Predicting Average Temperature in Southern Michigan

Matt Wojno A51414982
Project URL:
https://github.com/mswojno/weatherprediction

ABSTRACT
The main focus of the project that I picked was weather prediction. I was to analyze weather data from days prior to accurately predict the future weather. There are a few attributes you could pick to analyze when trying to predict weather. The attribute I chose to focus on was the average temperature for a particular day. However, the model that I trained could be used to predict another attribute that was collected if desired. The region I chose to predict weather for was southern Michigan. I needed to pick an isolated enough area with enough data to get a precise prediction. This can also be changed within my model if another region of the world needs to be studied.

1. INTRODUCTION
Weather prediction has been an exciting field of study as computers have increasingly got more powerful and new models to assist in the prediction. It is one of the most important areas of prediction in the sense that every person around the world is impacted by weather in their daily lives. The task of checking the weather before someone starts their day has become a commonality. There are a few weather predictions that can be made, such as precipitation percentage, snow accumulation, humidity, the classification of the sky whether its cloudy or sunny, average temperature, max temperature, min temperature, and so on. There is a plethora of these attributes that can be calculated based on the weather data collected. In the scope of this project, I have chosen to focus on average temperature as my predictor value. The reason I chose average temperature was because I believe this is the most important weather attribute. Once I chose this attribute, I had to pick an area to study because the area has to be condensed enough to get an accurate reading. Since I have lived in southern Michigan, all of my life so far, I thought this would be an interesting place to study. I am no stranger to the various weather diversity southern Michigan experiences and was excited to look at some weather data from this region. My goal for this project was to predict the average temperature of the next day in southern Michigan based on weather data from 1988-2018 that I have collected. The weather data also consists of around twelve stations around southern Michigan. I am going to apply linear regression to this data about the preprocessing steps. Some challenges I encountered while getting this data was the NOAA website [2] changed half way through the project. They the format of the columns for the data, which threw off my preprocessing code because I was cleaning data based on a format that was no longer there. On a positive note, the NOAA website did add a better search bar functionality, but they had a download threshold. So, I had to only download a year for each weather station and going back thirty years took some time to collect. Nonetheless, after I collected the data it was pretty straightforward to redo my data preprocessing that I will explain more in section two. After I applied the linear regression, I was pretty happy with the results that I have received as I saw only a two-degree Celsius error. However, there still is room for improvement as the data was filled with outliers as weather data usually has in it.

2. DATA
Data Gathering
In order to get a high accuracy with my model, I had to go back thirty years in my data collection. Therefore, the data that I collected ranges from 1988 to 2018. I collected this data by using the NOAA website [2]. I used their search portal to get a .csv file for a year time span. In total, I collected my data from around twelve stations in southern Michigan. The reason I picked only the southern part of Michigan was because it shares similar weather patterns. This lets me focus on training my model well rather than taking into account outliers, like there would be if I collected data from the upper peninsula of Michigan. The twelve stations/locations I focused my data collection around was Grand Rapids Gerald R Ford International Airport, Saint Joseph Coast Guard Station, Kalamazoo Battle Creek, Jackson Reynolds Field, Detroit Metro Airport, Belle Isle Coast Guard Station, Selfridge ANGB Train, Oakland Country International Airport, Flint FCWOS, Lansing Capital City Airport, Port Huron Coast Guard Station, and Muskegon County Airport. These stations can be seen better in Figure 1 below.

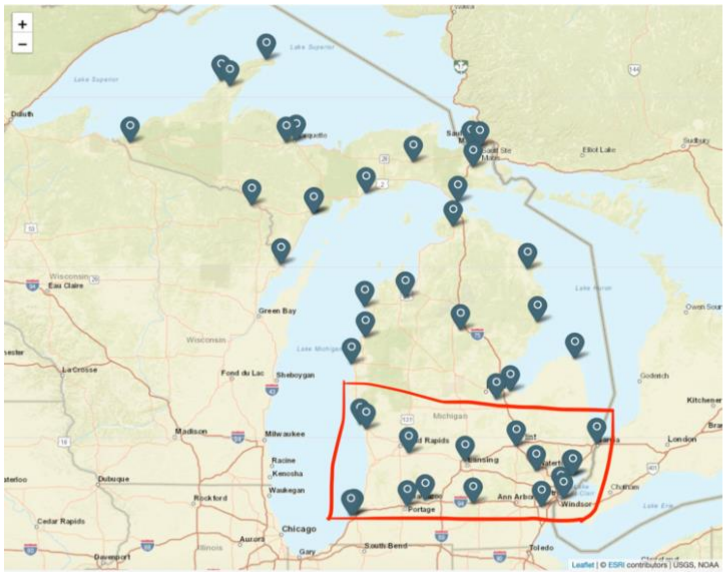
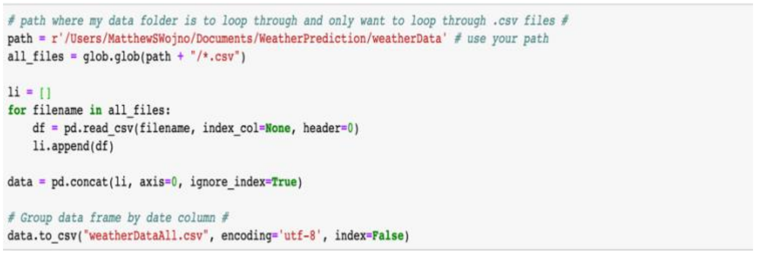


Figure 1. Stations that were used for data collection

The data was initially read into using a for-loop that was looping through the specific folder where the data was kept. This code can be seen in Figure 2 below.



```
# path where my data folder is to loop through and only want to loop through .csv files #
path = r'/Users/MatthewSWojno/Documents/WeatherPrediction/weatherData' # use your path
all_files = glob.glob(path + "/*.csv")

li = []
for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)
    li.append(df)

data = pd.concat(li, axis=0, ignore_index=True)

# Group data frame by date column #
data.to_csv("weatherDataAll.csv", encoding='utf-8', index=False)
```

Figure 2. Code to loop through folder for data

It did not come up until later, that I needed the data to be grouped by the date in order to successfully get the past three days of data. So, I had to write the files into one dataframe that I then wrote to a combined .csv file that you can see in the GitHub called weatherDataAll.csv. I used Excel to sort on the data based on the DATE column. I could have used pandas to complete this task, but I figured it would be much more efficient to use Excel because this was a one-time task. With only one .csv file now, I just read in that and put its contents into a dataframe. This concludes the data gathering step as data is ready to be used for analysis.

Data Cleaning

The next process once the data was collected was the clean it of any outliers, missing values, and anything else that would obstruct the analysis. The columns that the data originally game with was the following: Average Dew Point – DEWP, Average Sea Level Pressure – SLP, Average Station Pressure – STP, Average Temperature – TEMP, Average Visibility – VISIB, Average Wind Speed – WDSP, Indicators, Maximum Sustained Wind Speed – MXSPD, Maximum Temperature – MAX, Maximum Wind Gust – GUST, Minimum Temperature – MIN, Precipitation – PRCP, and Snow Depth – SNDP. The indicators in the column list refers to the numerous attribute columns that would follow most of the attributes. This would indicate some additional information for that particular attribute. For example, precipitation would have an attribute column that could signify if it was foggy or something similar. Therefore, the first step that I took was to remove the attribute column. Though it was interesting, it was not useful within the scope of my study. The next cleaning, I had to carry out was various columns had missing data. This missing data was signified typically by a 9999.9 or 999.9 in the column depending on the attribute. In this case, I could not just impute these values because that was skew my data by getting rid of quality data. [4] What I ended up doing was taking the average of the data that didn't equal the missing indicator and assigned the average instead. This would give a much better reading for these columns now. I did this for the columns of PRCP, SNDP, SLP, WDSP, GUST, MXSPD, DEWP, STP, VISIB, MIN, and MAX. Once this was completed, I now had to check if there were any less obvious outliers in my data. I decided to use a boxplot for each attribute to solve this issue. I was able to see that a few columns had a fair number of outliers. These columns were SLP, STP, VISIB, WDSP, MXSPD, GUST, PRCP, and SNDP. This is quite of few of outliers to deal with in this dataset. Luckily, the columns listed would not be the attributes I am going to focus on later on in this study. If they had been, I would have taken care of the outliers like I had the missing data, which was equal them to the average.

3. METHODOLOGY

Data Correlation

With all the data finally cleaned, it is time to start the analysis to choose what attributes to focus on for the regression model. The first thing I did was to get a graphical representation of the relationship between the various columns and the TEMP column. I compared it to the TEMP column because this was the attribute I am trying to predict, the average temperature. The charts were good resources to picture if there was a strong correlation, but I needed to validate this with numerical values. So, it was time to standardize the data by subtracting the values by the previous value and taking the mean. Then, it was divided by the standard deviation of these values. I could now get a nice reading from the correlation by calling .corr() on this new dataframe object. The relationship the correlation values told me were the following: 0.0-0.2 very weak, 0.2-0.4 weak, 0.4-0.6 moderate, 0.6-0.8 strong, 0.8-1.0. very strong. So, the values that I was looking for was between 0.6-1.0. It can be seen in my python notebook the chart that I generated and chose the values. The columns I chose were TEMP 1, TEMP 2, TEMP 3, MIN 1, MIN 2, MIN 3, DEWP 1, DEWP 2, DEWP 3, MAX 1, MAX 2, and MAX 3. It is important to note here that I needed to drop the current days attributes. That wouldn't be weather forecasting if I took into my model the attributes for that current day. Therefore, I have went back three days and added those metrics into my model. I dropped all the other columns that weren't listed above because they are not needed anymore.

Linear Regression

The final step of this project was to put this cleaned data and picked attributes into a model that will assist in the prediction. The model that I chose was a classic linear regression model. The reason I chose this was because it could be seen in the correlation the linear relationship between these attributes, rather than another form like clustering. Before inputting into the model, I split the TEMP into its own dataframe, and the columns listed above into its own. I then fed these variables into the train_test_split from sklearn. I also chose to split the data with 20% test data for this project, which would train it well.

Code Format & Auxiliary Software

The code I have written for this project is condensed into a lengthy python notebook: Weather Prediction Final Notebook.ipynb. The sections are split up with various comments, but I thought it was condensed enough to be in one notebook.

There was no auxiliary software used within this project. I utilized built-in packages within python to achieve everything.

4. EXPERIMENTAL EVALUATION

This section describes the experimental setup and results that were obtained in this project.

4.1 Experimental Setup

The computing platform that was used in this project was the local computer that the python notebook is run on. There were roughly around 106,000 rows in the data collected, which is a fair amount of data to process. However, it was not enough to have to run on anything besides the local computer.

The baseline methods that were employed in this project to compare my approach against was the industry standard for weather prediction. I research what typical weather prediction algorithms error would be around and judged my own against it. I found that if the model is within a couple degrees of error, then it can be considered a good model [3]. The evaluation metric that I used to then compare was the regressor score (explained variance), mean absolute error, and the median absolute error. These metrics were gotten from using the sklearn.metrics package [1].

4.2 Experimental Results

I was pretty happy with the final results I have received and how it ended up. After running the linear regression model, I calculated an explained variance of 0.96. The explained variance is how far the values are away from the mean. The higher the value is the better for the explained variance, with the maximum value being 1.0. The model also produced a mean absolute error of 2.53 degrees Celsius and a median absolute error of 1.80 degrees Celsius. The error rate that was calculated is fairly decent as this can be seen from some weather predication services currently in the market.

5. CONCLUSIONS

The results of this study yielded a decent error rate as most weather prediction algorithms would yield. However, there is always room for improvement. Some areas I believe this model could be improved is especially with the handling of outliers. It is pretty obvious to notice that weather patterns are pretty unpredictable sometimes and presents an interesting challenge to people trying to predict the weather accurately. There has to be decisions about how to handle these outliers better and when it is considered perfectly cleaned of outliers. Another area I can see improvement here, is getting more exact data. I believe if I had used hourly data then my model could become a little more accurate because I could track changes in the day. This adds its own array of complexities when collecting this data, though. It is also a little more difficult to collect this data and would double the storage. In the end, I was pretty happy with my results and robustness of my project. Like I said earlier, this can be applied to any other attribute within the dataset as well as any other region in the world.

6. REFERENCES

[1] "API Reference." *Scikit*, 2018, scikit-learn.org/stable/modules/classes.html#regression-metrics.

[2] "Global Surface Summary of the Day - GSOD." *Index of /*, NOAA National Centers for Environmental Information (NCEI), 1 Apr. 2019, data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.ncdc%3AC00516.

[3] Seman, Steven. "Assessing Forecast Accuracy." *Assessing Forecast Accuracy | METEO 3: Introductory Meteorology*, 2018, www.e-education.psu.edu/meteo3/node/2285.

[4] Sharma, Natasha. "Ways to Detect and Remove the Outliers." *Towards Data Science*, Towards Data Science, 22 May 2018, towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba.