Machine Learning Analysis of Australia Rain

Riley Mault

May 7, 2020

Abstract

This study is intended to explore which factors contribute most to rainfall in different locations throughout Australia. In particular, I am interested in using the factors to predict whether it will rain or not the day after. The following research questions are of interest to me: Could machine learning be used to predict the pattern of rain in Australia? Which attributes affect rainfall the most? How much rainfall is it considered to be raining?

Data Overview

I am using the Rain in Australia Data Set, available through Kaggle (https://www.kaggle.com/jsphyg/weather-dataset-rattle-package). It is a collection of about 10 years of daily weather observations from numerous Australian weather stations. The creator is Joe Young, a Data Scientist from Lexington, South Carolina. The daily observations are available from http://www.bom.gov.au/climate/data. Copyright Commonwealth of Australia 2010, Bureau of Meteorology.

Data Preprocessing

The original data set had 24 attributes and 142,193 observations. After removing missing data and unnecessary predictors to avoid the curse of dimensionality, I was left 112,925 observations with 17 predictors and 1 response variable.

```
## Observations: 112,925
## Variables: 18
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
## $ Location      <fct> Albury, Albury, Albury, Albury, Albury, Albury, Albury,...
## $ MinTemp       <dbl> 13.4, 7.4, 12.9, 9.2, 17.5, 14.6, 14.3, 7.7, 9.7, 13.1,...
## $ MaxTemp       <dbl> 22.9, 25.1, 25.7, 28.0, 32.3, 29.7, 25.0, 26.7, 31.9, 3...
## $ Rainfall      <dbl> 0.6, 0.0, 0.0, 0.0, 1.0, 0.2, 0.0, 0.0, 0.0, 1.4, 0.0, ...
## $ WindGustDir   <fct> W, WNW, WSW, NE, W, WNW, W, W, NNW, W, N, NNE, W, SW, E...
## $ WindGustSpeed <int> 44, 44, 46, 24, 41, 56, 50, 35, 80, 28, 30, 31, 61, 44,...
## $ WindDir9am    <fct> W, NNW, W, SE, ENE, W, SW, SSE, SE, S, SSE, NE, NNW, W,...
## $ WindDir3pm    <fct> WNW, WSW, WSW, E, NW, W, W, W, NW, SSE, ESE, ENE, NNW, ...
## $ WindSpeed9am  <int> 20, 4, 19, 11, 7, 19, 20, 6, 7, 15, 17, 15, 28, 24, 11,...
## $ WindSpeed3pm  <int> 24, 22, 26, 9, 20, 24, 24, 17, 28, 11, 6, 13, 28, 20, 9...
## $ Humidity9am   <int> 71, 44, 38, 45, 82, 55, 49, 48, 42, 58, 48, 89, 76, 65,...
## $ Humidity3pm   <int> 22, 25, 30, 16, 33, 23, 19, 19, 9, 27, 22, 91, 93, 43, ...
## $ Pressure9am   <dbl> 1007.7, 1010.6, 1007.6, 1017.6, 1010.8, 1009.2, 1009.6,...
## $ Pressure3pm   <dbl> 1007.1, 1007.8, 1008.7, 1012.8, 1006.0, 1005.4, 1008.2,...
## $ Temp9am       <dbl> 16.9, 17.2, 21.0, 18.1, 17.8, 20.6, 18.1, 16.3, 18.3, 2...
## $ Temp3pm       <dbl> 21.8, 24.3, 23.2, 26.5, 29.7, 28.9, 24.6, 25.5, 30.2, 2...
## $ RainToday     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0...
## $ RainTomorrow  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0...
```

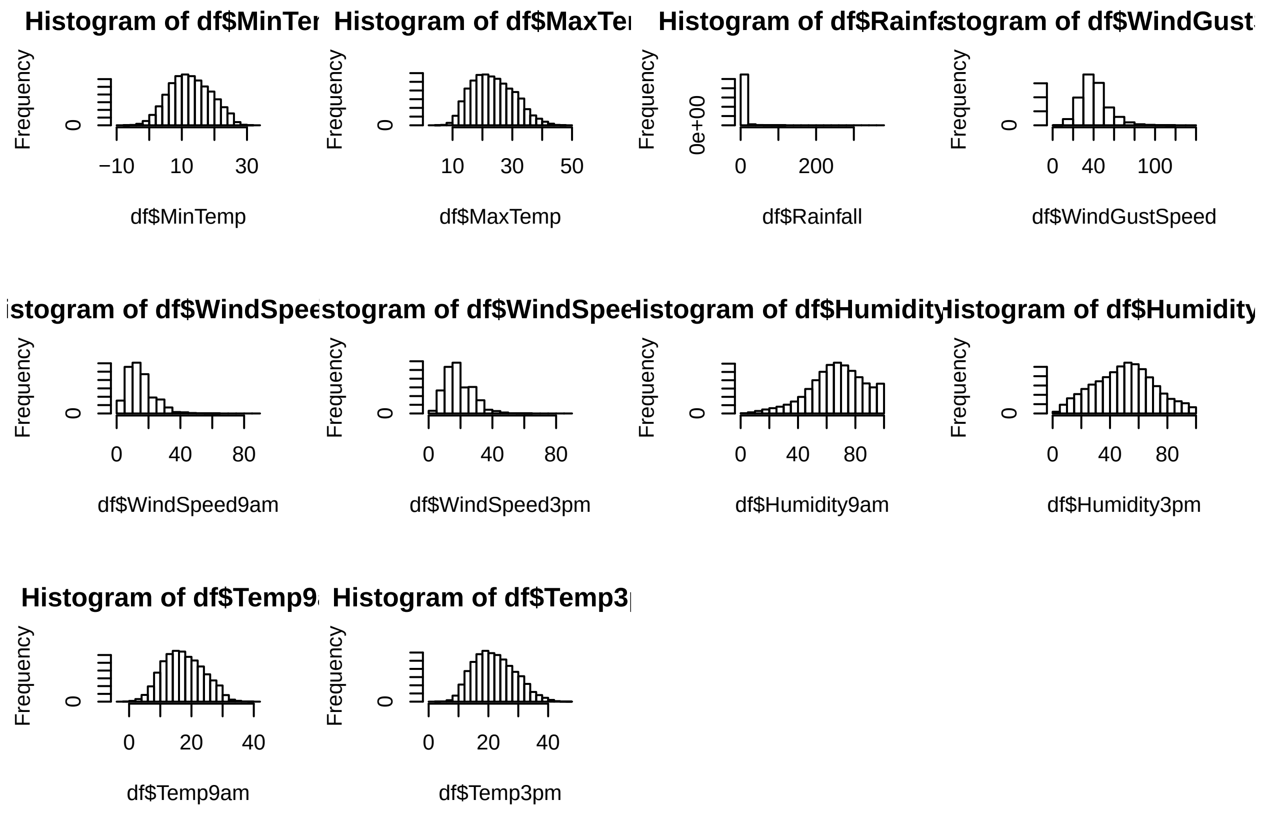
The response variable is **RainTomorrow**. This variable is a binary variable (yes or no if it rained the following day) for the purpose of classification. RainTomorrow was created from a numerical variable RISK_MM that recorded the amount of next day rainfall in mm.

The following are the final covariates used:

Location: The common name of the location of the weather station
MinTemp: The minimum temperature in degrees Celsius
MaxTemp: The maximum temperature in degrees Celsius
Rainfall: The amount of rainfall recorded for the day in mm
WindGustDir: The direction of the strongest wind gust in the 24 hours to midnight
WindGustSpeed: The speed (km/h) of the strongest wind gust in the 24 hours to midnight
WindDir9am: Direction of the wind at 9am
WindDir3pm: Direction of the wind at 3pm
WindSpeed9am: Wind speed (km/hr) averaged over 10 minutes prior to 9am
WindSpeed3pm: Wind speed (km/hr) averaged over 10 minutes prior to 3pm
Humidity9am: Humidity (percent) at 9am
Humidity3pm: Humidity (percent) at 3pm
Pressure9am: Atmospheric pressure (hpa) reduced to mean sea level at 9am
Pressure3pm: Atmospheric pressure (hpa) reduced to mean sea level at 3pm
Temp9am: Temperature (degrees C) at 9am
Temp3pm: Temperature (degrees C) at 3pm
RainToday: Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

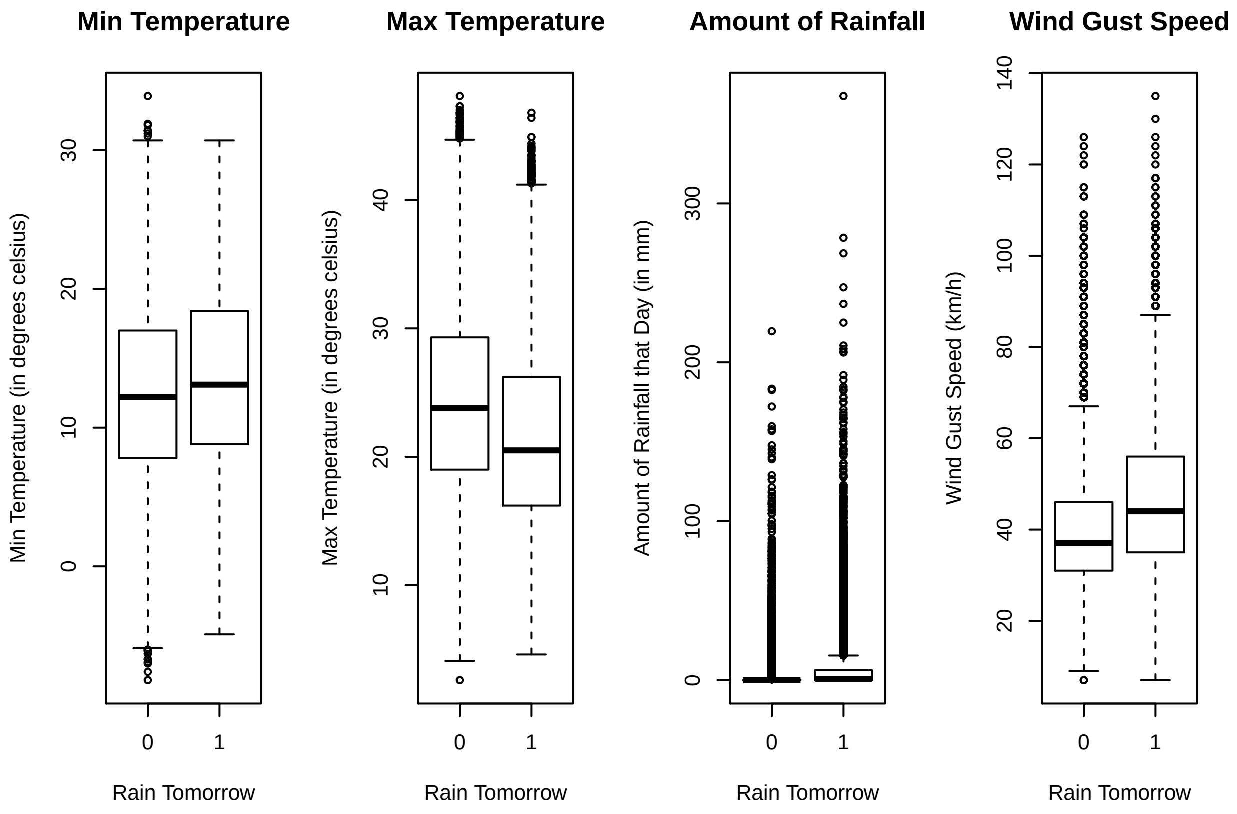
Exploratory Data Analysis

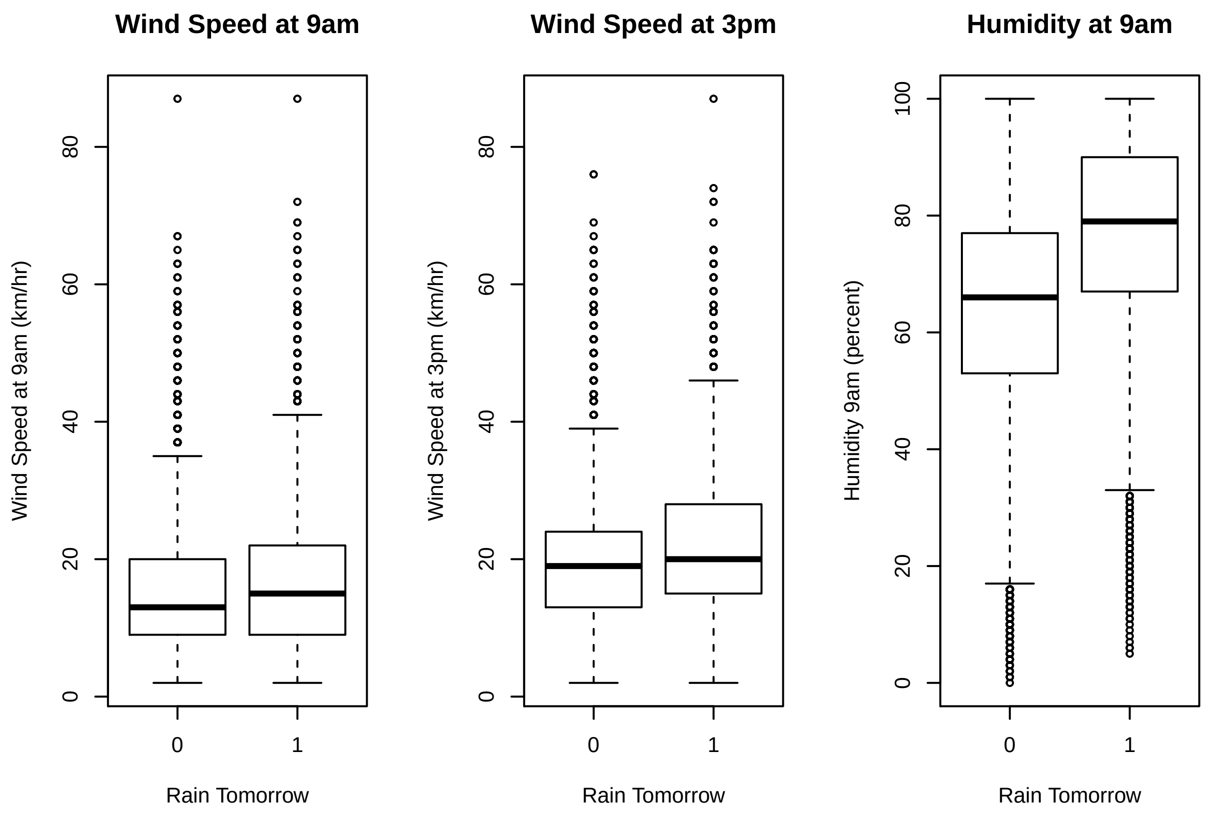
In order to explore and summarize characteristics of each continuous variable, I constructed histograms. Histograms reveal frequencies of predictors to analyze skewness and help in identify outliers.

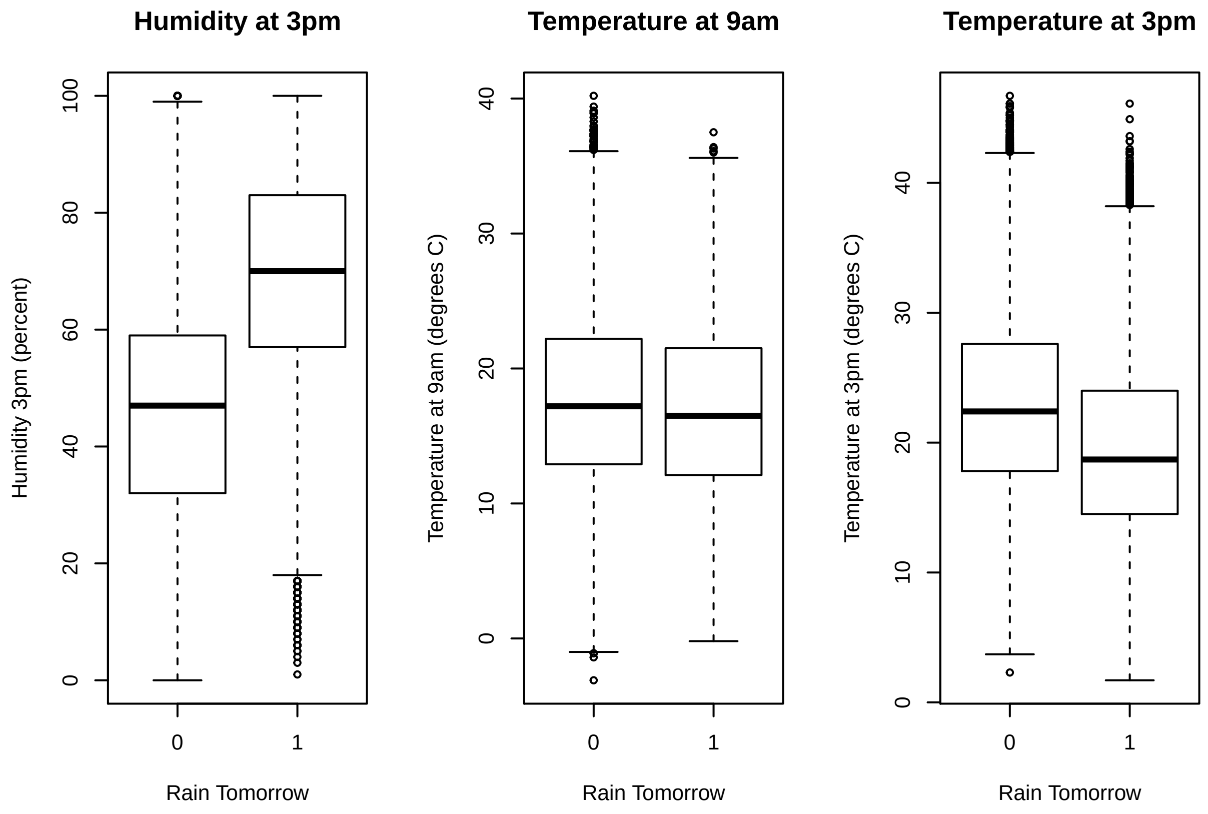


MinTemp, MaxTemp, Humidity3pm, Temp9am, and Temp3pm seem to be normally distributed. The rest of the variables seem to be right-skewed. There don't appear to be any noticeable outliers.

I then constructed boxplots based on the response variable **RainTomorrow**. Boxplots are a useful way to explore comparisons across the response variable in this case. With the comparisons, we can maybe find out if there are some predictors that are more significant than others. They are also useful for identifying outliers that are not shown in histograms.


Min Temperature
Max Temperature
Amount of Rainfall
Wind Gust Speed
Min Temperature (in degrees celsius)
Max Temperature (in degrees celsius)
Amount of Rainfall that Day (in mm)
Wind Gust Speed (km/h)
Rain Tomorrow
0
1


Wind Speed at 9am
Wind Speed at 9am (km/hr)
Rain Tomorrow
0
1
0
20
40
60
80
Wind Speed at 3pm
Wind Speed at 3pm (km/hr)
Rain Tomorrow
0
1
0
20
40
60
80
Humidity at 9am
Humidity 9am (percent)
Rain Tomorrow
0
1
0
20
40
60
80
100



In examining the boxplots, we can see that most predictors do have noticeable differences across the response variable. The two that stand out the most are Humidity at 9am and Humidity at 3pm. The amount of rainfall variable looks to have a lot of outliers, while the rest of the variables don't have many.

Analysis

I performed supervised learning on this data set fitting two different types of models including K-Nearest Neighbors Classification and Logistic Regression with Lasso regularization. To evaluate performance, I compared misclassification error rates on the test sets.

K-Nearest Neighbors Classification

My goal with K-Nearest Neighbors analysis was to investigate the relationship between RainTomorrow and all continuous numeric variables. This is due to KNN not being able to account for categorical features, so I removed them before proceeding with model fitting.

The training set contained 75% of our data, or 84,693 observations, and the test set consisted of the remaining 28,232 observations. I next trained the KNN classifier with 2, 5, and 10 neighbors and compared their test error rates. I choose these values of K, because they are common and tend to optimize bias-variance tradeoff.

The training error for k=2 was 0.1008, for k=5 it was 0.1201, and for k=10 it was 0.1351. Once we trained the KNN models, we fit the test sets to each model and calculated the test error rates. This is what we used to quantify the success of our models because it shows if our results are reproducible and if the model fits to other data sets well.

The test error for our KNN model with K=2 neighbors was 0.2060, for K=5 it was 0.1624, and for K=10 it was 0.1560. The best K to use out of these 3 options was K=10 because it yielded the lowest misclassification error rate on our test set.

Table 1: Table of Misclassification Error for Train and Test Sets

Number of Neighbors	Training Misclassification Error	Testing Misclassification Error
2	0.100834779733862	0.206042788325305
5	0.120151606390138	0.162475205440635
10	0.135182364540163	0.156064040804761

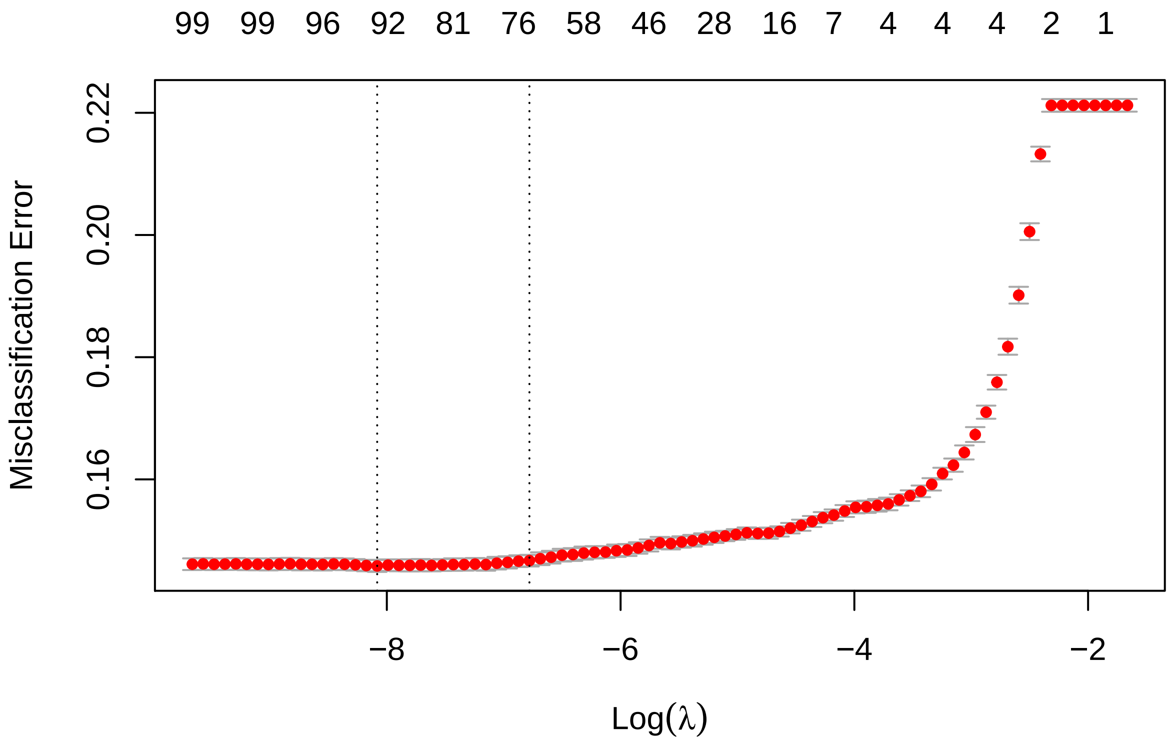
Penalized Logistic Regression

Logistic Regression using regularization is a powerful tool useful in determining variable importance. In particular, I used Lasso Regularization to eliminate insignificant predictors to fit my model.

In making the model, I first found the optimal value of lambda that minimizes cross-validation error.

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```



The plot displays the cross-validation error according to the log of lambda. The left dashed vertical line indicates that the log of the optimal value of lambda is approximately -9.2, which is the one that minimizes the prediction error. This lambda value will give the most accurate model.

Generally, the purpose of regularization is to balance accuracy and simplicity. This means, a model with the smallest number of predictors that also gives a good accuracy. To this end, the function cv.glmnet() finds also the value of lambda that gives the simplest model but also lies within one standard error of the optimal value of lambda - **lambda.1se**.

After fitting the model with lamda.1se, we result in these coefficients for our predictors.

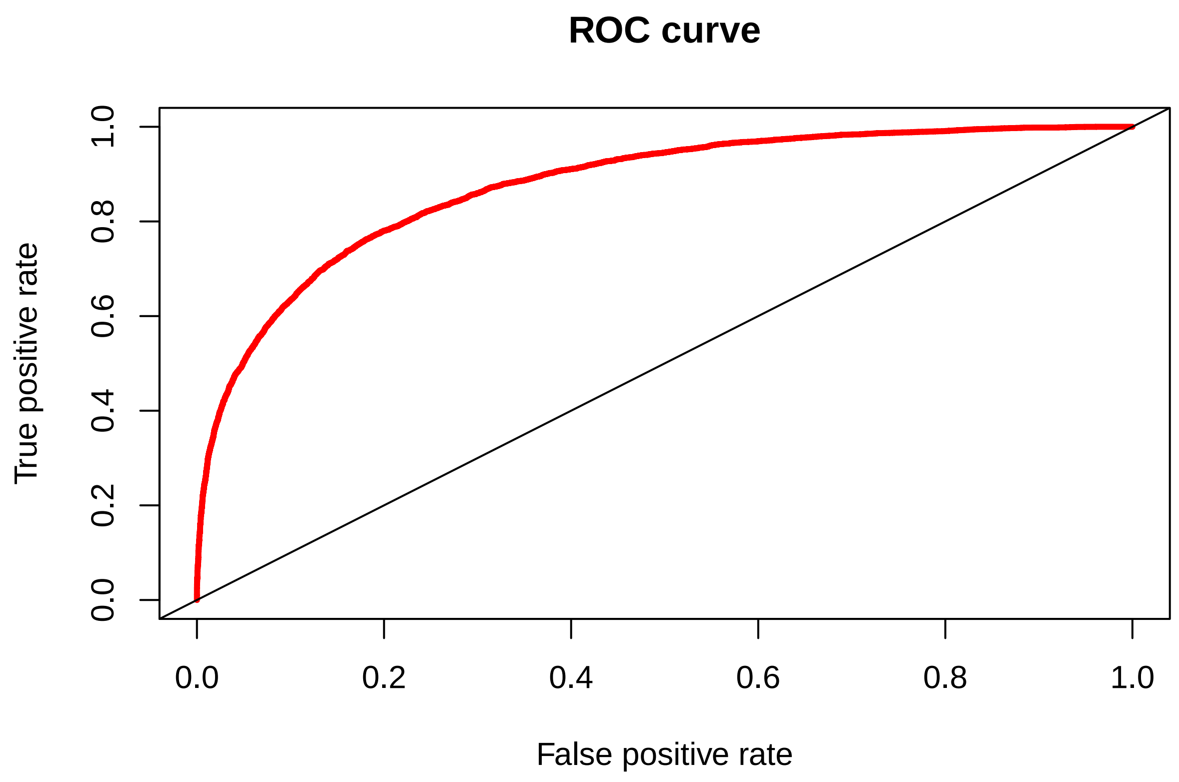
```
## 107 x 1 sparse Matrix of class "dgCMatrix"
##                                    1
## (Intercept)              49.625748158
## LocationAlbany             .
## LocationAlbury            0.223605555
## LocationAliceSprings       .
## LocationBadgerysCreek      .
## LocationBallarat         -0.467288483
## LocationBendigo           0.011080253
## LocationBrisbane          0.241127080
## LocationCairns           -0.088001922
## LocationCanberra         -0.041651264
## LocationCobar             0.304988327
## LocationCoffsHarbour     -0.070920411
## LocationDartmoor         -0.059810676
## LocationDarwin           -0.489523731
## LocationGoldCoast        -0.367551408
## LocationHobart           -0.465847788
## LocationKatherine        -0.795776889
## LocationLaunceston       -0.535132404
## LocationMelbourne        -0.010375987
## LocationMelbourneAirport -0.254503586
## LocationMildura           0.018507357
## LocationMoree              .
## LocationMountGambier      0.034716183
## LocationMountGinini        .
## LocationNewcastle          .
## LocationNhil             -0.160720102
## LocationNorahHead        -0.797846280
## LocationNorfolkIsland    -0.596404635
## LocationNuriootpa          .
## LocationPearceRAAF        0.068743674
## LocationPenrith            .
## LocationPerth             0.259123261
## LocationPerthAirport      0.203606502
## LocationPortland         -0.029430111
## LocationRichmond           .
## LocationSale             -0.350838251
## LocationSalmonGums         .
## LocationSydney            0.014374431
## LocationSydneyAirport      .
## LocationTownsville       -0.680586633
## LocationTuggeranong        .
## LocationUluru              .
## LocationWaggaWagga        0.246501580
## LocationWalpole          -0.158466340
## LocationWatsonia         -0.093749612
## LocationWilliamtown        .
## LocationWitchcliffe       0.173997845
## LocationWollongong       -0.991341364
## LocationWoomera            .
## MinTemp                   0.027414160
## MaxTemp                  -0.021987267
## Rainfall                  0.006963169
## WindGustDirENE           -0.039620655
## WindGustDirESE            0.003165128
## WindGustDirN               .
## WindGustDirNE            -0.238348014
## WindGustDirNNE           -0.173002908
## WindGustDirNNW            0.065149921
## WindGustDirNW              .
## WindGustDirS               .
## WindGustDirSE             0.021219526
## WindGustDirSSE             .
## WindGustDirSSW           -0.008823803
## WindGustDirSW              .
## WindGustDirW              0.001123926
## WindGustDirWNW             .
## WindGustDirWSW             .
## WindGustSpeed             0.059594729
## WindDir9amENE             0.199554218
## WindDir9amESE            -0.028819349
## WindDir9amN               0.172193576
## WindDir9amNE              0.229722945
## WindDir9amNNE             0.396352242
## WindDir9amNNW              .
## WindDir9amNW             -0.041818127
## WindDir9amS              -0.092295808
## WindDir9amSE             -0.066673068
## WindDir9amSSE            -0.055511024
## WindDir9amSSW            -0.041089745
## WindDir9amSW              0.035200139
## WindDir9amW                .
## WindDir9amWNW              .
## WindDir9amWSW             0.034803408
## WindDir3pmENE              .
## WindDir3pmESE              .
## WindDir3pmN               0.202252324
## WindDir3pmNE             -0.162674359
## WindDir3pmNNE             0.018351385
## WindDir3pmNNW             0.399146522
## WindDir3pmNW              0.297489065
## WindDir3pmS                .
## WindDir3pmSE               .
## WindDir3pmSSE            -0.076315433
## WindDir3pmSSW              .
## WindDir3pmSW             -0.087066649
## WindDir3pmW               0.066025108
## WindDir3pmWNW             0.168207598
## WindDir3pmWSW              .
## WindSpeed9am             -0.007475342
## WindSpeed3pm             -0.030693933
## Humidity9am               0.002932285
## Humidity3pm                0.071218370
## Pressure9am                0.103277638
## Pressure3pm               -0.159581410
## Temp9am                    .
## Temp3pm                    .
## RainToday                  0.471107069
```

Some seemingly significant predictors such as the temperature records had their coefficients set to zero by the lasso algorithm, reducing the complexity of the model.

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```



The ROC curve illustrates the diagnostic ability of a binary classifier to distinguish between the two classes. The higher area under the curve results in a better distinguishing model. The area under the curve of the model is 0.8755. Since it is closer to 1 than 0.5, I concluded that this model does a pretty good job at distinguishing between whether or not it will rain tomorrow in Australia.

```
##     obs
## pred     0     1
##    0 20873  3144
##    1  1076  3139
```

This table revealed as follows:

85.4633% of total cases were correctly classified
87.4832% of non-rainy days were correctly classified
74.2085% of rainy days were correctly classified

The test error rate is therefore 0.145367.
This test error rate of misclassification is slightly higher than what we saw in the KNN classification.

Conclusions

Based on the results, machine learning was fairly strong at predicting whether or not it will rain the next day in Australia. The models were all very close in performance. Even though the knn model didn't have any of the categorical variables that the logistic regression model had, it still performed nearly as well when k=10. It seems that the humidity levels (especially Humidity at 3pm) are the most significant predictors. I initially hypothesized that the Rainfall variable would be the most significant, but it turned out to be in line with most of the others. Logistic regression with lasso regularization was the second strongest after eliminating some predictors that were unimportant. Nevertheless, when looking at model performance, all algorithms show strong results for this data.

Table 2: Table of Test Missclassification Error

Model	Testing Missclassification Error
KNN 2	0.206042788325305
KNN 5	0.162475205440635
KNN 10	0.156064040804761
Logistic Regression	0.14947577217342

I was surprised that KNN performed as well as it did due to it typically performing best with smaller data sets with not as many features. After removing the categorical predictors, KNN was still left with 13 numeric covariates.

Future Research Directions

Expanding to other types of classification models such as Support Vector Machine or Naive Bayes could be useful predicting tools. Unsupervised learning methods could also be implemented, such as clustering. This could help reveal new patterns that were not noticeable with the supervised methods, and our models would also be less complex, potentially giving us a lower variance. It would be interesting to see the results of predicting on the amount of rainfall and not simply if it will rain or not.

References

Joe Young. (2018; December). Rain in Australia. Retrieved May 7, 2020 from https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

http://www.sthda.com/english/articles/36-classification-methods-essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-elastic-net/

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(tree)
# Reading in Data
df = read.csv('weatherAUS.csv')

set.seed(321)
dim(df)
head(df)
# Counting all the NA Values for each column
#sapply(df, function(x) sum(is.na(x))) %>% sort(decreasing=TRUE)
df = df %>% select(-Sunshine, -Evaporation, -Cloud3pm,
                   -Cloud9am, -Date, -RISK_MM) %>%
  na.omit()
df$RainToday = ifelse(df$RainToday == "Yes", 1, 0)
df$RainTomorrow = ifelse(df$RainTomorrow == "Yes", 1, 0)

# Set the Cat. vars to Factors
df$Location = as.factor(df$Location)
df$WindGustDir = as.factor(df$WindGustDir)
df$WindDir9am = as.factor(df$WindDir9am)
df$WindDir3pm = as.factor(df$WindDir3pm)


glimpse(df)
# histograms
par(mfrow=c(3,4))

hist(df$MinTemp)
hist(df$MaxTemp)
hist(df$Rainfall)
hist(df$WindGustSpeed)
hist(df$WindSpeed9am)
hist(df$WindSpeed3pm)
hist(df$Humidity9am)
hist(df$Humidity3pm)
hist(df$Temp9am)
hist(df$Temp3pm)
# boxplots

par(mfrow=c(1,4))
boxplot(MinTemp ~ RainTomorrow, data = df, ylab="Min Temperature (in degrees celsius)",
        xlab="Rain Tomorrow", main = "Min Temperature")
boxplot(MaxTemp ~ RainTomorrow, data = df, ylab="Max Temperature (in degrees celsius)",
        xlab="Rain Tomorrow", main = "Max Temperature")
boxplot(Rainfall ~ RainTomorrow, data = df, ylab="Amount of Rainfall that Day (in mm)",
        xlab="Rain Tomorrow", main = "Amount of Rainfall")
boxplot(WindGustSpeed ~ RainTomorrow, data = df, ylab="Wind Gust Speed (km/h)",
        xlab="Rain Tomorrow", main = "Wind Gust Speed")

par(mfrow=c(1,3))
```

```r
boxplot(WindSpeed9am ~ RainTomorrow, data = df, ylab="Wind Speed at 9am (km/hr)",
        xlab="Rain Tomorrow", main = "Wind Speed at 9am")
boxplot(WindSpeed3pm ~ RainTomorrow, data = df, ylab="Wind Speed at 3pm (km/hr)",
        xlab="Rain Tomorrow", main = "Wind Speed at 3pm")
boxplot(Humidity9am ~ RainTomorrow, data = df, ylab="Humidity 9am (percent)",
        xlab="Rain Tomorrow", main = "Humidity at 9am")

par(mfrow=c(1,3))
boxplot(Humidity3pm ~ RainTomorrow, data = df, ylab="Humidity 3pm (percent)",
        xlab="Rain Tomorrow", main = "Humidity at 3pm")
boxplot(Temp9am ~ RainTomorrow, data = df, ylab="Temperature at 9am (degrees C)",
        xlab="Rain Tomorrow", main = "Temperature at 9am")
boxplot(Temp3pm ~ RainTomorrow, data = df, ylab="Temperature at 3pm (degrees C)",
        xlab="Rain Tomorrow", main = "Temperature at 3pm")
library(class)
set.seed(321)
# KNN Train/Test Split
smp_size = floor(0.75 * nrow(df))

train_ind = sample(seq_len(nrow(df)), size=smp_size)

knn.train = df[train_ind, ]
knn.test = df[-train_ind, ]

knn.train = knn.train %>% select(-Location, -WindGustDir, -WindDir9am,
                                 -WindDir3pm, -RainToday)
knn.test = knn.test %>% select(-Location, -WindGustDir, -WindDir9am,
                               -WindDir3pm, -RainToday)



knn.XTrain = knn.train %>% select(-RainTomorrow)
knn.YTrain = knn.train$RainTomorrow

#XTrain is design matrix
knn.XTrain = scale(knn.XTrain, center= TRUE, scale= TRUE)

meanvec = attr(knn.XTrain, 'scaled:center')
sdvec = attr(knn.XTrain, 'scaled:scale')
knn.YTest = knn.test$RainTomorrow
knn.XTest = knn.test %>% select(-RainTomorrow)
knn.XTest = scale(knn.XTest, center= TRUE, scale= TRUE)

set.seed(321)
# k=2
# Train classifier and make predictions on the TRAINING set (k=2)
pred.YTrain2 = knn(train=knn.XTrain, test=knn.XTrain, cl=knn.YTrain, k=2)

# Calculate the confusion matrix
conf.train2 = table(predicted=pred.YTrain2, observed=knn.YTrain)
conf.train2

# Train accuarcy rate
```

```r
accuracy.train2 = sum(diag(conf.train2)/sum(conf.train2))

# Train error rate
train.error.knn2 = 1-accuracy.train2
train.error.knn2
set.seed(321)
# k=5
# Train classifier and make predictions on the TRAINING set (k=5)
pred.YTrain5 = knn(train=knn.XTrain, test=knn.XTrain, cl=knn.YTrain, k=5)

# Calculate the confusion matrix
conf.train5 = table(predicted=pred.YTrain5, observed=knn.YTrain)
conf.train5

# Train accuarcy rate
accuracy.train5 = sum(diag(conf.train5)/sum(conf.train5))

# Train error rate
train.error.knn5 = 1-accuracy.train5
train.error.knn5
set.seed(321)
# k=10
# Train classifier and make predictions on the TRAINING set (k=10)
pred.YTrain10 = knn(train=knn.XTrain, test=knn.XTrain, cl=knn.YTrain, k=10)

# Calculate the confusion matrix
conf.train10 = table(predicted=pred.YTrain10, observed=knn.YTrain)
conf.train10

# Train accuarcy rate
accuracy.train10 = sum(diag(conf.train10)/sum(conf.train10))

# Train error rate
train.error.knn10 = 1-accuracy.train10
train.error.knn10
set.seed(321)
# k=2
# train the classifier on TRAINING set and make predictions on the TEST set
pred.YTest2 = knn(train=knn.XTrain, test=knn.XTest, cl=knn.YTrain, k=2)


# Confusion Matrix
conf.test2 = table(predicted=pred.YTest2, observed=knn.YTest)
conf.test2

# Test accuracy rate
accuracy.test2 = sum(diag(conf.test2)/sum(conf.test2))

# Test error rate
test.error.knn2 = 1-accuracy.test2
test.error.knn2
set.seed(321)
# k=5
# train the classifier on TRAINING set and make predictions on the TEST set
pred.YTest5 = knn(train=knn.XTrain, test=knn.XTest, cl=knn.YTrain, k=5)


# Confusion Matrix
conf.test5 = table(predicted=pred.YTest5, observed=knn.YTest)
conf.test5

# Test accuracy rate
accuracy.test5 = sum(diag(conf.test5)/sum(conf.test5))

# Test error rate
test.error.knn5 = 1-accuracy.test5
test.error.knn5
set.seed(321)
# k=10
# train the classifier on TRAINING set and make predictions on the TEST set
pred.YTest10 = knn(train=knn.XTrain, test=knn.XTest, cl=knn.YTrain, k=10)


# Confusion Matrix
conf.test10 = table(predicted=pred.YTest10, observed=knn.YTest)
conf.test10

# Test accuracy rate
accuracy.test10 = sum(diag(conf.test10)/sum(conf.test10))

# Test error rate
test.error.knn10 = 1-accuracy.test10
test.error.knn10
###validation.error = NULL
###allK = 1:25

#### for each number in allK, use LOOCV to find a validation error
###for (i in allK){
###pred.Yval = knn.cv(train=knn.XTrain, cl=knn.YTrain, k=i)
###validation.error = c(validation.error, mean(pred.Yval!=knn.YTrain))
###}

####Best number of neighbors
###(if there is a tie, use larger number of neighbors for a simpler model)
###numneighbor = max(allK[validation.error == min(validation.error)])
###numneighbor
table = matrix(c("2", train.error.knn2, test.error.knn2,
                 "5", train.error.knn5, test.error.knn5,
                 "10", train.error.knn10, test.error.knn10), ncol=3, byrow=TRUE)

colnames(table) = c("Number of Neighbors","Training Misclassification Error",
                    "Testing Misclassification Error")
rownames(table) = c(1:3)
table2 = as.table(table)
library(knitr)
kable(table2, caption="Table of Misclassification Error for Train and Test Sets")
```

```r
set.seed(321)
# Train/Test Split
smp_size = floor(0.75 * nrow(df))

train_ind = sample(seq_len(nrow(df)), size=smp_size)

train = df[train_ind, ] %>% as_tibble()
test = df[-train_ind, ] %>% as_tibble()

XTrain = train %>% select(-RainTomorrow)
XTest = test$RainTomorrow
YTrain = test %>% select(-RainTomorrow)
YTest = test$RainTomorrow
library(glmnet)
set.seed(321)

# Lasso Regularization
x = model.matrix(RainTomorrow ~ ., data=train)[,-1]
cv.lasso = cv.glmnet(x, train$RainTomorrow, alpha = 1,
                     family = "binomial", type.measure = "class")
plot(cv.lasso)
set.seed(321)
# Final model with lambda.1se (min lamda)
lasso.model = glmnet(x, train$RainTomorrow, alpha = 1, family = "binomial",
                      lambda = cv.lasso$lambda.1se)

coef(lasso.model, cv.lasso$lambda.1se)

# Make prediction on test data
x.test = model.matrix(RainTomorrow ~ ., data=test)[,-1]

yhat = predict(lasso.model, x.test, s = cv.lasso$lambda.1se, type = "class")
set.seed(321)
# ROC Curve
library(ROCR)

prob.training = predict(lasso.model, x.test, s = cv.lasso$lambda.1se, type = "response")

# We want TPR on the y axis and FPR on the x axis
perf = performance(prediction(prob.training,test$RainTomorrow),
                   measure="tpr", x.measure="fpr")

# Lastly, plot the object you obtained above, and you will have the ROC curve
plot(perf, col=2, lwd=3, main="ROC curve")
abline(0,1)
# Logistic Regularization Results
tab = table(pred=yhat, obs=test$RainTomorrow) # confusion matrix
tab

# Missclassification TEST error rate
test.error.lasso = 1 - sum(diag(tab))/sum(tab)
table.total = matrix(c("KNN 2", test.error.knn2,
                       "KNN 5", test.error.knn5,
                        "KNN 10", test.error.knn10,
                        "Logistic Regression", test.error.lasso), ncol=2, byrow=TRUE)

colnames(table.total) = c("Model", "Testing Missclassification Error")
rownames(table.total) = c(1:4)
table.total2 = as.table(table.total)
library(knitr)
kable(table.total2, caption="Table of Test Missclassification Error")
```