Forecasting the Future: Electricity Load and Price Prediction Based on Weather

May 2, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

%matplotlib inline
```

1 Reading .csv files

[2]:
```
weather_dataframe = pd.read_csv('/content/weather_features.csv',␣
 ↪parse_dates=['dt_iso'])

energy_dataframe = pd.read_csv('/content/energy_dataset.csv',␣
 ↪parse_dates=['time'])
```

2 Data Cleaning & Pre-processing

[3]:
```
columns_to_be_converted = weather_dataframe.select_dtypes(include=['int64']).
 ↪columns
for col in columns_to_be_converted:
    weather_dataframe[col] = weather_dataframe[col].values.astype('float64')
```

[4]:
```
energy_dataframe['time'] = pd.to_datetime(energy_dataframe['time'], utc=True,␣
 ↪infer_datetime_format=True)
energy_dataframe = energy_dataframe.set_index('time')
```

```
<ipython-input-4-7590cbad87a7>:1: UserWarning: The argument
'infer_datetime_format' is deprecated and will be removed in a future version. A
strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.
  energy_dataframe['time'] = pd.to_datetime(energy_dataframe['time'], utc=True,
infer_datetime_format=True)
```

[5]:
```
columns_to_remove = [
    "generation biomass",
    "generation fossil brown coal/lignite",
    "generation fossil gas",
    "generation fossil hard coal",
    "generation fossil oil",
    "generation hydro pumped storage consumption",
    "generation hydro run-of-river and poundage",
    "generation hydro water reservoir",
    "generation nuclear",
    "generation other",
    "generation other renewable",
    "generation solar",
    "generation waste",
    "generation wind onshore",
    "generation fossil coal-derived gas",
    "generation fossil oil shale",
    "generation fossil peat",
    "generation geothermal",
    "generation hydro pumped storage aggregated",
    "generation marine",
    "generation wind offshore",
    "forecast wind offshore eday ahead",
    "forecast solar day ahead",
    "forecast wind onshore day ahead"
]
# Experimental Features
    # "price day ahead",
    # "total load forecast",
energy_dataframe = energy_dataframe.drop(columns=[col for col in␣
 ↪columns_to_remove if col in energy_dataframe.columns])
```

[6]:
```
energy_dataframe.duplicated(keep='first').sum()
```

[6]:
```
0
```

[7]:
```
energy_dataframe.interpolate(method='linear', limit_direction='forward',␣
 ↪inplace=True, axis=0)

print('Non-zero values in each column:\n', energy_dataframe.astype(bool).
 ↪sum(axis=0), sep='\n')
energy_dataframe.isnull().values.any()
```

```
Non-zero values in each column:

total load forecast    35064
total load actual      35064
```

```
price day ahead        35064
price actual           35064
dtype: int64
```

[7]:
```
False
```

[8]:
```
energy_dataframe.interpolate(method='linear', axis=0,␣
 ↪limit_direction='forward', inplace=True)
```

[9]:
```
weather_dataframe['time'] = pd.to_datetime(weather_dataframe['dt_iso'],␣
 ↪utc=True)
weather_dataframe = weather_dataframe.drop(['dt_iso'], axis=1)
weather_dataframe = weather_dataframe.set_index('time')
```

[10]:
```
weather_dataframe.duplicated(keep='first').sum()
```

[10]:
```
8622
```

[11]:
```
weather_dataframe = weather_dataframe.drop(['weather_main', 'weather_id',
                                'weather_description', 'weather_icon'], axis=1)
```

[12]:
```
weather_dataframe.interpolate(limit_direction='forward', inplace=True, axis=0)
```

3 Merging Datasets

[13]:
```
city_names = weather_dataframe.city_name.unique()
for city in city_names:
    print(city)
```

```
Valencia
Madrid
Bilbao
 Barcelona
Seville
```

[14]:
```
city_1, city_2, city_3, city_4, city_5 = [x for _, x in weather_dataframe.
 ↪groupby('city_name')]
cities = [city_1, city_2, city_3, city_4, city_5]
```

[15]:
```
merged_dataframe = energy_dataframe

for c in cities:
    city = c['city_name'].unique()
    city_str = str(city).replace("'", "").replace('[', '').replace(']', '').
 ↪replace(' ', '')
    c = c.add_suffix('_{}'.format(city_str))
```

```
    merged_dataframe = merged_dataframe.merge(c, on=['time'], how='outer')
    merged_dataframe = merged_dataframe.drop('city_name_{}'.format(city_str),␣
 ↪axis=1)

final_cols = merged_dataframe.columns
print(final_cols)
```

```
Index(['total load forecast', 'total load actual', 'price day ahead',
       'price actual', 'temp_Barcelona', 'temp_min_Barcelona',
       'temp_max_Barcelona', 'pressure_Barcelona', 'humidity_Barcelona',
       'wind_speed_Barcelona', 'wind_deg_Barcelona', 'rain_1h_Barcelona',
       'rain_3h_Barcelona', 'snow_3h_Barcelona', 'clouds_all_Barcelona',
       'temp_Bilbao', 'temp_min_Bilbao', 'temp_max_Bilbao', 'pressure_Bilbao',
       'humidity_Bilbao', 'wind_speed_Bilbao', 'wind_deg_Bilbao',
       'rain_1h_Bilbao', 'rain_3h_Bilbao', 'snow_3h_Bilbao',
       'clouds_all_Bilbao', 'temp_Madrid', 'temp_min_Madrid',
       'temp_max_Madrid', 'pressure_Madrid', 'humidity_Madrid',
       'wind_speed_Madrid', 'wind_deg_Madrid', 'rain_1h_Madrid',
       'rain_3h_Madrid', 'snow_3h_Madrid', 'clouds_all_Madrid', 'temp_Seville',
       'temp_min_Seville', 'temp_max_Seville', 'pressure_Seville',
       'humidity_Seville', 'wind_speed_Seville', 'wind_deg_Seville',
       'rain_1h_Seville', 'rain_3h_Seville', 'snow_3h_Seville',
       'clouds_all_Seville', 'temp_Valencia', 'temp_min_Valencia',
       'temp_max_Valencia', 'pressure_Valencia', 'humidity_Valencia',
       'wind_speed_Valencia', 'wind_deg_Valencia', 'rain_1h_Valencia',
       'rain_3h_Valencia', 'snow_3h_Valencia', 'clouds_all_Valencia'],
      dtype='object')
```

[16]:
```
columns = ['temp_Barcelona', 'humidity_Barcelona', 'wind_speed_Barcelona',␣
 ↪'price actual']
print(merged_dataframe.index[0])
# Display the first row of the selected columns
first_row = merged_dataframe[columns].iloc[0]
print(first_row)
```

```
2014-12-31 23:00:00+00:00
temp_Barcelona          281.625
humidity_Barcelona      100.000
wind_speed_Barcelona      7.000
price actual             65.410
Name: 2014-12-31 23:00:00+00:00, dtype: float64
```

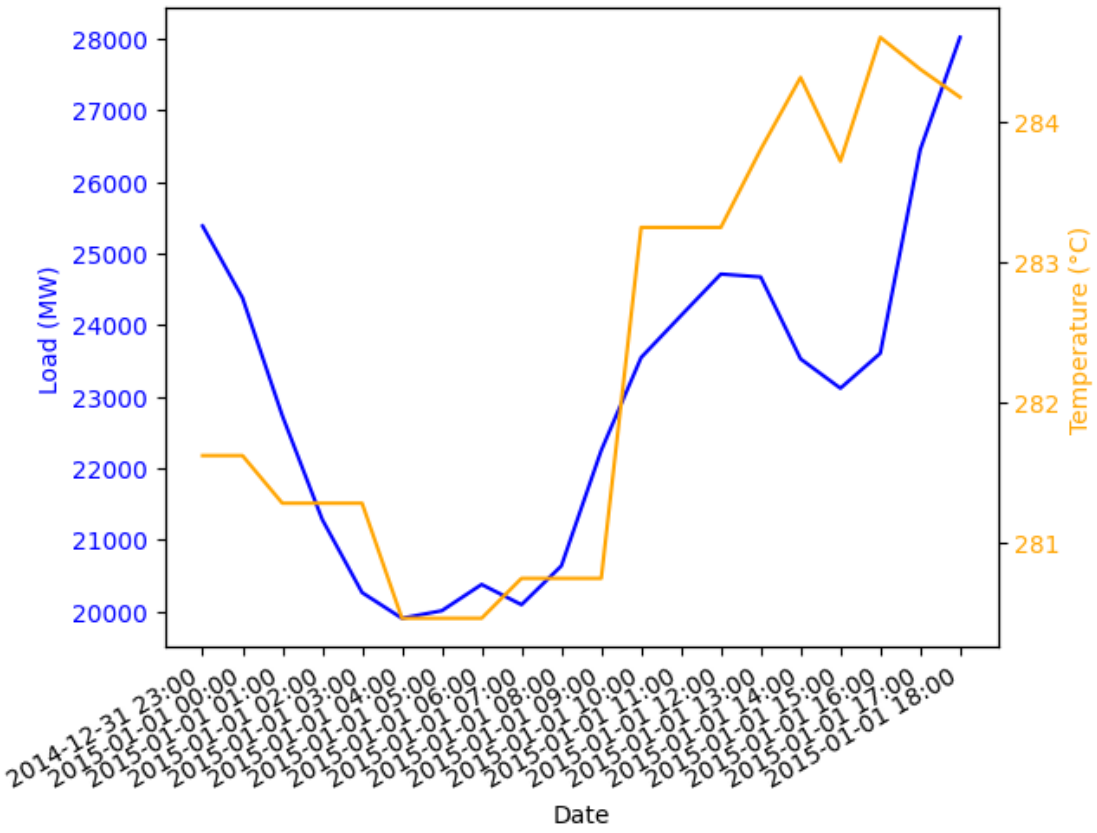
[17]:
```
merged_dataframe = merged_dataframe.interpolate()
```

[18]:
```
merged_dataframe.isnull().values.any()
```

[18]: False

4 Data Visualization

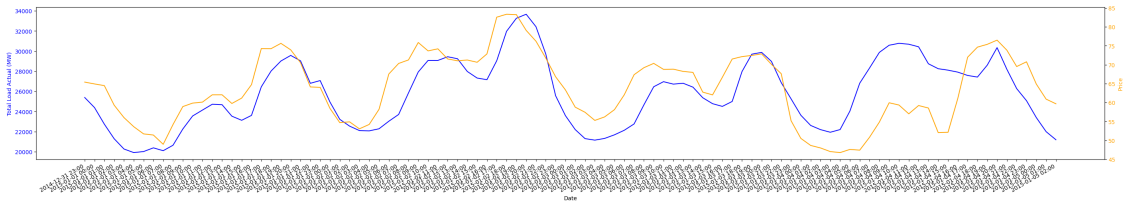
[19]:
```
from datetime import datetime

dates = [str(date)[:16] for date in merged_dataframe.index[:20]]
load = merged_dataframe['total load actual'][:20]
temperature = merged_dataframe['temp_Barcelona'][:20]
fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Load (MW)', color='blue')
ax1.plot(dates, load, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Temperature (°C)', color='orange')
ax2.plot(dates, temperature, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
plt.figure(figsize=(15, 7))

plt.show()
```



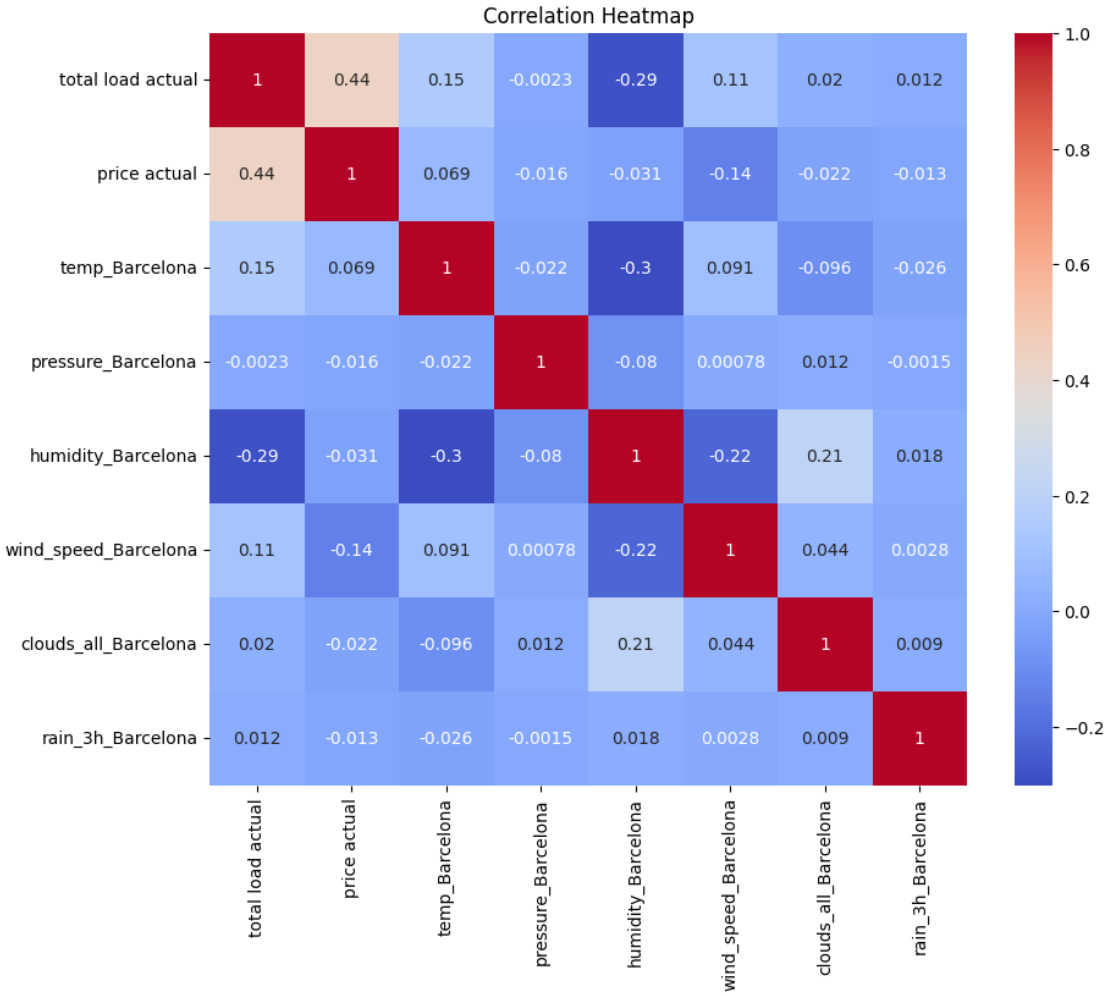

```
<Figure size 1500x700 with 0 Axes>
```

[20]:
```
from datetime import datetime

dates = [str(date)[:16] for date in merged_dataframe.index[:100]]
load = merged_dataframe['total load actual'][:100]
price = merged_dataframe['price actual'][:100]
fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Load Actual (MW)', color='blue')
ax1.plot(dates, load, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Price', color='orange')
ax2.plot(dates, price, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
```

```
fig.set_size_inches(35, 5, forward=True)

plt.show()
```

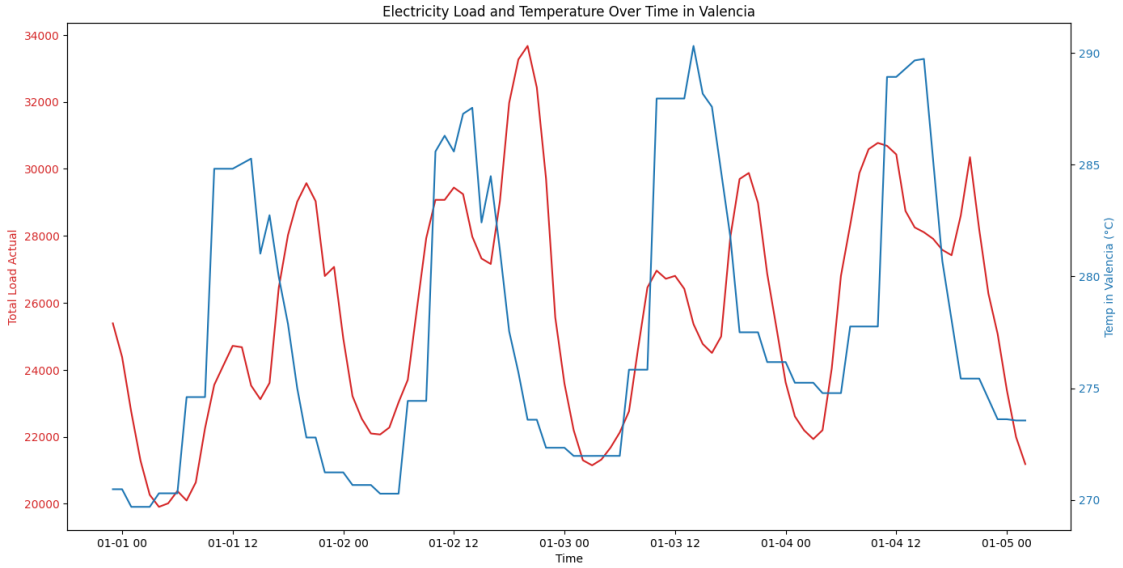


[21]:
```
import seaborn as sns
corr_matrix = merged_dataframe[['total load actual', 'price actual',
                  'temp_Barcelona', 'pressure_Barcelona', 'humidity_Barcelona',
                  'wind_speed_Barcelona', 'clouds_all_Barcelona',␣
 ↪'rain_3h_Barcelona'
                 ]].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



[22]:
```
fig, ax1 = plt.subplots(figsize=(14, 7))

color = 'tab:red'
ax1.set_xlabel('Time')
ax1.set_ylabel('Total Load Actual', color=color)
ax1.plot(merged_dataframe.index[:100], merged_dataframe['total load actual'][:
 ↪100], color=color, label='Total Load Actual')
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Temp in Valencia (°C)', color=color)
ax2.plot(merged_dataframe.index[:100], merged_dataframe['temp_Valencia'][:100],␣
 ↪color=color, label='Temperature in Valencia')
ax2.tick_params(axis='y', labelcolor=color)
```

```
fig.tight_layout()
plt.title('Electricity Load and Temperature Over Time in Valencia')
plt.show()
```



[23]:
```
import plotly.express as px

fig = px.scatter_3d(merged_dataframe, x='temp_Madrid', y='price actual',␣
 ↪z='total load actual',
                    color='temp_Madrid',
                    title='3D Scatter Plot: Weather Conditions in Madrid vs.␣
 ↪Electricity Load',
                    labels={'temp_Madrid': 'Temperature in Madrid (°F)',
                            'humidity_Madrid': 'Humidity in Madrid (%)',
                            'total load actual': 'Total Load Actual'})

fig.update_layout(margin=dict(l=0, r=0, b=0, t=30))
fig.update_traces(marker=dict(size=1.5))
fig.show()
```

5 ML Models

5.1 Random Forest Regressor

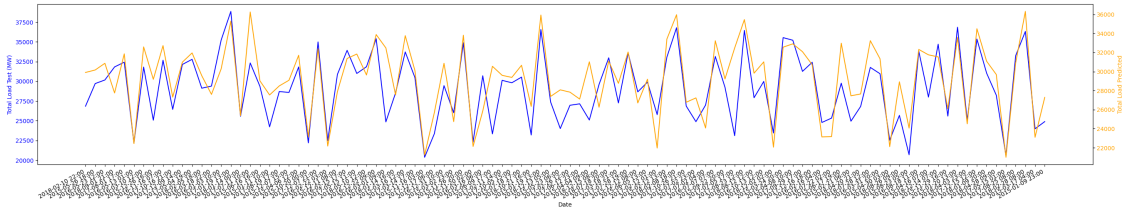
[24]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from math import sqrt

merged_dataframe['hour'] = merged_dataframe.index.hour
merged_dataframe['day_of_week'] = merged_dataframe.index.dayofweek
merged_dataframe['month'] = merged_dataframe.index.month

features = [
    'temp_Barcelona',
    'humidity_Barcelona',
    'wind_speed_Barcelona',
    'hour',
    'day_of_week',
    'month'
]
X_load = merged_dataframe[features]
y_load = merged_dataframe['total load actual']

X_price = merged_dataframe[features]
y_price = merged_dataframe['price actual']

scaler = StandardScaler()
X_load_scaled = scaler.fit_transform(X_load)
X_price_scaled = scaler.fit_transform(X_price)

X_load_train, X_load_test, y_load_train, y_load_test =␣
 ↪train_test_split(X_load_scaled, y_load, test_size=0.2, random_state=42)

X_price_train, X_price_test, y_price_train, y_price_test =␣
 ↪train_test_split(X_price_scaled, y_price, test_size=0.2, random_state=42)

regressor_load = RandomForestRegressor(n_estimators=100, random_state=42)
regressor_load.fit(X_load_train, y_load_train)

regressor_price = RandomForestRegressor(n_estimators=100, random_state=42)
regressor_price.fit(X_price_train, y_price_train)

y_load_pred = regressor_load.predict(X_load_test)
y_price_pred = regressor_price.predict(X_price_test)

mse_load = mean_squared_error(y_load_test, y_load_pred)
rmse_load = sqrt(mse_load)
mae_load = mean_absolute_error(y_load_test, y_load_pred)
r2_load = r2_score(y_load_test, y_load_pred)

mse_price = mean_squared_error(y_price_test, y_price_pred)
```

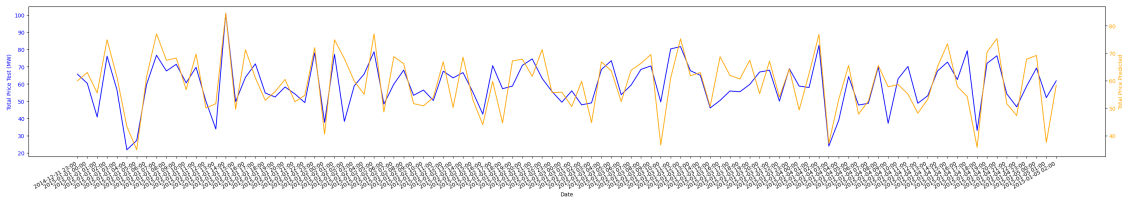
```
rmse_price = sqrt(mse_price)
mae_price = mean_absolute_error(y_price_test, y_price_pred)
r2_price = r2_score(y_price_test, y_price_pred)

print('Load Prediction RMSE:', rmse_load)
print('Load Prediction MAE:', mae_load)
print('Load Prediction R²:', r2_load)

print('Price Prediction RMSE:', rmse_price)
print('Price Prediction MAE:', mae_price)
print('Price Prediction R²:', r2_price)
```

```
Load Prediction RMSE: 2365.997895341652
Load Prediction MAE: 1616.8233389426882
Load Prediction R²: 0.7384301915759535
Price Prediction RMSE: 8.582923314509832
Price Prediction MAE: 6.216318472341475
Price Prediction R²: 0.6364657908535211
```

[25]:
```
dates = [str(date)[:16] for date in y_load_test.index][:100]
test = y_load_test.values[:100]
pred = y_load_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Load Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')


ax2 = ax1.twinx()
ax2.set_ylabel('Total Load Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



[26]:
```
dates = [str(date)[:16] for date in merged_dataframe.index][:100]
test = y_price_test[:100]
pred = y_price_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Price Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Total Price Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



5.1.1 Experimental Features Added Model

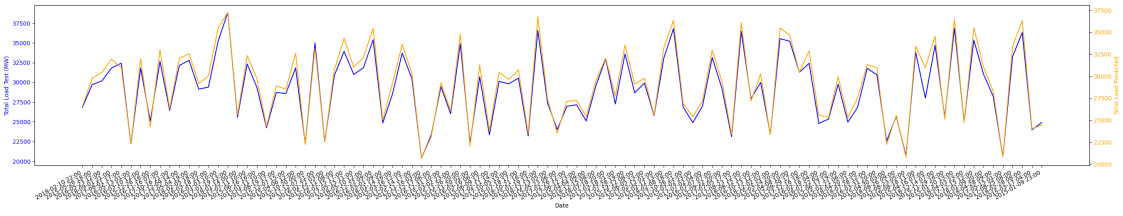
[27]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from math import sqrt

# Assuming 'merged_dataframe' has a DateTimeIndex and the necessary columns
merged_dataframe['hour'] = merged_dataframe.index.hour
merged_dataframe['day_of_week'] = merged_dataframe.index.dayofweek
merged_dataframe['month'] = merged_dataframe.index.month
```

```
# Adding 'total load forecast' and 'price day ahead' to the features list
features = [
    'temp_Barcelona',
    'humidity_Barcelona',
    'wind_speed_Barcelona',
    'hour',
    'day_of_week',
    'month',
    'total load forecast',  # added as a feature
    'price day ahead'       # added as a feature
]

X_load = merged_dataframe[features]
y_load = merged_dataframe['total load actual']

X_price = merged_dataframe[features]
y_price = merged_dataframe['price actual']

scaler = StandardScaler()
X_load_scaled = scaler.fit_transform(X_load)
X_price_scaled = scaler.fit_transform(X_price)

X_load_train, X_load_test, y_load_train, y_load_test =␣
 ↪train_test_split(X_load_scaled, y_load, test_size=0.2, random_state=42)
X_price_train, X_price_test, y_price_train, y_price_test =␣
 ↪train_test_split(X_price_scaled, y_price, test_size=0.2, random_state=42)

regressor_load = RandomForestRegressor(n_estimators=100, random_state=42)
regressor_load.fit(X_load_train, y_load_train)

regressor_price = RandomForestRegressor(n_estimators=100, random_state=42)
regressor_price.fit(X_price_train, y_price_train)

y_load_pred = regressor_load.predict(X_load_test)
y_price_pred = regressor_price.predict(X_price_test)

mse_load = mean_squared_error(y_load_test, y_load_pred)
rmse_load = sqrt(mse_load)
mae_load = mean_absolute_error(y_load_test, y_load_pred)
r2_load = r2_score(y_load_test, y_load_pred)

mse_price = mean_squared_error(y_price_test, y_price_pred)
rmse_price = sqrt(mse_price)
mae_price = mean_absolute_error(y_price_test, y_price_pred)
r2_price = r2_score(y_price_test, y_price_pred)
```

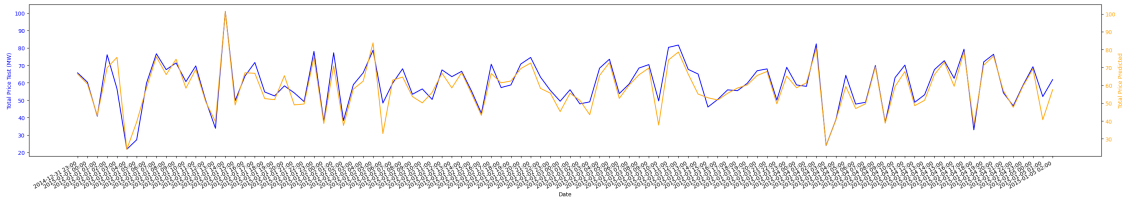
```
print('Load Prediction RMSE:', rmse_load)
print('Load Prediction MAE:', mae_load)
print('Load Prediction R²:', r2_load)

print('Price Prediction RMSE:', rmse_price)
print('Price Prediction MAE:', mae_price)
print('Price Prediction R^2:', r2_price)
```

```
Load Prediction RMSE: 412.75608752068956
Load Prediction MAE: 288.0417306916717
Load Prediction R²: 0.9920393985625786
Price Prediction RMSE: 5.534410806834109
Price Prediction MAE: 3.3579434534612407
Price Prediction R^2: 0.8488467095406759
```

[28]:
```
dates = [str(date)[:16] for date in y_load_test.index][:100]
test = y_load_test.values[:100]
pred = y_load_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Load Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')


ax2 = ax1.twinx()
ax2.set_ylabel('Total Load Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



[29]:
```
dates = [str(date)[:16] for date in merged_dataframe.index][:100]
test = y_price_test[:100]
pred = y_price_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Price Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Total Price Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



5.2 LSTM - Long Short-Term Memory Model

[30]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from math import sqrt
from sklearn.metrics import mean_squared_error

dataframe = merged_dataframe.copy()

dataframe['hour'] = dataframe.index.hour
dataframe['day_of_week'] = dataframe.index.dayofweek
dataframe['month'] = dataframe.index.month
```

```
features = [
    'temp_Barcelona',
    'humidity_Barcelona',
    'wind_speed_Barcelona',
    'hour',
    'day_of_week',
    'month',
]

X_load = dataframe[features]
y_load = dataframe['total load actual']
X_price = dataframe[features]
y_price = dataframe['price actual']

scaler = StandardScaler()
X_load_scaled = scaler.fit_transform(X_load)
X_price_scaled = scaler.fit_transform(X_price)

X_load_scaled = X_load_scaled.reshape(X_load_scaled.shape[0], 1, X_load_scaled.
 ↪shape[1])
X_price_scaled = X_price_scaled.reshape(X_price_scaled.shape[0], 1,␣
 ↪X_price_scaled.shape[1])

# Splitting the data
X_load_train, X_load_test, y_load_train, y_load_test =␣
 ↪train_test_split(X_load_scaled, y_load, test_size=0.2, random_state=42)
X_price_train, X_price_test, y_price_train, y_price_test =␣
 ↪train_test_split(X_price_scaled, y_price, test_size=0.2, random_state=42)

# Defining the LSTM model
def build_model(input_shape):
    model = Sequential()
    model.add(LSTM(50, activation='relu', input_shape=input_shape))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model

model_load = build_model((1, len(features)))
model_load.fit(X_load_train, y_load_train, epochs=150, batch_size=32, verbose=1)

model_price = build_model((1, len(features)))
model_price.fit(X_price_train, y_price_train, epochs=30, batch_size=32,␣
 ↪verbose=1)

y_load_pred = model_load.predict(X_load_test)
y_price_pred = model_price.predict(X_price_test)
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

Epoch 1/150
965/965 [==============================] - 8s 5ms/step - loss: 844268416.0000
Epoch 2/150
965/965 [==============================] - 6s 6ms/step - loss: 831181120.0000
Epoch 3/150
965/965 [==============================] - 4s 5ms/step - loss: 808970624.0000
Epoch 4/150
965/965 [==============================] - 4s 4ms/step - loss: 780310656.0000
Epoch 5/150
965/965 [==============================] - 5s 6ms/step - loss: 746336576.0000
Epoch 6/150
965/965 [==============================] - 6s 6ms/step - loss: 708226624.0000
Epoch 7/150
965/965 [==============================] - 5s 6ms/step - loss: 666662016.0000
Epoch 8/150
965/965 [==============================] - 5s 6ms/step - loss: 622232000.0000
Epoch 9/150
965/965 [==============================] - 5s 5ms/step - loss: 575646208.0000
Epoch 10/150
965/965 [==============================] - 5s 6ms/step - loss: 527536640.0000
Epoch 11/150
965/965 [==============================] - 4s 5ms/step - loss: 478496800.0000
Epoch 12/150
965/965 [==============================] - 5s 6ms/step - loss: 429316288.0000
Epoch 13/150
965/965 [==============================] - 4s 4ms/step - loss: 380664864.0000
Epoch 14/150
965/965 [==============================] - 4s 5ms/step - loss: 333248704.0000
Epoch 15/150
965/965 [==============================] - 6s 6ms/step - loss: 287854464.0000
Epoch 16/150
965/965 [==============================] - 4s 5ms/step - loss: 245225664.0000
Epoch 17/150
965/965 [==============================] - 5s 5ms/step - loss: 206054016.0000
Epoch 18/150
965/965 [==============================] - 5s 5ms/step - loss: 170955248.0000
Epoch 19/150
965/965 [==============================] - 4s 5ms/step - loss: 140436784.0000
Epoch 20/150
965/965 [==============================] - 5s 6ms/step - loss: 114814640.0000
Epoch 21/150
965/965 [==============================] - 4s 5ms/step - loss: 94171240.0000
Epoch 22/150
965/965 [==============================] - 4s 4ms/step - loss: 78210576.0000
Epoch 23/150
```

```
965/965 [==============================] - 5s 6ms/step - loss: 66141444.0000
Epoch 24/150
965/965 [==============================] - 4s 5ms/step - loss: 56561812.0000
Epoch 25/150
965/965 [==============================] - 5s 5ms/step - loss: 49051028.0000
Epoch 26/150
965/965 [==============================] - 5s 5ms/step - loss: 43275848.0000
Epoch 27/150
965/965 [==============================] - 4s 5ms/step - loss: 38601608.0000
Epoch 28/150
965/965 [==============================] - 5s 6ms/step - loss: 34470576.0000
Epoch 29/150
965/965 [==============================] - 4s 4ms/step - loss: 30658590.0000
Epoch 30/150
965/965 [==============================] - 4s 4ms/step - loss: 27103148.0000
Epoch 31/150
965/965 [==============================] - 5s 6ms/step - loss: 24299174.0000
Epoch 32/150
965/965 [==============================] - 4s 5ms/step - loss: 22008446.0000
Epoch 33/150
965/965 [==============================] - 5s 5ms/step - loss: 20104840.0000
Epoch 34/150
965/965 [==============================] - 5s 5ms/step - loss: 18485674.0000
Epoch 35/150
965/965 [==============================] - 4s 5ms/step - loss: 17113330.0000
Epoch 36/150
965/965 [==============================] - 5s 6ms/step - loss: 15943345.0000
Epoch 37/150
965/965 [==============================] - 5s 5ms/step - loss: 14941593.0000
Epoch 38/150
965/965 [==============================] - 6s 7ms/step - loss: 14070681.0000
Epoch 39/150
965/965 [==============================] - 10s 11ms/step - loss: 13321219.0000
Epoch 40/150
965/965 [==============================] - 7s 7ms/step - loss: 12690568.0000
Epoch 41/150
965/965 [==============================] - 4s 4ms/step - loss: 12151755.0000
Epoch 42/150
965/965 [==============================] - 5s 5ms/step - loss: 11690325.0000
Epoch 43/150
965/965 [==============================] - 5s 5ms/step - loss: 11296863.0000
Epoch 44/150
965/965 [==============================] - 8s 8ms/step - loss: 10957512.0000
Epoch 45/150
965/965 [==============================] - 7s 7ms/step - loss: 10663139.0000
Epoch 46/150
965/965 [==============================] - 8s 8ms/step - loss: 10404774.0000
Epoch 47/150
```

```
965/965 [==============================] - 4s 5ms/step - loss: 10176591.0000
Epoch 48/150
965/965 [==============================] - 4s 5ms/step - loss: 9975067.0000
Epoch 49/150
965/965 [==============================] - 6s 6ms/step - loss: 9791847.0000
Epoch 50/150
965/965 [==============================] - 5s 5ms/step - loss: 9626210.0000
Epoch 51/150
965/965 [==============================] - 5s 5ms/step - loss: 9477335.0000
Epoch 52/150
965/965 [==============================] - 5s 5ms/step - loss: 9343717.0000
Epoch 53/150
965/965 [==============================] - 4s 5ms/step - loss: 9226880.0000
Epoch 54/150
965/965 [==============================] - 5s 6ms/step - loss: 9122160.0000
Epoch 55/150
965/965 [==============================] - 4s 5ms/step - loss: 9027031.0000
Epoch 56/150
965/965 [==============================] - 4s 5ms/step - loss: 8940723.0000
Epoch 57/150
965/965 [==============================] - 5s 6ms/step - loss: 8861579.0000
Epoch 58/150
965/965 [==============================] - 4s 5ms/step - loss: 8790133.0000
Epoch 59/150
965/965 [==============================] - 5s 5ms/step - loss: 8725825.0000
Epoch 60/150
965/965 [==============================] - 5s 5ms/step - loss: 8668127.0000
Epoch 61/150
965/965 [==============================] - 4s 5ms/step - loss: 8619406.0000
Epoch 62/150
965/965 [==============================] - 5s 6ms/step - loss: 8576242.0000
Epoch 63/150
965/965 [==============================] - 4s 5ms/step - loss: 8535568.0000
Epoch 64/150
965/965 [==============================] - 4s 5ms/step - loss: 8499282.0000
Epoch 65/150
965/965 [==============================] - 5s 5ms/step - loss: 8465172.0000
Epoch 66/150
965/965 [==============================] - 4s 5ms/step - loss: 8435812.0000
Epoch 67/150
965/965 [==============================] - 5s 5ms/step - loss: 8406971.0000
Epoch 68/150
965/965 [==============================] - 5s 5ms/step - loss: 8378587.5000
Epoch 69/150
965/965 [==============================] - 4s 5ms/step - loss: 8354360.5000
Epoch 70/150
965/965 [==============================] - 5s 6ms/step - loss: 8329016.0000
Epoch 71/150
```

```
965/965 [==============================] - 4s 5ms/step - loss: 8307485.0000
Epoch 72/150
965/965 [==============================] - 4s 5ms/step - loss: 8286033.5000
Epoch 73/150
965/965 [==============================] - 5s 6ms/step - loss: 8265575.0000
Epoch 74/150
965/965 [==============================] - 4s 5ms/step - loss: 8245196.0000
Epoch 75/150
965/965 [==============================] - 5s 6ms/step - loss: 8226439.5000
Epoch 76/150
965/965 [==============================] - 4s 5ms/step - loss: 8208152.0000
Epoch 77/150
965/965 [==============================] - 4s 5ms/step - loss: 8190999.0000
Epoch 78/150
965/965 [==============================] - 6s 6ms/step - loss: 8173929.0000
Epoch 79/150
965/965 [==============================] - 9s 9ms/step - loss: 8156379.5000
Epoch 80/150
965/965 [==============================] - 6s 6ms/step - loss: 8140449.0000
Epoch 81/150
965/965 [==============================] - 4s 5ms/step - loss: 8125249.0000
Epoch 82/150
965/965 [==============================] - 5s 6ms/step - loss: 8110147.0000
Epoch 83/150
965/965 [==============================] - 4s 5ms/step - loss: 8095022.5000
Epoch 84/150
965/965 [==============================] - 4s 5ms/step - loss: 8080880.5000
Epoch 85/150
965/965 [==============================] - 5s 6ms/step - loss: 8066549.0000
Epoch 86/150
965/965 [==============================] - 6s 6ms/step - loss: 8052484.5000
Epoch 87/150
965/965 [==============================] - 5s 6ms/step - loss: 8037979.0000
Epoch 88/150
965/965 [==============================] - 4s 5ms/step - loss: 8024775.5000
Epoch 89/150
965/965 [==============================] - 4s 5ms/step - loss: 8010820.5000
Epoch 90/150
965/965 [==============================] - 5s 6ms/step - loss: 7998759.0000
Epoch 91/150
965/965 [==============================] - 4s 5ms/step - loss: 7984903.0000
Epoch 92/150
965/965 [==============================] - 5s 5ms/step - loss: 7973406.5000
Epoch 93/150
965/965 [==============================] - 5s 5ms/step - loss: 7962102.0000
Epoch 94/150
965/965 [==============================] - 4s 5ms/step - loss: 7950611.5000
Epoch 95/150
```

```
965/965 [==============================] - 5s 6ms/step - loss: 7940289.5000
Epoch 96/150
965/965 [==============================] - 4s 5ms/step - loss: 7928611.5000
Epoch 97/150
965/965 [==============================] - 4s 5ms/step - loss: 7919992.0000
Epoch 98/150
965/965 [==============================] - 5s 6ms/step - loss: 7909681.0000
Epoch 99/150
965/965 [==============================] - 4s 5ms/step - loss: 7900046.5000
Epoch 100/150
965/965 [==============================] - 5s 5ms/step - loss: 7890134.5000
Epoch 101/150
965/965 [==============================] - 5s 5ms/step - loss: 7882691.0000
Epoch 102/150
965/965 [==============================] - 4s 5ms/step - loss: 7873348.5000
Epoch 103/150
965/965 [==============================] - 5s 6ms/step - loss: 7864945.5000
Epoch 104/150
965/965 [==============================] - 4s 5ms/step - loss: 7856652.0000
Epoch 105/150
965/965 [==============================] - 4s 5ms/step - loss: 7849142.0000
Epoch 106/150
965/965 [==============================] - 5s 6ms/step - loss: 7842027.5000
Epoch 107/150
965/965 [==============================] - 4s 5ms/step - loss: 7834519.0000
Epoch 108/150
965/965 [==============================] - 5s 6ms/step - loss: 7826598.5000
Epoch 109/150
965/965 [==============================] - 4s 5ms/step - loss: 7820998.5000
Epoch 110/150
965/965 [==============================] - 4s 5ms/step - loss: 7814022.0000
Epoch 111/150
965/965 [==============================] - 5s 6ms/step - loss: 7806936.5000
Epoch 112/150
965/965 [==============================] - 4s 5ms/step - loss: 7801558.0000
Epoch 113/150
965/965 [==============================] - 5s 5ms/step - loss: 7793419.0000
Epoch 114/150
965/965 [==============================] - 5s 5ms/step - loss: 7787553.0000
Epoch 115/150
965/965 [==============================] - 4s 5ms/step - loss: 7782393.5000
Epoch 116/150
965/965 [==============================] - 6s 6ms/step - loss: 7775685.5000
Epoch 117/150
965/965 [==============================] - 4s 5ms/step - loss: 7771157.5000
Epoch 118/150
965/965 [==============================] - 4s 5ms/step - loss: 7765889.0000
Epoch 119/150
```

```
965/965 [==============================] - 5s 6ms/step - loss: 7760515.5000
Epoch 120/150
965/965 [==============================] - 4s 5ms/step - loss: 7754141.0000
Epoch 121/150
965/965 [==============================] - 5s 5ms/step - loss: 7750179.0000
Epoch 122/150
965/965 [==============================] - 5s 5ms/step - loss: 7744426.5000
Epoch 123/150
965/965 [==============================] - 4s 5ms/step - loss: 7739504.0000
Epoch 124/150
965/965 [==============================] - 5s 6ms/step - loss: 7734555.5000
Epoch 125/150
965/965 [==============================] - 4s 5ms/step - loss: 7728997.5000
Epoch 126/150
965/965 [==============================] - 4s 5ms/step - loss: 7723568.0000
Epoch 127/150
965/965 [==============================] - 5s 6ms/step - loss: 7719936.5000
Epoch 128/150
965/965 [==============================] - 5s 6ms/step - loss: 7714753.0000
Epoch 129/150
965/965 [==============================] - 6s 6ms/step - loss: 7709456.0000
Epoch 130/150
965/965 [==============================] - 4s 5ms/step - loss: 7705122.5000
Epoch 131/150
965/965 [==============================] - 4s 5ms/step - loss: 7699668.0000
Epoch 132/150
965/965 [==============================] - 5s 6ms/step - loss: 7696206.5000
Epoch 133/150
965/965 [==============================] - 4s 5ms/step - loss: 7691608.0000
Epoch 134/150
965/965 [==============================] - 5s 5ms/step - loss: 7687152.0000
Epoch 135/150
965/965 [==============================] - 5s 5ms/step - loss: 7681317.0000
Epoch 136/150
965/965 [==============================] - 4s 5ms/step - loss: 7677290.0000
Epoch 137/150
965/965 [==============================] - 5s 6ms/step - loss: 7673454.5000
Epoch 138/150
965/965 [==============================] - 4s 5ms/step - loss: 7667896.5000
Epoch 139/150
965/965 [==============================] - 4s 5ms/step - loss: 7664659.5000
Epoch 140/150
965/965 [==============================] - 5s 6ms/step - loss: 7659515.0000
Epoch 141/150
965/965 [==============================] - 5s 5ms/step - loss: 7655173.5000
Epoch 142/150
965/965 [==============================] - 5s 6ms/step - loss: 7650776.5000
Epoch 143/150
```

```
965/965 [==============================] - 5s 5ms/step - loss: 7645352.0000
Epoch 144/150
965/965 [==============================] - 4s 5ms/step - loss: 7641648.0000
Epoch 145/150
965/965 [==============================] - 5s 6ms/step - loss: 7637246.0000
Epoch 146/150
965/965 [==============================] - 4s 5ms/step - loss: 7633182.0000
Epoch 147/150
965/965 [==============================] - 5s 5ms/step - loss: 7628160.5000
Epoch 148/150
965/965 [==============================] - 5s 5ms/step - loss: 7624115.5000
Epoch 149/150
965/965 [==============================] - 4s 5ms/step - loss: 7619110.5000
Epoch 150/150
965/965 [==============================] - 6s 6ms/step - loss: 7615818.5000

WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

Epoch 1/30
965/965 [==============================] - 7s 5ms/step - loss: 1808.2083
Epoch 2/30
965/965 [==============================] - 5s 5ms/step - loss: 234.9438
Epoch 3/30
965/965 [==============================] - 4s 5ms/step - loss: 176.8414
Epoch 4/30
965/965 [==============================] - 5s 6ms/step - loss: 150.4218
Epoch 5/30
965/965 [==============================] - 5s 5ms/step - loss: 132.3914
Epoch 6/30
965/965 [==============================] - 4s 5ms/step - loss: 124.4236
Epoch 7/30
965/965 [==============================] - 5s 6ms/step - loss: 120.7148
Epoch 8/30
965/965 [==============================] - 4s 5ms/step - loss: 118.3600
Epoch 9/30
965/965 [==============================] - 6s 6ms/step - loss: 116.7449
Epoch 10/30
965/965 [==============================] - 5s 5ms/step - loss: 115.4196
Epoch 11/30
965/965 [==============================] - 5s 5ms/step - loss: 114.4296
Epoch 12/30
965/965 [==============================] - 5s 6ms/step - loss: 113.5574
Epoch 13/30
965/965 [==============================] - 4s 5ms/step - loss: 112.8764
Epoch 14/30
965/965 [==============================] - 4s 5ms/step - loss: 112.0536
Epoch 15/30
965/965 [==============================] - 6s 6ms/step - loss: 111.5052
```

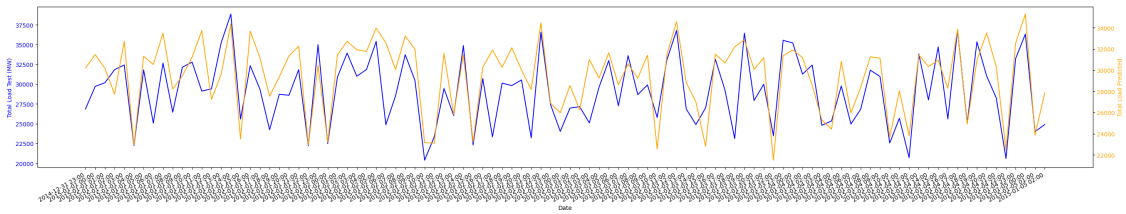
```
Epoch 16/30
965/965 [==============================] - 4s 5ms/step - loss: 110.9802
Epoch 17/30
965/965 [==============================] - 6s 6ms/step - loss: 110.4379
Epoch 18/30
965/965 [==============================] - 4s 5ms/step - loss: 109.9545
Epoch 19/30
965/965 [==============================] - 4s 5ms/step - loss: 109.4888
Epoch 20/30
965/965 [==============================] - 6s 6ms/step - loss: 108.9708
Epoch 21/30
965/965 [==============================] - 4s 5ms/step - loss: 108.6124
Epoch 22/30
965/965 [==============================] - 5s 5ms/step - loss: 108.2162
Epoch 23/30
965/965 [==============================] - 5s 5ms/step - loss: 107.8305
Epoch 24/30
965/965 [==============================] - 4s 5ms/step - loss: 107.4932
Epoch 25/30
965/965 [==============================] - 5s 6ms/step - loss: 107.2165
Epoch 26/30
965/965 [==============================] - 4s 5ms/step - loss: 106.7778
Epoch 27/30
965/965 [==============================] - 4s 5ms/step - loss: 106.4995
Epoch 28/30
965/965 [==============================] - 5s 6ms/step - loss: 106.2498
Epoch 29/30
965/965 [==============================] - 4s 5ms/step - loss: 105.8680
Epoch 30/30
965/965 [==============================] - 5s 6ms/step - loss: 105.6578
242/242 [==============================] - 1s 2ms/step
242/242 [==============================] - 1s 2ms/step
```

[31]:
```
mse_load = mean_squared_error(y_load_test, y_load_pred)
rmse_load = sqrt(mse_load)
mae_load = mean_absolute_error(y_load_test, y_load_pred)
r2_load = r2_score(y_load_test, y_load_pred)

mse_price = mean_squared_error(y_price_test, y_price_pred)
rmse_price = sqrt(mse_price)
mae_price = mean_absolute_error(y_price_test, y_price_pred)
r2_price = r2_score(y_price_test, y_price_pred)

print('Load Prediction RMSE:', rmse_load)
print('Load Prediction MAE:', mae_load)
print('Load Prediction R²:', r2_load)
```

```
print('Price Prediction RMSE:', rmse_price)
print('Price Prediction MAE:', mae_price)
print('Price Prediction R²:', r2_price)
```

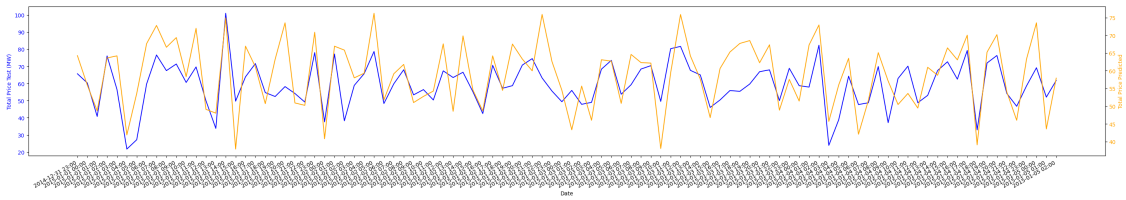
```
Load Prediction RMSE: 2780.5586216102947
Load Prediction MAE: 2083.865261241511
Load Prediction R²: 0.6387373880618912
Price Prediction RMSE: 10.395138419270296
Price Prediction MAE: 8.187366886618596
Price Prediction R²: 0.46674446284165594
```

[32]:
```
dates = [str(date)[:16] for date in merged_dataframe.index][:100]
test = y_load_test[:100]
pred = y_load_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Load Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Total Load Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



[33]:
```
dates = [str(date)[:16] for date in merged_dataframe.index][:100]
test = y_price_test[:100]
pred = y_price_pred[:100]

fig, ax1 = plt.subplots()
```

```
ax1.set_xlabel('Date')
ax1.set_ylabel('Total Price Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Total Price Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



5.2.1 Experimental Features Added Model - LSTM

[34]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from math import sqrt
from sklearn.metrics import mean_squared_error

dataframe = merged_dataframe.copy()

dataframe['hour'] = dataframe.index.hour
dataframe['day_of_week'] = dataframe.index.dayofweek
dataframe['month'] = dataframe.index.month

features = [
    'temp_Barcelona',
    'humidity_Barcelona',
    'wind_speed_Barcelona',
```

```
    'hour',
    'day_of_week',
    'month',
    'total load forecast',
    'price day ahead'
]

X_load = dataframe[features]
y_load = dataframe['total load actual']
X_price = dataframe[features]
y_price = dataframe['price actual']

scaler = StandardScaler()
X_load_scaled = scaler.fit_transform(X_load)
X_price_scaled = scaler.fit_transform(X_price)

X_load_scaled = X_load_scaled.reshape(X_load_scaled.shape[0], 1, X_load_scaled.
 ↪shape[1])
X_price_scaled = X_price_scaled.reshape(X_price_scaled.shape[0], 1,␣
 ↪X_price_scaled.shape[1])

# Splitting the data
X_load_train, X_load_test, y_load_train, y_load_test =␣
 ↪train_test_split(X_load_scaled, y_load, test_size=0.2, random_state=42)
X_price_train, X_price_test, y_price_train, y_price_test =␣
 ↪train_test_split(X_price_scaled, y_price, test_size=0.2, random_state=42)

# Defining the LSTM model
def build_model(input_shape):
    model = Sequential()
    model.add(LSTM(50, activation='relu', input_shape=input_shape))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model

model_load = build_model((1, len(features)))
model_load.fit(X_load_train, y_load_train, epochs=150, batch_size=32, verbose=1)

model_price = build_model((1, len(features)))
model_price.fit(X_price_train, y_price_train, epochs=30, batch_size=32,␣
 ↪verbose=1)

y_load_pred = model_load.predict(X_load_test)
y_price_pred = model_price.predict(X_price_test)
```

```
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

```
Epoch 1/150
965/965 [==============================] - 6s 5ms/step - loss: 843243520.0000
Epoch 2/150
965/965 [==============================] - 5s 5ms/step - loss: 824747200.0000
Epoch 3/150
965/965 [==============================] - 4s 4ms/step - loss: 794040448.0000
Epoch 4/150
965/965 [==============================] - 5s 6ms/step - loss: 754900544.0000
Epoch 5/150
965/965 [==============================] - 4s 5ms/step - loss: 709685376.0000
Epoch 6/150
965/965 [==============================] - 4s 5ms/step - loss: 660062464.0000
Epoch 7/150
965/965 [==============================] - 5s 5ms/step - loss: 607299648.0000
Epoch 8/150
965/965 [==============================] - 4s 5ms/step - loss: 552708672.0000
Epoch 9/150
965/965 [==============================] - 5s 5ms/step - loss: 497497600.0000
Epoch 10/150
965/965 [==============================] - 5s 5ms/step - loss: 442970080.0000
Epoch 11/150
965/965 [==============================] - 4s 5ms/step - loss: 390311872.0000
Epoch 12/150
965/965 [==============================] - 5s 6ms/step - loss: 340556864.0000
Epoch 13/150
965/965 [==============================] - 4s 5ms/step - loss: 294628320.0000
Epoch 14/150
965/965 [==============================] - 4s 5ms/step - loss: 253214432.0000
Epoch 15/150
965/965 [==============================] - 5s 6ms/step - loss: 216628176.0000
Epoch 16/150
965/965 [==============================] - 4s 5ms/step - loss: 184854752.0000
Epoch 17/150
965/965 [==============================] - 5s 5ms/step - loss: 157490384.0000
Epoch 18/150
965/965 [==============================] - 5s 5ms/step - loss: 133793712.0000
Epoch 19/150
965/965 [==============================] - 4s 5ms/step - loss: 113339392.0000
Epoch 20/150
965/965 [==============================] - 6s 6ms/step - loss: 95409064.0000
Epoch 21/150
965/965 [==============================] - 5s 5ms/step - loss: 79476400.0000
Epoch 22/150
965/965 [==============================] - 4s 5ms/step - loss: 66037836.0000
Epoch 23/150
965/965 [==============================] - 5s 6ms/step - loss: 53720412.0000
Epoch 24/150
965/965 [==============================] - 4s 5ms/step - loss: 43173800.0000
```

```
Epoch 25/150
965/965 [==============================] - 4s 5ms/step - loss: 34864988.0000
Epoch 26/150
965/965 [==============================] - 5s 6ms/step - loss: 28257098.0000
Epoch 27/150
965/965 [==============================] - 4s 4ms/step - loss: 22454118.0000
Epoch 28/150
965/965 [==============================] - 5s 6ms/step - loss: 17577678.0000
Epoch 29/150
965/965 [==============================] - 4s 5ms/step - loss: 13748504.0000
Epoch 30/150
965/965 [==============================] - 4s 5ms/step - loss: 10632154.0000
Epoch 31/150
965/965 [==============================] - 5s 6ms/step - loss: 8096102.5000
Epoch 32/150
965/965 [==============================] - 4s 5ms/step - loss: 6094292.0000
Epoch 33/150
965/965 [==============================] - 5s 5ms/step - loss: 4508940.5000
Epoch 34/150
965/965 [==============================] - 5s 5ms/step - loss: 3252093.2500
Epoch 35/150
965/965 [==============================] - 4s 5ms/step - loss: 2335523.0000
Epoch 36/150
965/965 [==============================] - 5s 6ms/step - loss: 1704043.1250
Epoch 37/150
965/965 [==============================] - 4s 5ms/step - loss: 1275917.7500
Epoch 38/150
965/965 [==============================] - 4s 5ms/step - loss: 991510.1875
Epoch 39/150
965/965 [==============================] - 5s 6ms/step - loss: 796972.0625
Epoch 40/150
965/965 [==============================] - 4s 5ms/step - loss: 660292.6250
Epoch 41/150
965/965 [==============================] - 5s 5ms/step - loss: 561779.2500
Epoch 42/150
965/965 [==============================] - 5s 5ms/step - loss: 489914.3750
Epoch 43/150
965/965 [==============================] - 4s 5ms/step - loss: 437159.3438
Epoch 44/150
965/965 [==============================] - 6s 6ms/step - loss: 397652.4062
Epoch 45/150
965/965 [==============================] - 4s 5ms/step - loss: 367084.3125
Epoch 46/150
965/965 [==============================] - 4s 5ms/step - loss: 343440.5000
Epoch 47/150
965/965 [==============================] - 5s 6ms/step - loss: 324615.4375
Epoch 48/150
965/965 [==============================] - 4s 5ms/step - loss: 309983.4375
```

```
Epoch 49/150
965/965 [==============================] - 5s 6ms/step - loss: 298292.2188
Epoch 50/150
965/965 [==============================] - 4s 5ms/step - loss: 288735.9062
Epoch 51/150
965/965 [==============================] - 4s 5ms/step - loss: 280821.2188
Epoch 52/150
965/965 [==============================] - 5s 6ms/step - loss: 273847.9062
Epoch 53/150
965/965 [==============================] - 4s 5ms/step - loss: 267865.5000
Epoch 54/150
965/965 [==============================] - 5s 5ms/step - loss: 262537.0938
Epoch 55/150
965/965 [==============================] - 5s 5ms/step - loss: 257806.7344
Epoch 56/150
965/965 [==============================] - 4s 5ms/step - loss: 253638.0156
Epoch 57/150
965/965 [==============================] - 5s 6ms/step - loss: 250011.3281
Epoch 58/150
965/965 [==============================] - 5s 5ms/step - loss: 246549.0781
Epoch 59/150
965/965 [==============================] - 5s 5ms/step - loss: 243494.3125
Epoch 60/150
965/965 [==============================] - 5s 6ms/step - loss: 240796.6719
Epoch 61/150
965/965 [==============================] - 5s 5ms/step - loss: 238335.3906
Epoch 62/150
965/965 [==============================] - 6s 6ms/step - loss: 236315.7344
Epoch 63/150
965/965 [==============================] - 6s 6ms/step - loss: 234347.5625
Epoch 64/150
965/965 [==============================] - 5s 5ms/step - loss: 232455.8438
Epoch 65/150
965/965 [==============================] - 6s 6ms/step - loss: 230755.6094
Epoch 66/150
965/965 [==============================] - 4s 5ms/step - loss: 229237.6406
Epoch 67/150
965/965 [==============================] - 5s 5ms/step - loss: 227836.8594
Epoch 68/150
965/965 [==============================] - 6s 6ms/step - loss: 226588.7656
Epoch 69/150
965/965 [==============================] - 5s 5ms/step - loss: 225507.7344
Epoch 70/150
965/965 [==============================] - 5s 6ms/step - loss: 224316.6406
Epoch 71/150
965/965 [==============================] - 5s 5ms/step - loss: 223218.8281
Epoch 72/150
965/965 [==============================] - 5s 5ms/step - loss: 222442.3281
```

```
Epoch 73/150
965/965 [==============================] - 5s 6ms/step - loss: 221485.7812
Epoch 74/150
965/965 [==============================] - 5s 5ms/step - loss: 220664.1875
Epoch 75/150
965/965 [==============================] - 5s 5ms/step - loss: 219922.0000
Epoch 76/150
965/965 [==============================] - 5s 5ms/step - loss: 219025.1875
Epoch 77/150
965/965 [==============================] - 5s 5ms/step - loss: 218421.5156
Epoch 78/150
965/965 [==============================] - 6s 6ms/step - loss: 217814.7656
Epoch 79/150
965/965 [==============================] - 5s 5ms/step - loss: 217113.8438
Epoch 80/150
965/965 [==============================] - 5s 5ms/step - loss: 216563.1719
Epoch 81/150
965/965 [==============================] - 5s 6ms/step - loss: 215955.4688
Epoch 82/150
965/965 [==============================] - 5s 5ms/step - loss: 215584.7969
Epoch 83/150
965/965 [==============================] - 6s 6ms/step - loss: 214929.6406
Epoch 84/150
965/965 [==============================] - 5s 5ms/step - loss: 214618.8438
Epoch 85/150
965/965 [==============================] - 5s 5ms/step - loss: 214079.9375
Epoch 86/150
965/965 [==============================] - 6s 6ms/step - loss: 213619.7188
Epoch 87/150
965/965 [==============================] - 5s 5ms/step - loss: 213201.8906
Epoch 88/150
965/965 [==============================] - 5s 5ms/step - loss: 212904.7812
Epoch 89/150
965/965 [==============================] - 5s 5ms/step - loss: 212465.7969
Epoch 90/150
965/965 [==============================] - 5s 5ms/step - loss: 212116.6406
Epoch 91/150
965/965 [==============================] - 5s 6ms/step - loss: 211781.0625
Epoch 92/150
965/965 [==============================] - 4s 5ms/step - loss: 211362.9844
Epoch 93/150
965/965 [==============================] - 5s 5ms/step - loss: 211088.5156
Epoch 94/150
965/965 [==============================] - 5s 5ms/step - loss: 210695.9062
Epoch 95/150
965/965 [==============================] - 4s 5ms/step - loss: 210572.7188
Epoch 96/150
965/965 [==============================] - 6s 6ms/step - loss: 210205.7969
```

```
Epoch 97/150
965/965 [==============================] - 4s 5ms/step - loss: 209856.2500
Epoch 98/150
965/965 [==============================] - 4s 5ms/step - loss: 209625.3750
Epoch 99/150
965/965 [==============================] - 5s 6ms/step - loss: 209187.1094
Epoch 100/150
965/965 [==============================] - 5s 5ms/step - loss: 209102.5938
Epoch 101/150
965/965 [==============================] - 5s 6ms/step - loss: 208757.3438
Epoch 102/150
965/965 [==============================] - 5s 5ms/step - loss: 208585.6719
Epoch 103/150
965/965 [==============================] - 5s 5ms/step - loss: 208220.9688
Epoch 104/150
965/965 [==============================] - 6s 7ms/step - loss: 208009.1562
Epoch 105/150
965/965 [==============================] - 4s 5ms/step - loss: 207979.5625
Epoch 106/150
965/965 [==============================] - 5s 5ms/step - loss: 207646.0156
Epoch 107/150
965/965 [==============================] - 5s 5ms/step - loss: 207521.6562
Epoch 108/150
965/965 [==============================] - 4s 5ms/step - loss: 207171.2969
Epoch 109/150
965/965 [==============================] - 6s 6ms/step - loss: 206976.9688
Epoch 110/150
965/965 [==============================] - 4s 5ms/step - loss: 206917.7656
Epoch 111/150
965/965 [==============================] - 4s 5ms/step - loss: 206716.9219
Epoch 112/150
965/965 [==============================] - 6s 6ms/step - loss: 206467.8281
Epoch 113/150
965/965 [==============================] - 5s 5ms/step - loss: 206299.3750
Epoch 114/150
965/965 [==============================] - 5s 6ms/step - loss: 206166.9062
Epoch 115/150
965/965 [==============================] - 5s 5ms/step - loss: 205966.2812
Epoch 116/150
965/965 [==============================] - 4s 5ms/step - loss: 205749.4688
Epoch 117/150
965/965 [==============================] - 5s 6ms/step - loss: 205658.1406
Epoch 118/150
965/965 [==============================] - 4s 5ms/step - loss: 205564.5312
Epoch 119/150
965/965 [==============================] - 5s 5ms/step - loss: 205343.7656
Epoch 120/150
965/965 [==============================] - 5s 6ms/step - loss: 205137.5000
```

```
Epoch 121/150
965/965 [==============================] - 5s 5ms/step - loss: 205039.2969
Epoch 122/150
965/965 [==============================] - 5s 6ms/step - loss: 204777.0781
Epoch 123/150
965/965 [==============================] - 4s 5ms/step - loss: 204800.2500
Epoch 124/150
965/965 [==============================] - 4s 5ms/step - loss: 204564.2812
Epoch 125/150
965/965 [==============================] - 5s 6ms/step - loss: 204270.2344
Epoch 126/150
965/965 [==============================] - 4s 5ms/step - loss: 204337.0469
Epoch 127/150
965/965 [==============================] - 5s 5ms/step - loss: 204227.0625
Epoch 128/150
965/965 [==============================] - 5s 5ms/step - loss: 204120.0625
Epoch 129/150
965/965 [==============================] - 5s 5ms/step - loss: 203995.8125
Epoch 130/150
965/965 [==============================] - 5s 6ms/step - loss: 203730.9844
Epoch 131/150
965/965 [==============================] - 4s 5ms/step - loss: 203668.0156
Epoch 132/150
965/965 [==============================] - 4s 5ms/step - loss: 203449.6250
Epoch 133/150
965/965 [==============================] - 5s 6ms/step - loss: 203501.8906
Epoch 134/150
965/965 [==============================] - 5s 5ms/step - loss: 203344.5469
Epoch 135/150
965/965 [==============================] - 6s 6ms/step - loss: 203290.4219
Epoch 136/150
965/965 [==============================] - 4s 5ms/step - loss: 203173.8125
Epoch 137/150
965/965 [==============================] - 4s 5ms/step - loss: 203003.9531
Epoch 138/150
965/965 [==============================] - 5s 6ms/step - loss: 202896.5000
Epoch 139/150
965/965 [==============================] - 4s 5ms/step - loss: 202818.3594
Epoch 140/150
965/965 [==============================] - 5s 5ms/step - loss: 202843.9219
Epoch 141/150
965/965 [==============================] - 5s 5ms/step - loss: 202523.2656
Epoch 142/150
965/965 [==============================] - 5s 5ms/step - loss: 202384.7188
Epoch 143/150
965/965 [==============================] - 5s 6ms/step - loss: 202495.9375
Epoch 144/150
965/965 [==============================] - 5s 5ms/step - loss: 202273.1406
```

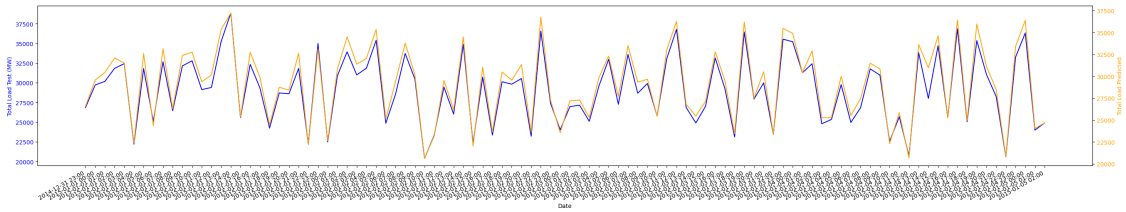
```
Epoch 145/150
965/965 [==============================] - 4s 5ms/step - loss: 202309.8281
Epoch 146/150
965/965 [==============================] - 7s 7ms/step - loss: 202030.1719
Epoch 147/150
965/965 [==============================] - 4s 5ms/step - loss: 202081.2500
Epoch 148/150
965/965 [==============================] - 5s 5ms/step - loss: 201805.0938
Epoch 149/150
965/965 [==============================] - 5s 5ms/step - loss: 201886.2656
Epoch 150/150
965/965 [==============================] - 5s 5ms/step - loss: 201685.2344

WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

Epoch 1/30
965/965 [==============================] - 7s 5ms/step - loss: 1577.5231
Epoch 2/30
965/965 [==============================] - 5s 5ms/step - loss: 176.6387
Epoch 3/30
965/965 [==============================] - 5s 5ms/step - loss: 106.2749
Epoch 4/30
965/965 [==============================] - 5s 6ms/step - loss: 86.5514
Epoch 5/30
965/965 [==============================] - 5s 5ms/step - loss: 76.9268
Epoch 6/30
965/965 [==============================] - 5s 5ms/step - loss: 72.1151
Epoch 7/30
965/965 [==============================] - 5s 6ms/step - loss: 69.3468
Epoch 8/30
965/965 [==============================] - 4s 5ms/step - loss: 67.3499
Epoch 9/30
965/965 [==============================] - 5s 6ms/step - loss: 65.6609
Epoch 10/30
965/965 [==============================] - 5s 5ms/step - loss: 64.1747
Epoch 11/30
965/965 [==============================] - 4s 5ms/step - loss: 62.8679
Epoch 12/30
965/965 [==============================] - 5s 6ms/step - loss: 61.6571
Epoch 13/30
965/965 [==============================] - 5s 5ms/step - loss: 60.7111
Epoch 14/30
965/965 [==============================] - 5s 6ms/step - loss: 59.8158
Epoch 15/30
965/965 [==============================] - 5s 5ms/step - loss: 59.0259
Epoch 16/30
965/965 [==============================] - 5s 5ms/step - loss: 58.3575
Epoch 17/30
```

```
965/965 [==============================] - 6s 6ms/step - loss: 57.7680
Epoch 18/30
965/965 [==============================] - 5s 5ms/step - loss: 57.2156
Epoch 19/30
965/965 [==============================] - 5s 5ms/step - loss: 56.7431
Epoch 20/30
965/965 [==============================] - 5s 5ms/step - loss: 56.3262
Epoch 21/30
965/965 [==============================] - 5s 5ms/step - loss: 55.8964
Epoch 22/30
965/965 [==============================] - 6s 6ms/step - loss: 55.5548
Epoch 23/30
965/965 [==============================] - 4s 5ms/step - loss: 55.1401
Epoch 24/30
965/965 [==============================] - 5s 5ms/step - loss: 54.7848
Epoch 25/30
965/965 [==============================] - 5s 6ms/step - loss: 54.4886
Epoch 26/30
965/965 [==============================] - 5s 5ms/step - loss: 54.1969
Epoch 27/30
965/965 [==============================] - 6s 6ms/step - loss: 53.8992
Epoch 28/30
965/965 [==============================] - 4s 5ms/step - loss: 53.6641
Epoch 29/30
965/965 [==============================] - 5s 5ms/step - loss: 53.3988
Epoch 30/30
965/965 [==============================] - 6s 6ms/step - loss: 53.1133
242/242 [==============================] - 1s 3ms/step
242/242 [==============================] - 1s 3ms/step
```

[35]:
```
mse_load = mean_squared_error(y_load_test, y_load_pred)
rmse_load = sqrt(mse_load)
mae_load = mean_absolute_error(y_load_test, y_load_pred)
r2_load = r2_score(y_load_test, y_load_pred)

mse_price = mean_squared_error(y_price_test, y_price_pred)
rmse_price = sqrt(mse_price)
mae_price = mean_absolute_error(y_price_test, y_price_pred)
r2_price = r2_score(y_price_test, y_price_pred)

print('Load Prediction RMSE:', rmse_load)
print('Load Prediction MAE:', mae_load)
print('Load Prediction R²:', r2_load)

print('Price Prediction RMSE:', rmse_price)
print('Price Prediction MAE:', mae_price)
print('Price Prediction R²:', r2_price)
```

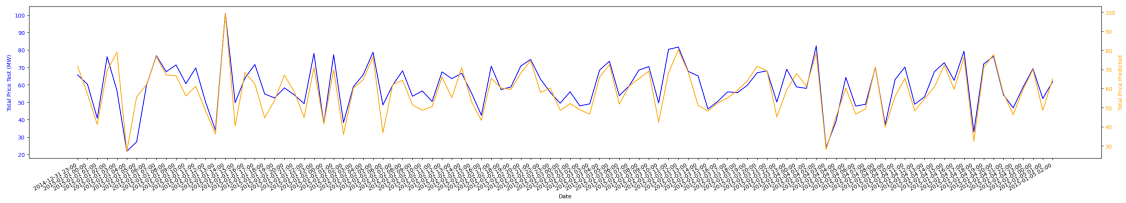
```
Load Prediction RMSE: 444.6452474466085
Load Prediction MAE: 325.5630641404607
Load Prediction R²: 0.9907618243407674
Price Prediction RMSE: 7.392546424026405
Price Prediction MAE: 5.102001977430467
Price Prediction R²: 0.7303111810386816
```

[36]:
```
dates = [str(date)[:16] for date in merged_dataframe.index][:100]
test = y_load_test[:100]
pred = y_load_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Load Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Total Load Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



[37]:
```
dates = [str(date)[:16] for date in merged_dataframe.index][:100]
test = y_price_test[:100]
pred = y_price_pred[:100]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Total Price Test (MW)', color='blue')
ax1.plot(dates, test, color='blue')
```

```
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('Total Price Predicted', color='orange')
ax2.plot(dates, pred, color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

fig.tight_layout()
fig.autofmt_xdate()
fig.set_size_inches(35, 5, forward=True)

plt.show()
```



5.3 Exporting Models Created

[38]:
```
import pickle

with open('rf_load.pkl', 'wb') as file:
    pickle.dump(regressor_load, file)

with open('rf_price.pkl', 'wb') as file:
    pickle.dump(regressor_price, file)

model_load.save('lstm_load_model.h5')
model_price.save('lstm_price_model.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning:

You are saving your model as an HDF5 file via `model.save()`. This file format
is considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
```

6 Export to PDF

[43]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
!jupyter nbconvert --to pdf '/content/Forecasting the Future: Electricity Load␣
↪and Price Prediction Based on Weather.ipynb'
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
texlive-fonts-recommended is already the newest version (2021.20220204-1).
texlive-plain-generic is already the newest version (2021.20220204-1).
texlive-xetex is already the newest version (2021.20220204-1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
[NbConvertApp] Converting notebook /content/Forecasting the Future: Electricity
Load and Price Prediction Based on Weather.ipynb to pdf
/usr/local/lib/python3.10/dist-packages/nbconvert/filters/datatypefilter.py:41:
UserWarning: Your element with mimetype(s) dict_keys(['text/html']) is not able
to be represented.
  warn(
[NbConvertApp] Support files will be in Forecasting the Future: Electricity Load
and Price Prediction Based on Weather_files/
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Making directory ./Forecasting the Future: Electricity Load and
Price Prediction Based on Weather_files
[NbConvertApp] Writing 179664 bytes to notebook.tex
```

```
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 2007251 bytes to /content/Forecasting the Future:
Electricity Load and Price Prediction Based on Weather.pdf
```