ELECTRICITY PRICE PREDICTION

Building an electricity price prediction model is a complex task that typically involves several steps, including data preprocessing, feature engineering, model training, and evaluation. I can provide a general outline of how you might approach this task, but keep in mind that the specific techniques and models you should use can vary based on the dataset and the goals of your prediction.

Here's a step-by-step guide to continue building your electricity price prediction model:

1. Data Preprocessing:

Load the dataset from the provided link (Kaggle).

Inspect the dataset to understand its structure, including the columns, data types, and any missing values.

Handle missing data: Depending on the dataset, you might need to impute missing values or remove rows with missing data.

Convert date/time features to datetime objects.

Explore the data distribution and check for outliers.

2. Feature Engineering:

Create new features that can help improve the model's performance. For electricity price prediction, potential features might include:

Time-based features: hour of the day, day of the week, month, etc.

Lag features: historical electricity prices at different time intervals.

Weather-related features: temperature, humidity, and other relevant weather data.

Calendar-based features: holidays, special events, and seasonal indicators.

Normalize or scale features if needed.

Consider using domain knowledge to engineer features that are relevant to electricity price behavior.

3. Split the Data:
Divide the dataset into training, validation, and test sets. A common split ratio is 70% for training, 15% for validation, and 15% for testing.

4. Model Selection:
Choose an appropriate machine learning or time series forecasting model for electricity price prediction. Some common models include:

- Linear Regression
- Decision Trees
- Random Forest
- XGBoost
- LSTM or other recurrent neural networks for time series data

Experiment with different models to find the one that works best for your dataset.

5. Model Training:
Train your selected model on the training dataset using the engineered features.
Tune hyperparameters to optimize model performance. Consider using techniques like cross-validation and grid search for hyperparameter tuning.

6. Model Evaluation:
Evaluate your model's performance on the validation set using appropriate evaluation metrics. Common metrics for regression tasks include:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R-squared (R2) score

Visualize the model's predictions against the actual electricity prices to understand how well it's performing.

7. Model Fine-Tuning and Validation:
Based on the validation results, fine-tune your model as needed. This may include adjusting hyperparameters, adding more features, or trying different algorithms.

8. Final Testing:
Once you are satisfied with your model's performance on the validation set, evaluate it on the test set to assess its real-world performance.

Dataset Link:
https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

Here's a Python program that demonstrates the basic steps for your electricity price prediction task:

Program:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

data = pd.read_csv('Electricity.csv')

# Data Preprocessing
# Specify the date format for the 'DateTime' column
data['DateTime'] = pd.to_datetime(data['DateTime'],
format='%d/%m/%Y %H:%M')
```

```
# Handle non-numeric values (replace '?' with NaN)
data = data.replace('?', np.nan)

# Convert numeric columns to float
data = data.apply(pd.to_numeric, errors='coerce')

# Drop rows with missing values
data = data.dropna()

# Feature Engineering
# Select relevant numeric features for model training
numeric_features = ['DayOfWeek', 'WeekOfYear', 'Day', 'Month',
'Year', 'PeriodOfDay', 'ORKTemperature', 'ORKWindspeed',
'CO2Intensity']
X = data[numeric_features]  # Features
y = data['SystemLoadEA']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.10, random_state=42)

# Model Training
model = LinearRegression()
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)

# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared (R2) Score: {r2}")
```

Use Cross-Validation: When dealing with a small dataset, you can consider using cross-validation techniques, such as k-fold cross-validation, to make the most of the available data while evaluating your model's performance.

Program:

```
from sklearn.model_selection import cross_val_score

# Define your model (e.g., LinearRegression)
model = LinearRegression()

# Perform 5-fold cross-validation
scores = cross_val_score(model, X, y, cv=5,
scoring='neg_mean_squared_error')
mse = -scores.mean()  # Calculate the mean of negative MSE
values

print(f"Mean Squared Error (Cross-Validation): {mse}")
```

This script will load the dataset, preprocess it using the specified columns, and perform electricity price prediction based on the 'SystemLoadEA' column.

Overall, this program aims to predict electricity prices using a linear regression model and provides an evaluation of the model's accuracy.