FUTURE SALES PREDICTION USING PYTHON

Begin building the electricity prices prediction model by loading and preprocessing the dataset of future sales prediction

Creating an electricity price prediction model involves several steps, including data loading, preprocessing, feature engineering, and model building. I'll guide you through these steps. However, please note that I don't have access to specific future sales prediction data since my knowledge is up to date only until September 2021. You'll need to provide your own dataset or access to a suitable one for this task.

Here's a general outline of the steps to build an electricity price prediction model:

1. **Data Loading**:
   - Import the necessary libraries (e.g., pandas, numpy, scikit-learn).
   - Load your dataset containing historical electricity prices and relevant features. Ensure the dataset includes columns like timestamp, electricity price, and any relevant predictors (e.g., demand, weather, market conditions).

import pandas as pd

 # Load your dataset

df = pd.read_csv('your_dataset.csv')

2. **Data Preprocessing**:
   - Check for missing values and handle them appropriately (e.g., impute, remove, or interpolate).
   - Convert the timestamp column to a datetime data type.
   - Explore and clean the data, addressing any anomalies or outliers.

Python

```
    # Check for missing values
df.isnull().sum()

# Convert timestamp to datetime
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Handle missing values
df.fillna(method='ffill', inplace=True)  # Forward fill missing values
```

Feature Engineering:
   - Create relevant features for prediction, such as lag features, moving averages, and any domain-specific features.
   - Normalize or scale the features as needed.

 # Example: Create lag features for electricity price

df['price_lag_1'] = df['electricity_price'].shift(1)

df['price_lag_7'] = df['electricity_price'].shift(7)

 # Normalize features (if necessary)

from sklearn.preprocessing import MinMaxScaler

```
scaler = MinMaxScaler()

df[['electricity_price', 'demand', 'weather']] = scaler.fit_transform(df[['electricity_price', 'demand', 'weather']])
```

Train-Test Split:

- Split the data into training and testing sets to evaluate the model's performance.

```
from sklearn.model_selection import train_test_split

X = df.drop('electricity_price', axis=1)  # Features

y = df['electricity_price']  # Target variable


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

Model Selection and Training:

- Choose a suitable machine learning or time series forecasting model for prediction, like Linear Regression, Random Forest, or ARIMA.
- Train the model using the training data.

```
# Example using Linear Regression

from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, y_train)
```

Model Evaluation:

- Evaluate the model's performance on the test dataset using appropriate metrics (e.g., Mean Absolute Error, Root Mean Squared Error, R-squared).

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_pred = model.predict(X_test)


mae = mean_absolute_error(y_test, y_pred)

rmse = mean_squared_error(y_test, y_pred, squared=False)

r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')

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

print(f'R-squared: {r2}')
```

7. **Hyperparameter Tuning and Optimization**:
    - Fine-tune model hyperparameters to improve prediction accuracy.
8. **Prediction**:

- Use the trained model to make predictions for future electricity prices based on new or unseen data.

Remember that this is a simplified outline, and the specific preprocessing, feature engineering, and modeling techniques may vary depending on your dataset and requirements. Also, for predicting future sales, you might want to use a time series forecasting model like ARIMA, Prophet, or LSTM if your data has a strong temporal component.

Load the historical electricity prices dataset and preprocess the data for analysis.

I can certainly guide you on how to preprocess a historical electricity prices dataset for analysis and future sales prediction. However, I cannot load or fetch specific datasets as my knowledge is based on information available up until September 2021. You would need to obtain the dataset from a reliable source.

Here are the general steps to preprocess a historical electricity prices dataset:

1. **Data Collection**: Obtain a historical electricity prices dataset from a reliable source. You can often find such datasets from government agencies, energy companies, or research organizations. Make sure the dataset includes relevant information such as date, time, and electricity prices.
2. **Data Loading**: Load the dataset into your preferred data analysis environment or tool, such as Python with Pandas, R, or any other suitable tool.
3. **Data Inspection**: Examine the dataset to get a sense of its structure and the type of information it contains. You should check for missing values, outliers, and data types.
4. **Data Cleaning**:
    - Handle Missing Data: Decide how to handle missing values. You can either impute missing values or remove rows or columns with missing data, depending on the extent of the missing values.
    - Handle Outliers: Identify and address any outliers in the data. Outliers can significantly affect the performance of predictive models.
    - Data Type Conversion: Ensure that date and time columns are in the appropriate date-time format. Convert any numerical data to the correct data types.
5. **Feature Engineering**:
    - Create Relevant Features: Depending on your analysis and prediction goals, you might need to create new features from the existing data. For example, you can add features like day of the week, hour of the day, or holidays as binary indicators.
    - Lag Features: Generate lag features if necessary. These are historical price values for a previous time period, which can be helpful in time series analysis.
6. **Data Transformation**:
    - Normalization or Standardization: Depending on the algorithms you plan to use for prediction, you may need to normalize or standardize your data to bring all features to the same scale.
    - Encoding Categorical Variables: If your dataset contains categorical variables (e.g., region or type of electricity source), you may need to encode them numerically using techniques like one-hot encoding.
7. **Data Splitting**: Split the data into training and testing datasets. The training data will be used to build your predictive model, while the testing data will be used to evaluate the model's performance.

8. **Data Visualization**: Visualize the data to gain insights into its patterns, trends, and seasonality. This step can help you make informed decisions during feature selection and model building.
9. **Modeling**: Depending on your specific goals, you can apply various machine learning or time series analysis techniques to predict future electricity prices or sales.
10. **Evaluation**: Evaluate the performance of your model using appropriate metrics. Common metrics for regression tasks include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).
11. **Hyperparameter Tuning**: Fine-tune your model's hyperparameters to improve its predictive accuracy.
12. **Deployment**: If the model meets your performance criteria, you can deploy it to make future sales predictions or inform decision-making.

Remember that the preprocessing steps can vary based on the nature of your dataset and the specific requirements of your analysis and prediction task.