Predicting Precipitation in Australian Cities

Jessica Eloy, Chris Robinson, Andrew Zazueta

University of San Diego

Master of Science, Applied Data Science

502

Section 2

April 19, 2021

Author Note

We have no known conflict of interest to disclose.

Correspondence concerning this article should be addressed to Jessica Eloy, Chris Robinson, or Andrew Zazueta, at University of San Diego, 5998 Alcala Park, San Diego, CA 92110. Emails: jeloy@sandiego.edu, christopherrobinson@sandiego.edu, azazueta@sandiego.edu

Abstract

Predicting weather patterns is important for agriculture, businesses, and for everyday people. In this paper, the objective is to predict whether there will be rain on specific days or not using different machine learning algorithms. There are multiple methods for conducting these types of predictions. Commonly, the use of a Doppler radar is deployed to monitor could movement, or other methods like examining high- and low-pressure zones can be utilized to see how a storm might move. The methodology used in this paper examines factors like humidity, cloud cover, precipitation amount, and atmospheric pressure to accurately predict if rain will occur the following day or not. The results gave accuracies above 80% for multiple models and in multiple locations around Australia. The cities that were used to train and test these models were Perth, Melbourne, and Sydney. The machine learning algorithms that made these predictions were Random Forests, C5.0, and Naïve Bayes.

Keywords: Australia, machine learning, precipitation, modeling

Contents

Data Set and the Objective of Study

The data set for this project was taken from a Kaggle competition page. There are 23 variables within the data set and each row is a different date corresponding to a certain location. The dataset has 10 years of weather predictions from different locations across Australia, and in total, there are 145,460 rows (not including the header) and 23 columns. The question presented for this study was: Can you "predict next-day rain by training classification models on the target variable 'RainTomorrow'" (Young, 2021). The columns that were helpful in making these predictions were 'Rainfall,' 'Humidity,' 'Pressure,' and 'Cloud.'

The 'Rainfall' feature gave a numerical value in millimeters of how much rain happened that day in a city in Australia. The 'Humidity' feature is a percentage of the humidity at 9am and 3pm that day. The 'Pressure' feature is measured in hectopascal pressure units (hPa) and was "reduced to [the] mean sea level" (2021) at 9am and 3pm. The 'Cloud' feature is the fraction of sky obscured by cloud cover at 9am and 3pm. Lastly, the response variable 'RainTomorrow' is a binary feature taking the values 'Yes' or 'No.' The rainfall must be more than 1mm for the 'RainTomorrow' variable to be yes. The objective of this study is to use different data cleaning methods, data prepping methods, and predictive modeling strategies learned this semester and apply them to this data set using R.

Overview and Plan of Project

Before machine learning algorithms could be applied to the data set, it needed to be cleaned and prepared. Once these steps were taken, 3 data frames were made from the original set. Each data frame was in a different location in Australia (Perth, Melbourne, and Sydney). Since 'RainTomorrow' is a categorical/logical variable, machine learning algorithms like decision trees, C5.0, Naive Bayes, CART, neural networks, etc. can be used to predict if it will take the values ‘Yes’ or ‘No.’ The three algorithms that were chosen to predict our response variable were Random Forests, Naïve Bayes, and C5.0. These were applied to the three data frames to compare the effectiveness of each one and to examine which one yielded the most accurate results.

Data Preparation

Cleaning and Preparatory Phase

The data preparation phase consists of cleaning and preparing the data for analysis. For our data mining project, it was necessary to check different features in the dataset to see if they were needed in the prediction in order to reduce the dimensionality. We began by identifying missing values, partitioning training and test sets for each city, finally checking the class balance to ensure the values were not imbalanced.

Missing Values

Many of the values in ‘Evaporation’ and ‘Sunshine’ were found to be missing, 98.7% and 98.2% respectively, these values were omitted from the data. To reduce dimensionality, the features ‘WindGustDir’, ‘WindDir9am’, and ‘WindDir3pm’ were removed. Since we are analyzing one location at a time the wind direction was found to not have influence on our response variable ‘RainTomorrow’. However, if we were comparing two cities in close proximity, wind direction could have an influence. When analyzing ‘RainToday’, it was noticed that when this particular value contained a missing value, a lot of other features in the same row also contained missing values. It was determined the best course of action was to delete the rows containing this pattern.

The original data from Kaggle was then broken out into three separate data frames, one for each city in Australia. Once broken up we identified that Sydney was the only dataset missing a large portion of the values in the feature ‘WindGustSpeed’ (31%), and thus the feature was removed.

Finally, in some instances, it was required to replace missing values with the average of those values during a certain week. For example, if a value in the ‘Cloud3pm’ feature was missing, the data from three days prior and three days after was added together and then divided by six to find the average and put back into the data set to replace the missing values. This step is performed last due to the size of the original data set.

Training and Test Data Sets

For the three data sets the data was partitioned into training and testing data sets. Since our data set was large, it was necessary to have more records in the training set, around 75-90% of the original data. How this task was accomplished can be seen in the code in our appendix.

Checking class balance

The last step performed during the data preparation phase was checking the class balance. This was performed to ensure the data was not too imbalanced for modeling. Performing the appropriate code, we received the following return:

Table 1

Number of "Yes" and "No" responses for the variables "RainToday" and "RainTomorrow"

Values	No	Yes
Melbourne Rain Today	1718	580
	0.75	0.25

Sydney Rain Today	2465 0.75	866 0.26
Perth Rain Today	2548 0.8	645 0.2
Melbourne Rain Tomorrow	1765 0.77	533 0.23
Sydney Rain Tomorrow	2468 0.74	863 0.26
Perth Rain Tomorrow	2548 0.8	645 0.2

Looking at the outcome, 20-26% of the values were 'Yes' in each data frame, so rebalancing was not necessary.

Data Mining and Evaluation of Results

The preliminary results from the data mining task to predict rainfall in Australian cities seemed encouraging. Each of the three models had high accuracy and low error rates, however the precision was lower and the sensitivity for each city and model were all 0.5 or below. We were trying to predict precipitation, so accuracy was an important evaluation metric, but because there are much fewer rainy days than sunny days it is also important to look at the sensitivity and precision as well. Precision and sensitivity are both key measures for our model because the cost of false positives and false negatives are relatively equal when predicting rainfall. In addition, we have an uneven distribution between "Yes" and "No" values in our dataset with a large portion being true negatives. According to Koo Ping Shung, "F1 Score might be a better measure to use if we need to seek a balance between precision and recall and there is an uneven class distribution" (2018). The $F_{0.5}$ and F_2 scores are less important because, as stated earlier, sensitivity and precision are equally important to us. We created three models, random forest, naïve bays, and C5.0, in attempt to predict rainfall in Sydney, Perth, and Melbourne.

As shown in figure 1.2, the random forest model for the Sydney subset had an accuracy of 0.831 with an error rate of 0.169. Sensitivity, specificity, and precision were 0.5, 0.945, and 0.759 respectively. The F_1 score was .61. The random forest model for the Perth subset had an accuracy of 0.857 with an error rate of 0.143. Sensitivity, specificity, and precision were 0.468, 0.962, and 0.769 respectively. The F_1 score was .582. The random forest model for the Melbourne subset had an accuracy of 0.797 with an error rate of 0.203. Sensitivity, specificity, and precision were 0.319, 0.940, and 0.614 respectively. The F_1 score was 0.420. The random forest model performed better for the Sydney and Perth subsets, but had a much lower F_1 score with the Melbourne subset.

Sydney (Random Forest)				
		Predicted		
		No	Yes	Total
Actual	No	TN 553	FP 32	585
	Yes	FN 101	TP 101	202
	Total	654	133	787

Perth (Random Forest)				
		Predicted		
		No	Yes	Total
Actual	No	TN 607	FP 24	631
	Yes	FN 91	TP 80	171
	Total	698	104	802

Melbourne (Random Forest)				
		Predicted		
		No	Yes	Total
Actual	No	TN 425	FP 27	452
	Yes	FN 92	TP 43	135
	Total	517	70	587

Figure 1.1 Random Forest Model Contingency Tables.

Random Forest Model			
Evaluation Measure	**Sydney**	**Perth**	**Melbourne**
Accuracy	0.831	0.857	0.797
Error Rate	0.169	0.143	0.203
Sensitivity	0.500	0.468	0.319
Specificity	0.945	0.962	0.940
Precision	0.759	0.769	0.614
F_1	0.603	0.582	0.420
F_2	0.537	0.508	0.352
$F_{0.5}$	0.688	0.681	0.518

Figure 1.2 Random Forest Model evaluation measures.

As shown in figure 1.4, the naïve bays model for the Sydney subset had an accuracy of 0.831 with an error rate of 0.187. Sensitivity, specificity, and precision were 0.421, 0.949, and 0.739 respectively. The F_1 score was 0.536. The naïve bays model for the Perth subset had an accuracy of 0.848 with an error rate of 0.152. Sensitivity, specificity, and precision were 0.509, 0.940, and 0.696 respectively. The F_1 score was 0.588. The naïve bays model for the Melbourne subset had an accuracy of 0.785 with an error rate of 0.215. Sensitivity, specificity, and precision were 0.363, 0.912, and 0.551 respectively. The F_1 score was 0.438. Similar to the random forest model, the naïve bays model performed better with both the Sydney and Perth subsets, but had the lowest F_1 score with the Melbourne subset. The naïve bays model performed better for Melbourne and worse for Sydney but had a similar result for Perth.

Sydney (Naïve Bayes)

Actual		Predicted: No	Predicted: Yes	Total
	No	TN 555	FP 30	585
	Yes	FN 117	TP 85	202
	Total	672	115	787

Perth (Naïve Bayes)

Actual		Predicted: No	Predicted: Yes	Total
	No	TN 593	FP 38	631
	Yes	FN 84	TP 87	171
	Total	677	125	802

Melbourne (Naïve Bayes)

Actual		Predicted: No	Predicted: Yes	Total
	No	TN 412	FP 40	452
	Yes	FN 86	TP 49	135
	Total	498	89	587

Figure 1.3 Naïve Bays Model Contingency Tables.

Naïve Bayes Model

Evaluation Measure	**Sydney**	**Perth**	**Melbourne**
Accuracy	0.813	0.848	0.785
Error Rate	0.187	0.152	0.215
Sensitivity	0.421	0.509	0.363
Specificity	0.949	0.940	0.912
Precision	0.739	0.696	0.551
F_1	0.536	0.588	0.438
F_2	0.460	0.538	0.390
$F_{0.5}$	0.642	0.648	0.499

Figure 1.4 Naïve bays evaluation measures.

As shown in figure 1.6, the C5.0 model for the Sydney data subset had an accuracy of 0.836 with an error rate of 0.164. Sensitivity, specificity, and precision were 0.500, 0.952, and 0.765 respectively. The F_1 score was 0.605. The C5.0 model for the Perth data subset had an accuracy of 0.852 with an error rate of 0.148. Sensitivity, specificity, and precision were 0.433, 0.965, and 0.771 respectively. The F_1 score was 0.554. The C5.0 model for the Melbourne data subset had an accuracy of 0.813 with an error rate of 0.187. Sensitivity, specificity, and precision were 0.311, 0.962, and 0.712 respectively. The F_1 score was 0.433. Like the previous two models, the C5.0 model performed better with both the Sydney and Perth subsets, but had much lower F_1 score with the Melbourne subset.

Sydney (C5.0)				
		Predicted		
		No	Yes	Total
Actual	No	TN 557	FP 28	585
	Yes	FN 98	TP 101	202
	Total	655	132	787

Perth (C5.0)				
		Predicted		
		No	Yes	Total
Actual	No	TN 609	FP 22	631
	Yes	FN 97	TP 74	171
	Total	706	96	802

Melbourne (C5.0)				
		Predicted		
		No	Yes	Total
Actual	No	TN 435	FP 17	452
	Yes	FN 93	TP 42	135
	Total	528	59	587

Figure 1.5 C5.0 Model Contingency Table for Melbourne Australia.

C5.0 Model			
Evaluation Measure	**Sydney**	**Perth**	**Melbourne**
Accuracy	0.836	0.852	0.813
Error Rate	0.164	0.148	0.187
Sensitivity	0.500	0.433	0.311
Specificity	0.952	0.965	0.962
Precision	0.765	0.771	0.712
F_1	0.605	0.554	0.433
F_2	0.537	0.474	0.351
$F_{0.5}$	0.692	0.667	0.566

Figure 1.6 C5.0 Model evaluation measures.

Conclusion

Overall, the random forest model had a better result for both Sydney and Perth, but the lowest score for Melbourne. If we had to choose a single model to run on all three cities the random forest would be our choice. There may be some factors associated with Melbourne that make it more difficult to predict which led to the higher percentage of false negatives in our models. The fact that Melbourne had the smallest sample of data may have also contributed to the difference in outcomes compared to Sydney and Perth which had a similar subset size. Geography may also play a factor. While we tried to choose cities with similar latitude and topography, other factors based on location may be contributing to the difference in performance between Melbourne and the other two cities.

Ultimately, given the results, different variables may have to be chosen for each city. Additionally, information not contained in the current dataset may also need to be considered in order to produce the best results. "There are many factors that influence weather, many of which we cannot see" (Climate and Weather, n.d.), which is why weather prediction is such an interesting topic for data mining.

References

Young, J. (2021, January). *Rain in Australia*. Kaggle. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

Shung, K. (2018, March 15). Accuracy, Precision, Recall or F1? *Towards Data Science*. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Climate and Weather. (n.d.). *Factors that Influence Weather*. Climate and Weather. https://www.climateandweather.net/world-weather/factors-that-influence-weather/

Appendix
Predicting Rainfall in Australian Cities Code

Jesica Eloy Chris Robinson Andrew Zazueta

Obtaining Data and Setting up Libraries

```
setwd("C:/Users/mzazu/OneDrive/Documents/USD papers/502/weatherAUS.csv")
library("tidyverse")
library("randomForest")
library("lubridate")
library("e1071")
library("C50")
library("rpart")
weatherAUS <- read_csv("weatherAUS.csv")
```

Cleaning and Preporation Phase

Part 1: Exploratory Data Analysis and and Handling Missing Values

The first step is to explore the data to see if there are any missing values that need to be handled. Also, checking different features to see if they are needed in the prediction is necessary to reduce the dimensionality.

```
# Evaporation values are missing from 98.7%  of the data set and Sunshine values
# are missing from 98.2% of the data set, so they will be removed

length(which(is.na(weatherAUS$Evaporation))) / dim(weatherAUS)[1]
```

```
## [1] 0.9874605
```

```
length(which(is.na(weatherAUS$Sunshine))) / dim(weatherAUS)[1]
```

```
## [1] 0.9815138
```

```
weatherAUS <- weatherAUS %>%
  select(-c(Evaporation, Sunshine))

# To reduce the dimensionality of the data set, the features which contain
# information on wind direction will be removed. This is because we will be
# separating the data by location. Since we will only be looking at one location
# at a time, the direction of the wind would not influence on the "RainTomorrow"
# feature. The wind direction could have an influence if we were comparing two
# cities in close proximity with on another.
```

```
weatherAUS <- weatherAUS %>%
  select(-c(WindGustDir, WindDir9am, WindDir3pm))

# When the feature "RainToday" contains an NA, a lot of other features in the
# data set also have missing values. For this reason, the simplest solution is
# to delete these rows to avoid fudging data.

weatherAUS <- weatherAUS[complete.cases(weatherAUS$RainToday),]

# The feature "RainTomorrow" is our response variable, so any missing value from
# this column cannot be replaced.

weatherAUS <- weatherAUS[complete.cases(weatherAUS$RainTomorrow),]

# Making a data frame for different cities

weatherSydney <- weatherAUS %>%
  filter(Location == "Sydney")

weatherMelbourne <- weatherAUS %>%
  filter(Location == "Melbourne")

weatherPerth <- weatherAUS %>%
  filter(Location == "Perth")

# Unlike other locations, Sydney is missing a large portion of the values (31%)
# in the "WindGustSpeed" feature, so it will be removed.

length(which(is.na(weatherSydney$WindGustSpeed) == TRUE)) / dim(weatherSydney)[1]
```

```
## [1] 0.3104173
```

```
weatherSydney <- weatherSydney %>%
  select(-c(WindGustSpeed))

# The rest of the missing values will be replaced with the average of those
# values during a certain week. For example, if a "MinTemp" value is missing,
# the data from 3 days prior and 3 days after will be added together and then
# divided by 6 to find the average "MinTemp." This step is performed now and not
# earlier because of how large the entire data set is. There might be instances
# where we divide by 6 even though the sum of six numbers was not found (due to
# NA's being close together). This is alright because we are already making
# assumptions on what number should be filling the NA.

NA_replace <- function(df) {
  for(j in 1:ncol(df)){
    for(i in 1:nrow(df)){
      if(is.na(df[i,j]) == TRUE && i > 3){
        avg <- sum(df[(i-3):(i+3),j], na.rm = TRUE) / 6
        df[i,j] <- avg
      }
    }
```

```
  }
  return(df)
}

weatherSydney <- NA_replace(weatherSydney)
weatherMelbourne <- NA_replace(weatherMelbourne)
weatherPerth <- NA_replace(weatherPerth)
```

Part 2: Making training and test sets

Due to the data set being large, we will want to have more records in the training set (75-90 percent of original data).

```
# setting seed

set.seed(7)

# Weather Melbourne

# identify how many records

MEL <- dim(weatherMelbourne)[1]

# determine which records are in training set

train_ind <- runif(MEL) < 0.75

# create training and test sets

MELtrain <- weatherMelbourne[ train_ind, ]
MELtest <- weatherMelbourne[ !train_ind, ]

# Weather Sydney

SYD <- dim(weatherSydney)[1]
train_ind <- runif(SYD) < 0.75
SYDtrain <- weatherSydney[ train_ind, ]
SYDtest <- weatherSydney[ !train_ind, ]

# Weather Perth

PER <- dim(weatherPerth)[1]
train_ind <- runif(PER) < 0.75
PERtrain <- weatherPerth[ train_ind, ]
PERtest <- weatherPerth[ !train_ind, ]
```

Part 3: Checking Class Balance

The last step in the data preparation phase is making sure classes are not too imbalanced for our modeling.

```r
# RainToday and RainTomorrow Yes and No counts

t1 <- table(weatherMelbourne$RainToday)
t2 <- table(weatherSydney$RainToday)
t3 <- table(weatherPerth$RainToday)
t4 <- table(weatherMelbourne$RainTomorrow)
t5 <- table(weatherSydney$RainTomorrow)
t6 <- table(weatherPerth$RainTomorrow)

t7 <- rbind(t1, round(prop.table(t1), 2), t2, round(prop.table(t2), 2), t3,
            round(prop.table(t3), 2), t4, round(prop.table(t4), 2), t5,
            round(prop.table(t5), 2), t6, round(prop.table(t6), 2))
rownames(t7) <- c("Melbourne Rain Today", " ","Sydney Rain Today", " ",
                  "Perth Rain Today", " ", "Melbourne Rain Tomorrow", " ",
                  "Sydney Rain Tomorrow", " ", "Perth Rain Tomorrow", " ")
t7
```

```
##                              No    Yes
## Melbourne Rain Today    1718.00 580.00
##                            0.75   0.25
## Sydney Rain Today       2465.00 866.00
##                            0.74   0.26
## Perth Rain Today        2548.00 645.00
##                            0.80   0.20
## Melbourne Rain Tomorrow 1765.00 533.00
##                            0.77   0.23
## Sydney Rain Tomorrow    2468.00 863.00
##                            0.74   0.26
## Perth Rain Tomorrow     2548.00 645.00
##                            0.80   0.20
```

Taking a look at the values, 20-26% of the values are 'Yes' in each data frame, so no re-balancing is needed.

Choosing Data Mining Task

The purpose of this project is to predict the 'RainTomorrow' future. Since this is a categorical/logical variable, machine learning algorithms like decision trees, C5.0, Naive Bayes, CART, neural networks, etc. can be used. The three algorithms that were chosen to predict our response variable are Random Forests, Naive Bayes, and C5.0. These will be applied to the three data frames we made to compare the effectiveness of each one and to examine which one yields the most accurate results. In total, 9 models will be made.

Applying Algorithms to Find Best Model

Each model made will have a contingency table made with it.

```r
# Sydney Models

# Setting up data set for model usage

SYDtrain$RainTomorrow <- factor(SYDtrain$RainTomorrow)
SYDtest$RainTomorrow <- factor(SYDtest$RainTomorrow)
```

```r
# Random Forest

rf01 <- randomForest(formula = RainTomorrow ~ Rainfall + Humidity3pm
                     + Cloud3pm, data = SYDtrain, ntree = 100, type = "classification")



ypred <- predict(rf01, SYDtest)

t_n <- table(SYDtest$RainTomorrow, ypred)
row.names(t_n) <- c("Actual: no", "Actual: yes")
colnames(t_n) <- c("Predicted: no", "Predicted: yes")
t_n <- addmargins(A = t_n, FUN = list(Total = sum), quiet = TRUE)
t_n
```

```
##              ypred
##               Predicted: no Predicted: yes Total
##   Actual: no            553             32   585
##   Actual: yes           101            101   202
##   Total                 654            133   787
```

```r
# Naive Bayes

nb01 <- naiveBayes(formula = RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm,
                   data = SYDtrain)

ypred2 <- predict(object = nb01, newdata = SYDtest)

t_n2 <- table(SYDtest$RainTomorrow, ypred2)
row.names(t_n2) <- c("Actual: no", "Actual: yes")
colnames(t_n2) <- c("Predicted: no", "Predicted: yes")
t_n2 <- addmargins(A = t_n2, FUN = list(Total = sum), quiet = TRUE)
t_n2
```

```
##              ypred2
##               Predicted: no Predicted: yes Total
##   Actual: no            555             30   585
##   Actual: yes           117             85   202
##   Total                 672            115   787
```

```r
# C5.0

C5 <- C5.0(RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm, data = SYDtrain)

ypred3 <- predict(object = C5, newdata = SYDtest)

t_n3 <- table(SYDtest$RainTomorrow, ypred3)
row.names(t_n3) <- c("Actual: no", "Actual: yes")
colnames(t_n3) <- c("Predicted: no", "Predicted: yes")
t_n3 <- addmargins(A = t_n3, FUN = list(Total = sum), quiet = TRUE)
t_n3
```

```
##             ypred3
##              Predicted: no Predicted: yes Total
##   Actual: no            557             28   585
##   Actual: yes            98            104   202
##   Total                 655            132   787
```

```
# Perth Models

PERtrain$RainTomorrow <- factor(PERtrain$RainTomorrow)
PERtest$RainTomorrow <- factor(PERtest$RainTomorrow)

# Random Forest

rf02 <- randomForest(formula = RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm,
                     data = PERtrain, ntree = 100, type = "classification")

ypred4 <- predict(rf02, PERtest)

t_n4 <- table(PERtest$RainTomorrow, ypred4)
row.names(t_n4) <- c("Actual: no", "Actual: yes")
colnames(t_n4) <- c("Predicted: no", "Predicted: yes")
t_n4 <- addmargins(A = t_n4, FUN = list(Total = sum), quiet = TRUE)
t_n4
```

```
##             ypred4
##              Predicted: no Predicted: yes Total
##   Actual: no            607             24   631
##   Actual: yes            91             80   171
##   Total                 698            104   802
```

```
# Naive Bayes

nb02 <- naiveBayes(formula = RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm,
                   data = PERtrain)

ypred5 <- predict(object = nb02, newdata = PERtest)

t_n5 <- table(PERtest$RainTomorrow, ypred5)
row.names(t_n5) <- c("Actual: no", "Actual: yes")
colnames(t_n5) <- c("Predicted: no", "Predicted: yes")
t_n5 <- addmargins(A = t_n5, FUN = list(Total = sum), quiet = TRUE)
t_n5
```

```
##             ypred5
##              Predicted: no Predicted: yes Total
##   Actual: no            593             38   631
##   Actual: yes            84             87   171
##   Total                 677            125   802
```

```
# C5.0

C5_PERTH <- C5.0(RainTomorrow ~ Humidity3pm, data = PERtrain)
```

```r
ypred6 <- predict(object = C5_PERTH, newdata = PERtest)

t_n6 <- table(PERtest$RainTomorrow, ypred6)
row.names(t_n6) <- c("Actual: no", "Actual: yes")
colnames(t_n6) <- c("Predicted: no", "Predicted: yes")
t_n6 <- addmargins(A = t_n6, FUN = list(Total = sum), quiet = TRUE)
t_n6
```

```
##              ypred6
##               Predicted: no Predicted: yes Total
##   Actual: no            609             22   631
##   Actual: yes            97             74   171
##   Total                 706             96   802
```

```r
# Melbourne Models

MELtrain$RainTomorrow <- factor(MELtrain$RainTomorrow)
MELtest$RainTomorrow <- factor(MELtest$RainTomorrow)

# Random Forest

rf03 <- randomForest(formula = RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm,
                     data = MELtrain, ntree = 100, type = "classification")

ypred7 <- predict(rf03, MELtest)

t_n7 <- table(MELtest$RainTomorrow, ypred7)
row.names(t_n7) <- c("Actual: no", "Actual: yes")
colnames(t_n7) <- c("Predicted: no", "Predicted: yes")
t_n7 <- addmargins(A = t_n7, FUN = list(Total = sum), quiet = TRUE)
t_n7
```

```
##              ypred7
##               Predicted: no Predicted: yes Total
##   Actual: no            425             27   452
##   Actual: yes            92             43   135
##   Total                 517             70   587
```

```r
# Naive Bayes

nb03 <- naiveBayes(formula = RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm,
                   data = MELtrain)

ypred8 <- predict(object = nb03, newdata = MELtest)

t_n8 <- table(MELtest$RainTomorrow, ypred8)

row.names(t_n8) <- c("Actual: no", "Actual: yes")
colnames(t_n8) <- c("Predicted: no", "Predicted: yes")
t_n8 <- addmargins(A = t_n8, FUN = list(Total = sum), quiet = TRUE)
t_n8
```

```
##              ypred8
##               Predicted: no Predicted: yes Total
##   Actual: no            412             40   452
##   Actual: yes            86             49   135
##   Total                 498             89   587
```

```r
# C5.0

C5_MEL <- C5.0(RainTomorrow ~ Rainfall + Humidity3pm + Cloud3pm, data = MELtrain)

ypred9 <- predict(object = C5_MEL, newdata = MELtest)

t_n9 <- table(MELtest$RainTomorrow, ypred9)
row.names(t_n9) <- c("Actual: no", "Actual: yes")
colnames(t_n9) <- c("Predicted: no", "Predicted: yes")
t_n9 <- addmargins(A = t_n9, FUN = list(Total = sum), quiet = TRUE)
t_n9
```

```
##              ypred9
##               Predicted: no Predicted: yes Total
##   Actual: no            435             17   452
##   Actual: yes            93             42   135
##   Total                 528             59   587
```