Assessment 1 - Time series forecasting

Model used - Facebook Prophet

About the model: Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Prophet is open source software released by Facebook's Core Data Science team. It is available for download on CRAN and PyPI.

Python code

Importing the necessary libraries

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
from prophet import Prophet
import requests
import json
```

Defining the API endpoint url for data loading and extraction

In [9]:
```
api_url = "https://www.data.act.gov.au/resource/x7dn-77he.json"

# Fetch data from the API
response = requests.get(api_url)
```

Data loading, pre processing and feature engineering

In [10]:
```
data = pd.read_json(response.text)

# Convert the 'date' column to datetime
data['date'] = pd.to_datetime(data['date'])

# Rename columns for Prophet compatibility
data = data.rename(columns={'date': 'ds', 'total': 'y', 'myway': 'myway', 'paper_ticket': 'paper_ticket'})
```

EDA

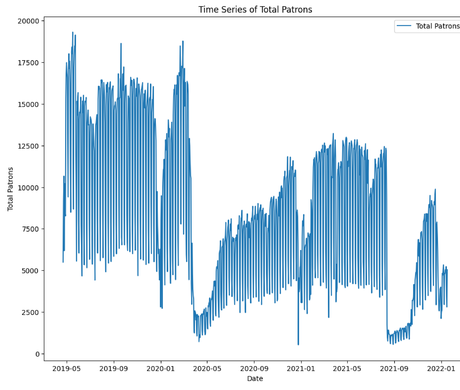
In [14]:
```
#Descriptive statistics

print(data.describe())
```

```
                myway  paper_ticket             y
count   1000.000000   1000.000000   1000.000000
mean    8123.180000    355.943000   8479.123000
std     4696.895156    243.715344   4803.810514
min      512.000000      8.000000    522.000000
25%     4131.750000    213.000000   4436.000000
50%     7504.000000    326.000000   7947.000000
75%    11932.500000    460.750000  12315.000000
max    18936.000000   2491.000000  19318.000000
```

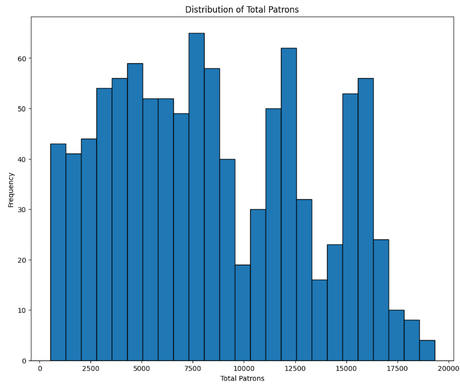
In [16]:
```
#Date and total

plt.figure(figsize=(11, 9))
plt.plot(data['ds'], data['y'], label='Total Patrons')
plt.xlabel('Date')
plt.ylabel('Total Patrons')
plt.title('Time Series of Total Patrons')
plt.legend()
plt.show()
```



In [17]:
```
#Distribution of total

plt.figure(figsize=(11, 9))
plt.hist(data['y'], bins=25, edgecolor='black')
plt.xlabel('Total Patrons')
plt.ylabel('Frequency')
plt.title('Distribution of Total Patrons')
plt.show()
```



In [18]:
```
#Correlation matrix

correlation_matrix = data[['y', 'myway', 'paper_ticket']].corr()
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
                   y     myway  paper_ticket
y           1.000000  0.998937      0.459175
myway       0.998937  1.000000      0.417738
paper_ticket  0.459175  0.417738      1.000000
```

Model training

In [11]:
```
# Splitting the dataset into training and testing sets
train_data = data[:-30]  # Training data
test_data = data[-30:]   # Testing data

# Create a Prophet model
model = Prophet()
model.add_regressor('myway')
model.add_regressor('paper_ticket')

# Fitting the model to the training data
model.fit(train_data)
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpge9ev00d/3vt11c2w.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpge9ev00d/64nvi_a3.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=3
33977', 'data', 'file=/tmp/tmpge9ev00d/3vt11c2w.json', 'init=/tmp/tmpge9ev00d/64nvi_a3.json', 'output', 'file=/tmp/tmpge9ev00d/pr
ophet_modelybgmmf7_/prophet_model-20230924044047.csv', 'method=optimize', 'algorithm=lbfgs', 'iter=10000']
04:40:47 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
04:40:48 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

Out[11]:
```
<prophet.forecaster.Prophet at 0x7a4389304460>
```

Forecasting for next day, 7 days, next 14 days, next 30 days with three predictions P10, P50, and P90 and printing the metrics

In [13]:
```
forecast_horizon = [1, 7, 14, 30]

# Generating quantile forecasts (P10, P50, P90)
quantiles = [0.1, 0.5, 0.9]

for day in forecast_horizon:

    forecast_date = data['ds'].max() + pd.DateOffset(days=day)
    future = model.make_future_dataframe(periods=day)
    future['myway'] = test_data['myway'].values[-1]
    future['paper_ticket'] = test_data['paper_ticket'].values[-1]

    # Forecasting the values
    forecast = model.predict(future)

    # Extracting the quantile forecasts
    quantile_forecasts = forecast[['ds', 'yhat_lower', 'yhat', 'yhat_upper']].tail(1).values[0]

    # Calculating the error metrics
    actual_values = test_data['y'].values[-day:]
    forecast_values = forecast['yhat'].tail(day).values
    mae = mean_absolute_error(actual_values, forecast_values) #Mean absolute error
    mse = mean_squared_error(actual_values, forecast_values) #Mean squared error
    rmse = np.sqrt(mse) #Root mean squared error

    # Printing the forecast for the specific prediction horizon
    print(f"Forecast for {day} days:")
    #Printing for that date
    print(f"Forecast Date: {forecast_date}")
    #Printing for P10
    print(f"P10: {quantile_forecasts[1]}")
    #Printing for P50
    print(f"P50 (Median): {quantile_forecasts[2]}")
    #Printing for P90
    print(f"P90: {quantile_forecasts[3]}\n")

    #Printing the metrics

    print(f"MAE: {mae}")
    print(f"MSE: {mse}")
    print(f"RMSE: {rmse}\n")
```

```
Forecast for 1 days:
Forecast Date: 2022-01-18 00:00:00
P10: 5030.747317931984
P50 (Median): 5033.627772839321
P90: 5036.365480777402

MAE: 0.6277728393206417
MSE: 0.39409873778870014
RMSE: 0.6277728393206417

Forecast for 7 days:
Forecast Date: 2022-01-24 00:00:00
P10: 5030.912943198585
P50 (Median): 5033.844851846275
P90: 5036.805001222978

MAE: 619.7655668675923
MSE: 1079897.367754565
RMSE: 1039.181104406044

Forecast for 14 days:
Forecast Date: 2022-01-31 00:00:00
P10: 5030.887447682910 5
P50 (Median): 5034.039192996939
P90: 5036.778121966934

MAE: 552.8017474691017
MSE: 877998.1944323495
RMSE: 937.0155785430409

Forecast for 30 days:
Forecast Date: 2022-02-16 00:00:00
P10: 5030.774039417556
P50 (Median): 5034.172260269608
P90: 5037.753477110828

MAE: 1239.2252115449635
MSE: 2621910.9040207565
RMSE: 1619.2315782557962
```