Cherry Blossom Prediction Competition

Spencer Retcher

Importing Tidyverse

```
library(tidyverse)
library(caret)
library(openmeteo)
library(readr)
library(kernlab)
```

Measuring Bloom Dates for New York City

The peak bloom dates for Yoshino cherry trees from 2019 - 2023 were determined by using data from the *USA National Phenology Network* and by following this method:

1. The bloom date for each tree was measured as the average date between the day the volunteer first noticed open flowers and the number of days since the prior observation.
   i) If a volunteer noticed open flowers on a certain day, that does not mean that the bloom date was that day. The bloom date could have taken place anytime between these two dates, so this average attempts to minimize potential volunteer error.
2. The peak bloom dates for New York City in 2019-2023 were the earliest bloom dates observed among Yoshino cherry trees during those years.

```
nyc <- read_csv("data/USA-NPN_individual_phenometrics_data.csv") |>
       filter(Site_ID == 32789,Species_ID == 228) |>
       mutate(NumDays_Since_Prior_No = if_else(NumDays_Since_Prior_No == -9999,
                                               0,
                                               NumDays_Since_Prior_No))

nyc_data <- nyc |>
```

```r
  mutate(
    year = First_Yes_Year,
    bloom_date = as.Date(paste(First_Yes_Year, First_Yes_Month, First_Yes_Day, sep = "-"))
    bloom_doy = floor(First_Yes_DOY - (NumDays_Since_Prior_No / 2))
  ) |>
  group_by(year) |>
  summarise(bloom_date = min(bloom_date),
            bloom_doy = min(bloom_doy),
            .groups = "drop") |>
  mutate(
    location = 'newyorkcity',
    lat = 40.73082,
    long = -73.99733,
    alt = 5,
  ) |>
  select(location, lat, long, alt, year, bloom_date, bloom_doy)

nyc_data
```

```
# A tibble: 4 x 7
  location      lat  long   alt  year bloom_date bloom_doy
  <chr>       <dbl> <dbl> <dbl> <dbl> <date>         <dbl>
1 newyorkcity  40.7 -74.0     5  2019 2019-04-05        95
2 newyorkcity  40.7 -74.0     5  2021 2021-04-04        94
3 newyorkcity  40.7 -74.0     5  2022 2022-03-26        85
4 newyorkcity  40.7 -74.0     5  2023 2023-03-26        85
```

Historical Bloom Dates

The tibble `historical_bloom_dates` contains past peak bloom dates and information about the five locations of interest:

- Kyoto (Japan)
- Liestal-Weideli (Switzerland)
- Washington, D.C. (USA)
- Vancouver, BC (Canada)
- New York City, NY (USA)

```r
historical_bloom_dates <- tibble(read_csv("data/washingtondc.csv") |>
  bind_rows(read_csv("data/liestal.csv")) |>
```

```r
  bind_rows(read_csv("data/kyoto.csv")) |>
  bind_rows(read_csv("data/vancouver.csv")) |>
  bind_rows(nyc_data))

historical_bloom_dates
```

```
# A tibble: 1,074 x 7
   location       lat  long   alt  year bloom_date bloom_doy
   <chr>        <dbl> <dbl> <dbl> <dbl> <date>         <dbl>
 1 washingtondc  38.9 -77.0     0  1921 1921-03-20        79
 2 washingtondc  38.9 -77.0     0  1922 1922-04-07        97
 3 washingtondc  38.9 -77.0     0  1923 1923-04-09        99
 4 washingtondc  38.9 -77.0     0  1924 1924-04-13       104
 5 washingtondc  38.9 -77.0     0  1925 1925-03-27        86
 6 washingtondc  38.9 -77.0     0  1926 1926-04-11       101
 7 washingtondc  38.9 -77.0     0  1927 1927-03-20        79
 8 washingtondc  38.9 -77.0     0  1928 1928-04-08        99
 9 washingtondc  38.9 -77.0     0  1929 1929-03-31        90
10 washingtondc  38.9 -77.0     0  1930 1930-04-01        91
# i 1,064 more rows
```

Open-Meteo API

Data for this project was collected from the *Open-Meteo API*, which can be accessed from R by using the library `openmeteo` (API key not required).Using this API, we will retrieve hourly and daily weather conditions from 1940 to the present day for each of our five locations. Later on, we will forecast the same weather conditions 16 days into the future for each of our locations, which we will use to help us make our 2024 peak bloom predictions.

Note: The API can be slow and allows around 10-15 requests per day. I will convert most of these requests to csv files and save them to the folders `data/hourly_data`, `data/daily_data`, and `data/forecast_data` to make the report easier to reproduce.

```r
# Used to get daily weather data for a location
get_daily_data <- function(lat, long, date_min, date_max, elements) {
  weather_history(
    location = c(lat, long),
    start = date_min,
    end = date_max,
    daily = elements
```

```
    )
}

# Used to get hourly weather data for a location
get_hourly_data <- function(lat, long, date_min, date_max, elements) {
  weather_history(
    location = c(lat, long),
    start = date_min,
    end = date_max,
    hourly = elements
    )
}
```

Accumulative Growing Degree Days

Growing degree day A measurement of heat accumulation which can be used to predict the development rates of plants.

`add_gdd()` calculates the growing degree day for each day by using the

1. Daily maximum temperature
2. Daily minimum temperature
3. Base Temperature of 10°C

$$GDD = \frac{(T_{max} + T_{min})}{2} - 10°C$$

If the average daily temperature is less than the base temperature, the growing degree day for that day is zero.

`get_agdd()` filters the daily weather data for a particular year and location and sums up the growing degree days for that year between January 1st to the bloom_date.

```
add_gdd <- function(daily_temp_data) {
  daily_temp_data |>
      mutate(
        year = year(date),
        growing_degree_day =
          ((daily_temperature_2m_max + daily_temperature_2m_min) / 2) - 10
      ) |>
    mutate(growing_degree_day = case_when(growing_degree_day < 0 ~ 0,
                                          .default = growing_degree_day)) |>
```

```r
    relocate(year, .after = date)
}


get_agdd <- function(place, year, bloom_date) {

  data <- daily_data %>%
          filter(location == place)


  date_min = as.Date(paste(year, "-01-01",sep = ""),format = "%Y-%m-%d")

  check_date_range <- data %>%
    filter(date >= date_min & date <= bloom_date)

  if (nrow(check_date_range) == 0) {
    return(NA)
  }


  check_date_range |>
    group_by(year) |>
    summarise(agdd = sum(growing_degree_day, na.rm = TRUE)) |>
    pull(agdd)
}
```

Total Sunshine Duration

Sunshine duration The number of seconds of sunshine per day which is determined by calculating direct normalized irradiance exceeding 120 W/m^2.

`sunshine_to_hours()` converts the sunshine duration of each day into hours.

`get_sunshine()` filters the daily weather data for a particular year and location and sums up the number of hours of sunshine duration for that year between January 1st to the bloom_date.

```r
sunshine_to_hours <- function(sunshine_data) {
    sunshine_data |>
    mutate(daily_sunshine_duration = daily_sunshine_duration / 3600)
}
```

```
get_sunshine <- function(place, year, bloom_date) {

  data <- daily_data %>%
          filter(location == place)

  date_min = as.Date(paste(year, "-01-01",sep = ""),format = "%Y-%m-%d")

  check_date_range <- data %>%
    filter(date >= date_min & date <= bloom_date)

  if (nrow(check_date_range) == 0) {
    return(NA)
  }

  check_date_range |>
    group_by(year) |>
    summarise(sunshine_duration = sum(daily_sunshine_duration, na.rm = TRUE)) |>
    pull(sunshine_duration)
}
```

Total Precipitation

Precipitation The amount of rain, showers, and snowfall for a day measured in millimeters.

`get_precip()` filters the daily weather data for a particular year and location and sums up the amount of precipitation for that year between January 1st to the bloom_date.

```
get_precip <- function(place, year, bloom_date) {

  data <- daily_data %>%
          filter(location == place)

  date_min = as.Date(paste(year, "-01-01",sep = ""),format = "%Y-%m-%d")

  check_date_range <- data %>%
    filter(date >= date_min & date <= bloom_date)

  if (nrow(check_date_range) == 0) {
    return(NA)
  }
```

```
  check_date_range |>
    group_by(year) |>
    summarise(total_precipitation = sum(daily_precipitation_sum, na.rm = TRUE)) |>
    pull(total_precipitation)

}
```

Chill Hours

Chill hours The number of hours in the winter that a plant spends exposed to certain temperatures.

For this project, chill hours is the number of hours between 0°C to 7°C from November to February.

`get_chills()` filters the hourly weather data for a particular year and location and sums up the number of chill hours in the winter for that year between November to Feburary.

```
add_chill <- function(data) {
  data |>
    mutate(
      chill_hour = case_when(
        hourly_temperature_2m >= 0 & hourly_temperature_2m <= 7 ~ 1,
        .default = 0
      ),
      date = as.Date(datetime),
      year = as.integer(format(datetime, "%Y")),
      month = as.integer(strftime(date, '%m')) %% 12,
      # make December "0"
      year = if_else(month == 0 | month == 11, year + 1L, year)
    ) |>
    filter(month %in% c(11, 0, 1, 2)) |>
    group_by(year, location) |>
    summarize(chill_hours = sum(chill_hour))
}



get_chill <- function(place, chill_year) {
```

```
  data <- hourly_data %>%
          filter(location == place)

  check_date_range <- data %>%
    filter(year == chill_year)

  if (nrow(check_date_range) == 0) {
    return(NA)
  }

    data |>
    filter(year == chill_year) |>
    pull(chill_hours)
}
```

Forecasting Weather Data

The *Open-Meteo API* can forecast the daily weather conditions stated above 16 days into the future. These forecasts were downloaded for each location from https://open-meteo.com/ and stored in csv files.

We do not need to forecast hourly temperature data to calculate this winter's chill hours since the competition ends on the last day of winter and we already have a good estimate of the chill hours.

```
washingtondc_forecast_daily_data <- read_csv("data/forecast_data/open-meteo-38.89N77.03W1m
  rename(date = time,
         daily_temperature_2m_max = `temperature_2m_max (°C)`,
         daily_temperature_2m_min = `temperature_2m_min (°C)`,
         daily_sunshine_duration = `sunshine_duration (s)`,
         daily_precipitation_sum = `precipitation_sum (mm)`) |>
  mutate(location = "washingtondc")


liestal_forecast_daily_data <- read_csv("data/forecast_data/open-meteo-47.48N7.74E344m.csv
  rename(date = time,
         daily_temperature_2m_max = `temperature_2m_max (°C)`,
         daily_temperature_2m_min = `temperature_2m_min (°C)`,
         daily_sunshine_duration = `sunshine_duration (s)`,
         daily_precipitation_sum = `precipitation_sum (mm)`) |>
```

```
  mutate(location = "liestal")



kyoto_forecast_daily_data <- read_csv("data/forecast_data/open-meteo-35.00N135.69E31m.csv"
  rename(date = time,
         daily_temperature_2m_max = `temperature_2m_max (°C)`,
         daily_temperature_2m_min = `temperature_2m_min (°C)`,
         daily_sunshine_duration = `sunshine_duration (s)`,
         daily_precipitation_sum = `precipitation_sum (mm)`) |>
  mutate(location = "kyoto")


vancouver_forecast_daily_data <- read_csv("data/forecast_data/open-meteo-49.23N123.18W28m.
  rename(date = time,
         daily_temperature_2m_max = `temperature_2m_max (°C)`,
         daily_temperature_2m_min = `temperature_2m_min (°C)`,
         daily_sunshine_duration = `sunshine_duration (s)`,
         daily_precipitation_sum = `precipitation_sum (mm)`) |>
  mutate(location = "vancouver")



newyorkcity_forecast_daily_data <- read_csv("data/forecast_data/open-meteo-40.74N73.98W14m
  rename(date = time,
         daily_temperature_2m_max = `temperature_2m_max (°C)`,
         daily_temperature_2m_min = `temperature_2m_min (°C)`,
         daily_sunshine_duration = `sunshine_duration (s)`,
         daily_precipitation_sum = `precipitation_sum (mm)`) |>
  mutate(location = "newyorkcity")

bind_rows(washingtondc_forecast_daily_data,
          liestal_forecast_daily_data,
          kyoto_forecast_daily_data,
          vancouver_forecast_daily_data,
          newyorkcity_forecast_daily_data)
```

```
# A tibble: 80 x 6
   date       daily_temperature_2m_max daily_temperature_2m_min
   <date>                        <dbl>                    <dbl>
 1 2024-02-29                     14.6                     -0.3
 2 2024-03-01                     11.1                      0
```

```
 3 2024-03-02                     9.4                       6.5
 4 2024-03-03                    16.5                       6.4
 5 2024-03-04                    16.8                      10.2
 6 2024-03-05                    13.9                       8.3
 7 2024-03-06                    14.5                      10
 8 2024-03-07                    20                        12.7
 9 2024-03-08                    14.4                       6.8
10 2024-03-09                    10.9                       7.3
# i 70 more rows
# i 3 more variables: daily_sunshine_duration <dbl>,
#   daily_precipitation_sum <dbl>, location <chr>
```

Combining and Transforming Daily Weather Data and Forecast Data

1. The *Open-Meteo API* was used to retrieve the daily maximum temperature, daily minimum temperature, daily sunshine duration, and daily precipitation for all days from January 1, 1940 to Yesterday for each location.
2. All of the daily weather data was joined with the 16 day weather forecast for each location and placed into `daily_data`.
   i) This means if we submit our project on the last day of the competition, we will have the daily weather data from January 1, 1940 to March 15, 2024.
3. `daily_data` is then transformed.
   i) The sunshine duration for each day is converted to hours.
   ii) The growing degree days are added for each day.

```
# washingtondc_daily_data <- get_daily_data(lat = 38.8853,
#                                            long = -77.0386,
#                                            date_min = "1940-01-01",
#                                            date_max = Sys.Date() - 1,
#                                            elements = c("temperature_2m_max","temperature
#                                            mutate(location = "washingtondc")
# write.csv(washingtondc_daily_data, file = "washingtondc_daily_data.csv", row.names = TRU

washingtondc_daily_data <- read_csv("data/daily_data/washingtondc_daily_data.csv")
```

```
# liestal_daily_data <- get_daily_data(lat = 47.4814,
#                                       long =7.730519 ,
```

```
#                                       date_min = "1940-01-01",
#                                       date_max = Sys.Date() - 1,
#                                       elements = c("temperature_2m_max","temperature_2m_m
#                                       mutate(location = "liestal")
# write.csv(liestal_daily_data, file = "liestal_daily_data.csv", row.names = TRUE)

liestal_daily_data <- read_csv("data/daily_data/liestal_daily_data.csv")
```

```
# kyoto_daily_data <- get_daily_data(lat = 35.0120,
#                                     long = 135.6761,
#                                     date_min = "1940-01-01",
#                                     date_max = Sys.Date() - 1,
#                                     elements = c("temperature_2m_max","temperature_2m_min
#                                     mutate(location = "kyoto")
# write.csv(kyoto_daily_data, file = "kyoto_daily_data.csv", row.names = TRUE)

kyoto_daily_data <- read_csv("data/daily_data/kyoto_daily_data.csv")
```

```
#
# vancouver_daily_data <- get_daily_data(lat = 49.2237,
#                                         long = -123.1636,
#                                         date_min = "1940-01-01",
#                                         date_max = Sys.Date() - 1,
#                                         elements = c("temperature_2m_max","temperature_2m
#                                         mutate(location = "vancouver")
# write.csv(vancouver_daily_data, file = "vancouver_daily_data.csv", row.names = TRUE)

vancouver_daily_data <- read_csv("data/daily_data/vancouver_daily_data.csv")
```

```
#
# newyorkcity_daily_data <- get_daily_data(lat = 40.73040 ,
#                                           long = -73.99809,
#                                           date_min = "1940-01-01",
#                                           date_max = Sys.Date() - 1,
#                                           elements = c("temperature_2m_max","temperature_
#                                           mutate(location = "newyorkcity")

# write.csv(newyorkcity_daily_data, file = "newyorkcity_daily_data.csv", row.names = TRUE)

newyorkcity_daily_data <- read_csv("data/daily_data/newyorkcity_daily_data.csv")
```

```r
daily_data = bind_rows(washingtondc_daily_data, liestal_daily_data, kyoto_daily_data, vanc
  add_gdd() |>
  sunshine_to_hours()

daily_data
```

```
# A tibble: 153,775 x 9
    ...1 date        year daily_temperature_2m_max daily_temperature_2m_min
   <dbl> <date>     <dbl>                    <dbl>                    <dbl>
 1     1 1940-01-01  1940                     -2.3                     -7.6
 2     2 1940-01-02  1940                     -3.3                     -8.1
 3     3 1940-01-03  1940                     -2                       -8.8
 4     4 1940-01-04  1940                     -0.2                     -7.5
 5     5 1940-01-05  1940                      6.3                     -7.3
 6     6 1940-01-06  1940                     -1.2                     -8.1
 7     7 1940-01-07  1940                     -0.8                     -7.9
 8     8 1940-01-08  1940                      2.1                     -2.6
 9     9 1940-01-09  1940                      0.1                     -5.7
10    10 1940-01-10  1940                      1.5                     -7
# i 153,765 more rows
# i 4 more variables: daily_sunshine_duration <dbl>,
#   daily_precipitation_sum <dbl>, location <chr>, growing_degree_day <dbl>
```

Hourly Weather Data

The *Open-Meteo API* was used to retrieve the hourly temperatures for all days from January 1, 1940 to Yesterday for each location.

These hourly temperatures were joined together, placed into `hourly_data`, and than the chill hours for each hour were calculated.

```r
#
# washingtondc_hourly_data <- get_hourly_data(lat = 38.8853,
#                                             long = -77.0386,
#                                             date_min = "1940-01-01",
#                                             date_max = Sys.Date()- 1,
#                                             elements = c("temperature_2m"))  %>%
#                                             mutate(location = "washingtondc")
# write.csv(washingtondc_hourly_data, file = "washingtondc_hourly_data.csv", row.names = T
```

```r
washingtondc_hourly_data <- read_csv("data/hourly_data/washingtondc_hourly_data.csv")
```

```r
#
# liestal_hourly_data <- get_hourly_data(lat = 47.4814,
#                                         long =7.730519 ,
#                                         date_min = "1940-01-01",
#                                         date_max = Sys.Date() - 1,
#                                         elements = c("temperature_2m"))  %>%
#                                         mutate(location = "liestal")

# write.csv(liestal_hourly_data, file = "liestal_hourly_data.csv", row.names = TRUE)

liestal_hourly_data <- read_csv("data/hourly_data/liestal_hourly_data.csv")
```

```r
#
# kyoto_hourly_data <- get_hourly_data(lat = 35.0120,
#                                       long = 135.6761,
#                                       date_min = "1940-01-01",
#                                       date_max = Sys.Date() - 1,
#                                       elements = c("temperature_2m" ))  %>%
#                                       mutate(location = "kyoto")
# write.csv(kyoto_hourly_data, file = "kyoto_hourly_data.csv", row.names = TRUE)

kyoto_hourly_data <- read_csv("data/hourly_data/kyoto_hourly_data.csv")
```

```r
#
# vancouver_hourly_data <- get_hourly_data(lat = 49.2237 ,
#                                           long = -123.1636 ,
#                                           date_min = "1940-01-01",
#                                           date_max = Sys.Date() - 1,
#                                           elements = c("temperature_2m" ))  %>%
#                                           mutate(location = "vancouver")
# write.csv(vancouver_hourly_data, file = "vancouver_hourly_data.csv", row.names = TRUE)

vancouver_hourly_data<- read_csv("data/hourly_data/vancouver_hourly_data.csv")
```

```r
#
# newyorkcity_hourly_data <- get_hourly_data(lat = 40.73040 ,
#                                             long = -73.99809,
#                                             date_min = "1940-01-01",
```

```
#                                            date_max = Sys.Date() - 1,
#                                            elements = c("temperature_2m" )) %>%
#                                            mutate(location = "newyorkcity")
# write.csv(newyorkcity_hourly_data, file = "newyorkcity_hourly_data.csv", row.names = TRU

newyorkcity_hourly_data <- read_csv("data/hourly_data/newyorkcity_hourly_data.csv")
```

```
hourly_data = bind_rows(washingtondc_hourly_data,liestal_hourly_data,kyoto_hourly_data,van
  add_chill()
```

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

```
hourly_data
```

```
# A tibble: 425 x 3
# Groups:   year [85]
    year location     chill_hours
   <int> <chr>              <dbl>
 1  1940 kyoto                785
 2  1940 liestal              444
 3  1940 newyorkcity          417
 4  1940 vancouver           1178
 5  1940 washingtondc         520
 6  1941 kyoto               1380
 7  1941 liestal             1283
 8  1941 newyorkcity         1370
 9  1941 vancouver           2272
10  1941 washingtondc        1422
# i 415 more rows
```

Creating the Complete Dataset

We have all the daily and hourly weather data for each of our locations from January 1, 1940 to March 15, 2024.

We then use `rowise()` and `mutate()` to essentially take each row of data in `historical_bloom_dates` and compute the aggregations for that year.

For each year and location in `historical_bloom_dates`, we calculate:

1. the number of chill hours between November and February.
2. the number of sunshine hours between January 1st to the bloom_date.
3. the number of growing degree days between January 1st to the bloom_date.
4. the amount of precipitation between January 1st to the bloom_date.

```r
complete_dataset <- historical_bloom_dates %>%
  filter(year >=1940) |>
  rowwise() |>
  mutate(chill_hours = get_chill(location,year),
         accumulative_growing_degree_days = get_agdd(location, year, bloom_date),
         total_sunshine_duration = get_sunshine(location,year,bloom_date),
         total_precipitation = get_precip(location, year, bloom_date)
         )

complete_dataset
```

```
# A tibble: 257 x 11
# Rowwise:
   location       lat  long   alt  year bloom_date bloom_doy chill_hours
   <chr>        <dbl> <dbl> <dbl> <dbl> <date>         <dbl>       <dbl>
 1 washingtondc  38.9 -77.0     0  1940 1940-04-13       104         520
 2 washingtondc  38.9 -77.0     0  1941 1941-04-12       102        1422
 3 washingtondc  38.9 -77.0     0  1942 1942-04-05        95        1321
 4 washingtondc  38.9 -77.0     0  1943 1943-04-04        94        1384
 5 washingtondc  38.9 -77.0     0  1944 1944-04-09       100        1419
 6 washingtondc  38.9 -77.0     0  1945 1945-03-20        79        1381
 7 washingtondc  38.9 -77.0     0  1946 1946-03-23        82        1270
 8 washingtondc  38.9 -77.0     0  1947 1947-04-12       102        1246
 9 washingtondc  38.9 -77.0     0  1948 1948-03-28        88        1258
10 washingtondc  38.9 -77.0     0  1949 1949-03-29        88        1281
# i 247 more rows
# i 3 more variables: accumulative_growing_degree_days <dbl>,
#   total_sunshine_duration <dbl>, total_precipitation <dbl>
```

Bayesian Regularized Neural Network

We did a 70/30 split of `complete_dataset` and stored the observations into a training set and test_set respectively. We trained the model using the training set and we will test how our model performs on unseen data with the test set. We will also use five fold cross-validation to tune any hyperparameters.

A bayesian regularized neural network was used to predict bloom_doy based on location, year, chill_hours, accumulative_growing_degree_days, total_sunshine_duration, and total_precipitation. The model with two neurons was chosen as the optimal model due to the low RMSE value achieved during cross-validation.

The test R-squared was approximately 0.884 which indicates that 88.4% of the variation in bloom dates can be explained by our model.

The test RMSE was about 2.7 which indicates that on average the bloom date predictions were off by 2.7 days.

```r
# split data into a training set(70%) and a test set(30%)
set.seed(123)
index <- createDataPartition(y = complete_dataset$bloom_doy,p = .70, list = FALSE)
training_set <- complete_dataset[index, ]
test_set <- complete_dataset[-index, ]





# model will perform five fold cross-validation five times
five_fold_cross_validation <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
```

```r
brnn_model <- train(bloom_doy ~ location + year + chill_hours +
                      accumulative_growing_degree_days + total_sunshine_duration +
                      total_precipitation, data = training_set, method = "brnn",
                      trControl = five_fold_cross_validation)
```

```r
print(brnn_model)
```

```
Bayesian Regularized Neural Networks

182 samples
  6 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 146, 146, 145, 145, 146, 146, ...
Resampling results across tuning parameters:

  neurons  RMSE      Rsquared   MAE
```

```
  1        3.954411  0.8113908  3.201479
  2        3.840946  0.8198470  3.075145
  3        3.958073  0.8101347  3.148061

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was neurons = 2.
```

```r
test_set_predictions <- predict(brnn_model, newdata = test_set)

test_set_performance <- postResample(pred = test_set_predictions, obs = test_set$bloom_doy

cat("\nTest Set Performance\n")
```

```
Test Set Performance
```

```r
test_set_performance
```

```
     RMSE  Rsquared       MAE
2.7024147 0.8844012 2.0381902
```

2024 Predictions

We used the same technique we used with `complete_dataset` to generate the daily and hourly weather aggregations for 2024 for each location. We then make predictions using our model trained on `complete_dataset` and output our predictions to a csv file.

```r
brnn_model_all_data <- train(bloom_doy ~ location + year + chill_hours +
                      accumulative_growing_degree_days + total_sunshine_duration +
                      total_precipitation, data = complete_dataset, method = "brnn",
                      trControl = five_fold_cross_validation)
```

```
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3728   alpha= 1.854    beta= 22.1614
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
```

```
Scaling factor= 0.7023708
gamma= 18.1144   alpha= 2.0888   beta= 26.2571
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 24.0938   alpha= 2.4621   beta= 27.4828
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3987   alpha= 1.9438   beta= 22.7664
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023479
gamma= 17.9342   alpha= 2.3438   beta= 26.6186
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.703725
gamma= 22.7067   alpha= 2.7762   beta= 26.841
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.429    alpha= 1.866    beta= 23.5163
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 18.474    alpha= 2.214    beta= 27.0853
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 24.2031   alpha= 2.1957   beta= 27.9005
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3166   alpha= 1.6751   beta= 19.9288
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 17.9334   alpha= 2.1029   beta= 23.4232
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 23.2342   alpha= 2.4919   beta= 23.638
Number of parameters (weights and biases) to estimate: 11
```

```
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3705   alpha= 1.6779   beta= 19.6764
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 16.6116   alpha= 2.3655   beta= 20.9063
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 20.9813   alpha= 2.4208   beta= 21.2811
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3849   alpha= 2.0358   beta= 23.0616
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 16.6242   alpha= 2.6332   beta= 25.3296
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 22.2348   alpha= 2.7472   beta= 26.0427
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.2644   alpha= 1.7561   beta= 21.6585
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.7155   alpha= 2.3504   beta= 24.1794
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 22.3694   alpha= 2.7348   beta= 24.382
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3969   alpha= 1.8382   beta= 20.088
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 18.2161   alpha= 1.5008   beta= 23.4191
```

```
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 23.712    alpha= 2.2235   beta= 24.4075
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.4267   alpha= 1.7753   beta= 24.0614
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.7575   alpha= 2.364    beta= 26.9343
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 22.4748   alpha= 2.5832   beta= 27.2535
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3477   alpha= 1.7138   beta= 22.2494
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.9102   alpha= 2.1269   beta= 26.9819
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 22.8616   alpha= 2.4912   beta= 27.1781
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.4892   alpha= 1.7253   beta= 22.151
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.881    alpha= 2.0044   beta= 23.1886
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 24.2975   alpha= 1.8837   beta= 24.4613
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
```

```
gamma= 10.3206   alpha= 1.8446   beta= 22.9951
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.2342   alpha= 2.3401   beta= 25.5086
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 22.6233   alpha= 2.4742   beta= 26.3865
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3115   alpha= 1.7504   beta= 22.4591
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 18.0597   alpha= 2.1556   beta= 26.4831
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 22.6076   alpha= 2.6723   beta= 27.0575
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3632   alpha= 2.0044   beta= 22.1233
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 16.9528   alpha= 2.376    beta= 25.6405
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 21.1642   alpha= 2.7326   beta= 25.6417
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.2922   alpha= 1.7832   beta= 20.8722
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 17.9548   alpha= 2.155    beta= 24.2104
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
```

```
Scaling factor= 0.7037614
gamma= 23.492    alpha= 2.3218   beta= 24.979
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3423   alpha= 1.7347   beta= 20.3603
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.5189   alpha= 2.2822   beta= 22.4593
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 22.8624   alpha= 2.38     beta= 23.0011
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3833   alpha= 1.8915   beta= 21.4346
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.2563   alpha= 2.1696   beta= 24.8405
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 23.9223   alpha= 2.2661   beta= 25.963
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3572   alpha= 1.8147   beta= 23.5775
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 17.821    alpha= 2.1901   beta= 28.094
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 23.0247   alpha= 2.6686   beta= 28.7677
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3775   alpha= 1.8395   beta= 23.6055
Number of parameters (weights and biases) to estimate: 22
```

```
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.7243   alpha= 2.2256   beta= 26.803
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 23.0467   alpha= 2.5506   beta= 27.241
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3534   alpha= 1.827    beta= 21.6844
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 17.4331   alpha= 2.3369   beta= 24.2312
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 22.3494   alpha= 2.6837   beta= 24.6244
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3589   alpha= 1.818    beta= 23.5589
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.7751   alpha= 2.2881   beta= 26.5833
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 22.4586   alpha= 2.6872   beta= 26.6256
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.4268   alpha= 1.8993   beta= 22.7685
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.9981   alpha= 2.2904   beta= 26.7874
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 23.6325   alpha= 2.5411   beta= 27.9419
```

```
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3759   alpha= 1.7456   beta= 20.7865
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 17.3721   alpha= 2.2502   beta= 22.4702
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 21.939    alpha= 2.5026   beta= 22.7192
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3049   alpha= 1.8576   beta= 21.969
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023593
gamma= 17.31     alpha= 2.3922   beta= 25.5431
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037431
gamma= 21.5337   alpha= 2.7081   beta= 25.6131
Number of parameters (weights and biases) to estimate: 11
Nguyen-Widrow method
Scaling factor= 0.7
gamma= 10.3622   alpha= 1.7628   beta= 21.2871
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7023708
gamma= 18.2868   alpha= 2.0831   beta= 24.8191
Number of parameters (weights and biases) to estimate: 33
Nguyen-Widrow method
Scaling factor= 0.7037614
gamma= 23.1982   alpha= 2.468    beta= 25.238
Number of parameters (weights and biases) to estimate: 22
Nguyen-Widrow method
Scaling factor= 0.7018905
gamma= 18.0881   alpha= 2.2963   beta= 25.7842
```

```r
predictions_2024 <-
  tibble(
    location = c("washingtondc", "liestal", "kyoto", "vancouver", "newyorkcity"),
    year = 2024
  ) |>
  rowwise() |>
  mutate(
    bloom_date = Sys.Date() + 15,
    chill_hours = get_chill(location, year),
    accumulative_growing_degree_days = get_agdd(location, year, bloom_date),
    total_sunshine_duration = get_sunshine(location, year, bloom_date),
    total_precipitation = get_precip(location, year, bloom_date)
  )

predictions_2024 <- predictions_2024 |>
  dplyr::select(-bloom_date) |>
  bind_cols(predict(brnn_model_all_data, newdata = predictions_2024)) %>%
  rename(prediction = `...7`) |>
  mutate(prediction = round(prediction)) |>
  dplyr::select(location, prediction)
```

```
New names:
* `` -> `...7`
```

```r
  predictions_2024
```

```
# A tibble: 5 x 2
# Rowwise:
  location     prediction
  <chr>             <dbl>
1 washingtondc         77
2 liestal              76
3 kyoto                83
4 vancouver            84
5 newyorkcity          76
```

```r
write.csv(predictions_2024, file = "cherry-predictions.csv", row.names = FALSE)
```