AVILA

Python For Data Analysis

Kevin Gomes

DATA VISUALIZATION

COMPOSITION

Shape

20867 rows & 11 columns

Target

Last column ('monk' → class)

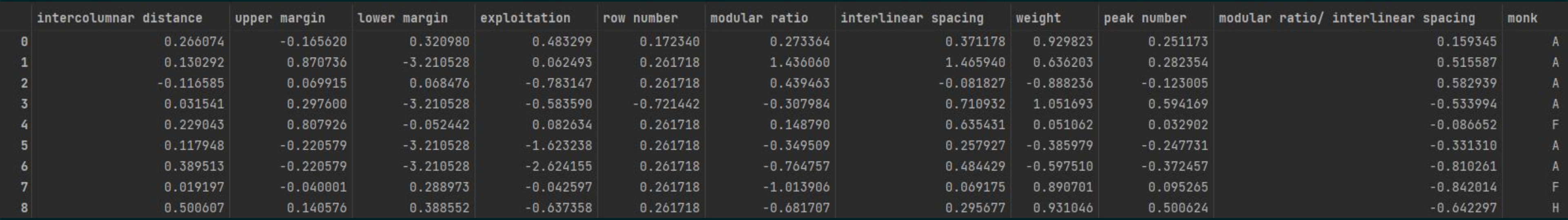
NaN

No NaN in all the Data Set

QUICK OVERVIEW

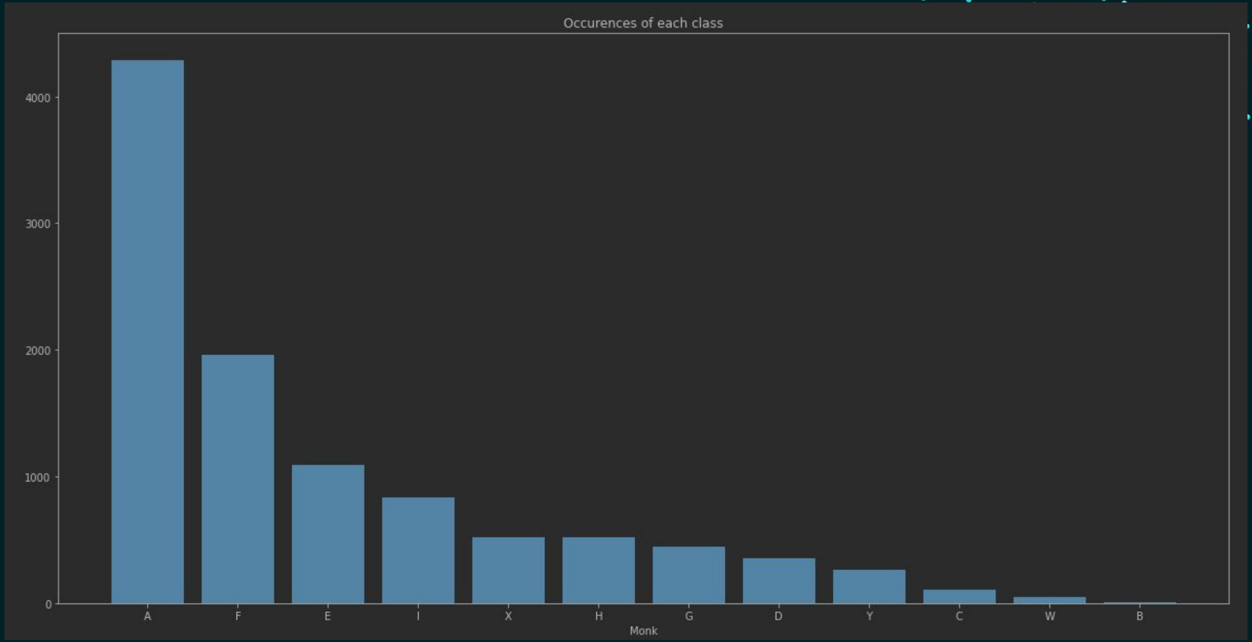
- Data already normalized by using the Z-Normalization method
- Divided in 2 data sets, train & test, with 10430 and 10437 samples respectively

Preview of the first 9 lines of the test set to see what the dataset looks like :



	intercolumnar distance	upper margin	lower margin	exploitation	row number	modular ratio	interlinear spacing	weight	peak number	modular ratio/ interlinear spacing	monk
0	0.266074	-0.165620	0.320980	0.483299	0.172340	0.273364	0.371178	0.929823	0.251173	0.159345	A
1	0.130292	0.870736	-3.210528	0.062493	0.261718	1.436060	1.465940	0.636203	0.282354	0.515587	A
2	-0.116585	0.069915	0.068476	-0.783147	0.261718	0.439463	-0.081827	-0.888236	-0.123005	0.582939	A
3	0.031541	0.297600	-3.210528	-0.583590	-0.721442	-0.307984	0.710932	1.051693	0.594169	-0.533994	A
4	0.229043	0.807926	-0.052442	0.082634	0.261718	0.148790	0.635431	0.051062	0.032902	-0.086652	F
5	0.117948	-0.220579	-3.210528	-1.623238	0.261718	-0.349509	0.257927	-0.385979	-0.247731	-0.331310	A
6	0.389513	-0.220579	-3.210528	-2.624155	0.261718	-0.764757	0.484429	-0.597510	-0.372457	-0.810261	A
7	0.019197	-0.040001	0.288973	-0.042597	0.261718	-1.013906	0.069175	0.890701	0.095265	-0.842014	F
8	0.500607	0.140576	0.388552	-0.637358	0.261718	-0.681707	0.295677	0.931046	0.500624	-0.642297	H

CLASS DISTRIBUTION



The number of occurrences of each class shows an unequal distribution of classes.

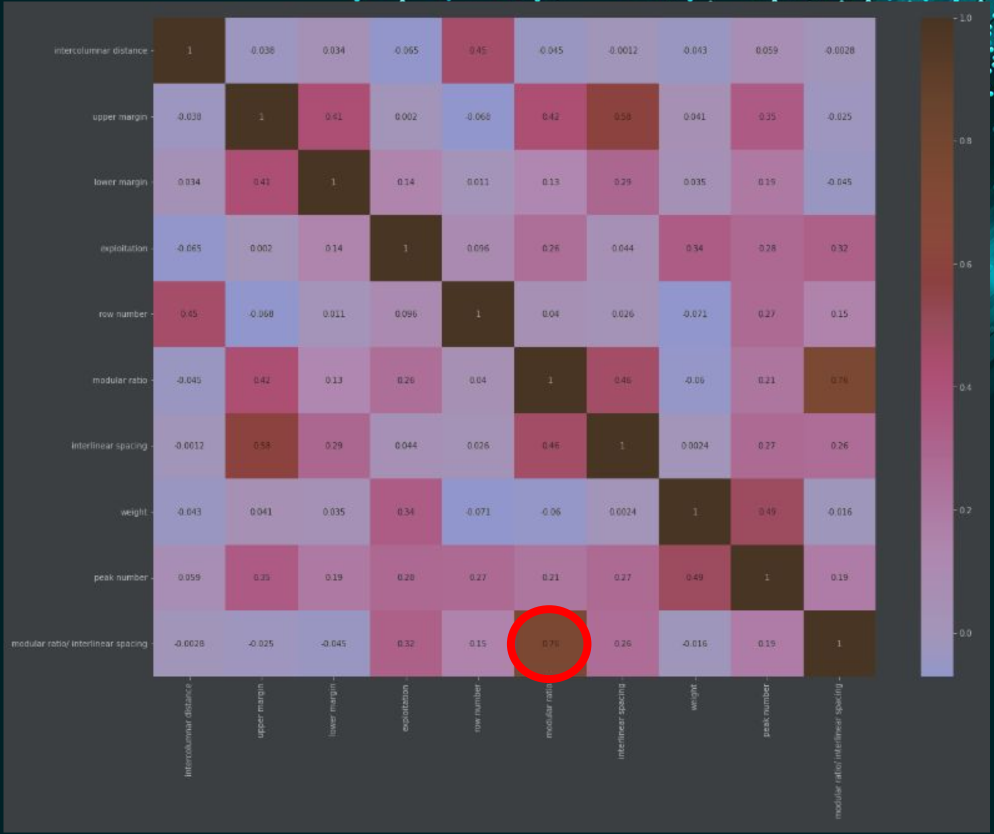
While the most present class, A, appears more than 4000 times, class B appears only 5 times (training set).

CORRELATION MATRIX

The Correlation Matrix is used to observe the links between the attributes.

In our case, the only 2 attributes that are too strongly correlated are ‘modular ratio’ and ‘modular ratio / interlinear spacing’ with 76% of correlation. (red circle)

To overcome this correlation, I decided to remove ‘modular ratio / interlinear spacing’ from the dataset.

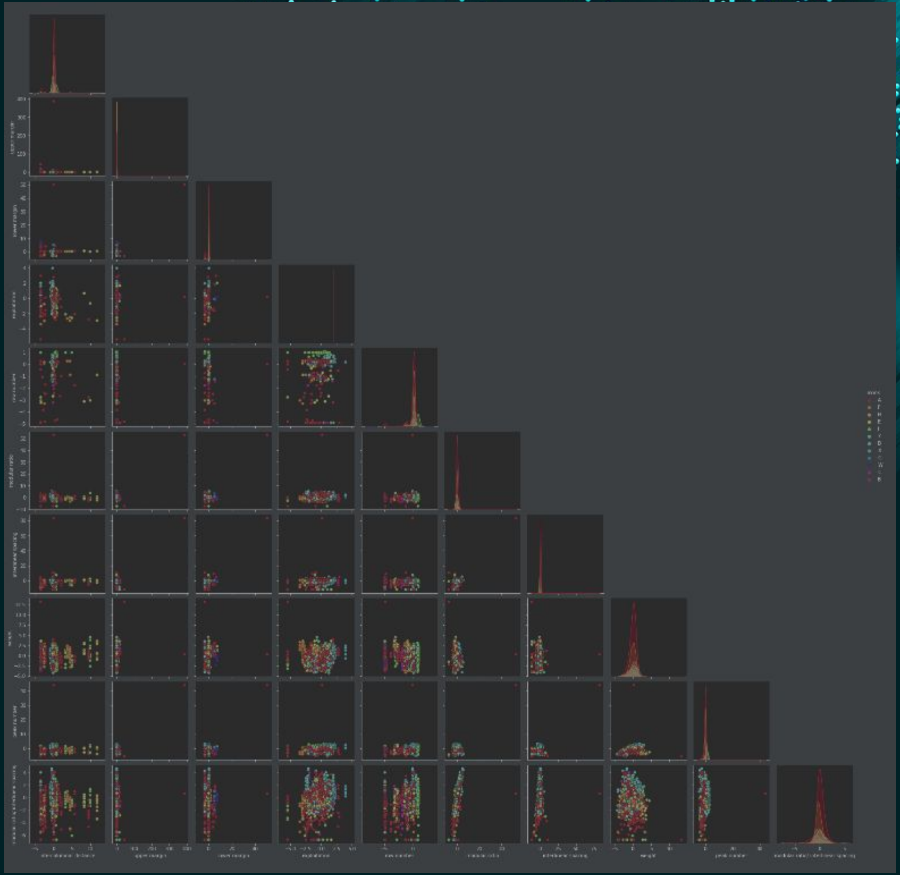


	intercolumnar distance	upper margin	lower margin	exploitation	row number	modular ratio	interlinear spacing	weight	peak number	modular ratio/ interlinear spacing
intercolumnar distance	1	-0.038	0.034	-0.065	0.45	-0.045	-0.0012	-0.043	0.059	-0.0028
upper margin	-0.038	1	0.41	0.002	-0.068	0.42	0.58	0.041	0.35	-0.025
lower margin	0.034	0.41	1	0.14	0.011	0.13	0.29	0.035	0.19	-0.045
exploitation	-0.065	0.002	0.14	1	0.096	0.26	0.044	0.34	0.28	0.32
row number	0.45	-0.068	0.011	0.096	1	0.04	0.026	-0.071	0.27	0.15
modular ratio	-0.045	0.42	0.13	0.26	0.04	1	0.46	-0.06	0.21	0.76
interlinear spacing	-0.0012	0.58	0.29	0.044	0.026	0.46	1	0.0024	0.27	0.26
weight	-0.043	0.041	0.035	0.34	-0.071	-0.06	0.0024	1	0.49	-0.016
peak number	0.059	0.35	0.19	0.28	0.27	0.21	0.27	0.49	1	0.19
modular ratio/ interlinear spacing	-0.0028	-0.025	-0.045	0.32	0.15	0.76	0.26	-0.016	0.19	1

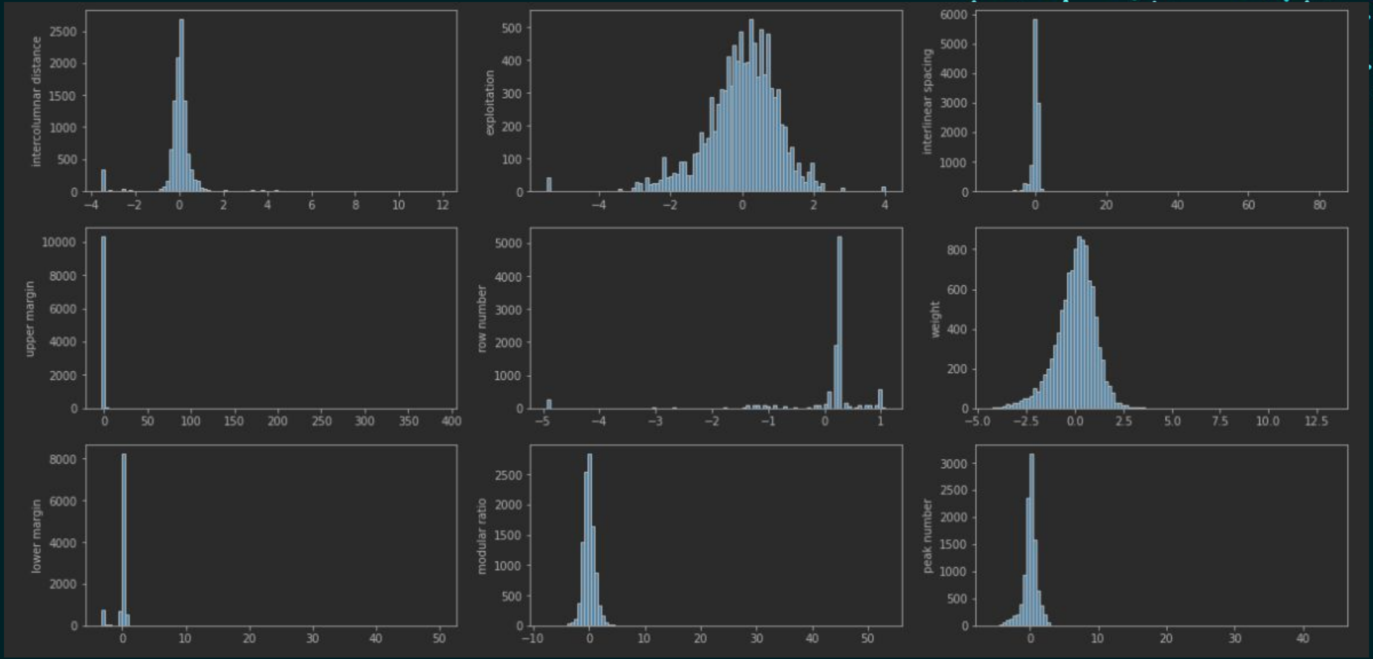
PAIR PLOT

Before deleting the attribute of the previous slide, I made pair plot between all attributes to observe in another way the correlation between attributes.

These pair plots confirm the correlation between our two attributes and therefore allow us to validate the deletion of the column ‘modular ratio / interlinear spacing’.

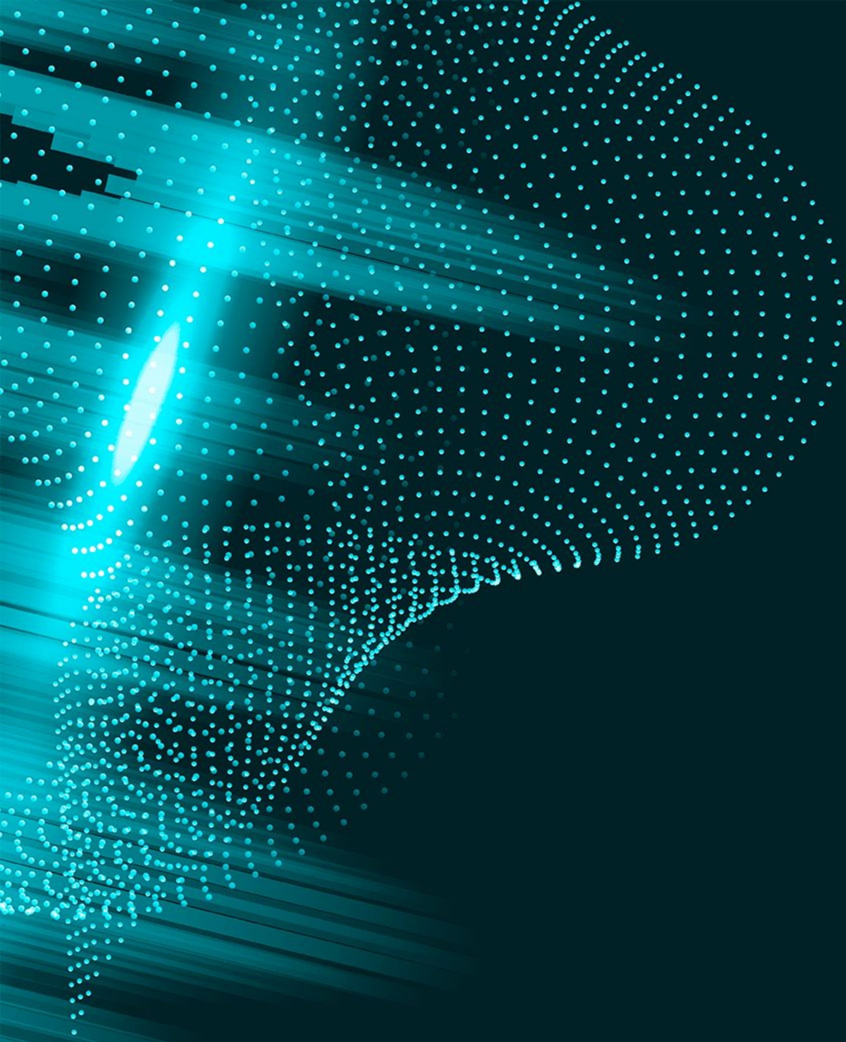


ATTRIBUTES DISTRIBUTION



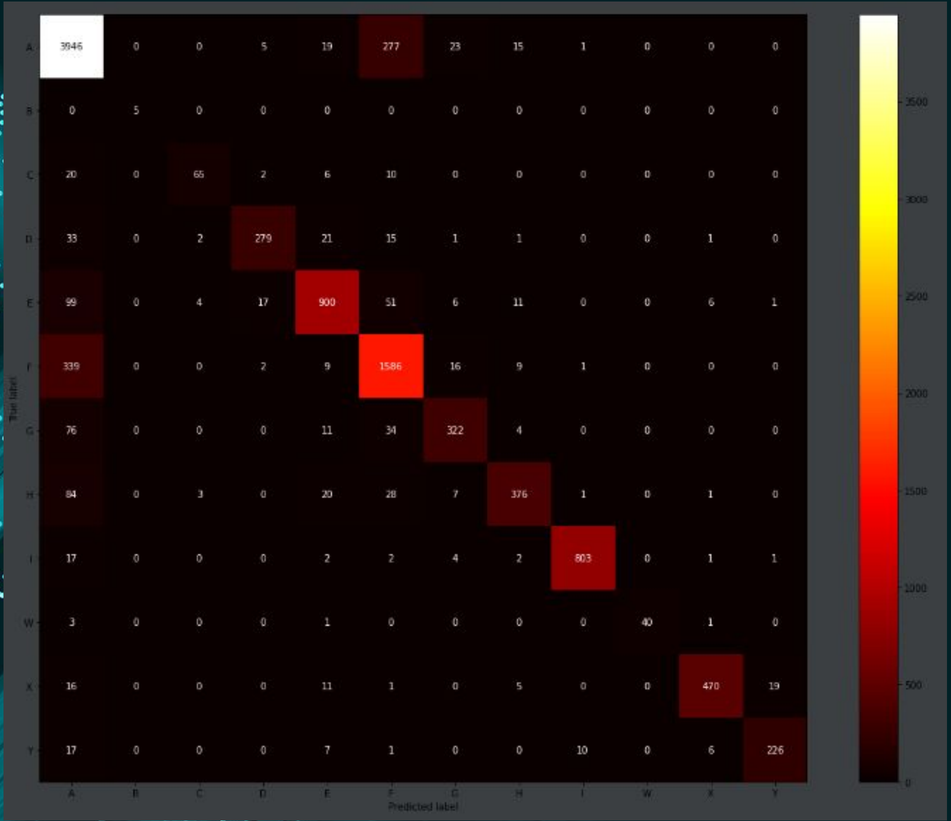
Thanks to the observation of the distribution of the attributes, we observe some extreme values for a majority of the attributes.

As for example ‘upper margin’ which has a graph that goes up to 400 whereas almost all the data are close to 0



MODELING

K-NN



	A	B	C	D	E	F	G	H	I	W	X	Y
A	3946	0	0	5	19	277	23	15	1	0	0	0
B	0	5	0	0	0	0	0	0	0	0	0	0
C	20	0	65	2	6	10	0	0	0	0	0	0
D	33	0	2	279	21	15	1	1	0	0	1	0
E	99	0	4	17	900	51	6	11	0	0	6	1
F	339	0	0	2	9	1586	16	9	1	0	0	0
G	76	0	0	0	11	34	322	4	0	0	0	0
H	84	0	3	0	20	28	7	376	1	0	1	0
I	17	0	0	0	2	2	4	2	803	0	1	1
W	3	0	0	0	1	0	0	0	0	40	1	0
X	16	0	0	0	11	1	0	5	0	0	470	19
Y	17	0	0	0	7	1	0	0	10	0	6	226

Confusion Matrix

Best Score : 0.837967

Works better with :

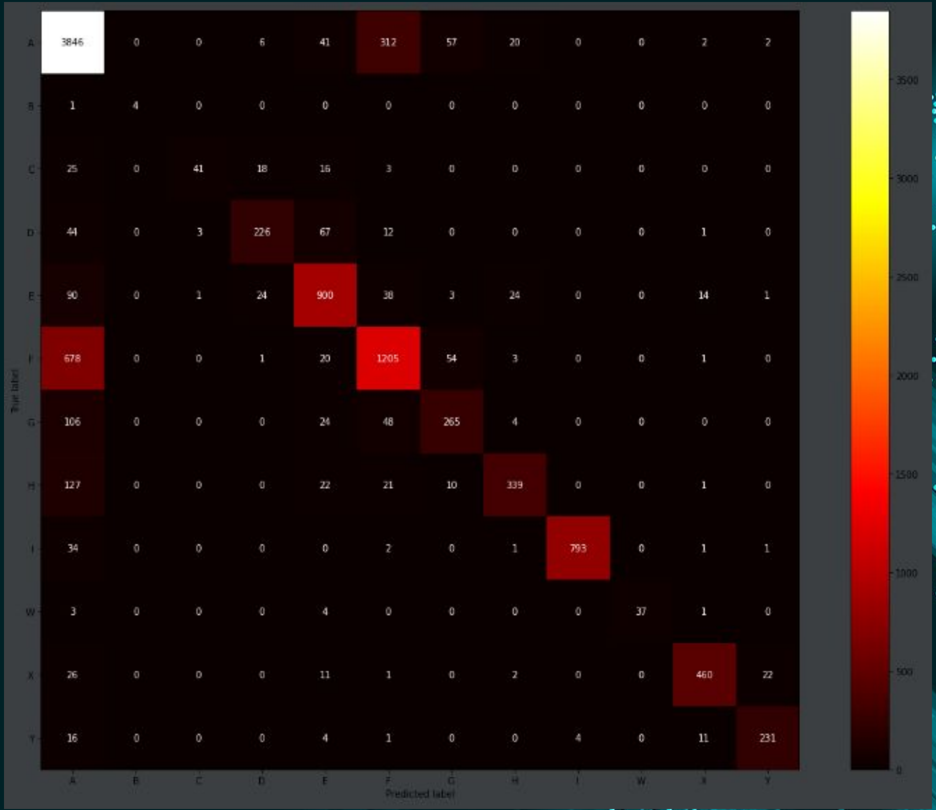
- leaf_size = 15
- n_neighbors = 5
- p = 1

SVC

Best Score : 0.788591

Works better with :

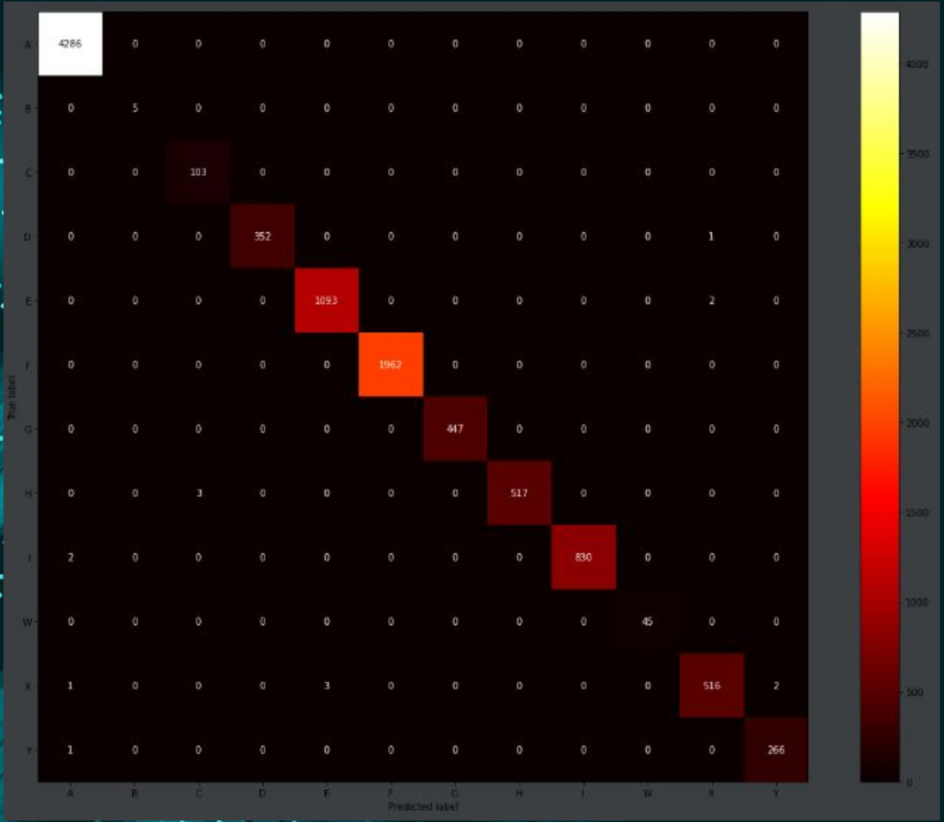
- C = 2
- gamma = 0.5
- kernel = rbf



	A	B	C	D	E	F	G	H	I	W	X	Y
A	3846	0	0	6	41	312	57	20	0	0	2	2
B	1	4	0	0	0	0	0	0	0	0	0	0
C	25	0	41	18	16	3	0	0	0	0	0	0
D	44	0	3	226	67	12	0	0	0	0	1	0
E	90	0	1	24	900	38	3	24	0	0	14	1
F	678	0	0	1	20	1205	54	3	0	0	1	0
G	106	0	0	0	24	48	265	4	0	0	0	0
H	127	0	0	0	22	21	10	339	0	0	1	0
I	34	0	0	0	0	2	0	1	793	0	1	1
W	3	0	0	0	4	0	0	0	0	37	1	0
X	26	0	0	0	11	1	0	2	0	0	460	22
Y	16	0	0	0	4	1	0	0	4	0	11	231

Confusion Matrix

XGBOOST



	A	B	C	D	E	F	G	H	I	W	X	Y
A	4286	0	0	0	0	0	0	0	0	0	0	0
B	0	5	0	0	0	0	0	0	0	0	0	0
C	0	0	103	0	0	0	0	0	0	0	0	0
D	0	0	0	352	0	0	0	0	0	0	1	0
E	0	0	0	0	1093	0	0	0	0	0	2	0
F	0	0	0	0	0	1962	0	0	0	0	0	0
G	0	0	0	0	0	0	447	0	0	0	0	0
H	0	0	3	0	0	0	0	517	0	0	0	0
I	2	0	0	0	0	0	0	0	830	0	0	0
W	0	0	0	0	0	0	0	0	0	45	0	0
X	1	0	0	0	3	0	0	0	0	0	516	2
Y	1	0	0	0	0	0	0	0	0	0	0	266

Confusion Matrix

Best Score : 0.996357

Works better with :

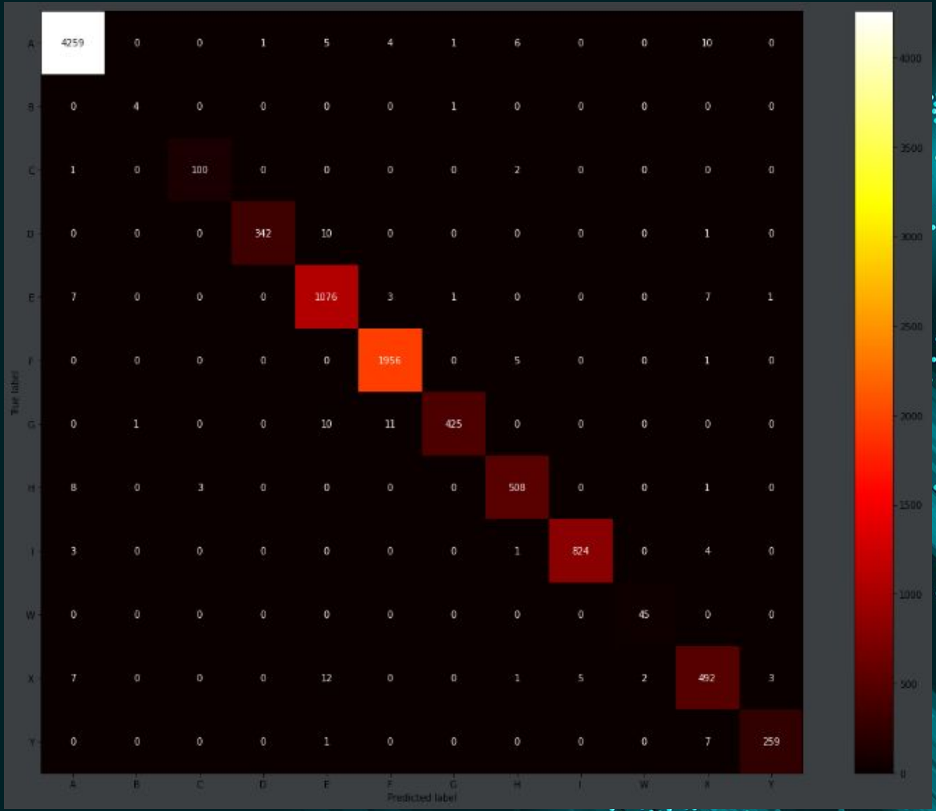
- learning_rate = 0.05
- n_estimators = 1500

DECISION TREE

Best Score : 0.982933

Works better with :

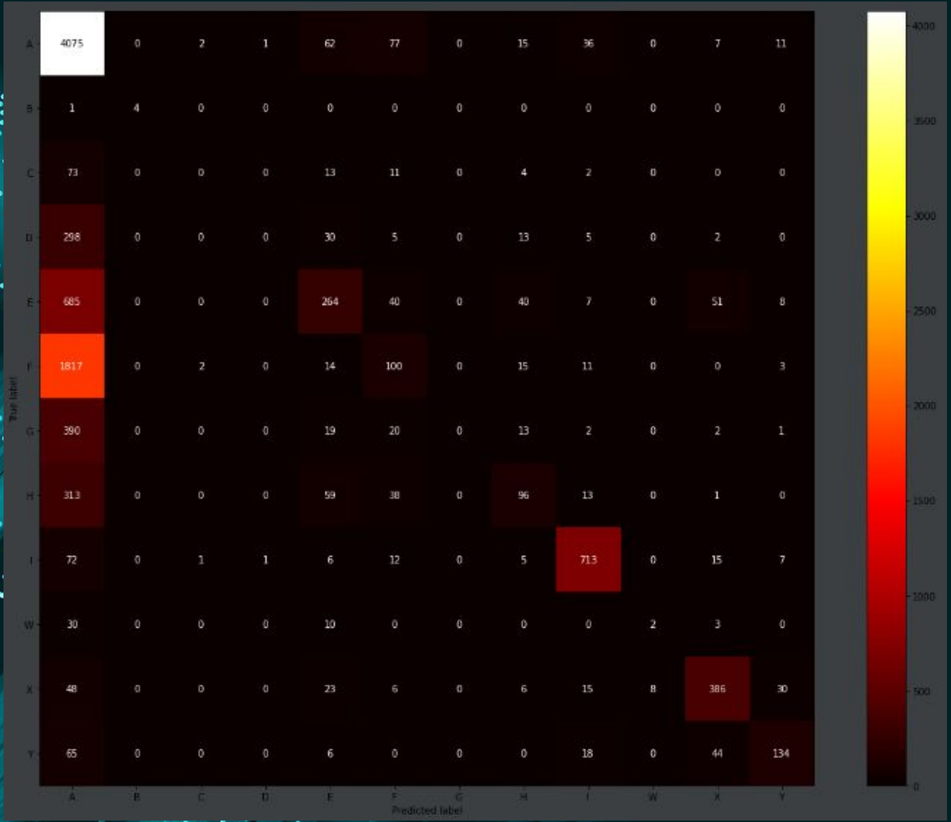
- criterion = entropy
- max_features = 9



	A	B	C	D	E	F	G	H	I	W	X	Y
A	4259	0	0	1	5	4	1	6	0	0	10	0
B	0	4	0	0	0	0	1	0	0	0	0	0
C	1	0	100	0	0	0	0	2	0	0	0	0
D	0	0	0	342	10	0	0	0	0	0	1	0
E	7	0	0	0	1076	3	1	0	0	0	7	1
F	0	0	0	0	0	1956	0	5	0	0	1	0
G	0	1	0	0	10	11	425	0	0	0	0	0
H	8	0	3	0	0	0	0	508	0	0	1	0
I	3	0	0	0	0	0	0	1	824	0	4	0
W	0	0	0	0	0	0	0	0	0	45	0	0
X	7	0	0	0	12	0	0	1	5	2	492	3
Y	0	0	0	0	1	0	0	0	0	0	7	259

Confusion Matrix

LOGISTIC REGRESSION



	A	B	C	D	E	F	G	H	I	W	X	Y
A	4075	0	2	1	62	77	0	15	36	0	7	11
B	1	4	0	0	0	0	0	0	0	0	0	0
C	73	0	0	0	13	11	0	4	2	0	0	0
D	298	0	0	0	30	5	0	13	5	0	2	0
E	685	0	0	0	264	40	0	40	7	0	51	8
F	1817	0	2	0	14	100	0	15	11	0	0	3
G	390	0	0	0	19	20	0	13	2	0	2	1
H	313	0	0	0	59	38	0	96	13	0	1	0
I	72	0	1	1	6	12	0	5	713	0	15	7
W	30	0	0	0	10	0	0	0	0	2	3	0
X	48	0	0	0	23	6	0	6	15	8	386	30
Y	65	0	0	0	6	0	0	0	18	0	44	134

Confusion Matrix

Best Score : 0.552733

Works better with :

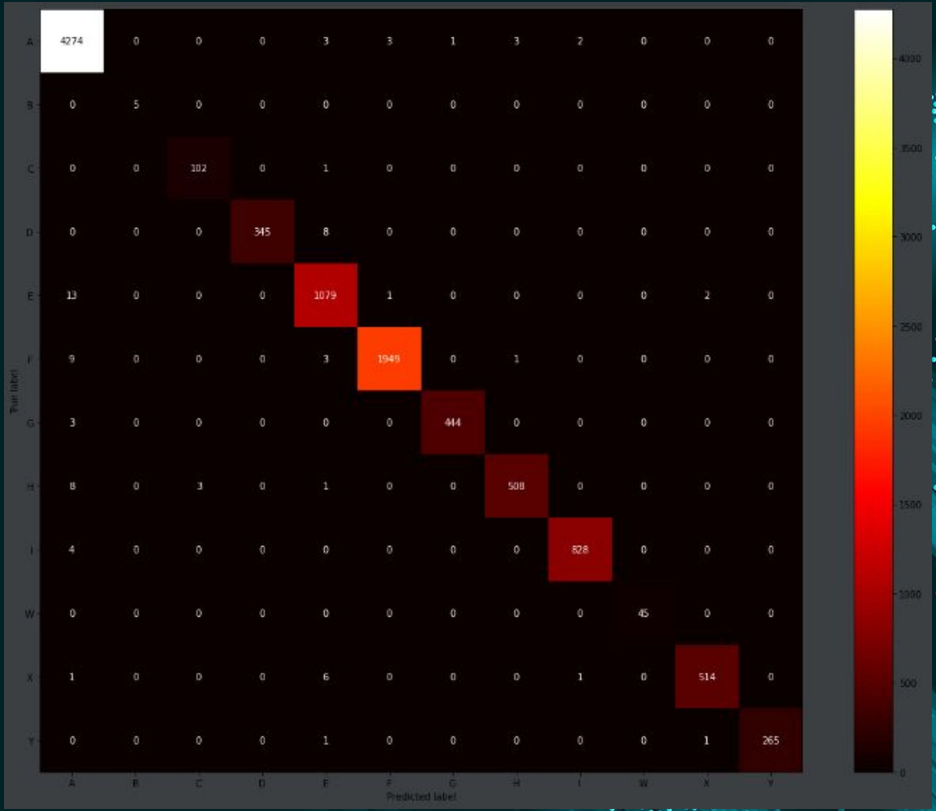
- max_iter = 500
- nb_jobs = 2

RANDOM FOREST

Best Score : 0.988878

Works better with :

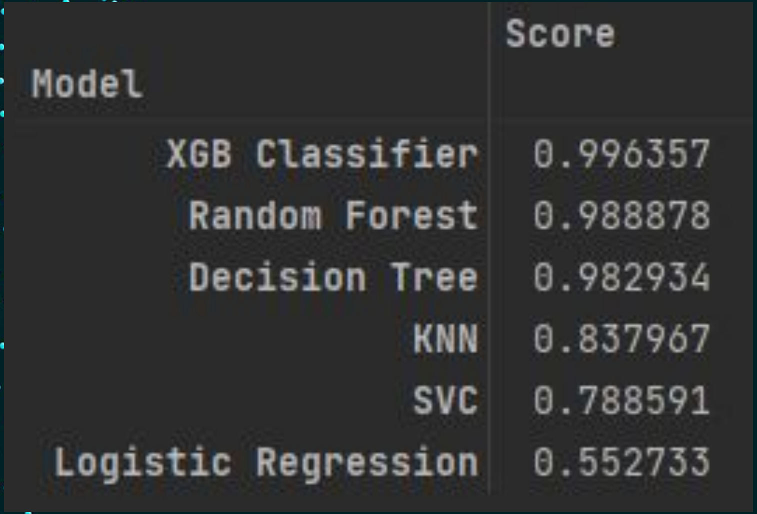
- min_samples_split = 2
- n_estimators = 150



	A	B	C	D	E	F	G	H	I	W	X	Y
A	4274	0	0	0	3	3	1	3	2	0	0	0
B	0	5	0	0	0	0	0	0	0	0	0	0
C	0	0	102	0	1	0	0	0	0	0	0	0
D	0	0	0	345	8	0	0	0	0	0	0	0
E	13	0	0	0	1079	1	0	0	0	0	2	0
F	9	0	0	0	3	1949	0	1	0	0	0	0
G	3	0	0	0	0	0	444	0	0	0	0	0
H	8	0	3	0	1	0	0	508	0	0	0	0
I	4	0	0	0	0	0	0	0	828	0	0	0
W	0	0	0	0	0	0	0	0	0	45	0	0
X	1	0	0	0	6	0	0	0	1	0	514	0
Y	0	0	0	0	1	0	0	0	0	0	1	265

Confusion Matrix

COMPARISON



Model	Score
XGB Classifier	0.996357
Random Forest	0.988878
Decision Tree	0.982934
KNN	0.837967
SVC	0.788591
Logistic Regression	0.552733

- XGB Classifier seems to be the best model but Random Forest & Decision Tree algorithms are also very accurate.
- KNN & SVC still have pretty good results.
- Logistic Regression has an accuracy of a little over 50% which is not very good.

CONCLUSION

CONCLUSION

This project allowed me to better understand how to work on a data set, how to exploit it and predict its data by comparing several models.

This project, as a whole, went well and I am satisfied with the results I obtained during my predictions.

My only regret is that I didn't have the time to make an API for my project due to health problems (COVID).