ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

# Python for data analysis

## SkillCraft 1

Laurine SALLE & Maélis YONES DIA1

# Presentation of our dataset

# Presentation of our dataset

Our dataset, created in 2013 by Mark Blair, Joe Thompson, Andrew Henrey and Bill Chen, is a study of players from the video game StarCraft2 which is a game of real-time science-fiction strategy developed and published by Blizzard Entertainment.
Our dataset is composed of 20 features (19 variables and our target which is LeagueIndex) and 3395 instances of players.

The dataset contained some technical features such as :
- 1 second = 88,1 timestamp (game time system).
- PAC which means Perception-Action Cycle, this is all the actions done by a player in a specific area of the map.
- Hotkeys that are keys used as keyboard shortcut for perform specific actions on the game.

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

# Presentation of our dataset

The first step of our analysis was to clean the data if needed. We saw some inconsistencies (3 rows only) and decided to delete them, for example, we deleted a player who played more than 24 hours a day, which is impossible. We also chose to delete the GameId because it has no signification for the data analysis.
For the "?" values in Age, HoursPerWeek and TotalHours, we decided to replace them by the mean value of the corresponding column.

Then, we needed to analyse the variables and to see the ones that has impacts on the LeagueIndex and to make some conclusions. The league index is an integer between 1 and 8 associated with a league

| League Index | League Name |
|---|---|
| 8 | Professional |
| 7 | Grandmaster |
| 6 | Master |
| 5 | Diamond |
| 4 | Platinum |
| 3 | Gold |
| 2 | Silver |
| 1 | Bronze |

Finally, we tried to predict our target: the LeagueIndex of players thanks to classification models.

# Our priliminary thoughts

As a first, we thought that some features would have influence on the target like:

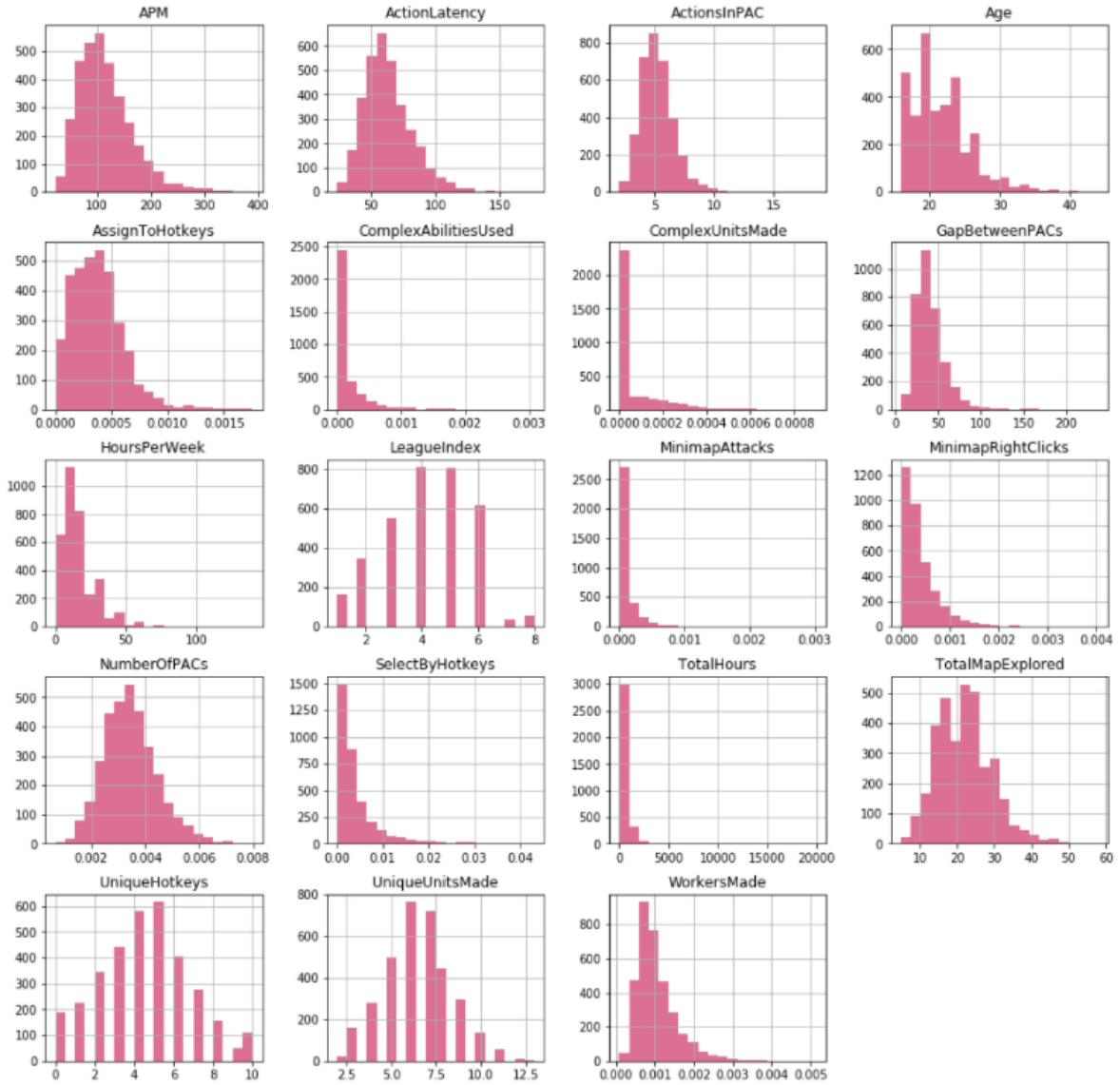
- The Age. Indeed, we thought that young player will be more reactive than old one so they would have a better level since we are in a strategic video game.

- The features related with the playing time such as HoursPerWeek and TotalHours. We thought that these would have a good impact on the player's level. Indeed, the more time he spends on the game, the more experience he gains and the better his level will be.

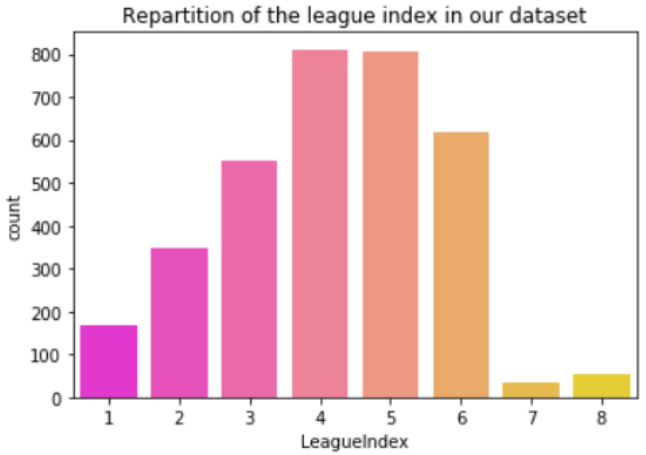
- We also thought that features like APM (Action Per Minute) and NumberOfPACs would have a positive impact and would be correlated with the LeagueIndex because it show the time it takes for the player to perceive an information and to react by taking action.

# Data Visualisation

# A global visualisation of all the variables of our dataset and a focus on our target

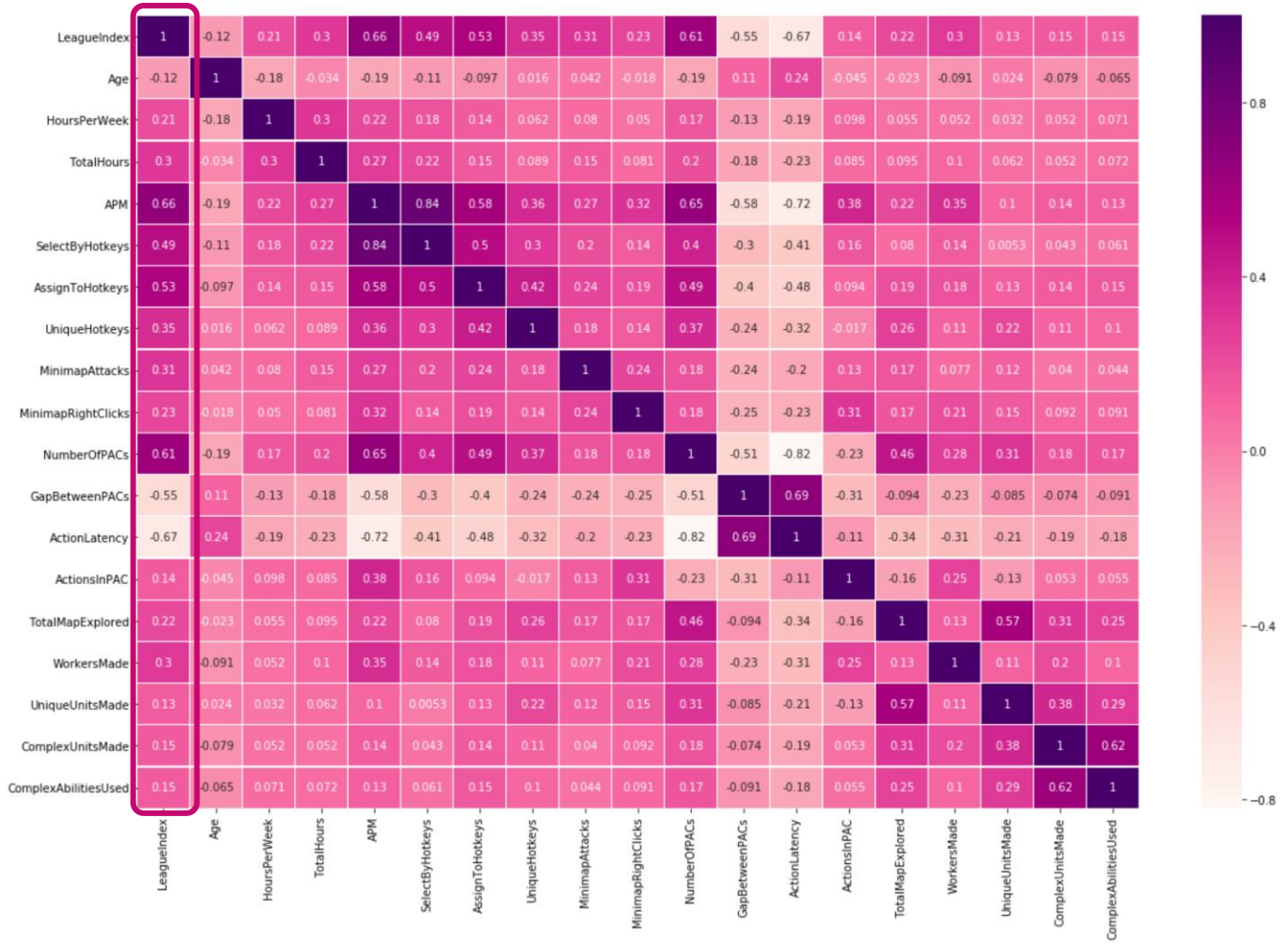



Our dataset seems to be mostly composed of a lot of player having a LeagueIndex between 3 and 6, then the number drops drastically, this might be a problem to well predict the LeagueIndex 7 and 8, because those players are underrepresented in the dataset (as we can see below).




We can also observe that the majority of the variables follows a normal distribution, according to the shape of the graphs.

# Correlations between all the variables of our dataset




We can observe on the correlation matrix the correlations between the variables and the LeagueIndex.
Contrary to our expectations on age, we can see that its correlation with the target is not significant because very close to 0 (-0.12)

The correlation matrix shows that some variables are significatively positively or negatively correlated with LeagueIndex.
Significant positive correlations with LeagueIndex: APM, SelectByHotkeys, AssignToHotkeys, NumberOfPACs
Significant negative correlations with LeagueIndex: GapBetweenPACs, ActionLatency.

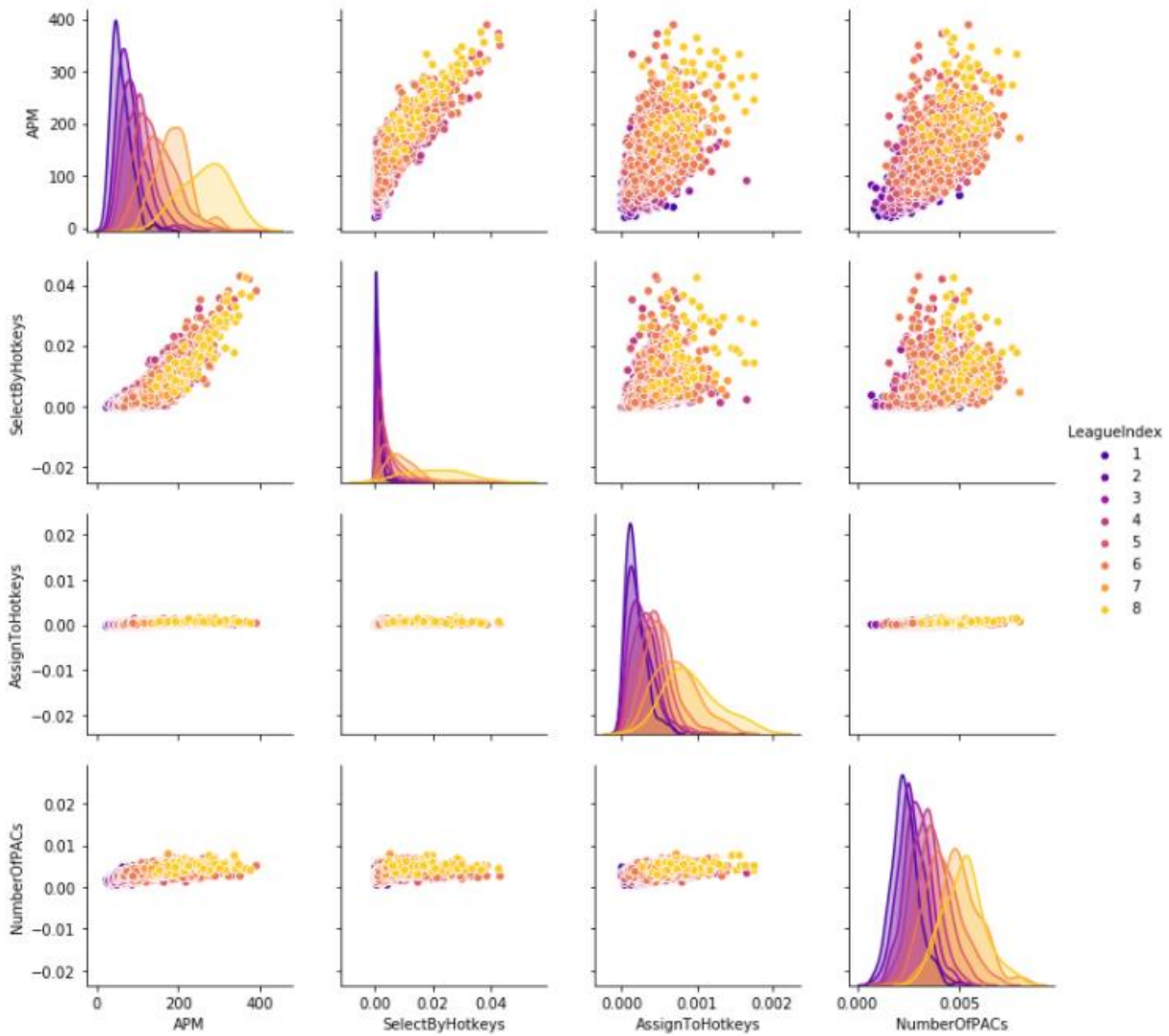
We can therefore anticipate the profile of a good player with a low latency, quick reaction, a big number of action, etc.

# Relationships between values positively correlated and the LeagueIndex

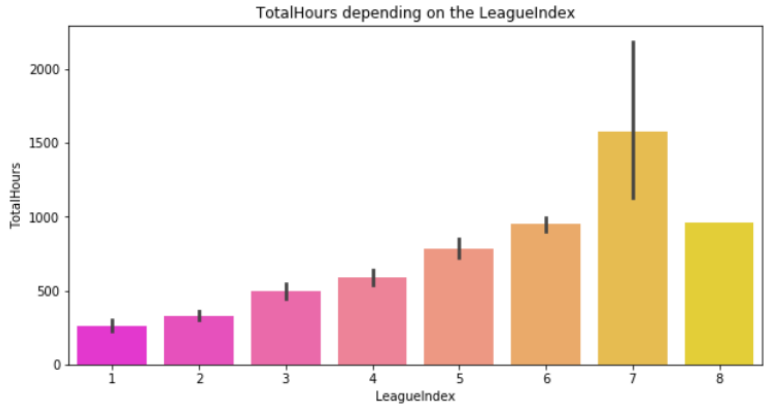
A low-level player (leagueIndex 1 to 3) has a low NumberOfPacs which induces few action cycles and therefore not too many activities in the game (low APM). This is supported by little use of Hotkeys keyboard shortcuts (SelectByHotkeys and AssignToHotkeys low).

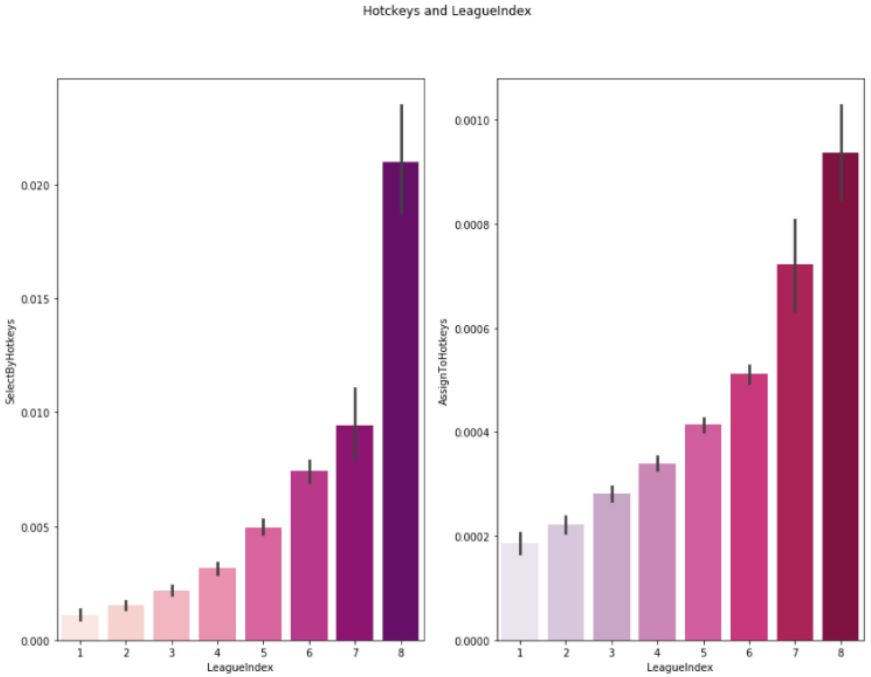
From this pairplot, we can deduce that a good player will have many actions per minute (APM) and NumberOfPACs.

We can also observe that, on any plane , when we try to represent the population of the dataset there is no true separation between the observations, and we can only observe that players having a LeagueIndex of 8 are slightly more grouped in certain plots. For the rest of the population the groups are overlapping each other a lot.



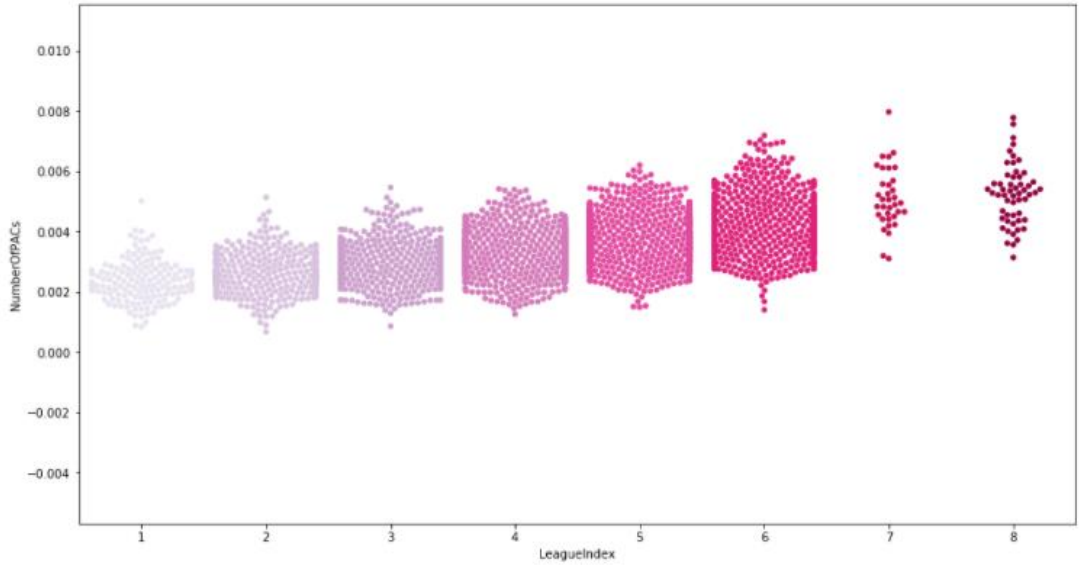
# The profil of a good player




Through all these graphics, we can clearly see the importance of keyboard shortcut (SelectByHotckeys and AssignToHotckeys variables) because it allows the players to go faster and thus optimizes the playing time by increasing the actions performed (NumberOfPACs). Logically, players spending the most time on the game have higher LeagueIndex (not for League 8 which represents only 1% of the dataset so it might not represent the reality.

NumberOfPACs according to the LeagueIndex



# Machine Learning

# Machine Learning : 1st approach : Prediction of the LeagueIndex

Our 1st goal was to predict the LeagueIndex

- We created a function that will train all the models and return the accuracies for each, and compare them in a graphic.
- We applied this function with cross validation to have a result that reflects the training and testing on all the available data.

We tried to model our data with different algorithms, which are:

| classifier |
|---|
| Random Forest |
| Gradient Boosting Classifier |
| Naive Bayes |
| Logistic Regression |
| Nearest Neighbors |
| Decision Tree |
| Linear SVM |

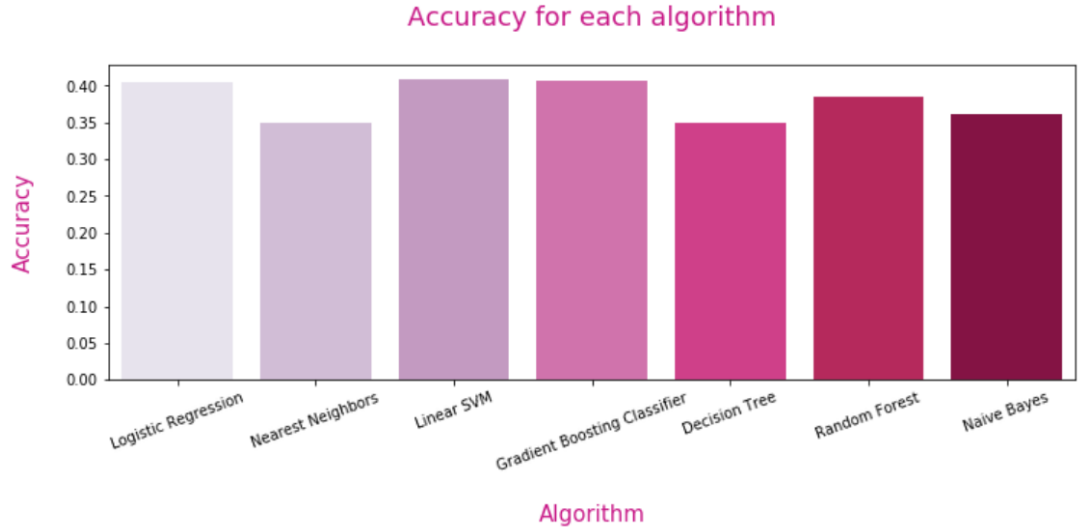
Our 1st results:

- We tried all the models in normalized and not normalized values, and compared the accuracies

| Not Normalized values | Normalized values |
|---|---|
| Gradient Boosting (0.400) | Linear SVM (0.408) |
| Logistic Regression (0.397) | Gradient Boosting (0.407) |
| Naive Bayes (0.394) | Logistic Regression (0.394) |

The best results are around 0.40, which is very low. The results does not change a lot with normalization.

Comparision of all the models accuracies for Normalized values:



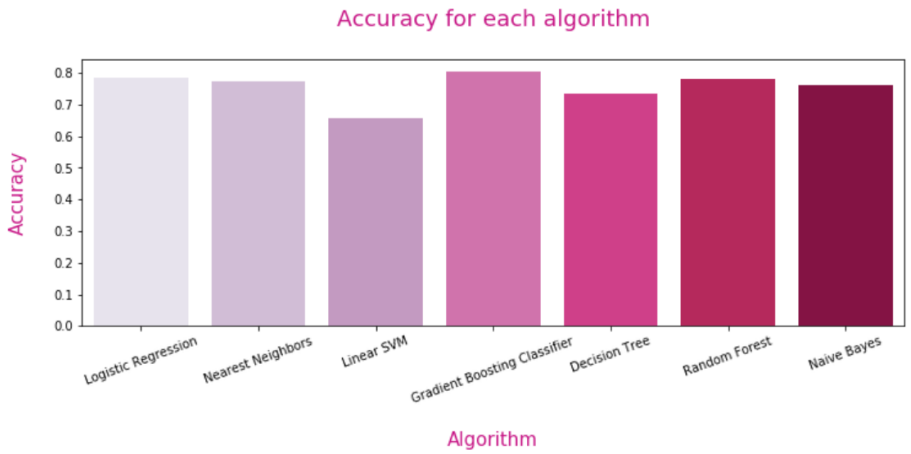
# Machine Learning : 2nd approach : Prediction of a player's Level

Considering the low accuracy obtained when trying to predict the exact LeagueIndex of players, we tried a new approach.
We grouped League Indexes by levels, as showns on the table

| | | |
|---|---|---|
| 1 | Beginner Player | League 1- 3 |
| 2 | Intermediate Player | League 4 – 6 |
| 3 | Advanced Player | League 7 - 8 |

We trained the same models as before to predict this new output. We tried them with normalized and not normalized values.

Comparision of all the models accuracies for not normalized values:




Top 3 :
Gradient Boosting (0.803)
Random Forest (0.784)
Logistic Regression (0.782)

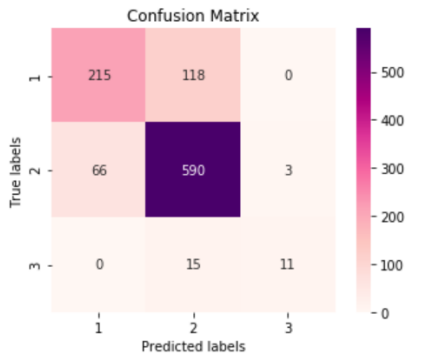
From those results, we performed a Grid search on the Gradient Boosting classifier and the Logistic Regression, which are the best model for all the predictions we made. We chose to do the grid search with non scaled values.

| | |
|---|---|
| Accuracy for Logistic Regression | 0.793 |
| Accuracy for Gradient Boosting | 0.804 |

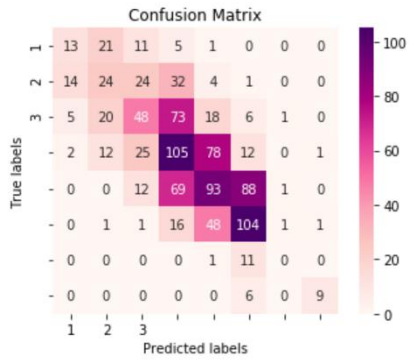
So, after the grid search, the Gradient Boosting Classifier is the best model, and, combined with the new labels mapping, we obtained 80.4% accuracy.

We applied the Gradient boosting model, to predict the new target variable, and we did the same grid search to predict the exact league indexes. Here are the confusion matrices we obtained.

Confusion matrix for grouped output




Confusion matrix for LeagueIndex output



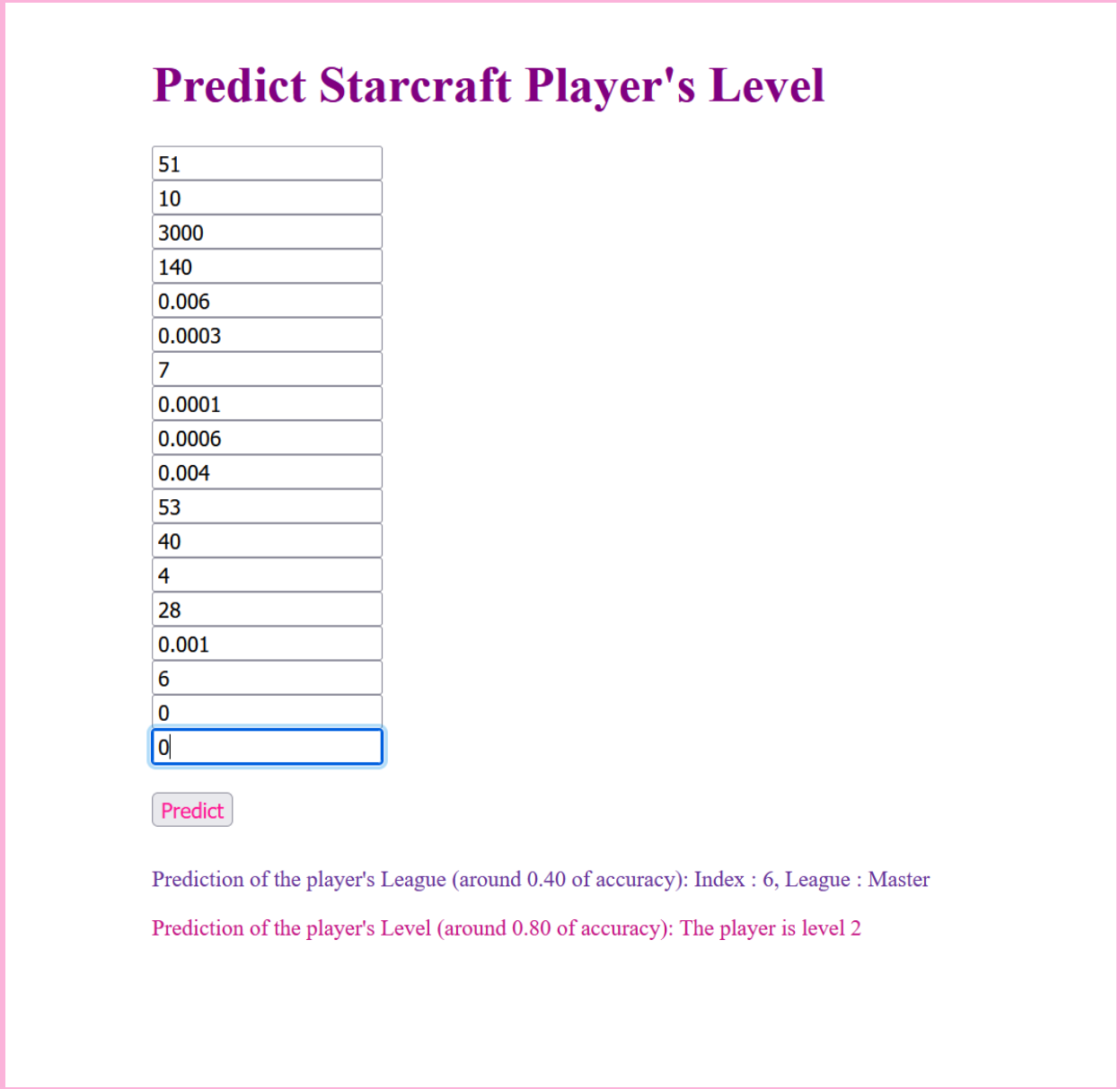
# API - Interface Flask

# API Flask : Our prediction model

For the Flask API, we allow the user to enter a list of player's parameters (all the variables), and then the application will predict the player's League (from 1 to 8), and the player's Level (from 1 to 3)

Here are the steps to use the api:

Download the entire projet, open your Anaconda Prompt (or any python supported cmd) and go to our project's folder (with cd command).
Then enter "python app.py", and copy the localhost link (like http://127.0.0.1:8000/) in your browser.
Fill the form and use the "Predict" button to see the results.



# Thank you for reading!

Laurine SALLE & Maélis YONES DIA1