excel-analysis

April 2, 2024

[1]:
```python
# importing of common library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```python
data = pd.read_excel("data/passenger-bus-data.xlsx")
```

1 Decriptive Statistics

[3]:
```python
data.head()
```

[3]:
```
        date  week_before_bus_departures  bus_departures  \
0 2020-12-07                      2406.0            2489
1 2020-12-14                      2489.0            2326
2 2020-12-21                      2326.0            2407
3 2020-12-28                      2407.0            2276
4 2021-01-04                      2207.0            2431

   week_over_week_variance_number_bus  week_over_week_variance_perc_bus  \
0                                 -83                              43.1
1                                 163                              52.9
2                                 -81                               1.4
3                                 131                              19.4
4                                -224                             -98.3

   week_before_passenger_departures  passenger_departures  \
0                             32301                 32779
1                             32779                 32046
2                             32046                 30884
3                             30884                 27864
4                             26692                 31629

   week_over_week_variance_number_passenger  \
0                                       478
1                                      -733
2                                     -1162
```

```
3                                       -3020
4                                        4937

   week_over_week_variance_perc_passenger  passengers_per_bus_date  \
0                                   -23.5                   213.05
1                                   -12.7                   223.08
2                                    11.6                   233.42
3                                  -114.3                   214.60
4                                   166.3                   223.46

   fall_2019_for_carrier
0                  317.4
1                  317.4
2                  317.4
3                  317.4
4                  298.7
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 11 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   date                                    123 non-null    datetime64[ns]
 1   week_before_bus_departures              123 non-null    float64
 2   bus_departures                          123 non-null    int64
 3   week_over_week_variance_number_bus      123 non-null    int64
 4   week_over_week_variance_perc_bus        123 non-null    float64
 5   week_before_passenger_departures        123 non-null    int64
 6   passenger_departures                    123 non-null    int64
 7   week_over_week_variance_number_passenger  123 non-null    int64
 8   week_over_week_variance_perc_passenger  123 non-null    float64
 9   passengers_per_bus_date                 123 non-null    float64
 10  fall_2019_for_carrier                   123 non-null    float64
dtypes: datetime64[ns](1), float64(5), int64(5)
memory usage: 10.7 KB
```

[5]:
```
data.describe()
```

[5]:
```
                      date  week_before_bus_departures  bus_departures  \
count                  123                  123.000000      123.000000
mean   2022-02-07 00:00:00                 2574.962602     2578.699187
min    2020-12-07 00:00:00                 2207.000000     2276.000000
25%    2021-07-08 12:00:00                 2546.000000     2547.500000
50%    2022-02-07 00:00:00                 2600.000000     2600.000000
75%    2022-09-08 12:00:00                 2653.000000     2654.000000
max    2023-04-10 00:00:00                 2738.000000     2750.000000
```

```
std                    NaN                 105.461103      104.315013

       week_over_week_variance_number_bus  week_over_week_variance_perc_bus  \
count                          123.000000                        123.000000
mean                            -3.739837                         -1.636585
min                           -231.000000                       -106.000000
25%                            -20.000000                        -13.100000
50%                             -5.000000                          0.000000
75%                              9.500000                          8.100000
max                            200.000000                        121.500000
std                             60.416565                         31.559606

       week_before_passenger_departures  passenger_departures  \
count                        123.000000            123.000000
mean                       60022.016260          60450.658537
min                        26692.000000          27864.000000
25%                        45300.500000          46956.500000
50%                        63695.000000          65959.000000
75%                        73971.500000          74345.500000
max                        84119.000000          84119.000000
std                        17173.394246          17115.461182

       week_over_week_variance_number_passenger  \
count                                123.000000
mean                                 428.642276
min                               -19114.000000
25%                                 -657.500000
50%                                  349.000000
75%                                 1463.500000
max                                12083.000000
std                                 3159.085391

       week_over_week_variance_perc_passenger  passengers_per_bus_date  \
count                              123.000000               123.000000
mean                                -5.298374               289.964553
min                               -806.800000               198.650000
25%                                -24.500000               262.245000
50%                                  0.000000               306.360000
75%                                 29.800000               321.350000
max                                336.900000               340.380000
std                                105.164457                38.953815

       fall_2019_for_carrier
count             123.000000
mean              323.900813
min               298.700000
25%               330.400000
```

```
50%              330.400000
75%              330.400000
max              330.400000
std               12.455794
```

[6]:

```python
# Extracting of month and year from the date columnb
data['month'] = data['date'].dt.month
data['year'] = data['date'].dt.year

# Setting date as index
data.set_index('date', inplace=True)
```

[7]:

```python
# Printing the new dataset
data
```

[7]:

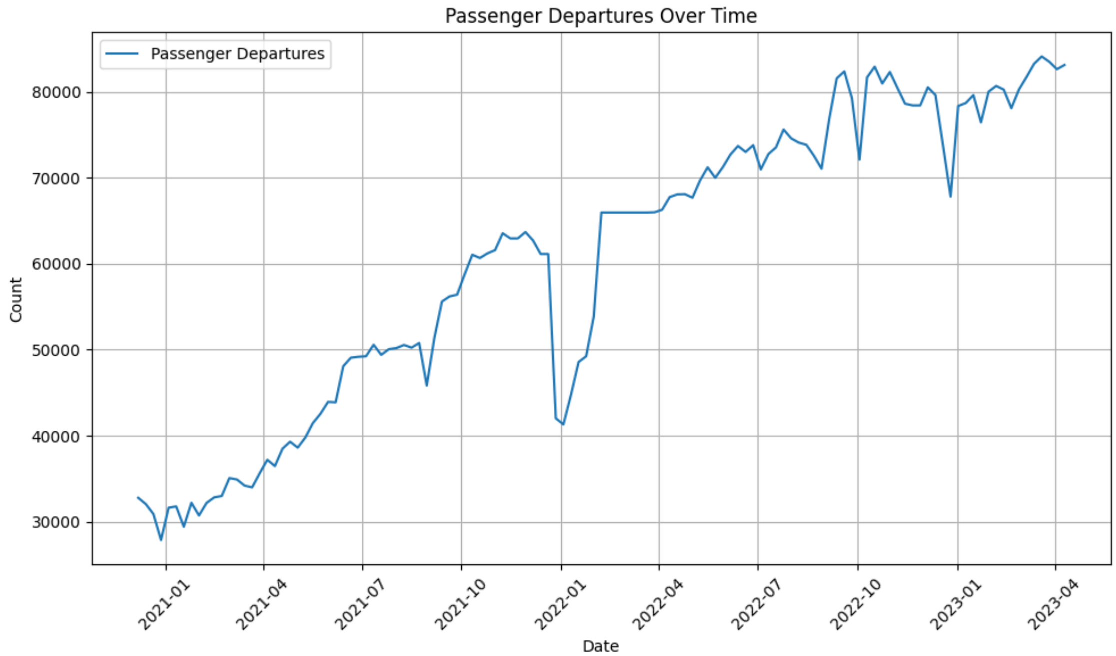
```
            week_before_bus_departures  bus_departures  \
date
2020-12-07                      2406.0            2489
2020-12-14                      2489.0            2326
2020-12-21                      2326.0            2407
2020-12-28                      2407.0            2276
2021-01-04                      2207.0            2431
…                                  …              …
2023-03-13                      2720.0            2715
2023-03-20                      2715.0            2730
2023-03-27                      2730.0            2728
2023-04-03                      2645.0            2700
2023-04-10                      2700.0            2750

            week_over_week_variance_number_bus  \
date
2020-12-07                                 -83
2020-12-14                                 163
2020-12-21                                 -81
2020-12-28                                 131
2021-01-04                                -224
…                                          …
2023-03-13                                   5
2023-03-20                                 -15
2023-03-27                                   2
2023-04-03                                 -55
2023-04-10                                 -50

            week_over_week_variance_perc_bus  \
date
2020-12-07                              43.1
2020-12-14                              52.9
```

```
2020-12-21                                 1.4
2020-12-28                                19.4
2021-01-04                               -98.3
…                                          …
2023-03-13                                52.3
2023-03-20                               -53.2
2023-03-27                                 7.2
2023-04-03                               -73.8
2023-04-10                                -6.3

            week_before_passenger_departures  passenger_departures  \
date
2020-12-07                             32301                 32779
2020-12-14                             32779                 32046
2020-12-21                             32046                 30884
2020-12-28                             30884                 27864
2021-01-04                             26692                 31629
…                                      …                     …
2023-03-13                             81732                 83269
2023-03-20                             83269                 84119
2023-03-27                             84119                 83498
2023-04-03                             82009                 82635
2023-04-10                             82635                 83126

            week_over_week_variance_number_passenger  \
date
2020-12-07                                       478
2020-12-14                                      -733
2020-12-21                                     -1162
2020-12-28                                     -3020
2021-01-04                                      4937
…                                              …
2023-03-13                                      1537
2023-03-20                                       850
2023-03-27                                      -621
2023-04-03                                       626
2023-04-10                                       491

            week_over_week_variance_perc_passenger  passengers_per_bus_date  \
date
2020-12-07                                   -23.5                   213.05
2020-12-14                                   -12.7                   223.08
2020-12-21                                    11.6                   233.42
2020-12-28                                  -114.3                   214.60
2021-01-04                                   166.3                   223.46
…                                            …                       …
2023-03-13                                   -33.0                   322.31
```

```
2023-03-20                                    105.6                   322.31
2023-03-27                                    -72.4                   323.28
2023-04-03                                    100.0                   306.02
2023-04-10                                     17.9                   316.75

            fall_2019_for_carrier  month  year
date
2020-12-07                  317.4     12  2020
2020-12-14                  317.4     12  2020
2020-12-21                  317.4     12  2020
2020-12-28                  317.4     12  2020
2021-01-04                  298.7      1  2021
…                           …    …    …
2023-03-13                  330.4      3  2023
2023-03-20                  330.4      3  2023
2023-03-27                  330.4      3  2023
2023-04-03                  312.1      4  2023
2023-04-10                  298.7      4  2023

[123 rows x 12 columns]
```

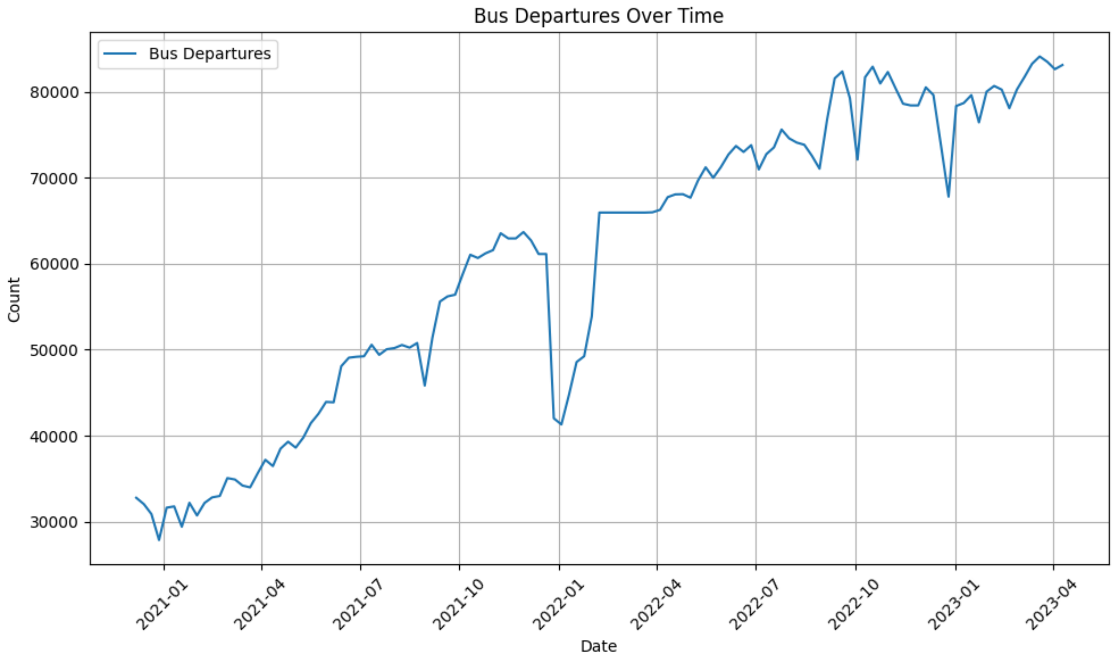
2 Data Visualization

[8]:
```python
# Plotting the Passenger Depatures
plt.figure(figsize=(10, 6))
plt.plot(data['passenger_departures'], label='Passenger Departures')
plt.xlabel('Date')
plt.ylabel('Count')
plt.title('Passenger Departures Over Time')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[9]:

```
# Plotting the Bus Depatures
plt.figure(figsize=(10, 6))
plt.plot(data['passenger_departures'], label='Bus Departures')
plt.xlabel('Date')
plt.ylabel('Count')
plt.title('Bus Departures Over Time')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



3 Forecasting

3.1 Bus Passenger Forecasting

[10]:
```
# Generate all possible combinations of p, d, and q using pmdauto Auto ARIMA
import pmdarima as pm

passenger_model = pm.auto_arima(data['passenger_departures'], seasonal=True,␣
↪m=7)
passenger_model
```

[10]:
```
ARIMA(order=(0, 1, 0), scoring_args={}, seasonal_order=(0, 0, 0, 7),
      suppress_warnings=True)
```

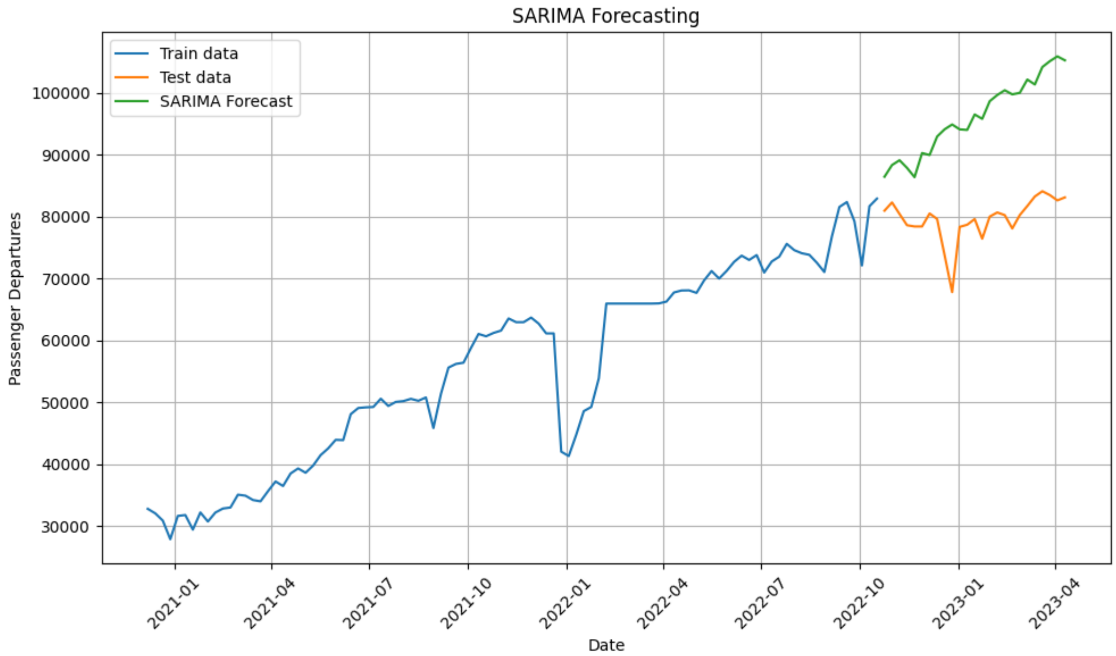
[11]:
```
# importing library
from sklearn.model_selection import train_test_split
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Splitting data into train and test sets
train_data, test_data = train_test_split(data, test_size=0.2, shuffle=False)
train_data.index.freq = train_data.index.inferred_freq

# Fit SARIMA model
passenger_model = SARIMAX(train_data['passenger_departures'], order=(0, 1, 0),␣
↪seasonal_order=(1, 1,1, 7))
```

```
passenger_results = passenger_model.fit()

# Forecasting
passenger_forecast = passenger_results.forecast(steps=len(test_data))

# Forecast future values to 2030
start_date = data.index[-1] + pd.Timedelta(days=1)
end_date = start_date + pd.offsets.DateOffset(years=9)  # Extend forecast␣
↪horizon to 2030
passenger_forecast_to_2030 = passenger_results.predict(start=start_date,␣
↪end=end_date)
```

[12]:
```
# Find the first value where bus departures exceed 3900
for i, value in enumerate(passenger_forecast_to_2030):
    if value > 125000:
        print("First forecasted value where bus departures exceed 125000:")
        print(f"Date: {passenger_forecast_to_2030.index[i]}, Value: {value}")
        break
```

```
First forecasted value where bus departures exceed 125000:
Date: 2023-10-02 00:00:00, Value: 126047.16991158869
```

[13]:
```
# Plotting of Passenger Departure Forecasting
plt.figure(figsize=(10, 6))
plt.plot(train_data.index, train_data['passenger_departures'], label='Train␣
↪data')
plt.plot(test_data.index, test_data['passenger_departures'], label='Test data')
plt.plot(test_data.index, passenger_forecast, label='SARIMA Forecast')
plt.title('SARIMA Forecasting')
plt.xlabel('Date')
plt.ylabel('Passenger Departures')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



3.2 Bus Departures Forecasting

[14]:
```
# Generate all possible combinations of p, d, and q using pmdauto Auto ARIMA
import pmdarima as pm

bus_model = pm.auto_arima(data['passenger_departures'], seasonal=True, m=7)
bus_model
```

[14]:
```
ARIMA(order=(0, 1, 0), scoring_args={}, seasonal_order=(0, 0, 0, 7),
      suppress_warnings=True)
```

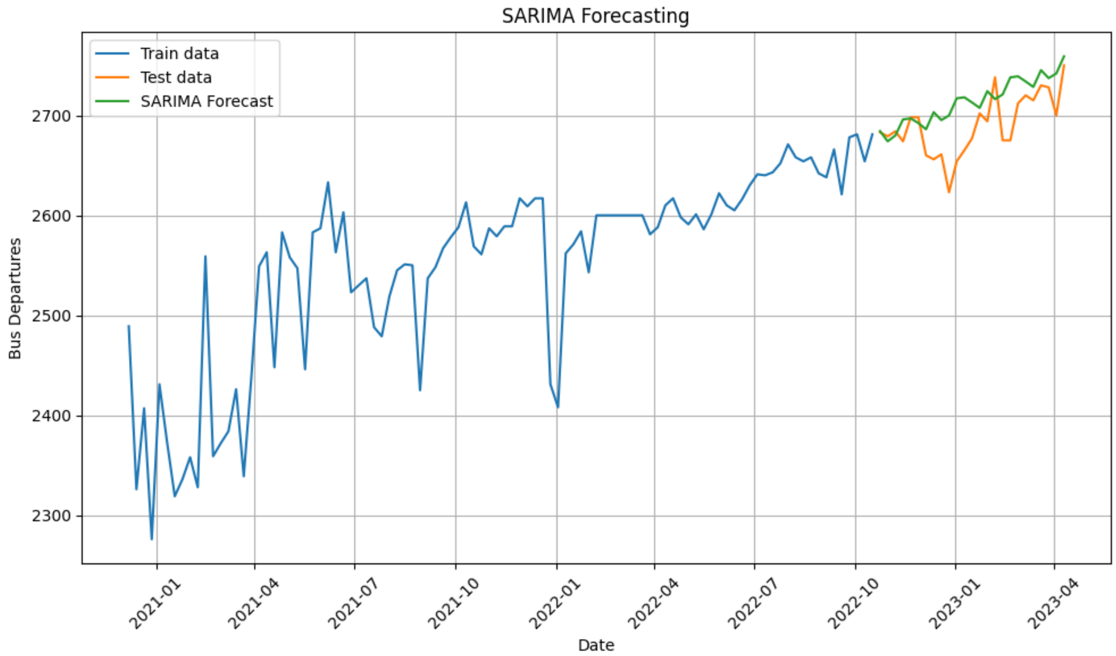
[15]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Splitting data into train and test sets
train_data, test_data = train_test_split(data, test_size=0.2, shuffle=False)
train_data.index.freq = train_data.index.inferred_freq



# Fit SARIMA model
model = SARIMAX(train_data['bus_departures'], order=(1, 1, 1),␣
 ↪seasonal_order=(1, 1, 1, 7))
results = model.fit()

# Forecasting
bus_forecast = results.forecast(steps=len(test_data))
```

```
# Forecast future values to 2030
start_date = data.index[-1] + pd.Timedelta(days=1)
end_date = start_date + pd.offsets.DateOffset(years=8)  # Extend forecast␣
↪horizon to 2030
bus_forecast_to_2030 = results.predict(start=start_date, end=end_date)
```

[16]:
```
# Find the first value where bus departures exceed 3900
for i, value in enumerate(bus_forecast_to_2030):
    if value > 3900:
        print("First forecasted value where bus departures exceed 3900:")
        print(f"Date: {bus_forecast_to_2030.index[i]}, Value: {value}")
        break
```

```
First forecasted value where bus departures exceed 3900:
Date: 2030-08-05 00:00:00, Value: 3900.046954986401
```

[17]:
```
# Plotting of Bus Departure Forecasting
plt.figure(figsize=(10, 6))
plt.plot(train_data.index, train_data['bus_departures'], label='Train data')
plt.plot(test_data.index, test_data['bus_departures'], label='Test data')
plt.plot(test_data.index, bus_forecast, label='SARIMA Forecast')
plt.title('SARIMA Forecasting')
plt.xlabel('Date')
plt.ylabel('Bus Departures')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



4 Models Forecasting.

[32]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

X = data[['week_before_bus_departures', 'week_over_week_variance_number_bus', 'week_over_week_variance_perc_bus',
          'week_before_passenger_departures', 'week_over_week_variance_number_passenger',
          'week_over_week_variance_perc_passenger', 'passengers_per_bus_date', 'fall_2019_for_carrier']]
y_bus = data['bus_departures']
y_passenger = data['passenger_departures']

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_bus_train, y_bus_test, y_passenger_train, y_passenger_test = train_test_split(data, y_bus, y_passenger, test_size=0.2, random_state=42)

# Train a single Linear Regression model for both targets using the training data
lr_model = LinearRegression()
lr_model.fit(X_train, y_bus_train)
lr_model.fit(X_train, y_passenger_train)
```

```
# Make predictions on the testing data
predictions_bus_test = lr_model.predict(X_test)
predictions_passenger_test = lr_model.predict(X_test)

# Evaluate the model performance on testing data
rmse_bus_test = np.sqrt(mean_squared_error(y_bus_test, predictions_bus_test))
rmse_passenger_test = np.sqrt(mean_squared_error(y_passenger_test,␣
↪predictions_passenger_test))

mae_bus_test = mean_absolute_error(y_bus_test, predictions_bus_test)
mae_passenger_test = mean_absolute_error(y_passenger_test,␣
↪predictions_passenger_test)

r2_bus_test = r2_score(y_bus_test, predictions_bus_test)
r2_passenger_test = r2_score(y_passenger_test, predictions_passenger_test)

print("Metrics for Bus Departures on Testing Data:")
print("RMSE:", rmse_bus_test)
print("MAE:", mae_bus_test)
print("R-squared:", r2_bus_test)
print("\nMetrics for Passenger Departures on Testing Data:")
print("RMSE:", rmse_passenger_test)
print("MAE:", mae_passenger_test)
print("R-squared:", r2_passenger_test)
```

```
Metrics for Bus Departures on Testing Data:
RMSE: 57198.814581073275
MAE: 54669.56
R-squared: -438171.8797319983

Metrics for Passenger Departures on Testing Data:
RMSE: 1.1999816708318134e-11
MAE: 8.149072527885438e-12
R-squared: 1.0
```

[33]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Convert DataFrame to numpy arrays
X = np.array(data[['week_before_bus_departures',␣
↪'week_over_week_variance_number_bus', 'week_over_week_variance_perc_bus',
          'week_before_passenger_departures',␣
↪'week_over_week_variance_number_passenger',
```

```
        'week_over_week_variance_perc_passenger', 'passengers_per_bus_date',␣
 ↪'fall_2019_for_carrier']])
y_bus = np.array(data['bus_departures'])
y_passenger = np.array(data['passenger_departures'])

# Split the data into training and testing sets for bus departures
X_train_bus, X_test_bus, y_bus_train, y_bus_test = train_test_split(X, y_bus,␣
 ↪test_size=0.2, random_state=42)

# Create a Random Forest Regressor model for bus departures
rf_model_bus = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the bus departures model
rf_model_bus.fit(X_train_bus, y_bus_train)

# Make predictions on the testing data for bus departures
predictions_bus_rf = rf_model_bus.predict(X_test_bus)

# Evaluate the bus departures model performance
rmse_bus_rf = np.sqrt(mean_squared_error(y_bus_test, predictions_bus_rf))
print("RMSE for Bus Departures (Random Forest):", rmse_bus_rf)

# Split the data into training and testing sets for passenger departures
X_train_passenger, X_test_passenger, y_passenger_train, y_passenger_test =␣
 ↪train_test_split(X, y_passenger, test_size=0.2, random_state=42)

# Create a Random Forest Regressor model for passenger departures
rf_model_passenger = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the passenger departures model
rf_model_passenger.fit(X_train_passenger, y_passenger_train)

# Make predictions on the testing data for passenger departures
predictions_passenger_rf = rf_model_passenger.predict(X_test_passenger)

# Evaluate the passenger departures model performance
rmse_passenger_rf = np.sqrt(mean_squared_error(y_passenger_test,␣
 ↪predictions_passenger_rf))
print("RMSE for Passenger Departures (Random Forest):", rmse_passenger_rf)
```

```
RMSE for Bus Departures (Random Forest): 63.33881085085193
RMSE for Passenger Departures (Random Forest): 4920.212948759434
```

[ ]:
```
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
```

```
# Extract features and target variables
X1 = data[['week_before_bus_departures', 'week_over_week_variance_number_bus',␣
 ↪'week_over_week_variance_perc_bus',
           'week_before_passenger_departures',␣
 ↪'week_over_week_variance_number_passenger',
           'week_over_week_variance_perc_passenger', 'passengers_per_bus_date',␣
 ↪'fall_2019_for_carrier']]
y_passenger1 = data['passenger_departures']

# Normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
X_scaled = scaler.fit_transform(X1)
y_bus_scaled = scaler.fit_transform(np.array(y_bus1).reshape(-1, 1))
y_passenger_scaled = scaler.fit_transform(np.array(y_passenger1).reshape(-1, 1))

# Prepare data for LSTM (reshape input to [samples, time steps, features])
X_lstm = X_scaled.reshape(X_scaled.shape[0], 1, X_scaled.shape[1])

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_bus_train, y_bus_test = train_test_split(X_lstm,␣
 ↪y_bus_scaled, test_size=0.2, random_state=42)
_, _, y_passenger_train, y_passenger_test = train_test_split(X_lstm,␣
 ↪y_passenger_scaled, test_size=0.2, random_state=42)

# Define the LSTM model architecture
model = Sequential()
model.add(LSTM(units=50, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
history = model.fit(X_train, y_bus_train, epochs=10, batch_size=32,␣
 ↪validation_data=(X_test, y_bus_test), verbose=1)

# Make predictions on the testing data
predictions_bus_scaled = model.predict(X_test)

# Inverse scale the predictions
predictions_bus = scaler.inverse_transform(predictions_bus_scaled)

# Evaluate the model performance
rmse_bus = np.sqrt(mean_squared_error(y_bus, predictions_bus))
print("RMSE for Bus Departures (LSTM):", rmse_bus)
```