Vancouver Weather Station Data

Data range: April 20, 1998, up until June 26, 2023

Dinara Salikhadenova

7-4-2023

Contents

Introduction

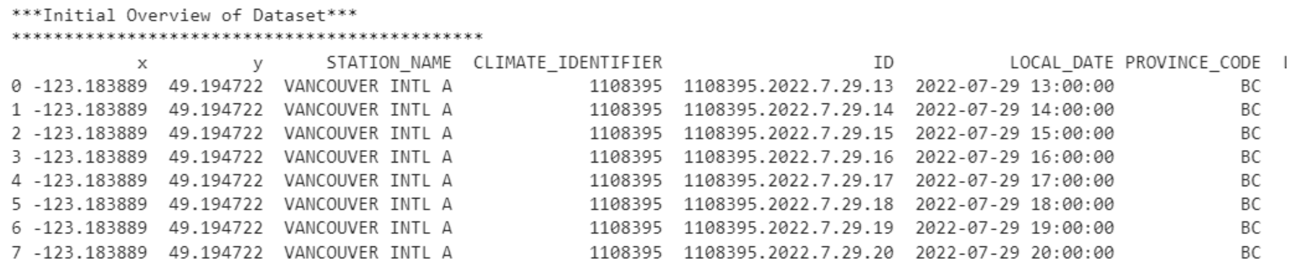
For this assignment I obtained climate data from the Government of Canada. Data can be extracted using a online tool on the following website: https://climate-change.canada.ca/climate-data/#/hourly-climate-data.

In particular, I have selected weather data from the weather station located in Richmond BC, at Vancouver International Airport. Station name is Vancouver Intl A, ID 1108395 and Vancouver Int'l, ID1108447. Data is presented in hourly format and includes data from April 20, 1998, up until June 26, 2023.

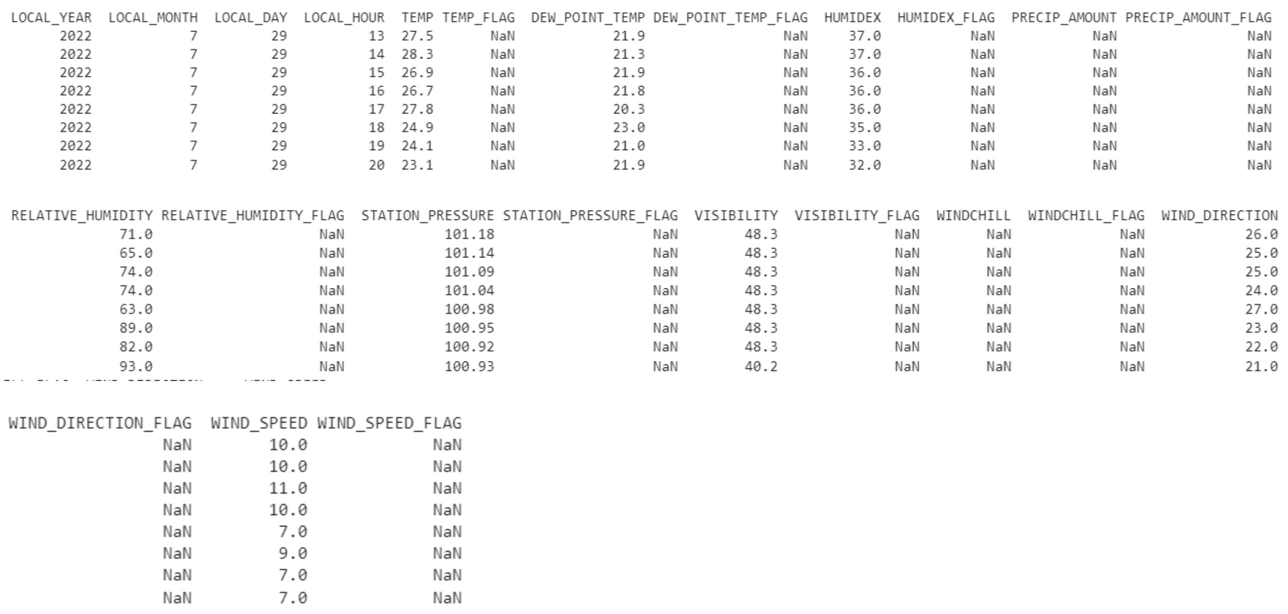
Using this data for the past ~25 years, I will attempt to build a model that will predict mean daily temperature 10 days into the future.

Data Exploration

Weather data is available in csv format. It is hourly data meaning that weather observations are updated every hour. Each csv file contains 10,000 rows of observations. I combined a total of 23 files into one dataset. Here is the initial overview.



```
***Initial Overview of Dataset***
*********************************************
            x          y      STATION_NAME  CLIMATE_IDENTIFIER                    ID           LOCAL_DATE PROVINCE_CODE  I
0 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.13  2022-07-29 13:00:00            BC
1 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.14  2022-07-29 14:00:00            BC
2 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.15  2022-07-29 15:00:00            BC
3 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.16  2022-07-29 16:00:00            BC
4 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.17  2022-07-29 17:00:00            BC
5 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.18  2022-07-29 18:00:00            BC
6 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.19  2022-07-29 19:00:00            BC
7 -123.183889  49.194722  VANCOUVER INTL A             1108395  1108395.2022.7.29.20  2022-07-29 20:00:00            BC
```

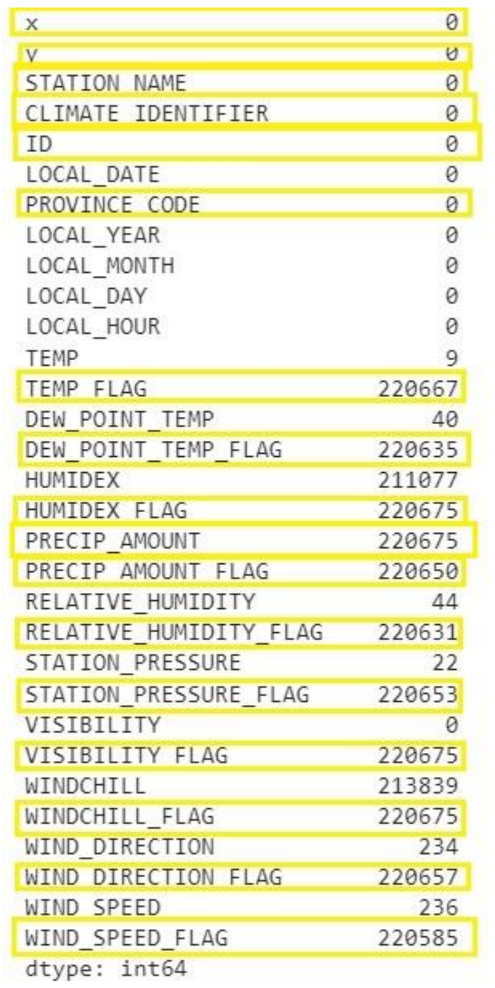


```
LOCAL_YEAR  LOCAL_MONTH  LOCAL_DAY  LOCAL_HOUR  TEMP TEMP_FLAG  DEW_POINT_TEMP DEW_POINT_TEMP_FLAG  HUMIDEX  HUMIDEX_FLAG  PRECIP_AMOUNT PRECIP_AMOUNT_FLAG
      2022            7         29          13  27.5       NaN            21.9                 NaN     37.0           NaN            NaN                NaN
      2022            7         29          14  28.3       NaN            21.3                 NaN     37.0           NaN            NaN                NaN
      2022            7         29          15  26.9       NaN            21.9                 NaN     36.0           NaN            NaN                NaN
      2022            7         29          16  26.7       NaN            21.8                 NaN     36.0           NaN            NaN                NaN
      2022            7         29          17  27.8       NaN            20.3                 NaN     36.0           NaN            NaN                NaN
      2022            7         29          18  24.9       NaN            23.0                 NaN     35.0           NaN            NaN                NaN
      2022            7         29          19  24.1       NaN            21.0                 NaN     33.0           NaN            NaN                NaN
      2022            7         29          20  23.1       NaN            21.9                 NaN     32.0           NaN            NaN                NaN
```

```
RELATIVE_HUMIDITY RELATIVE_HUMIDITY_FLAG  STATION_PRESSURE STATION_PRESSURE_FLAG  VISIBILITY  VISIBILITY_FLAG  WINDCHILL  WINDCHILL_FLAG  WIND_DIRECTION
             71.0                    NaN            101.18                   NaN        48.3              NaN        NaN             NaN            26.0
             65.0                    NaN            101.14                   NaN        48.3              NaN        NaN             NaN            25.0
             74.0                    NaN            101.09                   NaN        48.3              NaN        NaN             NaN            25.0
             74.0                    NaN            101.04                   NaN        48.3              NaN        NaN             NaN            24.0
             63.0                    NaN            100.98                   NaN        48.3              NaN        NaN             NaN            27.0
             89.0                    NaN            100.95                   NaN        48.3              NaN        NaN             NaN            23.0
             82.0                    NaN            100.92                   NaN        48.3              NaN        NaN             NaN            22.0
             93.0                    NaN            100.93                   NaN        40.2              NaN        NaN             NaN            21.0
```

```
WIND_DIRECTION_FLAG  WIND_SPEED WIND_SPEED_FLAG
                NaN        10.0             NaN
                NaN        10.0             NaN
                NaN        11.0             NaN
                NaN        10.0             NaN
                NaN         7.0             NaN
                NaN         9.0             NaN
                NaN         7.0             NaN
                NaN         7.0             NaN
```

There are 31 columns and 220,675 rows of data. Quiet a few of the columns have missing values:



```
x                              0
y                              0
STATION_NAME                   0
CLIMATE_IDENTIFIER             0
ID                             0
LOCAL_DATE                     0
PROVINCE_CODE                  0
LOCAL_YEAR                     0
LOCAL_MONTH                    0
LOCAL_DAY                      0
LOCAL_HOUR                     0
TEMP                           9
TEMP_FLAG                 220667
DEW_POINT_TEMP                40
DEW_POINT_TEMP_FLAG       220635
HUMIDEX                   211077
HUMIDEX_FLAG              220675
PRECIP_AMOUNT             220675
PRECIP_AMOUNT_FLAG        220650
RELATIVE_HUMIDITY             44
RELATIVE_HUMIDITY_FLAG    220631
STATION_PRESSURE              22
STATION_PRESSURE_FLAG     220653
VISIBILITY                     0
VISIBILITY_FLAG           220675
WINDCHILL                 213839
WINDCHILL_FLAG            220675
WIND_DIRECTION               234
WIND_DIRECTION_FLAG       220657
WIND_SPEED                   236
WIND_SPEED_FLAG           220585
dtype: int64
```

Some of the columns have repeating data such as x and y – coordinates of the weather station, station name, and identifiers, province code. Because my data is coming from one source only, these columns would be irrelevant, so I delete them. "Flag" columns have too many missing values, some of them are completely empty, hence I delete them too. "PRECIP_AMOUNT" columns is empty, so it contains no useful information, this columns is dropped from the dataset too. "HUMIDEX" and "WINDCHILL" columns have 9598 and 6836 observations respectively. The standard deviation is not very high for both of these features, so I keep them in the dataset, as they might prove to be useful. Humidex is a measure of how hot we feel, it is used to express how the combined effects of warm temperatures and humidity are perceived.

My dataset contains about 25 years of weather data. First, I want to see whether there have been any abnormal spikes in average temperatures in that period. First, I look at yearly data. Below are statistical process control charts for Average Temperatures (aggregated by Year), Maximum Temperature, and Minimum temperature. I want to see whether there was an unusually cold or warm year, or whether some days had unusually cold or warm days.

Chart 1.

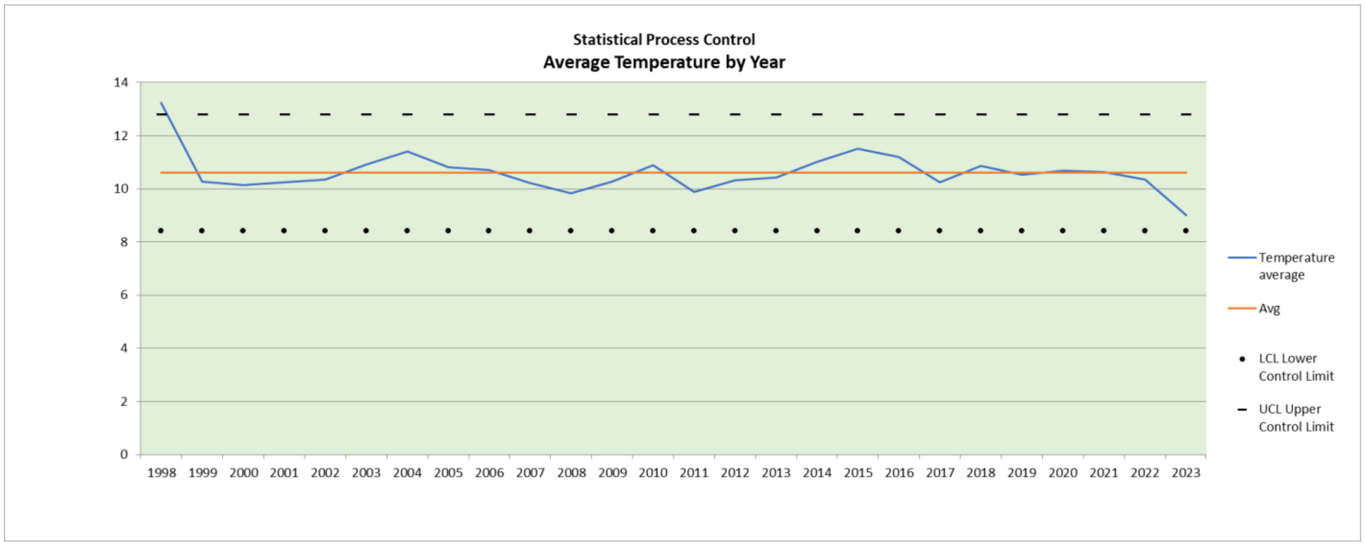


Chart 2.

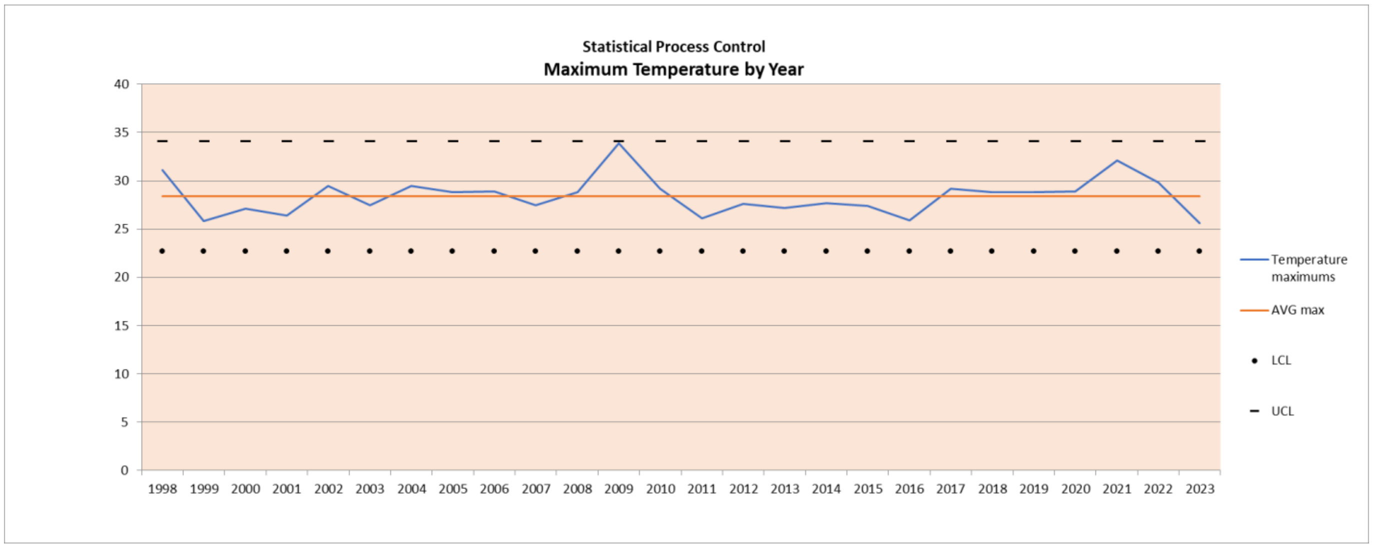
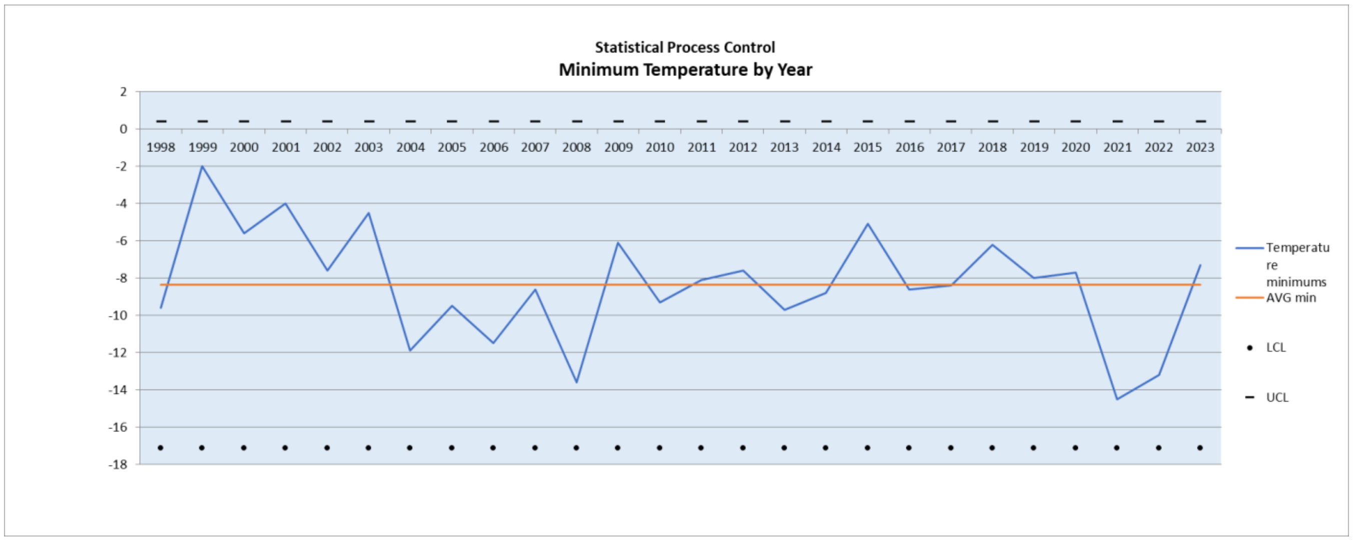


Chart 3.



In all three of the above charts, blue line shows temperatures, red line is the average. LCL – lower control limit is equal to Average – 3*(Standard Deviation), and UCL - upper control limit is equal to Average + 3*(Standard Deviation). In Chart 1 we see that average temperature year by year has remained within normal borders with no unusual spikes and falls. It does appear like in 1998 temperatures were unusually high, but this can be disregarded since we only have data from April of 1998 which makes the average temperature for that year appear abnormally high. The average temperature for 2023 seems low and approaching LCL, but 2023 data is also only partial.

Maximum temperature in 1998 is right at the upper control limit. According to the weather station at Vancouver International Airport in 1998 there was an unusually hot day. Interestingly, this dataset does not reflect unusually hot days we experience in June of 2021. This weather station is located very close to the ocean, so perhaps this influenced the observations at this weather station. Minimal temperatures remained within the 6 standard deviations range.

From the charts below we can observe no sudden spikes in temperatures when we aggregate the data by month. We no longer see a spike in the maximum temperatures in Chart 5 as we did in Chart 2. This is explained by the fact that we now look at a higher number of observations (monthly vs yearly) and hence the standard deviation and LCL and UCL are different, making range of normal observations wider. We can now see maximum temperatures during winter months, not just maximum in a year. This brings the average line down and makes the limits wider.

Chart 4.

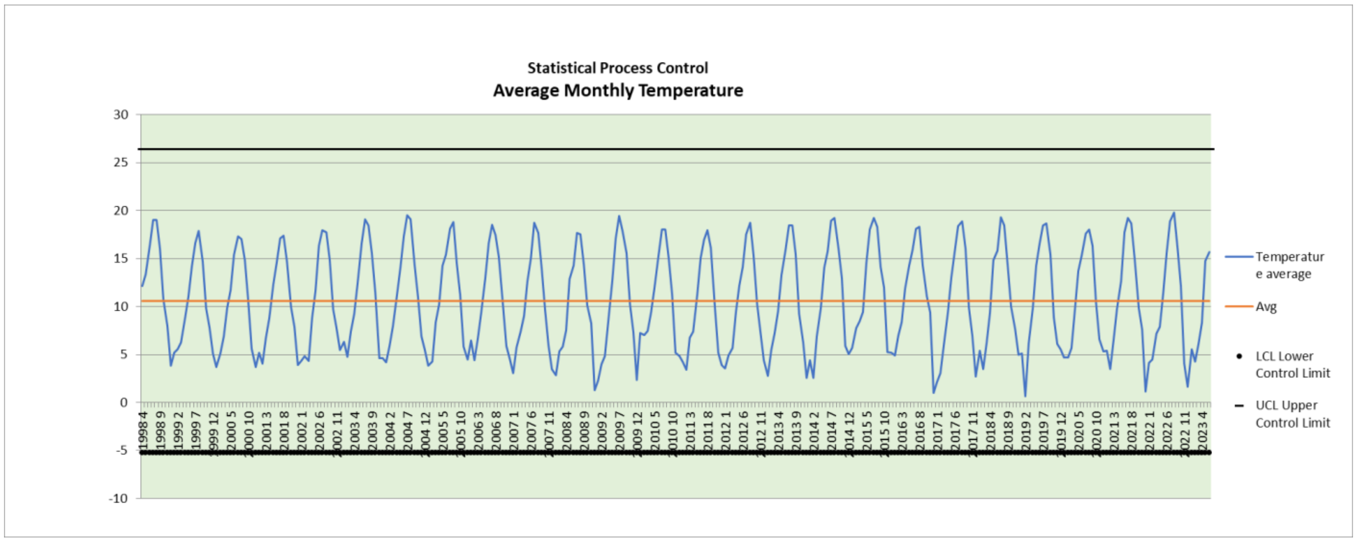


Chart 5.

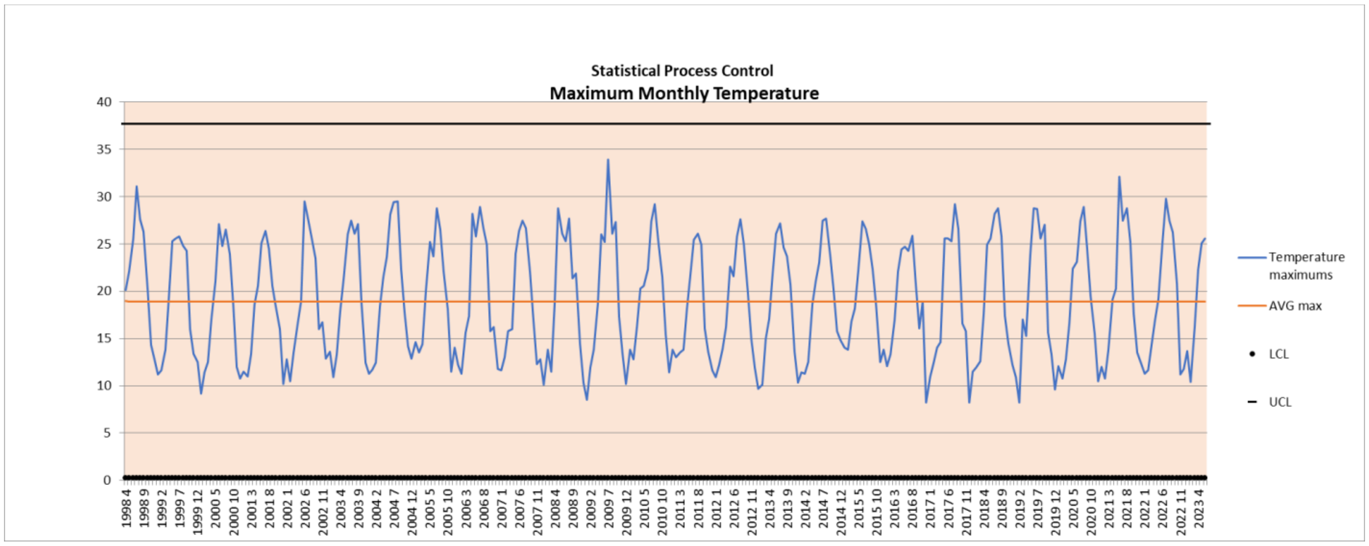
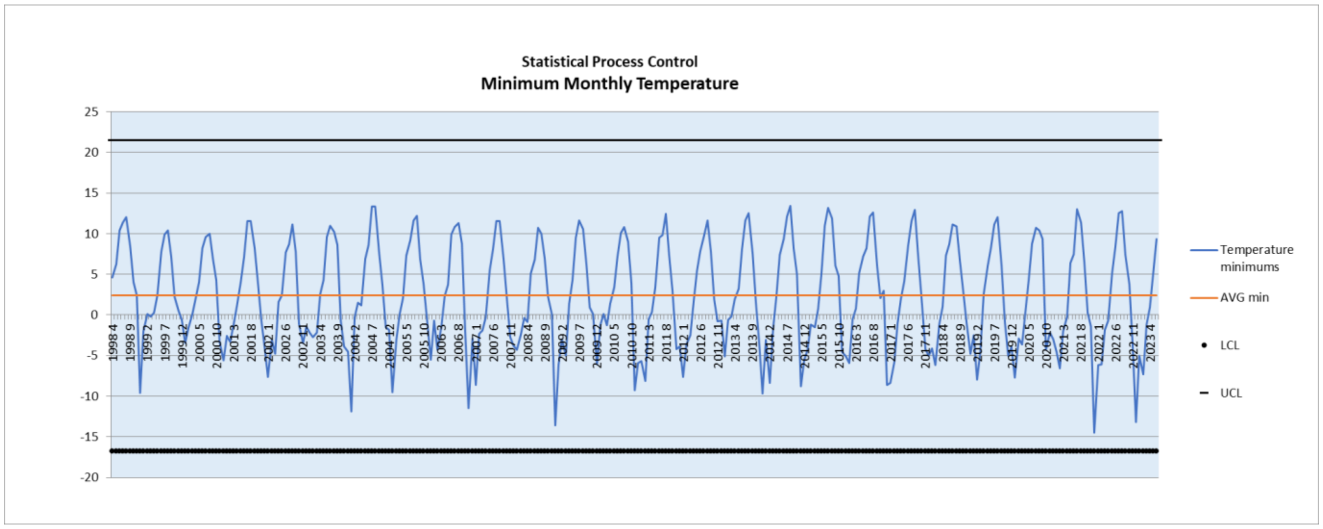
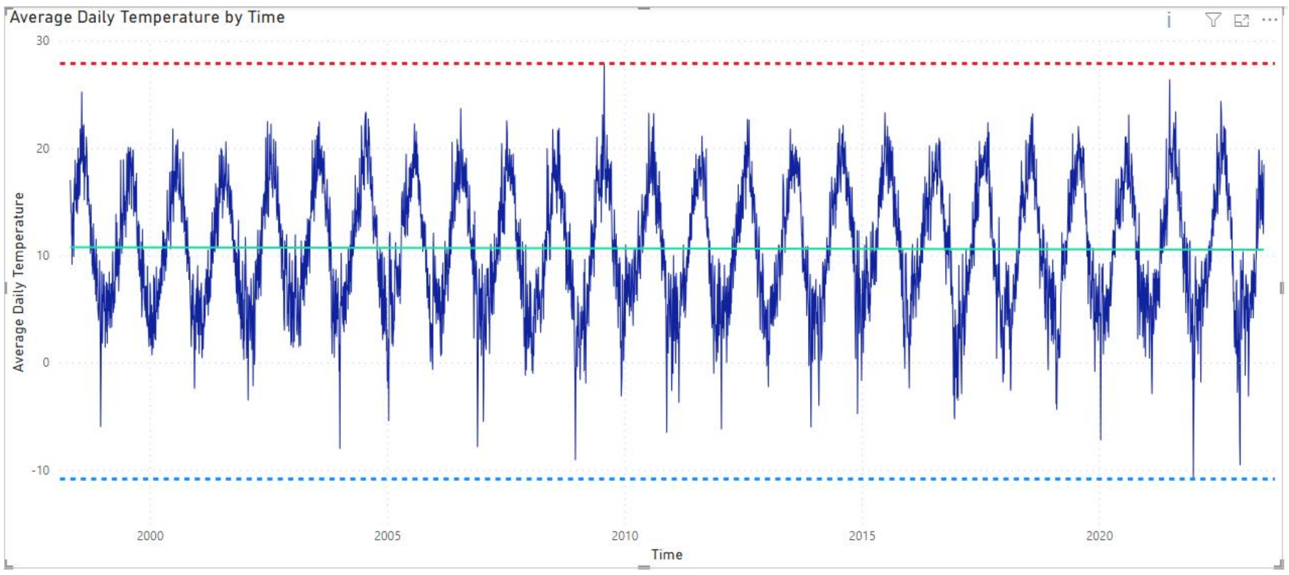


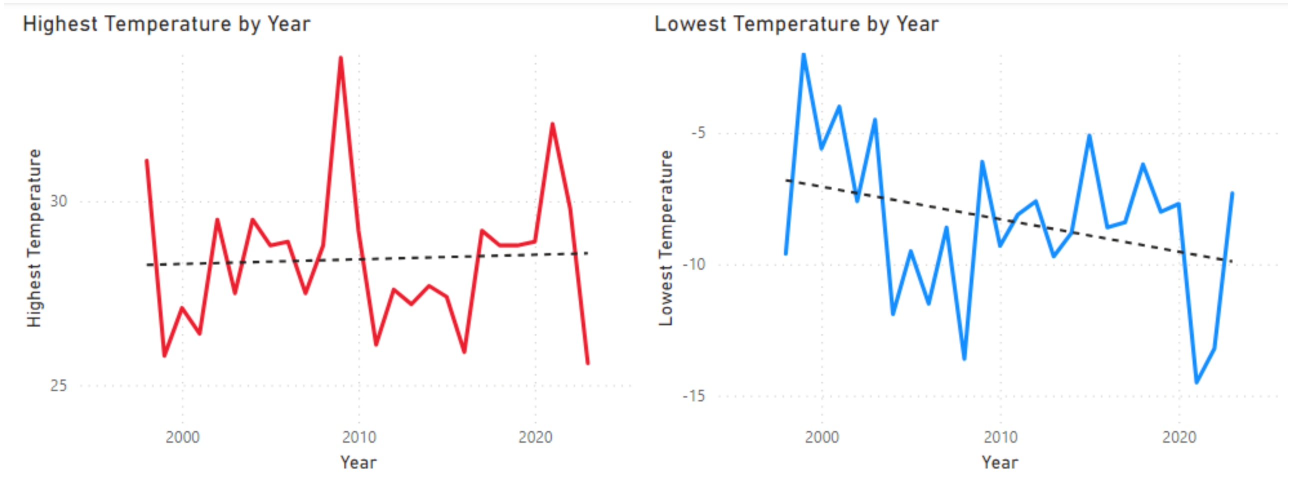
Chart 6.



I want to see if there are any trends noticeable in our data. I will be using daily data that I extracted after data preparation stage.

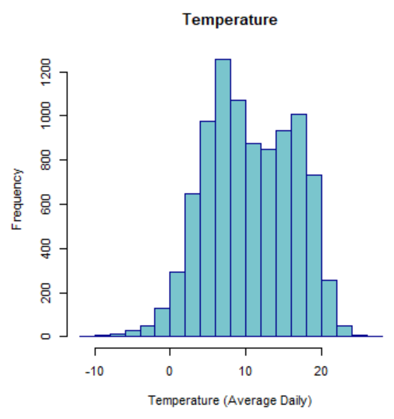


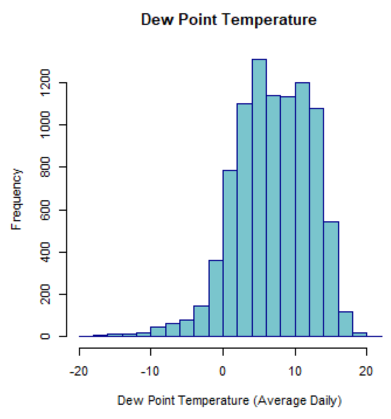
Here we can see daily temperature variations over the 25-year period. We can see definite cyclical variations in temperature as expected, since Vancouver is located in temperate area and has 4 distinct seasons. The green line in the middle is the trend line. There is a barely noticeable lowering trend. We can also notice a slight increase in temperature variations and lowest temperatures seem to get lower, and highest slightly higher. Let's have a look at the lowest and highest temperatures next.

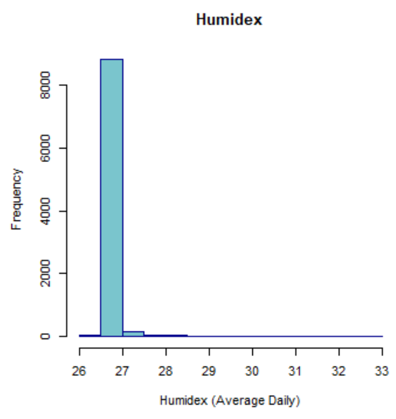


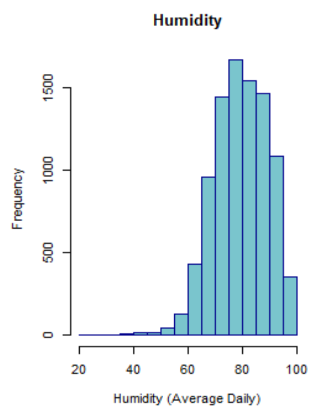
While it is hard to see from the Highest Temperature line alone, when we place a trend line across it, we can see that over the last 25 years, maximum yearly temperatures have been slightly rising. From the plot on the right, we see a stronger declining trend in the lowest temperatures. We can tell from our data that over the last 25 years, coldest days in a year have been getting colder, and warmest days have been getting a little warmer.

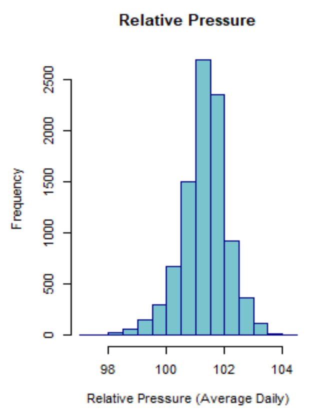
Now, lets have a look at the way our potential explanatory variables are distributed.

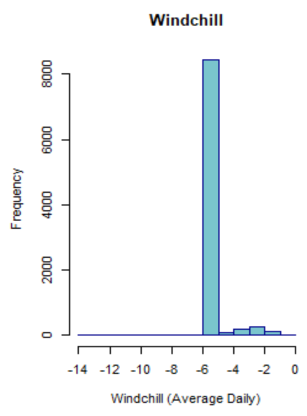


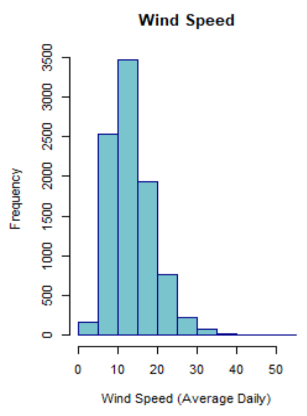








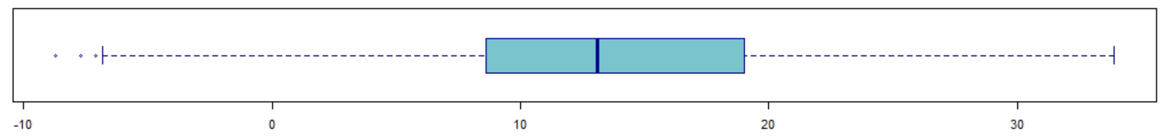




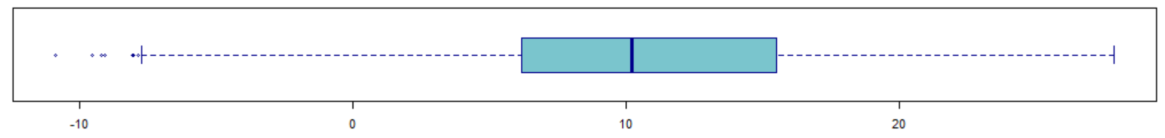
As we can see from above Temperature, Dew Point Temperature, Humidity and Relative Pressure are almost normally distributed, though all of these distribution are slightly skewed to the right. Wind Speed distribution and strongly skewed to the left. Windchill and Humidex are not normally distributed, though their distributions might look this peculiar because I imputed a lot of the missing variables and used medians to do so. In the initial dataset, before imputing, both Windchill and Humidex are also strongly skewed toward one end. The distribution shapes have changed after imputing, but they were no normal initially.

I will be using scaling to bring the distribution shapes closer to normal. To help me determine which type of scaler I should use, I will next construct and inspect the box and whisker plots for these features. I want to see if there are many outliers in my dataset. Below I will present whisker plots for my explanatory variables after I imputed missing variables and after I aggregated the data to show daily averages, minimums, and maximums, instead of using hourly data. Box and whisker plots and histograms for the initial data are attached in the appendix, as well as the code to be used in R to recreate the plots.

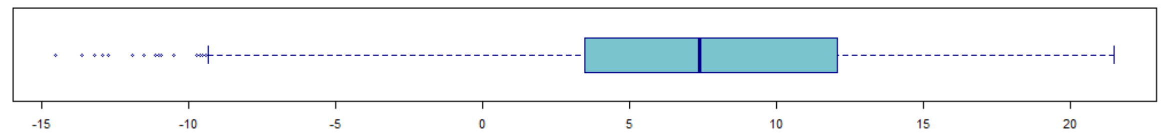
Maximum Temperature



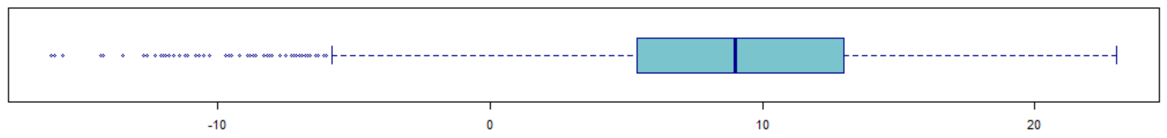
Mean Temperature



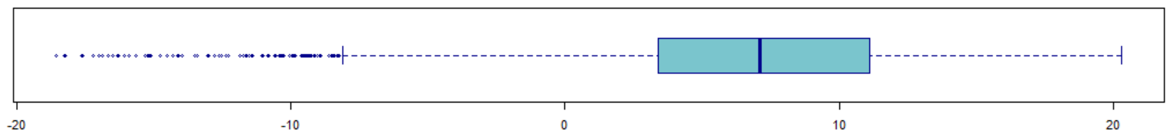
Minimum Temperature



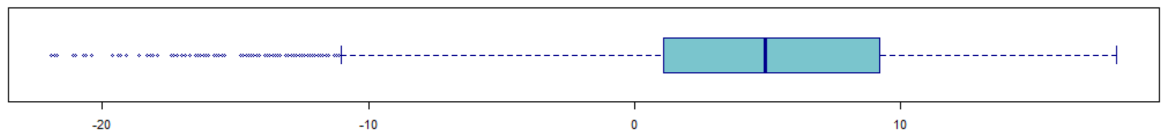
Maximum Dew Point Temperature

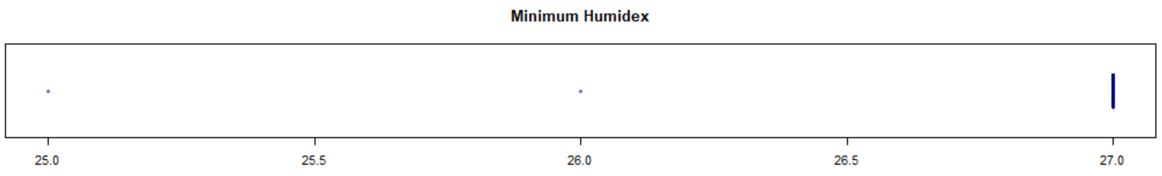


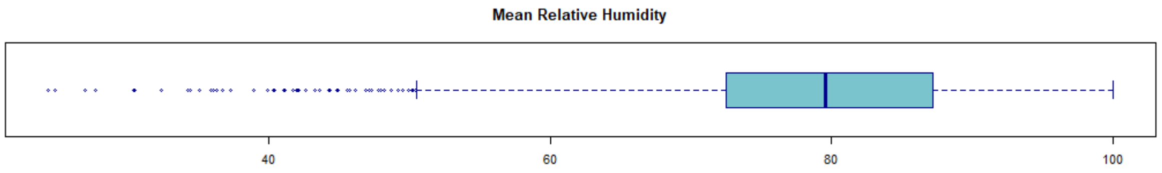
Mean Dew Point Temperature

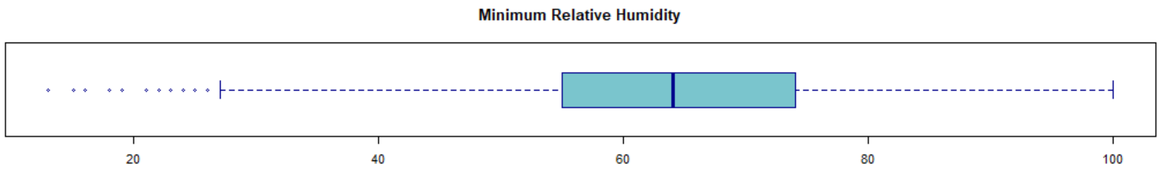


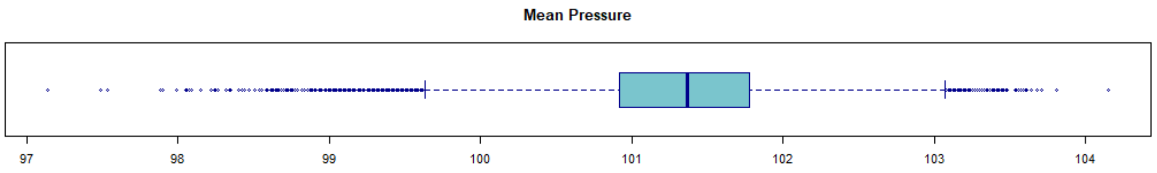
Minimum Dew Point Temperature

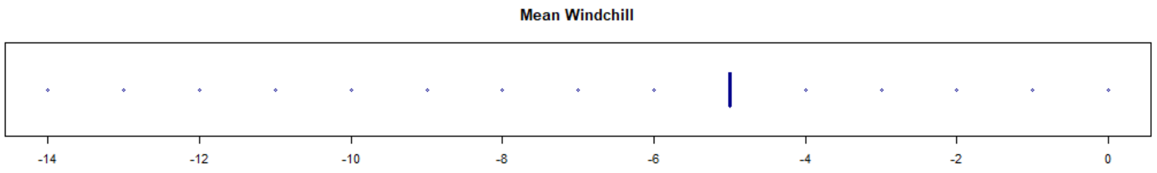


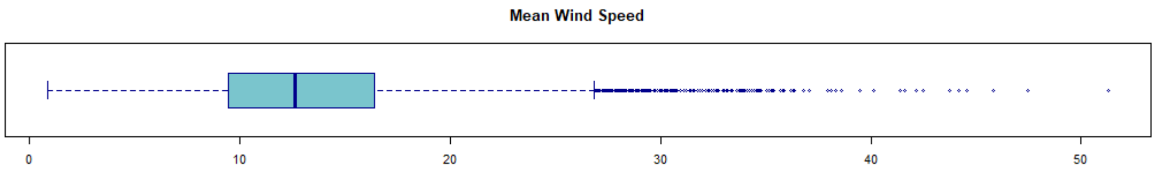






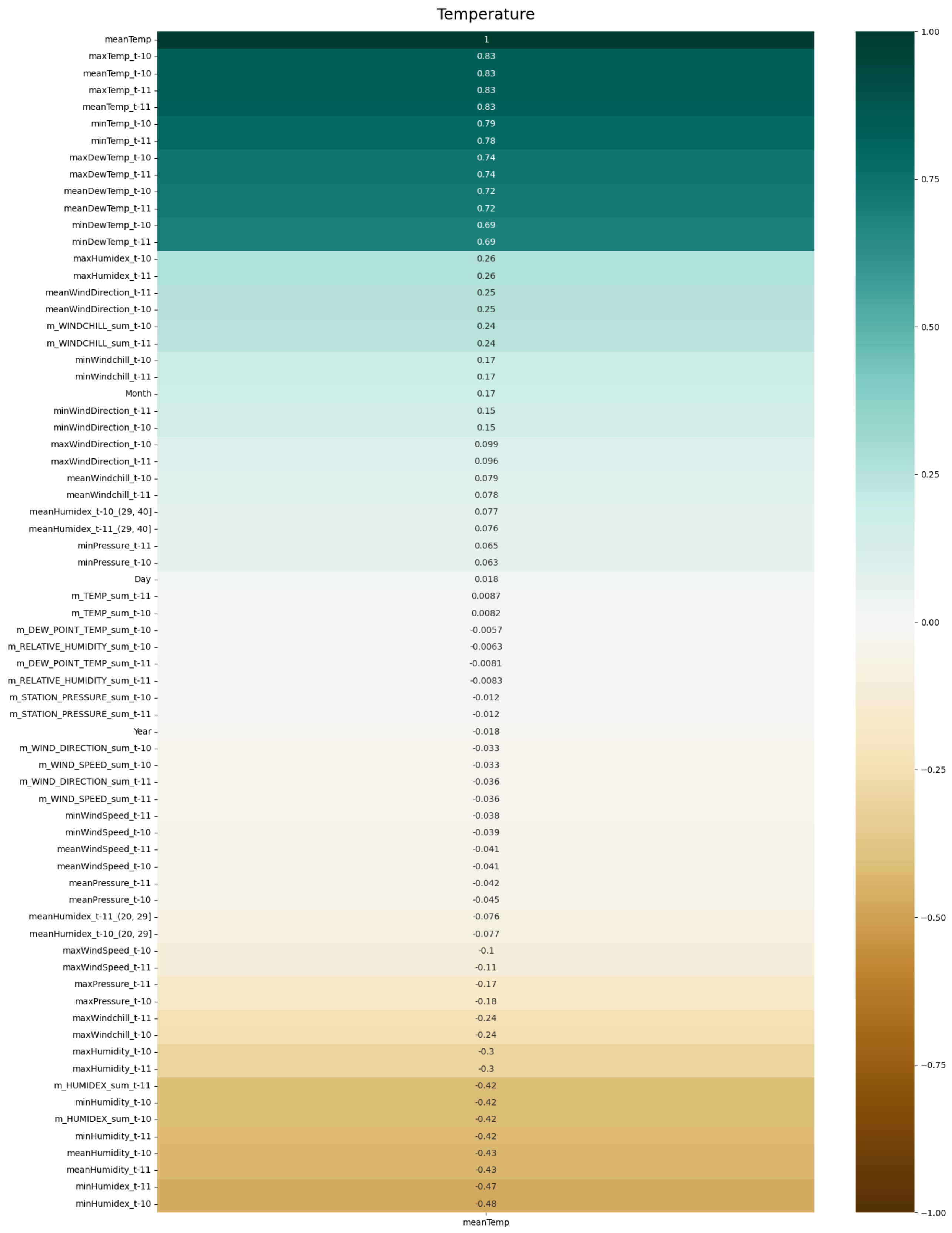






As we can see from the plots above, there are plenty of outliers in our features, so to prevent outliers from affecting our results too much, I will be using Robust Scaler. The ranges for the variables are also quite different, so scaling will bring all features into the same ranges.

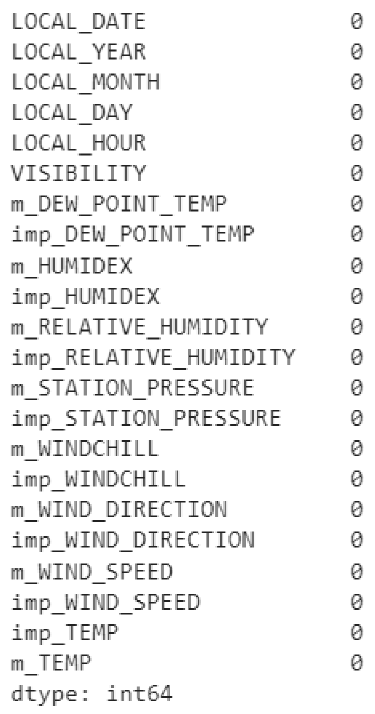
In this project I will use the dataset to try and predict the average temperature 10 days into the future. To achieve this, I will first aggregate the data to daily observations. I will then create time shifter variables. Below is a heatmap that demonstrates correlations between my target variable of meanTemp and other features in the dataset. I have a large number of potential explanatory variables and will be using different methods to select explanatory variables.


Temperature
meanTemp 1
maxTemp_t-10 0.83
meanTemp_t-10 0.83
maxTemp_t-11 0.83
meanTemp_t-11 0.83
minTemp_t-10 0.79
minTemp_t-11 0.78
maxDewTemp_t-10 0.74
maxDewTemp_t-11 0.74
meanDewTemp_t-10 0.72
meanDewTemp_t-11 0.72
minDewTemp_t-10 0.69
minDewTemp_t-11 0.69
maxHumidex_t-10 0.26
maxHumidex_t-11 0.26
meanWindDirection_t-11 0.25
meanWindDirection_t-10 0.25
m_WINDCHILL_sum_t-10 0.24
m_WINDCHILL_sum_t-11 0.24
minWindchill_t-10 0.17
minWindchill_t-11 0.17
Month 0.17
minWindDirection_t-11 0.15
minWindDirection_t-10 0.15
maxWindDirection_t-10 0.099
maxWindDirection_t-11 0.096
meanWindchill_t-10 0.079
meanWindchill_t-11 0.078
meanHumidex_t-10_(29, 40] 0.077
meanHumidex_t-11_(29, 40] 0.076
minPressure_t-11 0.065
minPressure_t-10 0.063
Day 0.018
m_TEMP_sum_t-11 0.0087
m_TEMP_sum_t-10 0.0082
m_DEW_POINT_TEMP_sum_t-10 -0.0057
m_RELATIVE_HUMIDITY_sum_t-10 -0.0063
m_DEW_POINT_TEMP_sum_t-11 -0.0081
m_RELATIVE_HUMIDITY_sum_t-11 -0.0083
m_STATION_PRESSURE_sum_t-10 -0.012
m_STATION_PRESSURE_sum_t-11 -0.012
Year -0.018
m_WIND_DIRECTION_sum_t-10 -0.033
m_WIND_SPEED_sum_t-10 -0.033
m_WIND_DIRECTION_sum_t-11 -0.036
m_WIND_SPEED_sum_t-11 -0.036
minWindSpeed_t-11 -0.038
minWindSpeed_t-10 -0.039
meanWindSpeed_t-11 -0.041
meanWindSpeed_t-10 -0.041
meanPressure_t-11 -0.042
meanPressure_t-10 -0.045
meanHumidex_t-11_(20, 29] -0.076
meanHumidex_t-10_(20, 29] -0.077
maxWindSpeed_t-10 -0.1
maxWindSpeed_t-11 -0.11
maxPressure_t-11 -0.17
maxPressure_t-10 -0.18
maxWindchill_t-11 -0.24
maxWindchill_t-10 -0.24
maxHumidity_t-10 -0.3
maxHumidity_t-11 -0.3
m_HUMIDEX_sum_t-11 -0.42
minHumidity_t-10 -0.42
m_HUMIDEX_sum_t-10 -0.42
minHumidity_t-11 -0.42
meanHumidity_t-10 -0.43
meanHumidity_t-11 -0.43
minHumidex_t-11 -0.47
minHumidex_t-10 -0.48
meanTemp
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Data Preparation

Imputing missing values

Instead of missing values in columns 'DEW_POINT_TEMP', 'HUMIDEX', 'RELATIVE_HUMIDITY', 'STATION_PRESSURE', 'WINDCHILL', 'WIND_DIRECTION', 'WIND_SPEED' input their median values. As a result of imputing, new column station with 'imp_' are created that contains our original data and medians instead of nulls. Another set of columns beginning with 'm_' are created that contain a binary value which indicates whether value was imputed or not. As we can see below, there are no longer any missing values in the dataset.



```
LOCAL_DATE               0
LOCAL_YEAR               0
LOCAL_MONTH              0
LOCAL_DAY                0
LOCAL_HOUR               0
VISIBILITY               0
m_DEW_POINT_TEMP         0
imp_DEW_POINT_TEMP       0
m_HUMIDEX                0
imp_HUMIDEX              0
m_RELATIVE_HUMIDITY      0
imp_RELATIVE_HUMIDITY    0
m_STATION_PRESSURE       0
imp_STATION_PRESSURE     0
m_WINDCHILL              0
imp_WINDCHILL            0
m_WIND_DIRECTION         0
imp_WIND_DIRECTION       0
m_WIND_SPEED             0
imp_WIND_SPEED           0
imp_TEMP                 0
m_TEMP                   0
dtype: int64
```

Combining into daily data

The original data is hourly since the measurements are updated every hour at the weather station. I would like to combine the data into daily data. I create aggregated columns for temperature, dew point temperature, humidex, relative humidity, station pressure, windchill, wind direction and wind speed: three columns for each variable containing mean, max and min. The indicator columns (indicator whether data was imputed or not) will be summed up. We still want to know whether the data was imputed or not. After aggregating that data, I rename the columns for more convenient use.

Time shift

As I want my models to predict average temperature 10 days into the future, I create columns that contained values shifted by 10 days into the past. This initially reduced number of wors to 9199. Together with aggregated columns and indicator columns, we now have 99 columns in the dataset and date is used as index. Of course, in the first 10 days of our dataset we don't have data for the previous 10 days. So now we have several rows of missing data. I run df.dropna() command to delete the rows with missing data. Since I will be using past data to predict meanTemp, I delete the columns containing data for the present period and only keep the time shifted columns. I only keep the meanTemp values for current period since this my target variable.

After running my initial models and recording their results, I decided to create two more possible explanatory variables, namely I create temperature on the same day last year and one day before that columns.

Binning

Environment Canada uses humidex ratings to inform the general public when conditions of heat and humidity are possibly uncomfortable. I split the humidex column into bins by relying on the table below.

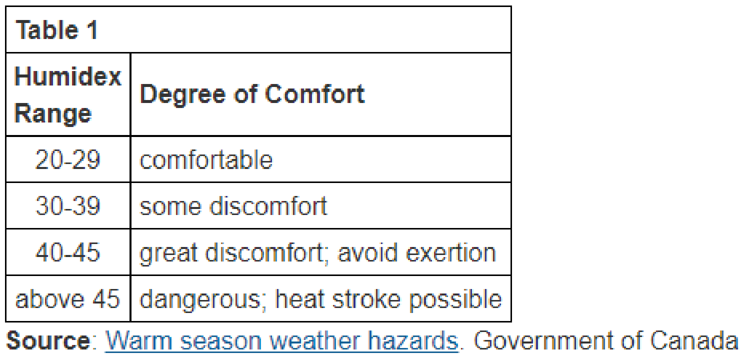


Table 1

Humidex Range	Degree of Comfort
20-29	comfortable
30-39	some discomfort
40-45	great discomfort; avoid exertion
above 45	dangerous; heat stroke possible

Source: Warm season weather hazards. Government of Canada

The maximum value of Humidex in the dataset is 40, therefore I did not create an above 40 bin.

Model Development

Determining significant variables

I now have 69 potential explanatory variables. To help me determine which of the variables to use I will try three different methods. As mentioned above, to avoid the influence of outliers in the dataset and to bring the features into the same ranges, I use Robust scaler. I then run an OLS regression using all of the columns. I then find out which of the coefficients are significant and print out the names of significant features to be used further in analysis. The significant features according to OLS regression are as follows:

```
'meanHumidex_t-11_(29, 40]', 'Month', 'meanDewTemp_t-11', 'maxHumidex_t-10',
'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11', 'minHumidity_t-
11', 'maxPressure_t-11', 'minPressure_t-10', 'meanWindchill_t-10',
'maxWindchill_t-10', 'maxWindchill_t-11', 'm_WINDCHILL_sum_t-11',
'm_TEMP_sum_t-10'.
```

The second method I use for feature selection if recursive feature elimination. This method begins y using all features, then least significant variables are dropped. According to this method, I should be using following features:

```
'meanTemp_t-11', 'maxTemp_t-10', 'meanDewTemp_t-10', 'meanDewTemp_t-11',
'meanHumidity_t-10', 'meanHumidity_t-11', 'm_DEW_POINT_TEMP_sum_t-10',
'm_RELATIVE_HUMIDITY_sum_t-10', 'm_TEMP_sum_t-10', 'm_TEMP_sum_t-11'
```

As a third method I use the forward feature selection method. This method starts by finding the single best feature, and then add next best etc. This method suggests that I use following variables for my models:

```
meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'maxTemp_t-11', 'minTemp_t-
10', 'minTemp_t-11', 'meanDewTemp_t-10', 'meanDewTemp_t-11', 'maxDewTemp_t-
10', 'maxDewTemp_t-11', 'minDewTemp_t-10', 'minDewTemp_t-11', 'minHumidex_t-
10', 'minHumidex_t-11', 'meanHumidity_t-10', 'meanHumidity_t-11',
'maxHumidity_t-10', 'maxHumidity_t-11', 'minHumidity_t-10', 'minHumidity_t-
11', 'm_HUMIDEX_sum_t-10', 'm_HUMIDEX_sum_t-11'
```

I also used Principal Component Analysis method and saved the results of the model created using PCA.

Using each set of explanatory variables, I ran three different models for each: Stacked Model with Scaling, Linear Regression with Cross Fold Validation, and Stacked Model with no Scaling. The results of these models are presented in the “Model comparisons” section of this assignment.

I then decided to create additional two columns with meanTemp 365 and 365 days prior to the current day.

Using same feature selections methods as above, I found these significant features:

Using OLS:

```
'meanTemp_t-366', 'Month', 'Day', 'meanDewTemp_t-11', 'meanHumidity_t-11',
'maxHumidity_t-11', 'meanPressure_t-10', 'minPressure_t-10',
'meanWindchill_t-10', 'meanWindchill_t-11', 'maxWindchill_t-10',
'maxWindchill_t-11', 'm_WINDCHILL_sum_t-10', 'm_WINDCHILL_sum_t-11',
'm_TEMP_sum_t-10', 'meanTemp_t-365', 'meanTemp_t-366'
```

Using RFE:

```
meanTemp_t-10, meanTemp_t-11, maxTemp_t-10, meanDewTemp_t-11,
meanHumidity_t-11, m_DEW_POINT_TEMP_sum_t-10, m_RELATIVE_HUMIDITY_sum_t-10,
m_TEMP_sum_t-10, meanTemp_t-365, meanTemp_t-366
```

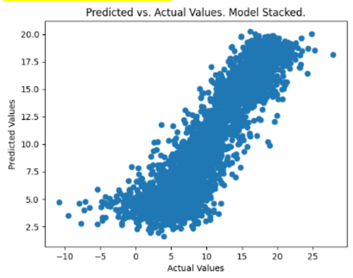
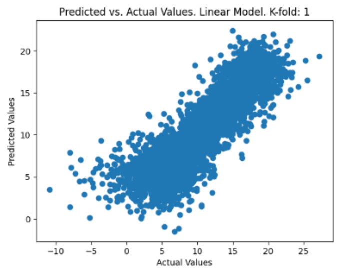
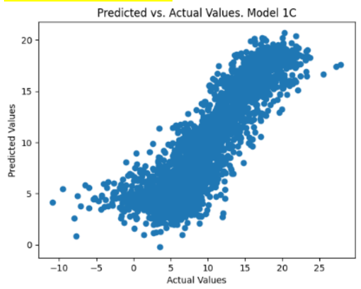
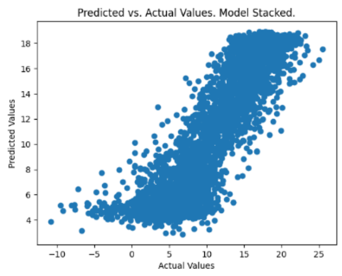
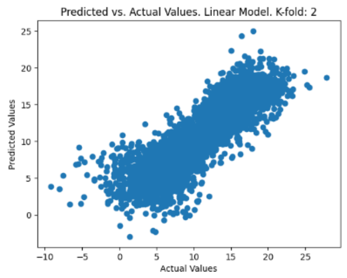
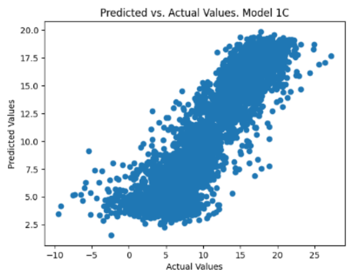
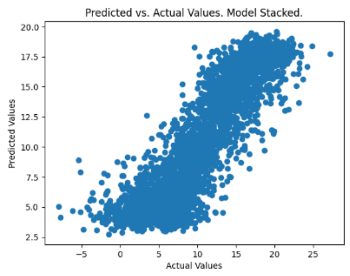
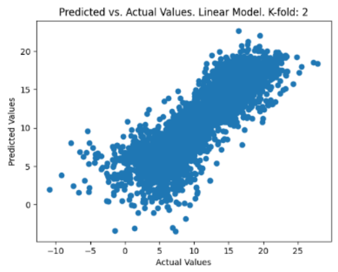
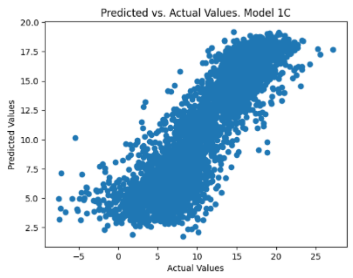
Using FFS:

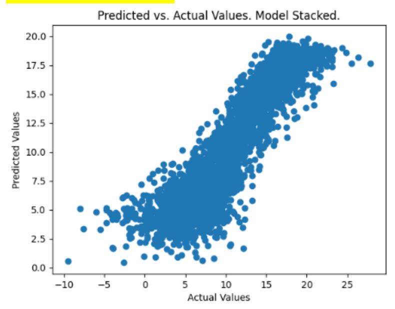
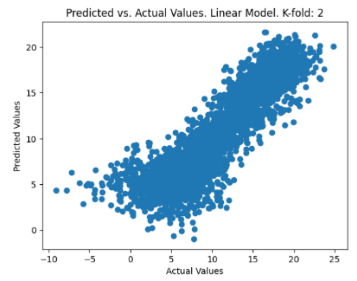
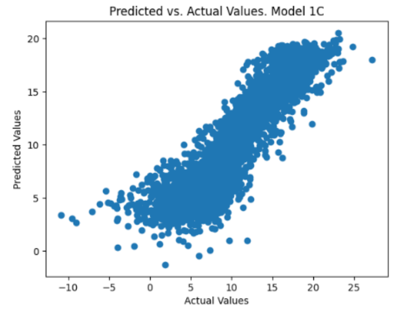
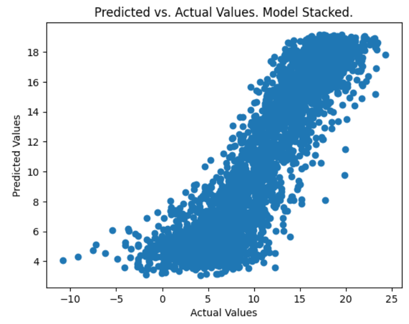
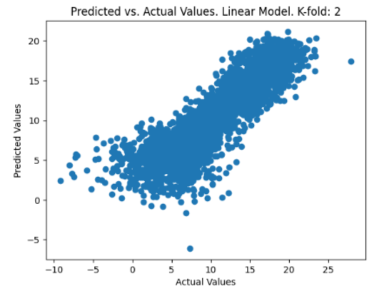
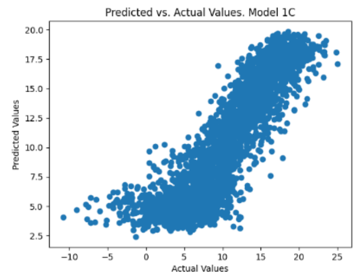
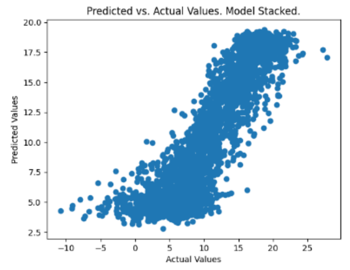
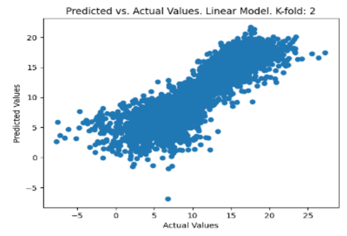
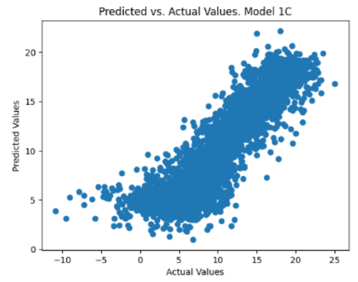
```
'meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'maxTemp_t-11',
'minTemp_t-10', 'minTemp_t-11', 'meanDewTemp_t-10', 'meanDewTemp_t-11',
'maxDewTemp_t-10', 'maxDewTemp_t-11', 'minDewTemp_t-10', 'minDewTemp_t-11',
'minHumidex_t-10', 'minHumidex_t-11', 'meanHumidity_t-10', 'meanHumidity_t-
11', 'maxHumidity_t-10', 'maxHumidity_t-11', 'minHumidity_t-10',
'minHumidity_t-11', 'm_HUMIDEX_sum_t-10', 'm_HUMIDEX_sum_t-11', 'meanTemp_t-
365'
```

I again made Stacked Model with scaling, Linear Regression model with cross fold validation, and Stacked model with no scaling. I also tried using PCA. Results are presented in the next section.

Model Comparisons

The following table presents the results of regressions for easy comparisons.

Features used (common features highlighted)	Stacked, Scaled	Linear, Scaled, Cross-Fold validation	Stacked, no scaling
OLS: 'meanHumidex_t-11_(29, 40]', 'Month', 'meanDewTemp_t-11', 'maxHumidex_t-10', 'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11', 'minHumidity_t-11', 'maxPressure_t-11', 'minPressure_t-10', 'meanWindchill_t-10', 'maxWindchill_t-10', 'maxWindchill_t-11', 'm_WINDCHILL_sum_t-11', 'm_TEMP_sum_t-10'	RMSE:2.563 	RMSE mean: 2.956 RMSE std: 0.0165 	RMSE:2.548 
RFE: 'meanTemp_t-11', 'maxTemp_t-10', 'meanDewTemp_t-10', 'meanDewTemp_t-11', 'meanHumidity_t-10', 'meanHumidity_t-11', 'm_DEW_POINT_TEMP_sum_t-10', 'm_RELATIVE_HUMIDITY_sum_t-10', 'm_TEMP_sum_t-10', 'm_TEMP_sum_t-11'	RMSE:2.833 	RMSE mean: 2.9576 RMSE std: 0.04783 	RMSE:2.784 
FFS: meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'maxTemp_t-11', 'minTemp_t-10', 'minTemp_t-11', 'meanDewTemp_t-10', 'meanDewTemp_t-11', 'maxDewTemp_t-10', 'maxDewTemp_t-11', 'minDewTemp_t-10', 'minDewTemp_t-11', 'minHumidex_t-10', 'minHumidex_t-11', 'meanHumidity_t-10', 'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11', 'minHumidity_t-10',	RMSE:2.755 	RMSE mean: 2.943 RMSE std: 0.048 	RMSE:2.808 

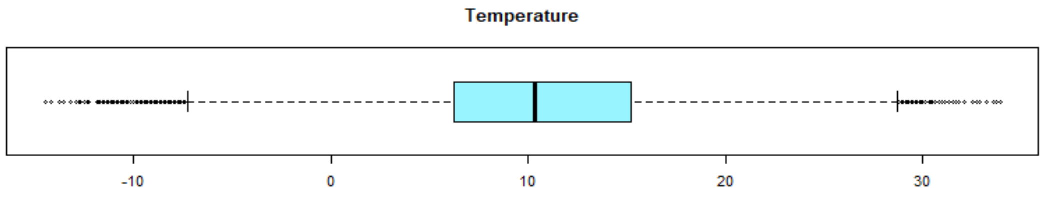
'minHumidity_t-11', 'm_HUMIDEX_sum_t-10', 'm_HUMIDEX_sum_t-11'			
PCA:	RMSE mean: 2.9158 RMSE Standard Deviation: 0.0415		
Second run (added meanTemp_t-365 and meanTemp_t-366)			
OLS: 'meanTemp_t-366', 'Month', 'Day', 'meanDewTemp_t-11', 'meanHumidity_t-11', 'maxHumidity_t-11', 'meanPressure_t-10', 'minPressure_t-10', 'meanWindchill_t-10', 'meanWindchill_t-11', 'maxWindchill_t-10', 'maxWindchill_t-11', 'm_WINDCHILL_sum_t-10', 'm_WINDCHILL_sum_t-11', 'm_TEMP_sum_t-10', 'meanTemp_t-365', 'meanTemp_t-366'	RMSE:2.436 	MSE mean: 2.737 RMSE std: 0.0549 	RMSE:2.378 
RFE: meanTemp_t-10, meanTemp_t-11, maxTemp_t-10, meanDewTemp_t-11, meanHumidity_t-11, m_DEW_POINT_TEMP_sum_t-10, m_RELATIVE_HUMIDITY_sum_t-10, m_TEMP_sum_t-10, meanTemp_t-365, meanTemp_t-366	RMSE:2.581 	RMSE mean: 2.754 RMSE std: 0.0086 	RMSE:2.678 
FFS: 'meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'maxTemp_t-11', 'minTemp_t-10', 'minTemp_t-11', 'meanDewTemp_t-10', 'meanDewTemp_t-11', 'maxDewTemp_t-10', 'maxDewTemp_t-11', 'minDewTemp_t-10', 'minDewTemp_t-11', 'minHumidex_t-10', 'minHumidex_t-11',	RMSE:2.701 	RMSE mean: 2.7616 RMSE std: 0.0168 	RMSE:2.719 

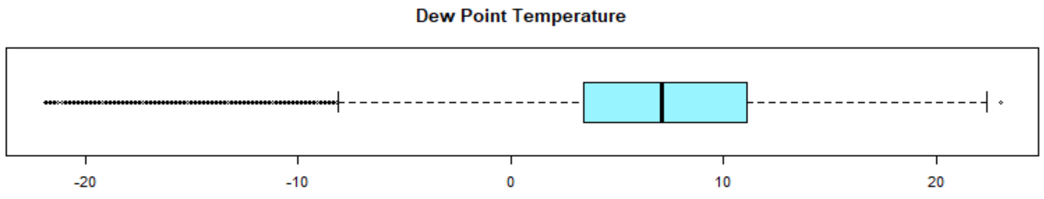
'meanHumidity_t-10', 'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11', 'minHumidity_t-10', 'minHumidity_t-11', 'm_HUMIDEX_sum_t-10', 'm_HUMIDEX_sum_t-11', 'meanTemp_t-365'			
PCA:	RMSE mean: 2.7596 RMSE Standard Deviation: 0.0707		

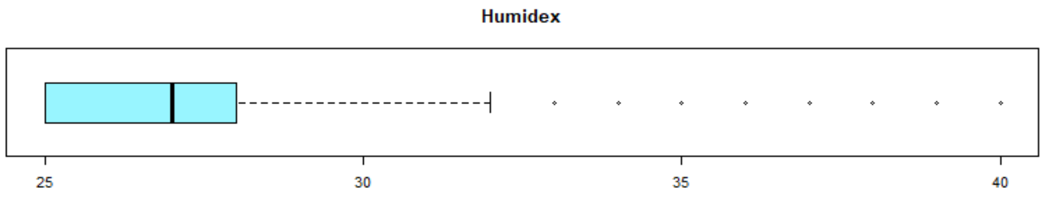
In both cases (with t-365 columns, and without) Stacked models perform the best, with or without Robust Scaler, when I use significant features determined using OLS regression. Stacked model with unscaled data performs the best when we include meanTemp_t-365 and meanTemp_t-366 using features determined using OLS regression. OLS regression determined features with statistically significant coefficients and those features led to better results.

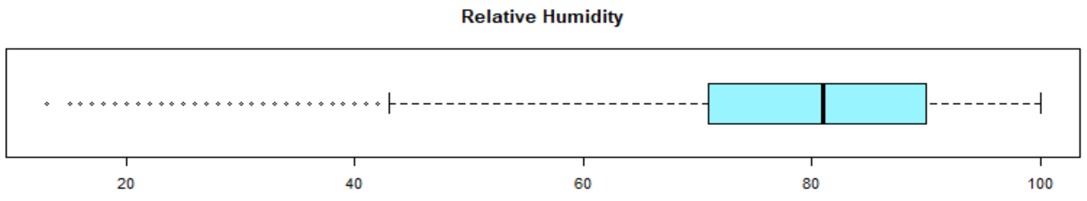
Appendix 1. Whisker plots and histograms of initial data

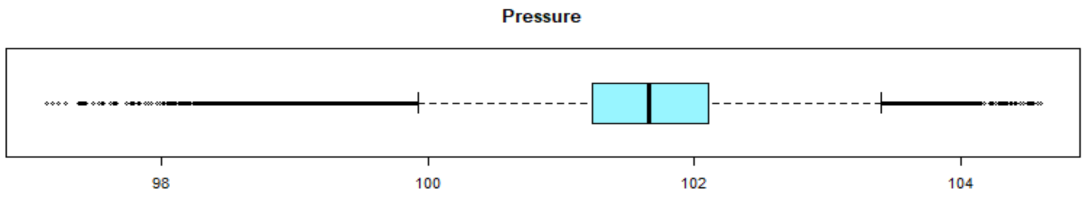
Box and whisker plots of initial data:

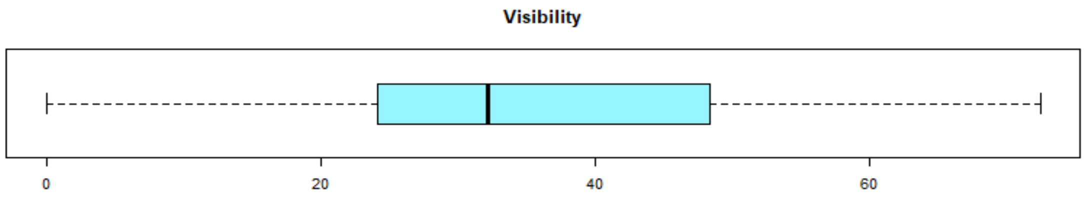


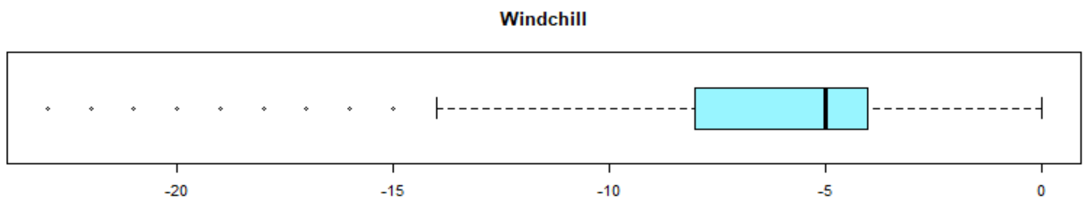


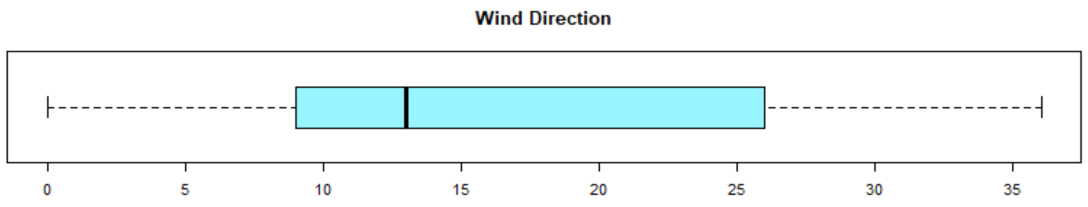


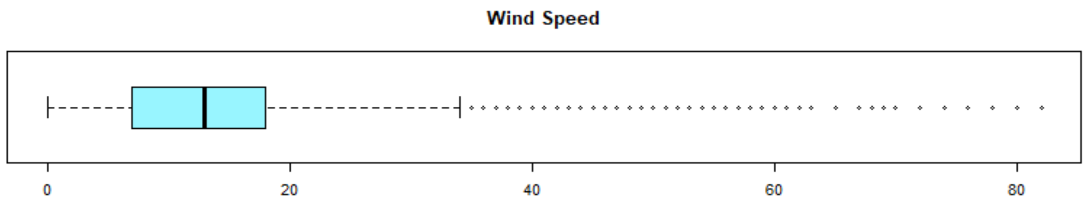




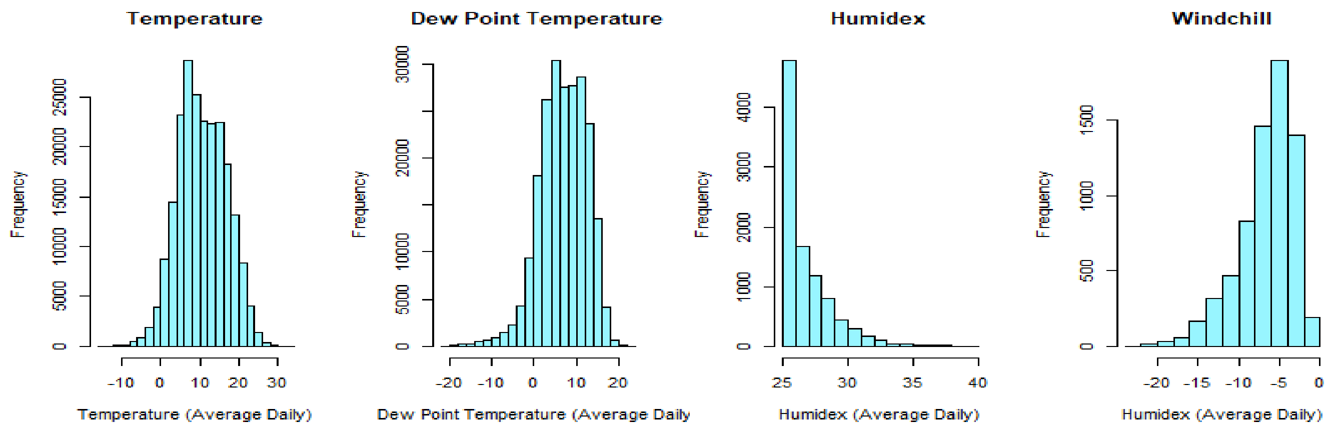


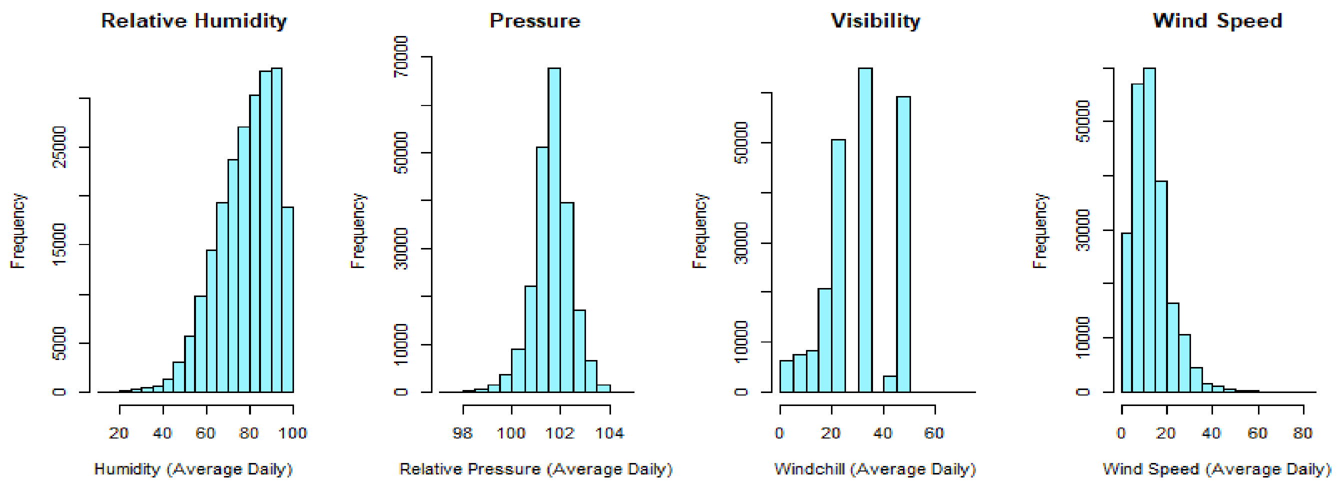

Windchill
-20
-15
-10
-5
0


Wind Direction
0
5
10
15
20
25
30
35


Wind Speed
0
20
40
60
80

Histograms:


Temperature
Frequency
Temperature (Average Daily)
Dew Point Temperature
Dew Point Temperature (Average Daily
Humidex
Humidex (Average Daily)
Windchill
Humidex (Average Daily)



Appendix 2. Code for R

```
#Initial hourly data
d1 <- read.csv("C:/Python/C4254/Vancouver Intl/initial.csv", header=TRUE,
stringsAsFactors=FALSE)

par(mfrow=c(3,1))
boxplot(d1$TEMP, main = "Temperature", col = "cadetblue1", border = "black", horizontal =
TRUE)
boxplot(d1$DEW_POINT_TEMP, main = "Dew Point Temperature", col = "cadetblue1", border =
"black", horizontal = TRUE)
boxplot(d1$HUMIDEX, main = "Humidex", col = "cadetblue1", border = "black", horizontal =
TRUE)
par(mfrow=c(3,1))
boxplot(d1$RELATIVE_HUMIDITY, main = "Relative Humidity", col = "cadetblue1", border =
"black", horizontal = TRUE)
boxplot(d1$STATION_PRESSURE, main = "Pressure", col = "cadetblue1", border = "black",
horizontal = TRUE)
boxplot(d1$VISIBILITY, main = "Visibility", col = "cadetblue1", border = "black", horizontal =
TRUE)
par(mfrow=c(3,1))
boxplot(d1$WINDCHILL, main = "Windchill", col = "cadetblue1", border = "black", horizontal =
TRUE)
boxplot(d1$WIND_DIRECTION, main = "Wind Direction", col = "cadetblue1", border = "black",
horizontal = TRUE)
boxplot(d1$WIND_SPEED, main = "Wind Speed", col = "cadetblue1", border = "black",
horizontal = TRUE)

par(mfrow=c(1,4))
```

```
hist(d1$TEMP, main = "Temperature", xlab = "Temperature (Average Daily)",col = "cadetblue1",
border = "black")
hist(d1$DEW_POINT_TEMP, main = "Dew Point Temperature",xlab = "Dew Point Temperature
(Average Daily)", col = "cadetblue1", border = "black")
hist(d1$HUMIDEX, main = "Humidex",xlab = "Humidex (Average Daily)", col = "cadetblue1",
border = "black")
hist(d1$WINDCHILL, main = "Windchill",xlab = "Humidex (Average Daily)", col = "cadetblue1",
border = "black")
par(mfrow=c(1,4))
hist(d1$RELATIVE_HUMIDITY, main = "Relative Humidity", xlab = "Humidity (Average Daily)", col
= "cadetblue1", border = "black")
hist(d1$STATION_PRESSURE, main = "Pressure", xlab = "Relative Pressure (Average Daily)", col =
"cadetblue1", border = "black")
hist(d1$VISIBILITY, main = "Visibility", xlab = "Windchill (Average Daily)",col = "cadetblue1",
border = "black")
hist(d1$WIND_SPEED, main = "Wind Speed", xlab = "Wind Speed (Average Daily)", col =
"cadetblue1", border = "black")

#Daily data.

d <- read.csv("C:/Python/C4254/Vancouver Intl/df_daily.csv", header=TRUE,
stringsAsFactors=FALSE)
head(d)

par(mfrow=c(3,1))
boxplot(d$maxTemp, main = "Maximum Temperature", col = "cadetblue3", border =
"darkblue", horizontal = TRUE)
boxplot(d$meanTemp, main = "Mean Temperature", col = "cadetblue3", border = "darkblue",
horizontal = TRUE)
boxplot(d$minTemp, main = "Minimum Temperature", col = "cadetblue3", border = "darkblue",
horizontal = TRUE)

par(mfrow=c(3,1))
boxplot(d$maxDewTemp, main = "Maximum Dew Point Temperature", col = "cadetblue3",
border = "blue4", horizontal = TRUE)
boxplot(d$meanDewTemp, main = "Mean Dew Point Temperature", col = "cadetblue3", border
= "blue4", horizontal = TRUE)
boxplot(d$minDewTemp, main = "Minimum Dew Point Temperature", col = "cadetblue3",
border = "blue4", horizontal = TRUE)
par(mfrow=c(3,1))
boxplot(d$minHumidex, main = "Minimum Humidex", col = "cadetblue3", border = "blue4",
horizontal = TRUE)
boxplot(d$meanHumidity, main = "Mean Relative Humidity", col = "cadetblue3", border =
"blue4", horizontal = TRUE)
```

```
boxplot(d$minHumidity, main = "Minimum Relative Humidity", col = "cadetblue3", border =
"blue4", horizontal = TRUE)
par(mfrow=c(3,1))
boxplot(d$minPressure, main = "Mean Pressure", col = "cadetblue3", border = "blue4",
horizontal = TRUE)
boxplot(d$maxWindchill, main = "Mean Windchill", col = "cadetblue3", border = "blue4",
horizontal = TRUE)
boxplot(d$meanWindSpeed, main = "Mean Wind Speed", col = "cadetblue3", border = "blue4",
horizontal = TRUE)

par(mfrow=c(1,3))
hist(d$meanTemp, main = "Temperature", xlab = "Temperature (Average Daily)",col =
"cadetblue3", border = "darkblue")
hist(d$meanDewTemp, main = "Dew Point Temperature",xlab = "Dew Point Temperature
(Average Daily)", col = "cadetblue3", border = "darkblue")
hist(d$meanHumidex, main = "Humidex",xlab = "Humidex (Average Daily)", col = "cadetblue3",
border = "darkblue")
par(mfrow=c(1,4))
hist(d$meanHumidity, main = "Humidity", xlab = "Humidity (Average Daily)", col = "cadetblue3",
border = "darkblue")
hist(d$minPressure, main = "Relative Pressure", xlab = "Relative Pressure (Average Daily)", col =
"cadetblue3", border = "darkblue")
hist(d$maxWindchill, main = "Windchill", xlab = "Windchill (Average Daily)",col = "cadetblue3",
border = "darkblue")
hist(d$meanWindSpeed, main = "Wind Speed", xlab = "Wind Speed (Average Daily)", col =
"cadetblue3", border = "darkblue")
```

Appendix 3: Python Code

```
import pandas as pd
#Our data comes in 23 separate csv files. First we combine these into one
file.
files = ["climate-hourly-1.csv", "climate-hourly-2.csv", "climate-hourly-
3.csv",
         "climate-hourly-4.csv", "climate-hourly-5.csv", "climate-hourly-
6.csv",
         "climate-hourly-7.csv", "climate-hourly-8.csv", "climate-hourly-
9.csv",
         "climate-hourly-10.csv", "climate-hourly-11.csv", "climate-hourly-
12.csv",
         "climate-hourly-13.csv", "climate-hourly-14.csv", "climate-hourly-
15.csv",
         "climate-hourly-16.csv", "climate-hourly-17.csv", "climate-hourly-
18.csv",
         "climate-hourly-19.csv", "climate-hourly-20.csv", "climate-hourly-
21.csv",
         "climate-hourly-22.csv", "climate-hourly-23.csv"]
df = pd.DataFrame()
for file in files:
    PATH = '/content/sample_data/Weather/'
    data = pd.read_csv(PATH + file)
    df = pd.concat([df,data], axis=0)
pd.set_option('display.max_columns', None, 'display.max_columns', None)
pd.set_option('display.width', 1000)
#Let's have a look at our data
print('***Initial Overview of Dataset***')
print("*********************************************")
print(df.head(8))
print(df.describe())
#Check columns with any missing values
print(df.isnull().sum())
df.to_csv('/content/sample_data/initial.csv')

#The following columns are not important since they are either constant or
empty
df.drop(['x', 'y', 'STATION_NAME',  'CLIMATE_IDENTIFIER', 'ID',
'PROVINCE_CODE', 'TEMP_FLAG',
        'DEW_POINT_TEMP_FLAG', 'HUMIDEX_FLAG', 'PRECIP_AMOUNT',
'PRECIP_AMOUNT_FLAG', 'RELATIVE_HUMIDITY_FLAG',
         'STATION_PRESSURE_FLAG', 'VISIBILITY_FLAG', 'WINDCHILL_FLAG',
'WIND_DIRECTION_FLAG', 'WIND_SPEED_FLAG'], axis=1, inplace=True)
print(df.head(4))

#Impute missing values
def imputeNullValues(colName, df):
    indicatorColName = 'm_'   + colName # Tracks whether imputed.
    imputedColName   = 'imp_' + colName # Stores original & imputed data.

    #Use median for imputing
    imputedValue = df[colName].median()

    imputedColumn = df[colName].fillna(imputedValue)
    indicatorColumn = df[colName].isnull().astype(int)
    df[indicatorColName] = indicatorColumn
    df[imputedColName]   = imputedColumn
    del df[colName]
    return df

df = imputeNullValues('DEW_POINT_TEMP', df)
df = imputeNullValues('HUMIDEX', df)
df = imputeNullValues('RELATIVE_HUMIDITY', df)
df = imputeNullValues('STATION_PRESSURE', df)
df = imputeNullValues('WINDCHILL', df)
df = imputeNullValues('WIND_DIRECTION', df)
df = imputeNullValues('WIND_SPEED', df)

#impute missing values in the target variable
imp_TEMP = df['TEMP'].fillna(method='bfill')
m_TEMP = df['TEMP'].isnull().astype(int)
df['imp_TEMP'] = imp_TEMP
df['m_TEMP'] = m_TEMP
del df['TEMP']
```

```
print('Dataset with Imputed Values')
print('***********************')
print(df.head(4))
print(df.describe())
#Check to see there are no more missing values.
print(df.isnull().sum())

df['LOCAL_DATE'] = pd.to_datetime(df['LOCAL_DATE'])
#Create tables for data exploration using Excel
df1 = df.copy() #copy the dataset to avoid unintended changes
df1['year'] = df1['LOCAL_DATE'].dt.year
df1['month'] = df1['LOCAL_DATE'].dt.month
averages = df1.groupby(['year', 'month'])['imp_TEMP'].mean().reset_index()
maximums = df1.groupby(['year', 'month'])['imp_TEMP'].max().reset_index()
minimums = df1.groupby(['year', 'month'])['imp_TEMP'].min().reset_index()
avgy = df1.groupby(['year'])['imp_TEMP'].mean().reset_index()
maxy = df1.groupby(['year'])['imp_TEMP'].max().reset_index()
miny = df1.groupby(['year'])['imp_TEMP'].min().reset_index()
new_df = pd.DataFrame({
    'Year': averages['year'],
    'Month': averages['month'],
    'Temperature average': averages['imp_TEMP'],
    'Temperature maximums':maximums['imp_TEMP'],
    'Temperature minimums':minimums['imp_TEMP'],
})
new_df1 = pd.DataFrame({
    'Year': avgy['year'],
    'Temperature average': avgy['imp_TEMP'],
    'Temperature maximums':maxy['imp_TEMP'],
    'Temperature minimums':miny['imp_TEMP'],
})
new_df.to_csv('/content/sample_data/for_excel_monthly.csv')
new_df1.to_csv('/content/sample_data/for_excel_yearly.csv')

#combine data into daily weather data
df.set_index('LOCAL_DATE', inplace=True)
indicator_columns  = df[['m_DEW_POINT_TEMP', 'm_HUMIDEX',
'm_RELATIVE_HUMIDITY',
                        'm_STATION_PRESSURE', 'm_WINDCHILL',
'm_WIND_DIRECTION', 'm_WIND_SPEED', 'm_TEMP']]
aggregation_function = {col: 'sum' for col in indicator_columns}
df_daily = df.resample('D').agg({'LOCAL_YEAR':['mean'], 'LOCAL_MONTH':
['mean'], 'LOCAL_DAY':['mean'],
                                'imp_TEMP':['mean', 'max',
'min'],'imp_DEW_POINT_TEMP':['mean', 'max', 'min'],
                                'imp_HUMIDEX':['mean', 'max', 'min'],
'imp_RELATIVE_HUMIDITY':['mean', 'max', 'min'],
                                'imp_STATION_PRESSURE': ['mean', 'max',
'min'], 'imp_WINDCHILL': ['mean', 'max', 'min'],
                                'imp_WIND_DIRECTION': ['mean', 'max',
'min'],'imp_WIND_SPEED': ['mean', 'max', 'min'],
                                 **aggregation_function })
df_daily.columns = df_daily.columns.map('_'.join)
#Save to csv for further use in RStudio or PowerBi
df_daily.to_csv('/content/sample_data/daily_weather_data.csv')
print('Daily Weather Data')
print('***********************')
print(df_daily.head(5))

#Rename column names for the convenience of use.
df_daily.rename(columns={'LOCAL_YEAR_mean':'Year', 'LOCAL_MONTH_mean':'Month',
'LOCAL_DAY_mean': 'Day',
                         'imp_TEMP_mean': 'meanTemp',
'imp_TEMP_max':'maxTemp', 'imp_TEMP_min':'minTemp',
                         'imp_DEW_POINT_TEMP_mean': 'meanDewTemp',
'imp_DEW_POINT_TEMP_max':'maxDewTemp', 'imp_DEW_POINT_TEMP_min':'minDewTemp',
                         'imp_HUMIDEX_mean': 'meanHumidex',
'imp_HUMIDEX_max':'maxHumidex', 'imp_HUMIDEX_min':'minHumidex',
                         'imp_RELATIVE_HUMIDITY_mean': 'meanHumidity',
'imp_RELATIVE_HUMIDITY_max':'maxHumidity',
'imp_RELATIVE_HUMIDITY_min':'minHumidity',
                         'imp_STATION_PRESSURE_mean': 'meanPressure',
'imp_STATION_PRESSURE_max':'maxPressure',
'imp_STATION_PRESSURE_min':'minPressure',
                         'imp_WINDCHILL_mean': 'meanWindchill',
'imp_WINDCHILL_max':'maxWindchill', 'imp_WINDCHILL_min':'minWindchill',
                         'imp_WIND_DIRECTION_mean': 'meanWindDirection',
'imp_WIND_DIRECTION_max':'maxWindDirection',
'imp_WIND_DIRECTION_min':'minWindDirection',
                         'imp_WIND_SPEED_mean': 'meanWindSpeed',
'imp_WIND_SPEED_max':'maxWindSpeed', 'imp_WIND_SPEED_min':'minWindSpeed'},
inplace=True)

#Shifting the variables in time
def timeShiftColumns(colName, df_daily):
    t1Name=colName + '_t-10'
    t2Name = colName + '_t-11'
    t1 = df_daily[colName].shift(periods=10)
    t2 = df_daily[colName].shift(periods=11)
    df_daily[t1Name] = t1
    df_daily[t2Name] = t2
    return df_daily
timeShiftColumns('meanTemp', df_daily)
timeShiftColumns('maxTemp', df_daily)
timeShiftColumns('minTemp', df_daily)
timeShiftColumns('meanDewTemp', df_daily)
timeShiftColumns('maxDewTemp', df_daily)
timeShiftColumns('minDewTemp', df_daily)
timeShiftColumns('meanHumidex', df_daily)
timeShiftColumns('maxHumidex', df_daily)
timeShiftColumns('minHumidex', df_daily)
timeShiftColumns('meanHumidity', df_daily)
timeShiftColumns('maxHumidity', df_daily)
timeShiftColumns('minHumidity', df_daily)
timeShiftColumns('meanPressure', df_daily)
timeShiftColumns('maxPressure', df_daily)
timeShiftColumns('minPressure', df_daily)
timeShiftColumns('meanWindchill', df_daily)
timeShiftColumns('maxWindchill', df_daily)
timeShiftColumns('minWindchill', df_daily)
timeShiftColumns('meanWindDirection', df_daily)
timeShiftColumns('maxWindDirection', df_daily)
timeShiftColumns('minWindDirection', df_daily)
timeShiftColumns('meanWindSpeed', df_daily)
timeShiftColumns('maxWindSpeed', df_daily)
timeShiftColumns('minWindSpeed', df_daily)
timeShiftColumns('m_DEW_POINT_TEMP_sum', df_daily)
timeShiftColumns('m_HUMIDEX_sum', df_daily)
timeShiftColumns('m_RELATIVE_HUMIDITY_sum', df_daily)
timeShiftColumns('m_STATION_PRESSURE_sum', df_daily)
timeShiftColumns('m_WINDCHILL_sum', df_daily)
timeShiftColumns('m_WIND_DIRECTION_sum', df_daily)
timeShiftColumns('m_WIND_SPEED_sum', df_daily)
timeShiftColumns('m_TEMP_sum', df_daily)

#Binning HUMIDEX column according Environment Canada humidex ratings
df_daily['meanHumidex_t-10'] = pd.cut(x=df_daily['meanHumidex_t-10'],
bins=[20,29,40])
df_daily = pd.get_dummies(df_daily, columns=['meanHumidex_t-10'])
df_daily['meanHumidex_t-11'] = pd.cut(x=df_daily['meanHumidex_t-11'],
bins=[20,29,40])
df_daily = pd.get_dummies(df_daily, columns=['meanHumidex_t-11'])
#Dropping rows with null values
df_daily = df_daily.dropna()
df_daily.to_csv('/content/sample_data/df_daily.csv')

df2 = df_daily.copy()
#Remove current values of various since they won't be used for predicting
df2.drop(['maxTemp', 'minTemp', 'meanDewTemp', 'maxDewTemp','minDewTemp',
'meanHumidex', 'maxHumidex', 'minHumidex',
          'meanHumidity','maxHumidity', 'minHumidity', 'meanPressure',
'maxPressure','minPressure', 'meanWindchill',
          'maxWindchill', 'minWindchill','meanWindDirection',
'maxWindDirection', 'minWindDirection', 'meanWindSpeed',
          'maxWindSpeed', 'minWindSpeed', 'm_DEW_POINT_TEMP_sum',
'm_HUMIDEX_sum', 'm_RELATIVE_HUMIDITY_sum',
          'm_STATION_PRESSURE_sum', 'm_WINDCHILL_sum',
'm_WIND_DIRECTION_sum','m_WIND_SPEED_sum', 'm_TEMP_sum'], axis=1,
inplace=True)
#Create a heatmap to look at correlations
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(16, 20))
heatmap = sns.heatmap(df2.corr() [['meanTemp']].\
            sort_values(by='meanTemp',
            ascending=False), vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title('Temperature',
                  fontdict={'fontsize':18}, pad=14)
plt.tight_layout()
plt.savefig('heatmap.png', dpi=300, bbox_inches='tight')
plt.show()

print(df2.shape)

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api  as sm
from sklearn.model_selection import KFold
```

```
X = df2.copy()
del X['meanTemp']

y = df2['meanTemp']

from sklearn.preprocessing import RobustScaler
def featureSelection(X,y):
  X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
  #Use scaling, use Robust Scaler
  #Run OLS and identify significant explanatory variables
  sc_x     = RobustScaler()
  X_scaled = sc_x.fit_transform(X)
  X_train_scaled = sc_x.fit_transform(X_train)
  X_test_scaled  = sc_x.transform(X_test)
  X_train_scaled = sm.add_constant(X_train_scaled)
  X_test_scaled = sm.add_constant(X_test_scaled)
  model = sm.OLS(y_train, X_train_scaled)
  results = model.fit()
  print(results.summary())
  p_values = results.pvalues
  significance_level = 0.05
  significant_features = np.where(p_values < significance_level)
  print("Significant Features using OLS:")
  for feature in significant_features:
      i = feature-1
      print(X.keys()[i])
  y_pred = results.predict(X_test_scaled)
  mse = mean_squared_error(y_test, y_pred)
  rmse = round(np.sqrt(mse),3)
  print("RMSE:  " + str(rmse))
  plt.scatter(y_test, y_pred)
  plt.xlabel("Actual Values")
  plt.ylabel("Predicted Values")
  plt.title("Predicted vs. Actual Values")
  plt.show()
  #Use recursive feature elimination to identify significant features
  model = LinearRegression()
  rfe = RFE(estimator=model, n_features_to_select=10)
  rfe.fit(X_scaled,y)
  selected_features_indices = rfe.support_
  feature_ranking = rfe.ranking_
  print("Selected feature indices:", selected_features_indices)
  print("Feature ranking:", feature_ranking)
  # Show top features.
  print("Features selected by RFE")
  for i in range(0, len(X.keys())):
      if(rfe.support_[i]):
          print(X.keys()[i])
  #Use forward feature selection to identify significant features
  from sklearn.feature_selection import f_regression
  ffs = f_regression(X_scaled, y)
  variable = []
  for i in range(0, len(X.columns) - 1):
      if ffs[0][i] >= 700:
          variable.append(X.columns[i])
  print("Features selected by FFS")
  print(variable)
featureSelection(X,y)

from sklearn.linear_model    import ElasticNet
from sklearn.tree            import DecisionTreeRegressor
from sklearn.svm             import SVR
from sklearn.ensemble        import AdaBoostRegressor
from sklearn.ensemble        import RandomForestRegressor
from sklearn.ensemble        import ExtraTreesRegressor

print("\n***********************")
print("Model using significant features identified using OLS regression.")
print("***********************")

X = df2[['meanHumidex_t-11_(29, 40]', 'Month', 'meanDewTemp_t-11',
'maxHumidex_t-10',
         'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11',
'minHumidity_t-11', 'maxPressure_t-11',
         'minPressure_t-10', 'meanWindchill_t-10', 'maxWindchill_t-10',
'maxWindchill_t-11', 'm_WINDCHILL_sum_t-11', 'm_TEMP_sum_t-10']]
y = df2['meanTemp']

def modelResultsPlots(X,y):
  sc_x     = RobustScaler()
  X_scaled = sc_x.fit_transform(X)
  def getUnfitModels():
      models = list()
      models.append(ElasticNet())
      models.append(SVR(gamma='scale'))
      models.append(DecisionTreeRegressor())
      models.append(AdaBoostRegressor())
      models.append(RandomForestRegressor(n_estimators=10))
      models.append(ExtraTreesRegressor(n_estimators=10))
      return models

  def evaluateModel(y_test, predictions, model):
      mse = mean_squared_error(y_test, predictions)
      rmse = round(np.sqrt(mse),3)
      print(" RMSE:" + str(rmse) + " " + model.__class__.__name__)

  def fitBaseModels(X_train, y_train, X_test, models):
      dfPredictions = pd.DataFrame()

      # Fit base model and store its predictions in dataframe.
      for i in range(0, len(models)):
          models[i].fit(X_train, y_train)
          predictions = models[i].predict(X_test)
          colName = str(i)
          # Add base model predictions to column of data frame.
          dfPredictions[colName] = predictions
      return dfPredictions, models

  def fitStackedModel(X, y):
      model = LinearRegression()
      model.fit(X, y)
      return model
  # Split data into train, test and validation sets.
  X_scaled_train, X_scaled_temp, y_train, y_temp = train_test_split(X_scaled,
y, test_size=0.70)
  X_scaled_test, X_scaled_val, y_test, y_val = train_test_split(X_scaled_temp,
y_temp, test_size=0.50)

  # Get base models.
  unfitModels = getUnfitModels()

  # Fit base and stacked models.
  dfPredictions, models = fitBaseModels(X_scaled_train, y_train,
X_scaled_test, unfitModels)
  stackedModel          = fitStackedModel(dfPredictions, y_test)

  # Evaluate base models with validation data.
  print("\n*Stacked Model with Robust Scaler")
  print("\n** Evaluate Base Models **")
  dfValidationPredictions = pd.DataFrame()
  for i in range(0, len(models)):
      predictions = models[i].predict(X_scaled_val)
      colName = str(i)
      dfValidationPredictions[colName] = predictions
      evaluateModel(y_val, predictions, models[i])

  # Evaluate stacked model with validation data.
  stackedPredictions = stackedModel.predict(dfValidationPredictions)
  print("\n** Evaluate Stacked Model **")
  evaluateModel(y_val, stackedPredictions, stackedModel)

  plt.scatter(y_val, stackedPredictions)
  plt.xlabel("Actual Values")
  plt.ylabel("Predicted Values")
  plt.title("Predicted vs. Actual Values. Model Stacked.")
  plt.show()

  print("Cross-fold validation")
  count = 0

  kfold = KFold(n_splits=3, shuffle= True)
  RMSElist = []

  for train_index, test_index in kfold.split(X_scaled):
      X_train, X_test = X_scaled[train_index], X_scaled[test_index]
      y_train, y_test = y.iloc[train_index], y.iloc[test_index]

      #  In the case of predicting mean temperature based on weather data, it
is not necessary to scale the target variable
      #  (mean temperature). in the context of weather data, the mean
temperature is often already presented on a
      #  consistent scale (e.g., Celsius or Fahrenheit) that is interpretable
and directly comparable to the predictions.
      y_train, y_test = y[train_index], y[test_index]

      # Perform linear regression.
      linearModel = LinearRegression()
      # Fit the model.
      linearModel.fit(X_train, y_train)
      y_pred = linearModel.predict(X_test)
      coefficients = linearModel.coef_
      intercept = linearModel.intercept_
      plt.scatter(y_test, y_pred)
      plt.xlabel("Actual Values")
      plt.ylabel("Predicted Values")
      plt.title("Predicted vs. Actual Values. Linear Model. K-fold: " +
str(count))
      plt.show()

      # Show confusion matrix and accuracy scores.
      count += 1
      print("***K-fold: " + str(count))
      print("Coefficients: " + str(coefficients))
      print("Intercept: " + str(intercept))

      # Calculate scores and add to the list.
      rmse = round(np.sqrt(mean_squared_error(y_test, y_pred)), 3)

      RMSElist.append(rmse)

      print('RMSE: ', rmse)

  # Show averages of scores over multiple runs.
  print("**********************")
  print("Linear Regression with Folding, with Robust Scaler.")
  print("**********************")
  print("\nRMSE and RMSE Standard Deviation For All Folds (RobustScaler):")
  print("*********************************************")
  print("RMSE mean:  " + str(np.mean(RMSElist)))
  print("RMSE std:      " + str(np.std(RMSElist)))

  print("**********************")
  print("Stacked Model, no Scaling")
  print("**********************")
  # Split data into train, test and validation sets.
  X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.70)
  X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp,
test_size=0.50)

  # Get base models.
  unfitModels = getUnfitModels()

  # Fit base and stacked models.
  dfPredictions, models = fitBaseModels(X_train, y_train, X_test, unfitModels)
  stackedModel = fitStackedModel(dfPredictions, y_test)

  # Evaluate base models with validation data.
  print("\n** Evaluate Base Models **")
  dfValidationPredictions = pd.DataFrame()
  for i in range(0, len(models)):
      predictions = models[i].predict(X_val)
      colName = str(i)
      dfValidationPredictions[colName] = predictions
      evaluateModel(y_val, predictions, models[i])
  # Evaluate stacked model with validation data.
  stackedPredictions = stackedModel.predict(dfValidationPredictions)
  print("\n** Evaluate Stacked Model **")
  evaluateModel(y_val, stackedPredictions, stackedModel)

  plt.scatter(y_val, stackedPredictions)
  plt.xlabel("Actual Values")
  plt.ylabel("Predicted Values")
  plt.title("Predicted vs. Actual Values. Model Stacked. No Scaling")
  plt.show()

modelResultsPlots(X,y)

print("\n***********************")
print("Models using significant features identified using RFE.")
print("***********************")


X = df2[['meanTemp_t-11', 'maxTemp_t-10', 'meanDewTemp_t-10', 'meanDewTemp_t-
11', 'meanHumidity_t-10', 'meanHumidity_t-11',
         'm_DEW_POINT_TEMP_sum_t-10', 'm_RELATIVE_HUMIDITY_sum_t-10',
'm_TEMP_sum_t-10', 'm_TEMP_sum_t-11']]
y = df2['meanTemp']
modelResultsPlots(X,y)

print("\n***********************")
print("Model using significant features identified using FFS.")
print("***********************")

X = df2[['meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'maxTemp_t-11',
'minTemp_t-10', 'minTemp_t-11', 'meanDewTemp_t-10',
         'meanDewTemp_t-11', 'maxDewTemp_t-10', 'maxDewTemp_t-11',
'minDewTemp_t-10', 'minDewTemp_t-11', 'minHumidex_t-10', 'minHumidex_t-11',
'meanHumidity_t-10',
         'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11',
'minHumidity_t-10', 'minHumidity_t-11', 'm_HUMIDEX_sum_t-10',
'm_HUMIDEX_sum_t-11']]
y = df2['meanTemp']
modelResultsPlots(X,y)

from sklearn.decomposition  import PCA
from sklearn.preprocessing  import StandardScaler
from sklearn                import model_selection
X = df2.copy()
del X['meanTemp']
y = df2['meanTemp']

def pca(runs, X, y):
  num_runs = runs
  rmse_scores = []

  for run in range(num_runs):
    pca = PCA(n_components=15)
    X_scaled = StandardScaler().fit_transform(X)
    # Split into training and test sets
    X_train, X_test , y_train, y_test =
model_selection.train_test_split(X_scaled, y,
                                                      test_size=0.3)

    X_reduced_train = pca.fit_transform(X_train)
    X_reduced_test  = pca.transform(X_test)
    # Train regression model on training data
    model = LinearRegression()
    model.fit(X_reduced_train, y_train)

    # Prediction with test data
    pred = model.predict(X_reduced_test)
    mse = mean_squared_error(y_test, pred)
    RMSE = round(np.sqrt(mse), 3)
    rmse_scores.append(RMSE)
  print("\n***********************")
  print("\nPrincipal Component Analysis")
  print("***********************")
  print("RMSE mean: " + str(np.mean(rmse_scores)))
  print("RMSE Standard Deviation: " + str(np.std(rmse_scores)))
pca(5, X, y)

#Add columns with average temperature on the same day 1 year previous
def timeShiftColumns(colName, df2):
    t1Name=colName + '_t-365'
    t2Name = colName + '_t-366'
    t1 = df2[colName].shift(periods=365)
    t2 = df2[colName].shift(periods=366)
    df2[t1Name] = t1
    df2[t2Name] = t2
    return df2
timeShiftColumns('meanTemp', df2)

df2 = df2.dropna()

X = df2.copy()
del X['meanTemp']

y = df2['meanTemp']

featureSelection(X,y)

print("\n***********************")
print("Model using significant features identified using OLS regression.")
print("***********************")

X = df2[['meanTemp_t-366', 'Month', 'Day', 'meanDewTemp_t-11',
'meanHumidity_t-11', 'maxHumidity_t-11',
         'meanPressure_t-10', 'minPressure_t-10', 'meanWindchill_t-10',
'meanWindchill_t-11',
         'maxWindchill_t-10', 'maxWindchill_t-11', 'm_WINDCHILL_sum_t-10',
'm_WINDCHILL_sum_t-11', 'm_TEMP_sum_t-10', 'meanTemp_t-365', 'meanTemp_t-
366']]
y = df2['meanTemp']

modelResultsPlots(X,y)

print("\n***********************")
print("Models using significant features identified using RFE.")
print("***********************")


X = df2[['meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'meanDewTemp_t-11',
         'meanHumidity_t-11', 'm_DEW_POINT_TEMP_sum_t-10',
'm_RELATIVE_HUMIDITY_sum_t-10',
         'm_TEMP_sum_t-10', 'meanTemp_t-365', 'meanTemp_t-366']]
y = df2['meanTemp']
modelResultsPlots(X,y)

print("\n***********************")
print("Model using significant features identified using FFS.")
print("***********************")

X = df2[['meanTemp_t-10', 'meanTemp_t-11', 'maxTemp_t-10', 'maxTemp_t-11',
'minTemp_t-10',
         'minTemp_t-11', 'meanDewTemp_t-10', 'meanDewTemp_t-11',
'maxDewTemp_t-10', 'maxDewTemp_t-11',
         'minDewTemp_t-10', 'minDewTemp_t-11', 'minHumidex_t-10',
'minHumidex_t-11', 'meanHumidity_t-10',
         'meanHumidity_t-11', 'maxHumidity_t-10', 'maxHumidity_t-11',
'minHumidity_t-10', 'minHumidity_t-11',
         'm_HUMIDEX_sum_t-10', 'm_HUMIDEX_sum_t-11', 'meanTemp_t-365']]
y = df2['meanTemp']
modelResultsPlots(X,y)

X = df2.copy()
del X['meanTemp']
y = df2['meanTemp']

pca(5,X,y)
```