How statistics could help you to predict tomorrow's weather

Have you ever complained about the incorrection of the weather forecast, and want to develop a forecast system about whether it will rain tomorrow by yourself? This report will show you how to make it realistic. In this report, we will first introduce the dataset we use, then review the statistical model used to predict whether it will rain tomorrow, and at the end, choose a final prediction approach.

Description of Data and Data pre-pocessing

The dataset collects weather information in Australia between 2007-2017. In total, there are 142193 observations and 23 predictor variables, containing information like Date, Location, Cloud, WindSpeed and so on. There is one variable called "RainTomorrow", which use "Yes" or "No" to indicate whether Australia will rain on the next day, is regarded as our response variable.

i. Data Transformation and Data Pre-processing

Firstly, some pre-possessing work has been taken to make our data more useful and efficient. There is one variable named "RISK_MM", which is the amount of rainfall in millimeters for the next day. For example, if "RISK_MM" has value greater than 0, then "RainTomorrow" will have a response "Yes". "Date" variable has been separated into "Year", "Month" and "Day", to make it more useful. As the percentage of "Yes" response in dataset without NA is 22.03%, which is around the same with that of the full dataset 22.42%. Therefore, we use the dataset without NA to make judge the variables.

ii. Feature Selection

Firstly, for numeric data, we selected the function findCorrelation from package(caret) to help us check the mean absolute correlation of each variables. We set cut-off threshold as 0.75, and removed variables MaxTemp, Pressure3pm, Temp3pm and Temp9am. Secondly, for categorical data, we did chi-square test and retained all. In addition, for variables having missing value percentage greater than 40%, we purely drop them: Sunshine (47.7%), Evaporation (42.8%) and Cloud3pm (40%).

iii. Missing Value Fill-in

For numeric variables, as we observed that there are outliers in predictor observations, especially in predictor variable: "Rainfall", "WindSpeed9am", therefore, we decided to fill in missing values with their corresponding median. For categorical observations, we filled in missing values with the most frequent observations, respectively. After all these pre-works, we have 142193 observations and 16 variables.

Statistical Model Review

80% observations of the dataset have been randomly selected as training dataset, the rest 20% goes into the testing dataset. Some of the approaches we attempted will be demonstrated as below:

i. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)

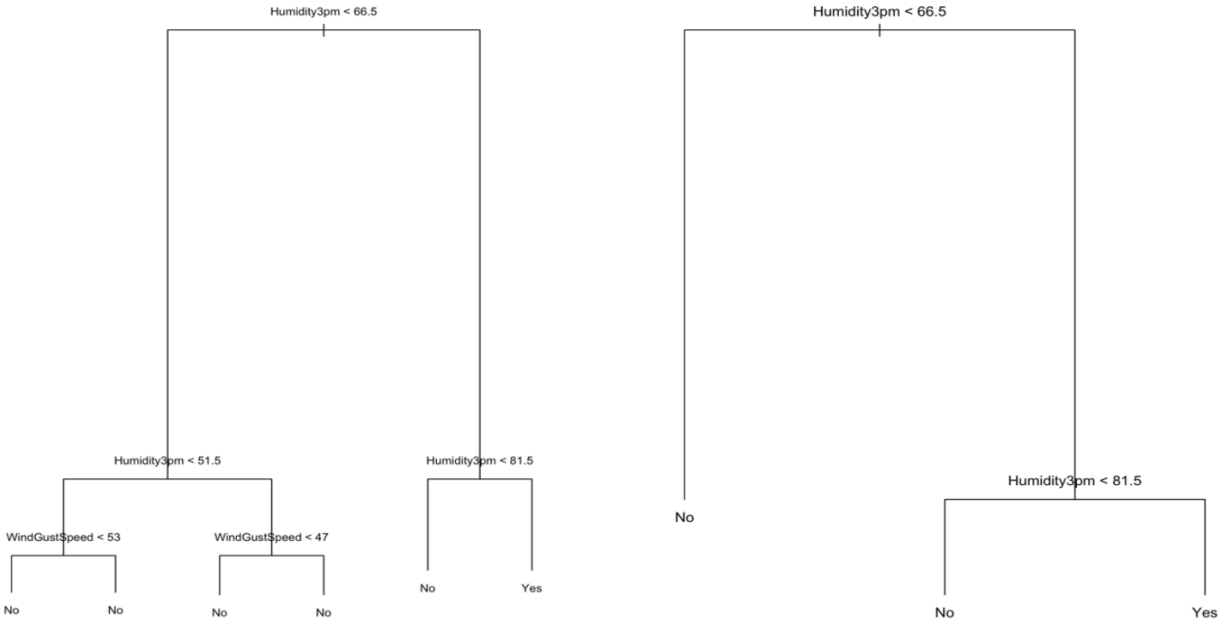
Both LDA and QDA are commonly used probabilistic methods for classifications, derived from Bayes Classifiers. LDA is like PCA, while it focuses on maximizing the separability among groups and minimizing the

distance of sample points within the same group. Its discriminant function is normally linear, and hence the resulting decision boundaries are normally linear. For LDA, we make assumptions that firstly, observations of each class are normally distributed data. Secondly, there is a common covariance across all groups. QDA follows the same assumption about the normally distributed data, while it allows each class has its own covariance matrix. As its name shows, the discriminant function for QDA is quadratic, therefore, the decision boundaries for QDA classification are quadratic as well.

The function "lda" and "qda" in R is used to generate the model. Then we predicted test dataset based on the model, and computed confusion matrix. The result illustrates that LDA has prediction accuracy 84.36% on test dataset, while QDA has that of 65.06%. Their respective misclassification rate is 15.64% and 34.94%. What is the reason behind, made QDA performs much worse than LDA? There is probability that the covariance matrixes between different classes are quite similar, thus LDA outperforms than QDA.

ii. Classification Tree

Classification tree, which is a type of decision tree, is also being applied to our dataset. What decision tree normally do is by setting a sequence of decisions to reach a result. In our case, by using decisions like whether humidity3pm is greater or smaller than 66.5%, to reach a final prediction: whether tomorrow will rain or not.



Tree.bb

Figure 1: Classification Tree

As classification tree has a limit of the amount of factor predictors, thus the location variable was dropped before the application of model. After running the R code, the model structure has been plotted out. As you could observe from the Figure 1, most of the internal nodes are asking questions relating to Humidity3pm and WindGustSpeed only. What is more, the region for Humidity3pm $< 66.5\%$ returns four terminal nodes, all showing the result that tomorrow will not rain. This is because by going through different internal nodes, there is a relatively large gap between probabilities of rain or not. To be more specific. For WindGustSpeed < 51.5 km/h, the probability that tomorrow will not rain is 0.93995, while that of WindGustSpeed > 51.5 km/h is 0.79793.

Next, cross validation was being applied to find the best tree, it turns out both size 6 and 3 returns the smallest deviance. Thus, to achieve a simpler tree, we will look at the tree with size 3. As shown in Figure 1 (attached in the appendix), the tree only depends on the predictor Humidity3pm. As a re- sult, we

have accuracy rate: 82.25%. During the previous analysis, we could conclude that as tree can be displayed graphically, it is highly interpretable. However, the accuracy rate is not very high as an expense compared with other approaches. Apart from this, the tree is dominated by the strong pre- dictor and not very robust. In other words, a small change in the data may cause a large modification to the final tree.

Due to these disadvantages, we set our sights onto the Bagging and Random Forest, which use trees as building blocks to construct prediction models and tends to give a better prediction.

iii. Random Forest

As Random Forest provide an improvement over bagged trees in some aspect, thus we will mainly talk about Random Forest here. Random Forest forces the split of variables, making sure that every predictor will have the same probability of exposure under the model as building blocks. On average, there is $(p-m)/p$ of the split will not include the strong predictors in, where m is a random sample selection of predictors and p is the number of predictors. The function randomForest is being used in the analysis. For the setup of m, we chose integer 4, which is equal to the square root of the total number of predictors. (4 √15) Then the Random Forest forces each split of subset to have 4 predictors. For the number of trees to grow, we choose the default value of the function: 500.

As a result, the accuracy rate for Random Forest is 85.66%, which improved 3.41% accuracy compared with Classification. The advantage of Random Forest is that the setting of splitting candidates' predictors decorrelates the tree, making the average of the resulting trees less variable and hence more reliable. What is more, by combining the results from many trees results leads to an improvement in prediction accuracy. The disadvantage is distinct that it has a poor interpretability.

iv. Adaboost

Fitting Adaboost on the dataset is also attempted. Adaboost is constructed on the basis of the concept of Decision Trees and Random Forests. There are three main differences between Random Forest and Adaboost: Firstly, the Random Forest tree has no limits on depth, while in a forest of trees made by Adaboost are usually have one. Secondly, in Random Forest, trees are all treated equally. In contrast, different trees in Adaboost weights differently. Thirdly, order in Adaboost is vital, each tree is taking the previous trees' mistake into account. We used the function "boosting" from package "adabag" and used the default value: 100, for the number of iterations. As a result, we received an 84.92% accuracy, which is slightly lower than that of Bagging and Random Forest. The error evolution was obtained after 100 iterations and the error is reducing. The main objective of Adaboosting is highlighting the errors. We start with the error value 0.1706554 and after 100 iteration we obtain the error as 0.1490369. Here we can see as the user increases the iterations, the error value reduces and hence shows that this is a better model for interpretations.

Summary of the final approach

The purpose of our analysis is to make a precise prediction on whether tomorrow's Australia will rain or not. Therefore, accuracy is the main indicator we are interested in. Thus, we selected 1. Accuracy: the portion of correctly classified examples, 2. Misclassification Rate: the portion of wrongly classified examples, 3. Sensitivity (True positive rate): portion of correctly classified true positive examples 4. Positive Predictive Value: the proportions of positive results that are true positive.

Indicator /Model	Logistic Regression	Classification Tree	LDA	QDA	Bagging	Random Forest	Adaboost
Accuracy	84.39	82.25	84.36	65.06	85.41	85.66	84.92
Misclassification Rate	15.61	17.75	15.64	34.94	14.59	14.34	15.08
Pos Pred Value	86.58	82.62	86.79	88.62	88.22	88.60	87.35

Indicator /Model	Logistic Regression	Classification Tree	LDA	QDA	Bagging	Random Forest	Adaboost
Sensitivity	94.57	97.71	94.22	63.14	93.73	93.59	94.23

To have a clear view of the above table, the ranking is displayed as below:

Indicator /Rank	1	2	3	4	5	6	7
Accuracy	Random Forest	Bagging	Adaboost	Logistic Regression	LDA	Classification Tree	QDA
Misclassification Rate	Random Forest	Bagging	Adaboost	Logistic Regression	LDA	Classification Tree	QDA
Pos Pred Value	QDA	Random Forest	Bagging	Adaboost	LDA	Logistic Regression	Classification Tree
Sensitivity	Classification Tree	Logistic Regression	Bagging	Random Forest	Adaboost	LDA	QDA

Then by judging the overall performance, we found that Random Forest always outperforms. Given our aim is to predict accurately, we will select Random Forest as the final approach.

Conclusion

Australia's climate is governed mostly by the hot, sinking air of the subtropical high-pressure belt, which moves north-west and north-east with the seasons. Because it has a wide variety of climates due to its large geographical size, thus practically, if you want to make a more precise prediction of the weather, you could extract information from the local meteorological observatory of your province.

Apart from this practical sense, several more points to conclude:

To begin with, although the running time was not specifically recorded, but during the code production, around 113k training dataset took Random Forest, Adaboost around 20 minutes to deliver a model output. When dealing with moderate or large dataset, data pre-processing is extremely vital. Feature extraction techniques could help us to refine data. Many times, we also need the help of dimensionality reduction.

In addition, logistic regression, LDA, QDA, classification tree, random forest, bagging, and Ada boost are attempted to help us to make predictions. We could observe that there is always a trade-off between implement and prediction accuracy. Classification tree, which could be directly shown graphically, has a bad performance in prediction accuracy. While Random Forest, Bagging, which predict relatively more precise, are difficult to explain to non-experts in real life. The model delivers the most accurate results, does not always be the best choice consistently. Depending on your analysis purpose, you need to balance between interpretation and prediction accuracy.