MLOPS

Project Report

Name	Roll-Num	Section
Muhammad Osama Fahim	21i-0439	G

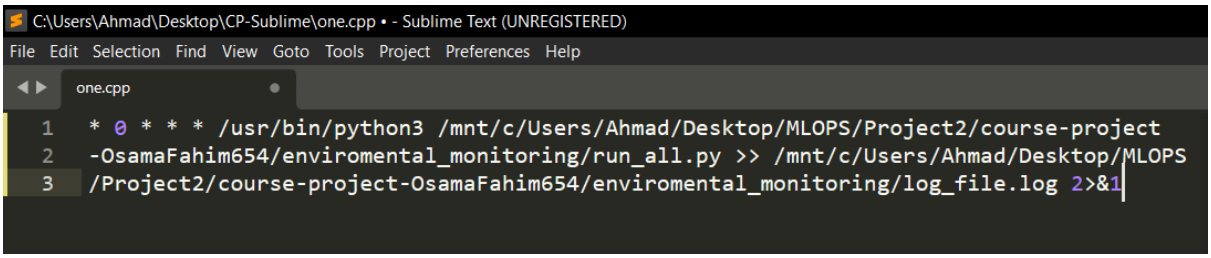
Department of Computer Science

National University of Computer and Emerging Sciences Islamabad, Pakistan

Dated: 15 December 2024

Integration Process:

- OpenWeatherMap API: First we signed in their website and registered an API for environmental change.
- Initialized DVC repository for versioning collected data and integrated it with Google Drive Folder so that new data can be versioned.
- Data collection script: Data_collection.py uses the obtained API key to fetch data of the given city (latitudes and longitudes are given).
- Version control with DVC: The Data_collection.py contains the logic for adding the newly obtained data file to DVC G drive.
- Regular Data Fetching:



The * 0 * * * indicates that the provided script will run after every hour

- Update Data with DVC: Data has been updated with DVC in the run_DVC() function which is called as soon as new file is fetched using the API which automatically add, commit and push to version and push changes to remote storage.

Model Training:

- ARIMA Model: ARIMA model has been used to forecast “components.pm2_5”.
- Hyperparameters used:
  Performed GridSearch on ARMIA (using TimeSeriesSplit for the time series cross-validation)

```
# Define the grid of hyperparameters (p, d, q)
p_values = [0, 1, 2]
d_values = [0, 1]
q_values = [0, 1, 2]
```

tscv = TimeSeriesSplit(n_splits=5), Time series cross validation which is important to keep time order.

- Logging Model in MLFlow: Model has been logged as an artifact in MLFlow.

```
# Save and log the ARIMA model as a .joblib file
model_path = "/mnt/c/Users/Ahmad/Desktop/MLOPS/Projec
joblib.dump(best_arima_model, model_path)
mlflow.log_artifact(model_path)
```

- Evaluation Metrics:
  Rmse, mae, and r2 have been tracked of the evaluation metrics.

```
#Evaluation metrics of the best model
rmse = np.sqrt(mean_squared_error(y_test, predictions))
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f"RMSE: {rmse}")
print(f"MAE: {mae}")
print(f"R2: {r2}")

mlflow.log_metric('rmse', rmse)
mlflow.log_metric('mae', mae)
mlflow.log_metric('r2', r2)
```

  Evaluation metrics have been logged in MLFlow as well

```
plt.plot(range(start_index, end_index + 1), predictions, color='red', label="Predicted Values")
plt.title("ARIMA Model Predictions vs Actual Values")
plt.legend()
plt.savefig(plot_path)
```

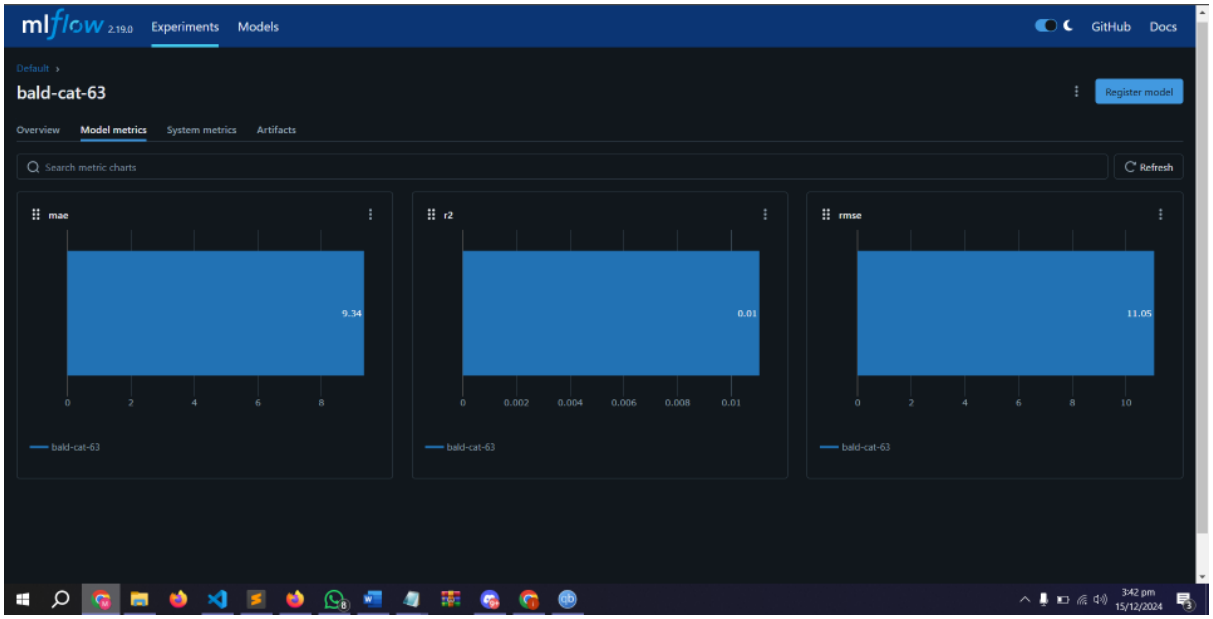
  Model Predictions vs Actual Values Graph has also been saved locally and logged in MLFlow.

Model Deployment:

- FASTAPI has been used to deploy the model
- The latest model dynamically loads from the MLFlow
- The predicted model values along with metrics are shown in the Swagger UI.
- The “unicorn app:app –reload” command launches the deployed API.

Summary report on the system's live performance:

- MLFLOW showing model metrics which was uploaded to it.



- Model's MSE and RME values

- App requests, Model’s r2 value and app requests crearted