Supervised Learning to Inform a Move - A Weather Prediction Model

We can all recognise some common challenges concerning a big move from. The climate seems to be one of those features affects us all when considering a new home. Someone who loves the weather in always-sunny LA is not very likely to love the weather during a snowed-in Minneapolis winter. A New Yorker might not be quite prepared for that second winter in Seattle that sends everyone running for sun. For me, the question is: “Can Colorado winters prepare me for a Nor’easter in Vermont?”

Much to do About the Weather

I find that when people tell me they think an area has rough winters I say I can take it. But what if I’m wrong?. Among the faults that nay-sayers and root-setters will choose from to dissuade a curious explorer is the brutal Burlington winters, and muddy North East springs. I don’t buy into the hype. People are fickle, and our memories don’t work very well for the strictly statistical task of recalling the weather of years past. The average person, when recalling a winter, tends to think about the most brutal storm of the year and focus on that. We are drawn to outliers.

Rather than relying on emotional opinions and half recollected winters of past. I thought to collect and plot the actual data regarding the weather in Burlington. The US government collects troves of data, for the benefit of no private investors. This data is available for everyone and every year work is being done to make that information more accessible. NOAA has decades worth of historical weather data and it is ripe for practicing, testing, and using to inform a move.

This is a proven interest and companies companies like AccuWeather make their dollars by repackaging this government data and selling it with strategic PR witchery and competitive claims. I myself, enlisted the help of a less sharkish company to help me build the data to answer my question, Dark Sky. Of course, the progress of 5G mobile carrier services is interfering with satellite imagery of water vapour which we rely on heavily for our modern forecasts and opening us to risk of unforeseen natural disaster. Tornadoes of typical short warnings but imagine having little to no warning of a hurricane. It isn’t likely that people will be

using less data, rather, we will find new ways to innovate and predict the forecast more accurately again.

```
        <
# sets an API call and collects all the features in one dataframe
        def collect_data(time, runs):
            resp =
requests.get('https://api.darksky.net/forecast/{0}/44.4759,-73.2121,{1}?exclude=flags,alerts,hour
ly'.format(api_keys["darksky_key"][0], time))

            print(f"init : {resp.status_code == requests.codes.ok}")

            first_call = resp.json()

            features = get_features(first_call)

            midnight = first_call['currently']['time']
            midnight

            df = pd.DataFrame([features.values()], columns = features.keys())

        ########################################
        ####  Loop  ( ͡° ͜ʖ ͡°)         ####
        ########################################

            for i in range(0, runs):
                df_rise = get_sunrise(df)
                df = create_new_row(df, df_rise)
                df_noon = get_noon(midnight)
                df = create_new_row(df, df_noon)
                df_set = get_sunset(df)
                df = create_new_row(df, df_set)
                midnight, df_night = get_next_midnight(midnight)
```

```
            df = create_new_row(df, df_night)
            print(f"Day: {i}")
        print(f"Time: {midnight}")
        df.reset_index(drop=True)
        return df
```

>

That work starts here, with naive weather prediction algorithms, and there is a lot of work to do. For a link to my github repo containing all the code for this project -click here- The goal of this module is to predict the outlook of a given day, given no extra ordinary features that a standard weather tower wouldn't have. It will do this in a rather naive way using the data from just one sensor bundle at a single gps location. This model is naive in the sense that it considers the weather of a single location is only one point in a closed global system, and that to work in real life a weather prediction must attempt to account for that of that.

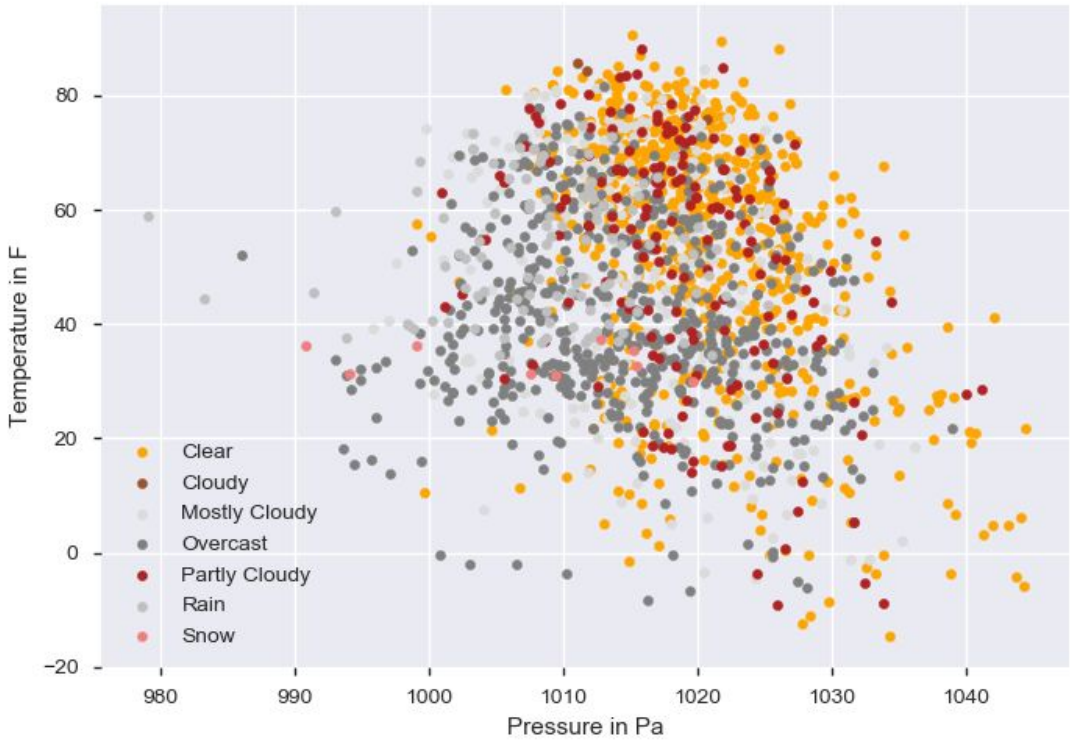
This model is in climate change denial.

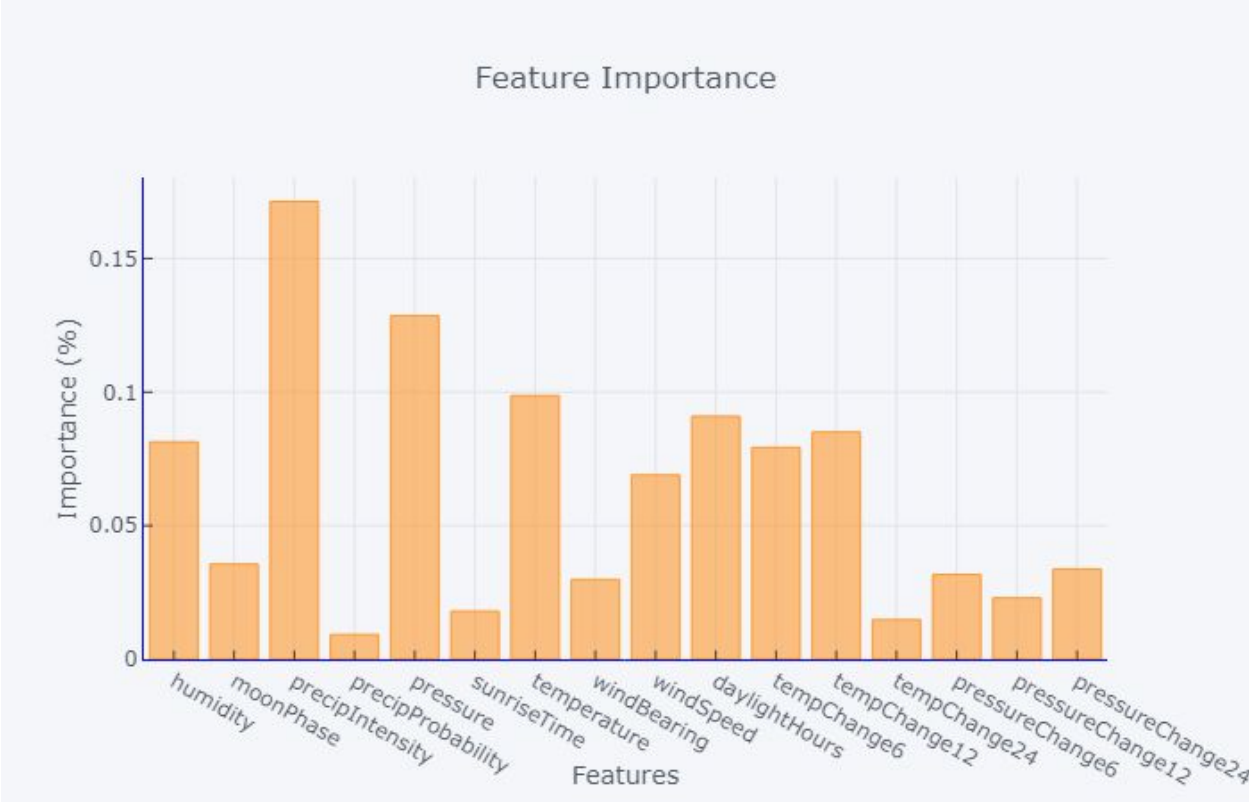
That said, it did surprisingly well predicting a defined set of weather patterns.

	Value	Count	Frequency (%)
Clear	689	38.1%	
Overcast	518	28.7%	
Mostly Cloudy	268	14.8%	
Partly Cloudy	164	9.1%	
Drizzle	54	3.0%	
Light Rain	48	2.7%	
Foggy	42	2.3%	
Humid	6	0.3%	
Flurries	5	0.3%	
Light Snow	4	0.2%	
Other values (4)	9	0.5%	

Repackaging the Weather

For simplicity, these outcomes were condensed into 7 different types of weather.


Temperature in F
80
60
40
20
0
−20
980
990
1000
1010
1020
1030
1040
Pressure in Pa
Clear
Cloudy
Mostly Cloudy
Overcast
Partly Cloudy
Rain
Snow

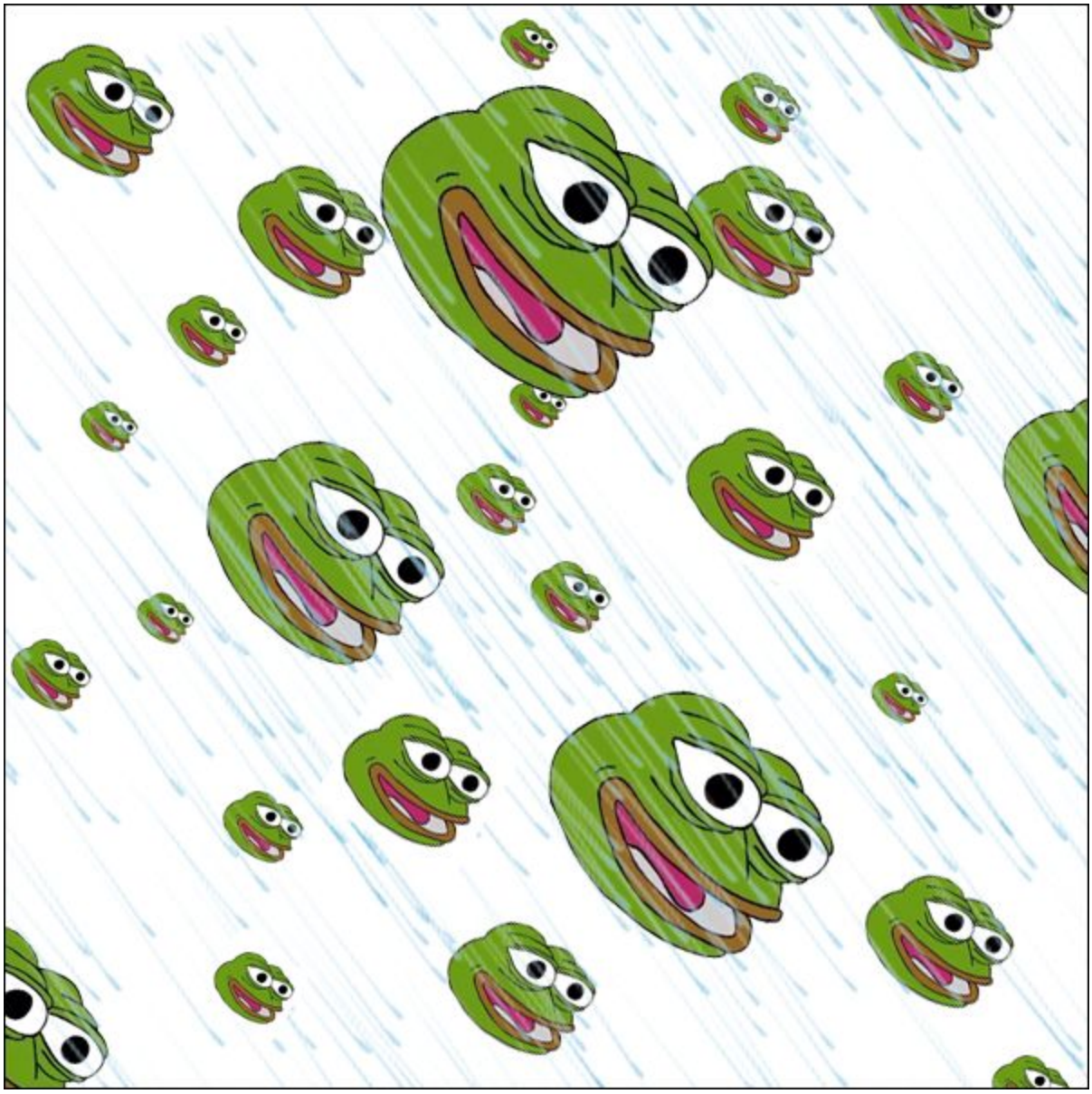


One interesting and unexpected result was that the Precip Probability - calculated by further feature engineering on the part of Dark Sky, turned out to be one of the least influential features of the data. What does this mean? One thought is that the this data is all historic, meaning that precip probability was akin the the percent (%) chance of rain on the day that the information was sourced from. Ever a weather skeptic, I like to joke that a 50% chance of rain really means that it is just as likely for the weather to be great than is it to be precipitate at all, and if then, how much? A drizzle is hardly the same thing as being caught out in a downpour. These subtle distinctions are beyond the scope of this first-time model.

I've Never Seen it Rain Cats or Dogs

The first step in predicting weather is appropriately categorising weather upon agreed upon outcomes. Supervised learning is fantastic in this regard because we know the different outcomes of the weather, it can be sunny, rain, snow etc. If it decided, quite suddenly, to rain something we've never seen before it would hurt the accuracy of our supervised learning model.

Kmeans, one of the methods explored in this module, can be used as an unsupervised model and correctly classify raining frogs as their own category distinct from other types of rain.



<

from sklearn.ensemble import, RandomForestClassifier

forestclf = RandomForestClassifier()

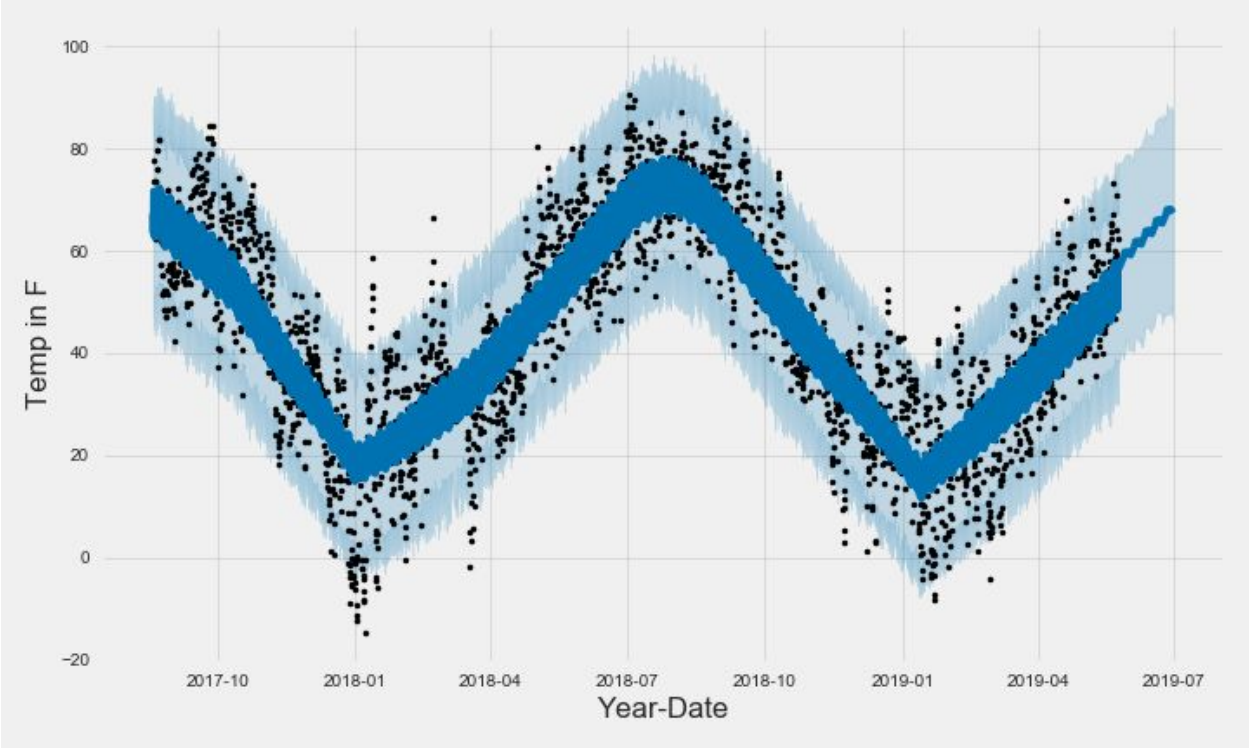
param_grid = {

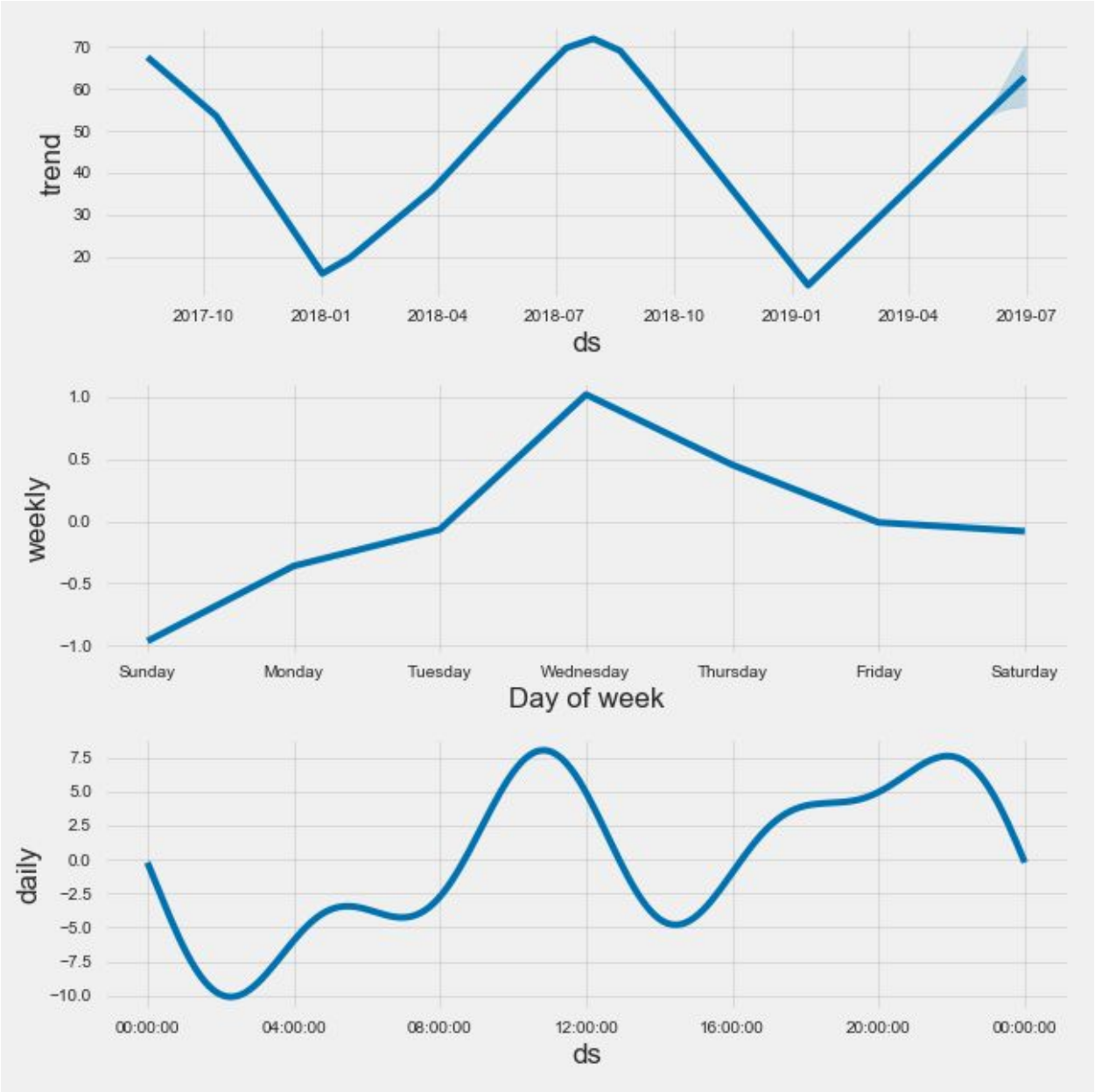
```
    "criterion": ["gini", "entropy"],
    "max_depth" : [4,5,7,8],
    "max_features" : ['auto'],
    "max_leaf_nodes" : [None],
    "min_impurity_decrease" : [0.0],
    "min_impurity_split" : [None],
    "min_samples_leaf" : [1,0.5],
    "min_samples_split" : [2,3,4],
    "min_weight_fraction_leaf" : [0.0],
    "n_estimators" : [75,100,200,250],
}

grid_forest = GridSearchCV(forestclf, param_grid, cv=3)
grid_forest.fit(x_tr, y_tr)

best_parameters = grid_forest.best_params_

print("Grid Search found the following optimal parameters: ")
for param_name in sorted(best_parameters.keys()):
    print("%s: %r" % (param_name, best_parameters[param_name]))

training_preds = grid_forest.predict(x_tr)
val_preds = grid_forest.predict(x_ts)
training_accuracy = accuracy_score(y_tr, training_preds)
val_accuracy = accuracy_score(y_ts, val_preds)

print("")
print("Training Accuracy: {:.4}%".format(training_accuracy * 100))
print("Validation Accuracy: {:.4}%".format(val_accuracy * 100))
>
```

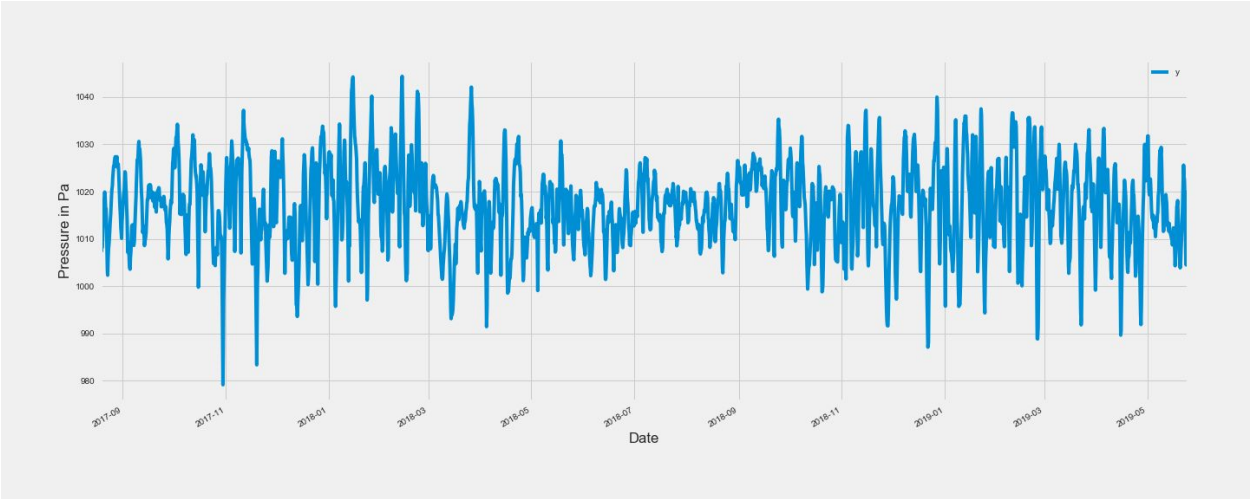
What was arrived at when running through ensemble methods was hovering around a 67.47% accuracy for the best performing models. A random guess would yield 17% accuracy.

Now Here's Time Series for the Weather

As this weather experiment continues we turn our eyes to the wind and look to see what's coming. If we look at individual features we can build on our 2 years' experience to predict what what tomorrow will look like. The following are is a break down of key features used in this dataset and a reasonable prediction as to what they will look like going forward through June.




trend
70
60
50
40
30
20
2017-10
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
ds
weekly
1.0
0.5
0.0
−0.5
−1.0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
daily
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
00:00:00
04:00:00
08:00:00
12:00:00
16:00:00
20:00:00
00:00:00
ds



Where as the temperature looks very cyclical, pressure changes frequently and rapidly. This makes sense as temperature is very dependant on sunlight and season. Given more time it would be interesting to see if we can explain some of the noise on the temperature charts using the corresponding pressure readings.

A lot was learned from this work. Yes, Burlington does get cold. No, The snow/rain isn't all that extra ordinary. Sure the winter's might suck, but for me, this weather experiment is far from over. There improvements to be made on the classification models by feature engineering and resampling. There are more detailed predictions to be made by shifting the dataset. Climate conditions as they are feel even more precarious and it can be very inviting to try and seek out the trends in the numbers. If ever we needed a reminder that math was the language of the universe, we might look to the weather and hopefully someday (soon) we will understand.