IESO Data Pipeline and Data Visualization

Methodology Doc

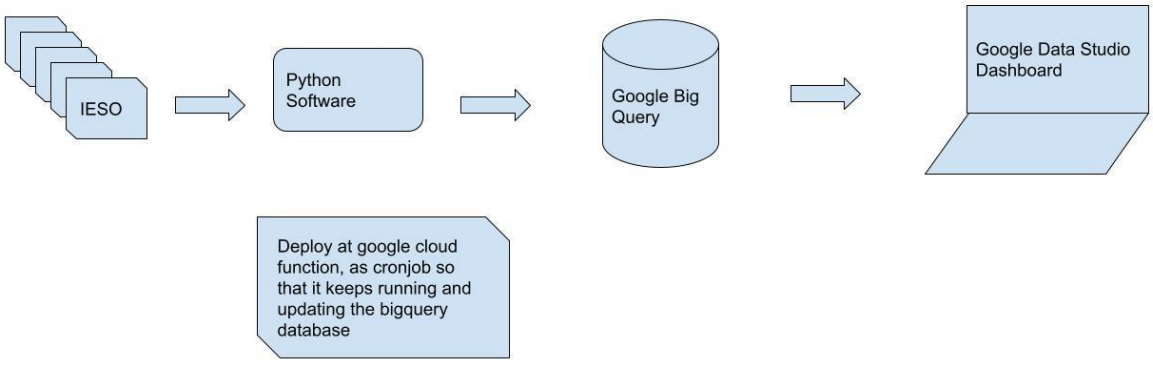
Table of Contents

➔ Project Brief

Complete end to end data pipeline –

1) Main link for pulling various reports http://reports.ieso.ca/public/
2) Every link under this might have a slightly different report layout, some are in csv and some are in xml
3) The main idea is to get the data in the database by day and by hour from past couple of ye
4) Links to capture the data { http://reports.ieso.ca/public/Adequacy2/
   ::http://reports.ieso.ca/public/RealtimeMktPrice/ }
5) The first link has the independent variables and the second one is the dependent one for price forecasting in the future
6) Once we have this data in our database in GCP, we look at building the dashboard and training/testing the model based on user defined inputs on the front end of the dashboard.

➔ Tech Architecture



➔ Methodology

- DATA EXTRACTION

➔ We chose Python as the primary programming language to develop the software in. Other tools and libraries used are: pandas, BeautifulSoup, Google Cloud Platform {Bigquery, Scheduler, Cloud Functions}, Tensorflow.

➔ All the data is scraped and fetched from the relevant reports links in a structured manner and stored in their respective dataframes.

Relevant reports being: Adequacy2, GenOutHourly, Demand and RealtimeMarketPrice

➔ Separate buckets are then made in Cloud Storage which are then filled with their respective dataframes.

➔ Data is then loaded from Cloud Storage in Bigquery tables for efficient processing. All this is done through a python function/client, automated with a Cronjob/Google Cloud Scheduler.

➔ We now use this hourly data to forecast the ont_ene price for next 24 hours. Now we can schedule the prediction to happen on an hourly basis as well as on a daily basis. Normalization for the data is done using min-max scaling and mae is chosen as our performance metric in all the models

- FORECASTING USING MACHINE LEARNING

➔ Once our data is processed and loaded in GCP's BigQuery we build and run our various ML Models and pick the best one out to predict the future fuel/energy price.

➔ The models tested are {LSTM univariate/multivariate, GRU} for modeling it as a time series problem and {ANN Regressor, Random Forests regression} for modeling it as a regression problem given we have some assumptions about future supply demand variables. Of these models, ANN regression turns out to be the best model for our use case with an MAE(mean average error) in the range of (7.51 - 12)

MAE is chosen as a preferred metric over RMSE to not punish the outliers by a lot.

➔ LSTM/GRU :: For our time-series models we have 3 main config parameters:

look_back - { how much past data to train the model on};

n_steps_in, n_steps_out - { at each step how many past hours to look at and how many future hours to predict on}. Note - With increase in n_steps_out our models complexity and computation time increases

nb_epochs - { the number of passes of the entire training dataset the ML algorithm completes}

The data is first loaded from the past old adequacy and mkt_price data as well the bigquery current data and stacked vertically. It is then split into a Sequential Series data for our input in time series models and then finally passed through our LSTM neural network layers. A dropout of 0.2 is set to prevent our model from overfitting.

The results are then pushed to a predictions* table inside our Bigquery dataset.

➔ ANN :: The way ANN regressor works is - It takes the past few months (look_back is that config parameter) data from Bigquery as a training set and finds the appropriate weights and relations to model the dependency of ont_ene price variable onto the various independent variables.

The data is then normalized using MinMaxScaler and clipped using z-score analysis to make it more robust.

Once loaded it is then split into train test datasets with an 80:20 ratio, and finally passed through our neural network layers. Once trained it then predicts the next look_ahead future price outputs from the already present independent variable values from the data.

➔ After completion we then report the predictions back to a Bigquery dataset named 'Predictions_24hr.ANN' on our GCP dashboard.

◆ DATA VISUALIZATION REPORT

➔ Once done with ML Modeling, we then generate a data visualization report on Google Data Studio to gain further insights.

➔ This report consists of :

A pie chart about the distribution of the distribution of fuel generated by each fuel type. Nuclear fuel appears to be the most generated fuel type (60%) while solar and biofuel the least (~0.5%)

A stacked column chart about the distribution of fuel generated each month indicating fuels being generated the most around the months January and August.

Fuel Generation output by hour ending for each fuel type is then plotted to gain further insights for which times each fuel is generated the most and when it's the least.

With the disproportionate scaling between the various fuel generation outputs we use a log scale line chart to better understand and view the time series visualization of fuel generated during each quarter of the year.

➔ Manual Configuration

➔ STEPS TO ACCESS AND CONFIGURE GOOGLE BIGQUERY

Link to Bigquery Dashboard --

https://console.cloud.google.com/bigquery?authuser=4&project=ieso-dashboard

Authenticated users can access all the datasets on this dashboard under 'External Connections'.

One can also run manual SQL queries through the query tab after selecting a dataset.

➔ GOOGLE CLOUD STORAGE

Link to Cloud Storage Buckets --

https://console.cloud.google.com/storage/browser?authuser=4&project=ieso-dashboard

All the csv files to the datasets, before being uploaded to the bigquery, are loaded here under 'amol_javahire' for faster and optimal processing. The remaining buckets are there for cloud functions files.

➔ GOOGLE CLOUD FUNCTIONS

Link to Cloud Functions --

https://console.cloud.google.com/functions/list?authuser=4&project=ieso-dashboard

The cloud functions to extract and load the data from the report links {Adequacy, Demand, etc.} are stored here.

Also the ML models for forecasting and predicting the future values of the market fuel price are saved in Cloud Functions.

1st Gen Cloud functions offer a max timeout period of 540 seconds while 2nd Gen offers a period of 3600 seconds and better compute instances. So accordingly all the functions are deployed.

To Trigger the cloud function for ml-ann,gru,lstm one has to follow the following steps:

-- Open the cloud function > Navigate to 'Testing Tab' > Click 'Test in cloud shell' > Press Enter

Following which one sees through the ‘Logs’ tab for any errors and bugs, if any. The code for the cloud functions can be viewed under the ‘Source’ tab.

To edit and test a cloud function with new parameters ::

1. Open the cloud function. Head to the ‘Edit’ tab. Click next after reviewing the memory allocated and timeout parameters.
2. For certain cloud functions, one has to download the cloud function’s .zip file, while for others user can directly edit the function’s main.py file and redeploy it.
3. Open/Extract the .zip file. Change the parameters inside the main.py file and recompress all the files in the same structure.
4. Go again to the browser window under the Cloud Function’s edit tab and choose upload zip file. For storage bucket choose ‘gcf-v2*’ as the bucket source.

   And then deploy and test the updated function.

➔ GOOGLE CLOUD SCHEDULER

Link to Cloud Scheduler --

https://console.cloud.google.com/cloudscheduler?authuser=4&project=ieso-dashboard

All the cronjobs/schedulers for the above cloud functions are deployed here. User can change the frequency of the cronjob by modifying the respective frequency parameter with a cron schedule expression.