FEATURE ENGINEERING

```
import pandas as pd

# Load your dataset into a Pandas Data Frame
df = pd.read_csv('your_dataset.csv')

# Convert the timestamp to a datetime object (assuming it's in a standard format)
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Extract time-related features
df['hour'] = df['timestamp'].dt.hour

df['day_of_week'] = df['timestamp'].dt.dayofweek

df['month'] = df['timestamp'].dt.month

df['year'] = df['timestamp'].dt.year

# Lag features (past prices)
for lag in range (1, 25):

df[f'lag_{lag}'] = df['price']. shift(lag)

# Rolling statistics (e.g., moving averages)
df['rolling_mean_7'] = df['price']. rolling(window=7).mean()

df['rolling_std_7'] = df['price']. rolling(window=7). std ()

# Weather data (if available)
# You can merge weather data based on a common timestamp

# df = pd.merge(df, weather_data, on='timestamp', how='left')

# Market indicators (if available)
# df = pd.merge(df, market_data, on='timestamp', how='left')
```

```
# Drop rows with missing values created by lag and rolling features
df = df.dropna()


# Optionally, you can scale/normalize features
# from sklearn.preprocessing import StandardScaler

# scaler = StandardScaler()

# df[feature_columns] = scaler.fit_transform(df[feature_columns])


# Now, you have an enriched dataset with engineered features for modeling
```

In this code:

timestamp is assumed to be the column containing the timestamp.

Features like hour, day_of_week, month, and year are extracted from the timestamp to capture time-related patterns.

Lag features represent past prices, which can help capture autocorrelation in the time series data.

Rolling statistics like the moving average and standard deviation can help smooth the data and capture trends.

Weather data and market indicators can be added to the dataset if available.

TRAINING:

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Assuming 'X' is your feature matrix and 'y' is the target variable (electricity prices)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = LinearRegression()

model.fit(X_train, y_train)


# Make predictions
y_pred = model.predict(X_test)
```

```python
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

print (f"Root Mean Squared Error: {rmse}")

# Import necessary libraries
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, mean_absolute_error


# Load and preprocess the dataset
# Replace 'X' and 'y' with your feature matrix and target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Initialize and train the model
model = LinearRegression()

model.fit(X_train, y_train)


# Make predictions
y_pred = model.predict(X_test)


# Evaluate the model
Mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)


print(f"Mean Absolute Error: {mae}")

print (f"Mean Squared Error: {mse}")

print (f"Root Mean Squared Error: {rmse}")
```