UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Machine Learning (APA)

Laboratory Project: SkillCraft

Group 12
David Nogales Pérez
Álex Ochoa Blasco

Barcelona January 8, 2022

Contents

1 Introduction

In this project, given some in-game telemetry data of the famous Real Time Strategy (**RTS**) game "**Starcraft 2**", we will study the feasibility of predicting the level of expertise of a player from this data by training some Machine Learning models and evaluating their performance.

In the following chapters we firstly explore the data while pre-processing it, Afterwards we will re-sample the data for our models, and to finish with, we will compare the results of the models and chose the one that we consider to be the best. In order to do this, we are tasked to train 5 different models which 2 of them are non-linear models and the rest Lineal or Quadratic models and, additionally, find the best hyperparameters for each one them.

We will use a dataset [4] that comes from the UCI data set repository. It contains the data of 3395 games of the "**Starcraft 2**" from players with different levels of mastery. Most of the variables describe in-game metrics of players, but we also have external variables, such as age and hours spent in the game. The target variable, which has 8 categories that represent the level of expertise of the player, is League Index. Each category represents a different skill of the player gradually incrementing, being the lower numbers for amateur players and the higher ones for players in professional leagues.

1.1 Related previous work and results

The "**Skillcraft**" dataset has been generated for the study described in the paper by Thompson JJ, Blair MR, Chen L, Henrey AJ, [6]. In said paper, the study sets as its goal to identify meaningful variables which could explain the expertise of the different players.

The authors remark, as their primary finding that variables do change in importance across the skill levels and give results to support their initial hypothesis. They also concluded that RTS game replays can track abilities of interest to cognitive science.

2 Data Exploration

In this chapter we dive into exploring the dataset characteristics and also providing soome solutions to problems found troughout the analysis.

2.1 Dataset Variables Description

Before starting the exploration we provide a brief summary of each variable's description present in the dataset in order to better understand the nature of the data:

- **GameID**: Unique ID number for each game (integer)
- **LeagueIndex**: Bronze, Silver, Gold, Platinum, Diamond, Master, GrandMaster, and Professional leagues coded 1-8 (Ordinal)
- **Age**: Age of each player (integer)
- **HoursPerWeek**: Reported hours spent playing per week (integer)
- **TotalHours**: Reported total hours spent playing (integer)
- **APM**: Action per minute (continuous)
- **SelectByHotkeys**: Number of unit or building selections made using hotkeys per timestamp (continuous)
- **AssignToHotkeys**: Number of units or buildings assigned to hotkeys per timestamp (continuous)
- **UniqueHotkeys**: Number of unique hotkeys used per timestamp (continuous)
- **MinimapAttacks**: Number of attack actions on minimap per timestamp (continuous)
- **MinimapRightClicks**: number of right-clicks on minimap per timestamp (continuous)
- **NumberOfPACs**: Number of Perception Action Cycles per timestamp (continuous)
- **GapBetweenPACs**: Mean duration in milliseconds between Perception Action Cycles (continuous)
- **ActionLatency**: Mean latency from the onset of a Perception Action Cycles to their first action in milliseconds (continuous)
- **ActionsInPAC**: Mean number of actions within each PAC (continuous)
- **TotalMapExplored**: The number of 24x24 game coordinate grids viewed by the player per timestamp (continuous)
- **WorkersMade**: Number of SCVs, drones, and probes trained per timestamp (continuous)
- **UniqueUnitsMade**: Unique unites made per timestamp (continuous)
- **ComplexUnitsMade**: Number of ghosts, infestors, and high templars trained per timestamp (continuous)
- **ComplexAbilitiesUsed**: Abilities requiring specific targeting instructions used per timestamp (continuous)

2.2 Balancing Data and Dealing with Null values

Observing the histogram plot 2.1 of the target variable , one can see that the data set is heavily unbalanced mainly due to the extreme categories. Given that we can't obtain more data we should study if we can afford to drop these categories, or try another methods such as merging them given that some categories in the **Starcraft** ladder aren't too different in terms of skill.

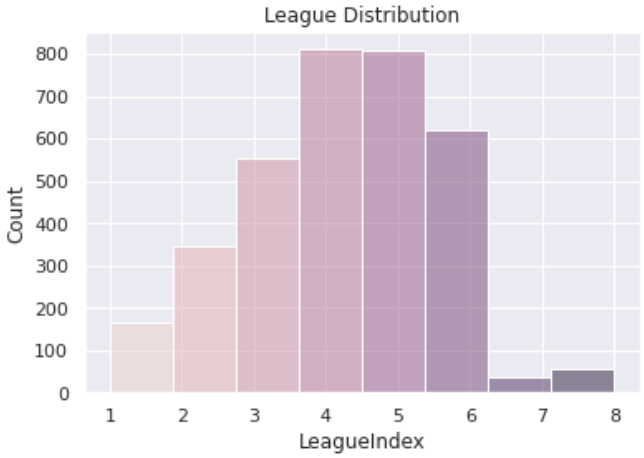

Fig. 2.1: Histogram Plot of LeagueIndex

Looking at the information of the dataframe, there are some variables which are numeric but appear as object and also there seemingly aren't any null values. After converting the variables to numeric we found that there are indeed null values in the data set. We can see that most of our missing data is located in the *'Professional'* category (8) and only 2 rows in the 5th category have missing data. A solution to this would be imputing all the missing data in the corresponding categories instead of dropping them since we could be dropping valuable data. But given that our data sample of these categories are to low there is no way to know if the values we would be imputing will be correct, so we will drop them.

To solve the problem regarding the unbalanced data we will merge categories 1,2 and 6,7 as we can see in the plot 2.2, given that according to the global statistics of ranks of the population of active players in the game through different seasons [1], these categories have a smaller number of players in general so by merging them we will be able to better predict the remaining ranks.

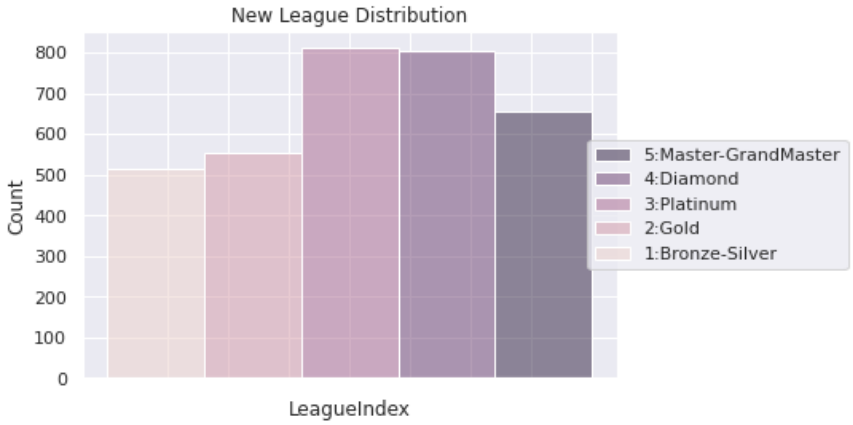

Fig. 2.2: Plot of New League Distribution

2.3 Detecting Outliers

Looking at the description of the variables one can rapidly tell the presence of outliers. For example in the variable *TotalHours* we have that someone played the game a million hours which is roughly 114 years. Similarly in *HoursPerWeek* the maximum of hours played in a week is 168 hours which would mean a week of playing the game without rest. Analyzing the violin plots of the data set 2.3, we can clearly observe the presence of outliers in different variables.

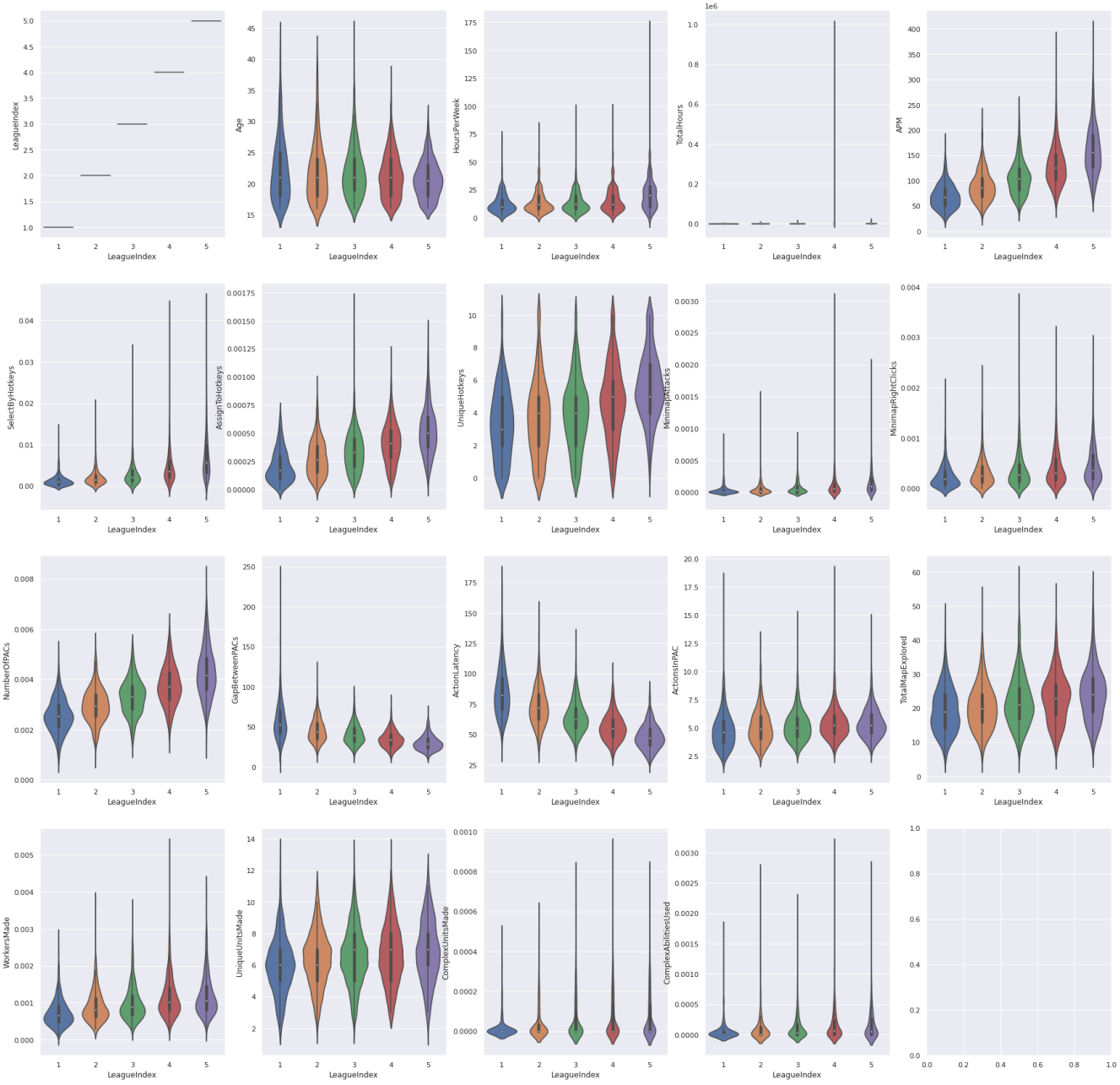

Fig. 2.3: Violin Plots of the dataset

Since most of the extreme values appear to be in the upper points of the violin plots we will erase values greater to the 99 percentile. Although outliers are still present in our data we can't erase or modify them since those data points may pertain to players who are close to climb the classification ladder.

2.4 Discretizing Age

Since our data does not have a discrete variable we are tasked to create one. To do that we chose *Age* which will be automatically discretized by the **KBinsDiscretizer**, which will create 3 new categories of same size.

2.5 Feature Selection

We will perform **Feature Selection** instead of **Feature Extraction** avoiding the change of variables' space performed by applying **PCA**. Observing the **Correlation Heatmap** 2.4 one can tell that the target variable has a relatively high correlation with *APM* and *NumberOfPACs*, in contrast, we have that is inversely correlated with *ActionLatency* and *GapBetweenPACs*

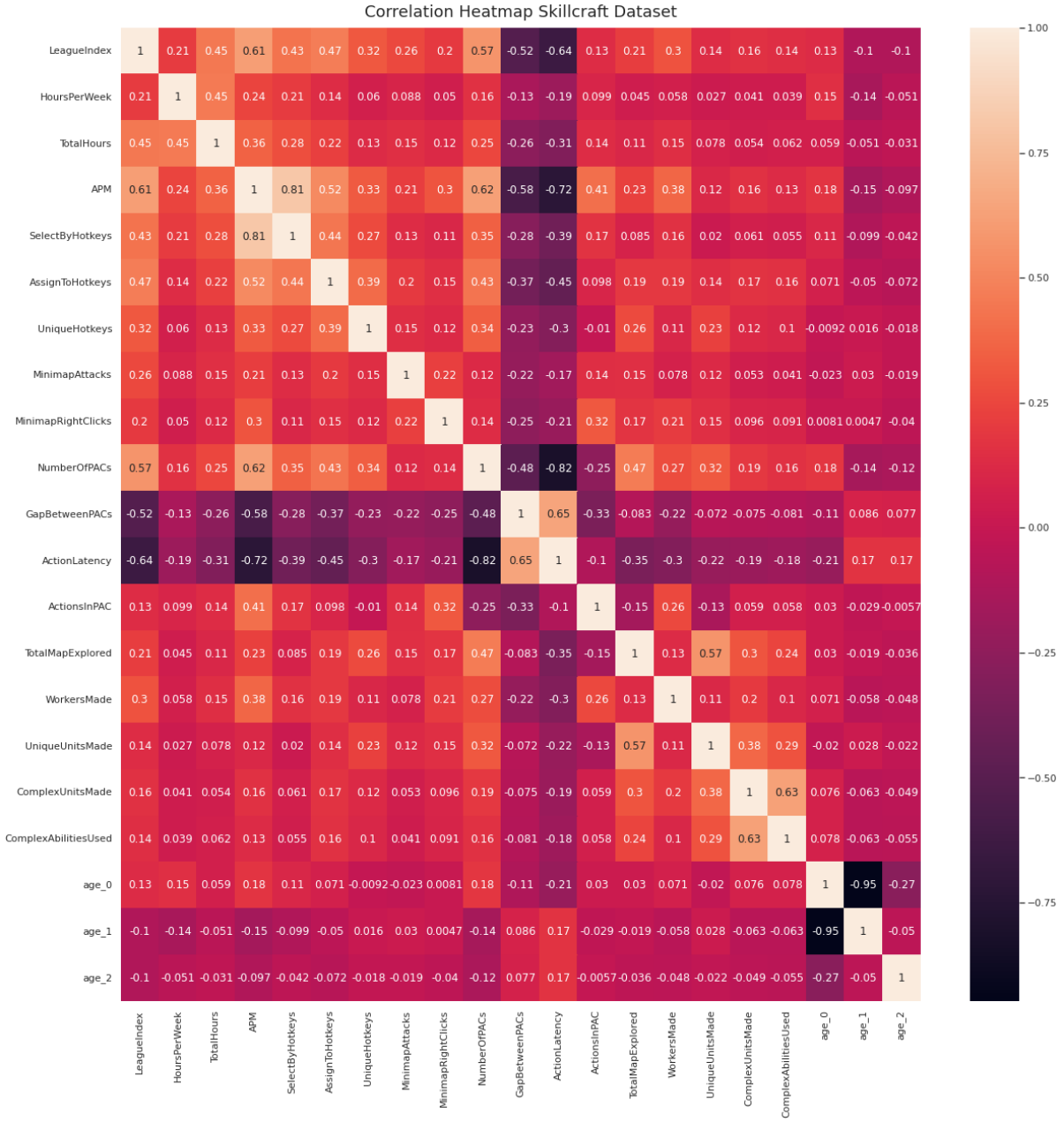

Fig. 2.4: Heatmap of the data set

Looking at the absolute value of the correlations 2.5 we have that at least 9 features have a relatively big impact on the target variable, meanwhile the rest does not seem to be that important.

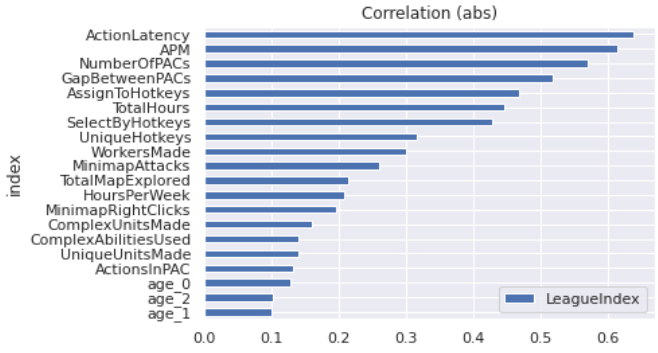

Fig. 2.5: Absolute value of the correlations

2.5.1 Feature Importance

According to the **Random Forest Regressor** 2.6 we can explain approximately 60% of the variance using only *APM*, *ActionLatency* and *TotalHours* variables.

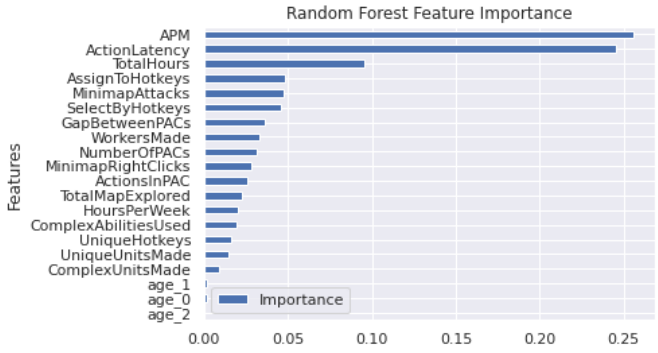

Fig. 2.6: Random forest features importance

Looking at the cumulative sum of the features importance's 2.7 we can see that we can explain 90% of the variance using 12 features instead of 20.

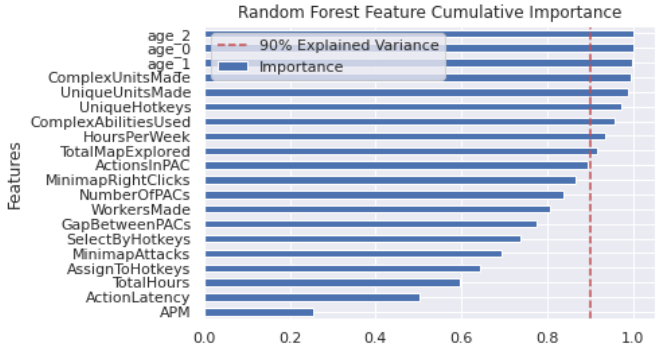

Fig. 2.7: Random forest features Cumulative importance

By intersecting the 10 most important features found with the correlation matrix and features importance's of the **Random Forest**, we end up with 9 features instead of 20, which is a great reduction of the variables.

3 Training Methodology (Resampling protocol)

Our target variable is ordinal so the models chosen are regression models in order to avoid losing the inherent relationship between the ranks of the players and also lose any prediction power that could provide this relationship.

First we split the training and test data into 70% for training and 30% for testing so it ensures the test set representativeness and keep the computational cost of training the models relatively low.

Next, to validate the obtained results **K-Fold Cross-Validation** is used with K-Folds of 5 in order to keep the time of validation at a conservative amount. Subsequently, in order to fine tune each model we picked **RandomizedSearchCV** to find the best hyperparameters to train our models given that the parameters space search was large enough to avoid the searching exhaustively provided by **GridSearchCV**.

Then, for each of the trained models, we generate their training and prediction **MAE** metrics and the **R2** score for the predictions. Finally we plot the predicted results with the real values in order to visually see the differences.

The models that we are tasked to train are the following:

Linear/Quadratic

- Linear Regression
- Linear SVM
- Quadratic SVM

Non-linear

- SVM with RBF kernel
- Random Forest

4 Results Obtained

4.1 Linear Regression

First method we are using is the **Linear Regression** 4.1

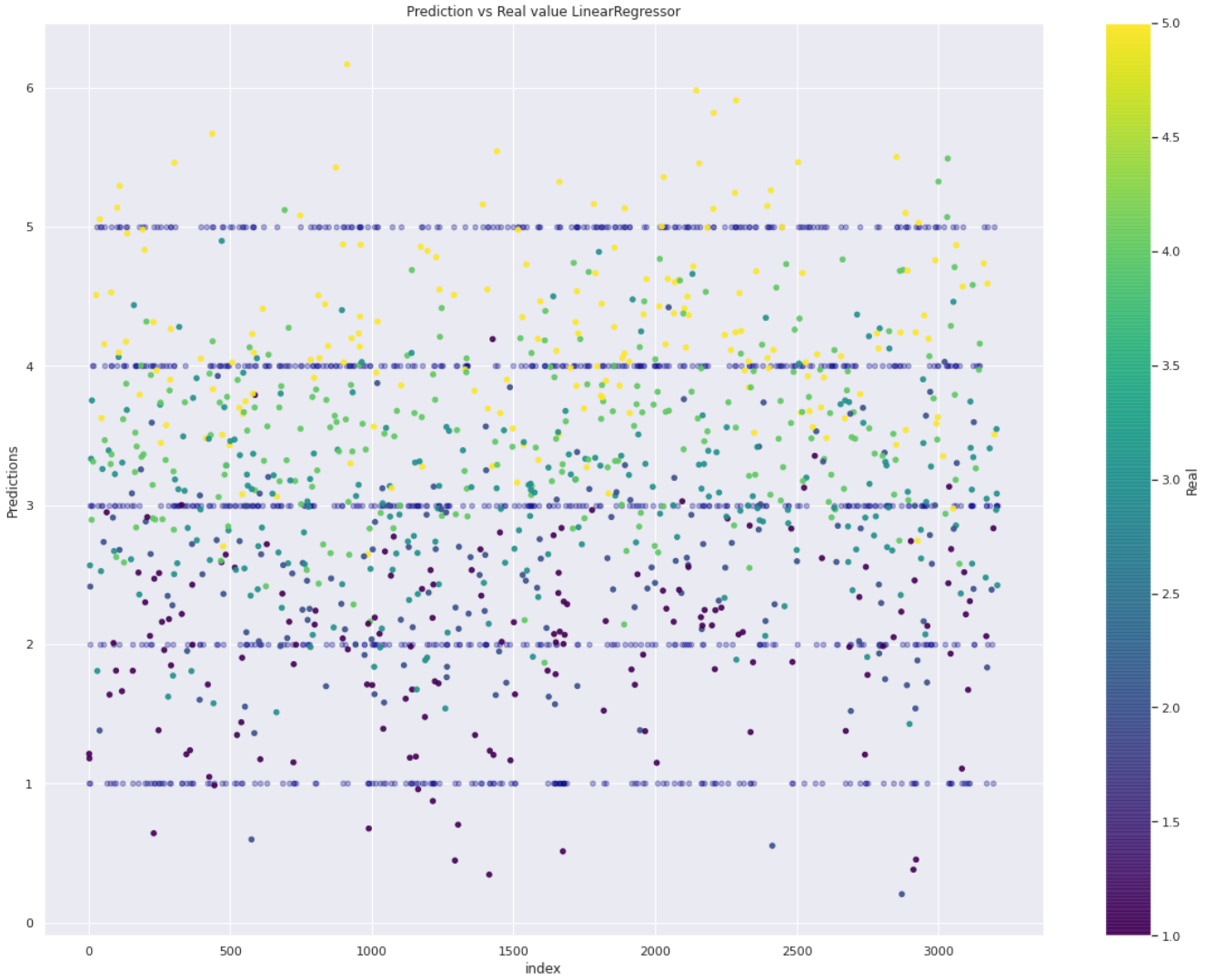

Fig. 4.1: Prediction vs Real value Linear Regression

Using this model we achieved a R2 test score of 0.56, and a MAE test score of 0.73.

4.2 Linear SVM

Second method we are using is the **Linear SVM** 4.2. The best hyperparameters we found for this model are the following:

```
{'random_state': [42], 'max_iter': [100000], 'epsilon': [0.8], 'C': [25]}
```

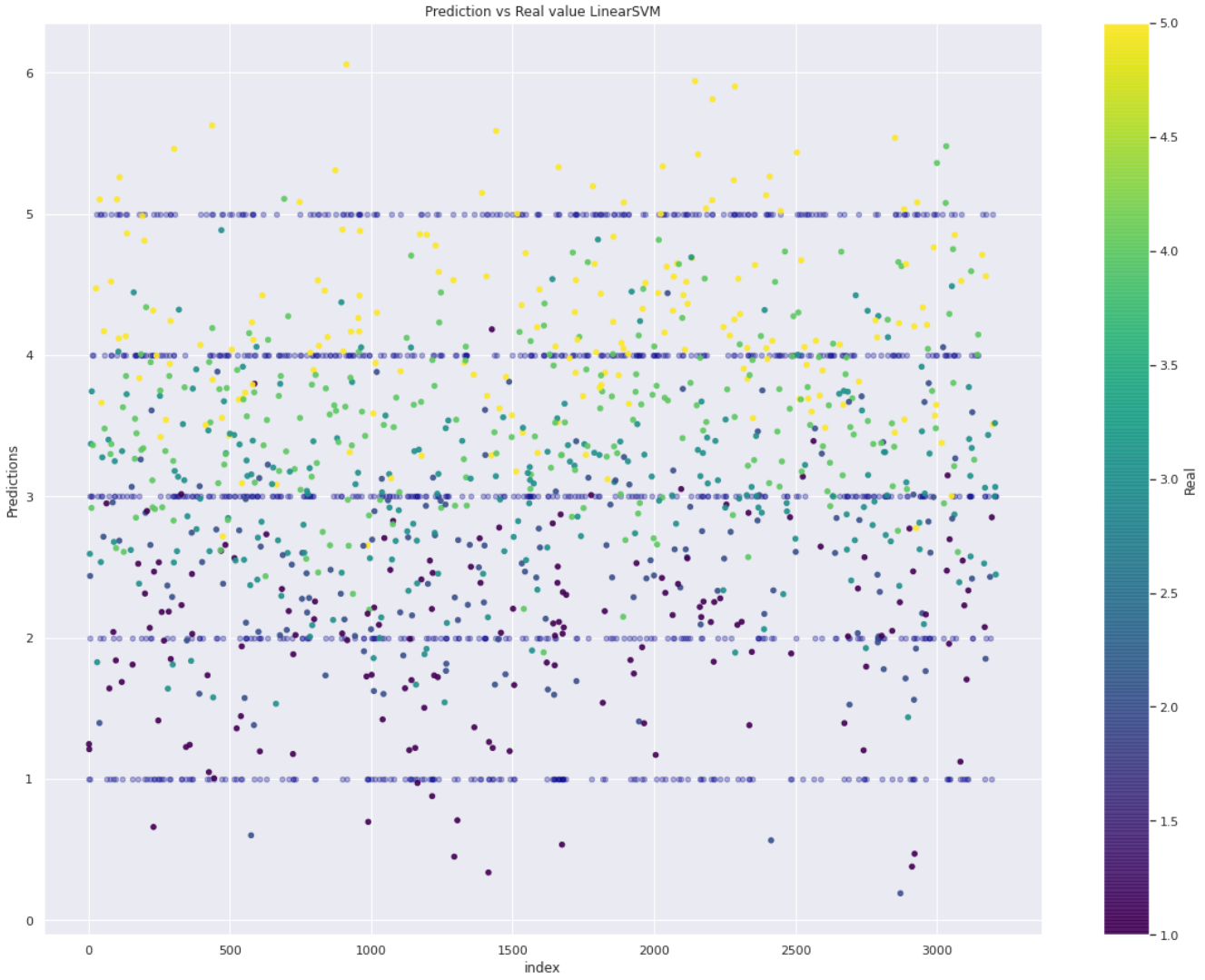

Fig. 4.2: Prediction vs Real value Linear SVM

Using the **Linear SVM** model with best hyperparameters, we achieved a R2 test score of 0.56, and a MAE test score of 0.73.

4.3 Quadratic SVM

Third method we are using is the **Quadratic SVM** 4.3. The best hyperparameters we found for this model are the following:

```
{'kernel': ['poly'], 'epsilon': [1e-05], 'degree': [2], 'coef0': [50], 'C': [1]}
```

Fig. 4.3: Prediction vs Real value Quadratic SVM

Using the **Quadratic SVM** model with best hyperparameters, we achieved a R2 test score of 0.58, and a MAE test score of 0.69.

4.4 RBF SVM

First of the non-linear methods we are using is the **SVM with RBF kernel** 4.4. The best hyperparameters we found for this model are the following:

```
{'kernel': ['rbf'], 'gamma': [0.001], 'epsilon': [0.0001], 'C': [1000]}
```

Fig. 4.4: Prediction vs Real value SVM with RBF kernel

Using the **SVM with RBF kernel** model with best hyperparameters, we achieved a R2 test score of 0.59, and a MAE test score of 0.68.

4.5 Random Forest

Last method we are using is the **Random Forest** 4.5. The best hyperparameters we found for this model are the following:

```
{'random_state': [44], 'n_estimators': [225], 'max_features': ['log2'], 'max_depth': [23]}
```

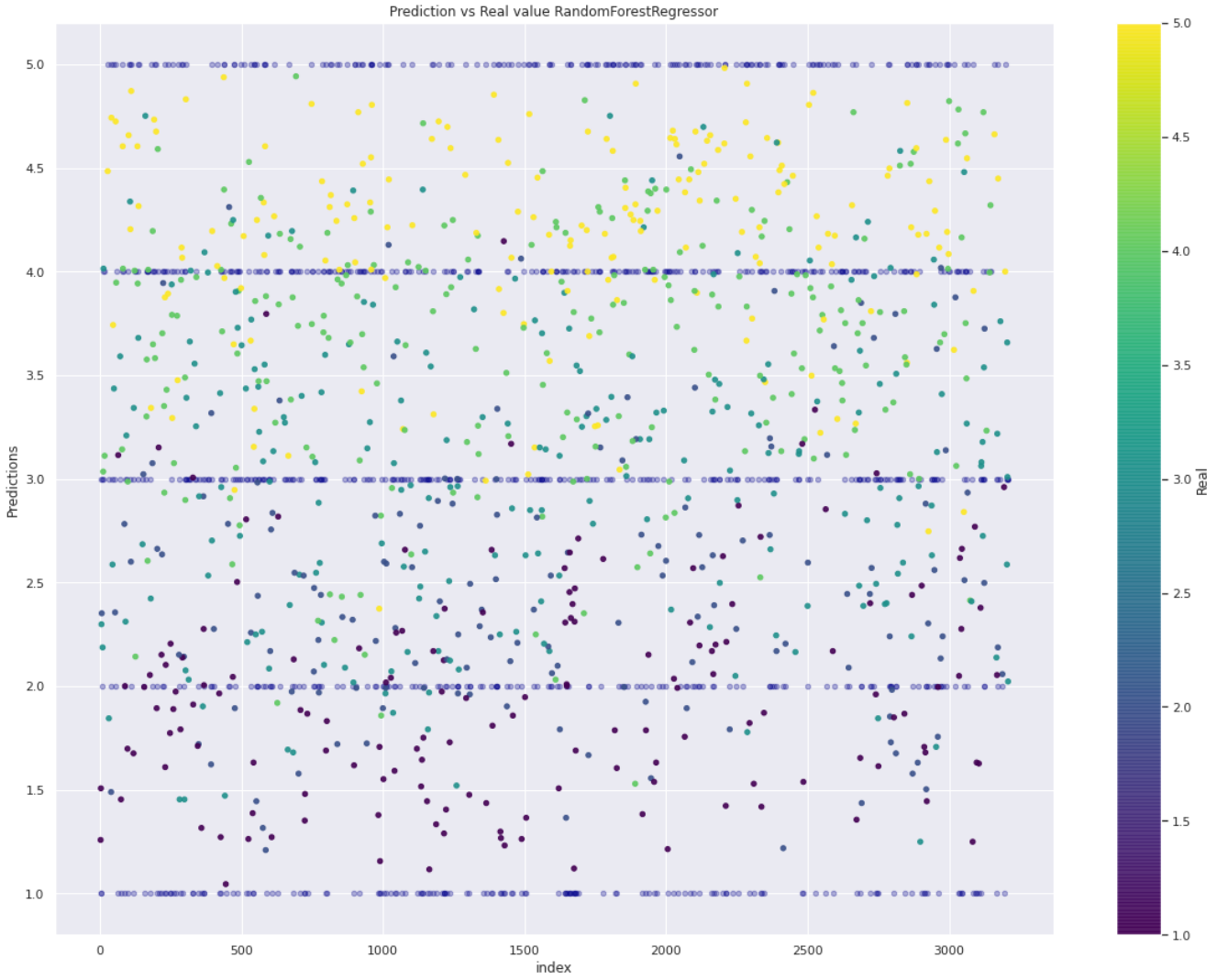

Fig. 4.5: Prediction vs Real value Random Forest

Using the **Random Forest** model with best hyperparameters, we achieved a R2 test score of 0.58, and a MAE test score of 0.70

5 Model Chosen

Analyzing the final results, we can see that all 5 models act pretty similar in terms of score, despite that, the **SVM with RBF kernel** produces the best score with the least absolute mean error. Although it offers the best results, by a low margin compared to the others, it also has the greatest prediction time which wouldn't be desirable in real-time applications given that one can get similar results with simpler models. As a final model, and following the **Ockham's Razor** principle, we are inclined to pick the simplest model which is the **LinearRegression** as it has an acceptable prediction score and is the fastest model regarding prediction time.

	Test score R2	Test score MAE	Train score MAE	Prediction Time
RbfSVM	0.587767	0.682977	-0.692864	1.352230
QuadraticSVM	0.582313	0.689182	-0.698299	0.567482
RandomForestRegressor	0.579276	0.702062	-0.709131	0.545400
LinearRegressor	0.559078	0.725275	-0.724747	0.001171
LinearSVM	0.558238	0.725961	-0.724471	0.002330

6 Self-Assessment

Looking back at the overall performance obtained with the models, the poor performance could be explained simply because the motor abilities of the players can't fully predict their corresponding rank in the league. But before arriving to such conclusions further studies could be performed, such as:

- Instead of performing **Feature Selection** one could analyze the performance doing **Feature Extraction**.
- Choosing different models like K-Nearest **Neighbours** or a **Neural Network**.
- Using the models provided by the libraries **mord**[7] or **statsmodels**[8] which allow **Ordinal Regression** in Python.
- Instead of training models with all the categories at once, training the models by pairing ranks may increase the separability between categories (As seen in the original paper of the data set [4]) and by doing so arrive to our own conclusions.

All in all, this project provided an enlightening experience by looking at extracted data from such a popular game as **Starcraft 2** and see the different characteristics of the game that could be analyzed and experience first-hand the problems that one can encounter with data in the 'wild'.

7 References

[1] 1v1 league distribution. https://www.rankedftw.com/stats/leagues/1v1/#v=2&r=-2&sx=a. Data actualized daily.

[2] Aprenentatge automàtic (APA) FIB web. https://sites.google.com/upc.edu/aprenentatge-automatic.

[3] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[4] Mark Blair, Joe Thompson, Andrew Henrey, and Bill Chen. Skillcraft1 master table dataset data set. http://archive.ics.uci.edu/ml/datasets/SkillCraft1+Master+Table+Dataset, 2013.

[5] Jason Brownlee. How to remove outliers for machine learning. https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data, 2018. Last Updated on August 18, 2020.

[6] Thompson JJ, Blair MR, Chen L, and Henrey AJ. Video game telemetry as a critical tool in the study of complex skill learning. *PLoS ONE*, 2013.

[7] Fabian Pedregosa-Izquierdo. *Feature extraction and supervised learning on fMRI : from practice to theory*. Theses, Université Pierre et Marie Curie - Paris VI, February 2015.

[8] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.