HarvardX: PH125.9x Data Science Choose Your Own Project Australian Rain Prediction

Chanathip Eamdeengamlert

5/22/2021

Contents

Introduction **2**

The Problem Statement . . . 2

Essential Libraries . . . 2

The Dataset . . . 2

Executive Summary . . . 4

Method and Analysis **5**

Import Library . . . 5

Data Manipulation . . . 5

Exploratory Data Analysis . . . 12

Modeling approach . . . 23

Results **46**

Conclusion **46**

Introduction

This is a project as a part of the ninth and final course, HarvardX PH125.9x - Data Science: Capstone, in HarvardX's multi-part Data Science Professional Certificate series.

The Problem Statement

In this report, we will try to answer the question that whether or not it will rain tomorrow in Australia. We implement Logistic Regression with Python and Scikit-Learn.

To answer the question, we build a classifier to predict whether or not it will rain tomorrow in Australia. We train a binary classification model using Logistic Regression. I have used the Rain in Australia dataset for this project.

So, let's get started.

Essential Libraries

The first step in building the model is to import the necessary libraries.

```
# Importing required package
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(MLeval)) install.packages('MLeval', repos = "http://cran.us.r-project.org")
if(!require(pROC)) install.packages('pROC', repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages('glmnet', repos = "http://cran.us.r-project.org")
if(!require(Rborist)) install.packages('Rborist', repos = "http://cran.us.r-project.org")
if(!require(xgboost)) install.packages('xgboost', repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages('e1071', repos = "http://cran.us.r-project.org")
```

The Dataset

The next step is to import the dataset.

```
# Importing Data
url <- "https://raw.githubusercontent.com/chanathipea/AustralianRainProject/main/weatherAUS.csv"
tmp_filename <- tempfile()
download.file(url, tmp_filename)
dat <- read_csv(tmp_filename)
file.remove(tmp_filename)

ausrain <- as.data.frame(dat)
```

View dimensions of dataset

```
dim(ausrain)
```

```
## [1] 145460     23
```

Preview the dataset

```r
head(ausrain)
```

```
##         Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir
## 1 2008-12-01   Albury    13.4    22.9      0.6          NA       NA           W
## 2 2008-12-02   Albury     7.4    25.1      0.0          NA       NA         WNW
## 3 2008-12-03   Albury    12.9    25.7      0.0          NA       NA         WSW
## 4 2008-12-04   Albury     9.2    28.0      0.0          NA       NA          NE
## 5 2008-12-05   Albury    17.5    32.3      1.0          NA       NA           W
## 6 2008-12-06   Albury    14.6    29.7      0.2          NA       NA         WNW
##   WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am
## 1            44          W        WNW           20           24          71
## 2            44        NNW        WSW            4           22          44
## 3            46          W        WSW           19           26          38
## 4            24         SE          E           11            9          45
## 5            41        ENE         NW            7           20          82
## 6            56          W          W           19           24          55
##   Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm
## 1          22      1007.7      1007.1        8       NA    16.9    21.8
## 2          25      1010.6      1007.8       NA       NA    17.2    24.3
## 3          30      1007.6      1008.7       NA        2    21.0    23.2
## 4          16      1017.6      1012.8       NA       NA    18.1    26.5
## 5          33      1010.8      1006.0        7        8    17.8    29.7
## 6          23      1009.2      1005.4       NA       NA    20.6    28.9
##   RainToday RainTomorrow
## 1        No           No
## 2        No           No
## 3        No           No
## 4        No           No
## 5        No           No
## 6        No           No
```

View column names

```r
colnames(ausrain)
```

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

View summary of dataset

```r
str(ausrain, give.attr = FALSE)
```

```
## 'data.frame':    145460 obs. of  23 variables:
##  $ Date         : Date, format: "2008-12-01" "2008-12-02" ...
##  $ Location     : chr  "Albury" "Albury" "Albury" "Albury" ...
##  $ 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  : logi  NA NA NA NA NA NA ...
##  $ Sunshine     : logi  NA NA NA NA NA NA ...
##  $ WindGustDir  : chr  "W" "WNW" "WSW" "NE" ...
##  $ WindGustSpeed: num  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindDir9am   : chr  "W" "NNW" "W" "SE" ...
##  $ WindDir3pm   : chr  "WNW" "WSW" "WSW" "E" ...
##  $ WindSpeed9am : num  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : num  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : num  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : num  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     : num  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : num  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    : chr  "No" "No" "No" "No" ...
##  $ RainTomorrow : chr  "No" "No" "No" "No" ...
```

We can see that the dataset contains mixture of date, character logical and numerical variables. Also, there are some missing values in the dataset.

Executive Summary

After import and load all required library, data cleansing and data manipulation was done in first step.

The target variable is RainTomorrow. So, firstly, The missing value in Raintomorrow will be removed. Then check the categorical column and convert to categorical type. The column with more 50% of missing value also will be removed.

Missing value is also explored with correlation to other variable. More column is removed since the missing value is affect to other variable such as missing value in pressure column is covered the whole category in location column. The rest of the missing value is removed by row.

Time stamp column is formatted to date-time data type. Month feature were extracted from existing column in order to further explore in exploratory analysis.

Then, exploratory data analysis was done in second step. The continuous variables are explored to see the distribution by target variable and other catagorical variables. The categorical variables are explored by target variable and its ration to target variable.

Insights from EDA has shown that the correlation between continuous variable and target variable is not very significant correlated. By exploring categorical variables, the difference in ration by target variable are noticable but also not very clear separator for target variable.

Since the current predictor is not very correlated and not very separation for target variable, The final approach is to use the ensemble classification algorithm. The logistic regression and normalized logistic regression with be used as the benchmark then random forest and extreme gradient boost will be used as the final model.

The autralian rain data is separated into 80% training set and 20% test set to fit model and test the algorithm. The ROC will be used for evaluation since the target is imbalance date and the objective is focused on predicting both positive condition and negative condition.

Random forest model shown the best performance for this project with all variable in the model.

Method and Analysis

Import Library

The first step in building the model is to import the necessary libraries.

```
library(tidyverse)
library(caret)
library(lubridate)
library(pROC)
library(MLeval)
library(glmnet)
library(Rborist)
library(xgboost)
library(e1071)
library(doParallel)
```

Data Manipulation

Firstly, trying to predict raining on tomorrow day.So NA of RainTomorrow will be filtered out.

```
# So NA of RainTomorrow will be filtered out.
ausrain <- ausrain %>% filter(!is.na(RainTomorrow))
```

Confirm no incorrect record by seeing the unique value before convert to factor.

```
# Confirm no incorrect record by seeing the unique value before convert to factor
ausrain %>% summarize(Today = unique(RainToday)) %>% as.list(.)
```

```
## $Today
## [1] "No"  "Yes" NA
```

```
ausrain %>% summarize(Tomorrow = unique(RainTomorrow)) %>% as.list(.)
```

```
## $Tomorrow
## [1] "No"  "Yes"
```

```
ausrain %>% summarize(WindDir = unique(WindGustDir)) %>% as.list(.)
```

```
## $WindDir
##  [1] "W"   "WNW" "WSW" "NE"  "NNW" "N"   "NNE" "SW"  "ENE" "SSE" "S"   "NW"
## [13] "SE"  "ESE" NA    "E"   "SSW"
```

```
ausrain %>% summarize(Wind9am = unique(WindDir9am)) %>% as.list(.)
```

```
## $Wind9am
##  [1] "W"   "NNW" "SE"  "ENE" "SW"  "SSE" "S"   "NE"  NA    "SSW" "N"   "WSW"
## [13] "ESE" "E"   "NW"  "WNW" "NNE"
```

```
ausrain %>% summarize(Wind3pm = unique(WindDir3pm)) %>% as.list(.)
```

```
## $Wind3pm
##  [1] "WNW" "WSW" "E"   "NW"  "W"   "SSE" "ESE" "ENE" "NNW" "SSW" "SW"  "SE"
## [13] "N"   "S"   "NNE" NA    "NE"
```

```
ausrain %>% summarize(Location = unique(Location)) %>% as.list(.)
```

```
## $Location
##  [1] "Albury"           "BadgerysCreek"    "Cobar"            "CoffsHarbour"
##  [5] "Moree"            "Newcastle"        "NorahHead"        "NorfolkIsland"
##  [9] "Penrith"          "Richmond"         "Sydney"           "SydneyAirport"
## [13] "WaggaWagga"       "Williamtown"      "Wollongong"       "Canberra"
## [17] "Tuggeranong"      "MountGinini"      "Ballarat"         "Bendigo"
## [21] "Sale"             "MelbourneAirport" "Melbourne"        "Mildura"
## [25] "Nhil"             "Portland"         "Watsonia"         "Dartmoor"
## [29] "Brisbane"         "Cairns"           "GoldCoast"        "Townsville"
## [33] "Adelaide"         "MountGambier"     "Nuriootpa"        "Woomera"
## [37] "Albany"           "Witchcliffe"      "PearceRAAF"       "PerthAirport"
## [41] "Perth"            "SalmonGums"       "Walpole"          "Hobart"
## [45] "Launceston"       "AliceSprings"     "Darwin"           "Katherine"
## [49] "Uluru"
```

Then convert all character and logical value to factor with specific level.

```
ausrain <- ausrain %>%
  mutate(RainToday = factor(RainToday, levels = c("Yes", "No")),
         RainTomorrow = factor(RainTomorrow, levels = c("Yes", "No")),
         WindGustDir = as.factor(WindGustDir),
         WindDir9am = as.factor(WindDir9am),
         WindDir3pm = as.factor(WindDir3pm),
         Location = as.factor(Location))
summary(ausrain)
```

```
##       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   Mode :logical   Mode :logical   W      : 9780
##  1st Qu.:  0.00   FALSE:240       FALSE:2308      SE     : 9309
##  Median :  0.00   TRUE :1563      TRUE :326       E      : 9071
##  Mean   :  2.35   NA's :140390    NA's :139559    N      : 9033
##  3rd Qu.:  0.80                                   SSE    : 8993
##  Max.   :371.00                                   (Other):86677
##  NA's   :1406                                     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   Yes : 31455   Yes: 31877
##  1st Qu.:16.60   No  :109332   No :110316
##  Median :21.10   NA's:  1406
##  Mean   :21.69
##  3rd Qu.:26.40
##  Max.   :46.70
##  NA's   :2726
```

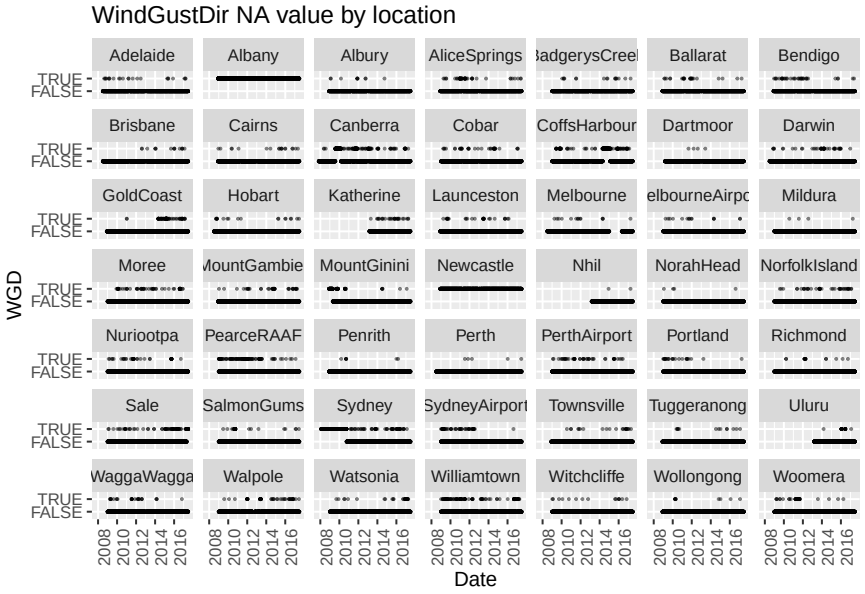
It seem that some of the column contain more than 30% of NA value such as Evaporation, Sunshine NA, Cloud9am, and Cloud3pm. So, trying to remove those columns from analysis since to impute the value to this much of missing value may deviate the actual observation.

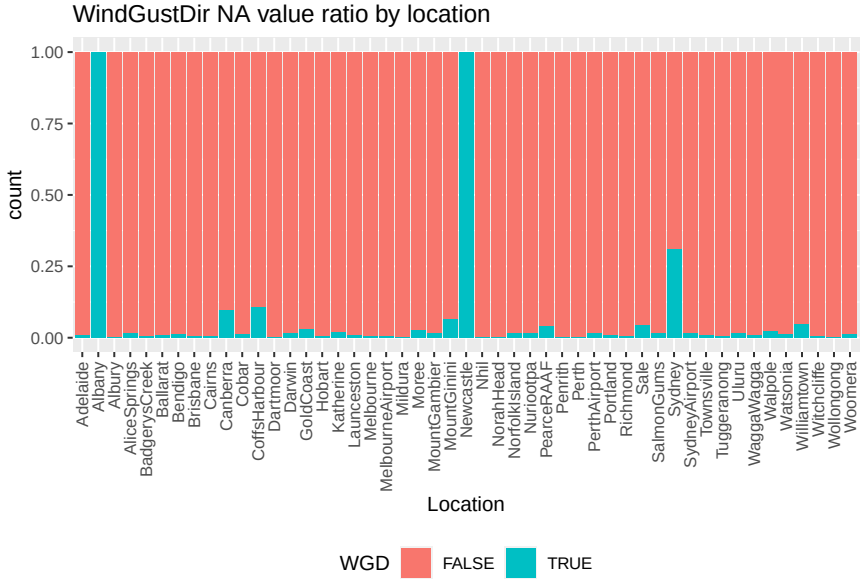
```
ausrain <- ausrain %>% select(-c(Evaporation , Sunshine, Cloud9am, Cloud3pm))
```

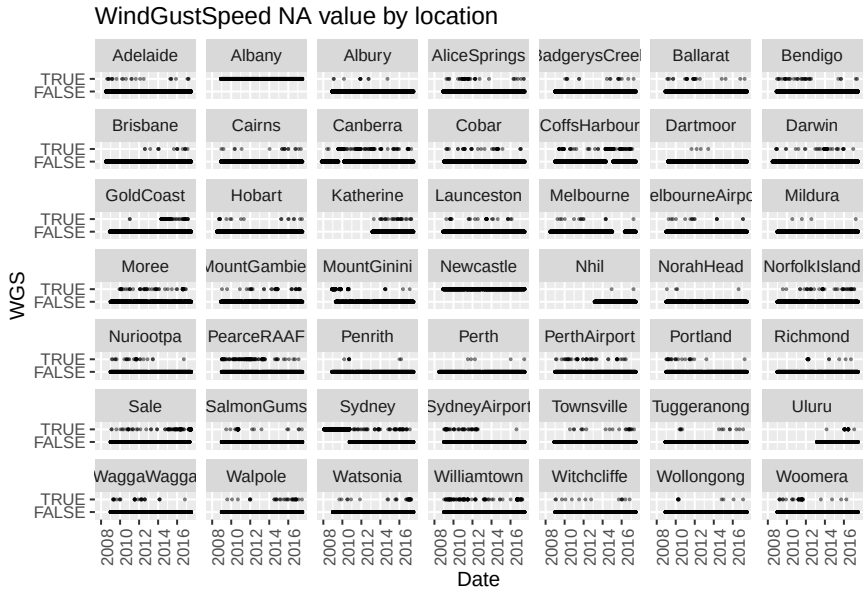
```
# The rest of missing values are explore separately
nadata <- ausrain %>% mutate(WGD = is.na(WindGustDir),
                             WGS = is.na(WindGustSpeed),
                             WD9 = is.na(WindDir9am),
                             WD3 = is.na(WindDir3pm),
                             WS9 = is.na(WindSpeed9am),
                             WS3 = is.na(WindSpeed3pm),
                             HD9 = is.na(Humidity9am),
                             HD3 = is.na(Humidity3pm),
                             P9na = is.na(Pressure9am),
                             P3na = is.na(Pressure3pm),
                             T9 = is.na(Temp9am),
                             T3 = is.na(Temp3pm))
```

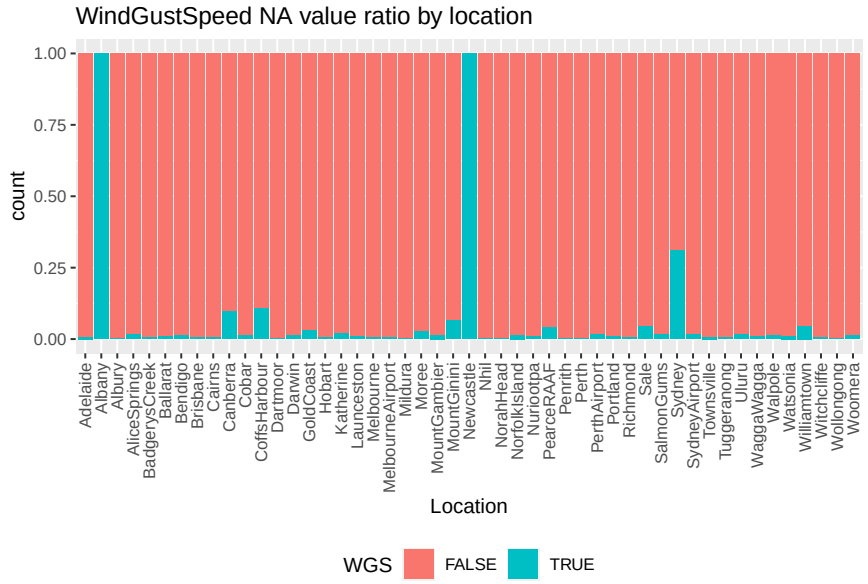
All variable were explore in the same manor but only 4 significant variables are shown in this report the rest will be contain in the original r script, WindGustDir, WindGustSpeed, Pressure9am and Pressure3pm.

Whole Newcastle and Albany do not have WindGustDir and WindGustSpeed observation. If we filter out the NA observations in WindGustDir and WindGustSpeed, the whole 2 locations observation will be missing. In order to retain 2 locations, we select to removed those 2 columns.

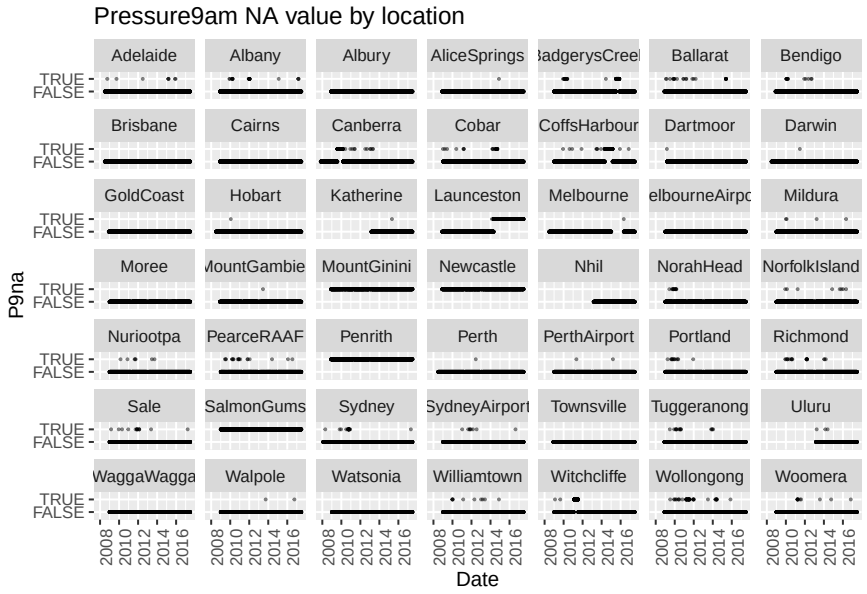


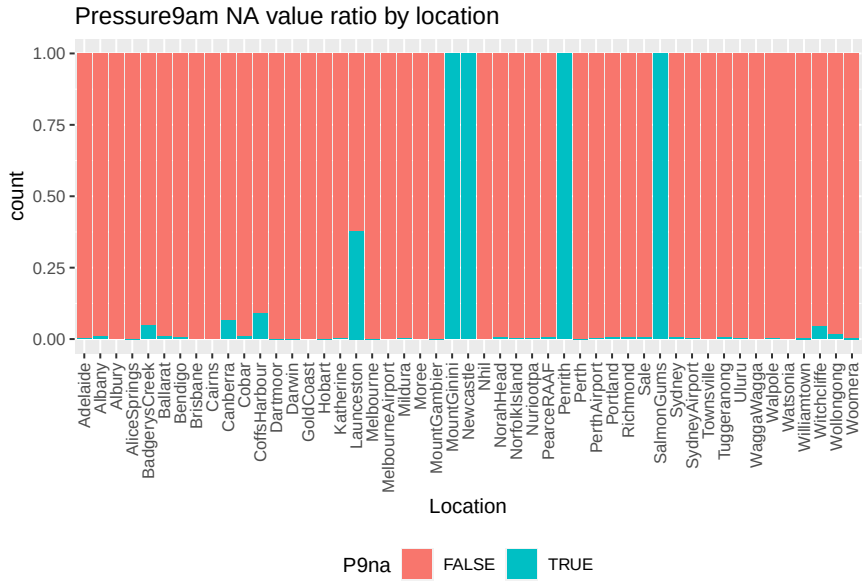


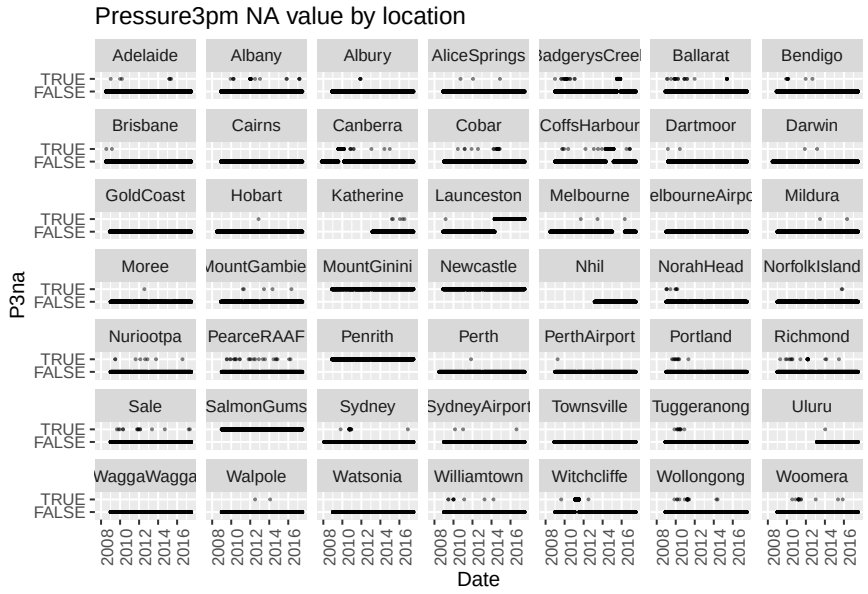


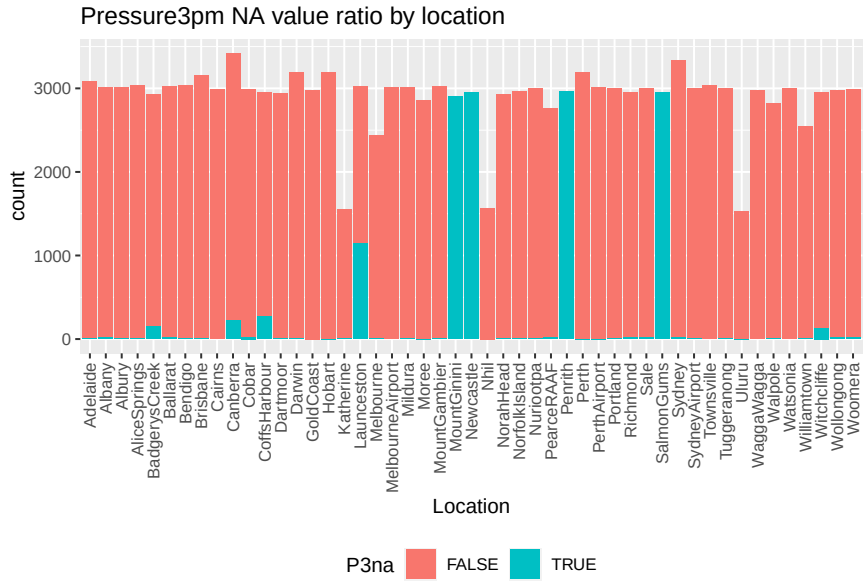


Even Pressure9am and Pressure3pm have about 10% of NA value which is a small number.But when looking into MountGinini, Newcastle, Penrith and SalmonGums, they do not have Pressure9am and Pressure3pm observation. If we filter out the NA observations, the whole 4 locations observation will be missing. In order to retain 4 locations, those 2 columns will be removed the same way as WindGustDir and WindGustSpeed.


Pressure9am NA value by location
Adelaide
Albany
Albury
AliceSprings
BadgerysCreek
Ballarat
Bendigo
Brisbane
Cairns
Canberra
Cobar
CoffsHarbour
Dartmoor
Darwin
GoldCoast
Hobart
Katherine
Launceston
Melbourne
MelbourneAirpo
Mildura
Moree
MountGambier
MountGinini
Newcastle
Nhil
NorahHead
NorfolkIsland
Nuriootpa
PearceRAAF
Penrith
Perth
PerthAirport
Portland
Richmond
Sale
SalmonGums
Sydney
SydneyAirport
Townsville
Tuggeranong
Uluru
WaggaWagga
Walpole
Watsonia
Williamtown
Witchcliffe
Wollongong
Woomera
TRUE
FALSE
P9na
2008
2010
2012
2014
2016
Date


Pressure9am NA value ratio by location
1.00
0.75
0.50
0.25
0.00
count
Adelaide
Albany
Albury
AliceSprings
BadgerysCreek
Ballarat
Bendigo
Brisbane
Cairns
Canberra
Cobar
CoffsHarbour
Dartmoor
Darwin
GoldCoast
Hobart
Katherine
Launceston
Melbourne
MelbourneAirport
Mildura
Moree
MountGambier
MountGinini
Newcastle
Nhil
NorahHead
NorfolkIsland
Nuriootpa
PearceRAAF
Penrith
Perth
PerthAirport
Portland
Richmond
Sale
SalmonGums
Sydney
SydneyAirport
Townsville
Tuggeranong
Uluru
WaggaWagga
Walpole
Watsonia
Williamtown
Witchcliffe
Wollongong
Woomera
Location
P9na
FALSE
TRUE





As we explore above, 4 variables are removed

```
# 4 variables are removed
ausrain <- ausrain %>% select(-c(Pressure9am, Pressure3pm, WindGustDir, WindGustSpeed))
ausrain <- ausrain %>% drop_na()
```

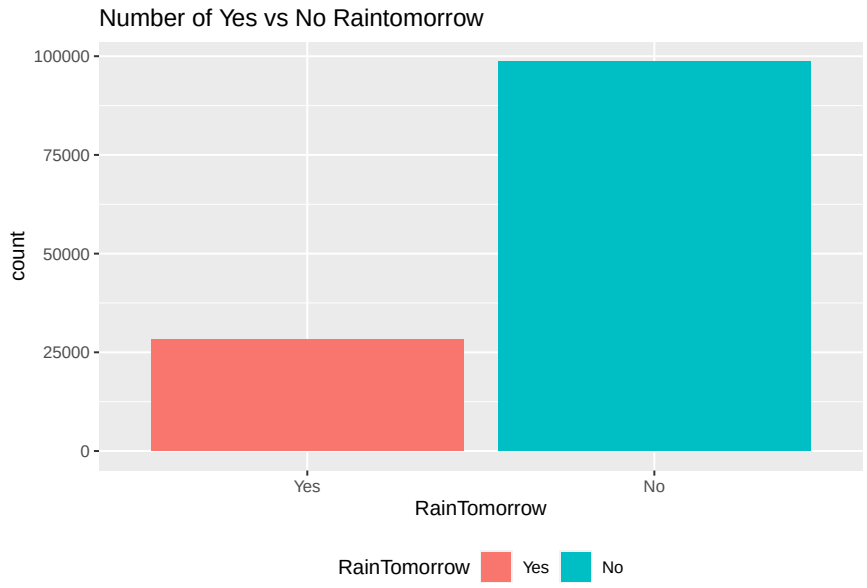
Next, we look into Date variable. We can see that there is a Date variable which is still not very useful for prediction since they are all unique for each column and there will be no repeated date in the future. So, Month data is extracted from Date column to use as predictor in order to see if they are some seasonal effect.

```
# Get month variable
ausrain <- ausrain %>% mutate(Month = month(Date)) %>% mutate(Month = as.factor(Month))
```

Exploratory Data Analysis

We have imported the data. Now, its time to explore the data to gain insights about it.

Target Variable Analysis




```
ausrain %>% group_by(RainTomorrow) %>% summarize(count = n())
```

```
## # A tibble: 2 x 2
##   RainTomorrow count
##   <fct>        <int>
## 1 Yes          28287
## 2 No           98746
```

The target variable is RainTomorrow which is binary categorical variable. The above univariate plot confirms our findings that the No variable have 98746 entries, and the Yes variable have 28287 entries. The target variable is in imbalance since it is more than 3 times difference in target factor levels.

Predictors Analysis

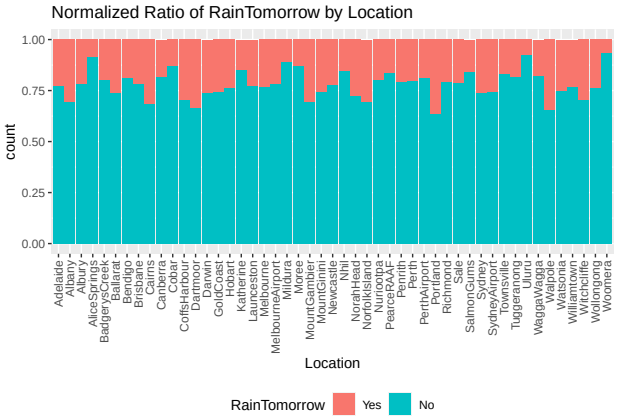
In this section, I segregate the dataset into categorical and numerical variables. There are a mixture of categorical and numerical variables in the dataset. First of all, I will find categorical variables.

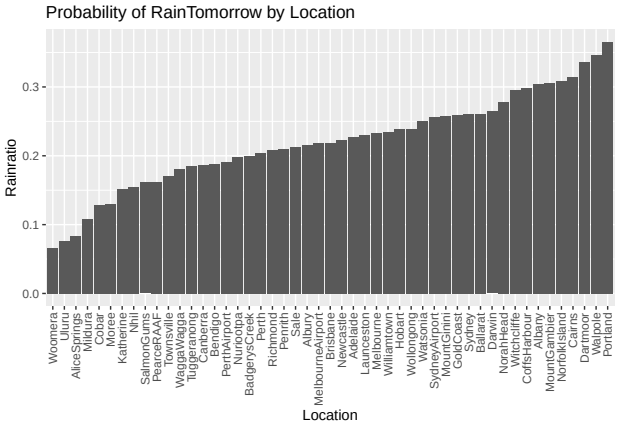
Explore Categorical Variables

There are 5 categorical variables. These are given by Location, WindDir9am, WindDir3pm, RainToday and Month. Excluding the target variable, there are one binary categorical predictor variable - RainToday. Each categorical variable will be explored by count frequency of each category and compare the ratio between target variable.

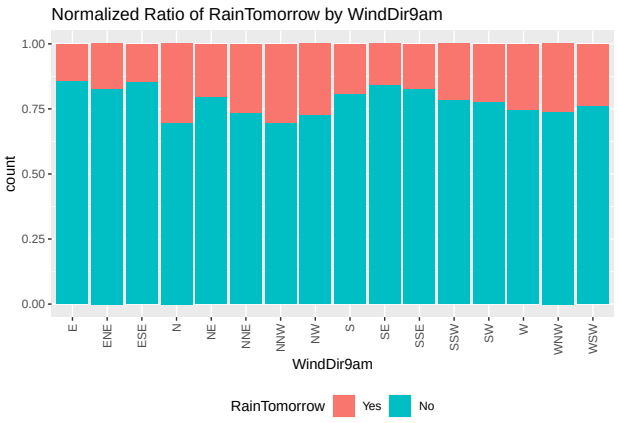
We would like to if there is any difference in ratio between target variable factor, Yes, it will rain tomorrow and No, it will not rain tomorrow, and also difference in ratio between each its own factor levels.

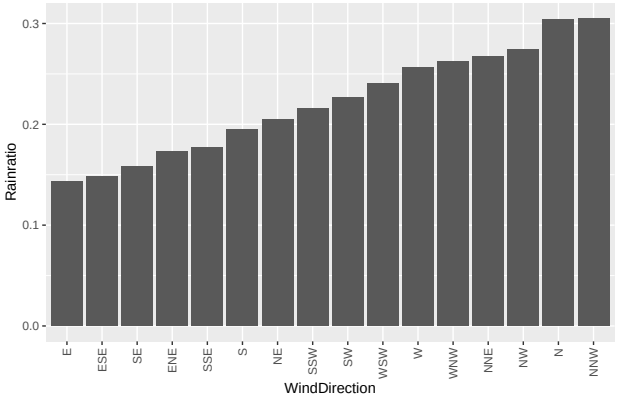
Explore by Location There is significant difference of rain probability on each Location.



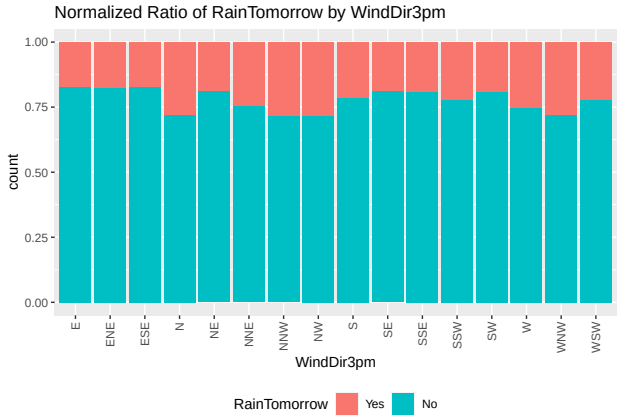


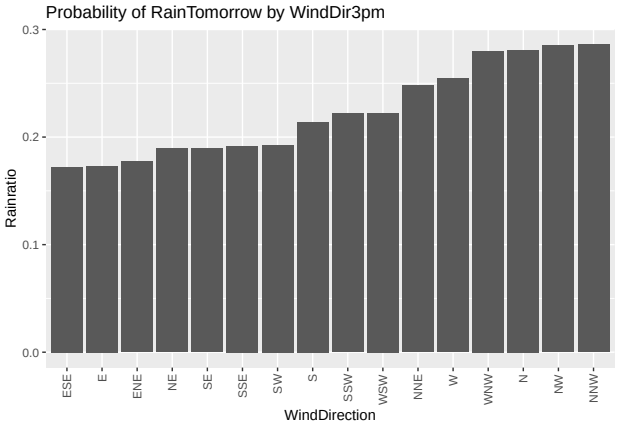
Explore by WindDir9am The RainTomorrow is also separable by WindDir9am variable. The ratio between Yes and No case of RainTomorrow is show the difference on each WindDir9am factor.



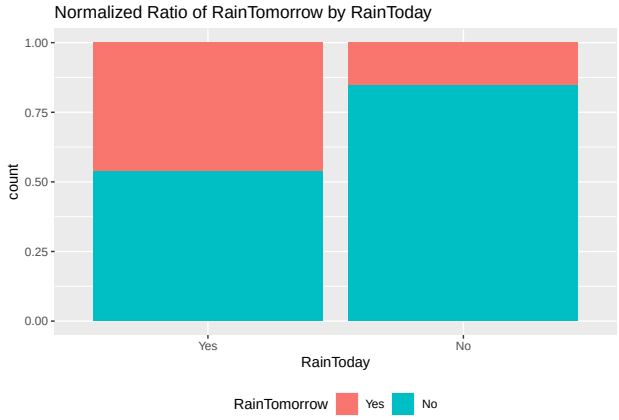


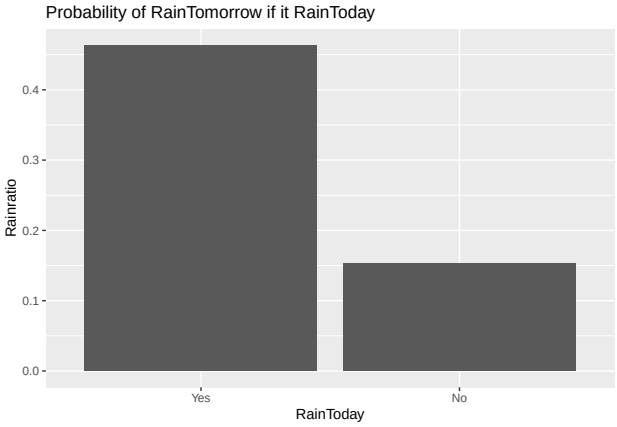
Explore by WindDir3pm The RainTomorrow is also separable by WindDir9am variable, but the difference between each wind direction is quite small.



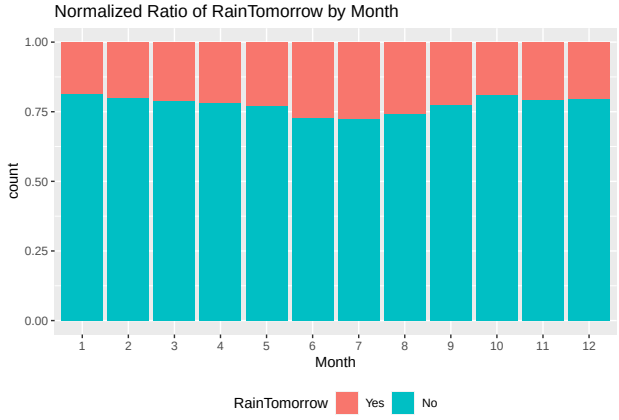


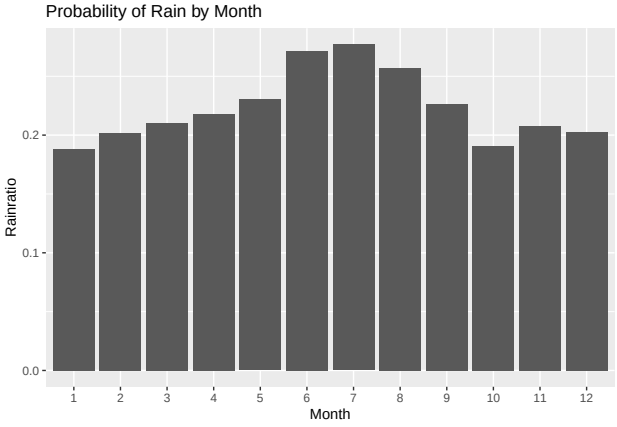
Explore by RainToday The RainTomorrow is also separable by RainToday if today is not raining, but if today is raining a chance of raining tomorrow is 50%.





Explore by Month The RainTomorrow is separable by Month variable. The ratio between Yes and No case of RainTomorrow is show the difference on each month.

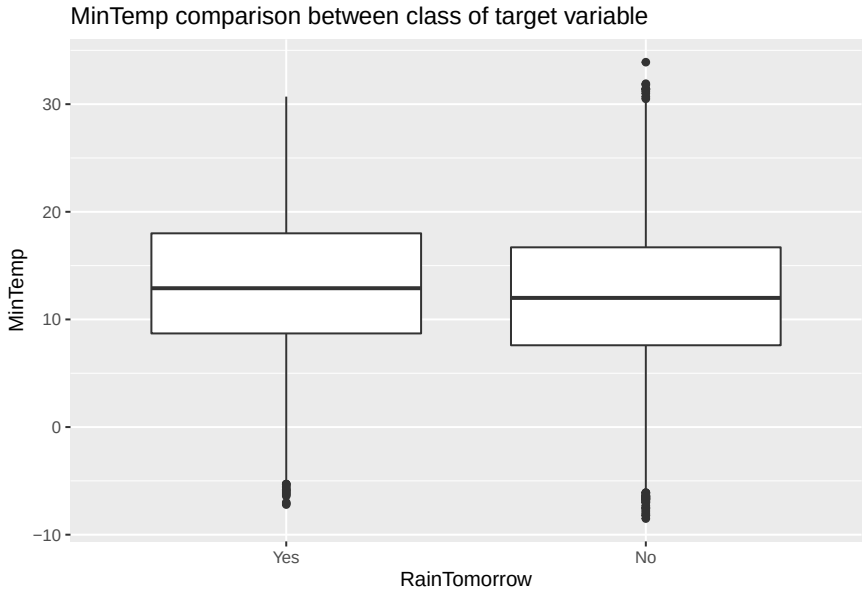


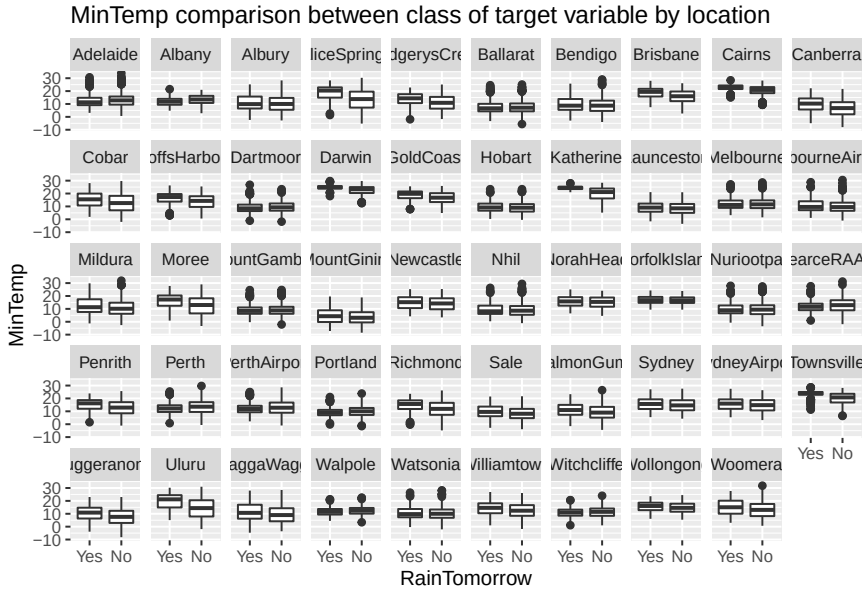


Explore Continuous Variables

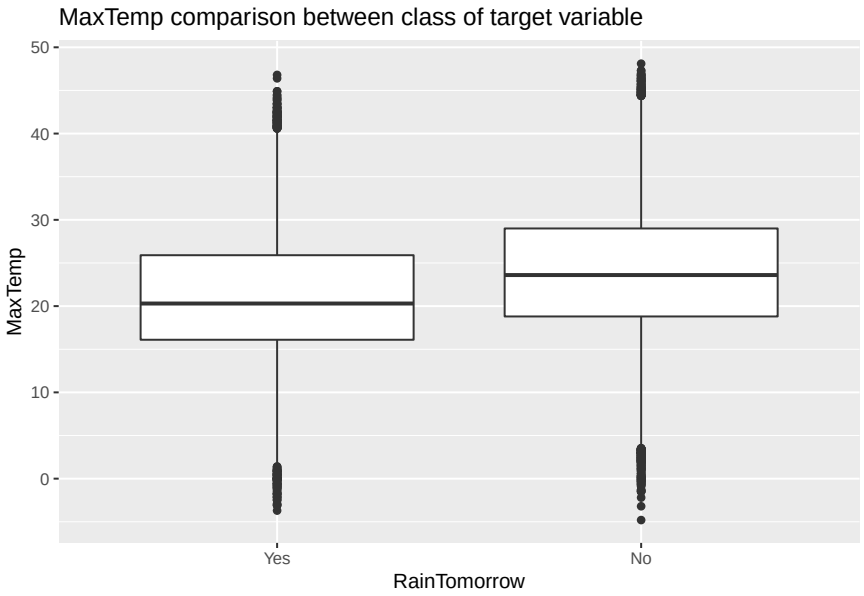
There are 9 continuous variables left after data cleansing. These are given by MinTemp, MaxTemp, Rainfall, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Temp9am, and Temp3pm. The continuous variables will be visualized by boxplots to check the distribution and compare the distribute between 2 target variable. Further more, some variable will be faceted with other variable to check any significant relationship.

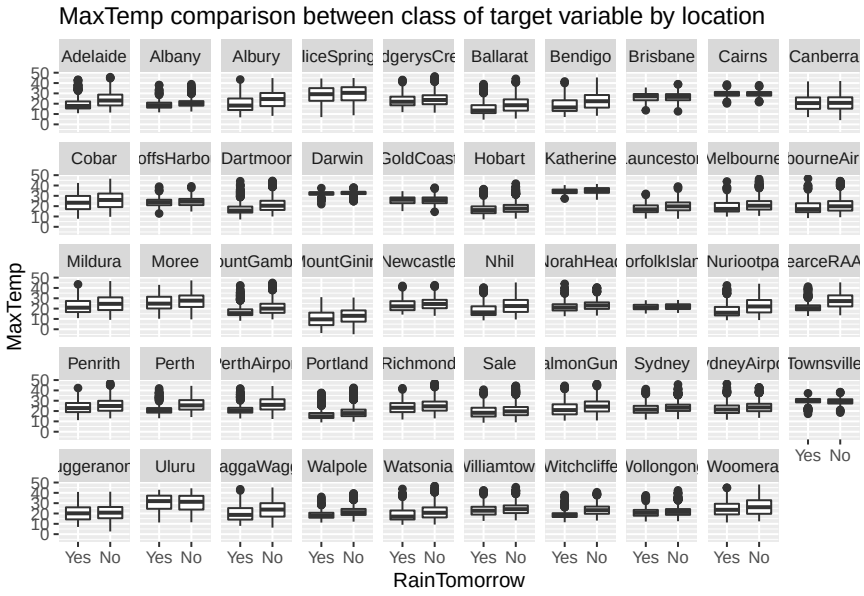
Explore Mintemp There is not a clear difference in mean of Mintemp when compare between 2 class of RainTomorrow. But when classify deeper into each location, there are some distinct relationship such as in AliceSprings, mean mintemp is higher if tomorrow is raining or in Albany, mintemp is lower if tomorrow is raining. Then Mintemp may be use full in the prediction.



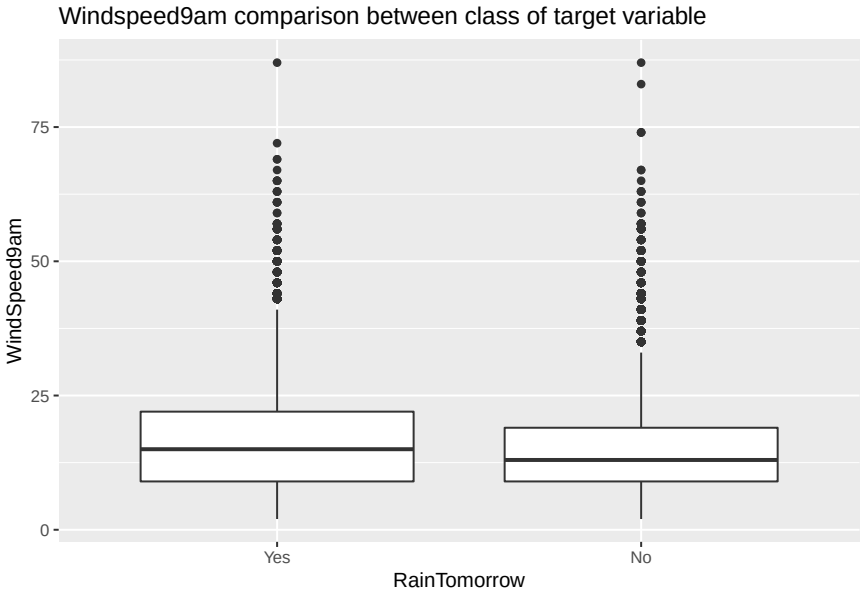


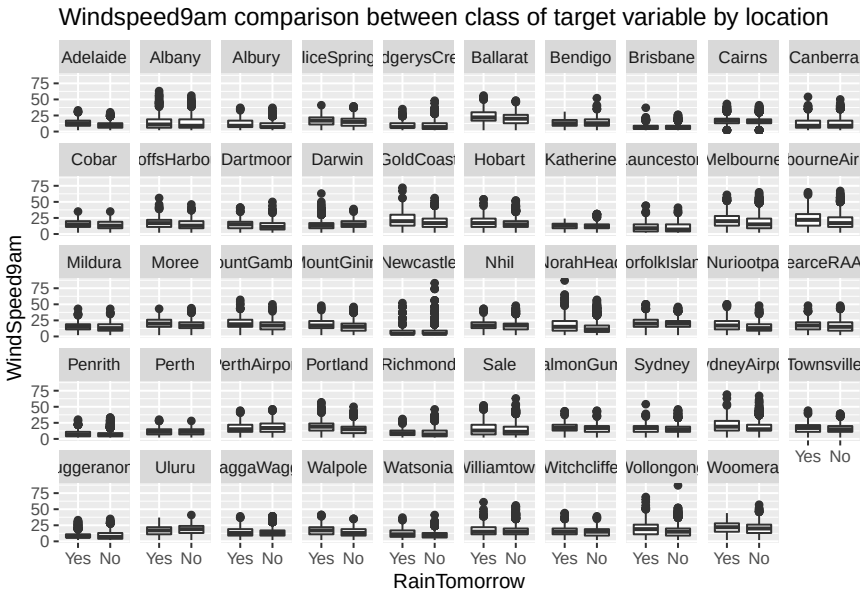
Explore Maxtemp There is not a clear difference in mean of Maxtemp when compare between 2 class of RainTomorrow. But when classify deeper into each location, there is a common relationship in every location. When there will be rain tomorrow, Maxtemp will be lower.



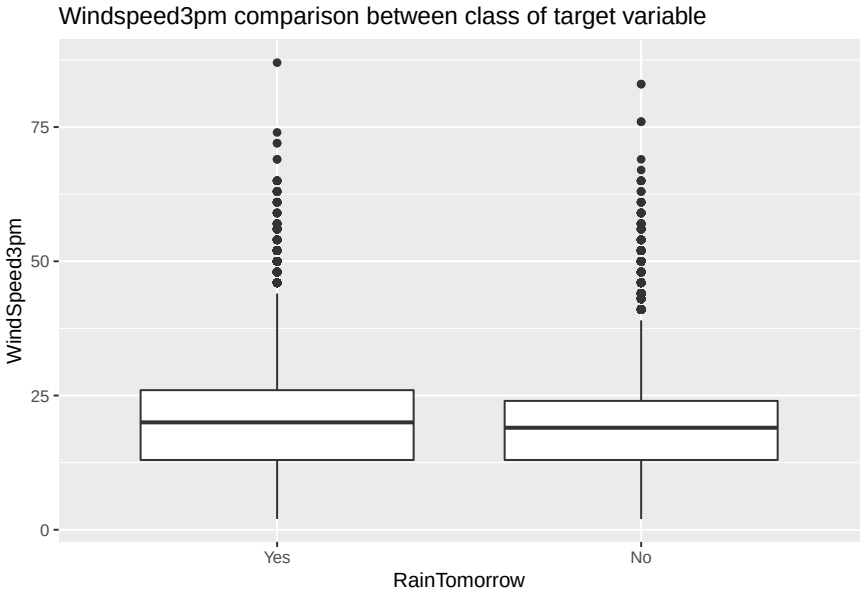


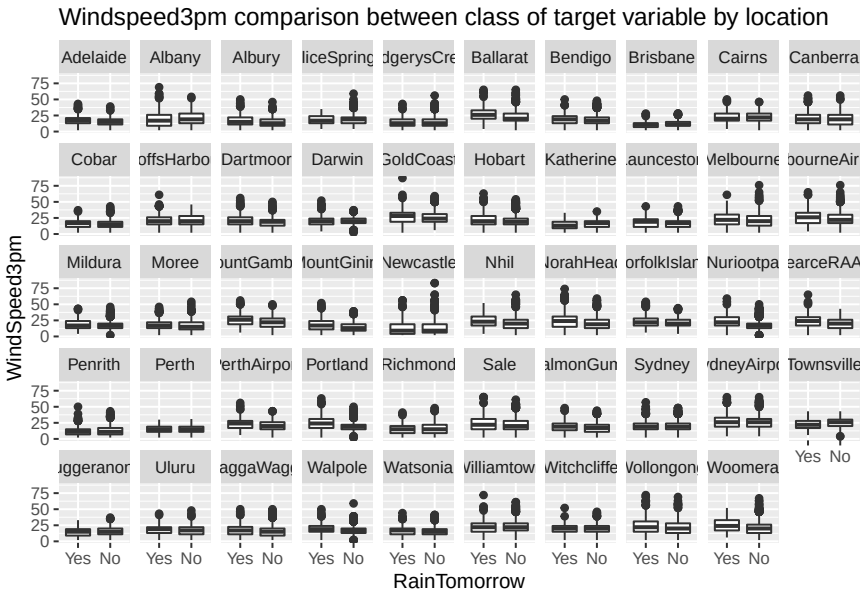
Explore Windspeed9am There is not a clear difference in mean of Windspeed9am when compare between 2 class of RainTomorrow.



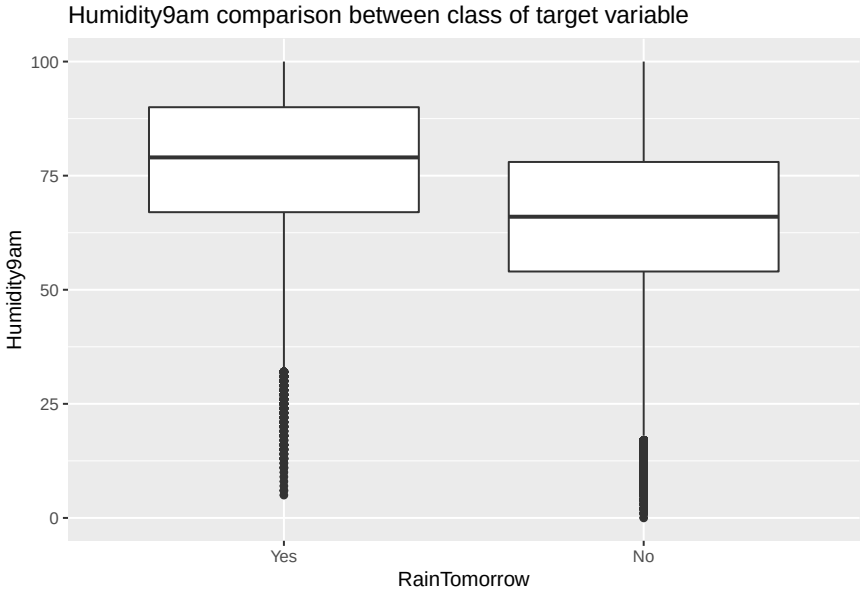


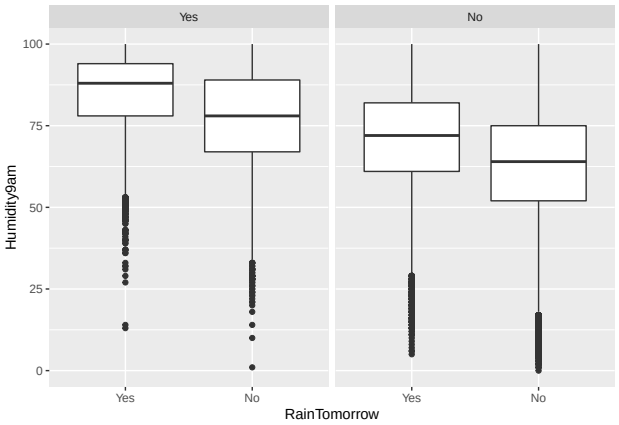
Explore Windspeed3pm There is not a clear difference in mean of Windspeed3pm when compare between 2 class of RainTomorrow.

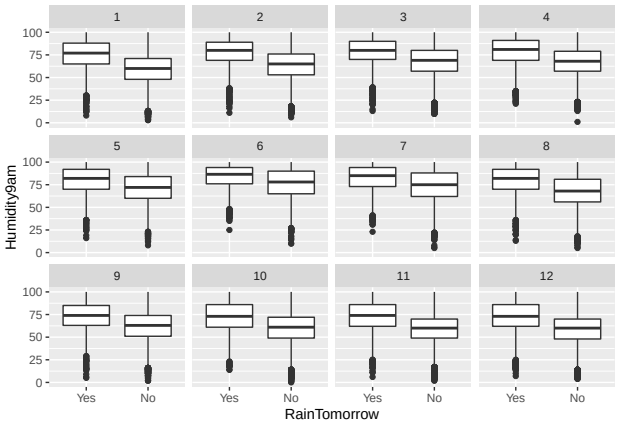




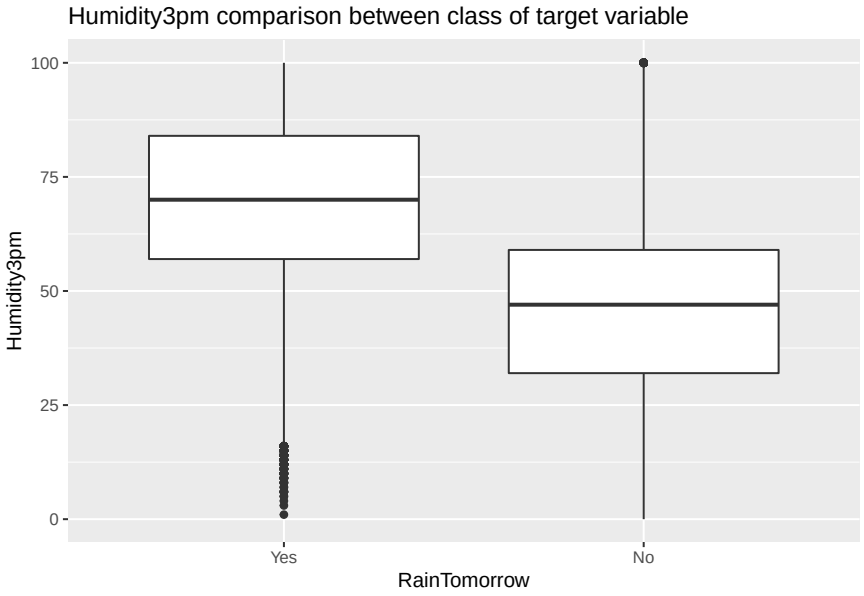
Explore Humidity9am There is quite significant difference in mean of Humidity9am when compare between 2 class of RainTomorrow. it is actually explore deeply into each variable such as Location, Month and Raintoday variable. Those will not shows in this report to keep this report not to long but the comparison shows similar relationship.

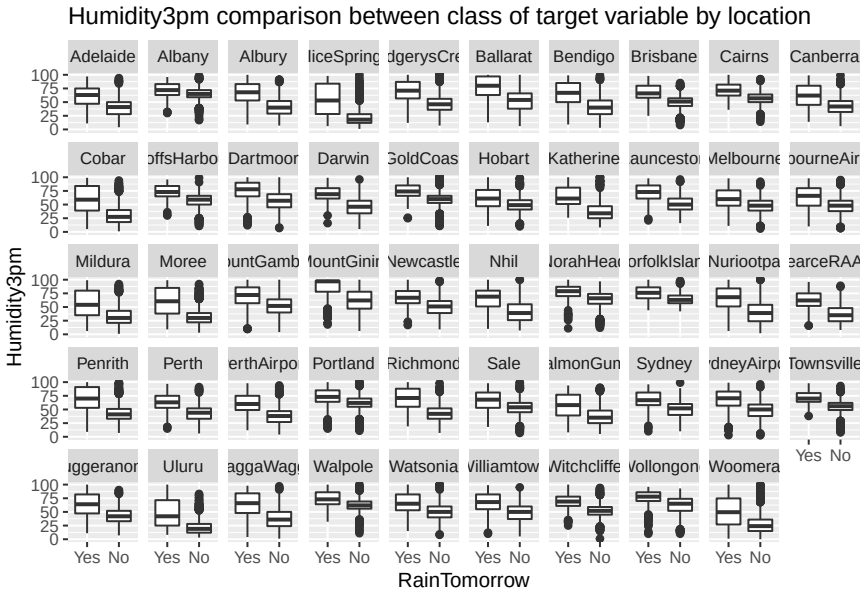




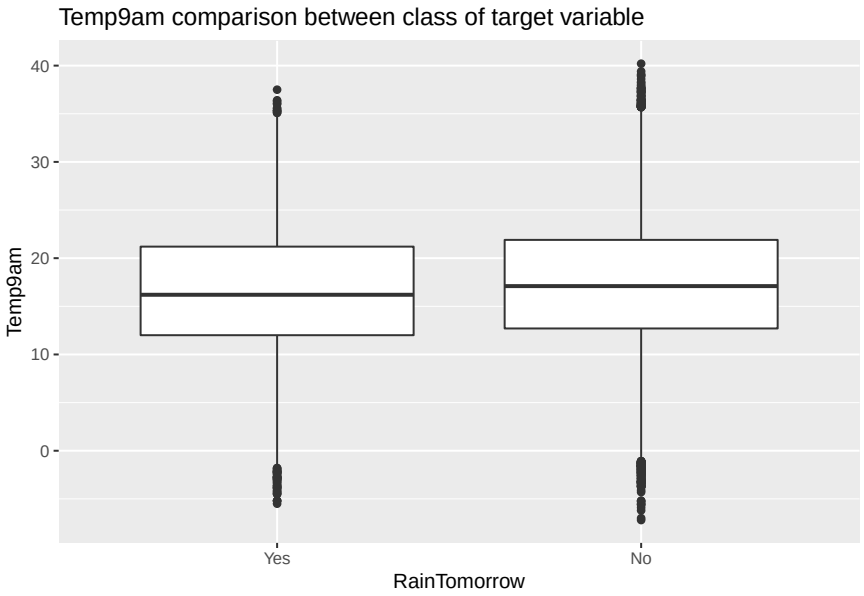


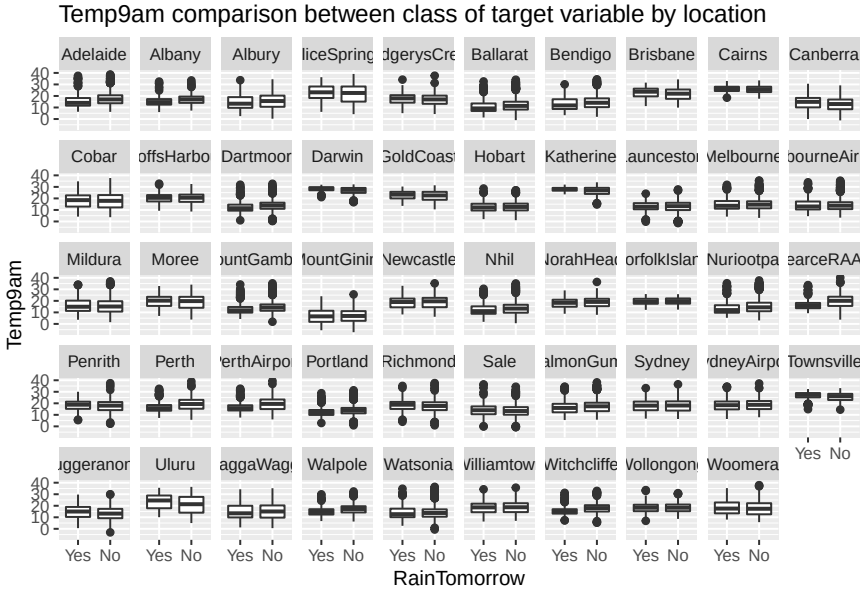
Explore Humidity3pm Humidity3pm even show more significant difference in mean of Humidity9am when compare between 2 class of RainTomorrow.



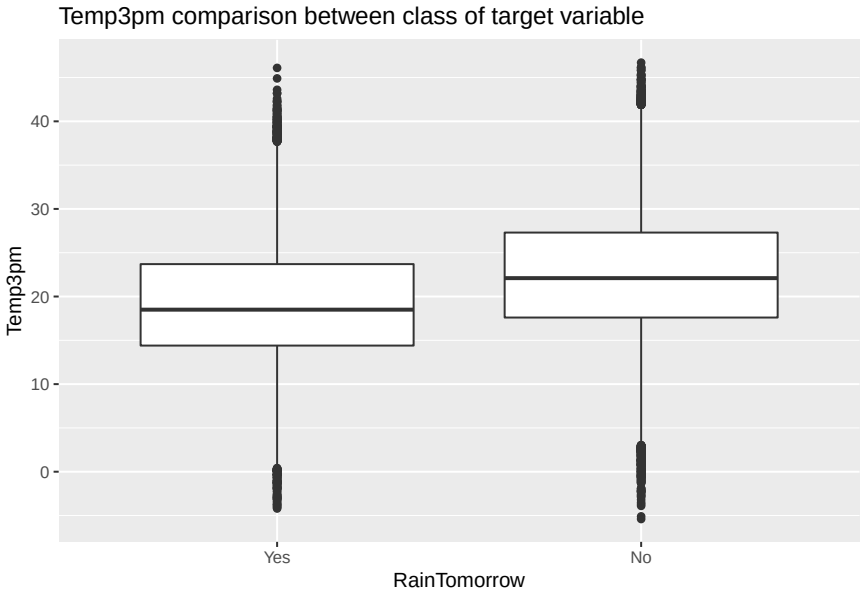


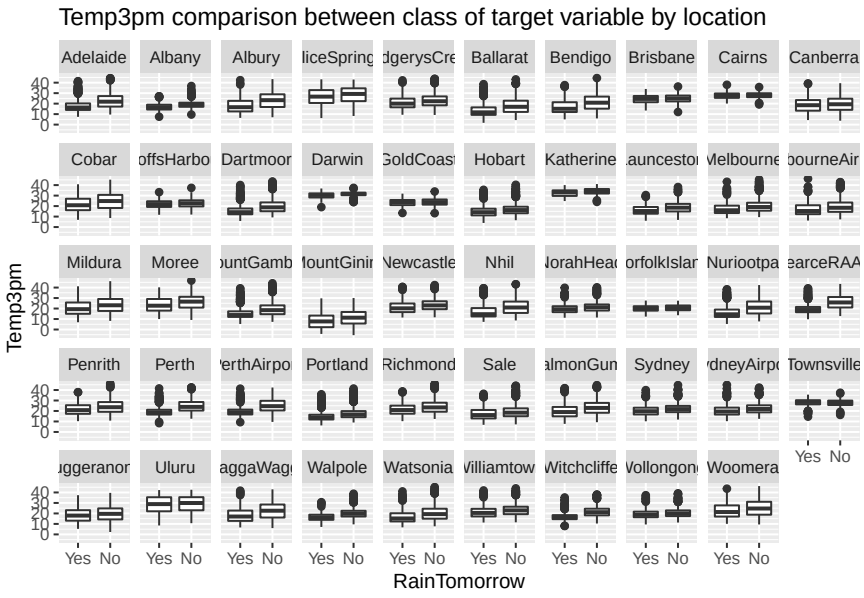
Explore Temp9am There is not a clear difference in mean of Temp9am when compare between 2 class of RainTomorrow. But when classify deeper into each location, there are some distinct relationship such as in Albany, Dartmoor, Ballarat and Walpole, mean Temp9am is lower if tomorrow is raining .



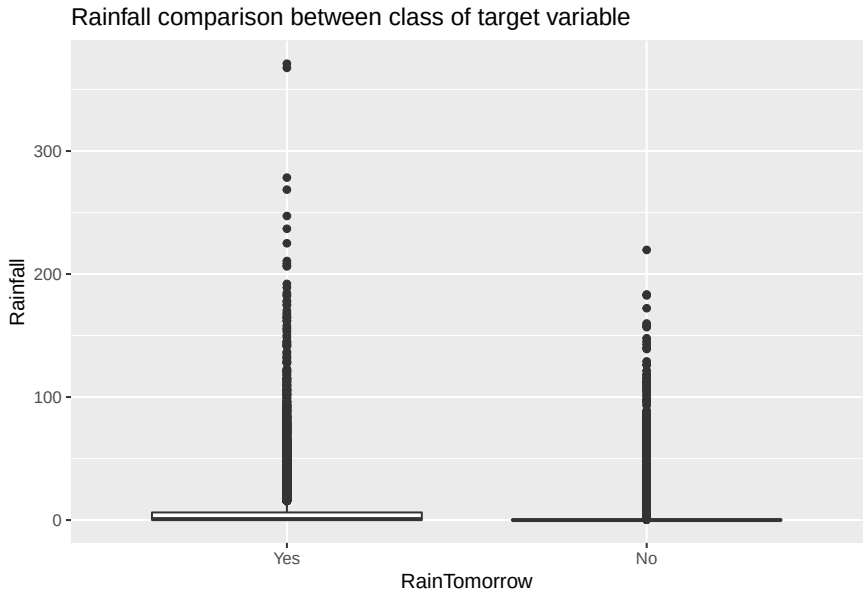


Explore Temp3pm There is quite significant difference in mean of Temp3pm when compare between 2 class of RainTomorrow.





Explore Rainfall Rainfall variable is full of outlier since when it is no rain the data will be close to zero but when it rain the data can goes up to 200. With is kind of data, the visualization may not shows much of the information but i decide to keep this variable because it is the actual data with the extreme difference.



Remark On closer inspection, we can see that the Rainfall, Evaporation, WindSpeed9am and WindSpeed3pm columns may contain some outliers. But it is not really a bad data outlier, it is actually a true data which may help in classification so it is decided not to filter it out.

Modeling approach

Since the current predictor is not very correlated and not very good separation for target variable, The final approach is to use the ensemble classification algorithm. Both bagging and boosting algorithm will be used in this project to compare the performance. The logistic regression and normalized logistic regression with be used as the benchmark then random forest and extreme gradient boost will be used as the final model.

The australian rain data is separated into 80% training set and 20% test set to fit model and test the algorithm. The ROC will be used for evaluation since the target is imbalance date and the objective is focused on predicting both positive condition and negative condition.

Train Test Split

We will divide the dataset into training (80%) and test (20%) sets respectively to train the rainfall prediction model:

```
set.seed(1, sample.kind = 'Rounding')
```

```
TrainIndex <- createDataPartition(ausrain$RainTomorrow, times = 1, p = 0.80, list = FALSE)
Trainset <- ausrain[TrainIndex,]
Testset <- ausrain[-TrainIndex,]
```

Model Improvement and Evaluation Method

Several techniques are employed to improve the model performance. k-fold cross validation and hyperparameter optimization using GridSearchCV are used in this project. Train control parameter is set.

The ROC will be used for evaluation since the target is imbalance date and the objective is focused on predicting both positive condition and negative condition. Twoclasssummary method is set in train control in order to get the probability result and ROC_AUC

```
TrainCon <- trainControl(method = "cv",
                         number = 5,
                         search = 'grid',
                         summaryFunction = twoClassSummary,
                         classProbs = TRUE,
                         savePredictions = TRUE)
```

The importance function is manually define to show to importance of each variable without separated into every factor level as varimp function did.

```
## Initiate variable importance function.
importance <- function(a){
  df <- as.data.frame(a)
  df <- df %>% mutate(name = rownames(df))
  df1 <- df %>%
    mutate(name = ifelse(str_detect(df$name, "^Location"), 'Location', name),
           name = ifelse(str_detect(df$name, "^WindDir3pm"), 'WindDir3pm', name),
           name = ifelse(str_detect(df$name, "^WindDir9am"), 'WindDir9am', name),
           name = ifelse(str_detect(df$name, "^Month"), 'Month', name),
           name = ifelse(str_detect(df$name, "^RainToday"), 'RainToday', name))
  df1 <- df1 %>% group_by(name) %>%
    summarize(importance = sum(Overall)) %>%
    arrange(desc(importance))
  return(df1)
}
```

Training Rainfall Prediction Model with Different Models

Logistic Regression

Normal logistic regression is used as base line model to see the prediction performance of the variable.

```
# GLM
fitGLM <- train(RainTomorrow ~ .,
                data = Trainset %>% select(-Date),
                method = "glm",
                trControl = TrainCon)
```

```
fitGLM
```

```
## Generalized Linear Model
##
## 101627 samples
##     14 predictor
```

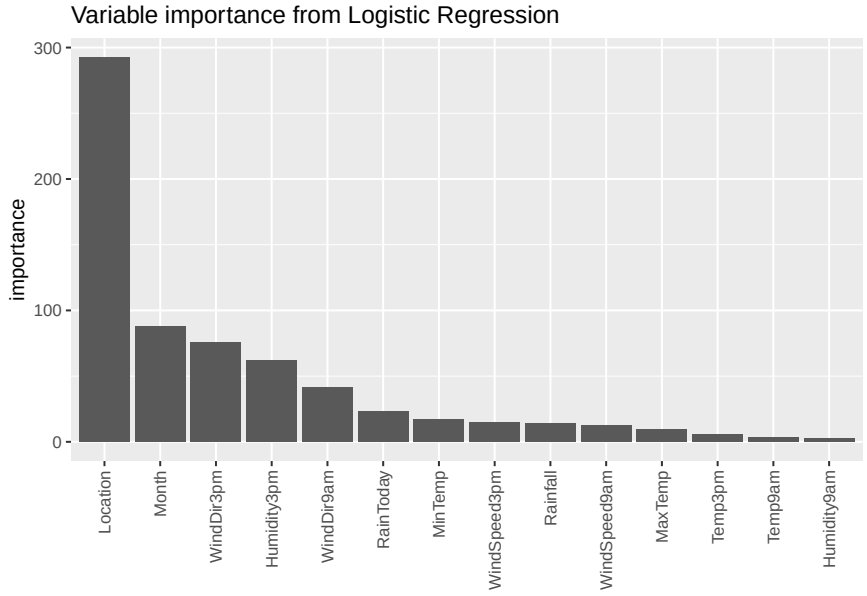
```
##      2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 81302, 81302, 81301, 81302, 81301
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8420155  0.4582413  0.9469726
```

```
confusionMatrix(fitGLM, norm = 'none')
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   Yes    No
##        Yes 10370  4189
##        No  12260 74808
##
##  Accuracy (average) : 0.8381
```

Even the accuracy is high but sensitivity is quite low since the data set is imbalance. So, twoclasssummary and ROC_AUC is used because of this imbalance issue.

```
# GLM varImp
varImpGLM <- varImp(fitGLM, scale = FALSE)
importance(varImpGLM$importance) %>%
  ggplot(aes(x = reorder(name, -importance), y = importance)) +
  geom_col() +
  scale_x_discrete(guide = guide_axis(angle = 90)) +
  labs(title = 'Variable importance from Logistic Regression') +
  xlab(NULL)
```



```r
# GLM testset prediction
predGLM <- predict(fitGLM, Testset)
postResample(pred = predGLM, obs = Testset$RainTomorrow)
```

```
##  Accuracy     Kappa
## 0.8414548 0.4766472
```

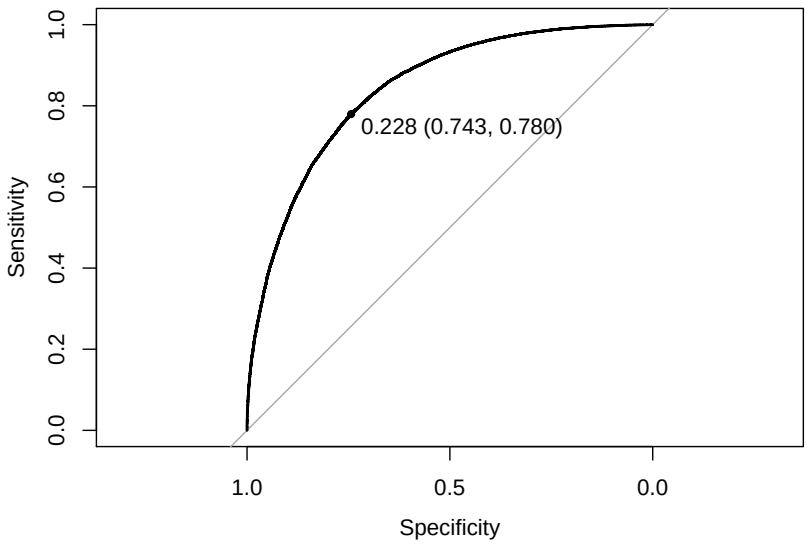
```r
confusionMatrix(data = predGLM, reference = Testset$RainTomorrow, positive = 'Yes')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  2653  1024
##        No   3004 18725
##
##                Accuracy : 0.8415
##                  95% CI : (0.8369, 0.8459)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4766
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4690
##             Specificity : 0.9481
##          Pos Pred Value : 0.7215
##          Neg Pred Value : 0.8618
##              Prevalence : 0.2227
##          Detection Rate : 0.1044
##    Detection Prevalence : 0.1447
##       Balanced Accuracy : 0.7086
##
##        'Positive' Class : Yes
##
```

It is confirmed with test set that the result of model accuracy is high but the sensitivity is very low because the negative class is so much more than positive class.

The ROC_AUC is used to evaluate the model performance instead of accuracy.

```r
# GLM ROC trainset prediction
roctrainGLM <- roc(response = fitGLM$pred[,'obs'],
                   predictor = fitGLM$pred[,'Yes'])
plot(roctrainGLM, print.thres = 'best')
```




```
auctrainGLM <- auc(roctrainGLM)
print(auctrainGLM)
```

```
## Area under the curve: 0.842
```

```
thresholdGLM = coords(roctrainGLM, "best")[1,'threshold']
print(thresholdGLM)
```

```
## [1] 0.2282991
```

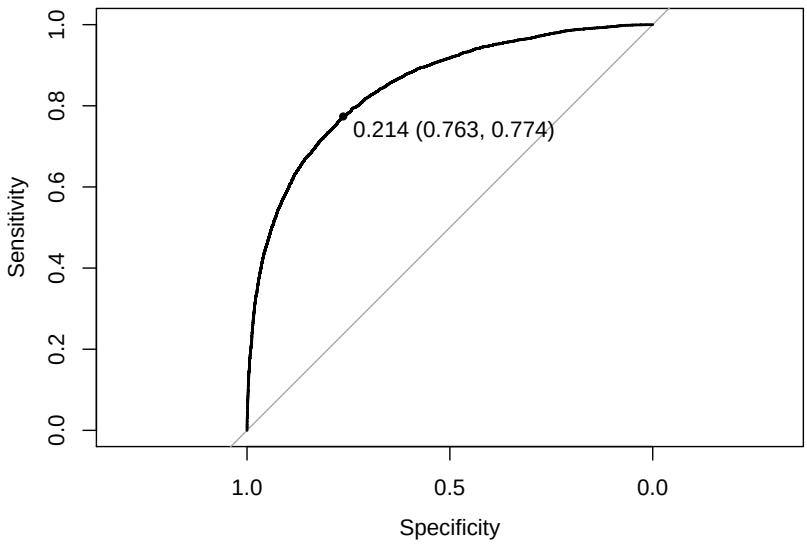
From the ROC above, the best threshold to get the best sensitivity while still high specificity is 0.2282991.

To confirm the performance, the threshold is selected and used to predict the testset again. Now, the 'type' argument is set to 'prob' to get the probability prediction of both Yes and No class. If the probability of positive class is higher than the selected threshold, the prediction will be Yes.

```
# GLM ROC testset prediction
probtestGLM <- predict(fitGLM, Testset, type = 'prob')
head(probtestGLM)
```

```
##           Yes        No
## 2  0.01889593 0.9811041
## 3  0.05432972 0.9456703
## 7  0.03202641 0.9679736
## 11 0.01665788 0.9833421
## 25 0.07170695 0.9282931
## 30 0.01386012 0.9861399
```

```
roctestGLM <- roc(response = Testset$RainTomorrow,
                  predictor = probtestGLM[, "Yes"],
                  levels = rev(levels(Testset$RainTomorrow)))
plot(roctestGLM, print.thres = 'best')
```




```
auc(roctestGLM)
```

```
## Area under the curve: 0.8494
```

ROC for testset is shown above. The best threshold is quite similar to the selected threshold from trainset. This shows that the model is fitted quite well to both trainset and testset.

```
predtestGLM <- as.factor(ifelse(probtestGLM["Yes"] >= thresholdGLM, 'Yes', 'No'))
confMatGLM <- confusionMatrix(data = predtestGLM, reference = Testset$RainTomorrow, positive = 'Yes')
print(confMatGLM)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  4269  4359
##        No   1388 15390
##
##                Accuracy : 0.7738
##                  95% CI : (0.7686, 0.7789)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : 0.9136
##
##                   Kappa : 0.4497
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7546
##             Specificity : 0.7793
##          Pos Pred Value : 0.4948
##          Neg Pred Value : 0.9173
##              Prevalence : 0.2227
```

```
##          Detection Rate : 0.1680
##    Detection Prevalence : 0.3396
##       Balanced Accuracy : 0.7670
##
##        'Positive' Class : Yes
##
```

The confusion matrix above shows that when compare with standard evaluation, when select the best ROC threshold, the model can predict more positive class which is good for weather forecast even the overall accuracy is lower.

```
auctestGLM <- auc(roctestGLM)
print(auctestGLM)
```

```
## Area under the curve: 0.8494
```

The final AUC is 0.849432 from logistic regression model.

```
result <- data_frame(Model = 'Logistic Regression',
                     TestAUC = auctestGLM,
                     TrainAUC = auctrainGLM,
                     Sensitivity = confMatGLM$byClass['Sensitivity'],
                     Specificity = confMatGLM$byClass['Specificity'],
                     Accuracy = confMatGLM$overall['Accuracy'])
result %>% knitr::kable()
```

Model	TestAUC	TrainAUC	Sensitivity	Specificity	Accuracy
Logistic Regression	0.849432	0.8420291	0.7546403	0.77928	0.7737936

The rest of the model in this report will be run in the same pattern. The final performance and result will be shown and discuss in the Result section.

Regularized Logistic Regression

Next, try to do regularized logistic regression to see is there any preformance improvement. First is to initiate the tuning grid with alpha to (0 and 1) and sequence of lambda from 0 to 1 by 0.2 each step.

```
### GLMNET
tuneGridGLMNET <- expand.grid(alpha = c(seq(0,1,0.2)),
                          lambda = c(seq(0,1,0.2)))
```

Fit the model with specified tuning grid and train control.

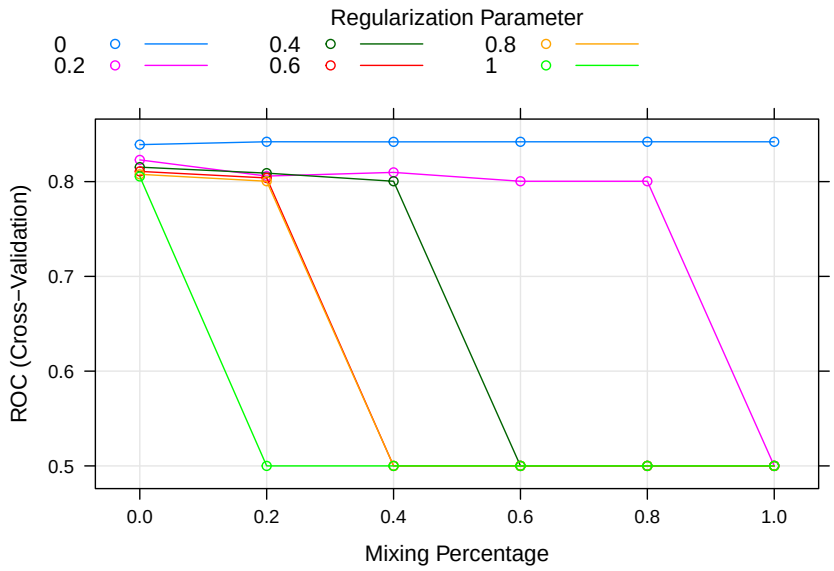
```
fitGLMNET <- train(RainTomorrow ~ .,
                 data = Trainset %>% select(-Date),
                 method = "glmnet",
                 trControl = TrainCon,
                 tuneGrid = tuneGridGLMNET,
                 family = 'binomial')
```

Best tune model hyperparameter is shown below

```
knitr::kable(fitGLMNET$bestTune)
```

	alpha	lambda
7	0.2	0

```
plot(fitGLMNET)
```



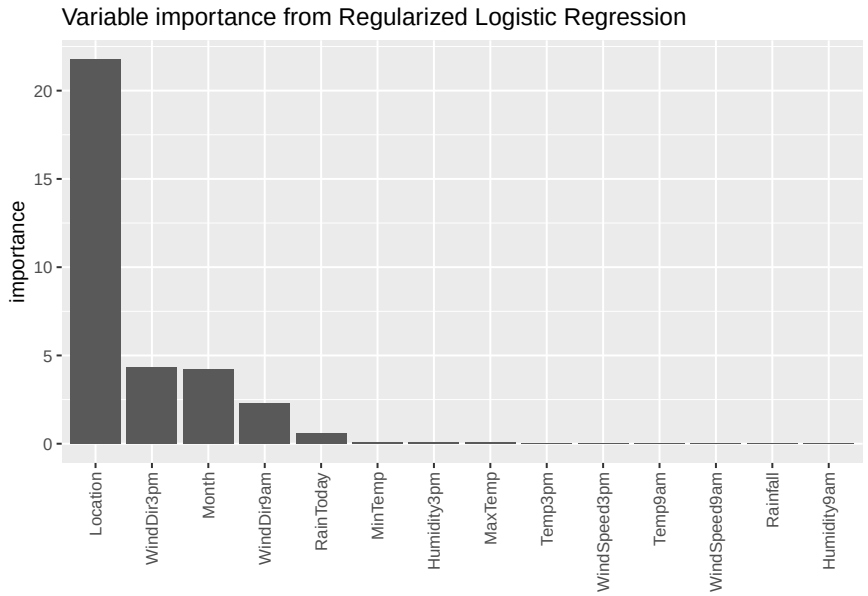

```
confusionMatrix(fitGLMNET, norm = 'none')
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##            Reference
## Prediction   Yes    No
##        Yes 10312  4130
##        No  12318 74867
##
##  Accuracy (average) : 0.8382
```

Similar to normal logistic regression, the accuracy is high but sensitivity is quite low since the data set is imbalance. So, twoclasssummary and ROC_AUC is used because of this imbalance issue.

```
# GLMNET varImp
varImpGLMNET <- varImp(fitGLMNET, scale = FALSE)
importance(varImpGLMNET$importance) %>%
  ggplot(aes(x = reorder(name, -importance), y = importance)) +
  geom_col() +
```

```
  scale_x_discrete(guide = guide_axis(angle = 90)) +
  labs(title = 'Variable importance from Regularized Logistic Regression') +
  xlab(NULL)
```




```
# GLMNET testset prediction
predGLMNET <- predict(fitGLMNET, Testset)
postResample(pred = predGLMNET, obs = Testset$RainTomorrow)
```

```
##  Accuracy     Kappa
## 0.8412186 0.4751111
```

```
confusionMatrix(data = predGLMNET, reference = Testset$RainTomorrow, positive = 'Yes')
```

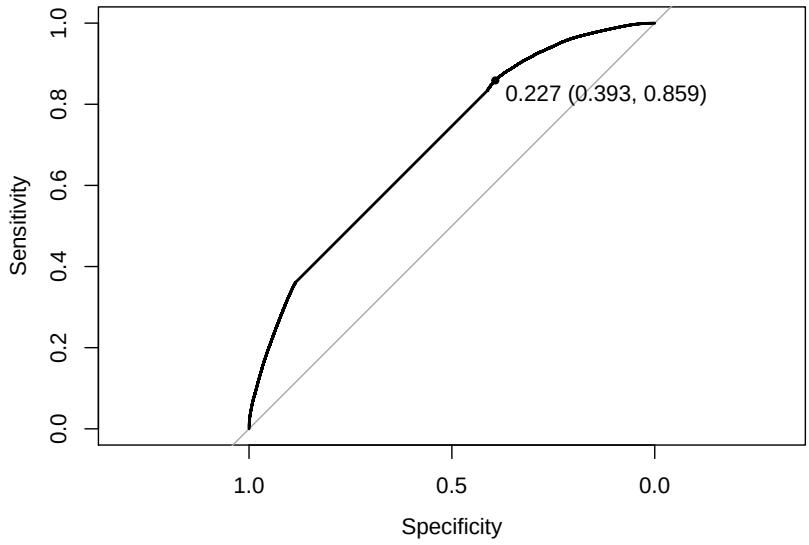
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  2640  1017
##        No   3017 18732
##
##                Accuracy : 0.8412
##                  95% CI : (0.8367, 0.8457)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4751
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4667
##             Specificity : 0.9485
##          Pos Pred Value : 0.7219
```

```
##          Neg Pred Value : 0.8613
##              Prevalence : 0.2227
##          Detection Rate : 0.1039
##    Detection Prevalence : 0.1439
##       Balanced Accuracy : 0.7076
##
##        'Positive' Class : Yes
##
```

It is confirmed with test set that the sensitivity is very low because the negative class is so much more than positive class.

The ROC_AUC is used to evaluate the model performance instead of accuracy.

```
# GLMNET ROC
# GLMNET ROC trainset prediction
roctrainGLMNET <- roc(response = fitGLMNET$pred[,'obs'],
                      predictor = fitGLMNET$pred[,'Yes'])
plot(roctrainGLMNET, print.thres = 'best')
```




```
auctrainGLMNET <- auc(roctrainGLM)
print(auctrainGLMNET)
```

```
## Area under the curve: 0.842
```

```
thresholdGLMNET = coords(roctrainGLMNET, "best")[1,'threshold']
print(thresholdGLMNET)
```

```
## [1] 0.2269947
```

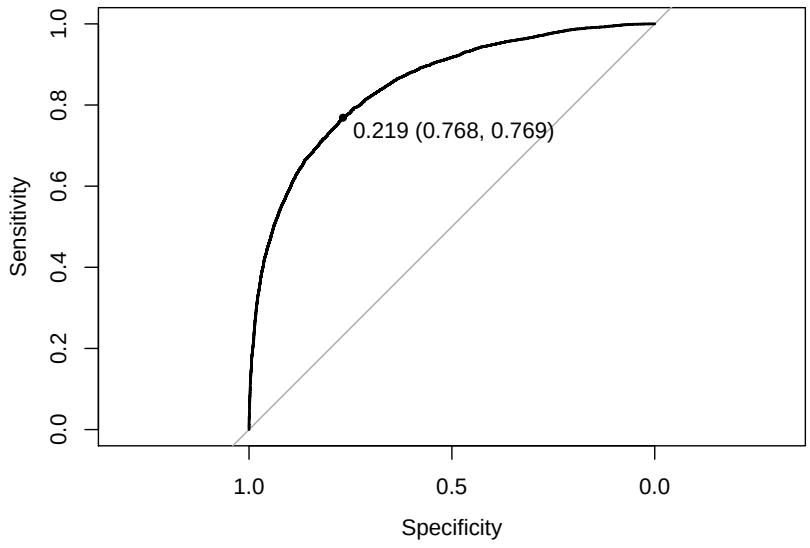
From the ROC above, the best threshold to get the best sensitivity while still high specificity is 0.2269947.

To confirm the performance, the threshold is selected and used to predict the testset again. Now, the 'type' argument is set to 'prob' to get the probability prediction of both Yes and No class. If the probability of positive class is higher than the selected threshold, the prediction will be Yes.

```
# GLMNET ROC testset prediction
probtestGLMNET <- predict(fitGLMNET, Testset, type = 'prob')
head(probtestGLMNET)
```

```
##          Yes        No
## 1 0.01970810 0.9802919
## 2 0.05548347 0.9445165
## 3 0.03244007 0.9675599
## 4 0.01661062 0.9833894
## 5 0.07240727 0.9275927
## 6 0.01428799 0.9857120
```

```
roctestGLMNET <- roc(response = Testset$RainTomorrow,
                     predictor = probtestGLMNET[, "Yes"],
                     levels = rev(levels(Testset$RainTomorrow)))
plot(roctestGLMNET, print.thres = 'best')
```



ROC for testset is shown above. The best threshold is quite similar to the selected threshold from trainset. This shows that the model is fitted quite well to both trainset and testset.

```
predtestGLMNET <- as.factor(ifelse(probtestGLMNET["Yes"] >= thresholdGLMNET, 'Yes', 'No'))
confMatGLMNET <- confusionMatrix(data = predtestGLMNET, reference = Testset$RainTomorrow, positive = 'Y
print(confMatGLMNET)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
```

```
##        Yes  4286  4394
##        No   1371 15355
##
##               Accuracy : 0.7731
##                 95% CI : (0.7679, 0.7782)
##    No Information Rate : 0.7773
##    P-Value [Acc > NIR] : 0.9489
##
##                  Kappa : 0.4495
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7576
##             Specificity : 0.7775
##          Pos Pred Value : 0.4938
##          Neg Pred Value : 0.9180
##              Prevalence : 0.2227
##          Detection Rate : 0.1687
##    Detection Prevalence : 0.3417
##       Balanced Accuracy : 0.7676
##
##        'Positive' Class : Yes
##
```

The confusion matrix above shows that when compare with standard evaluation, when select the best ROC threshold, the model can predict more positive class which is good for weather forecast even the overall accuracy is lower.

```
auctestGLMNET <- auc(roctestGLMNET)
print(auctestGLMNET)
```

```
## Area under the curve: 0.8494
```

The final AUC is 0.849428 from regularized logistic regression model.

```
result <- rbind(result, data.frame(Model = 'Regularized Logistic Regression',
                                   TestAUC = auctestGLMNET,
                                   TrainAUC = auctrainGLMNET,
                                   Sensitivity = confMatGLMNET$byClass['Sensitivity'],
                                   Specificity = confMatGLMNET$byClass['Specificity'],
                                   Accuracy = confMatGLMNET$overall['Accuracy']))
rownames(result) <- NULL
result %>% knitr::kable()
```

Model	TestAUC	TrainAUC	Sensitivity	Specificity	Accuracy
Logistic Regression	0.849432	0.8420291	0.7546403	0.7792800	0.7737936
Regularized Logistic Regression	0.849428	0.8420291	0.7576454	0.7775077	0.7730851

Random Forest (Bagging)

Next, try to use bagging algorithm in this case random forest to see is there any preformance improvement use method Rborist. The tuning parameter, predFixed, is set to 2, 3, and 4. The number of tree is set to 500.

```
tuneGridRF <- expand.grid(minNode = c(1,5), predFixed = c(2,3,4))
```

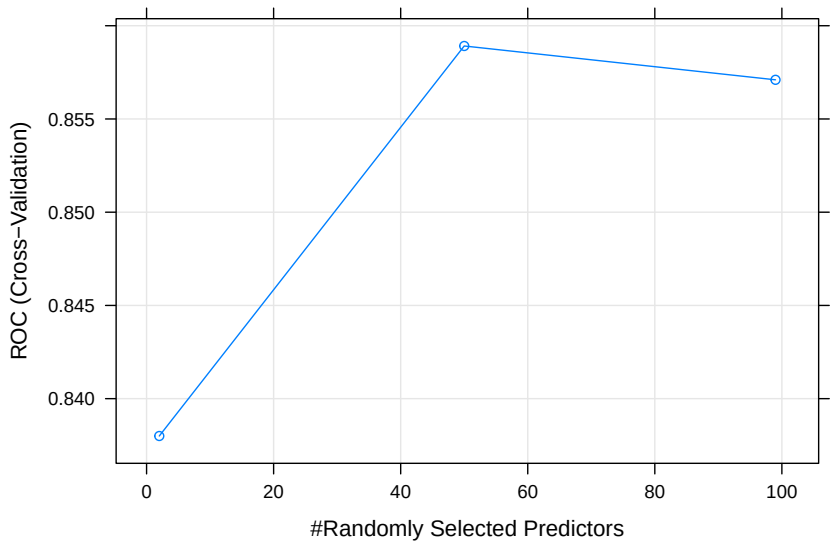
```
# Random Forest
fitRF <- train(RainTomorrow ~ .,
               data = Trainset %>% select(-Date),
               method = "Rborist",
               importance=TRUE,
               nTree = 500,
               tunegrid = tuneGridRF,
               trControl = TrainCon)
```

Best tune model hyperparameter is shown below.

```
knitr::kable(fitRF$bestTune)
```

	predFixed	minNode
2	50	2

```
plot(fitRF)
```



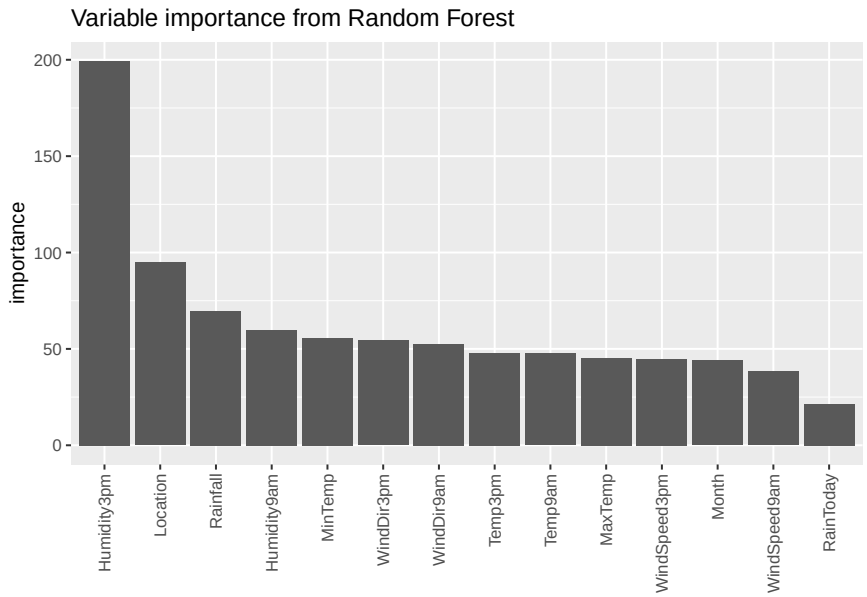

```
confusionMatrix(fitRF, norm = 'none')
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
```

```
## (entries are un-normalized aggregated counts)
##
##            Reference
## Prediction   Yes    No
##        Yes 10515  3723
##        No  12115 75274
##
##  Accuracy (average) : 0.8442
```

With random forest algorithm, the accuracy is still high with low sensitivity from imbalance the target variable. So, twoclasssummary and ROC_AUC is used because of this imbalance issue.

```
# RF varImp
varImpRF <- varImp(fitRF, scale = FALSE)
importance(varImpRF$importance) %>%
  ggplot(aes(x = reorder(name, -importance), y = importance)) +
  geom_col() +
  scale_x_discrete(guide = guide_axis(angle = 90)) +
  labs(title = 'Variable importance from Random Forest') +
  xlab(NULL)
```




```
# RF testset prediction
predRF <- predict(fitRF, Testset)
postResample(pred = predRF, obs = Testset$RainTomorrow)
```

```
##  Accuracy     Kappa
## 0.8475163 0.4959297
```

```
confusionMatrix(data = predRF, reference = Testset$RainTomorrow, positive = 'Yes')
```

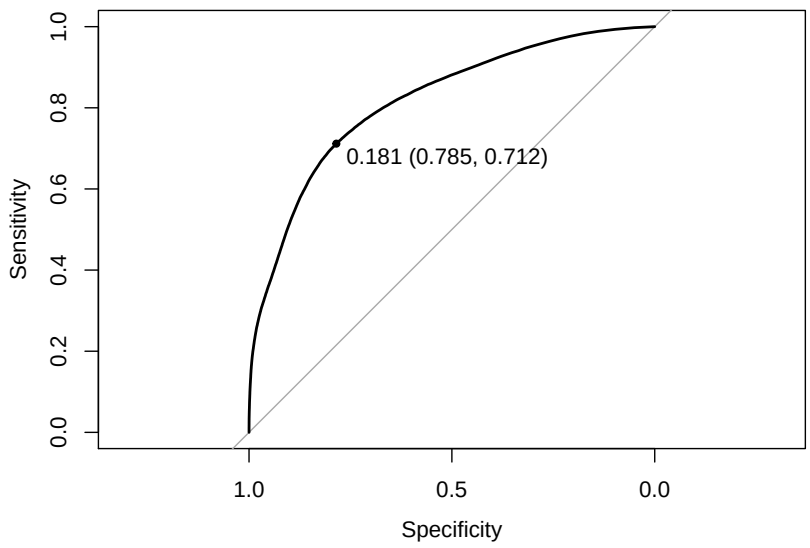
```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   Yes    No
##        Yes  2720   937
##        No   2937 18812
##
##                Accuracy : 0.8475
##                  95% CI : (0.843, 0.8519)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4959
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4808
##             Specificity : 0.9526
##          Pos Pred Value : 0.7438
##          Neg Pred Value : 0.8650
##              Prevalence : 0.2227
##          Detection Rate : 0.1071
##    Detection Prevalence : 0.1439
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : Yes
##
```

It is confirmed with test set that the sensitivity is very low because the negative class is so much more than positive class.

The ROC_AUC is used to evaluate the model performance instead of accuracy.

```
# RF ROC trainset prediction
roctrainRF <- roc(response = fitRF$pred[,'obs'],
                  predictor = fitRF$pred[,'Yes'],
                  levels = rev(levels(Trainset$RainTomorrow)))
plot(roctrainRF, print.thres = 'best')
```




```
auctrainRF <- auc(roctrainRF)
print(auctrainRF)
```

```
## Area under the curve: 0.817
```

```
thresholdRF = coords(roctrainRF, "best")[1,'threshold']
print(thresholdRF)
```

```
## [1] 0.181
```

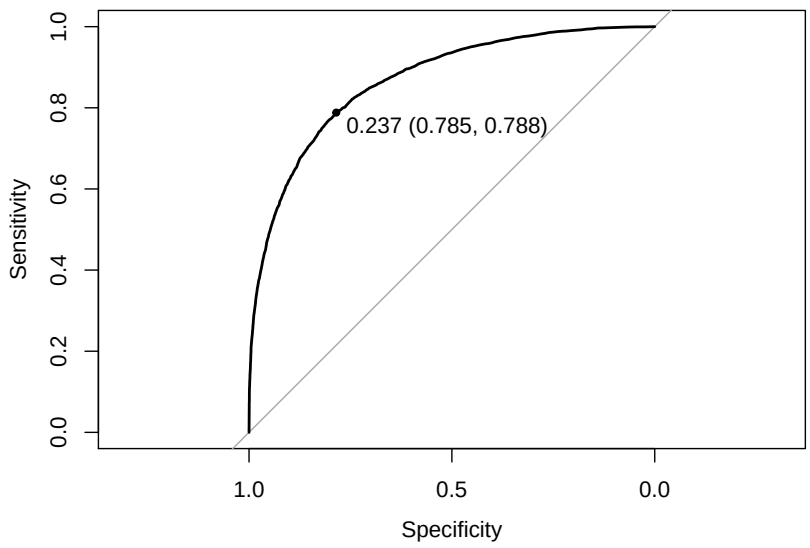
From the ROC above, the best threshold to get the best sensitivity while still high specificity is 0.181.

To confirm the performance, the threshold is selected and used to predict the testset again. Now, the 'type' argument is set to 'prob' to get the probability prediction of both Yes and No class. If the probability of positive class is higher than the selected threshold, the prediction will be Yes.

```
# RF ROC testset prediction
probtestRF <- predict(fitRF, Testset, type = 'prob')
head(probtestRF)
```

```
##      Yes    No
## 2  0.142 0.858
## 3  0.024 0.976
## 7  0.012 0.988
## 11 0.026 0.974
## 25 0.074 0.926
## 30 0.008 0.992
```

```
roctestRF <- roc(response = Testset$RainTomorrow,
                 predictor = probtestRF[, "Yes"],
                 levels = rev(levels(Testset$RainTomorrow)))
plot(roctestRF, print.thres = 'best')
```



ROC for testset is shown above. The best threshold is quite similar to the selected threshold from trainset. This shows that the model is fitted quite well to both trainset and testset.

```
predtestRF <- as.factor(ifelse(probtestRF["Yes"] >= thresholdRF, 'Yes', 'No'))
confMatRF <- confusionMatrix(data = predtestRF, reference = Testset$RainTomorrow, positive = 'Yes')
print(confMatRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  4783  5776
##        No    874 13973
##
##                Accuracy : 0.7383
##                  95% CI : (0.7328, 0.7436)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4224
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8455
##             Specificity : 0.7075
##          Pos Pred Value : 0.4530
##          Neg Pred Value : 0.9411
##              Prevalence : 0.2227
##          Detection Rate : 0.1883
##    Detection Prevalence : 0.4156
##       Balanced Accuracy : 0.7765
##
##        'Positive' Class : Yes
##
```

The confusion matrix above shows that when compare with standard evaluation, when select the best ROC threshold, the model can predict more positive class which is good for weather forecast even the overall accuracy is lower.

```
auctestRF <- auc(roctestRF)
print(auctestRF)
```

```
## Area under the curve: 0.8674
```

The final AUC is 0.8674408 from regularized logistic regression model.

```
result <- rbind(result, data.frame(Model = 'Random Forest',
                                   TestAUC = auctestRF,
                                   TrainAUC = auctrainRF,
                                   Sensitivity = confMatRF$byClass['Sensitivity'],
                                   Specificity = confMatRF$byClass['Specificity'],
                                   Accuracy = confMatRF$overall['Accuracy']))
rownames(result) <- NULL
result %>% knitr::kable()
```

Model	TestAUC	TrainAUC	Sensitivity	Specificity	Accuracy
Logistic Regression	0.8494320	0.8420291	0.7546403	0.7792800	0.7737936
Regularized Logistic Regression	0.8494280	0.8420291	0.7576454	0.7775077	0.7730851
Random Forest	0.8674408	0.8170209	0.8455011	0.7075295	0.7382508

Extreme Gradient Boosting (Boosting)

Last, boosting algorithm in this case extreme gradient boosting is selected to see is there any preformance improvement.

the xgbTree method has many hyperparameter for tuning. Only eta, gamma and colsample_bytree are tune in this project. Nround is set to 100 and the rest is remain default.

```
# XGBoost
tuneGridXGB <- expand.grid(
  nrounds = 100,
  max_depth = 6,
  eta = c(0.1, 0.3, 0.5),
  gamma = c(0, 10, 30, 50),
  colsample_bytree = c(0.1,0.3,0.5,0.7,1),
  min_child_weight = 1,
  subsample = 1
)
```

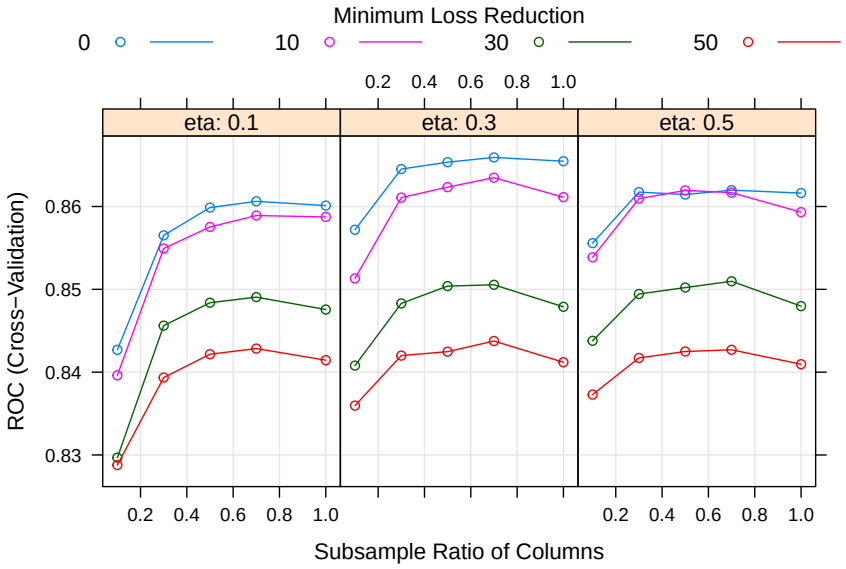
```
fitXGB <- train(RainTomorrow ~ .,
                data = Trainset,
                trControl = TrainCon,
                tuneGrid = tuneGridXGB,
                method = "xgbTree",
                verbose = TRUE
)
```

Best tune model hyperparameter is shown below.

```
knitr::kable(fitXGB$bestTune)
```

	nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight	subsample
24	100	6	0.3	0	0.7	1	1

```
plot(fitXGB)
```



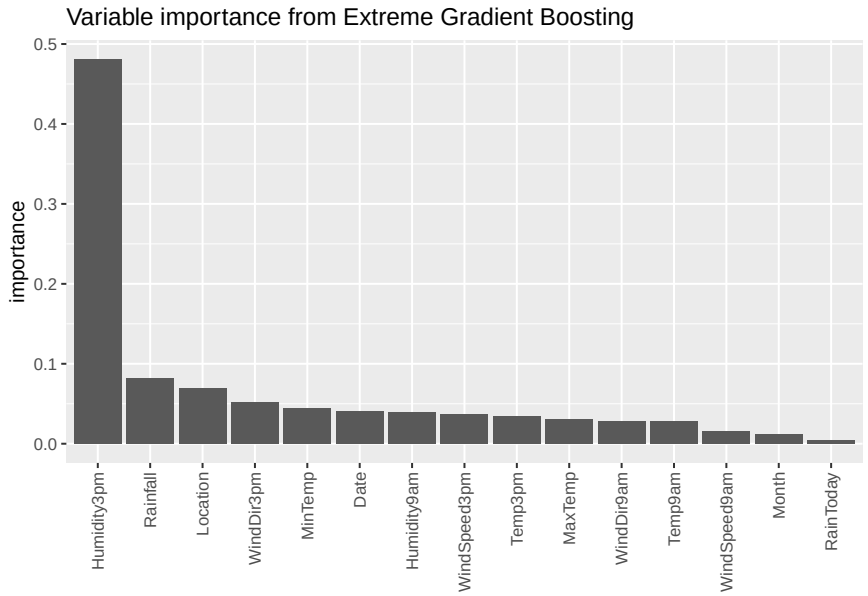

```
confusionMatrix(fitXGB, norm = 'none')
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##            Reference
## Prediction   Yes    No
##        Yes 11078  3850
##        No  11552 75147
##
##  Accuracy (average) : 0.8484
```

With extreme gradient boosting algorithm, the accuracy is also high with low sensitivity from imbalance the target variable. So, twoclasssummary and ROC_AUC is used because of this imbalance issue.

```
# XGB varImp
varImpXGB <- varImp(fitXGB, scale = FALSE)
importance(varImpXGB$importance) %>%
  ggplot(aes(x = reorder(name, -importance), y = importance)) +
  geom_col()+
```

```
  scale_x_discrete(guide = guide_axis(angle = 90)) +
  labs(title = 'Variable importance from Extreme Gradient Boosting') +
  xlab(NULL)
```




```
# XGB testset prediction
predXGB <- predict(fitXGB, Testset)
postResample(pred = predXGB, obs = Testset$RainTomorrow)
```

```
##  Accuracy     Kappa
## 0.8535779 0.5195048
```

```
confusionMatrix(data = predXGB, reference = Testset$RainTomorrow, positive = 'Yes')
```

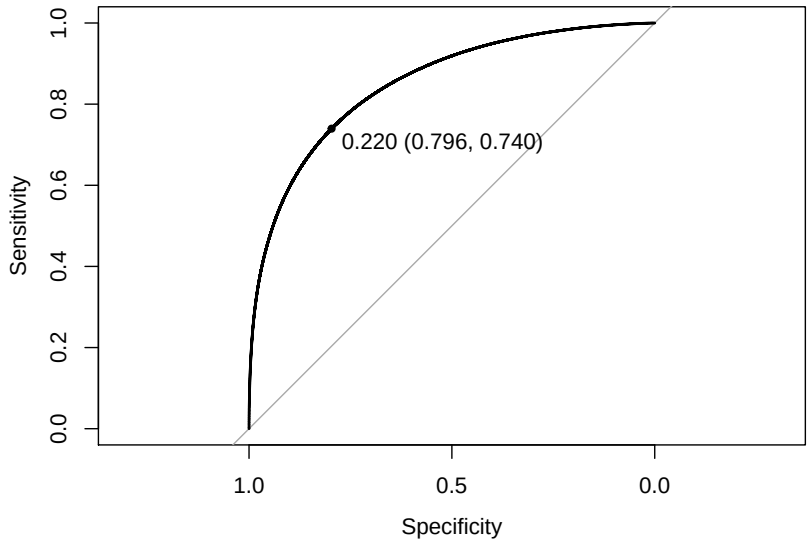
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  2848   911
##        No   2809 18838
##
##                Accuracy : 0.8536
##                  95% CI : (0.8492, 0.8579)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5195
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5034
##             Specificity : 0.9539
##          Pos Pred Value : 0.7576
```

```
##          Neg Pred Value : 0.8702
##              Prevalence : 0.2227
##          Detection Rate : 0.1121
##    Detection Prevalence : 0.1480
##       Balanced Accuracy : 0.7287
##
##        'Positive' Class : Yes
##
```

It is confirmed with test set that the sensitivity is very low because the negative class is so much more than positive class.

The ROC_AUC is used to evaluate the model performance instead of accuracy.

```
## XGB ROC
# XGB ROC trainset prediction
roctrainXGB <- roc(response = fitXGB$pred[,'obs'],
                   predictor = fitXGB$pred[,'Yes'],
                   levels = rev(levels(Trainset$RainTomorrow)))
plot(roctrainXGB, print.thres = 'best')
```




```
auctrainXGB <- auc(roctrainXGB)
print(auctrainXGB)
```

```
## Area under the curve: 0.8507
```

```
thresholdXGB = coords(roctrainXGB, "best")[1,'threshold']
print(thresholdXGB)
```

```
## [1] 0.2199673
```

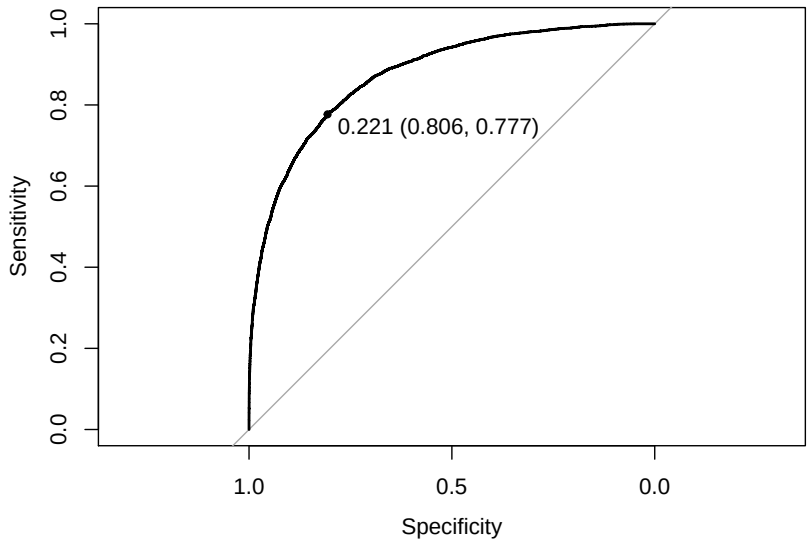
From the ROC above, the best threshold to get the best sensitivity while still high specificity is 0.2199673.

To confirm the performance, the threshold is selected and used to predict the testset again. Now, the 'type' argument is set to 'prob' to get the probability prediction of both Yes and No class. If the probability of positive class is higher than the selected threshold, the prediction will be Yes.

```
# XGB ROC testset prediction
probtestXGB <- predict(fitXGB, Testset, type = 'prob')
head(probtestXGB)
```

```
##           Yes        No
## 1 0.031386763 0.9686132
## 2 0.026120728 0.9738793
## 3 0.018601809 0.9813982
## 4 0.018789582 0.9812104
## 5 0.066558257 0.9334417
## 6 0.007150693 0.9928493
```

```
roctestXGB <- roc(response = Testset$RainTomorrow,
                  predictor = probtestXGB[, "Yes"],
                  levels = rev(levels(Testset$RainTomorrow)))
plot(roctestXGB, print.thres = 'best')
```



ROC for testset is shown above. The best threshold is quite similar to the selected threshold from trainset. This shows that the model is fitted quite well to both trainset and testset.

```
predtestXGB <- as.factor(ifelse(probtestXGB["Yes"] >= thresholdXGB, 'Yes', 'No'))
confMatXGB <- confusionMatrix(data = predtestXGB, reference = Testset$RainTomorrow, positive = 'Yes')
print(confMatXGB)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   Yes    No
##        Yes  4395  3855
##        No   1262 15894
##
##                Accuracy : 0.7986
##                  95% CI : (0.7936, 0.8035)
##     No Information Rate : 0.7773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7769
##             Specificity : 0.8048
##          Pos Pred Value : 0.5327
##          Neg Pred Value : 0.9264
##              Prevalence : 0.2227
##          Detection Rate : 0.1730
##    Detection Prevalence : 0.3247
##       Balanced Accuracy : 0.7909
##
##        'Positive' Class : Yes
##
```

The confusion matrix above shows that when compare with standard evaluation, when select the best ROC threshold, the model can predict more positive class which is good for weather forecast even the overall accuracy is lower.

```
auctestXGB <- auc(roctestXGB)
print(auctestXGB)
```

```
## Area under the curve: 0.875
```

The final AUC is 0.8749632 for Extreme Gradient Boosting algorithm.

```
result <- rbind(result, data.frame(Model = 'Extreme Gradient Boosting',
                                   TestAUC = auctestXGB,
                                   TrainAUC = auctrainXGB,
                                   Sensitivity = confMatXGB$byClass['Sensitivity'],
                                   Specificity = confMatXGB$byClass['Specificity'],
                                   Accuracy = confMatXGB$overall['Accuracy']))
rownames(result) <- NULL
result %>% knitr::kable()
```

Model	TestAUC	TrainAUC	Sensitivity	Specificity	Accuracy
Logistic Regression	0.8494320	0.8420291	0.7546403	0.7792800	0.7737936
Regularized Logistic Regression	0.8494280	0.8420291	0.7576454	0.7775077	0.7730851
Random Forest	0.8674408	0.8170209	0.8455011	0.7075295	0.7382508
Extreme Gradient Boosting	0.8749632	0.8507287	0.7769136	0.8048002	0.7985909

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   3032087  162.0    4739312  253.2         NA   3032087  162.0
## Vcells 275046176 2098.5  473659154 3613.8     102400 275046176 2098.5
```

Results

```
result %>% knitr::kable()
```

Model	TestAUC	TrainAUC	Sensitivity	Specificity	Accuracy
Logistic Regression	0.8494320	0.8420291	0.7546403	0.7792800	0.7737936
Regularized Logistic Regression	0.8494280	0.8420291	0.7576454	0.7775077	0.7730851
Random Forest	0.8674408	0.8170209	0.8455011	0.7075295	0.7382508
Extreme Gradient Boosting	0.8749632	0.8507287	0.7769136	0.8048002	0.7985909

The best model is Extreme Gradient Boosting model. Random forest model also have a very good performance . But when compare between train set and test set performance, random forest model is underfitting. If we select this underfitting model, the testset or future actual data prediction performance will be not reliable. Baseline model logistic regression is the poorest performance. Regularized logistic regression is quite interesting since the performance is very similar to baseline model which will be discussed in conclusion section.

Conclusion

The best performance model is Extreme GRadient Boosting model which is boosting algorithm with the best AUC score.

Ensemble Algorithm

Since the data shows very low significant and correlation with the target variable, RainTomorrow, the ensemble algorithm works better when compare to single decision algorithm, in this case logistic regression.

ROC_AUC Score

When explore the target variable, RainTomorrow, we see a small number of observation that there will be rain tomorrow. Majority of observations is no rain tomorrow. This is a kind of imbalance data which will affect to the our true prediction performance.

Since most of the time it will be no rain tomorrow, even if the model predict all observation to no rain tomorrow we still get high accuracy score. But when explore deeply into the prediction, the performance when predicting rain tomorrow is very low as it shows in low sensitivity score. That is why the ROC_AUC score is used for evaluate and compare the performance of each model, to get the best prediction performance for both positive and negative case while the data is imbalance.

Bagging or Boosting

When focus on ensemble algorithm, both bagging and boosting algorithm is used in this project. Both bagging and boosting algorithm are trying to increasing the overall performance but in different way.

The bagging algorithm, in this project is random forest, in general, is trying to improve overall performance by voting the result from different decision tree. In general, bagging algorithm help generalized the model, reduce the overfitting of training model.

On the other hand, boosting algorithm, in this project is extreme gradient boosting, is trying to improve overall performance by stacking up the previous decision tree with weighting to create next decision tree. It is focusing on reducing error between the prediction and actual data. In some case this may causing overfitting of the train model

From baseline model, logistic regression, there is no sign of overfitting and it is confirmed by result of regularized logistic regression which is no improvement from normal logistic regression. When random forest is fit to the data, we even see a sign of underfitting since the train dataset performance is quite low when compare to its performance on test dataset. But when extreme gradient boosting help reducing bias or error of the prediction, so the best model performance in this project is extreme gradient boosting.

Improvement

Finally, the final performance in this project still not the best since the time and computing performance is limited. There still a lot of room for improvement. Since the original variable correlation is quite low to target variable, to use more data or gather more variable may help improve the overall prediction performance. Deeper data exploration to get more insight of the dataset and it will be useful for more feature engineering. To do more cross validation and adjust more tuning parameter may help reducing the overfitting for boosting algorithm.

This is a project as a part of the ninth and final course, HarvardX PH125.9x - Data Science: Capstone, in HarvardX's multi-part Data Science Professional Certificate series. This project helps me for more understanding of data science workflow and technical method. It is inspired me for dig deep into the data science world. And i hope this report will inspired other who read this report also.