Name : Suji.G

Reg No : 410121104052

NM ID : Au410121104052

Department : CSE-III

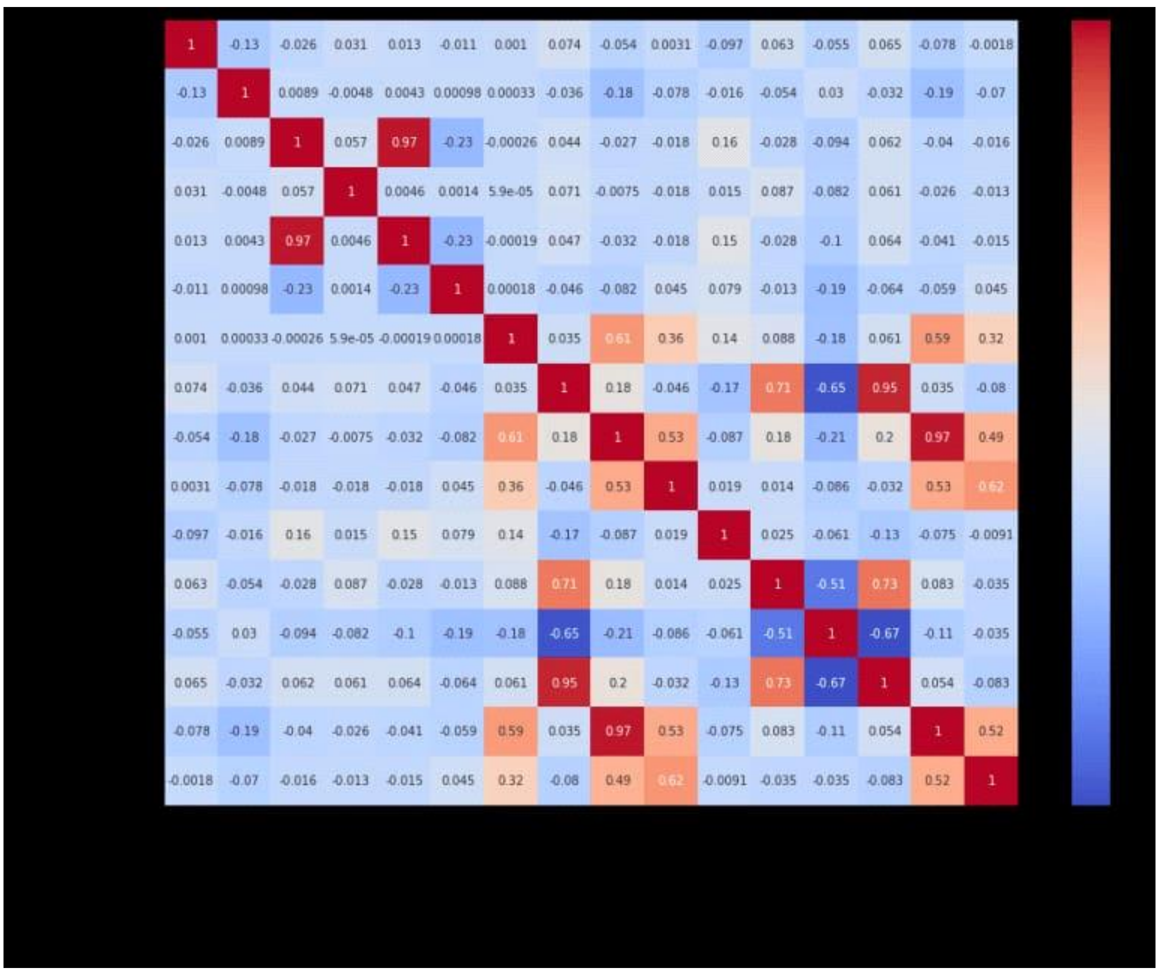
Domain : Data Science

Project : Electricity Price prediction

Introduction:

Electricity price prediction is a crucial application of data science that involves using historical and real-time data to forecast future electricity prices. Accurate predictions can be valuable for various stakeholders, including energy companies, consumers, and policymakers, as they can help in making informed decisions regarding energy production, consumption, and investment. This introduction outlines the key components and steps involved in electricity price prediction using data science techniques.

Dataset :



Necessary steps to follow:

Import libraries:

1.import pandas as pd

2.import numpy as np

LOAD THE DATASET:

1.data=pd.read_csv("ttps://raw.githubusercontent.com/amankharwal/website-data/master/electricity.csv")

2.Print(data.head())

1.DATA PREPROCESSING:

- Handle missing data by either removing or imputing missing values.
- Check for outliers and consider how to handle them

Drop rows with missing values

data.dropna(inplace=True)

Handle outliers (optional)

You can use statistical methods or domain knowledge to identify and handle outliers.

2.FEATURE ENGINEERING :

Create relevant features that can help in price prediction, such as day of the week, time of day, and lagged prices.

Extract date and time features

data['datetime'] = pd.to_datetime(data['timestamp'])

data['hour'] = data['datetime'].dt.hour

data['day_of_week'] = data['datetime'].dt.dayofweek

Create lag features (e.g., lagged prices)

```
data['price_lag1'] = data['price'].shift(1)
data['price_lag2'] = data['price'].shift(2)
```

3.DATA VIZUALIZATION AND EXPLORATORY DATA ANALYSIS:

Explore the data through visualization to understand patterns and relationships.

```
import matplotlib.pyplot as plt

# Visualize the data
plt.figure(figsize=(12, 6))
plt.plot(data['datetime'], data['price'])
plt.title('Electricity Price Over Time')
plt.xlabel('Time')
plt.ylabel('Price')
plt.show()
```

4.DATA SPLIT:

Split the data into training and testing sets.

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X = data[['hour', 'day_of_week', 'price_lag1', 'price_lag2']]
y = data['price']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, rand
```

5.MODEL SELECTION:

Choose an appropriate machine learning model for regression. For this example, we'll use a Random Forest Regressor.

```
from sklearn.ensemble import RandomForestRegressor

# Choose a model and create an instance
```

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
```

6.MODEL TRAINING:

Train the selected model on the training data.

```
# Train the model
model.fit(X_train, y_train)
```

7.MODEL EVALUATION:

Evaluate the model's performance on the test data using appropriate regression metrics.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# 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}')
```

8. HYPERPARAMETER TUNING AND MODEL IMPROVEMENNT:

Depending on the model's performance, you can fine-tune hyperparameters and experiment with different models to achieve better results.

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_absolute_error

# Load and preprocess the data
data = pd.read_csv('electricity_price_data.csv')
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)

# Feature selection and engineering
data['hour'] = data.index.hour
data['day_of_week'] = data.index.dayofweek

# Split the data
X = data[['hour', 'day_of_week']]
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.3, random_state=42)
```

```
# Train a simple linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, predictions)
print(f'Mean Absolute Error: {mae}')
```

CONCLUSION:

In conclusion, predicting electricity prices using data science is a complex yet valuable task with a wide range of applications, from energy trading to resource management.