Rain in Australia. Classification Prediction Model

by Sumaira Afzal, Viraja Ketkar, Murlidhar Loka, Vadim Spirkov

Abstract Many native cultures comprise an institution of "rainmakers" – people who would not as much invoke the rains, but anticipate them based on ethno-meteorology. The forecasting was based on skillful art of observing the natural environment as expressed in the timing or flowering of plants, hatching of insects, arrival of migratory birds, etc., which enables farmers to make adjustments in farming calendar and crop selection types in any given season. This indigenous knowledge was often passed down from one generation to the other. We are going to employ CRISP-DM framework (Ref: Jiawei Han (2012)) along with the latest scientific methods and prediction algorithms to achieve the same very goal without thorough knowledge of forces of nature, hopefully with the same accuracy as the aboriginal people.

Background

Weather forecasting is a complex and often challenging skill that involves observing and processing vast amounts of data. Weather systems can range from small, short lived thunderstorms only a few kilometers in diameter that last a couple hours to large scale rain and snow storms up to a thousand kilometers in diameter and lasting for days.

A very important component of modern weather forecasting is the use of numerical weather prediction (NWP) models. In the last years, the forecast quality of those models constantly improved, mostly due to major improvements in high performance computing. NWP focuses on taking current observations of weather and processing these data with computer models to forecast the future state of weather. Knowing the current state of the weather is just as important as the numerical computer models processing the data. Current weather observations serve as input to the numerical computer models through a process known as data assimilation to produce outputs of temperature, precipitation, and hundreds of other meteorological elements from the oceans to the top of the atmosphere. Society

Objective

The objective of this research is to find a supervised, binary classification model that would provide accurate forecast of the rain in Australia next day, having today's weather observations and historical data. In addition to being accurate the model should be easily interpretable and flexible enough to accept limited number of input features without diminishing its prediction power.

Data Analysis

The data set we are going to use for our research contains daily weather observations from numerous Australian weather stations collected from 2007 till 2017. There are over 142000 records. It has been sourced from Kaggle

Data Dictionary

We exclude the variable *Risk-MM* when training your binary classification model. If we don't exclude it, you will leak the answers to our model and reduce its predictability

Column Name	Column Description
Date	Date of observation
Location	Common name of the location of the weather station
MinTemp	Minimum temperature in degrees Celsius
MaxTemp	Maximum temperature in degrees Celsius
Rainfall	Amount of rainfall recorded for the day in mm
Evaporation	So-called Class A pan evaporation (mm) in the 24 hours to 9am

Column Name	Column Description
Sunshine	Number of hours of bright sunshine in the day
WindGustDir	Direction of the strongest wind gust in the 24 hours to midnight
WindGustSpeed	Speed (km/h) of the strongest wind gust in the 24 hours to midnight
WindDir9amDirection	Of the wind at 9am
WindDir3pmDirection	Of the wind at 3pm
WindSpeed9amWind	Wind speed (km/hr) averaged over 10 minutes prior to 9am
WindSpeed3pmWind	Wind Speed (km/hr) averaged over 10 minutes prior to 3pm
Humidity9amHumidity	Humidity (percent) at 9am
Humidity3pmHumidity	Humidity (percent) at 3pm
Pressure9amAtmospheric	Pressure (hpa) reduced to mean sea level at 9am
Pressure3pmAtmospheric	Pressure (hpa) reduced to mean sea level at 3pm
Cloud9amFraction	Area of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eights. It records how many eights of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast
Cloud3pmFraction	Area of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cloud9am for a description of the values
Temp9amTemperature	Temperature (degrees C) at 9am
Temp3pmTemperature	Temperature (degrees C) at 3pm
RainTodayBoolean	Rainy today. 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
RISK_MM	Amount of rain. A kind of measure of the "risk". This column is redundant and will be dropped
RainTomorrow	**Class label. Will it rain tomorrow?**

Data Exploration

Let's take a close look at the data set. We start with loading weather observations from the file into a data frame. We remove *RISK_MM* as explained and convert *Date* column to "date"' data type.

```
weatherData = read.csv("../data/weatherAUS.csv", header = TRUE, na.strings = c("NA","","#NA"),sep=",")
weatherData = subset(weatherData, select = -RISK_MM)
weatherData$Date = as.Date(as.character(weatherData$Date),"%Y-%m-%d")
```

Now we are going to load coordinates of the weather stations and have a bird-eye view of the weather station locations.

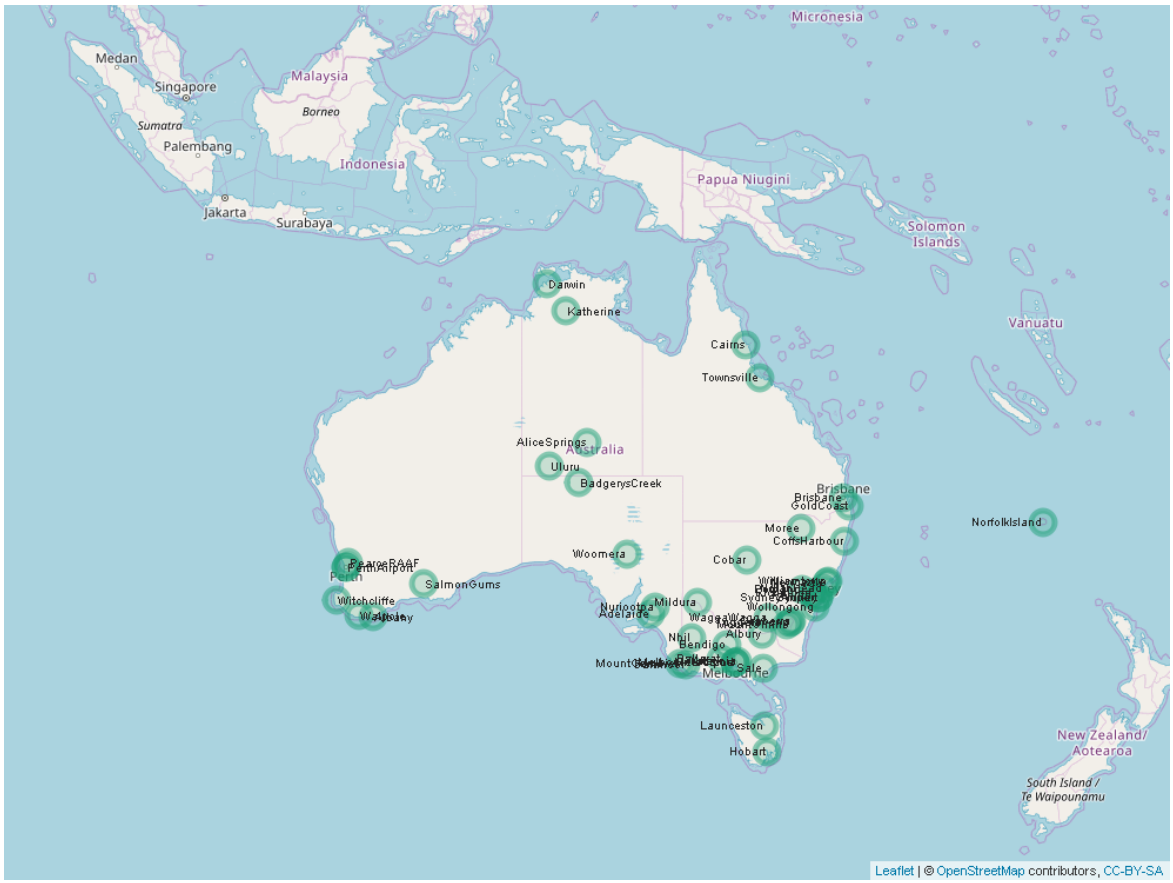

Figure 1: Australian Weather Stations

To have the full picture of the data let's print the data summary and sample.

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir
Min. :2007-11-01	Canberra: 3418	Min. :-8.50	Min. :-4.80	Min. : 0.00	Min. : 0.00	Min. : 0.00	W : 9780
1st Qu.:2011-01-06	Sydney : 3337	1st Qu.: 7.60	1st Qu.:17.90	1st Qu.: 0.00	1st Qu.: 2.60	1st Qu.: 4.90	SE : 9309
Median :2013-05-27	Perth : 3193	Median :12.00	Median :22.60	Median : 0.00	Median : 4.80	Median : 8.50	E : 9071
Mean :2013-04-01	Darwin : 3192	Mean :12.19	Mean :23.23	Mean : 2.35	Mean : 5.47	Mean : 7.62	N : 9033
3rd Qu.:2015-06-12	Hobart : 3188	3rd Qu.:16.80	3rd Qu.:28.20	3rd Qu.: 0.80	3rd Qu.: 7.40	3rd Qu.:10.60	SSE : 8993
Max. :2017-06-25	Brisbane: 3161	Max. :33.90	Max. :48.10	Max. :371.00	Max. :145.00	Max. :14.50	(Other):86677
	(Other) :122704	NA's :637	NA's :322	NA's :1406	NA's :60843	NA's :67816	NA's : 9330

WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am
Min. : 6.00	N :11393	SE :10663	Min. : 0	Min. : 0.00	Min. : 0.00	Min. : 0.00	Min. : 980.5
1st Qu.: 31.00	SE : 9162	W : 9911	1st Qu.: 7	1st Qu.:13.00	1st Qu.: 57.00	1st Qu.: 37.00	1st Qu.:1012.9
Median : 39.00	E : 9024	S : 9598	Median : 13	Median :19.00	Median : 70.00	Median : 52.00	Median :1017.6
Mean : 39.98	SSE : 8966	WSW : 9329	Mean : 14	Mean :18.64	Mean : 68.84	Mean : 51.48	Mean :1017.7
3rd Qu.: 48.00	NW : 8552	SW : 9182	3rd Qu.: 19	3rd Qu.:24.00	3rd Qu.: 83.00	3rd Qu.: 66.00	3rd Qu.:1022.4
Max. :135.00	(Other):85083	(Other):89732	Max. :130	Max. :87.00	Max. :100.00	Max. :100.00	Max. :1041.0
NA's :9270	NA's :10013	NA's : 3778	NA's :1348	NA's :2630	NA's :1774	NA's :3610	NA's :14014

Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
Min. : 977.1	Min. :0.00	Min. :0.0	Min. :-7.20	Min. :-5.40	No :109332	No :110316
1st Qu.:1010.4	1st Qu.:1.00	1st Qu.:2.0	1st Qu.:12.30	1st Qu.:16.60	Yes : 31455	Yes: 31877
Median :1015.2	Median :5.00	Median :5.0	Median :16.70	Median :21.10	NA's: 1406	
Mean :1015.3	Mean :4.44	Mean :4.5	Mean :16.99	Mean :21.69		
3rd Qu.:1020.0	3rd Qu.:7.00	3rd Qu.:7.0	3rd Qu.:21.60	3rd Qu.:26.40		
Max. :1039.6	Max. :9.00	Max. :9.0	Max. :40.20	Max. :46.70		
NA's :13981	NA's :53657	NA's :57094	NA's :904	NA's :2726		

Table 2: `Weather Obesrvations Data Summary`

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am
1	14214.00	Albury	13.40	22.90	0.60			W	44	W	WNW	20
2	14215.00	Albury	7.40	25.10	0.00			WNW	44	NNW	WSW	4
3	14216.00	Albury	12.90	25.70	0.00			WSW	46	W	WSW	19
4	14217.00	Albury	9.20	28.00	0.00			NE	24	SE	E	11
5	14218.00	Albury	17.50	32.30	1.00			W	41	ENE	NW	7
6	14219.00	Albury	14.60	29.70	0.20			WNW	56	W	W	19
7	14220.00	Albury	14.30	25.00	0.00			W	50	SW	W	20
8	14221.00	Albury	7.70	26.70	0.00			W	35	SSE	W	6
9	14222.00	Albury	9.70	31.90	0.00			NNW	80	SE	NW	7
10	14223.00	Albury	13.10	30.10	1.40			W	28	S	SSE	15

WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
24	71	22	1007.70	1007.10	8		16.90	21.80	No	No
22	44	25	1010.60	1007.80			17.20	24.30	No	No
26	38	30	1007.60	1008.70		2	21.00	23.20	No	No
9	45	16	1017.60	1012.80			18.10	26.50	No	No
20	82	33	1010.80	1006.00	7	8	17.80	29.70	No	No
24	55	23	1009.20	1005.40			20.60	28.90	No	No
24	49	19	1009.60	1008.20	1		18.10	24.60	No	No
17	48	19	1013.40	1010.10			16.30	25.50	No	No
28	42	9	1008.90	1003.60			18.30	30.20	No	Yes
11	58	27	1007.00	1005.70			20.10	28.20	Yes	No

Table 3: `Weather Obesrvations Data Sample`

Next set of plots renders distribution of a few important features. Overall all the features of the data set could be split into a few groups: temperature observations, humidity, wind speed, wind direction, cloud coverage, pressure, evaporation and sunshine. We picked one parameter from each group assuming that they represent well the remaining group attributes.

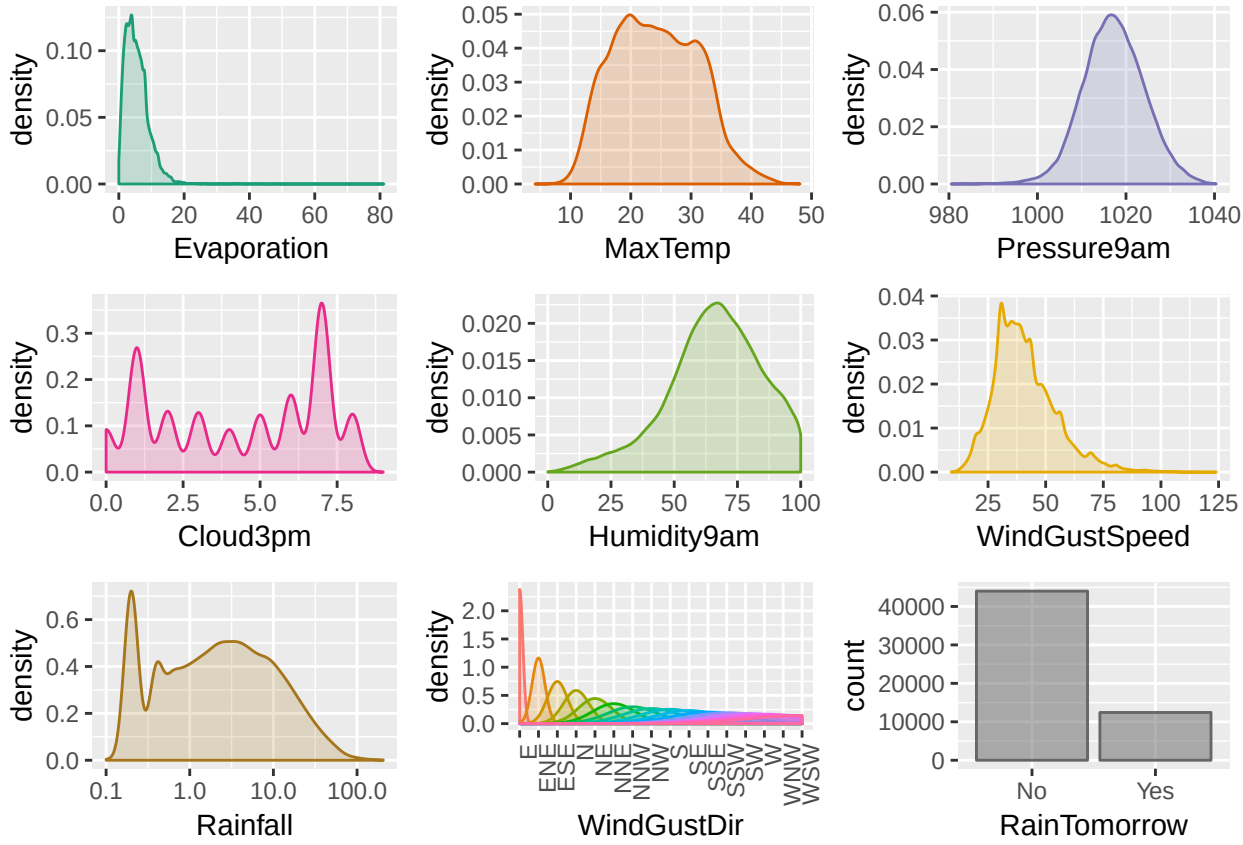

Figure 2: Distribution of Important Features

Missing Data

Data summary shows that some features miss data. Some data losses are very significant. We are going to identify what data is missing and if it is feasible to recover the missing data.

```
sort(colSums(is.na(weatherData)), decreasing = T)
```

```
#>      Sunshine   Evaporation      Cloud3pm      Cloud9am   Pressure9am
#>         67816         60843         57094         53657         14014
#>   Pressure3pm    WindDir9am   WindGustDir WindGustSpeed    WindDir3pm
#>         13981         10013          9330          9270          3778
#>   Humidity3pm      Temp3pm  WindSpeed3pm   Humidity9am      Rainfall
#>          3610          2726          2630          1774          1406
#>     RainToday  WindSpeed9am      Temp9am       MinTemp       MaxTemp
#>          1406          1348           904           637           322
#>          Date      Location  RainTomorrow
#>             0             0             0
```

To speed up data processing and plot rendering we are going to use a data sample. For population of 142K observations, 30K sample size would be sufficient for 99% confidence level with the confidence interval 1.

```
weatherSample = sample_n(weatherData, SampleSize)
aggr(weatherSample, numbers = F, prop = T, col = mainPalette, sortVars = T, bars = F, varheight = T)
```

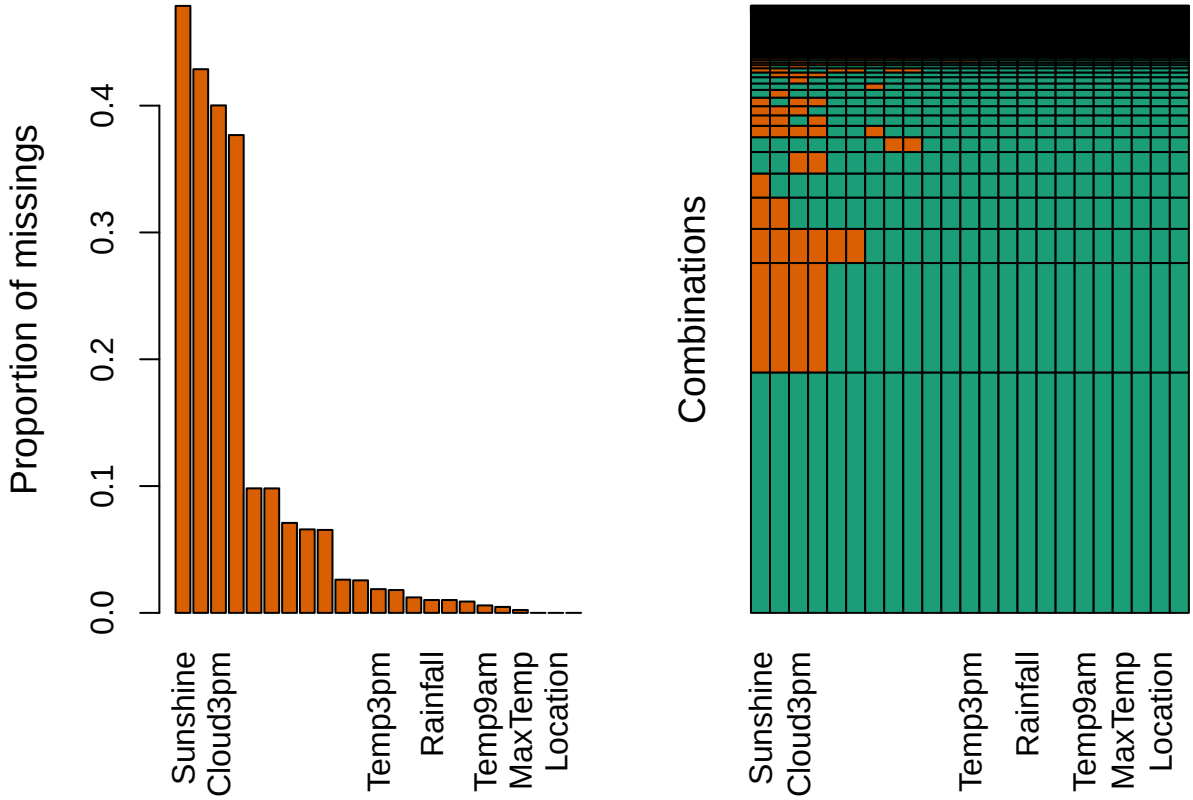

Figure 3: Missing Data Summary

```
#>
#>  Variables sorted by number of missings:
#>       Variable       Count
#>       Sunshine 0.478666667
#>    Evaporation 0.428733333
#>       Cloud3pm 0.400200000
#>       Cloud9am 0.376800000
#>    Pressure9am 0.098200000
#>    Pressure3pm 0.098166667
#>     WindDir9am 0.070966667
#>    WindGustDir 0.065833333
#>  WindGustSpeed 0.065400000
#>     WindDir3pm 0.026166667
```

```
#>    Humidity3pm 0.025666667
#>       Temp3pm 0.018733333
#>   WindSpeed3pm 0.018000000
#>    Humidity9am 0.012200000
#>       Rainfall 0.010133333
#>      RainToday 0.010133333
#>   WindSpeed9am 0.008933333
#>       Temp9am 0.005900000
#>       MinTemp 0.004700000
#>       MaxTemp 0.002266667
#>          Date 0.000000000
#>      Location 0.000000000
#>   RainTomorrow 0.000000000
```

As demonstrated in Figure 3 *Sunshine*, *Evaporation* and *Clouds* attributes suffer the loss of data between **48%** and **38%**. This is significant! Since we are dealing with the weather we should be observing cyclical patterns. The next group of plots illustrate missing data distribution for the most damaged features. We employ VIM (Ref: Matthias Templ) package to plot the missing data.

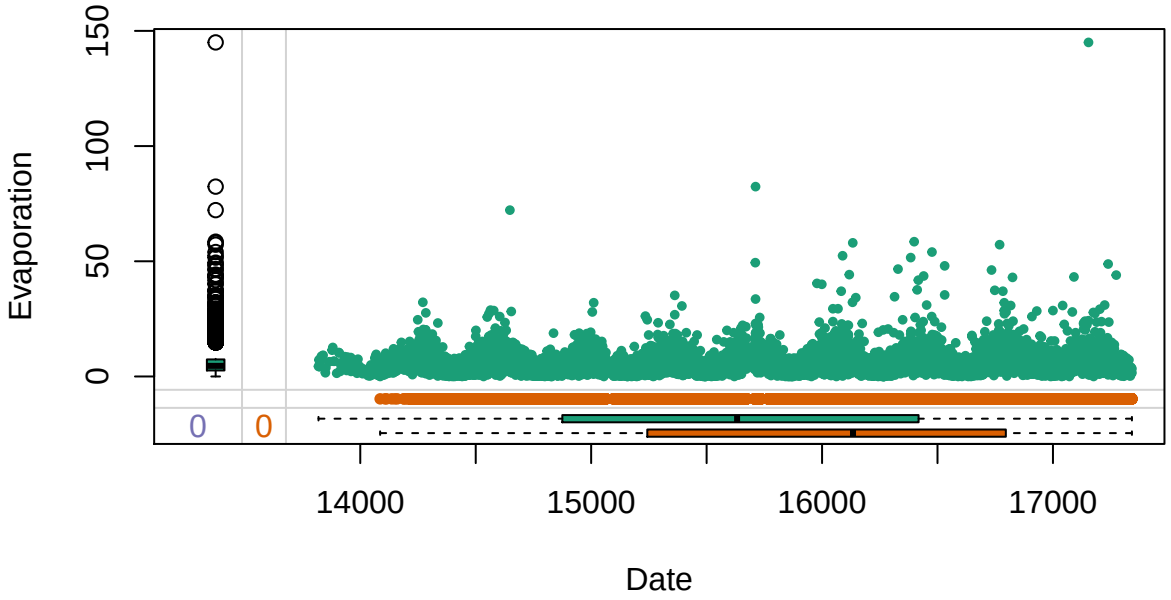

Figure 4: Date/Evaporation Margin Plot

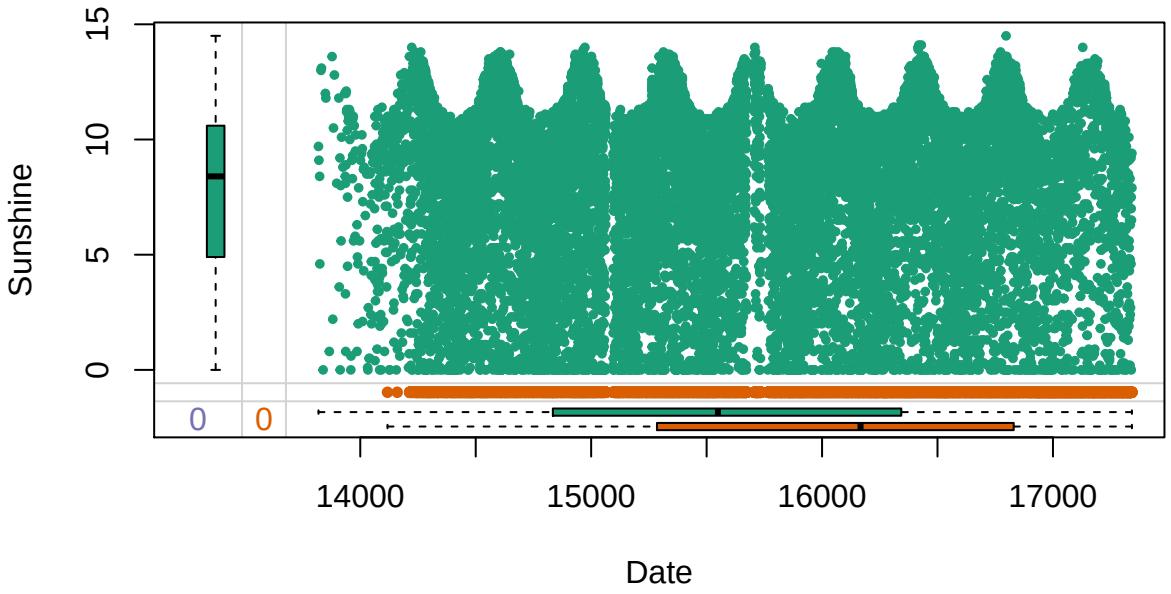

Figure 5: Date/ Sunshine Margin Plot

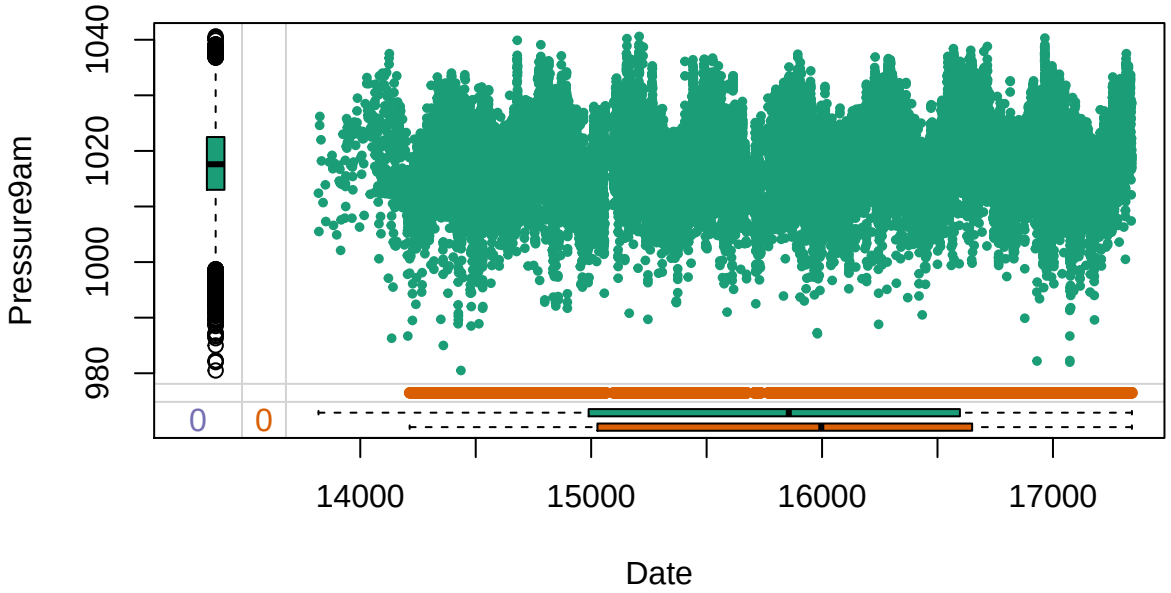

Figure 6: Date/ Pressure3pm Margin Plot

So what do the margin plots tell us? First of all let's take a look at *Date* axis. The *Date* has been converted to number to ensure continuous flow of the data. All features we picked exhibit cyclical pattern as expected. Along the vertical axis we observe the box plot of the respective feature. *Evaporaton* data is quite remarkable (Figure 4); it has very narrow distribution and a lot of so-called

outliers. Though the forces of nature follow seasonal patters they often exhibit wide range of seasonal anomalies, which the plots highlight. Thus we opt to keep the data as is. The distribution of the missing data of a given feature is depicted along the horizontal axis. In all three cases the missing data is randomly distributed over the observed date range. Along the horizontal axis we may see the box plots of the date and a given feature. Pressure readings at 9 AM *Presure9am* (Figure 6) distributed evenly across the observed date range. *Evaporation* and *Sunshine* exhibit more data losses towards the end of the observed period.

The next plot examines another dimension of the missing data, namely missing date grouped by location.

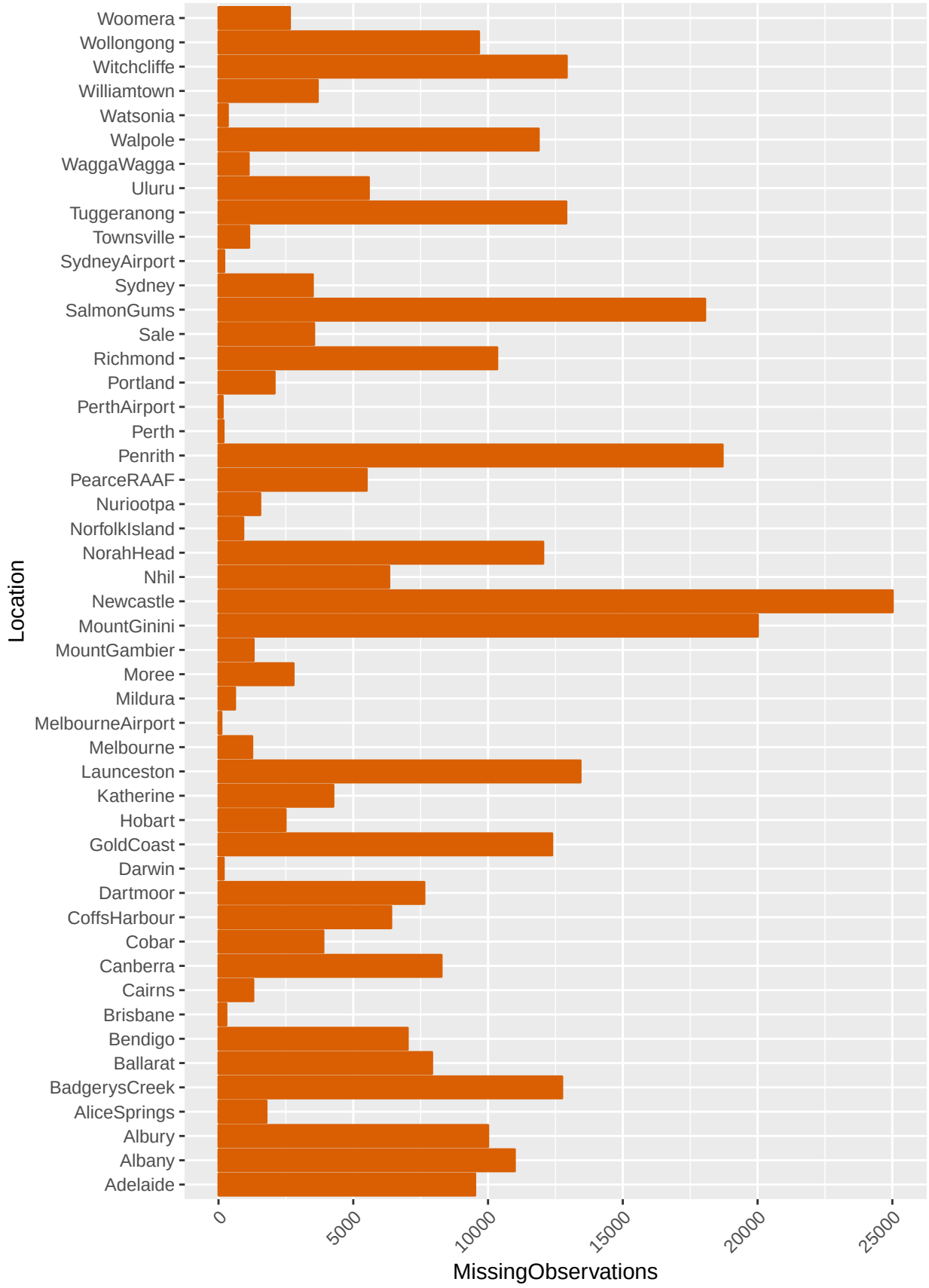

Figure 7: Missing Data By Location

Figure 7 shows that on average a location misses **6460** observations. This is significant! Though if we take a second look at the weather station map 1 we would see that Mount Gini (the station that

misses the most data), Bendigo and Ballarat are close to Melbrun, where the staff kept observing data on regular basis. Newcastle to Sydney and so on... Thus knowing that the stations are relatively close geographically we could potentially employ the weather reading collected by one station to approximate the missing data of the other station, provided they are located nearby.

Data correlation and other observations

Let's examine how the features are correlated to each other. Knowing weather we can make an assumption that the temperature features should be highly correlated, as well as pressure, wind speed, clouds and humidity groups.

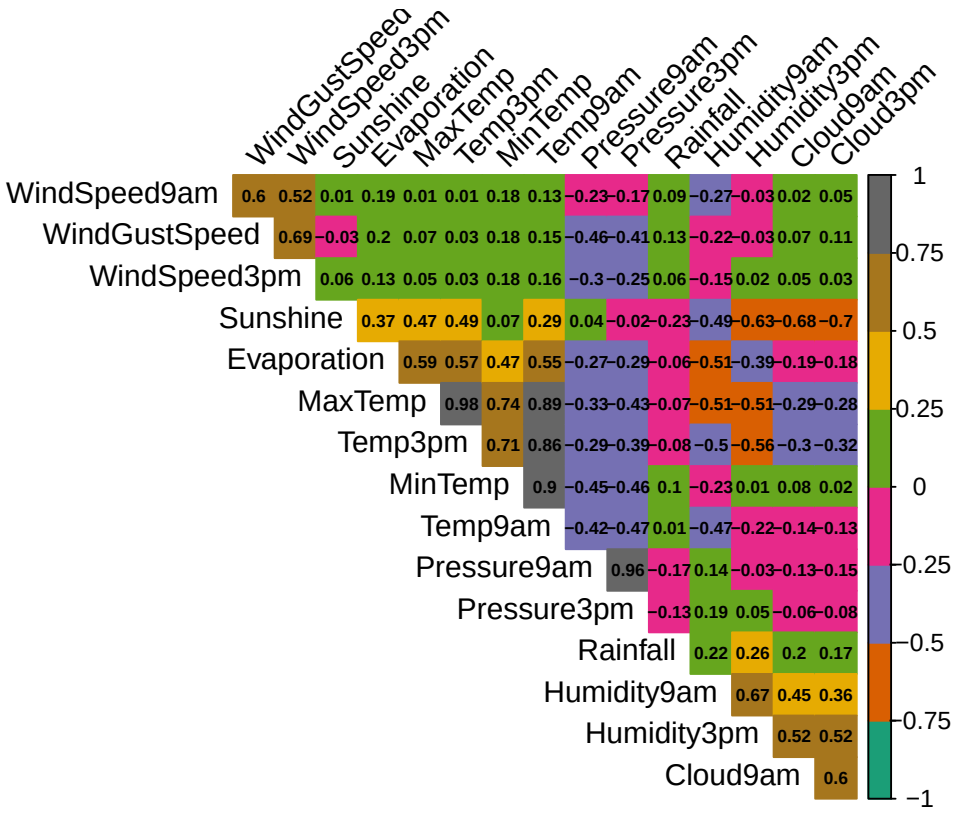

Figure 8: Data Correlation

Figure 8 confirms our initial guess. This observation will help us to eliminate redundant features later when we get to the point of selecting useful predictors for our model.

Takeaways from Data Exploration Excersize

- The data we are dealing with suffer major observation losses (Figure 3)
- The least represented features are
  - *Sunshine* **48%**.
  - *Evaporation* **43%**.
  - *Cloud* group (**40%** and **38%** respectively).
- The rest of the features exhibit medium to minor data losses, where *Pressure* group leads the way with 10%
- The missing data is distributed randomly over the observed time frame (Figures 4, 5, 6).
- We also witnessed that some weather stations recorded less data and some were almost prefect at record keeping (Figure 7). Luckily majority of the weather stations situate relatively close to each other (see figure 1). Thus if a station has data gaps the neighboring station data could be used to approximate the missing data with plausible accuracy.
- We have also noticed that many features are either positively or negatively correlated (Figure 8), where
  - *MaxTemp*, *Temp3pm* and *Temp9am* exhibit correlation of **0.86** to **0.98**.

  - *Pressure9am* and *Pressure3pm* have correlation coefficient of **0.96**.
  - *Sunshine* and *Cloud* group correlated negatively with coefficient of **-0.7**.
  - *Rainfall* feature is of particular interest since this is what we are trying to predict. Unfortunately it does not demonstrate any strong correlations with any other features.

- Examining the data we have also seen seasonal patterns and the data that fall outside of the normal distribution range by far (outliers). Those are anomalies of nature which we opt to keep.
- The last but not least the target feature *RainTomorrow* (the value we are trying to predict) is unbalanced. So we are dealing with unbalanced data set. See Figure 2.

Data Preparation

Data exploration confirmed that despite significant data loss we should be able to impute missing data with high degree of accuracy.

Data Imputing

To impute the missing data we employ **MICE** package (Ref: Stef van Buuren). Our imputation strategy is to employ **Predictive mean matching** model. It is a robust, fast imputation algorithm that works with numeric values. Prior to applying the algorithm we have to do some data transformations, which will

- Increase imputation processing speed.
- Improve model training performance and hopefully accuracy.

First of all let's get rid of *Date* column. Outside of the presentation it does not carry too mach information. What would be useful indeed is a feature that captures seasonal fluctuations. That would bee *month* and *day* combined, giving us year-round (365) days of observations.

Secondly we convert *Location* categorical feature (Factor) to plain numeric column. Indeed there are 49 locations. If we dummy-encode locations how much would it improve the performance of the imputation algorithm and the processing speed (we are talking about 49 new columns…)? We believe it is more harmful than helpful. Thus we go for numeric presentation. Another topic for pondering is whether we shall employ weather station coordinates… After some deliberation we can conclude that the coordinates will not add much knowledge in the context of the model training. But they will certainly break the categorical nature of the locations. Each coordinates have 4 - 6 decimal places, which effectively makes them continuous. So we stick with numeric presentation of the locations.

We also convert other factor data types to numbers in order to make *MICE* pick **Predictive mean matching** model (vs *Multinomial logit* model, that exhibits poor speed and low performance dealing with the categorical features of 20 levels or more).

This is our original set.

```
#> 'data.frame':    142193 obs. of  23 variables:
#>  $ Date         : Date, format: "2008-12-01" "2008-12-02" ...
#>  $ Location     : Factor w/ 49 levels "Adelaide","Albany",..: 3 3 3 3 3 3 3 3 3 3 ...
#>  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
#>  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
#>  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
#>  $ Evaporation  : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ Sunshine     : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ WindGustDir  : Factor w/ 16 levels "E","ENE","ESE",..: 14 15 16 5 14 15 14 14 7 14 ...
#>  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
#>  $ WindDir9am   : Factor w/ 16 levels "E","ENE","ESE",..: 14 7 14 10 2 14 13 11 10 9 ...
#>  $ WindDir3pm   : Factor w/ 16 levels "E","ENE","ESE",..: 15 16 16 1 8 14 14 14 8 11 ...
#>  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
#>  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
#>  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
#>  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
#>  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
#>  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
#>  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
#>  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
#>  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
#>  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
```

```
#>  $ RainToday    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 2 ...
#>  $ RainTomorrow : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 2 1 ...
```

Transformation.

```
data = mutate(weatherData,MMDD = as.numeric( format(Date, "%m%d")),Location = unclass(Location),
              WindGustDir = unclass(WindGustDir),
              WindDir9am = unclass(WindDir9am), WindDir3pm = unclass(WindDir3pm),
              RainToday = unclass(RainToday)-1, RainTomorrow = unclass(RainTomorrow)-1)
data =  subset(data, select = -Date)
```

Resulting data frame structure.

```
#> 'data.frame':    142193 obs. of  23 variables:
#>  $ Location     : int  3 3 3 3 3 3 3 3 3 3 ...
#>  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
#>  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
#>  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
#>  $ Evaporation  : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ Sunshine     : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ WindGustDir  : int  14 15 16 5 14 15 14 14 7 14 ...
#>  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
#>  $ WindDir9am   : int  14 7 14 10 2 14 13 11 10 9 ...
#>  $ WindDir3pm   : int  15 16 16 1 8 14 14 14 8 11 ...
#>  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
#>  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
#>  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
#>  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
#>  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
#>  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
#>  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
#>  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
#>  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
#>  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
#>  $ RainToday    : num  0 0 0 0 0 0 0 0 0 1 ...
#>  $ RainTomorrow : num  0 0 0 0 0 0 0 0 1 0 ...
#>  $ MMDD         : num  1201 1202 1203 1204 1205 ...
```

Lets do a dry run first to see what predictors and methods for each target feature *MICE* software chooses. we will be employing **quickpred()** method of the *MICE* package. As before we will be working with a 30K data sample. Imputation process on the whole set takes about 3 hours and 20 minutes to complete!

```
meta = mice(data, maxit = 0, print = FALSE)
weatherSample = sample_n(data, SampleSize)
methods = meta$method
predictors = quickpred(data)
print(methods)
```

```
#>      Location       MinTemp       MaxTemp      Rainfall   Evaporation
#>            ""          "pmm"         "pmm"         "pmm"         "pmm"
#>      Sunshine   WindGustDir WindGustSpeed    WindDir9am    WindDir3pm
#>         "pmm"         "pmm"         "pmm"         "pmm"         "pmm"
#>  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm   Pressure9am
#>         "pmm"         "pmm"         "pmm"         "pmm"         "pmm"
#>   Pressure3pm      Cloud9am      Cloud3pm       Temp9am       Temp3pm
#>         "pmm"         "pmm"         "pmm"         "pmm"         "pmm"
#>     RainToday  RainTomorrow          MMDD
#>         "pmm"            ""            ""
```

The code output above shows that

- the features without missing data will not be imputed.
- the imputation targets will all be treated with *Predictive mean matching* algorithm ("pmm").

This is exactly what we need. Now let's review the predictors (*Code output is not included into report to save space*).

The matrix of predictors has the predictors in the columns and the features to be imputed in the rows. If the cell value equals **1** the predictor will be employed in calculations of the respective imputation target. Surprisingly *MMDD* is not used widely to predict the missing data, nether do the *Location*.

Now we are going to start the imputation process. **Note: it might take about 4 - 5 minutes even for a sample**. We have disabled the output of the function as we do not want to pollute the report with irrelevant messages

```
imputed = mice(weatherSample, pred = predictors, meth = methods, seed = 38019,
               nnet.MaxNWts = 2000, printFlag = F)
```

Now it is time to analyze the imputed values. In general, a good imputed value is a value that could have been observed had it not been missing. The MAR assumption can never be tested from the observed data. To check whether the imputations created by **MICE** algorithm are plausible we employ density charts and compare the distribution of the imputed values vs real observations. Let's do this (*again the plots take time, patience...*).

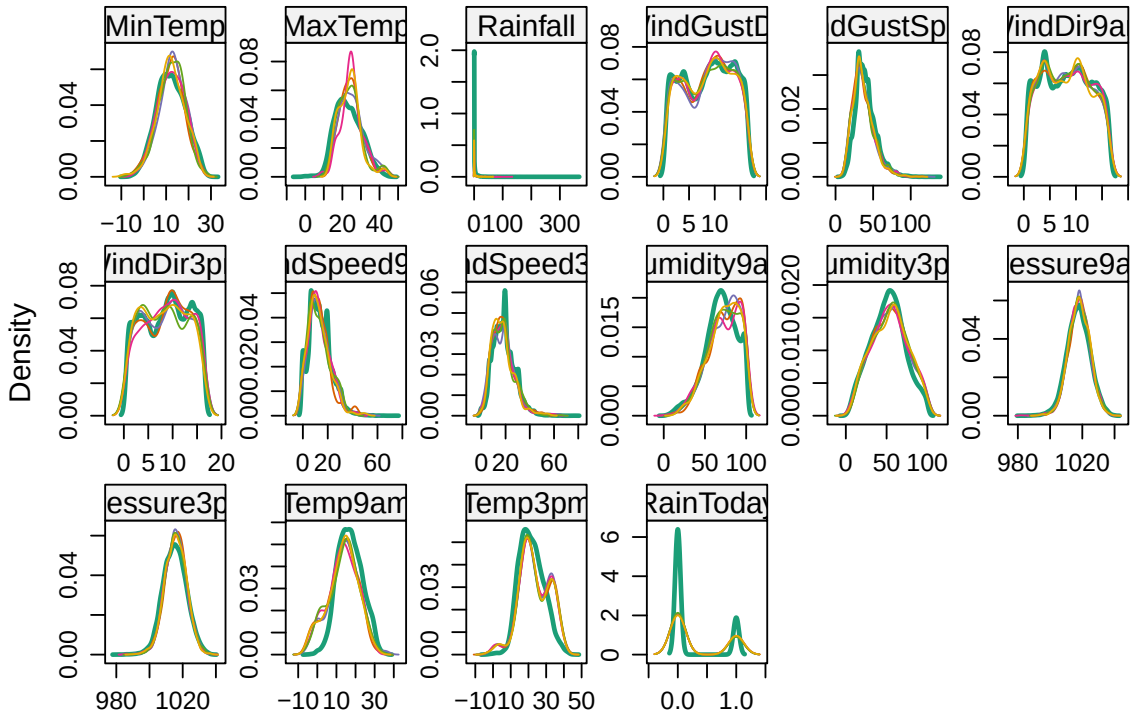

Figure 9: Imputed Values Distribution vs Real Observations

Figure 9 illustrates imputed value distribution for each imputed feature vs observed data. The fat green line renders the real data distribution and the thin lines of other colors show the distribution of the imputed data after each imputation cycle (*there are five of them by default*). The last imputation run is rendered in yellow; it should be shadowing the contour of the green one as close as possible. And it does which give us an indication that the result of the imputation is plausible. So looking at the charts we can conclude that the imputation has been successful! Let's apply imputed values to our sample set and verify if there are any *NAs* left.

```
weatherSample = complete(imputed)
print(colSums(is.na(weatherSample)))
```

```
#>      Location       MinTemp       MaxTemp      Rainfall   Evaporation
#>             0             0             0             0             0
#>      Sunshine   WindGustDir WindGustSpeed    WindDir9am    WindDir3pm
#>             0             0             0             0             0
#>  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm   Pressure9am
#>             0             0             0             0             0
```

```
#>   Pressure3pm      Cloud9am      Cloud3pm       Temp9am       Temp3pm
#>             0             0             0             0             0
#>     RainToday  RainTomorrow          MMDD
#>             0             0             0
```

Outstanding! There are no missing values. Now we move on to the next part - model training.

Modeling and Evalutation

Finally we have reached the stage where we can start training and evaluating classification models. At this point we have clear understanding of our data. We have gotten rid of the features that did not present much value. We have filled the gaps in our data set employing sophisticated imputation technique.

Feature Selection

The weather observation data set originally had 24 features. We have removed *RISK_MM* and *Date* as explained earlier and added *MMDD*. Now the data set has 22 features and one label - *RainTomorrow*. Let's see if we can reduce the number of predictors without significant information loss. This would make our models faster and more interpretable for users. We shall keep in mind that at the data exploration phase we discovered that many features were correlated (Figure 8). Hopefully this knowledge will help us to identify and remove redundant features.

Generally speaking feature evaluation methods can be separated into two groups: those that use the model information and those that do not. Clearly at this stage of our research the models are not ready. Thus we will be exploring the methods that do not require model.

This group of the method could be spit further as follows:

- wrapper methods that evaluate multiple models adding and/or removing predictors. These are some examples:
  - recursive feature elimination
  - genetic algorithms
  - simulated annealing
- filter methods which evaluate the relevance of the predictors outside of the predictive models.

The evaluation of various feature selection methods is not in the scope of this paper. Thus we opt for a recursive feature elimination method (Ref: Caret Max Kuhn) using accuracy as a target metric.

Before we proceed any further let's ensure that all categorical values get converted back to the factors. This is useful for dimentiality reduction algorithms and model training.

```
weatherSample = mutate(weatherSample, Location = as.factor(unclass(Location)),
          WindGustDir = as.factor(unclass(WindGustDir)),
          WindDir9am = as.factor(unclass(WindDir9am)), WindDir3pm = as.factor(unclass(WindDir3pm)),
          RainToday = as.factor(unclass(RainToday)), RainTomorrow = as.factor(unclass(RainTomorrow)))
```

It is time to run feature selection algorithm.

```
predictors = subset(weatherSample,select = -RainTomorrow)
label = weatherSample[,22]

# run the RFE algorithm
rfePrediction = rfe(predictors, label, sizes=c(1:22),
                    rfeControl = rfeControl(functions=rfFuncs, method="cv", number=3))
print(rfePrediction)
```

```
#>
#> Recursive feature selection
#>
#> Outer resampling method: Cross-Validated (3 fold)
#>
#> Resampling performance over subset size:
#>
#>  Variables Accuracy  Kappa AccuracySD  KappaSD Selected
```

```
#>          1   0.8239 0.3962  0.0019800 0.010263
#>          2   0.8250 0.4157  0.0007035 0.008486
#>          3   0.8332 0.4653  0.0086453 0.020116
#>          4   0.8413 0.5223  0.0048532 0.016212
#>          5   0.8489 0.5463  0.0026914 0.009575
#>          6   0.8497 0.5486  0.0019925 0.008716
#>          7   0.8516 0.5550  0.0034123 0.011395
#>          8   0.8522 0.5588  0.0026354 0.010375
#>          9   0.8559 0.5721  0.0028881 0.010820
#>         10   0.8552 0.5715  0.0017251 0.007309
#>         11   0.8547 0.5719  0.0026369 0.008775
#>         12   0.8562 0.5743  0.0033018 0.011060
#>         13   0.8557 0.5722  0.0045096 0.014191
#>         14   0.8559 0.5731  0.0029471 0.008654
#>         15   0.8558 0.5738  0.0026397 0.011328
#>         16   0.8553 0.5711  0.0021236 0.007330
#>         17   0.8554 0.5712  0.0035871 0.011682
#>         18   0.8568 0.5734  0.0043802 0.013652       *
#>         19   0.8562 0.5717  0.0022864 0.008480
#>         20   0.8566 0.5724  0.0034128 0.011871
#>         21   0.8556 0.5686  0.0030669 0.010513
#>         22   0.8558 0.5675  0.0035224 0.012731
#>
#> The top 5 variables (out of 18):
#>    Humidity3pm, Sunshine, WindGustSpeed, Location, Pressure3pm
```

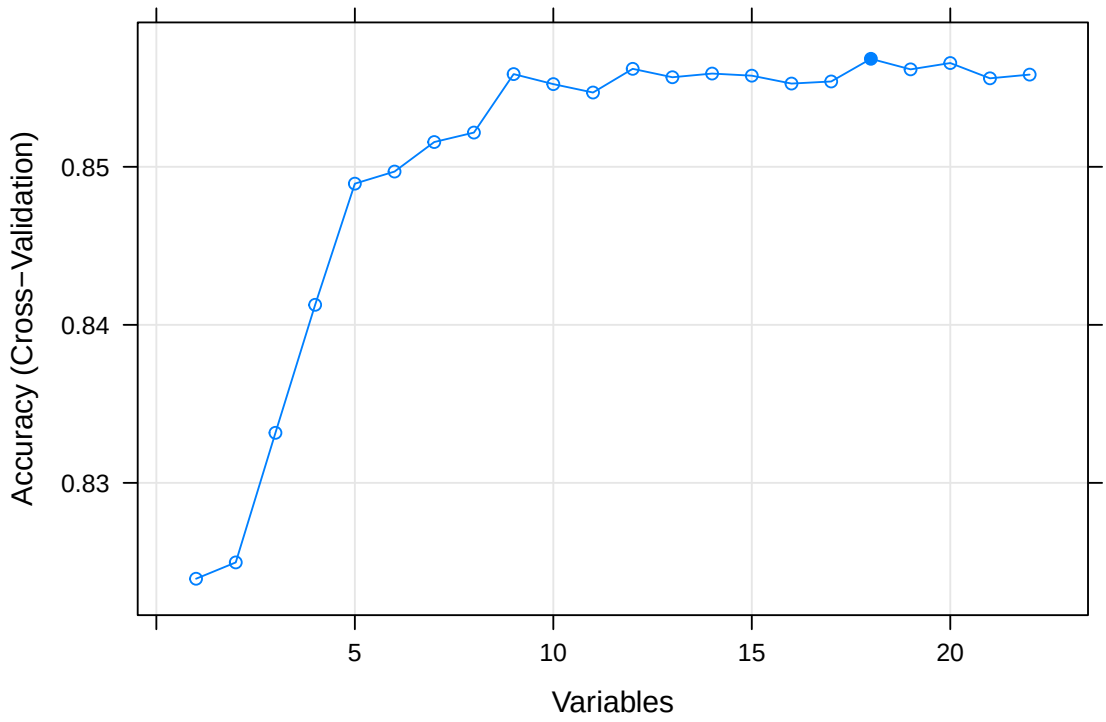

Figure 10: Number of Predictors vs Accuracy

Figure 10 illustrates that the accuracy practically flattens out when a number of predictors reaches 9. The accuracy improves a bit more when a number of features reaches 15 but the gain is negligible. Here is the list of features ordered by importance. We take first nine for model training.

```
#>  [1] "Humidity3pm"   "Sunshine"      "WindGustSpeed" "Location"
#>  [5] "Pressure3pm"   "Cloud3pm"      "Rainfall"      "Pressure9am"
#>  [9] "WindDir3pm"    "Humidity9am"   "WindGustDir"   "MinTemp"
```

```
#> [13] "Cloud9am"      "WindSpeed3pm"  "WindDir9am"    "Temp3pm"
#> [17] "RainToday"     "Temp9am"
```

It is counterintuitive that none of the temperature readings made it to the top, nether did our synthetic *MMDD* feature. Biases destroyed!

Data Upsampling

There is one more step to make before we get to the model training. As shown in Figure 2 our data set is unbalanced. This could cause model over-fitting. So let's split the data into the training and testing sets and up-sample the training set.

```
set.seed(1608)

# keep only the selected features
finalSample = weatherSample %>% dplyr::select(c(selectedPredictors,"RainTomorrow"));

splitIdx = createDataPartition(finalSample$RainTomorrow, p=0.7, list = F)  # 70% training data
trainData = finalSample[splitIdx, ]
testData = finalSample[-splitIdx, ]

set.seed(590045)
columns = colnames(trainData)
trainData = upSample(x = trainData[, columns[columns != "RainTomorrow"] ],
      y = trainData$RainTomorrow, list = F, yname = "RainTomorrow")

rm(splitIdx, columns, finalSample)
print(table(trainData$RainTomorrow))

#>
#>     0     1
#> 16228 16228
```

As we can see now the training set is balanced.

Thus we have prepared our training and test data sets. We have identified the most important features. We are ready to work on the prediction models.

Classification (Decision) Tree Model

Decision Tree algorithm is simple to understand, interpret and visualize. Effort required for data preparation is minimal. This is probably why the Decision Tree model tends to be the method of choice for predictive modeling of many.

```
#> Conditional Inference Tree
#>
#> 32456 samples
#>     9 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 25965, 25965, 25966, 25964, 25964
#> Resampling results across tuning parameters:
#>
#>   mincriterion  ROC        Sens       Spec
#>   0.01          0.8891238  0.7777305  0.8410156
#>   0.50          0.8849898  0.7764977  0.8288758
#>   0.99          0.8729268  0.7666361  0.8099585
#>
#> ROC was used to select the optimal model using the largest value.
#> The final value used for the model was mincriterion = 0.01.

confusionMatrix(data = pred.classTreeModel.raw, testDataCopy$RainTomorrow)
```

```
#> Confusion Matrix and Statistics
#>
#>          Reference
#> Prediction   no  yes
#>        no  5462  521
#>        yes 1492 1524
#>
#>               Accuracy : 0.7763
#>                 95% CI : (0.7676, 0.7849)
#>    No Information Rate : 0.7728
#>    P-Value [Acc > NIR] : 0.2143
#>
#>                  Kappa : 0.4545
#>  Mcnemar's Test P-Value : <2e-16
#>
#>             Sensitivity : 0.7854
#>             Specificity : 0.7452
#>          Pos Pred Value : 0.9129
#>          Neg Pred Value : 0.5053
#>              Prevalence : 0.7728
#>          Detection Rate : 0.6070
#>    Detection Prevalence : 0.6649
#>       Balanced Accuracy : 0.7653
#>
#>        'Positive' Class : no
#>
```

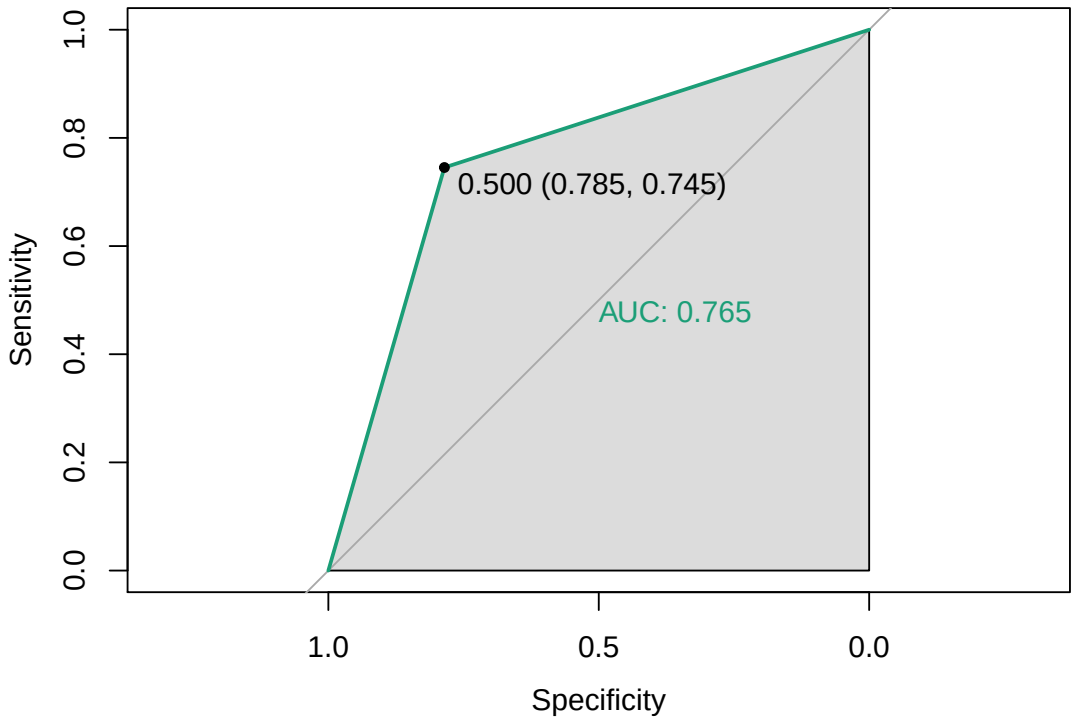

Figure 11: Classification Tree Model AUC and ROC Curve

Naive Bayes Model

Naïve Bayes classification is a kind of simple probabilistic classification methods based on Bayes' theorem with the assumption of independence between features.

It is simple (both intuitively and computationally), fast, performs well with small amounts of training data, and scales well to large data sets. The greatest weakness of the naïve Bayes classifier

is that it relies on an often-faulty assumption of equally important and independent features which results in biased posterior probabilities. Although this assumption is rarely met, in practice, this algorithm works surprisingly well and accurate; however, on average it rarely can compete with the accuracy of advanced tree-based methods (random forests & gradient boosting machines) but is definitely worth having in our toolkit.

```
#> Naive Bayes
#>
#> 32456 samples
#>     9 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 25965, 25965, 25966, 25964, 25964
#> Resampling results across tuning parameters:
#>
#>   usekernel  ROC        Sens       Spec
#>   FALSE      0.7402740  0.5754249  0.7620163
#>    TRUE      0.8591623  0.7780383  0.7736615
#>
#> Tuning parameter 'fL' was held constant at a value of 0
#> Tuning
#>  parameter 'adjust' was held constant at a value of 1
#> ROC was used to select the optimal model using the largest value.
#> The final values used for the model were fL = 0, usekernel = TRUE
#>  and adjust = 1.
```

```
confusionMatrix(data = pred.naiveBayesModel.raw, testDataCopy$RainTomorrow)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   no  yes
#>        no  5510  469
#>        yes 1444 1576
#>
#>                Accuracy : 0.7874
#>                  95% CI : (0.7788, 0.7958)
#>     No Information Rate : 0.7728
#>     P-Value [Acc > NIR] : 0.0004319
#>
#>                   Kappa : 0.4819
#>  Mcnemar's Test P-Value : < 2.2e-16
#>
#>             Sensitivity : 0.7923
#>             Specificity : 0.7707
#>          Pos Pred Value : 0.9216
#>          Neg Pred Value : 0.5219
#>              Prevalence : 0.7728
#>          Detection Rate : 0.6123
#>    Detection Prevalence : 0.6644
#>       Balanced Accuracy : 0.7815
#>
#>        'Positive' Class : no
#>
```

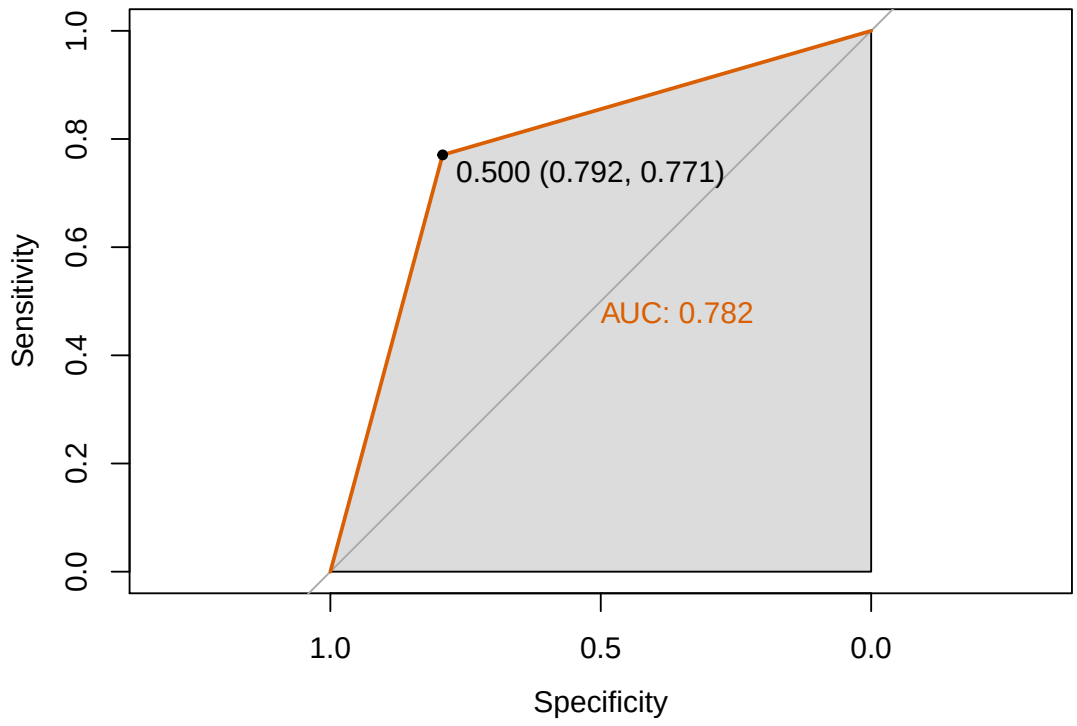

Figure 12: Naive Bayes Model AUC and ROC Curve

Random Forest Model

Random Forest is also considered as a very handy and easy to use algorithm, because it's default hyperparameters often produce a good prediction result. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained.

```
#>    user  system elapsed
#> 1094.91    3.42 1098.67

#> Random Forest
#>
#> 32456 samples
#>     9 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (3 fold)
#> Summary of sample sizes: 21638, 21637, 21637
#> Resampling results across tuning parameters:
#>
#>   mtry  ROC        Sens       Spec
#>    2    0.8848548  0.7910405  0.8086028
#>   36    0.9810928  0.8829804  0.9655531
#>   70    0.9795501  0.8762022  0.9654914
#>
#> ROC was used to select the optimal model using the largest value.
#> The final value used for the model was mtry = 36.

confusionMatrix(data = pred.randomForestModel.raw, testDataCopy$RainTomorrow)
```

```
#> Confusion Matrix and Statistics
#>
#>          Reference
#> Prediction   no  yes
#>        no  6311  725
#>        yes  643 1320
#>
#>               Accuracy : 0.848
#>                 95% CI : (0.8404, 0.8553)
#>    No Information Rate : 0.7728
#>    P-Value [Acc > NIR] : < 2e-16
#>
#>                  Kappa : 0.561
#>  Mcnemar's Test P-Value : 0.02853
#>
#>             Sensitivity : 0.9075
#>             Specificity : 0.6455
#>          Pos Pred Value : 0.8970
#>          Neg Pred Value : 0.6724
#>              Prevalence : 0.7728
#>          Detection Rate : 0.7013
#>    Detection Prevalence : 0.7819
#>       Balanced Accuracy : 0.7765
#>
#>        'Positive' Class : no
#>
```

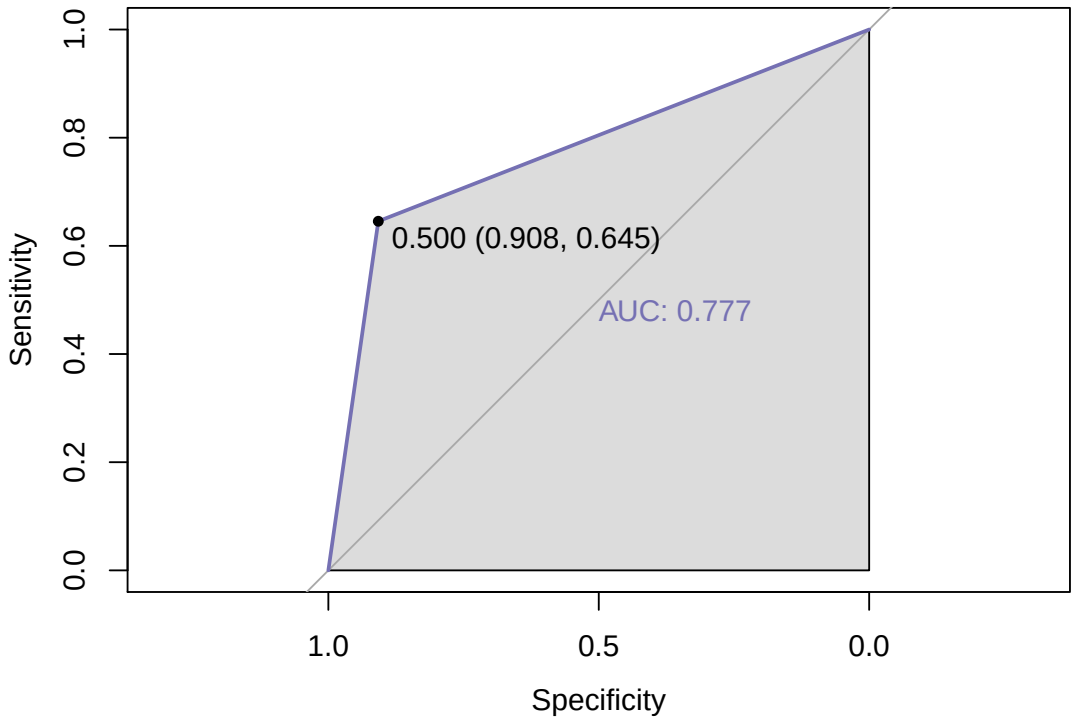

Figure 13: Random Forest Model AUC and ROC Curve

Logistic Regression Model

Logistic regression is an efficient, interpretable and accurate method, which fits quickly with minimal tuning. Logistic regression prediction accuracy will benefit if the data is close to Gaussian distribution. Thus we apply addition transformation to the training data set. We will also be employing 5-fold cross-validation resampling procedure to improve the model. In addition to the above we are going to

convert *Location* categorical value to numeric data type. We could have used dummy encoding but having 49 locations such approach does not seem beneficial.

```
confusionMatrix(data = pred.logRegModel.raw, testDataCopy$RainTomorrow)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction    0    1
#>          0 5586  446
#>          1 1368 1599
#>
#>                Accuracy : 0.7984
#>                  95% CI : (0.79, 0.8067)
#>     No Information Rate : 0.7728
#>     P-Value [Acc > NIR] : 2.083e-09
#>
#>                   Kappa : 0.5048
#>  Mcnemar's Test P-Value : < 2.2e-16
#>
#>             Sensitivity : 0.8033
#>             Specificity : 0.7819
#>          Pos Pred Value : 0.9261
#>          Neg Pred Value : 0.5389
#>              Prevalence : 0.7728
#>          Detection Rate : 0.6207
#>    Detection Prevalence : 0.6703
#>       Balanced Accuracy : 0.7926
#>
#>        'Positive' Class : 0
#>
```

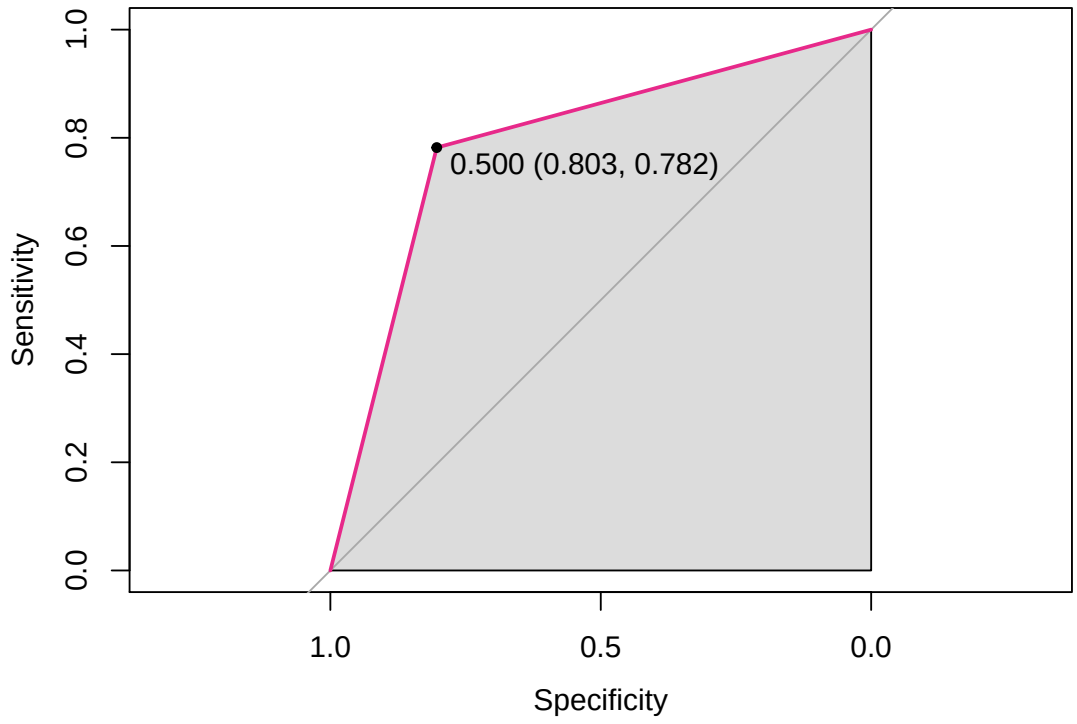

Figure 14: Logistic Regression Model AUC and ROC Curve

Confusion matrix and Figure 14 demonstrate the logistic model performance on the balanced data set. Using the proportion of positive data points that are correctly considered as positive (true

positives) and the proportion of negative data points that are mistakenly considered as positive (false negative), we generated a graphic that shows the trade off between the rate at which the model correctly predicts the rain tomorrow with the rate of incorrectly predicting the rain. The value around 0.80 indicates that the model does a good job in discriminating between the two categories.

Model Comparison

Now it is time to compare the models side by side and pick a winner.

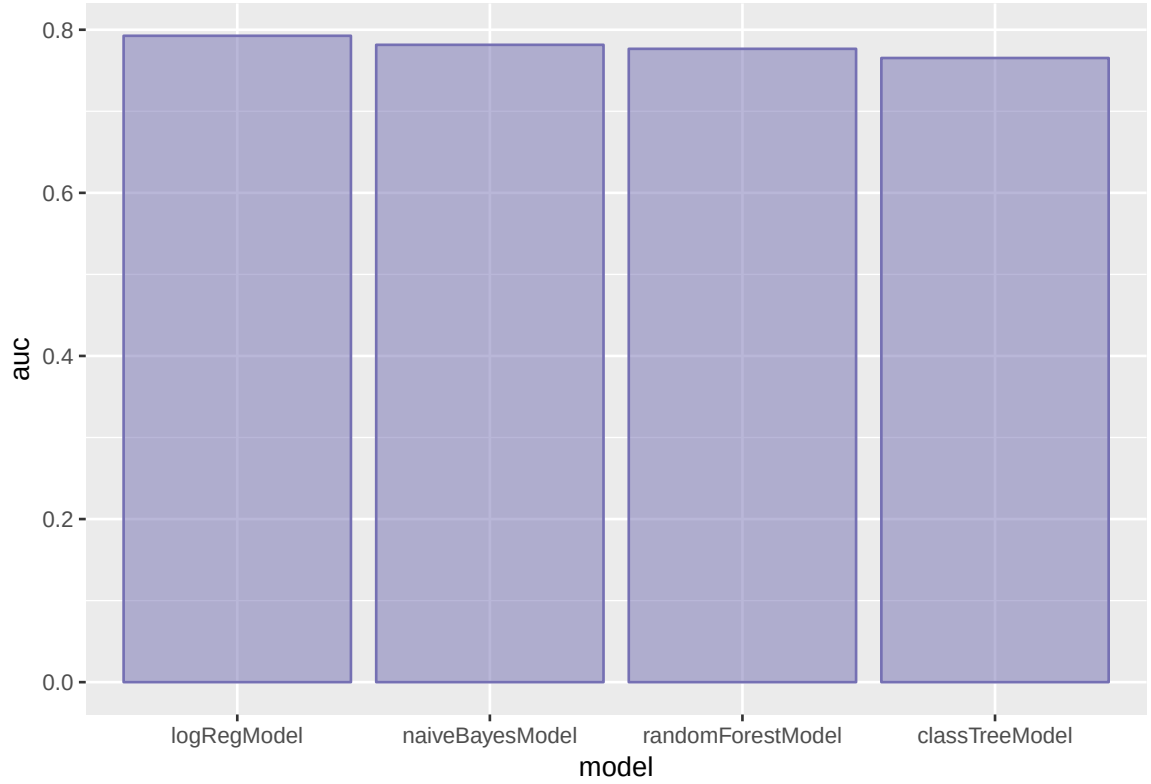

Figure 15: Model AUC Comparison

```
#>                model       auc
#> 1       logRegModel 0.7925929
#> 3   naiveBayesModel 0.7815049
#> 4 randomForestModel 0.7765779
#> 2    classTreeModel 0.7653397
```

AUC - ROC perfomance AUC stands for Area under the ROC Curve and ROC for Receiver operating characteristic curve. This is one of the most important KPIs of the classification algorithms. These two metrics measure how well the models distinguishing between the classes. The higher AUC the better model predicts positive and negative outcome.

Figures 11, 12, 13, 14 and accompanying data show that on the test data set the Random Forest model has the highest overall accuracy (85%) but performs poorly predicting rainy days (63%), thus the balanced accuracy is lower (about 76%).

Naive Bayes has lesser overall accuracy in comparison with the Random Forest one but is more balanced, demonstrating consistent power to predict rainy and sunny days with almost equal accuracy. It has balanced accuracy of 78%.

Logistic regression model scores the best having the highest AUC and all other metrics. It's balanced accuracy is over 80%.

The descision tree performance is close to the other models with the balanced accuracy of 76%.

Model interpretibility Logistic Regression, Decision Tree and Naive Bayes are all highly interpreatable models. It is easy to explain to the business what impact each input parameter has. The decision

tree could be visualized (provided if it is not too large).

Random Forest on the other hand is a black-box model, complex algorithm which is difficult to explain in simple terms.

Data Preparation Decision Tree, Random Forest and Naive Bayes can deal with missing data, outliers, numeric and alphanumeric values. Simply speaking they are not very demanding for data quality. It would be interesting to see how they perform on the original data set without data cleaning. But this is subject of another research...

Logistic regression does require conversion of alphanumeric values to numeric, struggles dealing with the outliers and performs best when fitted with the data that have normal distribution.

Verdict Despite sensitivity to data quality Logistic Regression outperforms other models in all other major categories. This is our choice!

Model Deployment

Without a doubt it would be a stretch to compare our model to the production numerical weather prediction models. But we do believe it might have a real live application as an educational tool. The model can demonstrate how various weather elements affect the probability of the rain.

It is simple to understand and deploy. The model does not require frequent updates because the weather patterns tend to be stable for a given geographical area (though this statement might be compromised in the context of the global warming). The model would benefit greatly if more complete data was available. Recall that we had to impute a lot of missing values.

Conclusion

Through exploring weather observations collected by 49 stations in Australia from 2007 to 2017 we selected and tuned a model to predict a rainy day tomorrow employing current day observations and historical data.

We commenced our research analyzing and understanding available data and geography of the weather stations. Then we identified the missing data, its distribution and feasibility of imputing it. We applied sophisticated data imputation algorithm to attack the problem. We continued our research selecting the most impactful data attributes to use as an input for our future model. Again we apply the feature identification algorithm to do the job.

When the data preparation phase was finished we picked and analysed four different classification models: Decision Tree, Naive Bayes, Random Forest and Logistic Regression. We conducted comparative analysis of the models, reviewed their strength and weaknesses. We fitted each model using K-fold cross-validation technique. Subsequently we evaluated performance of each model applying them to the test data set and comparing AUC - ROC and balanced accuracy metrics.

Finally we moved to identifying a winning model. In order to so we reviewed each model from different angles namely:

- performance
- interpretability
- data quality sensitivity and data preparation effort

The winning model scored the highest in the majority of the categories. It was Logistic Regression, which we employed to build a Shiny App Web application.

We consider the project to be a success. Being easily understood, with a balanced accuracy of 80% we conclude than our model could be applied for a short-term rain forecast. The last but not least we are confident that the model can predict the rain better than aboriginal "rainmakers".

Bibliography

J. P. Jiawei Han, Micheline Kamber. *Data Mining. Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems, 225 Wyman Street, Waltham, MA 02451, USA, 2012. ISBN 978-0-12-381479-1. [p1]

A. A. Matthias Templ, Alexander Kowarik. Vim (isualization and imputation of missing values) project. URL https://cran.r-project.org/web/packages/VIM/index.html. [p6]

S. W. Max Kuhn, Jed Wing. Caret (classification and regression training) project. URL https://cran.r-project.org/web/packages/caret/index.html. [p14]

A. M. Society. Weather analysis and forecasting. URL https://www.ametsoc.org/ams/index.cfm/about-ams/ams-statements/statements-of-the-ams-in-force/weather-analysis-and-forecasting. [p1]

A. R. Stef van Buuren, Karin Groothuis-Oudshoorn. Mice project (multivariate imputation by chained equations). URL https://cran.r-project.org/web/packages/mice/index.html. [p11]

Note from the Authors

This file was generated using *The R Journal* style article template, additional information on how to prepare articles for submission is here - Instructions for Authors. The article itself is an executable R Markdown file that could be downloaded from Github with all the necessary artifacts.

Sumaira Afzal
York University School of Continuing Studies

Viraja Ketkar
York University School of Continuing Studies

Murlidhar Loka
York University School of Continuing Studies

Vadim Spirkov
York University School of Continuing Studies