**ELECTRICITY PRICE PREDICTION**

**ABSTRACT:**

This study focuses on the development of a robust electricity price prediction model leveraging advanced machine learning techniques. The aim is to provide accurate forecasts that can enhance decision-making in energy markets. Historical electricity price data, incorporating temporal and external factors, is collected and preprocessed.

**PROBLEM STATEMENT :**

The problem is to develop a predictive model that uses historical electricity prices and relevant factors to forecast future electricity prices. The objective is to create a tool that assists both energy providers and consumers in making informed decisions regarding consumption and investment by predicting future electricity prices. This project involves data preprocessing, feature engineering, model selection, training, and evaluation

* Develop a machine learning model to predict electricity prices based on historical data, considering factors such as demand, weather conditions, and time of day.
* Explore the use of deep learning techniques to forecast short-term and long-term electricity prices, incorporating variables like market trends and renewable energy contributions.
* Investigate the impact of geopolitical events on electricity prices and build a predictive model to assess potential price fluctuations.
* Create a predictive analytics tool for energy traders by analyzing real-time data to forecast electricity prices, aiding in decision-making and risk management.
* Design a time series forecasting model to predict peak demand periods for electricity, enabling utilities to optimize resource allocation and prevent grid overloads.

**To predict electricity prices, you can follow these steps:**

**Data Collection:**

Gather historical data on electricity prices, considering factors like time of day, seasonality, weather conditions, and demand.

**Data Preprocessing:**

Clean and preprocess the data, handling missing values and outliers. Convert timestamps into a format suitable for time series analysis.

**Feature Engineering:**

Extract relevant features such as day of the week, time of day, holidays, and any external factors influencing electricity prices.

**Exploratory Data Analysis (EDA):**

Conduct exploratory analysis to understand patterns, trends, and correlations in the data. This step is crucial for selecting appropriate models.

**Model Selection:**

Choose a suitable model for time series forecasting. Common models include ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal ARIMA), or more advanced methods like Long Short-Term Memory (LSTM) networks for deep learning.

**Training the Model:**

Split the data into training and testing sets. Train the chosen model on the training data, adjusting parameters to optimize performance.

**Validation:**

Validate the model using the testing set to ensure it generalizes well to unseen data. Use metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) to evaluate performance.

**LONG SHORT TERM MEMORY (LSTM)**

LSTM is a type of recurrent neural network (RNN) architecture that has proven to be highly effective in handling sequences of data, making it particularly useful for time series forecasting, including electricity price prediction.

The working of LSTM is mentioned as the following,

1. MEMORY CELLS : Unlike traditional neural networks, LSTM networks have a more complex structure that includes memory cells. These cells can store information over long periods of time, allowing them to capture long-term dependencies in data.
2. GATES : LSTMs have three types of gates (input gate, forget gate, and output gate) that control the flow of information through the cell. These gates use sigmoid and tanh activation functions to regulate which information is passed through and which is discarded.
   * INPUT GATE : Determines how much of the new information is stored in the cell.
   * FORGET GATE: Controls which information should be discarded from the cell’s memory
   * OUTPUT GATE: Determines what part of the cell’s memory should be outputted.
3. SEQUENTIAL PROCESSING : LSTMs process data sequentially, which means they consider each element in the sequence one at a time. This allows them to capture complex temporal patterns in the data.
4. BACK PROPAGATION THROUGH TIME (BPTT) : LSTMs are trained using the backpropagation algorithm, which adjusts the weights of the network to minimize the prediction error. In the case of LSTMs, this process is extended over time to handle sequences.

**EFFECTS OF LSTM ON ELECTRICITY PRICES PREDICTION :**

* + CAPTURING LONG TERM DEPENDENCIES : LSTMs are designed to capture dependencies over long sequences,
  + which is crucial for modeling electricity price patterns influenced by various factor

-HANDLING SEASONING AND TRENDS : LSTMs can automatically learn and adapt to seasonal variations, trends, and other complex patterns present in electricity price data.

* + DYNAMIC PERFORMANCE : LSTMs are capable of adapting their internal state based on new information in the sequence, making them suitable for dynamic environments where patterns may change over time.
  + ROBUSTNESS TO IRREGULARITIES : LSTMs can handle missing data or irregularly sampled time series, which is common in real-world datasets.

It’s worth noting that while LSTMs are powerful, their effectiveness can be influenced by factors like data quality, feature engineering, and hyperparameter tuning. Therefore, careful preprocessing and experimentation are often crucial for achieving optimal results.

**DATA PREPROCESSING:**

The typical data preprocessing steps for building a model to predict electricity prices:

* Handling Missing Values:Identify and deal with any missing values in the dataset. This can involve techniques like imputation (filling in missing values with a calculated estimate) or removing rows with missing data.
* Outlier Detection:Identify and handle outliers in the data. Outliers can have a significant impact on the model’s performance. Techniques like z-score, IQR (Interquartile Range), or visualizations like box plots can be used to detect outliers.
* Data Scaling/Normalization:Depending on the algorithm you plan to use, it may be necessary to scale or normalize the data. For example, algorithms like Support Vector Machines and Neural Networks benefit from scaled data.
* Feature Selection:Choose the most relevant features for your model. This step is crucial as including irrelevant or redundant features can lead to overfitting.
* Handling Time Series Data:If your data involves time series (which is likely in the case of electricity prices), consider techniques like lagging to capture temporal patterns.
* Creating Additional Features: Generate new features if they might be informative for the model. For example, if you have data on holidays, create a binary feature indicating whether a given day is a holiday or not.
* Data Splitting:Divide the data into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate its performance.

The coding for electricity prices prediction using the LSTM (Long Short-Term Memory) algorithm. Here’s a basic outline in Python using the TensorFlow and Keras libraries:

# Import necessary libraries

Import numpy as np

Import pandas as pd

From sklearn.preprocessing import MinMaxScaler

From tensorflow.keras.models import Sequential

From tensorflow.keras.layers import LSTM, Dense

# Load your dataset

# Assuming you have a CSV file with a column ‘price’ representing the electricity prices

# Replace ‘your\_dataset.csv’ with the actual file path

Data = pd.read\_csv(‘your\_dataset.csv’)

Prices = data[‘price’].values.reshape(-1, 1)

# Normalize the data

Scaler = MinMaxScaler()

Prices = scaler.fit\_transform(prices)

# Create sequences for training

Sequence\_length = 10 # Adjust this based on your dataset and requirements

X, y = [], []

For I in range(len(prices) – sequence\_length):

X.append(prices[i:i+sequence\_length])

y.append(prices[i+sequence\_length])

X = np.array(X)

Y = np.array(y)

# Split data into training and testing sets

Train\_size = int(0.8 \* len(X))

X\_train, X\_test = X[:train\_size], X[train\_size:]

Y\_train, y\_test = y[:train\_size], y[train\_size:]

# Build the LSTM model

Model = Sequential([

LSTM(units=50, activation=’relu’, input\_shape=(sequence\_length, 1)),

Dense(units=1)

])

# Compile the model

Model.compile(optimizer=’adam’, loss=’mean\_squared\_error’)

# Train the model

Model.fit(X\_train, y\_train, epochs=50, batch\_size=32

# Evaluate the model

Loss = model.evaluate(X\_test, y\_test

# Predict future prices

Predicted\_prices = model.predict(X\_test)

# Inverse transform the predictions to get actual prices

Predicted\_prices = scaler.inverse\_transform(predicted\_prices)

**FEATURE ENGINEERING**

Feature engineering is a crucial step in building effective predictive models, especially for time series data like electricity prices

Import pandas as pd

# Load your dataset (assuming it has columns like ‘Date’ and ‘Price’)

Data = pd.read\_csv(‘electricity\_prices.csv’)

# Step 1: Extract Date Features

Data[‘Date’] = pd.to\_datetime(data[‘Date’])

Data[‘Year’] = data[‘Date’].dt.year

Data[‘Month’] = data[‘Date’].dt.month

Data[‘Day’] = data[‘Date’].dt.day

Data[‘Weekday’] = data[‘Date’].dt.weekday

# Step 2: Lag Features

Data[‘Price\_Lag1’] = data[‘Price’].shift(1) # Lag of 1 day

Data[‘Price\_Lag7’] = data[‘Price’].shift(7) # Lag of 1 week

# Step 3: Rolling Statistics

Data[‘Rolling\_Mean’] = data[‘Price’].rolling(window=7).mean() # 7-day rolling mean

Data[‘Rolling\_Std’] = data[‘Price’].rolling(window=7).std() # 7-day rolling standard deviation

# Step 4: Exponential Moving Average (EMA)

Data[‘EMA’] = data[‘Price’].ewm(span=7, adjust=False).mean() # 7-day EMA

# Step 5: Remove NaN values after feature engineering

Data = data.dropna()

# Save the engineered dataset

Data.to\_csv(‘engineered\_electricity\_prices.csv’, index=False)

**MODEL TRAINING**

To train your electricity prices prediction model using the engineered features, we can follow these steps:

Import pandas as pd

From sklearn.model\_selection import train\_test\_split

From sklearn.preprocessing import MinMaxScaler

From tensorflow.keras.models import Sequential

From tensorflow.keras.layers import LSTM, Dense

# Load your engineered dataset

Data = pd.read\_csv(‘engineered\_electricity\_prices.csv’)

# Define your features and target variable

X = data[[‘Year’, ‘Month’, ‘Day’, ‘Weekday’, ‘Price\_Lag1’, ‘Price\_Lag7’, ‘Rolling\_Mean’, ‘Rolling\_Std’, ‘EMA’]]

Y = data[‘Price’]

# Normalize the features

Scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Reshape the data for LSTM

X\_train = X\_train.reshape((X\_train.shape[0], 1, X\_train.shape[1]))

X\_test = X\_test.reshape((X\_test.shape[0], 1, X\_test.shape[1]))

# Build the LSTM model

Model = Sequential([

LSTM(50, activation=’relu’, input\_shape=(X\_train.shape[1], X\_train.shape[2])),

Dense(1)

])

Model.compile(optimizer=’adam’, loss=’mean\_squared\_error’)

# Train the model

Model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

**EVALUATION**:

To evaluate the performance of the electricity prices prediction model, you can use various metrics commonly used for regression tasks

# Evaluate the model

Loss = model.evaluate(X\_test, y\_test, verbose=0)

Print(f’Mean Squared Error (MSE): {loss}’)

# Calculate additional metrics if needed

From sklearn.metrics import mean\_absolute\_error, r2\_score

Predictions = model.predict(X\_test)

Mae = mean\_absolute\_error(y\_test, predictions)

R2 = r2\_score(y\_test, predictions)

Print(f’Mean Absolute Error (MAE): {mae}’)

Print(f’R-squared (R2) Score: {r2}’)

These metrics provide different perspectives on the model’s performance. Lower MSE and MAE values are better, while a higher R2 score is desirable

**DATASET LINK**

<https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction>

**SUMMARY**

Our project successfully developed a predictive model for electricity prices, achieving a satisfactory level of accuracy. The model is deployed and integrated into the relevant systems for real-time forecasting.