Vo_Fredrikstad

This project is basically working with **Fredrikstad_store** data. The steps of the process are broken down into sections. First, the preprocessed work has been accomplished, such as raw data concatenation, monthly and overall data visualization. The missing data can be easily captured using these representations. Second, the missing data can be repaired using various procedures, and new data will be obtained as a result. The final phase is to make predictions about future data. The instructions below will make working with scripts much easier.

Information about directory

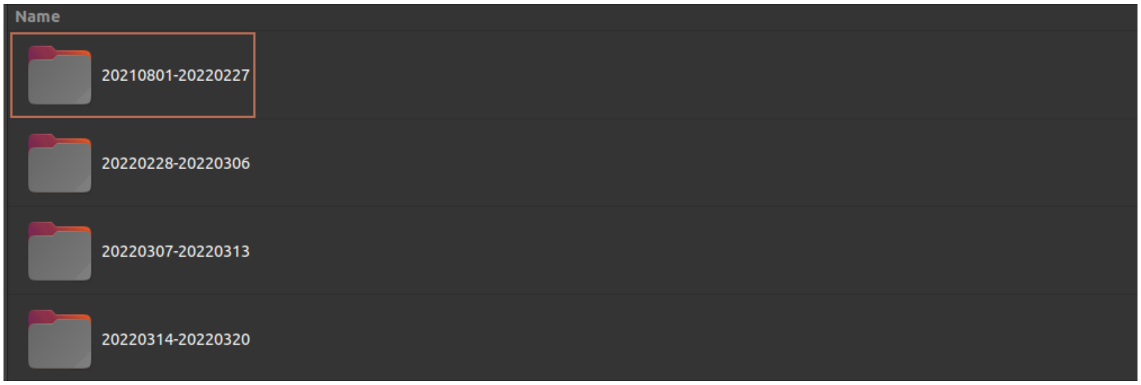
In the Fredrikstad directory, there are two folders:

1) Vo_fredrikstad_data
2) results

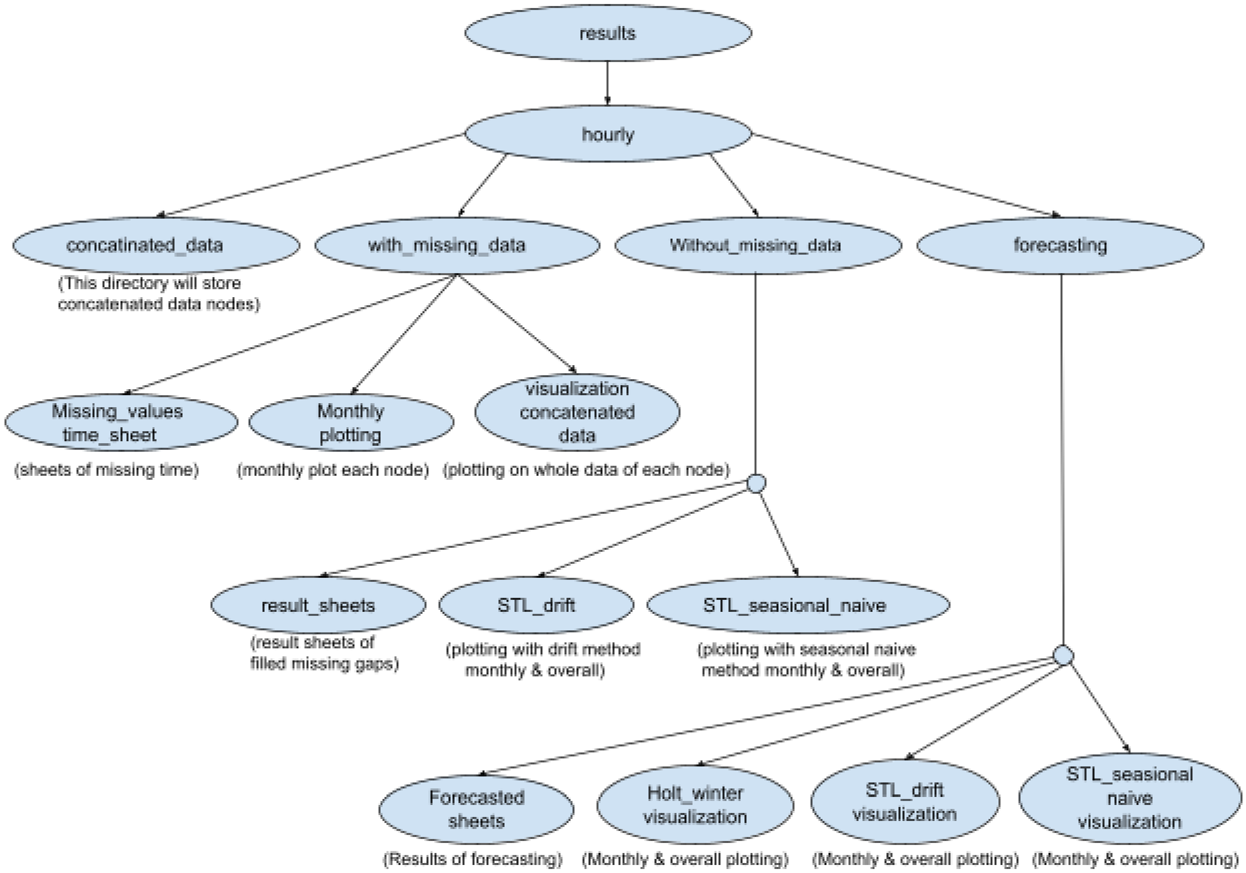
The three scripts are present in the directory including one requirement.txt:

3) preprocess_hourly_data.py
4) Fill_missing.py
5) Forecasting.py

1) Vo_fredrikstad_data The raw data in the various folders are stored in this directory. As indicated in the diagram, each folder must have moved with the name from the start date to the finish date. Every folder contains daily and hourly data of each node. The Python script will automatically work with the hourly data in the folder automatically.



2) Results This directory is used to keep the results carried out from each python script. The hierarchy below will easily help to understand the path of various results that will be stored when the python scripts will be run.


results
hourly
concatinated_data
(This directory will store concatenated data nodes)
with_missing_data
Without_missing_data
forecasting
Missing_values time_sheet
(sheets of missing time)
Monthly plotting
(monthly plot each node)
visualization concatenated data
(plotting on whole data of each node)
result_sheets
(result sheets of filled missing gaps)
STL_drift
(plotting with drift method monthly & overall)
STL_seasional_naive
(plotting with seasonal naive method monthly & overall)
Forecasted sheets
(Results of forecasting)
Holt_winter visualization
(Monthly & overall plotting)
STL_drift visualization
(Monthly & overall plotting)
STL_seasional naive visualization
(Monthly & overall plotting)

3) preprocess_hourly_data.py This python script will assist in preprocessing the data by first retrieving all data node-by-node and concatenating it according to time after that the visualizations will be performed on the overall concatenated data and by month. The outputs will be stored in the results folder

4) Fill_missing.py The purpose of this python script is to fill in data gaps. The moving average method will be used to fill the first week of each node data. Because each model requires at least one week of continuous data to train. Then the models will be trained in a loop in order to discover the missing data. The algorithms will use all prior continuous data as training data and forecast the next entities to fill in the gaps.

5) Forecasting.py This python script is solely used to forecast data in the future. We will get ongoing data as a result of executing the *Fill_missing.py* script, and there will be no missing entities. Future entities will be anticipated using current ongoing data.

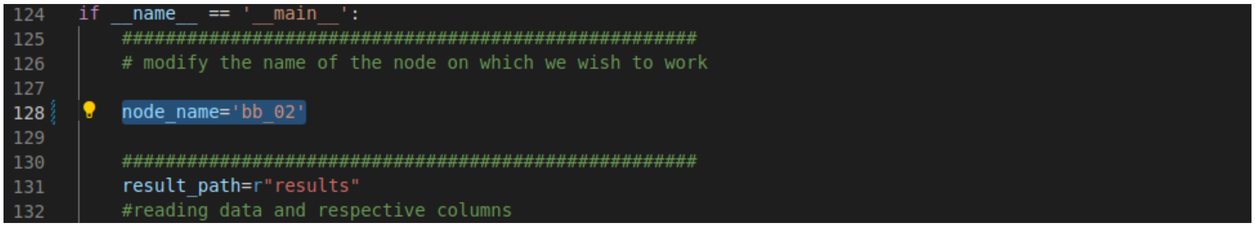
Steps to run the project

The following steps will help to run the project:

- **Create a virtual environment and install required packages** Open the project in any IDE, create a virtual environment, and install required packages by using the requirements.txt file. Run the following commands stepwise in the terminal:
  1) virtualenv fredrik_env
  2) source fredrik/bin/activate
  3) pip install -r requirements.txt

- **Concatenate raw data and perform initial visualizations** The script *preprocess_hourly_data.py* assists in concatenating the data and performing some preliminary analysis with visualizations. As shown in the upper figure, new raw data must be placed in the folder with the name of the starting date to the end date. The merged data will be saved in the results folder under the subfolder 'concatenated data.' Timesheets and visualizations that are missing will be saved in the 'with missing data' folder.

- **Fill missing gaps** The '*Fill_missing.py*' script will be used to fill the gaps. The data will be retrieved from the directory 'concatenated_data'. Before running the script, the user must update the node name as indicated in the figure below. Only that node name should be replaced in the script, as that is where the missing gaps will be filled. For example, to fill gaps in the bb 10 node, simply replace the node name with bb_10.



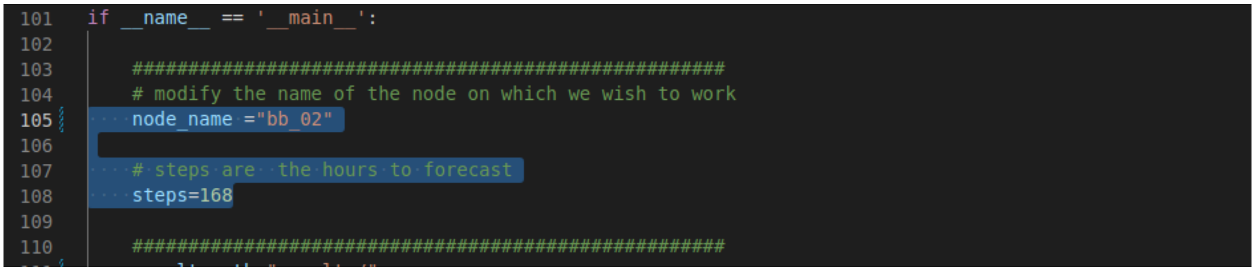
```
if __name__ == '__main__':
    #####################################################
    # modify the name of the node on which we wish to work

    node_name='bb_02'

    #####################################################
    result_path=r"results"
    #reading data and respective columns
```

The result sheets of filled missing gaps and visualizations of continued data will be stored in the *without_missing_data* directory in the results folder.

- **Forecasting** After obtaining data that is free of missing values, the data can be used to forecast the future using historical data. The *forecasting.py* script will assist in data forecasting. The node name and steps ('The hours are represented by the steps, so if we want to forecast for 18 hours, enter steps=18') must be defined in the script before it can be run. Replace the node name and steps with the following instructions, which are presented in a section of the script.



```
if __name__ == '__main__':

    #####################################################
    # modify the name of the node on which we wish to work
    node_name ="bb_02"

    # steps are  the hours to forecast
    steps=168

    #####################################################
```

The predicted outcomes are kept in the forecasting directory. The forecasted results sheet and visualizations will be saved in the forecasting directory's various folders.