Predicting Soccer Players' Market Value using FIFA Data

By Sammy Stern and Yash Kamat

Introduction & Motivation

FIFA is a soccer video game released by Electronic Arts annually. The game uses real-world individual and team performances to assign different attributes (out of 100) to each player licensed in the game. Attributes include offensive, defensive and physical attributes. Each player is also given a "market value" signifying their transfer value, a fee to be paid to their current club to transfer the player, which is based on their in-game attributes and other factors such as age and current team's reputation. We aim to use a player's information and in-game statistics from the latest version of the game, FIFA 22, to build models that predict their market value.

Our motivation to work on this problem comes from being a fan of both, the sport and the game. Data science plays a big role in this sport, as it does in many others, and the acquisition and sale of players is often aided by complex data analysis. For example, Liverpool, one of England's biggest teams, turned their woeful run of trophyless seasons around by smartly recruiting players using data analysis[1].

Data Source

The data for our project was sourced using web scraping. We used the BeautifulSoup package in Python to parse through the first 70 pages of fifaindex.com, a website that contains information about players in FIFA 22. Each page contained the url for about 30 players and the HTML representation for those was recursively collected. Each page was then scraped to retrieve information and attributes about each player, which was stored in a dictionary (one for each player). Due to the size of the data being collected, the response texts for each player and page were cached as a JSON file locally. The list containing "dictionary versions" of each player was then converted to a Pandas dataframe and exported as a CSV file.

This served as our main data source in our analysis notebook. Upon reading the data back into a Pandas dataframe, the data initially had a shape of (3000, 43). There were numerous columns, largely due to each player having an array of attributes for each skill, e.g. passing (Long Pass, Short Pass), dribbling (Ball Control, Reactions), shooting (Finishing, Long Shots) and so on. It was difficult to gauge which of these variables were "important", but it could be argued that the Overall Score (average of all stats) and Potential Score (potential Overall Score at their future peak) would serve as important features in determining a player's market value.

	Player	Overall Score	Potential Score	Market Value	Salary	Height	Weight	Age	Preferred Foot	Ball Control	...	Long Shots	Curve	FK Acc.	Penalties	Volleys
0	Lionel Messi	93	93	92000000.0	380000.0	170	72	34	Left	96.0	...	94.0	93.0	94.0	75.0	88.0
1	Robert Lewandowski	92	92	141500000.0	320000.0	185	81	33	Right	88.0	...	87.0	79.0	85.0	89.0	89.0
2	Kylian Mbappé	91	95	229000000.0	270000.0	182	73	22	Right	91.0	...	82.0	80.0	69.0	79.0	83.0

Figure 1 - Snippet of the overall dataframe

[1] How Data (and Some Breathtaking Soccer) Brought Liverpool to the Cusp of Glory, New York Times, (https://www.nytimes.com/2019/05/22/magazine/soccer-data-liverpool.html)

Methods

Supervised Learning Methods Used

Our problem provided us with a unique opportunity to observe how different ML methods worked in solving it. Along with a Dummy Regressor and a Naive-Bayes Regressor, we used 4 supervised learning methods to conduct our analysis: Linear Regression (linear), Decision Tree (tree-based), Random Forest (tree-based) and KNearestNeighbors (nearest-neighbor).

We also attempted to use Lasso/Ridge Regression and a Support Vector Regressor, but since both proved to be less effective on even a simple train-test split we did not consider them for further analysis (see code for more details).

Feature Representation

Based on the dataset we had, we created two initial representations. The first was called *df_info* and contained data about a player's information, i.e. their salary, age, preferred foot etc. The second was called *df_stat* and contained the data about a player's in-game statistics. The first dataset contained 5 non-target features and included categorical data, while the second contained 38 numerical, non-target features.

Feature Engineering

The numerical features in *df_stat* were individually separated by the type of skill, Agility was distinct from Balance, Shot Power was distinct from Long Passing. However, many of these individual features could be categorized into a broader category, like Passing or Defending. Hence, we created 6 custom features - Pace, Shooting, Passing, Dribbling, Defending and Physical. Each of these features represented an average value of all the sub features in it.

Missing Value Treatment

We did have some missing values in our data, although they didn't make up a significant portion of any individual feature or the dataset as a whole. Most importantly, our target feature column had only 67 missing values out of 3000. These values were Missing Completely At Random (MCAR) as they were due to the incompleteness of data on the source website and the shortcomings of the web scraper in handling these.

As such, rows with missing values in the Market Value column were dropped, as they comprised merely 2.23% of the entire dataset. The remaining values were imputed using a MinMaxScaler since each feature had different minimum and maximum values.

Categorical Feature Encoding

We only had one categorical feature in our dataset - the player's preferred foot. Due to the nature of the source website formatting and the original data from FIFA, we could not get each player's playing position (sometimes players play in multiple positions), another categorical feature that could have been useful. Since the preferred foot for each player was a binary variable (either Left or Right), we used binary encoding, with "Left" as 0 and "Right" as 1.

Feature Importance

Due to our dataframes having a combined 47 non-target columns, the need to distill important features from the rest was crucial to our case. We employed two methods of selecting features - Correlation and Tree-Based feature importance.

Using correlation we were able to get a single coefficient that described the numerical relationship between a feature and our target feature. Combining the inbuilt *DataFrame.corr()* method and a Seaborn heatmap, we were able to plot these values and distinguish the important ones. Below is an example of the heatmap for *df_info*:

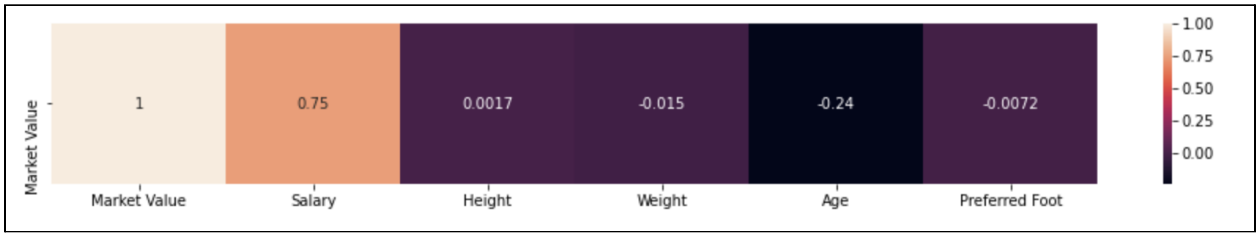

Figure 2 - Heatmap of correlation data for *df_info*

Determining tree based importance involved using an ExtraTreesRegressor and plotting its derived feature importance values.

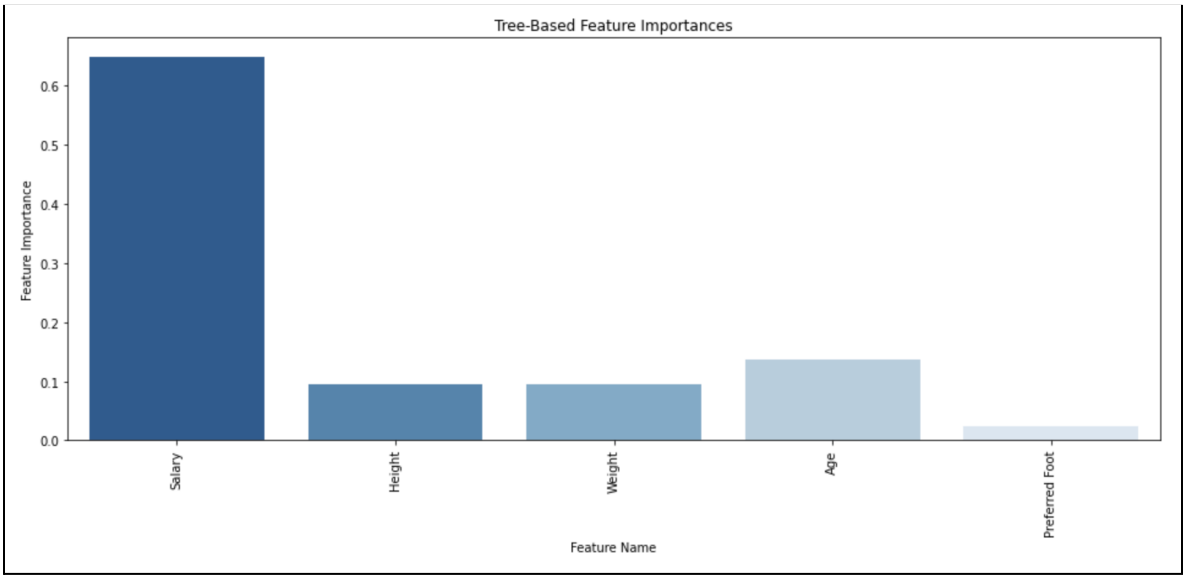

Figure 3 - Bar plot of tree based feature importances for *df_info*

After conducting these experiments, we decided to pick the features that had a correlation value (absolute value) greater than 0.25. This provided us with a challenge for the *df_info* dataframe, as only two features were close to this threshold. At this point, we decided that it wasn't feasible to produce meaningful results with this dataframe, and pivoted to creating a larger dataframe with all data and performing feature ablation.

For *df_stat*, however, we found 6 features that passed this threshold value. These were: Overall Score, Potential Score, Reactions, Composure, Dribbling (overall), Vision and Short Pass. These are all attributes that players, irrespective of playing position, need to have to succeed, so it was interesting yet unsurprising to notice.

Feature Ablation

For feature ablation, we joined *df_info* with *df_stat* to create *df_full*, a dataframe that looked similar to our original dataframe. Upon performing similar correlation and tree-based feature importance analysis, we found 8 variables that met our threshold of 0.25. Additionally, we decided to include “Age” as a feature, as its correlation value was very close to the threshold and it was the only other *df_info* feature close to this threshold.

After conducting our feature importance analysis we selected the following features for model building:

Feature Name	Correlation Value	In df_stat?	In df_full?
Overall Score	0.831	✓	✓
Potential Score	0.769	✓	✓
Salary	0.746	✓	✗
Reactions	0.640	✓	✓
Composure	0.370	✓	✓
Dribbling_Ovr	0.302	✓	✓
Vision	0.294	✓	✓
Short Pass	0.260	✓	✓
Age	-0.242	✓	✗

Table 1 - Final features selected for model building

Train-Test Split, Scaling and Base Model Selection

To prevent data leakage, we *always* conducted a train-test split before scaling values, at all stages in our notebook. We used the StandardScaler from sci-kit learn, fit it on our training data and transformed the test data accordingly. This was done for *df_stat* as well as *df_info*.

Before we delved into in-depth model building, we wanted to see which models performed well on our data at a simple level. Hence, we performed a simple, single train-test split (like in Figure 4 above) and tested a multitude of models on this data. This was done for both *df_stat* and *df_full*. The models were evaluated on their R^2 score.

Model	R^2 score (df_stat)	R^2 score (df_info)	Selected?
Dummy Regressor	0.0	0.0	✓
Gaussian Naive-Bayes	0.797	0.824	✓
Linear Regression	0.766	0.830	✓
Lasso Regression	0.766	0.830	✗
Ridge Regression	0.767	0.831	✗
Decision Tree	0.801	0.944	✓
Random Forest	0.921	0.982	✓
SVR	-0.166	-0.166	✗
KNN Regressor	0.913	0.958	✓

Table 2 - Initial R^2 scores for baseline models

Based on these metrics, we deemed that SVR did not perform well for our data. Moreover, Lasso/Ridge did not provide a much greater result compared to the regular Linear Regression model. Hence, those 3 models were excluded from the model building and evaluation process.

Evaluation

Evaluation Metrics

Since ours was a regression problem, we decided to use R^2 score, RMSE and MAE as our three evaluation metrics. R^2 was our main evaluation metric, as it provided a great way to gauge how well each model could explain the relationships between our independent and dependent features. Moreover, due to our target feature being a monetary figure in millions, RMSE and MAE would yield large values and thus, R^2 provided a great way of comparing different models on a similar and smaller scale.

Main Results

We tested the performance of the above selected models using a 5-fold cross-validation. For each fold, we split the data into a 75:25 train-test split and scaled the features accordingly.

Model	Mean R^2 score (df_stat)	R^2 score Std Dev (df_stat)	Mean R^2 score (df_info)	R^2 score Std Dev (df_info)
Dummy Regressor	0.0	0.0	0.0	0.0
Gaussian Naive-Bayes	0.457	0.076	0.527	0.089
Linear Regression	0.756	0.021	0.807	0.011

Decision Tree	0.746	0.039	0.823	0.058
Random Forest	0.920	0.021	0.950	0.021
KNN Regressor	0.783	0.033	0.773	0.030

Table 3 - Initial R^2 scores for baseline models on a 5-fold CV

The above models were still basic, i.e. without any hyper-parameter tuning. After getting the above values, we performed hyper-parameter tuning on our non-"dummy" models and replaced Linear Regression with Polynomial Regression.

Model	Tuned Hyperpameter	Mean R^2 score (df_stat)	R^2 score Std Dev (df_stat)	R^2 score (df_info)	R^2 score Std Dev (df_info)
Dummy Regressor	N/A	0.0	0.0	0.0	0.0
Gaussian Naive-Bayes	N/A	0.457	0.076	0.527	0.089
Polynomial Regression	Polynomial Degree = 3	0.817	0.027	0.957	0.01
Decision Tree	min_samples_split = 7	0.844	0.026	0.916	0.016
Random Forest	max_depth = 8, n_estimators = 15	0.917	0.025	0.942	0.027
KNN Regressor	n_neighbors = 6	0.779	0.027	0.776	0.023

Table 4 - R^2 scores for tuned models on a 5-fold CV

Based on the results above, we concluded that Polynomial Regression and Random Forest were the modes with the best overall results. The former had the highest average R^2 value while also having the lowest standard deviation for R^2. The Random Forest model also performed well in both metrics for both data representations.

On the other hand, the KNN Regressor seemed to perform worse, even with hyper-parameter tuning. It had the lowest R^2 values out of all non-dummy models for both representations. In fact, its performance got worse for the *df_info* representation, raising doubts about the model's scalability. Hence, for further analysis done below, we only considered the Polynomial Regression, Decision Tree and Random Forest models.

In terms of feature ablation, we noticed that except the KNN Regressor, the models performed better with the increased number of features, specifically Salary and Age. For both of these features, the nearest neighbors might not be highly indicative of the Market Value. For example, a player could be highly paid either due to their agent bargaining for a big contract, their loyalty to a club or their international brand/reputation - none of which are factors related to their actual performance or willingness of other clubs to buy them (aka Market Value).

Tradeoffs

For tradeoff analysis we did a simple bias-variance tradeoff analysis. Using the *mlxtend* library's *bias_variance_decomposition* method, we analysed what percentage of each model's RMSE value was bias and variance.

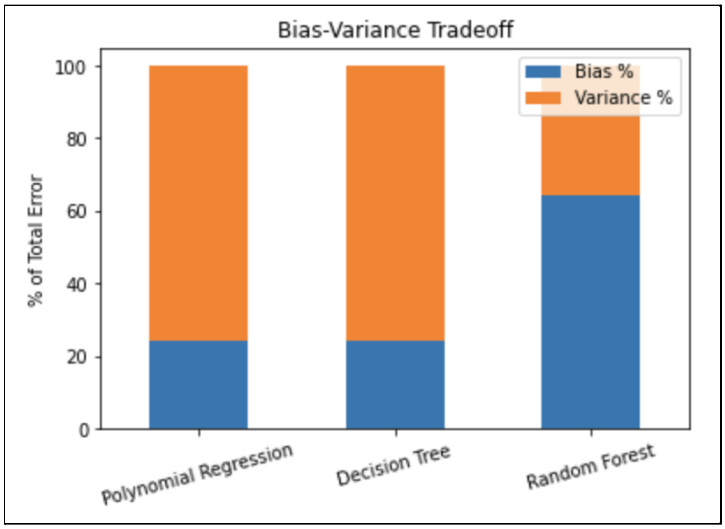

Figure 4 - Bias-Variance Tradeoff for tuned models

Here we found that the Polynomial Regression and Decision Tree models had over 70% of their error as a variance error. This meant that these models may be susceptible to overfitting. This can be explained by the fact that these models were highly tuned to a singular hyper-parameter

Random Forest, however, had a greater bias error than variance error. This does not mean that the model is underfitted, rather that it's variance error is so low that the bias error is comparatively higher. The nature of the Random Forest model and the inclusion of multiple trees means that the model is more scalable and thus, has a lower variance error.

Sensitivity Analysis

For our three models we tuned one key hyper-parameter for sensitivity analysis. They were - Polynomial Degree for Polynomial Regression, *min_samples_split* for Decision Tree and *n_estimators* for Random Forest. Unfortunately, while trying to conduct this analysis for Polynomial Regression, the kernel crashed multiple times, and we had to omit it from our final analysis.

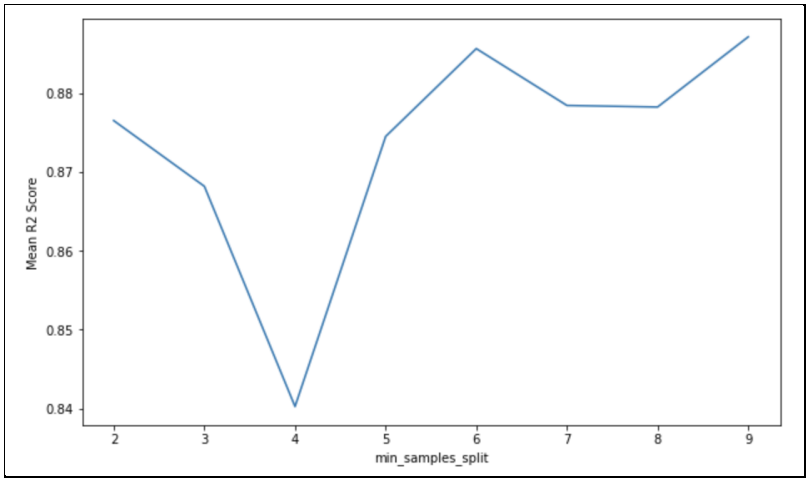

Figure 5 - Varying R^2 values for Decision Tree model

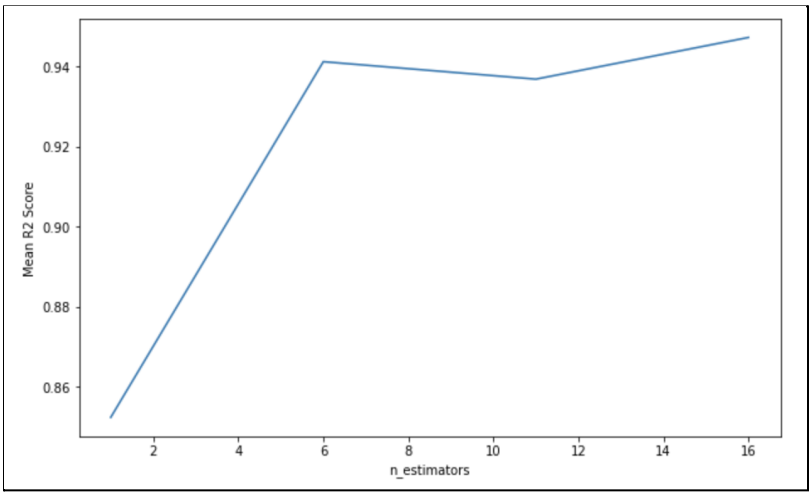

Figure 6 - Varying R^2 values for Random Forest model

For both models, we noted that the R^2 values didn't fluctuate heavily for changing hyper-parameters. In both cases, the values stayed roughly within 10% of our previous values. The ability of these models to perform fairly well with changing values indicated that the models have potential for good scalability.

Feature Importance

For this section, we decided to plot the feature importances of our Random Forest model. We found that the Overall and Potential scores, along with Age were the most important features.

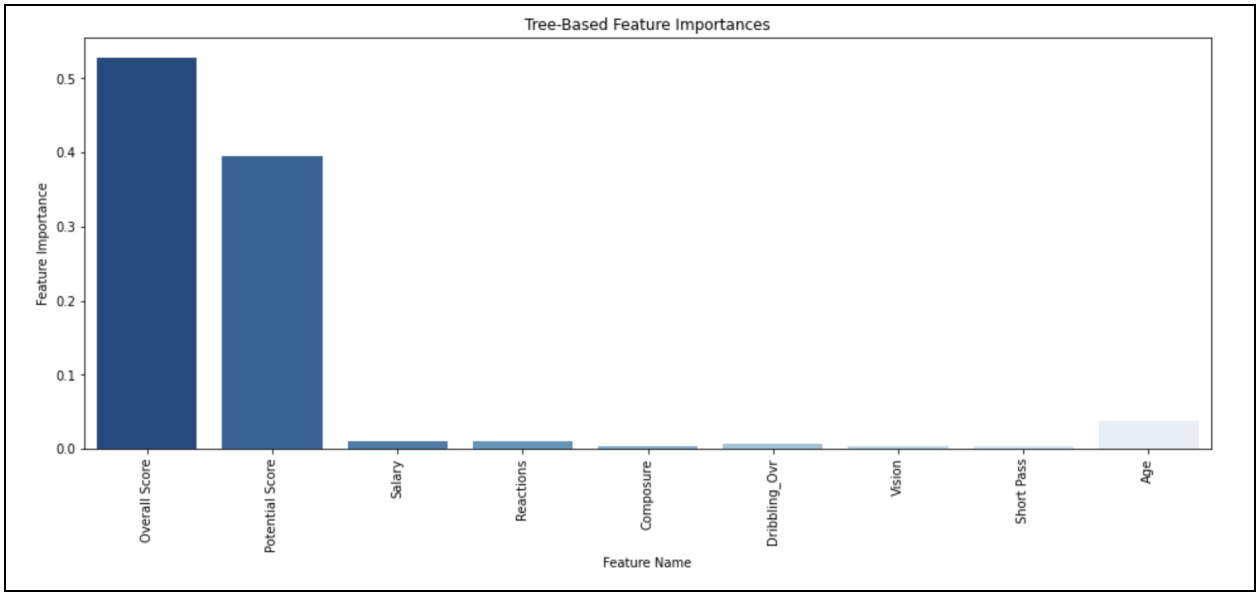

Figure 7 - Feature Importances of tuned Random Forest model

Failure Analysis

For this section, we looked at 5 players for whom our predicted values were furthest from their true values. Upon further observation we came across some explanations as to why that might be the case. Firstly, all 5 players were players who played for recent championship winning teams in the highest divisions of England and Spain - two of the biggest countries in terms of competitive club soccer. Their team's recent success could have led to FIFA giving them a higher (perhaps inflated) market value, something our model didn't account for since we didn't have individual team/league data in the dataset.

	True Value	Predicted Value	Difference	Player
15	101000000.0	1.264467e+08	2.544667e+07	Thibaut Courtois
22	53000000.0	7.686000e+07	2.386000e+07	Luis Suárez
27	111000000.0	8.922367e+07	2.177633e+07	Raheem Sterling
4	148000000.0	1.272000e+08	2.080000e+07	Kevin De Bruyne
9	96500000.0	1.152341e+08	1.873410e+07	Alisson

Figure 7 - Feature Importances of tuned Random Forest model

Secondly, two of these values belonged to goalkeepers. While the goalkeeping (GK) attributes in our dataset had values for each player, those values were especially applicable to actual goalkeepers, whose performance is determined largely by those attributes rather than by traditional outfield players' attributes like Dribbling or Shooting. Since the majority of the dataset consisted of outfield players who had low GK attributes, top goalies with GK attributes were likely valued and weight especially highly by our models, thus, leading to the exceedingly high predictions.

Discussion

1. *What interesting relationships or insights did you get from your analysis?*

Two insights particularly intrigued us. The first was that general physical statistics were valued higher by our models as compared to pure offensive or defensive attributes. This is interesting because offensive players are usually valued higher in the game than defensive players. Hence, it was expected that attacking attributes, like Shooting or Dribbling, may influence the models greatly. However, it was interesting, although unsurprising, to notice that greater importance was given to more general attributes such as Reactions and Composure, since these attributes are agnostic to a specific field position.

The second was that a player's salary was not highly indicative of their market value. This was intriguing because had postulated that a player valued highly in the market is likely also valued highly by their current club and thus, rewarded with a contract of a similar level. But upon further observation, we noticed that many highly paid players were those towards the end of their careers - rewarded with large contracts due to their existing reputation and/or loyalty. Meanwhile, many young players with high potential ceilings had lower wages, perhaps awaiting a new contract with their existing or new club that matches their expected development.

2. *What didn't work, and why?*

In terms of data collection, we encountered scraping issues that led to the web scraper code needing to impute NaN values wherever certain data couldn't be found. This led to data loss and

In terms of models, the KNN Regressor didn't perform as well as expected. This could be due to a number of reasons. While statistically, neighbors could indicate the target variable well due them being numerical and scaled similarly, some complexities of a player's market value couldn't be captured by KNN's simple algorithm - things like age. While age is linear in nature, a player's value may vary during different stages of their career - younger players have potential, older ones may have an established reputation, and so on. Hence, the KNN couldn't perform and scale on the given data well.

3. *What surprised you about your results?*

We were surprised to see that the Polynomial Regression model did so well. In terms of the chosen evaluation metrics, it performed the best across the entire dataset, having the highest mean R^2 score and the lowest variance within the CV results.

This was surprising because we expected the tree-based models to perform far better than regression-based models, due to the broad nature of our data. We hypothesized that flexibility of tree-based models would give them the edge over other models, but realize now that the majority of the data being numerical and scaled well suited regression models (specifically Polynomial) really well - leading to good results.

4. *How could you extend your solution with more time/resources?*

With more time, we could do a deeper diver using more complex models like neural networks (MLPRegressor). This could give us more options while selecting a final, best performing model, and more importantly, give us more ways of understanding how different features contribute and interact with our target variable.

We would also devote more time to improving our web-scaper. While ours worked well, we would have attempted to find and add features that we couldn't, such as a player's position. We would also try scraping different source websites to see if they yielded better results.

With more resources, we would perform more fine-grained GridSearch cross-validation for hyper-parameter tuning, which would hopefully lead to better performing models. It would also help with sensitivity analysis for our Polynomial Regression model, something that was curtailed due to memory issues.

5. *What ethical issues could arise in the course of this project work, or applying its results?*

There are two main ethical issues related to our project. Firstly, the source of the data may contain biases that we are unaware of. The market values in our dataset have been decided by EA Sports, the creators of the FIFA franchise, and the methodology behind determining those values isn't publicly available - so there might be biases in their methodology that have influenced our models.

Secondly, our models are, at the end of the day, based on numbers and statistics. They do not account for a variety of abstract factors, such as a player's play style, fitness levels and ability to fit into a specific team. If soccer teams were to use models such as ours, they need to account for these things as it may affect the amount they feel like paying for a certain player. An expensive player may not fit into a team and perform as well as one who's cheaper but with a play style suited to a team's tactics.

Statement of Work

Section	Sammy's Contribution	Yash's Contribution
Introduction & Motivation	75%	25%
Data Retrieval & Web Scraping	0%	100%
Data Cleaning	20%	80%
Feature Importance	60%	40%
Train-Test Split, Scaling, Base Model Selection	100%	0%
Main Results	20%	80%
Tradeoffs	40%	60%
Sensitivity Analysis	70%	30%
Failure Analysis	80%	20%
Discussion	50%	50%