PYTHON FOR DATA ANALYSIS

Study on

AVILA BIBLE

RAZZOUK Amine
OUARAS Yanis

Presentation of the Avila Problem

An XII century old problem…

Presentation of the Avila Dataset

- The Avila data set has been extracted from 800 images of the the "Avila Bible", a giant Latin copy of the whole Bible produced during the XII century between Italy and Spain.

- The palaeographic analysis of the manuscript has individuated the presence of 12 copyists. The pages written by each copyist are not equally numerous. Each pattern contains 10 features and corresponds to a group of 4 consecutive rows.

- The prediction task consists in associating each pattern to one of the 12 copyists (labeled as: A, B, C, D, E, F, G, H, I, W, X, Y). The data have has been normalized, by using the Z-normalization method, and divided in two data sets: a training set containing 10430 samples, and a test set containing the 10437 samples.

Data-Visualization

An XII century old problem…

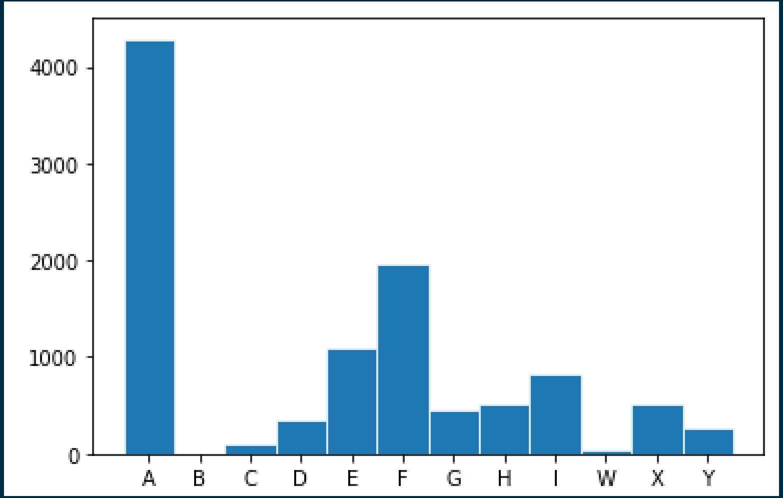
Data-Vizualisation

- As said earlier, our datasets are composed of 10 different features that each represent a particularity of a copyist.

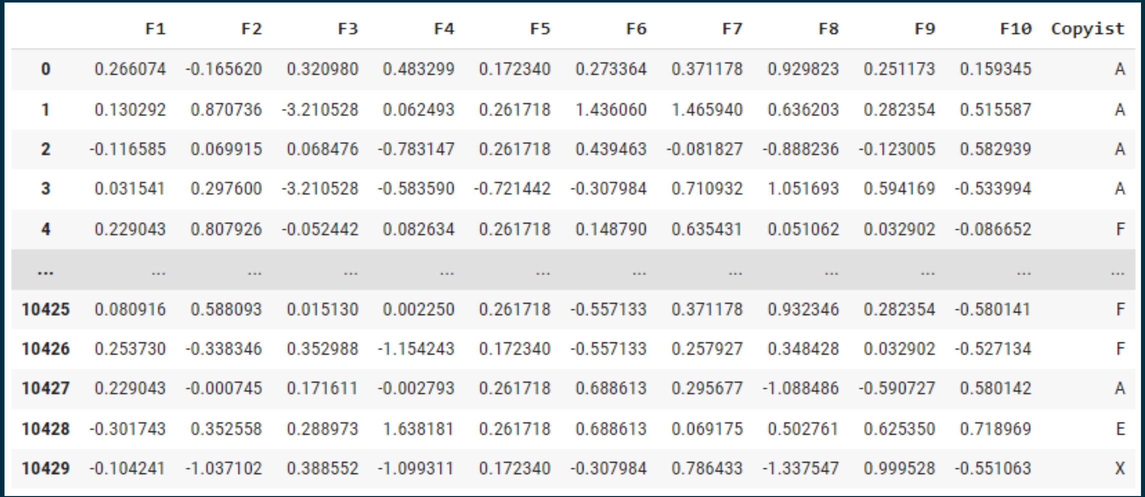
- The different features are :

F1 : Intercolumnar distance
F2 : upper margin
F3 : lower margin
F4 : exploitatoin
F5 : row number
F6 : modular ratio
F7 : interlinear spacing
F8 : weight
F9 : peak number
F10 : modular ratio/interlinear spacing

Distribution of samples by copyists



Composition of the training dataset

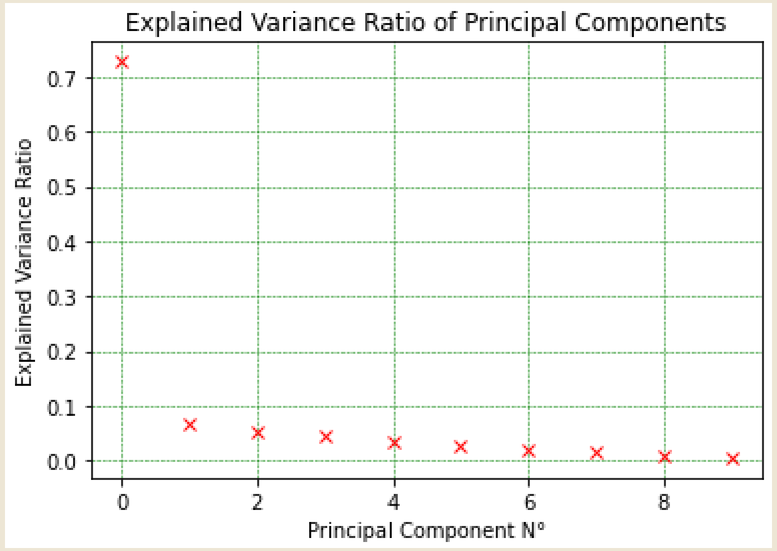


	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Copyist
0	0.266074	-0.165620	0.320980	0.483299	0.172340	0.273364	0.371178	0.929823	0.251173	0.159345	A
1	0.130292	0.870736	-3.210528	0.062493	0.261718	1.436060	1.465940	0.636203	0.282354	0.515587	A
2	-0.116585	0.069915	0.068476	-0.783147	0.261718	0.439463	-0.081827	-0.888236	-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
...	...	...	...	...	...	...	...	...	...	...	...
10425	0.080916	0.588093	0.015130	0.002250	0.261718	-0.557133	0.371178	0.932346	0.282354	-0.580141	F
10426	0.253730	-0.338346	0.352988	-1.154243	0.172340	-0.557133	0.257927	0.348428	0.032902	-0.527134	F
10427	0.229043	-0.000745	0.171611	-0.002793	0.261718	0.688613	0.295677	-1.088486	-0.590727	0.580142	A
10428	-0.301743	0.352558	0.288973	1.638181	0.261718	0.688613	0.069175	0.502761	0.625350	0.718969	E
10429	-0.104241	-1.037102	0.388552	-1.099311	0.172340	-0.307984	0.786433	-1.337547	0.999528	-0.551063	X

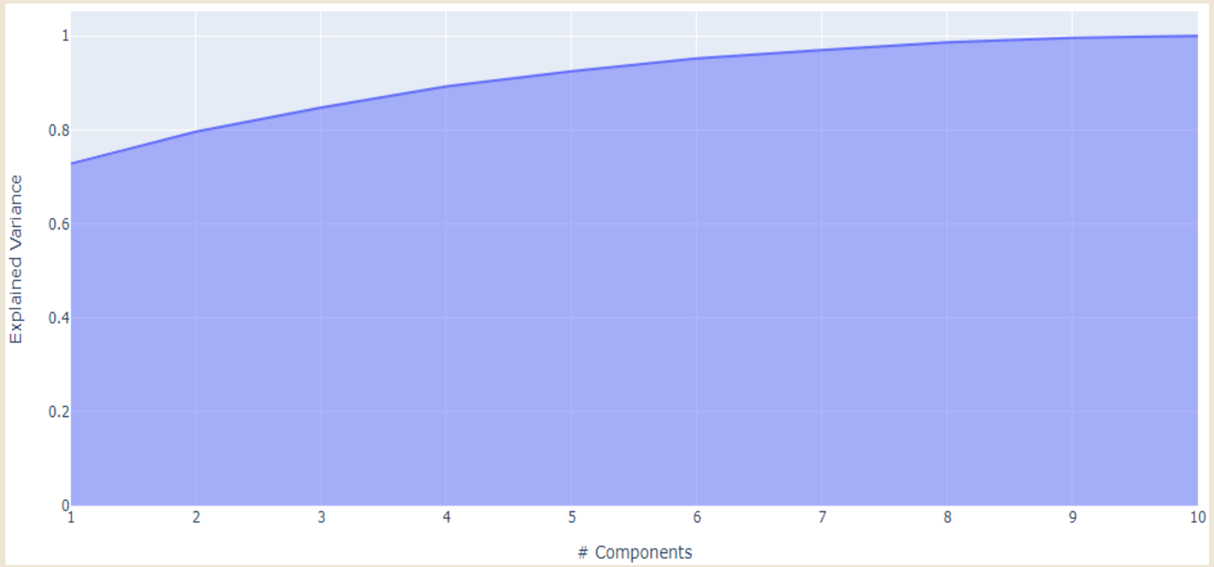
PCA & Link between the features

An XII century old problem…

PCA & Link between the features



Explained variance ratio of the first 10 components

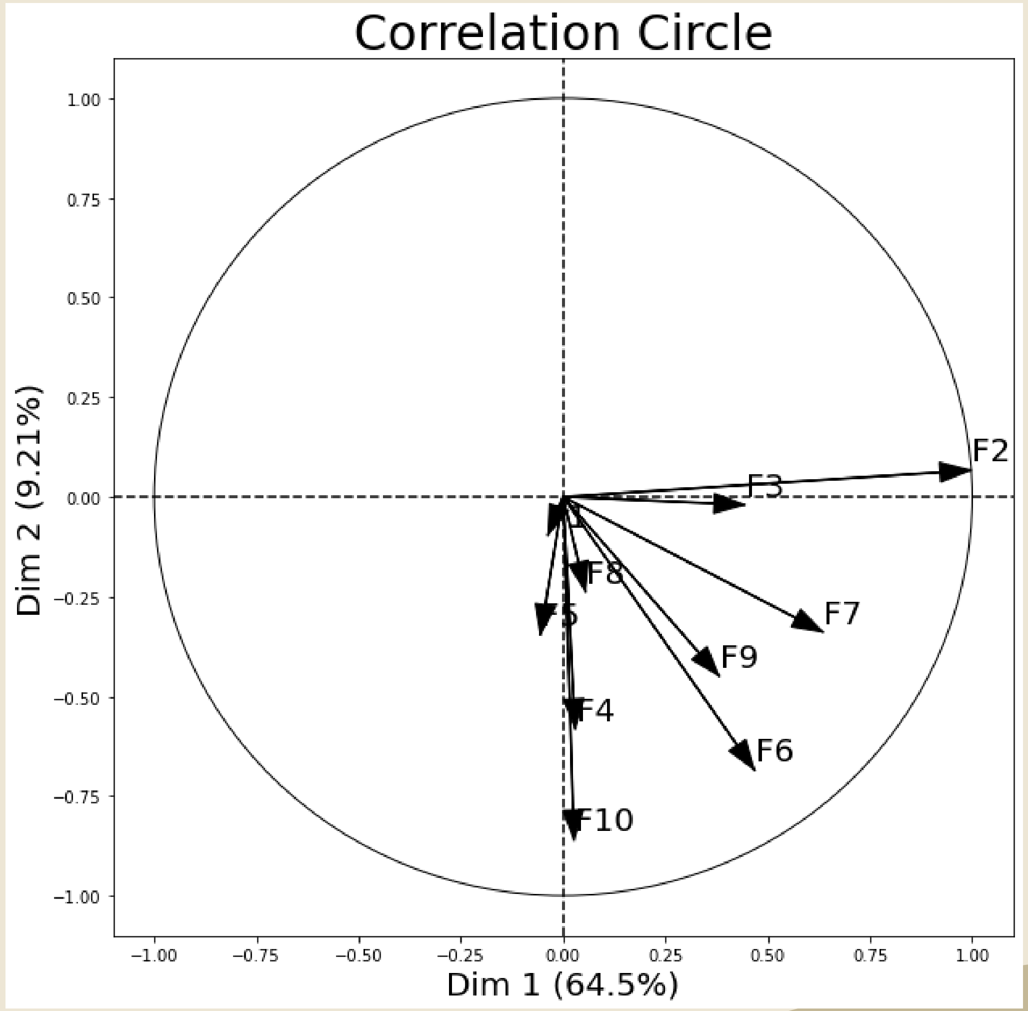


Cumulative Explained variance ratio of the first 10 components

- PCA makes maximum variability in the dataset more visible by rotating the axes. PCA identifies a list of the principal axes to describe the underlying dataset before ranking them according to the amount of variance captured by each.

- On our dataset, nearly 80% of the total variance is explained only by using the 2nd components of the PCA.

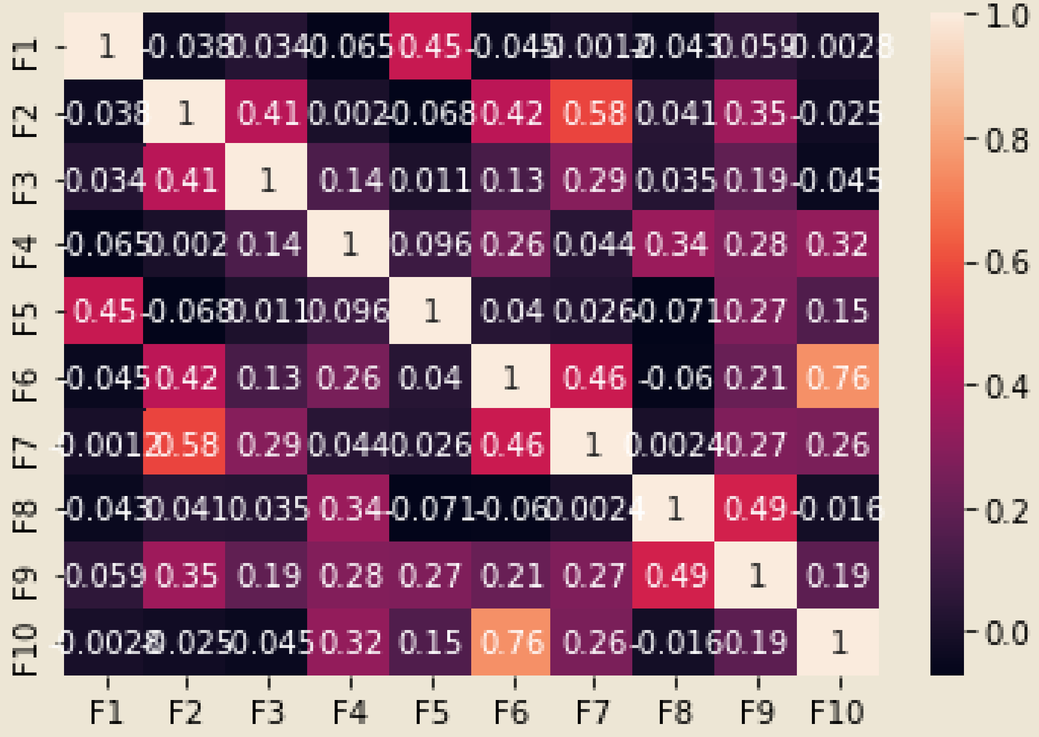
PCA & Link between the features



- Correlation circle help us to visualize the correlations between the original dataset features. The principal components are shown via coordinates.
- The correlation circle shows that some features are very correlated like F10 and F6 or F7 and F6, but some aren't correlated at all like F10 and F2 or F4 and F3.
- And this was predictable, because, for instance, F6 and F7 are representing the modular ratio and interlinear spacing of the copyist whereas F10 is a feature composed of the modular ratio over the interlinear spacing, so F10 is closely related to both features because it is directly composed by them.

PCA & Link between the features

Correlation Matrix

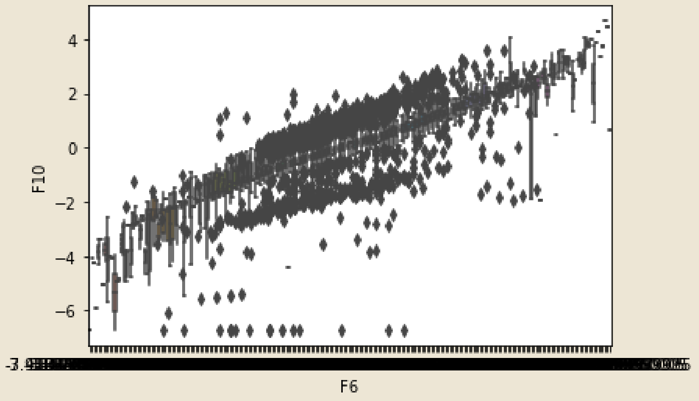


	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
F1	1	-0.038	-0.034	-0.065	0.45	-0.045	-0.0012	-0.043	-0.059	-0.0028
F2	-0.038	1	0.41	0.002	-0.068	0.42	0.58	0.041	0.35	-0.025
F3	-0.034	0.41	1	0.14	0.011	0.13	0.29	0.035	0.19	-0.045
F4	-0.065	0.002	0.14	1	0.096	0.26	0.044	0.34	0.28	0.32
F5	0.45	-0.068	0.011	0.096	1	0.04	0.026	-0.071	0.27	0.15
F6	-0.045	0.42	0.13	0.26	0.04	1	0.46	-0.06	0.21	0.76
F7	-0.0012	0.58	0.29	0.044	0.026	0.46	1	0.0024	0.27	0.26
F8	-0.043	0.041	0.035	0.34	-0.071	-0.06	0.0024	1	0.49	-0.016
F9	-0.059	0.35	0.19	0.28	0.27	0.21	0.27	0.49	1	0.19
F10	-0.0028	-0.025	-0.045	0.32	0.15	0.76	0.26	-0.016	0.19	1

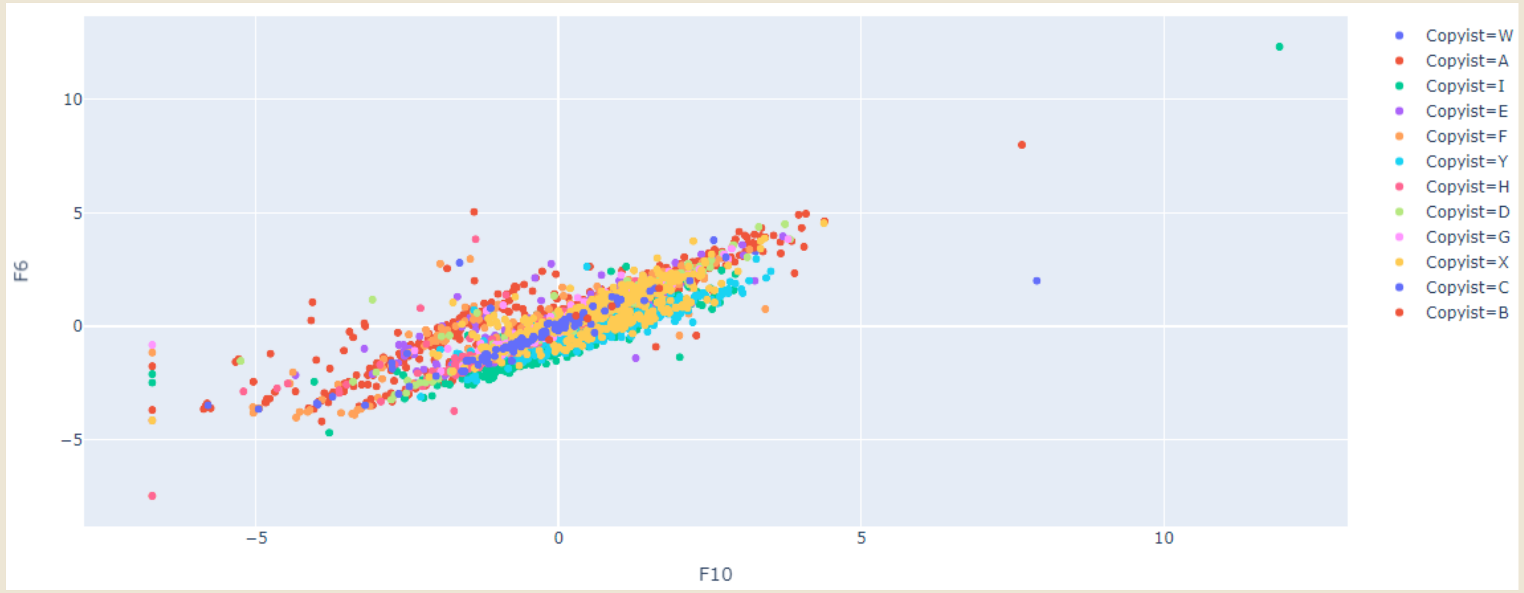
- With the correlation matrix of the two datasets, our earlier assumptions are now settled.
- F10, F6 and F7 are strongly correlated. But we discover other surprising correlations like between F5 and F1 which correspond to row number and intercolumnar distance. Or between F2 and F7 which correspond to upper margin and interlinear spacing.

PCA & Link between the features

- To illustrate the correlation between F6 and F10, we can use the following plots.

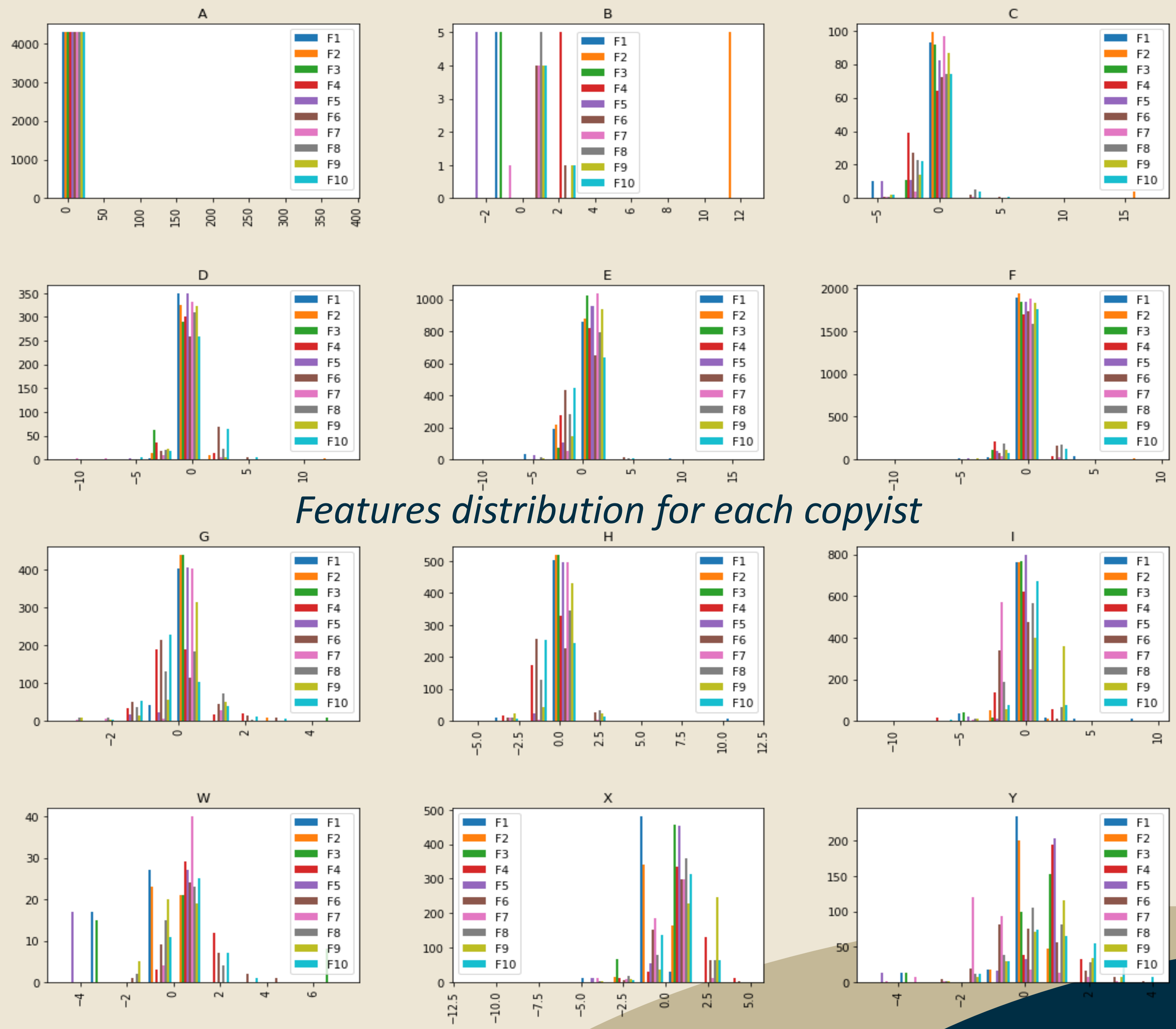


Seaborn plot of F10 correlation with F6



Plotly Express Scatter of the linear correlation between F6 and F10 features

PCA & Link between the features



Features distribution for each copyist

- Here are the features distribution for each copyist.
- We can see that some copyists have featured more and more normally scattered.
- But some behave very unpredictably : accordingly, these are the copyist with the least train samples (n.b. the histogram of 2.2 section)

Prediction

An XII century old problem…

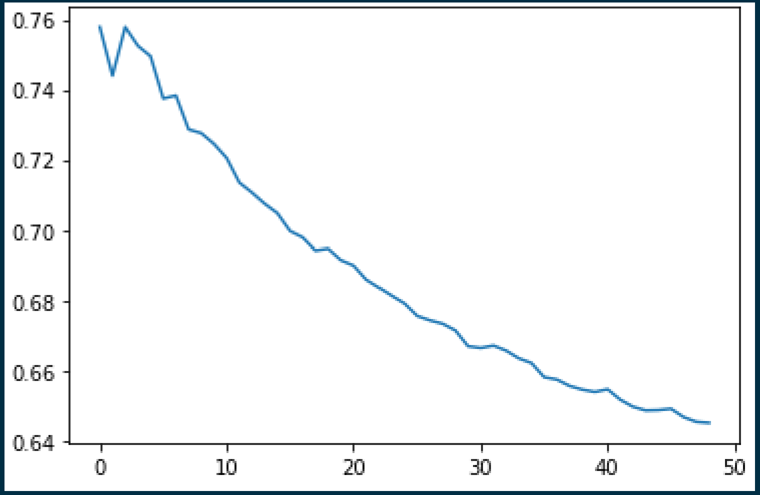
Prediction

- Data have already been Z-normalized, learning algorithms can also benefit from scaling the data.

- We began prediction using multiple models.

KNeighbors gives roughly a 74% accuracy.

Our most efficient hyperparameters with GridSearch method where (n_neighbors = 4, weights = "distance", metric = "euclidean")



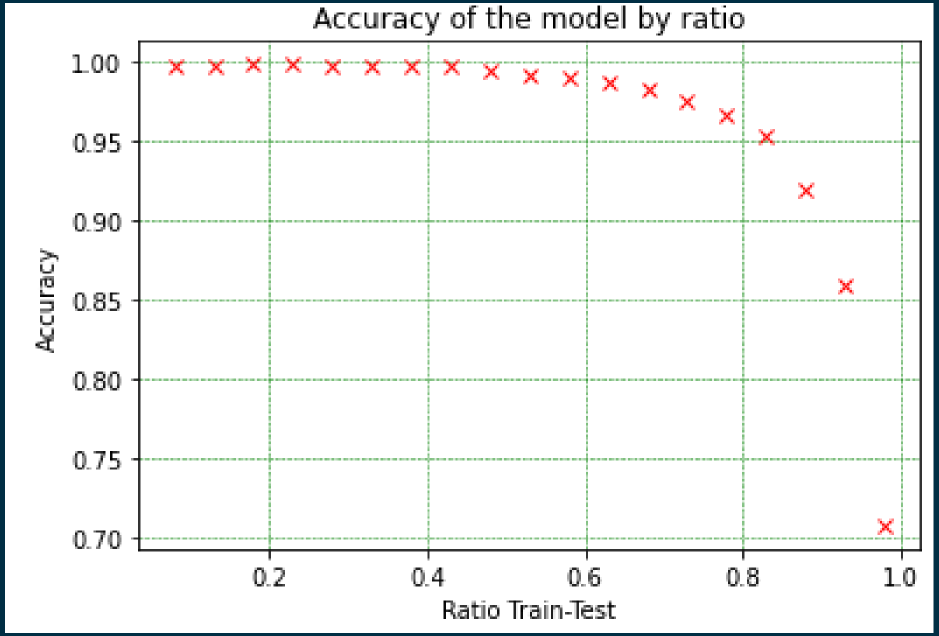
Kneighbors accuracy by number of neighbors

Prediction

RandomForest gives roughly a 98,02% accuracy.

Our most efficient hyperparameters with GridSearch method where (criterion='entropy', n_estimators=500, random_state=42)

Accuracy of RandomForest model by test/train ratio



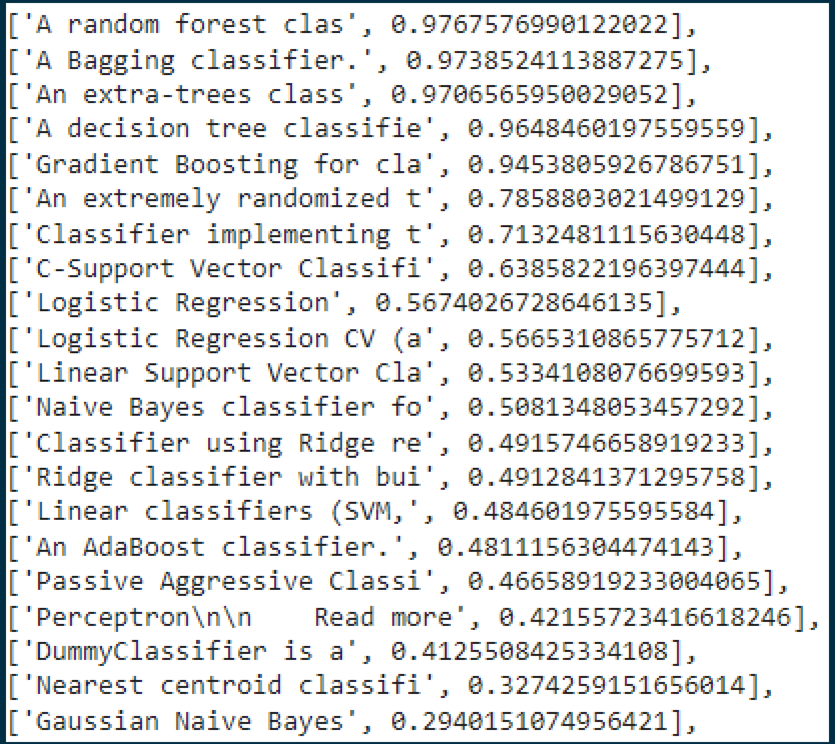
We can see that the test/train ratio is stable even approaching the 70%. But we can also conclude that a ratio of 33% test/train gives us a pretty good accuracy, which is optimal for our case.

Prediction

We needed to find the best sklearn algorithms for our dataset. So, we tested all the classifier models of the library and ranked them by their score on our dataset.

It cames out that the RandomForest is still the most accurate of the tested algorithms for our dataset, but we need to tune and test the Bagging and the GradientBoosting Classifiers with the best hyperparameters possible, as they were part of the five best algorithms tested.

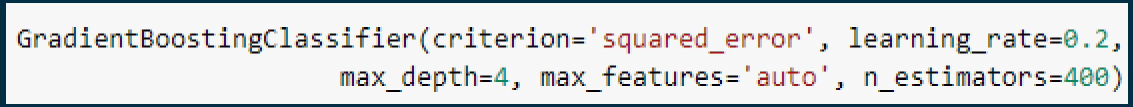
A list of the most accurate classifier models of the sklearn library on our dataset with their respective accuracy



```
['A random forest clas', 0.9767576990122022],
['A Bagging classifier.', 0.9738524113887275],
['An extra-trees class', 0.9706565950029052],
['A decision tree classifie', 0.9648460197559559],
['Gradient Boosting for cla', 0.9453805926786751],
['An extremely randomized t', 0.7858803021499129],
['Classifier implementing t', 0.7132481115630448],
['C-Support Vector Classifi', 0.6385822196397444],
['Logistic Regression', 0.5674026728646135],
['Logistic Regression CV (a', 0.5665310865775712],
['Linear Support Vector Cla', 0.5334108076699593],
['Naive Bayes classifier fo', 0.5081348053457292],
['Classifier using Ridge re', 0.4915746658919233],
['Ridge classifier with bui', 0.4912841371295758],
['Linear classifiers (SVM,', 0.484601975595584],
['An AdaBoost classifier.', 0.4811156304474143],
['Passive Aggressive Classi', 0.4665891923004065],
['Perceptron\n\n    Read more', 0.4215572341661 8246],
['DummyClassifier is a', 0.4125508425334108],
['Nearest centroid classifi', 0.327425915165601 4],
['Gaussian Naive Bayes', 0.2940151074956421],
```

Prediction

- BaggingClassifier gives us roughly a 97,99% accuracy.
- BaggingClassifier with the best hyperparameters possible were still less accurate than our modified RandomForestClassifier model.
- We needed to find the best hyperparameters for the GradientBoosting model, so we used again the GridSearch method to find the most efficient hyperparameters.

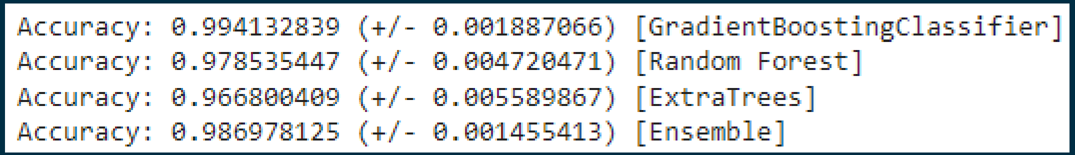


```
GradientBoostingClassifier(criterion='squared_error', learning_rate=0.2,
                           max_depth=4, max_features='auto', n_estimators=400)
```

- It cames out that the most efficient hyperparameters for the GradientBoosting model, in our case, were the hyperparameters above. At this point, we achieved an average accuracy of 99.44% with this model, which is greater than our previous RandomForest model.

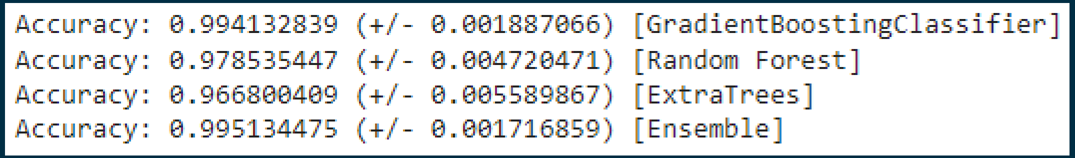
Prediction

Voting classifier with our three most accurate modified model, first with a hard voting :



```
Accuracy: 0.994132839 (+/- 0.001887066) [GradientBoostingClassifier]
Accuracy: 0.978535447 (+/- 0.004720471) [Random Forest]
Accuracy: 0.966800409 (+/- 0.005589867) [ExtraTrees]
Accuracy: 0.986978125 (+/- 0.001455413) [Ensemble]
```

And with a soft voting method :



```
Accuracy: 0.994132839 (+/- 0.001887066) [GradientBoostingClassifier]
Accuracy: 0.978535447 (+/- 0.004720471) [Random Forest]
Accuracy: 0.966800409 (+/- 0.005589867) [ExtraTrees]
Accuracy: 0.995134475 (+/- 0.001716859) [Ensemble]
```

We have now achieved an accuracy of 99.51% with our soft voting composed-base model using modified GradientBoosting RandomForest and ExtraTrees model. This was the best accuracy we were able to achieve and a nice trip into this machine learning branch.

Testing API

- Regarding the API, we had configuration problems and we were not able to go further at this level.
- We tried to confuse him on googlecollab using 'flask-ngrok' but also on Spyder.

Our thoughts

- It was a very interstiting project as it treats text recognition of old books. We had a surprinsignly furnished dataset, and so we were able to predict with very much accuracy which of the 12 copyist is writing.
- We had the chance to use a lot of the method we used in this class. A So, a big final project like this one was the best possible exercice to test our knowledge of the “data&machine learning” domain.
- It was a pleasure to work in group as well as we were able to help each other.
- Great thematic/dataset choice

Thank You !