**Energy consumption prediction using MLflow**

**Done by: Ayah kouider**

**Student ID: 2104620**

**Introduction:**

Energy plays a crucial role in our daily lives, powering everything from households to large scale industries. In recent years sustainability has become a global priority with a focus on responsible energy usage and environmental impact reduction.

In this project we will focus on forecasting daily energy consumption using MLflow to track experiments, manage models, and deploy the best-performing model in a reproducible and monitorable way. This system can be used for sustainable resource management, cost optimization, and the development of smart infrastructure.

**Dataset:**

The dataset used in this project is the Individual household electric power consumption dataset from UCI machine learning repository. It consists of ~2.1 million minute level data from 2006 to 2010.

**Features:**

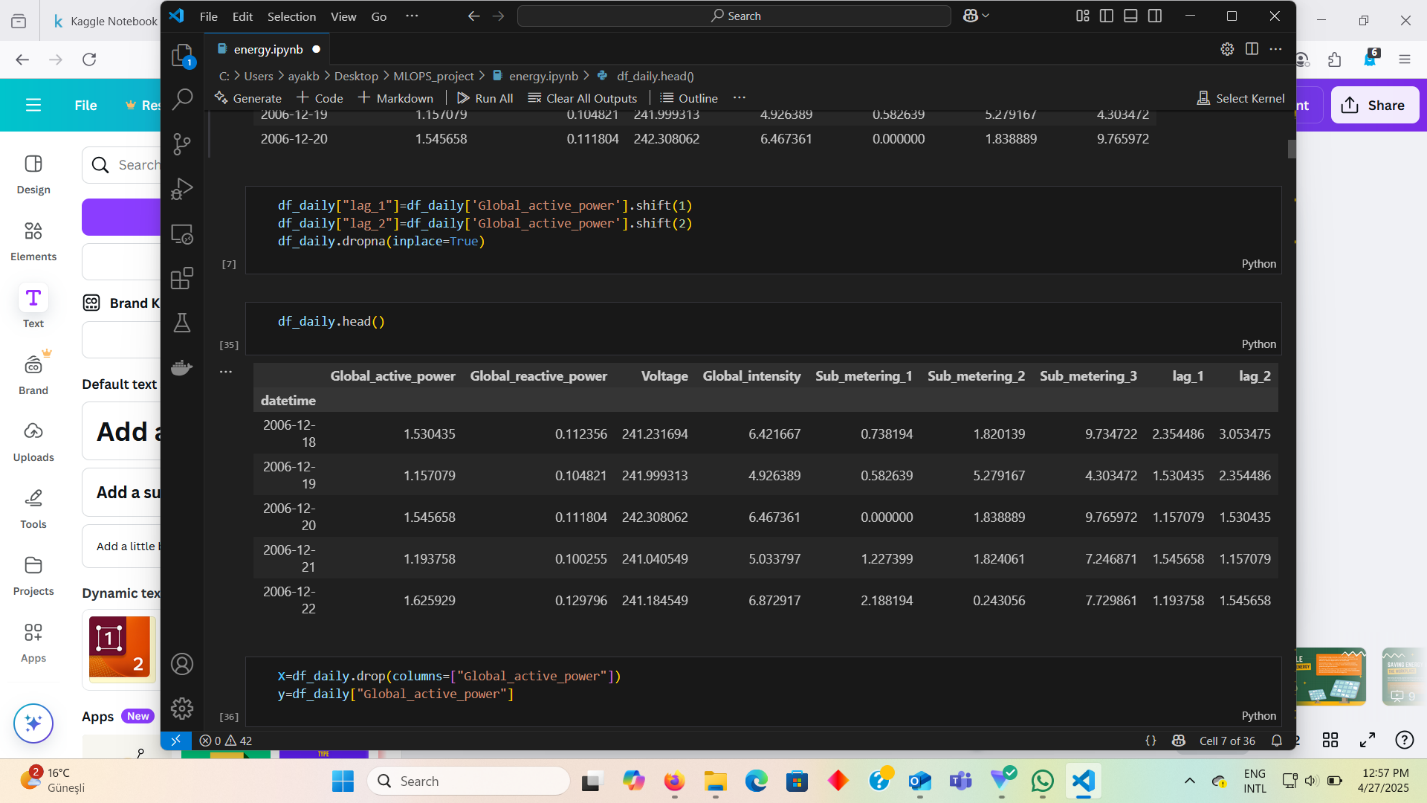
* Global\_active\_power (target)
* Global\_reactive\_power
* Voltage
* Global\_intensity
* Sub\_metering\_1,Sub\_metering\_2,Sub\_metering\_3 (energy consumption in different rooms)
* Date,Time

**Tools and Libraries used:**

* **IDE:**VS code
* **Language:** Python
* **Data preprocessing:** pandas, numpy, matplotlib, ucimlrepo
* **Modeling**: scikit-learn, xgboost
* **Hyperparameter Tunin**g: hyperopt
* **ML workflow:** mlflow , requests

**Data preprocessing:**

The First step in the Development process is data preprocessing. I combined Date and Time columns into a single datetime type column. Resampled data from minute to daily level, handled missing values, and created lag features(lag\_1, lag\_2) to predict trends. Lag features are the global active power of the previous days. Finally, I split the data 80% for training and 20% for testing.

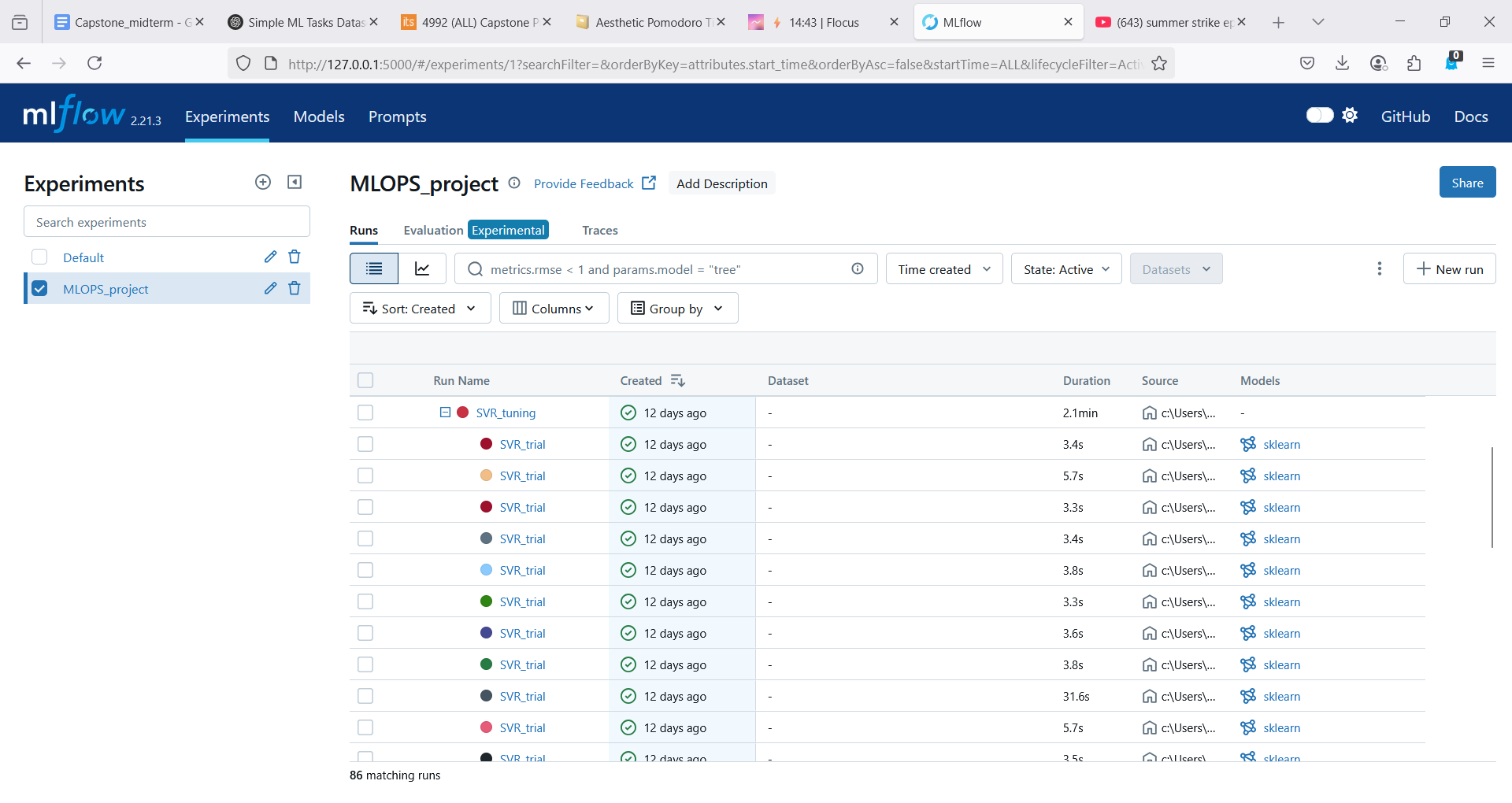


**Model training and hyperparameter tuning:**

4 models were initially tested:

* Linear Regresson
* Random Forest Regressor
* Support Vector Regressor (SVR)
* XGBoost Regressor.

MLflow was configured to log/track metrics MSE, RMSE and MSE, parameters and model artifacts. Each model was trained and evaluated in a loop. The best performing model at this stage was Linear regression.

To test if the other models performance can be improved hyperopt was to tune hyperparameters .To do that a space was designed for each model. A space is a dictionary with all the models and parameters we need for hyperparameter tuning. A loop was executed to evaluate relevant parameters. For each model a folder was created in the mlflow experiment that contains tuned models and their results after the tunning was done the best performing model’s info was saved and ready for registration.

Additionally, I created a histogram that compares the best performing result from each model. Despite improved performance Linear Regression still provides the best accuracy compared to other models.

A graph of different sizes of blue rectangular objects

AI-generated content may be incorrect.A graph of different types of data

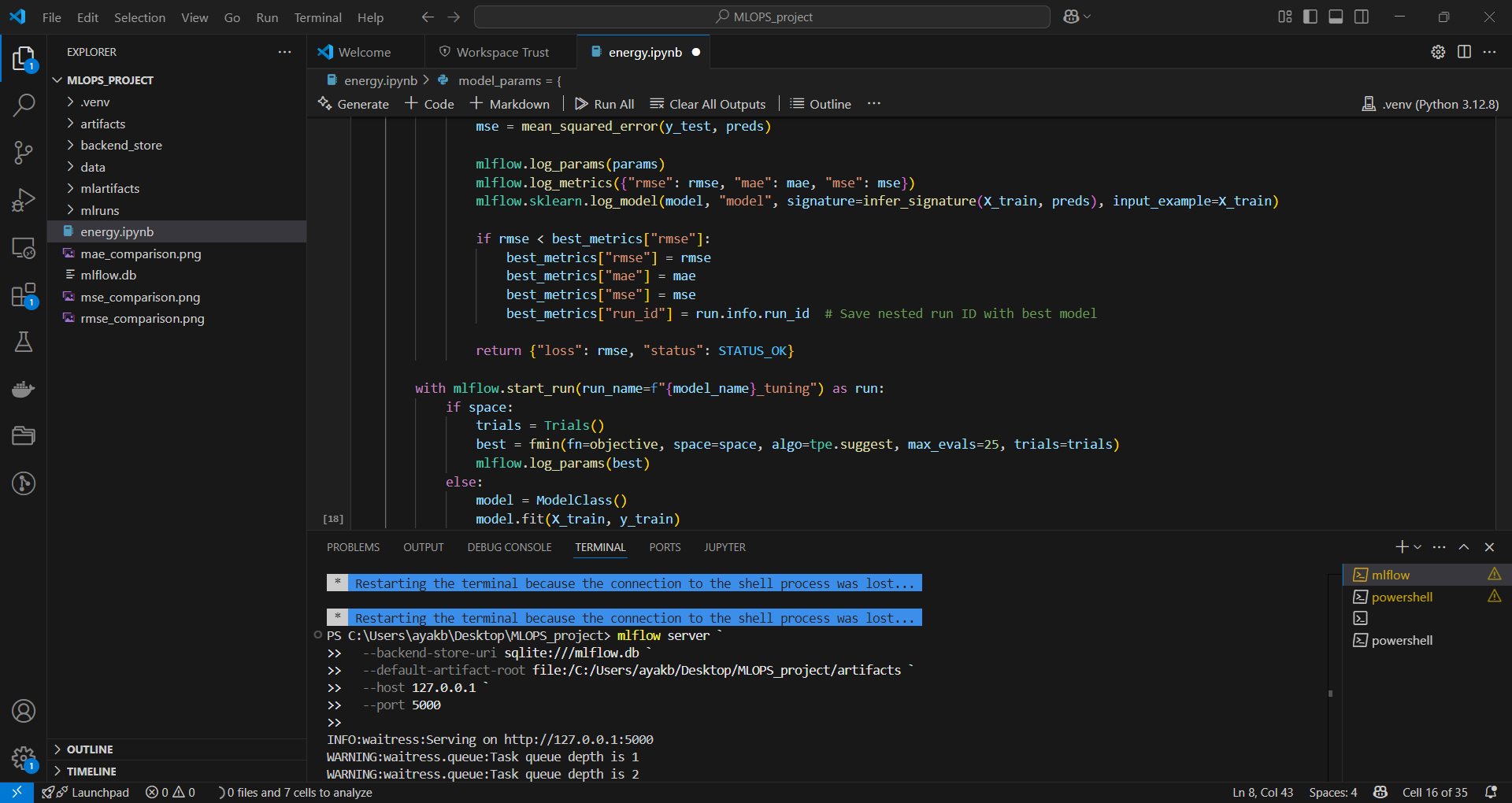
AI-generated content may be incorrect.

A graph of different sizes of blue rectangular bars

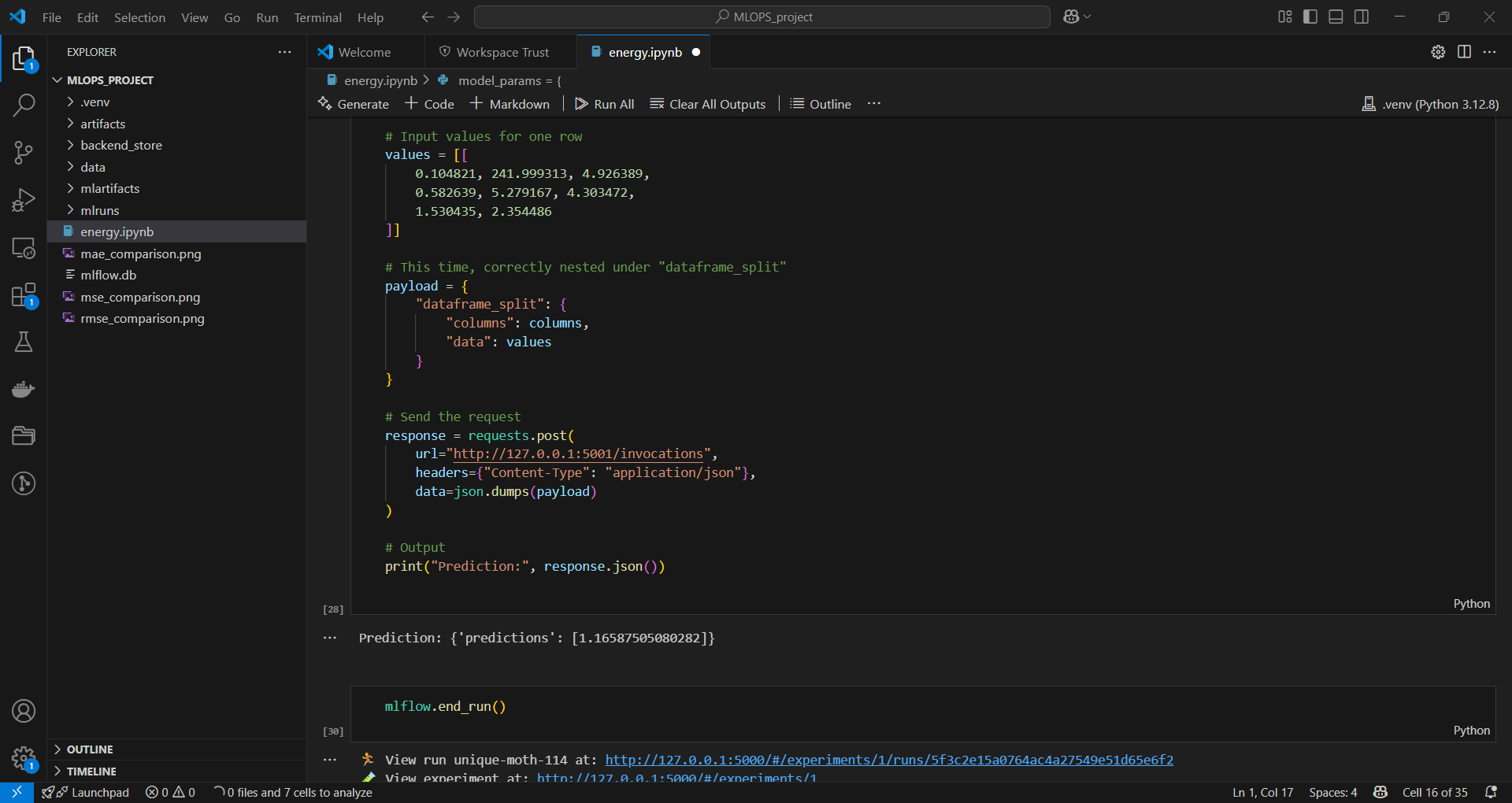
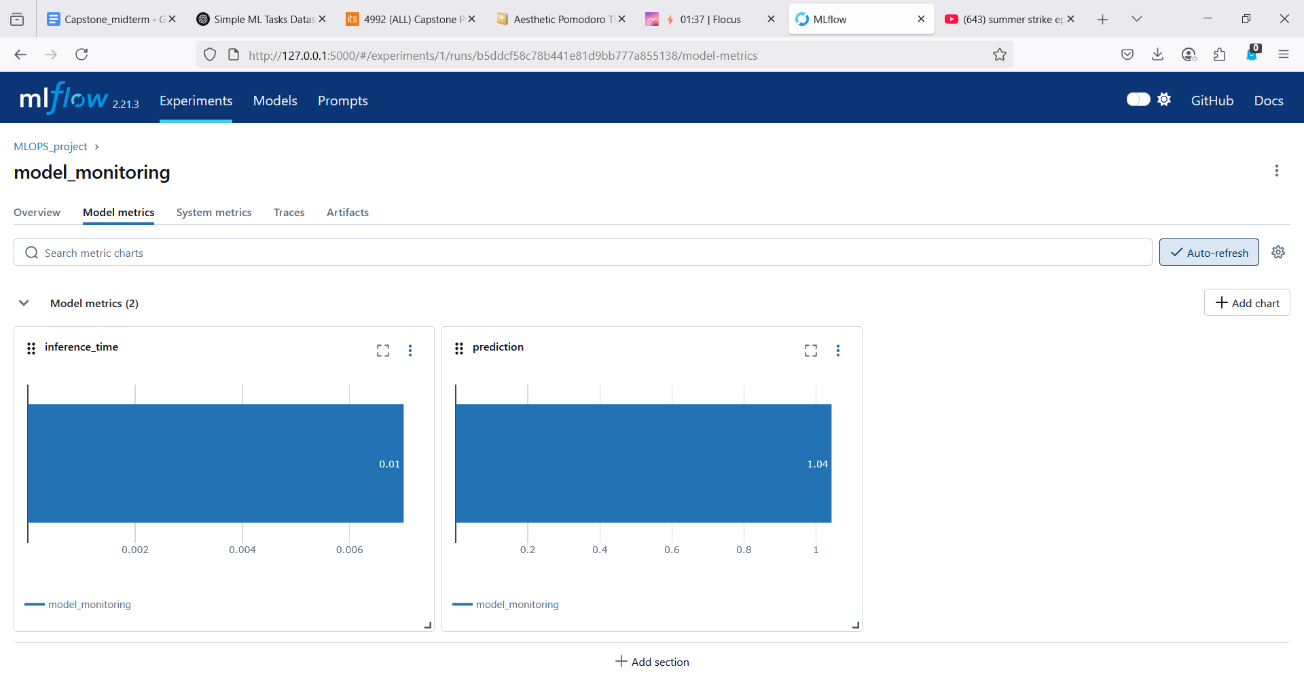
AI-generated content may be incorrect.

**Model deployment and registry:**

After hyperparameter tuning, the best model was registered and set to “production”. The model was deployed as a REST API using MLflow’s serve command. However, there were issues with the registry ,the serve command was unable to find the registered model despite everything being correct. I resolved this issue by initializing mlflow server instead of just running the mlflow UI.



**Model monitoring:**

To verify model deployment, test data was sent to the model via POST request. After successful prediction a monitoring loop was created, does predictions on a small batch of data, logs metrics and results under a monitoring run.  

**Conclusion and future steps:**

In conclusion, the Energy consumption prediction model was trained, Tuned and deployed successfully on MLflow. The project demonstrated the value of MLflow in managing a Machine Learning from training and experimentation to deployment and monitoring

Future improvements for real world use:

* Adding real time anomaly detection.
* Handling data from multiple sources and formats.
* Creating automated retraining pipelines.