Table of Content:

Objective 2

Dataset and Data Modeling 3

Tools and Technologies 4

Data Preprocessing and Analysis Steps 4

Data Updates 5

Scenario Analysis 5

Questions:

1. What's the goal? What queries will you want to run? How would Spark or Airflow be incorporated? Why did you choose the model you chose?
2. Clearly state the rationale for the choice of tools and technologies for the project.
3. Document the steps of the process.
4. Propose how often the data should be updated and why.
5. Post your write-up and final data model in a GitHub repo.
6. Include a description of how you would approach the problem differently under the following scenarios:
    6.1. If the data was increased by 100x.
    6.2. If the pipelines were run on a daily basis by 7am.
    6.3. If the database needed to be accessed by 100+ people.

PROJECT WRITE-UP

I. Objective

- The goals of this project are as follows:
    - To analyze sensor data collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004.
    - To combine internal and external weather data to improve our data analysis and predictions.
    - To apply time series and machine learning analysis to gain insights from the dataset and make accurate forecast.
    - To deploy machine learning model to the server and apply stream data from the sensors.
- Queries:
    - *Time-series forecasting*: being able to predict the right temperature in a facility based on internal and external weather data can help the users to allocate proper energy resources in the building that may lead to cost savings from energy efficiency.
        - For time-series analysis, we use AutoRegressive Integrated Moving Average (**ARIMA**) model. It is a class of model that captures a suite of different standard temporal structures in time series data.
        - The three components of ARIMA model are:
            - AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.
            - I: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
            - MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.
    - *Machine Learning classification*: Given a set of data and machine learning model, users will be able to predict which facility areas are using the most energy and be able to redesign the facility to distribute and deploy energy more efficiently.
        - We use **Logistic Regression** algorithm for the machine learning classification model. This algorithm is suitable to predict multiclass predictions (ie. sensor id), based on a set of input data (eg. temperature and time dimensions).

- Since the sensor data is streamed and collected continuously, Spark and Airflow can be very useful. Our sensor dataset has about 2.3 million rows for a two-month worth of data. Due to the sheer number of rows, Spark can help distribute our data ETL process and analysis on multiple servers. Also, since sensor data has a lot of variances, data preprocessing step requires tedious tasks. Airflow will be helpful to automate these tasks to reduce human errors and run schedules based on certain parameters.

II. Dataset and Data Modeling

- Dataset #1: Sensor Data
  - Format: text file
  - Source: Intel Lab
  - Data size: ~2.3 million rows
  - Schema: date (date), time (time), epoch (integer), mote_id (integer), temperature (float), light (float), humidity (float), voltage (float)
  - Raw data example:
    2004-03-31 03:38:15.757551 2 1 122.153 -3.91901 11.04 2.03397

- Dataset #2: Sensor Location Data
  - Format: text file
  - Source: Intel Lab
  - Data size: 58 rows
  - Schema:
    - sensor id (integer),
    - x coordinate (float),
    - y coordinate (float)
  - Raw data example:
    1 21.5 23

- Dataset #3: External Temperature Data
  - Format: json file
  - Source: OpenWeather
  - Data size: 588 rows
  - Schema: latitude (float), longitude (float), nested weather data
    - datetime (datetime),
    - temperature (float),
    - pressure (float),
    - dew point (float),
    - humidity (float),
    - precipitation (float),
    - wind (float)

    - Raw data example: {"lat":51.50853,"lon":-0.12574,"forecast":[{"forecast_dt":1546300800,"list":[{"main":{"temp":8.08,"pressure":1034.848,"dew_point":4.63,"grnd_pressure":1029.908,"humidity":78.61},"precipitation":null,"wind":{"speed":2.959,"deg":292.075},"dt":1546300800}
- Output data:
    - Format: parquet file
    - Partition: ‘year’ -> ‘month’ -> ‘sensor_id’

III. Tools and Technologies

- Data preprocessing: Python, Panda, Numpy, Matplotlib
- Data analysis: sklearn, statsmodels.tsa.arima_model
- Data storage: parquet file (AWS S3)

IV. Data Preprocessing and Analysis Steps

- Data Preprocessing
    1. Dataset #1: weather dataset from OpenWeather
        a) Import json file
        b) Inspect raw data format
        c) Convert nested dataset into Panda dataframe
        d) Convert ‘datetime’ column to Panda datetime format
    2. Dataset #2: sensor data
        a) Import csv file
        b) Inspect raw dataset
        c) Remove null and duplicate rows
        d) Merge `date` and `time` columns and convert it Panda `datetime` format
        e) Aggregate `datetime` by seconds
    3. Dataset #3: sensor location data
        a) Import csv file
    4. Combine all dataset into Panda DataFrame
        a) Remove null and duplicated rows
    5. Export clean data to parquet file
        a) Partition by ‘year’, ‘month’, and ‘mote_id’
- Machine Learning Training
    1. Import clean dataset from Parquet file
    2. Split `datetime` into `year`, `month`, `day`, `hour`, `minute` dimensions
    3. Prepare dataset for Machine Learning training (ie. input parameters and label)

4. Split training and test dataset (ie. ⅔ of dataset is used for training and ⅓ for test)
5. Scale `temperature`, `humidity`, `light`, and `voltage` using SKLearn's `StandardScaler`.
6. Instantiate SKLearn's Logistics Regression model to train dataset.
7. Test and evaluate model against test dataset.
8. Measure accuracy score (Note: our model achieves 96% accuracy).

- Time Series Forecast
  1. Load clean dataset
  2. Inspect `temperature` time series data
  3. Eliminate outliers and selecting time windows for training
  4. Train dataset using ARIMA model
  5. Forecast `temperature` using the model
- Deployment (note: this task is out of scope since Kafka is not covered in Data Engineering Nanodegree)
  1. Save clean dataset in parquet file to AWS S3
  2. Deploy Machine Learning and ARIMA model
  3. Set up data streaming tool, ie. Kafka, and apply model to make real-time predictions.

V. Data Updates

- Since sensor data is collected continuously through our sensors in milliseconds, it's important to update our data all the time. Kafka and other data streaming tools will need to be used to update our data.
- However, we may not need to update our time series analysis and machine learning model very often since the process can be time consuming and costly. We may need to update our models in longer time period, eg. monthly, to make sure the models capture the structural changes in our continuous dataset.

VI. Scenario Analysis

- *If the data was increased by 100x*: Since sensor data is not transactional by nature, we can use NoSQL database, instead of RDBMS. Also, since sensor data is very large, we need to carefully partitioned it carefully. If we intend to keep the data for a few years, we can partition it by time period. However, if we intend to add more sensors in the future, we can partition our data by sensor id.
- *If the pipelines were run on a daily basis by 7am*: We can set Airflow schedule to automatically run the ETL process.
- *If the database needed to be accessed by 100+ people*: We can use distributed database, eg. Amazon Redshift, and partition database so the queries from 100+

people are not necessarily targeted to a few partitions that may slowdown the queries.