Reproducing Results

This document details how to reproduce the results for the Power Laws Forecasting Challenge from start to finish!

1. **Structure of directories**

The main directory holding all project materials is called `project`. Within the `project` directory are three sub-directories:

- `data`: This holds the raw data as well as the data after formatting
- `scripts`: This holds the two scripts used to format the data and make the final predictions
- `submissions`: This initially starts out empty and is used to hold the predictions

2. **System Set-up**

This code requires **python 3.5.1**. I believe the code may work on newer versions of Python, but this is the only version on which is has been tested! There are three required Python packages:

- `numpy==1.14.1`
- `pandas==0.22.0`
- `scikit-learn==0.19.1`

The double equals (==) indicates the required version of the package. All 3 of these can be installed using pip from the command line with the following commands:

- `pip install numpy==1.14.1`
- `pip install pandas==0.22.0`
- `pip install scikit-learn==0.19.1`

After installing Python and the required packages, you are good to go. The code may also work with newer/older versions of these packages but these are the tested and verified versions!

3. **Directory Set-Up**

To set up the folder structure, create a main directory called `project` that holds three subdirectories: `data, scripts, submissions`. The raw data provided from the competition should be placed in the `data` directory. The four required files with the original names are:

- `metadata.csv`
- `weather.csv`
- `train.csv`
- `submission_format.csv`

It is important that the raw files names are not modified! (Or if they are, you will need to change the code to reflect the new names).

Place the following Python scripts in the `scripts` directory.

- `data_format.py`
- `predict.py`

The `submissions` directory should be initially empty.

4. Running the Model Pipeline

Generating the final predictions requires two steps:

1. Running the `data_format.py` script which loads in the raw data, cleans it, transforms the features, and joins the appropriate information together.
2. Running the `predict.py` script to build the final model, train the model on the formatted training data, make predictions on the test data, and save the predictions.

To run the `data_format.py` script, open a command window in the scripts directory and run `python data_format.py.` If all goes well, this should take about 15 minutes and will create two new files in the `data` directory. The files, along with their sizes are:

- `train_corrected.csv`: 1.4 GB
- `test_corrected.csv`: 263 MB

After this step, there will now be a total of six files in the data directory. After making sure that the data was correctly formatted, we can move on to generating predictions!

Before running the `predict.py` script, open the script and change the file name on line 162 (it's within single quotes in the call to `make_submission_file`) to the desired name of the submission file. This is the only modification required. Then, open a command prompt in the scripts directory and run the `predict.py` script using the command `python predict.py`. Depending on your machine, this may take a few hours.

At the conclusion of running the script, a message will appear in the command prompt indicating where the predictions were saved and the corresponding file should be in the `submissions` directory.

5. Remarks

That's all there is to it! The development process was completed across several Jupyter Notebooks, but I moved all the final code to scripts to run on a more powerful machine. This is my general workflow: start in a Jupyter Notebook to quickly iterate through ideas and move to Python scripts when I no longer need to make many changes. Python scripts are simpler to run on high-powered computing clusters which is where I did most of the predictions, allowing me to make small changes to the models and determine the performance.

As highly non-deterministic algorithms, the random forest and extra trees combined model will generate different predictions on each run. This can be addressed by setting the `random_state` parameter in the model creation. During the competition, I did not use a random seed for submissions.