In [829…
```
#All dependencies in the dependencies.txt file. This should work too
#!pip install numpy pandas IPython seaborn matplotlib folium statsmodels tensorflow scik
```

In [31]:
```
# DATA
import numpy as np
import pandas as pd

# DASHBOARDS
from IPython.display import HTML

# PLOTTERS
import seaborn as sns
import matplotlib.pyplot as plt

# MAP VISUALIZER
import folium
from folium.plugins import MarkerCluster


# STATISTICS
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# LSTM
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.layers import Input, Bidirectional, LSTM, Dense
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

BASIC Exploratory Data Analysis (EDA)

First Look at The Data

In [32]:
```
data = pd.read_csv("orders_autumn_2020.csv")
print(data.dtypes)
data.describe()
```

```
TIMESTAMP                                                 object
ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES       int64
ITEM_COUNT                                                 int64
USER_LAT                                                 float64
USER_LONG                                                float64
VENUE_LAT                                                float64
VENUE_LONG                                               float64
ESTIMATED_DELIVERY_MINUTES                                 int64
ACTUAL_DELIVERY_MINUTES                                    int64
CLOUD_COVERAGE                                           float64
TEMPERATURE                                              float64
WIND_SPEED                                               float64
PRECIPITATION                                            float64
dtype: object
```

Out[32]:

	ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES	ITEM_COUNT	USER_LAT	USER_LONG	VENUE_LAT	VENUE_LONG	ES
count	18706.000000	18706.000000	18706.000000	18706.000000	18706.000000	18706.000000	
mean	-1.201058	2.688228	60.175234	24.941244	60.175643	24.941214	

std	8.979834	1.886455	0.012674	0.016540	0.011509	0.014482
min	-41.000000	1.000000	60.153000	24.909000	60.149000	24.878000
25%	-7.000000	1.000000	60.163000	24.926000	60.167000	24.930000
50%	-2.000000	2.000000	60.175000	24.943000	60.170000	24.941000
75%	5.000000	4.000000	60.186000	24.954000	60.186000	24.950000
max	34.000000	11.000000	60.201000	24.980000	60.219000	25.042000

Spatial View

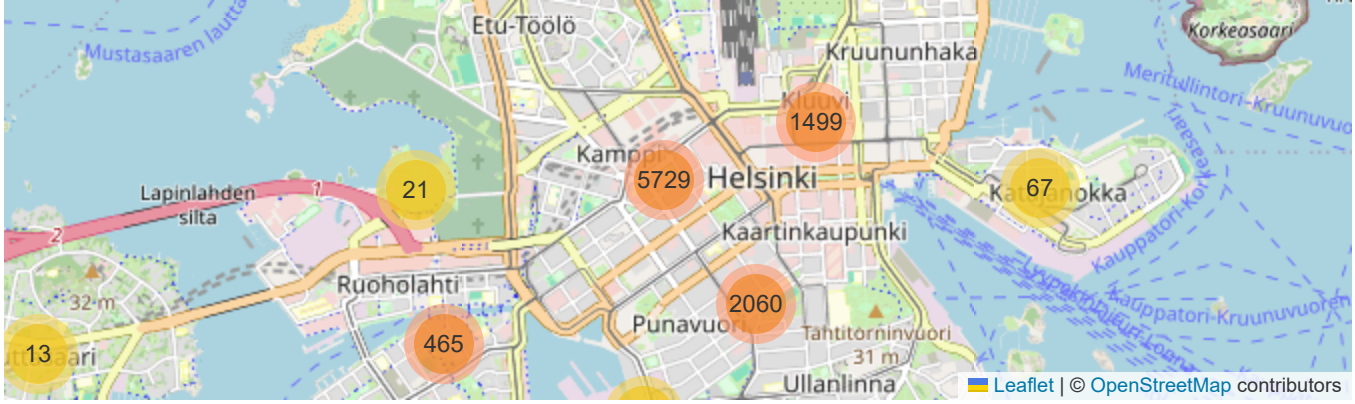
1. We have **coordinates**. We can **visualize the area** where the users and venues are located
2. Plotting users and venues locations we find the dataset concerns the city of **Helsinki**
3. It's hard to tell whether some areas are more populated than others in terms of users and venues. A heatmap could clarify better (below in the Advanced EDA).

In [60]:
```
# Create float columns for convenience, without losing the original ones

data['user_lat'] = pd.to_numeric(data['USER_LAT'])
data['user_lon'] = pd.to_numeric(data['USER_LONG'])
data['venue_lat'] = pd.to_numeric(data['VENUE_LAT'])
data['venue_lon'] = pd.to_numeric(data['VENUE_LONG'])
```

Venues

In [59]:
```
#import folium
#from folium.plugins import MarkerCluster


# Create a map centered at Helsinki (blue placeholder)
m = folium.Map(location=[60.1756, 24.9342], zoom_start=13)

# Define Venue Coordinates
coords = zip(data["venue_lat"], data["venue_lon"])

# Use MarkerCluster to group nearby markers
marker_cluster = MarkerCluster().add_to(m)

# Add points to the marker cluster
for coord in coords:
    folium.Marker(location=coord).add_to(marker_cluster)

# Display the map
m
```

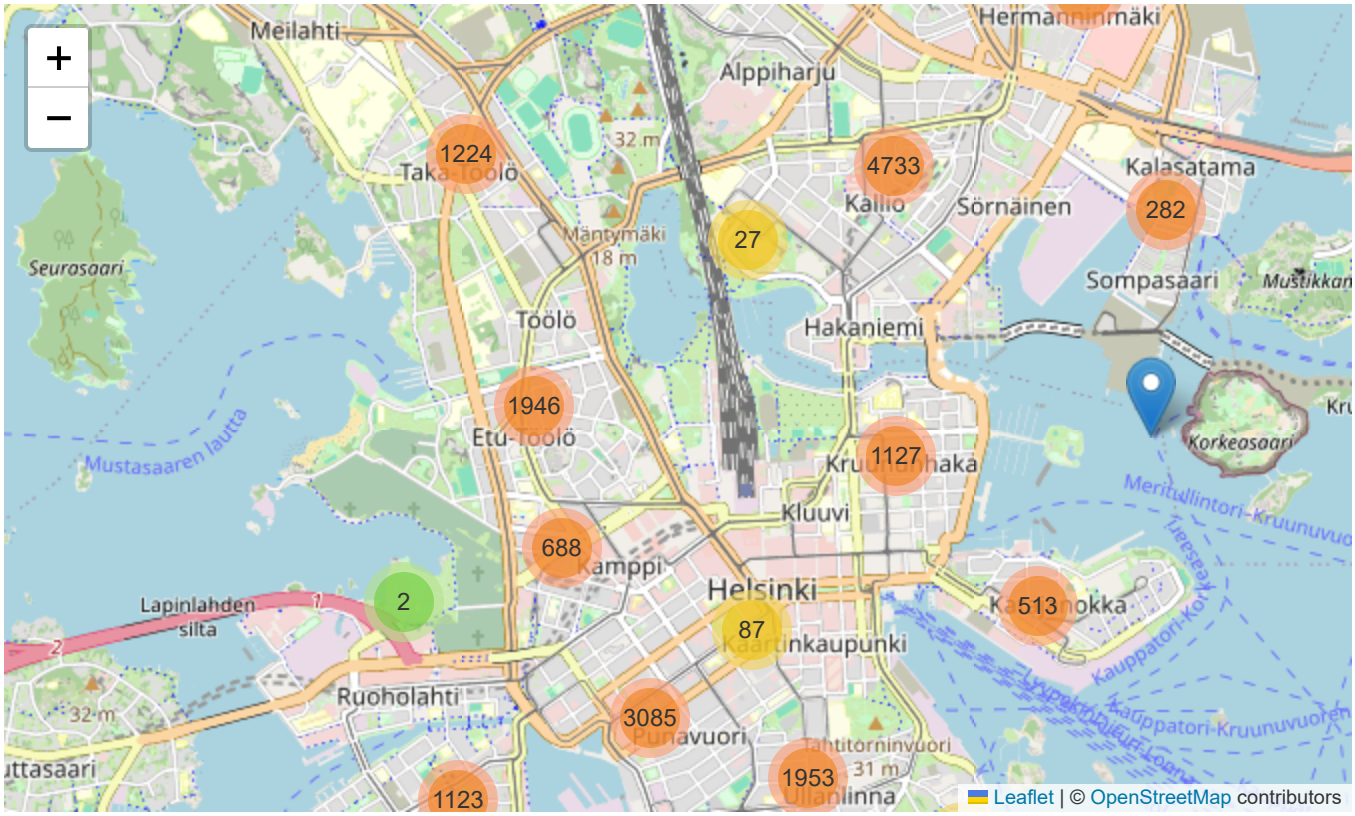
Out[59]:



Users

In [55]:
```
#import folium
#from folium.plugins import MarkerCluster


# Create a map centered at Helsinki (blue placeholder)
m = folium.Map(location=[60.1756, 24.9342], zoom_start=13)

# Define Venue Coordinates
coords = zip(data["user_lat"], data["user_lon"])

# Use MarkerCluster to group nearby markers
marker_cluster = MarkerCluster().add_to(m)

# Add points to the marker cluster
for coord in coords:
    folium.Marker(location=coord).add_to(marker_cluster)

# Display the map
m
```

Out[55]:



Distributions

Three variables carry information about **orders' delivery time**

1. "ACTUAL_DELIVERY_MINUTES"
2. "ESTIMATED_DELIVERY_MINUTES",
3. "ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES"

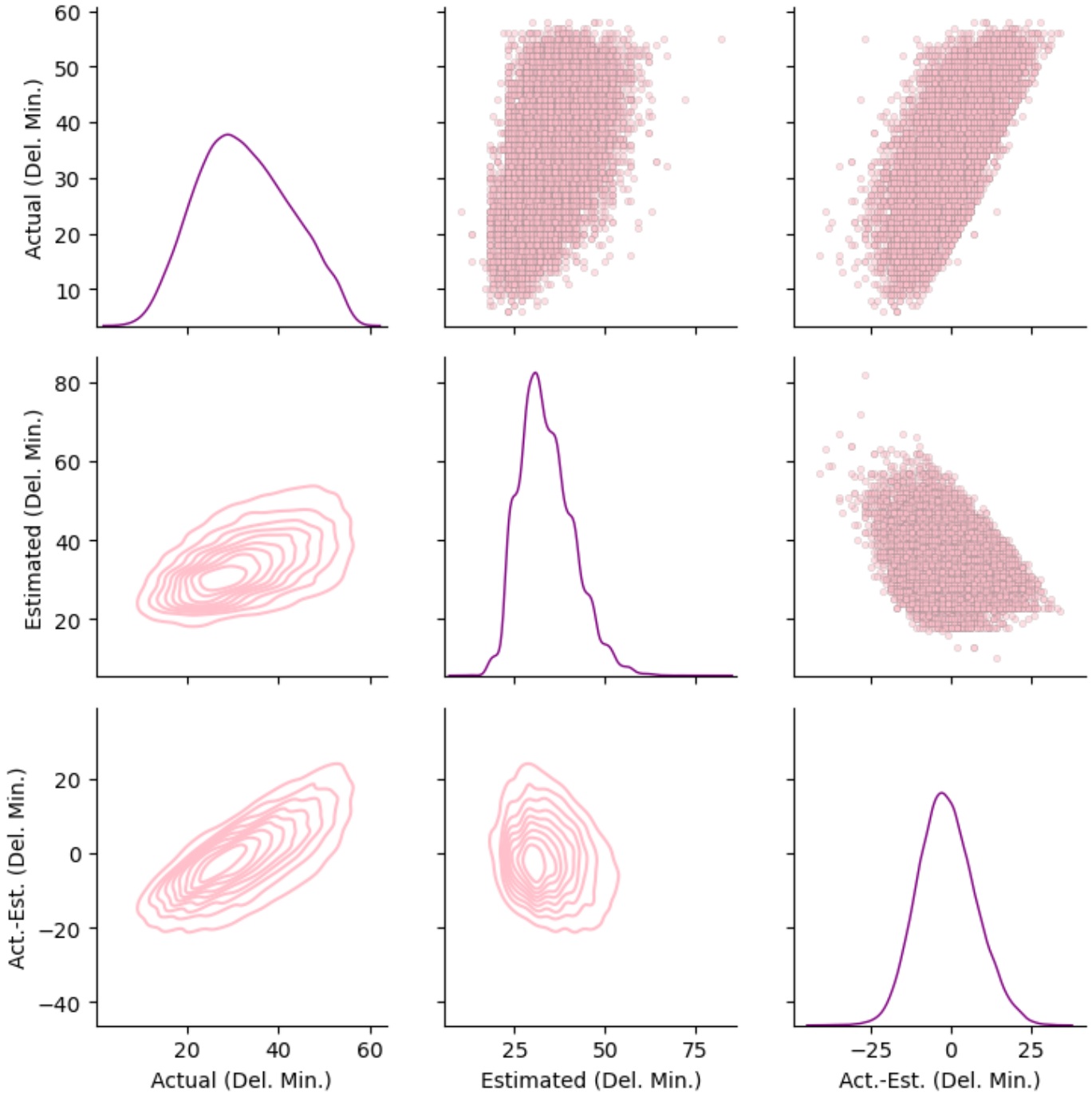
We can look at their single **distributions** (diagonal). Are they **gaussians**? Are they **skewed**?

What's the relation between each variable and the others (scatterplots and kdeplot, off-diagonal)

We don't see anything particularly surprising. The **(Actual - Estimated Delivery Minutes) variable is gaussian** and **centered below the 0**. Times are calculated so that **the user doesn't experience delays, on average.**

In [111…

```
# import seaborn as sns
sub_data = data[["ACTUAL_DELIVERY_MINUTES",
                 "ESTIMATED_DELIVERY_MINUTES",
                 "ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES"]]
sub_data.columns = ["Actual (Del. Min.)", "Estimated (Del. Min.)", "Act.-Est. (Del. Min.
g = sns.PairGrid(sub_data)
g.map_upper(sns.scatterplot,alpha=0.5, s=10, edgecolor='grey', color = 'pink')
g.map_lower(sns.kdeplot, color = 'pink')
g.map_diag(sns.kdeplot, lw=1, legend=False, color = 'purple')
plt.show()
```



In [114…]
```
sub_data.describe()
```

Out[114]:

	Actual (Del. Min.)	Estimated (Del. Min.)	Act.-Est. (Del. Min.)
count	18706.000000	18706.000000	18706.000000
mean	32.608254	33.809313	-1.201058
std	10.018879	7.340283	8.979834
min	6.000000	10.000000	-41.000000
25%	25.000000	28.000000	-7.000000
50%	32.000000	33.000000	-2.000000
75%	40.000000	38.000000	5.000000
max	58.000000	82.000000	34.000000

Time Series

We can look for interesting **patterns** across time. Should we focus on **weekly/daily/hourly trends**?

Some **variables** with interesting behaviours could be:

---

1. **(Actual - Estimated) Delivery Minutes (Delay)**

- Do delays happen more often in certain weekdays/hours of day/day of month?
- What's the time series like for the total window of time in the data? Are there patterns we can predict?

---

1. **Number of Orders**

- How does the number of orders depend on weekdays/hours of day/day of month?
- What's the time series like for the total window of time in the data? Are there patterns we can predict?

---

1. **Other**

- Should we expect any relation between time and coordinates or user-venue distance?

In [244…

```
# Convert into datetime format
data["Datetime"] = pd.to_datetime(data["TIMESTAMP"])

# Define new dataframe, for convenience
time_df = pd.DataFrame({})
time_df["date"] = data["Datetime"]
time_df["minute"] = data["Datetime"].dt.minute
time_df["hour"] = data["Datetime"].dt.hour
time_df["day"] = data["Datetime"].dt.day
time_df["month"] = data["Datetime"].dt.month
time_df["year"] = data["Datetime"].dt.year


time_df["Delay"] = data["ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES"]
time_df["Counts"] = len(time_df)*[1]

time_df
```

Out[244]:

	date	minute	hour	day	month	year	Delay	Counts
0	2020-08-01 06:07:00	7	6	1	8	2020	-19	1
1	2020-08-01 06:17:00	17	6	1	8	2020	-7	1
2	2020-08-01 06:54:00	54	6	1	8	2020	-17	1
3	2020-08-01 07:09:00	9	7	1	8	2020	-2	1
4	2020-08-01 07:10:00	10	7	1	8	2020	-1	1
...	...	...	...	...	...	...	...	...
18701	2020-09-30 19:27:00	27	19	30	9	2020	-1	1
18702	2020-09-30 19:36:00	36	19	30	9	2020	-8	1
18703	2020-09-30 19:39:00	39	19	30	9	2020	-14	1

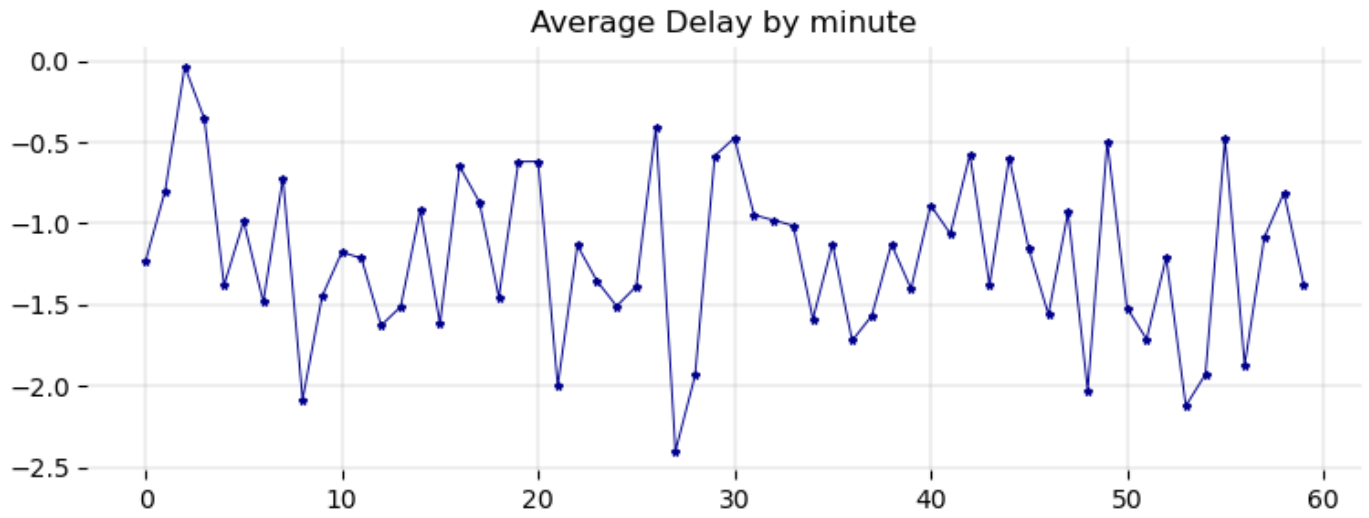
18704	2020-09-30 19:42:00	42	19	30	9	2020	-11	1
18705	2020-09-30 20:04:00	4	20	30	9	2020	-7	1

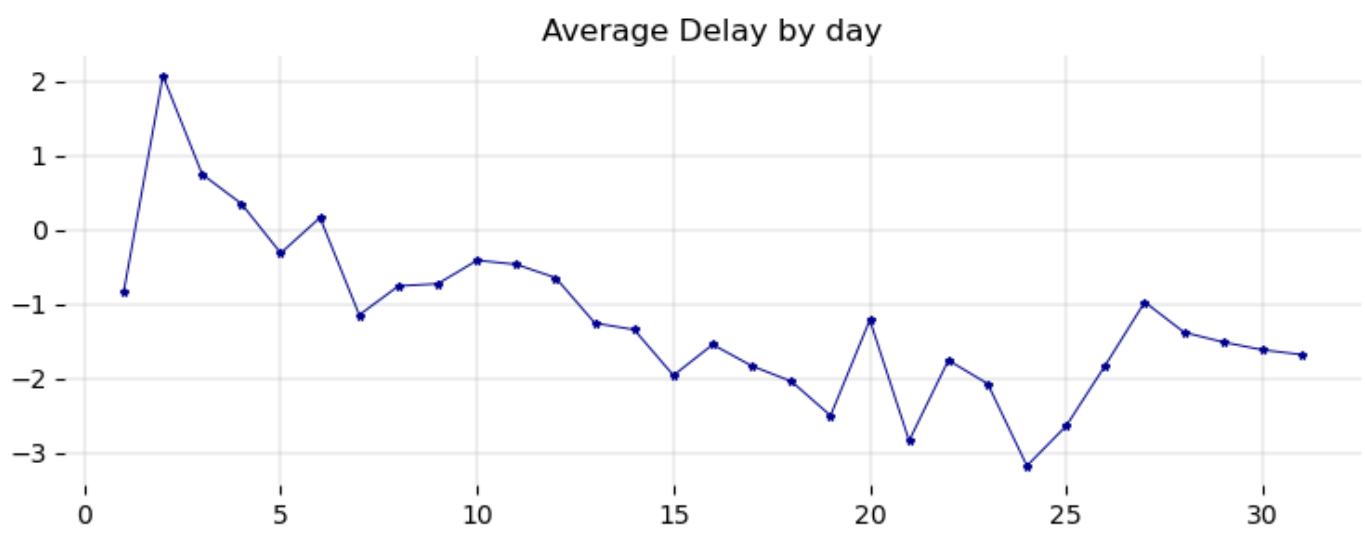
18706 rows × 8 columns

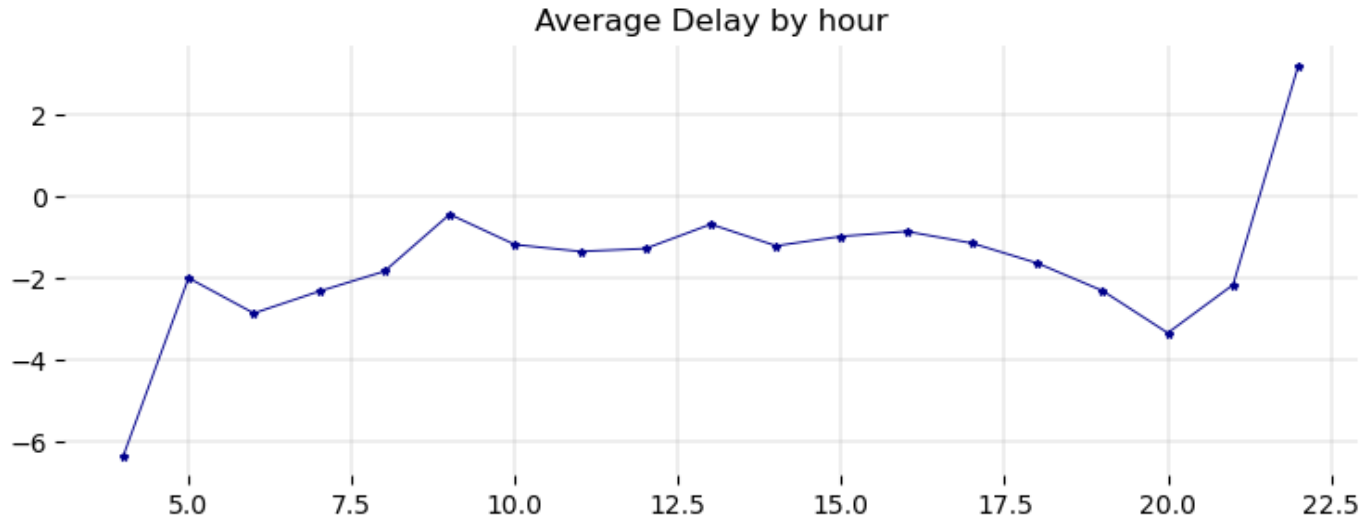
Delays

In [245…

```
def time_series_plotter(granularity = 'day'):
    result_df = time_df.groupby([granularity]).agg({'Delay': 'mean'}).reset_index()
    plt.figure(figsize = (9,3))
    plt.plot(result_df[[granularity]], result_df["Delay"],  linewidth = 0.69, color = 'd
             marker = '*', markersize = 3.1)
    plt.title(f"Average Delay by {granularity}")
    plt.grid(True, linewidth = 0.3)
    plt.box(False)
    plt.show()

def time_series_plotter_counts(granularity = 'day'):
    result_df = time_df.groupby([granularity]).agg({'Counts': 'sum'}).reset_index()
    plt.figure(figsize = (9,3))
    plt.plot(result_df[[granularity]], result_df["Counts"],  linewidth = 0.69, color = '
             marker = '*', markersize = 3.1)
    plt.title(f"Nr. of Orders by {granularity}")
    plt.grid(True, linewidth = 0.3)
    plt.box(False)
    plt.show()

time_series_plotter('minute')
time_series_plotter('day')
time_series_plotter('hour')
```



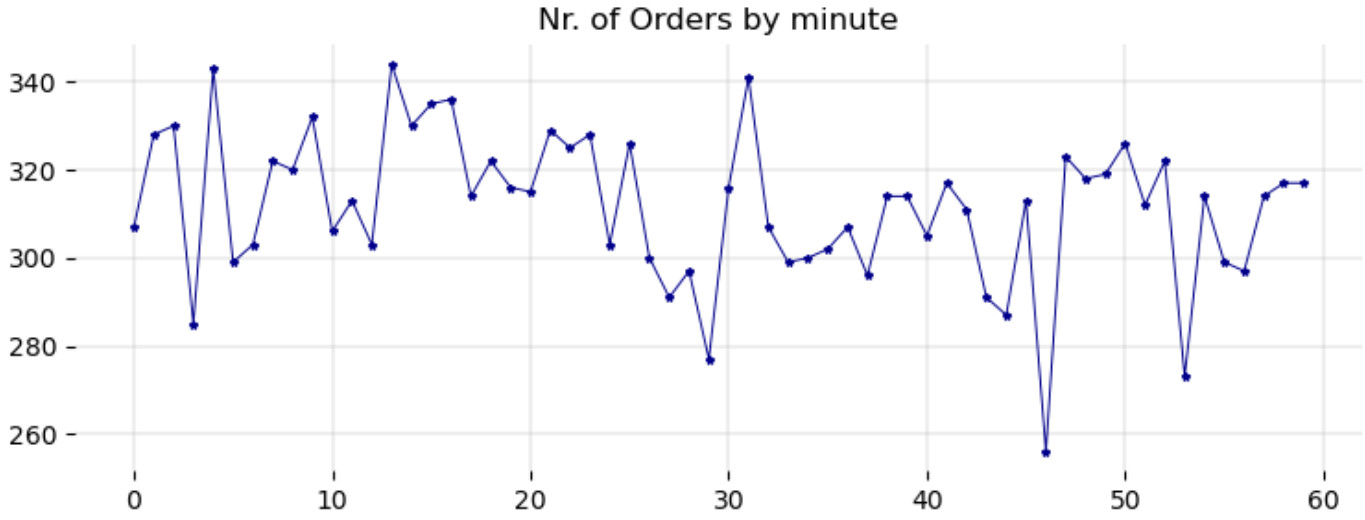


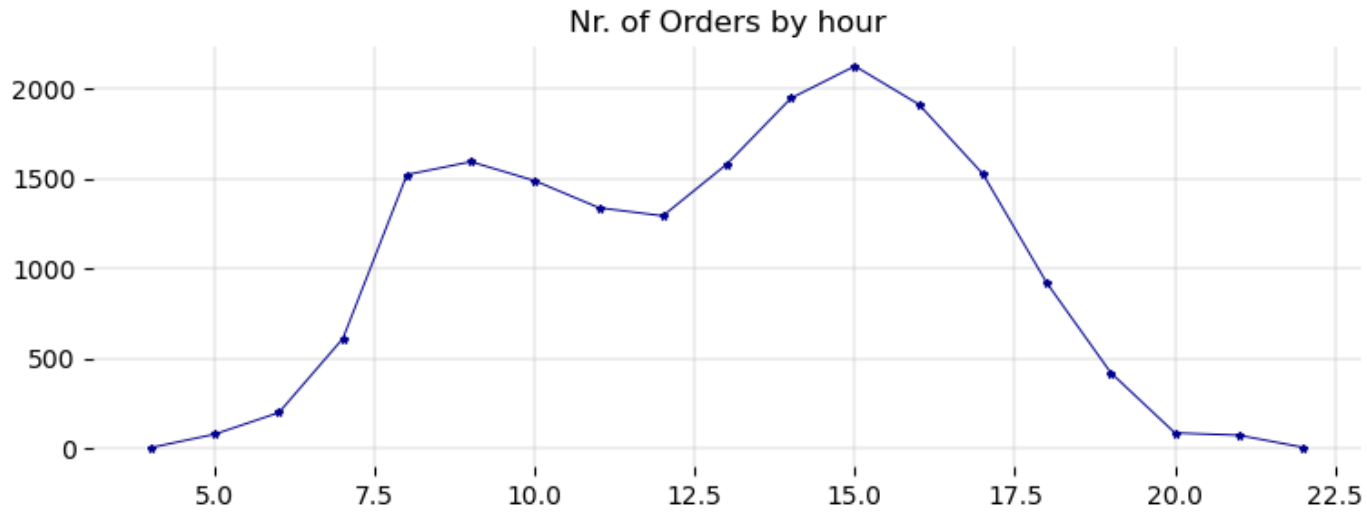


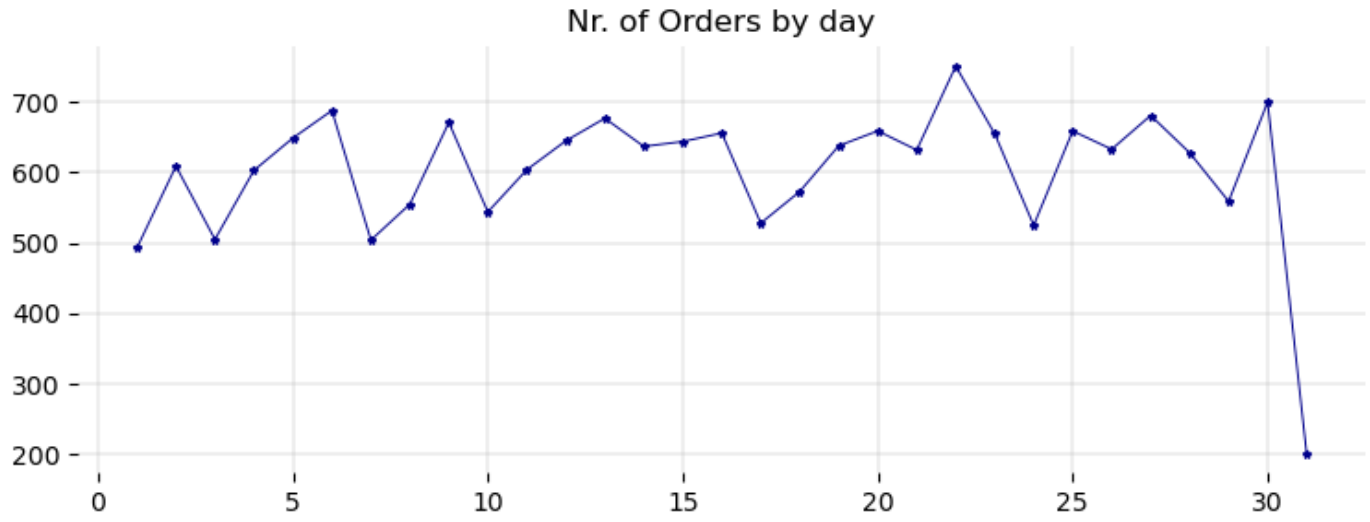
Orders

In [246…

```
time_series_plotter_counts('minute')
time_series_plotter_counts('hour')
time_series_plotter_counts('day')
```







In [247…

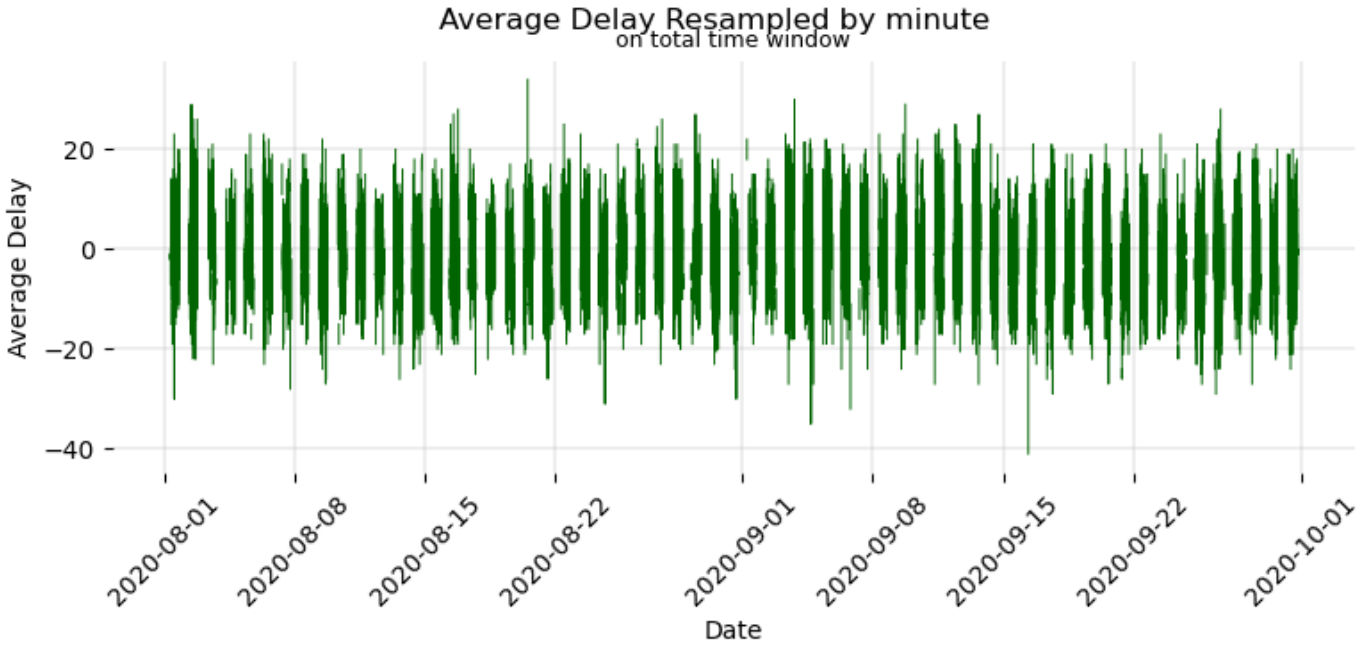
```
# This is needed for the resampling in the following plots. Set the "date" column as the

time_df_n = time_df
time_df_n.set_index('date', inplace=True)
```

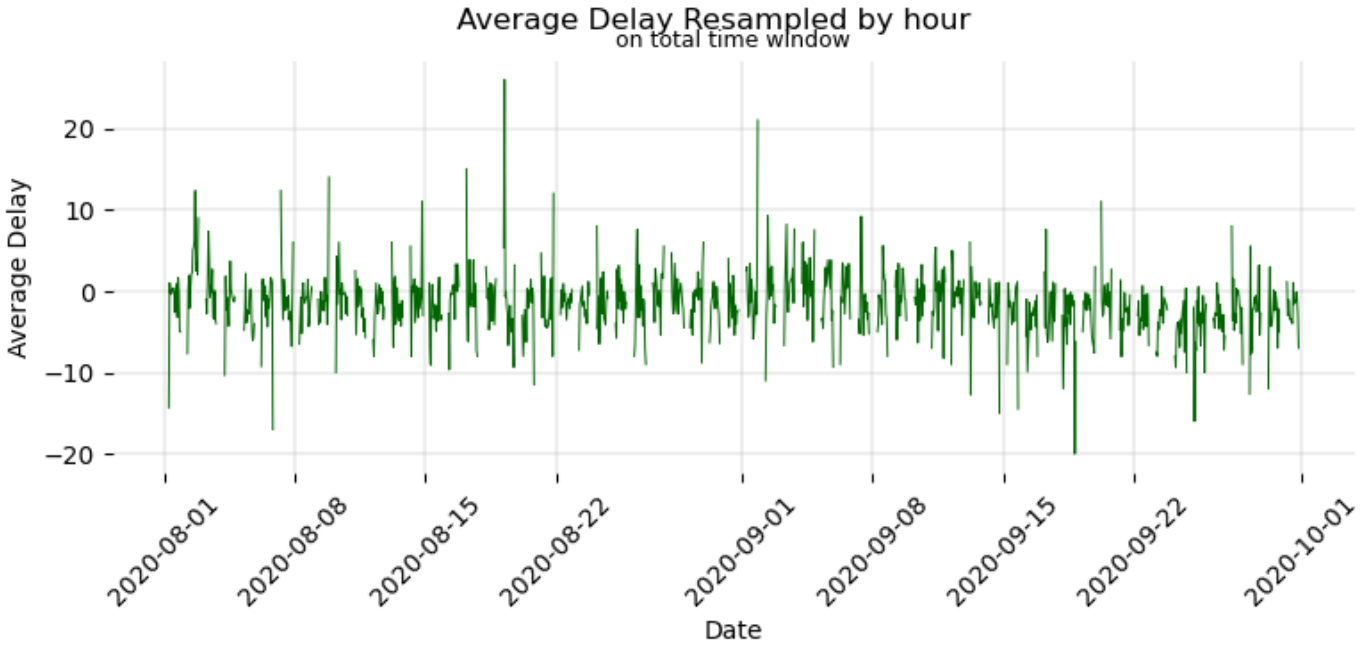
Delays

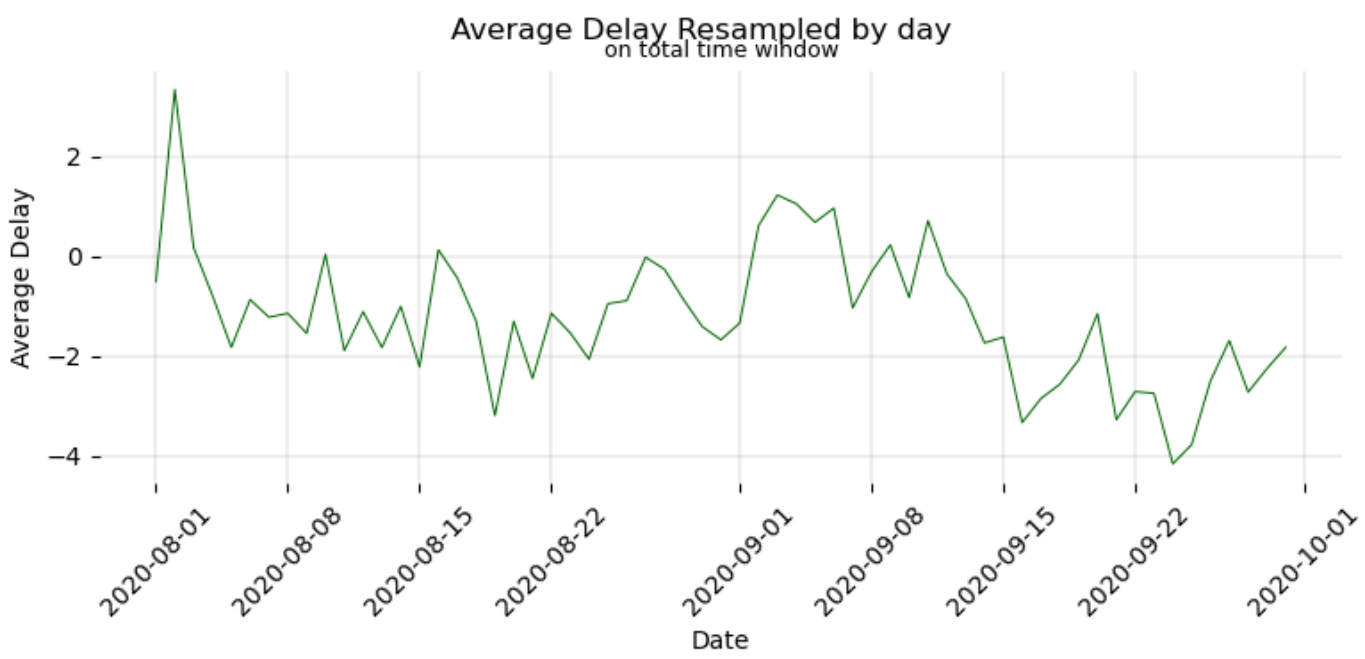
In [301…

```
def timeseries_plotter(resampled_by = 'day', indicator = 'mean'):
    dictio = {'minute': 'T','hour': 'H',"day":'D', 'month':'M'}

    # Resample by day and calculate the mean delay for each day
    if indicator == 'mean':
        resampled_df = time_df_n.resample(dictio[resampled_by]).mean()
    elif indicator == 'sum':
        resampled_df = time_df_n.resample(dictio[resampled_by]).sum()


    plt.figure(figsize = (9,3))
    # Plot the resampled data
    plt.plot(resampled_df.index, resampled_df['Delay'], linewidth = 0.69, color = 'darkg
    plt.xlabel('Date')
    plt.ylabel('Average Delay')
    plt.suptitle(f'Average Delay Resampled by {resampled_by}')
    plt.title('on total time window', fontsize = 9)
    plt.xticks(rotation = 45)
    plt.grid(True, linewidth = 0.3)
    plt.box(False)
    plt.show()
```

```
#'''
timeseries_plotter('minute')
timeseries_plotter('hour')
timeseries_plotter('day')
'''
timeseries_plotter('minute', indicator = 'sum')
timeseries_plotter('hour', indicator = 'sum')
timeseries_plotter('day', indicator = 'sum')
'''
```





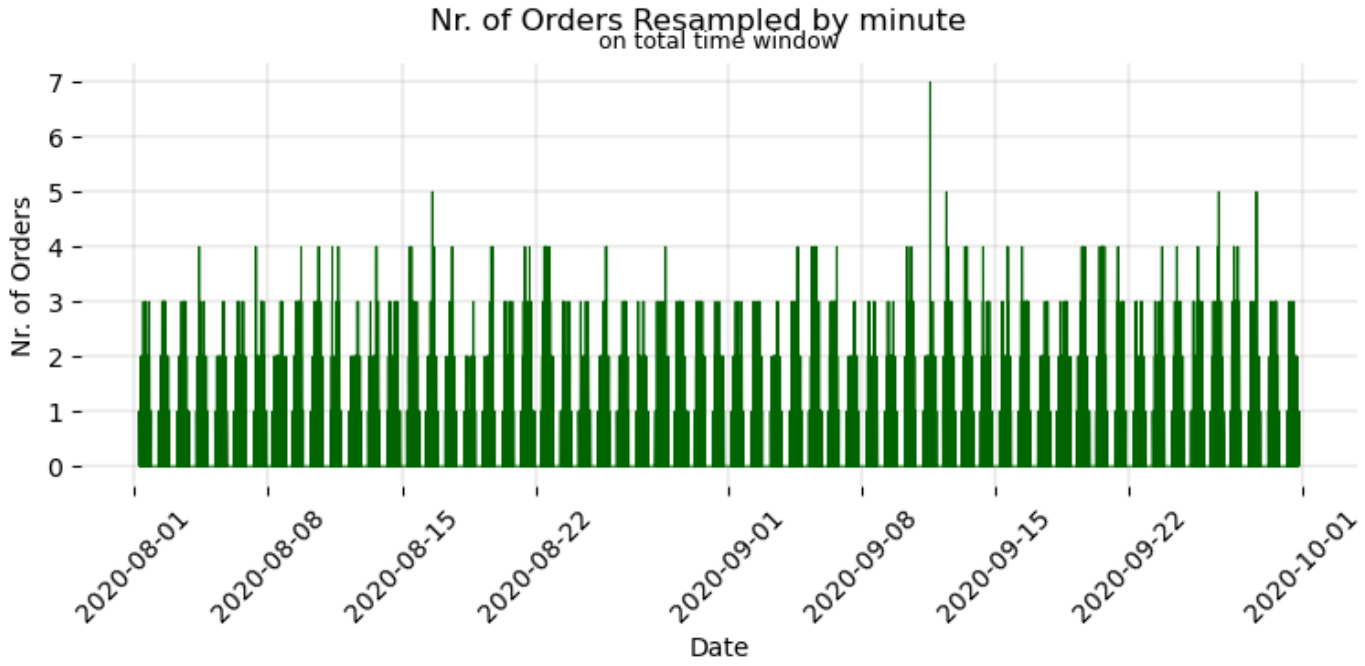


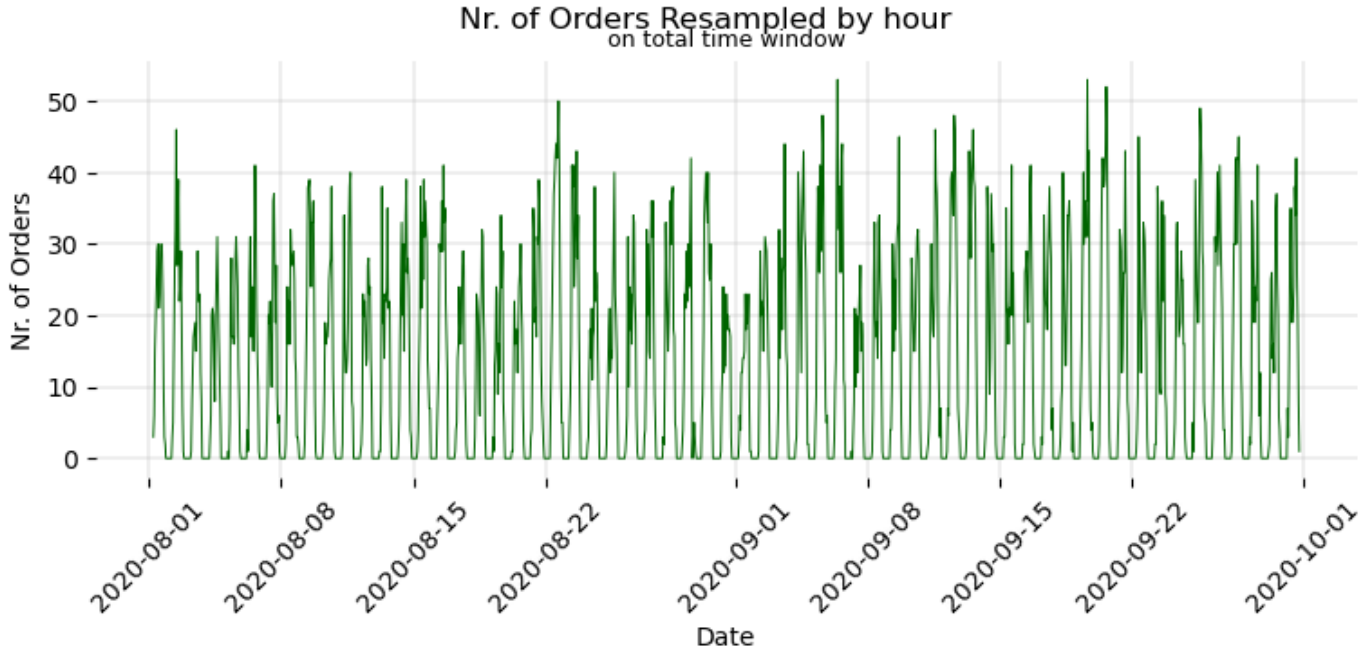
Out[301]:
```
"\ntimeseries_plotter('minute', indicator = 'sum')\ntimeseries_plotter('hour', indicator
= 'sum')\ntimeseries_plotter('day', indicator = 'sum')\n"
```

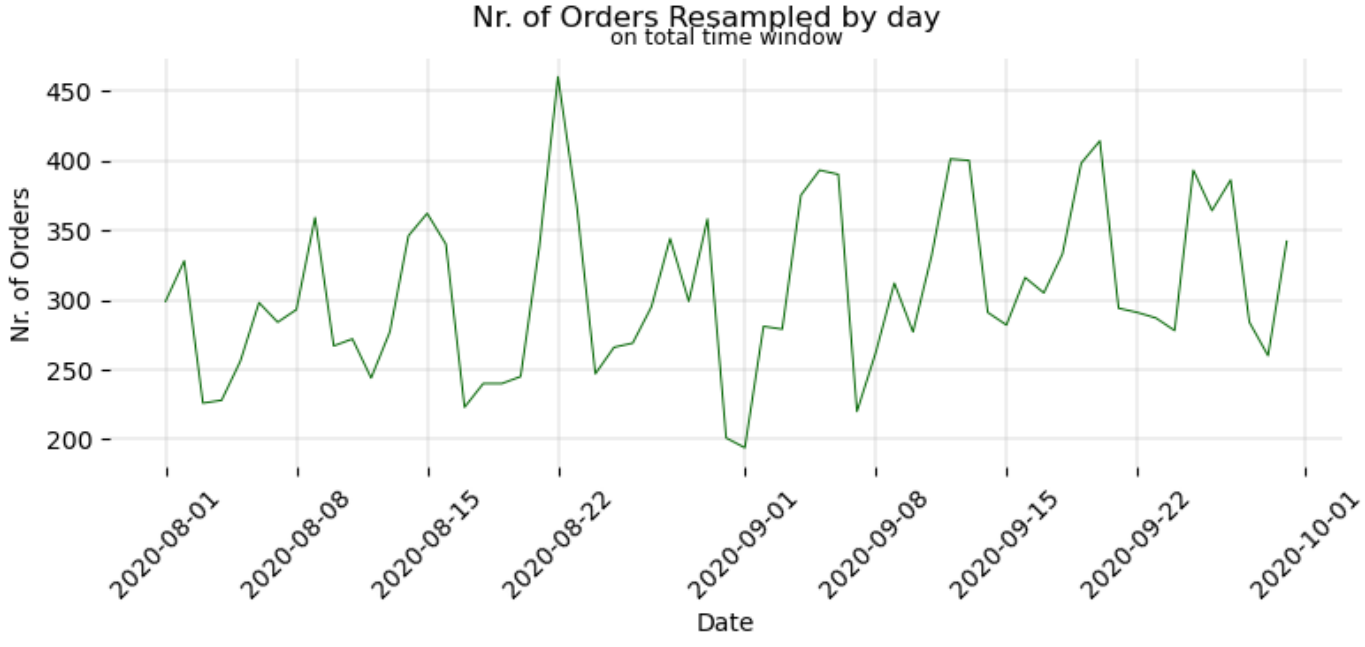
Orders

In [302…

```python
def timeseries_plotter_counts(resampled_by = 'day'):
    dictio = {'minute': 'T','hour': 'H',"day":'D', 'month':'M'}

    # Resample by day and calculate the mean delay for each day
    resampled_df = time_df_n.resample(dictio[resampled_by]).sum()

    plt.figure(figsize = (9,3))
    # Plot the resampled data
    plt.plot(resampled_df.index, resampled_df['Counts'], linewidth = 0.69, color = 'dark
    plt.xlabel('Date')
    plt.ylabel('Nr. of Orders')
    plt.suptitle(f'Nr. of Orders Resampled by {resampled_by}')
    plt.title('on total time window', fontsize = 9)
    plt.xticks(rotation = 45)
    plt.grid(True, linewidth = 0.3)
    plt.box(False)
    plt.show()

timeseries_plotter_counts('minute')
timeseries_plotter_counts('hour')
timeseries_plotter_counts('day')
```


Nr. of Orders Resampled by minute
on total time window
Nr. of Orders
0
1
2
3
4
5
6
7
2020-08-01
2020-08-08
2020-08-15
2020-08-22
2020-09-01
2020-09-08
2020-09-15
2020-09-22
2020-10-01
Date


Nr. of Orders Resampled by hour
on total time window
Nr. of Orders
0
10
20
30
40
50
2020-08-01
2020-08-08
2020-08-15
2020-08-22
2020-09-01
2020-09-08
2020-09-15
2020-09-22
2020-10-01
Date


Nr. of Orders Resampled by day
on total time window
Nr. of Orders
200
250
300
350
400
450
2020-08-01
2020-08-08
2020-08-15
2020-08-22
2020-09-01
2020-09-08
2020-09-15
2020-09-22
2020-10-01
Date

ADVANCED EDA

Interactive Dashboards

Links for the dashboards

- EDA 1
- EDA 2
- EDA 3

Orders and delays time trends

Analysis of Delays (Actual - Estimated Delivery Time > 0)

- Exploring **distributions** for **actual and estimated delivery times**, as well as their **difference**.
- The latter appears to have a gaussian distibution with average lower than zero. **Most of the times users don't experience delays**
- Are delays linked to specific venues? How are they impacted by venues popularity or location?

MODEL

Some variables are an obvious choice for a **prediction task**.
For instance, there's no doubt about the benefit and the need of forecasting the number of orders received in a day/hour.

Aim

- The EDA showed some interesting trends/patterns but didn't suggest any unexpected target variable.
- The best way to use our data is to predict a **well defined variable**, the knowledge of which is of obvious **utiliy**.
- **We thus focus on the average delay** (Actual - Estimated Delivery Minutes).

We try to complete the following **task**:
PREDICT THE AVERAGE DELAY IN THE NEXT DAY/HOUR

Benefits

This could be helpful for many reasons. For example:

- the company could encourage a **higher number of couriers** in those day/hours where more **delays are expected**, countering the latter by properly distributing deliveries.
- the app could **estimate higher delivery times**, in order not to disappoint users. In this case users would just be 'conscious' about the longer waiting times.

Approach

- After defining our predictive goal, we will **model the time series** considering both **endogenous and exogenous features.**

SARIMAX

Seasonal **A**uto**R**egressive **I**ntegrated **M**oving **Average** with He**x**ogenous features

---

!Remember!

The SARIMAX model is defined by the order parameters (p, d, q) for the ARIMA part and (P, D, Q, s) for the SARIMA part:

- **p**: The order of the AutoRegressive (AR) part. It represents the number of lags of the dependent variable to be used as predictors.
- **d**: The degree of differencing. It indicates how many times the data have had past values subtracted to make the series stationary.
- **q**: The order of the Moving Average (MA) part. It signifies the number of lagged forecast errors that should go into the ARIMA model.
- **P**: The order of the seasonal part of the AutoRegressive (AR) model.
- **D**: The degree of seasonal differencing.
- **Q**: The order of the seasonal part of the Moving Average (MA) model.
- **s**: The length of the seasonal cycle.

---

1. We can work with **daily or hourly** delays. We choose **hourly** time series since we have a good number of samples.

2. **We expect periodicity/seasonality**. At least we should observe trends on a **24-hours basis**

3. We can expect delays to be influenced by **hexogenous features**, such as **weather condition** or number of orders.
   -- We rely on **preciptations** assuming this feature is known in advance when performing the forecast. Weathermen are reliable nowadays :)
   -- We also consider **weekday and hour** of the day as hexogenous features (for example we know orders are not delivered in certain hours: that is additional information that the model may not completely capture in the naked time series).

We will find good orders for the model by looking at the Time Series **Autocorrelation** and **Partial Autocorrelation** for different lags.
The presence of 'high' (outside confidence interval) correlation will suggest the parameters for the SARIMAX model.

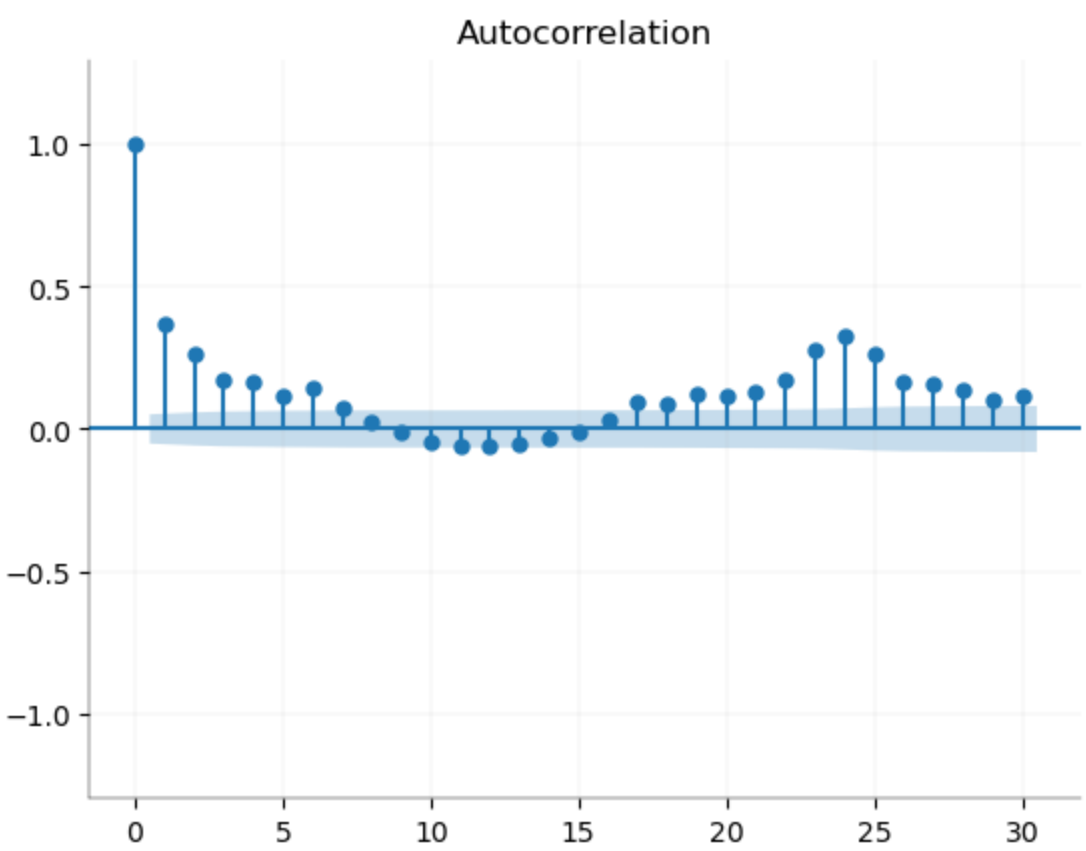
In [576…

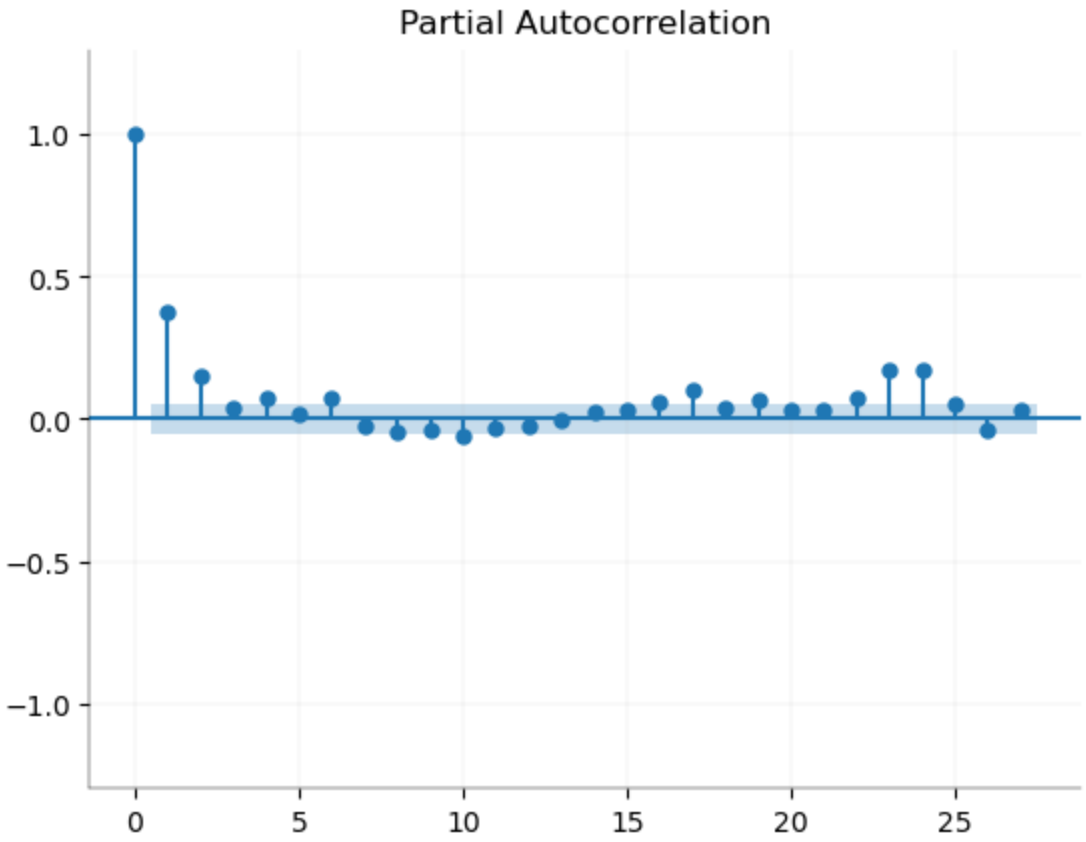
```
#import pandas as pd
#import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

resampled_df = time_df_n.resample('H').sum()
#resampled_df = time_df_n.resample('D').mean()

def correloplotter(x):
    # Compute the autocorrelation function (ACF)
    acf_vals = x#acf(x)
    pacf_vals = x#pacf(x)


    # Plot the ACF
    plot_acf(acf_vals, lags=30)
    plt.ylim([-1.3,1.3])
    plt.grid(linewidth = 0.1)
    #plt.xticks(range(0,20))
    plt.box(True)
    plt.gca().spines['right'].set_visible(False)
    plt.gca().spines['top'].set_visible(False)
    # Set the width of the axes
    plt.gca().spines['left'].set_linewidth(0.31)
    plt.gca().spines['bottom'].set_linewidth(0.31)

    # Plot the PACF
    plot_pacf(pacf_vals, lags =27)
    plt.ylim([-1.3,1.3])
    plt.grid(linewidth = 0.1)
    plt.box(True)
    plt.gca().spines['right'].set_visible(False)
    plt.gca().spines['top'].set_visible(False)
    # Set the width of the axes
    plt.gca().spines['left'].set_linewidth(0.31)
    plt.gca().spines['bottom'].set_linewidth(0.31)


    plt.show()
```

In [577…

```
correloplotter(resampled_df['Delay'])
```

```
C:\Users\diesi\anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureW
arning:

The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.
13, the default will change tounadjusted Yule-Walker ('ywm'). You can use this method no
w by setting method='ywm'.
```



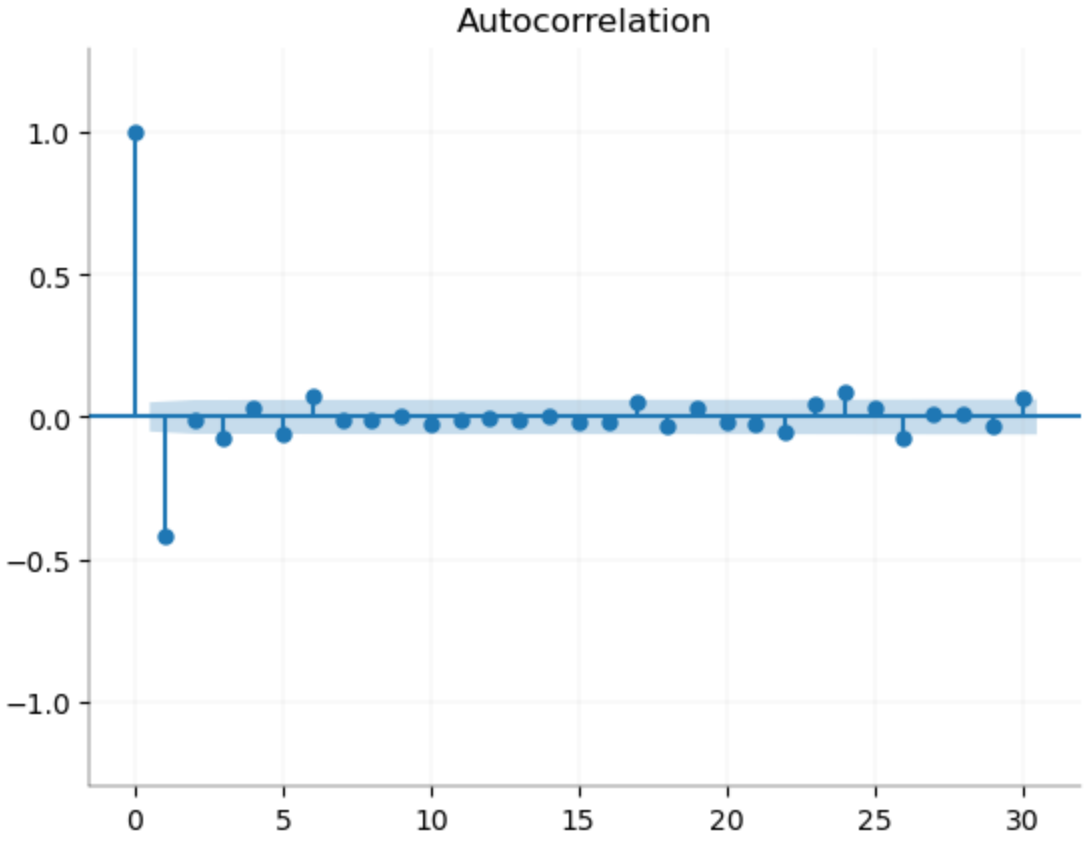


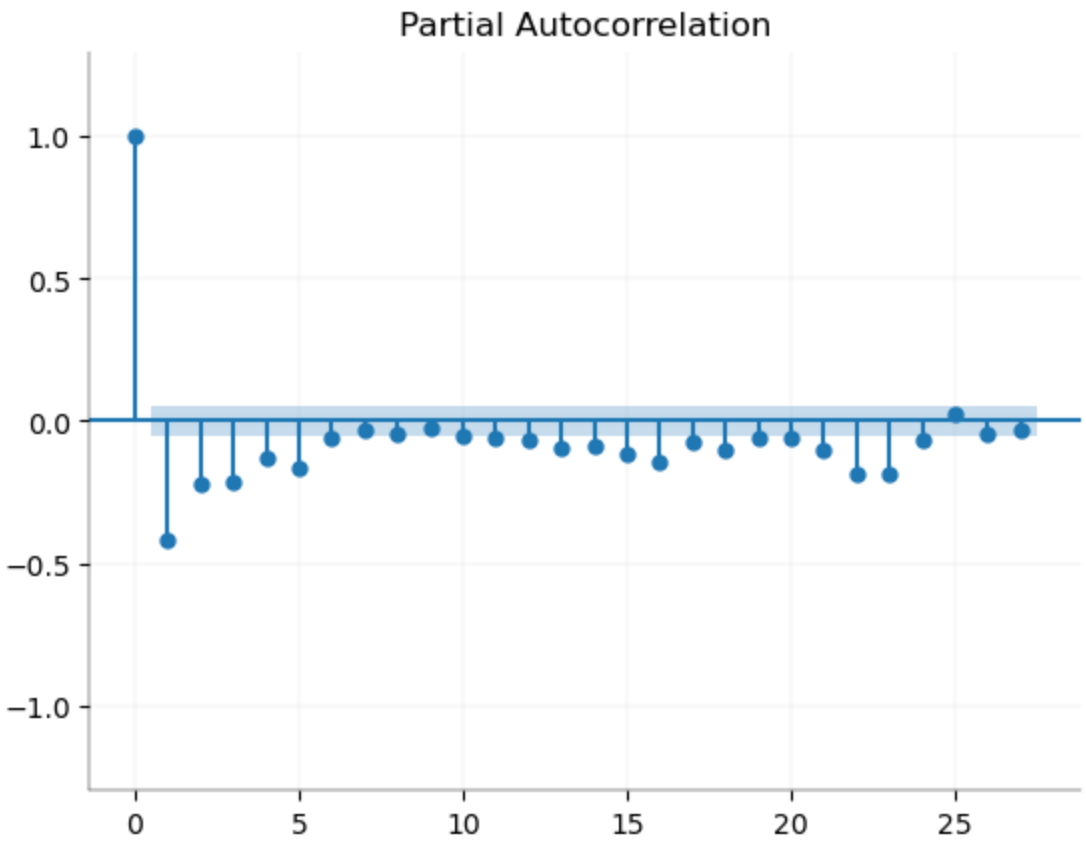
In [578…

```
resampled_df['delay_diff'] = resampled_df['Delay'].diff()
resampled_df_dropna = resampled_df.dropna()

correloplotter(resampled_df_dropna['delay_diff'])
```

```
C:\Users\diesi\anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureW
arning:

The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.
13, the default will change tounadjusted Yule-Walker ('ywm'). You can use this method no
```

```
w by setting method='ywm'.
```



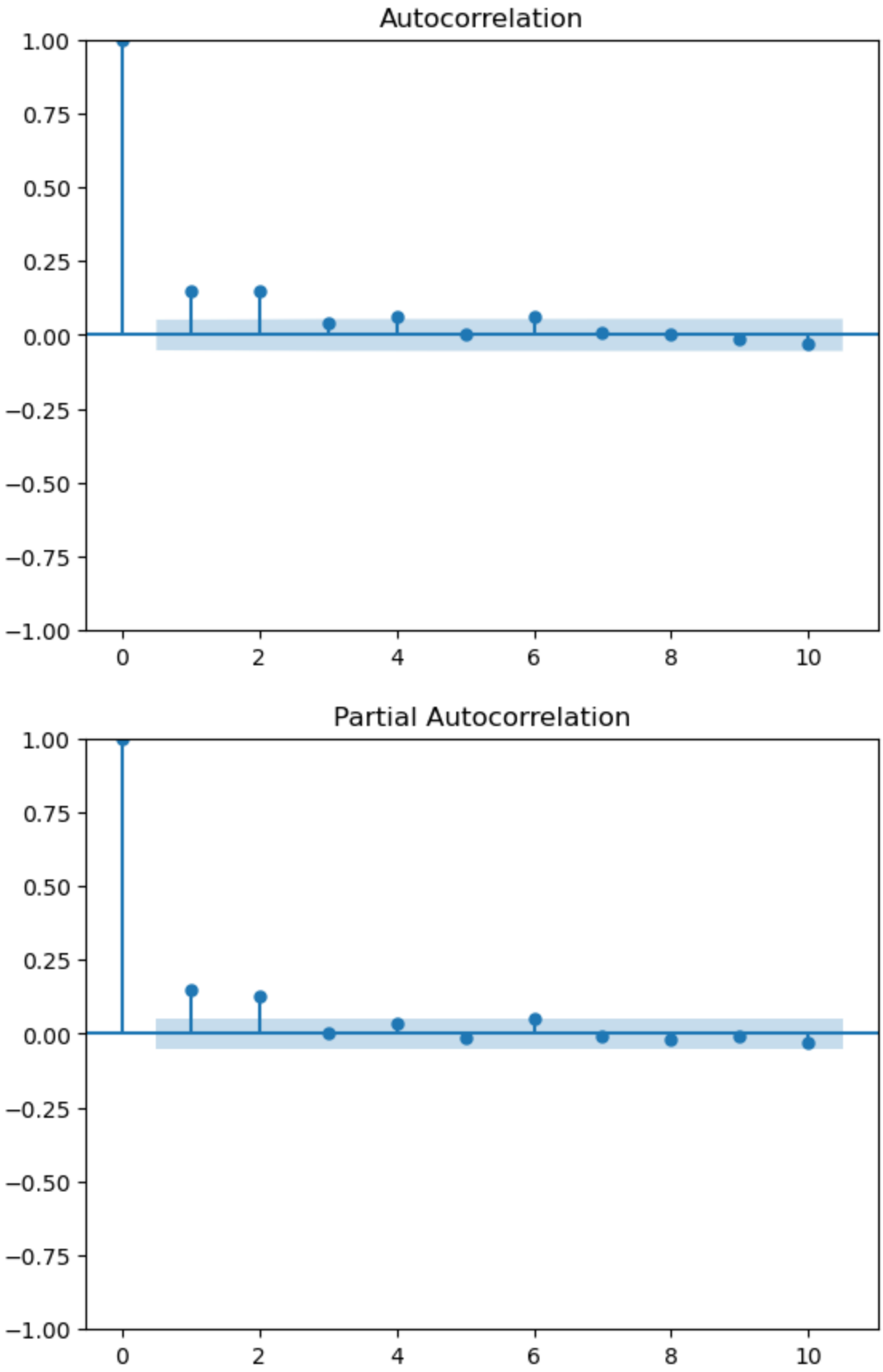


Sarima

In [537…

```
# ACF and PACF plots for seasonal component
seasonal_diff = resampled_df['Delay'].diff(24).dropna()

plot_acf(seasonal_diff, lags=10, zero=True)
plt.show()
```

```
plot_pacf(seasonal_diff, lags=10, zero=True)
plt.show()
```



Analysis

Let's analyze the findings:

Original Series:

- ACF shows significant spikes at lags 1, 2, 3, 4, 5, 6, and 16 to 27.
- PACF shows significant spikes at lags 1 and 2.

Differencing (non-seasonal):

- ACF after differencing shows significant spikes at lags 1, 6, and 24.
- PACF after differencing shows significant spikes at lags 1, 2, 3, 4, 5, and 16 to 24.

Seasonal Differencing (s=24):

- ACF after seasonal differencing shows significant spikes at lags 1 and 2.
- PACF after seasonal differencing shows significant spikes at lags 1 and 2.

Deductions:

- Considering the ACF and PACF spikes after non-seasonal differencing, it seems there is a seasonal component with a period of 24 hours, as one could expect.

Suggested Orders:

Based on the observations, a SARIMA model may be appropriate. The suggested orders could be as follows:

- **Seasonal Order (S)**: 24 (since the data is hourly and the seasonality is observed every 24 hours).
- **Autoregressive Order (p)**: Consider values around 2 for both the non-seasonal and seasonal components based on the PACF.
- **Integrated Order (d)**: 1 for the non-seasonal component (since differencing at lag 1 shows significance).
- **Seasonal Integrated Order (D)**: 1 for the seasonal component (since seasonal differencing at lag 24 shows significance).
- **Moving Average Order (q)**: Consider values around 2 for both the non-seasonal and seasonal components based on the ACF.

Model

Dataframe creation. We will have:

- **Target Variable** ->*Average Delay*
  Target Variable and time series variable
- **Hex. Features** -> *weekday (number from 1 to 7), hour of the day (1 to 24)*, *Rain (precipitaiotn in mm)*
  We could use categorical features but we retain the numerical form

---

Additional considerations:

- **order** = (2, 1, 2)
- **seasonal_order** = (2, 1, 2, 24)
- We handle **null values** (for hours where there are no orders at all) by replacing them with the average hourly delay across the whole series (= *-1.5*)

In [736…

```
# Define new dataframe, for convenience
time_df = pd.DataFrame({})
time_df["date"] = data["Datetime"]
time_df["minute"] = data["Datetime"].dt.minute
time_df["hour"] = data["Datetime"].dt.hour
time_df["day"] = data["Datetime"].dt.day
time_df["month"] = data["Datetime"].dt.month
time_df["year"] = data["Datetime"].dt.year
time_df["weekday"] = data["Datetime"].dt.weekday


time_df["Delay"] = data["ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES"]
time_df["Counts"] = len(time_df)*[1]
time_df["Rain"] = data["PRECIPITATION"]

# This is needed for the resampling in the following plots. Set the "date" column as the
time_df_n = time_df
time_df_n.set_index('date', inplace=True)

# Resample by considering hourly sum of delays
resampled_df = time_df_n.resample('H').sum()
#resampled_df = time_df_n.resample('D').mean()

#print(resampled_df['Delay'].dropna().mean())

for column in resampled_df.columns:
    if column != 'Counts':
        # Perform the division only when "Counts" is not 0, otherwise set the result to
        resampled_df[column] = np.where(
            resampled_df["Counts"] != 0,
            resampled_df[column] / resampled_df["Counts"], -1.5) # or 0)
```

In [737…

```
train_size = int(len(resampled_df) * 0.9)  # 95% for training, 5% for testing
train, test = resampled_df[:train_size], resampled_df[train_size:]
```

In [738…

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

### HOURLY FORECAST ###
order = (2, 1, 2)  # Non-seasonal component
seasonal_order = (2, 1, 2, 24)  # Seasonal component

''''
### DAILY FORECAST ###
order = (2, 1, 1)  # Non-seasonal component
seasonal_order = (2, 1, 2, 7)  # Seasonal component
'''

'''
order = (3, 1, 3)  # Non-seasonal component
seasonal_order = (2, 1, 2, 24)  # Seasonal component
'''

# Fit the SARIMA model
sarima_model = SARIMAX(train['Delay'],
                       order=order,  exog=train[['hour','weekday','Rain']],
                       seasonal_order=seasonal_order,
                       method_kwargs={'maxiter':1000})
fitted_sarima_model = sarima_model.fit()

# Forecast on the test set
sarima_predictions = fitted_sarima_model.get_forecast(len(test), exog=test[['hour','week
```

```
C:\Users\diesi\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: Us
```

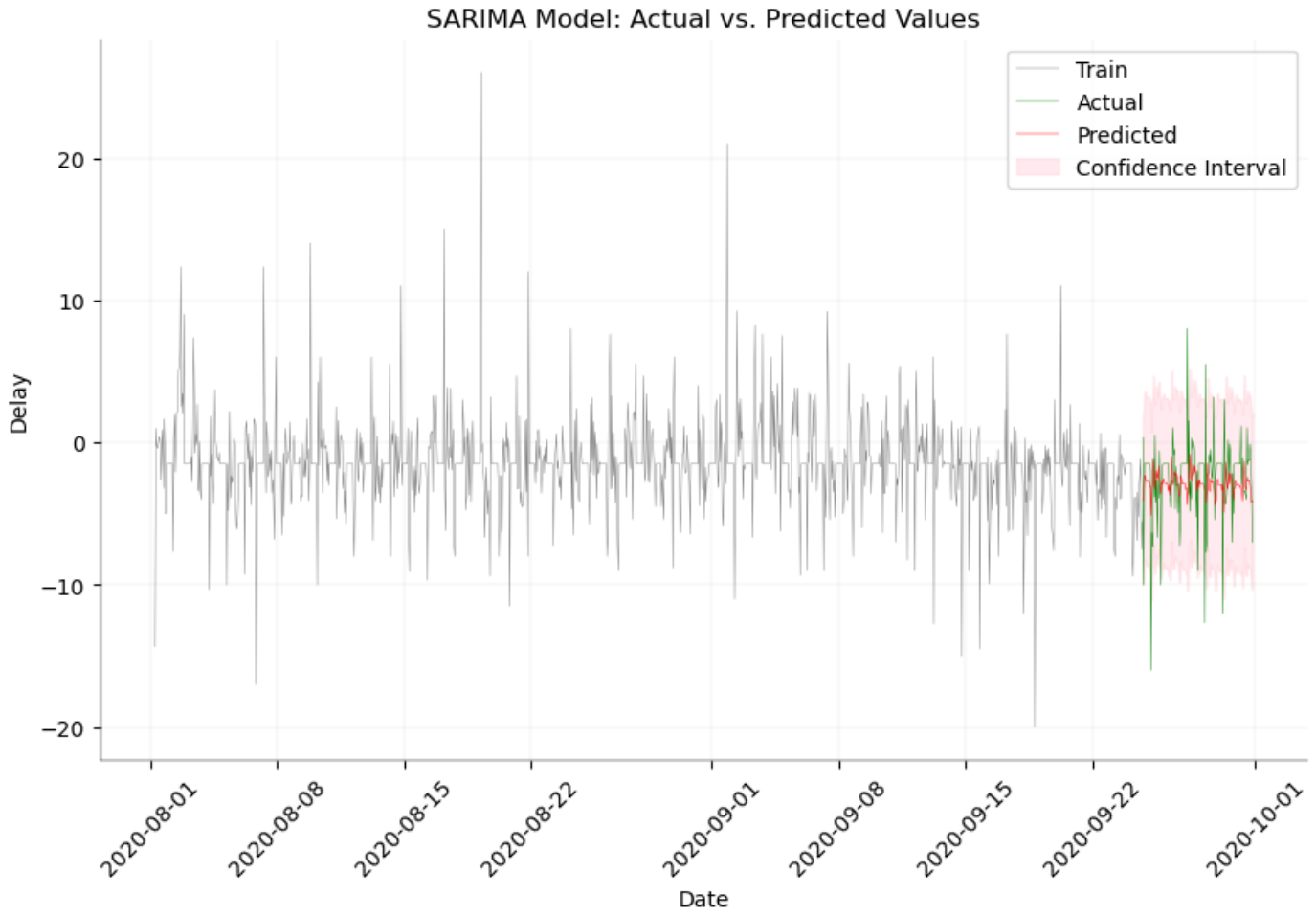
```
erWarning:

Non-stationary starting autoregressive parameters found. Using zeros as starting paramet
ers.

C:\Users\diesi\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning:

Non-invertible starting MA parameters found. Using zeros as starting parameters.

C:\Users\diesi\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWar
ning:

Maximum Likelihood optimization failed to converge. Check mle_retvals
```

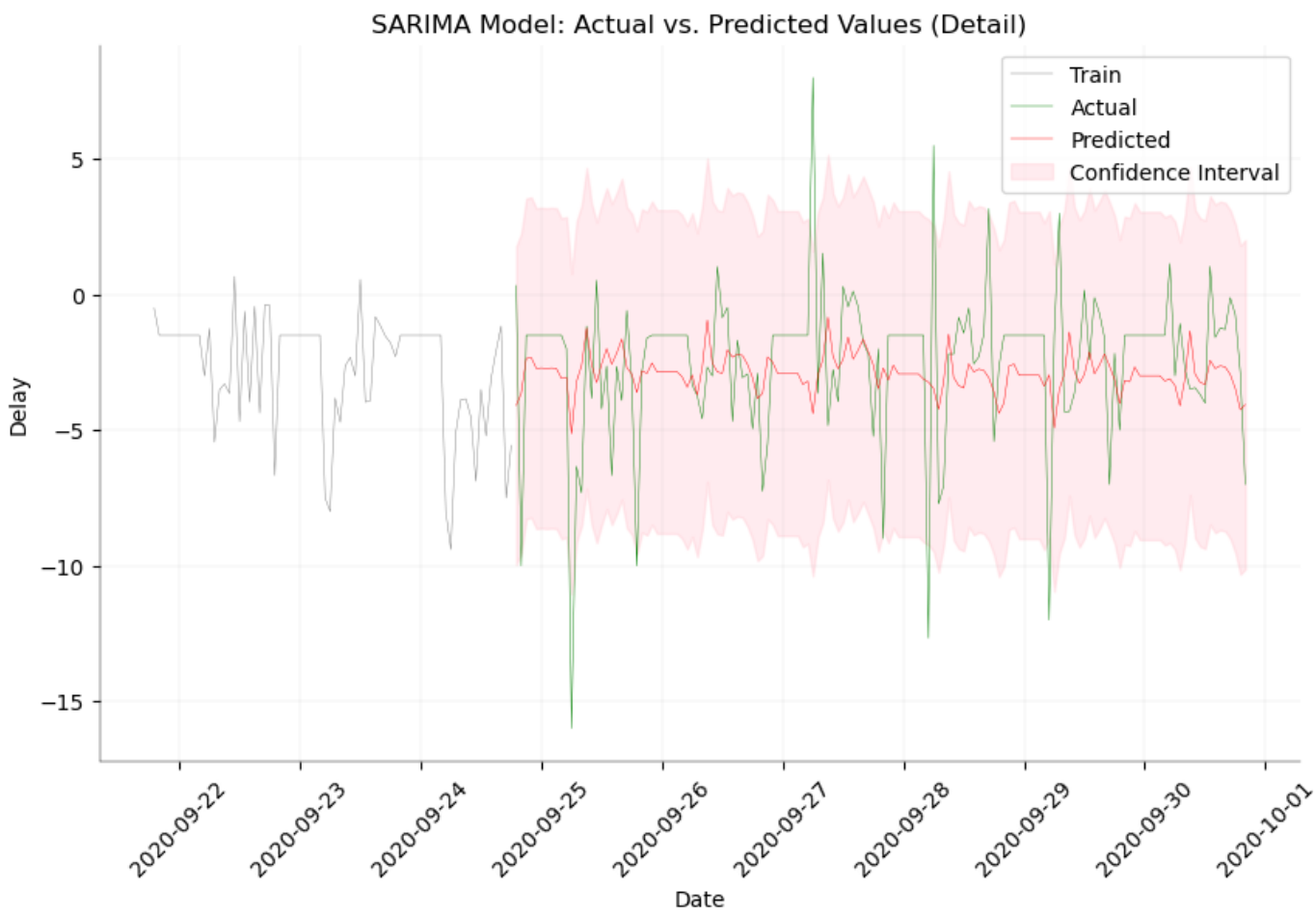
In [767…

```
# Calculate RMSE (Root Mean Squared Error)
rmse = sqrt(mean_squared_error(test['Delay'], sarima_predictions.predicted_mean))
print(f'Root Mean Squared Error (RMSE): {rmse}')

# Plotting the results
plt.figure(figsize=(10, 6))

# Plot training data
plt.plot(train.index, train['Delay'], label='Train', color='gray', linewidth = 0.31)

# Plot actual test data
plt.plot(test.index, test['Delay'], label='Actual', color='green', linewidth = 0.31)

# Plot predicted values
plt.plot(test.index, sarima_predictions.predicted_mean, label='Predicted', color='red',

# Plot confidence intervals
ci = sarima_predictions.conf_int()
plt.fill_between(test.index, ci['lower Delay'], ci['upper Delay'], color='pink', alpha=0

plt.title('SARIMA Model: Actual vs. Predicted Values')
plt.xlabel('Date')
plt.ylabel('Delay')
plt.xticks(rotation=45)
plt.legend()
plt.box(True)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
# Set the width of the axes
plt.gca().spines['left'].set_linewidth(0.31)
plt.gca().spines['bottom'].set_linewidth(0.31)
plt.grid(linewidth = 0.1)
plt.show()

##############
### DETAIL ###
##############

# Plotting the results
plt.figure(figsize=(10, 6))

# Plot training data
plt.plot(train.index[-24*3:], train['Delay'][-24*3:], label='Train', color='gray', linew

# Plot actual test data
plt.plot(test.index, test['Delay'], label='Actual', color='green', linewidth = 0.31)

# Plot predicted values
plt.plot(test.index, sarima_predictions.predicted_mean, label='Predicted', color='red',
```

```
# Plot confidence intervals
ci = sarima_predictions.conf_int()
plt.fill_between(test.index, ci['lower Delay'], ci['upper Delay'], color='pink', alpha=0


plt.title('SARIMA Model: Actual vs. Predicted Values (Detail)')
plt.xlabel('Date')
plt.ylabel('Delay')
plt.xticks(rotation=45)
plt.legend()
plt.box(True)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
# Set the width of the axes
plt.gca().spines['left'].set_linewidth(0.31)
plt.gca().spines['bottom'].set_linewidth(0.31)

plt.grid(linewidth = 0.1)
plt.show()
```

```
Root Mean Squared Error (RMSE): 2.9221514974542764
```





Diagnostic (below)

Dsitribution

- No trend observed
- No deviation from gaussian distribution, except for a narrower peak of the KDE.

Q-Q Plot:

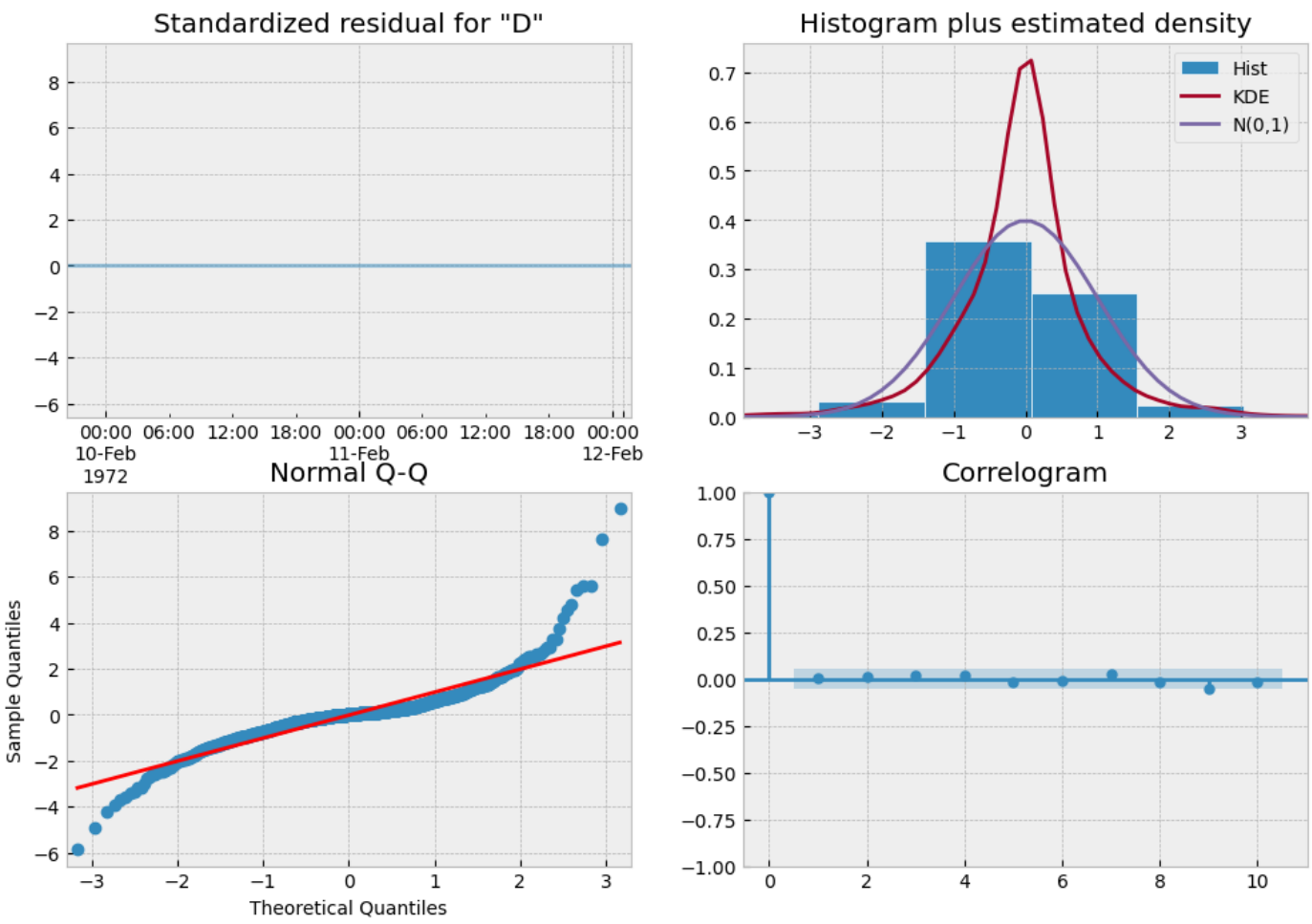
- The small "s" shape around the diagonal line in the Q-Q plot may indicate slight deviations from normality.
  This is not uncommon at the tails of the distribution.

Correlogram:

- No spikes just outside the confidence band in the correlogram
- The model likely won't benefit from changes in the orders

In [819…

```
plt.style.use('bmh') #ggplot')
fitted_sarima_model.plot_diagnostics(figsize=(12, 8))
plt.show()
plt.style.use('default')
```



Results

- The **model captures** global behaviours and **patterns**
- **High values for hourly delay fail to be predicted**, sometimes even trespassing outside of the 95% confidence band.
- **RMSE** is approx **2.9**

Deep Learning with LSTMs

Long **S**hort **T**erm **M**emory Recurrent Neural Networks

Model 1 (Vanilla LSTM)

Dataframe creation. We will have:

- **Target Variable** ->*Average Delay*. Target Variable and time series variable
- **Features**

> 1. *weekday (number from 1 to 7), hour of the day (1 to 24), Rain (precipitaiotn in mm)*
>   We could use categorical features but we retain the numerical form
> 2. **Past values of the target variable**, up to a fixed number (LOOKBACK).
>   We **set it to 24** but alternative values should be considered

Additional considerations:

- 27 features for one target variable. In addition the lstm will be sensitive to sequential patterns
- We handle **null values** (for hours where there are no orders at all) by replacing them with the average hourly delay across the whole series (= *-1.5*)

In [773…
```
import tensorflow.keras as keras
from tensorflow.keras.layers import Input, Bidirectional, LSTM, Dense
import tensorflow as tf

from sklearn.metrics import mean_squared_error
```

In [793…
```
# Define new dataframe, for convenience
time_df = pd.DataFrame({})
time_df["date"] = data["Datetime"]
time_df["minute"] = data["Datetime"].dt.minute
time_df["hour"] = data["Datetime"].dt.hour
time_df["day"] = data["Datetime"].dt.day
time_df["month"] = data["Datetime"].dt.month
time_df["year"] = data["Datetime"].dt.year
time_df["weekday"] = data["Datetime"].dt.weekday


time_df["Delay"] = data["ACTUAL_DELIVERY_MINUTES - ESTIMATED_DELIVERY_MINUTES"]
time_df["Counts"] = len(time_df)*[1]
time_df["Rain"] = data["PRECIPITATION"]

# This is needed for the resampling in the following plots. Set the "date" column as the
time_df_n = time_df
time_df_n.set_index('date', inplace=True)

# Resample by considering hourly sum of delays
resampled_df = time_df_n.resample('H').sum()
#resampled_df = time_df_n.resample('D').mean()

#print(resampled_df['Delay'].dropna().mean())

for column in resampled_df.columns:
    if column != 'Counts':
        # Perform the division only when "Counts" is not 0, otherwise set the result to
        resampled_df[column] = np.where(
            resampled_df["Counts"] != 0,
            resampled_df[column] / resampled_df["Counts"], -1.5) # or 0)
```

In [794…
```
hex_df = resampled_df[['hour','weekday','Rain']]
del_df = resampled_df["Delay"]
```

In [795…
```
# Set the lookback period to 24. This is the number of previous time steps
# to use as input variables to predict the next time period.
LOOKBACK = 24

# Define a function to wrap the data
def wrap_data(df, lookback):
  # Initialize an empty list to store the dataset
  dataset = []

  # Loop over the DataFrame from 'lookback' to the end of the DataFrame
  for index in range(lookback, len(df)+1):
    features = {
        f"col_{i}": float(val) for i, val in enumerate(
```

```
            df.iloc[index-lookback:index].values
        )
    }

    # Convert the dictionary to a DataFrame and append it to the 'dataset' list
    row = pd.DataFrame.from_dict([features])

    # Set the index of the row to be the index of the last period in the lookback window
    row.index = [df.index[index-1]]

    # Append the row to the dataset
    dataset.append(row)

  # Return a DataFrame obtained by concatenating all the DataFrames in the 'dataset' lis
  return pd.concat(dataset, axis=0)
```

In [796…
```
# Obtained features by lagging target variable
dataset = wrap_data(del_df, lookback=LOOKBACK)
# Traget Variable
dataset = dataset.join(del_df.shift(-1))
# Hex. Features
dataset = pd.concat([dataset,hex_df],axis = 1)
```

In [799…
```
dataset = dataset.dropna()
```

In [800…
```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)

def create_model():
    input_layer = Input(shape=(LOOKBACK + len(hex_df.columns), 1))
    recurrent = Bidirectional(LSTM(100, activation="tanh"))(input_layer)
    output_layer = Dense(1)(recurrent)
    model = keras.models.Model(inputs=input_layer, outputs=output_layer)
    model.compile(
        loss='mse', optimizer=keras.optimizers.Adagrad(),
        metrics=[
            keras.metrics.RootMeanSquaredError(),
            keras.metrics.MeanAbsoluteError()
    ])
    return model

model = create_model()
```

In [801…
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    dataset.drop(columns="Delay"),
    dataset["Delay"],
    shuffle=False, test_size = 0.1)
```

In [802…
```
model.fit(X_train, y_train, epochs=1000, callbacks=[callback])
```

```
Epoch 1/1000
41/41 [==============================] - 3s 16ms/step - loss: 10.7438 - root_mean_square
d_error: 3.2778 - mean_absolute_error: 2.2524
Epoch 2/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.6810 - root_mean_squared
_error: 3.1114 - mean_absolute_error: 1.9822
Epoch 3/1000
41/41 [==============================] - 1s 14ms/step - loss: 9.2782 - root_mean_squared
_error: 3.0460 - mean_absolute_error: 1.8392
Epoch 4/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.1602 - root_mean_squared
_error: 3.0266 - mean_absolute_error: 1.8040
Epoch 5/1000
```

```
41/41 [==============================] - 1s 15ms/step - loss: 9.1281 - root_mean_squared
_error: 3.0213 - mean_absolute_error: 1.8053
Epoch 6/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.1179 - root_mean_squared
_error: 3.0196 - mean_absolute_error: 1.8082
Epoch 7/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.1098 - root_mean_squared
_error: 3.0182 - mean_absolute_error: 1.8056
Epoch 8/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.1030 - root_mean_squared
_error: 3.0171 - mean_absolute_error: 1.8041
Epoch 9/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0964 - root_mean_squared
_error: 3.0160 - mean_absolute_error: 1.8019
Epoch 10/1000
41/41 [==============================] - 1s 16ms/step - loss: 9.0915 - root_mean_squared
_error: 3.0152 - mean_absolute_error: 1.8046
Epoch 11/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0844 - root_mean_squared
_error: 3.0140 - mean_absolute_error: 1.7959
Epoch 12/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0816 - root_mean_squared
_error: 3.0136 - mean_absolute_error: 1.8027
Epoch 13/1000
41/41 [==============================] - 1s 14ms/step - loss: 9.0766 - root_mean_squared
_error: 3.0127 - mean_absolute_error: 1.7968
Epoch 14/1000
41/41 [==============================] - 1s 14ms/step - loss: 9.0719 - root_mean_squared
_error: 3.0120 - mean_absolute_error: 1.7925
Epoch 15/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0686 - root_mean_squared
_error: 3.0114 - mean_absolute_error: 1.7939
Epoch 16/1000
41/41 [==============================] - 1s 14ms/step - loss: 9.0641 - root_mean_squared
_error: 3.0107 - mean_absolute_error: 1.7914
Epoch 17/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0588 - root_mean_squared
_error: 3.0098 - mean_absolute_error: 1.7914
Epoch 18/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0564 - root_mean_squared
_error: 3.0094 - mean_absolute_error: 1.7905
Epoch 19/1000
41/41 [==============================] - 1s 15ms/step - loss: 9.0528 - root_mean_squared
_error: 3.0088 - mean_absolute_error: 1.7907
Epoch 20/1000
41/41 [==============================] - 1s 21ms/step - loss: 9.0473 - root_mean_squared
_error: 3.0079 - mean_absolute_error: 1.7892
Epoch 21/1000
41/41 [==============================] - 1s 22ms/step - loss: 9.0450 - root_mean_squared
_error: 3.0075 - mean_absolute_error: 1.7889
Epoch 22/1000
41/41 [==============================] - 1s 24ms/step - loss: 9.0414 - root_mean_squared
_error: 3.0069 - mean_absolute_error: 1.7905
Epoch 23/1000
41/41 [==============================] - 1s 23ms/step - loss: 9.0390 - root_mean_squared
_error: 3.0065 - mean_absolute_error: 1.7881
Epoch 24/1000
41/41 [==============================] - 1s 20ms/step - loss: 9.0367 - root_mean_squared
_error: 3.0061 - mean_absolute_error: 1.7862
Epoch 25/1000
41/41 [==============================] - 1s 22ms/step - loss: 9.0328 - root_mean_squared
_error: 3.0055 - mean_absolute_error: 1.7861
Epoch 26/1000
41/41 [==============================] - 1s 22ms/step - loss: 9.0315 - root_mean_squared
_error: 3.0052 - mean_absolute_error: 1.7851
Epoch 27/1000
```

```
41/41 [==============================] - 1s 21ms/step - loss: 9.0250 - root_mean_squared
_error: 3.0042 - mean_absolute_error: 1.7815
Epoch 28/1000
41/41 [==============================] - 1s 19ms/step - loss: 9.0229 - root_mean_squared
_error: 3.0038 - mean_absolute_error: 1.7843
Epoch 29/1000
41/41 [==============================] - 1s 20ms/step - loss: 9.0201 - root_mean_squared
_error: 3.0034 - mean_absolute_error: 1.7834
Epoch 30/1000
41/41 [==============================] - 1s 22ms/step - loss: 9.0170 - root_mean_squared
_error: 3.0028 - mean_absolute_error: 1.7823
Epoch 31/1000
41/41 [==============================] - 1s 20ms/step - loss: 9.0142 - root_mean_squared
_error: 3.0024 - mean_absolute_error: 1.7845
Epoch 32/1000
41/41 [==============================] - 1s 19ms/step - loss: 9.0133 - root_mean_squared
_error: 3.0022 - mean_absolute_error: 1.7805
Epoch 33/1000
41/41 [==============================] - 1s 20ms/step - loss: 9.0098 - root_mean_squared
_error: 3.0016 - mean_absolute_error: 1.7791
Epoch 34/1000
41/41 [==============================] - 1s 21ms/step - loss: 9.0070 - root_mean_squared
_error: 3.0012 - mean_absolute_error: 1.7797
Epoch 35/1000
41/41 [==============================] - 1s 22ms/step - loss: 9.0053 - root_mean_squared
_error: 3.0009 - mean_absolute_error: 1.7754
Epoch 36/1000
41/41 [==============================] - 1s 21ms/step - loss: 9.0022 - root_mean_squared
_error: 3.0004 - mean_absolute_error: 1.7763
Epoch 37/1000
41/41 [==============================] - 1s 20ms/step - loss: 9.0003 - root_mean_squared
_error: 3.0000 - mean_absolute_error: 1.7750
Epoch 38/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9977 - root_mean_squared
_error: 2.9996 - mean_absolute_error: 1.7782
Epoch 39/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9956 - root_mean_squared
_error: 2.9993 - mean_absolute_error: 1.7723
Epoch 40/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9935 - root_mean_squared
_error: 2.9989 - mean_absolute_error: 1.7757
Epoch 41/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9909 - root_mean_squared
_error: 2.9985 - mean_absolute_error: 1.7761
Epoch 42/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9903 - root_mean_squared
_error: 2.9984 - mean_absolute_error: 1.7746
Epoch 43/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9872 - root_mean_squared
_error: 2.9979 - mean_absolute_error: 1.7760
Epoch 44/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9849 - root_mean_squared
_error: 2.9975 - mean_absolute_error: 1.7731
Epoch 45/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9820 - root_mean_squared
_error: 2.9970 - mean_absolute_error: 1.7740
Epoch 46/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9804 - root_mean_squared
_error: 2.9967 - mean_absolute_error: 1.7715
Epoch 47/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9781 - root_mean_squared
_error: 2.9964 - mean_absolute_error: 1.7721
Epoch 48/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.9761 - root_mean_squared
_error: 2.9960 - mean_absolute_error: 1.7693
Epoch 49/1000
```

```
41/41 [==============================] - 1s 21ms/step - loss: 8.9750 - root_mean_squared
_error: 2.9958 - mean_absolute_error: 1.7700
Epoch 50/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.9722 - root_mean_squared
_error: 2.9954 - mean_absolute_error: 1.7688
Epoch 51/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.9699 - root_mean_squared
_error: 2.9950 - mean_absolute_error: 1.7695
Epoch 52/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9674 - root_mean_squared
_error: 2.9946 - mean_absolute_error: 1.7684
Epoch 53/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.9676 - root_mean_squared
_error: 2.9946 - mean_absolute_error: 1.7684
Epoch 54/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9641 - root_mean_squared
_error: 2.9940 - mean_absolute_error: 1.7656
Epoch 55/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9628 - root_mean_squared
_error: 2.9938 - mean_absolute_error: 1.7650
Epoch 56/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9609 - root_mean_squared
_error: 2.9935 - mean_absolute_error: 1.7644
Epoch 57/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9579 - root_mean_squared
_error: 2.9930 - mean_absolute_error: 1.7651
Epoch 58/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9572 - root_mean_squared
_error: 2.9929 - mean_absolute_error: 1.7626
Epoch 59/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9538 - root_mean_squared
_error: 2.9923 - mean_absolute_error: 1.7636
Epoch 60/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9532 - root_mean_squared
_error: 2.9922 - mean_absolute_error: 1.7637
Epoch 61/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9510 - root_mean_squared
_error: 2.9918 - mean_absolute_error: 1.7644
Epoch 62/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.9508 - root_mean_squared
_error: 2.9918 - mean_absolute_error: 1.7590
Epoch 63/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9487 - root_mean_squared
_error: 2.9914 - mean_absolute_error: 1.7578
Epoch 64/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.9473 - root_mean_squared
_error: 2.9912 - mean_absolute_error: 1.7583
Epoch 65/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9459 - root_mean_squared
_error: 2.9910 - mean_absolute_error: 1.7583
Epoch 66/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9448 - root_mean_squared
_error: 2.9908 - mean_absolute_error: 1.7607
Epoch 67/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9433 - root_mean_squared
_error: 2.9905 - mean_absolute_error: 1.7575
Epoch 68/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9416 - root_mean_squared
_error: 2.9902 - mean_absolute_error: 1.7580
Epoch 69/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9394 - root_mean_squared
_error: 2.9899 - mean_absolute_error: 1.7578
Epoch 70/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9383 - root_mean_squared
_error: 2.9897 - mean_absolute_error: 1.7586
Epoch 71/1000
```

```
41/41 [==============================] - 1s 23ms/step - loss: 8.9364 - root_mean_squared
_error: 2.9894 - mean_absolute_error: 1.7571
Epoch 72/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9359 - root_mean_squared
_error: 2.9893 - mean_absolute_error: 1.7571
Epoch 73/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9328 - root_mean_squared
_error: 2.9888 - mean_absolute_error: 1.7576
Epoch 74/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9316 - root_mean_squared
_error: 2.9886 - mean_absolute_error: 1.7553
Epoch 75/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9308 - root_mean_squared
_error: 2.9884 - mean_absolute_error: 1.7566
Epoch 76/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9293 - root_mean_squared
_error: 2.9882 - mean_absolute_error: 1.7524
Epoch 77/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9287 - root_mean_squared
_error: 2.9881 - mean_absolute_error: 1.7551
Epoch 78/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9257 - root_mean_squared
_error: 2.9876 - mean_absolute_error: 1.7550
Epoch 79/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9248 - root_mean_squared
_error: 2.9874 - mean_absolute_error: 1.7555
Epoch 80/1000
41/41 [==============================] - 1s 24ms/step - loss: 8.9232 - root_mean_squared
_error: 2.9872 - mean_absolute_error: 1.7531
Epoch 81/1000
41/41 [==============================] - 1s 26ms/step - loss: 8.9218 - root_mean_squared
_error: 2.9869 - mean_absolute_error: 1.7518
Epoch 82/1000
41/41 [==============================] - 1s 25ms/step - loss: 8.9211 - root_mean_squared
_error: 2.9868 - mean_absolute_error: 1.7549
Epoch 83/1000
41/41 [==============================] - 1s 25ms/step - loss: 8.9186 - root_mean_squared
_error: 2.9864 - mean_absolute_error: 1.7535
Epoch 84/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9179 - root_mean_squared
_error: 2.9863 - mean_absolute_error: 1.7514
Epoch 85/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9159 - root_mean_squared
_error: 2.9860 - mean_absolute_error: 1.7496
Epoch 86/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9147 - root_mean_squared
_error: 2.9858 - mean_absolute_error: 1.7515
Epoch 87/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9140 - root_mean_squared
_error: 2.9856 - mean_absolute_error: 1.7476
Epoch 88/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9132 - root_mean_squared
_error: 2.9855 - mean_absolute_error: 1.7524
Epoch 89/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9112 - root_mean_squared
_error: 2.9852 - mean_absolute_error: 1.7469
Epoch 90/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9103 - root_mean_squared
_error: 2.9850 - mean_absolute_error: 1.7510
Epoch 91/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9080 - root_mean_squared
_error: 2.9846 - mean_absolute_error: 1.7535
Epoch 92/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9092 - root_mean_squared
_error: 2.9848 - mean_absolute_error: 1.7519
Epoch 93/1000
```

```
41/41 [==============================] - 1s 20ms/step - loss: 8.9064 - root_mean_squared
_error: 2.9844 - mean_absolute_error: 1.7518
Epoch 94/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9060 - root_mean_squared
_error: 2.9843 - mean_absolute_error: 1.7474
Epoch 95/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.9035 - root_mean_squared
_error: 2.9839 - mean_absolute_error: 1.7492
Epoch 96/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9032 - root_mean_squared
_error: 2.9838 - mean_absolute_error: 1.7464
Epoch 97/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.9009 - root_mean_squared
_error: 2.9834 - mean_absolute_error: 1.7494
Epoch 98/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.8996 - root_mean_squared
_error: 2.9832 - mean_absolute_error: 1.7499
Epoch 99/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8987 - root_mean_squared
_error: 2.9831 - mean_absolute_error: 1.7458
Epoch 100/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8974 - root_mean_squared
_error: 2.9828 - mean_absolute_error: 1.7469
Epoch 101/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8971 - root_mean_squared
_error: 2.9828 - mean_absolute_error: 1.7478
Epoch 102/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8955 - root_mean_squared
_error: 2.9825 - mean_absolute_error: 1.7461
Epoch 103/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8942 - root_mean_squared
_error: 2.9823 - mean_absolute_error: 1.7461
Epoch 104/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8929 - root_mean_squared
_error: 2.9821 - mean_absolute_error: 1.7459
Epoch 105/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8926 - root_mean_squared
_error: 2.9821 - mean_absolute_error: 1.7470
Epoch 106/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8929 - root_mean_squared
_error: 2.9821 - mean_absolute_error: 1.7494
Epoch 107/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8894 - root_mean_squared
_error: 2.9815 - mean_absolute_error: 1.7476
Epoch 108/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8888 - root_mean_squared
_error: 2.9814 - mean_absolute_error: 1.7442
Epoch 109/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8885 - root_mean_squared
_error: 2.9814 - mean_absolute_error: 1.7451
Epoch 110/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8863 - root_mean_squared
_error: 2.9810 - mean_absolute_error: 1.7493
Epoch 111/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8866 - root_mean_squared
_error: 2.9810 - mean_absolute_error: 1.7475
Epoch 112/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8841 - root_mean_squared
_error: 2.9806 - mean_absolute_error: 1.7494
Epoch 113/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8829 - root_mean_squared
_error: 2.9804 - mean_absolute_error: 1.7473
Epoch 114/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8829 - root_mean_squared
_error: 2.9804 - mean_absolute_error: 1.7415
Epoch 115/1000
```

```
41/41 [==============================] - 1s 21ms/step - loss: 8.8819 - root_mean_squared
_error: 2.9802 - mean_absolute_error: 1.7408
Epoch 116/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8809 - root_mean_squared
_error: 2.9801 - mean_absolute_error: 1.7429
Epoch 117/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8796 - root_mean_squared
_error: 2.9799 - mean_absolute_error: 1.7457
Epoch 118/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8786 - root_mean_squared
_error: 2.9797 - mean_absolute_error: 1.7437
Epoch 119/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8766 - root_mean_squared
_error: 2.9794 - mean_absolute_error: 1.7454
Epoch 120/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8758 - root_mean_squared
_error: 2.9792 - mean_absolute_error: 1.7423
Epoch 121/1000
41/41 [==============================] - 1s 27ms/step - loss: 8.8759 - root_mean_squared
_error: 2.9792 - mean_absolute_error: 1.7423
Epoch 122/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.8742 - root_mean_squared
_error: 2.9790 - mean_absolute_error: 1.7393
Epoch 123/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8735 - root_mean_squared
_error: 2.9788 - mean_absolute_error: 1.7441
Epoch 124/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8724 - root_mean_squared
_error: 2.9787 - mean_absolute_error: 1.7435
Epoch 125/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8715 - root_mean_squared
_error: 2.9785 - mean_absolute_error: 1.7415
Epoch 126/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8711 - root_mean_squared
_error: 2.9784 - mean_absolute_error: 1.7395
Epoch 127/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8697 - root_mean_squared
_error: 2.9782 - mean_absolute_error: 1.7415
Epoch 128/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8691 - root_mean_squared
_error: 2.9781 - mean_absolute_error: 1.7419
Epoch 129/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8678 - root_mean_squared
_error: 2.9779 - mean_absolute_error: 1.7383
Epoch 130/1000
41/41 [==============================] - 1s 25ms/step - loss: 8.8665 - root_mean_squared
_error: 2.9777 - mean_absolute_error: 1.7403
Epoch 131/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8677 - root_mean_squared
_error: 2.9779 - mean_absolute_error: 1.7376
Epoch 132/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8653 - root_mean_squared
_error: 2.9775 - mean_absolute_error: 1.7366
Epoch 133/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8644 - root_mean_squared
_error: 2.9773 - mean_absolute_error: 1.7400
Epoch 134/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8635 - root_mean_squared
_error: 2.9772 - mean_absolute_error: 1.7377
Epoch 135/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8623 - root_mean_squared
_error: 2.9770 - mean_absolute_error: 1.7380
Epoch 136/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8623 - root_mean_squared
_error: 2.9770 - mean_absolute_error: 1.7422
Epoch 137/1000
```

```
41/41 [==============================] - 1s 23ms/step - loss: 8.8605 - root_mean_squared
_error: 2.9767 - mean_absolute_error: 1.7378
Epoch 138/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.8595 - root_mean_squared
_error: 2.9765 - mean_absolute_error: 1.7421
Epoch 139/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8579 - root_mean_squared
_error: 2.9762 - mean_absolute_error: 1.7401
Epoch 140/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8584 - root_mean_squared
_error: 2.9763 - mean_absolute_error: 1.7426
Epoch 141/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8564 - root_mean_squared
_error: 2.9760 - mean_absolute_error: 1.7393
Epoch 142/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8557 - root_mean_squared
_error: 2.9759 - mean_absolute_error: 1.7393
Epoch 143/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8554 - root_mean_squared
_error: 2.9758 - mean_absolute_error: 1.7365
Epoch 144/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8551 - root_mean_squared
_error: 2.9757 - mean_absolute_error: 1.7397
Epoch 145/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8537 - root_mean_squared
_error: 2.9755 - mean_absolute_error: 1.7386
Epoch 146/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8539 - root_mean_squared
_error: 2.9755 - mean_absolute_error: 1.7357
Epoch 147/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8529 - root_mean_squared
_error: 2.9754 - mean_absolute_error: 1.7368
Epoch 148/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8504 - root_mean_squared
_error: 2.9750 - mean_absolute_error: 1.7353
Epoch 149/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8506 - root_mean_squared
_error: 2.9750 - mean_absolute_error: 1.7359
Epoch 150/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8490 - root_mean_squared
_error: 2.9747 - mean_absolute_error: 1.7383
Epoch 151/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8487 - root_mean_squared
_error: 2.9747 - mean_absolute_error: 1.7405
Epoch 152/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8478 - root_mean_squared
_error: 2.9745 - mean_absolute_error: 1.7342
Epoch 153/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8473 - root_mean_squared
_error: 2.9744 - mean_absolute_error: 1.7369
Epoch 154/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8469 - root_mean_squared
_error: 2.9744 - mean_absolute_error: 1.7349
Epoch 155/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8456 - root_mean_squared
_error: 2.9742 - mean_absolute_error: 1.7390
Epoch 156/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8449 - root_mean_squared
_error: 2.9740 - mean_absolute_error: 1.7385
Epoch 157/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8446 - root_mean_squared
_error: 2.9740 - mean_absolute_error: 1.7326
Epoch 158/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8432 - root_mean_squared
_error: 2.9737 - mean_absolute_error: 1.7322
Epoch 159/1000
```

```
41/41 [==============================] - 1s 20ms/step - loss: 8.8424 - root_mean_squared
_error: 2.9736 - mean_absolute_error: 1.7340
Epoch 160/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8420 - root_mean_squared
_error: 2.9736 - mean_absolute_error: 1.7335
Epoch 161/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8419 - root_mean_squared
_error: 2.9735 - mean_absolute_error: 1.7315
Epoch 162/1000
41/41 [==============================] - 1s 24ms/step - loss: 8.8399 - root_mean_squared
_error: 2.9732 - mean_absolute_error: 1.7320
Epoch 163/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8382 - root_mean_squared
_error: 2.9729 - mean_absolute_error: 1.7350
Epoch 164/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8379 - root_mean_squared
_error: 2.9729 - mean_absolute_error: 1.7332
Epoch 165/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8370 - root_mean_squared
_error: 2.9727 - mean_absolute_error: 1.7382
Epoch 166/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8371 - root_mean_squared
_error: 2.9727 - mean_absolute_error: 1.7348
Epoch 167/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8368 - root_mean_squared
_error: 2.9727 - mean_absolute_error: 1.7355
Epoch 168/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8349 - root_mean_squared
_error: 2.9724 - mean_absolute_error: 1.7360
Epoch 169/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8341 - root_mean_squared
_error: 2.9722 - mean_absolute_error: 1.7362
Epoch 170/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8334 - root_mean_squared
_error: 2.9721 - mean_absolute_error: 1.7359
Epoch 171/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8324 - root_mean_squared
_error: 2.9719 - mean_absolute_error: 1.7381
Epoch 172/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8321 - root_mean_squared
_error: 2.9719 - mean_absolute_error: 1.7319
Epoch 173/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8310 - root_mean_squared
_error: 2.9717 - mean_absolute_error: 1.7355
Epoch 174/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8317 - root_mean_squared
_error: 2.9718 - mean_absolute_error: 1.7357
Epoch 175/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8315 - root_mean_squared
_error: 2.9718 - mean_absolute_error: 1.7383
Epoch 176/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8297 - root_mean_squared
_error: 2.9715 - mean_absolute_error: 1.7351
Epoch 177/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8289 - root_mean_squared
_error: 2.9713 - mean_absolute_error: 1.7364
Epoch 178/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8281 - root_mean_squared
_error: 2.9712 - mean_absolute_error: 1.7341
Epoch 179/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8267 - root_mean_squared
_error: 2.9710 - mean_absolute_error: 1.7317
Epoch 180/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8257 - root_mean_squared
_error: 2.9708 - mean_absolute_error: 1.7360
Epoch 181/1000
```

```
41/41 [==============================] - 1s 20ms/step - loss: 8.8276 - root_mean_squared
_error: 2.9711 - mean_absolute_error: 1.7354
Epoch 182/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8248 - root_mean_squared
_error: 2.9707 - mean_absolute_error: 1.7343
Epoch 183/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8250 - root_mean_squared
_error: 2.9707 - mean_absolute_error: 1.7337
Epoch 184/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8244 - root_mean_squared
_error: 2.9706 - mean_absolute_error: 1.7381
Epoch 185/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8223 - root_mean_squared
_error: 2.9702 - mean_absolute_error: 1.7342
Epoch 186/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8228 - root_mean_squared
_error: 2.9703 - mean_absolute_error: 1.7300
Epoch 187/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8214 - root_mean_squared
_error: 2.9701 - mean_absolute_error: 1.7331
Epoch 188/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8212 - root_mean_squared
_error: 2.9701 - mean_absolute_error: 1.7337
Epoch 189/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8206 - root_mean_squared
_error: 2.9699 - mean_absolute_error: 1.7319
Epoch 190/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8195 - root_mean_squared
_error: 2.9698 - mean_absolute_error: 1.7337
Epoch 191/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8188 - root_mean_squared
_error: 2.9697 - mean_absolute_error: 1.7329
Epoch 192/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8181 - root_mean_squared
_error: 2.9695 - mean_absolute_error: 1.7310
Epoch 193/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8175 - root_mean_squared
_error: 2.9694 - mean_absolute_error: 1.7272
Epoch 194/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8169 - root_mean_squared
_error: 2.9693 - mean_absolute_error: 1.7292
Epoch 195/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8164 - root_mean_squared
_error: 2.9692 - mean_absolute_error: 1.7296
Epoch 196/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8155 - root_mean_squared
_error: 2.9691 - mean_absolute_error: 1.7332
Epoch 197/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8154 - root_mean_squared
_error: 2.9691 - mean_absolute_error: 1.7319
Epoch 198/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8134 - root_mean_squared
_error: 2.9687 - mean_absolute_error: 1.7324
Epoch 199/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8143 - root_mean_squared
_error: 2.9689 - mean_absolute_error: 1.7315
Epoch 200/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8134 - root_mean_squared
_error: 2.9687 - mean_absolute_error: 1.7313
Epoch 201/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.8126 - root_mean_squared
_error: 2.9686 - mean_absolute_error: 1.7280
Epoch 202/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8109 - root_mean_squared
_error: 2.9683 - mean_absolute_error: 1.7302
Epoch 203/1000
```

```
41/41 [==============================] - 1s 20ms/step - loss: 8.8112 - root_mean_squared
_error: 2.9684 - mean_absolute_error: 1.7291
Epoch 204/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8106 - root_mean_squared
_error: 2.9683 - mean_absolute_error: 1.7312
Epoch 205/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8104 - root_mean_squared
_error: 2.9682 - mean_absolute_error: 1.7311
Epoch 206/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8097 - root_mean_squared
_error: 2.9681 - mean_absolute_error: 1.7290
Epoch 207/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8088 - root_mean_squared
_error: 2.9680 - mean_absolute_error: 1.7342
Epoch 208/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8079 - root_mean_squared
_error: 2.9678 - mean_absolute_error: 1.7308
Epoch 209/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8078 - root_mean_squared
_error: 2.9678 - mean_absolute_error: 1.7279
Epoch 210/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8064 - root_mean_squared
_error: 2.9676 - mean_absolute_error: 1.7320
Epoch 211/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8065 - root_mean_squared
_error: 2.9676 - mean_absolute_error: 1.7249
Epoch 212/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8050 - root_mean_squared
_error: 2.9673 - mean_absolute_error: 1.7276
Epoch 213/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8048 - root_mean_squared
_error: 2.9673 - mean_absolute_error: 1.7282
Epoch 214/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8040 - root_mean_squared
_error: 2.9672 - mean_absolute_error: 1.7298
Epoch 215/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.8034 - root_mean_squared
_error: 2.9670 - mean_absolute_error: 1.7278
Epoch 216/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8039 - root_mean_squared
_error: 2.9671 - mean_absolute_error: 1.7276
Epoch 217/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8027 - root_mean_squared
_error: 2.9669 - mean_absolute_error: 1.7307
Epoch 218/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8026 - root_mean_squared
_error: 2.9669 - mean_absolute_error: 1.7309
Epoch 219/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8012 - root_mean_squared
_error: 2.9667 - mean_absolute_error: 1.7271
Epoch 220/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.8009 - root_mean_squared
_error: 2.9666 - mean_absolute_error: 1.7286
Epoch 221/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7997 - root_mean_squared
_error: 2.9664 - mean_absolute_error: 1.7265
Epoch 222/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7992 - root_mean_squared
_error: 2.9663 - mean_absolute_error: 1.7258
Epoch 223/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7991 - root_mean_squared
_error: 2.9663 - mean_absolute_error: 1.7266
Epoch 224/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7995 - root_mean_squared
_error: 2.9664 - mean_absolute_error: 1.7277
Epoch 225/1000
```

```
41/41 [==============================] - 1s 20ms/step - loss: 8.7983 - root_mean_squared
_error: 2.9662 - mean_absolute_error: 1.7262
Epoch 226/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7971 - root_mean_squared
_error: 2.9660 - mean_absolute_error: 1.7242
Epoch 227/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7970 - root_mean_squared
_error: 2.9660 - mean_absolute_error: 1.7270
Epoch 228/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7959 - root_mean_squared
_error: 2.9658 - mean_absolute_error: 1.7290
Epoch 229/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7976 - root_mean_squared
_error: 2.9661 - mean_absolute_error: 1.7299
Epoch 230/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7950 - root_mean_squared
_error: 2.9656 - mean_absolute_error: 1.7266
Epoch 231/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7948 - root_mean_squared
_error: 2.9656 - mean_absolute_error: 1.7270
Epoch 232/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7931 - root_mean_squared
_error: 2.9653 - mean_absolute_error: 1.7239
Epoch 233/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7926 - root_mean_squared
_error: 2.9652 - mean_absolute_error: 1.7323
Epoch 234/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7924 - root_mean_squared
_error: 2.9652 - mean_absolute_error: 1.7294
Epoch 235/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7918 - root_mean_squared
_error: 2.9651 - mean_absolute_error: 1.7295
Epoch 236/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7923 - root_mean_squared
_error: 2.9652 - mean_absolute_error: 1.7316
Epoch 237/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7917 - root_mean_squared
_error: 2.9651 - mean_absolute_error: 1.7278
Epoch 238/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7903 - root_mean_squared
_error: 2.9649 - mean_absolute_error: 1.7293
Epoch 239/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7900 - root_mean_squared
_error: 2.9648 - mean_absolute_error: 1.7276
Epoch 240/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7884 - root_mean_squared
_error: 2.9645 - mean_absolute_error: 1.7296
Epoch 241/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7882 - root_mean_squared
_error: 2.9645 - mean_absolute_error: 1.7302
Epoch 242/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7885 - root_mean_squared
_error: 2.9645 - mean_absolute_error: 1.7300
Epoch 243/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7869 - root_mean_squared
_error: 2.9643 - mean_absolute_error: 1.7246
Epoch 244/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7861 - root_mean_squared
_error: 2.9641 - mean_absolute_error: 1.7243
Epoch 245/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7862 - root_mean_squared
_error: 2.9642 - mean_absolute_error: 1.7280
Epoch 246/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7857 - root_mean_squared
_error: 2.9641 - mean_absolute_error: 1.7243
Epoch 247/1000
```

```
41/41 [==============================] - 1s 19ms/step - loss: 8.7859 - root_mean_squared
_error: 2.9641 - mean_absolute_error: 1.7278
Epoch 248/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7850 - root_mean_squared
_error: 2.9640 - mean_absolute_error: 1.7252
Epoch 249/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7845 - root_mean_squared
_error: 2.9639 - mean_absolute_error: 1.7258
Epoch 250/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7834 - root_mean_squared
_error: 2.9637 - mean_absolute_error: 1.7296
Epoch 251/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7829 - root_mean_squared
_error: 2.9636 - mean_absolute_error: 1.7251
Epoch 252/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7821 - root_mean_squared
_error: 2.9635 - mean_absolute_error: 1.7239
Epoch 253/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7819 - root_mean_squared
_error: 2.9634 - mean_absolute_error: 1.7259
Epoch 254/1000
41/41 [==============================] - 1s 19ms/step - loss: 8.7813 - root_mean_squared
_error: 2.9633 - mean_absolute_error: 1.7276
Epoch 255/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7805 - root_mean_squared
_error: 2.9632 - mean_absolute_error: 1.7289
Epoch 256/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7805 - root_mean_squared
_error: 2.9632 - mean_absolute_error: 1.7235
Epoch 257/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7807 - root_mean_squared
_error: 2.9632 - mean_absolute_error: 1.7239
Epoch 258/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7805 - root_mean_squared
_error: 2.9632 - mean_absolute_error: 1.7220
```

Out[802]:
```
<keras.src.callbacks.History at 0x2b28d615300>
```

In [825…

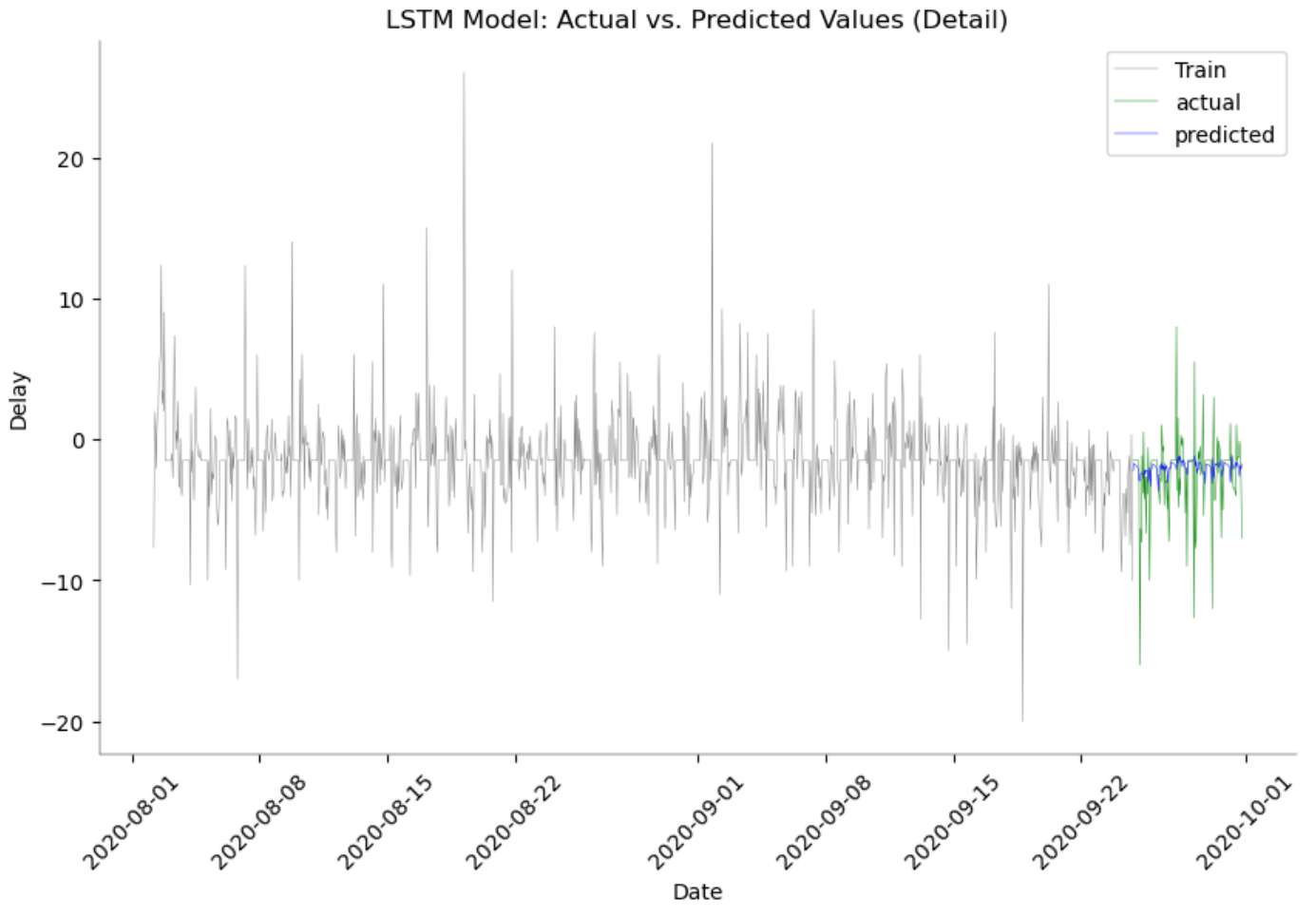
```python
def show_result(y_test, predicted):
    plt.figure(figsize=(10, 6))
    plt.plot(y_train.index, y_train, '-', label="Train",  color='gray', linewidth = 0.31
    plt.plot(y_test.index, y_test, '-', label="actual", color='green', linewidth = 0.31)
    plt.plot(y_test.index, predicted, '-', label="predicted", color='blue', linewidth =

    plt.title('LSTM Model: Actual vs. Predicted Values (Detail)')
    plt.xlabel('Date')
    plt.ylabel('Delay')
    plt.legend()
    plt.xticks(rotation=45)
    plt.box(True)
    plt.gca().spines['right'].set_visible(False)
    plt.gca().spines['top'].set_visible(False)
    # Set the width of the axes
    plt.gca().spines['left'].set_linewidth(0.31)
    plt.gca().spines['bottom'].set_linewidth(0.31)

    plt.grid(linewidth = 0.1)
    plt.show()

def show_result_detail(y_test, predicted):
    # Plotting the results
    plt.figure(figsize=(10, 6))

    # Plot training data
    #plt.plot(X_train.index[-24*3:], train['Delay'][-24*3:], label='Train', color='gray'
```

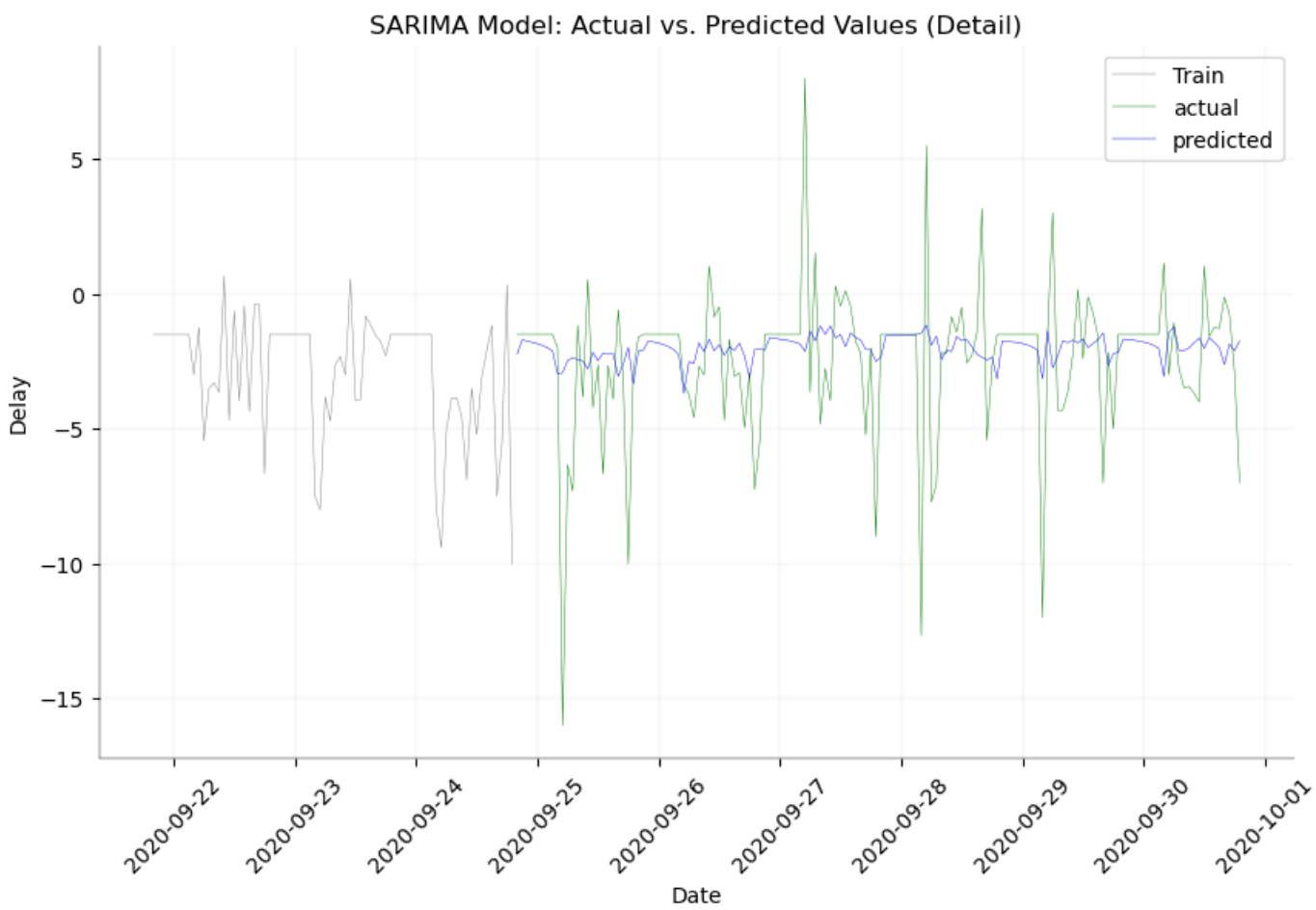
```
    plt.plot(y_train.index[-24*3:], y_train[-24*3:], '-', label="Train",  color='gray',
    plt.plot(y_test.index, y_test, '-', label="actual", color='green', linewidth = 0.31)
    plt.plot(y_test.index, predicted, '-', label="predicted", color='blue', linewidth =

    # Plot confidence intervals
    #ci = sarima_predictions.conf_int()
    #plt.fill_between(test.index, ci['lower Delay'], ci['upper Delay'], color='pink', al


    plt.title('LSTM Model: Actual vs. Predicted Values (Detail)')
    plt.xlabel('Date')
    plt.ylabel('Delay')
    plt.xticks(rotation=45)
    plt.legend()
    plt.box(True)
    plt.gca().spines['right'].set_visible(False)
    plt.gca().spines['top'].set_visible(False)
    # Set the width of the axes
    plt.gca().spines['left'].set_linewidth(0.31)
    plt.gca().spines['bottom'].set_linewidth(0.31)

    plt.grid(linewidth = 0.1)
    plt.show()

    results_df = pd.DataFrame({'test':y_test,'preds':list(predicted)})
    mse = mean_squared_error(results_df.dropna()['test'], results_df.dropna()['preds'])
    rmse = mse**(1/2)
    print(f'Test RMSE: {rmse}')
```

In [816…

```
predicted = model.predict(X_test)
show_result(y_test, predicted)
show_result_detail(y_test, predicted)
```

```
5/5 [==============================] - 0s 6ms/step
```






```
Test RMSE: 2.8147900311469574
```

Results

- The **model captures** global behaviours and **patterns**, but not better than SARIMAX.
- **High values for hourly delay fail to be predicted**. The predicted values tend to lay around the mean.
- **RMSE** is approx **2.8**

Further Developements

- Set **different LOOKBACK** values
- **Upgrade LSTM architecture, add layers and parameters** for richer non-linear models.
- Optimize training

Model 2 (Multiple Layers LSTM)

- Adding layers to **Model 1**

In [820…

```
def create_complex_model():
    input_layer = Input(shape=(LOOKBACK + len(hex_df.columns), 1))

    # Add multiple LSTM layers with dropout
    lstm1 = Bidirectional(LSTM(100, activation="tanh", return_sequences=True))(input_lay
    lstm1 = Dropout(0.2)(lstm1)

    lstm2 = Bidirectional(LSTM(50, activation="tanh", return_sequences=True))(lstm1)
```

```
    lstm2 = Dropout(0.2)(lstm2)

    lstm3 = Bidirectional(LSTM(25, activation="tanh"))(lstm2)
    lstm3 = Dropout(0.2)(lstm3)

    # Concatenate the output of the last LSTM layer with the input for skip connections
    #concat_layer = Concatenate()([lstm3, input_layer])

    # Dense layers for final prediction
    dense1 = Dense(50, activation='relu')(lstm3)
    #dense1 = Dense(50, activation='relu')(concat_layer)
    dense1 = Dropout(0.2)(dense1)

    dense2 = Dense(25, activation='relu')(dense1)
    dense2 = Dropout(0.2)(dense2)

    output_layer = Dense(1)(dense2)

    model = keras.models.Model(inputs=input_layer, outputs=output_layer)

    model.compile(
        loss='mse',
        optimizer=keras.optimizers.Adam(),
        metrics=[
            keras.metrics.RootMeanSquaredError(),
            keras.metrics.MeanAbsoluteError()
        ]
    )

    return model

model.fit(X_train, y_train, epochs=1000, callbacks=[callback])
```

```
Epoch 1/1000
41/41 [==============================] - 1s 15ms/step - loss: 8.7798 - root_mean_squared
_error: 2.9631 - mean_absolute_error: 1.7224
Epoch 2/1000
41/41 [==============================] - 1s 15ms/step - loss: 8.7784 - root_mean_squared
_error: 2.9628 - mean_absolute_error: 1.7238
Epoch 3/1000
41/41 [==============================] - 1s 15ms/step - loss: 8.7787 - root_mean_squared
_error: 2.9629 - mean_absolute_error: 1.7257
Epoch 4/1000
41/41 [==============================] - 1s 15ms/step - loss: 8.7776 - root_mean_squared
_error: 2.9627 - mean_absolute_error: 1.7268
Epoch 5/1000
41/41 [==============================] - 1s 16ms/step - loss: 8.7783 - root_mean_squared
_error: 2.9628 - mean_absolute_error: 1.7249
Epoch 6/1000
41/41 [==============================] - 1s 15ms/step - loss: 8.7778 - root_mean_squared
_error: 2.9627 - mean_absolute_error: 1.7271
Epoch 7/1000
41/41 [==============================] - 1s 14ms/step - loss: 8.7757 - root_mean_squared
_error: 2.9624 - mean_absolute_error: 1.7229
Epoch 8/1000
41/41 [==============================] - 1s 16ms/step - loss: 8.7759 - root_mean_squared
_error: 2.9624 - mean_absolute_error: 1.7238
Epoch 9/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7762 - root_mean_squared
_error: 2.9625 - mean_absolute_error: 1.7264
Epoch 10/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7736 - root_mean_squared
_error: 2.9620 - mean_absolute_error: 1.7204
Epoch 11/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7733 - root_mean_squared
_error: 2.9620 - mean_absolute_error: 1.7220
```

```
Epoch 12/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7733 - root_mean_squared
_error: 2.9620 - mean_absolute_error: 1.7259
Epoch 13/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7734 - root_mean_squared
_error: 2.9620 - mean_absolute_error: 1.7270
Epoch 14/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7727 - root_mean_squared
_error: 2.9619 - mean_absolute_error: 1.7230
Epoch 15/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7716 - root_mean_squared
_error: 2.9617 - mean_absolute_error: 1.7251
Epoch 16/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7702 - root_mean_squared
_error: 2.9615 - mean_absolute_error: 1.7291
Epoch 17/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7697 - root_mean_squared
_error: 2.9614 - mean_absolute_error: 1.7266
Epoch 18/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7696 - root_mean_squared
_error: 2.9614 - mean_absolute_error: 1.7219
Epoch 19/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7691 - root_mean_squared
_error: 2.9613 - mean_absolute_error: 1.7240
Epoch 20/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7692 - root_mean_squared
_error: 2.9613 - mean_absolute_error: 1.7205
Epoch 21/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7683 - root_mean_squared
_error: 2.9611 - mean_absolute_error: 1.7222
Epoch 22/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7677 - root_mean_squared
_error: 2.9610 - mean_absolute_error: 1.7203
Epoch 23/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7675 - root_mean_squared
_error: 2.9610 - mean_absolute_error: 1.7207
Epoch 24/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7673 - root_mean_squared
_error: 2.9610 - mean_absolute_error: 1.7236
Epoch 25/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7664 - root_mean_squared
_error: 2.9608 - mean_absolute_error: 1.7233
Epoch 26/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7653 - root_mean_squared
_error: 2.9606 - mean_absolute_error: 1.7225
Epoch 27/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7661 - root_mean_squared
_error: 2.9608 - mean_absolute_error: 1.7265
Epoch 28/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7647 - root_mean_squared
_error: 2.9605 - mean_absolute_error: 1.7254
Epoch 29/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7646 - root_mean_squared
_error: 2.9605 - mean_absolute_error: 1.7255
Epoch 30/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.7639 - root_mean_squared
_error: 2.9604 - mean_absolute_error: 1.7193
Epoch 31/1000
41/41 [==============================] - 1s 24ms/step - loss: 8.7633 - root_mean_squared
_error: 2.9603 - mean_absolute_error: 1.7220
Epoch 32/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7629 - root_mean_squared
_error: 2.9602 - mean_absolute_error: 1.7213
Epoch 33/1000
41/41 [==============================] - 1s 26ms/step - loss: 8.7634 - root_mean_squared
_error: 2.9603 - mean_absolute_error: 1.7240
```

```
Epoch 34/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7620 - root_mean_squared
_error: 2.9601 - mean_absolute_error: 1.7220
Epoch 35/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7616 - root_mean_squared
_error: 2.9600 - mean_absolute_error: 1.7246
Epoch 36/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7619 - root_mean_squared
_error: 2.9601 - mean_absolute_error: 1.7227
Epoch 37/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7604 - root_mean_squared
_error: 2.9598 - mean_absolute_error: 1.7222
Epoch 38/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7608 - root_mean_squared
_error: 2.9599 - mean_absolute_error: 1.7229
Epoch 39/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7595 - root_mean_squared
_error: 2.9596 - mean_absolute_error: 1.7239
Epoch 40/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7596 - root_mean_squared
_error: 2.9597 - mean_absolute_error: 1.7261
Epoch 41/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.7589 - root_mean_squared
_error: 2.9595 - mean_absolute_error: 1.7226
Epoch 42/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7588 - root_mean_squared
_error: 2.9595 - mean_absolute_error: 1.7263
Epoch 43/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7578 - root_mean_squared
_error: 2.9594 - mean_absolute_error: 1.7218
Epoch 44/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7576 - root_mean_squared
_error: 2.9593 - mean_absolute_error: 1.7211
Epoch 45/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7568 - root_mean_squared
_error: 2.9592 - mean_absolute_error: 1.7206
Epoch 46/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7567 - root_mean_squared
_error: 2.9592 - mean_absolute_error: 1.7182
Epoch 47/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7564 - root_mean_squared
_error: 2.9591 - mean_absolute_error: 1.7211
Epoch 48/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7561 - root_mean_squared
_error: 2.9591 - mean_absolute_error: 1.7194
Epoch 49/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7551 - root_mean_squared
_error: 2.9589 - mean_absolute_error: 1.7232
Epoch 50/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7539 - root_mean_squared
_error: 2.9587 - mean_absolute_error: 1.7222
Epoch 51/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7540 - root_mean_squared
_error: 2.9587 - mean_absolute_error: 1.7204
Epoch 52/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7536 - root_mean_squared
_error: 2.9586 - mean_absolute_error: 1.7203
Epoch 53/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7523 - root_mean_squared
_error: 2.9584 - mean_absolute_error: 1.7160
Epoch 54/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7522 - root_mean_squared
_error: 2.9584 - mean_absolute_error: 1.7231
Epoch 55/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7527 - root_mean_squared
_error: 2.9585 - mean_absolute_error: 1.7215
```

```
Epoch 56/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7515 - root_mean_squared
_error: 2.9583 - mean_absolute_error: 1.7167
Epoch 57/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7524 - root_mean_squared
_error: 2.9584 - mean_absolute_error: 1.7225
Epoch 58/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7504 - root_mean_squared
_error: 2.9581 - mean_absolute_error: 1.7219
Epoch 59/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7500 - root_mean_squared
_error: 2.9580 - mean_absolute_error: 1.7207
Epoch 60/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7493 - root_mean_squared
_error: 2.9579 - mean_absolute_error: 1.7217
Epoch 61/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7497 - root_mean_squared
_error: 2.9580 - mean_absolute_error: 1.7215
Epoch 62/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7480 - root_mean_squared
_error: 2.9577 - mean_absolute_error: 1.7216
Epoch 63/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7484 - root_mean_squared
_error: 2.9578 - mean_absolute_error: 1.7212
Epoch 64/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7466 - root_mean_squared
_error: 2.9575 - mean_absolute_error: 1.7215
Epoch 65/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7465 - root_mean_squared
_error: 2.9574 - mean_absolute_error: 1.7194
Epoch 66/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7473 - root_mean_squared
_error: 2.9576 - mean_absolute_error: 1.7211
Epoch 67/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7457 - root_mean_squared
_error: 2.9573 - mean_absolute_error: 1.7227
Epoch 68/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7462 - root_mean_squared
_error: 2.9574 - mean_absolute_error: 1.7220
Epoch 69/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7453 - root_mean_squared
_error: 2.9573 - mean_absolute_error: 1.7170
Epoch 70/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7449 - root_mean_squared
_error: 2.9572 - mean_absolute_error: 1.7196
Epoch 71/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7431 - root_mean_squared
_error: 2.9569 - mean_absolute_error: 1.7164
Epoch 72/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7432 - root_mean_squared
_error: 2.9569 - mean_absolute_error: 1.7244
Epoch 73/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7436 - root_mean_squared
_error: 2.9570 - mean_absolute_error: 1.7218
Epoch 74/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7426 - root_mean_squared
_error: 2.9568 - mean_absolute_error: 1.7226
Epoch 75/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7429 - root_mean_squared
_error: 2.9568 - mean_absolute_error: 1.7213
Epoch 76/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7416 - root_mean_squared
_error: 2.9566 - mean_absolute_error: 1.7178
Epoch 77/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7401 - root_mean_squared
_error: 2.9564 - mean_absolute_error: 1.7186
```

```
Epoch 78/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7400 - root_mean_squared
_error: 2.9563 - mean_absolute_error: 1.7198
Epoch 79/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7408 - root_mean_squared
_error: 2.9565 - mean_absolute_error: 1.7195
Epoch 80/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7390 - root_mean_squared
_error: 2.9562 - mean_absolute_error: 1.7235
Epoch 81/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7381 - root_mean_squared
_error: 2.9560 - mean_absolute_error: 1.7227
Epoch 82/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7382 - root_mean_squared
_error: 2.9560 - mean_absolute_error: 1.7165
Epoch 83/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7380 - root_mean_squared
_error: 2.9560 - mean_absolute_error: 1.7208
Epoch 84/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7378 - root_mean_squared
_error: 2.9560 - mean_absolute_error: 1.7161
Epoch 85/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7371 - root_mean_squared
_error: 2.9559 - mean_absolute_error: 1.7225
Epoch 86/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7368 - root_mean_squared
_error: 2.9558 - mean_absolute_error: 1.7198
Epoch 87/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7370 - root_mean_squared
_error: 2.9558 - mean_absolute_error: 1.7189
Epoch 88/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7358 - root_mean_squared
_error: 2.9556 - mean_absolute_error: 1.7188
Epoch 89/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.7351 - root_mean_squared
_error: 2.9555 - mean_absolute_error: 1.7182
Epoch 90/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7354 - root_mean_squared
_error: 2.9556 - mean_absolute_error: 1.7201
Epoch 91/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7344 - root_mean_squared
_error: 2.9554 - mean_absolute_error: 1.7176
Epoch 92/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7349 - root_mean_squared
_error: 2.9555 - mean_absolute_error: 1.7196
Epoch 93/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7350 - root_mean_squared
_error: 2.9555 - mean_absolute_error: 1.7205
Epoch 94/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7336 - root_mean_squared
_error: 2.9553 - mean_absolute_error: 1.7182
Epoch 95/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7343 - root_mean_squared
_error: 2.9554 - mean_absolute_error: 1.7170
Epoch 96/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7338 - root_mean_squared
_error: 2.9553 - mean_absolute_error: 1.7158
Epoch 97/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7314 - root_mean_squared
_error: 2.9549 - mean_absolute_error: 1.7140
Epoch 98/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7311 - root_mean_squared
_error: 2.9548 - mean_absolute_error: 1.7193
Epoch 99/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7307 - root_mean_squared
_error: 2.9548 - mean_absolute_error: 1.7173
```

```
Epoch 100/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7307 - root_mean_squared
_error: 2.9548 - mean_absolute_error: 1.7167
Epoch 101/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7306 - root_mean_squared
_error: 2.9548 - mean_absolute_error: 1.7172
Epoch 102/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7300 - root_mean_squared
_error: 2.9547 - mean_absolute_error: 1.7177
Epoch 103/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7291 - root_mean_squared
_error: 2.9545 - mean_absolute_error: 1.7177
Epoch 104/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7296 - root_mean_squared
_error: 2.9546 - mean_absolute_error: 1.7127
Epoch 105/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7289 - root_mean_squared
_error: 2.9545 - mean_absolute_error: 1.7140
Epoch 106/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7281 - root_mean_squared
_error: 2.9543 - mean_absolute_error: 1.7183
Epoch 107/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7281 - root_mean_squared
_error: 2.9543 - mean_absolute_error: 1.7176
Epoch 108/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7276 - root_mean_squared
_error: 2.9543 - mean_absolute_error: 1.7169
Epoch 109/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7269 - root_mean_squared
_error: 2.9541 - mean_absolute_error: 1.7159
Epoch 110/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7259 - root_mean_squared
_error: 2.9540 - mean_absolute_error: 1.7159
Epoch 111/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7262 - root_mean_squared
_error: 2.9540 - mean_absolute_error: 1.7167
Epoch 112/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7256 - root_mean_squared
_error: 2.9539 - mean_absolute_error: 1.7181
Epoch 113/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7244 - root_mean_squared
_error: 2.9537 - mean_absolute_error: 1.7174
Epoch 114/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7239 - root_mean_squared
_error: 2.9536 - mean_absolute_error: 1.7184
Epoch 115/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7245 - root_mean_squared
_error: 2.9537 - mean_absolute_error: 1.7192
Epoch 116/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7236 - root_mean_squared
_error: 2.9536 - mean_absolute_error: 1.7180
Epoch 117/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7236 - root_mean_squared
_error: 2.9536 - mean_absolute_error: 1.7150
Epoch 118/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7228 - root_mean_squared
_error: 2.9534 - mean_absolute_error: 1.7195
Epoch 119/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7226 - root_mean_squared
_error: 2.9534 - mean_absolute_error: 1.7176
Epoch 120/1000
41/41 [==============================] - 1s 24ms/step - loss: 8.7222 - root_mean_squared
_error: 2.9533 - mean_absolute_error: 1.7168
Epoch 121/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7216 - root_mean_squared
_error: 2.9532 - mean_absolute_error: 1.7218
```

```
Epoch 122/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7214 - root_mean_squared
_error: 2.9532 - mean_absolute_error: 1.7206
Epoch 123/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7210 - root_mean_squared
_error: 2.9531 - mean_absolute_error: 1.7180
Epoch 124/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7208 - root_mean_squared
_error: 2.9531 - mean_absolute_error: 1.7166
Epoch 125/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7203 - root_mean_squared
_error: 2.9530 - mean_absolute_error: 1.7158
Epoch 126/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7193 - root_mean_squared
_error: 2.9528 - mean_absolute_error: 1.7205
Epoch 127/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7195 - root_mean_squared
_error: 2.9529 - mean_absolute_error: 1.7157
Epoch 128/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7195 - root_mean_squared
_error: 2.9529 - mean_absolute_error: 1.7178
Epoch 129/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7189 - root_mean_squared
_error: 2.9528 - mean_absolute_error: 1.7144
Epoch 130/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7175 - root_mean_squared
_error: 2.9525 - mean_absolute_error: 1.7124
Epoch 131/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7176 - root_mean_squared
_error: 2.9526 - mean_absolute_error: 1.7201
Epoch 132/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7157 - root_mean_squared
_error: 2.9522 - mean_absolute_error: 1.7178
Epoch 133/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7170 - root_mean_squared
_error: 2.9525 - mean_absolute_error: 1.7182
Epoch 134/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7153 - root_mean_squared
_error: 2.9522 - mean_absolute_error: 1.7151
Epoch 135/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7156 - root_mean_squared
_error: 2.9522 - mean_absolute_error: 1.7168
Epoch 136/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7146 - root_mean_squared
_error: 2.9521 - mean_absolute_error: 1.7162
Epoch 137/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7141 - root_mean_squared
_error: 2.9520 - mean_absolute_error: 1.7182
Epoch 138/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7141 - root_mean_squared
_error: 2.9520 - mean_absolute_error: 1.7173
Epoch 139/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7151 - root_mean_squared
_error: 2.9521 - mean_absolute_error: 1.7205
Epoch 140/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7145 - root_mean_squared
_error: 2.9520 - mean_absolute_error: 1.7167
Epoch 141/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7131 - root_mean_squared
_error: 2.9518 - mean_absolute_error: 1.7191
Epoch 142/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7118 - root_mean_squared
_error: 2.9516 - mean_absolute_error: 1.7203
Epoch 143/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7118 - root_mean_squared
_error: 2.9516 - mean_absolute_error: 1.7174
```

```
Epoch 144/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7116 - root_mean_squared
_error: 2.9515 - mean_absolute_error: 1.7148
Epoch 145/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7117 - root_mean_squared
_error: 2.9516 - mean_absolute_error: 1.7201
Epoch 146/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7098 - root_mean_squared
_error: 2.9512 - mean_absolute_error: 1.7177
Epoch 147/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7107 - root_mean_squared
_error: 2.9514 - mean_absolute_error: 1.7162
Epoch 148/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7087 - root_mean_squared
_error: 2.9510 - mean_absolute_error: 1.7201
Epoch 149/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7090 - root_mean_squared
_error: 2.9511 - mean_absolute_error: 1.7169
Epoch 150/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7083 - root_mean_squared
_error: 2.9510 - mean_absolute_error: 1.7168
Epoch 151/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7076 - root_mean_squared
_error: 2.9509 - mean_absolute_error: 1.7207
Epoch 152/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7087 - root_mean_squared
_error: 2.9510 - mean_absolute_error: 1.7174
Epoch 153/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7072 - root_mean_squared
_error: 2.9508 - mean_absolute_error: 1.7183
Epoch 154/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7066 - root_mean_squared
_error: 2.9507 - mean_absolute_error: 1.7149
Epoch 155/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7076 - root_mean_squared
_error: 2.9509 - mean_absolute_error: 1.7169
Epoch 156/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7055 - root_mean_squared
_error: 2.9505 - mean_absolute_error: 1.7147
Epoch 157/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7058 - root_mean_squared
_error: 2.9506 - mean_absolute_error: 1.7158
Epoch 158/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7049 - root_mean_squared
_error: 2.9504 - mean_absolute_error: 1.7144
Epoch 159/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7051 - root_mean_squared
_error: 2.9504 - mean_absolute_error: 1.7133
Epoch 160/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7038 - root_mean_squared
_error: 2.9502 - mean_absolute_error: 1.7137
Epoch 161/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7042 - root_mean_squared
_error: 2.9503 - mean_absolute_error: 1.7184
Epoch 162/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.7033 - root_mean_squared
_error: 2.9501 - mean_absolute_error: 1.7134
Epoch 163/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7026 - root_mean_squared
_error: 2.9500 - mean_absolute_error: 1.7105
Epoch 164/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7019 - root_mean_squared
_error: 2.9499 - mean_absolute_error: 1.7130
Epoch 165/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.7027 - root_mean_squared
_error: 2.9500 - mean_absolute_error: 1.7141
```

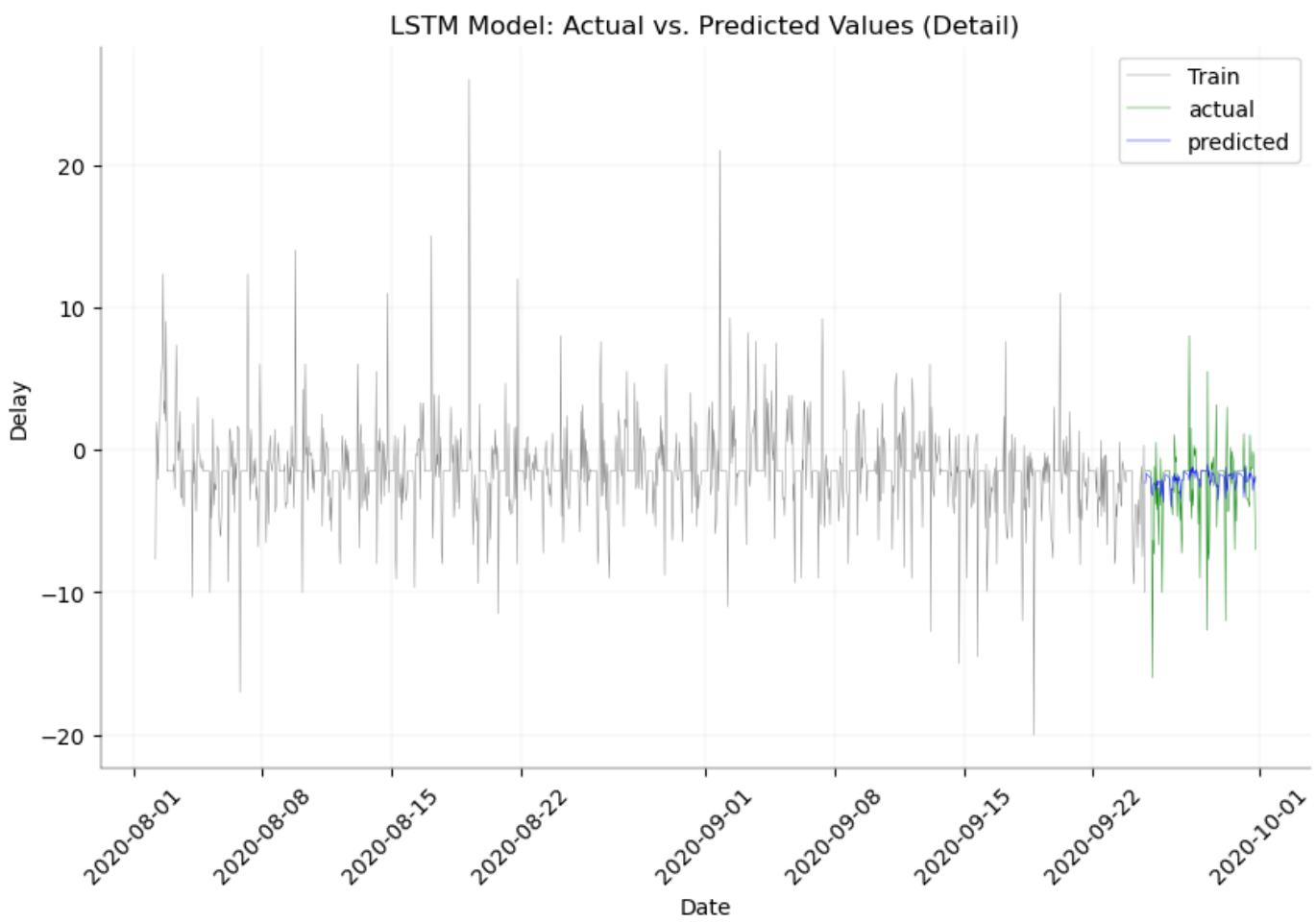
```
Epoch 166/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.7011 - root_mean_squared
_error: 2.9498 - mean_absolute_error: 1.7159
Epoch 167/1000
41/41 [==============================] - 1s 25ms/step - loss: 8.7002 - root_mean_squared
_error: 2.9496 - mean_absolute_error: 1.7149
Epoch 168/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.7005 - root_mean_squared
_error: 2.9497 - mean_absolute_error: 1.7151
Epoch 169/1000
41/41 [==============================] - 1s 28ms/step - loss: 8.6993 - root_mean_squared
_error: 2.9495 - mean_absolute_error: 1.7150
Epoch 170/1000
41/41 [==============================] - 1s 24ms/step - loss: 8.7009 - root_mean_squared
_error: 2.9497 - mean_absolute_error: 1.7174
Epoch 171/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6999 - root_mean_squared
_error: 2.9496 - mean_absolute_error: 1.7172
Epoch 172/1000
41/41 [==============================] - 1s 27ms/step - loss: 8.6989 - root_mean_squared
_error: 2.9494 - mean_absolute_error: 1.7146
Epoch 173/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.6982 - root_mean_squared
_error: 2.9493 - mean_absolute_error: 1.7169
Epoch 174/1000
41/41 [==============================] - 1s 22ms/step - loss: 8.6982 - root_mean_squared
_error: 2.9493 - mean_absolute_error: 1.7164
Epoch 175/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6970 - root_mean_squared
_error: 2.9491 - mean_absolute_error: 1.7144
Epoch 176/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6972 - root_mean_squared
_error: 2.9491 - mean_absolute_error: 1.7153
Epoch 177/1000
41/41 [==============================] - 1s 20ms/step - loss: 8.6967 - root_mean_squared
_error: 2.9490 - mean_absolute_error: 1.7152
Epoch 178/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6951 - root_mean_squared
_error: 2.9487 - mean_absolute_error: 1.7182
Epoch 179/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.6965 - root_mean_squared
_error: 2.9490 - mean_absolute_error: 1.7142
Epoch 180/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6941 - root_mean_squared
_error: 2.9486 - mean_absolute_error: 1.7163
Epoch 181/1000
41/41 [==============================] - 1s 23ms/step - loss: 8.6942 - root_mean_squared
_error: 2.9486 - mean_absolute_error: 1.7170
Epoch 182/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6959 - root_mean_squared
_error: 2.9489 - mean_absolute_error: 1.7134
Epoch 183/1000
41/41 [==============================] - 1s 21ms/step - loss: 8.6943 - root_mean_squared
_error: 2.9486 - mean_absolute_error: 1.7125
```

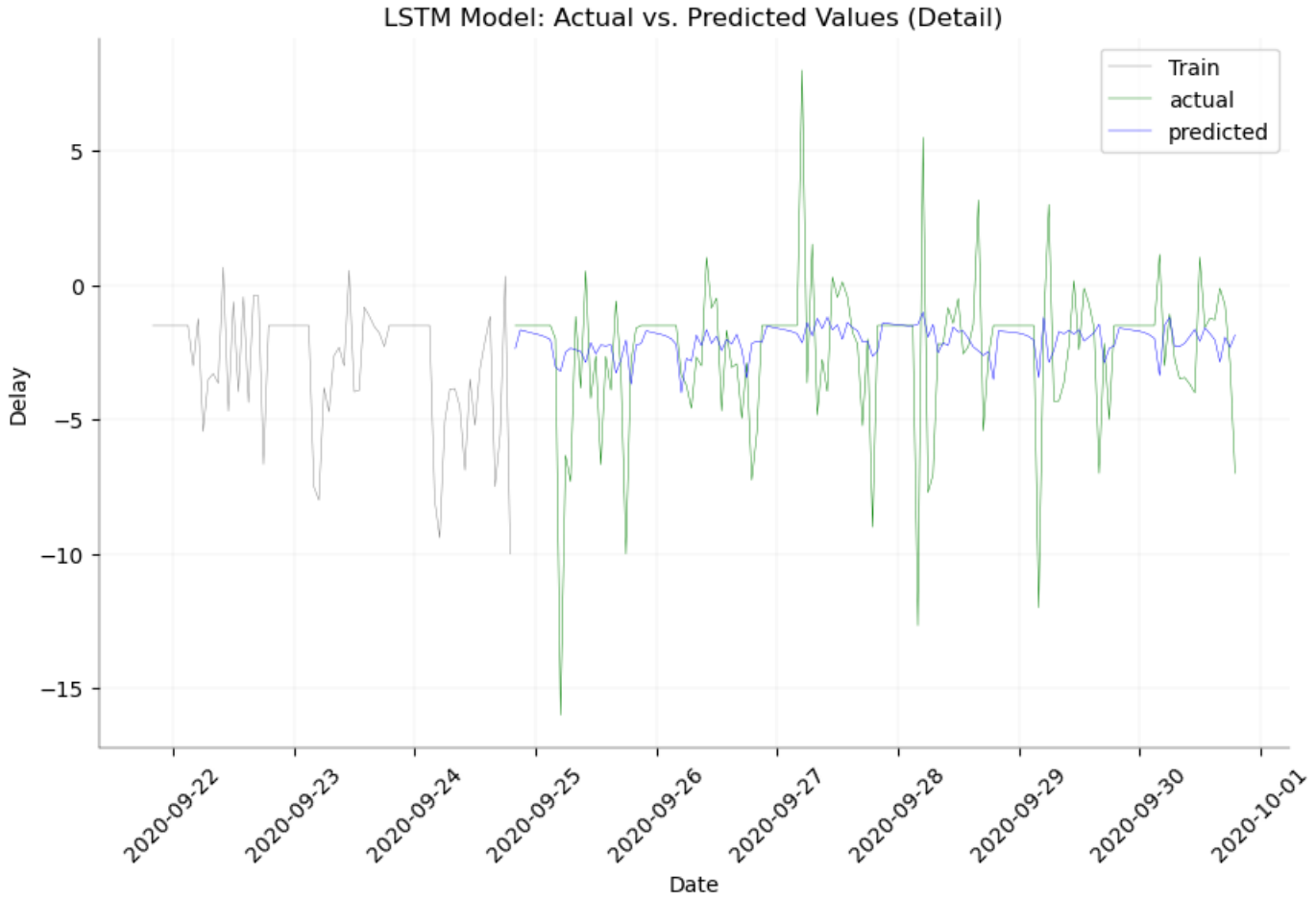
Out[820]: <keras.src.callbacks.History at 0x2b2c266c7c0>

In [826…

```
predicted = model.predict(X_test)
show_result(y_test, predicted)
show_result_detail(y_test, predicted)
```

```
5/5 [==============================] - 0s 9ms/step
```






```
Test RMSE: 2.797925942229235
```

Results

- The **model captures** global behaviours and **patterns**, but not better than SARIMAX.
- **High values for hourly delay fail to be predicted**. The predicted values tend to lay around the mean.
- **RMSE** is approx **2.8**

Further Developements

- Set **different LOOKBACK** values
- **Upgrade LSTM architecture, add layers and parameters** for richer non-linear models.
- Optimize training

CONCLUSIONS AND FURTHER DEVELOPMENTS

- **Forecasting daily delays** rather than hourly ones would be reasonable, and **likely yield better results**. **More data is needed** in order to do that.
- SARIMAX and LSTM models should be **optimized** by **tuning parameters** and modifying **architectures**.
- **Anomaly detection** could be employed to predict **spikes in the delivery time**.
- **More advanced models** should be considered. LSTMs could be replaced by **Transformers**.

```
#!pip freeze
```