# FINAL DELIVERY FOR PRACTICAL ASSIGNMENT 2

ELEMENTS OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

# DS PROJECT

## Work To Be Performed And How To Develop It

- Our current objective is to develop two Supervised Learning ML models (a Decision Tree Model and a K-NN Model) to predict the user rating of a game from a training dataset.

- The work is currently being developed in python using data science tools such as the *Pandas* and *Scikit-Learn* libraries to implement the models.

## What Was Done

- The project's workflow followed the order stated below:
  - Data cleaning and pre-processing: removing naturally unrelated characteristics, dropping missing values and encoding categorical features.
  - Data analysis: calculating the correlation matrix and representing it through a heatmap.
  - Model building and optimization: creating Decision Tree and KNN models, cross-validation, measuring correctness, determining the optimal parameters for each model.
  - Model comparison: comparision against a Dummy model, calculating learning curves, training and testing errors, and ROC curves.

## DATA ANALYSIS

- Firstly, we encoded categorical attributes to numerical, in order to use them in a correlation matrix. For that we've used the *LabelEnconder* function on some attributes and abstracted the others such as the "platforms" attribute to the number of platforms supported by each game.
- To calculate the correlation matrix, we used the *.corr()* function with the *"spearman"* method, and represented it through a heatmap, only showing values above 0.1 (scale -1 to 1).
- The *spearman* method is used for ordinal data, such as the categorical data present in this dataset, hence why we used it.

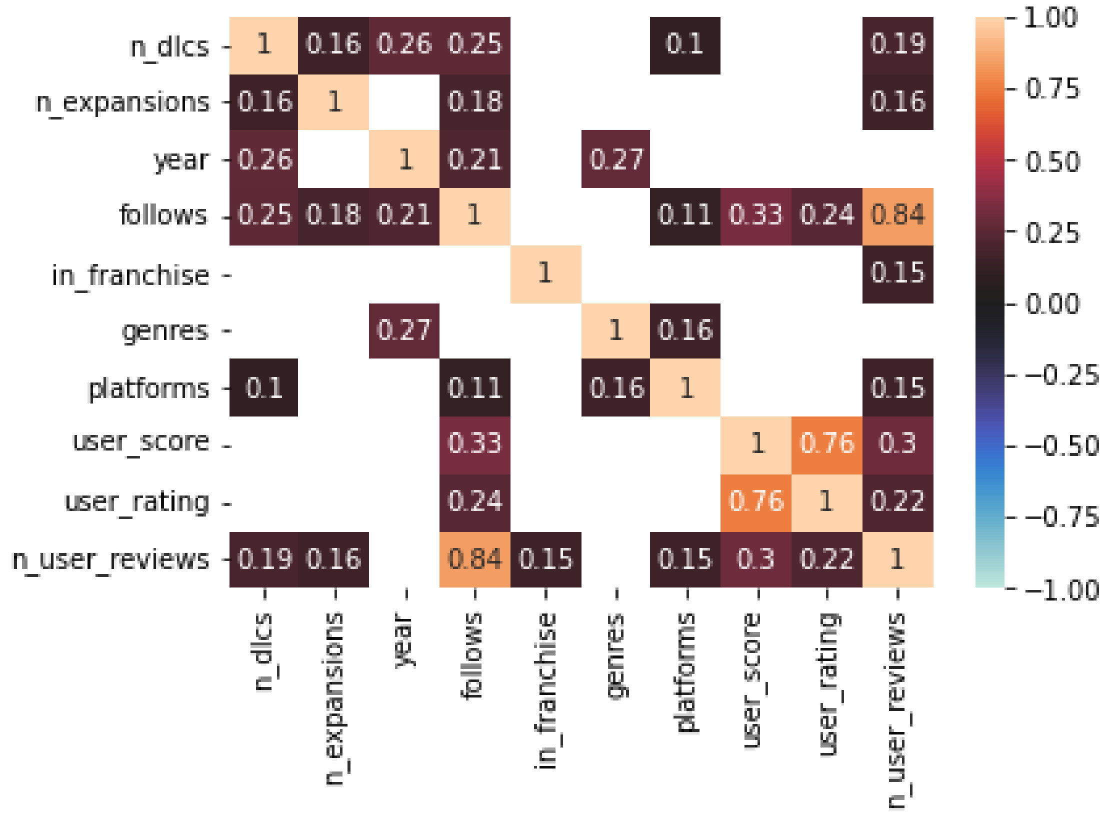
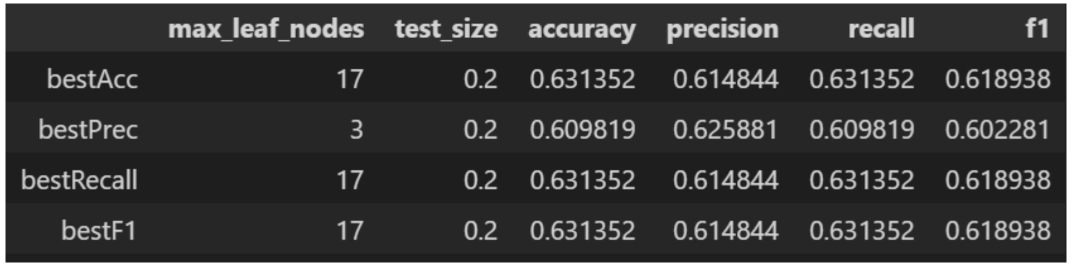


Fig. 1 – Correlation Matrix Heatmap

# DEVELOPMENT AND OPTIMIZATION OF DECISION TREE & KNN MODELS

## The DT Model

- After selecting which attributes to base the models on, in this case, "*user_rating*" in relation to "*n_dlcs*", "*follows*", "*in_franchise*" and "*n_user_reviews*", we set to build the models.
- In the DT building process, we first implemented a basic sklearn DT Classifier. Then we went on with optimizing it.
- To find the optimal parameters of the DT, we tested the best accuracy, precision, recall and F1 score for different *max_leaf_nodes* and *test_size* values (see Fig. 2 ).
- We concluded that the optimal DT model is for *max_leaf_nodes* = 17 and *test_size* = 0.2.
- The main issue with our DT was that it only classified as "Good" or "Great".



| | max_leaf_nodes | test_size | accuracy | precision | recall | f1 |
|---|---|---|---|---|---|---|
| bestAcc | 17 | 0.2 | 0.631352 | 0.614844 | 0.631352 | 0.618938 |
| bestPrec | 3 | 0.2 | 0.609819 | 0.625881 | 0.609819 | 0.602281 |
| bestRecall | 17 | 0.2 | 0.631352 | 0.614844 | 0.631352 | 0.618938 |
| bestF1 | 17 | 0.2 | 0.631352 | 0.614844 | 0.631352 | 0.618938 |

Fig. 2 – Best *max_leaf_nodes* and *test_size* for Accuracy, Precision, Recall and F1 score

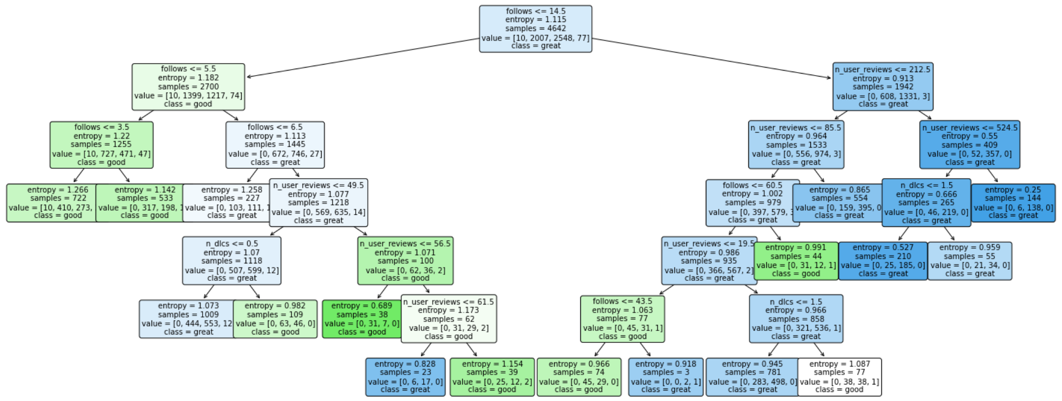
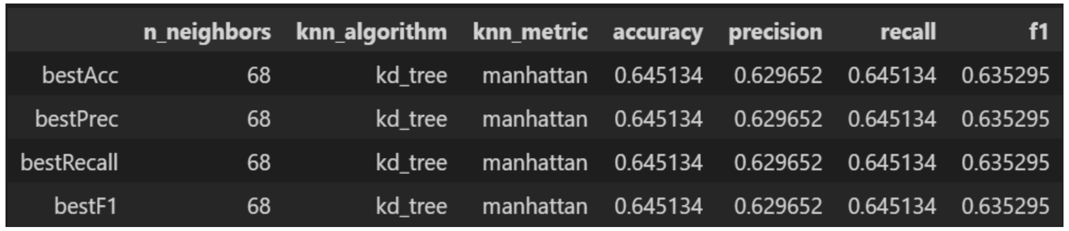

follows <= 14.5 entropy = 1.115 samples = 4642 value = [10, 2007, 2548, 77] class = great
follows <= 5.5 entropy = 1.182 samples = 2700 value = [10, 1399, 1217, 74] class = good
n_user_reviews <= 212.5 entropy = 0.913 samples = 1942 value = [0, 608, 1331, 3] class = great
follows <= 3.5 entropy = 1.22 samples = 1255 value = [10, 727, 471, 47] class = good
follows <= 6.5 entropy = 1.113 samples = 1445 value = [0, 672, 746, 27] class = great
n_user_reviews <= 85.5 entropy = 0.964 samples = 1533 value = [0, 556, 974, 3] class = great
n_user_reviews <= 524.5 entropy = 0.55 samples = 409 value = [0, 52, 357, 0] class = great
entropy = 1.266 samples = 722 value = [10, 410, 273, class = good
entropy = 1.142 samples = 533 value = [0, 317, 198, 1 class = good
entropy = 1.258 samples = 227 value = [0, 103, 111, 1 class = great
n_user_reviews <= 49.5 entropy = 1.077 samples = 1218 value = [0, 569, 635, 14] class = great
follows <= 60.5 entropy = 1.002 samples = 979 value = [0, 397, 579, 3 class = great
entropy = 0.865 samples = 554 value = [0, 159, 395, 0 class = great
n_dlcs <= 1.5 entropy = 0.666 samples = 265 value = [0, 46, 219, 0] class = great
entropy = 0.25 samples = 144 value = [0, 6, 138, 0] class = great
n_dlcs <= 0.5 entropy = 1.07 samples = 1118 value = [0, 507, 599, 12] class = great
n_user_reviews <= 56.5 entropy = 1.071 samples = 100 value = [0, 62, 36, 2] class = good
n_user_reviews <= 19.5 entropy = 0.986 samples = 935 value = [0, 366, 567, 2] class = great
entropy = 0.991 samples = 44 value = [0, 31, 12, 1] class = good
entropy = 0.527 samples = 210 value = [0, 25, 185, 0] class = great
entropy = 0.959 samples = 55 value = [0, 21, 34, 0] class = great
entropy = 1.073 samples = 1009 value = [0, 444, 553, 12 class = great
entropy = 0.982 samples = 109 value = [0, 63, 46, 0] class = good
entropy = 0.689 samples = 38 value = [0, 31, 7, 0] class = good
n_user_reviews <= 61.5 entropy = 1.173 samples = 62 value = [0, 31, 29, 2] class = good
follows <= 43.5 entropy = 1.063 samples = 77 value = [0, 45, 31, 1] class = good
n_dlcs <= 1.5 entropy = 0.966 samples = 858 value = [0, 321, 536, 1] class = great
entropy = 0.828 samples = 23 value = [0, 6, 17, 0] class = great
entropy = 1.154 samples = 39 value = [0, 25, 12, 2] class = good
entropy = 0.966 samples = 74 value = [0, 45, 29, 0] class = good
entropy = 0.918 samples = 3 value = [0, 0, 2, 1] class = great
entropy = 0.945 samples = 781 value = [0, 283, 498, 0] class = great
entropy = 1.087 samples = 77 value = [0, 38, 38, 1] class = good

Fig. 3 – Representation of the Decision Tree Model

## DEVELOPMENT AND OPTIMIZATION OF DECISION TREE & KNN MODELS

### The KNN Model

- For the KNN model, we started with a base sklearn KNeighborsClassifier model as well and then we repeated the optimization procedure to find the parameters which yield the best results.
- As we can see in Fig. 4, the model with *n_neighbors* = 68, *knn_algorithm* = "kd_tree" and *metric* = "manhattan" secures the best results for all measures.
- Contrary to the DT model, our KNN assigned each rating at least once.



| | n_neighbors | knn_algorithm | knn_metric | accuracy | precision | recall | f1 |
|---|---|---|---|---|---|---|---|
| bestAcc | 68 | kd_tree | manhattan | 0.645134 | 0.629652 | 0.645134 | 0.635295 |
| bestPrec | 68 | kd_tree | manhattan | 0.645134 | 0.629652 | 0.645134 | 0.635295 |
| bestRecall | 68 | kd_tree | manhattan | 0.645134 | 0.629652 | 0.645134 | 0.635295 |
| bestF1 | 68 | kd_tree | manhattan | 0.645134 | 0.629652 | 0.645134 | 0.635295 |

Fig. 4 – Best KNN parameters for Accuracy, Precision, Recall and F1 score

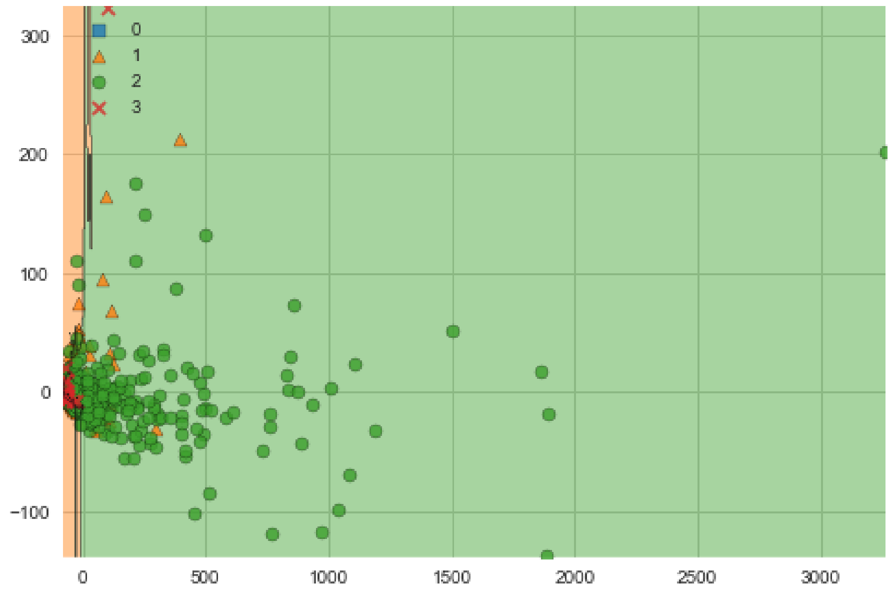
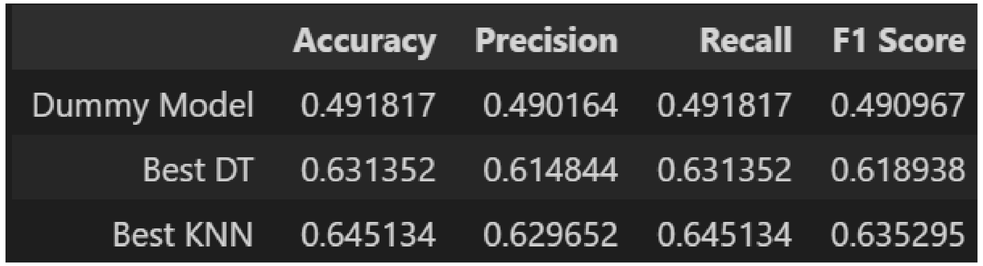


Fig. 5 – Representation of the KNN Decision Region

# COMPARISON BETWEEN OPTIMAL MODELS AND A DUMMY MODEL

- As a control sample to compare our best DT and KNN models, we created a Dummy model with sklearn’s DummyClassifier with *strategy*=“stratified”, since its weighted randomness will simulate a better random classifier according to the distribution of our data.
- From the table in Fig. 6, we can see that our Decision Tree and KNN model outperform the Dummy model by ~14% in all values of Accuracy, Precision, Recall and F1 Score.
- These differences indicate us that both models are slightly better than a random classifier.



| | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Dummy Model | 0.491817 | 0.490164 | 0.491817 | 0.490967 |
| Best DT | 0.631352 | 0.614844 | 0.631352 | 0.618938 |
| Best KNN | 0.645134 | 0.629652 | 0.645134 | 0.635295 |

Fig. 6 – Train / Test Error of DT

## EVALUATION AND COMPARISON BETWEEN BOTH MODELS

### Training and Testing Error

- As we can see in Fig. 7, the Decision Tree model from 30 *leaf nodes* up has a high variance increase rate, which means it is prone to overfit around the data. That is expectable given that a more complex DT tends to overfit.

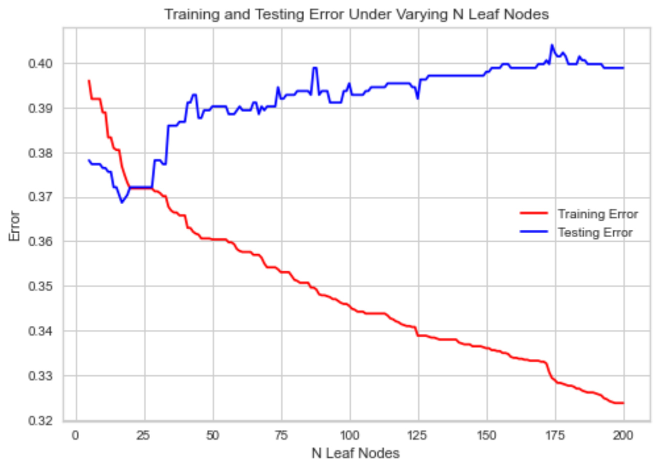


Fig. 7 – Train / Test Error of DT

- Since the KNN overfitting decreases with the increase of the number of neighbors, and larger values for K may lead to underfitting, we used cross-validation to find a suitable value for K. And as we can see in Fig. 8, the number of neighbors previously calculated and now verified to yield the lowest testing error is 68.

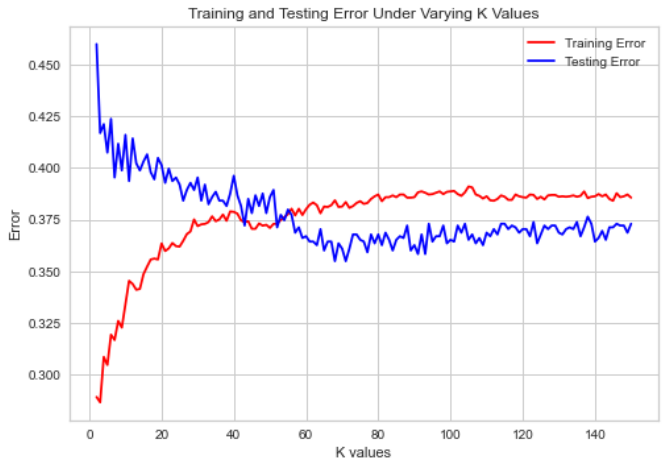


Fig. 8 – Train / Test Error of KNN

## EVALUATION AND COMPARISON BETWEEN BOTH MODELS

### Learning Curves

- Studying the Learning Curve for both models helps us to see how accuracy changes with varying sample size.

- From Fig. 9 we can conclude that both models have similar learning curves. However, the rate of change in accuracy per sample size is very small: in both DT and KNN, accuracy only increases ~6% with more 4000 samples to train the models on.
- Therefore, we can ascertain that giving more data to develop the models on will not make much of a difference in the models' accuracy.

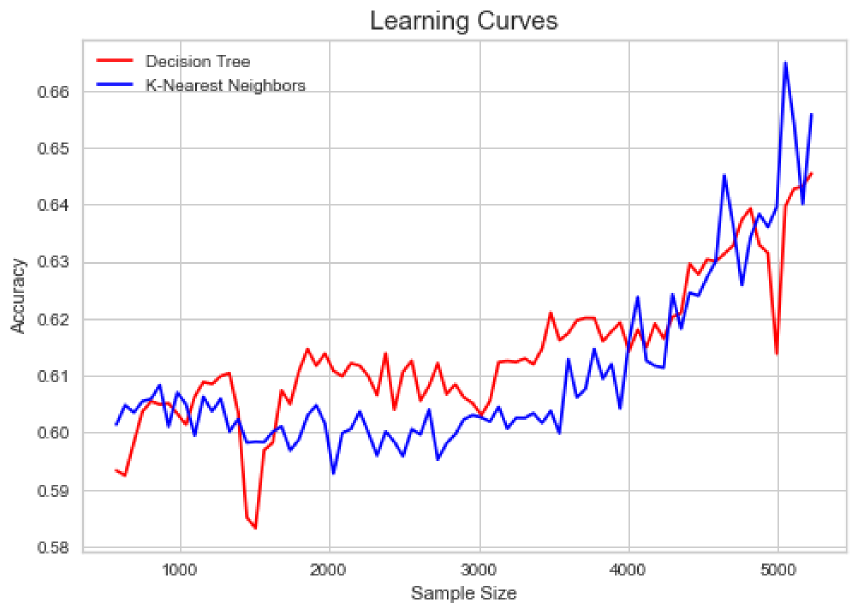


Fig. 9 – Learning Curves of DT and KNN Models

## EVALUATION AND COMPARISON BETWEEN BOTH MODELS

### ROC – Receiver Operating Characteristics

- The ROC gives us the connection between the True Positive Rate (TPR) in relation to the False Positive Rate (FPR), which can help us determine if a classifier is good – High TPR & Low FPR – or otherwise.
- All ROCs of the DT model show that its TPR has an acceptable relation to FPR, since the curve are relatively close to the diagonal line in Fig. 10. Thus, we can assess the DT model as a mediocre classifier.
- Just like the DT's ROCs, KNN has a slightly good TPR to FPR relation (Fig. 11), although for lower FPR has worse TPR in comparison to DT's.

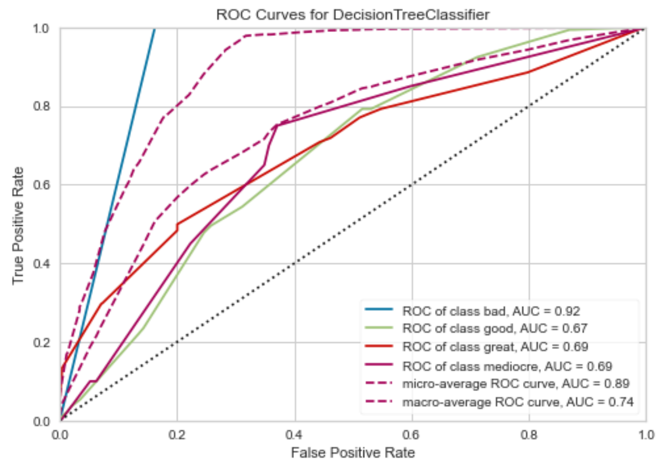


Fig. 10 – ROC and AUC of DT for each label

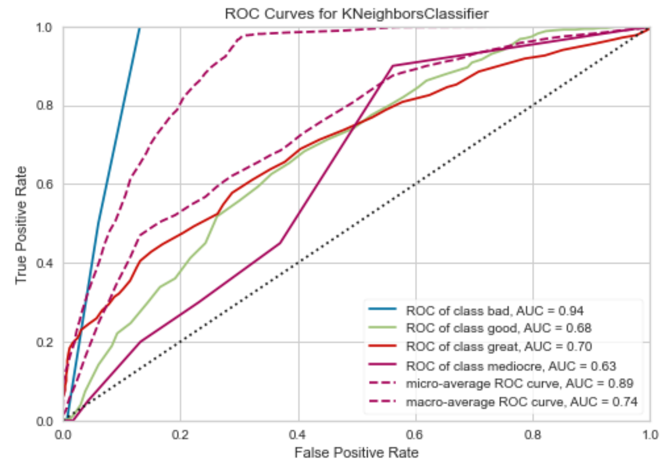


Fig. 11 – ROC and AUC of KNN for each label

## MAIN DIFFICULTIES AND CONCLUSIONS

- On a first stage the main difficulty was finding the most suited attributes on the dataset for the classification model due to weak correlation between attributes.
- Searching for a correctness measure was another struggle for us, since we were working on a non-binary classification model. We ended up settling for accuracy, precision, recall and F1 score as correctness measures.
- Initially while tidying the dataset, we observed that “bad” and “mediocre” labels only made up ~2% of all samples’ labeling, which, when creating the training sets lead to poor learning of the machine learning models.

- During this practical assignment, the group was able to experience what are the phases of the Data Science pipeline and its workflows.
- Working with Python’s broad libraries has been a good introduction to the tools we might use during a Data Science project.