Capstone Project Proposal: Chicago Weather Predictor

Problem Being Solved

The goal of the project is to come up with a solution that makes shorter term weather predictions for a Target Location. Chicago has been chosen for the Target Location due to better feasibility of that choice.

First, Chicago is far from any ocean (weather reports are readily available for land but not sea) and second, there are no mountain ranges in close proximity (mountains are known to make weather more locally variable). Finally, Chicago is known to experience “Lake Effect” due to the proximity to Great Lakes, which makes for a fun challenge by introducing a bit of chaos into the picture.

Weather is notoriously difficult to predict longer term so we shall focus on predicting weather **within the next 24 hours only**.

“Weather” comprises many different characteristics. For this project we shall focus only on those weather variables that make the most sense to a layperson planning their day:

- Temperature
- Wind Speed
- Absence or presence of precipitation (whether it is raining or snowing)
- Absence or presence of cloud cover (whether the weather is “fair”)

Data to be Used

To train the Models to be used to generate weather predictions we shall use historic hourly data. Such data is aggregated and maintained by the National Oceanic and Atmospheric Administration (NOAA). It is available for hundreds of North American locations, for several decades and for every hour until the current time. Examples of the variables included into each hourly report include but are not limited to:

- All 4 variables we aim to predict
- Wind Direction
- Dew Point
- Pressure
- Humidity

The data is available from NOAA directly for free through a data ordering mechanism. As that mechanism is unwieldy and does not land itself to automation we shall also use data from

sources that make NOAA data available in a more automatable fashion, such as **wunderground.com.** Using such machine-friendlier sources becomes especially important at the point of use of pre-trained models, when we need to pull recent weather data in real-time to feed to the models.

Approach

Theoretical Justification is as follows:

- Weather forecasting in use today treats Earth's atmosphere as a fluid that is governed by laws of physics
- Those laws are expressed as complex partial differential equations with recent readings at different positions plugged in as parameters; thus, given a number of readings at different locations, those equations yield weather characteristics of interest for the future
- Therefore, it appears that future weather at the Target Location can be expressed as a complex function of recent weather characteristics at that location, as well as locations at near and moderate distances away
- Supervised Machine Learning algorithms exist specifically to discover such complex functions from data sets
- As mentioned above, NOAA provides hourly historical data for all significant population centers in North America. Models will be trained on that data; recent records from that data will be fed to the models to make predictions.

We shall tackle both regression and classification problems. Predicting Temperature and Wind Speed are examples of the former while predicting Precipitation and Fairness are examples of the latter. Experimentation will be needed to determine the best Supervised Learning Algorithms for each model we shall build. We'll consider both simpler (e.g. Linear) and more complex (e.g. Multi-Layer Neural Networks) approaches.

We shall build a number of different models Specifically, we'll have a separate model per lookahead interval. For example:

- Model 1: Temperature 6 hours from now
- Model 2: Temperature 12 hours from now
- ..
- Model N: Precipitation 24 hours from now

MODEL_COUNT = VARIABLE_COUNT x LOOKAHEAD_INTERVAL_COUNT

As per the above, VARIABLE_COUNT is 4. A reasonable LOOKAHEAD_INTERVAL_COUNT is also 4: 6h, 12h, 18h and 24h from now. This gives us **16 as the total number of models** involved in forecasting weather.

Deliverable

End deliverable is twofold:

- A public website that displays weather forecast data in HTML
- A repeatable, documented process (pipeline) for training and re-training the models, at least partially automated. Specifically, the following shall be true: *given raw historic hourly NOAA reports, it is easy for the developer to train the required models and re-deploy them for use by the public website*.

The Public Website shall include the following features:

- Forecast Page: show hourly weather forecast for Chicago for the next 24 hours in tabular format
- Prediction Quality: for recent past (e.g. 30 days or 60 days) show how predicted weather variables differed from actual ones, including summary stats (e.g. what % of time we got Precipitation right)
- Model Info: a log containing information on when each model was trained, on what data (especially time frame, e.g. 01/01/2011 - 12/31/2017) and, optionally, model performance at training time

The following functionality is optional and may not be included:

- Automatic updating (retraining) of models with the most recent data: pre-trained models shall be included in the deployment and the only way to update them is to retrain offline using the pipeline process and re-deploy the application
- Support for multiple Target Locations: while the project idea is easily generalizable to cities other than Chicago, the end deliverable will not target such other cities
- Forecast Page showing forecast side-by-side with the forecast from a public service (such as wunderground). The Prediction Quality page does give adequate indication of historic forecast quality.

Deployment and Resources

- Deployed in an instance (or instances) provided by an established Cloud provider, such as AWS EC2
- No special computation requirements: resources shall not exceed what is provided by higher end Consumer hardware
- Deployment shall be properly containerized eliminating the need for manually installing any dependencies
- Pre-trained models shall be included in the deployment in addition to the Web Application that serves User requests