SkillCraft Dataset

Presented by Sami Laribi & Arslene Abdi

Date 10-01-2021

Plan

I. Dataset SkillCraft Description
II. Data Analyzing
III. Data cleaning
IV. Fiting models
    1. Logestic regression
    2. Random forest
    3. Multinomial
    4. KNN
V. Model comparisons
VI. Casting results on Flask
VII. conclusion

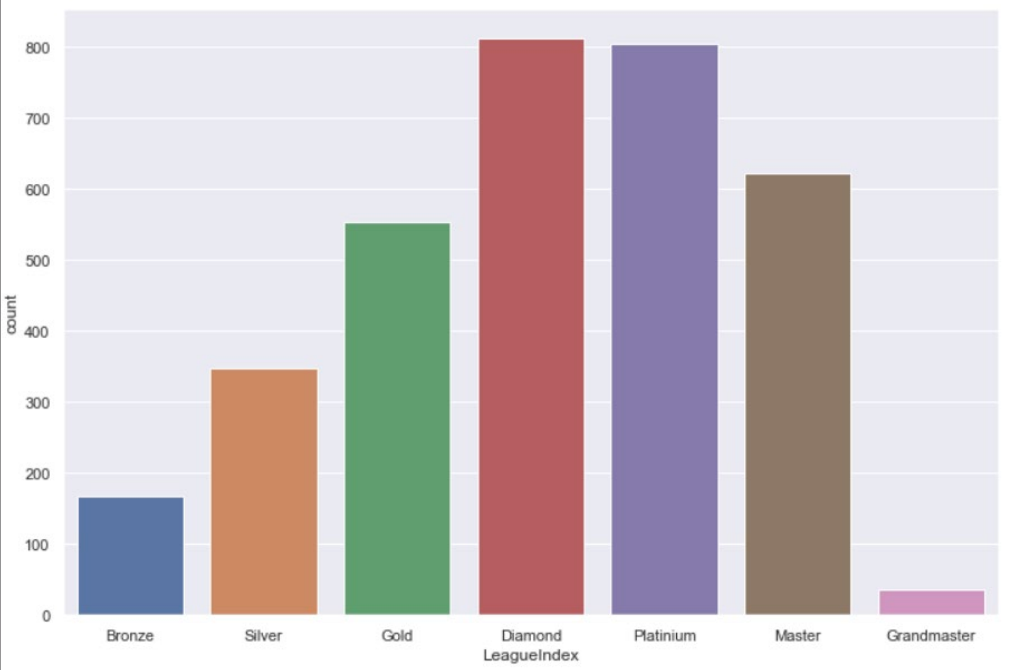
I. Dataset SkillCraft Description

- The dataset used in this project is the SkillCraft1 master table dataset from UCI machine learning repository. It aggregates screen movements into screen-fixations using a Salvucci & Goldberg (2000) dispersion-threshold algorithm, and defined Perception Action Cycles (PACs) as fixations with at least one action.
- In simple words, they actually tried to find the league of players based on their screen movement when playing the game.

I. Data Analyzing

After analyzing our Dataset , we came up with these caracteristics :

- Dimensions : 3395 rows × 20 columns
- Target : **LeagueIndex**
- Partitions of Data :



We have 7 classes in the league there is not the professionel leagues as mentioned in the dataset description and as we can see the Data varries a lot in this Dataset , and this is a little bit concerning in fitting models later as we can have an underfitted models.

- After analysing each variable with the target and ploting their graphs we noticed that we need to remove some features in order to have better model .
- As you can see there is a huge correlation between different features , so we had to clean our dataset , I can't spoil you but it did make a difference in improving the scores .
- Cor(gameid,leagueindex)=0.025
- Cor(age,leagueindex)=-0.13
- Cor(actioninPAC,leagueindex)=0.14
- Cor(complexabilitiesused,leagueindex)=0.16
- Cor(uniqueunitmade,leagueindex)=0.15
- Cor(complexunitmade,leagueindex)=0.17
- Cor(Totalhours,leagueindex)=0.024

	GameID	LeagueIndex	Age	HoursPerWeek	TotalHours	APM	SelectByHotkeys	AssignToHotkeys	UniqueHotkeys	MinimapAttacks	MinimapRightClicks	NumberOfPACs	GapBetweenPACs	ActionLatency	ActionsInPAC	TotalMapExplored	WorkersMade	UniqueUnitsMade	ComplexUnitsMade	ComplexAbilitiesUsed
GameID	1	0.025	0.089	-0.025	0.0042	0.076	0.07	0.064	0.081	0.022	0.0099	0.036	-0.066	-0.043	0.039	0.031	-0.015	0.029	-0.018	0.0041
LeagueIndex	0.025	1	-0.13	0.22	0.024	0.62	0.43	0.49	0.32	0.27	0.21	0.59	-0.54	-0.66	0.14	0.23	0.31	0.15	0.17	0.16
Age	0.089	-0.13	1	-0.18	-0.017	-0.21	-0.13	-0.1	0.015	0.043	-0.02	-0.2	0.11	0.24	-0.046	-0.024	-0.092	0.023	-0.08	-0.066
HoursPerWeek	-0.025	0.22	-0.18	1	0.024	0.25	0.21	0.16	0.07	0.084	0.049	0.17	-0.13	-0.19	0.095	0.065	0.051	0.039	0.059	0.075
TotalHours	0.0042	0.024	-0.017	0.024	1	0.073	0.082	0.042	0.0093	0.00087	0.0077	0.04	-0.021	-0.036	0.011	0.02	0.015	-0.0024	-0.0072	-0.0063
APM	0.076	0.62	-0.21	0.25	0.073	1	0.81	0.53	0.34	0.22	0.31	0.64	-0.57	-0.72	0.4	0.24	0.38	0.12	0.16	0.14
SelectByHotkeys	0.07	0.43	-0.13	0.21	0.082	0.81	1	0.45	0.27	0.13	0.11	0.36	-0.27	-0.39	0.17	0.097	0.16	0.028	0.065	0.064
AssignToHotkeys	0.064	0.49	-0.1	0.16	0.042	0.53	0.45	1	0.4	0.21	0.15	0.45	-0.38	-0.46	0.092	0.2	0.2	0.15	0.17	0.17
UniqueHotkeys	0.081	0.32	0.015	0.07	0.0093	0.34	0.27	0.4	1	0.15	0.12	0.35	-0.22	-0.3	-0.022	0.27	0.11	0.23	0.12	0.11
MinimapAttacks	0.022	0.27	0.043	0.084	0.00087	0.22	0.13	0.21	0.15	1	0.22	0.14	-0.21	-0.17	0.13	0.16	0.082	0.13	0.052	0.042
MinimapRightClicks	0.0099	0.21	-0.02	0.049	0.0077	0.31	0.11	0.15	0.12	0.22	1	0.14	-0.24	-0.22	0.32	0.17	0.21	0.15	0.098	0.096
NumberOfPACs	0.036	0.59	-0.2	0.17	0.04	0.64	0.36	0.45	0.35	0.14	0.14	1	-0.49	-0.82	-0.24	0.47	0.28	0.32	0.2	0.18
GapBetweenPACs	-0.066	-0.54	0.11	-0.13	-0.021	-0.57	-0.27	-0.38	-0.22	-0.21	-0.24	-0.49	1	0.68	-0.31	-0.095	-0.24	-0.09	-0.083	-0.092
ActionLatency	-0.043	-0.66	0.24	-0.19	-0.036	-0.72	-0.39	-0.46	-0.3	-0.17	-0.22	-0.82	0.68	1	-0.11	-0.35	-0.31	-0.22	-0.2	-0.19
ActionsInPAC	0.039	0.14	-0.046	0.095	0.011	0.4	0.17	0.092	-0.022	0.13	0.32	-0.24	-0.31	-0.11	1	-0.16	0.25	-0.13	0.054	0.053
TotalMapExplored	0.031	0.23	-0.024	0.065	0.02	0.24	0.097	0.2	0.27	0.16	0.17	0.47	-0.095	-0.35	-0.16	1	0.13	0.58	0.31	0.25
WorkersMade	-0.015	0.31	-0.092	0.051	0.015	0.38	0.16	0.2	0.11	0.082	0.21	0.28	-0.24	-0.31	0.25	0.13	1	0.11	0.2	0.1
UniqueUnitsMade	0.029	0.15	0.023	0.039	-0.0024	0.12	0.028	0.15	0.23	0.13	0.15	0.32	-0.09	-0.22	-0.13	0.58	0.11	1	0.38	0.29
ComplexUnitsMade	-0.018	0.17	-0.08	0.059	-0.0072	0.16	0.065	0.17	0.12	0.052	0.098	0.2	-0.083	-0.2	0.054	0.31	0.2	0.38	1	0.62
ComplexAbilitiesUsed	0.0041	0.16	-0.066	0.075	-0.0063	0.14	0.064	0.17	0.11	0.042	0.096	0.18	-0.092	-0.19	0.053	0.25	0.1	0.29	0.62	1

1.00 0.75 0.50 0.25 0.00 -0.25 -0.50 -0.75

I. Data Cleaning

- First of all , we noticed that there is some rows with '?' Inside witch we think its some corrupted data , so we had to remove them .
- Then we searched for missing values , we hadn't found anything .
- There is 3 features that have an object type that will make it difficult for the work so we had to change them to 'integer'

```
Out[6]: GameID                    int64
        LeagueIndex               int64
        Age                      object
        HoursPerWeek             object
        TotalHours               object
        APM                     float64
        SelectByHotkeys         float64
        AssignToHotkeys         float64
        UniqueHotkeys             int64
        MinimapAttacks          float64
        MinimapRightClicks      float64
        NumberOfPACs            float64
        GapBetweenPACs          float64
        ActionLatency           float64
        ActionsInPAC            float64
        TotalMapExplored          int64
        WorkersMade             float64
        UniqueUnitsMade           int64
        ComplexUnitsMade        float64
        ComplexAbilitiesUsed    float64
```

- Its time to remove the unwanted features :

"GameID","LeagueIndex","Age","TotalHours","ActionsInPAC",

"UniqueUnitsMade","ComplexUnitsMade","ComplexAbilitiesUsed"

Our dataset is clean and ready for some work .

I. Fitting models

1. Logistic regression :

We began our journey with logestic regression since it's the easiest one to come to mind and our target is categorical .

After implementing and fiting the model we had a score of **0.38** we thought that this score is too low so we tried cross-validation since it can give better results and we chose our fold number to be equal to 5 (its actually the default one it gives normally the best results) :

```
model_0 = LogisticRegressionCV(cv=5).fit(x_train, y_train)
```

and at last we got a score of **0.417**

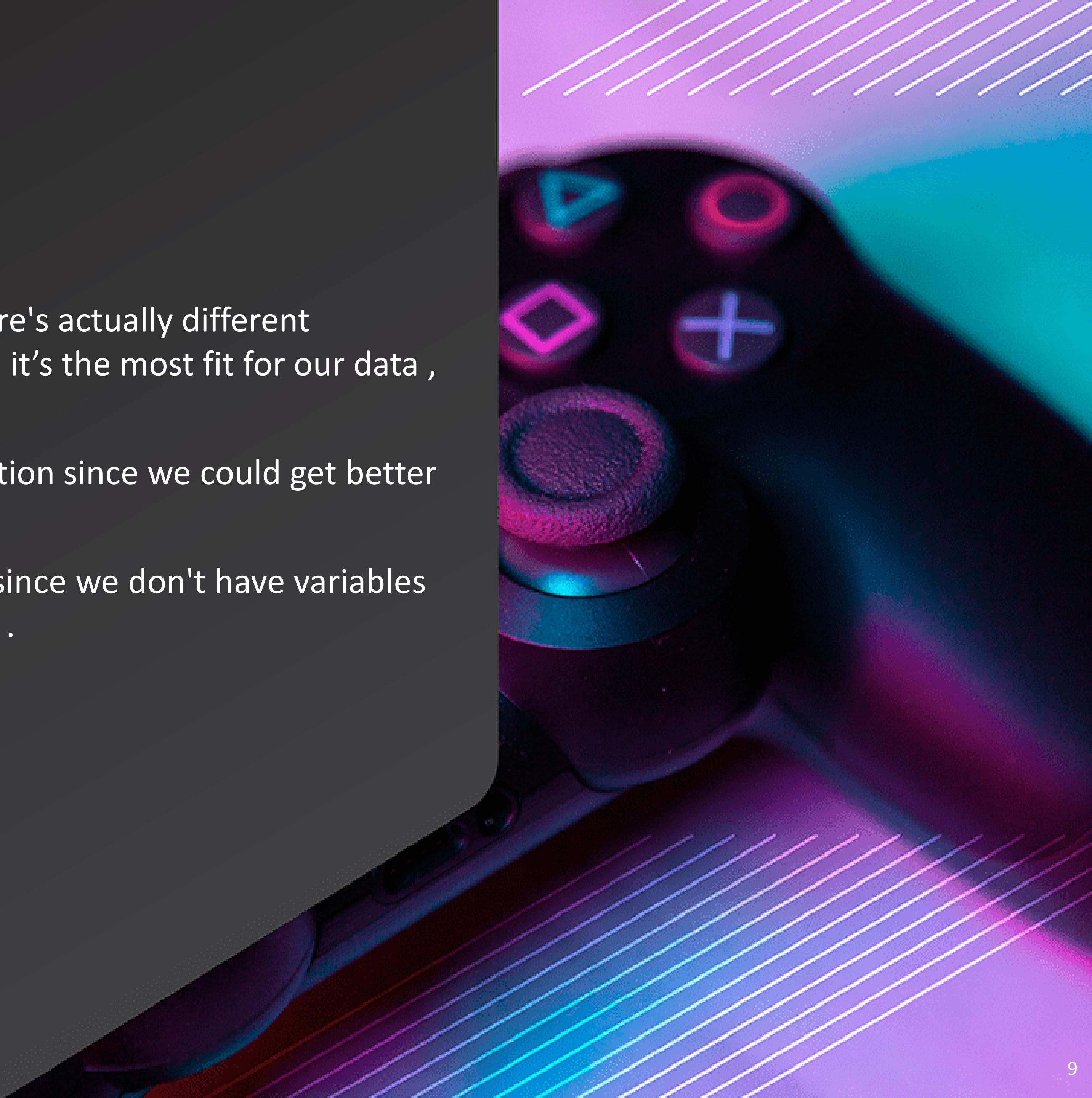
This score is actually very casual in such a dataset because of the variation of the distribution of the data that we saw before .

1. Naive Bayes :

We re going to use now the naive bayes model , there's actually different techniques but we chose the gaussian method since it’s the most fit for our data , and we got **0.34** ,

We didn't actually stop their we did the cross-validation since we could get better results and it was the case , we got **0.37** .

We couldn't plot anything with these two methods since we don't have variables to play with , so we tried the random forest method .



1. Random forest :

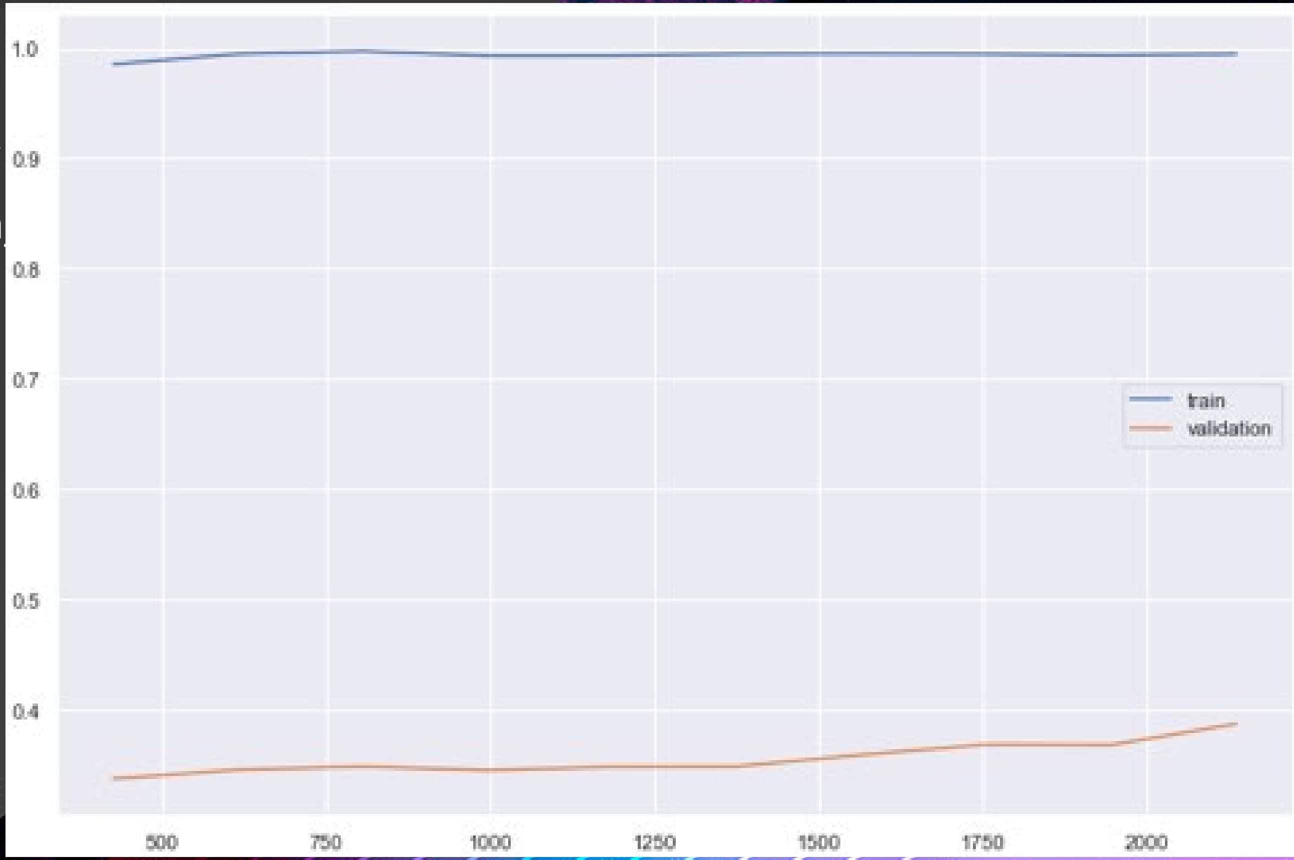
Its random forest turn , so we tried to implement the algorithm with random paremeters :

clf2 = RandomForestClassifier(n_estimators = 240, random_state = 123)

We got **0.41** its quiet good, but we wanted to go even further , we tried the grid search , it's a function that finds the best parameters for a model , but unfortunately we got only **0.39** because we didn't play with the random because it takes too much time for a single one and days if we do it 100 time .

This the graph of repeated testing on Random forest with different parameters :

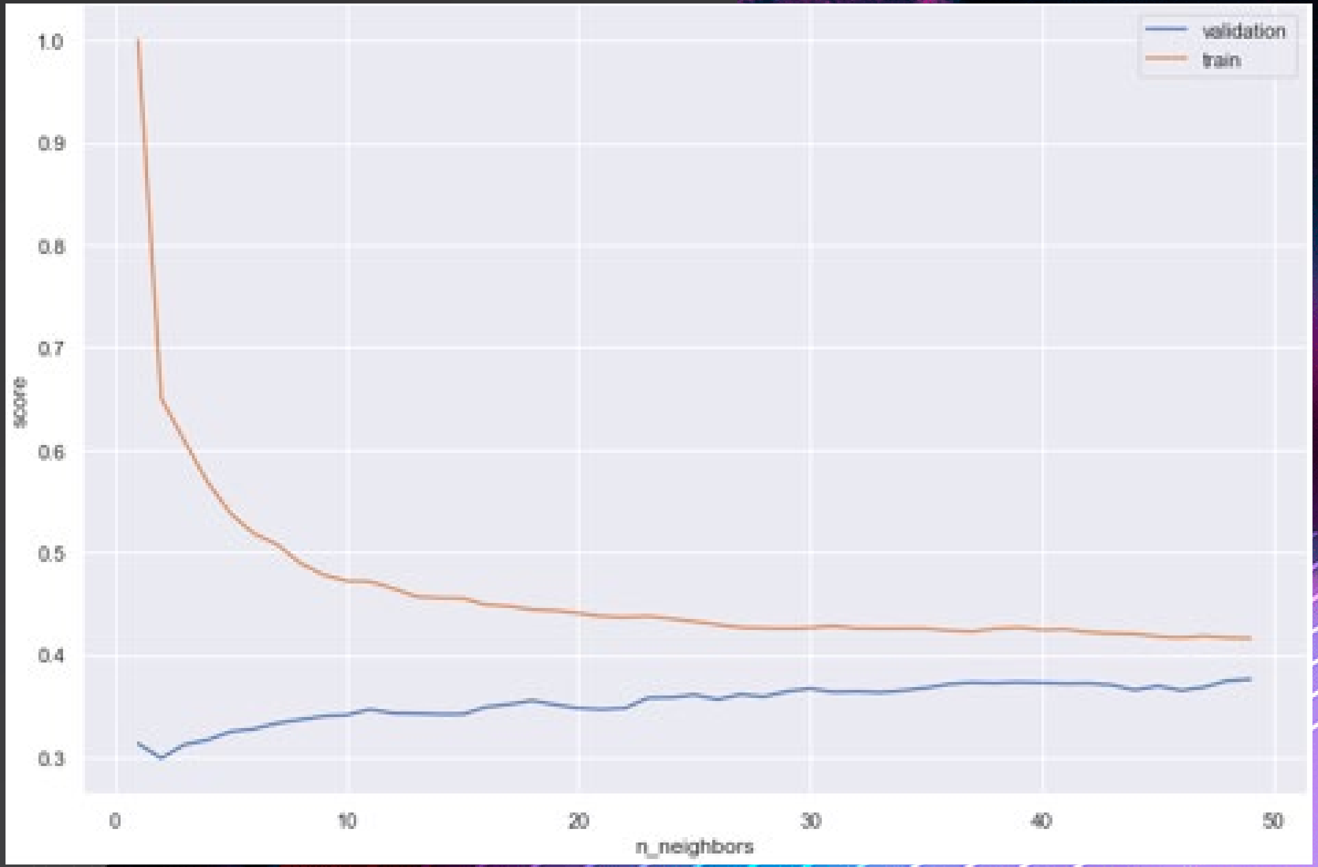
The graph is like that because of the lack of data in the dataset ! .



1. KNN :

Finally we went with our last hope KNN (k-nearest neighbors) to get the best scores, but sadly we got only **0.38** when k equal to 28 , and we got the same result with the grid search .

This is the graph of the algorith when k varies from 1 to 50 , as you can see the score is rising very slowly towards the **0.4** and this is due to the repartition of data in the Dataset .



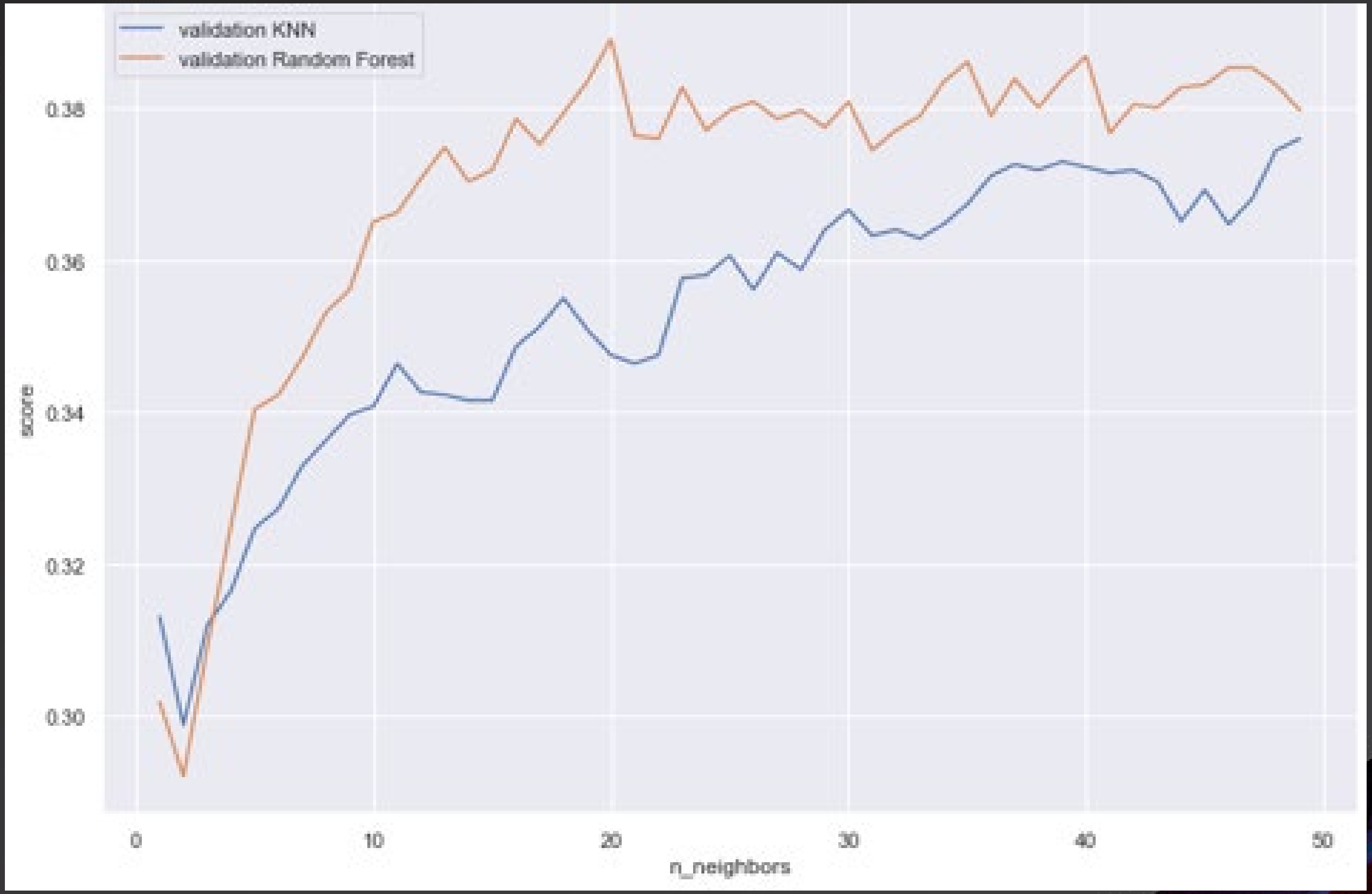
I. Models comparisons

We now will compare all the models that we've done in this table :

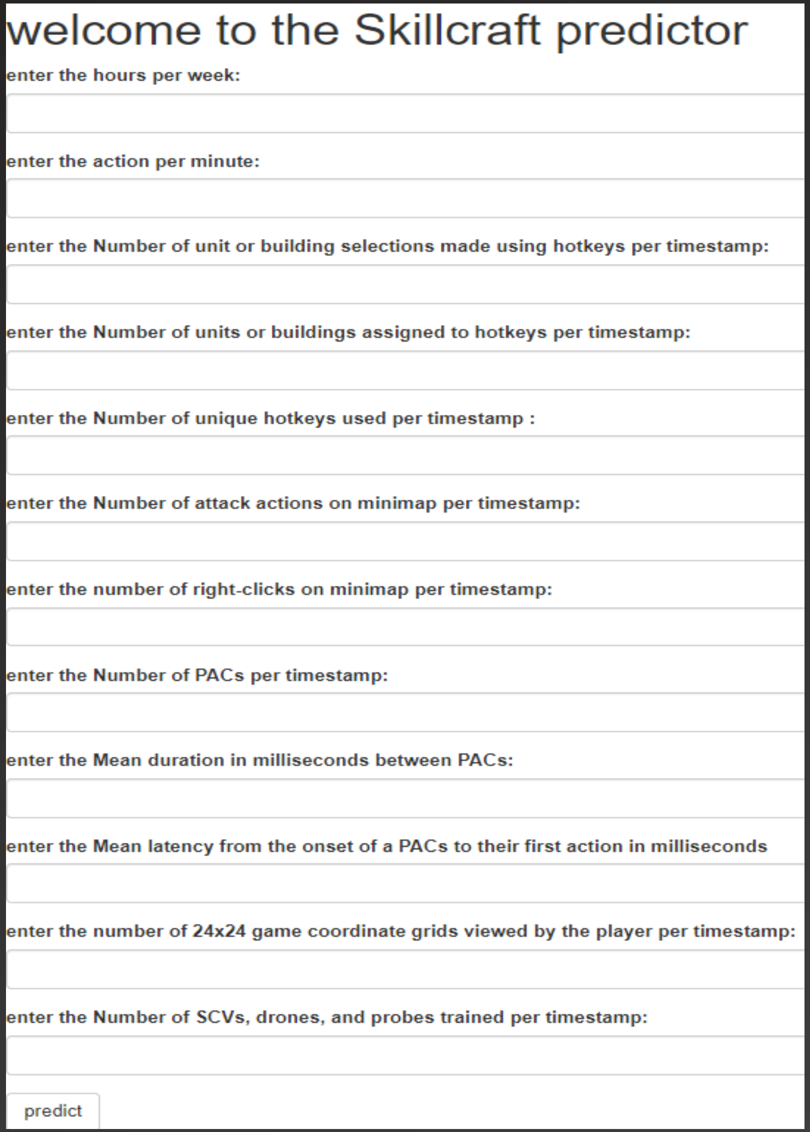
Logestic regression		Naive bayes		Random Forest		KNN	
normal	Cross-validation	Gaussian	Cross-validation	Random parameters	Grid search	normal	Grid search
0.38	0.417	0.34	0.37	0.410	0.39	0.38	0.38

The best score is with the logestic regression with the cross-validation , but we can see that it has similar score with the random forest, but we took the simpler and no time costing one which is the logestic regression .

This is a comparison graph between Random Forest and KNN which we can observe that random forest is better than KNN



I. Casting results on Flusk



welcome to the Skillcraft predictor

enter the hours per week:

enter the action per minute:

enter the Number of unit or building selections made using hotkeys per timestamp:

enter the Number of units or buildings assigned to hotkeys per timestamp:

enter the Number of unique hotkeys used per timestamp :

enter the Number of attack actions on minimap per timestamp:

enter the number of right-clicks on minimap per timestamp:

enter the Number of PACs per timestamp:

enter the Mean duration in milliseconds between PACs:

enter the Mean latency from the onset of a PACs to their first action in milliseconds

enter the number of 24x24 game coordinate grids viewed by the player per timestamp:

enter the Number of SCVs, drones, and probes trained per timestamp:

predict

We have created a flask application that it contains a form which has the features that the person need to fill in order to have his league level as showed in the two pictures.

I. conclusions

After analysing the dataset and implementiing different algorithms , and with the scores that we got , we can say that the dataset is lacking for some data , all models are underfitted , if you remember there is some classes with almost no data in them , we suggest that we remove thae class "7" and make a fusion with the class "6" and the class "1" with the class "2" , so we get better predictions and results , or they can update the dataset with more information that will give it the push that it needs .

Thank you !