Mathematics in Machine Learning

EEG Eye State dataset analysis

Gianvito Liturri
s290464

Professors: Francesco Vaccarino, Mauro Gasparini

Academic Year: 2021/2022

Summary

1 Introduction

One of the most interesting tools in our era is artificial intelligence. His potential is to transform the healthcare system as we know it, to a one where humans and machines work together in order to provide better treatment for the patients. The main goal is to improve the quality of prognostic and diagnostic models; it is useful to facilitate expert-level clinical decisions. The current analysis is about a real dataset, named EEG eye state [Roe13], related to continuous EEG measurement with Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. A camera records the eye state and subsequently the state was added manually to a file after analyzing the video frames.'1' indicates the eye-closed and '0' the eye-open state. All values are in chronological order with the first measured value at the top of the data.

2 Data exporation

2.1 Data overview and description

The dataset consists in 14980 records, each of them is characterized by 14 EEG measurements from the headset, originally labeled:

- AF3
- F7
- F3
- FC5
- T7
- P
- O1
- O2
- P8
- T8
- FC6
- F4
- F8
- AF4

All features are numerical.

2.2 Target distribution and statistics

In order to obtain a better understanding of the dataset, data exploration is a mandatory step of the machine learning pipeline. The first thing to check is the dataset distribution. We call "balanced" if the class labels have the same number of records instead "unbalanced", as in this case, if at least one class hasn't the same number of samples with respect to the other ones.

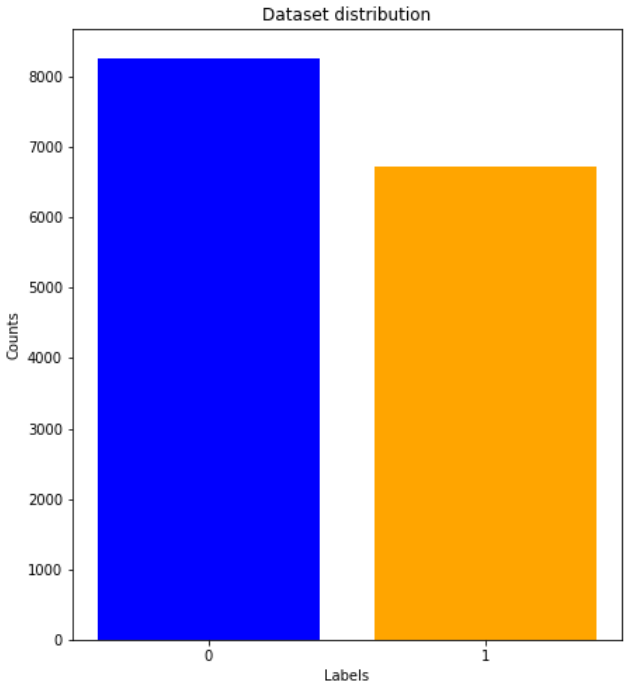


Figure 1: Labels distribution in the dataset.

The dataset is a little bit unbalanced. The number of samples that represents the open state of the eyes is 55% of the total, respectively the closed state is 45%. Since many machine learning algorithms work better with balanced data, we need certain methods to solve this gap. The dataset doesn't contain null values. These are some statistics about features:

	AF3	F7	F3	FC5	T7	P7	O1
count	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000
mean	4321.917777	4009.767694	4264.022433	4164.946326	4341.741075	4644.022379	4110.400160
std	2492.072174	45.941672	44.428052	5216.404632	34.738821	2924.789537	4600.926543
min	1030.770000	2830.770000	1040.000000	2453.330000	2089.740000	2768.210000	2086.150000
25%	4280.510000	3990.770000	4250.260000	4108.210000	4331.790000	4611.790000	4057.950000
50%	4294.360000	4005.640000	4262.560000	4120.510000	4338.970000	4617.950000	4070.260000
75%	4311.790000	4023.080000	4270.770000	4132.310000	4347.180000	4626.670000	4083.590000
max	309231.000000	7804.620000	6880.510000	642564.000000	6474.360000	362564.000000	567179.000000

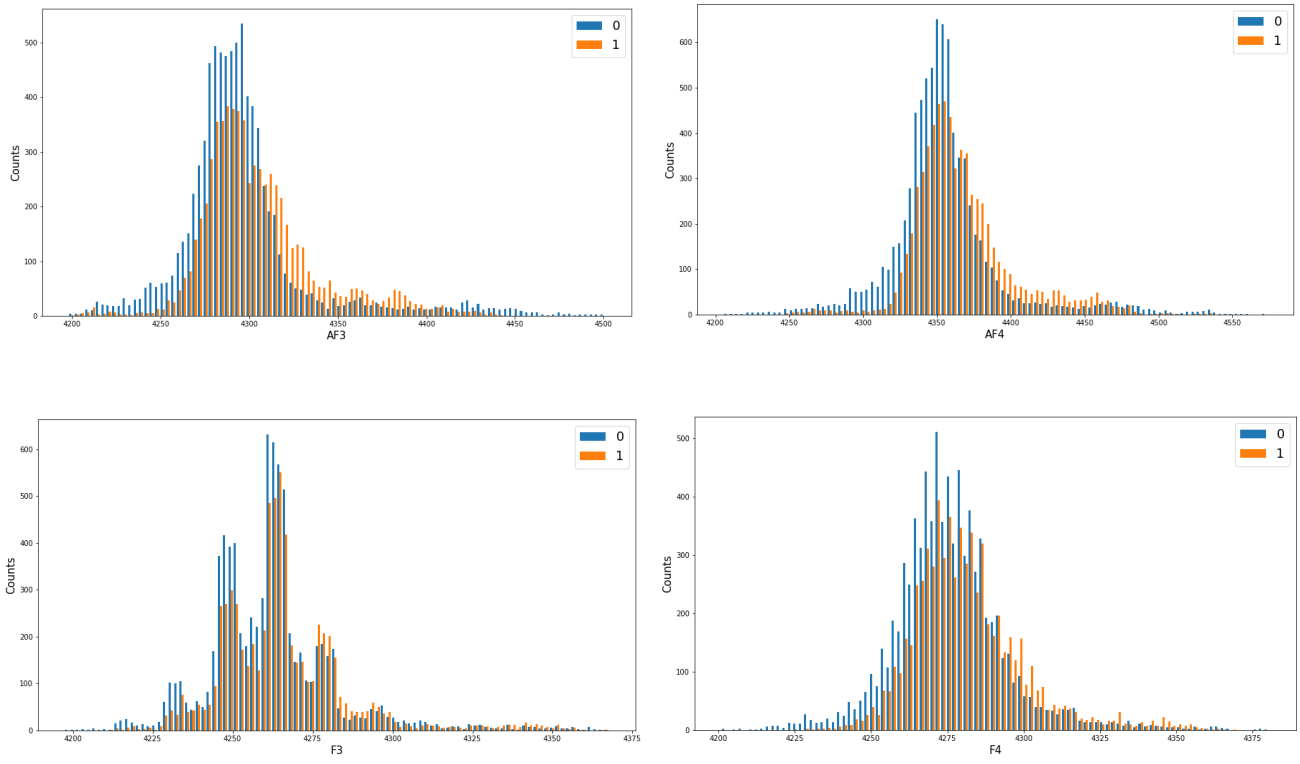
Figure 2: Statistics of the first seven features.

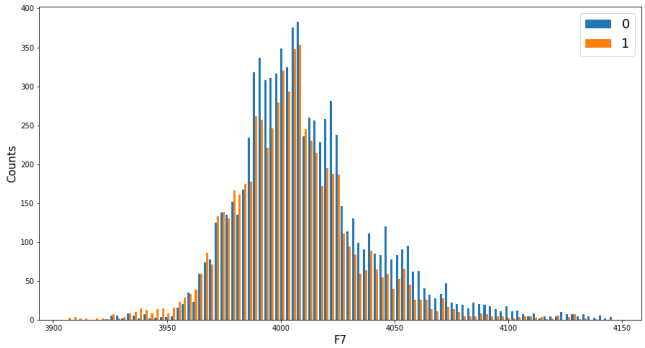
	O2	P8	T8	FC6	F4	F8	AF4
count	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000
mean	4616.056904	4218.826610	4231.316200	4202.456900	4279.232774	4615.205336	4416.435832
std	29.292603	2136.408523	38.050903	37.785981	41.544312	1208.369958	5891.285043
min	4567.180000	1357.950000	1816.410000	3273.330000	2257.950000	86.666700	1366.150000
25%	4604.620000	4190.770000	4220.510000	4190.260000	4267.690000	4590.770000	4342.050000
50%	4613.330000	4199.490000	4229.230000	4200.510000	4276.920000	4603.080000	4354.870000
75%	4624.100000	4209.230000	4239.490000	4211.280000	4287.180000	4617.440000	4372.820000
max	7264.100000	265641.000000	6674.360000	6823.080000	7002.560000	152308.000000	715897.000000

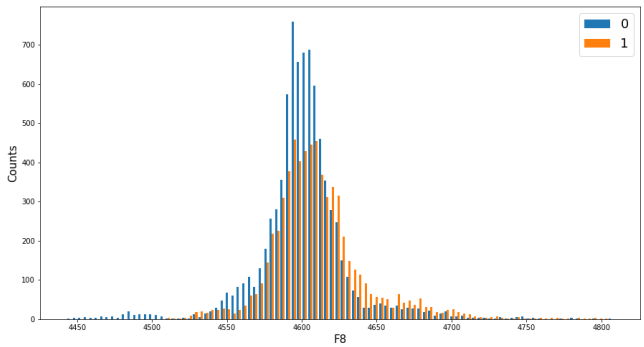
Figure 3: Statistics of the last seven features.

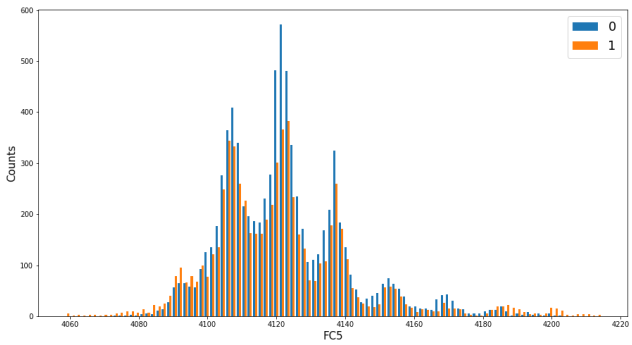
2.3 Domains and features distributions

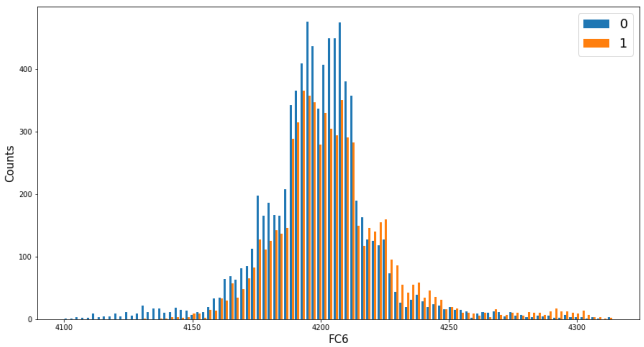
Further important analysis is about the features. We plot the distribution of each column group by class labels in order to compare the two eye states and display some differences. The chart chosen is the histogram.

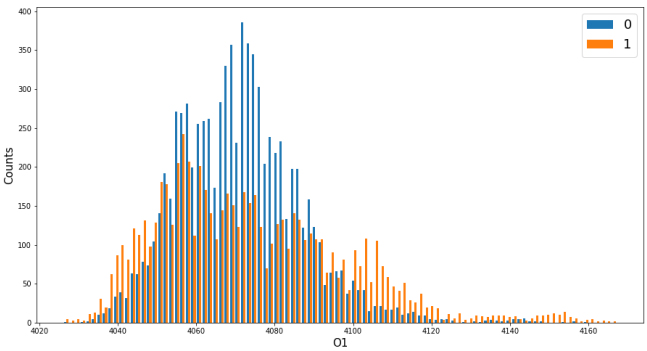


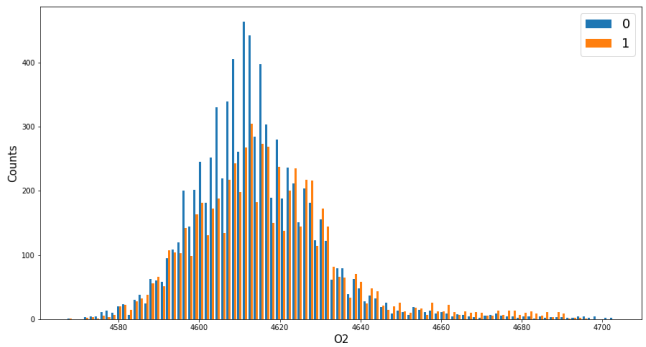

0
1
Counts
F7

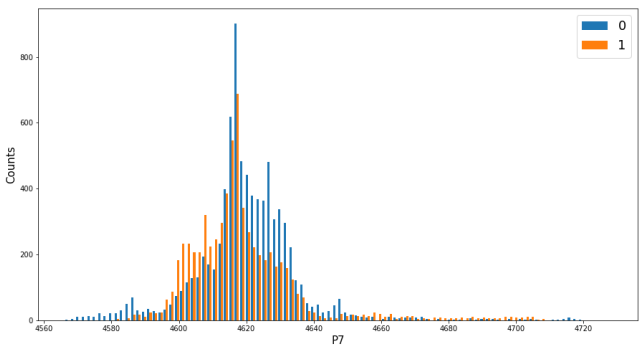

0
1
Counts
F8

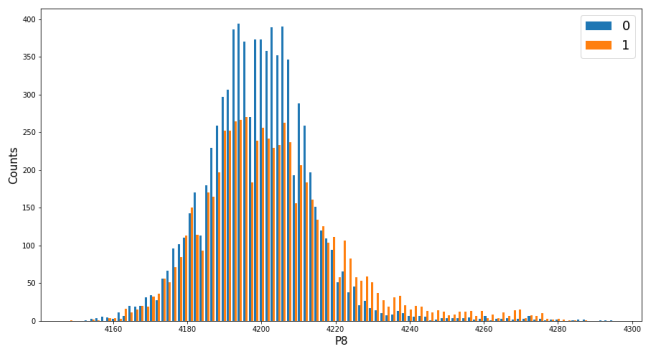

0
1
Counts
FC5

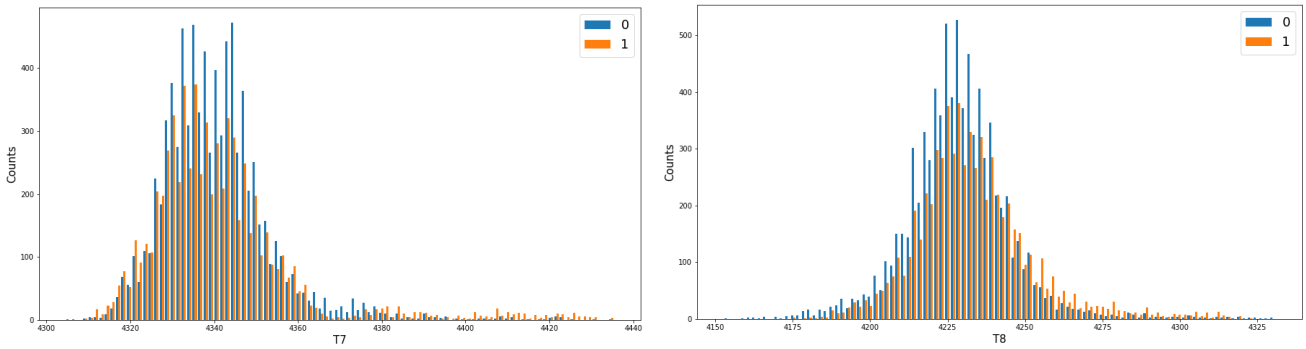

0
1
Counts
FC6


0
1
Counts
O1


0
1
Counts
O2


0
1
Counts
P7


0
1
Counts
P8



Analyzing the distributions we can notice a strange thing. In most of the features, it seems that the two Gaussian distributions are shifted in the orizontal ax. This is a useful detail due to, in extreme points, we are able to distinguish from one class to another.

2.4 Correlations

Another important analysis is the correlation among features.The correlations are calculated for each pair of features through the following formula: $cov(x, y)/\rho$. The confusion matrix is:

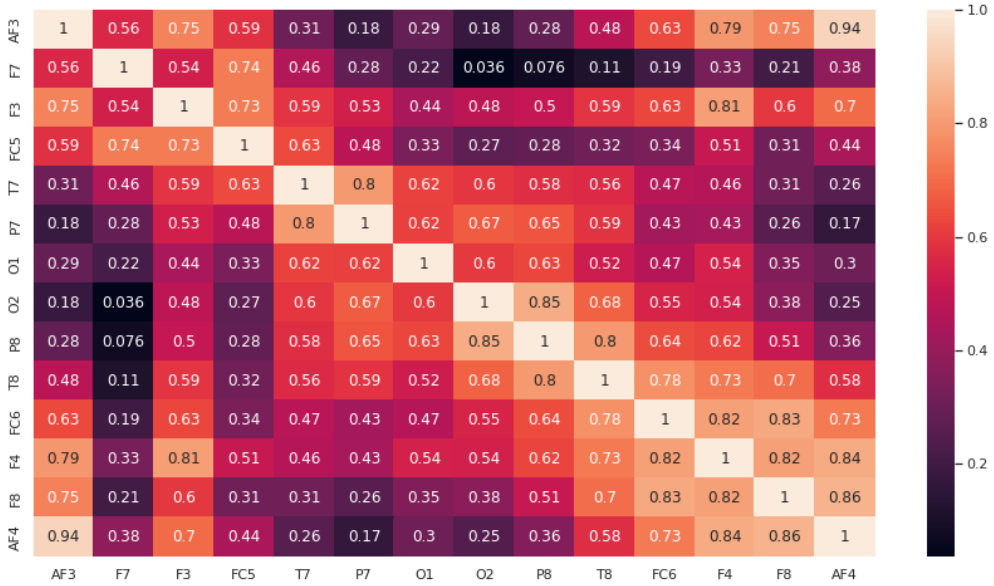


Figure 4: Correlation matrix.

As we can see, there are some pairs of features that have a correlation equal to one. In theory, it should not affect our ability to make predictions. We may have a problem if, in realistic situations, we have a finite number of samples. However,the

highly correlated feature can provide precious little pieces of information about class labels. This is proven if we compare results among models with or without highly correlated features.

3 Data preparation

3.1 Normalization

The first step into data preparation is data normalization. The features are transformed in other ones, in order to obtain the same range of values [0,1] for every column. It is a useful step since there are some machine learning methods that are able to predict better if data are normalized. The method chosen is the min-max normalization:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

3.2 Principal Component Analysis

The next step consists of dimensionality reduction. One of the most used methods is PCA (principal component analysis), an orthogonal linear transformation that rearranges features such that their number is reduced.The new generated attributes are called principal components. If all the attributes are independent, PCA is useless. Thanks to this, we can plot our distribution in two dimensions. The main properties of principal components are:

1. They are essentially linear combinations of original features.
2. They are orthogonal, among them.
3. Their variance decreases from the first to the last component, like their importance.

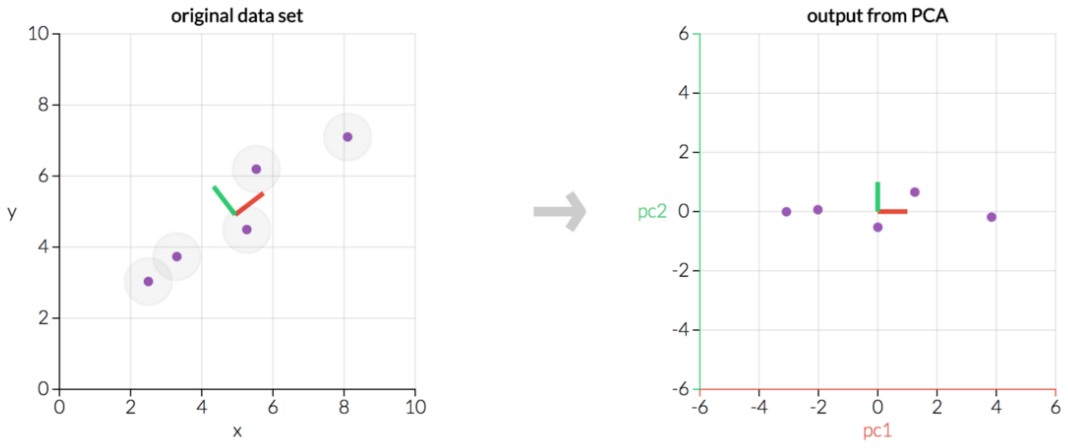


Figure 5: Principal Component Analysis.

Given this dataset, we do not apply dimensionality reduction because our features are only 14 and this approach could be counterproductive. This is the dataset representation by the usage of 3 principal component:

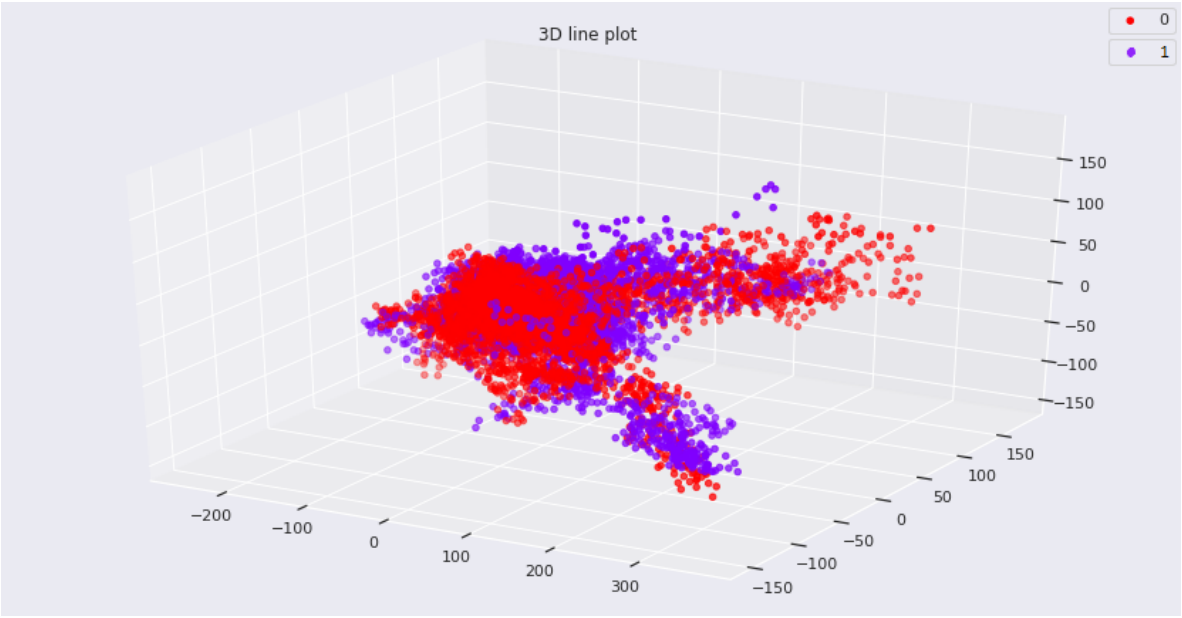


Figure 6: Dataset in 3 principal components.

3.3 Outliers management

The last step in data preparation is the outliers management. Outliers are unusual values in the dataset that can distort our analysis and consequently our predictions. We can manage outliers in two ways. The first one is to remove the entire sample where the error occurs; the second way is to replace the value with another one that does not represent an outlier (ex. the mean of that attribute) in order to keep the same distribution. The method used to find outliers is the z-score. It describes attribute values by how many standard deviation points are from the attribute mean. Once we define a certain threshold, each value is over, it will be removed. The threshold chosen is 3. In conclusion, we find 88 samples that contain at least one outlier. These samples will be discarded.

3.4 SMOTE

The dataset is a little bit unbalanced. This is not rare in real-word. An unbalanced dataset can provide some issues in the prediction task: overfitting and wrong predictions. That's why we must balance the dataset. There are two ways: oversampling and undersampling. Before applying one of these two methods, we keep in mind that rebalancing should be applied only on the training set, instead, the test set will be untouched. One of the most popular oversampling methods is SMOTE : Synthetic Minority Over-sampling Technique, described by Nitesh Chawla, et al. in their 2002 paper [CBHK02]. Given an unbalanced dataset with two classes. SMotE first selects the minority class samples and then creates new samples as a combination of the first point and his k neighbors on the high-dimensional lines that connect them:

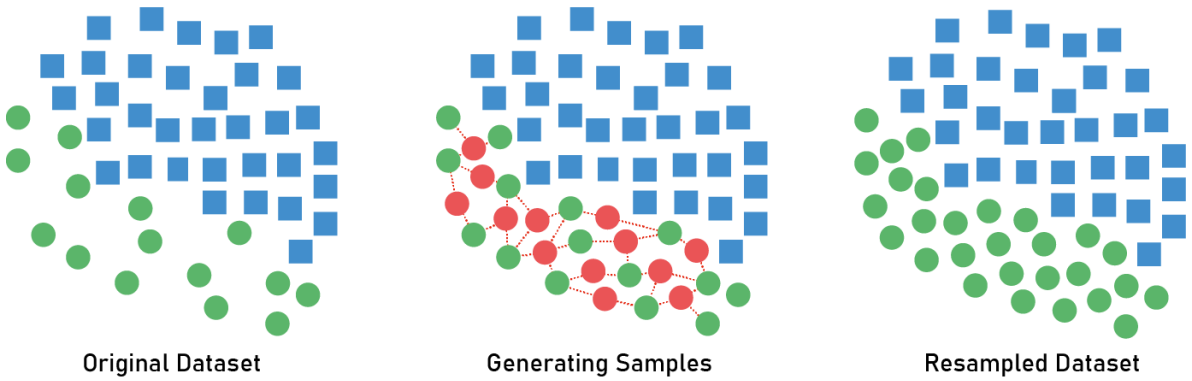


Figure 7: Synthetic Minority Oversampling Technique.

The second way to balance the dataset is by undersampling. In contrast to oversampling, this method rearranges the majority class, rather than the minority one, to obtain the same number of samples. The most popular technique to apply undersampling is based on cluster centroid. Starting from the majority classes, we compute the centroid, the mean point of the majority classes in the feature space, and remove those samples that are far from the centroid until the dataset is balanced. The biggest con of subsampling is the loss of huge useful information, that's why we don't use it. Undersampling techniques are useful only in the case that we have a large dataset and we afford to lose information.

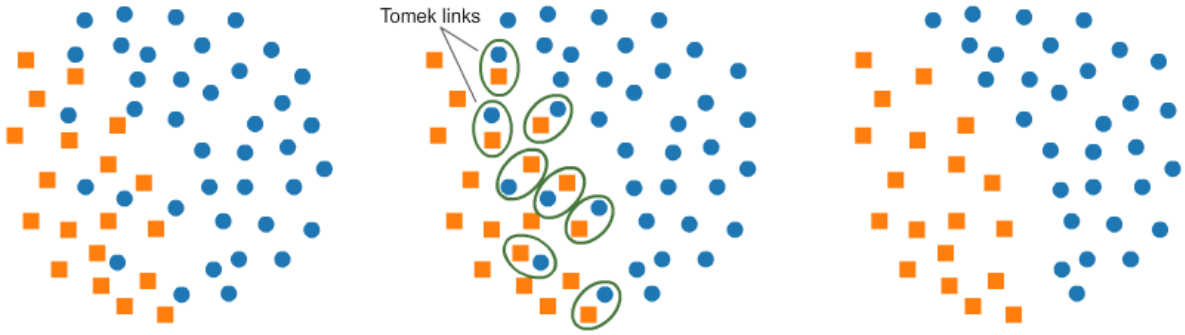


Figure 8: Underampling technique.

4 Metodology

4.1 Metrics

Performance metrics are a part of every machine learning pipeline. They are functions that we use in order to measure how well our machine learning methods work. In a few words metrics tell us if we are making progress or not. Classification Accuracy is the ratio between the number of correct predictions over the total number of predictions.

The biggest disadvantage of the accuracy metrics is that it works well only in the case we have a balanced dataset. In an unbalanced one,for example: medical datasets,

the accuracy gives us the false sense of achieving high accuracy. Other metrics that help us in this case, are :

- **Recall**: It gives us useful information. Recall describes how well the model performs on the minority class in order to fill the gap by accuracy metric.
- **F1 score**:it gives an interpretation as a harmonic mean of the precision(another metric) and recall. Its best value is 1 and the worst score is 0.

4.2 Cross validation

In order to obtain more accurate machine learning models and how they work on an independent test dataset. Cross-validation is a technique used to protect the model from overfitting, especially if the amount of sample is limited.In the process of cross-validation technique, the original dataset is divided into k subsets of samples with the same number of records. At each iteration, one of these k subsets is used as a validation set and the remaining ones as a training set. This step is repeated k times and for each of these ones it is computed the metrics. In conclusion, metrics are averaged. There is a special cross-validation called leave-one-out where the number of k subset of samples is equal to the total number of records. A useful parameter of cross-validation is the stratify option, in order to maintain the same proportion of different classes in both training and test sets. The cross-validation is computed at the same time in the hyperparameters tuning stage with k equal to 5.

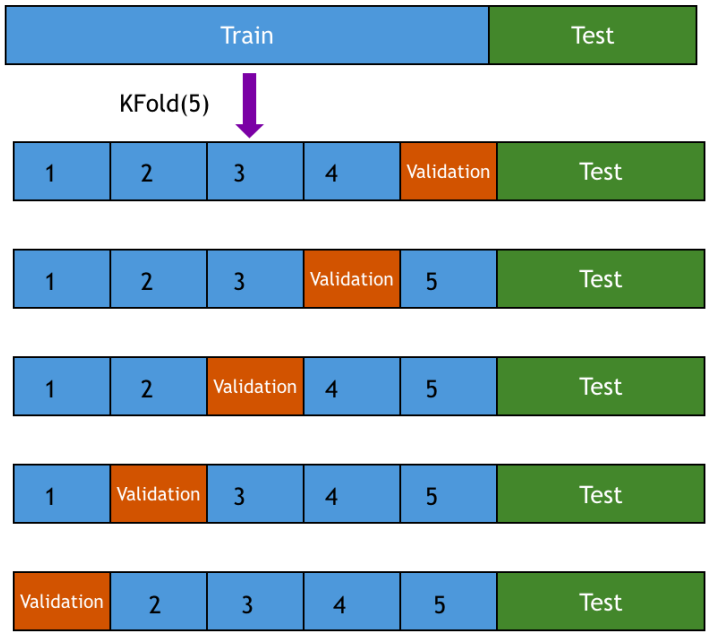


Figure 9: Cross validation with 5 K.

5 Classification models

5.1 Decision Tree

A decision tree is a supervised learning model,which is used for both classification and regression tasks. Its purpose is to predict the class or value based on a set of decision rules inferred from training data. This method consists of spitting the predictor space in non-overlapping regions, the label, that most occurs in a certain region, is assigned to the entire region. In the training phase, a decision tree uses a top-down greedy approach, top-down since the predictor space is split from the top of the tree, greedy because the splitting is performed to obtain the local best spit. The most common criteria used to obtain the best split are:

- **Gini Index**: $\sum_{k=1}^{K} p_{mk}(1 - p_{mk})$
- **Cross Entropy**:$-\sum_{k=1}^{K} p_{mk} \log(p_{mk})$

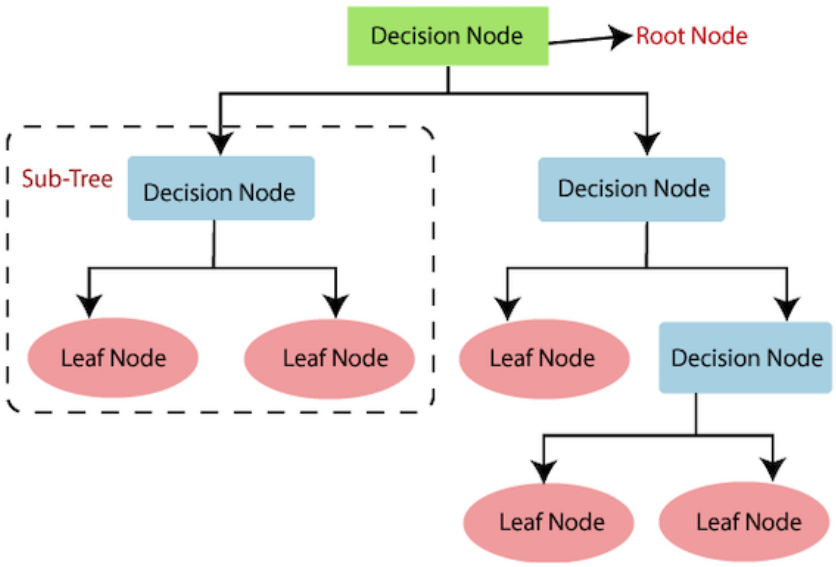


Figure 10: Decision tree

The good points of the decision tree are the simplicity and the interpretability of the model. On the other hand, the model could overfit on the test set. In order to solve this problem, an important hyperparameter is the maximum depth of the tree. To better generalize the problem is useful to use a small tree by stopping its growth. However, tree-based methods are not competitive with other supervised algorithms in terms of accuracy. For this reason, random forest is more used; it is based on combining different decision trees and produces an increasing of the accuracy.

In the hyperparameter tuning phase, these hyperparameters have been evaluated (in combination):

- **criterion**: ["gini", "entropy"]
- **depth**:[None, 10]
- **splitter**: ["best", "random"]

The best combination is: ['criterion': 'entropy', 'depth': None,'splitter': 'best'] The f1 score is: 0.836. The confusion matrix:

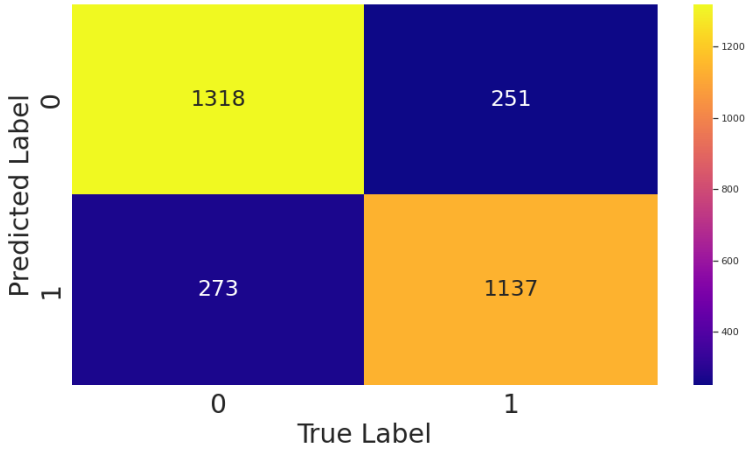


Figure 11: Confusion matrix.

5.2 Random forest

The random forest method is a supervised learning technique that uses decision trees by bagging. This method creates and trains different decision trees with a different set of bootstrapped training samples and choosing a random subset of all predictors as split candidates(typically $\sqrt{p}$). This allows us to better generalize the problem and obtain uncorrelated trees with smaller variances. In the prediction phase, the global prediction is computed by a majority vote on predictions of different trees.

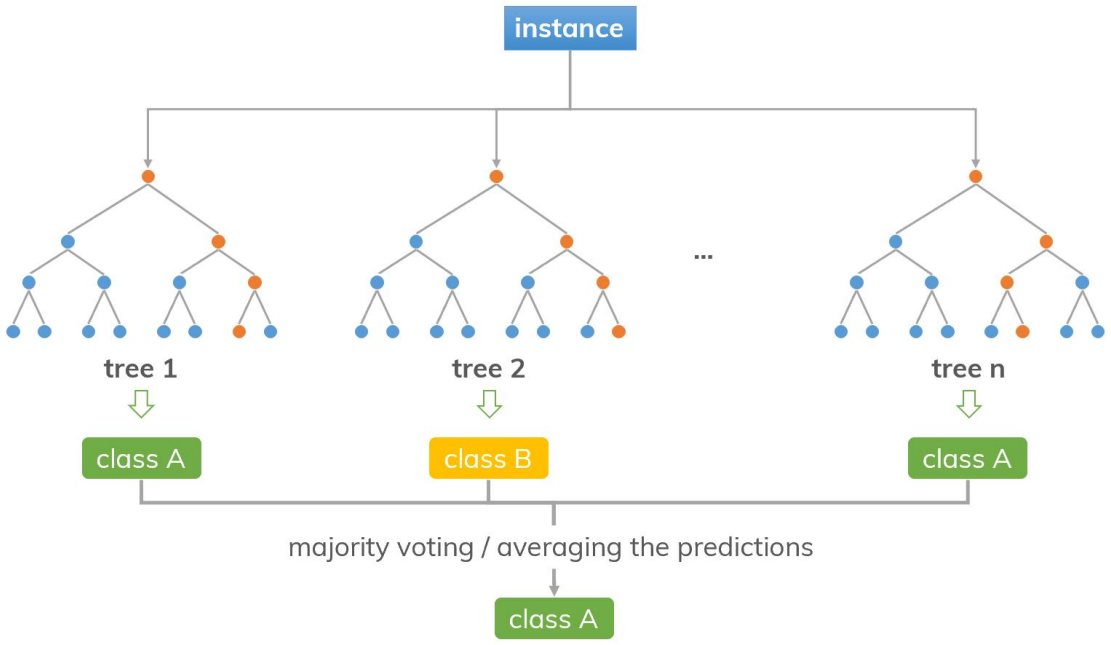


Figure 12: Random forest

In the hyperparameter tuning phase, these hyperparameter have been evaluated (in combination):

- **criterion**: [”gini”, ”entropy”]
- **n estimators**:[100,300,500]
- **max depth**: [None, ”sqrt”]

The best combination is:[’criterion’: ’gini’, ’max depth’: None, ’n estimators’: 500]
The f1 score is: 0.933. The confusion matrix:

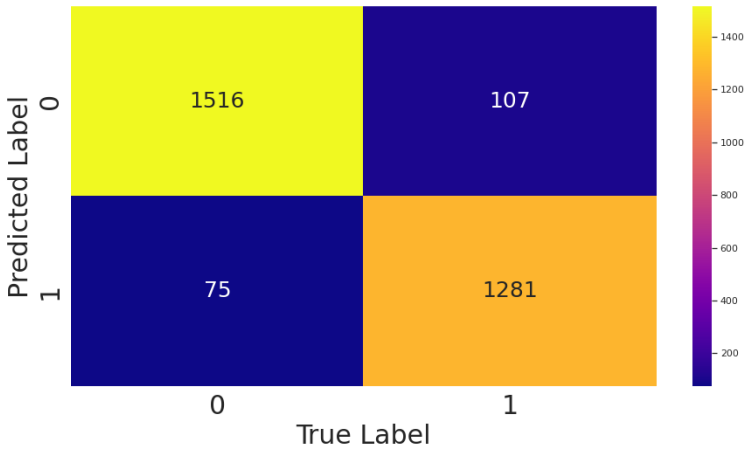


Figure 13: Confusion matrix.

5.3 SVM

Support Vector Machine is a supervised method and its purpose is to solve an optimization problem. The problem is: to find the hyperplane that maximizes the margin among classes in the feature space, in other words, the SVM finds the hyperplane with the biggest margins from the closest sample of each class. These points are called support vectors.

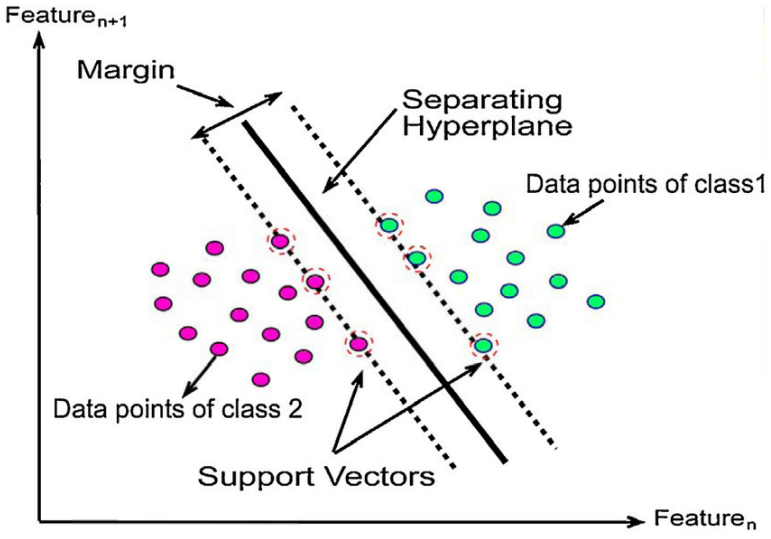


Figure 14: Support Vector Machine

Given an hyperplane:

$$f(X) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{2}$$

Beta is the variable that we want to find in order to maximize M (the margin) such that $y_i f(x_i) > 0 \vee i$ Sometimes, classes are not linearly separable, and so an alternative can be the soft margin problem in which a slack variable is used to allow some mistakes.

$$\min \frac{1}{2} w^t w + C \sum_i \max(0, 1 - y_i w^t x_i) \tag{3}$$

The hyperparameter C controls the tradeoff between a large margin and a small hinge loss. The first term of the function is a Regularization term that maximizes the margin and pushes for a better generalization: The second term is the empirical loss and it penalizes weight vectors that make mistakes.

Another possible solution for non-linear problems is to use the kernel trick. A kernel is a symmetric function that corresponds to a scalar product in a higher dimensional feature space in order to make the separation easier. Once we obtained the new space data could be linearly separable. Not all kernels are valid! A kernel is valid if it has two characteristics(Marcel Theorem): it is symmetric and the gram matrix $G_{ij} = K(x_i, x_j)$ is positive semidefinite. The f1 score is: 0.890. The confusion matrix:

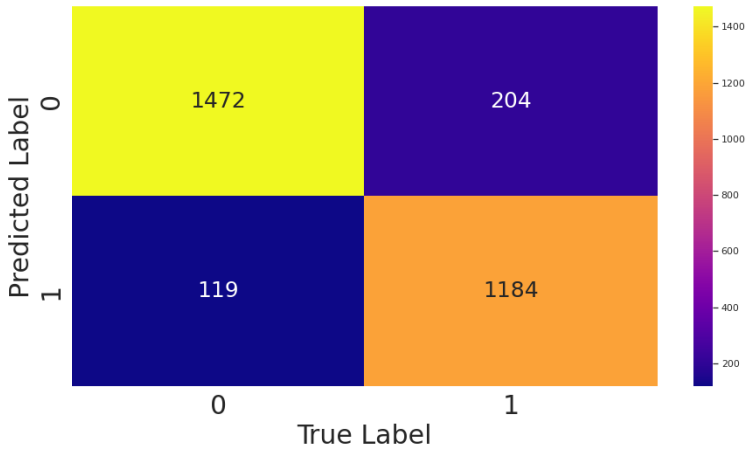


Figure 15: Confusion matrix.

5.4 K Nearest Neighbors

This model is called K-nearest neighbors. It is a type of supervised learning model used for regression and classification tasks. KNN predicts the correct class of a sample by computing the distance between the sample and the training points. The distance is computed generally by the Minkowski formula $(\sum_{i=1}^{D} |a_i - b_i|^p)^{1/p}$. Then, k training points are selected (the closest to the sample) and knn assigns the label to the sample by majority vote according to k training points labels.

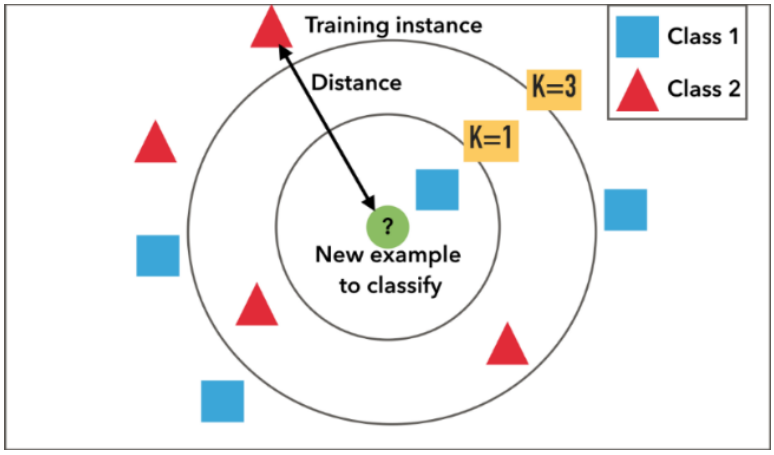


Figure 16: K-nearest neighbors

The K-nearest neighbor is easy to use and implement. A bad point is that for a large dataset is very expensive to compute all distances. The choice of K is very important since a small K can make the model more sensitive to outliers; in the opposite way, a big K can lead irrelevan training points into account. In the hyperparameter tuning phase, these hyperparameters have been evaluated (in combination):

- **n neighbors**: [5,10,15]
- **weights**:["uniform,"distance"]

The best combination is: ['n neighbors': 5, 'weights': 'distance']
The f1 score is: 0.96 The confusion matrix:

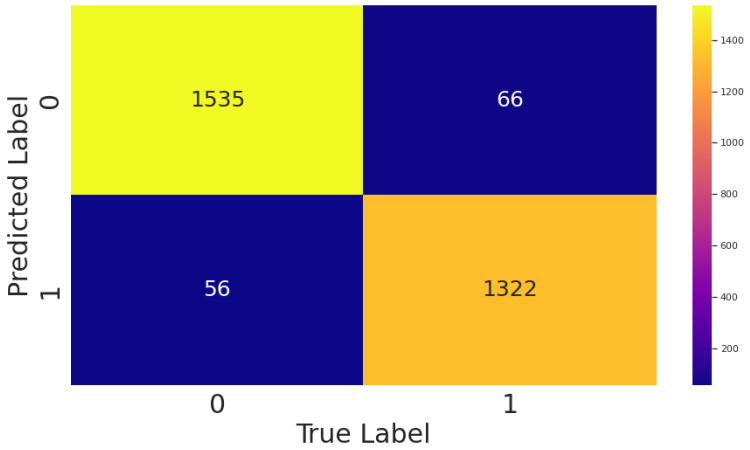


Figure 17: Confusion matrix.

5.5 Logistic regression

Based on probability, a logistic regressor is a regression model used for binary classification tasks. Since we want to map our results into 0,1 interval, we use the logistic function(sigmoid) and not the linear function. The logistic takes any value and gives

as an output a value between 0 and 1.

$$\log(\frac{p(X)}{(1-p(X)}) = \beta_0 + \beta_1 X \tag{4}$$

In order to estimate the B coefficients, maximum likelihood estimator is used.

$$l(\beta_0, \beta_1) = \prod_{i=0}^{n} p(x_i) \prod_{i=0}^{n} 1 - p(x_i) \tag{5}$$

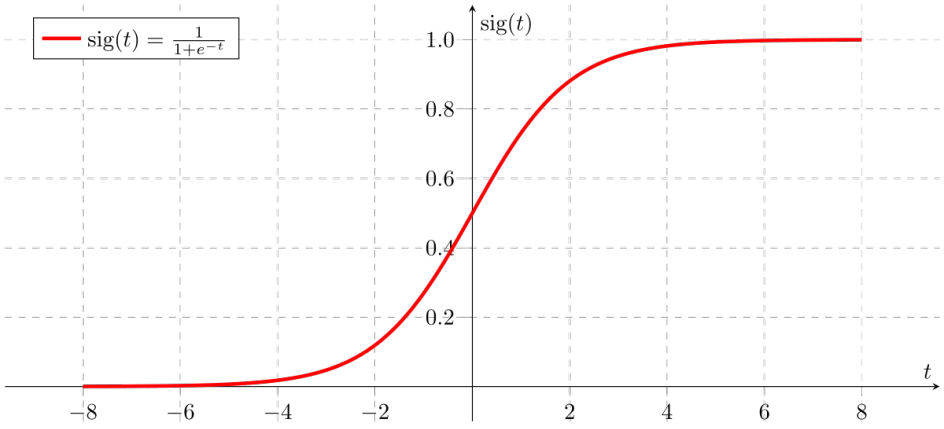


Figure 18: Logistic Regression

In conclusion, it is applied a threshold to the probability. For example: if the probability is less than 0.5, 0 will be assigned to the sample. In the hyperparameter tuning phase, these hyperparameters have been evaluated (in combination):

- **C**: [0.5,1]
- **penalty**:["l1","l2"]

The best combination is: ['C': 1, 'penalty': 'l2']
The f1 score is: 0.625 The confusion matrix:

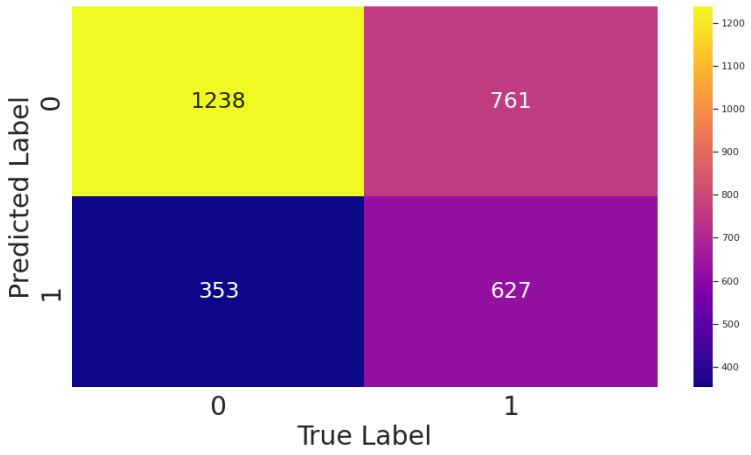


Figure 19: Confusion matrix.

6 Conclution

As we can see in the previous section, the best models are the Random Forest with 0.933 f1 score and the K-nearest neighbors with 0.96 f1 score. Other models with comparable accuracy are the svm with 0.89 f1 score and the decision tree with 0.84 f1 score. The worst model is the Logistic Regressor with 0.63 f1 score.

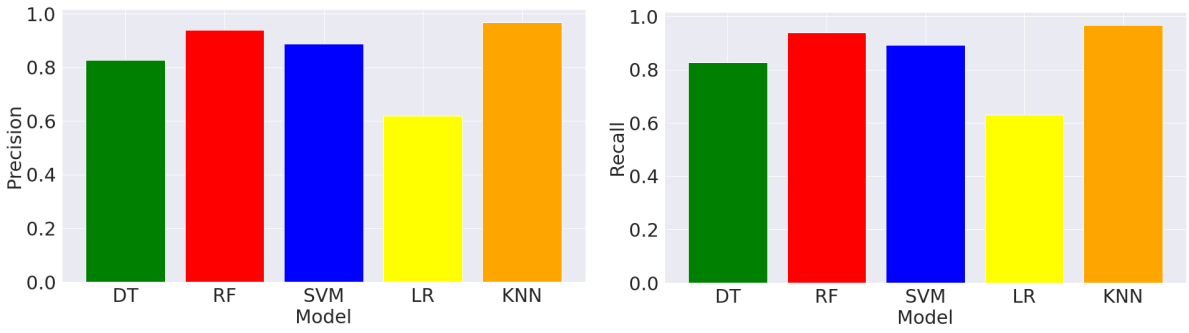


Figure 20: Precision and Recall for each model.

For completeness, the figures represent the precision and recall for each model. As we can see the precision and recall are similar for each model and similar to the f1 score related to each model.

References

[CBHK02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, jun 2002.

[Roe13] Oliver Roesler. Eeg eye state data set. https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State, June 2013.