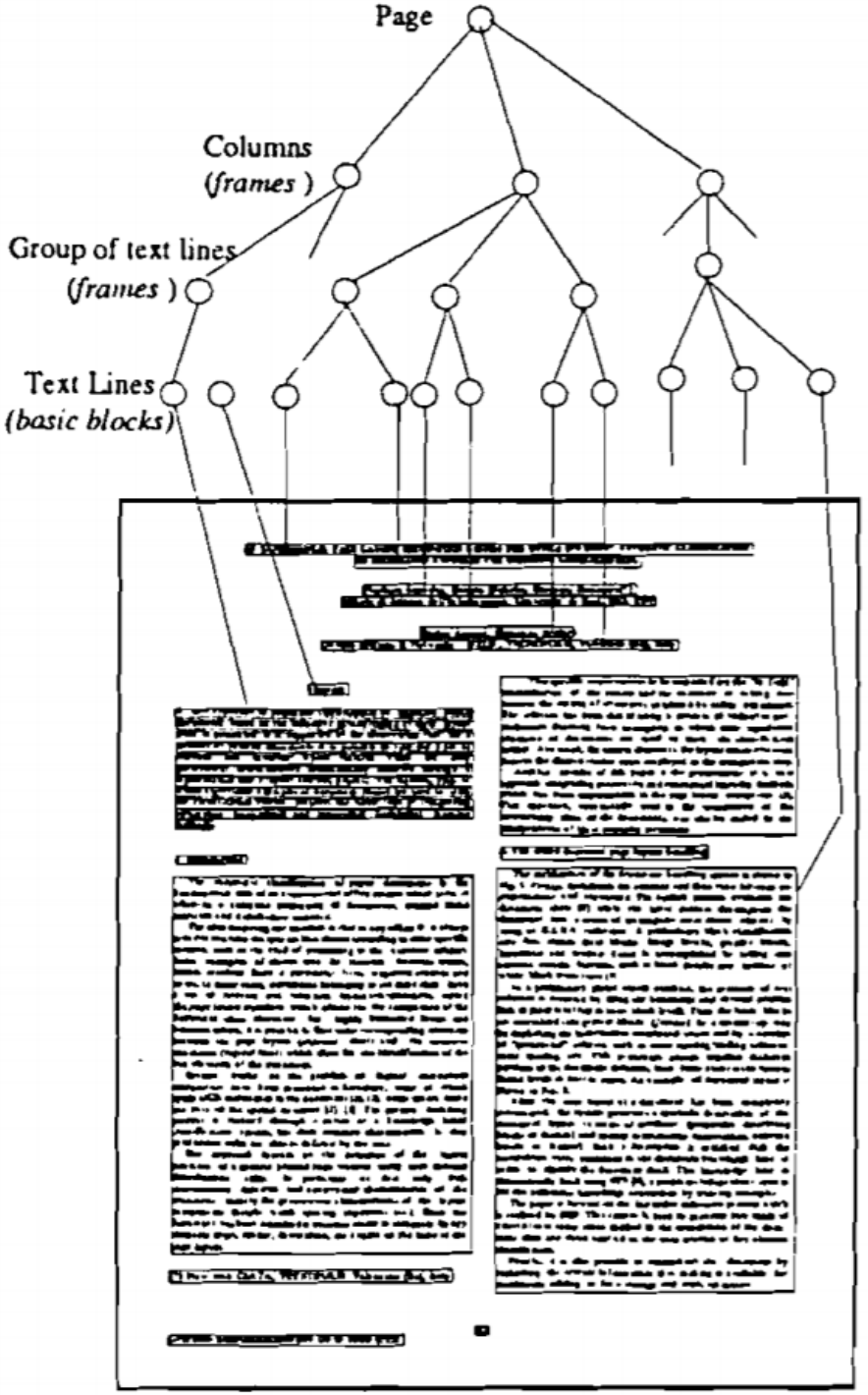
Python for Data Analysis Project

Mounir HAMMOUTI – Lies HAOUAS

ESILV – DIA3 – 2020/2021



Page block classification

- The problem consists in classifying all the blocks of the page layout of a document that has been detected by a segmentation process. This is an essential step in document analysis in order to separate text from graphic areas. Indeed, the five classes to predict are: text (1), horizontal line (2), picture (3), vertical line (4) and graphic (5).

- Link of the dataset: https://archive.ics.uci.edu/ml/datasets/Page+Blocks+Classification

The dataset

- The 10 features used to predict the page block class are :

  - height: integer. | Height of the block.
  - length: integer. | Length of the block.
  - area: integer. | Area of the block (height * lenght);
  - eccen: continuous. | Eccentricity of the block (lenght / height);
  - p_black: continuous. | Percentage of black pixels within the block (blackpix / area);
  - p_and: continuous. | Percentage of black pixels after the application of the Run Length Smoothing Algorithm (RLSA) (blackand / area);
  - mean_tr: continuous. | Mean number of white-black transitions (blackpix / wb_trans);
  - blackpix: integer. | Total number of black pixels in the original bitmap of the block.
  - blackand: integer. | Total number of black pixels in the bitmap of the block after the RLSA.
  - wb_trans: integer. | Number of white-black transitions in the original bitmap of the block.

- We have no categorical value, so there is no need to encode variables.

Data exploration

- After having downloaded and put the dataset into a pandas dataframe, we explored what the table contains and the characteristics of each feature. Here is the result of the describe() method from pandas:

	height	length	area	eccen	p_black	p_and	mean_tr	blackpix	blackand	wb_trans	label
count	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000	5473.000000
mean	10.473232	89.568244	1198.405628	13.753977	0.368642	0.785053	6.219278	365.930751	741.108167	106.662891	1.202631
std	18.960564	114.721758	4849.376950	30.703737	0.177757	0.170661	69.079021	1270.333082	1881.504302	167.308362	0.721470
min	1.000000	1.000000	7.000000	0.007000	0.052000	0.062000	1.000000	7.000000	7.000000	1.000000	1.000000
25%	7.000000	17.000000	114.000000	2.143000	0.261000	0.679000	1.610000	42.000000	95.000000	17.000000	1.000000
50%	8.000000	41.000000	322.000000	5.167000	0.337000	0.803000	2.070000	108.000000	250.000000	49.000000	1.000000
75%	10.000000	107.000000	980.000000	13.625000	0.426000	0.927000	3.000000	284.000000	718.000000	126.000000	1.000000
max	804.000000	553.000000	143993.000000	537.000000	1.000000	1.000000	4955.000000	33017.000000	46133.000000	3212.000000	5.000000

Data exploration

- Based on the result, some attributes are well distributed and some are not. For example, the height feature is not well distributed because its mean is 10 and its max value is 804. Unlike length feature which is well distributed.

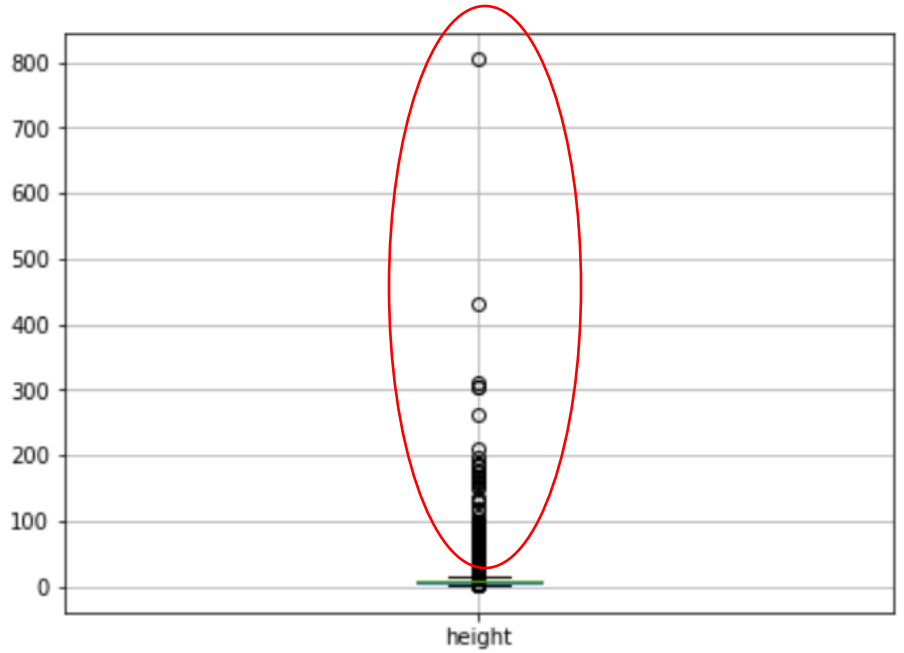
- We also counted null values in the dataset and we realized that there is not. Thus, we didn't need to process the missing values, and this made it easier to process the dataset.

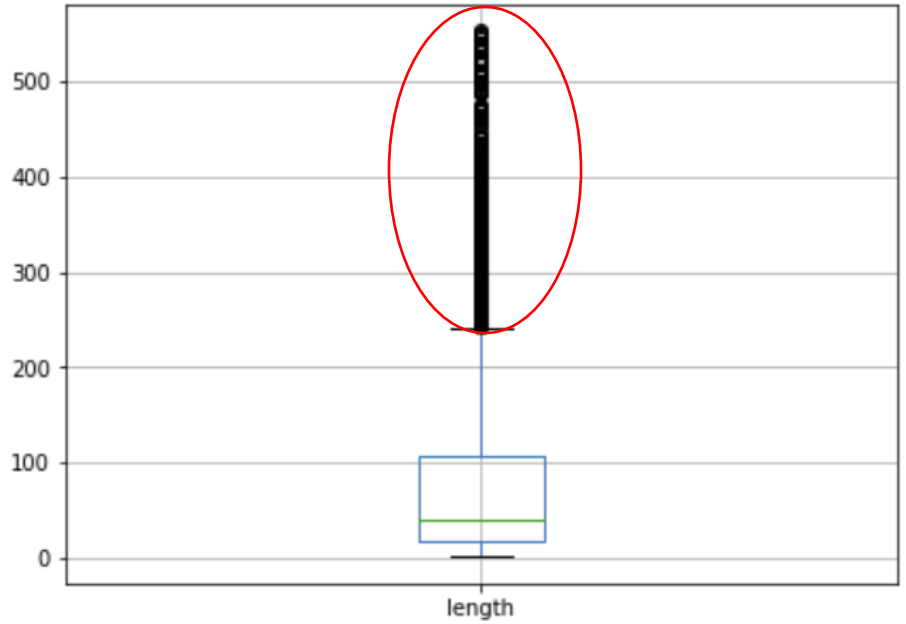
```
Null values per Column in Training Dataset:
 height      0
length       0
area         0
eccen        0
p_black      0
p_and        0
mean_tr      0
blackpix     0
blackand     0
wb_trans     0
label        0
dtype: int64
```

Data exploration

- Then we realized boxplots to vizualise the distribution of each feature. Here is the representation of height and length features:

- As we can see circled in red, the two features have many outliers. Those are data that are outside the range of what is expected and unlike the other data. Machine Learning models can be improved by understanding and removing these outliers values.



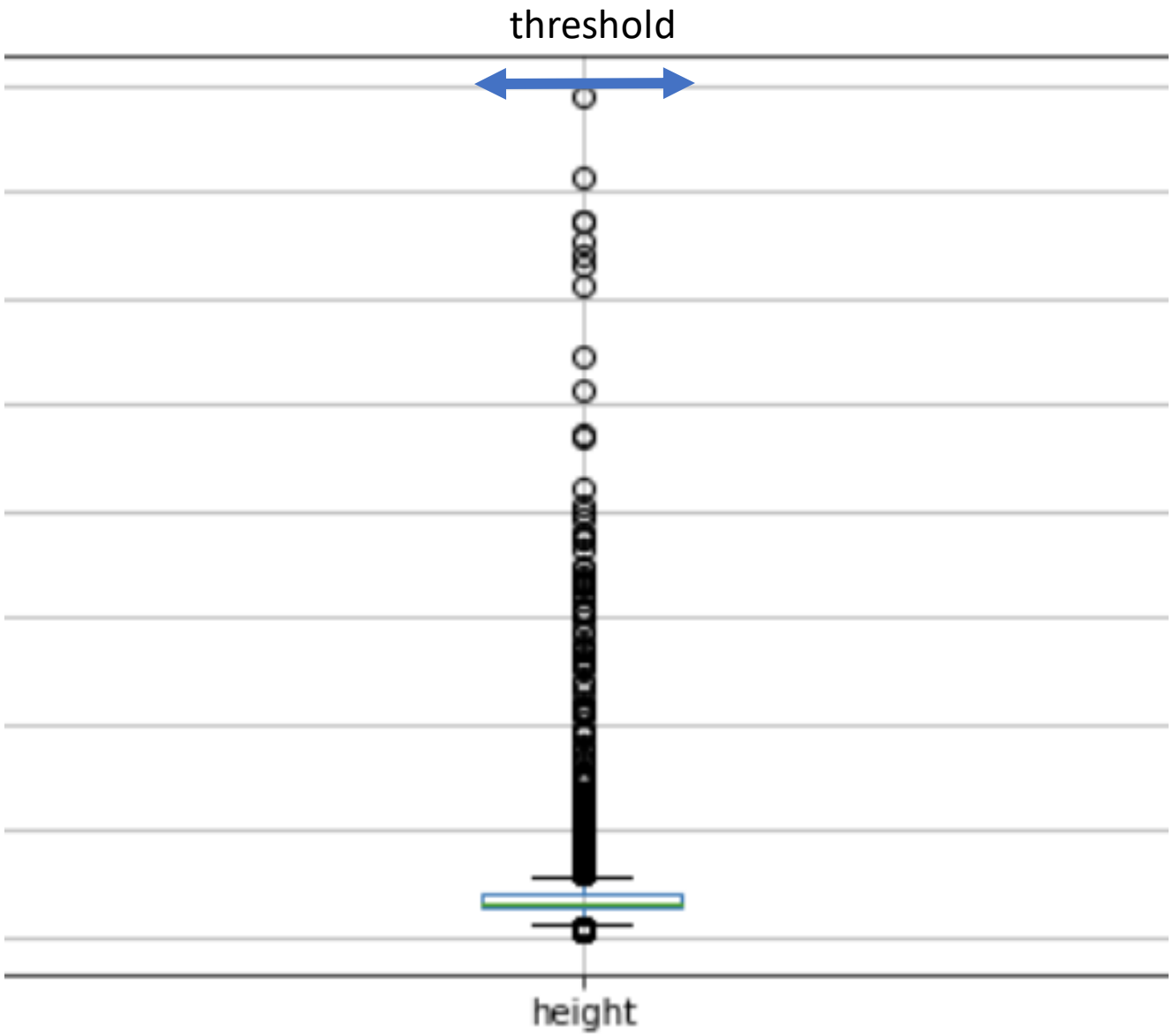


Data cleaning

- So we have decided to delete extreme values above a certain threshold to obtain a better distribution of features. Here is the example for height feature:

For example for the height feature, we have deleted values over 200 then we got a better representation. A particular filter has been set for every features that contains too many outlier values.

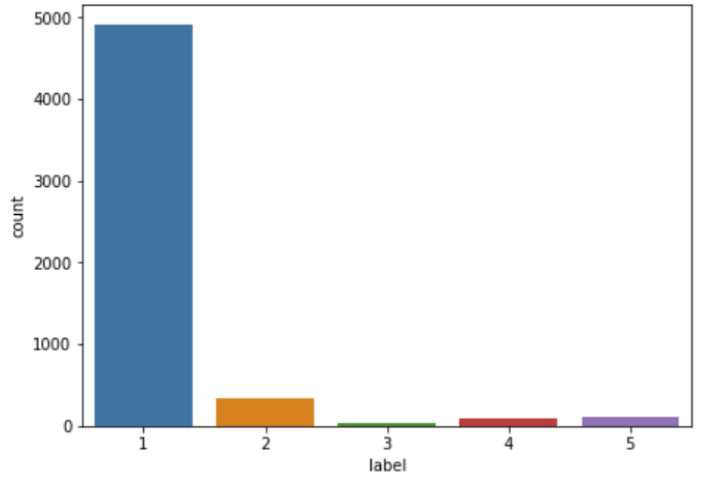
But we realized that by deleting outliers, we lost data that are important for the Machine Learning model. This will be explained in the next slide.



Data cleaning

- By deleting some outlier values we realized that we lost important data, as the following graph shows. These graphs represent the number of each classes in the label column.
- Indeed, we lost the minority labels. For example for the class "3" or picture class : we went from 28 to 22. We have lost more than 25% for this class. The only class that has not lost any data is the class "1", and this class already represent the majority of the dataset.
- This is not good for our Machine Learning models. So for the rest, we worked with the full dataset.

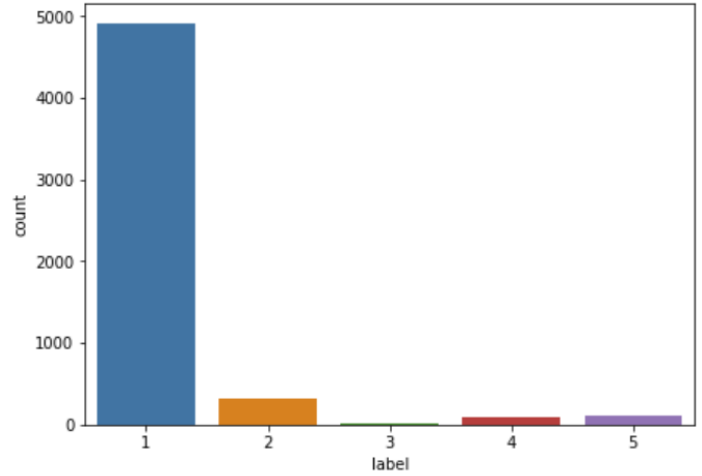
```
1    4913
2     329
5     115
4      88
3      28
Name: label, dtype: int64
```



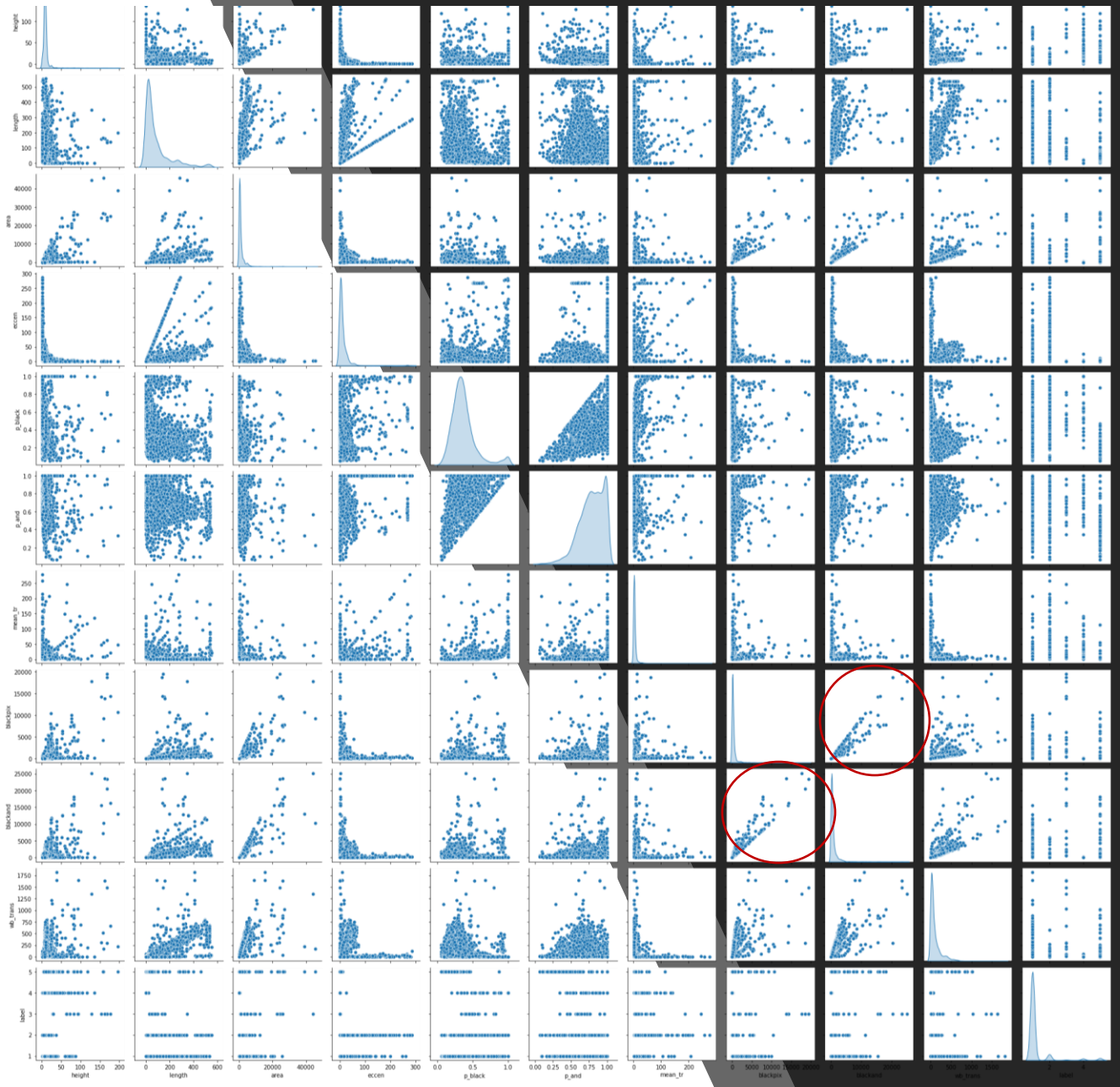
Full dataset

Cleaning outliers

```
1    4913
2     325
5     110
4      85
3      22
Name: label, dtype: int64
```



Cleaned dataset

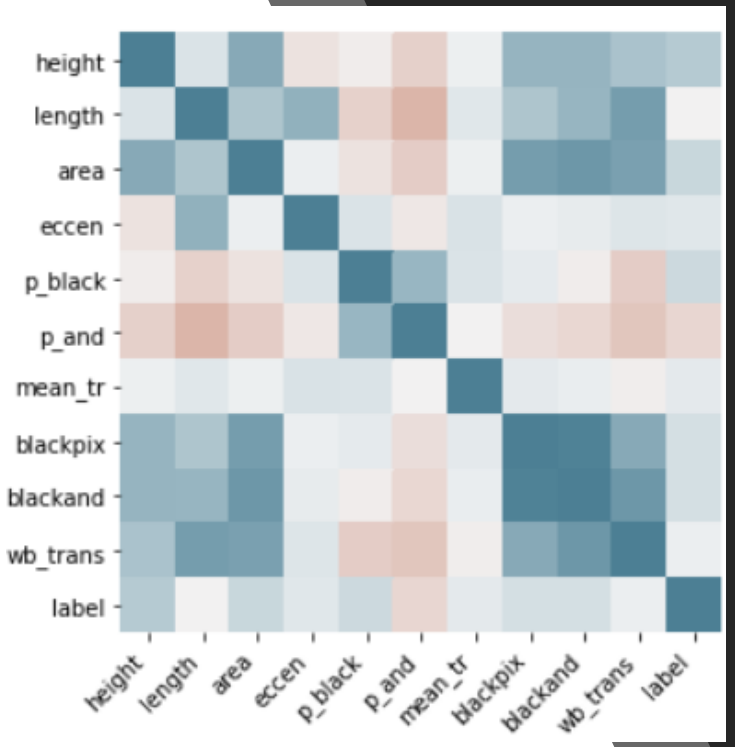


Correlation visualization

- Then we realized some plots to visualize correlation between features, with seaborn module we obtained this picture on the left.

- We observed a correlation between blackpix and blackand features as circled in red in the left picture. The other features do not seem to be correlated or just a little bit.

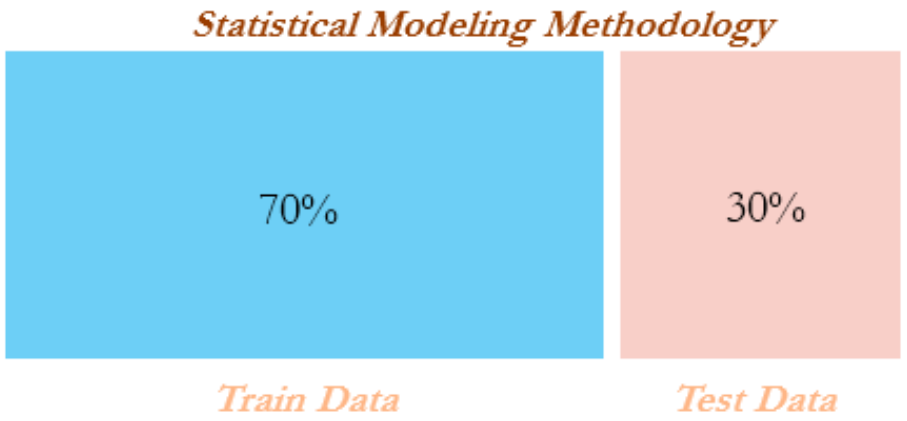
Correlation visualization



- The correlation matrix confirmed our observations.
- After having vizualised and processed the dataset, we moved to the Machine Learning Models. This is a classification model, so we have tested multiple classifiers to compare the scores. This will be explained in the next slides

Splitting data

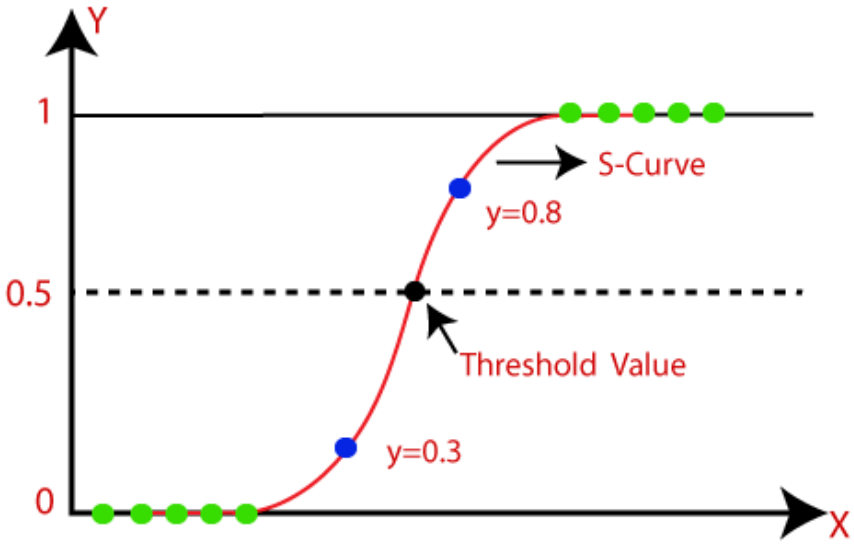
- Before making the models, we have separated the dataset into test and training datasets. Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing. In order to do so, we used the train_test_split method from the sklearn library. We keeped 30% of the dataset for the testing set and the rest for training.



Machine Learning Models

- Logistic regression

We started by the logistic regression model. Logistic Regression is used when the target variable is categorical. The output of a logistic regression model is a sigmoid function. To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.

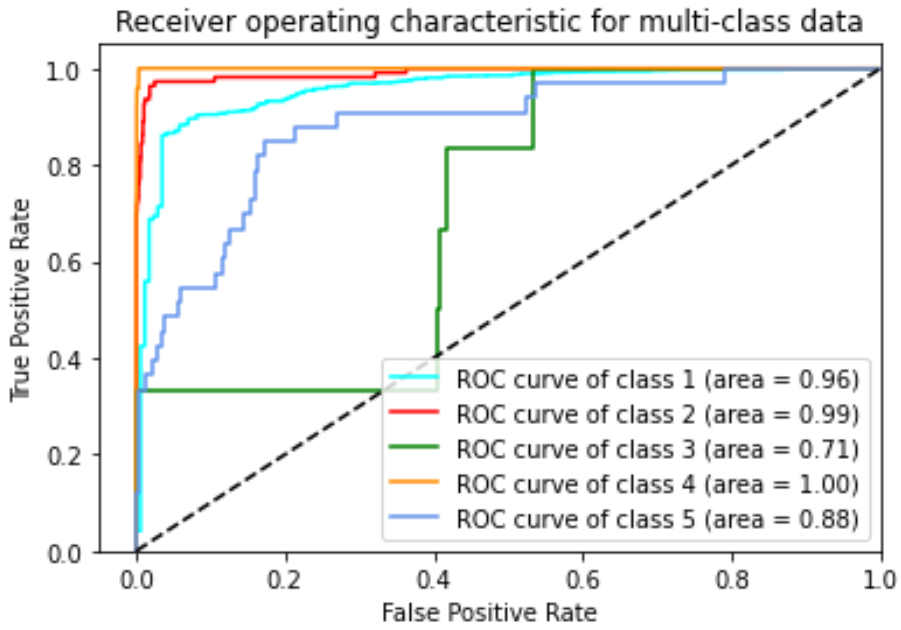


Machine Learning Models

- Logistic regression

We used the sklearn library to make this model.
Then we plotted the ROC curves of every class. ROC is a probability curve and AUC (Area Under the Curve) represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.

As we can see, the model is not very efficient with class 3. It is certainly due to a poor amount of data from this class. But the other class are well classified, with an Area Under Curve of at least 0.88.



Machine Learning Models

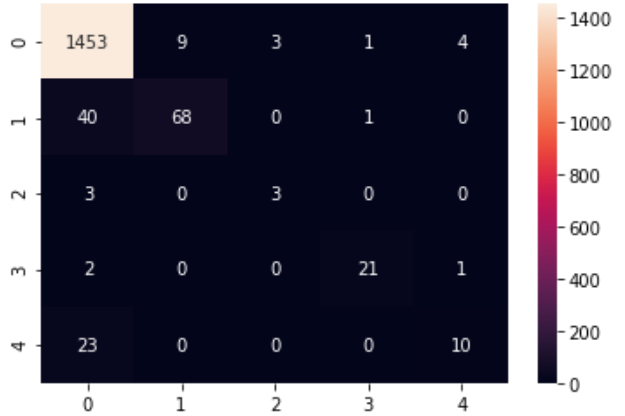
- Logistic Regression Hyperparameters tuning:

In order to tune the hyperparameters and get the better results from the the Logistic Regression model we have used Grid Search and Cross Validation methods. These processing tools are provided the Grid SearchCV method from sklearn.

Here are the results after Hyperparameter tuning, we have the classification report and the confusion matrix. We can see that the class "3" is not correctly classified in 50% of the cases. Indeed, the f1-score is very low. The accuracy is 95%.

The precision metrics are calculated by these formulas :

	precision	recall	f1-score	support
1	0.96	0.99	0.97	1470
2	0.88	0.62	0.73	109
3	0.50	0.50	0.50	6
4	0.91	0.88	0.89	24
5	0.67	0.30	0.42	33
accuracy			0.95	1642
macro avg	0.78	0.66	0.70	1642
weighted avg	0.94	0.95	0.94	1642



$$(10.1)\quad Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$(10.2)\quad Precision = \frac{T_p}{T_p + F_p}$$

$$(10.3)\quad Recall = \frac{T_p}{T_p + T_n}$$

$$(10.4)\quad F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

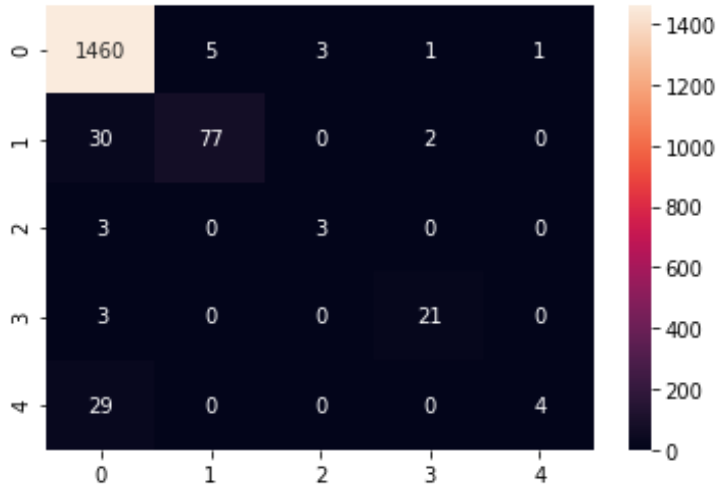
Machine Learning Models

- Support Vector Machine Model:

Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks. But, it is widely used in classification objectives. The sklearn library provide tools to create SVM models.

We directly applied a grid search and cross validation to this model to get the best parameters and the results are on the right. This model is not really better than the logistic regression model. The class 5 is very badly classified with a f1 score of 0.21. And the accuracy is 95%, it is the same than the previous model.

	precision	recall	f1-score	support
1	0.96	0.99	0.97	1470
2	0.94	0.71	0.81	109
3	0.50	0.50	0.50	6
4	0.88	0.88	0.88	24
5	0.80	0.12	0.21	33
accuracy			0.95	1642
macro avg	0.81	0.64	0.67	1642
weighted avg	0.95	0.95	0.95	1642



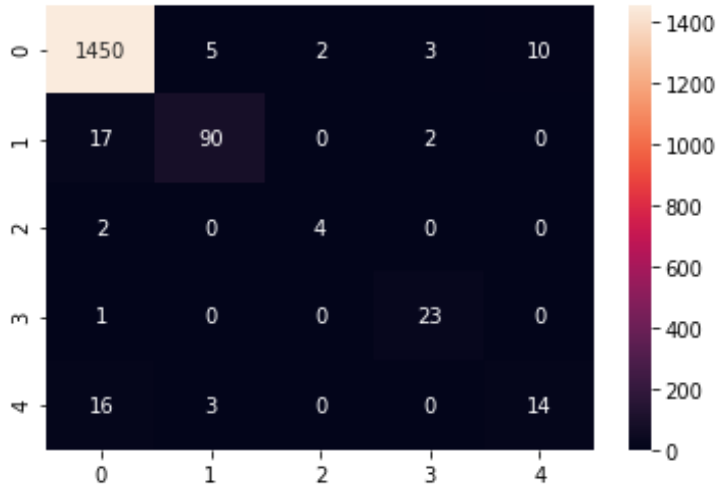
Machine Learning Models

- Decision Tree:

A decision tree is a largely used non-parametric effective machine learning modeling technique for regression and classification problems. To find solutions a decision tree makes sequential, hierarchical decision about the outcomes variable based on the predictor data.

After performing Cross Validation and Grid Search, we have the following results on the right. We have a better result than the 2 previous models with an accuracy of 97%, and a f1-score improved for classes "3" and "5".

	precision	recall	f1-score	support
1	0.98	0.99	0.98	1470
2	0.93	0.84	0.88	109
3	0.67	0.67	0.67	6
4	0.85	0.92	0.88	24
5	0.62	0.45	0.53	33
accuracy			0.97	1642
macro avg	0.81	0.77	0.79	1642
weighted avg	0.96	0.97	0.96	1642



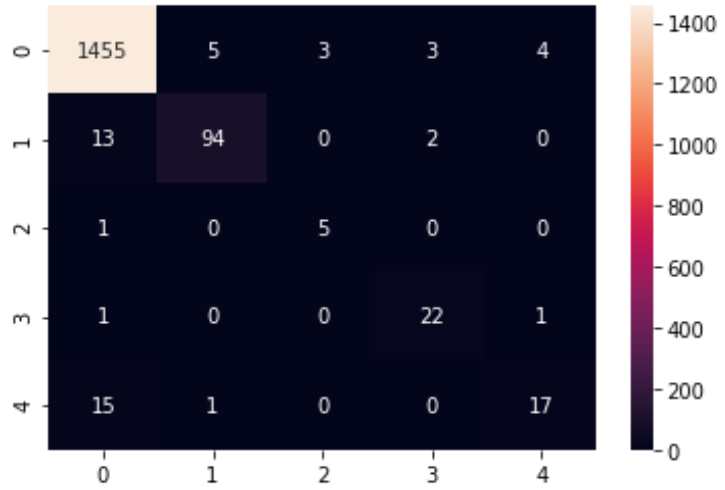
Machine Learning Models

- Random Forest:

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

After performing Cross Validation and Grid Search, we have the following results on the right. We have a better result than the Decision Tree in class "3" and "5". The accuracy is 97%, almost 98%. We chose this model to run with the flask API because it is the most accurate.

	precision	recall	f1-score	support
1	0.98	0.99	0.98	1470
2	0.94	0.86	0.90	109
3	0.62	0.83	0.71	6
4	0.81	0.92	0.86	24
5	0.77	0.52	0.62	33
accuracy			0.97	1642
macro avg	0.83	0.82	0.82	1642
weighted avg	0.97	0.97	0.97	1642



Conclusion

At the end of this project, we reached the final goal of programming a Flask application . The requested functionalities have been successfully achieved. We can only be satisfied to met the expectations of the project and to have completed it. We are happy to reach this stage for a first experimentation of a programming project of one month. We were able to get a brief overview of a professional project in the field of programming.

Moreover, the realization of this project has been more than beneficial to us. First of all, it allowed us to develop, better understand and put into practice the knowledge we learned during the beginning of the semester. In addition to that, this project allowed us to develop our abilities to manage our time, collaborate and work in groups to adopt the right working methods. Thus, it has been beneficial insofar as collaborative work is indispensable in most fields of the professional world.

However, being still novices, many difficulties were encountered and there are many points that can be further improved. This help us to realize that there is still a lot to learn and that we need to accumulate experience. The first difficulties we encountered were at the beginning of the project. We felt a bit lost as we didn't really know where to start. Thanks to the teachers and the courses we were given, we were soon able to be better directed and start our project on a good basis.

Finally, the last obstacle was during the coding of the Flask application, we were unable to find suitable solutions to this problem, which still needs to be improved: the responsiveness of the page, which we feel is not complete enough. Concerning the coding of this function, we must have had a lack of analysis and time.