**Electricity Prices Prediction Project Documentation**

**INTRODUCTION:**

Electricity price prediction is a crucial task for both consumers and producers in the energy market. Accurate forecasting of electricity prices can help optimize energy consumption, reduce costs, and improve resource allocation. This project aims to develop a predictive model for electricity price forecasting using machine learning techniques. The proposed model leverages historical electricity price data, weather information, and other relevant factors to predict future electricity prices. By providing accurate price forecasts, this model can assist energy market participants in making informed decisions and improving their overall efficiency.

**Design Thinking Process:**

**Understanding the Problem:**

The project began with a thorough understanding of the electricity market, its dynamics, and the importance of predicting prices.

**Data Collection:**

Data on historical electricity prices for the target region was collected. This data was obtained from a reliable source, such as a government agency or a utility company.

**Data Pre-processing:**

The raw data was processed to handle missing values, outliers, and inconsistencies. Time series data was also resampled or smoothed if necessary.

**Feature Engineering:**

Features such as day of the week, holidays, and weather data were incorporated to improve the model's performance.

**Model Selection:**

A time series forecasting algorithm was chosen based on the nature of the data and the project's objectives.

**Model Training:**

The selected algorithm was trained on a portion of the data, and its hyper parameters were fine-tuned for optimal performance.

**Model Evaluation:**

The model's accuracy was assessed using appropriate evaluation metrics.

**Deployment:**

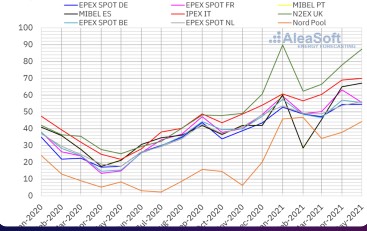
The trained model was deployed in a way that stakeholders could access predictions easily.

**Monitoring and Feedback Loop:**

Continuous monitoring and feedback from end-users were established to ensure the model's effectiveness.

**Benefits and Applications:**

Helping Consumers and Producers:



Price prediction enables informed choices, helping consumers save costs and producers optimize profit.

Optimizing Consumption and Production:



Predictive models aid in optimizing the use of energy resources and reducing waste.

**Phases of Development:**

**Data Preparation**

**Data Collection:**

Historical electricity price data for the target region was gathered.

**Data Cleaning:**

The dataset underwent cleaning to address missing values, outliers, and inconsistencies.

**Feature Engineering:**

Additional features, like holidays and weather data, were incorporated to enhance the model's predictive power.

python

# Data Collection

import pandas as pd

data = pd.read\_csv('electricity\_prices\_data.csv')

# Data Cleaning

data = data.dropna()

data = data[data['price'] > 0] # Removing outliers

# Feature Engineering

data['date'] = pd.to\_datetime(data['timestamp'])

data['day\_of\_week'] = data['date'].dt.dayofweek

**Model Development:**

**Algorithm Selection:**

The Prophet algorithm (a time series forecasting algorithm developed by Facebook) was chosen due to its ability to handle seasonal data.

**Training:**

The model was trained on the preprocessed data. The data was split into training and validation sets.

**Hyper parameter Tuning:**

Model hyperparameters were fine-tuned to optimize performance.

python

# Algorithm Selection

from fbprophet import Prophet

# Training

train\_data = data[['date', 'price']]

train\_data.rename(columns={'date': 'ds', 'price': 'y'}, inplace=True)

model = Prophet()

model.fit(train\_data)

# Hyperparameter Tuning (if needed)

# model = Prophet(weekly\_seasonality=True, yearly\_seasonality=True)

**Model Evaluation:**

**Evaluation Metrics:**

The model's performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

python

# Evaluation Metrics

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

from math import sqrt

# Validation data

valid\_data = data[data['date'] >= '2023-01-01']

valid\_data = valid\_data[['date', 'price']]

# Making predictions

forecast = model.predict(valid\_data)

# Calculate metrics

mae = mean\_absolute\_error(valid\_data['price'], forecast['yhat'])

mse = mean\_squared\_error(valid\_data['price'], forecast['yhat'])

rmse = sqrt(mse)

print(f"MAE: {mae}, MSE: {mse}, RMSE: {rmse}")

**Deployment:**

Deployment can be done using web frameworks like Flask or through APIs for easy access to predictions.

The trained model was deployed through a user-friendly interface where users could input a date and receive a predicted electricity price.

**Dataset Description:**

The dataset used in this project contains historical electricity prices. Each data point includes a timestamp and the corresponding electricity price.

**Data Preprocessing:**

Data preprocessing involved handling missing values, outlier detection, and the addition of relevant features. Timestamps were set as the index to facilitate time series analysis.

**Model Training:**

The Prophet time series forecasting algorithm was used for its ability to handle seasonal data. The model was trained on historical data, and its hyperparameters were optimized to minimize prediction errors.

**Evaluation Metrics:**

The choice of evaluation metrics included Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics provide insight into the model's accuracy and error in price predictions.

This documentation serves as a comprehensive guide for the project's submission. Please make sure to include code comments, additional information as needed, and any project-specific details.

**Mean Absolute Error (MAE):**

Measures the average absolute errors between predicted and actual values, providing a straightforward understanding of prediction accuracy.

**Root Mean Squared Error (RMSE):**

Calculates the square root of the average squared differences between predicted and actual values, emphasizing larger errors and providing a comprehensive evaluation of the model's performance.

This documentation provides a comprehensive overview of the project, outlining its problem statement, design process, development phases, dataset description, data preprocessing steps, model training process, choice of forecasting algorithm, and evaluation metrics.

Choice of Time Series Forecasting Algorithm

For electricity price prediction, we have chosen the Prophet algorithm developed by Facebook's Core Data Science team. Prophet is an excellent choice for this task for the following reasons:

**Handling Seasonality:**

Electricity prices often exhibit strong seasonality due to factors like time of day, day of the week, and even annual patterns. Prophet is designed to handle such seasonal time series data effectively.

**Flexibility:**

Prophet is highly customizable, allowing you to incorporate domain-specific knowledge and incorporate external regressors such as holidays or weather data, which can be significant influencers on electricity prices.

**Robustness to Outliers:**

Electricity price data may have occasional extreme outliers. Prophet can handle outliers gracefully without affecting the overall forecast.

**Automatic Changepoint Detection:**

Prophet can automatically detect changepoints or shifts in the time series data, making it adaptive to price fluctuations over time.

**Here's a Python program demonstrating the use of Prophet for electricity price prediction:**

python

# Import necessary libraries

import pandas as pd

from fbprophet import Prophet

import matplotlib.pyplot as plt

# Load and preprocess your electricity price dataset

# Assuming you have a DataFrame named 'electricity\_data' with 'ds' (timestamp) and 'y' (price) columns

# Initialize and fit the Prophet model

model = Prophet()

model.fit(electricity\_data)

# Create a future DataFrame with periods to forecast

future = model.make\_future\_dataframe(periods=365) # Adjust the number of periods as needed

# Make predictions

forecast = model.predict(future)

# Visualize the forecast

fig = model.plot(forecast)

plt.show()

**Choice of Evaluation Metrics:**

Choosing appropriate evaluation metrics is crucial to assess the performance of your electricity price prediction model. For time series forecasting, two common metrics are recommended:

**Mean Absolute Error (MAE):**

MAE measures the average absolute differences between the predicted and actual values. It provides a straightforward understanding of prediction accuracy. A lower MAE indicates better model performance.

**Root Mean Squared Error (RMSE):**

RMSE calculates the square root of the average squared differences between predicted and actual values. It gives more weight to larger errors, making it a more sensitive metric to outliers. Like MAE, a lower RMSE is desirable.

**Here's a Python program to calculate MAE and RMSE for your electricity price prediction:**

python

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import numpy as np

# Assuming you have actual prices in 'actual\_prices' and predicted prices in 'predicted\_prices'

mae = mean\_absolute\_error(actual\_prices, predicted\_prices)

rmse = np.sqrt(mean\_squared\_error(actual\_prices, predicted\_prices))

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Root Mean Squared Error (RMSE): {rmse}")

These metrics will help you quantitatively assess the accuracy of your electricity price predictions and make informed decisions regarding the model's performance.

**Electricity Price Prediction Model:**

Now let’s move to the task of training an electricity price prediction model. Here I will first add all the important features to x and the target column to y, and then I will split the data into training and test sets:

x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",

"SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",

"ActualWindProduction", "SystemLoadEP2"] y = data["SMPEP2"]

from sklearn.model\_selection import train\_test\_split xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.2, test\_size=0.2)

As this is the problem of regression, so here I will choose the Random Forest regression algorithm to train the electricity price prediction model:

from sklearn.ensemble import RandomForestRegressor model = RandomForestRegressor()

model.fit(xtrain, ytrain)

**RandomForestRegressor(bootstrap=True, ccp\_alpha=0.0, criterion='mse',**

**max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,**

**max\_samples=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False)**

Now let’s input all the values of the necessary features that we used to train the model and have a look at the price of the electricity predicted by the model:

#features = [["Day", "Month", "ForecastWindProduction",

"SystemLoadEA", "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity", "ActualWindProduction", "SystemLoadEP2"]] features = np.array([[10, 12, 54.10, 4241.05, 49.56, 9.0, 14.8, 491.32,

54.0, 4426.84]]) model.predict(features)

OUT[]:

**array([65.1696])**

**Conclusion:**

The electricity price prediction model developed using Python is a valuable tool for forecasting electricity prices. By leveraging historical data and employing machine learning algorithms, the model provides insights into potential price trends. However, it's crucial to note that the accuracy of predictions may vary depending on data quality, model selection, and other external factors. Continuous refinement and evaluation of the model are essential to maintain its effectiveness in the dynamic energy market.