Alma Mater Studiorum - University of Bologna
LM Informatica

Data Analytics Project

Cotugno Giosué [983620] - Pruscini Davide [1007343]

July 19, 2022

Contents

1 Introduction **5**

2 Methodology **7**

2.1 Data Acquisition 7

2.2 Data Pre-process 7

2.3 Modeling 13

2.4 Performance Analysis 15

Chapter 1

Introduction

In this report we will discuss about the whole data pipeline of a project that uses different techniques of machine learning to classify some tabular data. In particular, we choose the first project proposal that aims to predict the average mark of a film from its features. The first step of the data pipeline requires to download, save and load in memory the MovieLens [1], TMDB [2] and IMDb [3] datasets. Consequently, the datasets were elaborated with the objective to generate a unified dataset that can be used as an input for some machine learning algorithms. Afterwards, during the modelling phase we built an MLP model thanks to the PyTorch framework [4]. In addition, we used other techniques like SVM, tree methods and naive bayes methods that are available into the Scikit-Learn library [5]. In order to find a good configuration during the performance analysis, it was mandatory to define a large enough hyperparameters space for all the models that we defined. Moreover, in the last phase, the cross validation technique was used to obtain more robust statistics results that has been compared between the trained models.

Chapter 2

Methodology

In the next chapter there will be the explaination of the data pipeline that the project followed. In particular, each subsection will focus on a specific task, except for the data visualization that has been used only when needed.

2.1 Data Acquisition

The used datasets are downloaded at runtime directly from the sources. These datasets come directly from MovieLens' page and provides 6 different files. More informations about the nature of the data are available here.

Dataset	**Features**
ratings.csv	userId, movieId, rating, timestamp
tags.csv	userId, movieId, tag, timestamp
movies.csv	movieId, title, genres
links.csv	movieId, imdbId, tmdbId
genome-scores.csv	movieId, tagId, relevance
genome-tags.csv	tagId, tag

Most of these datasets provides information for approximately 60000 films. The links dataset provides two identifiers that allow to collect information from the IMDB and TMDB databases. Thanks to the links' features, it has been possible to collect some more information on the running times of the films that could provide more insight into them. Talking about the ground truth of the supervisioned models, the rating mean is missing. So the target feature will be computed during the pre-process phase thanks to the ratings dataset. Further information about the features usage and the cardinality of the datasets will be discussed in the Pre-Process section.

2.2 Data Pre-process

In this section, will be discussed the pre-process phase for each of the above presented datasets. In order to achieve a major clearity, the work on each dataset will be discussed in a specific subsection where the operation computed on them will be explained. At the beginning of each subsection the cardinality will be reported.

Movies - movies.csv

Cardinality: 58098×3

Inside this dataset, the title and generes features contains multiple information. In particular, the title has been splitted in two part, where on one hand there's only the title name, and on the other, there's the year of the film production. Since the title name is a string, that doesn't add more information to a classic machine learning model, this feature has been converted into its length meanwhile the production year just constitutes a new feature. The genres feature contains a pipe separated list of a fixed possible values. Since the list is just saying if a film has a specific genre or not, to each film, all the fixed values has been added as a feature, and if a genre appears into the genres feature, that column will result into 1 that indicate the presence of that genre, 0 otherwise. At the end of this initial phase, the first sample of the movies dataset appears as shown in the Table 2.1.

movieId	**title_len**	**year**	**action**	**adventure**	**...**	**Western**	**(no genres listed)**
1	16	1995	0	1	...	0	0

Table 2.1: First sample of movis interim dataset.

In order to finish the data pre-process on this dataset, the data cleaning is required. First of all, there are some films where the year of the film production is missing. Analyzing the distribuition of these, as showed in Figure 2.1, it is possible to see that the distribuition is right skewed, so the missing values can be filled with the **median**, as suggested during the lectures.

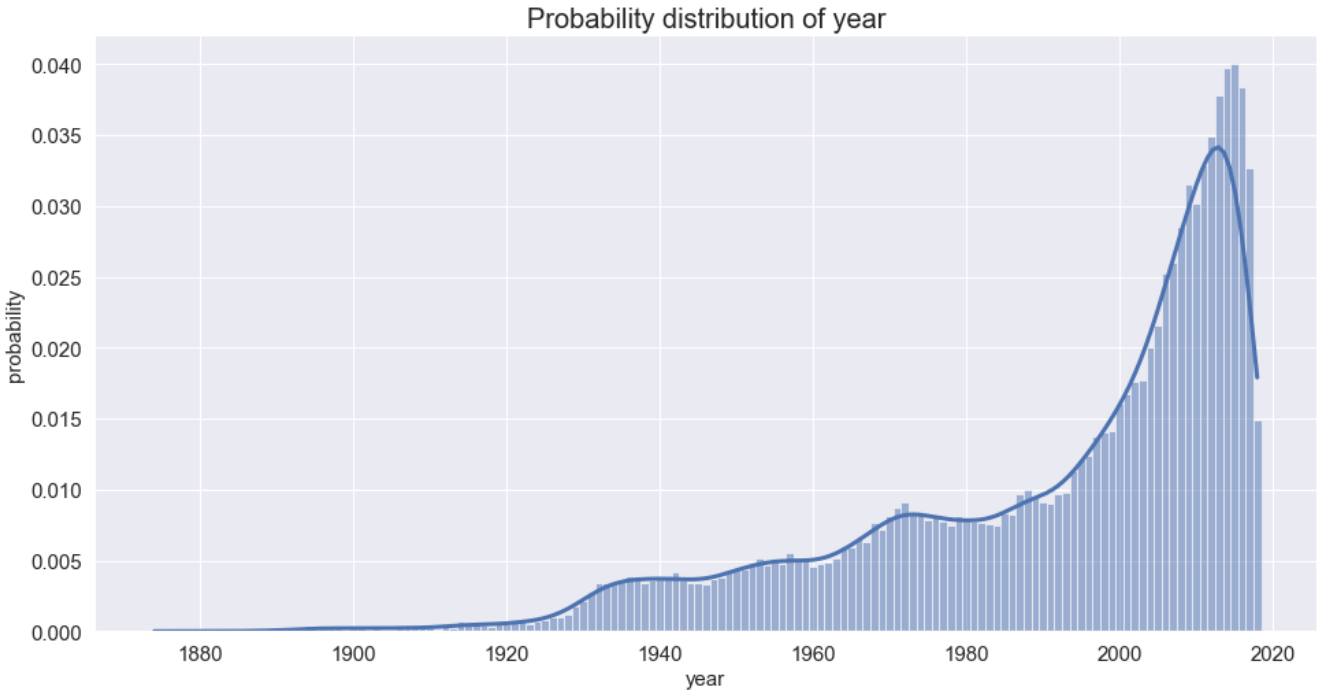


Figure 2.1: Right skewed probability distribution of the year feature.

Since the feature (no genres listed) and the films that has no genres provide no information about the film, they will be removed obtaining a final cardinality of 53832×22.

Tags - tags.csv

Cardinality: 1108997×4

In order to use the relevant data from this dataset the timestamp and userId features have been deleted because they don't explain anything more about the films. The information provided by each sample in the dataset are not very meaningful, for this reason it was decided to count the number of tags associated with each film, in order to understand how much interaction that film generated. When a film doesn't have any related tag, the tag_count can be setted to 0 because no users were interested on that film. After these operations the cardinality was reduced to 45981×2.

Ratings - ratings.csv

Cardinality: 27753443×2

The features in this dataset were necessary to find the target column. In order to compute it, the average of each films' ratings has been calculated. In addition, the number of the rating on each film has been counted. However, the task of this project is the classification, for this reason the rating_mean has been discretized in 10 bins, where each of them covers a rating range of 0.45 as showed in Figure 2.2.

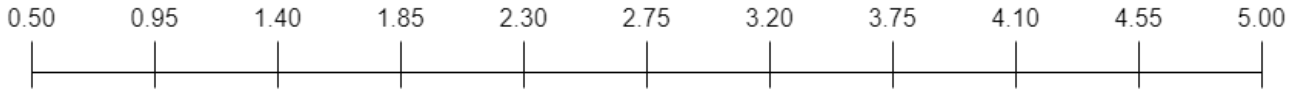


Figure 2.2: Discretization of the rating_mean feature.

At the end, the final cardinality is 53889×4.

Genome - genome-scores.csv and genome-tags.csv

Cardinality scores: 14862528×3
Cardinality tags: 1128×2

Talking about these two datasets, the merge operation over the tagId was needed because it was interesting to associate the tagId with its string name. After this union, on each sample there is the correspondence between a movieId, the tag name and the relevance of that tag. The final step consist on relate every film to the relevance of each tag, using the pivot function, as shown in Table 2.2.

movieId	007	18th century	action	absurd	...	addiction	adventure
1	0.02900	0.05425	0.66825	0.09725	...	0.07475	0.90700

Table 2.2: First sample of genome interim dataset.

The final dataset named genome.csv has the cardinality equal to 13176×1129.

Links - links.csv

Cardinality: 58098×3

On this dataset no pre-processing was needed because for each movie it contains two identifier that provide a link to external sources. The interesting one is tmdbId because it has been used to retrieve information from the TMDB database. Using the TMDB Api it has been possibile to build a new dataset that contains some additional features:

- budget
- revenue
- adult
- runtime

In order to avoid the expensive operation of the Api calls, the resulted dataset has been saved in a GitHub release. Due to the insufficient amount of valid samples, the IMDb title-basics.csv dataset was downloaded to try to fill in all missing values. After a brief exploration and analysis it has been possible to see that only the runtime feature has an enough amount of samples. Since there continue to be some missing values, the distribution of this feature was analyzed in Figure 2.3. It is possible to see that the distribuition is left skewed, so the missing values can be filled with the **median**.

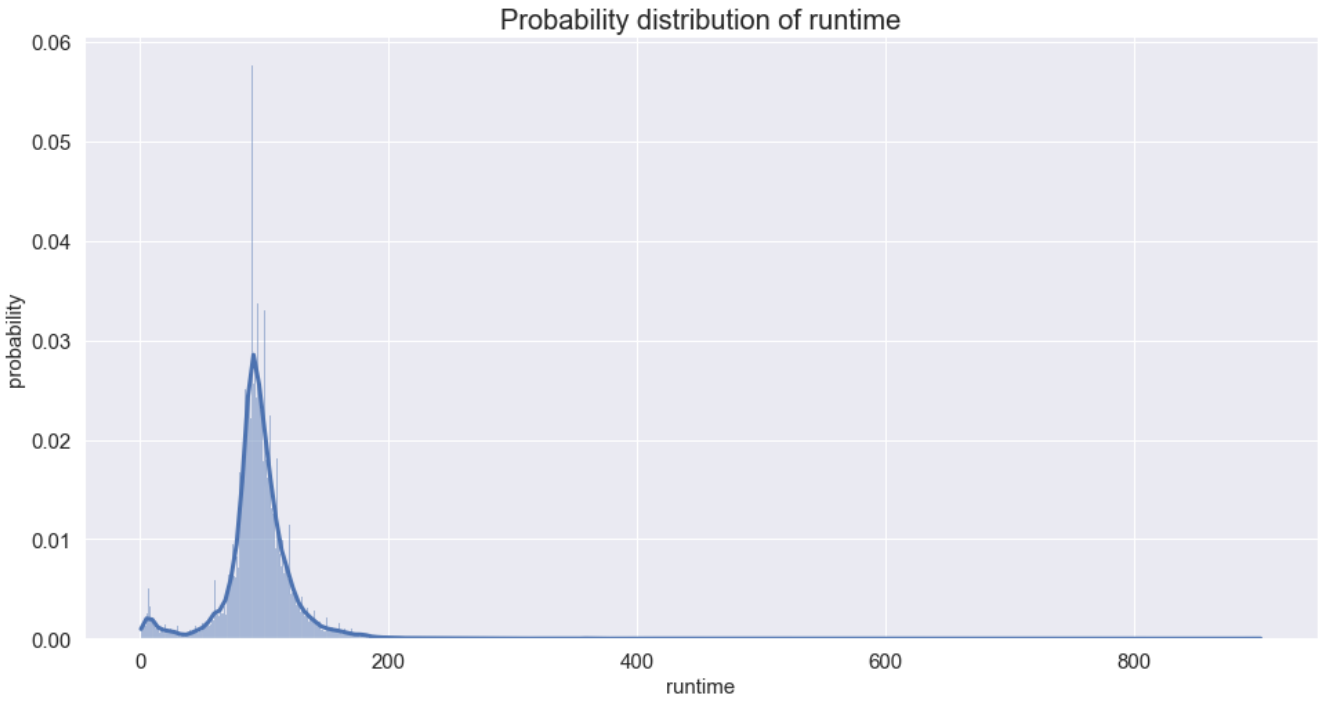


Figure 2.3: Left skewed probability distribution of the runtime feature.

The final cardinality of the acquired external resources is 58098×2.

Final - final.parquet

In this section it will be showed how the cardinality of the used dataset has been obtained, starting from all the previously processed datasets. In particular, the focus will be on the number of samples, because the number of columns will increase by the addition of the features coming from all the other datasets. Since for a classification task is interesting to know the information about a sample, the starting dataset will be the movies and then all the others will be merged, trying to preserve the maximum number of samples. The first merge has been between movies and ratings because if a film doesn't have the target feature there is no point to consider it. In addition, all films with no rating mean were discarded, resulting

in a cardinality of 50157×24. From this point on, all the merges will be performed using the previously calculated dataset, so only the new dataset will be specified. The second merge considers tags dataset, where there was an important number of samples that hasn't a tag count. In this case, these samples were not discarded because they were considered as a film that generated no users interaction, so all missing values were filled with 0, resulting in a cardinality of 50157×25. The third merge introduces external resources where for each film there is a related sample, so the cardinality depends from the previous one, resulting 50157×26. The final merge introduces the genome dataset, where there is a large gap between the cardinality of this dataset and the previous one. The choice for this merge was to discard all data that don't match with the genome dataset because it provides more interesting characteristics than the other datasets could provide. At the end, the cardinality of the final dataset is 13147×1154.

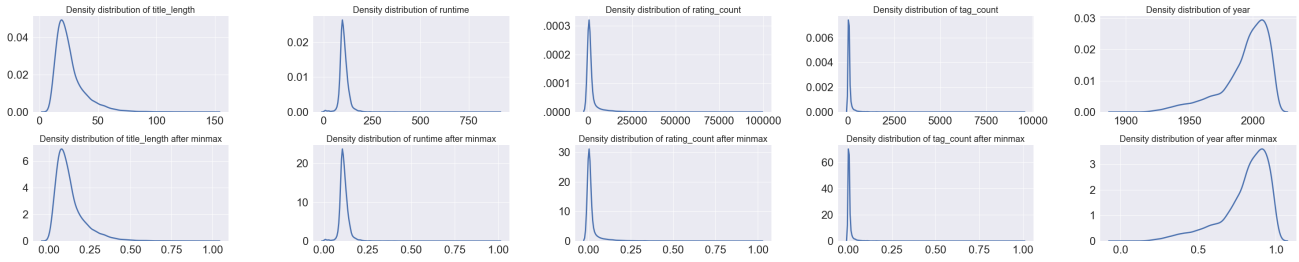
Data Transformation

After further exploration, it becomes apparent that some features did not belong to a well-defined range. For this reason, it was useful to apply certain transformation techniques such as min-max scaling and normalization. In order to apply some of these transformations, taking into account also the balancing task, the split of the dataset is needed because all these operations entail working on the training set. So, in this section, the dataset will be split with the ratios shown in Table 2.3.

Train set size	**Validation set size**	**Test set size**
72%	8%	20%

Table 2.3: Train/Validation/Test set ratios.

The features that have been analyzed are: title_length, runtime, rating_count, tag_count, year. It has been seen as the distribuition of these features are not Gaussian and for this reason the min-max scaling has been applied.



(a) Title length (b) Runtime (c) Rating count (d) Tag count (e) Year

Figure 2.4: Comparision of feature distribuition.

In order to have a greater reliability on the techniques to apply, a test function was created. This function uses the default version of the scikit models used in this project, which are: RandomForest, DecisionTree, GaussianNB, QDA and SVM. Only the scikit models were used due to their ease of use, which is why the Neural Network does not appear in these tests. From the results, some graphs were produced to better interpret the results. Thanks to this function, 4 configuration has tried: Normalization of the sample together with scaling, Normalization of the sample without categorical feature with scaling, Minmax scaling, normalization was also tried, but after some tests it turned out that the normalization technique in addition to minmax scaling did not provide any improvement, as shown in Figure 2.5. An

attempt was also made to apply only normalization without min-max scaling, but since the dataset with scaling gave slightly better results than the one with normalization, it was decided to apply only the min-max scaling technique to the final dataset.

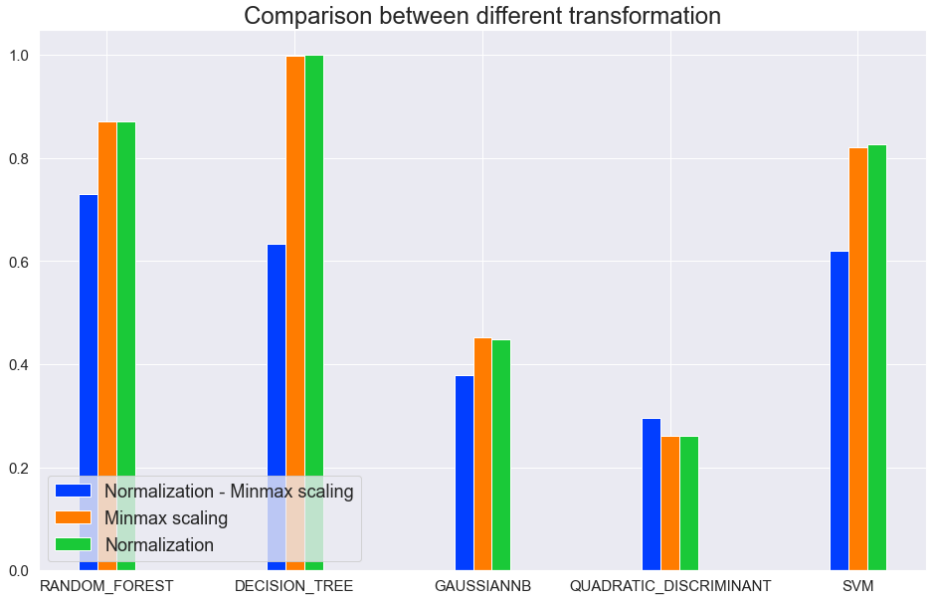


Figure 2.5: Comparision between normalization and minmax scaling

Data balancing

The dataset that came from all the previous stages is strongly unbalanced as showed in Figure 2.6.

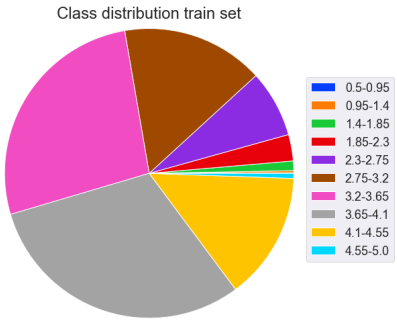


Figure 2.6: Initial class distribuition over the samples.

For this reason a balancing technique is needed to be applied on this dataset. Since, there are a lot of options in the Imbalanced-learn library [6], it has been applied several techniques of balancing, to try and find the best one. In order to do this task, the previously testing function has been used obtaining results as showed in Figure 2.7.

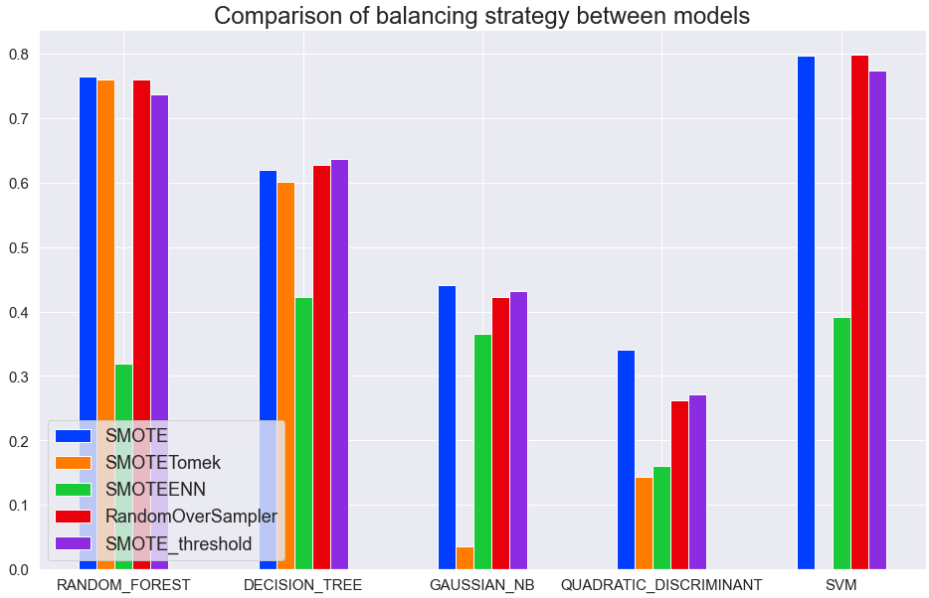


Figure 2.7: Dataset balancing comparison.

Starting from this results, the choosed balancing technique is the SMOTE, obtaining a class distribuition as showed in Figure 2.8

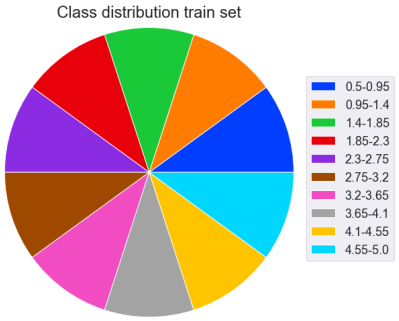


Figure 2.8: Class distribution with SMOTE.

2.3 Modeling

In order to train the proposed models with enhanced reliability the internal-external cross-validation is used. Each cross-validator has 5 folds and takes the classes distribution into account. As required by the assignment, the used models belong to the following categories:

1. Tree based
2. Naive Bayes

3. Support Vector Machine (SVM)
4. Multilayer Perceptron (MLP)

and for some of them, more than one model is used. The following lists show the set of hyperparameters used for each scikit model:

1.1 DecisionTree

- *criterion*: gini, entropy
- *max_depth*: [5, 10, 15]

1.2 RandomForest

- *n_estimator*: [100, 300, 500, 700, 900]
- *max_features*: sqrt, log2

2.1 GaussianNB

- *var_smoothing*: logspace(0, -9, num=100)

2.2 Quadratic Discriminant Analysis (QDA)

- *reg_param*: [0.00001, 0.0001, 0.001, 0.01, 0.1]
- *tol*: [0.0001, 0.001, 0.01, 0.1]

3.1 SVM

- *kernel*: rbf, poly, sigmoid
- *C*: [1, 10, 100]
- *gamma*: [1, 0.1, 0.01]

4.1 Neural Network

- *num_epochs*: 200
- *starting_lr*: 1e-3
- *batch_size*: [128, 256]
- *optim*: [Adam, SGD]
- *momentum*: [0.6, 0.9]
- *weight_decay*: [1e-5, 1e-7]

With regard to the Neural Network, it was necessary to define also the architecture:

- *input_act*: LeakyReLU
- *hidden_act*: LeakyReLU
- *hidden_size*: 512
- *num_hidden_layers*: [3, 5]
- *dropout*: [0.2, 0.3, 0.5]
- *batch_norm*: [False, True]
- *output_fn*: None

2.4 Performance Analysis

Bibliography

[1] GroupLens. MovieLens Latest Datasets. `https://grouplens.org/datasets/movielens/latest/`.

[2] TMDB Community. Api Overview. `https://www.themoviedb.org/documentation/api/`.

[3] IMDb.com. Imdb datasets. `https://www.imdb.com/interfaces/`.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.