Data Processing Lifecycle

Ingestion **Sourcing the Data:** This was performed by using API keys provided by the Met Office and the Twitter Feed. Both websites provided us the option to filter what we pulled. From the Met Office we pulled observation and forecast data from London and Glasgow, in order to achieve this we used the task scheduler to run Python scripts for observation and forecast data every hour and every three hours respectively. When it came to gathering tweets, we decided the more data the better, so decided to take all the tweets from the London and Glasgow regions and output them hourly to json files. This gave us the flexibility to reanalyse the data using various methods, we did however have to limit the metadata that came with the tweets as they were taking up too much storage.

**Staging the Data:** The scripts pulled tweets and weather forecasts/observations onto our local machines and soon after imported to MongoDB. We pushed these daily to GitHub to avoid potential loss of data.

**Profiling the Data:** We decided on what analysis we wanted to make. Decisions were made on how to make the querying most efficient. (creating a time period field – nearest 3 hour). Getting rid of unwanted fields.

Munging and WranglingCleaning/Processing the Data:

MetOffice Data: Removing unwanted fields and changing data types within fields from *String* to *Date.*

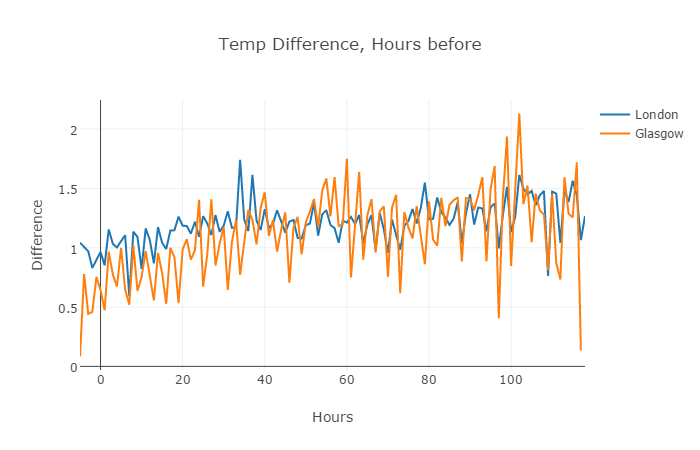
Tweets Data: The cleaning process largely intertwined with a *Machine Learning* toolkit we had developed. The toolkit relied on sensory analysis. We filtered through 2000 tweets containing weather related words (which had been filtered via regular expressions), inputting whether they were actually weather related or not. This built up an algorithm which could accurately determine whether a tweet was weather related or not.

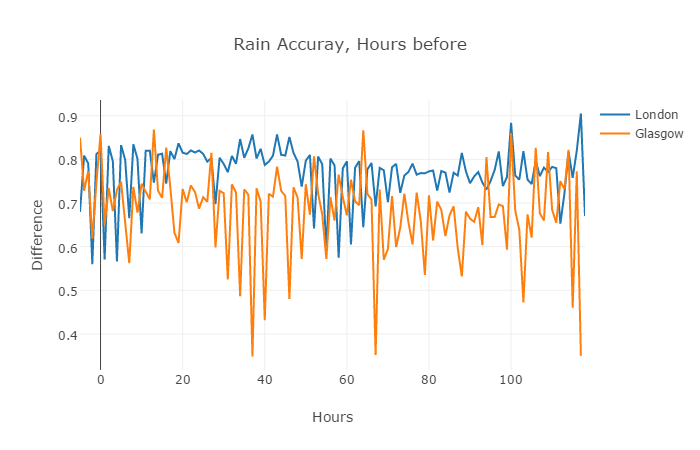
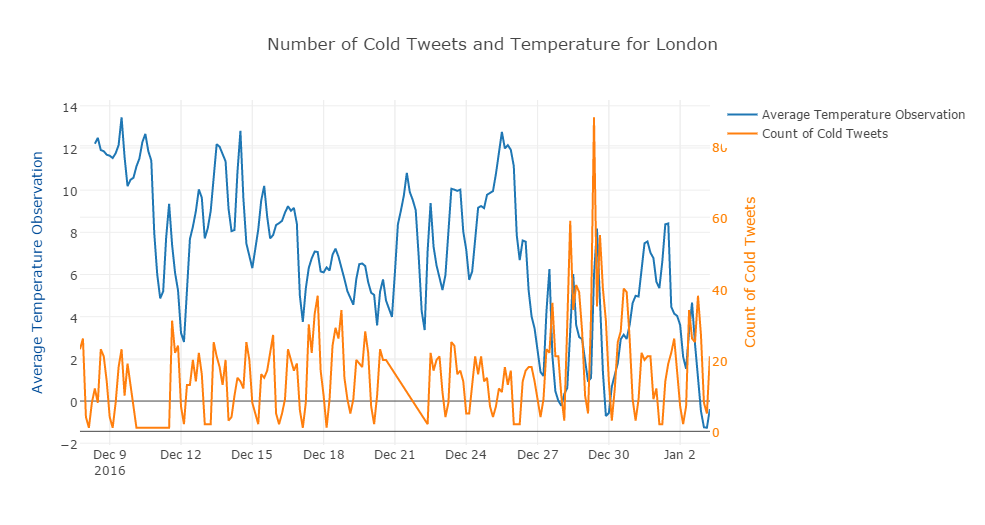
In order to run the algorithm accurately we first needed to filter out all tweets that did not contain any weather related words. This was achieved by updating fields in the MongoDB which matched the same regular expression we had used previously. These tweets could then be looped through via a cursor and given a status of *Weather* or *Non-Weather*. From the tweets which had passed both tests, we finally determined whether they were *Rain, Cold, Warm* or *Windy* via other regular expressions.

To make analysis easier, we included a Date-Time period field. This would allow all the tweets from a relevant period be linked to the corresponding weather observation.

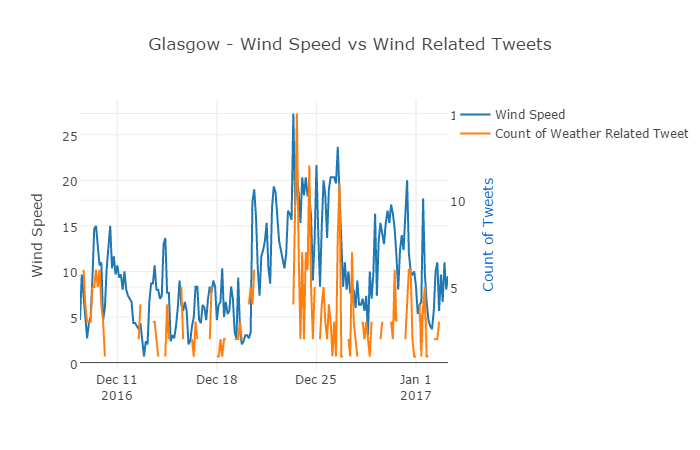
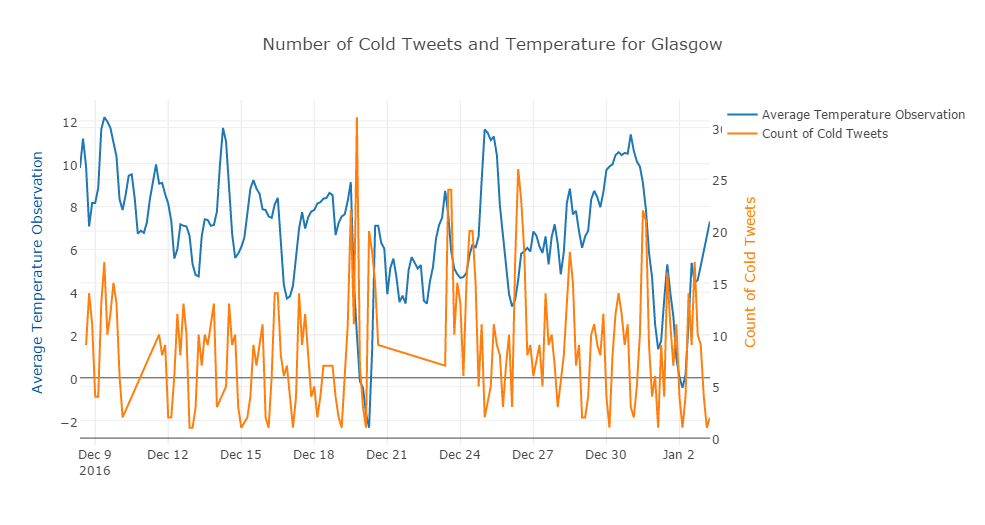
**Modelling the Data:** This was an easy step since we only had two collections (MongoDB tables) – Tweets and Weather Forecasts/Observations. All that was required was a count of a weather type related tweets and then to join the observed data. Aggregation was performed via MongoDB due to efficiency. We were able to easily move our data from MongoDB to a Pandas DataFrame by pickling a cursor list.

Analysis **Hypothesis and Test:** Our objective was largely open-ended. We set out to find trends between Tweets and weather as well as assessing how accurate weather forecasts were.

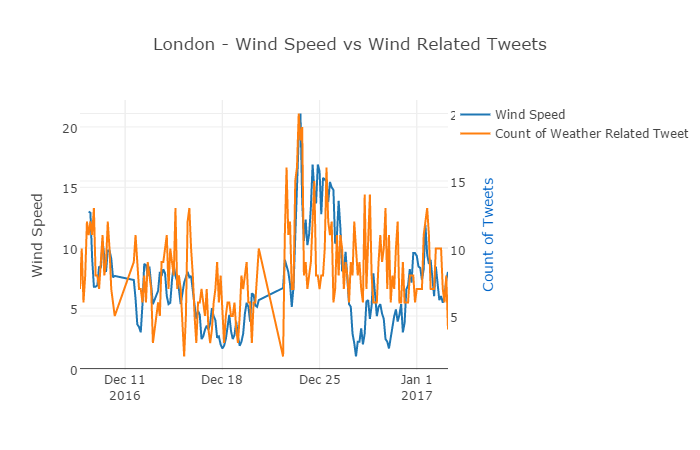
The chart was created by finding the average absolute difference between predicted and observed temperatures over three-hourly periods up to 120hours. As expected, there was a decrease in accuracy as the forecast period moved further into the future. However, there was surprisingly no trend to suggest that at 0 hours, the accuracy would be 100% as was assumed. This suggests a difference in forecast and observation techniques.

The rain accuracy difference was calculated by giving precipitation a value of 1 and no precipitation a value of 0. The absolute difference from the weather forecast was taken. For example, a forecast of 70% chance of rain would give a 0.3 difference if it did rain, and a 0.7 difference if it did not. The rain forecast was significantly more accurate in London, perhaps due to Glasgow’s geographical location. However, again, surprisingly there was no real strong correlation as the forecast period moved further into the future.

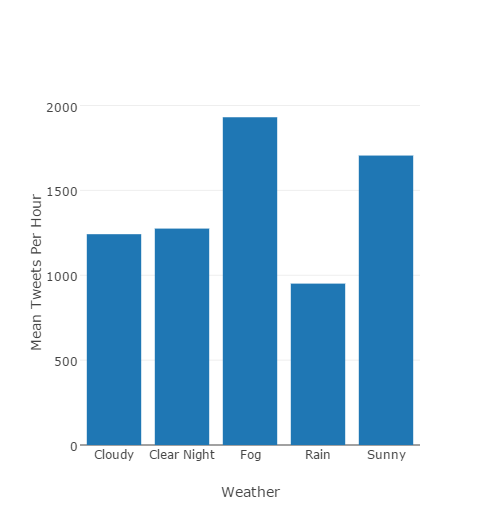
This graph shows the trend in tweets related to cold weather against the actual observed temperature during that time period. Unfortunately, we were not able to find and strong correlation in London.

****Similarly, no real correlation in Glasgow.

Unlike the temperature, people appear to be much more likely to tweet about the weather when it is windy.

****Again, there appears to be correlation between wind and incoming wind related tweets in London. The constant spikes are most likely due to a sharp drop in Twitter activity overnight.

Looking more broadly, we thought it would be interesting to see which weather types linked to the highest number of total tweets:

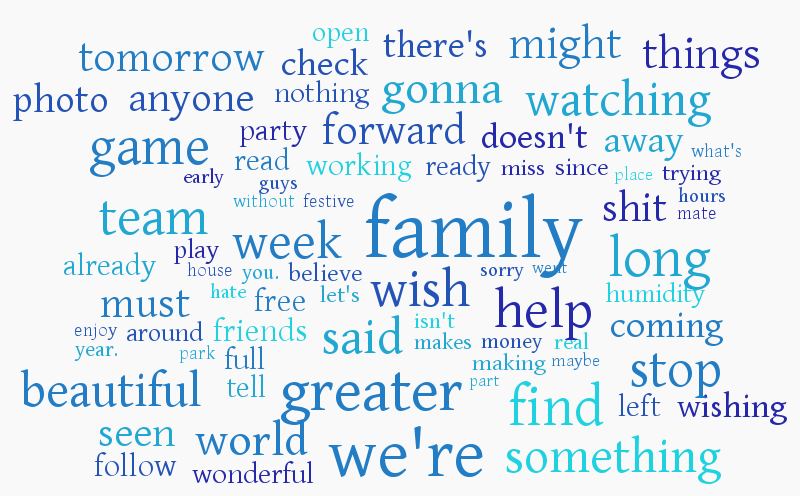


Clearly if there is foggy or sunny weather there is more tweets total than any other weather type, this disparity could be used as a different method of determining the weather occurring now.

The final two graphs below shows the words used most often during observed weather types across London and Glasgow in the month of December, discounting ‘common’ words (designated by the nltk library) and the top hundred most popular words across all weather types. In order to process, the 1.6 m tweets and generate the ‘word clouds’, we used a combination of aws s3, databricks and worditout.com.



Cloudy Tweets



Rainy Tweets

The graphs show the most common words are nothing to do with weather, which is to be expected, but there are many differences in both the words and their frequencies. Hence, it might be possible to determine what type of weather is occurring based on these properties as opposed to just looking at weather related words as we did previously.

By creating a new training set, on one side taking tweets we know were created during rainy periods and on the other side the rest of the tweets then we can use the machine learning algorithm to classify new tweets as rain or not. The same method could be applied to a variety of weather types.

**Assumptions:** That our machine learning was accurate.