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 the latest scientific methods, 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.

Objective

The objective of this research is to find a numerical weather prediction model that would provide accurate forecast of possibility of the rain next day having today weather pattern observations. In addition to accuracy the model should be easily interpretable and flexible enough to accept limited number of input features without diminishing its prediction power.

Data Analisys

The data set we are going to use for our research contains daily weather observations from numerous Australian weather stations 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
Sunshine	Number of hours of bright sunshine in the day

Column Name	Column Description
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
RainTomorrowThe	**Target variable. 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* format

```
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 let's load coordinates of the weather stations and have a bird-eye view of the weather station locations

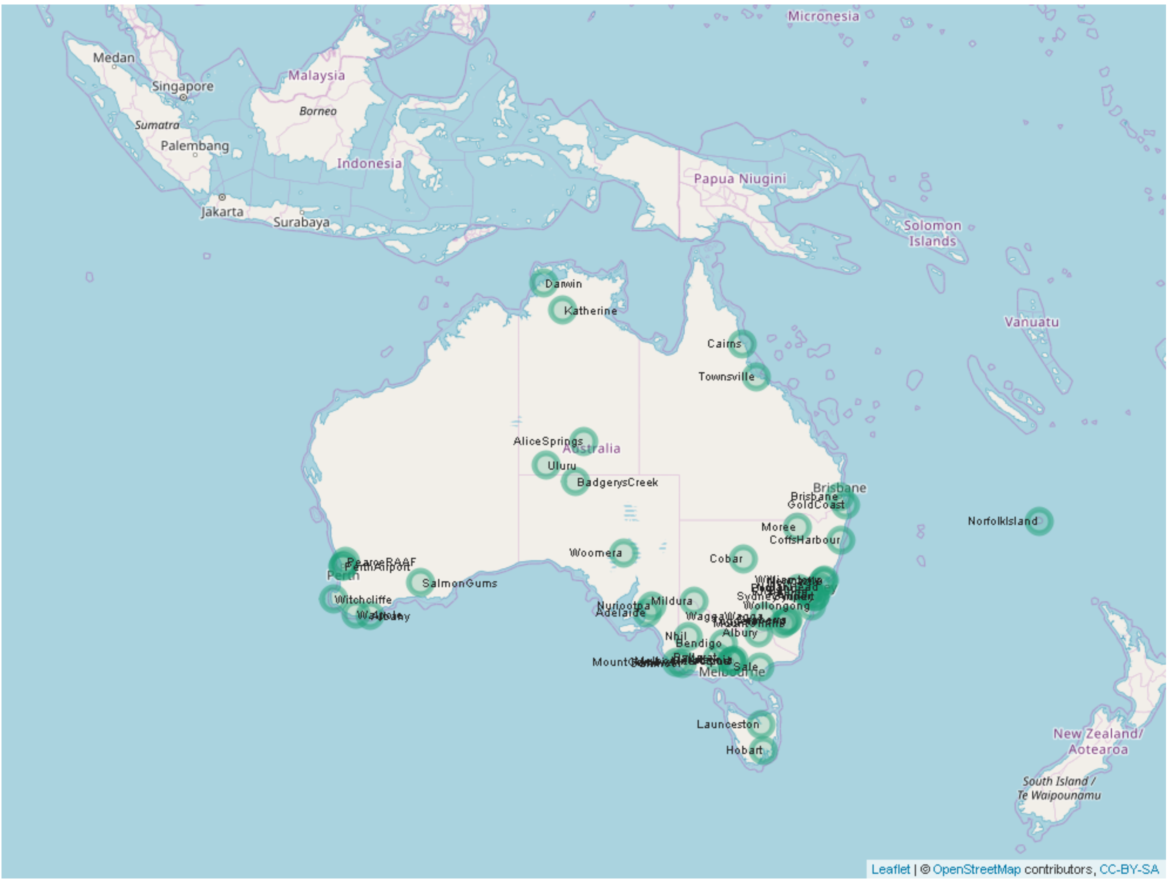

Figure 1: Australian Weather Stations

Let's review data summary

```
summary(weatherData)
```

```
#>       Date                  Location         MinTemp          MaxTemp
#>  Min.   :2007-11-01   Canberra:  3418   Min.   :-8.50   Min.   :-4.80
#>  1st Qu.:2011-01-06   Sydney  :  3337   1st Qu.: 7.60   1st Qu.:17.90
#>  Median :2013-05-27   Perth   :  3193   Median :12.00   Median :22.60
#>  Mean   :2013-04-01   Darwin  :  3192   Mean   :12.19   Mean   :23.23
#>  3rd Qu.:2015-06-12   Hobart  :  3188   3rd Qu.:16.80   3rd Qu.:28.20
#>  Max.   :2017-06-25   Brisbane:  3161   Max.   :33.90   Max.   :48.10
#>                       (Other) :122704   NA's   :637     NA's   :322
#>     Rainfall          Evaporation          Sunshine       WindGustDir
#>  Min.   :  0.00   Min.   :  0.00   Min.   : 0.00   W      : 9780
#>  1st Qu.:  0.00   1st Qu.:  2.60   1st Qu.: 4.90   SE     : 9309
#>  Median :  0.00   Median :  4.80   Median : 8.50   E      : 9071
#>  Mean   :  2.35   Mean   :  5.47   Mean   : 7.62   N      : 9033
#>  3rd Qu.:  0.80   3rd Qu.:  7.40   3rd Qu.:10.60   SSE    : 8993
#>  Max.   :371.00   Max.   :145.00   Max.   :14.50   (Other):86677
#>  NA's   :1406     NA's   :60843    NA's   :67816   NA's   : 9330
#>  WindGustSpeed      WindDir9am       WindDir3pm      WindSpeed9am
#>  Min.   :  6.00   N      :11393   SE     :10663   Min.   :  0
#>  1st Qu.: 31.00   SE     : 9162   W      : 9911   1st Qu.:  7
#>  Median : 39.00   E      : 9024   S      : 9598   Median : 13
#>  Mean   : 39.98   SSE    : 8966   WSW    : 9329   Mean   : 14
#>  3rd Qu.: 48.00   NW     : 8552   SW     : 9182   3rd Qu.: 19
#>  Max.   :135.00   (Other):85083   (Other):89732   Max.   :130
#>  NA's   :9270     NA's   :10013   NA's   : 3778   NA's   :1348
#>   WindSpeed3pm     Humidity9am       Humidity3pm       Pressure9am
#>  Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 980.5
#>  1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 37.00   1st Qu.:1012.9
#>  Median :19.00   Median : 70.00   Median : 52.00   Median :1017.6
```

```
#>  Mean   :18.64   Mean   : 68.84   Mean   : 51.48   Mean   :1017.7
#>  3rd Qu.:24.00   3rd Qu.: 83.00   3rd Qu.: 66.00   3rd Qu.:1022.4
#>  Max.   :87.00   Max.   :100.00   Max.   :100.00   Max.   :1041.0
#>  NA's   :2630    NA's   :1774     NA's   :3610     NA's   :14014
#>   Pressure3pm        Cloud9am        Cloud3pm        Temp9am
#>  Min.   : 977.1   Min.   :0.00    Min.   :0.0     Min.   :-7.20
#>  1st Qu.:1010.4   1st Qu.:1.00    1st Qu.:2.0     1st Qu.:12.30
#>  Median :1015.2   Median :5.00    Median :5.0     Median :16.70
#>  Mean   :1015.3   Mean   :4.44    Mean   :4.5     Mean   :16.99
#>  3rd Qu.:1020.0   3rd Qu.:7.00    3rd Qu.:7.0     3rd Qu.:21.60
#>  Max.   :1039.6   Max.   :9.00    Max.   :9.0     Max.   :40.20
#>  NA's   :13981    NA's   :53657   NA's   :57094   NA's   :904
#>     Temp3pm      RainToday     RainTomorrow
#>  Min.   :-5.40   No  :109332   No :110316
#>  1st Qu.:16.60   Yes : 31455   Yes: 31877
#>  Median :21.10   NA's:  1406
#>  Mean   :21.69
#>  3rd Qu.:26.40
#>  Max.   :46.70
#>  NA's   :2726
```

Next set of plots renders distribution of a few selected features.

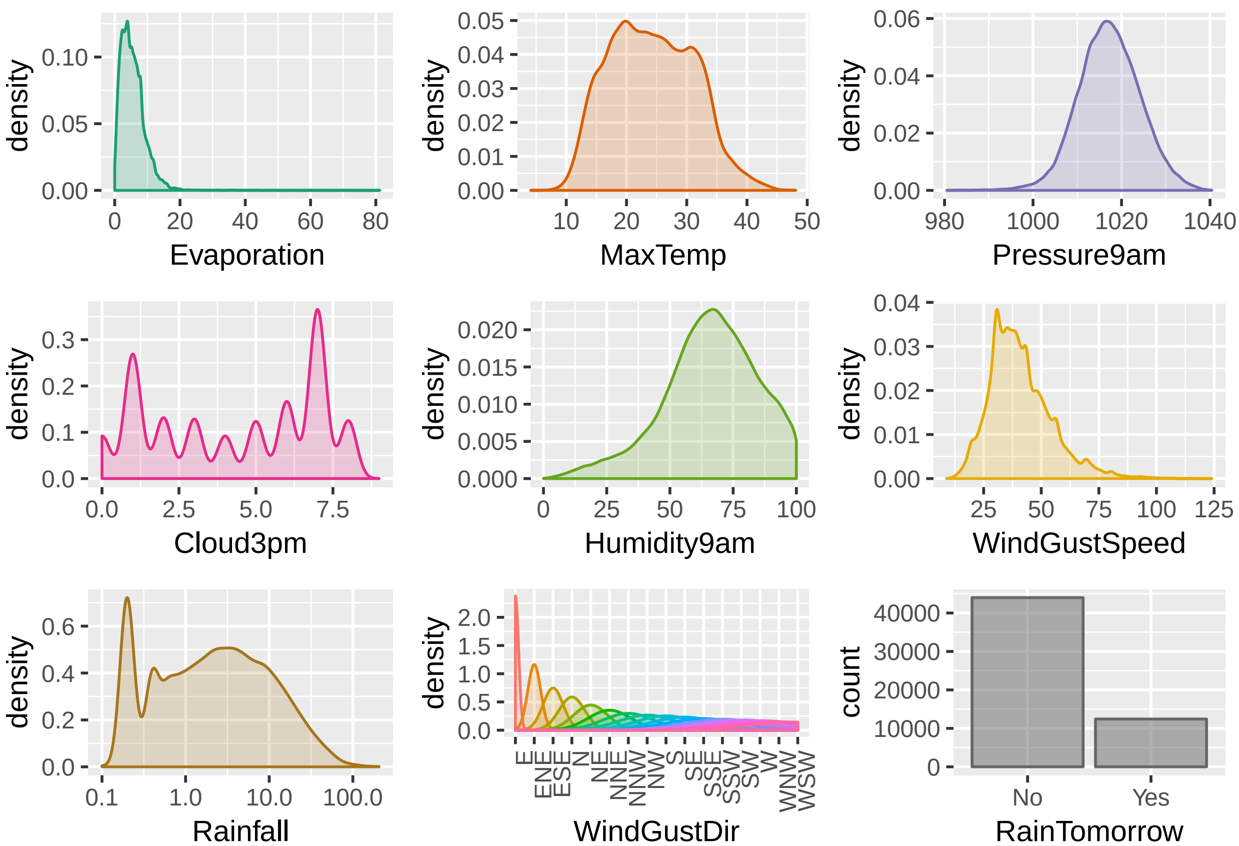

Figure 2: Observations Distribution

Missing Data

Further analysis of data shows that many features are missing. Some data losses are very significant. We are going to identify what data is missing and if it is feasible to recover the data.

```
print(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, 20K 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)
```

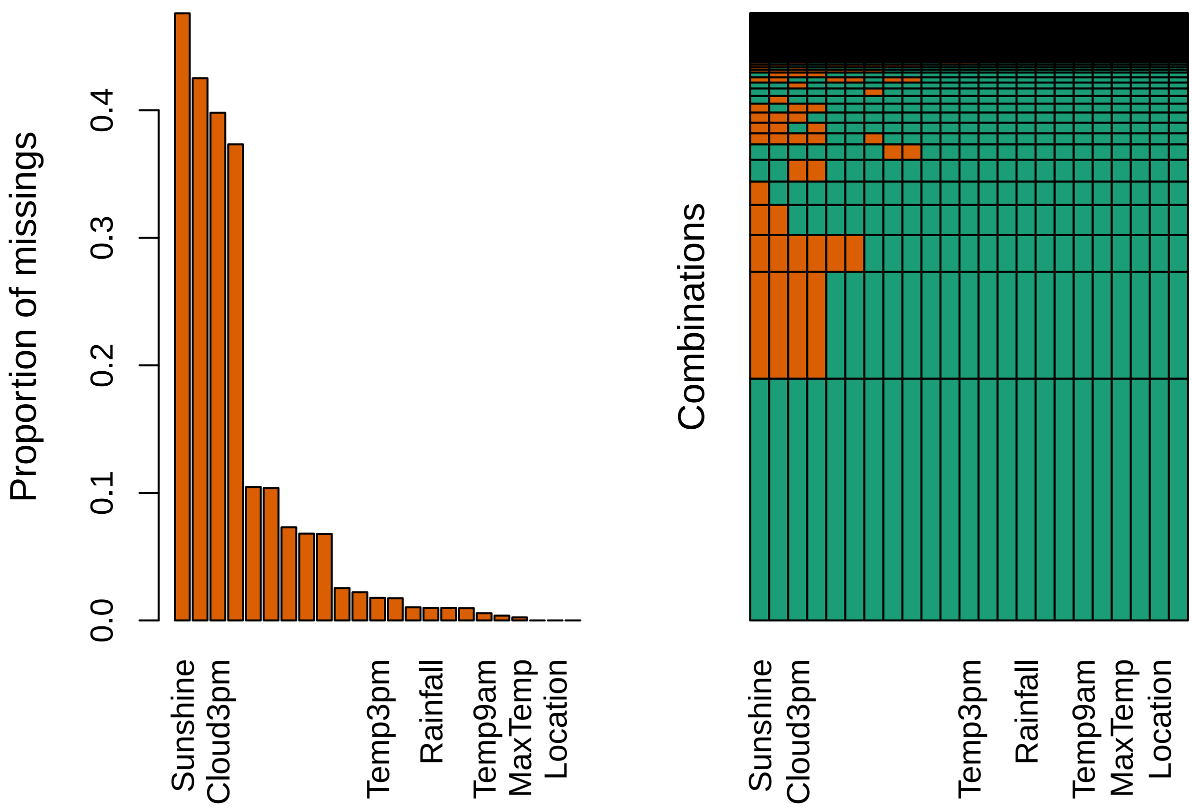

Figure 3: Missing Data Summary

```
#>
#>  Variables sorted by number of missings:
#>       Variable  Count
#>       Sunshine 0.4760
#>    Evaporation 0.4251
#>       Cloud3pm 0.3980
#>       Cloud9am 0.3733
#>    Pressure9am 0.1046
#>    Pressure3pm 0.1038
#>     WindDir9am 0.0730
#>    WindGustDir 0.0681
#>  WindGustSpeed 0.0679
#>     WindDir3pm 0.0254
#>    Humidity3pm 0.0221
#>        Temp3pm 0.0178
```

```
#>    WindSpeed3pm 0.0174
#>     Humidity9am 0.0103
#>        Rainfall 0.0099
#>       RainToday 0.0099
#>    WindSpeed9am 0.0097
#>         Temp9am 0.0057
#>         MinTemp 0.0038
#>         MaxTemp 0.0024
#>            Date 0.0000
#>        Location 0.0000
#>    RainTomorrow 0.0000
```

As demonstrated in Figure 3 *Sunshine*, *Evaporation* and *Clouds* columns safer the loss of data between **48%** and **38%**. This is significant! Since we are dealing with the weather patterns we should be observing cyclical data patterns. Let's review data distribution of features that damaged the most.

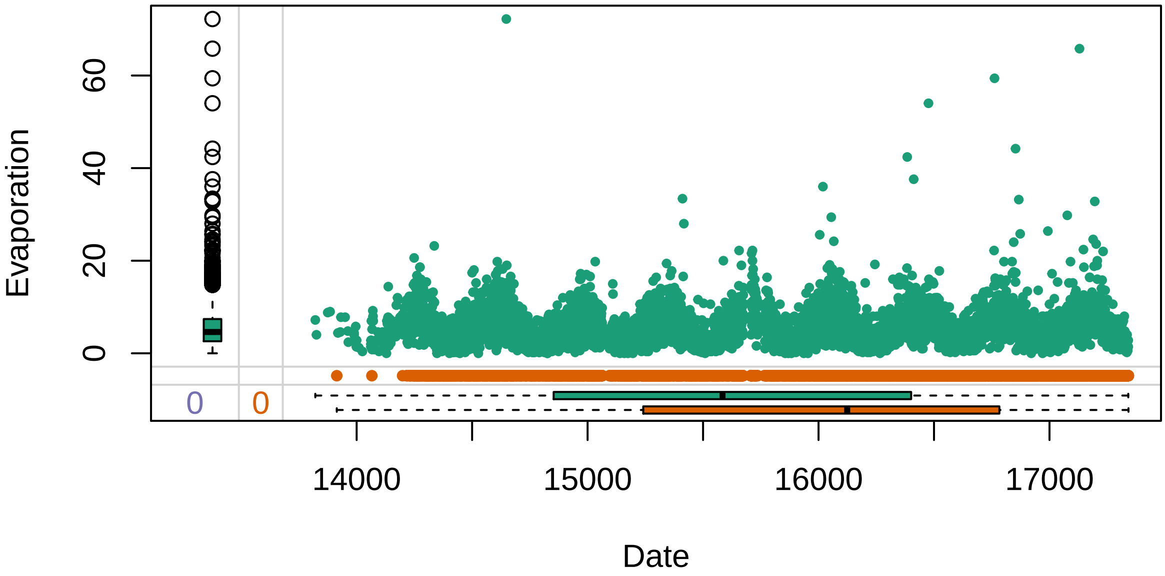

Figure 4: Date/Evaporation Margin Plot

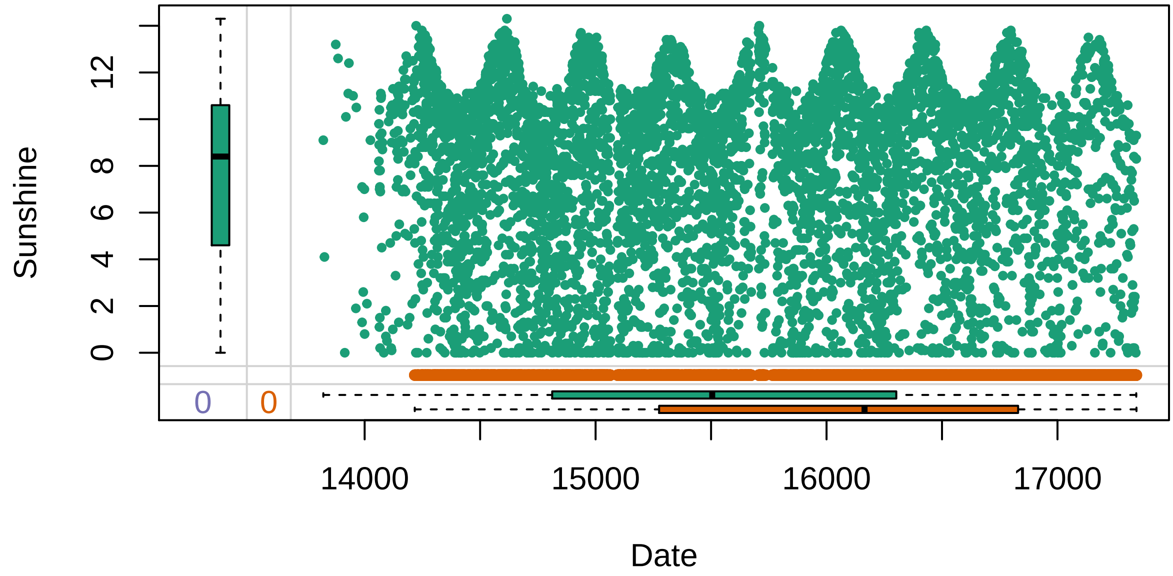

Figure 5: Date/ Sunshine Margin Plot

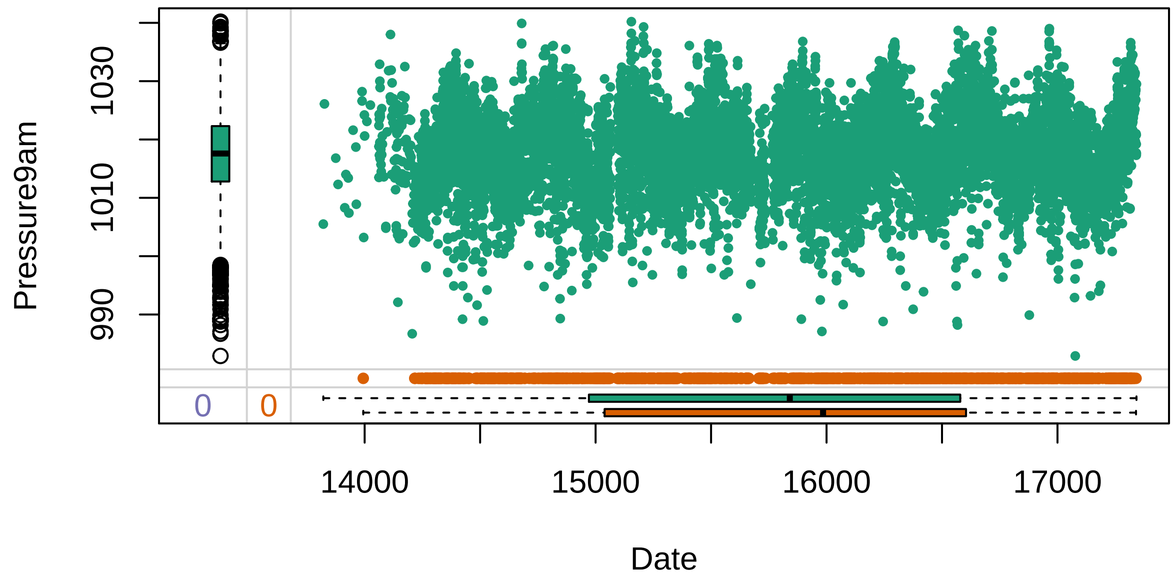

Figure 6: Date/ Pressure3pm Margin Plot

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 forces of nature follow seasonal patters they often exhibit wide range of seasonal anomalies, which the plots highlight. 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 along observed date range. Along the horizontal axis we may see box plots of the date and a given feature. *Presure9am* ((Figure 6)) distributed evenly across the observed date frame. *Evaporation* and *Sunshine* exhibit more data losses towards the end of the observed period

Let's examine one more dimension of the missing data, namely features vs feature vs location

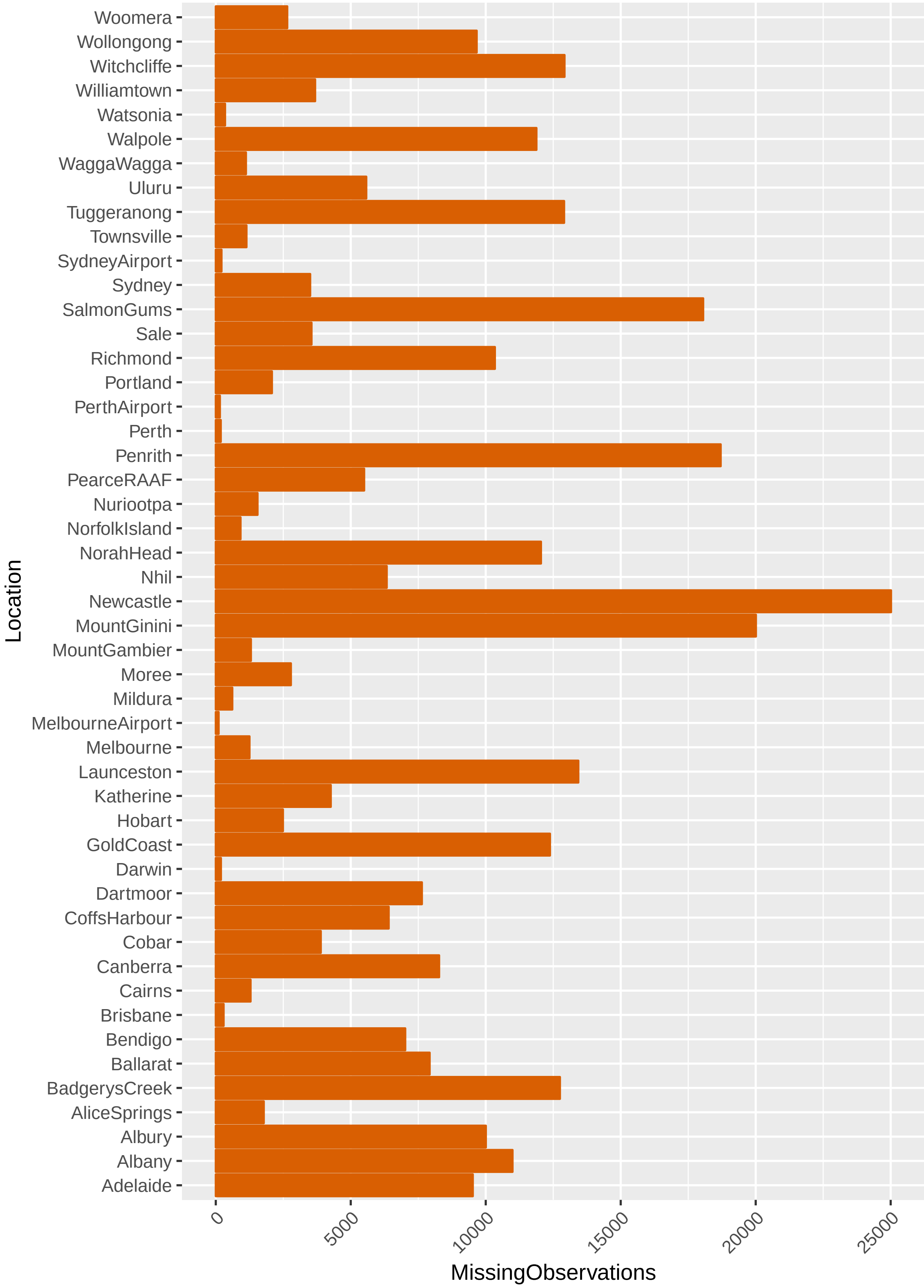

Figure 7: Missing Data By Location

Remarkably Figure 7 shows that **6460** observations are missing on average per location. Though if we take a second look at the weather station map 1 we would see that Mount Gini (the station that miss the most data), Bendigo and Ballarat are close to Melbrun, where the staff has kept observing data on regular basis. Newcastle to Sydney and so on...

Data correlation and other observations

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

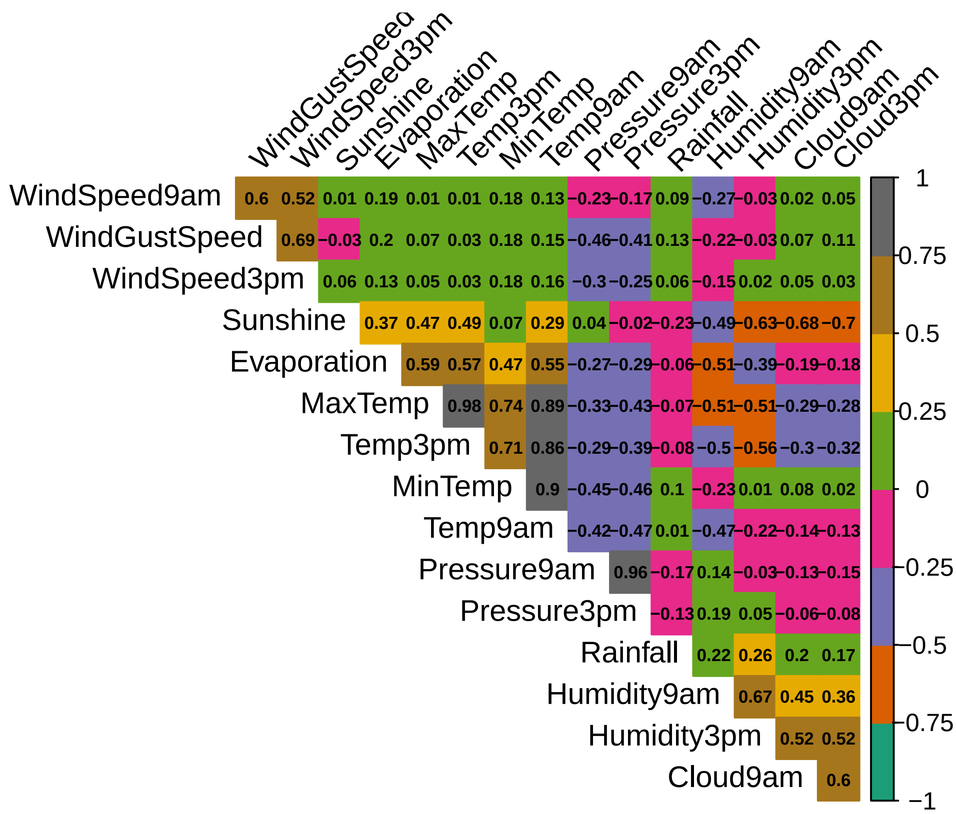

Figure 8: Data Correlation

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 suffers 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 many majority 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* exhibits 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 feature

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

Data Preparation

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

Datas Imputing

Before we start dealing with missing observations let's do some feature engineering, which will + improve 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 observation fluctuations. That would bee *month* and *day* combined, giving us year-round (365) days of observations

Secondly we convert categorical features to numbers. But before we do so we would like to ponder about *Location*. We have couple options here. Either we convert the locations to the numbers or we can replace them with the real geographical coordinates. After some deliberation we can conclude that the coordinates will not add too much knowledge in the context of the model training. But they will certainly break this categorical feature (coordinates have 4,6 decimal places, which effectively make them continuous). So we stick with categories.

This is our original set:

```
str(weatherData)
```

```
#> '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:

```
str(data)
```

```
#> '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 ...
```

To impute the missing data we employ **MICE** package. Our imputation strategy is to employ **Predictive mean matching** model which is a robust, fast imputation algorithm that works with numeric values (this is why we have converted all data to the numeric values) Lets do a dry run first to see what predictors and methods for each feature to cure *MICE* software chooses. As before we will be working with a 20K data sample. Imputation process on the whole set take about 3 hours and 20 minutes to complete! In addition we let *MICE* to choose predictors for us running **quickpred()** method

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

Let's review the methods chosen by the software making sure that they meet our requirements highlighted prior in the imputation strategy paragraph

```
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 1 the features without missing data will not be imputed 2 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 (*The command 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 has **1** the predictor will be employed in calculations for 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 smaple**. 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...*).

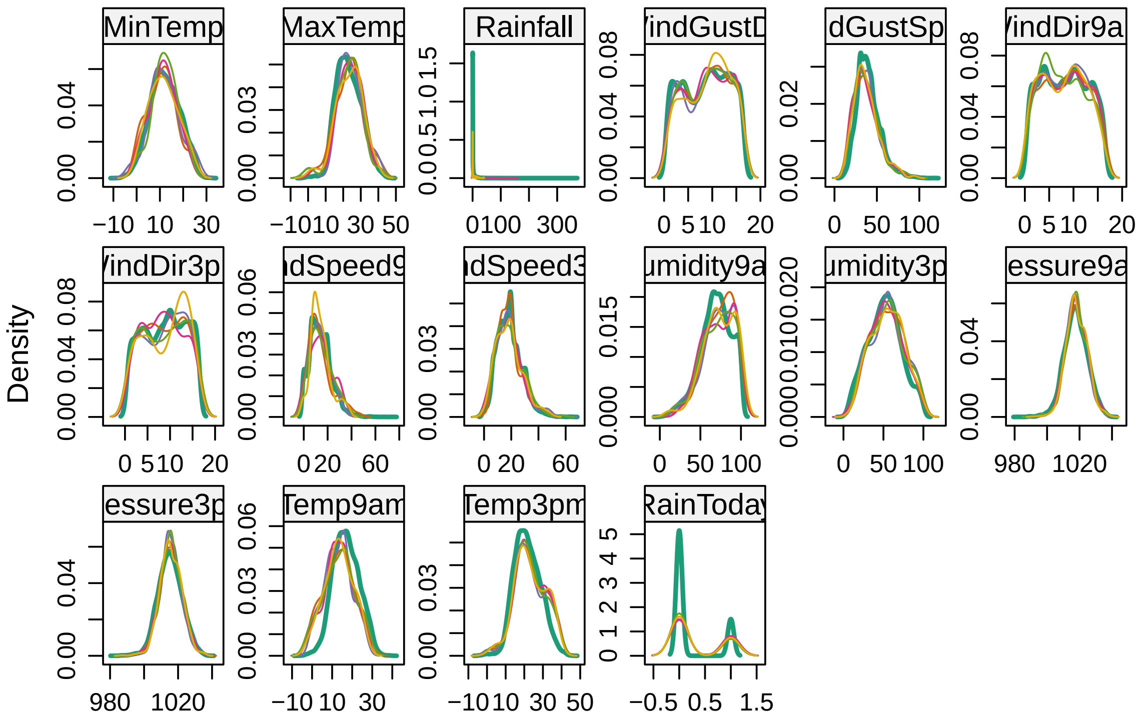

Figure 9: Imputed Values Distribution vs Real Observations

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 the other colors the distribution of imputed values after each imputation cycle (*there are five of them by default*). Where the last one is yellow. The yellow line should be shadowing the contour of the green one as close as possible, which give us an indication that the result of the imputation is plausible. 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. 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 have discovered that many features are correlated (Figure 8). hopefully this knowledge will help us 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 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 using accuracy as a target metric.

Before we precede any further let's ensure that all categorical values get converted to 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)))
```

Let's 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   0.8299 0.4029  0.0026465 0.011730
#>          2   0.8136 0.3815  0.0011240 0.008133
#>          3   0.8327 0.4434  0.0027761 0.026249
#>          4   0.8282 0.4605  0.0074554 0.035109
#>          5   0.8370 0.5018  0.0026736 0.007753
#>          6   0.8409 0.5175  0.0055642 0.014173
#>          7   0.8425 0.5222  0.0049109 0.012185
#>          8   0.8452 0.5277  0.0041731 0.011044
```

```
#>          9   0.8477 0.5305  0.0034504 0.010820
#>         10   0.8471 0.5346  0.0031549 0.011910
#>         11   0.8478 0.5389  0.0037956 0.010865
#>         12   0.8495 0.5429  0.0026821 0.008961
#>         13   0.8472 0.5347  0.0032822 0.011780
#>         14   0.8499 0.5436  0.0022456 0.011753
#>         15   0.8507 0.5477  0.0040748 0.008121
#>         16   0.8501 0.5440  0.0009483 0.007511
#>         17   0.8507 0.5470  0.0010324 0.005136
#>         18   0.8502 0.5443  0.0016291 0.005912
#>         19   0.8490 0.5422  0.0053511 0.017372
#>         20   0.8490 0.5396  0.0027612 0.011155
#>         21   0.8511 0.5444  0.0019786 0.007723
#>         22   0.8515 0.5458  0.0014777 0.006059        *
#>
#> The top 5 variables (out of 22):
#>    Humidity3pm, Sunshine, WindGustSpeed, Cloud3pm, Location
```

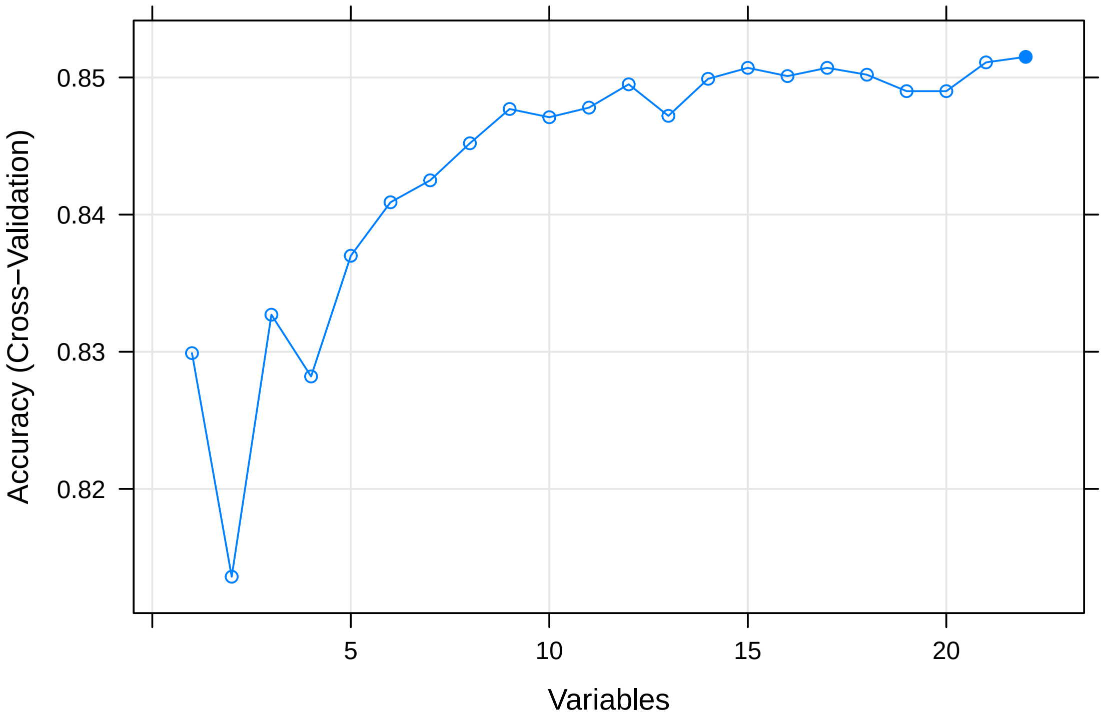

Figure 10: Number of Predictors vs Accuracy

Figure 10 shows that accuracy peaks a few times: with 9 predictors, 14 and tops at 22.The accuracy gain between 9 and 22 is negligible. Here is the list of features ordered by importance. We take first nine for model training.

```
print(predictors(rfePrediction))
```

```
#>  [1] "Humidity3pm"   "Sunshine"      "WindGustSpeed" "Cloud3pm"
#>  [5] "Location"      "Pressure3pm"   "Rainfall"      "Pressure9am"
#>  [9] "Humidity9am"   "WindDir3pm"    "Cloud9am"      "RainToday"
#> [13] "Temp3pm"       "Evaporation"   "WindSpeed3pm"  "MinTemp"
#> [17] "WindDir9am"    "WindGustDir"   "Temp9am"       "MaxTemp"
#> [21] "WindSpeed9am"  "MMDD"
```

```
l = length(predictors(rfePrediction))
selectedPredictors =  predictors(rfePrediction)[1:ifelse(l<9,l,9)]
```

```
# remove useless variables
rm(label,predictors,rfePrediction,l)
```

Data Upsampling

There is one more step 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

```
#>
#>    0    1
#> 5446 5446
```

As we can see 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 Tree Model

```
#> Conditional Inference Tree
#>
#> 10892 samples
#>     9 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 8714, 8713, 8714, 8714, 8713
#> Resampling results across tuning parameters:
#>
#>   mincriterion  ROC        Sens       Spec
#>   0.01          0.8814035  0.7734145  0.8207831
#>   0.50          0.8741582  0.7684555  0.8040773
#>   0.99          0.8579026  0.7636755  0.7769063
#>
#> 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  1715  165
#>        yes  618  501
#>
#>                Accuracy : 0.7389
#>                  95% CI : (0.7228, 0.7546)
#>     No Information Rate : 0.7779
#>     P-Value [Acc > NIR] : 1
#>
#>                   Kappa : 0.3921
#>  Mcnemar's Test P-Value : <2e-16
#>
#>             Sensitivity : 0.7351
#>             Specificity : 0.7523
#>          Pos Pred Value : 0.9122
#>          Neg Pred Value : 0.4477
#>              Prevalence : 0.7779
#>          Detection Rate : 0.5719
#>    Detection Prevalence : 0.6269
#>       Balanced Accuracy : 0.7437
#>
```

```
#>        'Positive' Class : no
#>
```

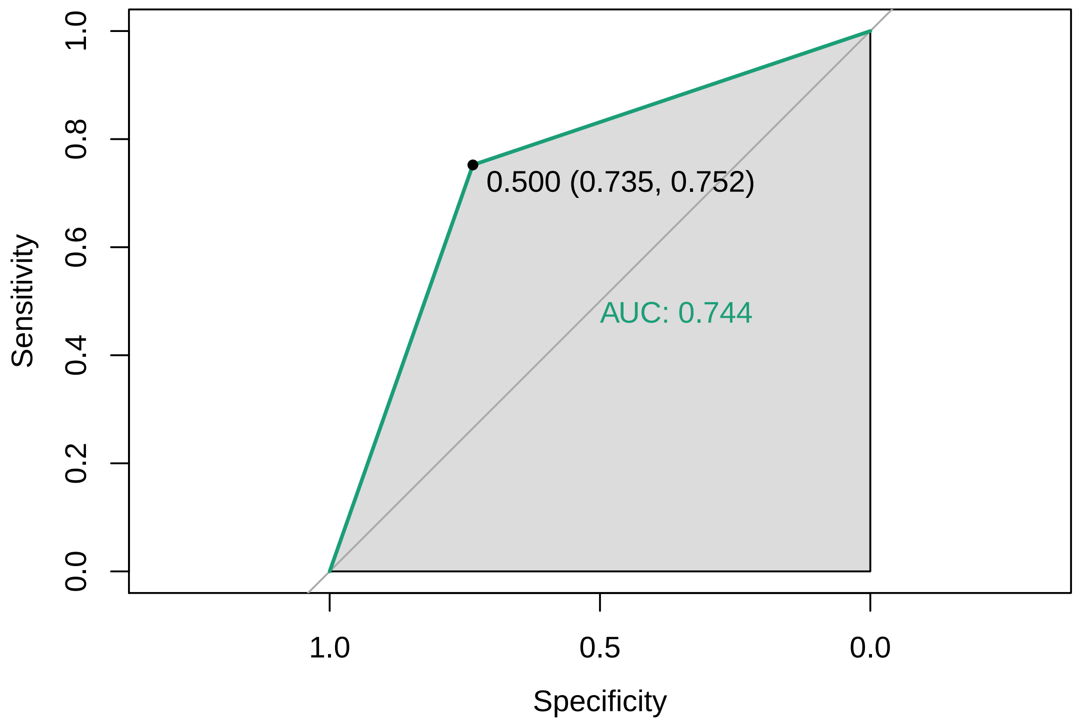

Figure 11: Classification Tree Model AUC and ROC Curve

Naive Bayes Model

```
#> Naive Bayes
#>
#> 10892 samples
#>     9 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 8714, 8713, 8714, 8714, 8713
#> Resampling results across tuning parameters:
#>
#>   usekernel  ROC        Sens       Spec
#>   FALSE      0.7281140  0.6044800  0.7264070
#>    TRUE      0.8555831  0.7868127  0.7550492
#>
#> 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  1846  140
#>        yes  487  526
#>
#>                Accuracy : 0.7909
#>                  95% CI : (0.7759, 0.8054)
#>     No Information Rate : 0.7779
#>     P-Value [Acc > NIR] : 0.04463
#>
#>                   Kappa : 0.4899
#>  Mcnemar's Test P-Value : < 2e-16
#>
#>             Sensitivity : 0.7913
#>             Specificity : 0.7898
#>          Pos Pred Value : 0.9295
#>          Neg Pred Value : 0.5192
#>              Prevalence : 0.7779
#>          Detection Rate : 0.6155
#>    Detection Prevalence : 0.6622
#>       Balanced Accuracy : 0.7905
#>
#>        'Positive' Class : no
#>
```

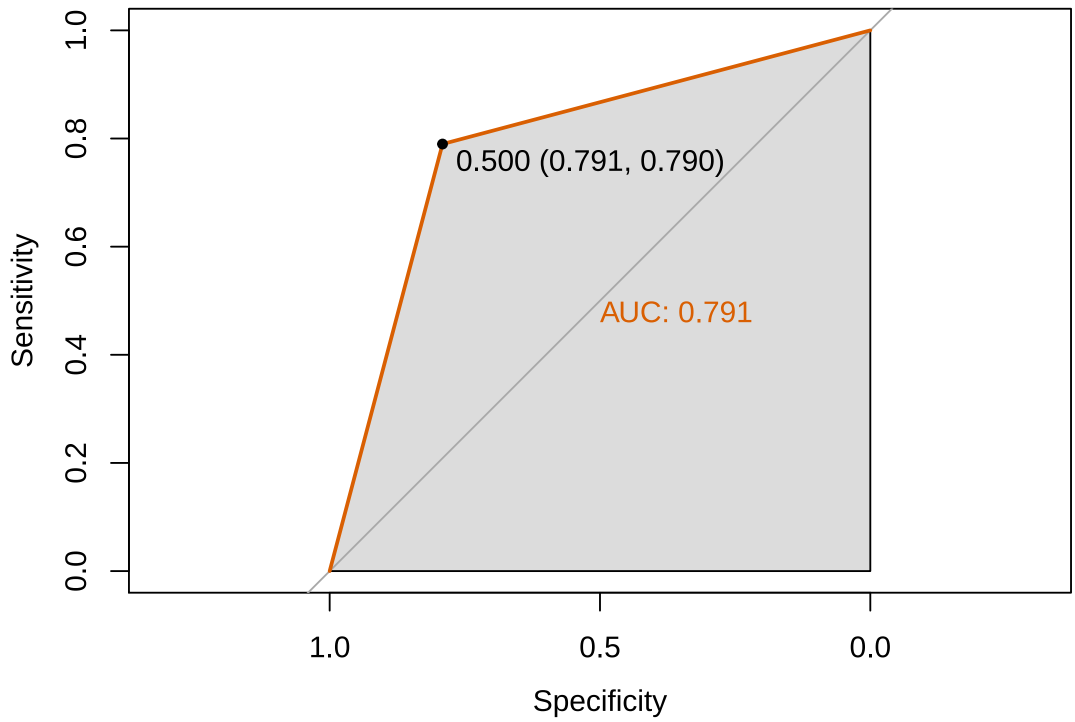

Figure 12: Naive Bayes Model AUC and ROC Curve

Random Forest Model

```
#> Random Forest
#>
#> 10892 samples
#>     9 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (2 fold)
```

```
#> Summary of sample sizes: 5446, 5446
#> Resampling results across tuning parameters:
#>
#>   mtry  ROC        Sens       Spec
#>    2    0.8892382  0.7882850  0.8158281
#>   29    0.9693558  0.8659567  0.9381197
#>   56    0.9669775  0.8545722  0.9388542
#>
#> ROC was used to select the optimal model using the largest value.
#> The final value used for the model was mtry = 29.
```

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

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   no  yes
#>        no  2108  273
#>        yes  225  393
#>
#>                Accuracy : 0.8339
#>                  95% CI : (0.8201, 0.8471)
#>     No Information Rate : 0.7779
#>     P-Value [Acc > NIR] : 1.405e-14
#>
#>                   Kappa : 0.5067
#>  Mcnemar's Test P-Value : 0.03519
#>
#>             Sensitivity : 0.9036
#>             Specificity : 0.5901
#>          Pos Pred Value : 0.8853
#>          Neg Pred Value : 0.6359
#>              Prevalence : 0.7779
#>          Detection Rate : 0.7029
#>    Detection Prevalence : 0.7939
#>       Balanced Accuracy : 0.7468
#>
#>        'Positive' Class : no
#>
```

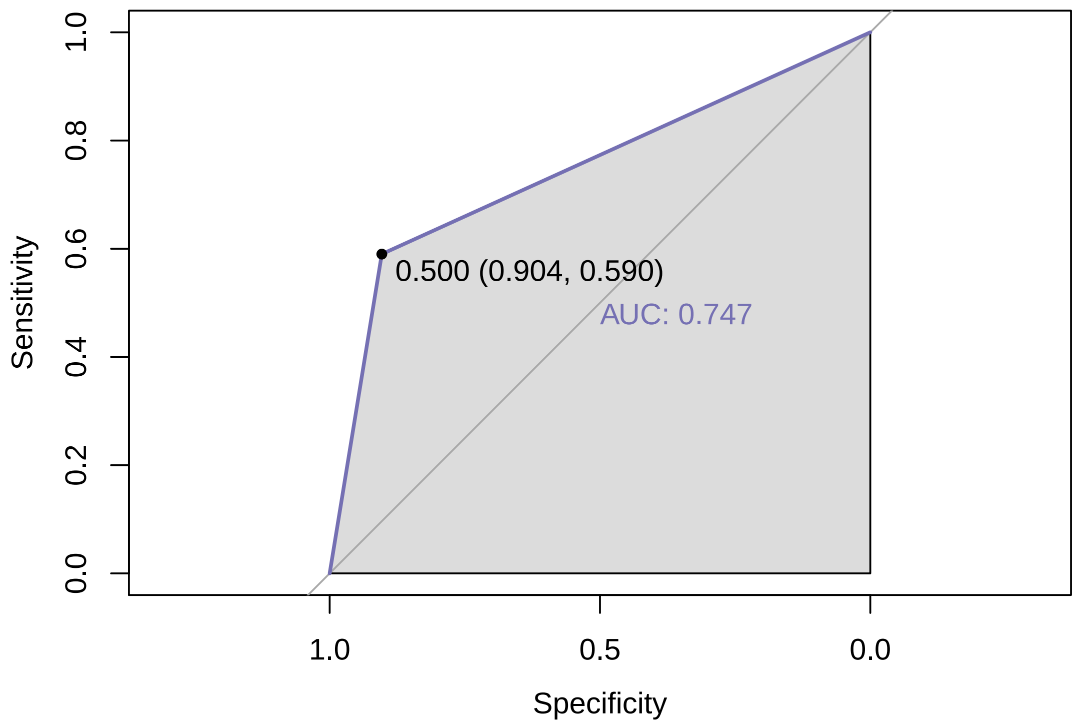

Figure 13: Random Forest Model AUC and ROC Curve

Logistic Regression Model

Logistic regression prediction accuracy will benefit if the data is normalized and are 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 categorical values (factors) to numeric data type

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

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction    0    1
#>          0 1832  121
#>          1  501  545
#>
#>                Accuracy : 0.7926
#>                  95% CI : (0.7776, 0.807)
#>     No Information Rate : 0.7779
#>     P-Value [Acc > NIR] : 0.02728
#>
#>                   Kappa : 0.5014
#>  Mcnemar's Test P-Value : < 2e-16
#>
#>             Sensitivity : 0.7853
#>             Specificity : 0.8183
#>          Pos Pred Value : 0.9380
#>          Neg Pred Value : 0.5210
#>              Prevalence : 0.7779
#>          Detection Rate : 0.6109
#>    Detection Prevalence : 0.6512
#>       Balanced Accuracy : 0.8018
#>
```

```
#>        'Positive' Class : 0
#>
```

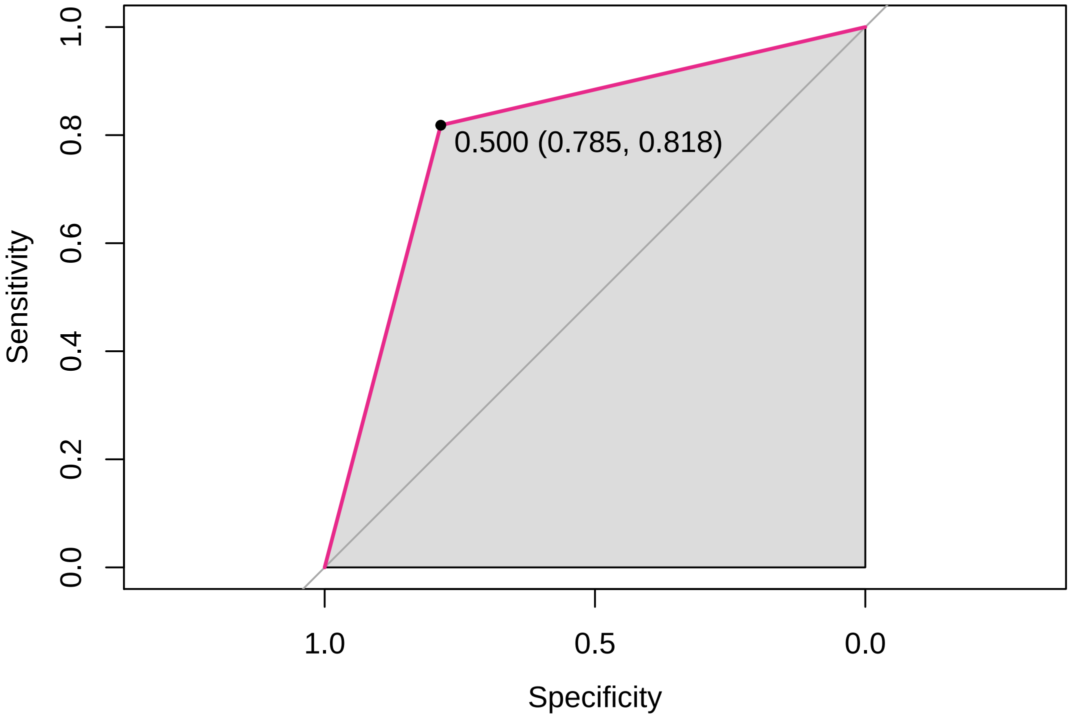

Figure 14: Logistic Regression Model AUC and ROC Curve

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

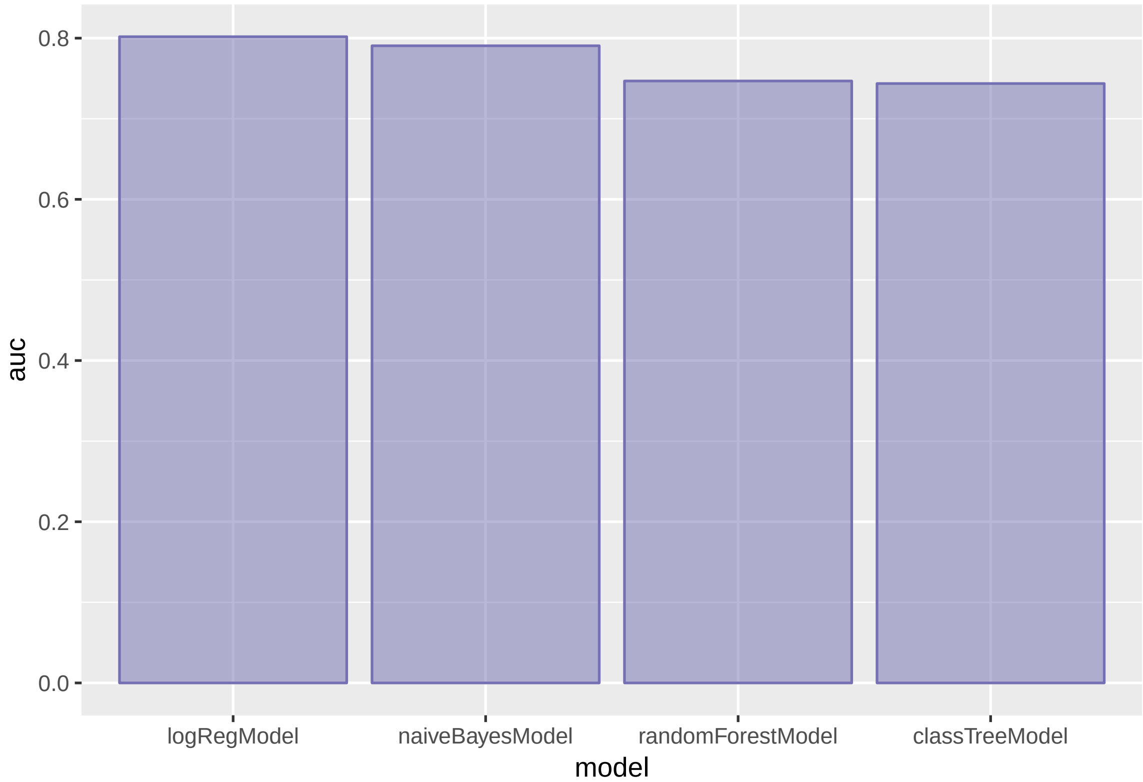

Figure 15: Model Accuracy Comparison

Model Deployment

Conclusion

Bibliography

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