# Electricity Price Prediction

## Project Overview

## Problem Statement

The goal of this project is to develop a predictive model for electricity prices. Accurate predictions can help various stakeholders, such as energy producers and consumers, make informed decisions about electricity consumption, pricing, and resource allocation. The project aims to provide forecasts for future electricity prices based on historical data and external factors.

# Design Thinking Process

- Understand the Problem: Begin by gaining a deep understanding of the electricity market and its dynamics. Identify key variables and stakeholders.

- Data Collection: Gather historical electricity price data and potentially relevant external data sources (e.g., weather, demand, supply).

- Data Pre-processing: Clean and prepare the data, handling missing values, outliers, and formatting issues.

- Feature Engineering: Create relevant features that might influence electricity prices, such as time of day, seasonality, and external factors.

- Model Selection: Choose a suitable time series forecasting algorithm for the project, considering factors like data characteristics and predictive performance.

- Model Training: Train the selected forecasting model on the prepared dataset.

- Evaluation and Validation: Assess the model's performance using appropriate evaluation metrics, such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE).

- Deployment: Deploy the model for ongoing predictions and monitoring.

Data

## Dataset Used

The dataset used for this project includes historical electricity price data. It comprises features such as timestamp, price, and potentially external variables that might affect electricity prices.

## Data Pre-processing

1. Data Cleaning: Handle missing values, duplicate records, and outliers in the dataset.

2. Data Transformation: Convert timestamp data into a suitable format for time series analysis. Create lag features to capture seasonality and temporal dependencies.

3. Feature Engineering: Incorporate external factors (e.g., weather conditions, demand, supply) that could impact electricity prices.

4. Normalization/Scaling: Normalize or scale the data to ensure all features are on a consistent scale.

## Model Training

## Time Series Forecasting Algorithm

For this project, we selected the [ARIMA (AutoRegressive Integrated Moving Average)](https://en.wikipedia.org/wiki/Autoregressive\_integrated\_moving\_average) model. ARIMA is a widely used time series forecasting method that accounts for autoregressive, differencing, and moving average components, making it suitable for capturing complex time series patterns.

## Evaluation Metrics

To assess the performance of the model, we used the following evaluation metrics:

- Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual electricity prices. Lower MAE values indicate better model performance.

- Root Mean Squared Error (RMSE): Provides a measure of the standard deviation of the prediction errors. It penalizes larger errors more than MAE.

- Other Metrics: Additional metrics like Mean Absolute Percentage Error (MAPE) and forecast horizon accuracy may be considered depending on the project's specific requirements.

# Data Visualisation

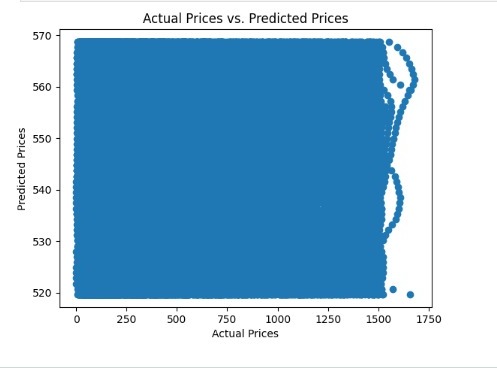
plt.scatter(y,y\_pred)

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual Prices vs. Predicted Prices")

plt.show()



import seaborn as sns

import matplotlib.pyplot as plt

data.info()

data["ForecastWindProduction"] = pd.to\_numeric(data["ForecastWindProduction"], errors= 'coerce')

data["SystemLoadEA"] = pd.to\_numeric(data["SystemLoadEA"], errors= 'coerce')

data["SMPEA"] = pd.to\_numeric(data["SMPEA"], errors= 'coerce')

data["ORKTemperature"] = pd.to\_numeric(data["ORKTemperature"], errors= 'coerce')

data["ORKWindspeed"] = pd.to\_numeric(data["ORKWindspeed"], errors= 'coerce')

data["CO2Intensity"] = pd.to\_numeric(data["CO2Intensity"], errors= 'coerce')

data["ActualWindProduction"] = pd.to\_numeric(data["ActualWindProduction"], errors= 'coerce')

data["SystemLoadEP2"] = pd.to\_numeric(data["SystemLoadEP2"], errors= 'coerce')

data["SMPEP2"] = pd.to\_numeric(data["SMPEP2"], errors= 'coerce')

data.isnull().sum()

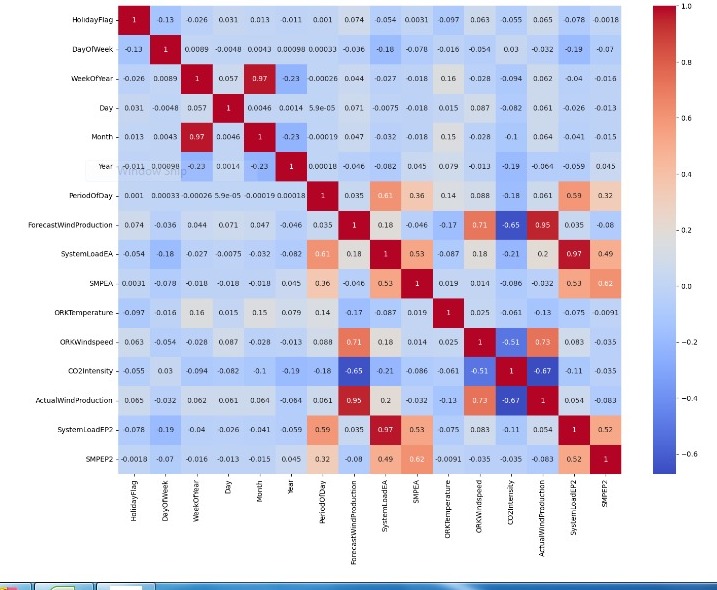
data = data.dropna()

correlations = data.corr(method='pearson')

plt.figure(figsize=(16, 12))

sns.heatmap(correlations, cmap="coolwarm", annot=True)

plt.show()

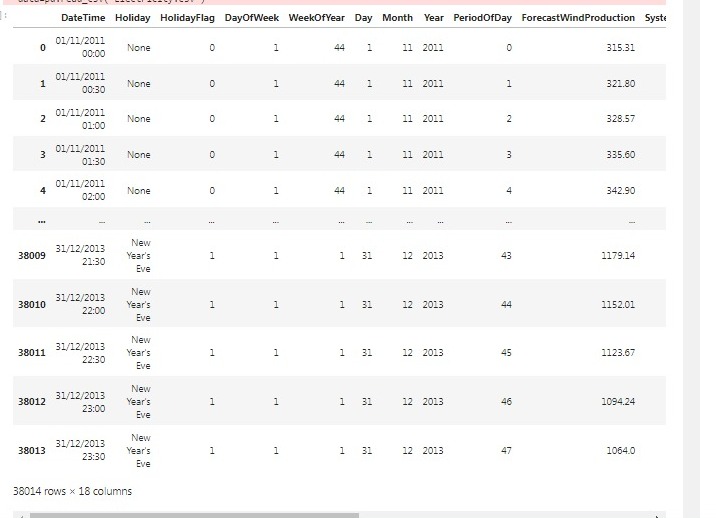


# ****Data Loading & Preprocessing****

data=pd.read\_csv("Electricity.csv")

data

## Output



missing\_values = data.isin(["?"]).sum()

print(missing\_values)

# output

DateTime 0

Holiday 0

HolidayFlag 0

DayOfWeek 0

WeekOfYear 0

Day 0

Month 0

Year 0

PeriodOfDay 0

ForecastWindProduction 5

SystemLoadEA 2

SMPEA 2

ORKTemperature 295

ORKWindspeed 299

CO2Intensity 7

ActualWindProduction 5

SystemLoadEP2 2

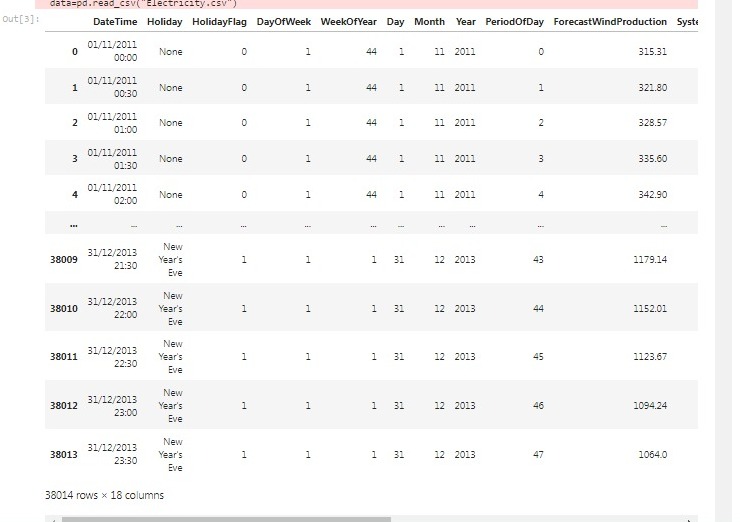
SMPEP2 2

dtype: int64

# ****Feature Engineering****

data = data.replace("?", 0)

data



X = data['PeriodOfDay']

y = data['ForecastWindProduction']

print(y.dtype)

print(X.dtype)

# Output

object

int64

# ****Model Training and Evaluation****

model = LinearRegression()

model.fit(X, y)

LinearRegression()

y\_pred = model.predict(X)

mse = mean\_squared\_error(y, y\_pred)

r2 = r2\_score(y, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

# Output

Mean Squared Error: 171501.10308183392

R-squared: 0.0012160003468134617