BEAT THE BOOKIE

A PREPRINT

Group Name: `J (COMP0036 2021-22)`
Department of Computer Science
University College London
London, WC1E 6BT

December 22, 2021

1 Introduction

In this report, we detail our approach to predicting the results of the ten future English Premier League (EPL) football matches taking place on the weekend of 15 January 2022 using Machine Learning (ML) techniques. A stratified 5-fold cross-validation on the final match result classifier yields a mean macro-averaged F_1 score of 43.07%, taken across all the folds, which is comparable to the 53% accuracy of bookmakers as detailed in the assignment brief.

Firstly, additional factors and metrics that could contribute towards the success of a team in a football match, such as per-player match-time data and seasonal player market values, were researched to inform the data gathering process. Such data was used to augment the originally provided training dataset. Any missing values in the newly combined dataset were then estimated via multivariate imputation.

Furthermore, several additional features were engineered and experimented with, such as the proportion of shots that were on target as well as the ratio between a team's number of interceptions and the opposing team's total number of passes. Correlation matrices were generated as a useful and visual way of identifying and selecting the features contributing the most towards the variance in the variables to be predicted.

To use the dataset in training, we transformed the match dataset into a new format, which contains entries for each team at each match, prior to shuffling and standardising it to remove the mean and scale the data to unit variance.

Four regression models were trained to predict the following target variables for a team based on their match-time performance and their players' seasonal market values: the number of goals the team could be expected to win (`xG-won`) and concede (`xG-con`); their expected number of interceptions per opponent's pass (`xDef`) as an extra defensive metric; and their expected proportion of shots that were on target (`xOff`) as an extra offensive metric.

For each team, post-match and hence pre-match Elo-like [1] scores, corresponding to each of these four variables, were computed to give a measure of each team's cumulative performance over time in the respective areas.

Finally, a multi-class match result classification model was trained on the Elo-like scores, and was then used to infer the results of the matches in January 2022, i.e. whether the home (`H`) or away (`A`) team wins, or the match is a draw (`D`). The final predictions produced by the model may be found in the Results section of this report.

This report is also accompanied with a Jupyter Notebook containing relevant Python source code for each step, along with in-line documentation.

2 Data Transformation & Exploration

2.1 Aggregating Across Multiple Data Sources

As part of the initial data exploration phase, the provided training data was examined, manipulated and visualised in Microsoft Excel. An initial linear support vector classification (`LinearSVC`) model trained on this data was found to be significantly overfitting (as detailed in the Prediction Strategy section), so we gathered more data as a way of improving the generalisability of the model.

Additional per-player, per-match and per-season data was gathered from various data sources, detailed in the Research and Data Gathering section. The originally provided training dataset and the new data were imported into Pandas dataframes, aggregated as appropriate and merged.

The data sources used varying date formats and team naming conventions — e.g. "Hull City" vs. "Hull" — which were standardised using a custom date parser and mapper in order to properly match rows across the Pandas dataframes.

Firstly, the 2021-22 season data was appended to the provided training data. Then, the passing, possession & attendance data was left-merged on the team names and match date in Pandas to produce an input dataframe.

Individual player performance data was aggregated across each match to compute additional team performance metrics (e.g., by summing the number of assists) and left-merged into the input dataframe. The exact aggregations for each attribute can be seen in Section 2.2 of the Jupyter Notebook.

Finally, seasonal player market values and wages were averaged across a team and similarly left-merged into the input dataframe. The Python code for the whole procedure can be found in Section 2.2 of the Jupyter Notebook.

2.2 Multivariate Imputation of Missing Values

Some features were only available for more recent matches, leading to missing data points in our combined dataset. As only a small proportion of the data points were missing, we performed a multivariate imputation using the `IterativeImputer` class in Scikit-Learn to estimate the missing values by modelling each feature as a function of all the others.

2.3 Splitting Match Statistics into Team Statistics

In order to train regression models to predict the `xG-won`, `xG-con`, `xDef` and `xOff` metrics for a team given their and their opposing team's performance and players' seasonal market values, we doubled the data so to create rows from the perspective of both the home and away teams (as seen in Section 3.2 of the Jupyter Notebook).

2.4 Feature Engineering

Domain knowledge was used to engineer additional features from the combined raw dataframe. A few examples of these features are (note that these are all pre-match matrics):

Shots on Target Proportion Each team's on target shots as a ratio of their total attempts.

Interceptions per opponent's pass Each team's interceptions made as a ratio of the opposing team's total passes.

Pressing Actions Each team's number of interceptions + winning tackles.

We also experimented with features that were computed as ratios of metrics corresponding to the home and away teams; however, as detailed in the Visualising the Training Dataset & Feature Selection section, many were removed as they contributed little to the variance of the variables to be predicted. The complete list of engineered features can be found in Section 3.1 of the Jupyter Notebook.

2.5 Visualising the Training Dataset & Feature Selection

Correlation matrices were produced to visualise the linear relationship between pairs of input features, including engineered features we experimented with, e.g., red and yellow card, penalty card score and possession ratios. The correlation matrices informed our selection of features that contributed the most towards the variance of the target variables we were interested in.

An example such matrix can be seen in Appendix A (Correlation Matrix) and Section 3.3.1 of the Jupyter Notebook.

2.6 Shuffling & Scaling

Prior to training, the dataframe was shuffled to prevent the chronological ordering of the data inadvertently introducing bias. Afterwards, the data was standardised by removing the mean, thereby centring the data around zero, and scaling it to unit variance using the `StandardScaler` class in Scikit-Learn.

Initially, the `MinMaxScaler` class in Scikit-Learn was used to scale feature values to the range $[0, 1]$; however we switched to the `StandardScaler` class as this improved the convergence of our models.

3 Methodology Overview

3.1 Research and Data Gathering

The details of the additionally gathered data discussed in the Aggregating Across Multiple Data Sources section can be found below:

- Per-match passing, possession & attendance data

  Obtained from fbref.com.

  A relationship between the home advantage and attendance has been suggested [2], which we wanted to consider in our models, despite the lack of spectators in the 2020-21 season due to the Covid-19 pandemic.
- Per-player per-match performance data

  Obtained from fbref.com.

  Players are vital to a match's success, so considering their performance metrics could improve our models.
- 2021-22 season data (including the above data)

  Obtained from fbref.com.

  As a new entrant to the EPL in the 2021-22 season, *Brentford* was not found in the provided training data [3].

  Consequently, to calculate a fairer set of Elo scores for *Brentford* to use for inference rather than using the default starting Elo values, we scraped data from the latest 2021-22 season until 6th December 2021.
- Seasonal player market values and wages

  Obtained from sofifa.com.

  Assuming an efficient market, the market value of a player can be expected to encode information on the value a player could add to a team and their desirability [4].

We wrote scripts to parse the HTML of the respective websites and export the scraped data as `CSV` files. An example of such a scraper can be found in Section 2.1 of the Jupyter Notebook.

The generated `CSV` files were then merged into a single `CSV` file with features aggregated where appropriate as detailed in the previous Data Transformation & Exploration section and shown in Section 2.2 of the Jupyter Notebook.

3.2 Background Reading

An interesting approach was that of Corentin Herbinet and Dr Jonathan Passerat-Palmbach [5], who used in-game match events to develop an "expected goals" (hereinafter `xG`) won by a team metric. Firstly, they separated each shot in their dataset by type (e.g. normal shots, volleys, headers, direct/indirect free kicks, etc.). From this, they trained a classifier for each shot type to predict whether a given shot resulted in a goal from the distance and angle of the shot to the goal.

These values were then adjusted for the team's goal lead during the match, the time into the match of each shot and the number of players on the pitch for that team. The team's shot-based match `xG` value was then defined as the sum of their adjusted shot `xG` values over the course of the match. This was combined with some "non-shot-based" metrics — such as possession, the number of corners and the number of yellow cards — to train a match `xG` regressor.

For each match, the `xG` won by a team and their opponent were used to compute defensive and offensive Elo ratings, respectively. Additional such defensive and offensive ratings that were only updated when a team was playing at home or away, respectively, were also generated. These six ratings were finally used to train both a match result classifier and a regressor to predict the score.

Another interesting methodology worth mentioning is that of FiveThirtyEight [6], who developed a soccer power index (SPI) to estimate a team's overall strength: an initial preseason SPI rating is determined from a team's rating at the end of the previous season and the team's market value, derived from their players' potential transfer prices; then, after each match, it gets updated using their shot-based and non-shot-based expected goal models based on the number of goals the team could be expected to win and concede against an average opponent on a neutral pitch as in [5], but with an extra adjusted goals metric instead of adjusting the `xG` values. They then project the match scores from the SPI ratings to model goal-scoring for each team as two Poisson processes in order to finally calculate the likelihood of a win, loss or draw for each team.

When researching potential defensive metrics, we came across various ones involving ratios between so-called "pressing actions", such as interceptions and tackles, and passes [7]. An adaptation of one such defensive metric ended up being used for both feature engineering and the implementation of the `xDef` model.

3.3 Prediction Strategy

We implemented all our models using Scikit-Learn [8] because it is widely used and has excellent documentation, which allowed us to speed up development.

At the beginning of the project, an initial linear support vector classification [9] model was trained on the provided training dataset to gauge its performance. In a stratified 5-fold cross-validation, the mean macro-averaged F_1-score across the folds was 97.36% (see the Results section), indicating that it was heavily overfitting; therefore, it seemed appropriate to investigate alternative match result prediction strategies and gather additional data.

Inspired by the approach of Corentin Herbinet and Dr Jonathan Passerat-Palmbach [5], and FiveThirtyEight [6], we trained two linear support vector regression models [10] on the doubled training dataset to predict the expected goals won `xG-won` and conceded `xG-con` by a team, respectively, given their and their opponent's performance, as well as their and the opposing team's mean player values.

We then sought to expand on this approach by developing two further `ElasticNet` [11] regression models on the same dataset to predict additional defensive (`xDef`) and offensive metrics (`xOff`) because we thought we were reaching the limit of the total information that could be encoded into a single Elo rating. Applying our research and domain knowledge, these were chosen as the number of interceptions per opposing team's pass and the proportion of shots that were on target, respectively.

These four expected metrics predicted by our models were then associated with their own corresponding Elo ratings. These were updated post-match for each team depending on the extent to which they surpassed the models' expectations for their performance according to the following formula:

$$\text{Elo}_{i+1} = \text{Elo}_i + K \cdot (\text{actual} - \text{expected}), \quad \text{Elo}_0 = 1000$$

For `xG-won` and `xG-won`, we modelled K as a function of the number of years since the match took place so that more recent matches would contribute more towards the `xG-won`- and `xG-con`-derived Elo scores. The K functions used, where δ is the number of years since the present, are the following:

$$K_{\texttt{xG-won}}(\delta) = 100e^{-0.3\delta}, \quad K_{\texttt{xG-con}}(\delta) = -50e^{-0.3\delta}$$

$$K_{\texttt{xDef}}(\delta) = 1000, \quad K_{\texttt{xOff}}(\delta) = 100$$

Having calculated the post-match Elo values for each team, we could then use the pre-match Elo values in order to train a multi-class K-Nearest Neighbours classifier using the `KNeighborsClassifier` [12] class in Scikit-Learn to predict whether the home (`H`) or away (`A`) team would win, or they would draw (`D`).

Finally, we used the most up-to-date post-match Elo values for each team as the pre-match Elo values input into our classifier to infer the results of the matches we were seeking to predict in January 2022. Details on why these specific models were chosen can be found in the Model Selection & Hyperparameter Tuning and Results sections.

3.4 Alternative approaches

We attempted various alternative approaches and techniques but decided against these due to worse performance.

Firstly, had we remained with our initial approach of using a single multi-class classifier (see the Prediction Strategy section) to predict match results directly, we could have then imputed the rest of the values in the test set for each team to predict the match results in January. We decided against this strategy when our initial model was heavily overfitting.

We also tried using Principal Component Analysis (PCA) [13] as a technique to project the training dataset into a lower dimensional subspace with maximal variance and hence reduce the complexity of our models using the `PCA` class in Scikit-Learn [8]. This resulted in the mean cross-validated macro-averaged F_1 scores of our models decreasing, which could have been a result of non-linear relationships between features. Thus, we abandoned this strategy.

We wanted to test the approach in [5] to use additional Elo scores that would only update after home and away matches; however, after calculating these for our `xG-won`-, `xG-con`-, `xDef`- and `xG-Off`-derived Elo scores and inputting them into our classification model, the macro-averaged F_1 either decreased slightly or was unchanged, so we did not pursue this further to maintain a simpler model. Nevertheless, our code to calculate these values has been made available in Section 7 of the Jupyter notebook.

4 Model Training & Validation

4.1 Model Selection & Hyperparameter Tuning

The initial implementation of our strategy adopted the `LinearSVR` and `LinearSVC` models in Scikit-Learn [9] as they were quick to train while still producing satisfactory predictions. We then performed a random and a grid search over the hyperparameter space using `RandomSearchCV` [14] and `GridSearchCV` [15] to improve the prediction quality of our models. Grid searching was our preferred method of the two, as it enabled us to test our code over a more consistent set of hyperparameters in development, allowing for faster iterative improvements.

Our next step involved implementing the `BayesSearchCV` class from the `scikit-optimize` package [16] to perform a Bayesian optimisation over the hyperparameter space for a wide variety of model types in order to improve our `xG-won`, `xG-con`, `xDef` and `xOff` regressors, the results of which are shown in table 1 for `xG-won` and `xG-con`.

While the tree-based regressors – such as the extreme gradient-boosted regressor `XGBoostRegressor` [17] and `RandomForestRegressor` [18] – seemed to give the highest mean R^2 value across the cross-validation folds, possibly due to overfitting, the resulting Elo scores seemed improbable with many teams not having performed strongly historically ranked highly, and the prediction quality of the match result classifier decreased. Moreover, when testing the non-tree-based model types for `xG-won` and `xG-con` in combination with our other models, a `LinearSVR` found via our grid search translated into the highest final match classifier macro-averaged F_1 score.

On the other hand, the best models for `xDef` and `xOff` when used in combination were found to be two `ElasticNet` models, which were the best non-tree-based models found in the Bayesian optimisation.

Model	**Mean Fit Time (s)**	**Mean Test R^2 Score**
`XGBRegressor(n_estimators=326, gamma=2.065831918109364, learning_rate=0.16335721089253233, max_depth=7, reg_alpha=0.00037570902238116463, reg_lambda=15.64414141772379)`	49.6	92.71%
`RandomForestRegressor(bootstrap=True, max_depth=402, max_features=auto, min_samples_leaf=6, min_samples_split=5, n_estimators=3052)`	457.0	91.77%
`ElasticNet(alpha=0.00016229434841034116, l1_ratio=1e-06, max_iter=2000)`	2.58	89.63%
`Ridge(max_iter=2000, alpha=2.108346459453851)`	0.0250	89.63%
`Lasso(max_iter=2000, alpha=0.001)`	0.740	89.60%
`SGDRegressor(max_iter=2000, alpha=0.0026880211421797875, epsilon=1.1434791445538757, learning_rate=invscaling, penalty=l2)`	0.0356	89.40%
`SVR(C=38.209242257769766, gamma=0.00010553595238726702, max_iter=2000, degree=3, kernel=rbf)`	5.47	88.87%
`LinearSVR(C=0.0036254778918504075, max_iter=2000)`	0.129	88.20%
`LogisticRegression(max_iter=2000, C=81.18627273985055, penalty=l2)`	33.7	88.09%
`AdaBoostRegressor(learning_rate=0.050611810172046065, n_estimators=3217)`	38.1	71.73%
`KNeighborsRegressor(n_neighbors=9, p=1.879543986431898)`	0.218	66.75%

Table 1: Best model hyperparameter tunings for `xG-won` and `xG-con` found via Bayesian optimisation.

4.2 Evaluation

The evaluation metric used to score the regressors (`xG-won`, `xG-con`, `xDef`, `xOff`) was the coefficient of determination R^2, as it could evaluate the extent to which the variance in the corresponding outputs were predictable from the training data using each model.

The evaluation metric for the classifier was the macro-averaged F_1-score, as it performs better than other metrics such as the accuracy score (which only measures the correct predictions percentage), because it is the harmonic mean of the precision and recall, weighing both the precision (percentage of predicted positives are actually positive) and recall (percentage of actual positives are predicted positive) equally [19]. Macro-averaging was chosen as we did not have a significant class imbalance between home-win, draw and away-win in our training data.

All models were evaluated using their corresponding evaluation metric and stratified k-fold cross-validation with $k = 5$. If we had only split the data into a single training and validation set, there would be a higher chance of overfitting to the validation set, weakening the generalisability of our models. Cross-validation is more suitable therefore as it trains and validates the model over different folds of the dataset. Stratified cross-validation improves this further so that each fold would be more representative of the dataset. Empirical analysis and standard practice suggests that $k = 5$ or $k = 10$ is a good number of folds to use, so 5 was chosen to also take into account speed and performance [20].

5 Results

5.1 Evolution of Approach

Table 2 details, in-order, the evolution of our approach, with rows showing the results of the cumulative changes made.

Progress Summary	**Regressor(s)**			**Classifier F_1**
	Model	**Type**	**R^2**	**(macro-avg.)**
Trained a `LinearSVC` on the provided training data, which led to significant overfitting and poor generalisability.	-	-	-	97.36%
Added the scraped passing, possession & attendance data to the training data, which reduced overfitting slightly.	-	-	-	53.39%
Created an `xG-won` Elo score regressor.	`xG-won`	`LinearSVR`	54.17%	33.35%
Added the scraped player data, which improved the regressor performance greatly and classifier performance slightly.	`xG-won`	`LinearSVR`	79.51%	34.28%
Added home and away Elo scores as explained in the Alternative approaches section.	`xG-won`	`LinearSVR`	82.73%	33.14%
Added the scraped matches from the 2021-22 season to the dataset. Created new `xG-con`, `xDef` and `xOff` regressors to calculate their corresponding Elo-like scores. Began to weight recent games higher in the Elo calculation.	`xG-won`	`LinearSVR`	68.34%	37.87%
	`xG-con`	`LinearSVR`	68.44%	
	`xDef`	`LinearSVR`	92.03%	
	`xOff`	`LinearSVR`	81.11%	
Tried using `LinearSVR` and `ElasticNet` regressors with a KNN classifier, to try improve convergence performance.	`xG-won`	`LinearSVR`	68.36%	41.66%
	`xG-con`	`LinearSVR`	68.34%	
	`xDef`	`ElasticNet`	78.87%	
	`xOff`	`LinearSVR`	80.80%	
Used `LinearSVR`s again and stop using home- & away- Elo scores, as they were detrimental to the overall classifier performance.	`xG-won`	`LinearSVR`	68.43%	43.01%
	`xG-con`	`LinearSVR`	68.42%	
	`xDef`	`LinearSVR`	92.04%	
	`xOff`	`LinearSVR`	81.36%	
Added player market values data.	`xG-won`	`LinearSVR`	88.08%	42.36%
	`xG-con`	`LinearSVR`	88.08%	
	`xDef`	`LinearSVR`	91.46%	
	`xOff`	`LinearSVR`	81.02%	

Used `LinearSVR` and `ElasticNet` regressors with a KNN classifier again.	`xG-won`	`LinearSVR`	88.10%	42.80%
	`xG-con`	`LinearSVR`	88.06%	
	`xDef`	`ElasticNet`	92.79%	
	`xOff`	`ElasticNet`	81.23%	

Table 2: Evolution of Approach.

Model Selection is discussed in more detail in the Model Selection & Hyperparameter Tuning section.

In summary, the experimentation showed that different models suited different regressors better and that the addition of data that had been scraped proved to improve the classifier's ability to generalise.

In particular, the player market values improved the R^2 score of the `xG-won` and `xG-con` regressors significantly, supporting our intuition outlined in section Research and Data Gathering. The inclusion of the scraped player data also greatly improved the R^2 score of the `xG-won` regressor, showing the benefits of using the Elo scores.

The experimentation also allowed recognition of what was detrimental to the models and why. For example, the inclusion of the home- and away- Elo ratings negatively affected the F_1 score. This could have been due to the new Elo ratings introducing a non-linear separation, making it more difficult for the model to separate data classes.

5.2 Final Classification Model Statistics

The final run of the script produced a classifier F_1 score of 43.07% and the following regressor R^2 mean (SD) scores:

`xG-won`: 0.8808 (0.0082); `xG-con`: 0.8814 (0.0070); `xDef`: 0.9281 (0.0053); `xOff`: 0.8122 (0.0077).

Table 3 shows further statistics after performing a stratified 5-fold cross-validation.

Metric	**Test folds — Mean (SD)**	**Train folds — Mean (SD)**
F_1 (macro-averaged)	43.07% (0.0205)	52.91% (0.0046)
F_1 (micro-averaged)	48.28% (0.0052)	57.38% (0.0035)
Balanced accuracy	43.68% (0.0057)	53.15% (0.0043)
Precision (macro-averaged)	43.32% (0.0058)	54.71% (0.0052)
Precision (micro-averaged)	48.28% (0.0052)	57.38% (0.0035)
Recall (macro-averaged)	43.68% (0.0057)	53.15% (0.0043)
Recall (micro-averaged)	48.28% (0.0052)	57.38% (0.0035)
Jaccard index (macro-averaged)	0.2850 (0.0034)	0.3700 (0.0039)
Jaccard index (micro-averaged)	0.3182 (0.0045)	0.4023 (0.0034)
Hamming loss	0.5172 (0.0052)	0.4262 (0.0035)

Table 3: Final result classifier statistics produced by performing a stratified 5-fold cross-validation.
Note that the low standard deviation indicates the consistency across the 5 folds.

Tables 4 & 5 and Section 6 of the Jupyter Notebook show confusion matrices generated using Scikit-Learn which display the number of times the model made correct and incorrect predictions. The left and top headings show the actual and predicted labels respectively (e.g., Table 4 shows the model correctly predicted the home team would win 1680 times). These are useful to quickly visualise how the model performs overall and when it tends to predict incorrectly.

	H	**A**	**D**
H	1680	423	205
A	481	899	148
D	518	382	353

Table 4: Confusion matrix.

	H	**A**	**D**
H	0.3301238	0.08312046	0.04028296
A	0.09451759	0.17665553	0.02908233
D	0.10178817	0.07506386	0.0693653

Table 5: Normalised confusion matrix.

6 Final Predictions on the Test Set

Table 6 shows the final match outcome predictions, using the latest `xG-won`, `xG-con`, `xDef`, and `xOff` Elo scores for both the home and away teams as input.

Home Team	Away Team	Result
Aston Villa	Man United	A
West Ham	Leeds	H
Norwich	Everton	A
Brighton	Crystal Palace	D
Wolves	Southampton	A
Liverpool	Brentford	H
Tottenham	Arsenal	H
Man City	Chelsea	H
Newcastle	Watford	D
Burnley	Leicester	A

Table 6: Final Predictions on the Test Set.

***D** represents a draw; and **H** and **A** represent the home and away teams winning, respectively.*

The raw output can be found in Section 7 of the Jupyter Notebook.

7 Conclusion

We finalise this report by presenting an analysis of our approach and results, and discuss potential improvements and the scope for further research.

Firstly, the January player transfer window may affect our results due to the inclusion of seasonal player metrics in the models. This could be explored further to investigate how potential misclassification could be mitigated as a result of players changing teams. Additionally, more metrics could be included to train the `xDef` and `xOff` models, if obtainable given the limited data available publicly.

Furthermore, a potential drawback to our current approach is that there may be a limit to how much information can be encoded into numeric Elo scores; therefore, extra non-Elo-rating-based features could be introduced to our final results classifier to improve the quality of its predictions. Given more data, a different, more suitable approach may need to be used building on our current strategy. Moreover, more sophisticated methods for estimating starting Elo values could be investigated and developed, akin to how FiveThirtyEight initialise their SPI ratings [6].

Finally, there is scope to investigate the use of neural networks, including the use of recurrent neural networks (RNNs) in order to better predict the performance of a team as it changes over time and potentially drawing on existing work using Deep Neural Networks [21].

Nevertheless, this report presents a novel and innovative solution to the problem outlined in the assignment brief. Considering the final classifier mean macro-averaged F_1 score in a stratified 5-fold cross-validation of 43.07%, the restricted public access to football data and our limited computational resources, we are satisfied with the predictive performance of our strategy, which is comparable to the 53% accuracy of bookmakers mentioned in the Introduction.

References

[1] Amy N Langville. “Chapter 5 - Elo’s System”. In: *Who’s #1?: The Science of Rating and Ranking*. Ed. by Carl D Myer. Princeton University Press, 2012, pp. 53–66.

[2] Chris Goumas. “Modelling home advantage for individual teams in UEFA Champions League football”. en. In: *Journal of Sport and Health Science* 6.3 (Sept. 2017), pp. 321–326. ISSN: 2095-2546. DOI: 10.1016/j.jshs.2015.12.008. URL: https://www.sciencedirect.com/science/article/pii/S2095254615001325 (visited on 12/13/2021).

[3] *Brentford: How Thomas Frank’s Bees reached the Premier League*. en. URL: https://www.skysports.com/football/news/11689/12033358/brentford-how-thomas-franks-bees-completed-a-remarkable-rise-to-the-premier-league (visited on 12/13/2021).

[4] Egon Franck, Erwin Verbeek, and Stephan Nüesch. “Prediction accuracy of different market structures - bookmakers versus a betting exchange”. In: *International Journal of Forecasting* (Mar. 2010). URL: https://www.sciencedirect.com/science/article/pii/S0169207010000105.

[5] Corentin Herbinet. “Predicting football results using machine learning techniques”. In: *MEng thesis, Imperial College London* (2018).

[6] FiveThirtyEight. *How Our Club Soccer Predictions Work*. Aug. 2018. URL: https://fivethirtyeight.com/methodology/how-our-club-soccer-predictions-work/.

[7] Colin Trainor. *Defensive Metrics - An Introduction*. Oct. 2013. URL: https://statsbomb.com/2013/10/defensive-metrics-an-introduction.

[8] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] *sklearn.svm.LinearSVC: Linear Support Vector Classification*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

[10] *sklearn.svm.LinearSVR: Linear Support Vector Regression*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html.

[11] *sklearn.linear_model.ElasticNet: Linear regression with combined L1 and L2 priors as regularizer*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html.

[12] *sklearn.neighbors.KNeighborsClassifier: Classifier implementing the k-nearest neighbors vote*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[13] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[14] *sklearn.model_selection.RandomizedSearchCV: Randomized search on hyper parameters*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[15] *sklearn.model_selection.GridSearchCV: Exhaustive search over specified parameter values for an estimator*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[16] *skopt.BayesSearchCV: Bayesian optimization over hyper parameters*. URL: https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html.

[17] *XGBoost Scikit-Learn API*. URL: https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.sklearn.

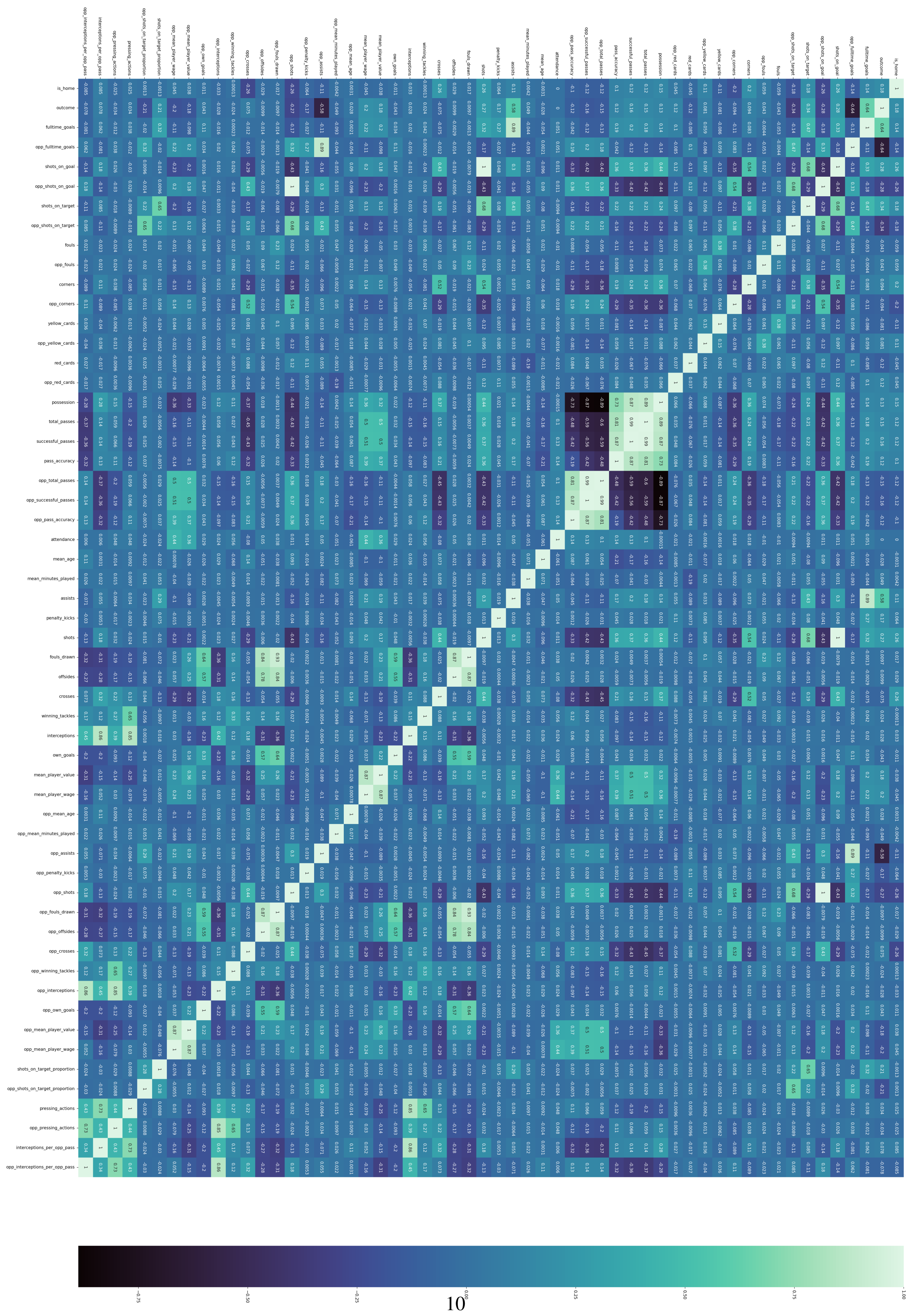
[18] *sklearn.ensemble.RandomForestRegressor: A random forest regressor*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[19] Joos Korstanje. *The F1 score*. Aug. 2021. URL: https://towardsdatascience.com/the-F1-score-bec2bbc38aa6.

[20] Dariush Hosseini. *Machine Learning: Model Selection & Assessment*. 2021. URL: https://moodle.ucl.ac.uk/course/view.php?id=1391.

[21] Dwijen Rudrapal et al. “A Deep Learning Approach to Predict Football Match Result”. In: Jan. 2020, pp. 93–99. ISBN: 978-981-13-8675-6. DOI: 10.1007/978-981-13-8676-3_9.

A Correlation Matrix


is_home
outcome
fulltime_goals
opp_fulltime_goals
shots_on_goal
opp_shots_on_goal
shots_on_target
opp_shots_on_target
fouls
opp_fouls
corners
opp_corners
yellow_cards
opp_yellow_cards
red_cards
opp_red_cards
possession
total_passes
successful_passes
pass_accuracy
opp_total_passes
opp_successful_passes
opp_pass_accuracy
attendance
mean_age
mean_minutes_played
assists
penalty_kicks
shots
fouls_drawn
offsides
crosses
winning_tackles
interceptions
own_goals
mean_player_value
mean_player_wage
opp_mean_age
opp_mean_minutes_played
opp_assists
opp_penalty_kicks
opp_shots
opp_fouls_drawn
opp_offsides
opp_crosses
opp_winning_tackles
opp_interceptions
opp_own_goals
opp_mean_player_value
opp_mean_player_wage
shots_on_target_proportion
opp_shots_on_target_proportion
pressing_actions
opp_pressing_actions
interceptions_per_opp_pass
opp_interceptions_per_opp_pass