Predicting Weather Forecast

Kruger 60 A

Research Skills: Programming with R

Name	u-number	e-mail
Vicky Mekes	u980683	v.j.b.mekes@tilburguniversity.edu
Sema Karan	u924823	s.karan@tilburguniversity.edu
Shelly van Erp	u1266624	s.d.a.vanerp@tilburguniversity.edu

Introduction

Weather forecasting has been attempted to be predicted formally since 19th century (wikipedia). Traditionally, this has been done through physical simulations in which the atmosphere is modeled as a fluid (Holmstrom, Liu & Vo, 2016). The samples from the present state of atmosphere are collected and prediction has been done by computing the numerical equations fluid dynamics & thermodynamics with the samples collected. Later on, a lot work and methods have been used to predict weather forecast with machine learning models since they are more robust to perturbations. In this project, a few different machine learning models are attempt to predict whether it'll rain tomorrow or not and the amount of rain that will rain tomorrow.

Data Description

The dataset used was retrieved from the competition website Kaggle. The original dataset contains 24 different variables about weather in Australia. Every observation represents one day at an Australian city and 20 of the variables represent specific measurements like humidity, wind speed, minimum and maximum temperature and so on. The dataset contains two target variables as presented as a challenge by Kaggle: a binary variable that states if it would rain the next day and a continuous variable that states the amount of rain that would fall the next day.

Research Questions

For the two target variables presented in the previous section, two separate research questions were constructed:

- R1: Which weather features can most accurately predict whether it will rain tomorrow?
- R2: How accurate can weather measurements from today predict the amount of rain that will fall tomorrow?

For both research questions, different types of modeling and feature selection methods were tried to find the optimal solution.

Packages Used

`caret` was used to make a test en training set for our dataset. Furthermore, we build a KNN model to predict research question 1. `tidyverse` is used to benefit from `dplyr` packages which is used to select and made new variables, subsetting or deleting certain unnecessary variables and also benefit from `ggplot2` which is used for visualization both in modelling and EDA parts, besides we followed tidy data principles with " %>% " in `tidyverse` package. `ggfortify` was needed for the PCA analysis. The package assists the `ggplot2` package and makes it possible to use the `autoplot()` function. The function made it possible to easily plot the

PCA graph. `leaps` and `MASS` are used by Caret to do the forward and backward feature selection methods. `MLmetrics` MLmetrics was used to get the mean squared error for all linear models tried to answer research question 2, to be able to compare the model performances. `neuralnet` The package neuralnet was used to build a simple neural network to answer question 2 and compare the results with the simple linear regression models. `Boruta` package is built to find important variables based on Boruta algoithm which is a wrapper built around random forest classification algorithm. It tries to bring variables as *important*, *unimportant* and *tentative*. We used the package to answer research question 1.

```
rm(list=ls(all=TRUE))
library(caret)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(ggfortify)
library(leaps)
library(MASS)
library(mice)
library(MLmetrics)
library(neuralnet)
library(Boruta)
```

Data Processing

First the data is loaded and structure is viewed to see what type of variables are available.

```
weather <- read.delim("weatherAUS.csv", sep=",", stringsAsFactors = FALSE)
str(weather) # dataset has int, num and chr values
```

```
## 'data.frame':    145460 obs. of  24 variables:
##  $ Date         : chr  "2008-12-01" "2008-12-02" "2008-12-03" "2008-12-04" ...
##  $ 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  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Sunshine     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ WindGustDir  : chr  "W" "WNW" "WSW" "NE" ...
##  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindDir9am   : chr  "W" "NNW" "W" "SE" ...
##  $ WindDir3pm   : chr  "WNW" "WSW" "WSW" "E" ...
##  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
##  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
##  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
##  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
##  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
##  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
##  $ RainToday    : chr  "No" "No" "No" "No" ...
##  $ RISK_MM      : num  0 0 0 1 0.2 0 0 0 1.4 0 ...
##  $ RainTomorrow : chr  "No" "No" "No" "No" ...
```

- **Variable classes**

The variables as presented in the original dataset were all integer, numerical and character values. We decided to convert all variables that contained character values to factors, except for the variable Date. This was done as character values are difficult to use in further analysis to answer the research questions.

```
weather$RainToday <- factor(weather$RainToday)
weather$RainTomorrow <- factor(weather$RainTomorrow)
weather$WindDir3pm <- factor(weather$WindDir3pm)
weather$WindDir9am <- factor(weather$WindDir9am)
weather$WindGustDir <- factor(weather$WindGustDir)
weather$Location <- factor(weather$Location)
```

- **Adding new variables**

As the Date variable is coded like *yyyy-mm-dd*, it was not very helpful to use this variable as a predictor, therefore we decided to break the variable down and make two new variables with it: *Month* and *Season*. First, we added the column Month by taking a substring from the *Date* column, after that we wrote our own function (`conver_season`) to convert a column with month numbers to the Australian season and we saved those in a new column: *Season*.

```
#Creating a new column with only the month number
weather$Season <- substr(weather$Date, 6, 7)
weather$Season <- as.numeric(weather$Season)
weather$Season[weather$Season == 12] <- 0

convert_season <- function(Month) {
  Month[Month <= 2] <- "Summer"
  Month[Month >= 3 & Month <= 5] <- "Autumn"
  Month[Month >= 6 & Month <= 8] <- "Winter"
  Month[!(Month %in% c("Autumn", "Winter", "Summer"))] <- "Spring"
  Month
}

weather$Season <- convert_season(weather$Season)
weather$Season <- factor(weather$Season )
```

- **Missing data**

The dataset contained quite a lot of missing data, especially for four of the variables: *Evaporation, Sunshine, Cloud9am and Cloud3pm.* Values of those four variables were missing for more than one third of the observations. Before deleting those variables from the dataset immediately we looked at some graphs to see if the variables could be good predictors for the target variables. Although *Cloud9am and Cloud3pm* seemed to be reasonably good predictors we decided to delete all four of them from consideration, as the fraction of data missing was too big.

For the other variables we tried to impute the missing values with multiple imputation using the `mice` package. Unfortunately, due to the size of the dataset the imputation was very computationally expensive so we decided to omit the observations that contained missing values. This still resulted in a dataset of **112.925 observations** (against 145.460 observations from the original dataset), which we think should be enough to get reasonable good results.

```
sapply(weather, function(x) sum(is.na(x))) # shows absolute numbers of missing values
```

```
##          Date      Location       MinTemp       MaxTemp      Rainfall
##             0             0          1485          1261          3261
##   Evaporation      Sunshine   WindGustDir WindGustSpeed    WindDir9am
##         62790         69835         10326         10263         10566
##    WindDir3pm  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm
##          4228          1767          3062          2654          4507
```

```
##   Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am 
##         15065         15028         55888         59358          1767 
##       Temp3pm     RainToday       RISK_MM  RainTomorrow        Season 
##          3609          3261          3267          3267             0
```

```r
colMeans(is.na(weather)) # shows percentage of missig values per column
```

```
##          Date      Location       MinTemp       MaxTemp      Rainfall 
##    0.00000000    0.00000000    0.01020899    0.00866905    0.02241853 
##   Evaporation      Sunshine   WindGustDir WindGustSpeed    WindDir9am 
##    0.43166506    0.48009762    0.07098859    0.07055548    0.07263853 
##    WindDir3pm  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm 
##    0.02906641    0.01214767    0.02105046    0.01824557    0.03098446 
##   Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am 
##    0.10356799    0.10331363    0.38421559    0.40807095    0.01214767 
##       Temp3pm     RainToday       RISK_MM  RainTomorrow        Season 
##    0.02481094    0.02241853    0.02245978    0.02245978    0.00000000
```

```r
weather_clean <- subset(weather, select = -c(Evaporation, Sunshine, Cloud3pm, Cloud9am))
weather_clean <- na.omit(weather_clean)

sapply(weather_clean, function(x) sum(is.na(x)))
```

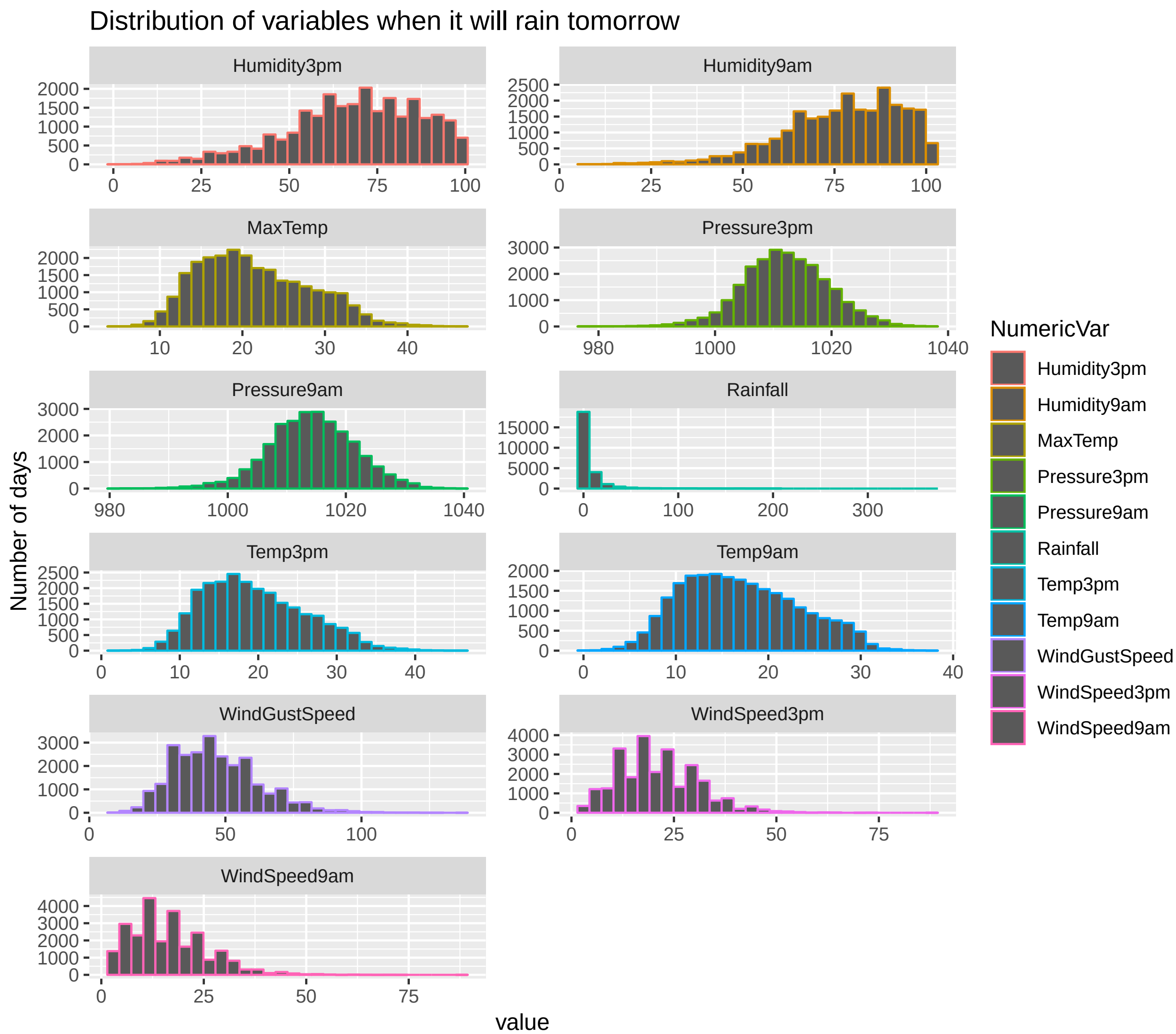
```
##          Date      Location       MinTemp       MaxTemp      Rainfall 
##             0             0             0             0             0 
##   WindGustDir WindGustSpeed    WindDir9am    WindDir3pm  WindSpeed9am 
##             0             0             0             0             0 
##  WindSpeed3pm   Humidity9am   Humidity3pm   Pressure9am   Pressure3pm 
##             0             0             0             0             0 
##       Temp9am       Temp3pm     RainToday       RISK_MM  RainTomorrow 
##             0             0             0             0             0 
##        Season 
##             0
```

Exploratory Data Analysis (EDA)

The graphs show the distribution of the numberic values per each binary target value. It can be observed tht skewnesses of **Humidity3pm** and **Humidty9am** are changing based on tomorrow is rainy or not. Pressure and temparature distributions are also slightly different that my help to predict whether it'll rain tomorrow or the amount of rain will it rain.

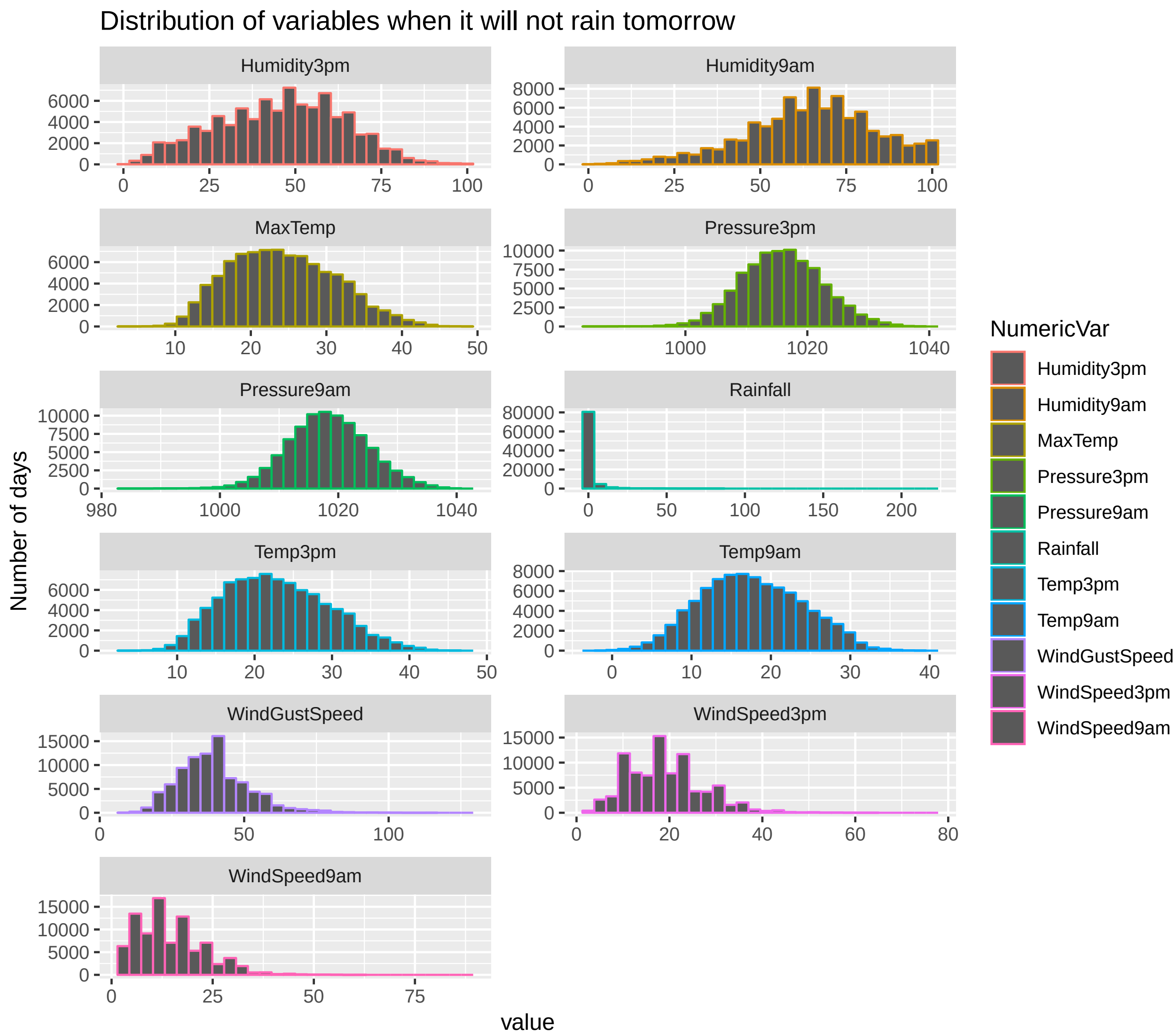
```r
num_values_paired <- weather_clean %>%
  gather("NumericVar", "Value", c(4,5,7, 10:17))

dist_yes <- ggplot(data = num_values_paired %>% filter(RainTomorrow == "Yes"),
              aes(x = Value, color = NumericVar)) +
  geom_histogram() +
  scale_y_continuous("Number of days") +
  scale_x_continuous("value") +
  facet_wrap( .~ NumericVar, ncol = 2, scales = "free") +
  ggtitle("Distribution of variables when it will rain tomorrow")
dist_yes
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
dist_no <- ggplot(data = num_values_paired %>% filter(RainTomorrow == "No"),
              aes(x = Value, color = NumericVar)) +
  geom_histogram() +
  scale_y_continuous("Number of days") +
  scale_x_continuous("value") +
  facet_wrap( .~ NumericVar, ncol = 2, scales = "free") +
  ggtitle("Distribution of variables when it will not rain tomorrow")
dist_no
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Modelling Research Question 1

When we approach the problem as **binary classification proplem** to predict whether it'll rain tomorrow or not, the algorithm will be **logistic regression** to apply on the problem. Since the research question

Target variable: *RainTomorrow* Predictor variables: all columns excluding *Date, Rainfall and RISK_MM*

```
weather_clean_r1 <- subset(weather_clean, select = -c(Date, RISK_MM))
weather_clean_r1$RainTomorrow <- relevel(factor(weather_clean_r1$RainTomorrow),
                                         ref = "Yes")
```

Splitting data to train and test

```
set.seed(1)
trn_index = createDataPartition(y = weather_clean_r1$RainTomorrow, p = 0.70, list = FALSE)
trn_weather = weather_clean_r1[trn_index, ]
tst_weather = weather_clean_r1[-trn_index, ]
```

Fitting training data to logistic regresssion

```r
set.seed(1)
weather_lgr = train(RainTomorrow ~ ., method = "glm",
  family = binomial(link = "logit"), data = trn_weather,
  trControl = trainControl(method = 'cv', number = 5))
weather_lgr
```

```
## Generalized Linear Model
##
## 79049 samples
##    18 predictor
##     2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 63239, 63239, 63240, 63239, 63239
## Resampling results:
##
##   Accuracy   Kappa
##   0.8526104  0.5224735
```

```r
set.seed(1)
predicted_outcomes <- predict(weather_lgr, tst_weather)
predicted_outcomes[1:10]
```

```
##  [1] No  No  No  No  Yes No  No  No  No  No
## Levels: Yes No
```

```r
accuracy <- sum(predicted_outcomes == tst_weather$RainTomorrow) /
  length(tst_weather$RainTomorrow)
accuracy
```

```
## [1] 0.8559747
```

```r
weather_confM <- confusionMatrix(predicted_outcomes, tst_weather$RainTomorrow)
weather_confM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  3913  1287
##        No   3592 25084
##
##                Accuracy : 0.856
##                  95% CI : (0.8522, 0.8597)
##     No Information Rate : 0.7785
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5309
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5214
##             Specificity : 0.9512
##          Pos Pred Value : 0.7525
##          Neg Pred Value : 0.8747
##              Prevalence : 0.2215
```

```
##          Detection Rate : 0.1155
##    Detection Prevalence : 0.1535
##       Balanced Accuracy : 0.7363
##
##        'Positive' Class : Yes
##
```

Modelling Research Question 2

Different linear models were tried and compared to predict the target variable **RISK_MM** (amount of rain that will fall tomorrow). First a linear regression model was tried, including all variables of the dataset, except Date. Furthermore, forward and backwards feature selection was tried using leapforward and leapbackward when training a model with the caret package in combination with the leap package. Finally a simple neural network was tried using the neuralnet package, with a minimum amount of steps and repetitions, as a neural network is very computationally expensive.

```
validation_index <- createDataPartition(weather_clean$RISK_MM, p=0.80,
  list=FALSE)
validation <- weather[-validation_index,]
training <- weather[validation_index,]
```

Stepwise feature selection using Caret, Leaps and MASS

1. Backward feature Selection

```
set.seed(1)

backwards_model <- train(RISK_MM ~. -Date, data = training,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:21),
                    trControl = trainControl(method = "cv", number = 10),
                    na.action = na.exclude)
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
backwards_model$results
```

```
##    nvmax     RMSE     Rsquared      MAE    RMSESD   RsquaredSD     MAESD
## 1      1 8.697830          NaN 3.630455 0.9591839           NA 0.1235640
## 2      2 8.697416 0.0002031529 3.631194 0.9588233 0.0002125965 0.1236830
## 3      3 8.697416 0.0002031529 3.631194 0.9588233 0.0002125965 0.1236830
## 4      4 8.697739 0.0002249690 3.632751 0.9582530 0.0002661360 0.1231116
## 5      5 8.698064 0.0003366549 3.632499 0.9576657 0.0002811147 0.1230688
## 6      6 8.698064 0.0003366549 3.632499 0.9576657 0.0002811147 0.1230688
## 7      7 8.697849 0.0003794793 3.632344 0.9569947 0.0004364977 0.1224418
```

```
## 8      8 8.695363 0.0007280285 3.633362 0.9571171 0.0004948038 0.1225979
## 9      9 8.695313 0.0007432527 3.633280 0.9570285 0.0005198907 0.1224274
## 10    10 8.695313 0.0007432527 3.633280 0.9570285 0.0005198907 0.1224274
## 11    11 8.695313 0.0007432527 3.633280 0.9570285 0.0005198907 0.1224274
## 12    12 8.694887 0.0008578391 3.633191 0.9566129 0.0006981803 0.1221934
## 13    13 8.694601 0.0009789304 3.632238 0.9567783 0.0009275987 0.1216595
## 14    14 8.690989 0.0019024542 3.623074 0.9576572 0.0013752577 0.1250086
## 15    15 8.687290 0.0027616919 3.618969 0.9589493 0.0020885857 0.1270121
## 16    16 8.684358 0.0034175109 3.615306 0.9588695 0.0020522074 0.1297516
## 17    17 8.684886 0.0033319565 3.610752 0.9587846 0.0021125003 0.1274870
## 18    18 8.683839 0.0035563641 3.606769 0.9588644 0.0019504692 0.1263319
## 19    19 8.680749 0.0042891776 3.603602 0.9591848 0.0022613752 0.1249355
## 20    20 8.679912 0.0044882975 3.602622 0.9598266 0.0024043552 0.1260605
## 21    21 8.678815 0.0048210382 3.600000 0.9598230 0.0025966240 0.1272538
```

2. Forward Feature Selection

```r
forwards_model <- train(RISK_MM ~. -Date, data = training,
                    method = "leapForward",
                    tuneGrid = data.frame(nvmax = 1:21),
                    trControl = trainControl(method = "cv", number = 10),
                    na.action = na.exclude)
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```r
forwards_model$results
```

```
##    nvmax     RMSE     Rsquared      MAE    RMSESD   RsquaredSD      MAESD
## 1      1 8.710901          NaN 3.630435 0.8158373           NA 0.09716823
## 2      2 8.710813 0.0005740259 3.631416 0.8159127 0.0006337442 0.09812566
## 3      3 8.709845 0.0011483461 3.633610 0.8153266 0.0014113852 0.09515250
## 4      4 8.709978 0.0010238798 3.633492 0.8154866 0.0011874112 0.09534003
## 5      5 8.709947 0.0009255191 3.634209 0.8157340 0.0009330428 0.09584392
## 6      6 8.709947 0.0009255191 3.634209 0.8157340 0.0009330428 0.09584392
## 7      7 8.709922 0.0008299370 3.635695 0.8147858 0.0007761237 0.09489148
## 8      8 8.707385 0.0014245905 3.636038 0.8155334 0.0009759725 0.09401778
## 9      9 8.707385 0.0014245905 3.636038 0.8155334 0.0009759725 0.09401778
## 10    10 8.707440 0.0014227357 3.636123 0.8155060 0.0009785388 0.09411625
## 11    11 8.707444 0.0014216861 3.636127 0.8155092 0.0009775753 0.09411560
## 12    12 8.707671 0.0013527013 3.636455 0.8153776 0.0009316779 0.09396287
## 13    13 8.703314 0.0021746093 3.628890 0.8164417 0.0011984250 0.09633563
## 14    14 8.701720 0.0025053144 3.625373 0.8151127 0.0016733235 0.09094472
## 15    15 8.701061 0.0026430864 3.624208 0.8157222 0.0015739746 0.09232809
## 16    16 8.700615 0.0028050713 3.622985 0.8167275 0.0016077425 0.09340926
## 17    17 8.660699 0.0111615395 3.621352 0.7897585 0.0239539002 0.09701419
## 18    18 8.613828 0.0215611509 3.615911 0.7949714 0.0375659870 0.09465045
```

```
## 19     19 8.558245 0.0332329689 3.606431 0.7586154 0.0477685065 0.08369568
## 20     20 8.461997 0.0553411369 3.611015 0.7661294 0.0547328817 0.07901003
## 21     21 8.418610 0.0643535862 3.609957 0.7239887 0.0523136338 0.07949351
```

Conclusion

- **Research Question 1**
- **Research Question 2**

When interpreting the model results for research question 2, the linear regression model including all variables from the dataset performed best in predicting the amount of rain tomorrow. To answer the research question, **"how accurate can weather measurements from today predict the amount of rain that will fall tomorrow?"**, our models did not predict the amount of rain very accurately. The best model had a high mean squared error *($MSE = 44.85$)* and the R^2 score *($R^2 = 0.31$)* was very low, which shows that the model did not explain a lot of variability in the target variable RISK_MM and thus does not fit the data well. We expect to get more accurate results with a more computationally expensive neural network, when normalizing the numerical variables on beforehand and when increasing the amount of steps and repetitions the network may make.