AIOps lab3

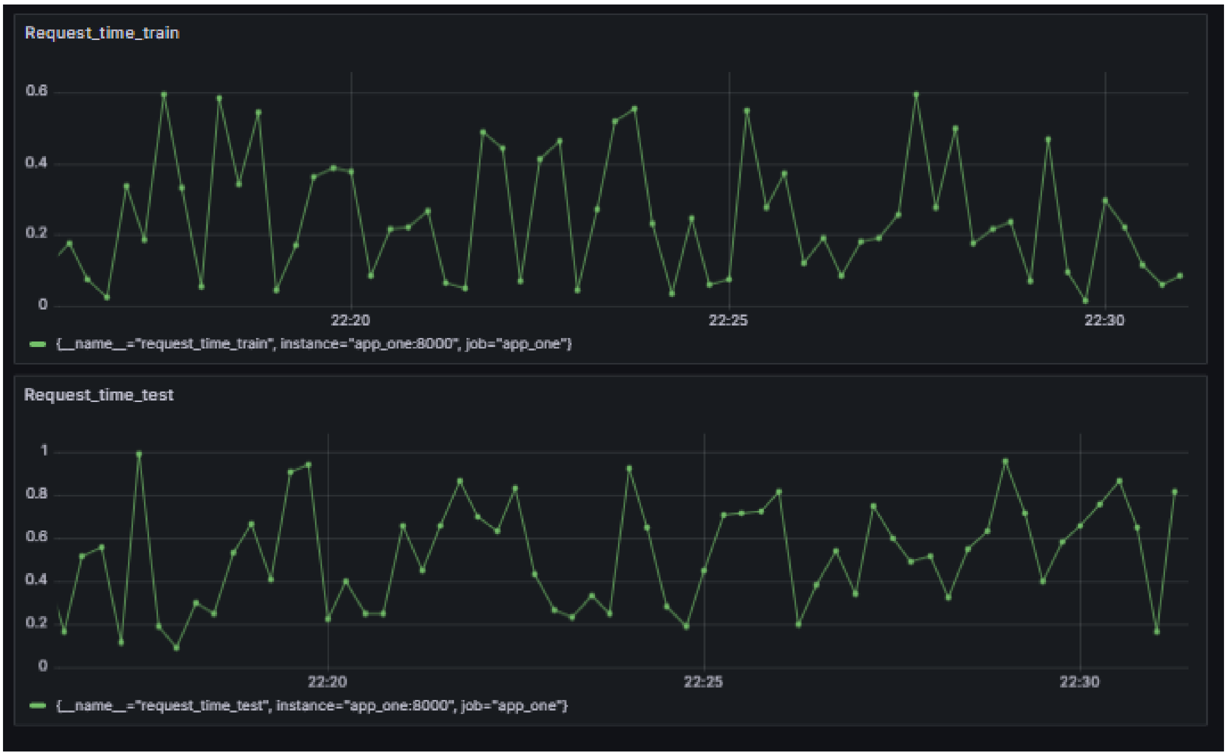
Building the Prophet model

1.Built the two new gauge metrics named request_time_train and request_time_test. The request_time_train will be used to train a prophet model and the request_time_test will be used to test for anomalies.

2.Rebuilt app_one with the two new metrics.

```
# HELP request_time_test Random gauge 0 to 1
# TYPE request_time_test gauge
request_time_test 0.7005027828699764
# HELP request_time_train Random gauge 0 to 0.6
# TYPE request_time_train gauge
request_time_train 0.5876152893158663
```

3. Add your two metrics to your Grafana dashboard as simple time series (graphs over time of a single value)



4

Prepared python application from the class notebook and this is a snippet of the anomalies detected in the logs

```
Anomalies:
                          ds         y      yhat  yhat_lower  yhat_upper  anomaly
1  2023-10-10 10:17:32.614  0.773619  0.289373   -0.129460    0.720548        1
5  2023-10-10 10:17:52.615  0.735464  0.292641   -0.224963    0.735264        1
10 2023-10-10 10:18:17.615  0.796649  0.296725   -0.140496    0.757864        1
```

Lab Task – Package model as Docker container/image

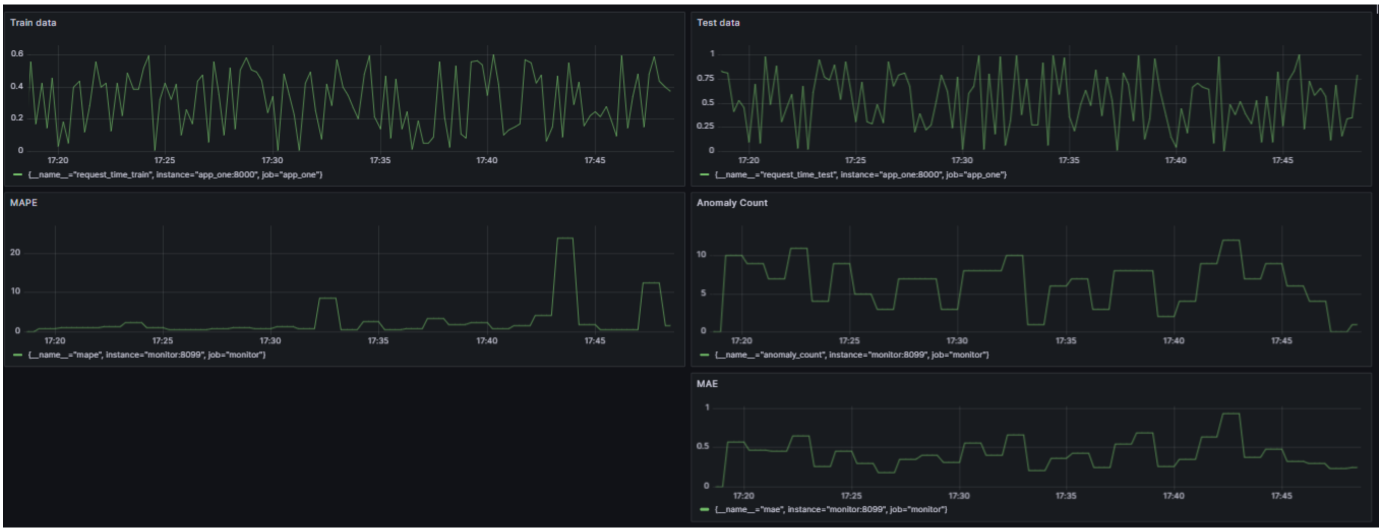
Visualisation of Grafana dashboard showing the anomaly count with the previous timeseries.

Lab Task – Explore model quality vs training time series and forecast durations

Construct a dataframe with a row for each future forecast step with columns: current time, number of anomalies detected, MAE, and MAPE). Print this dataframe to the console as your application is exiting as final output.

```
Final Output:
                   Timestamp  Anomalies       MAE      MAPE
0 2023-10-10 14:25:53.842018          3  0.258910  6.823358
1 2023-10-10 14:26:53.875449          7  0.463401  0.757637
```

Add time series visualizations to your Grafana dashboard for MAE/MAPE in addition to the original train/test and Anomaly count series – should have a total of 5 time series in your final dashboard.



Snippet of the logs after implementing the above changes

```
Fetching new training data:
http://prometheus:9090/api/v1/query?query=request_time_train[1m]
16:06:08 - cmdstanpy - INFO - Chain [1] start processing
16:06:08 - cmdstanpy - INFO - Chain [1] done processing
Sleeping for 1 minute before fetching test data (Iteration: 48)
Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Results:
The MAE for the model is 0.30737731854489
The MAPE for the model is 0.8290771802030529
Anomalies:
                       ds         y      yhat  yhat_lower  yhat_upper  anomaly
0 2023-10-10 16:06:10.256  0.838487  0.385096    0.009908    0.734033        1
1 2023-10-10 16:06:15.255  0.987362  0.394679    0.006123    0.755653        1
4 2023-10-10 16:06:30.255  0.932728  0.423435    0.061460    0.797093        1
7 2023-10-10 16:06:45.255  0.836313  0.452190    0.071006    0.802118        1
Sleeping for 1 minute before fetching test data (Iteration: 49)
Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Results:
The MAE for the model is 0.24850145516546515
The MAPE for the model is 6.7666410728861655
Anomalies:
                        ds         y      yhat  yhat_lower  yhat_upper  anomaly
2  2023-10-10 16:07:20.255  0.007631  0.519287    0.139349    0.892616        1
8  2023-10-10 16:07:50.255  0.060050  0.576798    0.212282    0.971484        1
11 2023-10-10 16:08:05.255  0.159750  0.605554    0.229810    0.957557        1
Fetching new training data:
http://prometheus:9090/api/v1/query?query=request_time_train[1m]
16:08:08 - cmdstanpy - INFO - Chain [1] start processing
16:08:08 - cmdstanpy - INFO - Chain [1] done processing
Sleeping for 1 minute before fetching test data (Iteration: 50)
Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Results:
The MAE for the model is 0.2826209751113355
The MAPE for the model is 1.2537115037425277
Anomalies:
                       ds         y      yhat  yhat_lower  yhat_upper  anomaly
2 2023-10-10 16:08:20.256  0.926756  0.406068    0.032373    0.764680        1
3 2023-10-10 16:08:25.255  0.909406  0.411602    0.022508    0.795427        1
Sleeping for 1 minute before fetching test data (Iteration: 51)
```

Lab Task – Answer the following questions

I experimented with the following variations

Test time	Train	Forecast iterations
1 minute	1 minute	2
10 minutes	1 minute	2
12 hours	1 minute	2
5 minutes	1 minute	5
5 minutes	1 minute	10

Did you find any type of seasonality to be helpful in assuring the best forecast from your model? Why or why not?

No.

Because these a randomly generated values.

2. How far into the future did you observe your forecast to be working? What is the effect

of adding a longer baseline of training data?

My forecast worked well up to 3 minutes because when I tested up to 10 minutes at minute 4 the MAPE was higher than the previous minutes.

```
Final Output:
                    Timestamp  Anomalies       MAE      MAPE
0 2023-10-10 21:13:22.496057          8  0.427862  0.770145
1 2023-10-10 21:14:22.532454          3  0.212338  0.928368
2 2023-10-10 21:15:22.562693          5  0.296385  0.429068
3 2023-10-10 21:16:22.598090          3  0.256895  4.718836
4 2023-10-10 21:17:22.637254          3  0.262748  1.561683
5 2023-10-10 21:18:22.672638          6  0.341220  1.028274
6 2023-10-10 21:19:22.707414          5  0.323010  0.543941
7 2023-10-10 21:20:22.749694          4  0.288352  0.398754
8 2023-10-10 21:21:22.782414          2  0.238049  2.184600
9 2023-10-10 21:22:22.812100          1  0.205596  1.346977
```

A longer baseline of training data, such as 12 hours, notably improved the model's performance by allowing it to learn intricate patterns and fluctuations. This is because the longer training time enables the model to adapt to changes more effectively. However, the choice of training duration should strike a balance between model performance and operational challenges, as longer training times lead to increased storage and computational requirements.

3. What do you estimate to be a “reasonable” baseline of data to use this type of Prophet

model in an actual running production system? Would that length of training time pose

any operational challenges?

I would say above 12 hours but in a production system, a "reasonable" baseline for the Prophet model can depend on the specific use case but often encompasses several days or weeks of training data. Such a length allows the model to capture seasonal and daily patterns efficiently. However, the exact training duration needs to be carefully determined to avoid overfitting to historical data and ensure adaptability to evolving trends. These longer training times may pose operational challenges in terms of resource consumption and responsiveness to real-time changes, which must be considered.

4. Do you think such a Prophet model should be allowed to retrain continuously in a pro-

duction setting or require some manual review/approval? What could be some pitfalls of

allowing a fully automatic operation?

Whether a Prophet model should retrain continuously in a production setting or require manual review/approval depends on the specific application and the consequences of incorrect forecasts. Continuous retraining can be useful for systems with rapidly changing data, but it may pose risks, especially if the model's behavior becomes unpredictable. Pitfalls of fully automatic operation include the model becoming overly sensitive to short-term fluctuations, leading to a high number of false positives. Manual review/approval can provide more control but may not be feasible in real-time applications. The best approach may involve a hybrid system with automated retraining and human oversight to balance accuracy and stability.

Appendix

```
Forecast_count : 2
Train minutes : 12
Test minutes : 1

Fetching new training data:
http://prometheus:9090/api/v1/query?query=request_time_train[12h]
20:45:31 - cmdstanpy - INFO - Chain [1] start processing
20:45:32 - cmdstanpy - INFO - Chain [1] done processing
Sleeping for 1 minute before fetching test data (Iteration: 0)

Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Anomalies:
                        ds         y      yhat  yhat_lower  yhat_upper  anomaly
0  2023-10-10 20:45:35.249  0.751987  0.296541   -0.113240    0.749591        1
1  2023-10-10 20:45:40.249  0.935659  0.296541   -0.161373    0.722911        1
5  2023-10-10 20:46:00.249  0.951944  0.296537   -0.156129    0.709019        1
7  2023-10-10 20:46:10.250  0.953396  0.296536   -0.143363    0.682466        1
10 2023-10-10 20:46:25.250  0.726195  0.296533   -0.136988    0.705471        1
11 2023-10-10 20:46:30.249  0.788453  0.296532   -0.177203    0.717618        1

The MAE for the model is 0.39650356759587085
The MAPE for the model is 0.5312874533496025
Sleeping for 1 minute before fetching test data (Iteration: 1)

Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Anomalies:
                       ds         y      yhat  yhat_lower  yhat_upper  anomaly
0 2023-10-10 20:46:35.249  0.746037  0.296532   -0.115981    0.724455        1
3 2023-10-10 20:46:50.250  0.876177  0.296529   -0.114379    0.768947        1
7 2023-10-10 20:47:10.249  0.901287  0.296526   -0.145689    0.753583        1
8 2023-10-10 20:47:15.249  0.715638  0.296525   -0.120339    0.709906        1

The MAE for the model is 0.27214002142006527
The MAPE for the model is 1.287290703847485

Final Output:
                   Timestamp  Anomalies       MAE      MAPE
0 2023-10-10 20:46:32.213898          6  0.396504  0.531287
1 2023-10-10 20:47:32.260659          4  0.272140  1.287291
```

```
Forecast_count : 2
Train minutes : 10
Test minutes : 1

Fetching new training data:
http://prometheus:9090/api/v1/query?query=request_time_train[10m]
20:51:45 - cmdstanpy - INFO - Chain [1] start processing
20:51:45 - cmdstanpy - INFO - Chain [1] done processing
Sleeping for 1 minute before fetching test data (Iteration: 0)

Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Anomalies:
                        ds         y      yhat  yhat_lower  yhat_upper  anomaly
5  2023-10-10 20:52:15.249  0.772526  0.267367   -0.168805    0.731677        1
7  2023-10-10 20:52:25.249  0.958727  0.266501   -0.173648    0.723772        1
11 2023-10-10 20:52:45.249  0.753103  0.264768   -0.196297    0.744599        1

The MAE for the model is 0.24964015429924372
The MAPE for the model is 0.6663466244447416
Sleeping for 1 minute before fetching test data (Iteration: 1)

Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Anomalies:
Empty DataFrame
Columns: [ds, y, yhat, yhat_lower, yhat_upper, anomaly]
Index: []

The MAE for the model is 0.19433063667373907
The MAPE for the model is 6.703189921023308

Final Output:
                   Timestamp  Anomalies       MAE      MAPE
0 2023-10-10 20:52:45.382625          3  0.249640  0.666347
1 2023-10-10 20:53:45.422611          0  0.194331  6.703190
```

```
Forecast_count : 2
Train minutes : 1
Test minutes : 1

Fetching new training data:
http://prometheus:9090/api/v1/query?query=request_time_train[1m]
20:26:48 - cmdstanpy - INFO - Chain [1] start processing
20:26:48 - cmdstanpy - INFO - Chain [1] done processing
Sleeping for 1 minute before fetching test data (Iteration: 0)

Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Anomalies:
                        ds         y      yhat  yhat_lower  yhat_upper  anomaly
3  2023-10-10 20:27:05.250  0.132599  0.595188    0.249825    0.922918        1
4  2023-10-10 20:27:10.251  0.160511  0.615403    0.233688    0.996498        1
5  2023-10-10 20:27:15.250  0.198782  0.635610    0.274338    0.971861        1
8  2023-10-10 20:27:30.249  0.246936  0.696239    0.340126    1.096195        1
11 2023-10-10 20:27:45.250  0.084100  0.756877    0.397321    1.159798        1

The MAE for the model is 0.2720145628829247
The MAPE for the model is 1.65529464193033
Sleeping for 1 minute before fetching test data (Iteration: 1)

Fetching test data:
http://prometheus:9090/api/v1/query?query=request_time_test[1m]
Anomalies:
                        ds         y      yhat  yhat_lower  yhat_upper  anomaly
2  2023-10-10 20:28:00.250  0.332613  0.817510    0.460509    1.183012        1
5  2023-10-10 20:28:15.250  0.286513  0.878143    0.492294    1.254031        1
6  2023-10-10 20:28:20.250  0.166385  0.898354    0.546564    1.223955        1
8  2023-10-10 20:28:30.250  0.529706  0.938776    0.596682    1.303626        1
11 2023-10-10 20:28:45.251  0.306251  0.999413    0.598779    1.314478        1

The MAE for the model is 0.33773497867115476
The MAPE for the model is 1.0519692849394404

Final Output:
                   Timestamp  Anomalies       MAE      MAPE
0 2023-10-10 20:27:49.017358          5  0.272015  1.655295
1 2023-10-10 20:28:49.058113          5  0.337735  1.051969
```

```
Final Output:
                   Timestamp  Anomalies       MAE      MAPE
0 2023-10-10 21:01:55.845443          2  0.178532  0.453395
1 2023-10-10 21:02:55.888518          1  0.249616  1.150684
2 2023-10-10 21:03:55.926098          3  0.289059  1.101072
3 2023-10-10 21:04:55.963771          2  0.211002  0.370083
4 2023-10-10 21:05:55.998574          2  0.281251  1.085215
```