UNIVERSITY OF COIMBRA

PATTERN RECOGNITION

FIRST MILESTONE

Weather Prediction in Australia

Author	*Students Number*
Leonardo VIEIRA	2015236155
Pedro CARVALHO	2015232484

April 24, 2019

Contents

1 Introduction

We were assigned a job to predict if rain will or wont fall at 44 different Australian locations. For that, we used the weather in Australia data-set available at *https://www.kaggle.com/jsphyg/weather-dataset-rattle-package*. There is also a possibility to use more data-sets later on this assignment to improve the classifiers quality.

1.1 Pre-processing

Although the data-set was pretty clean and usable from the start, some pre-processing was still needed. As recommended, we started by removing the *RISK MM* column from the data-set. This is done because this feature actually represents the amount of rainfall in millimeters for the next day and was the feature used to create the final target *RainTomorrow*. As so, using this feature would give a false sense of accuracy to the classifiers.

After that all the features that had more than 20% of missing values and all the measurements that had at least one missing value were removed from the final data-set. At the end of this process we had 16 features (7 were removed) and 112925 measurements (29268 were removed).

Finally, in order to have a usable data-frame, we needed to categorize and normalize our values. In order to categorize the data, we mapped out all the *yes* and *no* values to 1 and 0, and all the 16 cardinal points to integers 1 to 16. After that we scaled our data (using the scikit-learn function *StandardScaler*) to a range between -1 and 1. This was done so we could compare features in a meaningful way.

1.2 Data Analysis

In order to understand our data distributions we performed a *Kolmogorov Smirnov Test* for each feature. All of them showed a non parametric distribution.

2 Feature Selection

2.1 Kruskal-Wallis

Since we concluded that our features do not follow a normal distribution, we applied a Kruskal-Wallis test to try and relate each individual feature to the classification. We used the results of these tests to rank them and select the used features according to these rankings.

2.2 ROC

We used the ROC to obtain a ranking as described in the Kruskal-Wallis section. In order to use ROC, however, some changes must be made to the process described above. The value used to sort the features in the ROC area under curve, in order to obtain this we use each feature by itself to try and classify some examples. The classifier chosen for this was a simple LDA classifier. Once the AUC is known we sort the features accordingly

2.3 Select K Best

Select K Best is a function from *sklearn* used to select features. This function takes a *score_func* as parameter, that is used to sort the features. We decided to try and use a mutual information method as it is compatible with the non-parametric nature of our data.

3 Dimensionality Reduction

3.1 PCA

We implemented the Principal Component Analysis for dimensionality reduction using the built in *sklearn* function using the default solver. The variance obtained with each feature decreases with each increasing feature as can be seen in Figure 1. We can see that a small number of features is responsible for a big slice of the variation. This is very useful as it allows us to reduce the number of dimensions we'll be looking at our data from.

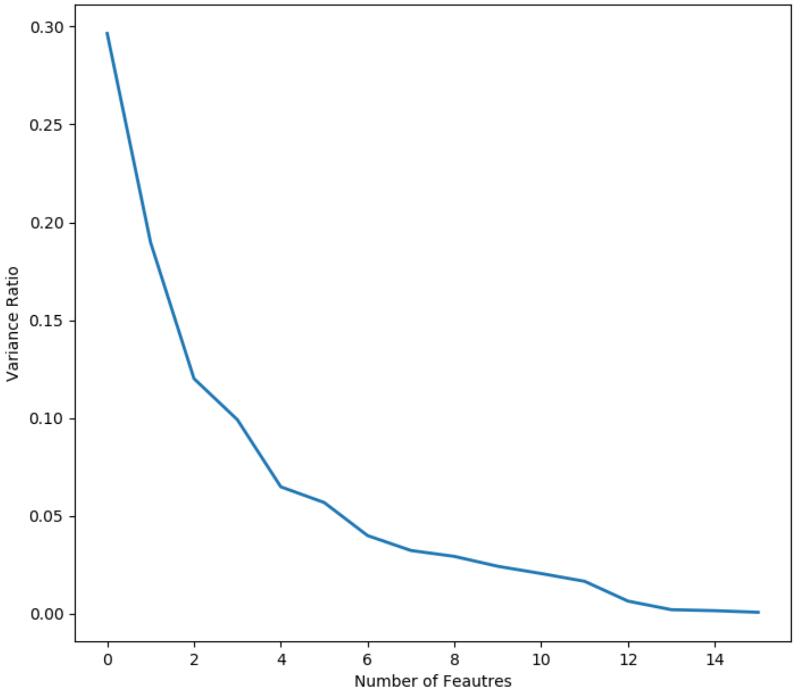

Figure 1: Variance for each of the 16 features.

3.2 LDA

A Linear Discriminant Analysis was also implemented as dimensionality reduction technique. Important to note that our problem has a binary target and as such, the LDA technique will only be able to generate 1 feature (*n_classes - 1*). Scikit-learn comes with a LDA dimensionality reduction tool out of the box so implementation was limited.

4 Classifiers

The following classifiers were implemented in order to classify the data resulting from the preprocessing.

- Euclidean MDC
- Mahalanobis MDC
- Fisher LDA

All the MDC (minimum distance classifier) were implemented using the *nearest_centroid* function from *sklearn*. Only the distance metric changed according the to desired classifier. The *Fisher LDA* classifier is a simple Linear Discriminant Analysis applied to classification instead of dimensionality reduction.

5 Pipeline and Performed Tests

A test structure was built using the *sklearn* pipeline. All the compatible combinations from the built techniques are made and registered into a *.CSV* file. Using this pipeline setup we performed the available tests to gather some initial results. The results of this initial testing can be found in the *results.csv* file included with the code. In order to compare algorithms in the next milestone we are keep track of the following values on each test run:

- F1 Score for both classes
- Precision for both classes
- Recall for both classes
- Overall accuracy
- Seed used (if appropriate)
- Pipeline used

6 Notes and Conclusions

So far we have met some challenges in our work. An initial analysis showed that no single feature can be used to obtain our target very effectively. The work described above lays the foundation from which we will polish our pattern recognition tool. Our plan going forward is to use the techniques described above to go into the rest of the project with an informed outlook, this will allow us to focus our testing on the most effective methods and make meaningful benchmarks.