

SKILLCRAFT DATASET ANALYSIS

By Valentin DUPUIS



Introduction

This dataset gathers data on the starcraft 2 video game. It is a real-time strategy game. Players can play against players or against the AI. Entries are various data that could be retrieved or calculated from a game. One line corresponds to a summary of a player's game. The output corresponds to the level of the player concerned. There are 8 different player level. The objective of this analysis is to understand and be able to predict the level of a player based on the most interesting entries.



DataSet Presentation

Attribute Information:

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

| | | | | | |
|----------------------------|---------------|-----------------------|------|---------------------|------------|
| Data Set Characteristics: | Multivariate | Number of Instances: | 3395 | Area: | Game |
| Attribute Characteristics: | Integer, Real | Number of Attributes: | 20 | Date Donated | 2013-10-22 |
| Associated Tasks: | Regression | Missing Values? | Yes | Number of Web Hits: | 76190 |

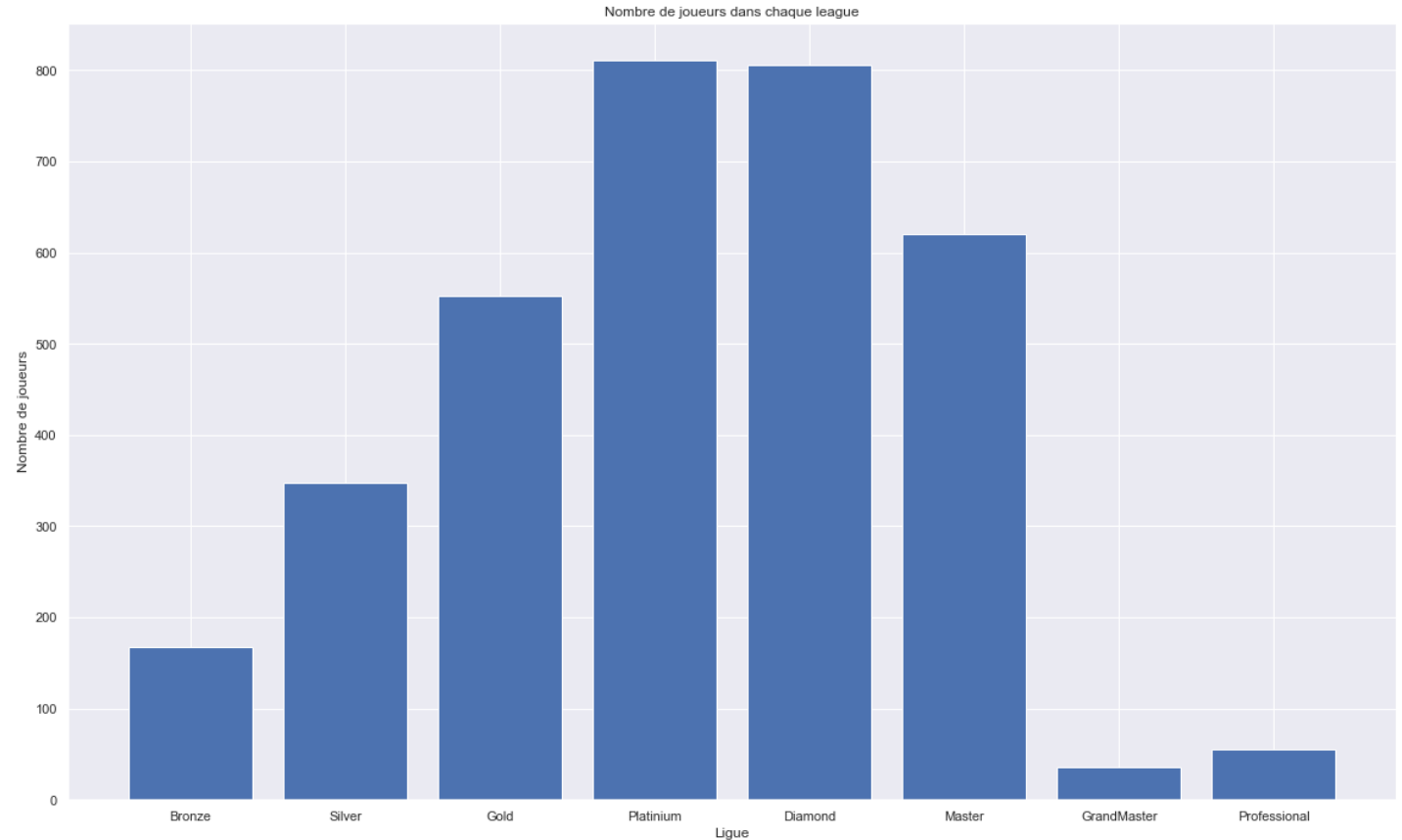


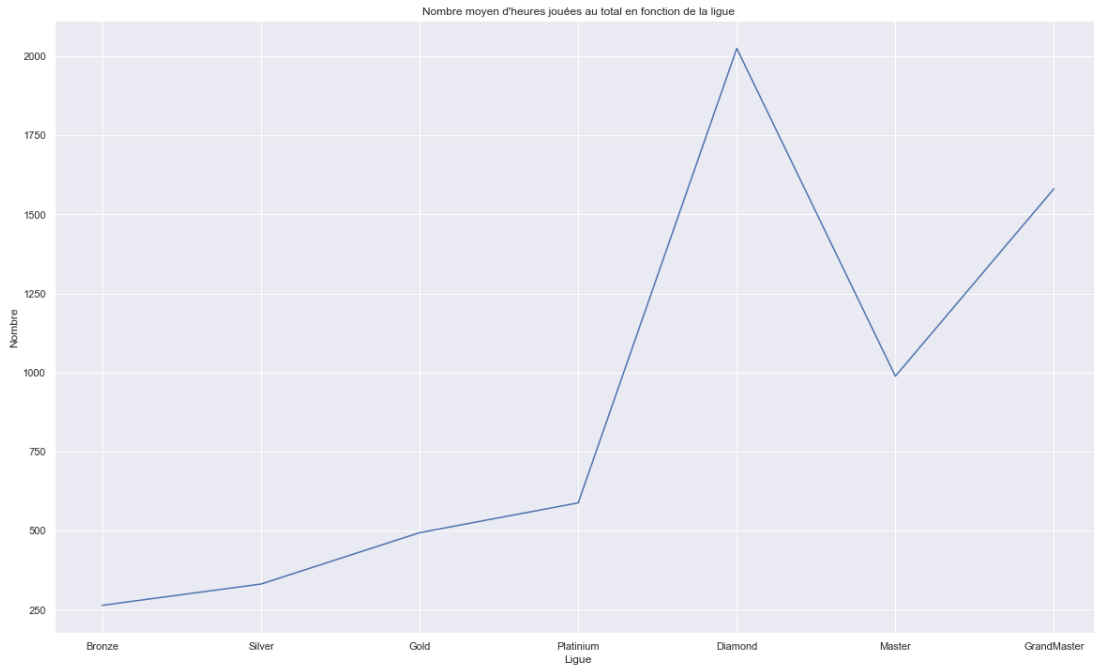
Thoughts on the problem and Data Exploration

- To start, we had to explore the dataset. For that I used the panda module.

I was able to do several visualizations to understand the dataset. For example, we can see the number of players per level here. I also used a maping to better understand the level of the players which goes from Bronze to GrandMaster / Professional, it is a level scale that we find in many video games nowadays.

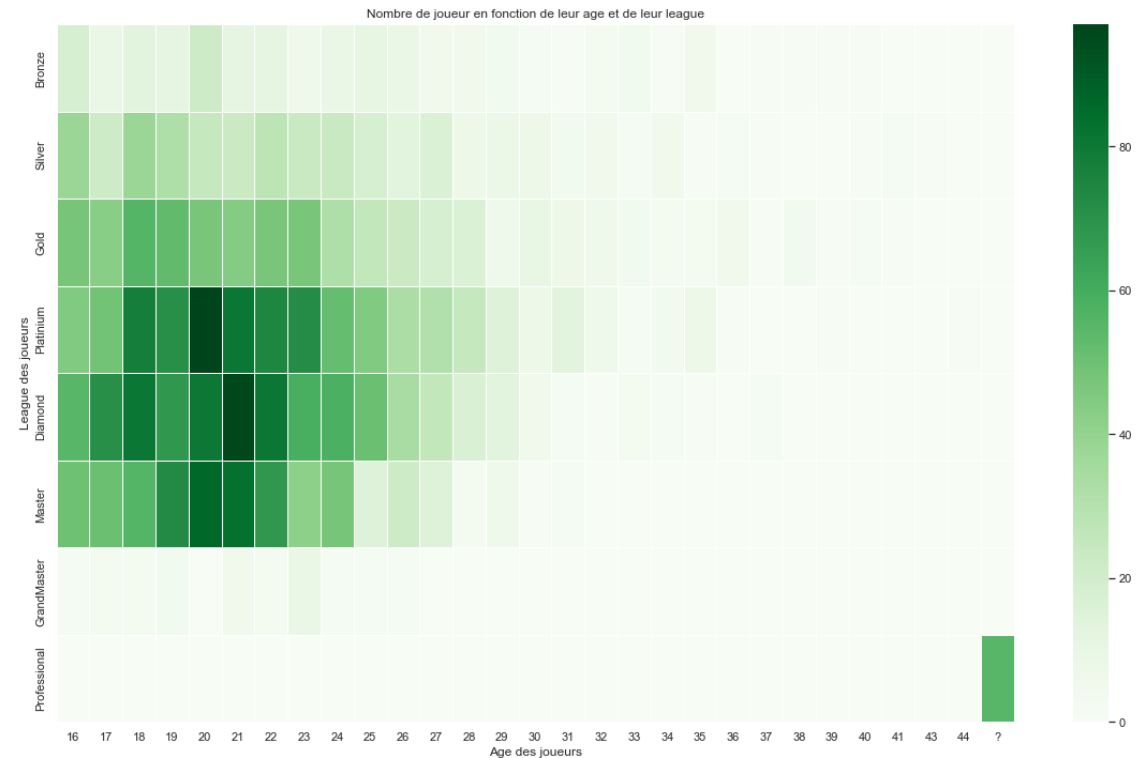
- Here we see that the distribution of player in the dataset seems to be of normal law in the dataset. Which may correspond to the real distribution of all players.

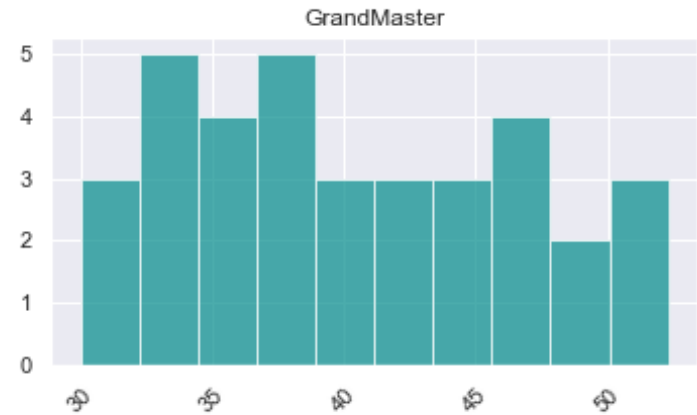
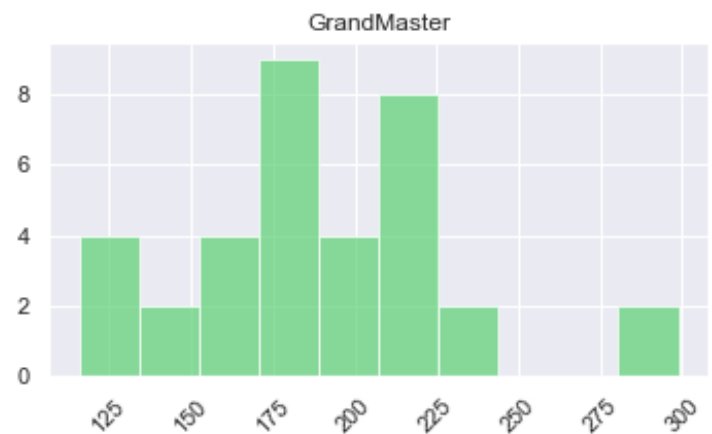
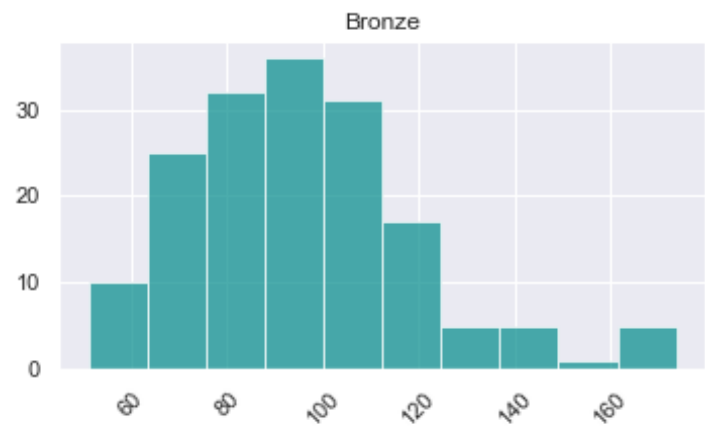
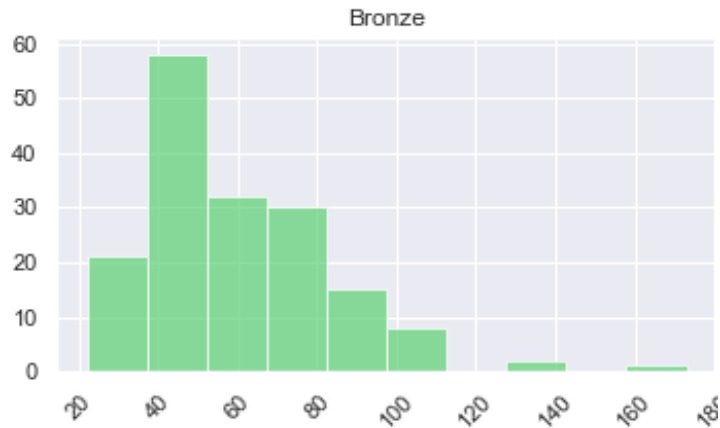




Here we can see the average number of hours played in total depending on the league. We can notice that up to the diamond level there is a linear relationship between playing time and level. But this is stopped by the Grand Master level. There would be a form of "talent" without a lot of playing time for these players. Knowing that the number of diamond and master players is appreciably close (800 and 650).

-
- Here I made a Heatmap of the age of players and their league.
 - We see that the game mainly interests young people and that many are between platinum and diamond as confirmed by the previous histogram.
 - We can also notice something very interesting: for the "professional" release we do not know the age of the players.
 - It was from this graph that I noticed that some features were missing data for these players, which is why I decided not to take them into account in the modeling part.





- I made several other visuals like histograms between bronze players and grandmaster players. By comparing these two types of players the differences are glaring.
- In green we have the APM, Action Per Minute, and we can clearly see that the grandmasters are much more efficient in this area than the bronze ones.
- In blue these are the Action Latency, which can be matched to the players' "thinking / action time". We can see that the GrandMasters are much faster.

-
- Here we have the comparison between the gold (blue) and the master (orange) which are almost as numerous. We can see that in the four features chosen the masters are always "better". The differences may seem slight but "all" the features together justify a better level of master players.
 - The visuals and graphics are available in the attached python notebook for better reading.



Modeling



- For the modeling part I used sklearn.
- I arbitrarily chose certain variables of the dataset to use. Then I divided my dataset (without the professionals) into training and testing set.
- I standardize the values using the minMaxScaler from sklearn.preprocessing. Finally I chose several models to test.

scikit-learn

Machine Learning in Python

[Getting Started](#) [Release Highlights for 1.0](#) [GitHub](#)

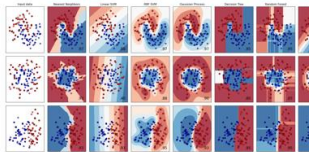
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



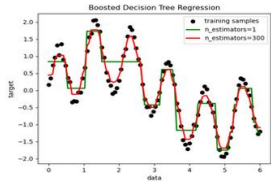
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



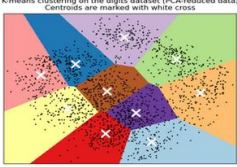
Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

-
- The scores obtained were not very interesting as shown here. This is probably due to a bad preprocessing on my part. I could have done a PCA to know the best features to use. I should have tested other sklearn transformers and done more testing.

```
modelTree.score(scaler.transform(X_test),y_test)
```

✓ 0.4s

0.3008982035928144

```
modelSVC.score(scaler.transform(X_test),y_test)
```

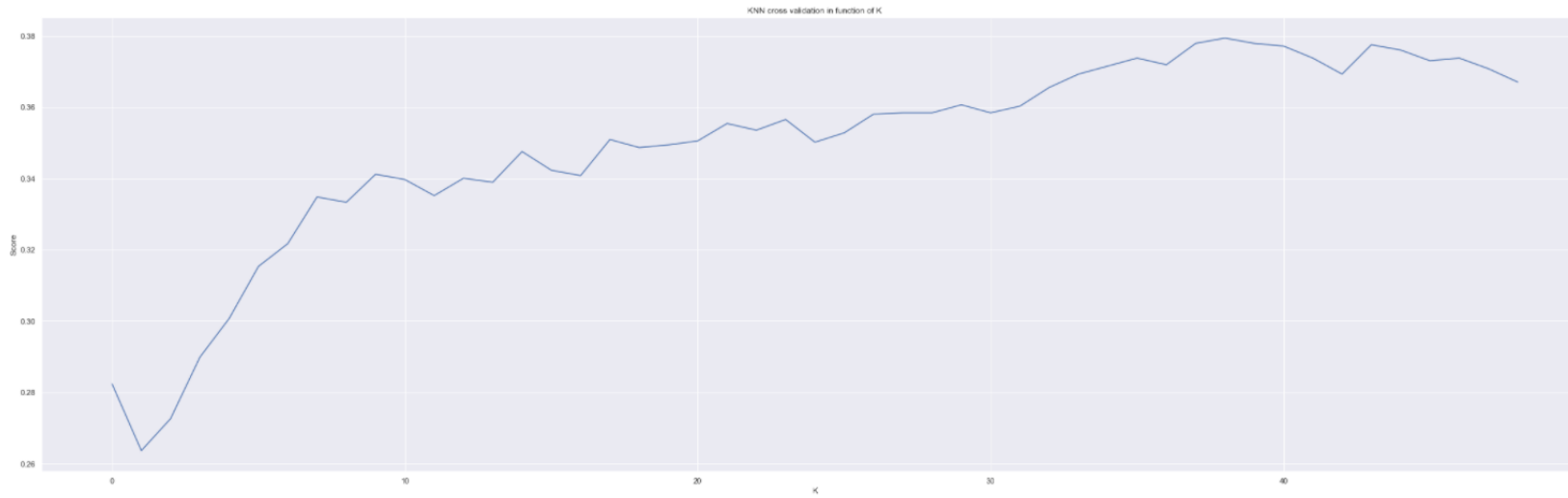
✓ 0.1s

0.39221556886227543

```
modelSGD.score(scaler.transform(X_test),y_test)
```

✓ 0.2s

0.36377245508982037



- Despite everything, I decided to test a cross validation on a KNN in addition to changing the K (from 1 to 50) and here is the result. This result is still not satisfactory but shows the concept of cross validation and change of hyperparameter of an algorithm.