GRT INSTITUTE OF

ENGINEERING AND

TECHNOLOGY, TIRUTTANI - 631209

Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PHASE 3

PROJECT TITLE

ELECTRICITY PRICE PREDICTION USING MACHINE LEARNING

COLLEGE CODE : 1103

LEELAVATHI N V

3rd yr, 5th sem

Reg no. : 110321104025

nvleela2004@gmail.com

3.Electricity Price Prediction Using Machine Learning:

Electricity price prediction! This is a fascinating problem that combines elements of time series forecasting and regression. The goal is to predict the future prices of electricity based on historical data.

Electricity price prediction is crucial for both consumers and providers in the energy market. Accurate predictions enable better decision-making, resource allocation, and cost management. The electricity market is influenced by various factors, including demand patterns, weather conditions, market policies, and more.

DATA PREPROCESSING:

Data Collection:

- Gather historical data on electricity prices. This data should include timestamps and corresponding price values. Additional features like weather conditions, holidays, and special events could also be collected.

Handling Missing Data:

- Check for missing values in the dataset. If any are found, decide on an appropriate strategy to handle them. This might involve interpolation, imputation, or removing the affected data points.

Time Series Decomposition:

- Decompose the time series data into its components: trend, seasonality, and residual. This step helps in understanding the underlying patterns and trends.

Feature Engineering:

- Create additional features that might impact electricity prices. For example, you could include features like day of the week, month, or special events. Lag features, representing past electricity prices, can also be useful.

Normalization/Scaling:

- Normalize or scale the numerical features to ensure that all variables contribute equally to the model. Common techniques include Min-Max scaling or standardization.

Handling Categorical Variables:

- If you have categorical variables like holidays or weekdays, encode them appropriately. One-hot encoding is a common technique for this purpose.

Outlier Detection and Removal:

- Identify and handle outliers in the data. Outliers can significantly impact model performance. Techniques like Z-score or IQR (Interquartile Range) can be employed.

Splitting Data:

- Split the dataset into training and testing sets. This helps assess the model's performance on unseen data.

Time Series Specific Splitting:

- For time series data, ensure that the training set consists of past data, and the testing set contains future data. This mimics the real-world scenario where the model needs to predict unseen future values.

Model-Specific Data Preparation:

- Depending on the chosen prediction model (e.g., ARIMA, LSTM, Random Forest), further data preparation may be required. For instance, reshaping the data for LSTM models or setting up lag features for autoregressive models.

3.1 Introduction to Electricity Price Prediction Dataset:

The electricity price prediction dataset sourced from Kaggle is a collection of historical data capturing the fluctuations in electricity prices over time. This dataset is invaluable for researchers, data scientists, and industry professionals aiming to develop models that can forecast future electricity prices.

Key Features of the Dataset:

Timestamps:

- The dataset includes timestamps corresponding to each observation. These timestamps could be in the format of date and time, providing a temporal dimension to the data.

Electricity Prices:

- The primary target variable is the electricity price. This is the value that the predictive model aims to forecast. Prices might be recorded at regular intervals, such as hourly or daily, depending on the granularity of the data.

Additional Features:

- The dataset may contain additional features that influence electricity prices. These could include weather conditions (temperature, humidity), special events (holidays, festivals), economic indicators, or other relevant factors impacting the energy market.

Missing Values:

- It's common for datasets to have missing values, and this dataset may be no exception. Handling missing data is a crucial step in the preprocessing phase.

Geographical Information:

- Depending on the scope of the dataset, it might include information about the geographical location for which the electricity prices are recorded. This could be at a national, regional, or even city level.

Source of Data:

- The dataset's source is Kaggle, a popular platform for sharing and discovering datasets. Kaggle provides a diverse range of datasets contributed by the community, including those related to energy and electricity markets.

3.2 Loading The Dataset

Loading the dataset is the first step in any data analysis or machine learning project. Let's assume you have a CSV file containing the electricity price prediction dataset. We'll use Python and the popular Pandas library for this task.

1. Import Necessary Libraries:

- You'll need to import the necessary Python libraries to work with the data set.

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns
```

2. Load The Dataset:

- Download the historical data collection about the Electricity price prediction from Kaggle.com and this contains an CSV file. We use Pandas library for loading the dataset. In pandas the function called read_CSV( ) is used for

reading the csv file into the program.We extract the dataset to our programming folder for easy access.

```
# Load the dataset
df = pd.read_csv('Electiricitly.csv')
```

3. Explore The Dataset:

- Now, let's start exploring the dataset to understand its structure and the type of information it contains.

```
# Display the first few rows of the dataset
print(df.head())
# Shape of dataset
print(df.shape)
# Check the basic statistics of the dataset
print(df.describe())
# Check for missing values
print(df.isnull().sum())
# Check the data types of each column
print(df.dtypes)
```

The above code describes the first 5 lines of the dataset, -head() method, shape of the dataset is printed, it sums the null values, and gives the datatype of the dataset.

4. DATA PREPROCESSING:

- Depending on your analysis goals and the segmentation method you plan to use, you may need to pre-process the data. This could include handling outliers, scaling features, and encoding categorical variables.

3.3 Preprocess Dataset

1. Import Libraries:

❖ First, import the necessary libraries, including Pandas for data manipulation.

```
import pandas as pd
```

2. Load The Dataset:

- Load the dataset from the CSV file. Make sure to download the dataset from Kaggle and place it in your working directory.

```
# Load the dataset

df = pd.read_csv('Electricity.csv')
```

3. Explore The Dataset:

- Explore the dataset to understand its structure, check for missing values, and review data types.

```
# Display the first few rows of the dataset

print(df.head())

# Check for missing values

print(df.isnull().sum())

# Check the data types of each column

print(df.dtypes)
```

4. Handle Missing Values:

Depending on the dataset, you may need to handle missing values. Options include imputation or removal of rows/columns.

```
df = df.dropna()  # Remove rows with missing values
```

5.Handling Categorical Variables:

- Convert categorical variables like PeriodOfDay into numerical representations (e.g., one-hot encoding).

```
df = pd.get_dummies(df, columns=['PeriodOfDay'])
```

- In the dataset of Electricity price prediction, there might be categorical data, handling those is an step of preprocessing data.

6. Train-Test Split:

- Split your dataset into training and testing sets.

```
from sklearn.model_selection import train_test_split

X = df[['DayOfWeek', 'Month', 'ForecastWindProduction', 'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed', 'CO2Intensity']]

y = df['SMPEP2']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

7. Save The Preprocessed Data :

- We can save the preprocessed data for future use, you can save it to a new CSV file.

```
df_preprocessed = pd.DataFrame(df_scaled, columns=selected_columns)

df_preprocessed.to_csv('mall_customers_preprocessed.csv', index=False)
```

3.4 PERFORMING DIFFERENT ANALYSIS NEEDED

1. Distribution of Electricity Prices:

```
import matplotlib.pyplot as plt

# Assuming 'SMPEP2' is your target variable

plt.hist(df['SMPEP2'], bins=30, color='blue', alpha=0.7)

plt.title('Distribution of Electricity Prices')

plt.xlabel('Electricity Prices')

plt.ylabel('Frequency')

plt.show()
```

2. Correlation Analysis:

```
# Correlation matrix

correlation_matrix = df.corr()

# Heatmap for visualization

import seaborn as sns

plt.figure(figsize=(12, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix')

plt.show()
```

3.Temporal Analysis:

```
# Assuming 'DateTime' is in datetime format

df['DateTime'] = pd.to_datetime(df['DateTime'])

plt.figure(figsize=(12, 6))

plt.plot(df['DateTime'], df['SMPEP2'], label='Electricity Prices')

plt.title('Time Series Plot of Electricity Prices')

plt.xlabel('Date')

plt.ylabel('Electricity Prices')

plt.legend()

plt.show()
```

4. Weather Impact:

```
# Scatter Plot of Temperature vs. Electricity Prices:

python

Copy code

plt.scatter(df['ORKTemperature'], df['SMPEP2'])

plt.title('Scatter Plot: Temperature vs. Electricity Prices')

plt.xlabel('Temperature')

plt.ylabel('Electricity Prices')

plt.show()
```

5. Wind Production Analysis:

```
#Scatter Plot of Forecasted Wind Production vs. Actual Wind Production:

plt.scatter(df['ForecastWindProduction'], df['ActualWindProduction'])

plt.title('Scatter Plot: Forecasted vs. Actual Wind Production')

plt.xlabel('Forecasted Wind Production')

plt.ylabel('Actual Wind Production')

plt.show()
```

6. Further Analysis:

Feature Importance (if using a machine learning model):

```
from sklearn.ensemble import RandomForestRegressor

# Assuming X and y are your feature matrix and target variable

model = RandomForestRegressor()

model.fit(X, y)

# Feature importances

feature_importances = model.feature_importances_

# Bar plot

plt.bar(X.columns, feature_importances)

plt.title('Feature Importances')

plt.xlabel('Features')

plt.ylabel('Importance')
```

```
plt.show()
```

These analyses provide a starting point for understanding your dataset and identifying potential patterns that can be useful for building an electricity price prediction model. Adjustments and further explorations may be needed based on the specifics of your data.