Predicting Footballer Market Values using Machine Learning

FYS-STK3155: Project 3

Bror Johannes Tidemand Ruud (640394)
Egil Furnes (693784)
Ådne Rolstad (693783)

University of Oslo
Date: 18.12.2025
GitHub: github.com/egil10/FYSSTK3155

Abstract

The professional football transfer market has grown into a global industry where player valuations play a central role in how clubs plan, compete, and remain financially sustainable. Accurately estimating a player's market value is challenging: while the overall market has increased rapidly over time, individual valuations fluctuate due to factors such as age, short-lived performance peaks, reputation, and changing tactical or strategic priorities at clubs. As transfer fees continue to reach new highs, decisions that were once driven largely by subjective scouting judgments are increasingly supported by quantitative analysis. In this setting, data-driven modeling of player market value has become an important tool for understanding and navigating the modern football economy.

In this project we seek to predict footballers market value applying machine learning (ML) methods on data from the *Football Data from Transfermarkt* dataset [1]. Evaluating the ML performance, we use R^2 and root mean squared error $RMSE$. Regarding ML models, we use Ridge regression and feed forward neural networks (FFNN) on our static dataset, before applying recurrent neural networks (RNNs) on a temporal version of the data.

First, we generate a *core dataset* consisting of player attributes (height, age, position, and preferred foot), and game events including both cumulative statistics up to this point and performance during the last 10 games (goals, assists, substitutions, and yellow/red cards). As from previous projects, we find that a simple FFNN outperform Ridge regression, yielding a R^2 of 0.6024 versus 0.5059 and RMSE of 0.9537 versus 1.0631 respectively. Using an *extended dataset* to include nationality, we find that the predictive value of our ML models increase, now with R^2 of 0.6335 versus 0.5484 and RMSE of 0.9157 versus 1.0165 for FFNN and Ridge respectively.

Based on domain knowledge and exploratory data analysis, we find that the general football market and transfer fees have increased massively over time. Therefore, to capture temporal changes, we introduce a RNN, and set up our dataset as being sequential and time-dependent. This massively improves our ML predictive powers, now yielding R^2 of 0.9760 and 0.9759 and RMSE of 0.2452 and 0.2548 for GRU and LSTM, respectively.

I. INTRODUCTION

Accurately valuating professional football players is a central component of the modern transfer market, where accurate assessments of player worth influence club strategy, financial planning, and long-term squad development. As transfer fees and overall market values continue to rise, see Figure 1, estimating a fair and consistent market value remains a challenging task, shaped by both on-field performance and broader market dynamics. This project investigates to what extent a player's market value can be predicted using historical in-game events (goals, assists, cards, substitutions), and if these predictions improve in a sequential context.

Our analysis is based on the *Football Data from Transfermarkt.com* dataset [1], from which we construct a tabular dataset containing player attributes, performance statistics, and contextual indicators. The feature set includes basic player characteristics such as age and height, categorical indicators for league and playing position, and binary variables describing attributes such as dominant foot and nationality. Player performance is summarized using both cumulative career statistics (e.g. total goals, assists, and disciplinary records) and short-term form metrics computed over a rolling window of the ten most recent matches. The final dataset consists of 278 558 player–valuation instances described by 212 predictor variables.

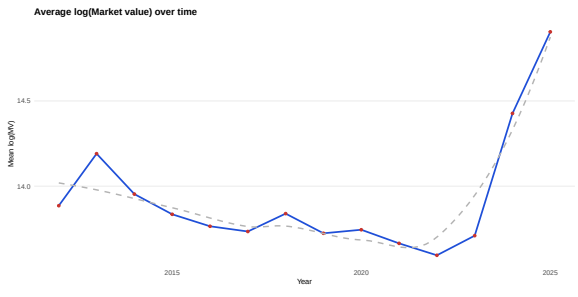

Figure 1: Time evolution of average log-transformed player market value across seasons, illustrating overall market growth and cyclical dynamics.

The target variable in this analysis is the estimated market value of each player, analyzed both in its raw form `y_raw` and after a logarithmic transformation, `y_log`, to account for the heavy right-tailed distribution of transfer values. We employ Ridge regression as a linear baseline model and compare its performance to that of feed-forward neural networks (FFNNs) capable of capturing non-linear relationships in the data. To assess whether model capacity limits predictive performance, we perform a systematic architecture study of the neural networks (NNs). Based on these findings, we further investigate the impact of feature engineering by extending the dataset with nationality information and re-evaluating both linear and non-linear models.

Going further than using the static dataset, we expand to a recurrent neural network (RNN), using a sequential time-dependent version of the dataset. With this improvement we seek to capture any temporal relationships, and use a players historical valuations to predict their future market value. As a purely anecdotal showcasing of our ML performance, we include a graphical representation of our predicted market values for the starting lineups of the Champions League Final of 2024, including 11 players from Borussia Dortmund and 11 from Real Madrid, as seen in Figure 14.

Beyond its predictive objective, this project serves as a practical exploration of applied machine learning (ML) techniques, emphasizing model selection, regularization, and reproducibility. The results provide empirical insight into the relative importance of model architecture and feature design in the context of football player valuation. The report is organized as follows. Section II presents the ML methods and evaluation framework used in the analysis, Section III presents and discusses the results, and Section IV summarizes the main findings and outlines directions for future work. References are shown in Section V, and additional figures and supplementary material are provided in Section VI.

II. METHODS

This section describes the methods employed in the project. We first outline the models and evaluation metrics used, followed by a description of data splitting and feature scaling procedures. We then summarize key implementation details related to data processing and model training, and conclude with a brief account of the use of AI-based tools during the project.

A. Method

1. Evaluation Metrics

To assess and compare model performance, we employ the mean squared error (MSE) and the coefficient of determination (R^2) throughout this report. The MSE measures the average squared deviation between the predicted values $\tilde{\mathbf{y}}$ and the true targets $\mathbf{y}$, and is defined in eq. (1). For interpretability, we also report the root mean squared error (RMSE), defined as $\text{RMSE} = \sqrt{\text{MSE}}$, which has the same units as the target variable.

The R^2 score quantifies the proportion of variance in the target variable explained by the model, relative to the empirical mean $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, as shown in eq. (2). When the target variable is log-transformed, all evaluation metrics, including MSE, RMSE, and R^2, are computed on the logarithmic scale. In this case, the R^2 score measures the fraction of variance explained in the log-transformed market value rather than in the raw monetary scale. Lower values of MSE (and RMSE) and higher values of R^2 therefore indicate improved predictive performance.

$$\text{MSE}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2, \tag{1}$$

$$R^2(\mathbf{y}, \tilde{\mathbf{y}}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{2}$$

2. Splitting and Scaling the Data

Prior to model training, the dataset is partitioned into disjoint subsets to enable an unbiased evaluation of predictive performance.

For the Ridge regression and FFNN models, we apply an 80/20 train–test split at the player level using grouped sampling based on player identifiers. This ensures that multiple valuation records for the same player do not appear in both sets, thereby preventing information leakage and overly optimistic performance estimates. A fixed random seed (`6114`) is used throughout to ensure reproducibility.

All input features are standardized using the `StandardScaler` from `scikit-learn` [2], which rescales each feature to zero mean and unit variance:

$$z = \frac{x - \mu}{\sigma}, \tag{3}$$

where x denotes the original feature value, and μ and σ are the mean and standard deviation estimated exclusively from the training data. The fitted scaler is then applied to the test data to avoid information leakage.

Feature scaling is essential for both model classes considered. For Ridge regression, standardization ensures that all coefficients are penalized uniformly under L_2 regularization. For FFNNs, scaling improves numerical stability during optimization by keeping gradient magnitudes within a reasonable range, facilitating faster and more reliable convergence.

For the RNN, a temporally ordered split is used. Data up to 2021 is used for training, data up to 2023 for validation, and 2024 data for testing. This results in 251694, 67018, and 18618 observations for the training, validation, and test sets, respectively.

Both static and sequential features are standardized using parameters derived solely from the training set. For

sequential inputs, features are flattened across time steps when computing global means and standard deviations, ensuring numerical stability while preventing information leakage.

3. Ridge Regression

We consider a linear regression model on matrix form,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{4}$$

where $\mathbf{y} \in \mathbb{R}^n$ denotes the response vector, $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the design matrix, $\boldsymbol{\beta} \in \mathbb{R}^p$ contains the regression coefficients, and $\boldsymbol{\varepsilon}$ is a noise vector satisfying $\mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{0}$ and $\mathrm{Var}[\boldsymbol{\varepsilon}] = \sigma^2\mathbf{I}$.

Ridge regression extends ordinary least-squares regression by introducing an L_2 regularization term that penalizes large coefficients. This is particularly well suited to the present setting, where the feature space is high-dimensional and exhibits substantial multicollinearity due to correlated performance statistics and categorical encodings. The regularization stabilizes the inversion of the normal equations and reduces estimator variance at the cost of introducing a small bias.

The Ridge estimator is obtained by minimizing the penalized objective function

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \arg\min_{\boldsymbol{\beta}} \left(\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right), \tag{5}$$

where $\lambda \geq 0$ is the regularization parameter. This optimization problem admits the closed-form solution

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{6}$$

The regularization parameter λ controls the strength of the coefficient shrinkage: for $\lambda = 0$, the estimator reduces to the unregularized least-squares solution, while increasing values of λ progressively shrink the coefficients toward zero. In this project, λ is selected using group K-fold cross-validation on the training data, with grouping performed at the player level to ensure that all observations associated with a given player are assigned to the same fold. For each candidate value of λ, the mean validation error across folds is computed, and the value minimizing this error is selected. The final Ridge model is subsequently trained on the full training set using the selected λ and evaluated once on the held-out test set.

4. Neural Networks

FFNNs are flexible function approximators capable of learning non-linear relationships between inputs and outputs by iteratively updating their weights during training [3]. This flexibility makes FFNNs well suited for modeling complex interactions between player attributes, performance statistics, and contextual variables in the present regression task.

Throughout this report, network architectures are denoted by $[h_1 - h_2 - \cdots - h_L - 1]$, where h_ℓ represents the number of neurons in hidden layer ℓ for $\ell = 1, \ldots, L$, and the final 1 corresponds to a single output neuron producing a scalar prediction. The input layer is omitted, as its dimensionality is determined by the feature set. An example architecture is shown in Figure 3.

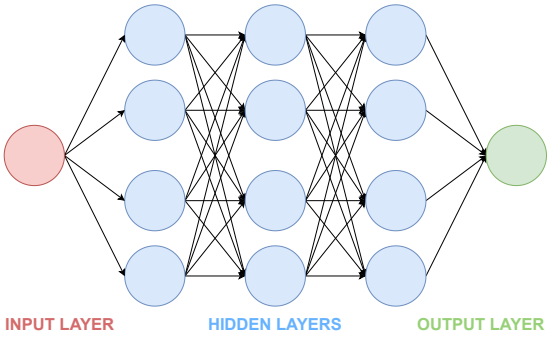

Figure 2: Schematic illustration of a FFNN with input, hidden, and output layers. Circles represent neurons, while lines indicate weighted connections. The example shown corresponds to a [4–4–4–1] architecture.

Each neuron computes an affine transformation of its inputs followed by a non-linear activation function. Model parameters are learned by minimizing the MSE loss,

$$\mathcal{C}_{\text{MSE}} = \frac{1}{2n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2, \tag{7}$$

using gradient-based optimization.

For the hidden layers, we consider the rectified linear unit (ReLU) and hyperbolic tangent (tanh) activation functions. ReLU is computationally efficient and promotes stable optimization, while tanh provides smooth, bounded activations. The output layer employs a linear activation function to allow unbounded continuous predictions.

All networks are trained using mini-batch gradient descent with the Adam optimizer implemented in PyTorch [4]. Model selection and generalization control are performed using validation-based early stopping, while the test set is reserved exclusively for final performance evaluation.

5. Recurrent Neural Networks

While the FFNNs considered in the previous section operate on fixed-size feature vectors, they do not explicitly model the temporal structure inherent in a football player's career. Player valuation is intrinsically dynamic, evolving as a function of accumulated performance, recent form, and age-related development. To explicitly account for such sequential dependencies, we also consider RNNs, which are designed to process ordered sequences of observations.

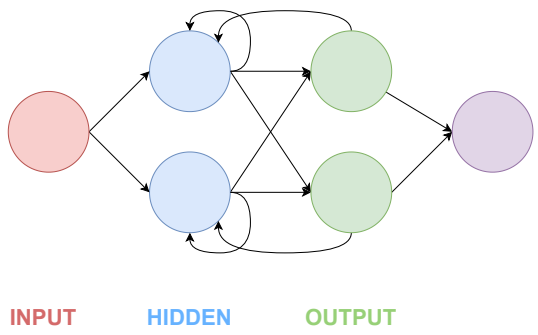

Figure 3: Schematic illustration of a RNN. Unlike standard feed-forward architectures, RNNs feature recurrent connections (indicated by loops) that allow information to persist across time steps. This enabling the model to capture temporal dependencies in sequential player performance data.

a. ***Sequential formulation:*** For each player, we construct a time-ordered sequence of feature vectors

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T,$$

where each $\mathbf{x}_t \in \mathbb{R}^p$ summarizes the player's attributes and performance statistics at valuation time t. These vectors may include cumulative career statistics, rolling-window form metrics, and static attributes such as height and position. The corresponding target is the player's market value y_t (or its logarithm) at the final time step.

Unlike FFNNs, which assume independence between samples, RNNs maintain a hidden state that evolves over time and serves as a memory of past observations. This allows the model to capture temporal patterns such as performance trends, sudden improvements, or gradual decline.

b. ***RNN architecture and forward dynamics:*** In a standard (vanilla) RNN, the hidden state $\mathbf{h}_t \in \mathbb{R}^m$ at time step t is updated according to

$$\mathbf{h}_t = \phi\left(W_x \mathbf{x}_t + W_h \mathbf{h}_{t-1} + \mathbf{b}_h\right), \tag{8}$$

$$\tilde{y}_t = W_y \mathbf{h}_t + b_y, \tag{9}$$

where W_x, W_h, and W_y are learnable weight matrices, $\mathbf{b}_h$ and b_y are bias terms, and $\phi(\cdot)$ is a non-linear activation function, typically tanh or ReLU. The initial hidden state $\mathbf{h}_0$ is either initialized to zero or treated as a learnable parameter.

The recurrence in Eq. (8) enables information from earlier time steps to influence later predictions, allowing the network to model long-range temporal dependencies that are inaccessible to static regression models.

c. ***Loss function and training:*** For regression, the RNN is trained by minimizing the MSE between the predicted and true target values. When predicting the market value at the final time step T, the loss function takes the form

$$C_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^{n} \left(\tilde{y}_{i,T} - y_{i,T}\right)^2,$$

where n denotes the number of player sequences in the training set.

Model parameters are optimized using gradient-based methods, with gradients computed through backpropagation through time (BPTT). In BPTT, the computational graph is unrolled across time steps, and gradients are propagated backward through the sequence. To ensure numerical stability, training is performed using the Adam optimizer together with gradient clipping.

d. ***Long Short-Term Memory (LSTM):*** To mitigate the vanishing gradient problem, the LSTM introduces a cell state $\mathbf{c}_t$ regulated by three gates: forget ($\mathbf{f}_t$), input ($\mathbf{i}_t$), and output ($\mathbf{o}_t$). These gates use sigmoid activations to control the persistence of information, allowing the network to maintain long-term dependencies across extended sequences by selectively updating or discarding historical data.

e. ***Gated Recurrent Unit (GRU):*** The GRU is a streamlined variant that merges the cell and hidden states into a single vector $\mathbf{h}_t$. It utilizes an update gate $\mathbf{z}_t$ and a reset gate $\mathbf{r}_t$ to govern information flow. By reducing the number of parameters compared to LSTM, the GRU offers improved computational efficiency and faster convergence while remaining highly effective at capturing temporal correlations.

B. Implementation

The project is implemented primarily in `Python` [5], with limited use of `R` [6] for descriptive statistics and exploratory analysis, using packages `tidyverse` [7] and `readr` [8]. The report is written in LaTeX using `Overleaf`, and the complete codebase is version-controlled and publicly available on `GitHub` [9] to ensure transparency and reproducibility.

Data processing, feature engineering, and model implementation rely on established scientific computing libraries, including `NumPy` [10], `Pandas` [11], `Matplotlib` [12], `Scikit-Learn` [2], `Jupyter` [13], and `PyTorch` [4]. To ensure reproducibility across all experiments, a fixed global random seed (`6114`) is used for data splitting, model initialization, and training.

1. Ridge Regression and Neural Networks

Raw player data from the *Football Data from Transfermarkt* dataset [1] are processed into a structured tabular format suitable for supervised learning. Feature engineering and aggregation are performed using custom Python scripts, producing an intermediate dataset stored in `.csv` format. The processed data are subsequently converted to the more compact `.parquet` format to reduce file size and facilitate version control, and for RNN we use a `.npz` datastructure.

When preparing the data used for Ridge and FFNNs, identifier and temporal variables such as `player_id` and valuation dates are explicitly removed in a dedicated

`prepare_data` function. In addition, the target variables corresponding to the raw market value and its logarithmic transformation are excluded from the feature matrix prior to model training. This design choice prevents information leakage and ensures that the models learn predictive relationships based solely on observable player attributes, performance statistics, and contextual variables.

Two distinct feature representations are used in the Ridge and FFNN analysis. A *core feature set* is constructed to capture fundamental player attributes, positional indicators, league context, and aggregated performance statistics, resulting in a total of 24 input features. In addition, an *extended feature set* is created by augmenting the core features with one-hot encoded nationality indicators, yielding a higher-dimensional representation with 208 input features. Both feature sets are generated within the same preprocessing pipeline and passed through identical training, validation, and evaluation procedures to ensure a fair and controlled comparison between models.

All preprocessing steps, including feature construction, encoding of categorical variables, and standardization, are applied consistently across models. The resulting feature matrices serve as input to both the Ridge regression and NN models described in Section II A. A complete overview of the included variables is provided in Section VI A.

2. Recurrent Neural Networks

In contrast to the Ridge regression and FFNN, which operate on static tabular representations, the RNN models are designed to explicitly capture temporal dynamics in player market valuations. To this end, a separate data preparation pipeline is implemented, producing a structured sequence-based dataset stored in `.npz` format.

For the RNN, each training sample corresponds to a single player valuation event and is constructed using information from previous valuation intervals of the same player. Rather than treating individual matches as timesteps, the sequence dimension is defined over *valuation dates*. Each timestep summarizes the player's performance and contextual information in the time interval between two consecutive market valuations. This design aligns the temporal resolution of the input data with the prediction target and ensures a strictly causal setup.

Let t_i denote the date of the i-th valuation for a given player. For each valuation t_i, the RNN input consists of a fixed-length sequence of the previous $T = 5$ valuation intervals, i.e. $(t_{i-5}, t_{i-4}], \ldots, (t_{i-1}, t_i]$, while the target is the logarithm of the market value at time t_i. Players with fewer than $T + 1$ valuation dates are excluded, as no valid sequence can be formed.

Each timestep in the sequence is represented by an 11-dimensional feature vector summarizing the interval between two valuation dates. These features include the logarithm of the previous market value, the number of days since the last valuation, the number of matches played during the interval, aggregated in-game performance statistics (goals, assists, yellow cards, red cards, substitutions in, substitutions out), the player's age at the previous valuation, and an indicator for whether the player was active in a top-five European league at that time. Match-level event data are aggregated using cumulative sums to efficiently compute interval statistics.

In addition to the sequential input, a set of static player features is provided to the model. These features remain constant across all timesteps and include player height as a continuous variable, as well as one-hot encoded representations of dominant foot and positional group. Static features are concatenated with the final hidden state of the recurrent encoder during model training.

The resulting dataset consists of four aligned components: a three-dimensional sequence tensor $\mathbf{X}_{\text{seq}} \in \mathbb{R}^{N \times T \times F}$, a static feature matrix $\mathbf{X}_{\text{static}} \in \mathbb{R}^{N \times F_s}$, a target vector $\mathbf{y}$ containing log-transformed market values, and metadata arrays containing player identifiers and target valuation dates. All samples are filtered to remove non-finite values before being saved to disk.

By constructing the data in this manner, the RNN models are able to learn temporal patterns in market valuation dynamics while avoiding information leakage, as each prediction depends only on information available prior to the target valuation date.

C. Use of AI tools

In the implementation of this report we have used several AI-tools, including traditional large language models (LLMs) on chat interface, but also coding agents. For LLMs we use ChatGPT from OpenAI [14], Grok from xAI [15], and the newly launched Gemini from [16]. For example, when making descriptive statistics in `R`, ChatGPT was very useful in helping with plotting [17], and in finding potential data issues and `NA`-values [18]. It also helped us aggregating and running data for RNN [19] [20].

Moreso, we included some chat exports from using Cursor [21] in the GitHub repository under `Use_of_AI_Tools`, one for aggregating data on nationality and one for completing the Champions League 2023 analysis. Finally, Gemini has been a useful companion in writing the OverLeaf report [22]. Regarding coding agents, we use Cursor from Anysphere [23] and the newly published Antigravity from Google [24], which are both seemingly forks on VSCode [25] with an added agentic chatbot interface.

III. RESULTS AND DISCUSSION

This section presents the empirical results of the models considered in this study. We begin with a concise summary of predictive performance across model classes, followed by a more detailed analysis of individual models, feature representations, and evaluation metrics.

A. Summary of Model Performance

To provide a consolidated comparison of model performance, we summarize the final predictive accuracy of all models in Table I. All reported metrics are computed on the held-out test set and include the coefficient of determination (R^2) and the RMSE. For NNs, hyperparameter selection and architecture tuning are performed exclusively using validation data, while the test set is reserved for a single, final evaluation of each selected model.

Model	Features	Arch./Reg.	R^2	$RMSE$
Ridge	Core	$\lambda = 1.78 \times 10^2$	0.5059	1.0631
NN	Core	ReLU [64-64-1]	0.6024	0.9537
Ridge	Ext.	$\lambda = 1.21 \times 10^3$	0.5484	1.0165
NN	Ext.	ReLU [64-64-1]	0.6335	0.9157
RNN GRU	Seq.	ReLU [64-64-1]	0.9760	0.2452
RNN LSTM	Seq.	ReLU [64-64-1]	0.9759	0.2548

Table I: Test set performance for Ridge regression, FFNNs and RNNs using the core and extended feature sets, and the sequential data set. For the FFNNs, the reported architecture corresponds to the best-performing configuration selected on the validation set.

B. Exploratory Data Analysis and Feature Motivation

We begin by examining the structure of the processed dataset and selected empirical patterns relevant to player market valuation. The dataset comprises 278 558 player–valuation instances and 212 predictor variables, including player attributes, performance statistics, and contextual indicators. Several core player attributes, such as age and height, exhibit approximately unimodal and symmetric distributions, as illustrated in Figure 4.

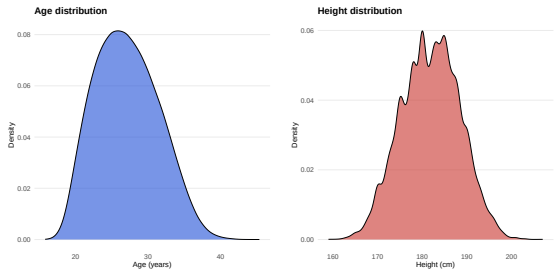

Figure 4: Empirical distributions of player age and height in the sample. Age is measured in years at observation time, while height is reported in centimeters.

A notable pattern emerges when comparing market valuations across competitive contexts. Players competing in the so-called Big-5 European leagues (England, Spain, Italy, Germany, and France) display systematically higher market values than players in other leagues. This effect is evident in Figure 5, where the distribution of logarithmic market value differs substantially between Big-5 and non–Big-5 leagues. This suggests that league-level factors influence valuation beyond individual performance metrics.

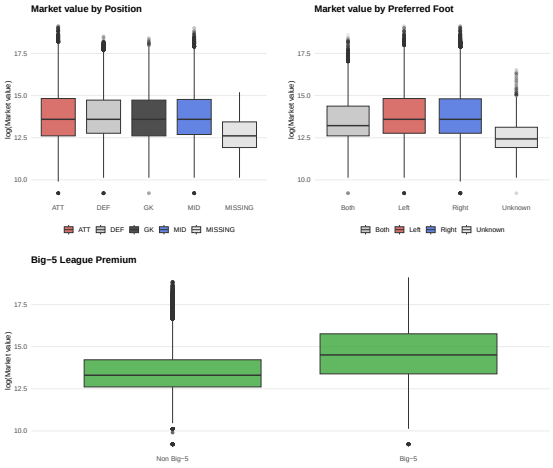

Figure 5: Distribution of player market values by categorical characteristics: playing position, preferred foot, and participation in a Big-5 European league. Values are shown on a logarithmic scale.

Further differences are observed when stratifying players by nationality. Figure 6, Figure 20, and Figure 21 show that both the level and age-profile of market values vary across countries. In particular, some nationalities exhibit higher median market values and earlier valuation peaks, indicating that nationality may act as a proxy for market reputation, scouting networks, or player development systems.

These observations indicate that player market value is shaped not only by measurable on-field performance, but also by broader contextual and market-related factors. Motivated by these findings, we extend the baseline feature set to include indicators for league status and player nationality, and investigate whether incorporating such information improves the predictive performance of both

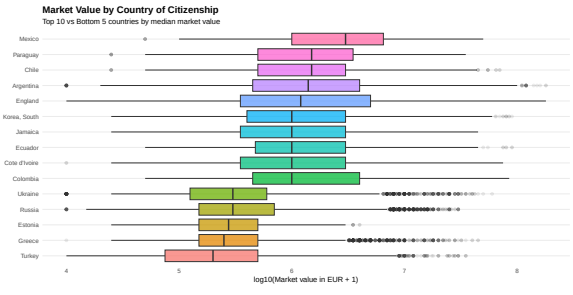

Figure 6: Cross-country comparison of player market values using boxplots. Each box summarizes the distribution of market values for players associated with a given country.

linear and non-linear models.

C. Ridge Regression: Core Feature Results

Figure 7 shows the training and cross-validation RMSE as approximate functions of the regularization parameter λ. Performance remains stable over a wide range of λ, with degradation only under strong regularization, indicating limited sensitivity to the bias–variance trade-off for this feature representation.

On the test set, the final Ridge model achieves an R^2 of approximately 0.51 and an RMSE of about 1.06 in logarithmic market value, implying that roughly half of the variance in player valuations is explained by the linear model.

Diagnostic plots in Figure 8 reveal systematic structure in the residuals. While predicted values track the overall trend well, residual variance increases for highly valued players, indicating heteroskedasticity and deviations from linearity. This suggests that important non-linear effects are not fully captured by the linear model, motivating the use of more flexible approaches in subsequent analyses.

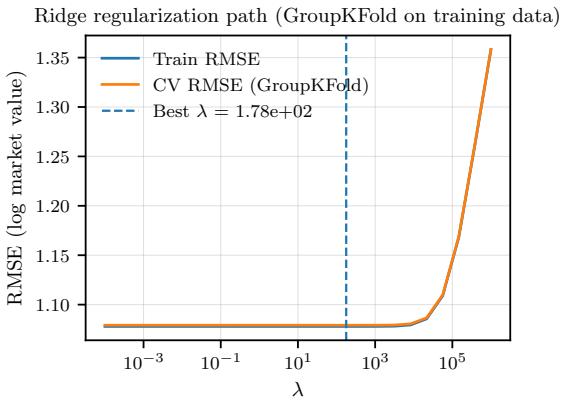

Figure 7: Training and validation RMSE as a function of the regularization parameter λ for Ridge regression on the core feature set. The dashed line indicates the selected value of λ.

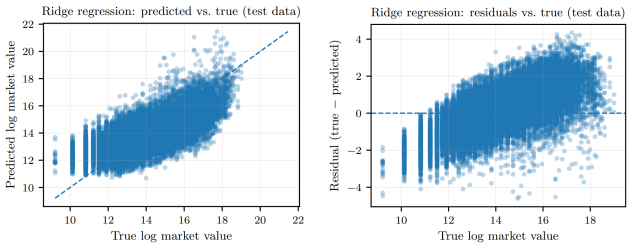

Figure 8: Diagnostic plots for Ridge regression on the core feature set evaluated on the test data. (a) Predicted versus true logarithmic market value. (b) Residuals as a function of the true logarithmic market value.

D. Neural Network on Core Features

Figure 9 shows validation R^2 across network architectures of varying depth and width for ReLU-activated networks. Performance varies smoothly across configurations, with shallow networks already achieving strong results and only marginal gains from increased depth or width. Similar behavior is observed for tanh-activated networks (see Figure 27 in Appendix). These results indicate that model capacity is not the primary limiting factor when using the core feature set.

The best-performing configuration achieves a validation R^2 of approximately 0.61 and corresponds to a network with two hidden layers and 64 units per layer. When evaluated on the held-out test set, this model attains an R^2 of 0.602 and an RMSE of 0.954 in logarithmic market value, representing a clear improvement over the linear Ridge baseline trained on the same features (Table I).

Despite this improvement, the relatively small variation in performance across architectures suggests that further architectural tuning alone is unlikely to yield substantial gains. Instead, these findings motivate the introduction of richer feature representations, explored in the next section.

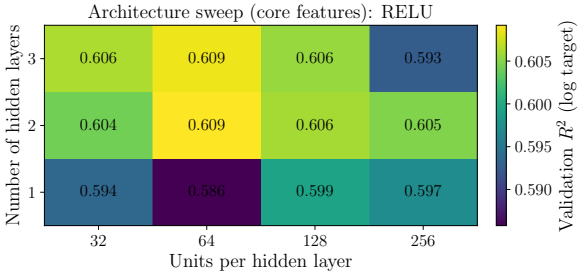

Figure 9: Validation R^2 for NNs trained on the core feature set, shown as a function of the number of hidden layers and hidden units per layer. Performance varies smoothly across architectures, with shallow networks already achieving strong results and diminishing returns observed for increased model complexity.

E. Extended Features: Nationality Effects

Results obtained using the core feature set show that a substantial fraction of the variance in player market value remains unexplained by both linear and non-linear models. Motivated by the exploratory data analysis, we therefore extend the feature set to include player nationality, encoded using one-hot variables.

While this substantially increases feature dimensionality, it introduces no additional temporal information. The extended feature set thus provides a controlled test of whether richer contextual information can improve predictive performance beyond what is achievable through model complexity alone.

1. *Ridge Regression*

Ridge regression is applied to the extended feature set to assess the effect of including nationality within a linear modeling framework. The regularization parameter is selected using the same group-wise cross-validation strategy as for the core feature set, yielding $\lambda = 1.21 \times 10^3$, as shown in Figure 24 in the appendix.

Compared to the core feature model, predictive performance improves on the held-out test set, with an R^2 score of 0.5484, indicating that nationality provides additional predictive signal beyond performance statistics and league context.

However, the overall gain remains moderate, suggesting that the linear structure of Ridge regression limits its ability to fully exploit the higher-dimensional feature space introduced by nationality indicators. Diagnostic plots for the extended Ridge model are shown in Figure 26 in the appendix.

2. *Neural Network*

The NN benefits more strongly from the inclusion of nationality information than the linear model, with an R^2 score of 0.6335. This indicates that the extended feature representation introduces additional structure that can be effectively exploited by non-linear models.

NNs are well suited to capture interactions between nationality, age, and performance-related variables that are inaccessible to linear approaches. These interactions are consistent with domain intuition, where player valuation depends not only on individual performance but also on broader market dynamics associated with player origin. Taken together, these results highlight the complementary roles of feature richness and model flexibility in predicting football player market values.

F. Recurrent Neural Network and Sequential Data

Figures 11 illustrate the training dynamics of the recurrent models in terms of R^2 and MSE. Both GRU and LSTM converge rapidly, with most performance gains achieved within the first training epoch. Subsequent epochs yield only minor improvements, indicating that the models quickly capture the dominant structure present in the data.

The validation metrics reported for the first epoch are computed after the model has already been updated during that epoch. Consequently, the initial validation MSE reflects a partially trained model, which explains why it is lower than the corresponding training error.

The observed rapid convergence suggests that the prediction task is relatively well-conditioned given the available inputs. Market values tend to evolve smoothly over time, and temporal information provides a strong constraint on future valuations. At the same time, the event-based input features are sparse for many players between valuation dates. With a richer and more complete set of recorded match events, more information could have been aggregated within each time interval, potentially reducing sparsity and allowing performance-related features to play a larger role in the prediction task.

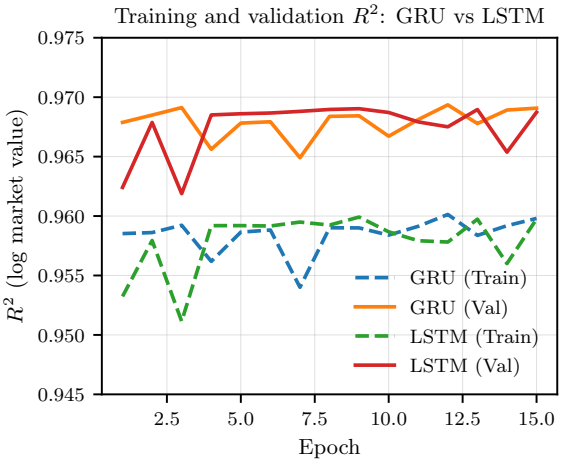

Figure 10: Training and validation R^2 across epochs for GRU and LSTM models. Both architectures achieve stable explanatory power after a few epochs, with validation performance closely tracking training performance, indicating limited overfitting.

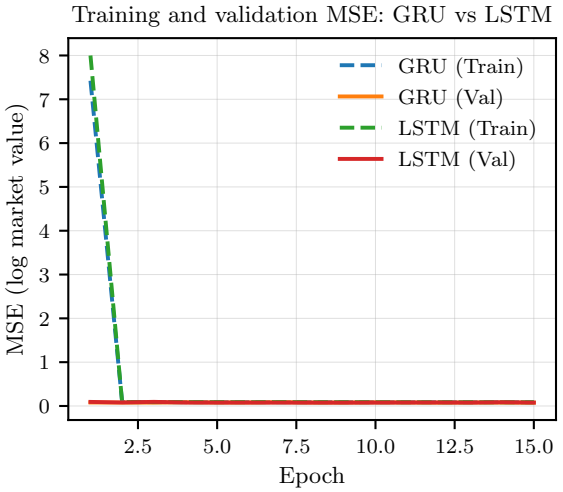

Figure 11: Training and validation MSE for GRU and LSTM models across epochs. After a sharp initial decrease from ~ 8 to ~ 0.09, both models converge rapidly, with training MSE stabilizing near 0.085 and validation MSE near 0.076 by the final epoch.

G. Model Comparison and Discussion

We conclude the results section by comparing predictive performance across all model classes and feature

representations. Table I summarizes test set results for Ridge regression, FFNNs, and RNNs.

For static feature representations, FFNNs consistently outperform Ridge regression on both the core and extended feature sets. This highlights the importance of modeling non-linear interactions between player attributes, performance statistics, and contextual variables. Extending the feature set with nationality improves performance for both models, with larger gains observed for FFNNs, indicating that richer contextual information is more effectively exploited by non-linear models.

These trends are reflected in Figures 12 and 13, which compare test set R^2 and RMSE across models and feature sets. In both cases, extended features lead to improved performance, while FFNNs achieve lower prediction error than linear baselines.

A distinct performance regime is observed for the RNN models trained on sequential data. Both GRU and LSTM architectures substantially outperform all static models, achieving test set R^2 values close to unity and RMSE values below 0.26 on the logarithmic scale. While GRU and LSTM perform nearly identically, these results demonstrate that incorporating temporal structure yields a large improvement over static representations.

The strong performance of the RNNs suggests that sequential information captures aspects of player valuation dynamics that are not accessible through static feature engineering alone. At the same time, the rapid convergence and similarity between GRU and LSTM indicate that the predictive task may be relatively well-conditioned given the available inputs. Future work could investigate whether richer or less sparse event data would shift predictive importance away from valuation history and toward performance-driven features.

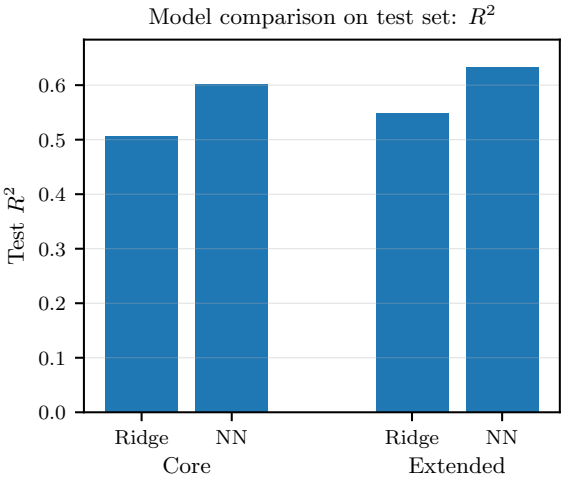

Figure 12: Comparison of test set R^2 scores across models and feature sets. NNs consistently achieve higher predictive accuracy than Ridge regression, with the largest gains observed when using the extended feature representation.

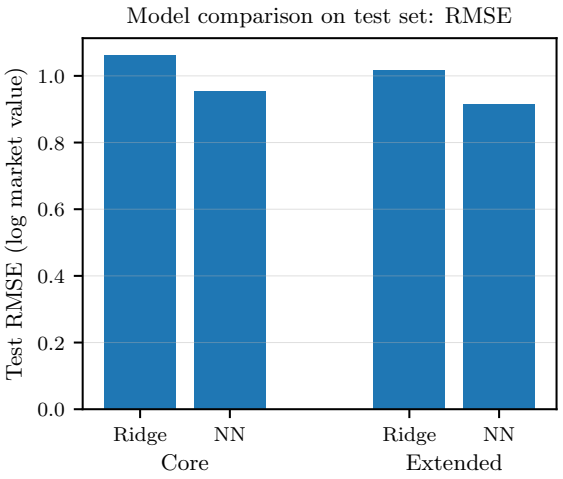

Figure 13: Comparison of test set RMSE across models and feature sets. Lower values indicate improved predictive performance. NNs achieve consistently lower error than Ridge regression, particularly when trained on the extended feature set.

H. Predicting UEFA Champions League 2024 Final

As an entertaining conclusion to the results section, we used the RNN models to predict the starting lineup of the Champions League 2024 finals between Borussia Dortmund and Real Madrid, shown in Figure 14. From the dataset, we see that the true values (y) are shown in whole 10 million euros, while our predictions (p) take any positive real values. Purely anecdotally we find that our model predictions are quite close to the true values, where we in 18/22 cases undervalue the footballer market values, except from the Borussia Dortmund keeper Kobel and midfielder Can, and Real Madrids midfielder Kroos and defender Nacho.

Of course, in this specific task we use data from the entire training and validation run from 2012 to 2023 to predict market values during the finals in june of 2024. One possible reason why our model underpredicts, is that all the player would have had 6 additional months to accrue their market values, and also done so leading up to the finals. As such, one might predict that most of their value would increase during 6 months of winning Champions League group stages, quarter finals, and semi finals.

IV. CONCLUSION

In this project, we investigated the extent to which professional football players' market values can be predicted using machine learning methods applied to performance data, player attributes, and contextual information. Starting from a static regression framework and progressively increasing both model complexity and data richness, we evaluated the performance of linear models, feed-forward

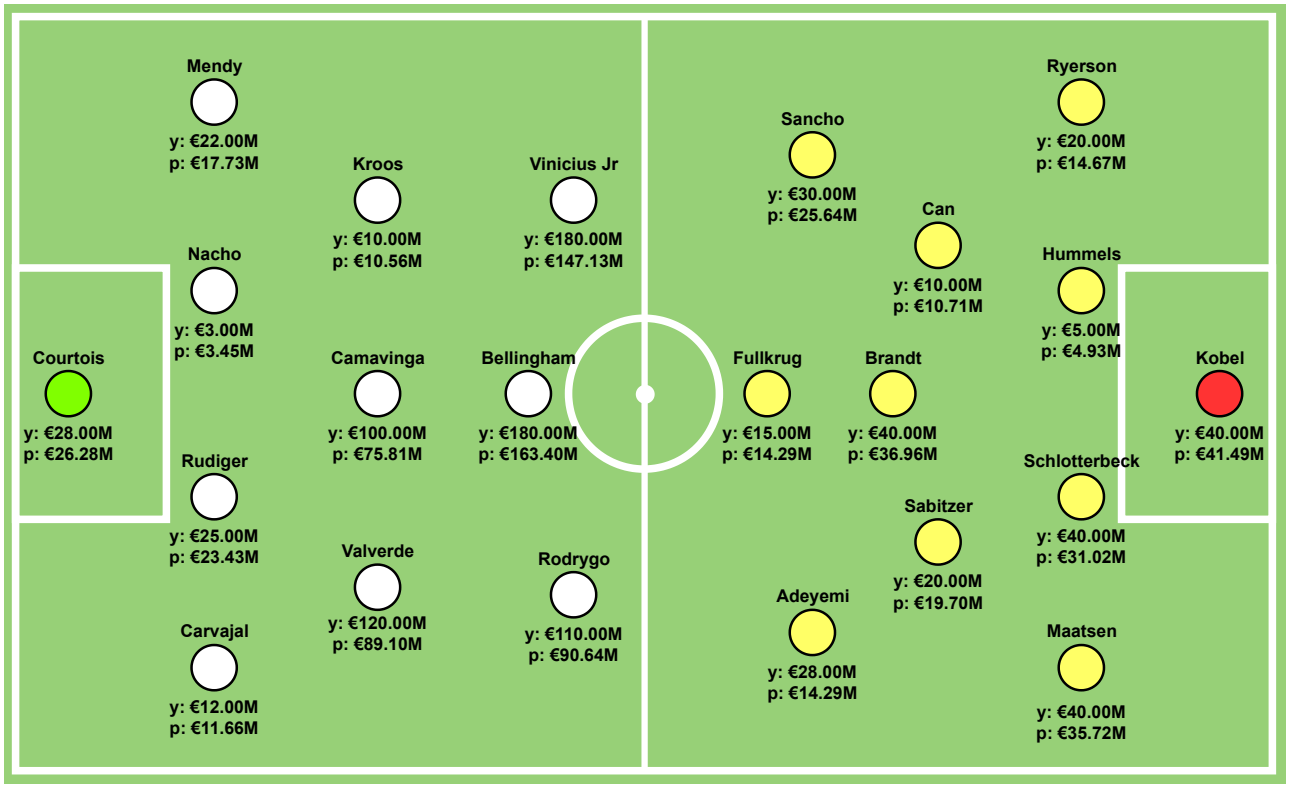

Figure 14: Predicted versus observed market values for the starting elevens in the 2024 UEFA Champions League final, visualized on the pitch. For each player, the true market value (y) and the RNN-predicted market value (p) are reported in millions of euros. This figure provides an intuitive, real-world illustration of the model's performance in a high-stakes elite setting, highlighting both its strong overall calibration and player-specific deviations.

neural networks, and recurrent neural networks on this task.

Using a core set of player attributes and aggregated match statistics, we found that feed-forward neural networks consistently outperform Ridge regression, demonstrating the importance of capturing non-linear interactions in football valuation data. Extending the feature set with nationality indicators led to further performance improvements for both model classes, with larger gains observed for neural networks. This indicates that contextual factors such as player origin encode additional information beyond individual performance metrics and league affiliation.

The most substantial improvement was achieved by explicitly modeling temporal structure through recurrent neural networks. By reformulating the dataset as a sequence of valuation intervals and incorporating historical market values, both GRU and LSTM architectures achieved near-perfect predictive performance on the held-out test set. These results highlight that player market value is strongly path-dependent and evolves smoothly over time, making sequential models particularly well suited for this prediction task. The comparable performance of GRU and LSTM suggests that the dominant temporal dynamics can be captured without the full complexity of an LSTM architecture.

Despite the strong predictive performance, several limitations should be acknowledged. The recurrent models rely heavily on previous market valuations, which may partly reflect self-reinforcing mechanisms in the transfer market rather than purely performance-driven changes in player value. In addition, match-level event data are relatively sparse between valuation dates for many players, limiting the influence of short-term performance fluctuations. Finally, the focus of this study is on predictive accuracy rather than causal inference, and the learned relationships should not be interpreted as direct drivers of market value.

Future work could address these limitations by incorporating richer and more granular performance data, such as advanced match statistics or team-level tactical context. Alternative temporal modeling approaches, including attention-based architectures or transformer models, may offer improved interpretability and flexibility. Extending the analysis to additional leagues, transfer windows, or valuation sources would also provide insight into the robustness and generalizability of the proposed methods.

Overall, this project demonstrates that combining expressive machine learning models with carefully con-

structed temporal representations yields substantial improvements in predictive performance for complex, real-world economic systems such as the professional football transfer market.

V. REFERENCES

Link to GitHub: github.com/egil10/FYSSTK3155

[1] David Cariboo. Football data from transfermarkt. Kaggle dataset, 2024. URL https://www.kaggle.com/datasets/davidcariboo/player-scores.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[3] Sebastian Raschka, Yuxi (Hayden) Liu, and Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*. Expert Insight. Packt Publishing, Birmingham, UK, 2022. ISBN 9781801819312.

[4] PyTorch Foundation. Pytorch: An optimized tensor library for deep learning. https://docs.pytorch.org/docs/stable/index.html, 2017. Accessed: 2025-11-10.

[5] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[6] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2024. URL https://www.R-project.org/.

[7] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686. URL https://tidyverse.org/.

[8] Hadley Wickham, Jim Hester, and Jennifer Bryan. *readr: Read Rectangular Text Data*, 2025. URL https://readr.tidyverse.org. R package version 2.1.6.

[9] Egil10. Fysstk3155. https://github.com/egil10/FYSSTK3155, 2025. GitHub repository.

[10] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2.

[11] The pandas development team. pandas-dev/pandas: Pandas, 2020. URL https://doi.org/10.5281/zenodo.3509134.

[12] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[13] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016.

[14] OpenAI. Chatgpt. https://chat.openai.com/. Accessed: 2025-11-09.

[15] xAI. Grok. https://grok.com/. Accessed: 2025-11-09.

[16] Google DeepMind. Gemini: A family of highly capable multimodal models. Online, 2025. URL https://deepmind.google/technologies/gemini/. Large language model developed by Google. Accessed via Gemini API.

[17] OpenAI. Chatgpt conversation (r code for descriptive stats), 2025. URL https://chatgpt.com/share/6943eba6-8198-8005-b9de-3af5e4e025a8. ChatGPT shared link / generative AI conversation.

[18] OpenAI. Chatgpt conversation (bibtex entry for shared link), 2025. URL https://chatgpt.com/share/6943ec16-f638-8005-94e2-0a8126e17c39. ChatGPT shared link / generative AI conversation.

[19] ChatGPT. Dataset aggregation code. https://chatgpt.com/share/694406bc-0be0-800e-bc80-d4db661cbc26, 2025. Initial aggregation script generated via ChatGPT.

[20] ChatGPT. Rnn market value prediction. https://chatgpt.com/share/69440695-fe08-800e-9713-83830ddd9634, 2025. Conversation and generated analysis/code accessed via ChatGPT.

[21] Cursor. Cursor. https://cursor.com/. AI coding assistant. Accessed: 2025-11-09.

[22] Google AI. Google gemini conversation (shared link), 2025. URL https://gemini.google.com/share/fab402816344. Google Gemini shared conversation / generative AI output.

[23] Cursor AI. Cursor: The ai code editor. https://cursor.com/, 2025. AI-assisted development environment for programming. Accessed October 2025.

[24] Google. Google antigravity. Agentic development platform and AI-powered IDE, 2025. URL https://antigravity.google/. Agent-first integrated development environment providing autonomous coding assistance.

[25] Microsoft. *Visual Studio Code*. Microsoft, 2025. URL https://code.visualstudio.com/.

VI. APPENDIX

A. Data tables

player_valuations.csv

- ⋆ player_id
- date
- market_value_in_eur
- current_club_id
- player_club_domestic_competition_id

players.csv

- ⋆ player_id
- first_name
- last_name
- name
- last_season
- current_club_id
- player_code
- country_of_birth
- city_of_birth
- country_of_citizenship
- date_of_birth
- sub_position
- position
- foot
- height_in_cm
- contract_expiration_date
- agent_name
- image_url
- url
- current_club_domestic_competition_id
- current_club_name
- market_value_in_eur
- highest_market_value_in_eur

game_events.csv

- ⋆ player_id
- game_event_id
- date
- game_id
- minute
- type
- club_id
- description
- player_in_id
- player_assist_id

processed data

- ⋆ player_id
- valuation_date
- y_raw
- y_log
- height_in_cm
- age_years
- is_big5_league
- foot_B,L,R,UNK
- pos_ATT,DEF,GK,MID,MISSING
- cumulative_goals,assists,yellow_cards,
- cumulative_red_cards,sub_in,sub_out
- lag_10_goals,assists,yellow_cards,
- lag_10_red_cards,sub_in,sub_out
- nat_Albania,Andorra,...,Zimbabwe

Figure 15: Data entity table

B. Descriptive statistics

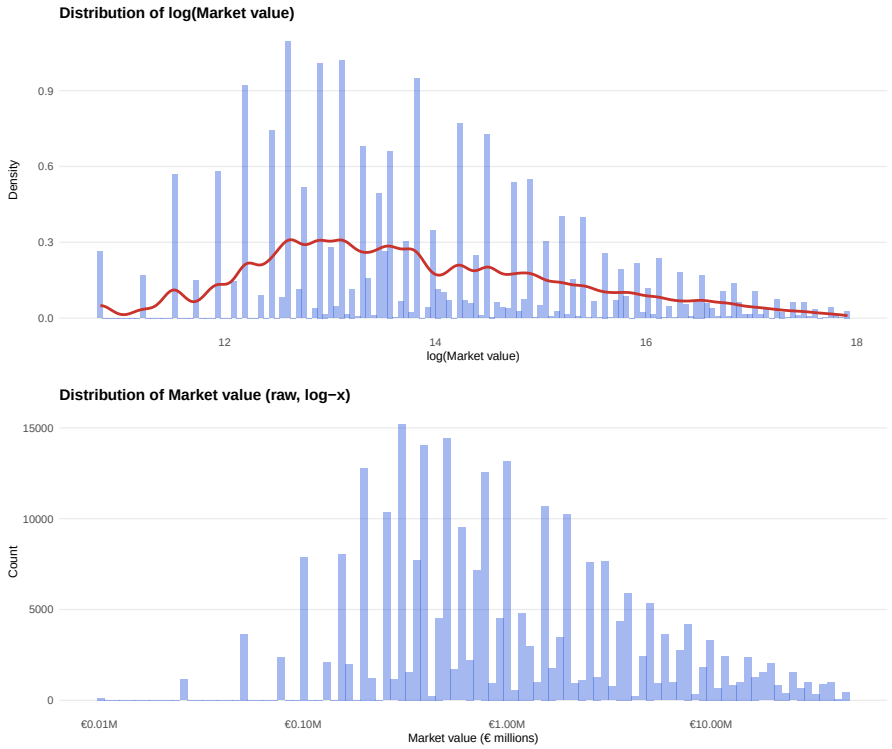

Figure 16: Distribution of professional football player market values, shown on both the raw scale and the logarithmic scale. The log transformation highlights the heavy right tail and improves comparability across players.

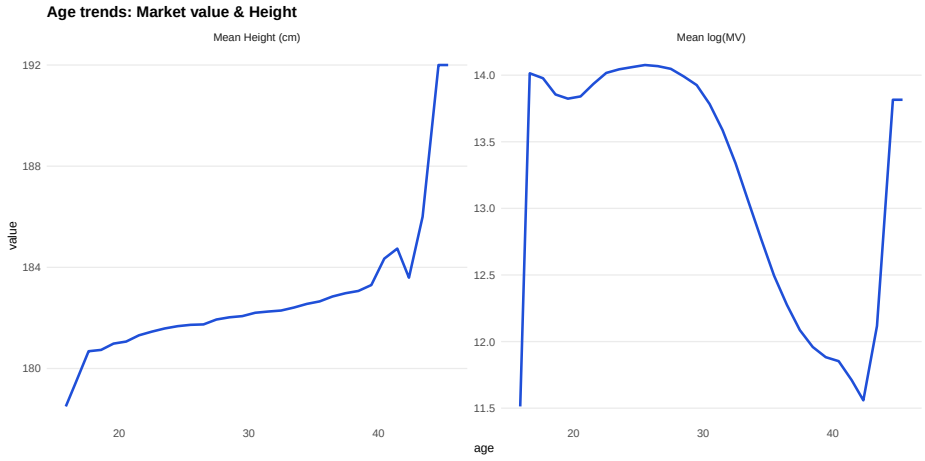

Figure 17: Average player market value and height as a function of age. Market values exhibit a clear peak in the prime playing years, while height remains stable across age cohorts.

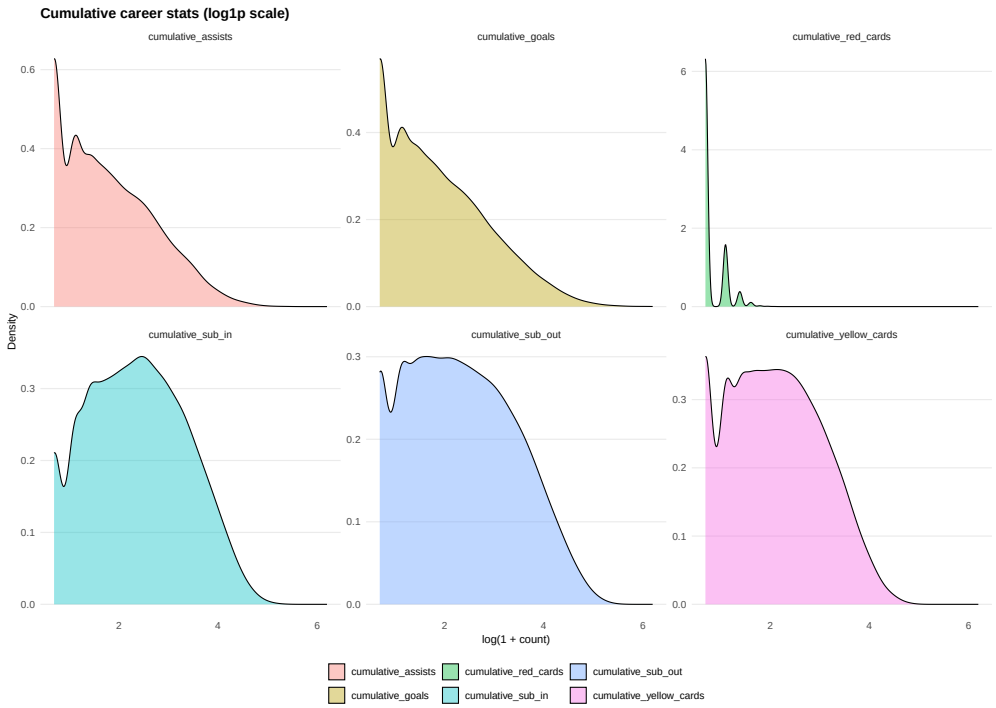

Figure 18: Distributions of cumulative career performance statistics (e.g., goals, assists, appearances), displayed using a $\log(1+x)$ transformation to account for extreme values.

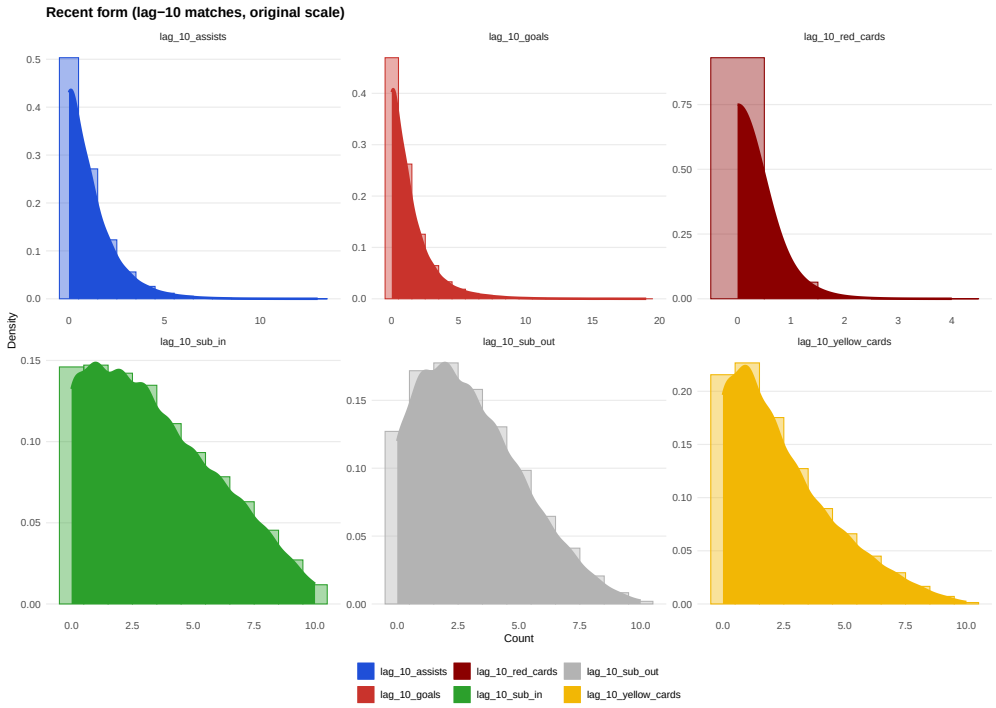

Figure 19: Distributions of recent player performance metrics based on the last ten matches (lag-10 window), shown on the original measurement scale.

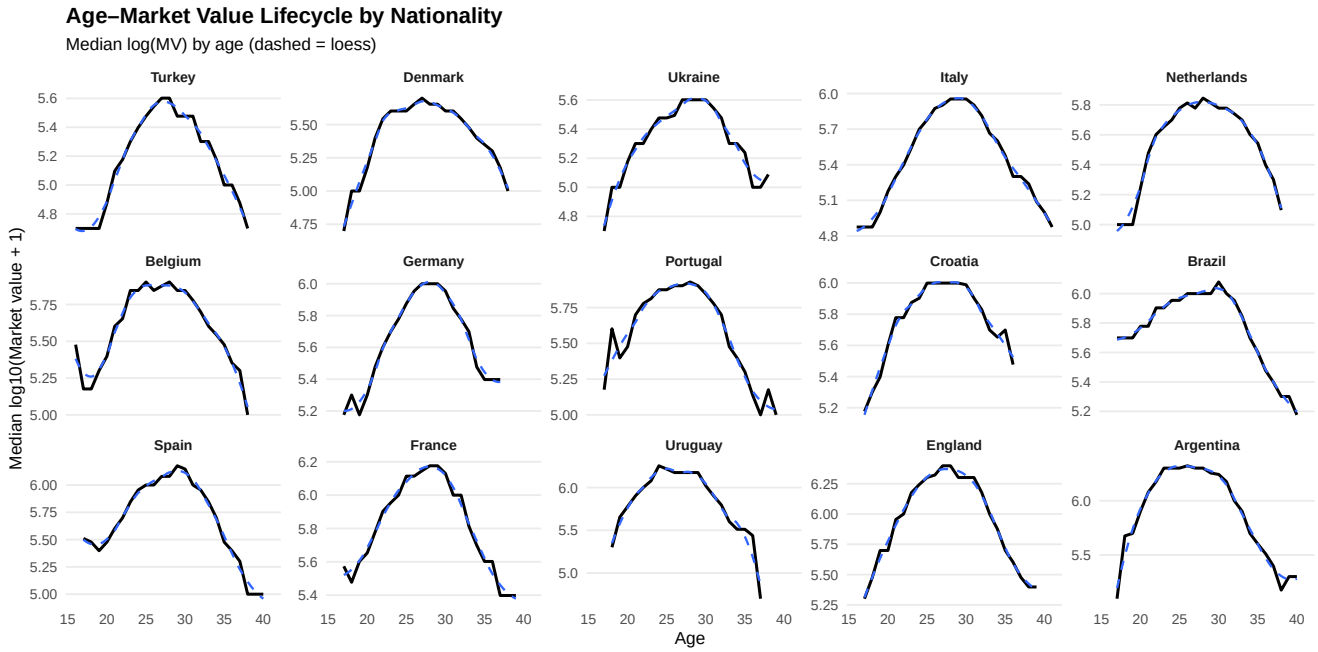

Figure 20: Median player market value by country and age group, highlighting differences in valuation profiles across national football systems.

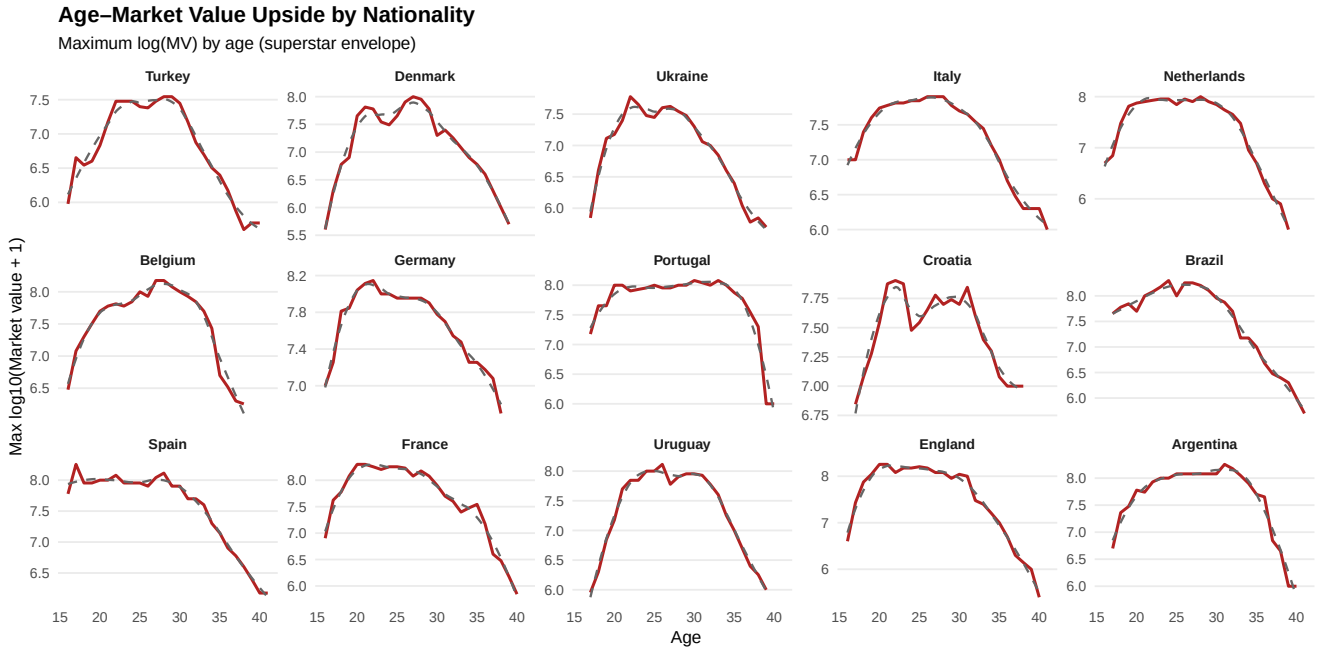

Figure 21: Maximum observed player market value by country and age group, illustrating the upper tail of talent valuation across countries.

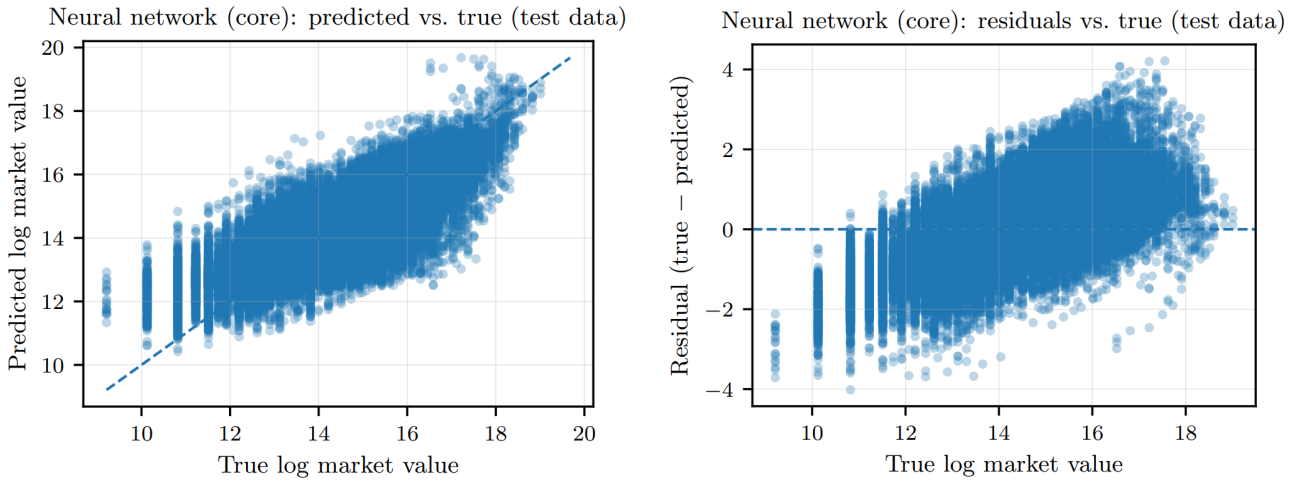

Figure 22: Predicted versus true logarithmic market values for the neural network model trained on the core feature set, evaluated on the test data. The alignment of points along the diagonal indicates the model's predictive accuracy across the range of observed market values. Residuals as a function of the true logarithmic market value for the neural network trained on the core feature set, evaluated on the test data. The plot is used to assess systematic biases, heteroscedasticity, and deviations from model assumptions.

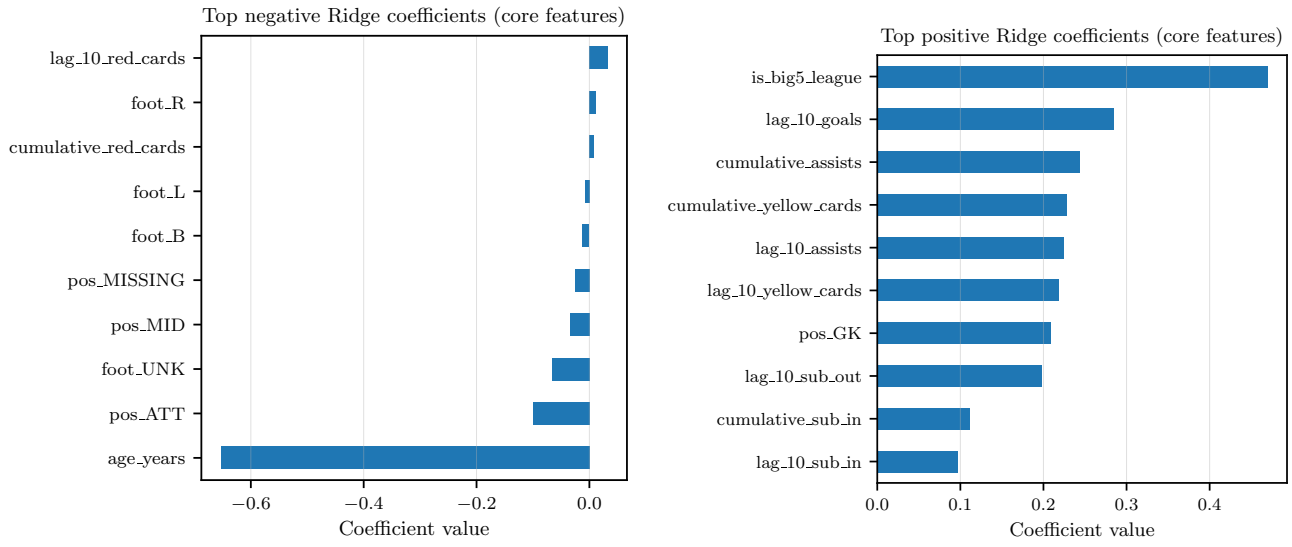

Figure 23: Top negative coefficient estimates from the Ridge regression model trained on the core feature set. The figure highlights features that are most strongly associated with a decrease in the predicted logarithmic market value. Top positive coefficient estimates from the Ridge regression model trained on the core feature set. The figure highlights features that are most strongly associated with an increase in the predicted logarithmic market value.

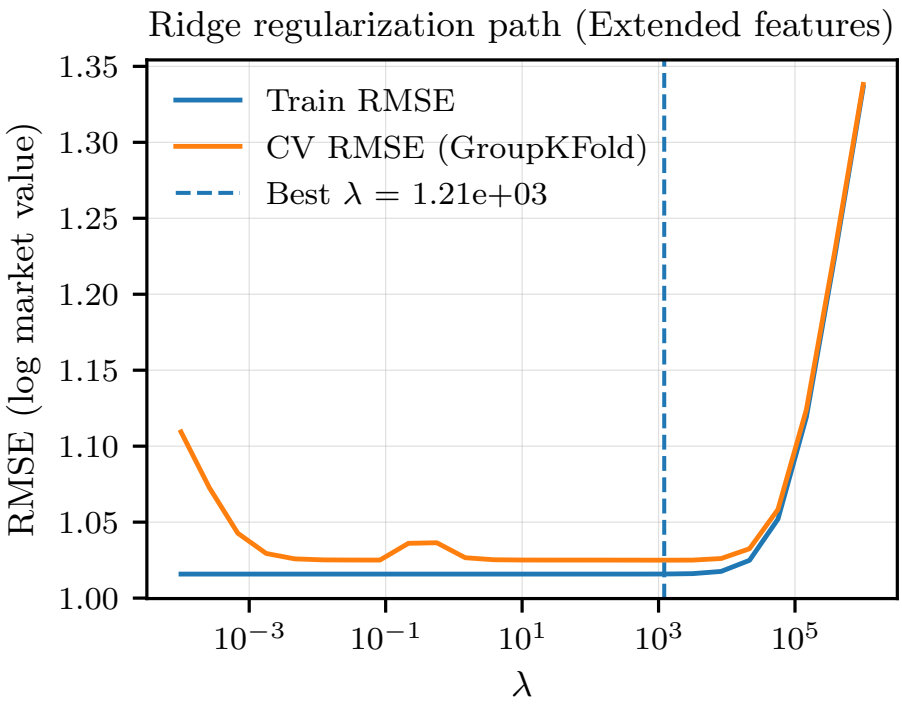

Figure 24: Cross-validation curve for Ridge regression on the extended feature set, showing the relationship between the regularization parameter λ and the validation error. The selected value of λ corresponds to the minimum cross-validated error.

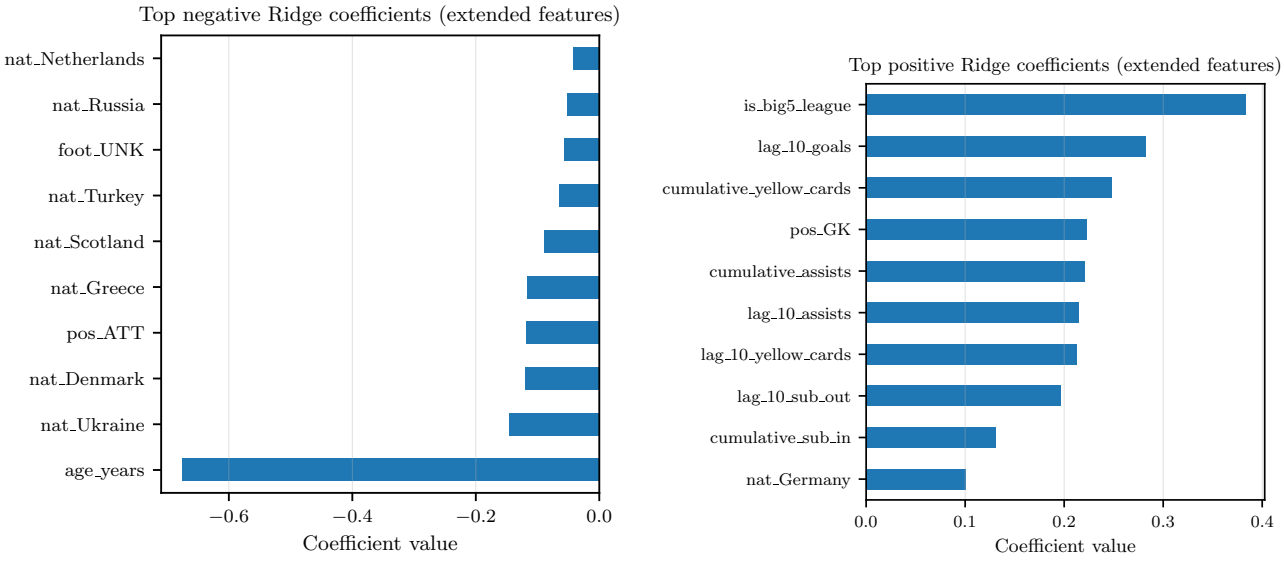

Figure 25: Top negative coefficient estimates from the Ridge regression model trained on the extended feature set. The figure highlights features that are most strongly associated with a decrease in the predicted logarithmic market value. Top positive coefficient estimates from the Ridge regression model trained on the extended feature set. The figure highlights features that are most strongly associated with an increase in the predicted logarithmic market value.

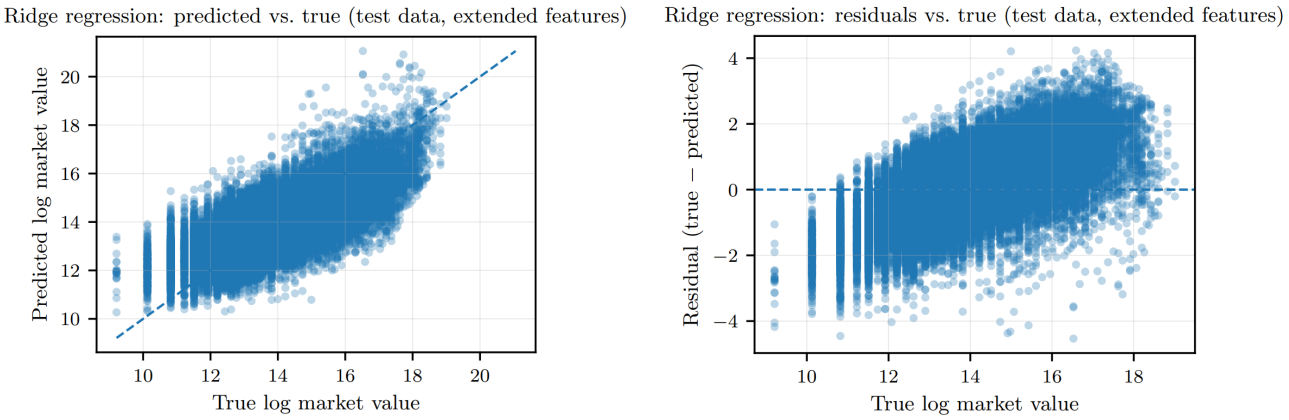

Figure 26: Diagnostic plots for Ridge regression on the extended feature set evaluated on the test data. (a) Predicted versus true logarithmic market value. (b) Residuals as a function of the true logarithmic market value.

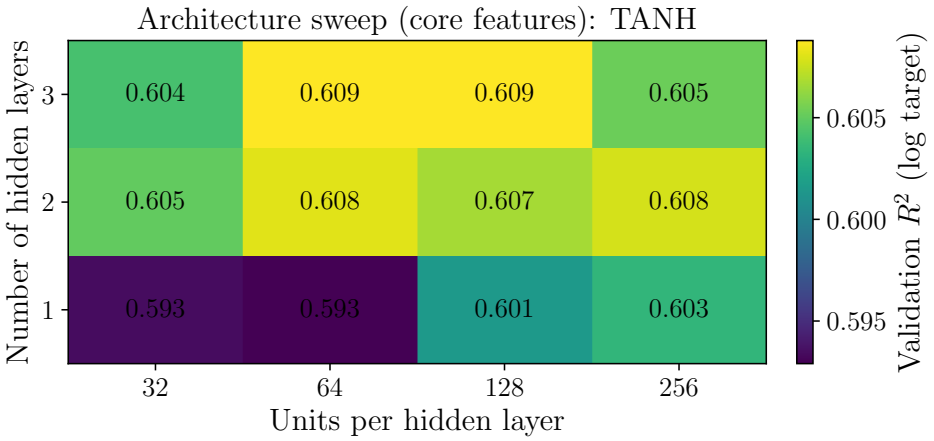

Figure 27: Heatmap visualization of the neural network architecture trained on the core feature set using `tanh` activation functions. The figure illustrates the depth of the network, the number of neurons in each hidden layer, and the overall connection structure of the final model configuration used for prediction.

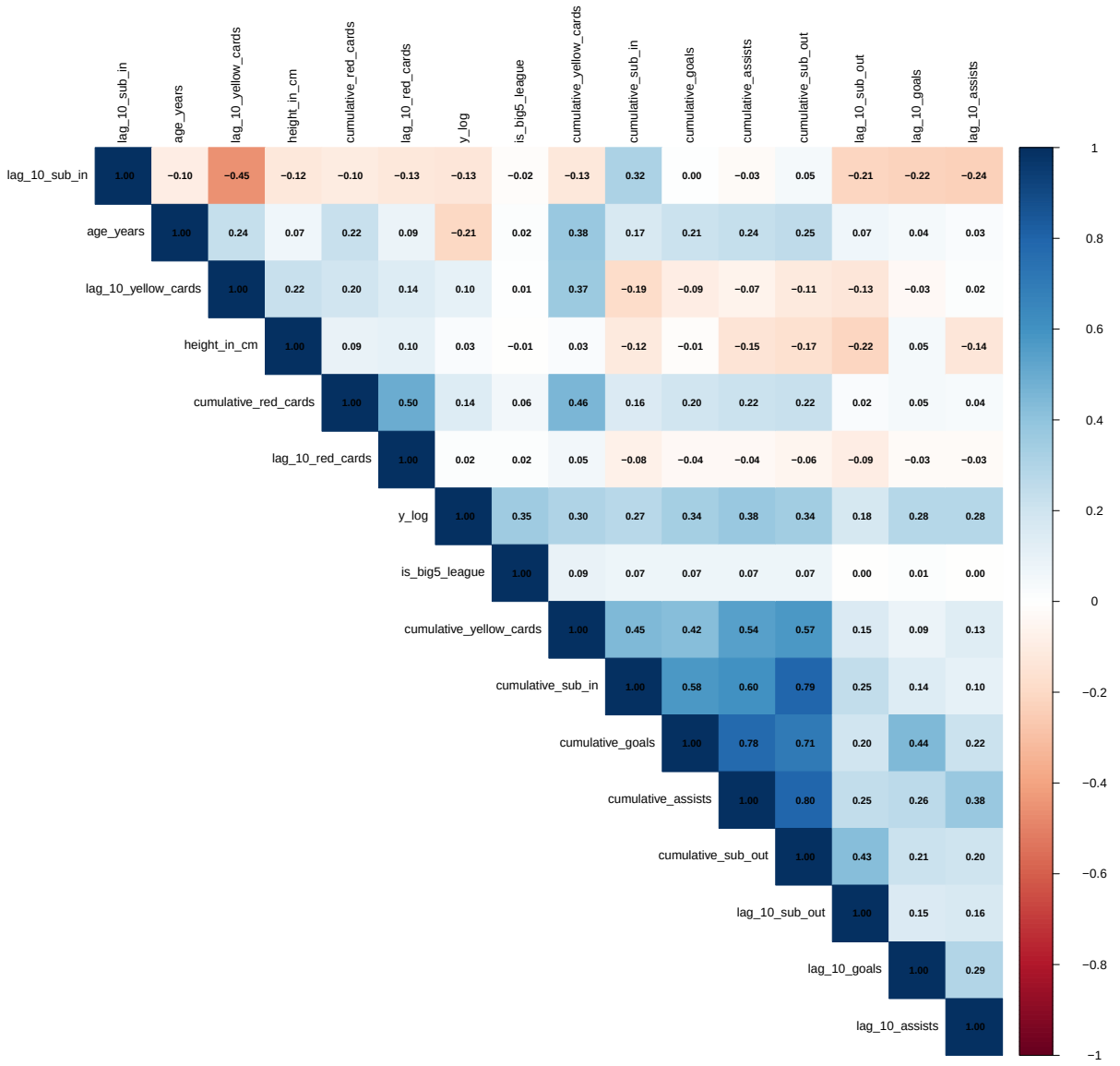

Figure 28: Correlation matrix showing pairwise correlations between model features and log-transformed player market value. Strong associations highlight key drivers of valuation.