Practical Application 1

Rafael Sojo García

rafael.sojo@alumnos.upm.es

October, 2022

1. Introduction

For this work a Dataset of Rain in Australia will be used to make predictions about whether or not it rains tomorrow. In order to do so, five non probabilistic algorithms have been used. These are:

- K-Nearest Neighbors
- Support Vector Machines
- Classification Trees
- Rule Induction
- Artificial Neural Networks

These methods have been implemented in conjunction with four different sets of features selected after the pre-processing steps. These are:

- All features
- Features selected with an univariate feature subset selection filter
- Features selected with a multivariate feature subset selection filter
- Features selected with a multivariate feature subset selection wrapper

2. Problem Description

This Dataset [1] contains data of Australian weather stations from the last 10 years, where there are stored more than 150.000 rows and a total of 23 columns including the target variable. Among others, there are dates, locations, and different weather-related measures. Here there is a brief explanation of them.

- *Date*: date where the observation was made.
- *Location*: location of the weather station.
- *MinTemp, MaxTemp*: minimum and maximum temperatures in degrees.
- *Rainfall*: amount of rainfall recorded in mm.
- *Evaporation*: Class A pan evaporation in mm.
- *Sunshine*: hours of bright sunshine in the day.
- *WinGustDir*: direction of the strongest wind in the 24 hours to midnight.
- *WindGustSpeed*: speed of the strongest wind in the 24 hours to midnight.
- *WindDir9pm, WinDir3pm*: direction of the wind at 9am and 3pm.
- *WindSpeed9am, WindSpeed3pm*: wind speed in km/h 10 min prior to 9am and 3pm.
- *Humidity9am, Humidity3pm*: humidity percent at 9am and 3pm.
- *Pressure9am, Pressure3pm*: atmospheric pressure at sea level at 9am and 3pm.
- *Cloud9am, Cloud3pm*: fraction of the sky obscured by cloud at 9am and 3pm.
- *Temp9am, Temp3pm*: temperature in degrees at 3pm.
- *RainToday*: yes if precipitation in the 24 hours to 9am exceeds 1mm, no otherwise.
- *RainTomorrow*: target variable.

3. Methodology

One of the biggest issues found for evaluating the performance of the different models and their selected features is the imbalance of the classes, since the raining days represents just the 15.5% of the whole Dataset after the pre-processing. In addition, the computational effort required in some cases was unfeasible or required too much time.

Therefore, all the models and subset of features have been tested and compared between each other for three different scenarios using a 10% of the Dataset (5.510 rows of the 55.104 remaining after the pre-processing), see **Figure 1**. These are the scenarios:

- Over Sampling of the minority class for the training set using *SMOTE* [2]algorithm.
- Under Sampling of the majority class for the training set using *Near-Miss* [2] algorithm.
- Default class weights.

On the other hand, the parameters of the models have been selected following a certain logic in conjunction with some trials, but without performing an exhaustive search. Thus, looking for the best results keeping as possible the *honesty*.

For the pre-processing steps, the missing values must be handled as well as the categorical variables and outliers.

There are missing values in almost every feature, therefore, part of them have been directly deleted, which is the case of *Shunshine, RainToday* and *RainTomorrow*, and the rest filled with the mean. Since *RainTomorrow,* is the target variable, it must not contain missing values, whereas the missing values of *RainToday* are almost the same. With respect to *Sunshine,* it is the feature with the biggest number of missing values, 69.835 out of 150.000, what makes unfeasible to use the mean to fill them.

The deletion of this rows solved most of the missing values for the remaining features. Nevertheless, before filling with the mean, the outliers were deleted, see **Figure 2**.

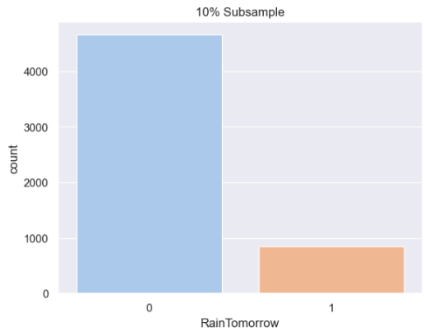


Figure 1. Count plot of the target variable.

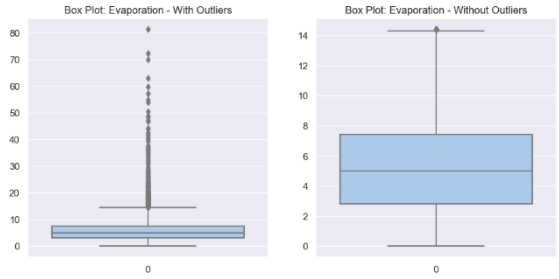


Figure 2. Example of features with and without outliers.

There are four categorical variables, three of them referring to the wind direction and one to locations. All of them have been encoded in relation to their frequency with *RainsTomorrow*, since there are certain locations and wind directions for which it tends to rain more frequently.

Regarding the *Date*, it gives us information about the frequency of rains with respect to the month, however, neither the years nor the days have any special relevance and can be removed.

On the other hand, all the input variables have been standardized or normalized for the non-rule based methods, i.e., **kNN**, **SVM** and **ANNs**, which are pretty sensitive to the domain of the features. However, in the case of **Classification Trees** or **Rule Induction** methods, it makes more sense to set the rules within the domain of each feature in order to preserve as much information as possible.

All the models have been implemented in Python with Scikit-Learn [3], TensorFlow [4], Pandas [5] and NumPy [6], while the results have been represented with Seaborn [7] and Matplotlib [8]. However, for the rule induction models and multivariate feature selection filters Wittgenstein [9] and ITMO FS [10] libraries have been use instead.

For the measurement of model performance, the AUC score has been used. However, the AUC score may perform relatively optimistic against unbalanced classes, therefore, the confusion matrix is calculated along with this metric in order to have a real knowledge of the quality of our models.

3.1. SMOTE vs Near-Miss

As it was explained previously, **SMOTE** and Near-Miss algorithms will be used along with the original weights of the classes to balance the training sets. Now a brief explanation of each is given.

SMOTE algorithm [11] is an over sampling method that creates artificial data for the minority class, based on the *k* nearest neighbors of a random selection of the minority class.

On the other hand, **Near-Miss** randomly eliminates samples from the majority class. "When two points belonging to different classes are very close to each other in the distribution, this algorithm eliminates the datapoint of the larger class" [12]

3.2. Models

For the **kNN** algorithm, a loop from 1 to 100 was performed to select the number of neighbors for which the AUC score was the highest. Moreover, the neighbors are weighted inverse to their distance. Once the number of neighbors is selected, the results are obtained after a cross validation process of 6 folds.

Regarding the configuration of the **SVM** algorithm, *class weights* are set *to* "balance", while *gamma* is set to "auto" instead of "scale", which is the default.

On the other hand, the data is not linearly separable, therefore an "rbf" kernel is chosen. Then, the results are obtained after a cross validation of 6 folds as in **kNN**.

In the configuration of the **Classification Trees** model, the classification criteria is set to entropy, while the best value of *alpha* is searched during a cross validation process of 5 folds. This parameter is related to the pruning of the tree and helps to obtain better results.

With respect to the **ANN**, it was created using Keras, from TensorFlow, due to the freedom that allows for its configuration. The model contains 6 hidden layers, for which every two are made of 64, 128 and 256 neurons respectively. A dropout layer of 0.5 is set between every two steps except for the final layer, which is reduced to 0.2.

The ReLu activation function is used for the hidden layers, whereas Sigmoid is set for the output and the **Adam** optimizer with a learning rate of 1e-5. Then, for the predictions everything above 0.5 will be consider as 1 and 0 otherwise.

Finally, for the Rule Induction model, RIPPER algorithm was selected. In its configuration, *k* represents the number of optimization iterations, which is set to 3 (2 by default) and the *prune size* is set to 1% of the Dataset, since after some tests it demonstrated to give better results. Then, as in **kNN** and **SVM**, the results are displayed after a cross validation process of 6 folds.

3.3. Feature Subset Selection

- For the univariate filter, **Mutual Information** has been used to calculate the importance of each feature. Then, a threshold of 0.01 was set for selecting the best feature subset.
- For the multivariate filter, the **CFS** algorithm was chosen. However, the multivariate filter from ITMO FS library requires to specify the number of features. Therefore, in order to be consistent with the rest of the feature selection methods, a similar number have been set. Normally, this number has tended to be around the half of features in every case, therefore, it was set to 10.
- For the wrapper approach, a **Sequential Forward Feature Selection** algorithm was used with the AUC score as metric.

4. Results

Now, begging with the results, the selected features for each method are displayed in **Table 1**, whereas the AUC scores of each model with the SMOTE, Near-Miss and Unbalanced training sets in **Tables 2, 3 and 4** respectively.

Univariate	Multivariate	Wrapper Knn	Wrapper SVM	Wrapper Classification Trees	Wrapper ANN	Wrapper RIPPER
Humidity3pm	Humidity3pm	Humidity3pm	Humidity3pm	Humidity3pm	Humidity3pm	Humidity3pm
Sunshine	Sunshine	Sunshine	Sunshine	Sunshine	Sunshine	Sunshine
Cloud3pm	Cloud3pm	Cloud3pm	Cloud3pm	Cloud3pm	Cloud3pm	Cloud3pm
Cloud9am	Cloud9am	Cloud9am	Cloud9am	Cloud9am	Cloud9am	Cloud9am
Pressure3pm	Pressure3pm	Pressure3pm	Pressure3pm	Pressure3pm	Pressure3pm	Pressure3pm
Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall
WindDir3pm	WindDir3pm	WindDir3pm	WindDir3pm	WindDir3pm	WindDir3pm	WindDir3pm
WindGustSpeed	WindGustSpeed	WindGustSpeed	WindGustSpeed	WindGustSpeed	WindGustSpeed	WindGustSpeed
Pressure9am	Pressure9am	Pressure9am	Pressure9am	Pressure9am	Pressure9am	Pressure9am
MaxTemp	MaxTemp	MaxTemp	MaxTemp	MaxTemp	MaxTemp	MaxTemp
Location	Location	Location	Location	Location	Location	Location
Humidity9am	Humidity9am	Humidity9am	Humidity9am	Humidity9am	Humidity9am	Humidity9am
WindGustDir	WindGustDir	WindGustDir	WindGustDir	WindGustDir	WindGustDir	WindGustDir
WindDir9am	WindDir9am	WindDir9am	WindDir9am	WindDir9am	WindDir9am	WindDir9am
Temp3pm	Temp3pm	Temp3pm	Temp3pm	Temp3pm	Temp3pm	Temp3pm
MinTemp	MinTemp	MinTemp	MinTemp	MinTemp	MinTemp	MinTemp
Month	Month	Month	Month	Month	Month	Month
Temp9am	Temp9am	Temp9am	Temp9am	Temp9am	Temp9am	Temp9am
WindSpeed9am	WindSpeed9am	WindSpeed9am	WindSpeed9am	WindSpeed9am	WindSpeed9am	WindSpeed9am
Evaporation	Evaporation	Evaporation	Evaporation	Evaporation	Evaporation	Evaporation
RainToday	RainToday	RainToday	RainToday	RainToday	RainToday	RainToday
WindSpeed3pm	WindSpeed3pm	WindSpeed3pm	WindSpeed3pm	WindSpeed3pm	WindSpeed3pm	WindSpeed3pm

Table 1. Subsets of features.

	kNN	SVM	Classification Trees	ANN	RIPPER
All Features	0.764	0.767	0.696	0.861	0.658
Univariate	0.756	0.774	0.668	0.85	0.687
Multivariate	0.767	0.777	0.752	0.867	0.726
Wrapper	0.709	0.767	0.678	0.856	0.708

Table 2. SMOTE balance results.

	kNN	SVM	Classification Trees	ANN	RIPPER
All Features	0.659	0.731	0.698	0.798	0.561
Univariate	0.682	0.734	0.612	0.772	0.622
Multivariate	0.720	0.747	0.694	0.726	0.568
Wrapper	0.678	0.737	0.594	0.821	0.626

Table 3. Near-Miss balance results.

	kNN	SVM	Classification Trees	ANN	RIPPER
All Features	0.643	0.777	0.717	0.858	0.7
Univariate	0.643	0.773	0.732	0.844	0.711
Multivariate	0.674	0.775	0.772	0.852	0.735
Wrapper	0.659	0.763	0.749	0.854	0.717

Table 4. Unbalanced results.

For a better understanding of the data and why some models perform better than others, we can draw a *pairplot* from seaborn to check the distribution of the points between all the variables along the axes. Even though it is not possible to draw a unique representative scatter plot when the dimensionality of our data is greater than 3, **Figure 3** shows that it is not a fully linearly separable problem. What explains the fact that **SVM** with the RBF kernel and the **ANN** models performed better than the others in most of the cases.

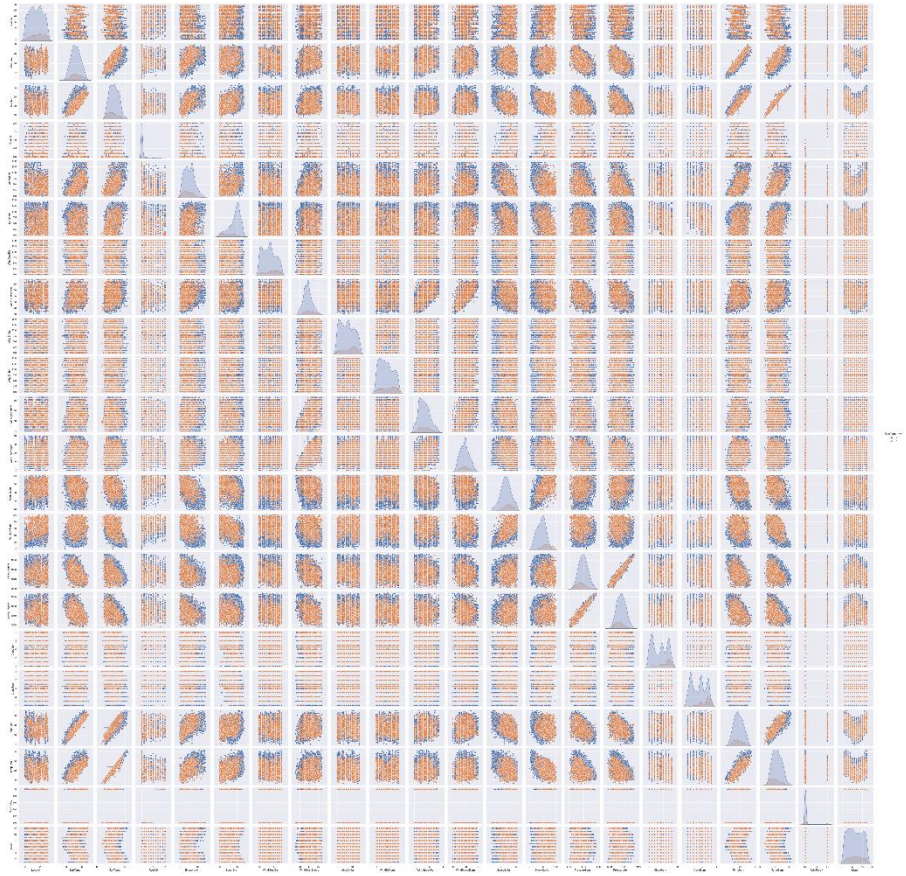


Figure 3. *Pairplot* of all the features.

Likewise, we can also draw a correlation matrix, **Figure 4,** in order to understand why some features where more frequently selected in all the subsets than others. For instance, the case of *Shunshine* and *Pressure3pm*, which appeared in all the subsets. They have a negative correlation with *RainTomorrow*, what means that there is less likely to rain when these values are higher.

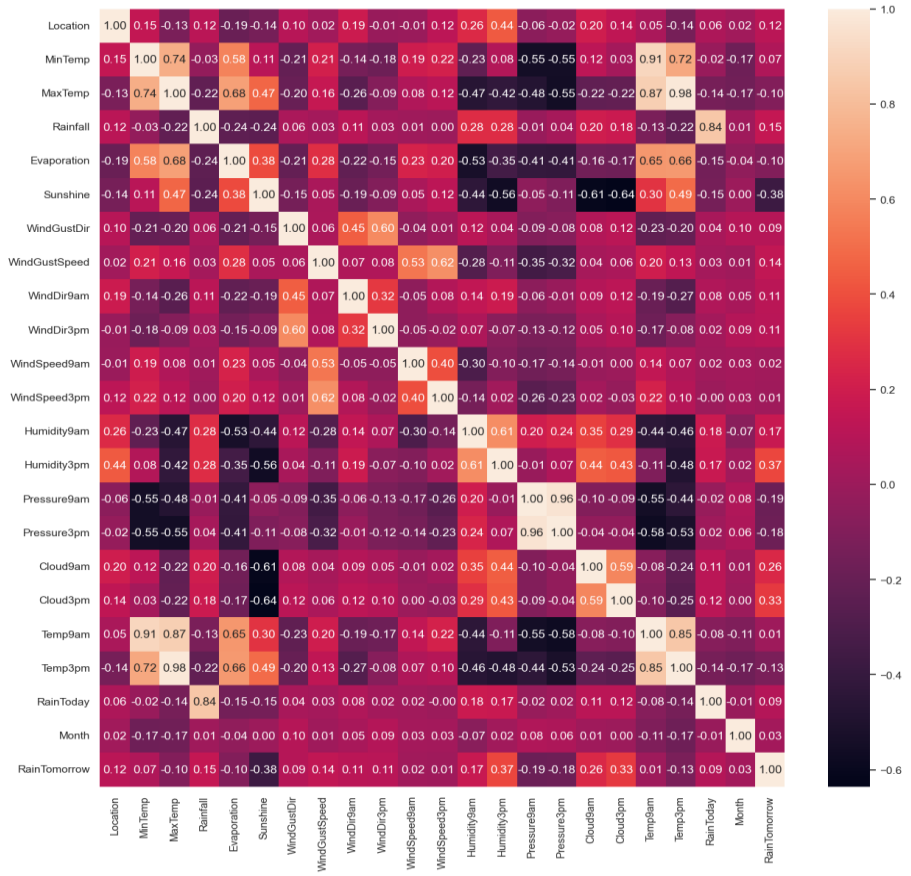


Figure 4. Correlation Matrix.

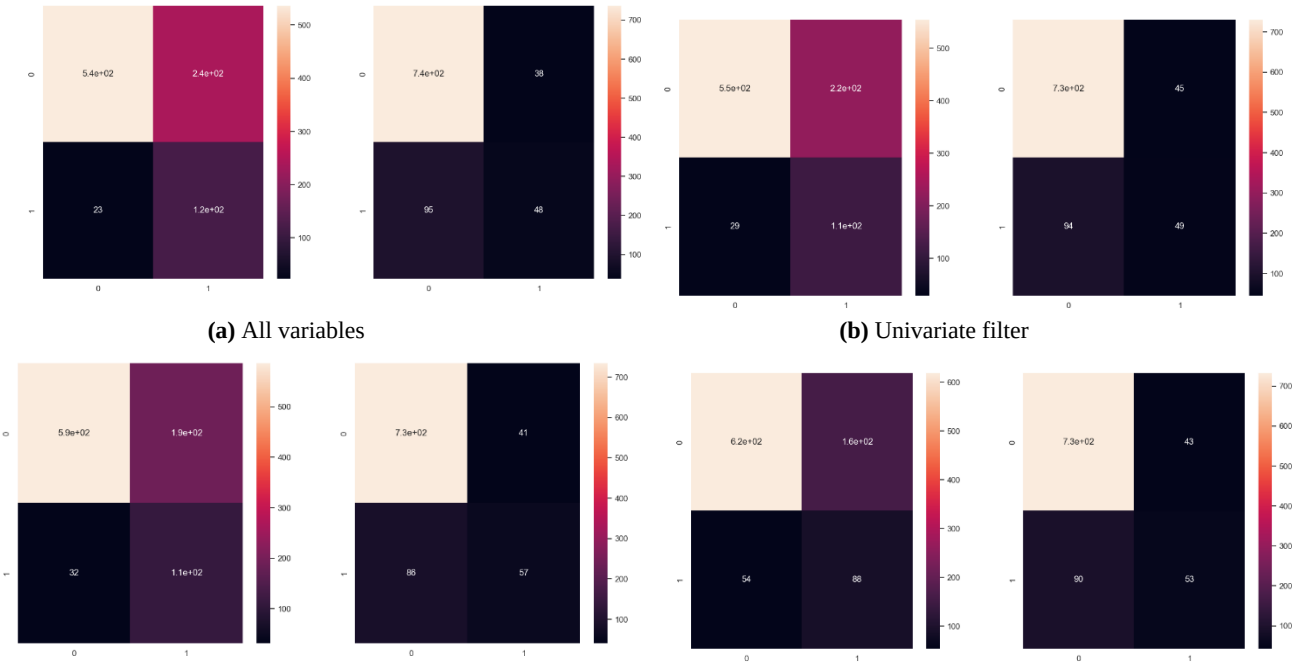
Taking this into account, we can now take a better look to our results and their costs matrixes.

5. Discussion

First of all, we can affirm that the **SMOTE** balance procedure achieved the significantly better results than **Near-Miss** in all the scenarios, therefore, it will not be included in the discussion of the performance of each model at each subset. However, the results are between 0.6 and 0.86, what means that the models will not be as good as expected.

Considering this fact, we can begin analyzing the results of the **kNN** model. Here, it is clear how poorly this model performs when we deal with unbalanced data, for which the balanced dataset increased the performance around a 10-15%, what could be interpreted as much higher if we compare their confusion matrixes, see **Figure 5**. With respect to the feature subsets, there is a slightly but clear improvement of the results when any of the filters is applied, where the **CFS** algorithm apparently selected the best features.

However, the Wrapper has even decreased the model performance, what could be caused by the number of neighbors used for this feature selection, since this parameter was left by default, i.e., 3 neighbors. This was done due to the computational effort required for looking a 100 times for the best number of neighbors with the best feature subset.

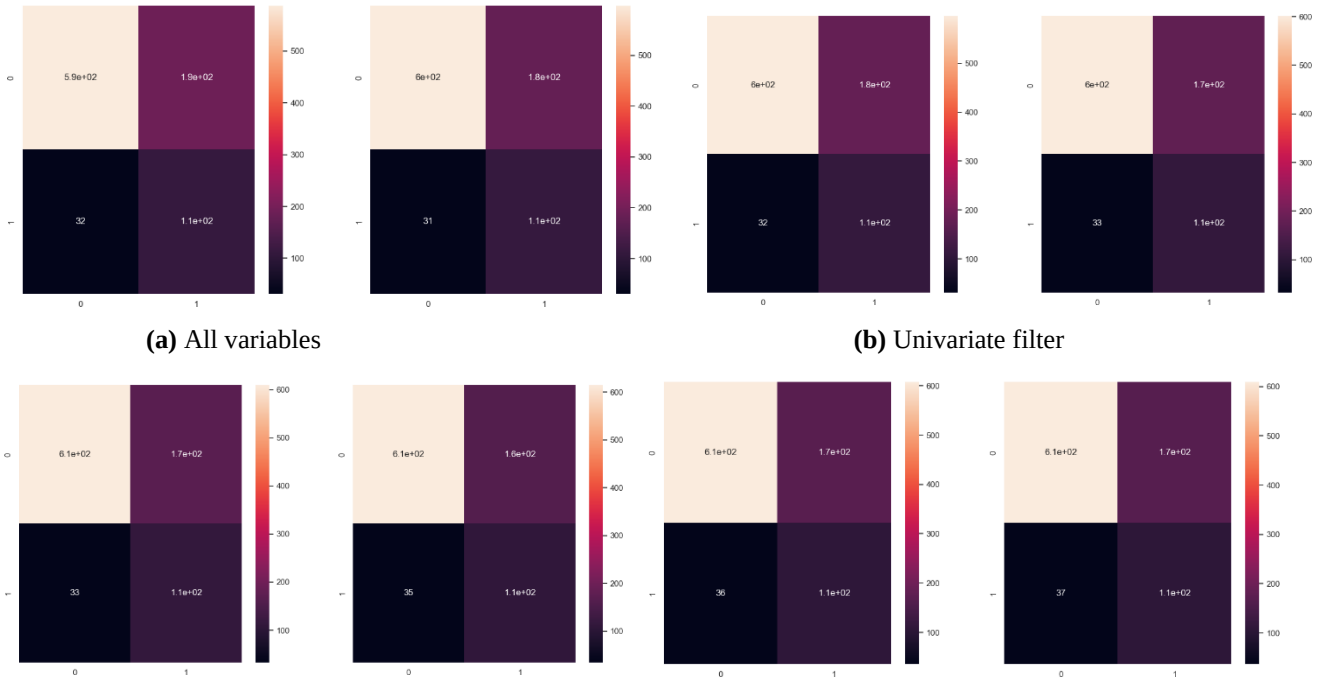


(a) All variables **(b)** Univariate filter

(c) Multivariate filter **(d).** Wrapper

Figure 5. Confusion matrixes for kNN algorithm.

On the other hand, for the **SVM** model we have practically the same results in the balanced training set as in the unbalanced. This is a clear effect of setting the *class weights* parameter to "balance". Regarding the dimensionality reduction, there is a little improvement in the performance when reducing the number of features, however, it is not as representative as it was in **kNN,** since it is so likely that the parameters selected in the configuration are having the greatest impact in the results, see the confusion matrixes at **Figure 6.**

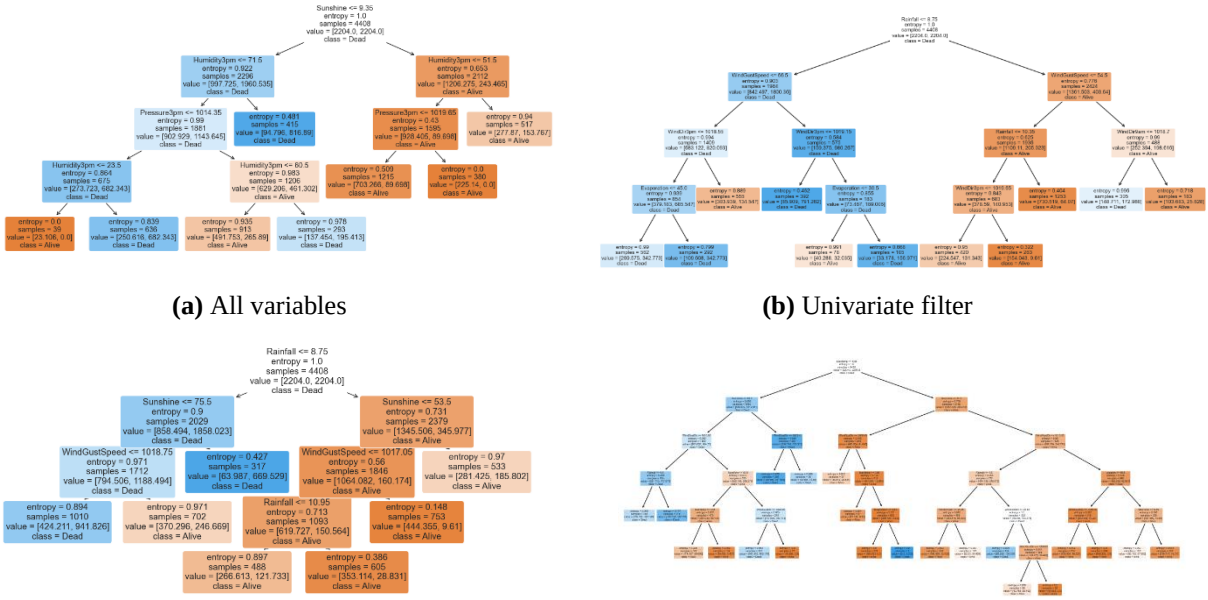


(a) All variables **(b)** Univariate filter

(c) Multivariate filter **(d).** Wrapper

Figure 6. Confusion matrixes for SVM algorithm.

In the case of **Classification Trees**, the unbalance training set seems to perform surprisingly better than the balance for all the feature subsets. What could be caused by the fact that when using SMOTE, we are generating synthetical samples with the same expected value but less variance, which "has little impact on the classifiers that base their classification rules on class-specific mean values and overall variances, while it has some (harmful) impact on the classifiers that use class-specific variances" [13]. Despite of this, the dimensionality reduction had a positive effect in the model performance, especially the case of the multivariate subset. Bellow, it can be seen the set of rules of each feature subset after the pruning of the tree.



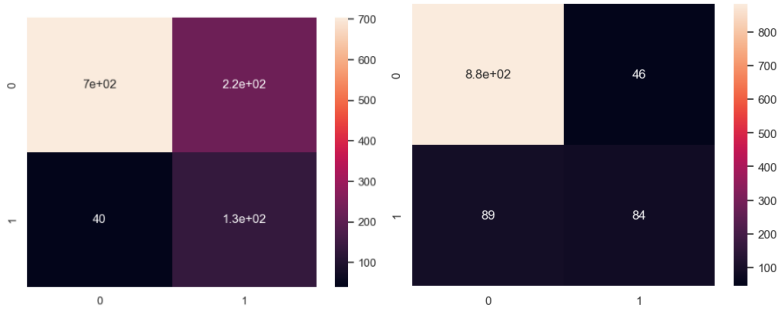
(a) All variables **(b)** Univariate filter

(c) Multivariate filter **(d).** Wrapper

Figure 7. Unbalance Classification Trees for each feature subset.

Continuing with the **ANNs**, it is clearly the model with the best performance ratio in both cases apparently. However, if we look to the confusion matrixes of both the balanced and unbalanced training sets, this last is clearly bias to the No rain tomorrow category, see **Figure 8**. These models are generally quite susceptible to class weights.

Regarding the feature selection, there is no apparent improvement on reducing the dimensionality and, since an ANN is by default a black box, it is quite difficult to guess the path within the net that has led to such results on each case. However, it could be case that the model itself is the optimal for this particular Dataset



(a) . Balanced **(b).** Unbalanced

Figure 8. Confusion matrixes for the multivariate feature subset.

With respect to the **RIPPER** model, here we can make the same assumption as we did in **Classification Trees** for the worsening of the performance in the balanced training set, since both are based on the same principle. Besides, the multivariate filter has achieved the best performance, returning slightly poorer results than in **Classification Trees**, but also with a smaller number of rules. Probably this performance could increase significantly with a good fine tuning.

```
FINAL RULESET:
[[13=>72.0 ^ 17=8.0] V
[17=7.0] V
[5=<2.2] V
[5=2.2-4.8] V
[5=4.8-6.7]]
```

(a). All Features

```
FINAL RULESET:
[[7=>71.0] V
[7=63.0-71.0] V
[4=>52.0] V
[3=<2.4] V
[3=2.4-5.0 ^ 7=30.0-37.0]]
```

(b). Univariate Filter

```
FINAL RULESET:
[[5=>72.0] V
[3=2.2-4.7] V
[3=<2.2] V
[3=4.7-6.7] V
[3=6.7-8.1 ^ 7=<1007.9] V
[3=8.1-9.2 ^ 9=26.1-28.58] V
[3=8.1-9.2 ^ 7=1010.4-1012.4]]
```

(c). Multivariate Filter

```
FINAL RULESET:
[[2=<2.2] V
[2=2.2-4.7] V
[2=4.7-6.6] V
[2=6.6-8.1 ^ 4=>52.0] V
[2=6.6-8.1 ^ 0=>21.5] V
[2=6.6-8.1 ^ 4=46.0-52.0]]
```

(d). Wrapper

Figure 9. RIPPER Rules for the unbalance training set of each feature subsets

6. Conclusion

All this project has proved that there are several things that must be considered before even thinking on defining a good model. The pre-processing of the dataset as well as the understanding of its variables and their correlation are crucial. For instance, variables like the *Wind Direction* and the *Location* were encoded considering their relationship with the target variable and ended being some of the most important for the feature selection techniques.

On the other hand, the weights of the classes have also had an enormous impact, returning very different performances depending on the model due to this precise fact, as it was shown for the rule-based methods tested, where balancing the training sets led to worse performances.

As a rule, the feature selection has also demonstrated to effectively increase the quality of the models, where the multivariate filter in conjunction with the **CFS** technique improved the performance of the simpler methods, i.e., **kNN**, **Classification Trees** and **Rule Induction**. However, there was not a significant difference for the case of **SVM** and **ANNs**, the methods that showed the best performance, 77% and 86% respectively. What could be related to the fact that weather predictions are one of the greatest challenges for Data Science even nowadays, requiring more complex techniques to achieve decent results.

Nevertheless, a final consideration could be to remark the fact that in all cases there is a clear worsening of the Precision in order to obtain better results for the Recall. Most of the times, the number or FP was higher than the number of TN, which demonstrated a clear lack of robustness for the model. Despite of this fact, and the drawbacks present during the data pre-processing, it gave decent results, and the improvement of them is set as lines of future works.

References

[1] «Kaggle,» [En línea]. Available: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package.

[2] scikit-learn-contrib, «Github,» [En línea]. Available: https://github.com/scikit-learn-contrib/imbalanced-learn.

[3] «Scikit-Learn,» [En línea]. Available: https://scikit-learn.org/stable/index.html.

[4] «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/.

[5] «Pandas,» [En línea]. Available: https://pandas.pydata.org/.

[6] «Numpy,» [En línea]. Available: https://numpy.org/.

[7] «Seaborn,» [En línea]. Available: https://seaborn.pydata.org/index.html.

[8] «Matplotlib,» [En línea]. Available: https://matplotlib.org/.

[9] Imoscovitz, «Github,» [En línea]. Available: https://github.com/imoscovitz/wittgenstein#useful-references.

[10] «ITMO-FS,» [En línea]. Available: https://itmo-fs.readthedocs.io/en/latest/index.html.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall y W. P. Kegelmeyer, «arxiv,» [En línea]. Available: https://arxiv.org/pdf/1106.1813.pdf.

[12] B. Madhukar, «analyticsindiamag,» [En línea]. Available: https://analyticsindiamag.com/using-near-miss-algorithm-for-imbalanced-datasets/.

[13] B. Rok y L. Lusa, «BCM Bioinformatics,» [En línea]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-106.