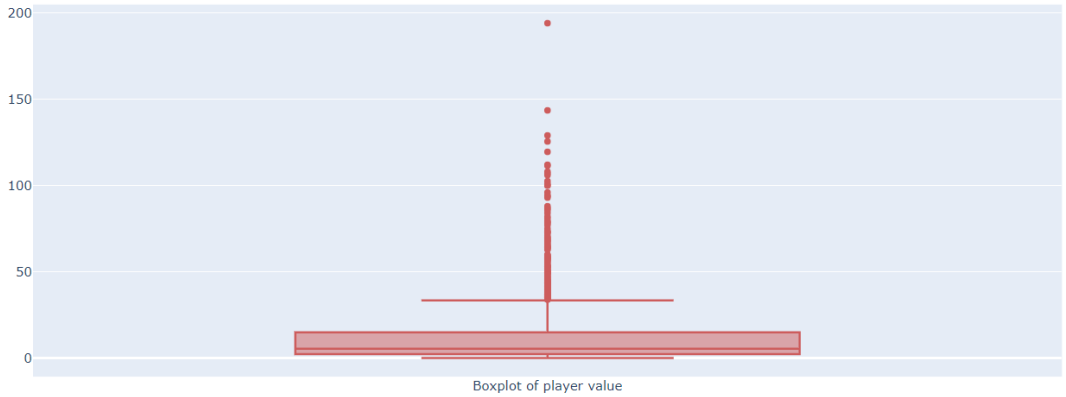
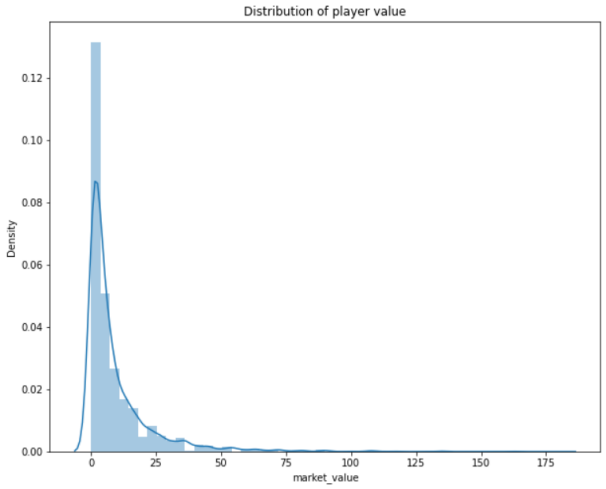
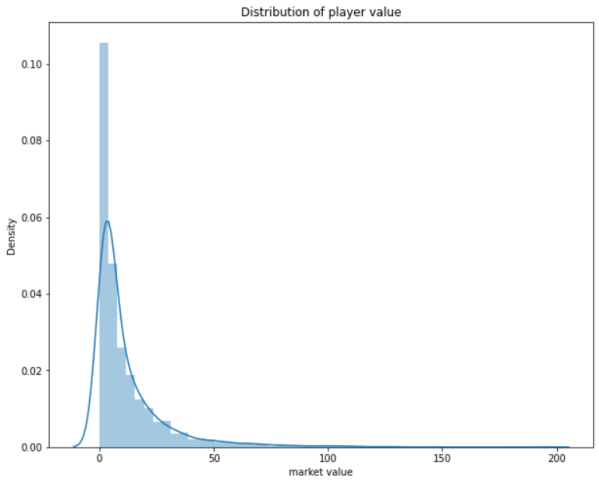



**Figure 1. Boxplot of player value on Dataset 2**




| | |
|---|---|
| mean | 9.342428 |
| std | 13.982631 |




| | |
|---|---|
| mean | 12.013012 |
| std | 17.195709 |

**Figure 2. Distribution of market value on Dataset 1 and Dataset 2, respectively**

We can see that: the distribution of player value is very positively skewed. It makes predicting high priced players very difficult. Compared to the highest player value, the lowest value nearly means nothing. Also, the standard deviation is even larger than the mean. Moreover, we see that the data has

many outliers because most of the players are of low value. Thus, we decided to use log transformation from the target variable to reduce variance in this column.
Figure 3 and 4 represent the result of log transformation to our data.

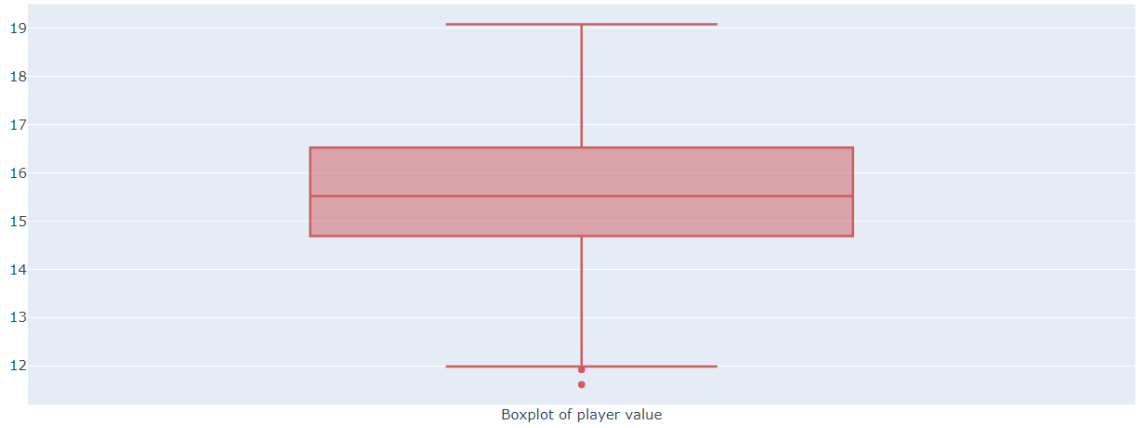



**Figure 3. Boxplot of player value on both datasets - post log transfer**

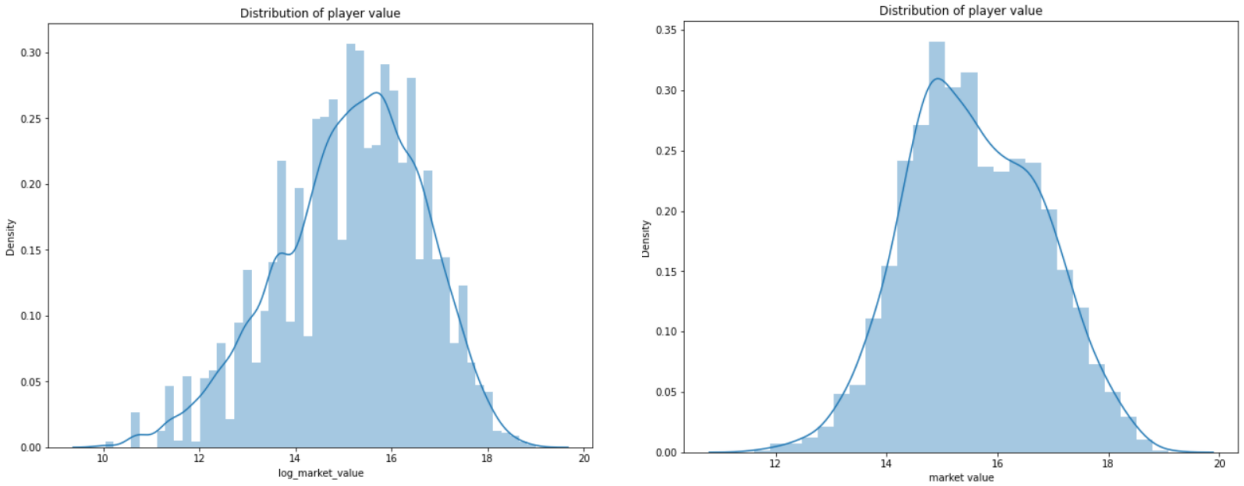



**Figure 4. Distribution of log transformed market value on Dataset 1 and Dataset 2, respectively**

From there, we can observe that the market log value variable is more normally distributed and has a much lower standard deviation compared to the mean. The value on dataset 1 seems to be more sparse, suggesting a better prediction on dataset 2. Nonetheless, with this technique, we also reduced the number of outliers to 2. Therefore, we decided to train our model with market log value as a target variable.

## 2.3.2. Position

A slight difference between the percentage of player positions is portrayed in figure 5.

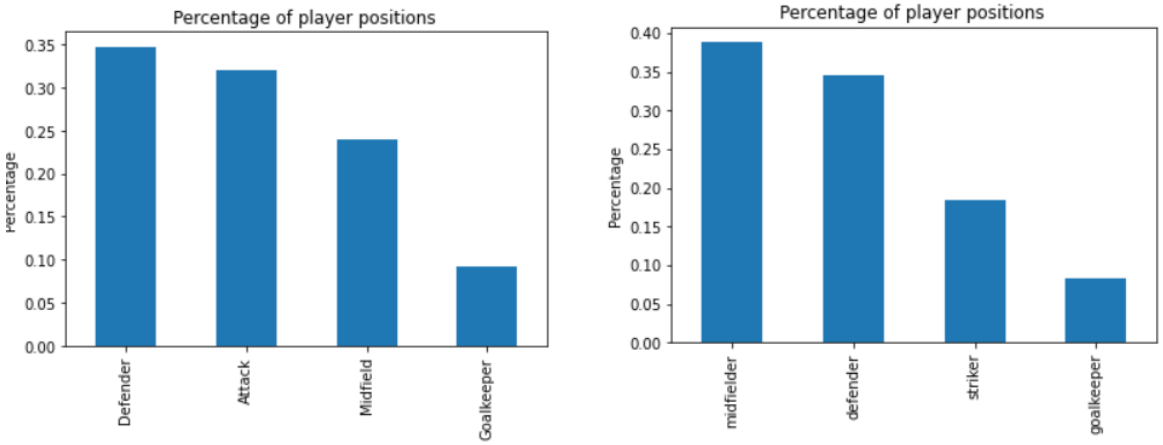



**Figure 5. The percentage of player positions in Dataset 1 and 2, respectively**

We can see that the number of Goalkeepers is always the smallest, because there's only 1 Goalkeeper present on the field at a time, and 2-3 reserved ones per team. In contrast, more than 30% of player data belongs to defenders since for most formations, namely 4-3-3 or 5-4-1, the number of defenders is overwhelming.

The difference between the characteristics of 2 datasets leads to the different distribution:

- The dataset 1 consists of historical values, thus the number of defenders are the largest, since many defenders can play at old age (30+), while midfielders are the ones doing most work on the field, thus it requires them a very high work rate and they cannot last long in big clubs.
- The dataset 2 only consists of players from 2021-2022 season, having the percentage midfielder the highest with approximately 38%. This is quite understandable because the teams in the top 5 European leagues have a lot of different styles of play depending on the game, including counter-attack, ball control,... Therefore, they need many midfielders.

Next , we see log value in each position in figure 6.

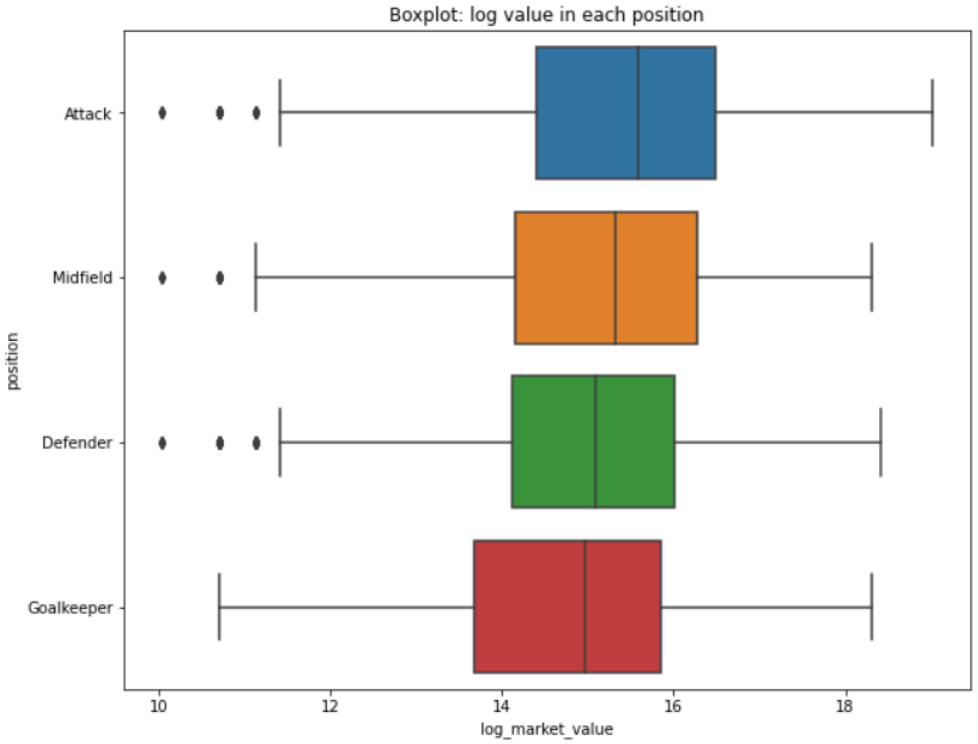



**Figure 6. Boxplot of log transformed value for each position**

We can see that: The price of strikers is higher compared to other positions and the price of goalkeepers is always the lowest. It means that the trend of football is to attack. Football teams and professionals focus on strikers. The proof is that most of the Ballon d'Or winners (best player of the year) play in the striker position.

### 2.3.2. Age

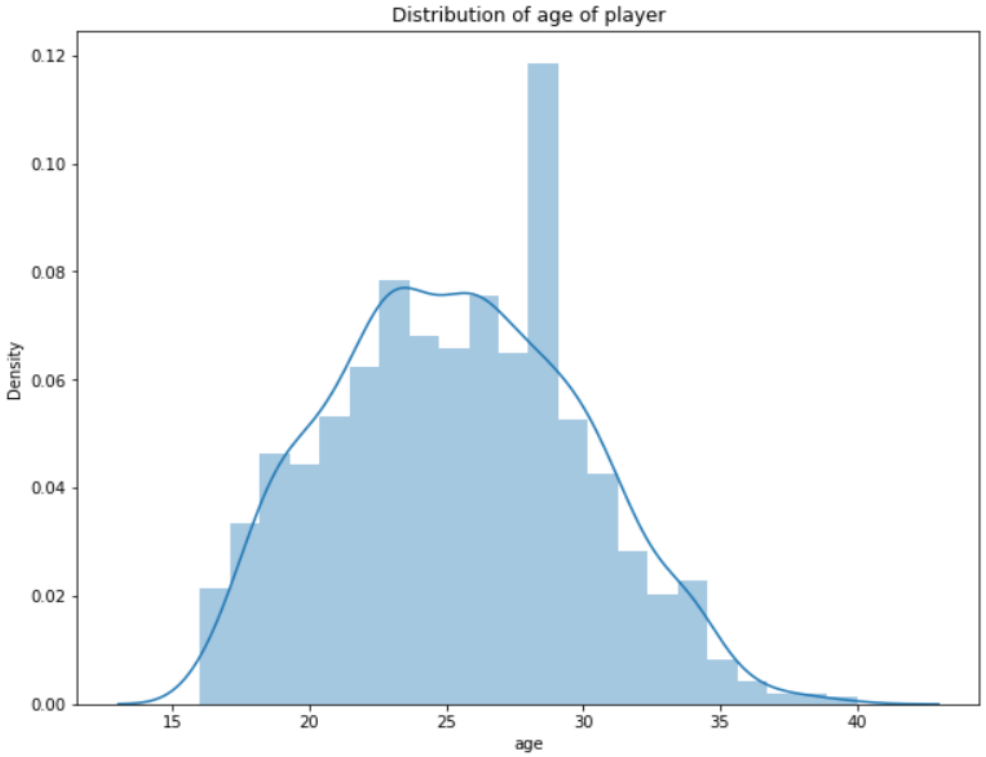



**Figure 7. Age distribution of players for Dataset 2**

The distribution of age is near-normal distribution.The number reduces drastically when age gets past 30 years old. Moreover, judging by figure 8, we can conclude that the age range from 27-29 is prime time for every player.

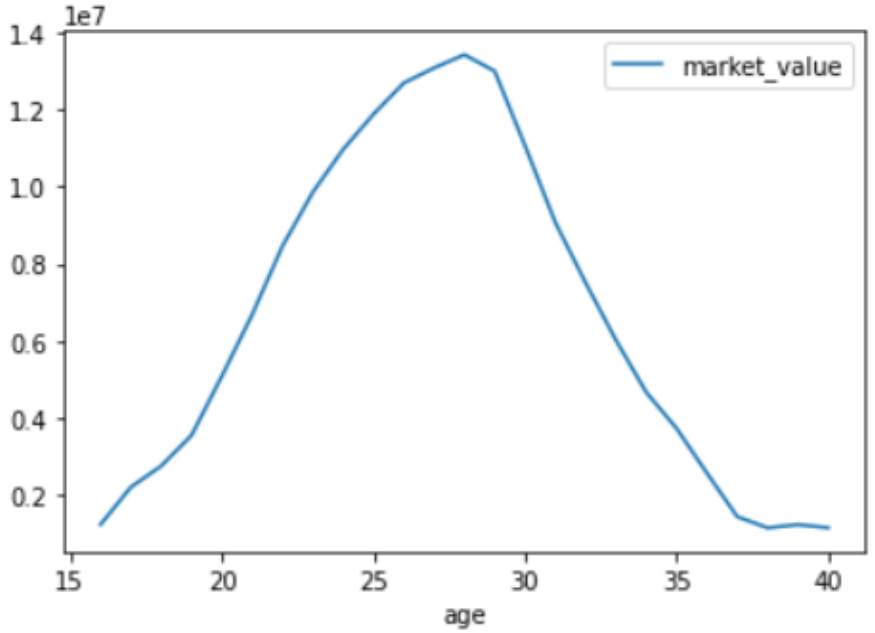



**Figure 8. Average market value for player at age (Dataset 1)**

Some may expect a phenomena when comparing figure 8 and 9

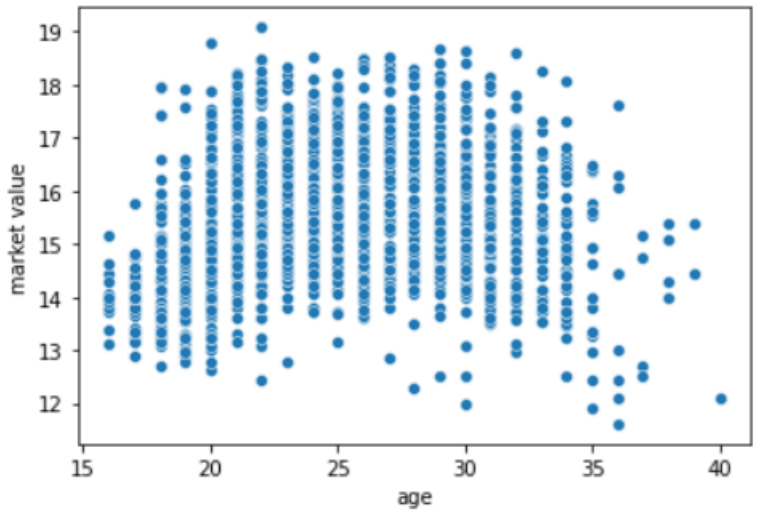



**Figure 9. Correlation between age and market value for Dataset 2**

Figure 9 says that age doesn't have much of an impact on the prices of players in the top 5 leagues. But does that disapprove our observation in figure 8? The answer is no, since the time frame of the two datasets are different. Instead, we can conclude that: it is hard to tell that a 28-year-old player can be worse or better than another 18-year-old player, however, he is most likely to be better than himself at 18.

## 2.3.4. Correlation between independent variables and target variables (Dataset 2)

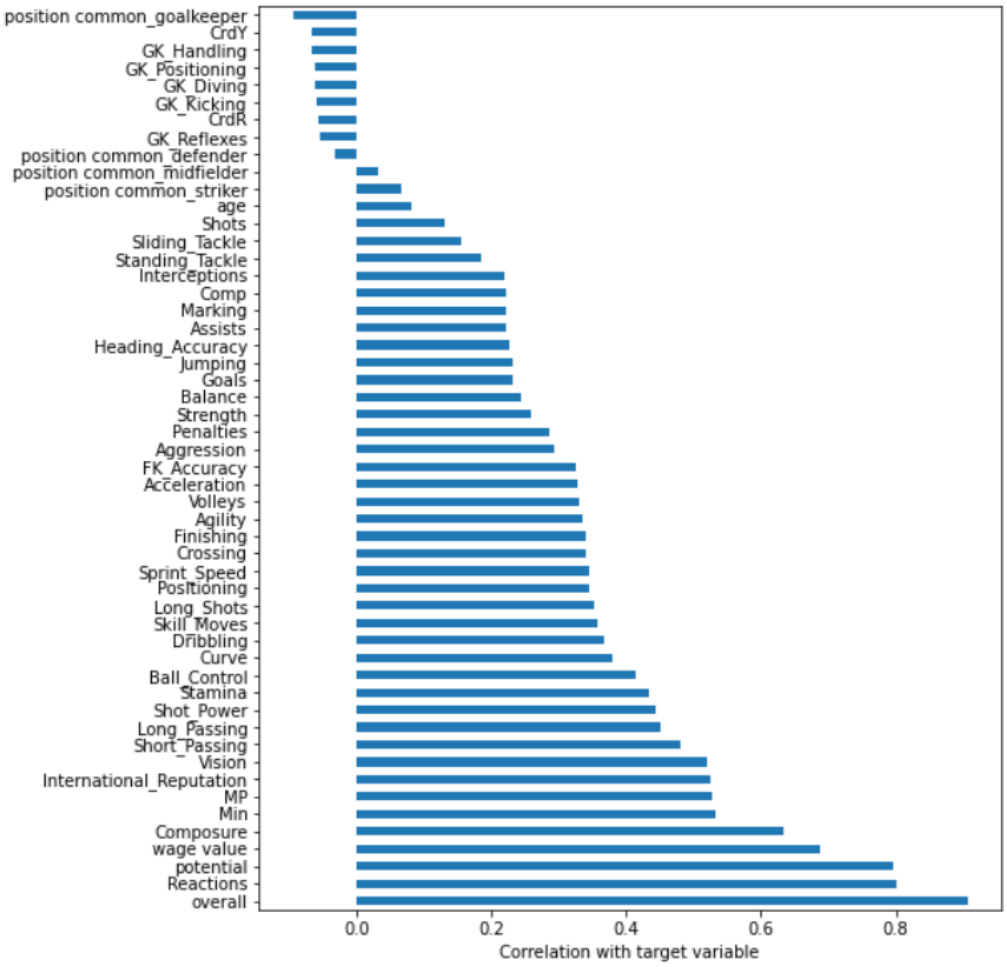



**Figure 10.**

We can see that: Some independent variables have high correlation with target variables, such as overall, reactions, potential, composure, min, MP. Which means, the more a player plays, and the higher his statistics in FIFA-22, the higher his value becomes. Besides, some independent variables have negative correlation with target variables, including position common ones, since they are strictly related to player position; CrdY, CrdR: since unfair plays affect ones' value negatively; GK_Handling, GK_Positioning, GK Diving, GK Kicking, GK_Reflexes are some values strictly for Goalkeepers and those value for other players often remains 0 or 1, therefore cannot correlate well with the data.

# 3. Metric for evaluation of models

In this project, we use two metrics for evaluation models: root mean squared error and R2 score

Root mean squared error: $RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$

R2 score: $R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$

where: + $y_i$ : actual value
+ $\hat{y}_i$ : predicted value
+ $\bar{y}$ : mean of actual values

For a regression model, the combination of these two metrics represent its accuracy very well:

- RMSE tells us how well a regression model can predict the value of the response variable in absolute terms
- R2 score tells us how well a regression model can predict the value of the response variable in percentage terms

# 4. Choosing data models & result

# 4.1. For dataset 1

### 4.4.1. Preprocessing

For this dataset, we first try to preprocess with the data by creating new features in the following pattern:

- From number of appearances and minutes played, get a feature of how much a player is fully played
- Number of goals/assists/MotM per match
- Percentile of each feature in the whole dataset
- For the hypothesis that a player's prime age is 28, a feature also measures how far is that player from his prime

We then split the data set into 4 positions, and use Spearman Correlation to select the best features for our model. In each set, data is split into training set (¾) and test set (¼).

### 4.1.2. Applied models

Linear Regression methods are used, based on the hypothesis that the better outcomes a player provides, the more expensive he would be. In particular, we used 3 different regularization methods: Lasso, Ridge and ElasticNet.
Moreover, K-nearest neighbor can be applied since players with the same performance should be evaluated the same.
Finally, Decision Trees and Random Forests are applicable because the architecture of trees can cluster data points, therefore the same logic from K-nearest neighbors can be applied.
The GridSearchCV module with RMSE is used for tuning hyperparameters suitable for our model.

### 4.1.3. Experimental results

Initially, we tested the Linear Regression methods on 4 separated positions. However, the model works better when all data are combined. The results are presented below

| Data | Best model | RMSE | R2 score |
|---|---|---|---|
| Attack/Striker | Ridge(alpha=10) | 1.26110 | 0.32127 |
| Midfielder | ElasticNet(alpha=0.002, l1_ratio=0.9) | 1.23395 | 0.31565 |
| Defender | Ridge(alpha=20) | 1.23229 | 0.29585 |
| Goalkeeper | Ridge(alpha=20) | 1.20931 | 0.35937 |
| All combined | Ridge(alpha=10) | 1.17645 | 0.40933 |

Since the combined data works the best, we would apply it to the remaining models tuned by GridSearchCV:

- KNeighborsRegressor(n_neighbors = 14, metric = 'euclidean', weights = 'distance')
  got R2 = 0.32826
- DecisionTreeRegressor(max_depth = 6,criterion = 'poisson', min_samples_leaf= 4, max_features= 1.0)
  got R2 = 0.34266
- RandomForestRegressor(n_estimators = 500, min_samples_leaf= 4, max_features= 0.5)
  got R2 = 0.34266

For the low value of R2-score, we would like to compare the result with the second dataset to choose which is better for training our model.

## 4.2. For dataset 2

### 4.2.1. Lasso - Ridge - KNN

#### 4.2.1.1. Preprocessing

First, we can convert category columns into numeric columns:
+ Comp column (name of league): we use ordinal encoder, specifically: Premier league into 1, La Liga into 2 , Bundesliga into 3, Serie A into 4 and Ligue 1 into 5.
+ Position (includes goalkeeper, defender, midfielder, striker): we use one-hot encoder

Then, drop columns that aren't related to building our model: name and preferred_foot columns. For feature scaling: we use standardization for independent variables (x). Finally, for train - test split, ⅔ of the dataset is used for training and ⅓ is used for testing.
The GridSearchCV module with RMSE is used for tuning hyperparameters suitable for our model.

#### 4.2.1.2. Experimental results

- **Lasso**: Best parameter: alpha = 0.005

```
Root mean squared 0.221936440341696
R2-scored  0.967581063962568
```

- **Ridge**: Best parameter: alpha = 0.4

```
Root mean squared 0.22603063142189242
R2-scored  0.9669988183986586
```

- **KNN**: Best parameter: n_neighbors = 7, weights = 'distance', metric = 'manhattan'

```
Root mean squared 0.38971366862658924
R2-scored  0.867663755288164
```

### 4.2.2. Decision Tree – Random Forest – AdaBoost

#### 4.2.2.1. Preprocessing

The same process of preprocessing is used for these methods, however we don't apply feature scaling.
The GridSearchCV module with RMSE is used for tuning hyperparameters suitable for our model.

#### 4.2.2.2. Experimental results

- **Decision tree**: Best parameter: criterion = 'friedman_mse', max-depth = 16, max_features = None, min_samples_leaf = 2

```
r2 score : 0.975740722816691
Root mean squared  : 0.19550146475369884
```

- **Random Forests:** Best parameter: n_estimators = 500, max_features = None, min_samples_leaf = 1

```
r2 score : 0.9920981681987293
Root mean squared error:  0.11025607993364234
```

- **AdaBoost**:Best parameter: base_estimator = DecisionTreeRegressor(max_depth=13,random_state=0), n_estimators = 94, learning_rate = 1.0, random_state = 3

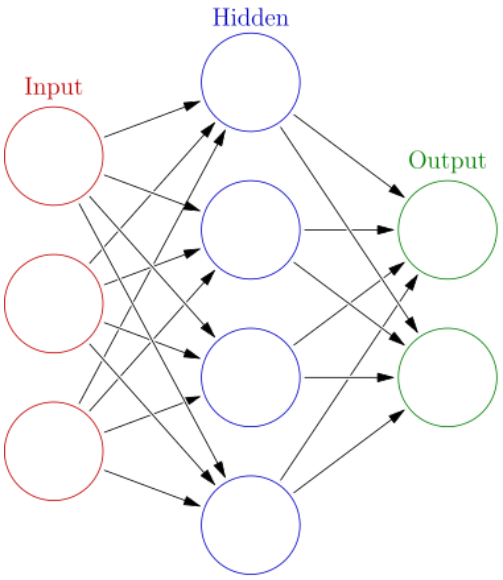
```
Root mean squared 0.12748218988490542
R2-scored  0.9894280792870761
```

### 4.2.3. Artificial Neural Network

#### 4.2.3.1. Layers of Artificial Neural-network

ANNs are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs and produces a single real-valued output.

Artificial Neural-network is typically organized in layers. Layers are being made up of many interconnected 'nodes' which contain an 'activation function'. A neural network may contain the following 3 layers.




#### 4.2.3.2. Algorithm for ANN

Each neuron is connected with all the neurons in the following layer. Each connection represents a weight that contributes to the fitting. With a proper activation function, a combination of optimized weights can generate the prediction of the dependent variable

$$NET = \sum_{i,j}^{n} w_{ij} x_i + b$$

where $w_{ij}$ represents the weight value of a connection, $x_i$ represents an inputted independent variable, and $b$ represents a bias. For the activation function $f(NET)$, the sigmoid function is one of the most popular forms that can introduce a smooth non-linear fitting to the training of an ANN. The training of an ANN is essentially the optimization of each weight contribution based on the data groups in the training set. The most commonly used weight optimization method is the back-propagation algorithm, which iteratively analyzes the errors and optimizes each weight value based on the errors generated by the next layer.

$$f(NET) = \frac{1}{1+e^{-NET}}$$

To validate the trained ANN, a testing process is necessary. The testing of a model should use the data groups that are not used for the training process. With the inputs of the testing set, the outputted data can be compared with the actual data in the testing set, with the root mean square error (RMSE)

#### 4.2.3.3. Experimental results

| Model | R2-score | Root-means_squared_error |
|---|---|---|
| ANN with batch size 64 and first hidden layer's nodes 1500 | 0.77 | 0.65 |
| ANN with batch size 64 and first hidden layer's nodes 1000 | 0.51 | 1.22 |

| | | |
|---|---|---|
| ANN with batch size 64 and first hidden layer's nodes 500 | 0.48 | 1.34 |

Best parameter: Activation_function = "relu",
loss_function = "mean_squared_logarithmic_error",
optimizer = "Adam",
number of hidden layers = 5,
first hidden layer = 1500,
last hidden layer = 10,
batch size = 64,
epochs = 10

```
r2 score : 0.7707037366414715
Root mean squared error:  0.650466074201643
```

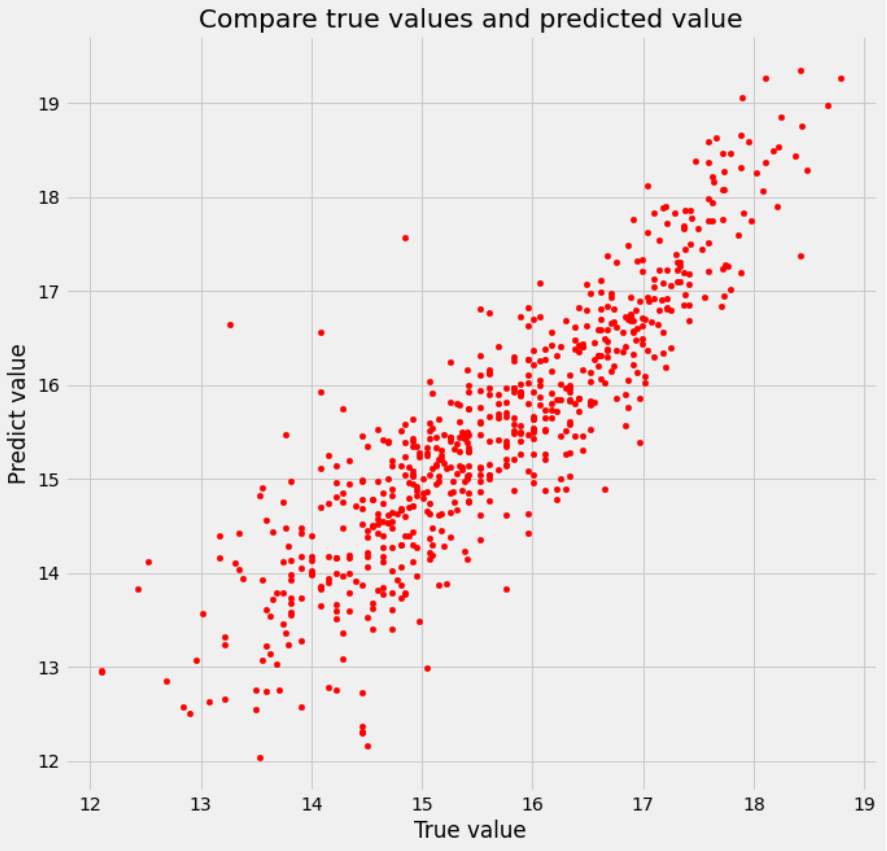



**Figure 11. Compare true values with predicted values.**

## 4.2.4. Gradient Boosting: XGBoost and LightGBM

### 4.2.4.1. Theory of gradient boosting and its variations

A gradient-boosted model is built in a **stage-wise** fashion as in other boosting methods. But it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.
Generic gradient boosting method:

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations $M$.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For $m$ = 1 to $M$:
   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

**Figure 12. Gradient boosting method**

Regularization: some important parameters

+ Shrinkage:

$F_m(x) = F_{m-1}(x) + v\gamma_m h_m(x),\ 0 < v \leq 1,$ where $v$ is called the "learning rate"

Empirically, it has been found that using small learning rate $v < 0.1$. But a lower learning rate requires more iterations, thus leading to increasing computation time.

+ number of trees: Increase M reduces the error on the training set but setting it too high may lead to overfitting. An optimal value M is often selected by a validation set.

+ Max depth: usually equal to 3, we need to choose from 3 to 9 for max depth.

+ Penalize complexity of tree: L2 penalty on the leaf values can also be added to avoid overfitting.

#### 4.2.4.1.1. XGBoost

XGBoost is an open-source software library which provides a regularizing gradient boosting.
Salient features of XGBoost which make it different from other gradient boosting algorithm include:

+ Clever penalization of trees
+ A proportional shrinking of leaf nodes
+ Newton Boosting
+ Extra randomization parameter
+ Implementation on single, distributed systems, and out-of-core computation
+ Automatic feature selection

#### 4.2.4.1.2. LightGBM

LightGBM has many of XGBoost's advantages, including sparse optimization, parallel training, multiple loss functions, regularization, bagging and early stopping.
LightGBM doesn't grow trees level-wise , instead it grows trees leaf-wise. It chooses a leaf that will yield the largest decrease in loss. Further explanations are provided in figure

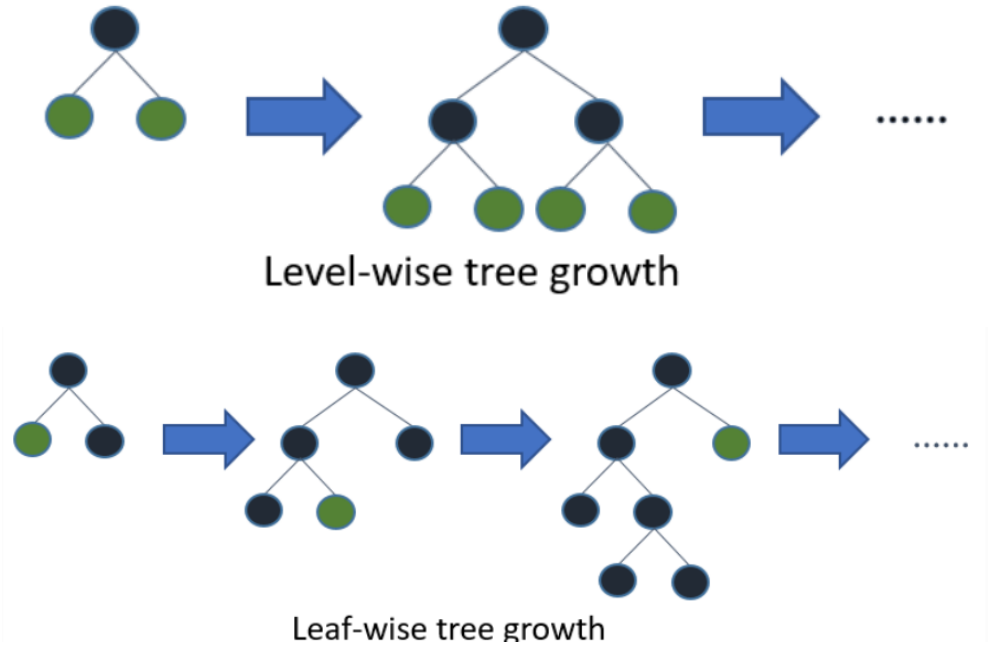



**Figure 13. Tree growth**

LightGBM implements a highly optimized histogram-based decision tree learning algorithm.
The LightGBM algorithm utilizes two novel techniques called Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) which allow the algorithm to run faster while maintaining a high level of accuracy

+ **GOSS**: that there is no native weight for data instance in GBDT, the instances with larger gradients will contribute more to the information gain.Thus, in order to retain the accuracy of the information, GOSS keeps the instances with large gradients and randomly drops the instances with small gradients.

+ **EFB**: is a near-lossless method to reduce the number of effective features. In a sparse feature space many features are nearly exclusive, implying they rarely take nonzero values simultaneously. EFB bundles these features, reducing dimensionality to improve efficiency while maintaining a high level of accuracy.

#### 4.2.4.2. Preprocessing

Preprocessing is conducted the same way with decision tree – random forest – AdaBoost.
The GridSearchCV module with RMSE is used for tuning hyperparameters suitable for our model.

#### 4.2.4.3.Experimental results

- **XGBoost**:

Best parameter: n_estimators = 2000, learning_rate = 0.03, gamma = 0.0, max_depth = 3.
( gamma : Gamma specifies the minimum loss reduction required to make a split. The values can vary depending on the loss function and should be tuned.)

```
Root mean squared 0.07013753747205126
R2-scored  0.9968735027541525
```

- **LightGBM**:

Best parameter: n_estimators = 1000, learning_rate = 0.05, reg_alpha = 0.1, reg_lambda = 1, max_depth = 3
(reg_alpha = L1 regularization, reg_lambda = L2 regularization)

```
Root mean squared 0.09735391781435403
R2-scored  0.9939875216024947
```

### 4.2.5. Stacking

#### 4.2.5.1. Preprocessing

Preprocessing is conducted the same way with decision tree – random forest – AdaBoost.
The GridSearchCV module with RMSE is used for tuning hyperparameters suitable for our model.

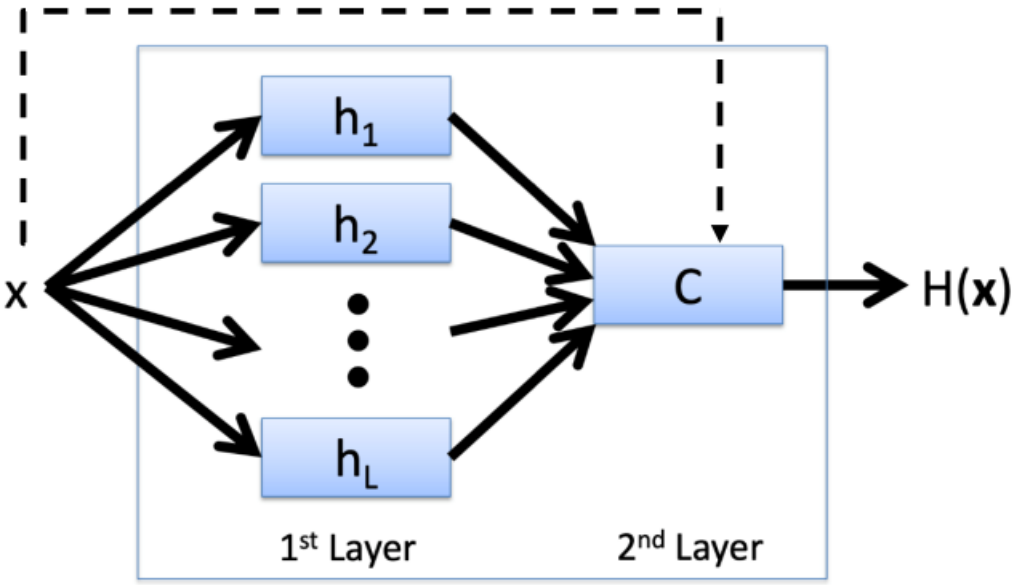



**Figure 14. Stacking model**

#### 4.2.5.2.Experimental results

Layer 1: we use 4 models, namely: Lasso, Ridge, Decision Tree and Random Forest. After we predict for 4 models, the result of 4 models will add into independent variables(x) as 4 new features.
Layer 2: we use XGBoost for hyperparameter tuning
Best parameter: n_estimators = 6000, learning_rate = 0.01, gamma = 0, max_depth = 3
Training models with hyperparameters, we have accuracy:

```
Root mean squared 0.07805544277635618
R2-scored  0.9962017279178491
```

# 5. Summary of results and analysis with best model

## 5.1. Summary of results

- From our observed results, we can conclude that the second dataset is more suitable for predicting players' value.
- Although the first dataset does not help us get the perfect prediction value, it can accurately list out most valuable players and world class ones.
- The accuracy of models from the second dataset are listed as follows

| | Name of models | RMSE | R2 score |
|---|---|---|---|
| 0 | Lasso | 0.221936 | 0.967581 |
| 1 | KNN | 0.389714 | 0.867664 |
| 2 | Ridge | 0.226031 | 0.966999 |
| 3 | Decision Tree | 0.195501 | 0.975741 |
| 4 | Random Forest | 0.110256 | 0.992098 |
| 5 | Adaboost | 0.127482 | 0.989428 |
| 6 | ANN | 0.650466 | 0.773704 |
| 7 | XGBoost | 0.070138 | 0.996874 |
| 8 | LightGBM | 0.097354 | 0.993988 |
| 9 | Stacking | 0.078055 | 0.996202 |

We can see that: XGBoost has the smallest RMSE and the largest R2_score. The barplot for RMSE portrays the best possible model.

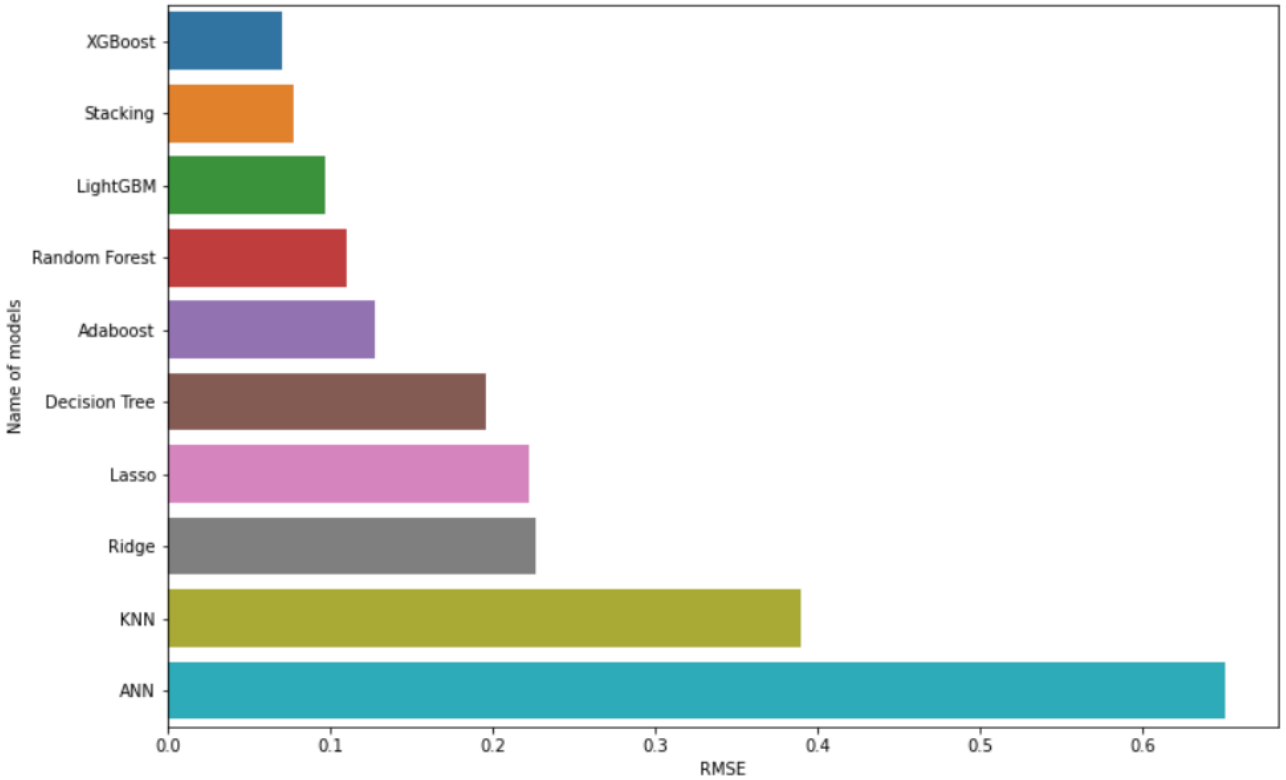



**Figure 15. Barplot of RMSE for models for Dataset 2**

## 5.2. Analysis with best model

XGBoost is the best model, so we decided to apply this model for our problem, we will use it for analysis results.

Since we used log transformation for market value variables, we can evaluate actual values using natural exponential function. The final RMSE and R2 score for actual market values and predicted market values are as below:

```
Root mean squared error :  1373136.7151043897
R2-score :   0.9940817312005764
```

Figure 14 plots the error between actual values and predicted values.

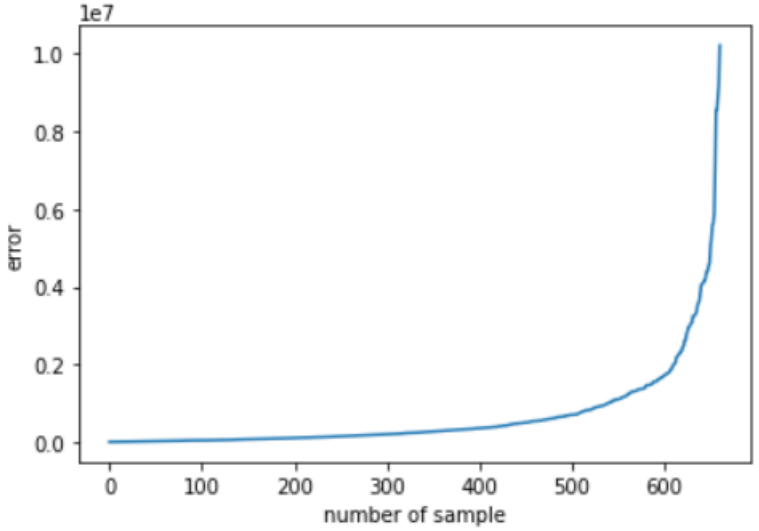



**Figure 16. n-closests approximation of player value**

Based on figure 14, we have approximately 610 players that the error of real value and prediction is smaller than 1 million euros.

Finally, we use a new metric to calculate the percentage of error of player prices:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i}$$

Applying this new metric for our model, we have :

```
percentage of error of player prices :  5.424107721238021 %
```

It means that: If a player is really worth 100 million euros, our best model would predict that player's value between 100 - 5.4 million euros to 100 + 5.4 million euros.Hence, it is a good prediction since in real life, clubs' deals may differ from real values tens of million euros.

# 6. Limitations and Future extensions

- We have only covered a part of a very huge player database around the world. Hopefully, in the future, we can apply our models to data for all players in the world, not just the top 5 European leagues.
- Due to time constraints, many models have not been perfect.
- More features should be considered in the model and data, such as contract deadline, contract fee, advertisement of players, players' fame,... which are also very important in deciding or predicting his value
- The model should be extended to other sports, such as basketball where the transfer market is also very active.

# 7. References


[1] https://docs.scrapy.org/en/latest/intro/tutorial.html
[2] Web Scraping in Python using Scrapy, retrieved at https://www.codementor.io/@mgalarny/using-scrapy-to-build-your-own-dataset-cz24hsbp5
[3] https://en.wikipedia.org/wiki/Gradient_boosting
[4] https://en.wikipedia.org/wiki/XGBoost
[5] https://en.wikipedia.org/wiki/LightGBM
[6] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
[8] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[10] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html
[11] https://xgboost.readthedocs.io/en/stable/
[12] https://lightgbm.readthedocs.io/en/v3.3.2/
[13] https://docs.scrapy.org/en/latest/intro/tutorial.html
[14] A Gentle Introduction to XGBoost for Applied Machine Learning, retrieved at https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
[15] Premier League announce three-year renewal of TV rights deal, retrieved at https://www.irishtimes.com/sport/soccer/english-soccer/premier-league-announce-three-year-renewal-of-tv-rights-deal-1.4563972
[16] Leicester City and the greatest underdog story ever told: a primer for Americans, retrieved at https://www.theguardian.com/football/2016/apr/30/leicester-city-premier-league-champions-underdog-story
[17] Measures of Model Fit for Linear Regression Models, retrieved at https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/
[18] How does Artificial Neural Network (ANN) algorithm work? Simplified! retrieved at https://www.analyticsvidhya.com/blog/2014/10/ann-work-simplified/.
[19] Activation functions in Neural Networks retrieved at https://www.geeksforgeeks.org/activation-functions-neural-networks/
[20] Algoritmo ANN | How the artificial neural network works | Datapeaker. retrieved at https://datapeaker.com/en/big--data/ann-algorithm-how-the-artificial-neural-network-works/