ELECTRICITY PRICES PREDICTION

PHASE 2

STEPS INVOLVED:

Step 1: Data Collection

Dataset available: https://skilluptechmy.sharepoint.com/:b:/g/personal/simrang_skillup_online/EZjznEGC5VVAvWNefQ-uERkBeFLCyboV4mAuleYE3ENxLA?e=2fqSfm

Step 2: Data Preprocessing

- Correct missing values by deciding whether to impute them using methods such as mean, median, or predictive imputation, or remove incomplete data points.
- Check the dataset for duplicates and remove them if necessary.

Step 3: Feature Engineering

Some potential feature engineering ideas might include

- Extract time-based features from 'DateTime,' such as day of the week, month, and hour.
- Use 'Holiday' and 'HolidayFlag' columns to create a binary flag for holidays.
- For 'PeriodOfDay,' categorize time periods based on domain knowledge.
- A key feature was the creation of a 7-day rolling average of electricity prices to capture short-term trends.

Step 4: Data Splitting

- Splitting data into training and testing sets for model evaluation.

- This split is to assess the model's performance on unseen data and evaluate the model's performance by making predictions and comparing them to the actual values.
- The training set might consist of 70-80% of the data, while the test set is the remaining 20-30%.
- Split will likely be done chronologically to simulate real-world forecasting conditions.
- In python:

```
from sklearn.model_selection import train_test_split

# Split your data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)
```

  In this code:
  X represents the feature data.
  y represents the target variable.
  test_size controls the proportion of data allocated to the test set.
  random_state is set for reproducibility. You can choose any integer value for it.
  After splitting the data, you will have X_train, y_train for training, X_val, y_val for validation, and X_test, y_test for testing.

Step 5: Model Selection

- LSTM (Long Short-Term Memory) neural network due to its effectiveness in capturing sequential patterns in time series data.
- However, it could be changed to other time series models like ARIMA or LSTM, depending on the dataset's complexity.

Step 6: Model Training

- Adjust model hyperparameters such as ARIMA order parameters or LSTM architecture options to optimize performance.

Step 7: Model Evaluation

Evaluate the model's performance using relevant metrics like

- Mean Absolute Error

  MAE is calculated as the average of the absolute differences between the predicted and actual values. It measures the average magnitude of errors in the predictions. A lower MAE indicates better model accuracy

- Root Mean Squared Error

  RMSE is calculated as the square root of the average of the squared differences between predicted and actual values. It penalizes larger errors more heavily than MAE and is sensitive to outliers.

  Both MAE and RMSE provide insights into the accuracy of the model's predictions. A lower value for either of these metrics indicates that the model's predictions are closer to the actual values.

- R-squared

Tuning and Optimization:

- Techniques like Grid Search or Random Search is used to tune the LSTM model for a specific dataset using the previously defined hyperparameters.
  - Grid Search: This method involves defining a set of hyperparameters and exploring all possible combinations. It can be exhaustive but guarantees finding the best combination within the defined search space.
  - Random Search: In this method, you randomly sample from the hyperparameter space. While it may not guarantee the best combination, it is often more efficient than grid search and can discover good hyperparameters more quickly.

- Hyperparameter tuning is crucial because the choice of hyperparameters can significantly impact the accuracy of the model. The goal is to find the hyperparameters that result in the best predictions for the electricity price dataset.

Step 8: Apply the model

Apply the trained model to predict real-time or future prices.

Step 9: Management and Maintenance

- Continuously monitor model performance with metrics such as MAE and RMSE.
- Retrain the model with updated data to maintain accuracy and relevance to changing conditions.

Libraries likely to be necessary:

- Pandas:
    - Pandas is used for data manipulation, loading, and preprocessing.
    - It will be used to read the dataset and perform initial data processing.

- NumPy:
    - NumPy is used for numerical operations and working with arrays.
    - It's commonly used for data manipulation and transformations.

- Scikit-Learn:
    - Scikit-Learn provides tools for machine learning, including data splitting (train-test split), evaluation metrics, and preprocessing tools.
    - In the code provided, it's used for data splitting and calculating the Mean Absolute Error (MAE).

- Keras (with TensorFlow backend):
    - Keras is a high-level neural networks API that is often used with TensorFlow as the backend.
    - It's essential for defining and training deep learning models like LSTM.

- Matplotlib or Seaborn:
    - These libraries are used for data visualization.
    - They allow plot training and validation curves, model performance, and other visualizations.

- scipy.stats:
    - If decided to implement a random search for hyperparameter tuning, the scipy.stats library can be useful for defining probability distributions for hyperparameters.

The following is a simplified code structure:

```
# Import necessary libraries


# Load and preprocess the dataset

# - Load the dataset

# - Preprocess data, handle missing values, and perform feature engineering


# Split the data into training, validation, and test sets

# - Define the split proportions


# Define hyperparameters to search over

# - Units in the LSTM layer

# - Sequence length

# - Learning rate

# - Batch size
# Hyperparameter tuning loop

# - Loop over hyperparameter combinations

# - Build an LSTM model with the given hyperparameters

# - Train the model on the training data

# - Evaluate the model on the validation set using MAE

# - Keep track of the best model based on the lowest MAE


# Once the hyperparameter tuning is complete, you have the best model.


# Test the best model on the test dataset


# Make future predictions using the best model

# - Prepare input sequences for future predictions

# - Use the trained model to make predictions


# Visualize the results

# - Plot training and validation curves

# - Plot actual vs. predicted electricity prices


# Save the best model for future use


# Documentation and reporting

# - Document the hyperparameters and model settings
# - Generate a report with model performance and insights


# Additional considerations:

# - Data preprocessing functions (e.g., scaling, differencing)

# - Model architecture definition

# - Training loop with early stopping

# - Inverse transforming predictions
```