Electricity price Prediction

Creating an electricity price prediction model involves several steps, including data collection, data pre processing, feature selection/engineering, model selection, training, and evaluation. It provide with a high-level outline of these steps

Step 1: Data Collection

1. **Gather Historical Data**: Collect historical electricity price data. This can include factors like time of day, day of the week, month, weather conditions, etc. You might also want to include data on factors that influence electricity prices, like demand, supply, and geopolitical events.

Step 2: Data Preprocessing

1. **Data Cleaning**: Handle missing values, outliers, and anomalies in the dataset.
2. **Data Transformation**: Convert categorical variables into numerical values (if any). Scale or normalize numerical features.
3. **Feature Engineering**: Create additional features that might be relevant for prediction. For example, you might want to extract features like time of day, day of the week, holidays, etc.

Step 3: Model Selection

1. **Choose Algorithm**: Decide on a suitable algorithm for your prediction task. Common choices include Linear Regression, Random Forest, Support Vector Machines, Neural Networks, etc.
2. **Time Series Consideration**: If the data has a time component, consider using time series models like ARIMA, SARIMA, or LSTM.

Step 4: Train-Test Split

1. **Divide Data**: Split the dataset into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate its performance.

Step 5: Model Training

1. **Fit the Model**: Use the training data to train the chosen model. The model learns to make predictions based on the features you provided.

Step 6: Model Evaluation

1. **Performance Metrics**: Evaluate the model's performance using appropriate metrics (e.g., Mean Absolute Error, Mean Squared Error, R-squared, etc.).
2. **Cross-validation (Optional)**: Perform k-fold cross-validation to assess the model's robustness.

Step 7: Hyper parameter Tuning (Optional)

1. **Optimize Parameters**: Fine-tune the model's hyperparameters to improve performance. This might involve techniques like grid search or random search.

Step 8: Model Deployment

1. **Deployment Environment**: Choose an environment to deploy your model. It could be a cloud service, a web application, or a local server.

Step 9: Monitoring and Maintenance

1. **Monitor Model Performance**: Continuously monitor the model's performance in the real world. Update the model or retrain it as needed.

Remember that this is a high-level overview, and each step can be quite involved depending on the specific details of your project. Additionally, the choice of algorithms, preprocessing techniques, and evaluation metrics can vary based on the nature of your data and the goals of your prediction model.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Step 1: Load the dataset (assuming you have a CSV file)

# Replace 'your_dataset.csv' with the actual path to your dataset

data = pd.read_csv(' https://github.com/Akash-Jeyachandran/Electricity-price-prediction/Electricity.csv')

# Step 2: Data Preprocessing

# Assuming your dataset contains features like temperature, demand, etc.

# X should be the features, and y should be the target variable (electricity price)

X = data[['Temperature', 'Demand']]

y = data['Price']

# Step 3: Split the data into training and testing sets

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

# Step 4: Initialize the model (Linear Regression)

model = LinearRegression()

# Step 5: Train the model

model.fit(X_train, y_train)

# Step 6: Predict on the test set

y_pred = model.predict(X_test)

# Step 7: Evaluate the model (using Mean Squared Error as an example)

mse = mean_squared_error(y_test, y_pred)

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

# Optional: You can save the model for future use
```

```
# Replace 'your_model_name.pkl' with a meaningful name

import joblib

joblib.dump(model, “Electricity.pkl')
```