Datasets and Resources Provided

This document contains information about the data provided and some additional resources to guide you.

1. Dataset Details

The main dataset provided is **Eskom data** sourced from https://www.eskom.co.za/dataportal/. The dataset is provided as a csv file and contains data regarding power generation and consumption for the past 5 years.

Note that, depending on the problem you are trying to solve, you will not need to use all the columns of the dataset. Be selective when importing the data so that you do not end up processing data that you do not need to complete your project. It is intentional that "too much" data is provided. The challenge faced by data scientists and data engineers is that they need to filter through available data and use only the data they require to solve the problem.

A **data dictionary** is provided for the Eskom dataset, giving details on each column.

2. Additional Datasets

A **weather dataset** is provided containing daily weather data across several South African cities. This includes information on temperature, solar energy rating and wind speed.

If you need additional data to solve your problem, do not hesitate to try and find that data. However, in the interest of time, if the data is not readily available, please mention what additional data you would need to solve your problem, and how you would obtain and use that data in your solution. You are scored on the completeness of your solution, not on how much code you have written!

Hint: If you are looking for cost data (say you wanted to know how much Eskom spends on diesel fuel), Eskom discloses these figures in their financial statements, which are available to the public.

3. Additional Resources

3.1 Telling a data story

- https://powerbi.microsoft.com/en-us/data-storytelling/
- https://online.hbs.edu/blog/post/data-storytelling

3.2 Data Solution Design

A useful way to represent your system is using the C4 Model. These are a way of describing and visualizing your system at different levels of detail:

- Context diagram – a starting point showing how the system fits into the world around it.
- Container Diagram – the scope of the system itself with high level building blocks.
- Component Diagram – the scope of a container, showing all its components.
- Code Diagram – usually UML class diagram to show how code components are implemented.

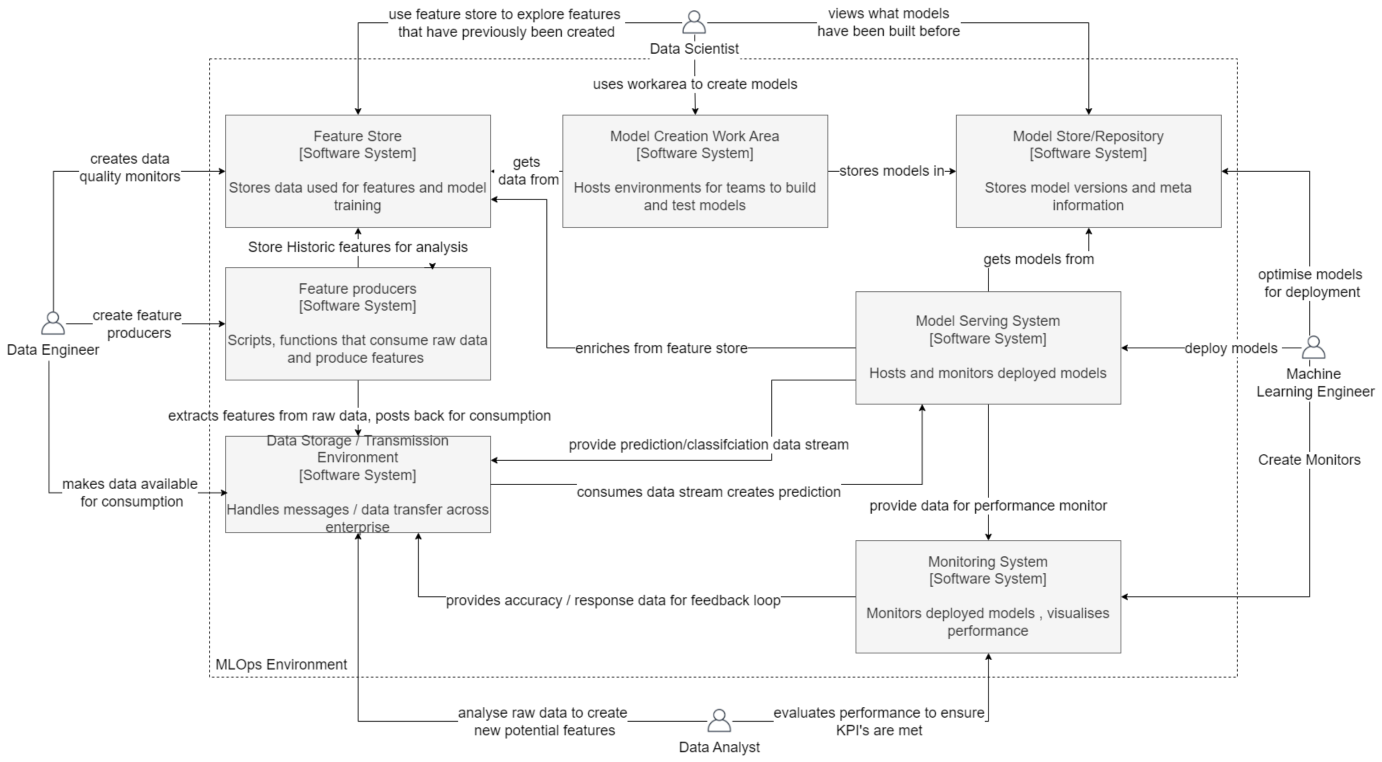
Documentation can be found here: https://c4model.com/

You may like to create the context and container diagrams. If you would like to go into more detail for the data science portion of the hackathon, you may create a component diagram for the code prototype.

3.3 MLOps

- https://towardsdatascience.com/a-gentle-introduction-to-mlops-7d64a3e890ff

The following diagram shows a general example of how data projects are broken down by component and by the role. With data solutions in industry, you are often expected to cover all the roles of Data Scientist, Data Engineer, ML Engineer and Data Analyst! Specialising in just one role is rare.



use feature store to explore features that have previously been created
Data Scientist
views what models have been built before
uses workarea to create models
Feature Store [Software System] Stores data used for features and model training
gets data from
Model Creation Work Area [Software System] Hosts environments for teams to build and test models
stores models in
Model Store/Repository [Software System] Stores model versions and meta information
creates data quality monitors
Store Historic features for analysis
gets models from
optimise models for deployment
Feature producers [Software System] Scripts, functions that consume raw data and produce features
create feature producers
Data Engineer
Model Serving System [Software System] Hosts and monitors deployed models
enriches from feature store
deploy models
Machine Learning Engineer
extracts features from raw data, posts back for consumption
Data Storage / Transmission Environment [Software System] Handles messages / data transfer across enterprise
provide prediction/classifciation data stream
consumes data stream creates prediction
makes data available for consumption
Create Monitors
provide data for performance monitor
Monitoring System [Software System] Monitors deployed models , visualises performance
provides accuracy / response data for feedback loop
MLOps Environment
analyse raw data to create new potential features
Data Analyst
evaluates performance to ensure KPI's are met

A typical data project has the following components:

Model creation workspace: An environment that hosts compute that allows Data Scientists/ML Engineers/Data Analysts to train and deploy models. This environment can be as simple as a cloud-based compute or a Virtual Machine (VM). Think of this as Google Colab for the Hackathon!

Model Store: Every trained model, even those that have not been deployed should be stored in a model store. A trained model is usually compiled as some form of object file, while this file will usually contain all the hyper parameters and transformations required to run the model. The model store should reference any artifacts and data that was used to create the version of the object. Storing non-deployed models allows the team to understand what has already been done and stops the team from re-inventing the wheel and wasting time.

Data/feature store: The data/feature store is more than just a simple SQL database. Each data source (sometimes called a feature) that is created as part of the modelling process should be stored and versioned. Models are based on their data and in order to recreate a model one needs to be able to access the same version of the data set that was used in the training process. This includes the validation and holdout sets that were used to test the models.

Model monitoring system: A central location to view the performance of any models historically trained, that monitors the current performance of models that are deployed. This can be a simple as a reporting service that allows stake holders to review performance.

Data monitors: ML models degrade over time due to various natural phenomena collectively known as data drift. The underlying statistics of most data sources are consistently changing, this could be due to a wide variety of reasons from inflation in economic systems or degradation of equipment in industrial processes. When this occurs, we define these models as drifting and their performance will start to slowly decay over time. To combat this performance degradation, the underlying data that models are trained on, can and should be monitored. Monitoring should also, in a mature environment, include downtime and quality monitors.

Automated retraining pipelines: Any experiment that is run on new or existing data should be triggered by an automated pipeline. This pipeline should by default store models generated in the model store. Data sets should be stored in the data store for later investigation should the need arise. Any performance evaluation should be automatically conducted without need for any manual intervention from the team.

Automated deployment pipelines: Just like any code project an ML project is no different, deployment should occur automatically with as little human intervention as possible.