Rainfall Prediction in New Zealand

A machine learning project

Alexandre Monteiro

December 21, 2021

Contents

Preface

This report is a submission for the capstone project of HarvardX's Data Science Professional Certificate[1].It is based on a dataset of rainfall registers for 30 sites across New Zealand[2] made avaiable on Kaggle[3] by user Pralabh Poudel[4]. The datasets are licensed under a Creative Commons Attribution 4.0 International License.

1 Introduction

Rain is a vital resource for life. Be it to drink, to grow our food or to generate energy, water supply greatly depends on rainfall and changes in ammount or timing can have great impact on how we live our lives. New Zealand makes avaiable data on annual and seasonal rainfall at 30 sites from 1960 to 2019 and these data will be used on this project to build a machine learning algorithm to predict rainfall for each of these avaiable sites, which are representative of the area were they are measured. Criteria used to select the 30 sites is described in Macara et al. (2020).

Machine learning techniques can help us to find patterns in data and transform it into useful information. In this case, our purpose is to find and train an algorithm to better predict rainfall.

1.1 Rainfall Dataset

From the three datasets made avaiable on Kaggle, we will focus on the *state_data* dataset, which contains the rainfall data - measured in milimiters - for each station, summarized by season and by annual precipitation.

1.2 Model Evaluation

The proposed model is to return the rainfall from the other predictors (site, season, year). That means the algorithm will returne a regression value and a loss function is ideal to evaluate our predictions. The loss function of our choice is the root mean square error (RMSE) since it is one of the most usual tools and fits well to our purpose.

The chosen function is defined by the following formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i,u}(\hat{y}_{i,u} - y_{i,u})^2}$$

[1]https://www.edx.org/professional-certificate/harvardx-data-science
[2]https://www.stats.govt.nz/indicators/rainfall
[3]https://www.kaggle.com/pralabhpoudel/nz-rainfall-dataset
[4]https://www.kaggle.com/pralabhpoudel

Where N is the total number of rainfall values in the dataset, $y_{i,u}$ is the registered rainfall for season i (Summer of 1970, Spring of 2015 or Year of 2000, for example) at the site u and $\hat{y}_{i,u}$ is the prediction of rainfall for that season i at that site u.

2 Data and Analysis

Our first step is to download our data and examine it to understand how they are related and understand which are the most useful columns to be used as predictors.

2.1 Data Aquisition

```
# Load data
state_data <- read_csv("state_data.csv")
```

The first few lines of the dataset are as follows:

```
# The first few lines
head(state_data, n=10) %>%
  kbl(booktabs = TRUE,
      caption = "The first 10 lines of the original data") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 1: The first 10 lines of the original data

agent_number	season	precipitation	period_start	period_end	lat	lon	site	anomaly	reference_period
1056	Spring	326.6	2019-09-01	2019-11-30	-35.18300	173.9260	Kerikeri	-42.47667	1961-1990
1287	Spring	308.2	2019-09-01	2019-11-30	-35.76900	174.3640	Whangarei	-11.12000	1961-1990
1400	Spring	210.7	2019-09-01	2019-11-30	-36.60600	174.8350	Whangaparaoa	-49.11000	1961-1990
15752	Spring	212.4	2019-09-01	2019-11-30	-45.90129	170.5147	Dunedin	58.93000	1961-1990
1615	Spring	225.6	2019-09-01	2019-11-30	-37.67300	176.1960	Tauranga	-69.21667	1961-1990
1770	Spring	219.0	2019-09-01	2019-11-30	-38.10595	176.3149	Rotorua	-122.31667	1961-1990
1858	Spring	158.0	2019-09-01	2019-11-30	-38.74263	176.0810	Taupo	-69.06000	1961-1990
1962	Spring	234.4	2019-09-01	2019-11-30	-37.00813	174.7887	Auckland	-21.06667	1961-1990
2112	Spring	224.2	2019-09-01	2019-11-30	-37.86088	175.3317	Hamilton	-76.34000	1961-1990
2250	Spring	350.4	2019-09-01	2019-11-30	-38.88784	175.2598	Taumarunui	3.33000	1961-1990

From this, we can see that the *reference_period* column has only one value (that can be verified by extracting the unique values), and that it can be removed from the table. According to the New Zealand Government Statitcs Site for the rainfall[5], Daily rainfall is measured from 9am to 9am of the following day, with missing data being replaced with adjusted daily Virtual Climate Station Network (VCSN) data from 1960 onwards. Then, the accumulated rainfall is calculated by aggregating daily rainfall data. This was done annually, and for each meteorological season of the year: Summrer = December (previous year) - Janyary - February, Autumn = March - April - May, Winter = June - July - August, and Spring = September - October - November. Using this information, the columns *period_start* and *period_end* can be replaced for simplicity by a column containing the year of *period_end*. Thus, for example, the period of 1979-12-01 to 1980-02-28 can be referenced as Summer of 1980. This is consistent with the calculation of the rainfall baseline (described bellow). The columns *agent_number*, *lat* & *lon* (considered as a pair), and *site* are ways to describe the same information: the identity of the meteorological stations that gave origin to the data, so, for this report's purpose, it suffices to use only the *site* name to describe the stations in a unique way.

[5] https://www.stats.govt.nz/indicators/rainfall

The *precipitation* column contains the calculated rainfall for that specific time period (Season or Year). In our model, *anomaly* will be our **outcome** variable and the other selected features will be our **predictors**. Since anomaly is a continuous variable, we will build a **predictive model**.

Finally, the original dataset can be simplified as follows:

```
# Data adjustments
state_data <- state_data %>%
  mutate(year = year(period_end)) %>%
  select(-reference_period, -agent_number, -period_start,
         -period_end, -lat, -lon) %>%
  select(site, season, year, precipitation, anomaly)
```

And the first few lines of our work dataset are:

```
# The first few lines
head(state_data, n=10) %>%
  kbl(booktabs = TRUE,
      caption = "The first 10 lines of the work dataset") %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 2: The first 10 lines of the work dataset

site	season	year	precipitation	anomaly
Kerikeri	Spring	2019	326.6	-42.47667
Whangarei	Spring	2019	308.2	-11.12000
Whangaparaoa	Spring	2019	210.7	-49.11000
Dunedin	Spring	2019	212.4	58.93000
Tauranga	Spring	2019	225.6	-69.21667
Rotorua	Spring	2019	219.0	-122.31667
Taupo	Spring	2019	158.0	-69.06000
Auckland	Spring	2019	234.4	-21.06667
Hamilton	Spring	2019	224.2	-76.34000
Taumarunui	Spring	2019	350.4	3.33000

2.2 Baseline

The baseline is the long term average of the precipitation values. It is used to calculate anomaly, which is the difference between the actual value for a period and the 30-year long average. The dataset reference baseline is the average of rainfall for each season, and annual, between 1961 and 1990.

```
# Baseline
#
# Baseline is used to calculate the rainfall anommaly. Simply put, anomaly
# is the diference between the actual value of rainfall and a long term
# average. Our baseline will be the average precipitation for a period of
# 30 years, namely 1961 to 1990.
# We have to take into account that, for purposes of methodology, Summer
# is the period encompassed between December, 1st and March, 1st (open
# end) of the following year and is counted for that year, ie, from
```

```r
# 1970-12-01 to 1971-02-28 the period is named Summer of 1971.
# That is why the filter is made using 'period_end' column.
# Although the data is avaiable as a column of the dataset, the
# calculation is performed for the sake of verification.
baseline <- state_data %>%
  filter(year >= 1961,
         year < 1991) %>%
  group_by(season, site) %>%
  summarize(base_precip = mean(precipitation))
```

```
## `summarise()` has grouped output by 'season'. You can override using the `.groups` argument.
```

2.3 Data preparation

Having settled on the dataset structure, definining which columns will be the features (site, season, year, precipitation) and which is the outcome (anomaly), it is time to divide the data set for training and verification of the algorithms.

The original *state_data* dataset will be divided into two sets: a *rainfall* dataset, that will contain 90% of the data and will be used to train the algorithms, and the *verification* dataset, that will contain the other 10% of the data and will be used only in the final verification.

In order to train the algorithms properly, the *rainfall* dataset will be divided again into two datasets, *train_set* and *test_set*, in order to execute proper cross validation of the training data while tuning the algorithms.

```r
# Now separate the state_data dataset into 'rainfall' and 'verification'
# datasets. The 'rainfall'dataset will be used to train the algorithm
# and will have 90% of the total data. The 'verification' dataset will
# be used in the final verification and will have the other 10% of the
# data.

# Setting the random seed
# if using R 3.5 or earlier, use `set.seed(2021)`
set.seed(2021, sample.kind = "Rounding")

test_ind <- createDataPartition(y = state_data$precipitation,
                                times = 1, p = 0.1, list = FALSE)
rainfall <- state_data[-test_ind,]
verification <- state_data[test_ind,]

# Dividing the 'rainfall' dataset again into a 'train_set' and a
# 'test_set' that will be used during the algorithm training.
# The 'train_set'will have 90% of the data in the 'rainfall' dataset amd
# the 'test_set' will have the other 10% of the data in the 'rainfall'
# dataset.
test_ind <- createDataPartition(y = rainfall$precipitation,
                                times = 1, p = 0.1, list = FALSE)

train_set <- rainfall[-test_ind,]
test_set <- rainfall[test_ind,]

remove(test_ind)
```

2.4 Data exploration

The *rainfall* dataset will be used to make data exploration, in order to better understand our data. By doing this exploration, we will have a beter idea on how to construct our algorithm.

The structure of the dataset is as follows:

```
# Data structure
str(rainfall)
```

```
## tibble [8,070 x 5] (S3: tbl_df/tbl/data.frame)
##  $ site         : chr [1:8070] "Kerikeri" "Whangarei" "Whangaparaoa" "Dunedin" ...
##  $ season       : chr [1:8070] "Spring" "Spring" "Spring" "Spring" ...
##  $ year         : num [1:8070] 2019 2019 2019 2019 2019 ...
##  $ precipitation: num [1:8070] 327 308 211 212 226 ...
##  $ anomaly      : num [1:8070] -42.5 -11.1 -49.1 58.9 -69.2 ...
```

The *rainfall* dataset is comprised of five columns: site, season, year, precipitation and anomaly. Each line of the dataset represents the precipitation (and its anomaly regarding the long term average) for a particular site during a particular season in a specific year. There are a total of 8070 rows of data. A data summary for the dataset:

```
# Data summary
summary(rainfall)
```

```
##      site              season               year      precipitation
##  Length:8070        Length:8070        Min.   :1960   Min.   :  23.4
##  Class :character   Class :character   1st Qu.:1975   1st Qu.: 191.6
##  Mode  :character   Mode  :character   Median :1990   Median : 291.2
##                                        Mean   :1990   Mean   : 512.7
##                                        3rd Qu.:2004   3rd Qu.: 574.5
##                                        Max.   :2019   Max.   :9258.8
##     anomaly
##  Min.   :-2259.6600
##  1st Qu.:  -65.7658
##  Median :   -8.0883
##  Mean   :   -0.0173
##  3rd Qu.:   56.0642
##  Max.   : 2756.2400
```

We can check if there is any blank in the dataset:

```
# Check for blank data
count_blank <- function(name) {
  sum(is.na(rainfall[, name]))
}

sapply(names(rainfall), count_blank)
```

```
##          site        season          year precipitation       anomaly
##             0             0             0             0             0
```

And the first lines of the *rainfall* dataset can be seen here:

```r
# The first few lines
head(state_data, n=10) %>%
  kbl(booktabs = TRUE,
      caption = "The first 10 lines of the rainfall dataset") %>%
  kable_styling(latex_options = "hold_position")
```

Table 3: The first 10 lines of the rainfall dataset

site	season	year	precipitation	anomaly
Kerikeri	Spring	2019	326.6	-42.47667
Whangarei	Spring	2019	308.2	-11.12000
Whangaparaoa	Spring	2019	210.7	-49.11000
Dunedin	Spring	2019	212.4	58.93000
Tauranga	Spring	2019	225.6	-69.21667
Rotorua	Spring	2019	219.0	-122.31667
Taupo	Spring	2019	158.0	-69.06000
Auckland	Spring	2019	234.4	-21.06667
Hamilton	Spring	2019	224.2	-76.34000
Taumarunui	Spring	2019	350.4	3.33000

2.4.1 Climate stations and seasons

The climate station names and season names are simple categorical variables, and the list of unique values that are in the *rainfall* dataset are listed bellow:

```r
# Exploration
unique(rainfall$site)
```

```
##  [1] "Kerikeri"      "Whangarei"     "Whangaparaoa"  "Dunedin"
##  [5] "Tauranga"      "Rotorua"       "Taupo"         "Auckland"
##  [9] "Hamilton"      "Taumarunui"    "New Plymouth"  "Dannevirke"
## [13] "Wellington"    "Napier"        "Waiouru"       "Whanganui"
## [17] "Masterton"     "Hokitika"      "Reefton"       "Nelson"
## [21] "Blenheim"      "Christchurch"  "Lake Tekapo"   "Timaru"
## [25] "Queenstown"    "Gore"          "Invercargill"  "Gisborne"
## [29] "Milford Sound" "Tara Hills"
```

```r
unique(rainfall$season)
```

```
## [1] "Spring" "Winter" "Autumn" "Annual" "Summer"
```

2.4.2 Precipitation

The average precipoitation for each site and for each season is:

```r
# Mean rainfall values by season and site
rainfall %>%
  group_by(season, site) %>%
  summarize(mean_rainfall = mean(precipitation)) %>%
  pivot_wider(names_from = season,
              values_from = mean_rainfall) %>%
  kbl(booktabs = TRUE,
      caption = "Average precipitation for each site and each season.") %>%
  kable_styling(latex_options = "hold_position")
```

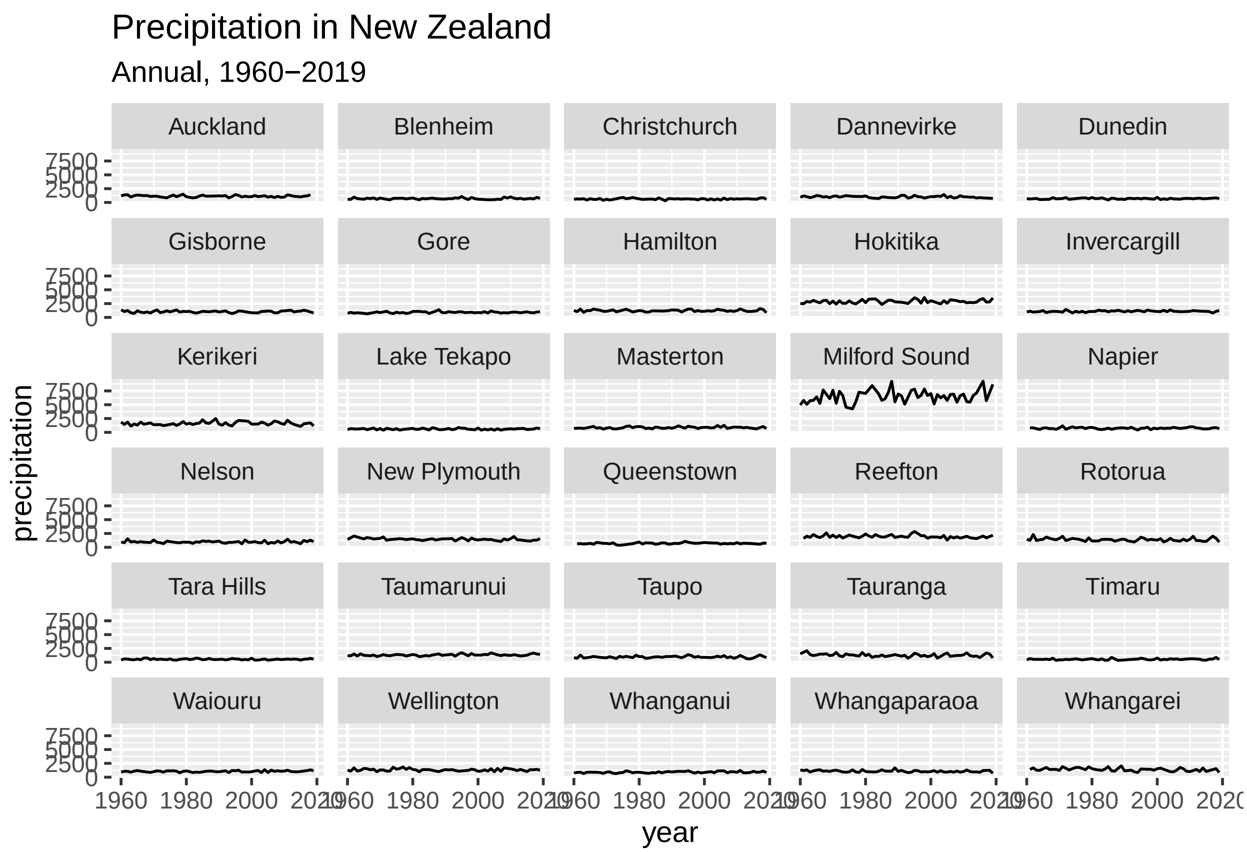
Table 4: Average precipitation for each site and each season.

site	Annual	Autumn	Spring	Summer	Winter
Auckland	1136.8981	278.1034	258.1466	224.9283	366.8927
Blenheim	705.3019	177.1370	177.3778	145.2882	199.6175
Christchurch	628.4821	162.0833	132.4824	133.6096	186.4642
Dannevirke	1019.1836	236.4528	258.9534	227.4558	285.0074
Dunedin	696.1673	177.1722	155.6037	188.5278	168.3071
Gisborne	1036.7698	283.2055	204.0426	203.3750	327.2396
Gore	915.1556	242.1038	222.4125	256.1755	185.8404
Hamilton	1220.5943	286.2589	292.0346	267.1472	381.5981
Hokitika	2868.8792	707.2712	767.4164	681.5696	694.0472
Invercargill	1105.1849	300.1185	267.1302	270.1449	253.1034
Kerikeri	1617.6722	401.9080	357.2439	322.2109	555.9404
Lake Tekapo	597.8382	151.1772	150.2404	128.0904	168.0396
Masterton	868.4148	219.1635	200.1436	164.9145	290.3692
Milford Sound	6524.1000	1763.7800	1732.7943	1765.9691	1295.6638
Napier	754.5518	197.1520	154.0527	164.8473	235.2000
Nelson	969.0836	262.1500	246.8353	221.4091	262.8945
New Plymouth	1457.0471	354.5463	349.9509	303.7643	451.8521
Queenstown	700.2327	179.4321	178.4768	172.5309	164.2855
Reefton	1970.8094	460.2836	536.0654	424.5333	527.9618
Rotorua	1407.9755	358.2000	306.4036	335.7389	396.7453
Tara Hills	537.4255	133.9882	130.7389	146.5698	117.9055
Taumarunui	1310.8370	295.0130	358.2286	288.5554	382.2946
Taupo	951.1473	215.7725	224.3315	243.1420	272.2132
Tauranga	1270.9310	354.1618	264.6889	279.0712	379.7075
Timaru	535.0100	139.8115	126.6698	143.9080	120.8648
Waiouru	1059.5727	243.7643	265.6389	242.9377	310.2407
Wellington	1319.5000	326.3000	319.6660	251.8347	420.0792
Whanganui	899.5127	220.4727	221.8375	208.2255	250.9870
Whangaparaoa	1081.2561	270.4655	244.5417	218.6462	350.8625
Whangarei	1386.2633	362.2500	302.3018	282.3074	459.1889

And the evolution of annual precipitation through the years for all sites can be put on a graph:

```r
# Annual precipitation data for all sites
rainfall %>%
  filter(season == "Annual") %>%
```

```
  ggplot(aes(x = year,
             y = precipitation)) +
  geom_line() +
  facet_wrap(. ~ site, ncol = 5) +
  ggtitle("Precipitation in New Zealand",
          subtitle = "Annual, 1960-2019")
```

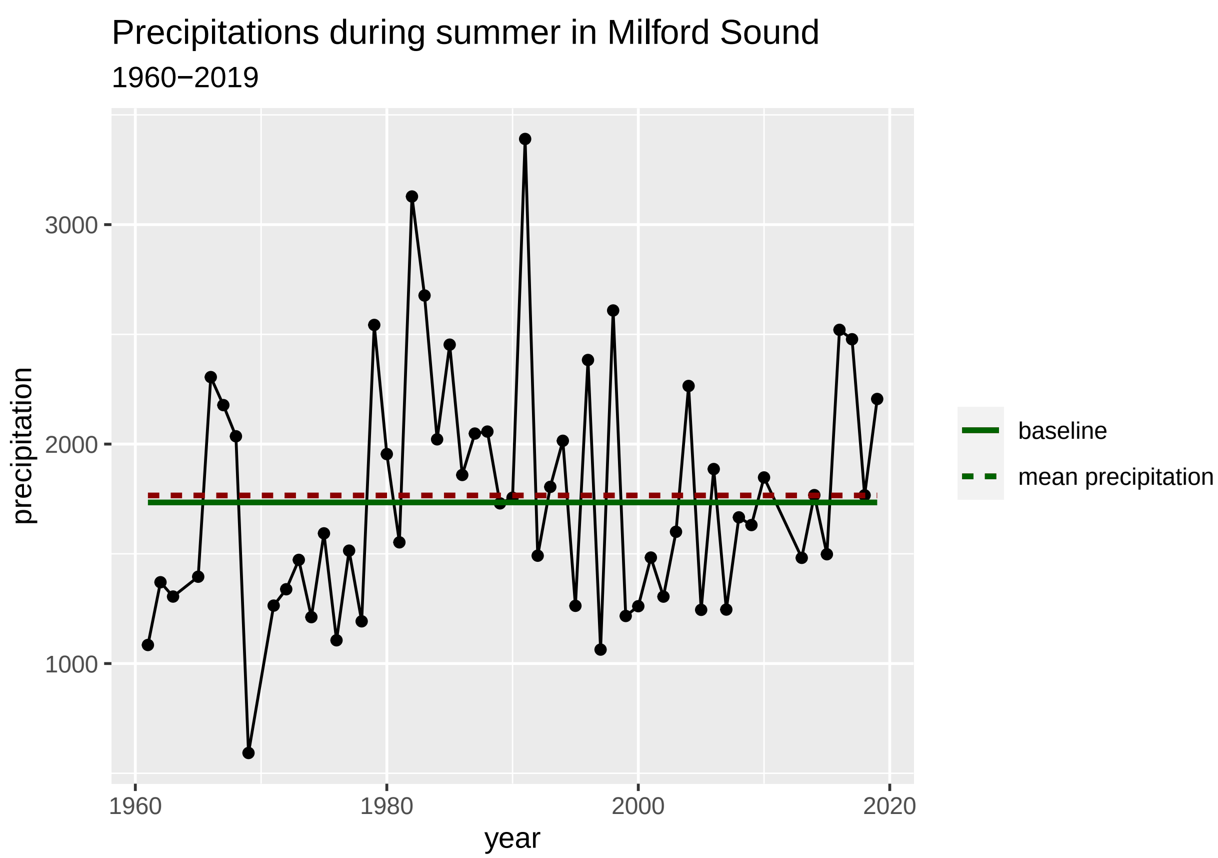


From this graph, we can see that Milford Sound registers values of precipitation much higher than the other climate sites.

To give an example, and better visualize the precipitation, curve, here is the plot for Summer rainfall through the years for Milford Sound climate station.

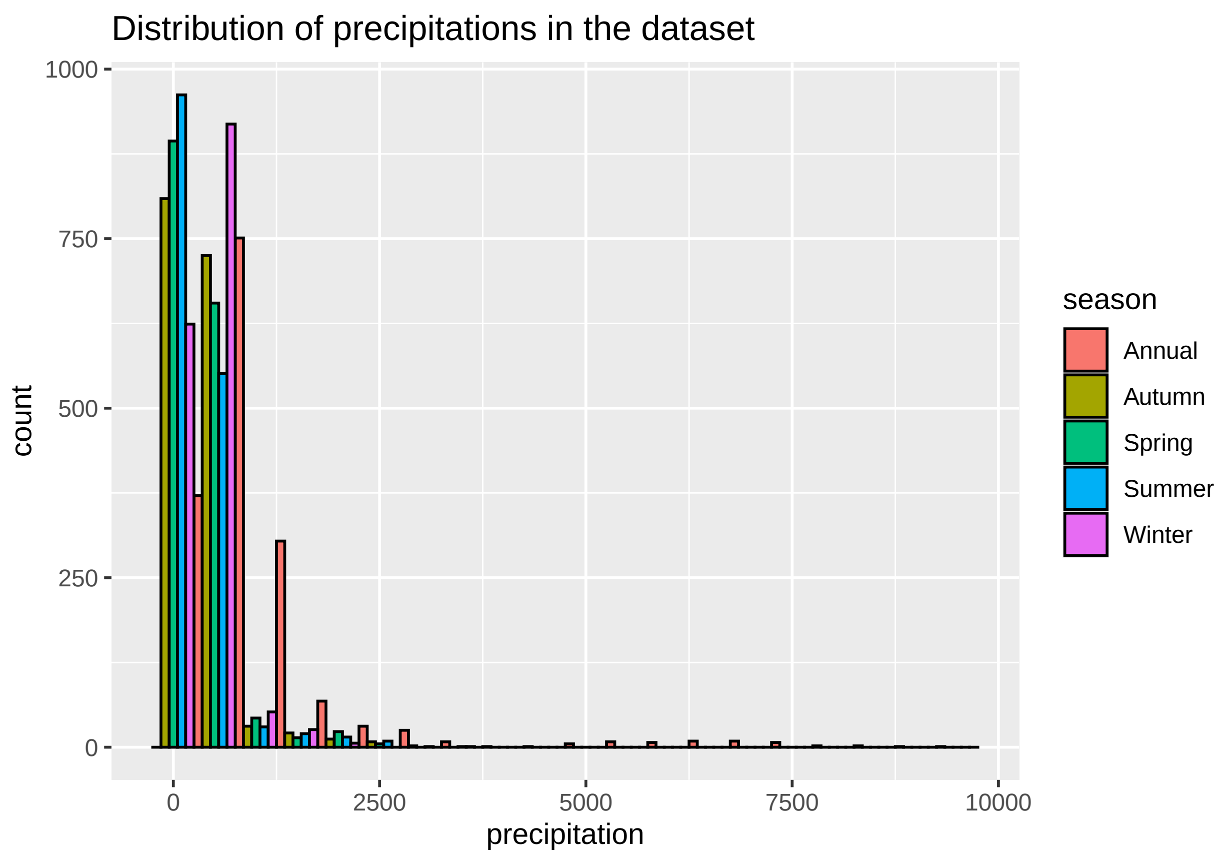
```
# Take one station as an examble

# Get its baseline value
base_precip <- baseline %>%
  filter(site == "Milford Sound",
         season == "Summer") %>%
  pull(base_precip)

# Plot the graph
rainfall %>%
  filter(season == "Summer",
         site == "Milford Sound") %>%
  ggplot(aes(x = year, y = precipitation)) +
```

```
  geom_line() +
  geom_point() +
  geom_line(aes(y = mean(precipitation),
                lty = "mean precipitation"),
            col = "darkred",
            size = 1) +
  geom_line(aes(y = base_precip,
                lty = "baseline"),
            col = "darkgreen",
            size = 1) +
  ggtitle("Precipitations during summer in Milford Sound",
          subtitle = "1960-2019") +
  labs(linetype = NULL)
```



The distribution of precipitation throughout the dataset is:

```
# Rain distribution
rainfall %>%
  ggplot(aes(x = precipitation,
             fill = season)) +
  geom_histogram(color = "black",
                 binwidth = 500,
                 position = "dodge") +
  ggtitle("Distribution of precipitations in the dataset")
```




```
rainfall %>%
  group_by(season) %>%
  summarize(mean = mean(precipitation),
            q75 = quantile(precipitation, 0.75)) %>%
  kbl(booktabs = TRUE,
      caption = "Rain distribution: mean and 75th percentile.") %>%
  kable_styling(latex_options = "hold_position")
```

Table 5: Rain distribution: mean and 75th percentile.

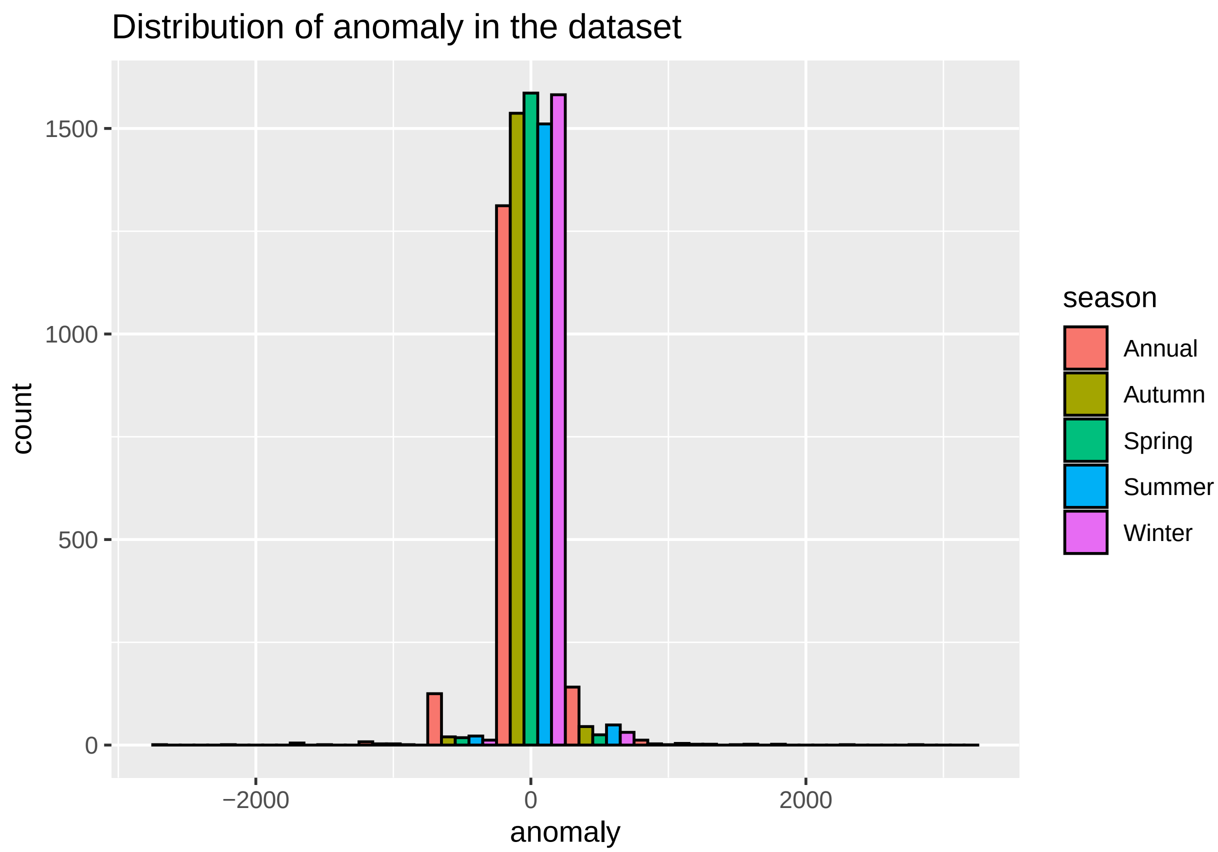
season	mean	q75
Annual	1281.4171	1333.00
Autumn	321.4389	343.80
Spring	312.0179	323.00
Summer	299.5026	312.30
Winter	350.3175	422.75

From this, it shows that there is not much difference in precipitation values between seasons, with slightly more rain in winter and slightly less rain during summer. It is usual to observe a rainfall of 420 mm or less in a typical season, and 1300 mm or less in a typical year.

2.4.3 Anomaly

The distribution of anomalies in the data set can be seen in the following graph.

```r
# Anomaly distribution
# Rain distribution
rainfall %>%
  ggplot(aes(x = anomaly,
             fill = season)) +
  geom_histogram(color = "black",
                 binwidth = 500,
                 position = "dodge") +
  ggtitle("Distribution of anomaly in the dataset")
```



3 Modeling

Our objective is to predict the anomaly for each of the 30 climate stations in our report. The *rainfall* dataset has three predictors (site, season, and year) and one outcome (precipitation). Since our outcome is continuous, we are going to model a few algorithms suitable to work with continuous variables. With this in mind, five algorythms were chosen, four of them were used before during the Machine Learning course, the last one was selected for being a robust algorithm with a criterion to avoid overtraining.

The models chosen to be applied to the model are: Linear Regression, k-Nearest Nighbors, Generalized Adictive Method with LOESS (gam loess), Regression Tree, and Bayesian Regularized Neural Network (brnn).Linear Regression, as the most basic model, will serve as a baseline of how algorythm will perform.

4 Results

4.1 Setup

Our first step before running the algorithms is to setup the random seed and define the formula of the root mean square error (RMSE), which will be used to evaluate each model.

```
##
## Machine Learning Algorithms
##

# Our objective is to use machine learning algorythms to predict
# precipitation anomaly for each of the 30 climate stations.

# For each algorithm, the predictors will be:
# season, precipitation, year, and site.

#Set the random seed again.
set.seed(1973, sample.kind = "Rounding")

# Loss function: mean square error
RMSE <- function(true_values, predicted_values){
  sqrt(mean((true_values - predicted_values)^2))
}
```

4.2 Linear Regression

First of all, we will gather some info on the method:

```
#
# First algorithm: linear regression
#

# Model info
modelLookup("lm")
```

model	parameter	label	forReg	forClass	probModel
lm	intercept	intercept	TRUE	FALSE	FALSE

And then we will train the algorithm using a simple bootstrap resampling with a 10-fold cross validation, repeated 3 times. After that, we apply the data on *train_set* to set the parameters.

```
# Setting the control parameters
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3)

# Train the algorithm
lmFit <- train(anomaly ~ .,
               data = train_set,
               method = "lm",
               trControl = ctrl)
```

The results of the method:

```r
# Results
lmFit
```

```
## Linear Regression
##
## 7262 samples
##    4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 6536, 6536, 6536, 6537, 6536, 6536, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   148.1351  0.1742271  83.67857
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Summary
summary(lmFit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2417.72   -55.89   -11.10    47.18  1626.32
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -4.373e+02  2.017e+02  -2.168 0.030215 *
## siteBlenheim         3.239e+01  1.344e+01   2.410 0.015993 *
## siteChristchurch     4.262e+01  1.338e+01   3.184 0.001457 **
## siteDannevirke       5.173e-01  1.342e+01   0.039 0.969248
## siteDunedin          3.283e+01  1.344e+01   2.442 0.014635 *
## siteGisborne         1.106e+01  1.332e+01   0.831 0.406186
## siteGore             2.003e+01  1.331e+01   1.505 0.132298
## siteHamilton        -7.134e+00  1.345e+01  -0.531 0.595725
## siteHokitika        -1.147e+02  1.379e+01  -8.322  < 2e-16 ***
## siteInvercargill     1.054e+01  1.329e+01   0.793 0.427679
## siteKerikeri        -3.982e+01  1.326e+01  -3.004 0.002675 **
## `siteLake Tekapo`    3.631e+01  1.336e+01   2.719 0.006570 **
## siteMasterton        2.658e+01  1.327e+01   2.003 0.045239 *
## `siteMilford Sound` -4.113e+02  1.694e+01 -24.286  < 2e-16 ***
## siteNapier           2.453e+01  1.329e+01   1.845 0.065080 .
## siteNelson           5.034e+00  1.329e+01   0.379 0.704858
## `siteNew Plymouth`  -4.149e+01  1.340e+01  -3.096 0.001968 **
## siteQueenstown       3.639e+01  1.336e+01   2.725 0.006455 **
## siteReefton         -7.567e+01  1.344e+01  -5.630 1.87e-08 ***
## siteRotorua         -3.724e+01  1.343e+01  -2.774 0.005554 **
```

```
## `siteTara Hills`     3.872e+01  1.353e+01   2.862 0.004220 **
## siteTaumarunui       5.171e+00  1.329e+01   0.389 0.697257
## siteTaupo            1.711e+01  1.336e+01   1.280 0.200436
## siteTauranga        -2.597e+01  1.333e+01  -1.948 0.051442 .
## siteTimaru           4.693e+01  1.358e+01   3.457 0.000549 ***
## siteWaiouru          1.545e+01  1.323e+01   1.167 0.243100
## siteWellington      -2.798e+01  1.331e+01  -2.103 0.035491 *
## siteWhanganui        3.247e+01  1.328e+01   2.445 0.014517 *
## siteWhangaparaoa    -1.279e+01  1.301e+01  -0.983 0.325534
## siteWhangarei       -5.702e+01  1.344e+01  -4.242 2.24e-05 ***
## seasonAutumn         1.813e+02  7.251e+00  25.001  < 2e-16 ***
## seasonSpring         1.826e+02  7.296e+00  25.028  < 2e-16 ***
## seasonSummer         1.919e+02  7.386e+00  25.984  < 2e-16 ***
## seasonWinter         1.837e+02  7.183e+00  25.570  < 2e-16 ***
## year                 1.031e-01  1.012e-01   1.019 0.308023
## precipitation        1.892e-01  4.969e-03  38.079  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 148.4 on 7226 degrees of freedom
## Multiple R-squared:  0.172,  Adjusted R-squared:  0.168
## F-statistic:  42.9 on 35 and 7226 DF,  p-value: < 2.2e-16
```

Calculating the loss function:

```
# RMSE
lmPredict <- predict(lmFit,newdata = test_set)
error_value <- RMSE(test_set$anomaly, lmPredict)

# Add result to tabe
results_RMSE <- tibble(algorithm = "linear regression",
                       RMSE = error_value)

results_RMSE %>%
   kbl(booktabs = TRUE,
      caption = "Results table") %>%
  kable_styling(latex_options = "hold_position")
```

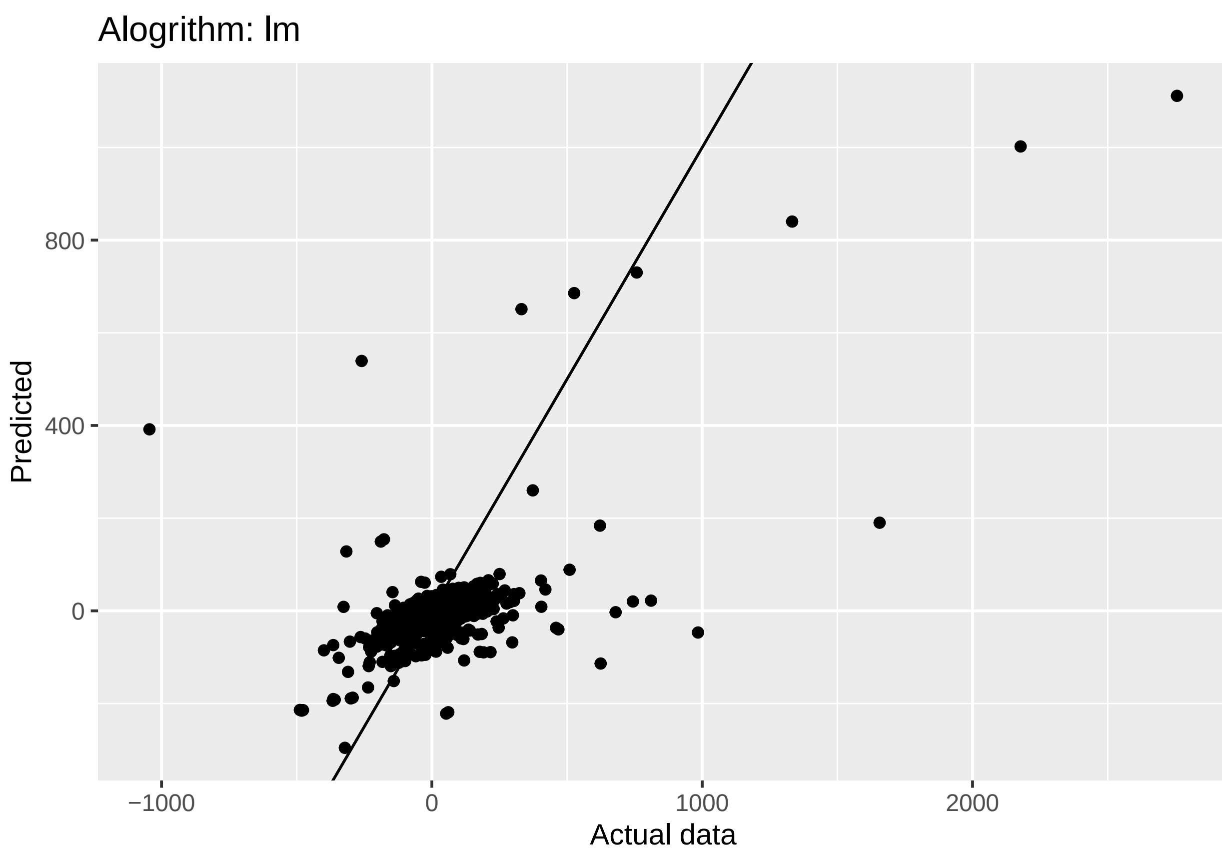
Table 6: Results table

algorithm	RMSE
linear regression	160.9478

And, finally, plotting the predicted values against the actual vaules on the *test_set*:

```
# Plot predicted vs actual values in the train_set
data.frame(x = test_set$anomaly,
           y = lmPredict) %>%
  ggplot(aes(x = x,
             y = y)) +
  geom_point() +
  geom_abline(slope = 1) +
```

```r
  ggtitle("Alogrithm: lm") +
  xlab("Actual data") +
  ylab("Predicted")
```



4.2.1 Findings

This is the most basic algorithm and was used to set a base of compasion. We can see that for categorical variables, it is created a new numerical variable (for example *seasonSummer*) that can assume value 0 or 1, thus the categorical value can be enconded into the regression formula.

The plot of the predicted data versus the real data shows the prediction is poor, specially when studyng annual values, as can be checked in the following table.

```r
# Error analysis per season
test_set %>%
  cbind(fitted = lmPredict) %>%
  group_by(season) %>%
  summarize(rmse = RMSE(anomaly, fitted)) %>%
   kbl(booktabs = TRUE,
      caption = "RMSE per season - Linear Regression.") %>%
  kable_styling(latex_options = "hold_position")
```

Table 7: RMSE per season - Linear Regression.

season	rmse
Annual	252.0652
Autumn	100.0710
Spring	112.5516
Summer	150.9840
Winter	135.0651

4.3 k-Nearest Neighbors

As with last, algorithm, we will first gather information on the method and its parameters:

```
#
# Second algorithm: knn
#

# Model info
modelLookup("knn")
```

model	parameter	label	forReg	forClass	probModel
knn	k	#Neighbors	TRUE	TRUE	TRUE

Once again, the algorithm will be trained with a 10-fold bootstrap repeated 3 times, applying the *train_set* data to optimize the number of neighbors.

```
# Setting the control parameters
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3)

# Train the algorithm
knnFit <- train(anomaly ~ .,
                data = train_set,
                method = "knn",
                trControl = ctrl,
                tuneGrid = data.frame(k = seq(1,15,2)),
                preProcess = c("center", "scale"),
                tuneLength = 20)
```

The results are:

```
# Results
knnFit
```

```
## k-Nearest Neighbors
##
## 7262 samples
##    4 predictor
##
## Pre-processing: centered (35), scaled (35)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
```

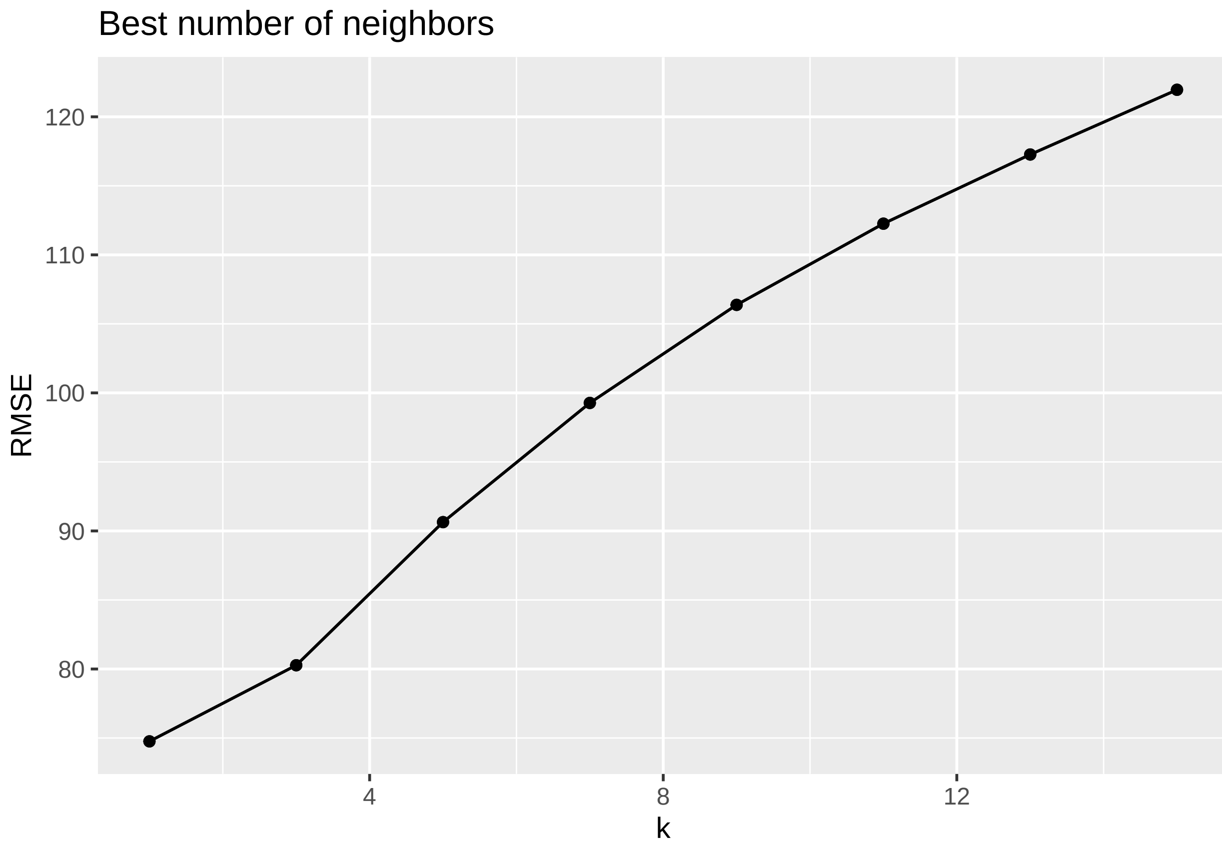
```
## Summary of sample sizes: 6535, 6536, 6536, 6536, 6536, 6536, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1   74.75790  0.7860963  52.60455
##    3   80.27223  0.7711005  55.59705
##    5   90.63955  0.7208402  63.11886
##    7   99.27230  0.6666728  69.14349
##    9  106.37716  0.6144823  73.73072
##   11  112.25916  0.5739101  77.42578
##   13  117.26773  0.5324819  80.32435
##   15  121.95737  0.4941384  82.79611
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 1.
```

```r
# Summary
summary(knnFit)
```

```
##             Length Class      Mode
## learn        2     -none-     list
## k            1     -none-     numeric
## theDots      0     -none-     list
## xNames      35     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    1     -none-     logical
## param        0     -none-     list
```

And the best number of neighbors can be seen on this graph.

```r
# Best number of neighbors
knnFit %>%
  ggplot(aes(x = .$results[k],
             y = .$results[RMSE])) +
  geom_point() +
  geom_line() +
  ggtitle("Best number of neighbors") +
  xlab("k") +
  ylab("RMSE")
```



Calculating the loss function and adding the result to the results table.

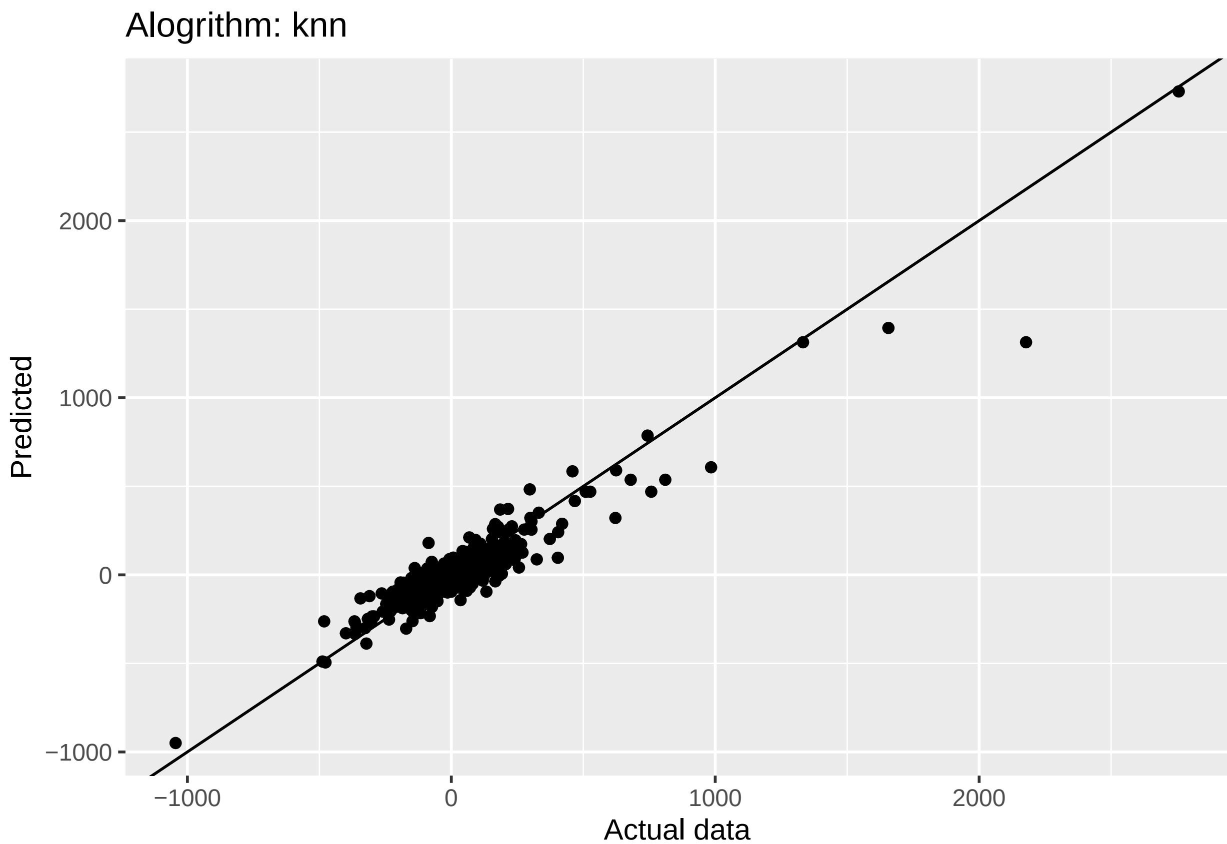
```
# Calculate RMSE
knnPredict <- predict(knnFit,newdata = test_set)
error_value <- RMSE(test_set$anomaly, knnPredict)

# Creater a table to group the resulst of all algorithms
results_RMSE <- results_RMSE %>%
  rbind(tibble(algorithm = "k-nearest neighbors",
               RMSE = error_value))

results_RMSE %>%
   kbl(booktabs = TRUE,
      caption = "Results table") %>%
  kable_styling(latex_options = "hold_position")
```

Table 8: Results table

algorithm	RMSE
linear regression	160.94783
k-nearest neighbors	74.69472

And plotting the predicted value of precipitation versus the actual values on the *test_set*.

```r
# Plot predicted vs actual values in the train_set
data.frame(x = test_set$anomaly,
           y = knnPredict) %>%
ggplot(aes(x = x,
           y = y)) +
  geom_point() +
  geom_abline(slope = 1) +
  ggtitle("Alogrithm: knn") +
  xlab("Actual data") +
  ylab("Predicted")
```



4.3.1 Findings

The k-Nearest Neighbors algorithm performs better than linear regression, yet it still performs poorly at the Annual strata. The optimal number of neighbors is

k
1

.

As per linear model, the error for the Annual series is about twice the error for each season.

```r
# Error analysis per season
test_set %>%
  cbind(fitted = knnPredict) %>%
```

```
  group_by(season) %>%
  summarize(rmse = RMSE(anomaly, fitted)) %>%
   kbl(booktabs = TRUE,
       caption = "RMSE per season - knn.") %>%
  kable_styling(latex_options = "hold_position")
```

Table 9: RMSE per season - knn.

season	rmse
Annual	110.79252
Autumn	65.73300
Spring	56.51545
Summer	57.57174
Winter	66.94702

4.4 Generalized Adictive Method with LOESS

The gamLoess algorithm is the next algorithm to be considered.

```
#
# Third algorithm: gamLoess
#

# Model info
modelLookup("gamLoess")
```

model	parameter	label	forReg	forClass	probModel
gamLoess	span	Span	TRUE	TRUE	TRUE
gamLoess	degree	Degree	TRUE	TRUE	TRUE

And the training will be conducted in similar way to the previous methods.

```
# Setting the control parameters
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3)

grid <- expand.grid(span = seq(0.15, 0.65, len = 10),
                    degree = 1)

# Train the algorithm
gamFit <- train(anomaly ~ .,
                data = train_set,
                method = "gamLoess",
                trControl = ctrl,
                tuneGrid = grid)
```

And the results for the algorithm.

```r
# Results
gamFit
```

```
## Generalized Additive Model using LOESS
##
## 7262 samples
##    4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 6536, 6536, 6535, 6536, 6536, 6536, ...
## Resampling results across tuning parameters:
##
##   span       RMSE      Rsquared    MAE
##   0.1500000  155.9317  0.08552609  87.25063
##   0.2055556  155.4865  0.08877861  86.90994
##   0.2611111  155.1872  0.09127187  86.64016
##   0.3166667  154.2985  0.09950439  86.21144
##   0.3722222  154.0771  0.10154476  86.03983
##   0.4277778  153.9236  0.10299654  85.95553
##   0.4833333  153.8206  0.10400413  85.89714
##   0.5388889  153.7580  0.10461278  85.86602
##   0.5944444  153.7165  0.10502701  85.84393
##   0.6500000  153.6937  0.10513611  85.80753
##
## Tuning parameter 'degree' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were span = 0.65 and degree = 1.
```

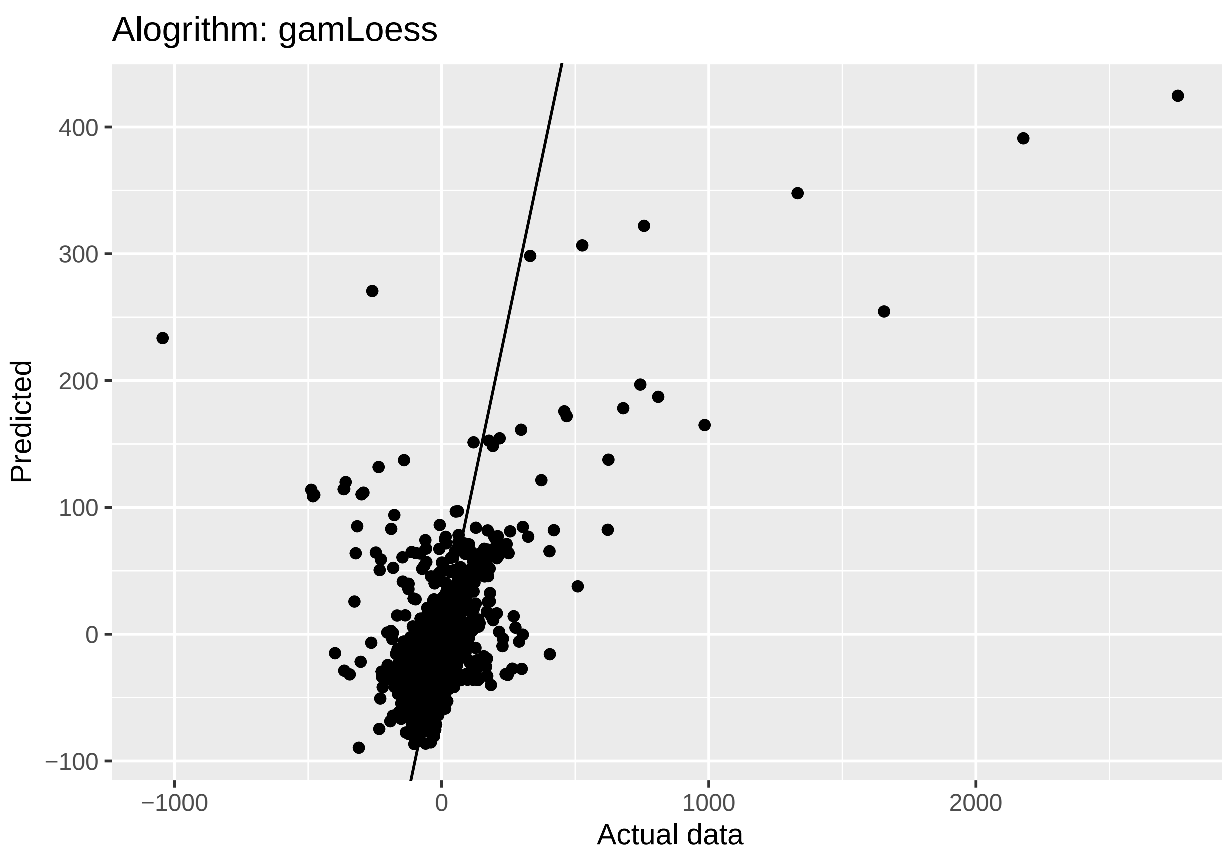
```r
# RMSE
gamPredict <- predict(gamFit,newdata = test_set)
error_value <- RMSE(test_set$anomaly, gamPredict)

# Add result to tabe
results_RMSE <- results_RMSE %>%
  rbind(tibble(algorithm = "generalized aditive model",
               RMSE = error_value))

results_RMSE %>%
   kbl(booktabs = TRUE,
      caption = "Results table") %>%
  kable_styling(latex_options = "hold_position")
```

Table 10: Results table

algorithm	RMSE
linear regression	160.94783
k-nearest neighbors	74.69472
generalized aditive model	178.34719

And finally plot the predicted data versus the precipitation value on the *test_set* dataset.

```r
# Plot predicted vs actual values in the train_set
data.frame(x = test_set$anomaly,
           y = gamPredict) %>%
  ggplot(aes(x = x,
             y = y)) +
  geom_point() +
  geom_abline(slope = 1) +
  ggtitle("Alogrithm: gamLoess") +
  xlab("Actual data") +
  ylab("Predicted")
```



4.4.1 Findings

This method performs poorly even when compared with the Linear Regression results. As other algorithms, the annual series show the largesr error when compared to each season series.

The algorithm determines the relationship of each individual predictor and the dependent variable.

```r
# Error analysis per season
test_set %>%
  cbind(fitted = gamPredict) %>%
  group_by(season) %>%
  summarize(rmse = RMSE(anomaly, fitted))  %>%
   kbl(booktabs = TRUE,
```

```
      caption = "RMSE per season - gamLoess.") %>%
  kable_styling(latex_options = "hold_position")
```

Table 11: RMSE per season - gamLoess.

season	rmse
Annual	306.4554
Autumn	119.3233
Spring	100.5266
Summer	150.7407
Winter	120.3532

4.5 Regression Tree

The info on the regression tree algorithm avaiable on the caret package is:

```
#
# Fourth algorithm: regression tree
#

# Model info
modelLookup("rpart")
```

model	parameter	label	forReg	forClass	probModel
rpart	cp	Complexity Parameter	TRUE	TRUE	TRUE

The trainning information for this algorithm:

```
# Setting the control parameters
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3)
grid <- expand.grid(cp = range(0, 3, 0.1))

# Train the algorithm
rpFit <- train(anomaly ~ .,
               data = train_set,
               method = "rpart",
               trControl = ctrl,
               tuneGrid = grid)
```

And the results:

```
# Results
rpFit
```

```
## CART
##
## 7262 samples
```

```
##    4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 6536, 6536, 6538, 6536, 6535, 6535, ...
## Resampling results across tuning parameters:
##
##   cp  RMSE       Rsquared   MAE
##   0    70.26619  0.8145697  38.38821
##   3   161.46713        NaN  95.78929
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.
```

```
# RMSE
rpPredict <- predict(rpFit,newdata = test_set)
error_value <- RMSE(test_set$anomaly, rpPredict)

# Add result to tabe
results_RMSE <- results_RMSE %>%
  rbind(tibble(algorithm = "Regression Tree",
               RMSE = error_value))
results_RMSE
```

algorithm	RMSE
linear regression	160.94783
k-nearest neighbors	74.69472
generalized aditive model	178.34719
Regression Tree	90.34400

4.5.1 Findings

Up to now, this is the algorithm with the best results and, like the other algorithms, it performs worse for the annual precipitation data.

```
# Error analysis per season
test_set %>%
  cbind(fitted = rpPredict) %>%
  group_by(season) %>%
  summarize(rmse = RMSE(anomaly, fitted))  %>%
   kbl(booktabs = TRUE,
      caption = "RMSE per season - regression tree.") %>%
  kable_styling(latex_options = "hold_position")
```

4.6 Bayesian Regularized Neural Network

The last algorithm used in this report is the Bayesian Regularized Neural Network.

```
#
# Fifth algorithm: Bayesian Regularized Neural Network
#
```

Table 12: RMSE per season - regression tree.

season	rmse
Annual	169.03463
Autumn	40.01646
Spring	37.94764
Summer	64.13099
Winter	62.78531

```
# Model info
modelLookup("brnn")
```

model	parameter	label	forReg	forClass	probModel
brnn	neurons	# Neurons	TRUE	FALSE	FALSE

The training of the algorithm is:

```
# Setting the control parameters
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3)
grid <- expand.grid(neurons = seq(2,3,1))

# Train the algorithm
brnnFit <- train(anomaly ~ .,
                 data = train_set,
                 method = "brnn",
                 trControl = ctrl,
                 tuneGrid = grid)
```

```
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.1205    alpha= 0.0259   beta= 2170.901
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.5691   alpha= 0.541    beta= 24427.69
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.7689    alpha= 0.3995   beta= 362.1554
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 106.7695   alpha= 0.8517   beta= 20417.6
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 73.2605    alpha= 0.4626   beta= 5102.81
## Number of parameters (weights and biases) to estimate: 111
```

```
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.8835   alpha= 1.101    beta= 19992.15
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 73.2371    alpha= 0.49     beta= 194.0248
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.941    alpha= 0.834    beta= 12280
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.4688    alpha= 1.2592   beta= 358.0429
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.5163   alpha= 0.7524   beta= 23734.64
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 73.8238    alpha= 0.4299   beta= 428.607
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 107.1399   alpha= 0.349    beta= 23622.95
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.2652    alpha= 0.0397   beta= 2116.171
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 105.8596   alpha= 0.161    beta= 23445.76
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 73.7986    alpha= 0.4301   beta= 379.1519
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.7457   alpha= 0.4874   beta= 24513.93
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.6361    alpha= 0.4001   beta= 5178.63
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 107.2305   alpha= 0.5214   beta= 25035.98
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
```

```
## gamma= 73.6686    alpha= 0.5821   beta= 346.9291
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.6202   alpha= 0.4911   beta= 20411.91
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.4541    alpha= 0.0475   beta= 218.1608
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.2391   alpha= 0.2725   beta= 21340.93
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.468     alpha= 1.2611   beta= 353.5546
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001176
## gamma= 108.8117   alpha= 0.432    beta= 24635.61
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 72.0675    alpha= 0.0218   beta= 2221.824
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.3028   alpha= 0.8952   beta= 24836.7
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.0775    alpha= 0.0236   beta= 2194.459
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 106.4594   alpha= 0.709    beta= 23565.79
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 73.4921    alpha= 1.2736   beta= 348.6197
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.9344   alpha= 1.1617   beta= 20011.01
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 72.1453    alpha= 0.0281   beta= 2145.762
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 106.8798   alpha= 1.2927   beta= 17292.52
## Number of parameters (weights and biases) to estimate: 74
```

```
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.1012    alpha= 0.0244   beta= 1912.033
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.7709   alpha= 0.2218   beta= 25876.83
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.6309    alpha= 0.3997   beta= 5151.366
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.7813   alpha= 0.4953   beta= 24635.06
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.6326    alpha= 0.3913   beta= 5224.92
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.852    alpha= 0.7336   beta= 25390.14
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 73.8643    alpha= 0.2651   beta= 407.6582
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.9321   alpha= 1.1302   beta= 19966.4
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.9229    alpha= 0.042    beta= 363.7987
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 107.4065   alpha= 0.8164   beta= 14672.31
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.0442    alpha= 0.0214   beta= 2190.253
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.0067   alpha= 0.5647   beta= 17262.62
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.8402    alpha= 0.4201   beta= 376.2044
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
```

```
## gamma= 110.0092   alpha= 1.0939   beta= 17342.53
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.5209    alpha= 0.4674   beta= 5063.309
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.8872   alpha= 1.1091   beta= 19846.02
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.0475    alpha= 0.0221   beta= 2187.262
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.9119   alpha= 1.7706   beta= 14237.43
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 72.4581    alpha= 0.0478   beta= 2661.021
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.8581   alpha= 1.1198   beta= 20286.96
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 73.8408    alpha= 0.1548   beta= 332.9856
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.7968   alpha= 0.293    beta= 30943.67
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000742
## gamma= 72.7895    alpha= 0.3174   beta= 4787.315
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.743    alpha= 0.5619   beta= 24286.54
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 72.6098    alpha= 0.0918   beta= 2814.296
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 108.9521   alpha= 0.924    beta= 17370.86
## Number of parameters (weights and biases) to estimate: 74
## Nguyen-Widrow method
## Scaling factor= 0.7000743
## gamma= 72.7981    alpha= 0.3165   beta= 4901.169
## Number of parameters (weights and biases) to estimate: 111
```

```
## Nguyen-Widrow method
## Scaling factor= 0.7001177
## gamma= 109.1208   alpha= 0.2994   beta= 24355.2
## Number of parameters (weights and biases) to estimate: 111
## Nguyen-Widrow method
## Scaling factor= 0.7001059
## gamma= 106.8643   alpha= 0.681    beta= 17124.9
```

And the results

```
# Results
brnnFit
```

```
## Bayesian Regularized Neural Networks
##
## 7262 samples
##    4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 6536, 6537, 6535, 6534, 6537, 6534, ...
## Resampling results across tuning parameters:
##
##   neurons  RMSE      Rsquared   MAE
##   2        62.66769  0.8167093  32.233759
##   3        12.05792  0.9942582   7.889802
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was neurons = 3.
```

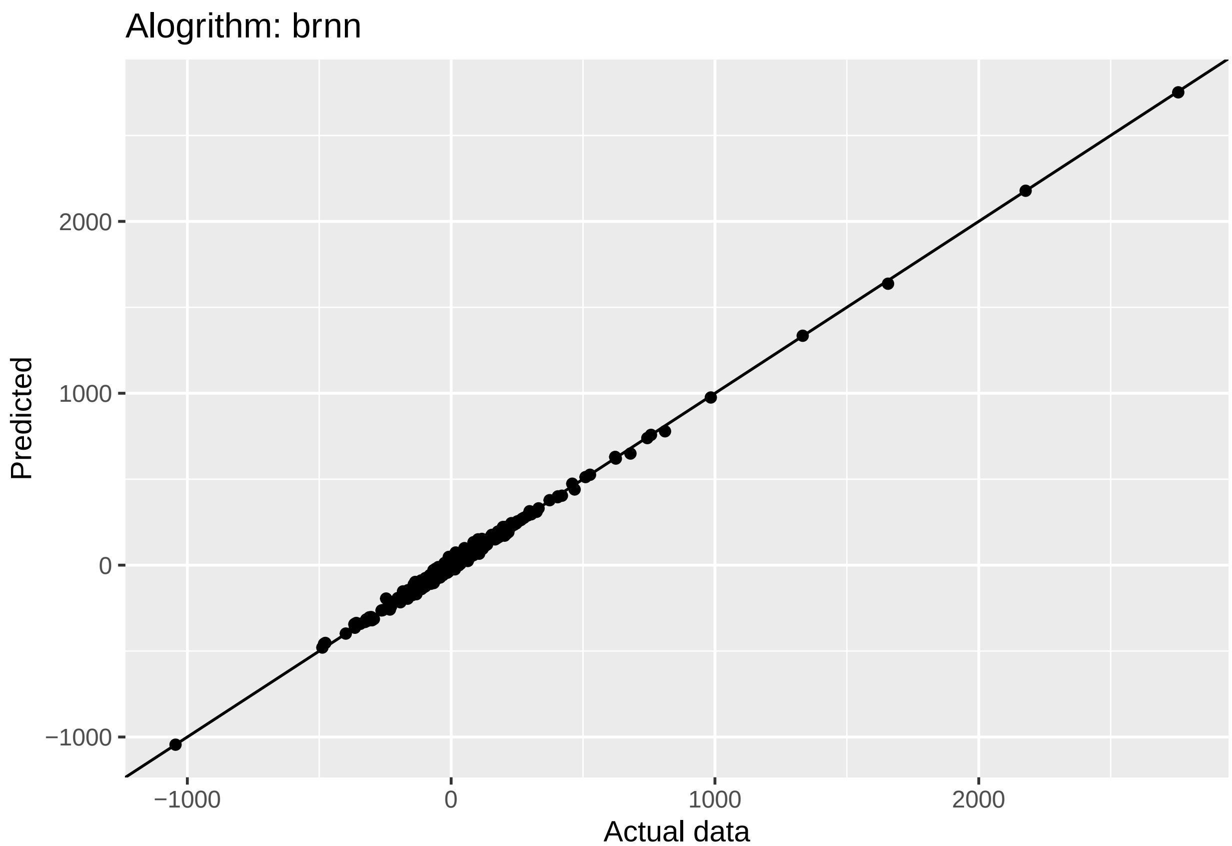
```
# RMSE
brnnPredict <- predict(brnnFit,newdata = test_set)
error_value <- RMSE(test_set$anomaly, brnnPredict)

# Add result to tabe
results_RMSE <- results_RMSE %>%
  rbind(tibble(algorithm = "Bayesian Regularized Neural Network",
               RMSE = error_value))
results_RMSE
```

algorithm	RMSE
linear regression	160.94783
k-nearest neighbors	74.69472
generalized aditive model	178.34719
Regression Tree	90.34400
Bayesian Regularized Neural Network	13.15758

4.6.1 Findings

```
# Plot predicted vs actual values in the train_set
data.frame(x = test_set$anomaly,
```

```r
          y = brnnPredict) %>%
  ggplot(aes(x = x,
             y = y)) +
  geom_point() +
  geom_abline(slope = 1) +
  ggtitle("Alogrithm: brnn") +
  xlab("Actual data") +
  ylab("Predicted")
```




```r
# Error analysis per season
test_set %>%
  cbind(fitted = brnnPredict) %>%
  group_by(season) %>%
  summarize(rmse = RMSE(anomaly, fitted))  %>%
   kbl(booktabs = TRUE,
      caption = "RMSE per season - brnn.") %>%
  kable_styling(latex_options = "hold_position")
```

4.7 Final Validation

Since the Bayesian algorythm had the best performance, we will use it to make the the final validation.

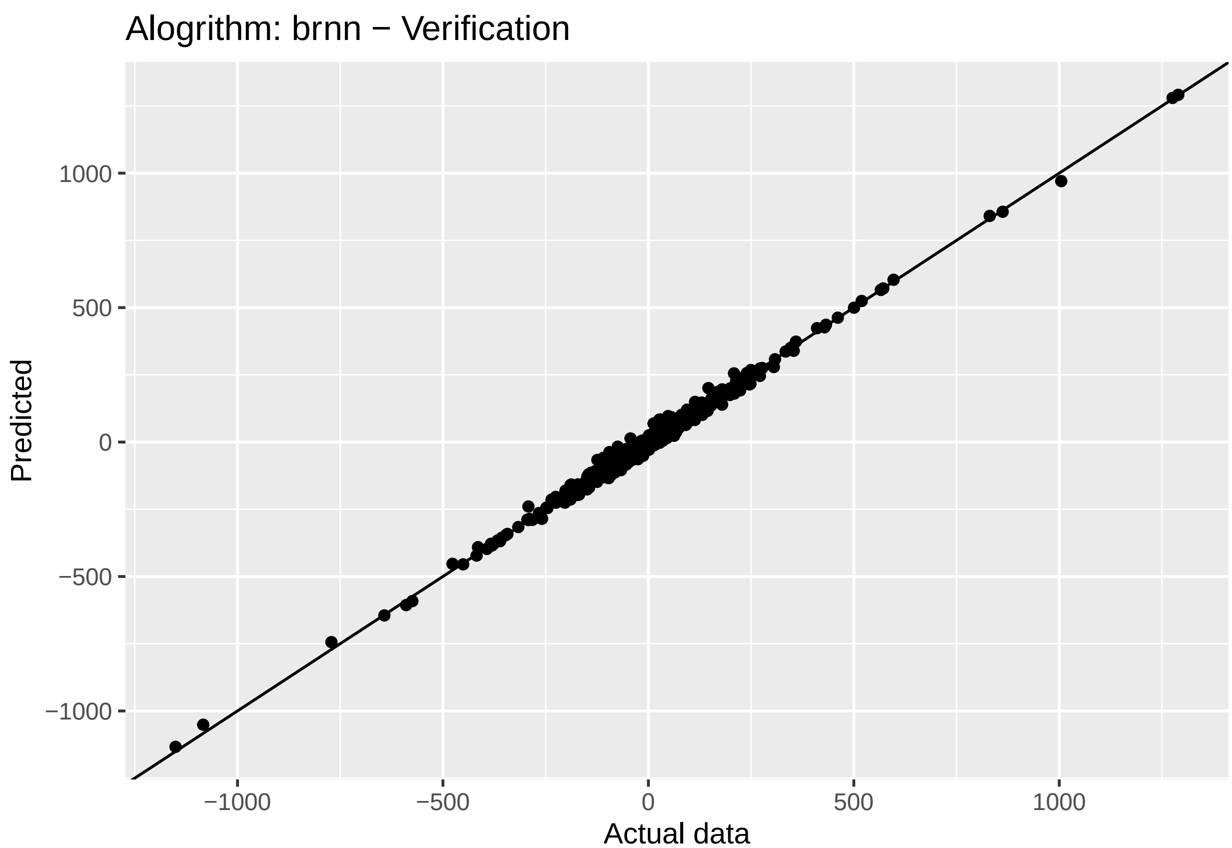
Table 13: RMSE per season - brnn.

season	rmse
Annual	2.040061
Autumn	17.226196
Spring	19.523657
Summer	12.861570
Winter	4.080947

```
# Given brnn is, by far, the minimum RMSE among the chosen algorithms, we
# will run the final verification only for it.

# RMSE
brnnPredict <- predict(brnnFit,newdata = verification)
error_value <- RMSE(verification$anomaly, brnnPredict)

# Add result to tabe
results_RMSE <- results_RMSE %>%
  rbind(tibble(algorithm = "BRNN - Verification",
               RMSE = error_value))
results_RMSE
```

algorithm	RMSE
linear regression	160.94783
k-nearest neighbors	74.69472
generalized aditive model	178.34719
Regression Tree	90.34400
Bayesian Regularized Neural Network	13.15758
BRNN - Verification	13.86329

```
# Plot predicted vs actual values in the train_set
data.frame(x = verification$anomaly,
           y = brnnPredict) %>%
  ggplot(aes(x = x,
             y = y)) +
  geom_point() +
  geom_abline(slope = 1) +
  ggtitle("Alogrithm: brnn - Verification") +
  xlab("Actual data") +
  ylab("Predicted")
```



5 Conclusion

This report describes how machine learning algorithms can be used as a tool to solve a problem. In this case, the idea was to propose a very simple model to help predict the rain anomaly in New Zealand based on historic data of precipitation along the country. Of course, a better model can be build using aditional data such as temperature, for example.

For the purpose of this report, a few algorithms were tested in order to understand how each one works. The chosen algorithms were simple linear regression, k-nearest neighnors, generalized aditive model, regression tree and bayesian regularized neural network. The last one had the best performance against the train data and was checked on the verification set and it seems to work better with different time scales, since most of the algorithms performed better for seasons but did poorly when predicting the early values. The decision to verify only the bayesian algorithm was called since the error criteria chosen to evaluate how good the models performed was so much better for brnn than the other algorithms.

The idea can be used to help prediction rainfall anywhere we wish as long we can gather information on the site of interest.

6 Reference

Macara, G., Nichol, S., Sutherland, D., Liley, B., Paul, V., & Srinivasan, R. (2020). Ministry for the Environment Atmosphere and Climate Report 2020: Updated Datasets supplied by NIWA (NIWA Client Report No. 2020100WN). Retrieved from https://www.mfe.govt.nz/publications/environmental-reporting/ministry-environment-atmosphere-and-climate-report-2020-updated.

Irizarri, R. A., Data Analysis and Prediction Algorithms with R, 2021. Avaiable on https://rafalab.github.io/dsbook/

Witten, I. H., Frank, E., Hall, M. A. & Pal, C.J. Data Mining: Practical Machine Learning Tools and Techniques, 4th Edition, Elsevier, 2016.