SOFE 4950U Final Project

Group 14

Julian Pascual (100707563)
April 2, 2023

The code, models, scripts, and Jupyter Notebooks can be found at https://github.com/jcpascual/SOFE4950U-Project.

Problem

Create a weather prediction model that can predict future quantitative values (low temperature, high temperature) and qualitative values (current condition – cloudy, rainy, sunny, etc.) in Toronto based on past data.

Motivation

Weather is something that every human has to deal with on a regular basis. It affects many aspects of our lives, often forcing us to adapt to the conditions and upcoming forecast.

Early and accurate prediction of the weather can help us in our day-to-day lives.

For example, knowing how the weather will be for the next day or week can help improve one's quality of life. If one knows that it will rain later today, they can prepare by bringing an umbrella or raincoat if they need to go out. If one knows that the temperature will be high, they would know that they do not need to put on as many layers of clothing before leaving the house.

In extreme situations, it could even save lives. For example, if a machine learning model predicts that there could be an extreme weather event (for example, a hurricane) based on input features, then preparations can be started well in advance and evacuations can be put into place if necessary, protecting life and property.

Related Work

The field of weather prediction has had significant study and investment over the past several decades.

Machine learning models are already in use in several forecasting centers and are used to supplement pre-existing simulators and prediction engines. For example, the European Centre for Medium-Range Weather Forecasts (ECMWF) are starting to use machine learning techniques to assist with their everyday forecasting. Some computationally expensive tools and calculations are being replaced with machine learning equivalents that can execute at much faster speeds. Many scientists believe

that machine learning will increasingly be used in the field, supplementing current forecasting techniques. [1]

Datasets

To facilitate accurate weather forecasting for the entirety of Canada, Environment Canada and NAV CANADA jointly operate hundreds of weather stations across the country. Humans and automated equipment collect information about the current weather conditions at these stations on an hourly basis. At the end of the day, these data points are then aggregated into a daily summary of the conditions. Environment Canada offers datasets containing historical data from these weather stations free of charge.

I elected to use the dataset for station ID 51459, which is located at Toronto Pearson International Airport. [2] The weather station located there has a significant amount of historical data which is helpful for training a machine learning model. It is also one of the major weather stations whose data is used to create forecasts for the City of Toronto and residents in surrounding areas.

To download the data, I used command lines provided by Environment Canada, which automates fetching the entire hourly and daily datasets within a given time range. [3] I chose to download all data between January 2013 and February 2023, since that's what was available. The resulting download consisted of multiple CSV files, each containing data for a specific time interval. For the daily dataset, each CSV file contains entries for 1 year (365 days). For the hourly dataset, each CSV file contains entries for 1 month (~744 hours).

Approach

After each dataset, I decided to attempt to predict the following features:

- Minimum temperature
- Maximum temperature
- Total precipitation
- Max gust
- Minimum humidity
- Maximum humidity
- Minimum wind speed
- Maximum wind speed
- Minimum visibility

- Maximum visibility
- Minimum atmospheric pressure
- Maximum atmospheric pressure

First, I unified the multiple CSV files into one for each type of dataset. While loading each CSV individually would've worked, I chose to do this to make the code to train the models simpler. Doing this was accomplished using two scripts. Each script does the same thing, except for minor changes to the input folder path and the date/time column name. Each CSV is loaded into a DataFrame using pandas. Then, each individual DataFrame is concatenated into one big DataFrame. Finally, the rows are sorted by the values in the date/time column. Once finished, a single CSV is written to the output directory. The code for this can be found in Notebooks/1_CombineDatasets.ipynb.

Second, I had to combine both datasets into one. This is because the hourly dataset includes some features that the daily dataset does not. This includes:

- Relative humidity
- Wind speed
- Visibility
- Atmospheric pressure
- Weather conditions

I wrote a third script to combine the datasets. The script reads the data in the unified daily dataset. For each day in that dataset, I load all rows related to that day from the unified hourly dataset. For features that are not present in the daily dataset, I take the minimum and maximum values of each corresponding column, and save those. Every "Weather" column from the hourly rows is checked, and if any of them contain precipitation (rain / snow / hail), then the day is marked as containing precipitation. Then, features from the daily dataset are extracted. Once all values have been loaded, I create a new row in an array. These rows are then converted into a DataFrame using pandas and written to an output CSV. All rows containing NaN values are dropped during this stage. (This is necessary because some days do not contain any data for some or all columns.) The code for this can be found in Notebooks/1_CombineDatasets.ipynb.

Third, I created a machine learning model for each numerical feature. For these features, I decided to use the Prophet library, which was made by Facebook engineers. [4] Prophet is a library which lets one easily forecast nonlinear time-series data using an additive regression model, with good support for seasonal trends. [5] This library is relatively popular, and is used internally at Facebook and at several other companies. It

also has an sklearn-alike API, which makes it easy to learn. I used Prophet to train one model on each numerical column, and saved each model out to a JSON file. A reference plot is also created, containing all past data points and future predictions for up to 365 days. The code for this can be found in Notebooks/2_Train.ipynb.

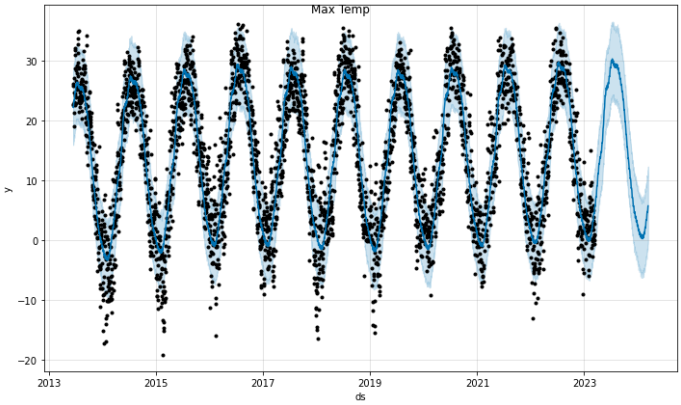
Fourth, I created a machine learning model which attempts to predict if the day will have precipitation ("condition") based on all of the other features. I used GradientBoostingClassifier from sklearn's classification API as my algorithm of choice. 80% of the combined dataset was used to train the model, while the leftover 20% was used to test for accuracy.

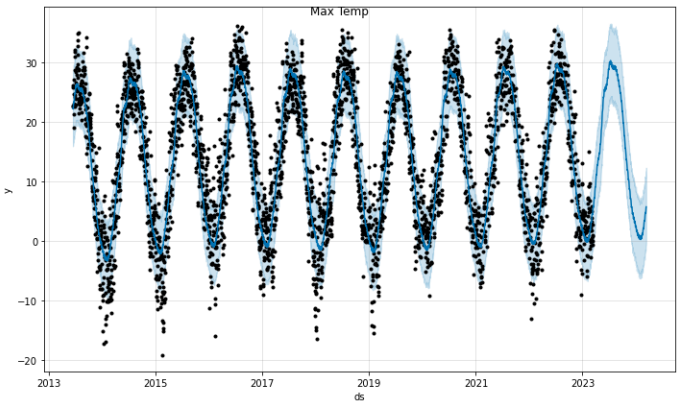
Results

I will first analyze the plots for each numerical feature, as they give an easy to understand overview of the input data points and Prophet's predictions.

Full size versions of the plots can be reviewed in Notebooks/2_Train.ipynb.

<u>Temperature</u>





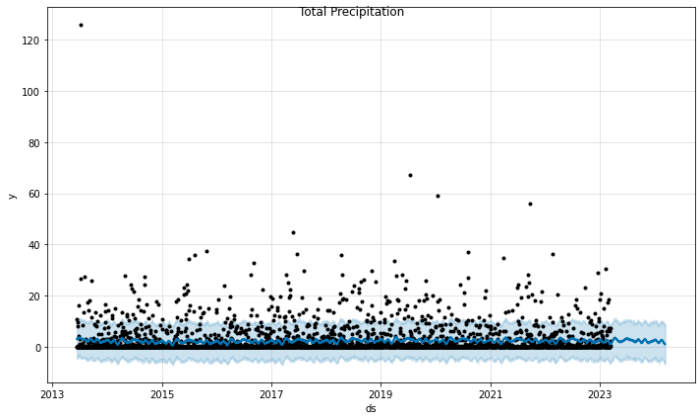
The plot of predicted values for temperature very closely matches the overall trend of the real data.

In the plot of actual data points, one can very clearly identify the high points of summer (where the temperature is likely to be of a higher value) and the low point of winter (where the temperature is likely to be colder). The prediction line follows these trends and has an acceptable margin of error.

Max Gust

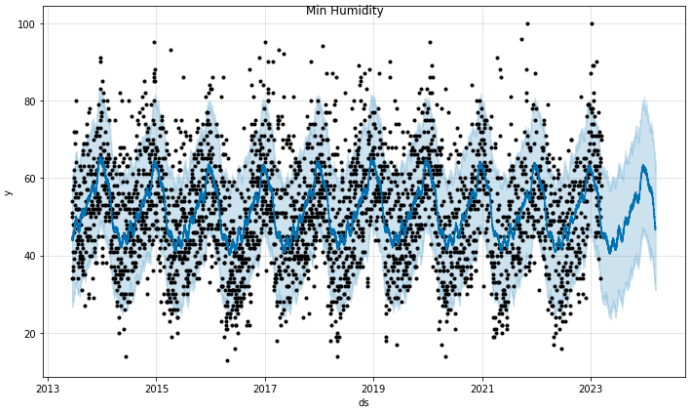
Unfortunately, during training, I found that “max gust” had some unusual numbers which originated from the datasets. It appears that either the equipment or the human operator at the weather station is sometimes not able to measure gust speeds below 31 km/h. When this happens, a “<31” data point is logged. There are around 420 of these data points in the hourly dataset. While I tested some workarounds like changing all occurrences of “<31” to “31” or “0”, these workarounds did not appear to be appropriate after I reviewed the plots. I ultimately elected to not attempt to train a model which predicts “max gust”.

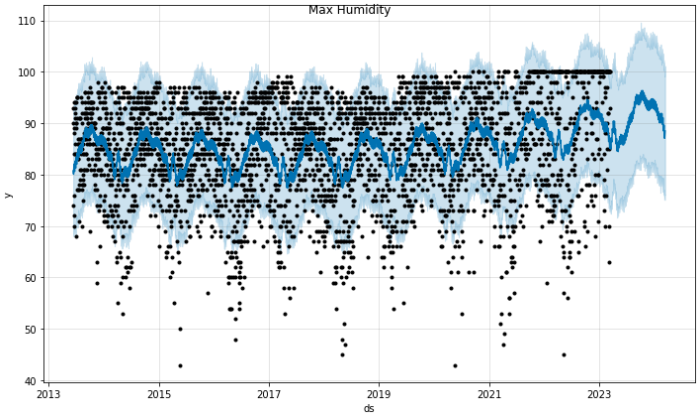
Total Precipitation



It appears that the total precipitation is often clustered around the zero mark (meaning no rain or snow occurred). There are occasional outliers, such as the ~130mm precipitation that occurred sometime during 2013.

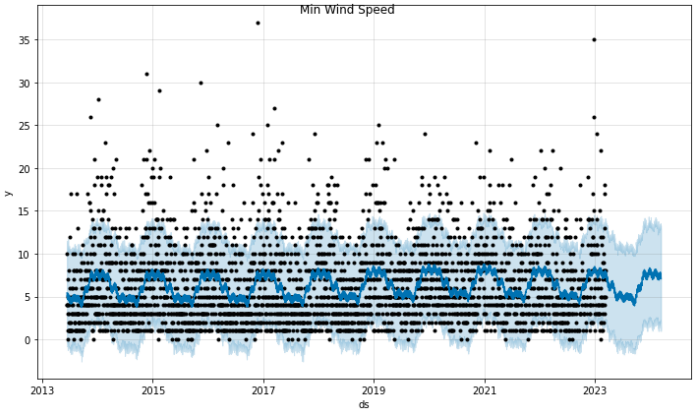
Humidity

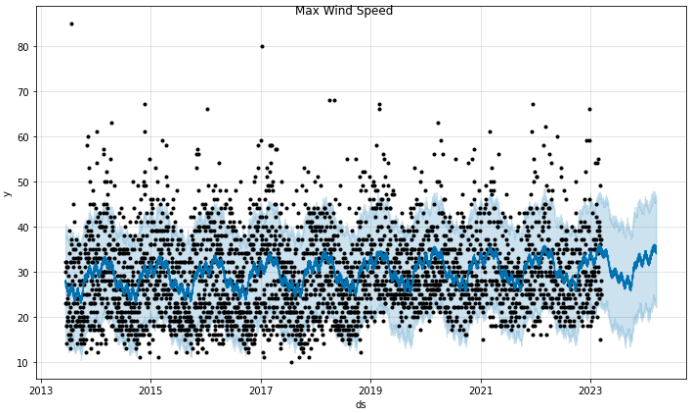




Prophet was able to clearly identify a trend in minimum and maximum humidity. The prediction lines follow a wave-like pattern.

Wind Speed

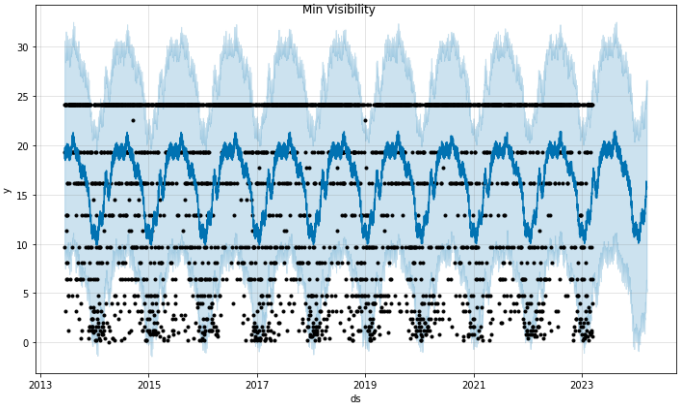


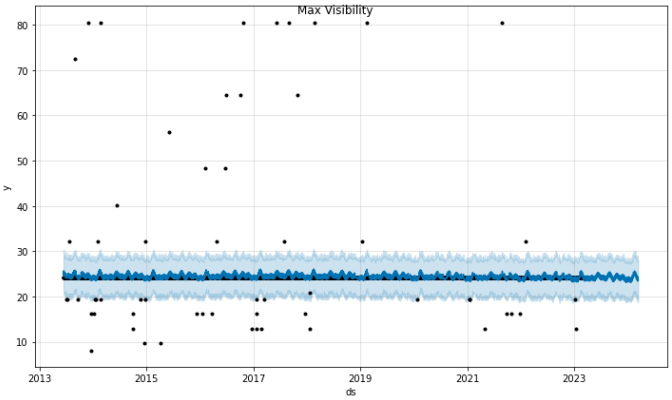


There appears to be an issue with the minimum wind speed feature. Many of the data points appear as stacked vertical lines, suggesting that the measuring equipment has limited granularity at these lower numbers. Because of this, this model might not be usable.

The maximum wind speed plot appears to be unaffected.

Visibility

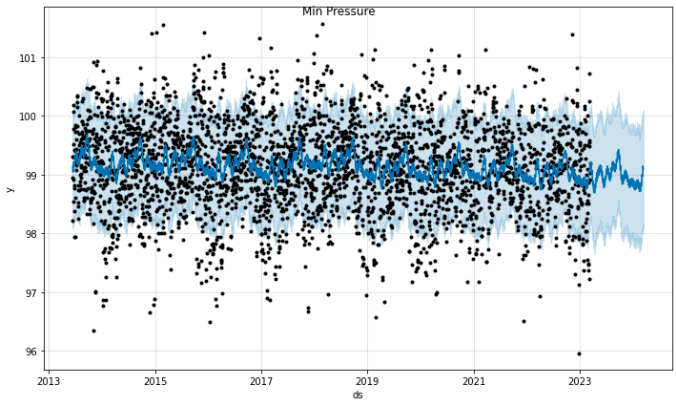


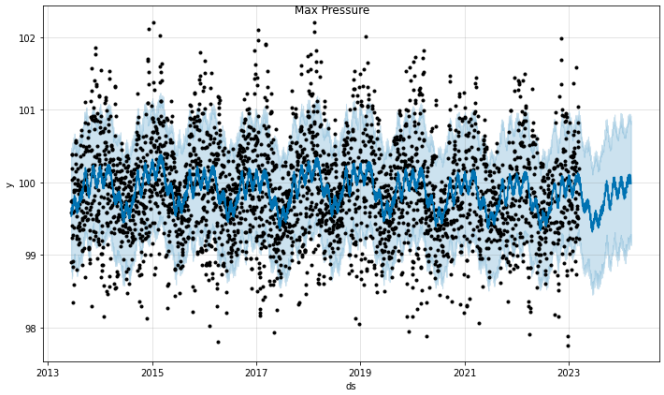


The minimum visibility feature appears to have a similar problem as the minimum wind speed, except it is much more obvious.

Both the minimum and maximum do not appear to have identifiable trends in the data, resulting in a large margin of error for the minimum prediction, and a straight line for the maximum prediction. These models are not usable.

Atmospheric Pressure





Prophet was able to clearly identify a trend in minimum and maximum atmospheric pressure. The prediction lines follow a wave-like pattern, though it is more pronounced in the maximum feature.

Condition

The classifier's accuracy, according to the accuracy_score function, is 86.63%.

Future Improvements

Some features were affected by various inaccuracies and oddities from the dataset. Max Gust's data had invalid data points which made it difficult to train a model with. The Wind Speed data was found to be inaccurate at the lower bounds of the data. This could be improved by using a different dataset that has more accurate data.

The accuracy of the model that predicts whether precipitation will occur could be improved with hyperparameter tuning. For example, I could attempt to tune the n_estimators, learning_rate, and max_depth parameters and see if I can increase the accuracy score with certain values. I could also attempt to check the various other algorithms and see if hyperparameter tuning helps with their accuracy scores.

References

[1] https://www.ecmwf.int/en/about/media-centre/news/2023/growing-role-machine-learning-ecmwf

[2] https://climate.weather.gc.ca/climate_data/daily_data_e.html?StationID=51459

[3] https://collaboration.cmc.ec.gc.ca/cmc/climate/Get_More_Data_Plus_de_donnees/

[4] https://facebook.github.io/prophet/

[5] https://research.facebook.com/blog/2017/2/prophet-forecasting-at-scale/