Big Data - Machine Learning Homework

Will it rain tomorrow?

Luca Scofano 1762509

1 Introduction

What I aimed to do for this project was to create a simple pipeline to estimate whether tomorrow it would rain or not (in Australia). In essence I'm talking about a Classification task, where the target variable will assume a value of **1** if it rains and **0** otherwise. The pipeline of the project is:

- Exploratory Data Analysis
  - Data Exploration
  - Data Analysis
  - Pre-Processing
- Classification
- Model Evaluation

The dataset that I chose has observations that were drawn from numerous weather stations. The daily observations are available from http://www.bom.gov.au/climate/data. It's historical data that goes from **2007** to **2017**.

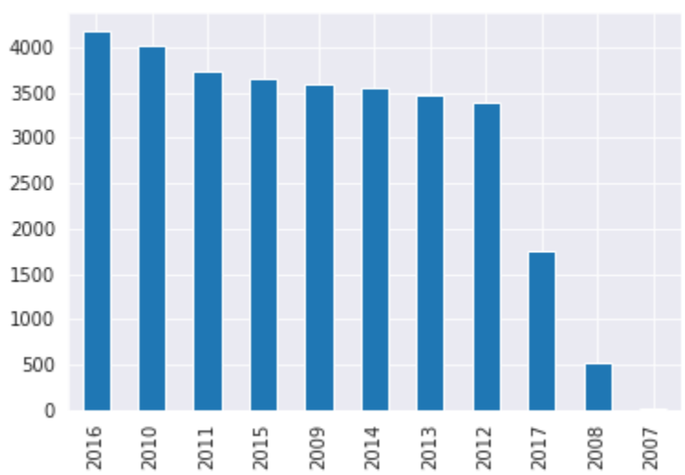

Figure 1: Number of rainy days sorted by values for each year (for all locations in Australia)

The plot clearly shows a very stable trend, without including **2008** and **2017**, being the start and the end of the dataset some values were not measured for certain months.

On the other hand **2016** and **2010** have higher values than the rest, double checking with the historical data this is true since these two years are in the top 5 most rainy years in Australia since 1900.

Before going in depth with the Data Analysis, some additional info on the dataset:

- **145460** entries with 23 columns, counting the *target variable as well*
- 16 columns are **continuous** and 7 are **categorical**

Two main issues rise from this first analysis, most variables have different scales, this could be an issue when modelling, that is why I'll address this problem later on by rescaling all variables. And the second problem is that in order to perform classification we should *encode* the categorical variables.

2 Exploratory Data Analysis

Let's start by analyzing the **target** variable, *RainTomorrow*.

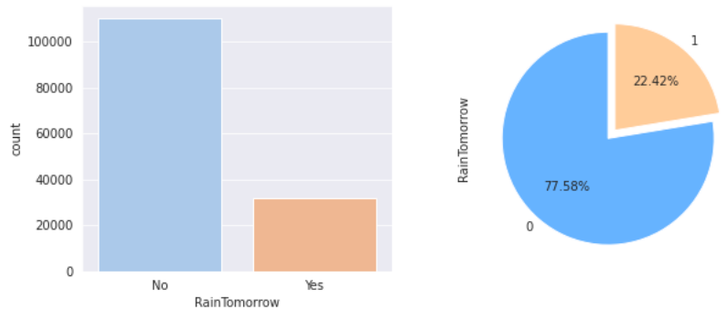

Figure 2: Target Variable

The target variable is really unbalanced, **77.6 percent** of the entire dataset has an output of **0**.

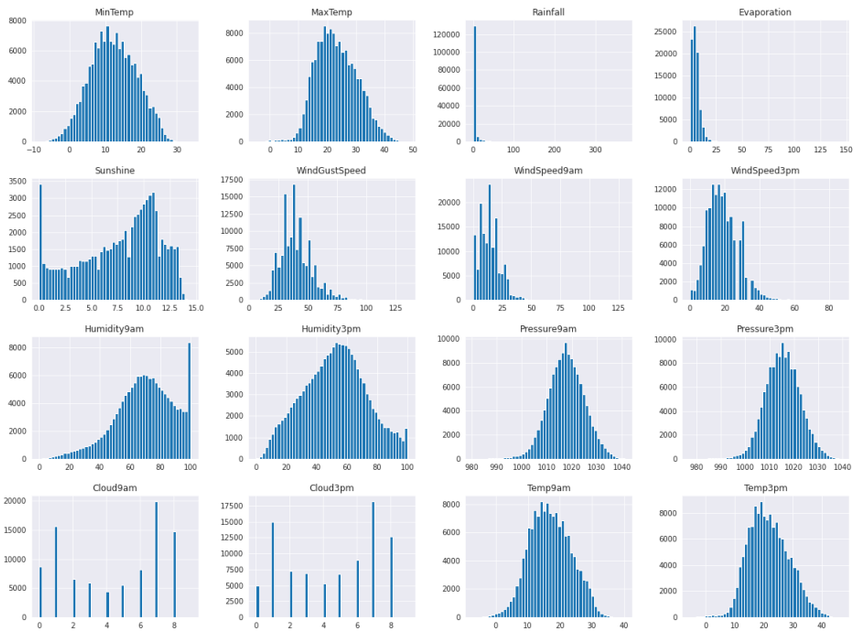

Figure 3: Distribution for all variables

- Many of these attributes, such as MinTemp, MaxTemp, Pressure9am, and Pressure3pm, have bell-shaped distributions.
- Rainfall and Evaporation are heavily positively skewed.
- WindGustSpeed, WindSpeed9am, and WindSpeed3pm are slightly less skewed to the right.
- Humidity9am and Humidity3pm are slightly skewed to the left.
- As we said before, these attributes have very different scales. For example, compare MinTemp and Pressure9am.
- Cloud9am and Cloud3pm are discrete attributes.
- The mode for Humidity9am is 100 percent. Also, Humidity3pm has an usually high number of 100 percent days given the bell-shaped distribution.

The above highlights the need for feature scaling and the transformation of attributes so they approximate to a normal distribution. Also outliers could be an issue for Rainfall, Evaporation, Sunshine, Humidity9am, and Humidity3pm.

Let's take a look at the relationship that the target variable has with some of the most significant variables. We are considering **Rainfall** and **Humidity3pm**, two of the *top 5* variables that have emerged from **SelectKBest** algorithm (based con Chi-squared)

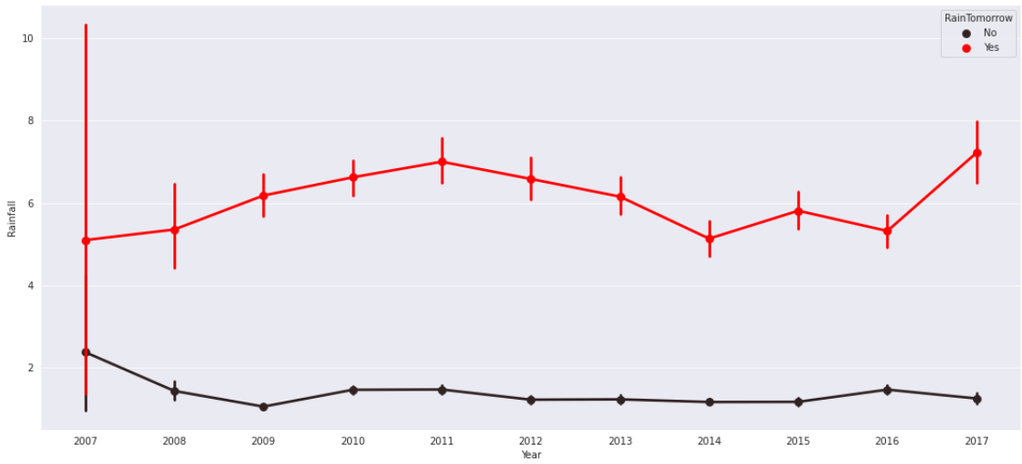

Figure 4: Cloud of word with top 15 words of both clusters.

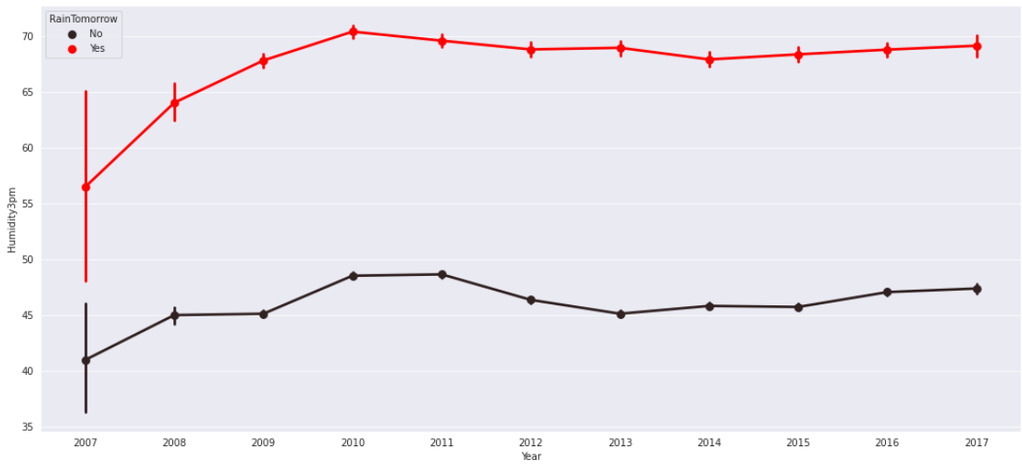

Figure 5: Cloud of word with top 15 words of both clusters.

As we thought these two variables are really significant, since both have positive trends when the values on the **x axis** are high (it's more likely to rain). The mirror explanation can be given when considering low values and the outcome of the target variable equal to **0**.

Analyzing the dataset another problem arised, **Missing Values**. This is a chart of the highest percentage of missing values.

Variable	Missing values(percentage)
Sunshine	48.0
Evaporation	43.0
Cloud3pm	41.0
Cloud9am	38.0

How did I address it? Depending on the usefulness of the data *NaN* values could be replaced with the mean, median or mode, or could be set to a particular value. In my case I decided to dismiss these variables , since the **top 3** variables are **Humidity3pm, Rainfall and RainToday**. These features were discovered thanks to **Select K best + Chi-squared**; This score can be used to select the n features with the highest values for the test chi-squared statistic from X

The chi-square test measures dependence between stochastic variables, so using this function prune the features that are the most likely to be independent of class and therefore irrelevant for classification.

Correlation between all variables

- Each pairwise correlation of MinTemp, MaxTemp, Temp9am, and Temp3pm has a moderate $(0.5 < r < 0.75) to\, strong (r > 0.75)$ relationship.
- The strongest association is between MaxTemp and Temp3pm ($r = 0.98$) followed by MinTemp and Temp9am ($r = 0.90$).
- WindGustSpeed, WindSpeed9am, WindSpeed3pm are moderately associated with each other $(0.5 < r < 0.75)$.
- There is a moderate relationship between Humidity9am and Humidity3pm $(0.5 < r < 0.75)$.

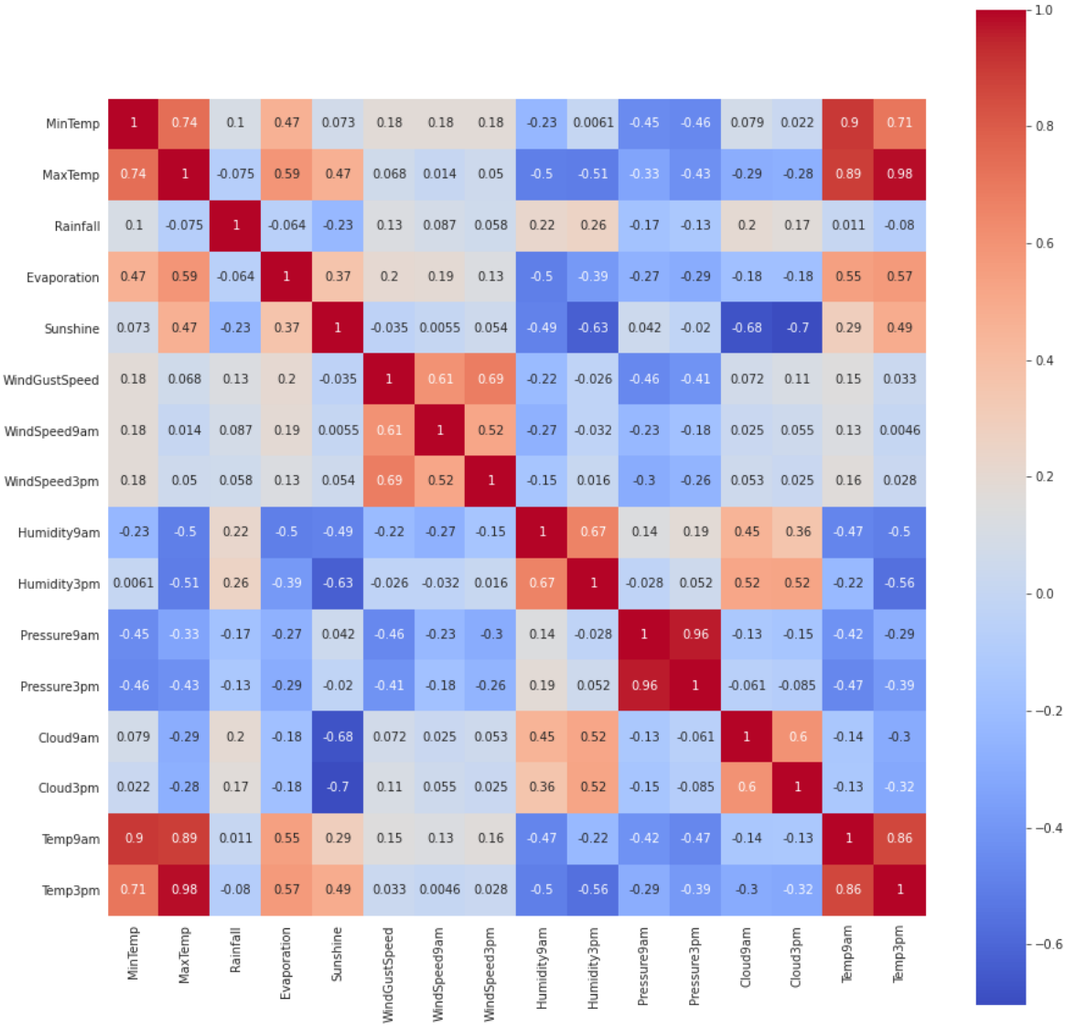

Figure 6: Correlation Heatmap

2.1 Pre-processing

I tackled different issues that I pointed out previously.

First off I drop the variables : **Sunshine,Evaporation,Cloud3pm,Cloud9am,Location,Date**. Afterwards I encode the categorical variables into dummy/indicator variables.

At last the rescaling takes place, and I picked *MinMaxScaler* for the task, it transform features by scaling each feature to a given range, in this case I went with the default one, from **0 to 1**.

3 Classification

Just as a reminder, the target variable has an outcome of **1 when it will rain** and **0 otherwise**. Since the dataset is very large the debate was whether to use the entire one, sacrificing speed (more computationally expensive) or perform some sort of dimensionality reduction, although I could have used SVD or PCA I preferred using the *Chi-square score* to measure the most dependent variables.

At the end we got **Humidity3pm, Rainfall and RainToday**

The models that I used are:

- Logistic Regression
- XGBoost
- Decision Tree

For all three of these models I used a GridSearch approach to find the best parameters in order to predict the test labels.

(Note: The dataset has been split into train 70 percent and test 30 percent)

3.1 GridSearch

Logistic Regression

In the Machine Learning world, Logistic Regression is a kind of parametric classification model.

This means that logistic regression models are models that have a certain fixed number of parameters that depend on the number of input features, and they output categorical prediction, like for example in our case if it will rain tomorrow or not.

In Logistic Regression, we don't directly fit a straight line to our data like in linear regression. Instead, we fit a S shaped curve, called Sigmoid, to our observations.

$sigmoid(x) = \frac{1}{1+e^{-x}}$

The parameters that we searched for are:

- **Penalty** That in our case could be either L1 or L2
- **C** Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
- **Solver** liblinear

XGBoost

This algorithm is based on Gradient Descent, minimizing the loss function. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks, especially classical Machine Learning algorithms. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

It can be seen as a Gradient Boosting algorithm, a special case of boosting where errors are minimized by gradient descent algorithm, but thanks to hardware and software optimization iit performs better and takes less time to compute. The key optimization is **parallelization, tree pruning, hardware optimization**

In this case I didn't perform any GridSearch because the default values are cut out for the job. Just to mention a few:

- **Max depth** Useful for tree pruning, the higher the value and the more complex and prone to overfitting the model will be, **default value = 6**
- **Gamma** The larger gamma is, the more conservative the algorithm will be, it's considered as minimum loss reduction **default value = 0**

Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning

simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

The parameters that we searched for are:

- **Criterion** The function to measure the quality of a split, **gini and entropy**
- **Max Depth 2,4,6,8,10,12**

4 Evaluation

When talking about Accuracy the final standings are:

Model	Score	Time(seconds)
XGBoost	0.8392	1.87
Logistic Regression	0.8373	0.21
Decision Tree	0.8286	0.07

Best Parameters for **Logistic Regression:** C equal to 1 and penalty equal to L2.

Best Parameters for **Decision Tree:** criterion equal to gini and max depth equal to None, which means that each tree will expand until every leaf is pure.

Further analysis

I took the analysis further to inspect how the best model performs with Precision and Recall.

The confusion matrix:

	Positive	Negative	Total
Positive	57573	2305	59878
Negative	9694	5935	15629
Total	67267	8240	75507

In other words:

$Precision = \frac{TP}{TP+FP} = 0.7202$ the accuracy of positive predictions

$Recall = \frac{TP}{TP+FN} = 0.3797$

When the model predicts it will rain tomorrow, it is correct about 72 percent of the time. Moreover, it correctly classifies 37 percent of instances in the training set where RainTomorrow is equal to Yes (i.e., by predicting Yes). Or, stated alternatively, 63 percent of the time the model predicts No for RainTomorrow when it should be Yes.

These numbers may seem disappointing, but at least the model is correctly predicting nearly 40 percent of Yes instances.

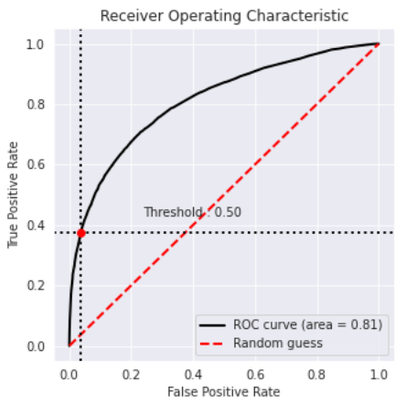

Figure 7: Roc curve

The **ROC curve** shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. As a baseline,

a random classifier is expected to give points lying along the diagonal (FPR = TPR). The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

One common approach is to calculate the area under the ROC curve, which is abbreviated to **AUC**. It is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance.

A classifier with high AUC can occassionally score worse in a specific region than another classifier with lower AUC. But generally, the AUC performs well as a measure of predictive accuracy.