PRACTICAL APPLICATION 1

Rocks and Mines Classification

Author:
Serena Alderisi
s.alderisi@alumnos.upm.es

Course:
Machine Learning
Master Degree in Data Science
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

June, 2020

Contents

Introduction

The aim of this work is focused on the application of Machine Learning procedures to assess a binary classification problem: determine whether an object is a mine or a rock given the strength of sonar returns at different angles.

The problem begins when an unknown object is encoured and the classification has to be performed. Then, another question emerges: which method gives the best results when dealing with low number of observations and relatively big amount of variables (in the case of this work - different angles of measurement)?

This paper will explore and compare different classification methods with the aim to show the one that gives the most accurate and reliable results in this particular application.

To accomplish this, unsupervised and supervised approaches are used, including meta classifiers.

Problem description

The dataset used for this work is “Sonar: Mines vs. Rocks” used by Gorman and Sejnowski in their study of the classification of sonar signals.

The aim is discriminating between rocks and metal structures such as sea mines on the seafloor by using sonar signals.

This data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for mines and 180 degrees for rocks. The inputs are not raw sonar data or spectograms but spectral envelopes. Unfortunately, there are no further information on the frequencies (It is a paper from Cold War times).

The data frame is hosted on the UCI machine learning repository consists of 208 observations on 61 variables, all numerical and one (the Class) nominal:

- Each row represent a pattern: a set of 60 numbers in the range 0.0 to 1.0. The numbers in the attributes are in increasing order of aspect angle, but they do not encode the angle directly.
- The used file contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions and 97 patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency.
- Each number represents the energy within a particular frequency band, integrated over a certain period of time.
- The label associated with e ach record contains the letter ”R” if the object is a rock and ”M” if it is a mine (metal cylinder).

Methodology

This work has been carried out using RStudio.
Before starting to apply machine learning techiniques, an exploratory analysis is performed. The data don't present missing values and the distribution of the classes is balanced : 111 observations are mines (53%) and 97 are rocks (47%). The dataset is normalized already and all the values are in the same range 0.0 to 1.0. Hence, the only preprocessing to do is encoding the classes of type string to integers and split the data into training and testing data. Instead of having M and R values in the Label Class, the values are transformated into 0 and 1 respectively. The used portion of data are 80% for training (166 observations) and 20% for final evaluation (42 observations). Also the training set is well balanced: 92 mines and 74 rocks, almost the same proportions of the whole dataset: 53% mines and 47% rocks. Anyway, since random division may results in subsets that do not represent the given domain properly, repeated random runs are necessary. That's why, to obtain more reliable results, for each algorithms is applied the resampling methods of k-fold cross validation, with k= 5.

Furthermore, a visualization analysis is performed. For each attribute have been generated the histograms and the density plots, Figure 3.1. Even if the visual output show that many of the attributes have asymetrical distributions, has been prefered to proceed without performing further changes. This choice came from the fact that the dataset consists of already normalized data. It would have been possible to normalize even more the data, standardizing each attribute to have mean equal to zero and standard deviation equal to 1, but has been prefered not to transform too much the dataset.

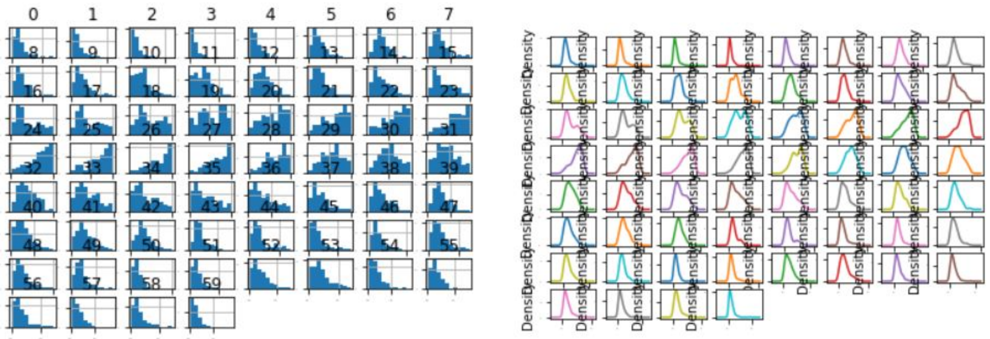


Figure 3.1: DataVisualization - Histograms and Density Plots

Then, different machine learning algorithms have been applied to the sonar dataset to determine whether an object on the seafloor is a mine or a rock. To do that 3 techiniques have been used: Supervised Classification, Classification with Metaclassifier and Unsupervised Classification.

In particular, Supervised Classification tecniques have been tested following three types of analysis:

1. with all the features available in the data set,

2. with a filter feature subset selection,

3. with a wrapper feature subset selection.

Lastly, all the performances obtained are discussed and comparated. All the details and the algorithms applied have been deepened in the following chapter.

Results and Discussion

The results are reported for three different sections: supervised classification, metaclassifiers and unsupervised classification.

Since the Class variable is a balanced class, the performance measure that is taken into account for all the algorithms is the Accuracy.
Moreover, every section contains a final benchmark to compare the performance obtained with the different algorithms.

4.1 Supervised Classification

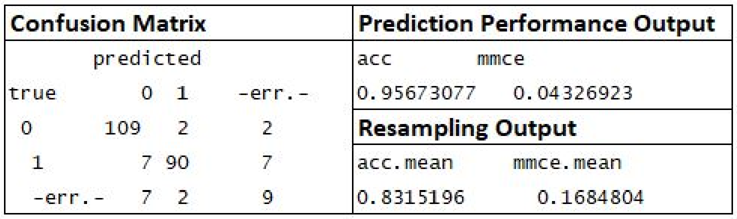
This section is dedicated to the outputs obtained by the application of both no-probabilistic and probabilistic supervised algorithms in the three analysis explained in the Methodology section 3. The algoritms used are both no-probabilistic and probabilistic and are the same for all the three analysis: K-Nearest Neighbors, Rule Induction, Classification Tree, Support Vector Machine, Neural Network, Logistic Regression, Naive Bayes and Discriminant Analysis.

4.1.1 Analysis with all the original variables

In this first analysis the features taken into account are all the 61 original variables of the sonar dataset.

K-Nearest Neighbors

This algorithm classifies a new observation taking into account its features and the features of the observations that looks like it: each new instance class will be predicted based on the similarity of its properties with the ones of other instances. In this analysis, the algorithm identified k=7 objects that have similar attributes to the instance that we are trying to define. To determine if it's a mine or a rock, this algorithms uses the most common class of those 7 objects. The result of the KNN algorithm can be seen in the Figure 4.1.



Confusion Matrix

```
        predicted
true         0  1    -err.-
 0       109   2      2
  1        7 90       7
  -err.-   7  2       9
```

Prediction Performance Output

```
acc         mmce
0.95673077   0.04326923
```

Resampling Output

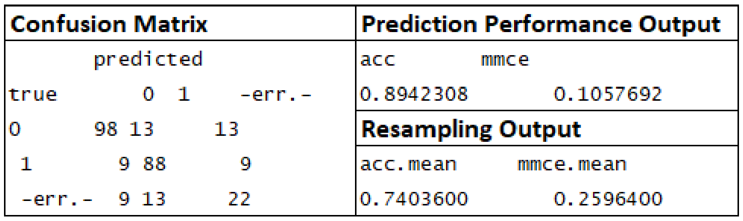
```
acc.mean      mmce.mean
0.8315196      0.1684804
```

Figure 4.1: K-Nearest Neighbors Algorithm Output

The performance of the k- nearest neighbor algorithm shows some signs of alarm, the accuracy of the forecast is extremely high: 0.96. Since this measure tracks the number of correct predictions of the model against the true class of the instances, the model made an accurate prediction in 96% of the observations. Hence, it's plausible to assume that the model is overfitting the training set. To avoid this problematic a resample techinique is performed: K-fold Cross Validation, with k=5. Indeed, the evidence of this issue can be seen in the results of the resampling aggregate, where the accuracy drops to 0.83, meaning that the built model classifies correctly the 83% of the instances. In resume, the model estimates almost perfectly in the training set but not that well after the resampling. Despite all these considerations The KNN algorithm generated a model with a good accuracy, making an accurate prediction in 83% of the observations.

Rule Induction

Rule induction: is a technique that creates "if–else–then"- statements from a set of input variables and an output variable to predict the class. In "mlr" package, the JRip algorithm implements a proportional rule learner algorithm, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which was proposed by William W. JRip uses sequential covering algorithms for creating ordered rule lists". This model doesn't exposed a good accuracy into predicting if an object is a rock or a mine, as the one achieved by the Knn algorithm. Contrarily it shows more consistency between the results obtaned during the training and the ones obtain in the generalization process after the resampling. As shown from the Figure 4.2 both accuracy values are very close.



Confusion Matrix

```
        predicted
true          0  1     -err.-
0       98 13      13
 1        9 88       9
 -err.-   9 13      22
```

Prediction Performance Output

```
acc         mmce
0.8942308         0.1057692
```

Resampling Output

```
acc.mean      mmce.mean
0.7403600         0.2596400
```

Figure 4.2: Rule Induction Algorithm Output

Classification Tree

It follows a tree structure to predict a class, where each tree has its root node (a relevant feature to predict the class), leaf nodes (following features) and branches (output of the test). Works by partitioning the feature space into a number of smaller scenarios with similar response values using a set of splitting rules. The segmentation process is typically carried out using only one explanatory variable at a time.

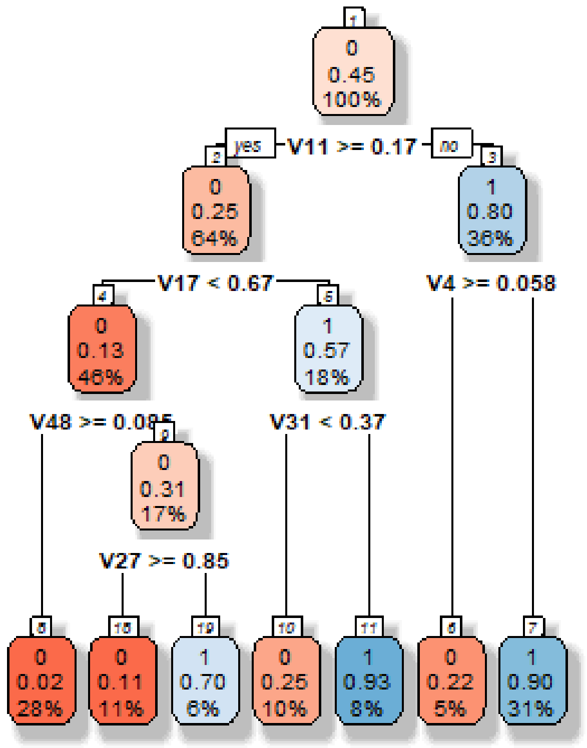
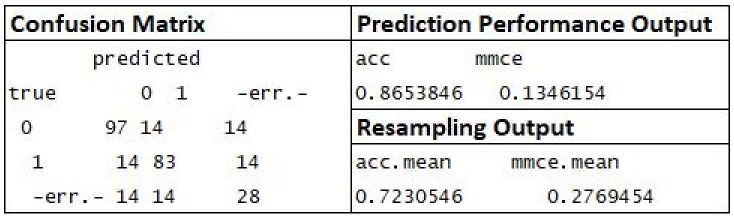


Figure 4.3: Classification Tree Graph

Likewise to the output of the KNN algorithm, the classification tree doesn't exposes a stable process of training and generalization. Also in this case, the performance measures for both summaries shown in Figure 4.4 are quite different: the training model predicted correctly 86% of the cases but this performance is lower after the application of the Resempling technique where the accuracy drops to 0.72. The resampling aggregates allow to assess the generalization of the model, in other words, how well that model it is going to predict the new data or unseen data.



Confusion Matrix

```
        predicted
true         0  1    -err.-
 0       97 14      14
  1        14 83      14
  -err.- 14 14      28
```

Prediction Performance Output

```
acc        mmce
0.8653846    0.1346154
```

Resampling Output

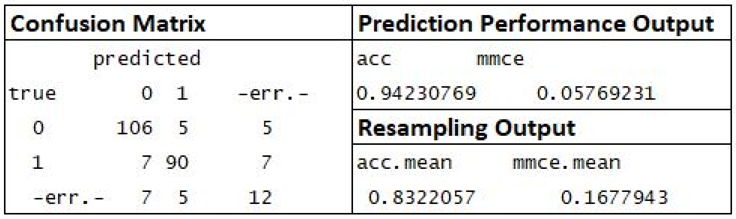
```
acc.mean      mmce.mean
0.7230546        0.2769454
```

Figure 4.4: Classification Tree Algorithm Output

The tree obtained in the first analysis is shown in the Figure 4.3. In this case, the algorithm applied for building the tree has used some of the available 60 features and the most relevant ones have been V11 and V17. Thus, according to this algorithm, the possibility to encounter a mine or a rock on the marine subsoil is associated with the angles of measurement described by the variables taken to built the tree: V11,V17,V4,V48,V31 and V27.

Support Vector Machine

The main purpose of this algorithm is to find a hyperplane in some feature space to build a classifier that allows us to separate the two classes or predict to which of the group belongs a new instance based on his attributes.



Confusion Matrix	Prediction Performance Output

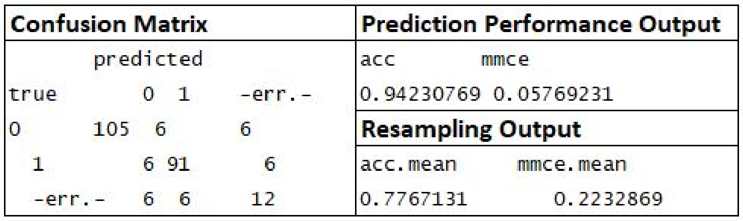
```predicted```
```true   0 1   -err.-```
``` 0   106 5   5```
``` 1   7 90   7```
``` -err.-   7 5   12```
	```acc mmce```
	```0.94230769 0.05769231```
	Resampling Output
	```acc.mean mmce.mean```
	``` 0.8322057 0.1677943```

Figure 4.5: Support Vector Machine Algorithm Output

So far, SVM algorithm, together with the KNN, are the ones with the highest accuracy values in the resampling aggregate: 0.83 (Figure 4.5). Thus, the deployment of this model will classify correctly a greater amount of new data into the detection of object under water surface, than the other algorithms analized so far.

Neural Network

This model is inspired by the way the human brain works, particularly, on the biological neuron structure. The approach holds all its power in a neuron, which is the central unit and the one in charge of the computations. It receives features as inputs and then operates a function of the weighted sum of these plus the bias to generate an output. Essentially, the prediction of the class it's done by a learning process that is based on the examples. 4.6



Confusion Matrix	Prediction Performance Output

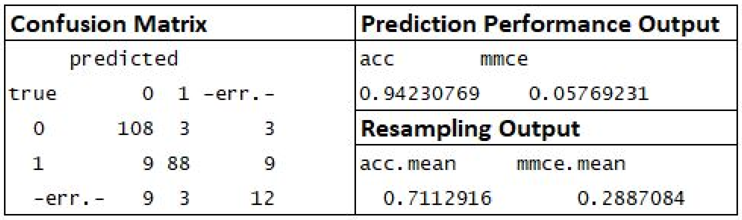
```predicted```
```true   0 1   -err.-```
```0   105 6   6```
``` 1   6 91   6```
``` -err.-   6 6   12```
	```acc mmce```
	```0.94230769 0.05769231```
	Resampling Output
	```acc.mean mmce.mean```
	```0.7767131 0.2232869```

Figure 4.6: Neural Network Algorithm Output

A single-hidden-layer neural network has been fitted, obtained after 100 iterations. With the Neural Network method, we were capable of predicting the object related to the sonar signals with an acceptable chance of misclassification. According to the performance measures, the model misclassificate an observation in 22% of the opportunities.

Logistic Regression

Here the focus is on the mean misclassification error: this performance measure evaluates the number of the wrong classifications from the total input of the training set (in this case). The objective in a machine learning project is to minimize this quantity. After the resampling this model produced a MMCE of 0.29, that means that this algorithm misclassified 29% of the instances according to the observed class. See Figure 4.7.



Confusion Matrix

```
     predicted
true       0  1 -err.-
  0      108  3      3
  1        9 88      9
  -err.-   9  3     12
```

Prediction Performance Output

```
acc        mmce
0.94230769    0.05769231
```

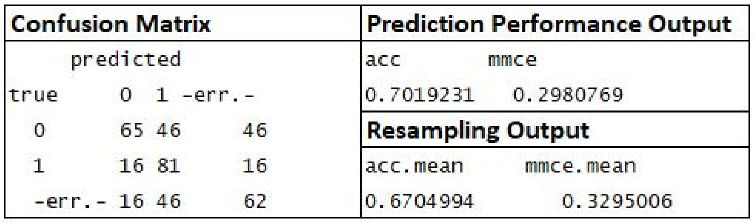
Resampling Output

```
acc.mean     mmce.mean
  0.7112916        0.2887084
```

Figure 4.7: Logistic Regression Algorithm Output

Naive Bayes

This approach is based on applying Bayes' theorem with strong (naïve) conditional independence assumptions between the features. It assumes that given the values of the other attributes, the effect of a particular feature on a class is independent from the first ones.



Confusion Matrix

```
     predicted
true      0  1 -err.-
  0      65 46     46
  1      16 81     16
  -err.- 16 46     62
```

Prediction Performance Output

```
acc        mmce
0.7019231   0.2980769
```

Resampling Output

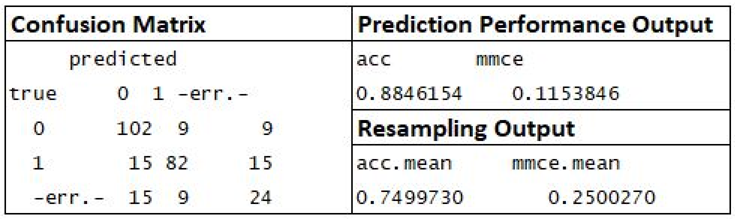
```
acc.mean     mmce.mean
0.6704994        0.3295006
```

Figure 4.8: Naive Bayes Algorithm Output

As it's possible to see in the Figure 4.8, the performance got with this model is the worst achieved so far. The accuracy values is the lowest compared to the ones obtained with the other algorithms: 0.67. This algorithm did an accurate prediction only in slidely more than the half of the times. Hence, with the Naive Bayes Classifier, it would be possible to predict the right object on the seafloor with the sonar signals, only the 67% of the time. This result in the accuracy of the prediction is explained because this model follows a parametric approach and it assumes normality and independence, two premises that are difficult to fulfill, especially the second one.

Discriminant Analysis

The aim of this model is to find linear combinations of the explanatory variables that characterizes or separates two or more classes of objects. This methos provides the best possible discrimination between groups, which in turn, are distributed as multivariate normals with a common variance–covariance matrix.



Confusion Matrix

```
     predicted
true      0  1 -err.-
  0     102  9      9
  1      15 82     15
  -err.-  15  9     24
```

Prediction Performance Output

```
acc        mmce
0.8846154    0.1153846
```

Resampling Output

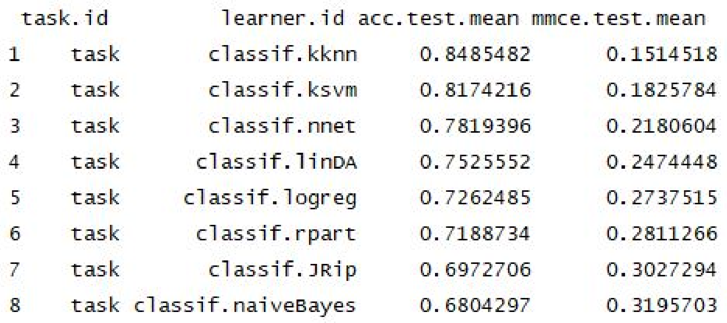
```
acc.mean     mmce.mean
0.7499730       0.2500270
```

Figure 4.9: Discriminant Analysis Algorithm Output

The linear and the quadratic approach have been examined, finding better results with the first. The model behaves more or less like the previous ones, the accuracy is above 70% (See Figure 4.9).

Benchmak Comparison

To judge, at the same time, the performances of all the algorithms used in this first analysis, a benchmark evaluation is performed (Figure 4.10). The results has been obtained with the benchmark function that by default, replicates the evaluations of the algorithms 100 times. The main idea is to compare and to rank the methods applied, according the highest accuracy values, to establish which one of them classified better in the training set and to determine which would generalize better in new data. It seems that all the algorithm have been affected to possible overfit problems. So, the evaluation of the best approach is based on the accuracy value obtained after the resampling techinique. The approaches that showed better performance to generalize on new data are: KNN and SVM.

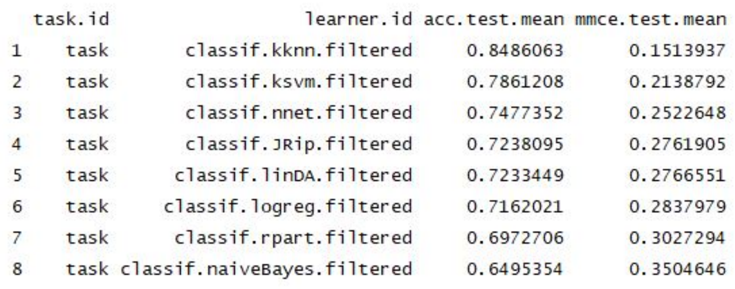


```
  task.id         learner.id acc.test.mean mmce.test.mean
1    task       classif.kknn     0.8485482      0.1514518
2    task       classif.ksvm     0.8174216      0.1825784
3    task       classif.nnet     0.7819396      0.2180604
4    task      classif.linDA     0.7525552      0.2474448
5    task     classif.logreg     0.7262485      0.2737515
6    task      classif.rpart     0.7188734      0.2811266
7    task       classif.JRip     0.6972706      0.3027294
8    task classif.naiveBayes     0.6804297      0.3195703
```

Figure 4.10: Benchmarch Comparison - Analysis with all the original variables

4.1.2 Analysis with a Univariate Filter Feature Subset Selection

The goal in this analysis is applying all the algorithms already seen in the first analysis (section 4.1.1) but only with the features that have more relevance. To do that a Filter Feature Subset Selection approach has been carried out, where the variables were selected according to the information gain criteria (in mlr package the information.gain filter corresponds to mutual information). The relevance of each variable is assessed individually. The threshold has been set manually to 60%: only 36 features have been selected between all the original ones.



```
  task.id                    learner.id acc.test.mean mmce.test.mean
1    task        classif.kknn.filtered     0.8486063      0.1513937
2    task        classif.ksvm.filtered     0.7861208      0.2138792
3    task        classif.nnet.filtered     0.7477352      0.2522648
4    task        classif.JRip.filtered     0.7238095      0.2761905
5    task       classif.linDA.filtered     0.7233449      0.2766551
6    task      classif.logreg.filtered     0.7162021      0.2837979
7    task       classif.rpart.filtered     0.6972706      0.3027294
8    task classif.naiveBayes.filtered     0.6495354      0.3504646
```

Figure 4.11: Benchmarch Comparison - Analysis with a univariate FSS

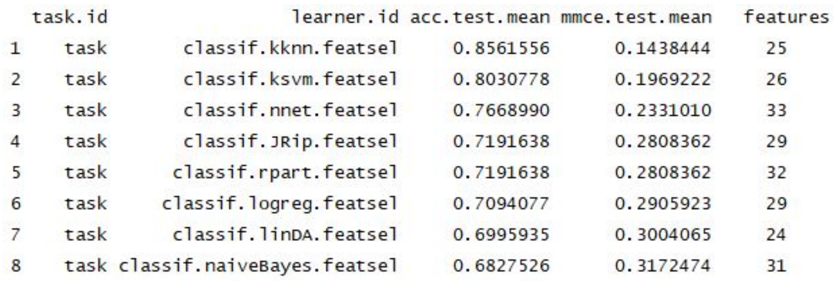
In the Figure 4.11 a Benchmark comparison is proposed: the algorithms have been sorted by the obtained accuracy values. Given the obtained results in the first analysis, also here these values refers to the algorithms applied after the resampling techinique.

Despite the feature subset selection the accuracy values obtained by all the models remains almost equal to the results obtained in the first analysis with all the original variables. In the KNN algorithm there is no change at all in the accuracy, all the others algorithms presented a slight reduction. Anyway the changes are relative only to the second digit after the decimal place. In particular, Naive Bayes classifier can be negatively affected by redundant variables and KNN is sensitive to irrilevant features, but since the results are basically the same for both algorithms, it means that each feature gives unique information to predict the Label class; there is no significant redundancy. Also with this approach the best result in terms of performance to generalize new data are KNN and SVM and ANN algorithms.

4.1.3 Analysis with a Multivariate Feature Subset Selection

The goal of the third analysis is focus always on the application of the all 8 supervised algorithms seen so far, but with a different type of selection between the variables: a Multivariate Feature Sub Selection in this case is performed. Univariate methods unavoidably discard features that, when taken in aggregate, would have provided useful information. Indeed, the wrapper feature selection approach detects the possible interactions between variables by subsetting the attributes.

Wrapper methods are based on greedy search algorithms as they evaluate all possible combinations of the features and select the combination that produces the best result for a specific machine learning algorithm. A downside to this approach is that testing all possible combinations of the features can be computationally very expensive, particularly if the feature set is very large.



```
  task.id                   learner.id acc.test.mean mmce.test.mean  features
1    task        classif.kknn.featsel     0.8561556      0.1438444       25
2    task        classif.ksvm.featsel     0.8030778      0.1969222       26
3    task        classif.nnet.featsel     0.7668990      0.2331010       33
4    task        classif.JRip.featsel     0.7191638      0.2808362       29
5    task       classif.rpart.featsel     0.7191638      0.2808362       32
6    task      classif.logreg.featsel     0.7094077      0.2905923       29
7    task       classif.linDA.featsel     0.6995935      0.3004065       24
8    task classif.naiveBayes.featsel     0.6827526      0.3172474       31
```

Figure 4.12: Benchmarch Comparison - Analysis with a multivariate FSS

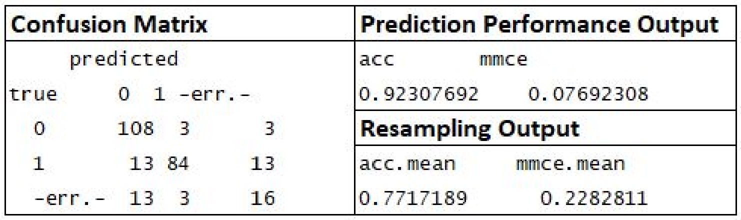
As said earlier, wrapper methods can find the best set of features for a specific algorithm. However, a downside is that these set of features may not be optimal for every other machine learning algorithm. That's why in Figure is possibile to see that each algorithm has its specific optimal set of features of different sizes obtained with a random search strategy. 5-fold cross-validated performances of the learners are replicated 100 times and summarized: The KNN algorithm has the highest performance in this analysis.

4.2 Metaclassifiers

This section is dedicated to the outputs obtained by the application of metaclassifier algorithms: Bagging, Ada Boosting and Random Forest.

Bagging

This term is derived on the method of bootstrap aggregation, which creates subsets of training observations by random selection with replacement. The subsets are slightly different training sets. Prediction works according the majority voting approach to create a discrete label, probabilities are predicted by considering the proportions of all predicted labels. Bagging is designed to improve the stability and accuracy of machine learning algorithms. It also reduces variance and helps to avoid overfitting. A generic bagging with the Artificial Neural Network algorithm is performed. NN is chosen because between the unstable classifiers seen so far is the one with the highest accuracy. The number of fitted models in bagging is set to 20, the results are displayed into Figure 4.13. Comparing this output with the ones obtained above, the performance measure did not improve.



Confusion Matrix

```
     predicted
true      0  1 -err.-
  0     108  3      3
  1      13 84     13
  -err.-  13  3     16
```

Prediction Performance Output

```
acc        mmce
0.92307692  0.07692308
```

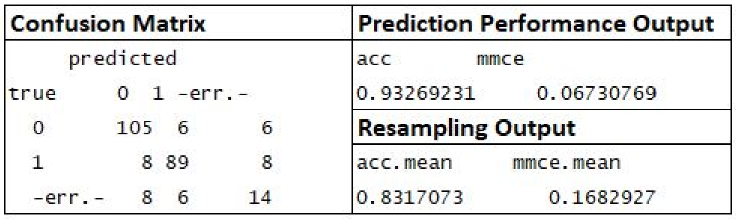
Resampling Output

```
acc.mean     mmce.mean
0.7717189     0.2282811
```

Figure 4.13: Generic Bagging Algorithm Output

Adaptive Boosting

The main weakness of Bagging is the randomness of its behavior. Adaboost instead, creates the classifiers one by one, each of them from a different training subset whose composition depends on the behavior of the previous classifiers. An other difference from the previous metaclassifier is that Adaboost chooses the examples probabilistically and the final decision is not achieved by the plain voting used in bagging. Instead, the so-called weighted majority voting scheme is used. It's based on the idea that each observation has a weight. The goal is minimizing the sum of the weights of the misclassified observations in each iteration. The errors of each iteration are used to update the weight of the observations in the training set, in such way the misclassified ones will have a heavier weight than the accurate classified ones. In this sense, the next model gives more relevance to the wrong predictions and it will try to minimize then. Looking at the Figure 4.14 is possible to see that the performance achieved with the AdaBoost algorithms are better than the one achieved with the Bagging algorithm.

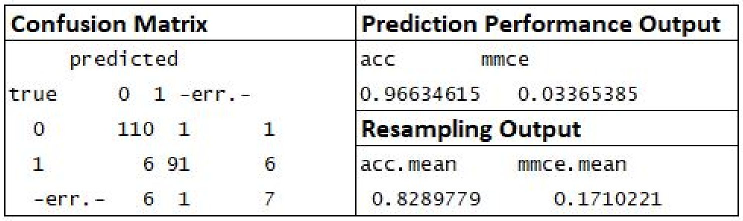


Confusion Matrix	Prediction Performance Output
predicted	acc mmce
true 0 1 -err.-	0.93269231 0.06730769
0 105 6 6	**Resampling Output**
1 8 89 8	acc.mean mmce.mean
-err.- 8 6 14	0.8317073 0.1682927

Figure 4.14: Adaptive Boosting Algorithm Output

Random Forest

Random forest is a modification of bagged decision trees that build a large collection of not correlated classification trees to improve predictive performance. According to Boehmke and Greenwell Bagging trees introduces a random component into the tree building process by building many trees on bootstrapped copies of the training data. Bagging then aggregates the predictions across all the trees; this aggregation reduces the variance of the overall procedure and results in improved predictive performance. Comparing the output obtained (Figure 4.15) with the results of the simple Classification tree, the performances are significantly improved. Better results are obtained even comparing them to the performance of the Bagging algorithm. Regarding the AdaBoost algorithm the accuracy values are almost the same.

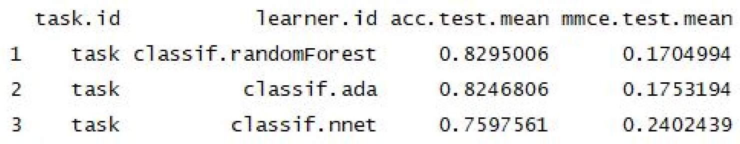


Confusion Matrix	Prediction Performance Output
predicted	acc mmce
true 0 1 -err.-	0.96634615 0.03365385
0 110 1 1	**Resampling Output**
1 6 91 6	acc.mean mmce.mean
-err.- 6 1 7	0.8289779 0.1710221

Figure 4.15: Random Forest Algorithm Output

Benchmak Comparison

After running 100 times the algorithms seen in this section, the results have been aggregated in the Figure (4.16). The output shows that in terms of performance the best metaclassifier is Random Forest.



```
  task.id          learner.id acc.test.mean mmce.test.mean
1    task classif.randomForest     0.8295006      0.1704994
2    task          classif.ada     0.8246806      0.1753194
3    task         classif.nnet     0.7597561      0.2402439
```

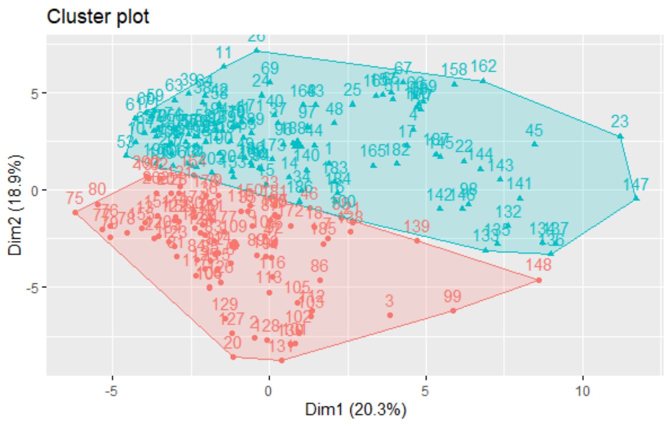
Figure 4.16: Metaclassifiers Benchmark Output

4.3 Unsupervised Classification

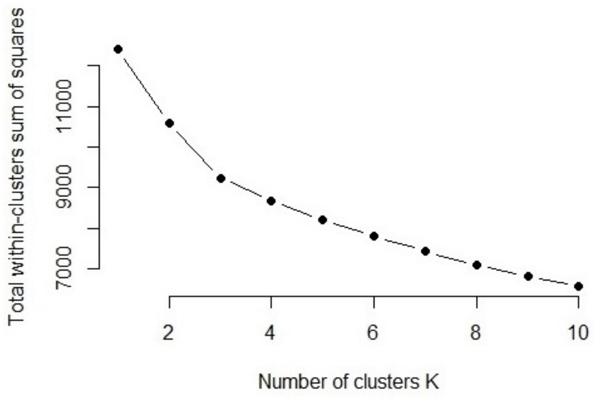
This section is dedicated to the outputs obtained by the application of the unsupervised algorithms: Kmeans, Hierarchical Clustering and Gaussian Mixture. The variable Label is removed.

K-Means

It's a partitional clustering method, based on the proximity or distance between observations. The idea is setting the k parameter and finding centroids in the cloud of data points and then assigned the observations to the nearest centroid to build the cluster. K in this case has been set equal to 2, since the Label variables has 2 classes. The output of this algorithm produced 2 groups of 93 and 115 observations (Figure 4.17(a)), that compared to the original data with labels, it identified the correct group only the 54% of the times. Usually the correct number of class labels is unknown and is estimated using the elbow plot, Figure 4.17(b). The elbow plot suggests to set k=3, that is different from the reality of the dataset.



(a) Kmeans Output with k=2



(b) Elbow Plot

Figure 4.17: Kmeans output with k=2 and Elbow Plot

Hierarchical Clustering

It's a classification method based on a tree structure following the bottom-upapproach: each observation starts in its own cluster and at each step it gets aggregate them, determining which ones are the observations belonging to each group until the end that there is only one big cluster containing all the observations. In this context the function hclust has been used that follows the complete linkage clustering: it computes all pairwise dissimilarities between the elements in 2 clusters, and considers the largest value (i.e., maximum value) of these dissimilarities as the distance between the two clusters. It tends to produce more compact clusters. The generate dendrogram, Figure 4.18, has been cut into 2 groups. This cutting generated 2 groups that have been compared with the original class variable that in this it has been removed to study the unsupervised algorithms. The first group contains 148 observations of which 74 are mines and 69 are rocks; the second group contains 65 observations of which 37 are mines and 28 are rocks. Hence, with the hierarchical approach it has been possible identify correctly only the 49% of the time the proper group where an observation belongs.

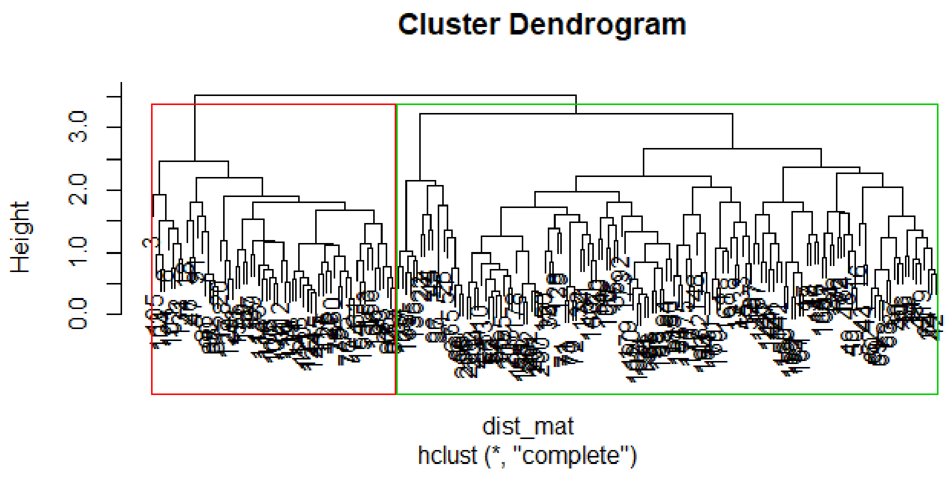
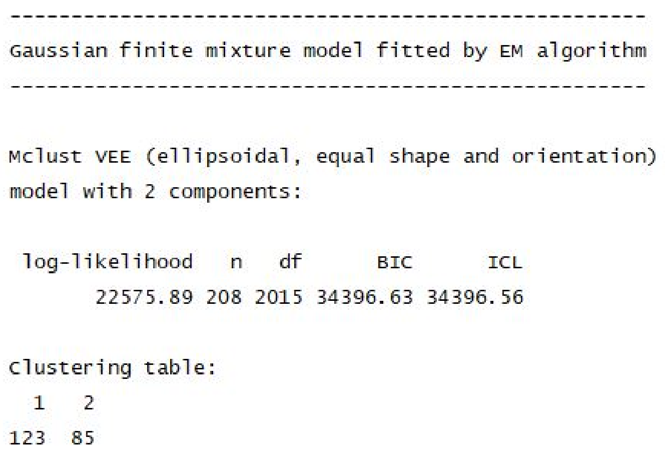


Figure 4.18: Hierarchical Clustering Algorithm Output

Gaussian Mixture

Expectation Massimization, EM, It's a probabilistic clustering algorithm and its goal is to maximize the overall probability or likelihood of the data, given the final clusters instead of maximizing the differences in means for continuous variables. As it's possible to see in Figure 4.19, the EM algorithm defined 2 clusters: the first of 123 observations and the second of 85. Comparing this results with the Label variable, it has been possible identify correctly only the 47% the right group to the observations belong: the lowest result achieved so far.

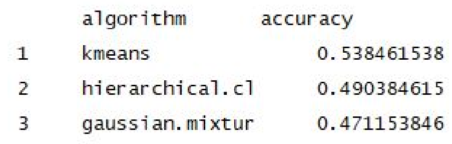


```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEE (ellipsoidal, equal shape and orientation)
model with 2 components:

 log-likelihood   n   df      BIC      ICL
       22575.89 208 2015 34396.63 34396.56

Clustering table:
  1   2
123  85
```

Figure 4.19: Gaussian Mixture Algorithm Output

Comparison

Accuracy values obtained in this section are assessed manually through comparing the resulting classifications with reference data. The Figure 4.20 shows a quick comparison bewteen the algorithms analized in this section: the best performance are obtained with Kmeans algortihm. Nevertheless, it can be considered satisfactory.



```
  algorithm         accuracy
1 kmeans            0.538461538
2 hierarchical.cl   0.490384615
3 gaussian.mixtur   0.471153846
```

Figure 4.20: Unsupervised Algorithms Comparison

Conclusion

Regardless the number of the variables used in the supervised classifications, the algorithms with the highest performance are the same in all the 3 analysis: K-Nearest Neighbor, Support Vector Machine and Artificial Neural Network. The performance measure of all the other algorithms applied to the sonar dataset in the Section 4.1 haven't been affected significatly after the variables reduction with filter and wrapper selection.

Note that in all the 8 supervised algorithms it was necassary resort to K-fold Cross Validation as resampling techinique to handle the overfitting.

Equally high performance, as the best ones obtained in the supervised classification, are observed also with the application of metaclassifiers algorithms. Compared to the general results obtained in this work all the 3 algorithms have produced some of the best results.

A different situation instead, has been observed regarding the unsupervised classification: all the 3 algorithms applied to the sonar dataset produced bad performance measures.

A reason why this kind of algorithms failed to divide in a satisfactory way the examples into groups, could be the composition of the dataset used in this work, where all the examples are numerical and whose values fall within a limited range. To have a global point of view, in the Figure 5.1 is proposed a bar chart with all the performance obtained in this work. To ensure an omogean comparison the measures relative to the supervised algorithms are the ones obtained in the first analysis with all the original variables.

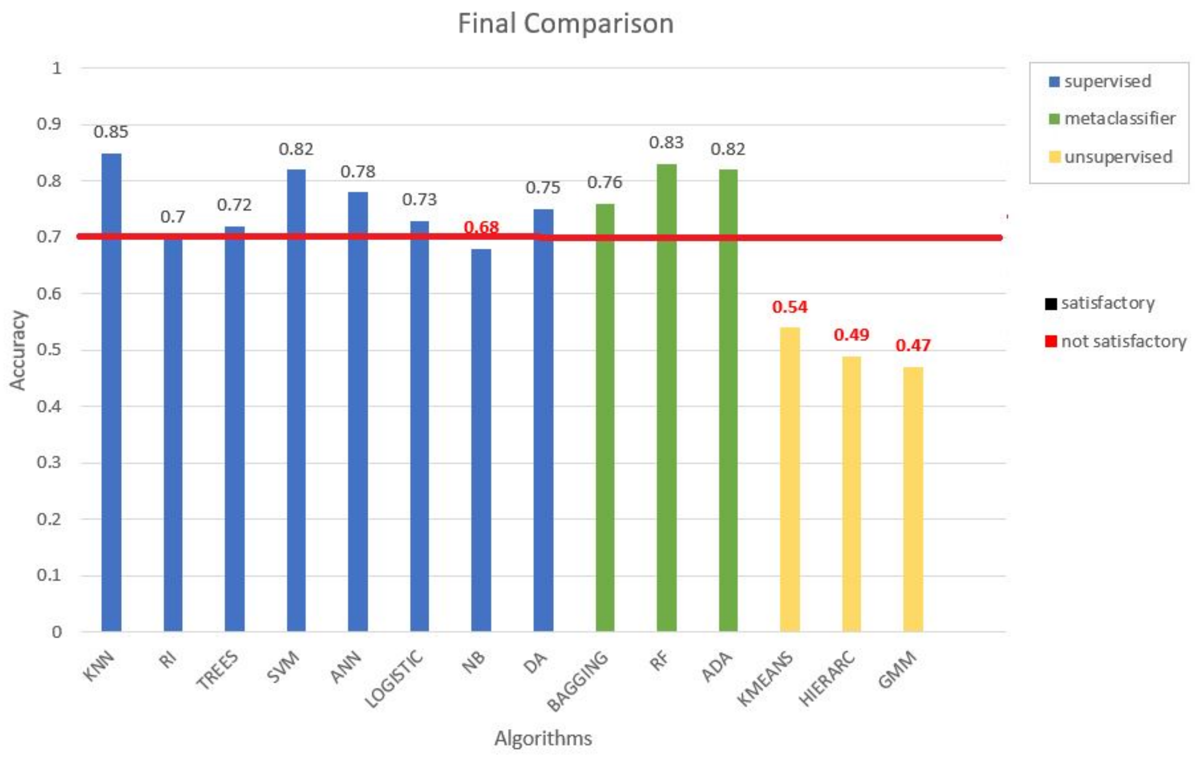


Figure 5.1: Final Comparison

Hence, the best algorithm in terms of performance is the KNN. Further investigation could be based on tuning the parameters of this algorithm to try to improve further the obtained results.

Since Random Forest, Support Vector Machine and Ada Bosting provided accuracy values very close to the one obtained with the K-Nearest Neighbor, and since all the algorithms used default tuning parameters, it would be intersting investigate how the results would change using different configurations of the parameters per each algorithm.

Finally, since the distributions of the attributes are skewed, it's plausible to suspect that the differing distributions of the raw data may be negatively impacting the skill of some of the algorithms. A further step could be to evaluate the same algorithms with a standardized copy of the dataset, transforming the data such that each attribute has a mean value of zero and a standard deviation of one.

References

1. Gorman, R. P., and Sejnowski, T. J. (1988). Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75-89

2. Miroslav Kubat. An Introduction to Machine Learning.

3. Max Kuhn, Kjell Johnson (2013). Applied Predictive Modeling.

4. V. López, A. Fernandez, S. Garcia, V. Palade and F. Herrera. An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics.

5. Sharma A., Dey S. A comparative study of feature selection and machine learning techniques for sentiment analysis. Proceedings of the 2012 ACM Research in Applied Computation Symposium, pp. 1–7. ACM (2012)

6. C. Bielza, P. Larrañaga (2014a). Discrete Bayesian network classifiers: A survey. ACM Computing Surveys, 47(1), Article 5

7. C. Bielza, P. Larrañaga (2014b). Bayesian networks in neuroscience: A survey. Frontiers in Computational Neuroscience, 8, Article 131

8. L. Breiman (2001). Statistical modeling: The two cultures. Statistical Science, 16(3), 199-231