Ionosphere prediction task

Matteo Cerutti

265476

Politecnico di Torino

Data spaces

Exam relation

14-07-2021

Indice

1. Introduction
2. Dataset
    a. Dataset balancing
    b. Attribute distribution
    c. Attribute correlation
3. Data preprocessing
4. Classification
    a. Cross validation
    b. Evaluation metrics
        i. Area under curve
        ii. Classification accuracy
        iii. F1 score
    c. Trained algorithms
        i. Logistic regression
        ii. Decision tree
        iii. Random forest
        iv. kNN
        v. SVM
            1. Linear SVM
            2. Kernel SVM
                a. Polynomial SVM
                b. RBF SVM
5. Conclusion
6. Additional tables and figures
7. Bibliography

1. Introduction

The relation proposed in the following document aims to describe a classification task based on the dataset *Ionosphere*.
The dataset contains different radar signal data collected by a system in Goose Bay, a location in the Labrador province in the state of Canada. The collection target was to measure free electrons in the ionosphere. For this reason the data were labeled with two possible values: *good* or *bad* records, based on the showing evidence of some structure passing through the ionosphere.
The objective of the classification task is to find a good classifier for distinguishing between the *good* or the *bad* records belonging to the result of this type of measurement, analyzing different classification algorithms.
The relation starts describing the dataset through its attributes, continuing with the data preprocessing methods used, the classification algorithms trained on the dataset, and it concludes with the consideration of classification results.
All the data preprocessing, training and testing procedures are made using the software *Orange*.

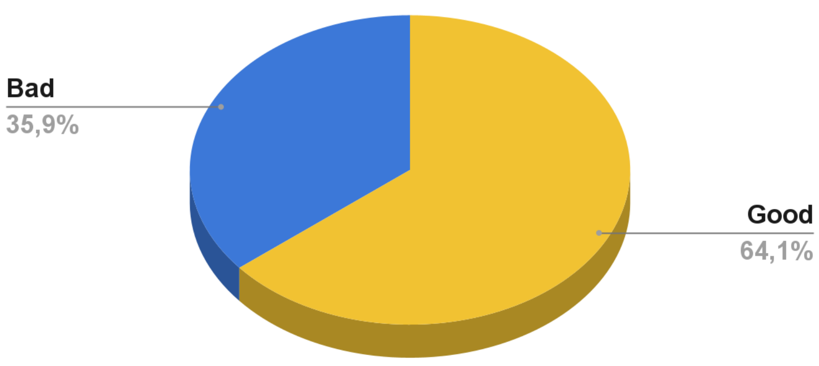
2. Dataset

The dataset is a set of radar signal data collected in the Goose Bay from a phased array of 17 high-frequency antennas. The dataset is composed of 351 instances. The received signals were processed using an autocorrelation function in order to extract the pulse number, which is a complex one composed of two values.
For that reason the attributes are 34, with 1 target attribute.
The values contained in the attributes are all continuous, in an interval of -1 and 1.
The target attribute has only two possible values: *good* (g) or *bad* (b).
There aren't missing values in the dataset rows.

a. Dataset balancing

The dataset is unbalanced between the good and bad classes: the instances are for the 64,1% of the good one, and of the remaining 35,9% of the bad one. For that reason, in data preprocessing the oversampling technique is used in order to balance the training dataset.

Figure 1: Ionosphere target distribution



b. Attribute distribution

All the attribute values are in the interval between -1 and 1.
Observing the distribution of the attributes, collected in Table 1, it's possible to extract two important considerations:

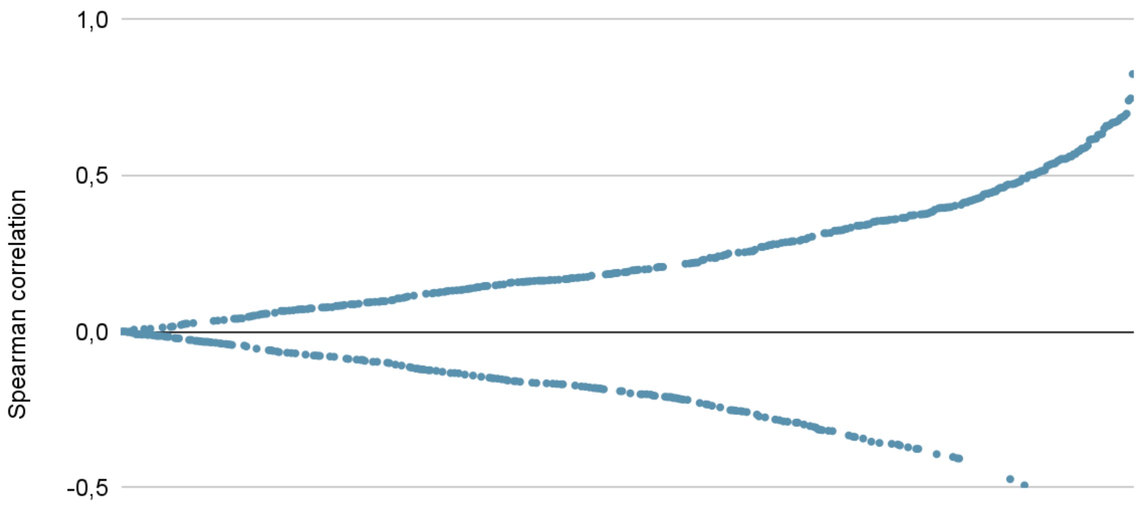
- The first one is that the attribute X.1 has only one value in all the dataset, that is 0. As a consequence this attribute is excluded from the classification task, because it doesn't contain any useful information for the classifiers.
- The second one regards the outliers. It's possible to observe that some distributions of data are centered not near 0, that is the medium value between -1 and 1, but their mean is near one of the two interval limits (for example the attributes X.2, X.4). But the instances whose attribute value is far from the mean are not in a limited number; in particular they are concentrated on the fairest interval limit. For that reason these instances aren't excluded from the dataset, despite that they could be considered as outliers.

In Figure 2 is possible to observe the graphs of each attribute distribution.

c. Attribute correlation

The correlation between the attributes of the dataset is evaluated using the Spearman correlation.
Observing the graph in Figure 3, it's possible to deduce that there aren't attributes greatly correlated: there is only one value greater than 0,8, which is the correlation value of the features *X.12* and *X.14*.
Therefore none feature is removed from the classification attributes considering the correlation with other ones.

Figure 3: Attribute correlation graph



3. Data preprocessing

The first step about the data preprocessing is the removal of X.1 from the attributes of the dataset.
After this step the instances are splitted in two parts: the training and test sets. The splitting percentage is 70% of instances in the training set, the remaining 30% in the test one. This splitting is stratified, meaning that the distribution of instances on the classies is maintained.
The sequent phase is the rebalancing of the training set, considering the number of instances per class. In fact at this point there are 158 instances with *good* label and 88 with *bad* label. The technique used is the oversampling: a random picking from

the set of bad labeled instances with replacement. In this way in the result set there are repeated instances. Considering that the missing instances are 70, this method is worth increasing the knowledge about the bad labeled instances in the training set, without having too much repetition of the same instance and consequently not giving new knowledge to the classifiers.
The final step is to concatenate the good labeled set and the oversampled bad set in order to have the complete training set.
At the end there are two sets with different distributions on the classies: the test one that maintains the original distribution and the training one with a balanced distribution.

4. Classification

a. Cross validation

In order to evaluate the best hyperparameters of each algorithm used for the classification task, the k-fold cross validation method is used.
The base idea of this method is to randomly divide the training set in k equal-sized subsets, and use in an iterative way k-1 subsets for training and the remaining one for the validation, until the k-th cycle. The final phase is to combine the validation results in an averaged way.
With this method it's possible to evaluate the hyperparameter values that give the best combination of evaluation metrics, without the necessity of testing a bigger quantity of data as in the test set. Moreover the evaluation on different k subsets permits to avoid the problem of a bad splitting of data.
For this classification task a 5-fold cross validation is used.

b. Evaluation metrics

In order to evaluate the best hyperparameters in the cross validation and the classifier goodness in the result analysis, different metrics are used: area under curve (AUC), classification accuracy (CA) and F1 score.

All these metrics are important, considering the fact that the dataset is not balanced. In fact the classification accuracy is less significant in evaluating classification results of an unbalanced dataset, so it’s important to consider also more resilient metrics like AUC or F1 score.

i. Area under curve

The area under curve is the measure of the area between the horizontal axis and the ROC curve (Receiver Operating Characteristics).

The ROC is a probability curve whose scope is to show a measure of separability of the analysed classifier. On the vertical axis there is the True Positive Rate (TPR) or Sensitivity, meaning the division between the number of correctly classified positive class’ points and the number of positive class’ points. On the horizontal axis there is the False Positive Rate (FPR), meaning the division between the number of wrongly classified negative class’ points and the number of negative class’ points.

$$FPR = \frac{FP}{FP+TN}$$

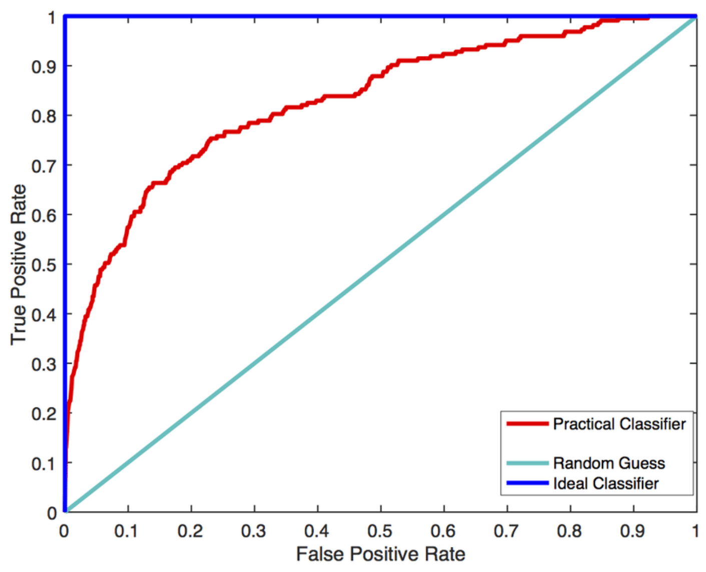
referring to Table 2

The perfect result is to have 0 as FPR and 1 as TPR, meaning a value of 1 for AUC. For that reason the best classifiers, in terms of class separability, are the ones whose AUC is near 1.

Table 2: Classification metrics’ matrix

		Predicted class		
		Positive	Negative	
Actual class	Positive	**True positive (TP)**	**False Negative (FN)**	Sensitivity $\frac{TP}{TP+TN}$
	Negative	**False positive (FP)**	**True Negative (TN)**	Specificity $\frac{TN}{TN+FP}$
		Precision $\frac{TP}{TP+FN}$	Negative Predictive Value $\frac{TN}{TN+FN}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$

Figure 4: ideal, random and example ROCs



ii. Classification accuracy

The classification accuracy is the most intuitive metric. It's calculated by dividing the number of correctly classified points and the number of set's points.
Thus it's a measure of the capability of the model in correctly classifying the test instances.
But this measure is particularly influenced by the distribution of the dataset, because assigning in a random way the predictions, it's probable to achieve the distribution percentage of the more representative class as value of accuracy. So if the dataset is greatly unbalanced, the accuracy can have a high value despite a not so performing classifier.

iii. F1 score

The F1 score is the harmonic mean between the precision, meaning the division between the number of correctly classified positive class' points and the number of positive classified points, and the recall (TPR, see Table 2).
This metric permits to measure the precision of the classifier in the predictions and its robustness.

$$F1 = 2 \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

referring to Table 2

c. Trained algorithm

In the following subchapters are presented different algorithms trained with the selected training set in order to find the best one for the classification task, evaluating them with the test set.
For each algorithm, after a short theoretical description, a table is presented with the grid search of the best hyperparameter, whose values are enumerated at the beginning row and column. In the middle of the table are indicated the metric measurements, extracted from the 5-fold cross validation, using the hyperparameter values indicated by their coordinates.
Finally the test metric measurements are indicated, using the best combination of hyperparameter values extracted by grid search.

i. Logistic regression

The logistic regression is a statistics method for binary classification problems. It's based on the logistic function that determines the probability of belonging to one of the classes. Based on a threshold, usually 0,5, a label is determined: one when y is less than the threshold, the other one when is higher.
The base formula for several variables (attributes) is:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + ... + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + ... + \beta_p X_p}}$$

This method can be applied also to a multiclass classification problem, having a set of functions for each class, whose denominator is the sum of the characteristic class exponential.
In order to find the best coefficients for the logistic function, a regularized loss one is used, composed of two elements, summed together: a loss function and a penalization.

The penalization depends on the two hyperparameters: the regularization type (RT) and the cost (C), that influence the strength of the penalization.

Table 3: Logistic regression grid search

		Regularization type (RT)					
		L1			L2		
		AUC	CA	F1	AUC	CA	F1
C	**0,001**	0,5	0,5	0,333	0,805	0,737	0,737
	0,01	0,5	0,5	0,333	0,846	0,804	0,802
	0,1	0,857	0,801	0,796	0,907	0,813	0,812
	1	0,911	0,848	0,847	0,922	0,826	0,825
	10	0,929	0,848	0,848	0,924	0,854	0,854
	100	**0,938**	**0,88**	**0,88**	0,934	0,873	0,873
	1000	0,937	0,886	0,886	0,938	0,88	0,88

Table 4: Logistic regression testing evaluation measures

Best RT	Best C	AUC	CA	F1
L1	100	0,877	0,867	0,865

ii. Decision tree

The base idea of the decision tree model is to divide the predictor space in high-dimensional rectangles, called boxes. Each point in a specific region is labeled with its barycenter for regression problems or with the same region class for classification ones.
The process to find the boxes is greedy: for each step the best split at the moment is made, based on a specific metric, until a stopping criterion is achieved.

The most common metrics used in a classification task to find the best attribute for splitting in each step are: the *Gini index*, that measures the total variance across the classes, and the *cross-entropy*, that measures the "disorder" through the regions.

Having a full tree, meaning a single point in each region, can be problematic due to the overfitting of the model. For that reason we can have two possible approaches: stop the tree definition before the end with a specific criterion, or pruning the full tree after, meaning considering trees with a determined number of leaves.

The hyperparameters considered for this grid search are two: the majority limit, meaning to not split again when the majority class distribution percentage achieves the limit, and the depth limit, a threshold on the depth of the tree.

Table 5: Decision tree grid search

		Majority limit (ML)											
		100%			**95%**			**90%**			**80%**		
		AUC	CA	F1	AUC	CA	F1	AUC	CA	F1	AUC	CA	F1
DL*	**3**	0,932	0,88	0,879	0,932	0,88	0,879	0,932	0,88	0,879	0,902	0,88	0,879
	5	0,943	0,911	0,911	0,94	0,918	0,918	0,939	0,902	0,902	0,907	0,883	0,883
	7	0,918	0,911	0,911	0,91	0,918	0,918	0,924	0,905	0,905	0,902	0,886	0,886
	8	0,918	0,918	0,918	**0,911**	**0,921**	**0,921**	0,924	0,905	0,905	0,902	0,886	0,886

*DL: depth tree limit

Table 6: Decision tree testing evaluation measures

Best DL	**Best ML**	**AUC**	**CA**	**F1**
8	95%	0,929	0,914	0,912

iii. Random forest

The random forest is a model based on the decision tree. The base concept is to train multiple trees based on the bagging method, meaning on a random subset of

the original training set. In order to improve the variability of the prediction, each tree is trained on a random subset of the attributes. Usually the square root of total attributes for each attribute subset is used.
In fact in the following grid search 6 attributes are used for each subset, considering an excess approximation of the square root of 34 dataset attributes.

The hyperparameters considered for the grid search are the number of trees trained in the random forest (NT) and the depth limit of each tree.

Table 7: Random forest grid search

		Number of trees (NT)														
		10			50			100			500			1000		
		AUC	CA	F1	AUC	CA	F1	AUC	CA	F1	AUC	CA	F1	AUC	CA	F1
	2	0,943	0,889	0,889	0,957	0,892	0,892	0,959	0,905	0,905	0,96	0,899	0,898	0,958	0,902	0,902
	3	0,969	0,911	0,911	0,972	0,911	0,911	0,973	0,902	0,902	0,973	0,908	0,908	0,973	0,908	0,908
DL*	4	0,979	0,918	0,918	0,98	0,927	0,927	0,98	0,927	0,927	0,982	0,93	0,93	0,981	0,924	0,924
	5	0,983	0,937	0,937	0,985	0,953	0,953	0,985	0,946	0,946	0,987	0,943	0,943	0,987	0,943	0,943
	6	0,984	0,943	0,943	0,989	0,946	0,946	0,99	0,953	0,953	0,99	0,949	0,949	0,99	0,949	0,949
	7	0,987	0,962	0,962	**0,992**	**0,956**	**0,956**	0,993	0,953	0,953	0,992	0,956	0,956	0,992	0,956	0,956

*DL: depth tree limit

Table 8: Random forest testing evaluation measures

Best DL	Best NT	AUC	CA	F1
7	50	0,979	0,943	0,942

iv. kNN

The k Nearest Neighbor is a classification model based on the simple idea of assigning to a point the majority label between the k nearest points for classification tasks, or the average y for the regression ones.

For this reason the concept of distance is important: the used metric influences the training and the prediction ability of the trained model.
The other hyperparameter is k, that determines the subset considered to assign the label to the points.
The last element considered, that distinguish the grid search in Table 9 from the one in Table 10, is the weight used in the label assignment: with a uniform one, all the k points have the same importance in the majority voting or average calculation, instead with a distanced weight, the importance is determined following the distance between the classifying point and the labeled ones, meaning that the nearest point is more influencing respecting a farther one.

Table 9: kNN grid search with uniform weight

		Distance metric					
		Euclidean			**Manhattan**		
		AUC	**CA**	**F1**	**AUC**	**CA**	**F1**
k	**1**	0,921	0,921	0,921	0,953	0,953	0,953
	3	0,945	0,867	0,877	0,963	0,889	0,889
	5	0,948	0,848	0,847	0,957	0,854	0,853
	7	0,944	0,845	0,842	0,951	0,858	0,856
	15	0,938	0,81	0,806	0,942	0,813	0,81
	25	0,891	0,788	0,781	0,91	0,801	0,795
	41	0,866	0,782	0,774	0,882	0,782	0,774

Table 10: kNN grid search with distanced weight

		Distance metric					
		Euclidean			Manhattan		
		AUC	CA	F1	AUC	CA	F1
k	1	0,921	0,921	0,921	0,953	0,953	0,953
	3	0,954	0,921	0,921	0,972	0,94	0,94
	5	0,964	0,921	0,921	0,972	0,934	0,934
	7	0,963	0,93	0,93	**0,973**	**0,943**	**0,943**
	15	0,969	0,934	0,933	0,976	0,93	0,93
	25	0,965	0,921	0,921	0,976	0,94	0,94
	41	0,963	0,915	0,914	0,972	0,921	0,921

Table 11: kNN testing evaluation measures

Best weight	Best k	Best metric	AUC	CA	F1
Distanced	7	Manhattan	0,957	0,905	0,902

v. SVM

The Semi Vector Machine algorithm has as basic concept the idea of finding the hyperplane able to divide the points in their belonging class.
This solution is known as hard margin one, because the hyperplane has to divide all points correctly and with the best margin possible. But in most of the cases, that solution is infeasible.
The optimization formula is:

$$argmin_{(w,b)}||w||^2 \; subject\ to\ y_i(< w, x_i >+ b) \geq 1 \; \forall\, i$$

For the previous reason the algorithm is modified in order to have a more flexible solution: the soft margin. In this way the algorithm can misclassify some of the points, putting them on the wrong side of the hyperplane.

$$argmin_{(w,b)}(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i)\ subject\ to\ y_i(< w, x_i >+ b) \geq 1 - \xi_i \,\forall\, i,\ \xi_i \geq 0$$

The cost (C) is a hyperparameter that permits to control the number misclassified points, called slack variables, in order to find the best feasible classifier.

$$\exists C \mid \sum_{i=1}^{m} \xi_i \leq C$$

1. Linear SVM

Table 12: Linear SVM grid search

		AUC	CA	F1
C	**0,1**	0,914	0,867	0,866
	1	0,915	0,864	0,864
	10	0,916	0,867	0,867
	50	**0,922**	**0,886**	**0,886**
	100	0,924	0,873	0,873
	500	0,930	0,889	0,889

Table 13: Linear SVM testing evaluation measures

Best C	AUC	CA	F1
50	0,858	0,886	0,883

2. Kernel SVM

The limit of linear SVM is to not perform well with no linearly separable problems. For that reason it’s possible to apply the kernel trick: we use the scalar product of a higher dimensional Hilbert space, without actually moving the points in that new

space. In this way the decision boundaries are not only linear, but can have any type of curve, depending on the kernel used in the algorithm.
Thus the choice of the kernel becomes one of the important factors to be considered in classification tasks.

a. Polynomial SVM

In this particular case, for the kernel trick is used a polynomial function, whose degree is one of the tuned hyperparameters in the cross validation.
The other hyperparameter is C, the cost, characteristic of the soft margin SVM.

Table 14: Polynomial SVM grid search

		C																	
		0,1			**1**			**10**			**50**			**100**			**500**		
		AUC	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**
d	**2**	0,97	0,725	0,702	0,976	0,829	0,824	**0,966**	**0,927**	**0,927**	0,957	0,93	0,93	0,946	0,937	0,937	0,914	0,905	0,905
	3	0,936	0,68	0,644	0,961	0,791	0,782	0,964	0,899	0,898	0,963	0,911	0,911	0,969	0,908	0,908	0,934	0,915	0,914
	4	0,973	0,658	0,613	0,976	0,728	0,706	0,983	0,835	0,831	0,971	0,877	0,875	0,964	0,905	0,905	0,961	0,915	0,914
	5	0,922	0,661	0,618	0,948	0,706	0,678	0,958	0,807	0,799	0,952	0,842	0,838	0,949	0,854	0,852	0,942	0,899	0,898

Table 15: Polynomial SVM testing evaluation measures

Best C	**Best d**	**AUC**	**CA**	**F1**
10	2	0,915	0,914	0,911

b. RBF SVM

The RBF SVM is based on a RBF kernel, meaning Radial Basis Function. This function is the exponential of the euclidean distance between the coordinates of the first space and the second one.

$$K(x,\ x') = e^{-\gamma||x-x'||^2}$$

For that reason can be seen as a measure of similarity: when the distance is high, the result tends to 0, instead when it is low, the value is near 1.
γ is a free parameter that influences the amplitude of the curve. As a consequence, it becomes a hyperparameter in the training phase to be determined by the cross validation.
The other hyperparameter is C, the cost, characteristic of the soft margin SVM.

Table 16: RBF SVM grid search

		C																	
		0,1			**1**			**10**			**50**			**100**			**500**		
		AUC	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**	**AUC**	**CA**	**F1**
γ	**0,01**	0,92	0,845	0,845	0,976	0,908	0,908	0,986	0,946	0,946	0,985	0,953	0,953	0,979	0,943	0,943	0,979	0,937	0,937
	0,1	0,961	0,797	0,791	0,983	0,953	0,953	0,996	0,953	0,953	**0,997**	**0,962**	**0,962**	0,997	0,962	0,962	0,997	0,962	0,962
	1	0,857	0,617	0,551	0,976	0,864	0,862	0,977	0,864	0,862	0,977	0,864	0,862	0,974	0,864	0,862	0,983	0,864	0,862
	5	0,487	0,544	0,543	0,908	0,864	0,861	0,915	0,864	0,861	0,913	0,864	0,861	0,91	0,864	0,861	0,917	0,864	0,861
	10	0,41	0,503	0,499	0,893	0,858	0,855	0,89	0,858	0,855	0,889	0,858	0,855	0,883	0,858	0,855	0,905	0,858	0,855

Table 17: RBF SVM testing evaluation measures

Best γ	**Best C**	**AUC**	**CA**	**F1**
0,1	50	0,976	0,933	0,932

5. Conclusion

Observing Table 18, it's possible to conclude that the best algorithms are the RBF SVM and the random forest, considering also a good prediction performance given by the logistic regression.
One motivation can be that the problem is not very linearly separable, making better performance for algorithms with non-linear decision boundaries like the best ones. This can also explain the worst performance given by the linear SVM.

Considering the poor performance of the kNN, respecting other algorithms, the motivation can be searched in the number of the attributes: the number is consistent, meaning that this algorithm can have suffered of the curse of dimensionality.
The values of the classification accuracy and the F1 score are very similar, indicating that the dataset unbalancing is compensated correctly with the oversampling method on the training set.
That affirmation is confirmed also by the ROC curve, where it's possible to see the highest values in the best algorithms for classification accuracy and F1 score.
In Figure 7 it's possible to observe the confusion matrix for each classifier, from which the ROC curves are made.

Table 18: Testing evaluation measures for each algorithm

Algorithm	AUC	CA	F1
Logistic regression	0,976	0,933	0,932
kNN	0,957	0,905	0,902
Decision tree	0,929	0,914	0,912
Random forest	0,979	0,943	0,942
Linear SVM	0,858	0,886	0,883
Polynomial SVM	0,915	0,914	0,911
RBF SVM	0,976	0,933	0,932

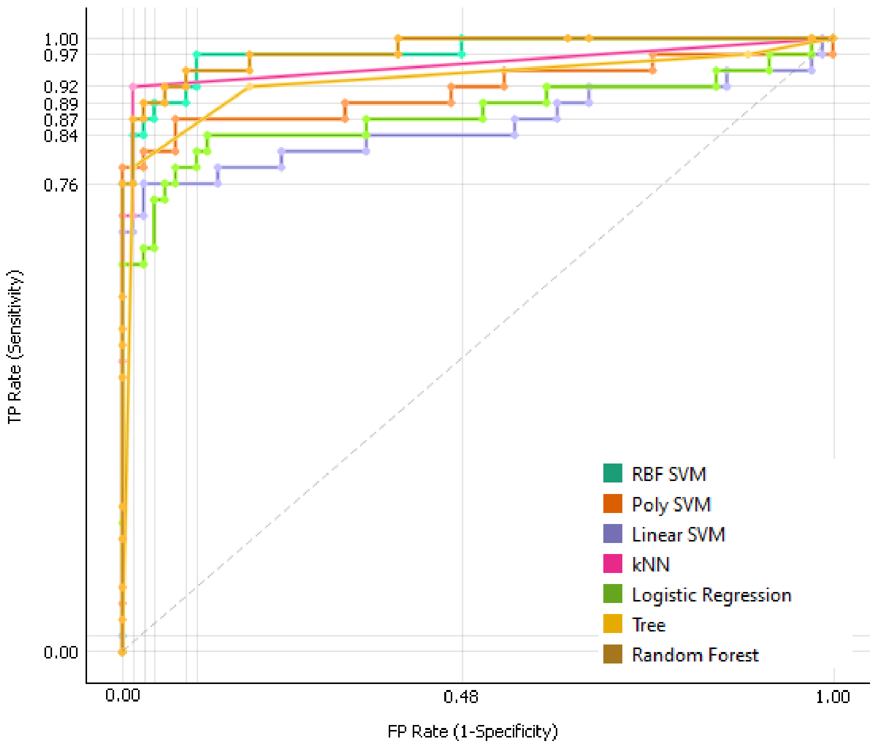


Figure 5: Classifier ROCs for bad label

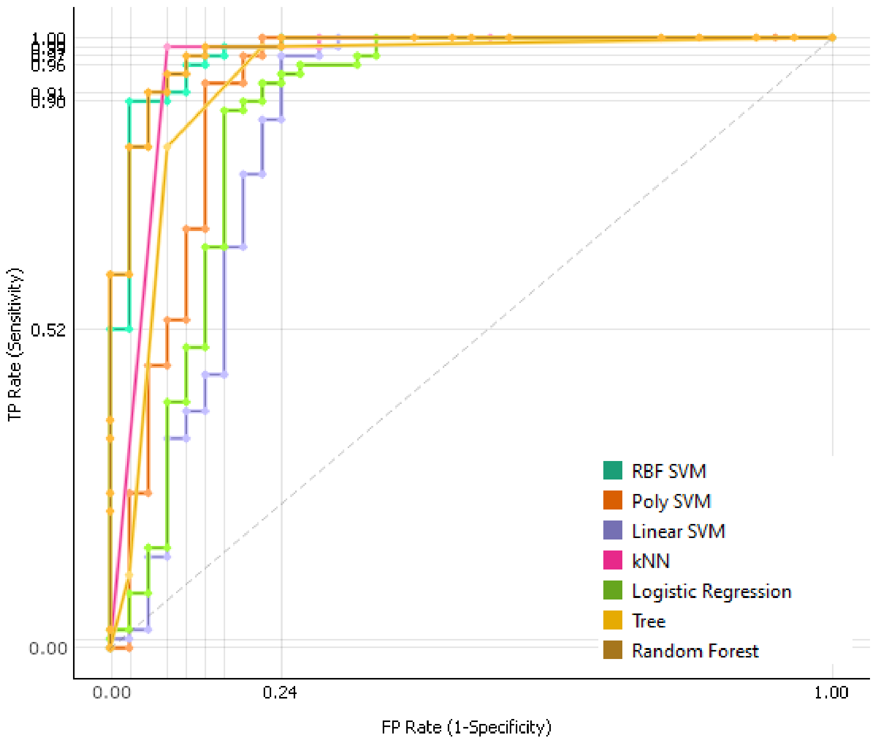
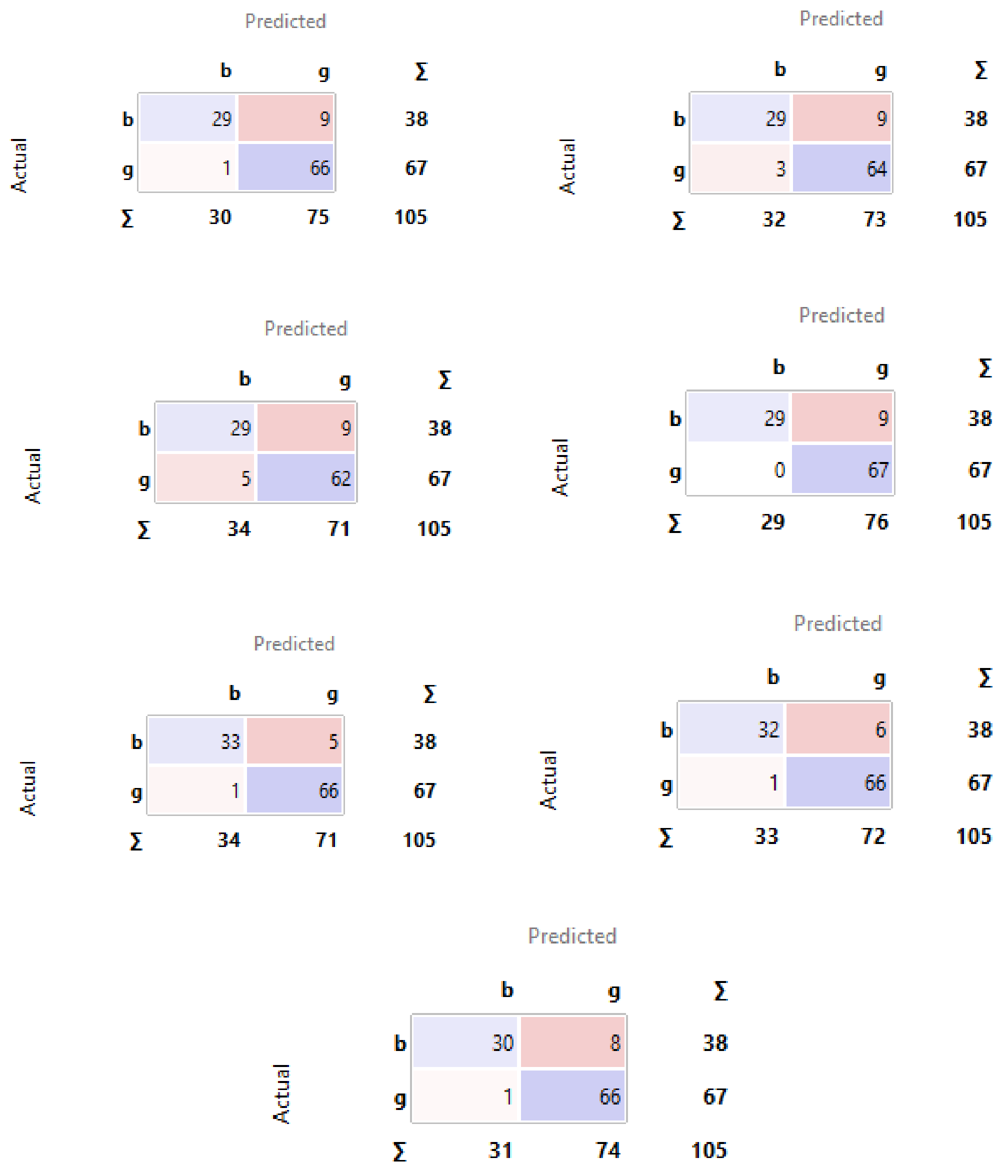


Figure 6: Classifier ROCs for good label

Figure 7: Confusion matrix for each classifier.
In order kNN, Linear SVM, Logistic regression, Polynomial SVM, Random forest, RBF SVM and Decision tree



		Predicted		
		b	g	Σ
Actual	b	29	9	38
	g	1	66	67
	Σ	30	75	105

		Predicted		
		b	g	Σ
Actual	b	29	9	38
	g	3	64	67
	Σ	32	73	105

		Predicted		
		b	g	Σ
Actual	b	29	9	38
	g	5	62	67
	Σ	34	71	105

		Predicted		
		b	g	Σ
Actual	b	29	9	38
	g	0	67	67
	Σ	29	76	105

		Predicted		
		b	g	Σ
Actual	b	33	5	38
	g	1	66	67
	Σ	34	71	105

		Predicted		
		b	g	Σ
Actual	b	32	6	38
	g	1	66	67
	Σ	33	72	105

		Predicted		
		b	g	Σ
Actual	b	30	8	38
	g	1	66	67
	Σ	31	74	105

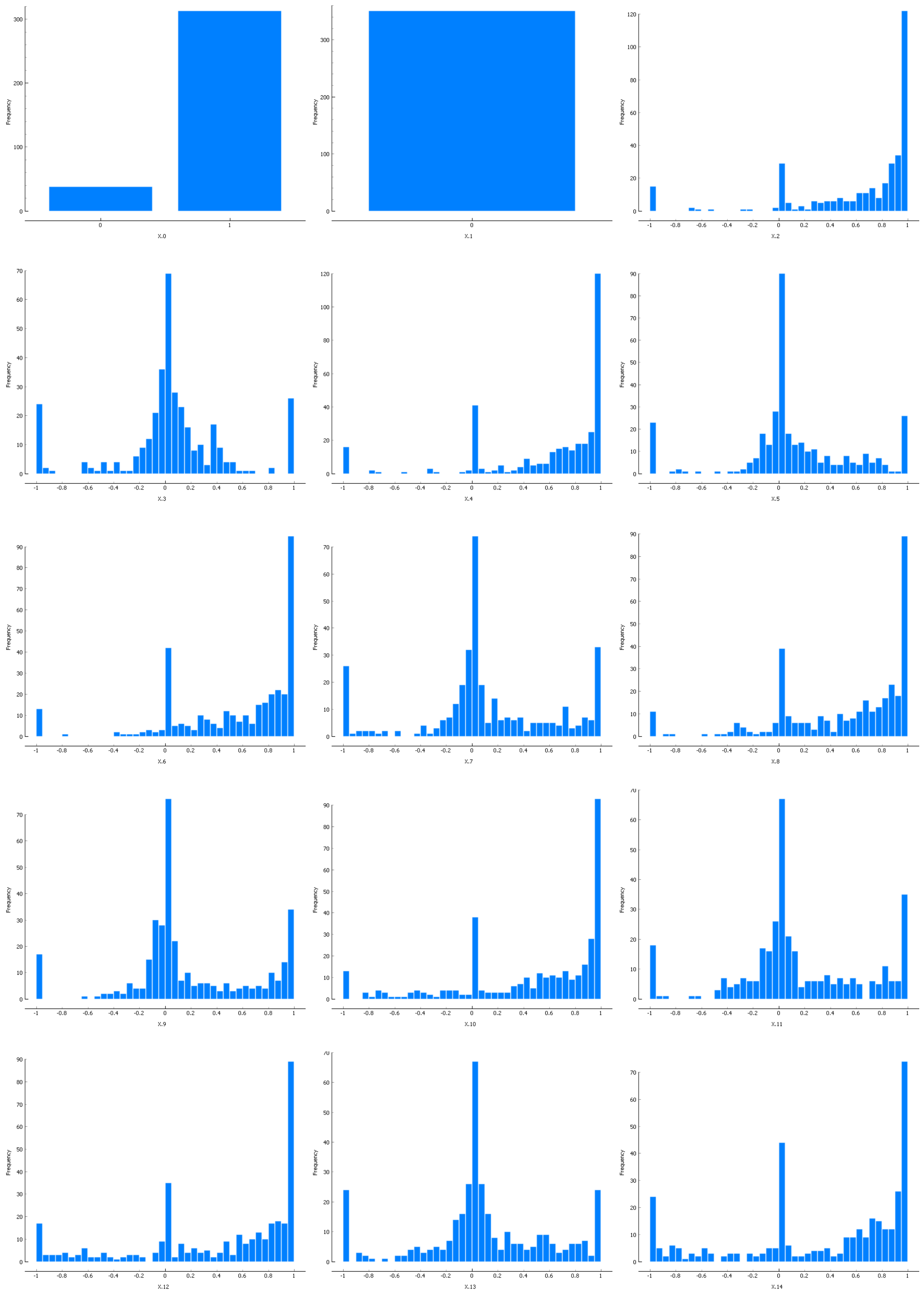
6. Additional tables and figures

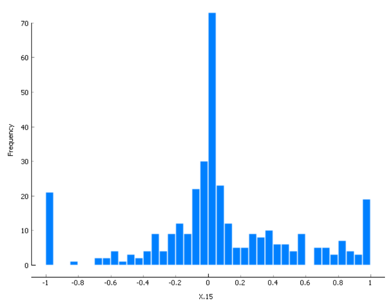
Table 1: Attribute statistics

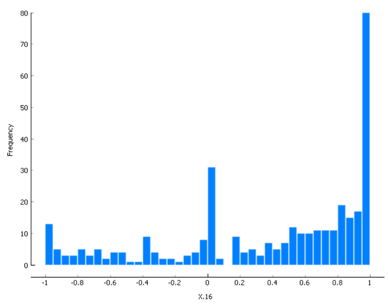
Name	Mean	Median	Dispersion
X.0	1	0,343	-
X.1	0	0	-
X.2	0,641	0,871	0,775
X.3	0,044	0,016	9,993
X.4	0,601	0,809	0,864
X.5	0,116	0,022	3,971
X.6	0,550	0,729	0,894
X.7	0,119	0,015	4,357
X.8	0,519	0,684	0,989
X.9	0,181	0,018	2,664
X.10	0,476	0,668	1,182
X.11	0,155	0,028	3,189
X.12	0,401	0,644	1,550
X.13	0,093	0,030	5,290
X.14	0,344	0,602	1,894
X.15	0,071	0,000	6,435
X.16	0,382	0,591	1,616
X.17	-0,004	0,000	-137,152

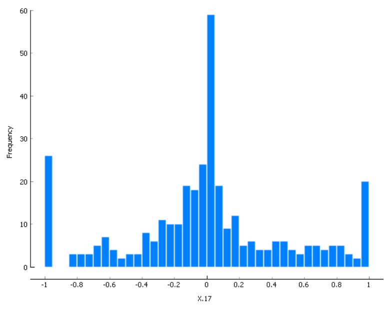
X.18	0,359	0,576	1,740
X.19	-0,024	0,000	-21,575
X.20	0,337	0,499	1,809
X.21	0,008	0,000	62,372
X.22	0,362	0,532	1,663
X.23	-0,057	0,000	-9,175
X.24	0,396	0,554	1,458
X.25	-0,071	-0,015	-7,133
X.26	0,542	0,708	0,952
X.27	-0,070	-0,018	-7,898
X.28	0,378	0,500	1,520
X.29	-0,028	0,000	-18,176
X.30	0,353	0,443	1,619
X.31	-0,004	0,000	-135,18
X.32	0,349	0,410	1,494
X.33	0,014	0,000	32,297
X.34 (target)		g	0,653

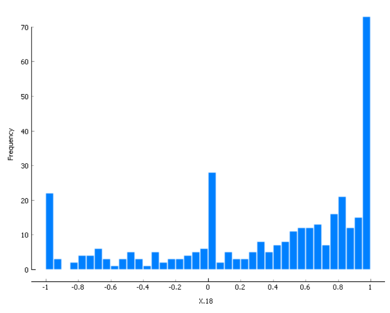
Figure 2: Attribute distribution graphs

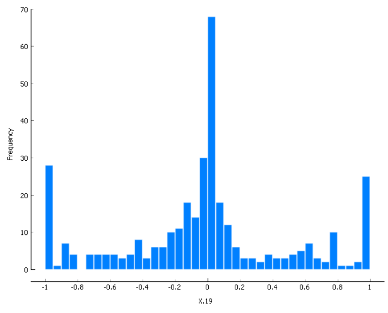


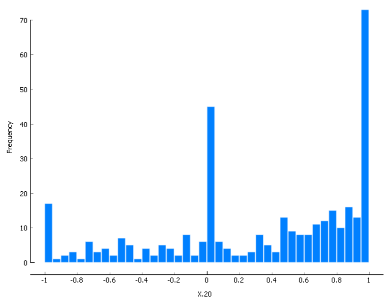

Frequency
X.15

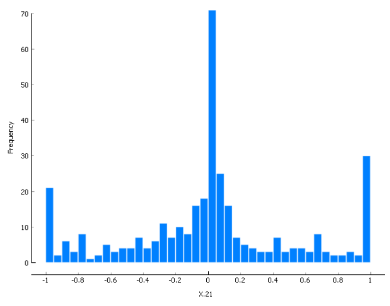

Frequency
X.16

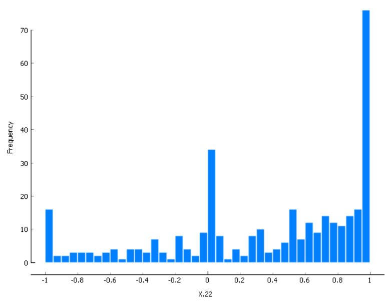

Frequency
X.17

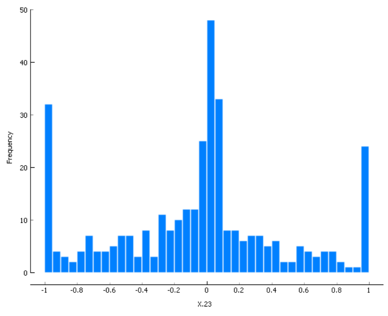

Frequency
X.18

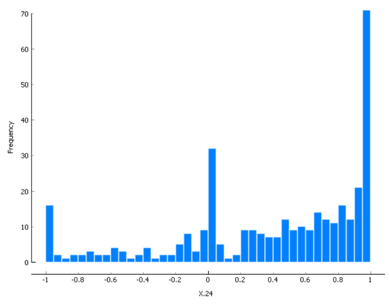

Frequency
X.19

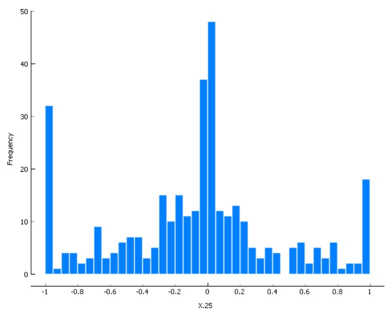

Frequency
X.20

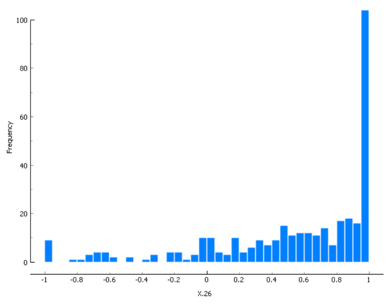

Frequency
X.21

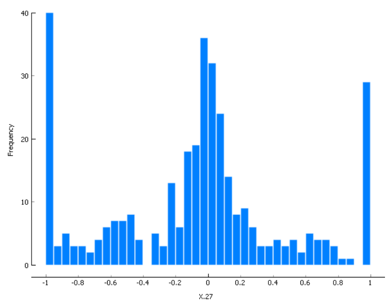

Frequency
X.22

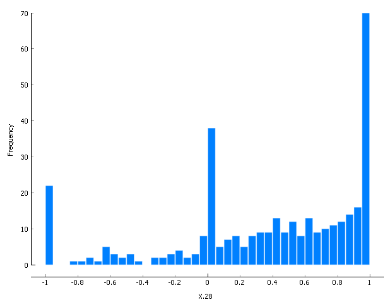

Frequency
X.23

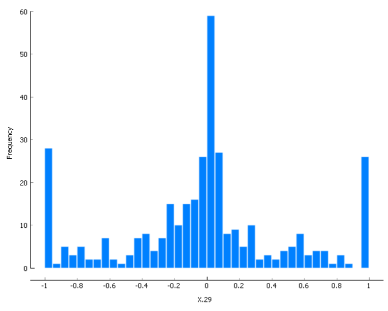

Frequency
X.24

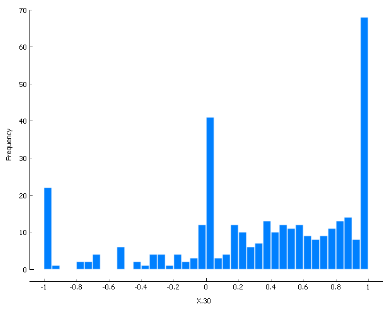

Frequency
X.25


Frequency
X.26

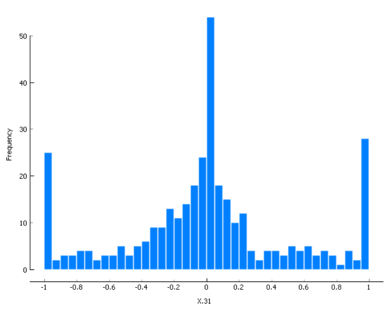

Frequency
X.27


Frequency
X.28

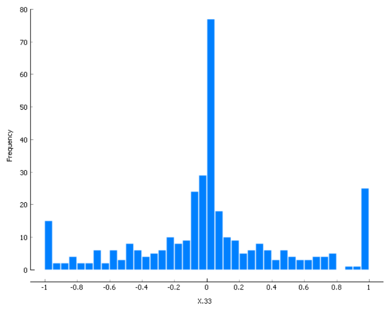

Frequency
X.29



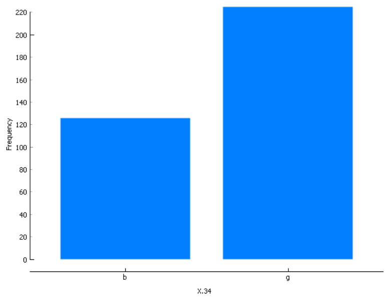
Frequency
X.30



Frequency
X.31



Frequency
X.33



Frequency
b
g
X.34

7. Bibliography

1. http://archive.ics.uci.edu/ml/datasets/Ionosphere
2. https://www.politocomunica.polito.it/corporate_image/marchio_e_identita_visiva
3. https://en.wikipedia.org/wiki/Support-vector_machine
4. https://en.wikipedia.org/wiki/Radial_basis_function_kernel
5. https://it.wikipedia.org/wiki/K-nearest_neighbors
6. https://www.datasciencesmachinelearning.com/2019/01/logistic-regression.html
7. https://privefl.github.io/blog/choosing-hyper-parameters-in-penalized-regression/
8. https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/index.html
9. https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8
10. https://en.wikipedia.org/wiki/Logistic_regression
11. https://www.researchgate.net/profile/Hongge-Chen/publication/320487716/figure/fig13/AS:582180318269440@1515814012533/6-The-Receiver-Operating-Characteristic-ROC-curves-The-ROC-curve-of-an-ideal.png