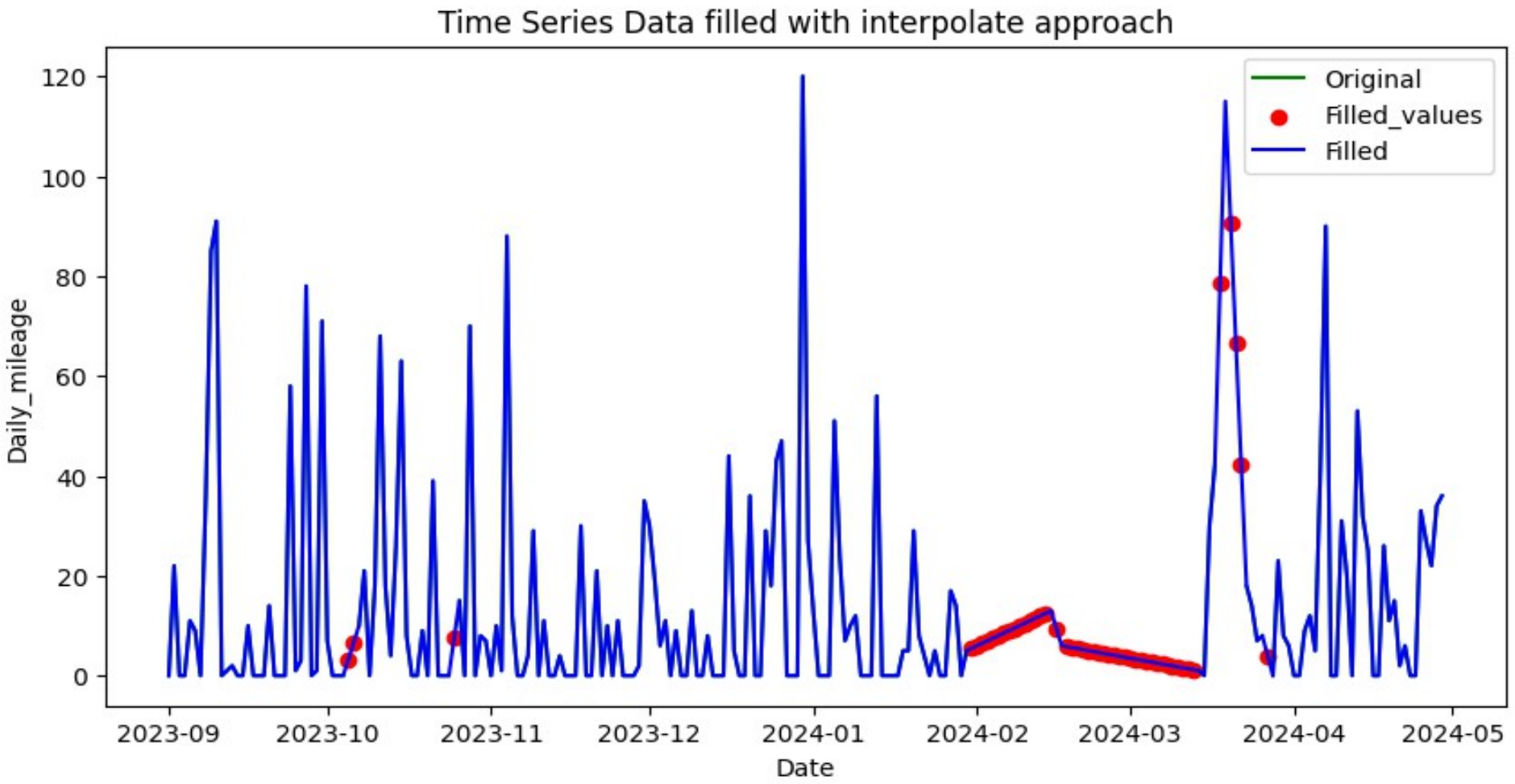
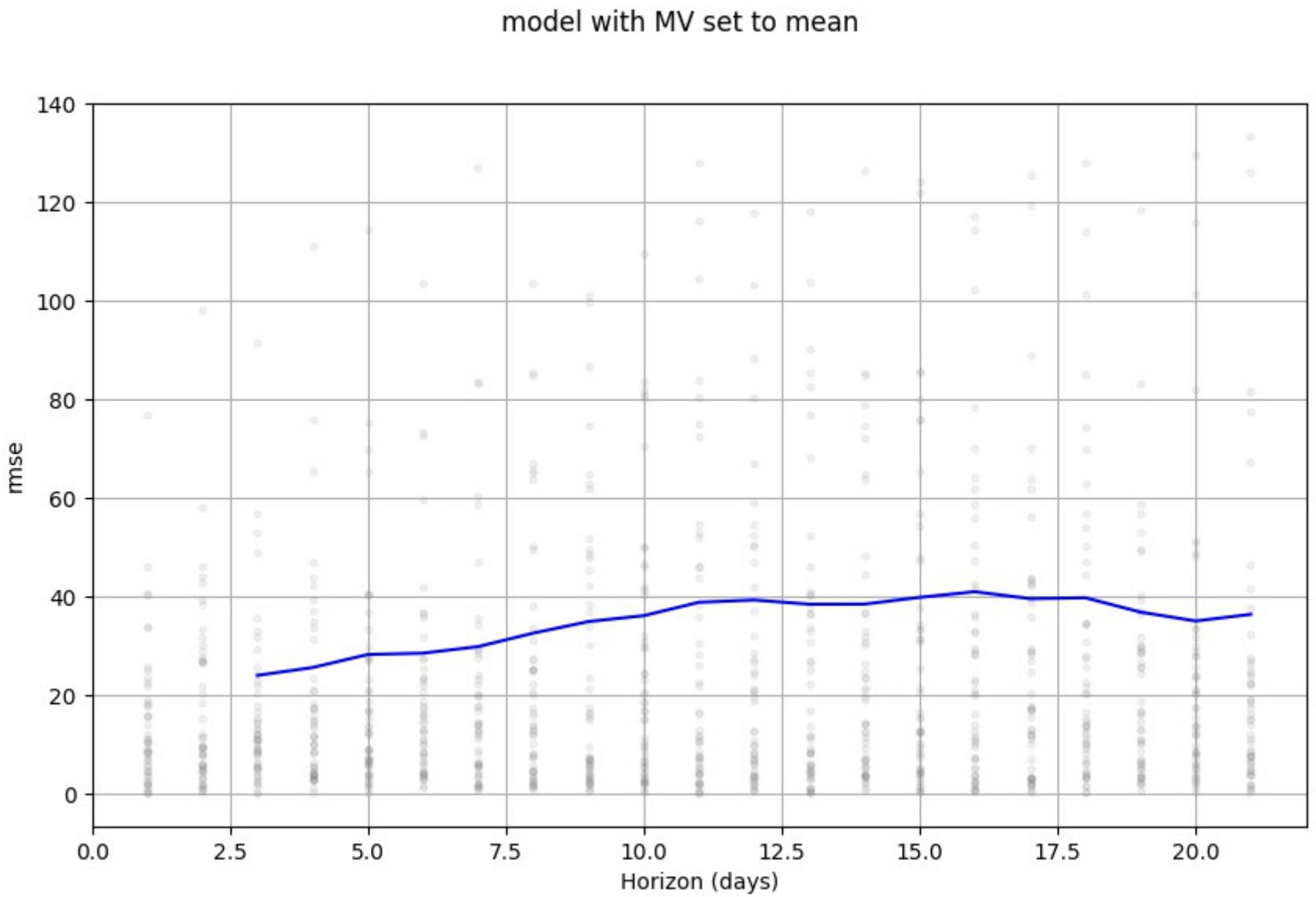
Mileage Forecasting

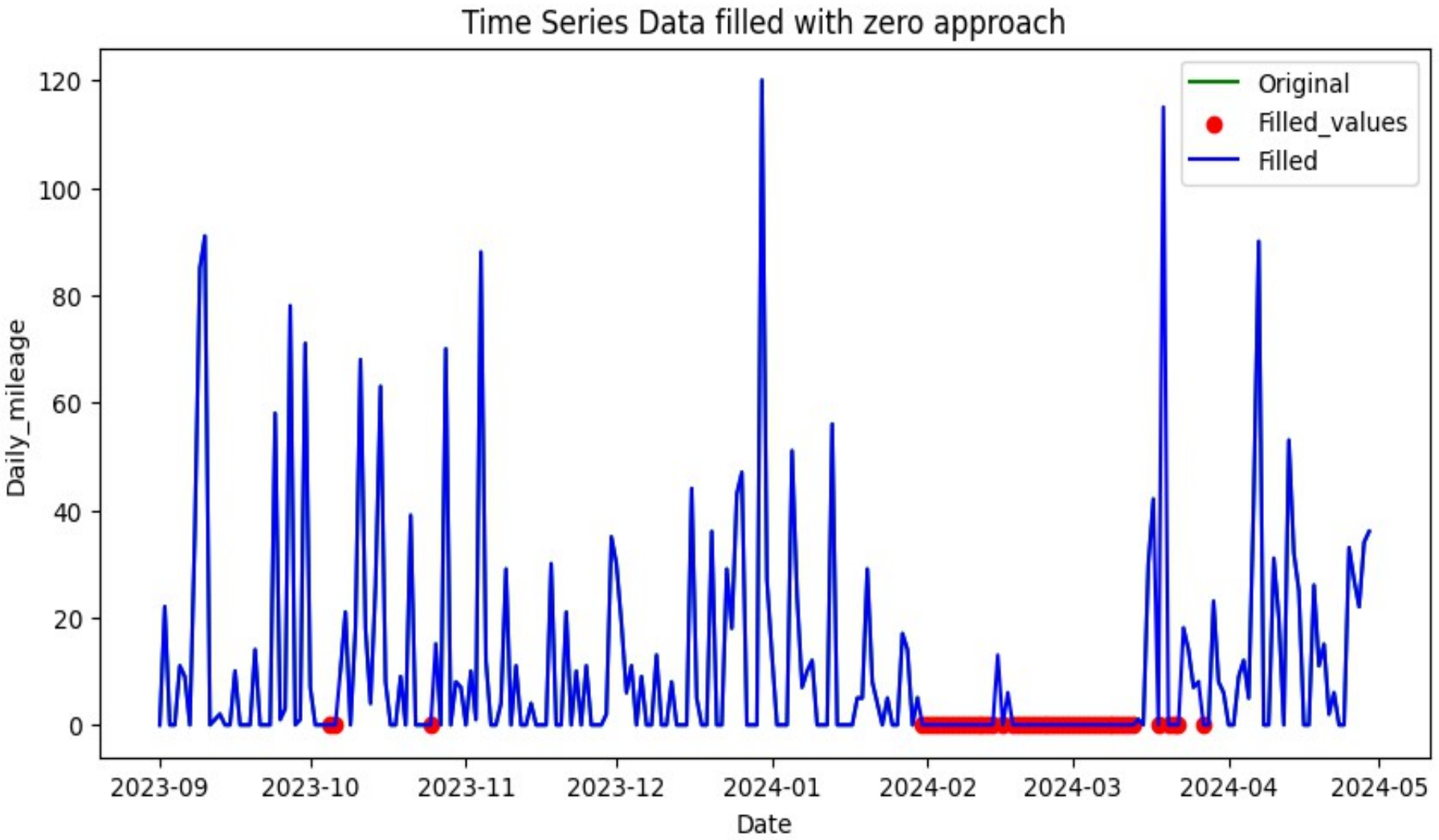
I.Preprocessing and Model Training
II.Model Deployment
III.Testing

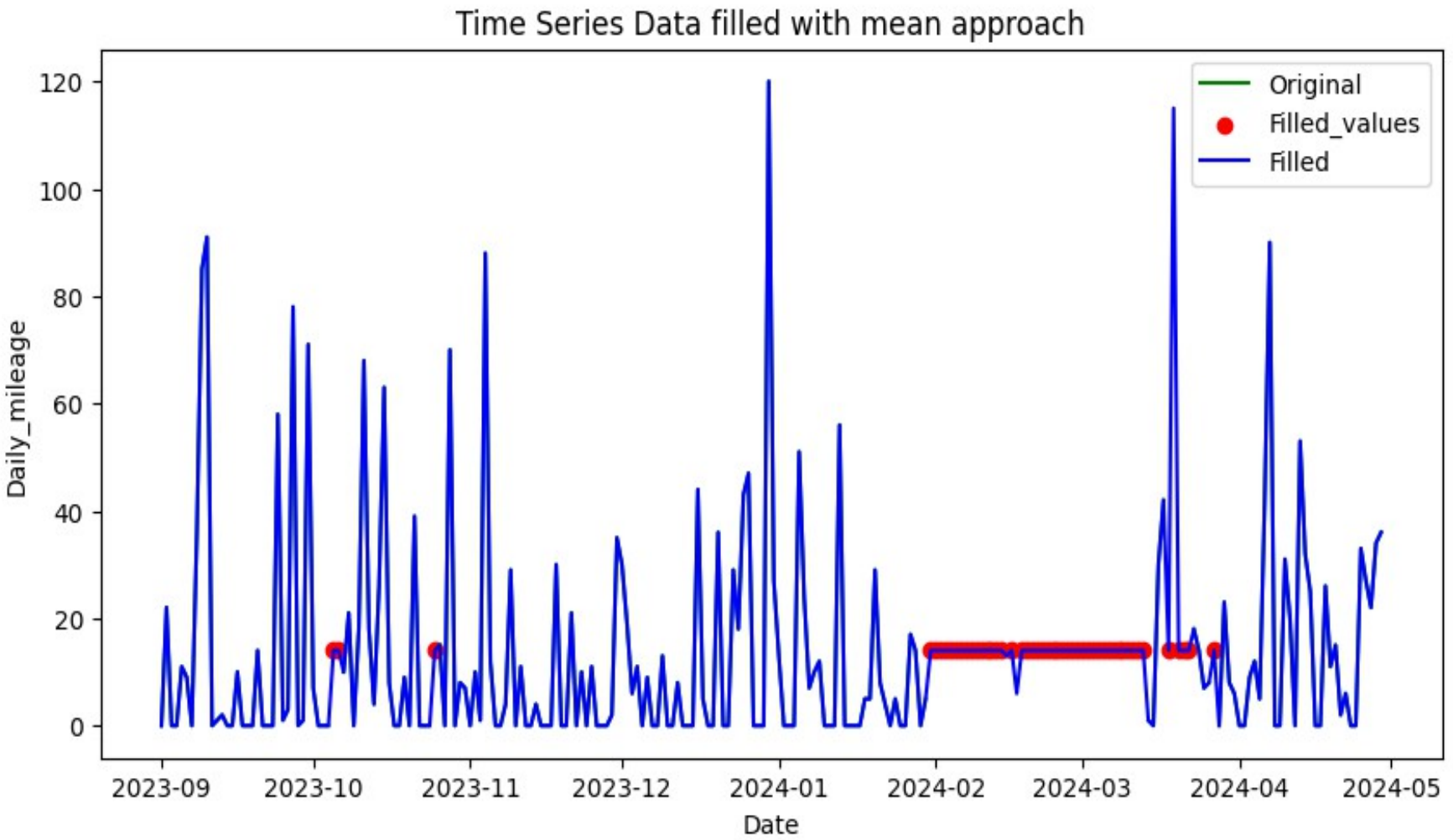
Preprocessing and Model Training

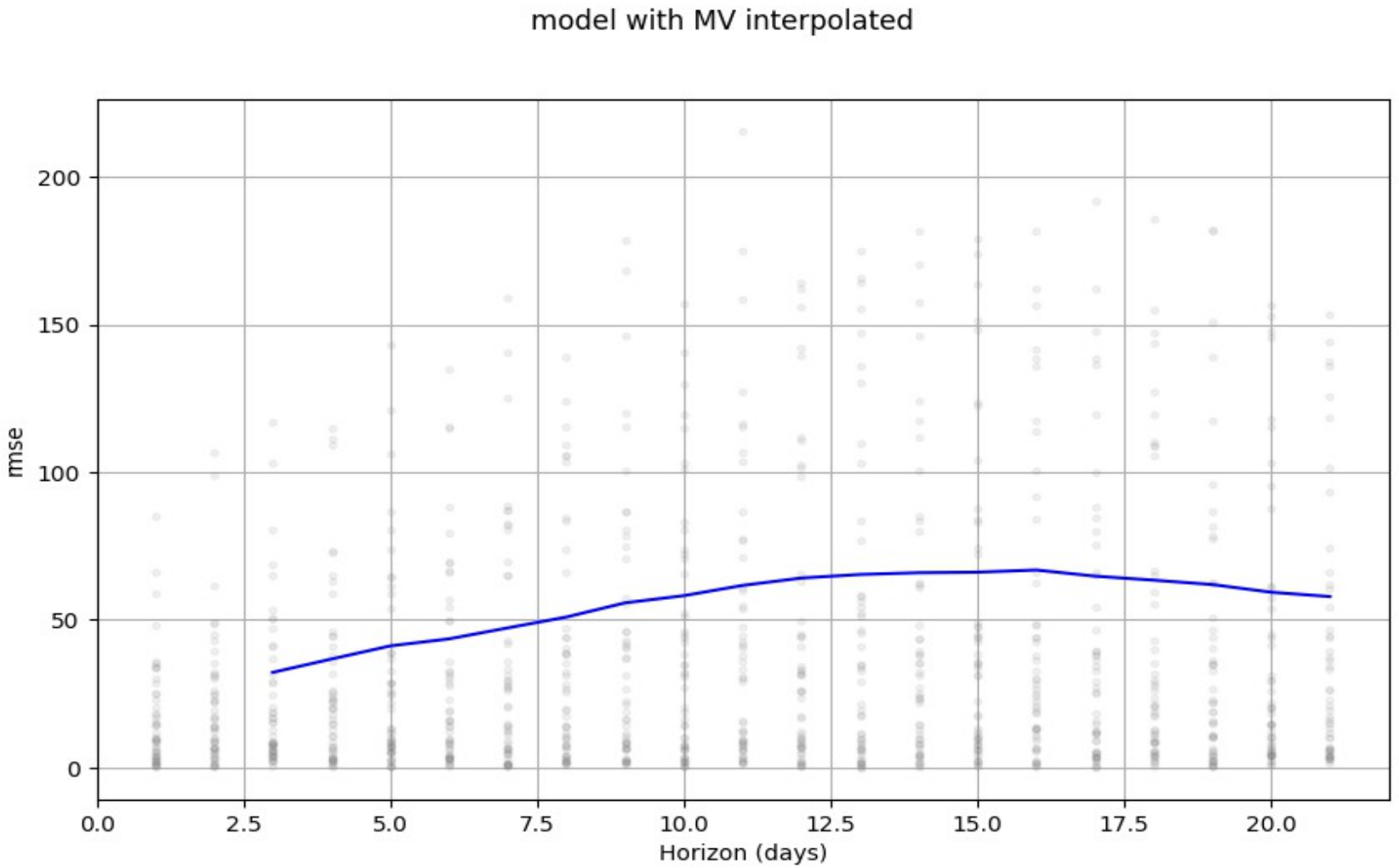
- First step: Exploratory Data Analysis
  - Exploring missing values
  - Identifying outliers
  - Exploring model (prophet) performance with different methods of handeling missing values
- Handelin of missing values
  - Best results achieved by setting missing values to constants like zero, mean or median
  - Assumption: Missing rows might indicate no vehicle activity
  - Choice of method: setting to zero

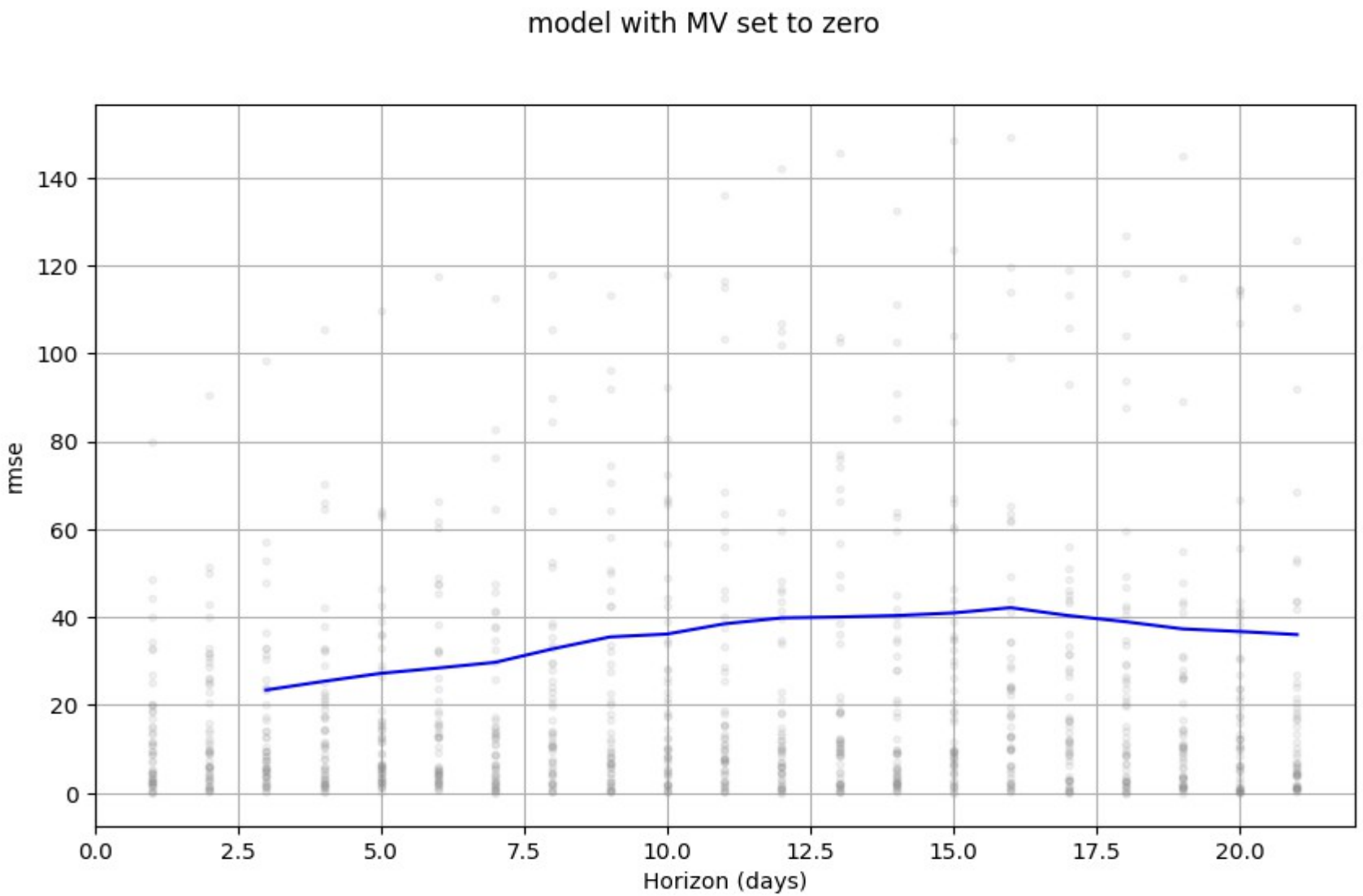

Time Series Data filled with interpolate approach
Original
Filled_values
Filled
Daily_mileage
Date
0
20
40
60
80
100
120
2023-09
2023-10
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05


model with MV set to mean
mse
Horizon (days)
0
20
40
60
80
100
120
140
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Time Series Data filled with zero approach
Original
Filled_values
Filled
Daily_mileage
Date
0
20
40
60
80
100
120
2023-09
2023-10
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05

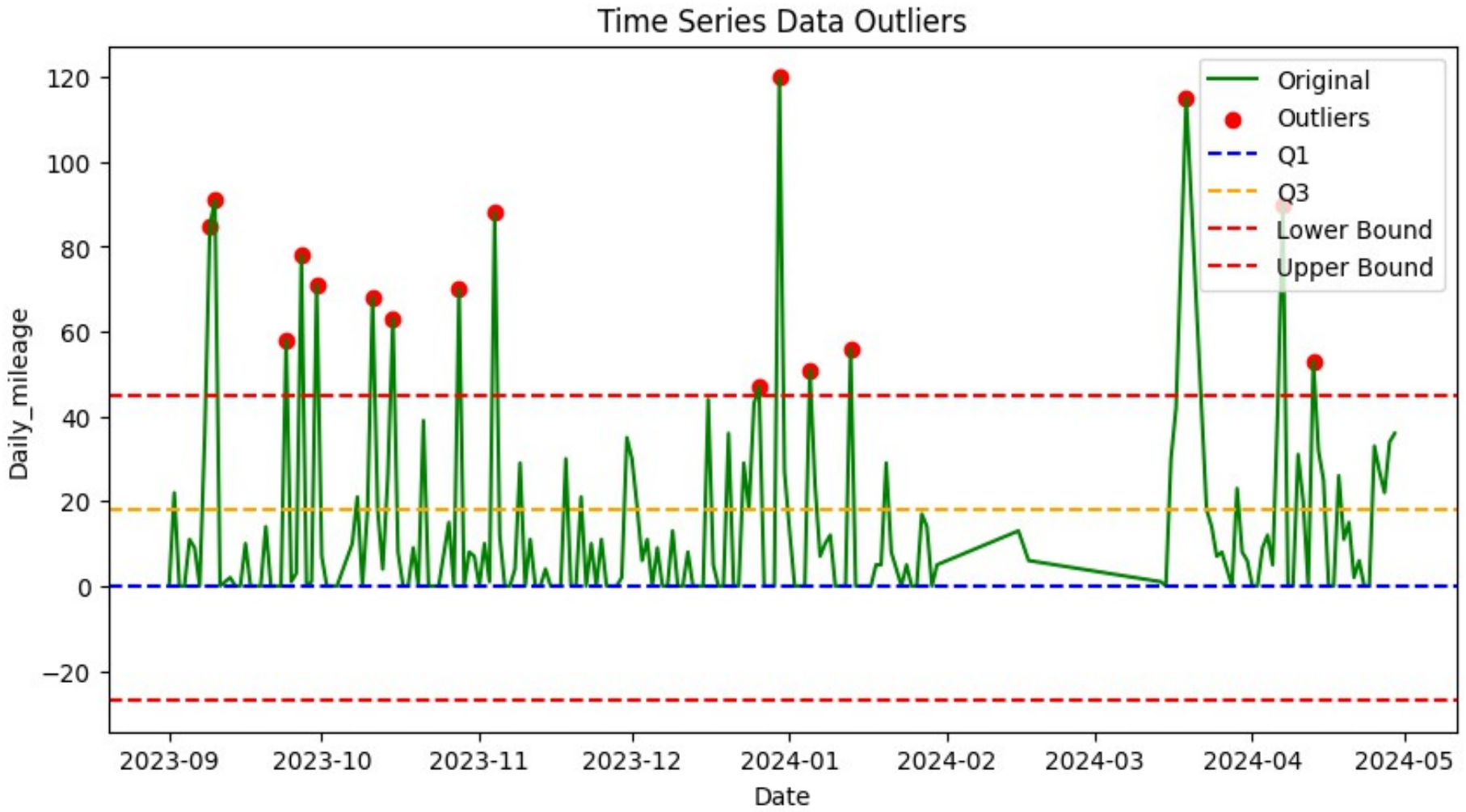

Time Series Data filled with mean approach
Original
Filled_values
Filled
Daily_mileage
Date
0
20
40
60
80
100
120
2023-09
2023-10
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05


model with MV interpolated
rmse
200
150
100
50
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Horizon (days)


model with MV set to zero
mse
0
20
40
60
80
100
120
140
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Horizon (days)

Preprocessing and Model Training

- Identify outliers by Interqualtile Range Method
  - Q1: 25th percentile: value below which 25% data falls
  - Q3: 75th percentile: value below 75% of data falls
  - Interqualtil: Range between Q1 and Q3
  - Outliers: everything that is below lower bound = 1,5*Q1 and above 1,5*Q3
- Chose method to handle outliers:
  - Set them to NaN as recommended in the Phrophet documentation


Time Series Data Outliers
Original
Outliers
Q1
Q3
Lower Bound
Upper Bound
Daily_mileage
Date
120
100
80
60
40
20
0
−20
2023-09
2023-10
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05

Model Evaluation

- Instead of splitting to train and testing sets: Cross Validation
  - Done by rolling time window for time series
  - Advantage:
    - More data for training
    - More than just a single evaluation
    - More relyablity on performance estimates
    - Better detection of overfitting
- Metric : RMSE Root mean square error
  - Intuitive interpretation as it is in the same unit as data
  - Penalize larger errors more then smaller errors

Hyperparameter Tuning

- Use grid search and cross-validation to identify best hyperparameters for
  - Changepoint:
    - Detects when trend changes and adapts predictions
    - Specify potential change points
  - Seasonality: model can try to identify seasonal patterns

•Model Deployment

- Core Idea
  - Deploy model automatically with Kubeflow Pipeline:
    - Takes care of containerization of components to have reusable components
    - Portablity : can be run locally or in cloud
    - Track and monitor models
    - Efficiency: Optimize resource usage
    - Runs on top of Kubernetes

- Ways I tried to do this:
  - Initial idea: Use Google Pipeline components to deploy model:(https://cloud.google.com/vertex-ai/docs/pipelines/build-pipeline)
    - Problem it does not support Prophet libary
  - Containerize model, load it to Vertex Ai Model registry and run Kubeflow pipeline from there:
    - Problem: Vertex Ai model registry does not support Prophet libary models
  - Load model files from Github by writing costum component and deploy it with a custom component
    - Problem: very difficult to set up Kubeflow locally on windows, lost alot of time fixing this

- Ways i tried to solve this issue with Kubeflow
  - Install kubernetes minikube locally and install Kubeflow on the minikube container:
    - 1.Problem: containersize by default too small
      - Solved by changing default container size in WSL
    - 2. Problem: kubeflow installation on minikube failes
  - Install kubeflow using kind following this tutorial :
    - https://www.kubeflow.org/docs/components/pipelines/v1/installation/localcluster-deployment/

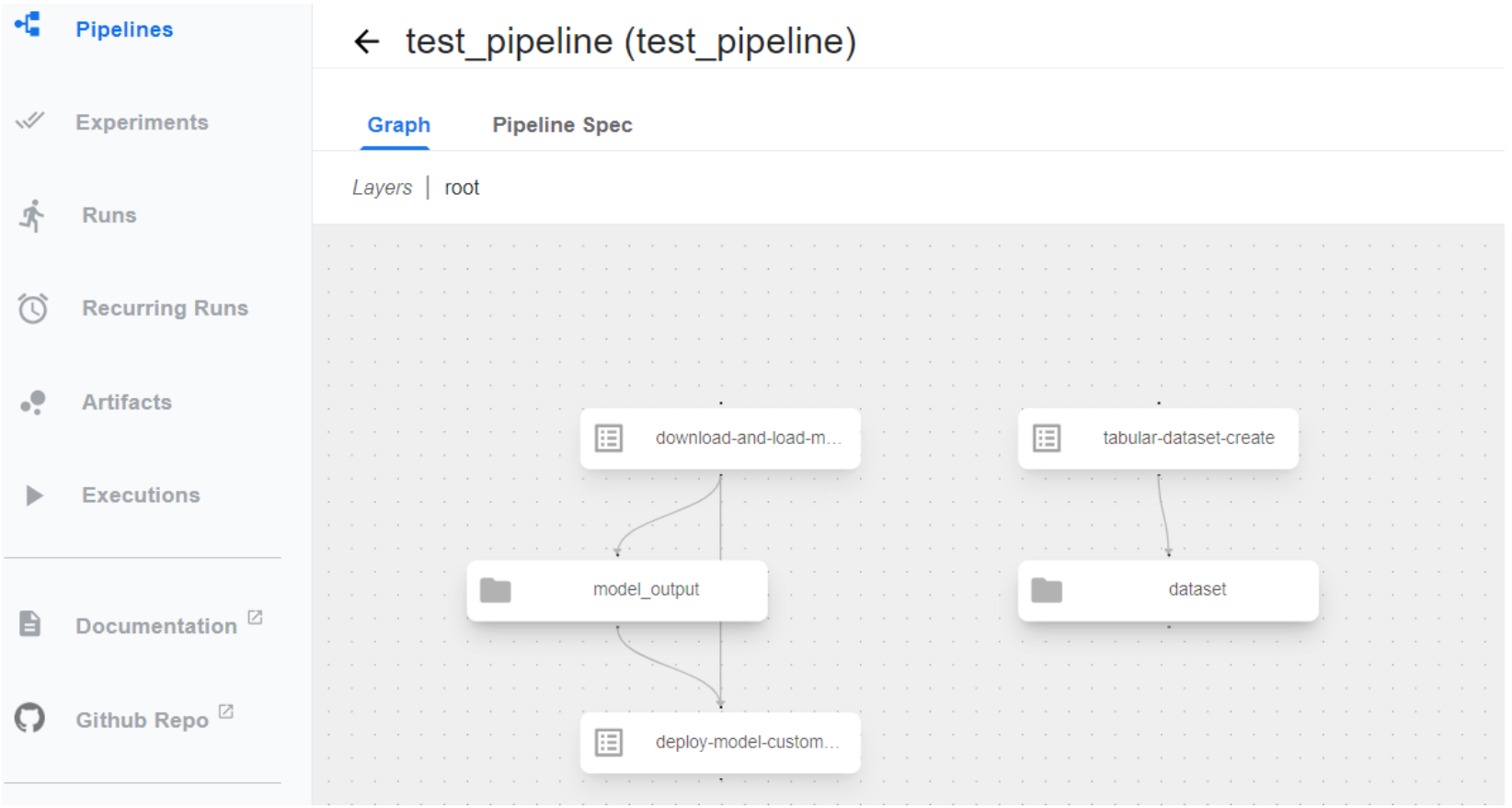
- Further problems: Compling the pipeline the Client could not find the host
  - Solved by forwarding port to 3000 :

mlpipeline)

  - Then kubeflow could be opend in the browser (localhost:3000)

```
C:\Users\aisha\MLPipeline\MLPipeline\src> kubectl
port-forward svc/ml-pipeline-ui 3000:80
--namespace kubeflow
```

- Then I ran out of time to properly debug my pipline



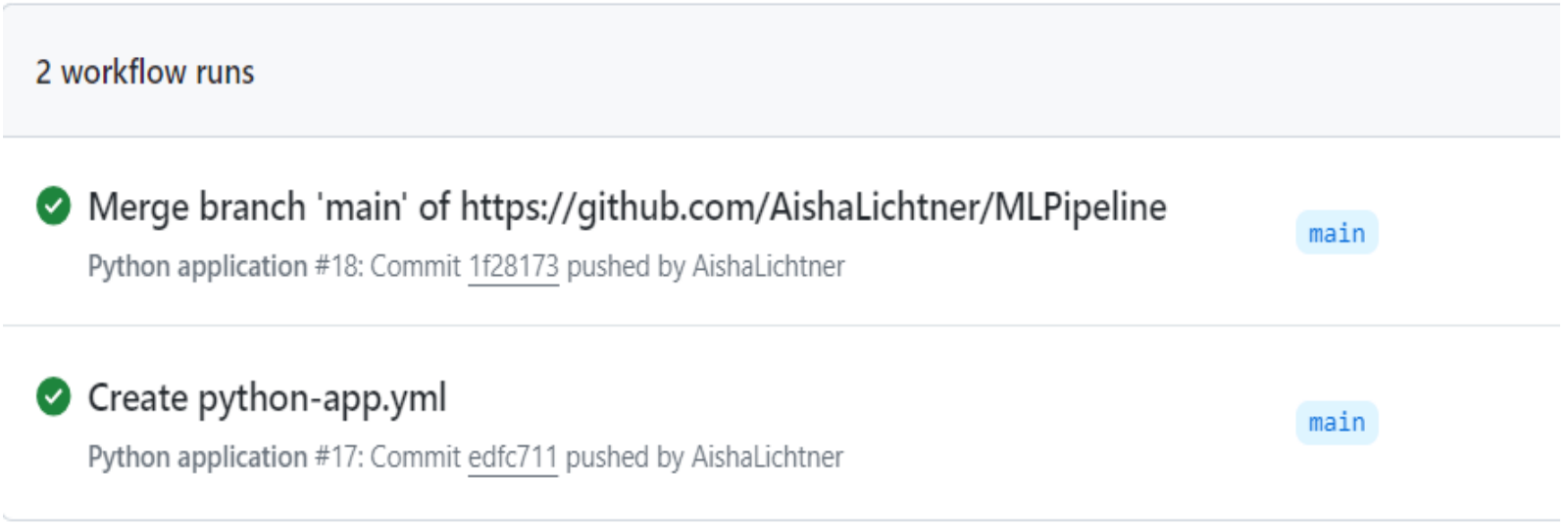
- Lesson learned:
  - When you want to use Google Best Practise better use a Tensorflow model

Testing

- First step: create a python project from the notebook
- Set up unit test for modules, preprocessing, training
- Kubeflow pipeline can also be covered with unit test

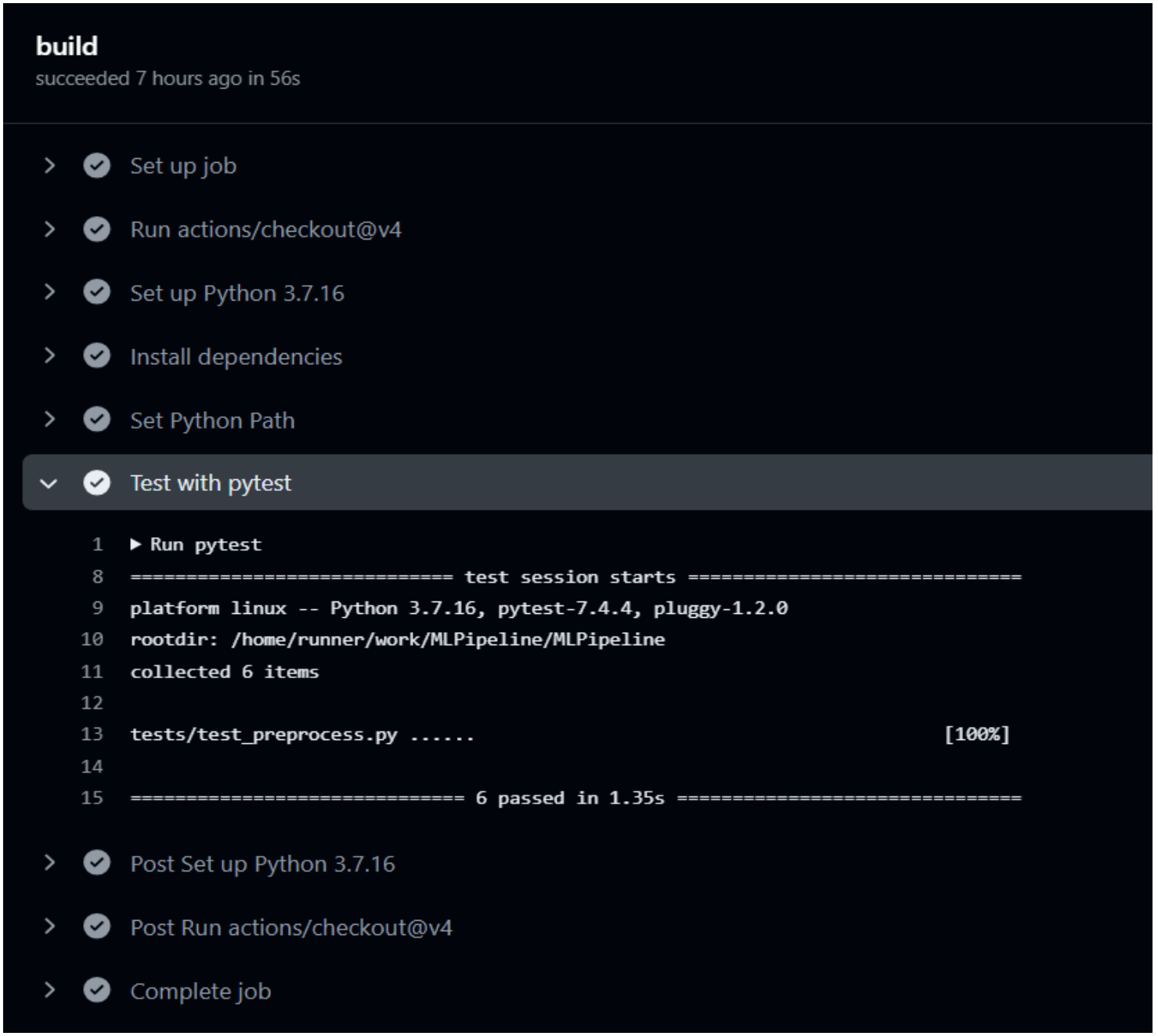
Testing

- Automated testing with github action

- 

- Problem setting up the automated testing
  - Python path wasnt set correctly, so my modules coud not be found and therefore unit tests could not run
  - Solved by adding a job to pipleline to set the python path

```
- name: Set Python Path
  run: |
    echo "PYTHONPATH=$(pwd)/src" >> $GITHUB_ENV
```


build
succeeded 7 hours ago in 56s
Set up job
Run actions/checkout@v4
Set up Python 3.7.16
Install dependencies
Set Python Path
Test with pytest
1 ▶ Run pytest
8 ============================= test session starts ==============================
9 platform linux -- Python 3.7.16, pytest-7.4.4, pluggy-1.2.0
10 rootdir: /home/runner/work/MLPipeline/MLPipeline
11 collected 6 items
12
13 tests/test_preprocess.py ...... [100%]
14
15 ============================== 6 passed in 1.35s ===============================
Post Set up Python 3.7.16
Post Run actions/checkout@v4
Complete job

Things I would have done with more time

- Modulize code better, eg.
  - Add config file to avoid hard coding
  - Make it more flexible for different structured input data
- Try github actions for model deployment
- Try combination of github action for containerization and loading to GCP and Kubeflow pipeline.