Electricity demand forecasting

Problem Description :

Electricity demand forecasting is a critical task in the energy industry, as it helps in efficiently planning and managing the generation, transmission, and distribution of electricity. Accurate forecasting enables power utilities and grid operators to make informed decisions about resource allocation, maintenance, and pricing strategies.

The goal of this project is to develop a model that can accurately predict the daily electricity demand in the Spanish market for the year 2018 based on historical data. The dataset used for this task contains information about the daily electricity demand in Spain, recorded at different timestamps over a period of time.

Dataset Information:

- The dataset is provided in a CSV format and is named "data.csv." It includes the following columns:
- datetime: Timestamp of the electricity demand reading.
- name: Name of the parameter (Demanda programada PBF total).
- value: The actual value of electricity demand recorded at each timestamp.

Background Information:

The electricity demand in any region is influenced by various factors, including seasonal patterns, economic activities, weather conditions, and special events. Forecasting electricity demand is challenging due to its non-linear and dynamic nature, making it a suitable application for machine learning techniques.

In this project, the RandomForestRegressor algorithm is used for demand forecasting. RandomForestRegressor is an ensemble learning method that combines multiple

decision trees to make predictions. The algorithm is well-suited for regression tasks and can handle non-linear relationships between input features and the target variable.

The code first performs data preparation by filtering the relevant data for electricity demand. It then extracts time-related features and calculates rolling statistics to provide meaningful insights into the demand patterns.

Next, the code performs data visualization to gain an understanding of the data distribution and identify any seasonality or trends.

The model evaluation is carried out using Mean Absolute Percent Error (MAPE), which measures the accuracy of the forecasts. The final step involves forecasting the electricity demand for the year 2018 using the trained model.

By accurately predicting electricity demand, power utilities can efficiently allocate resources, manage peak demand periods, and optimize energy production, resulting in cost savings and a more reliable electricity supply.

The key challenges in this project include handling non-linearity, detecting and dealing with seasonality, and choosing appropriate time windows for calculating rolling statistics.

Overall, electricity demand forecasting plays a crucial role in ensuring the stability and sustainability of power systems, making it a significant application of data science and machine learning in the energy sector.

Possible Framework :

1. **Importing Libraries and Loading Data:**
- Import the necessary Python libraries, including pandas, datetime, numpy, matplotlib, seaborn, and scipy.stats.
- Load the dataset from the CSV file using pandas, parse the "datetime" column as datetime objects, and set the "date" column as the index.
2. **Data Preprocessing and Filtering:**
- Filter the dataset to keep only the relevant data related to electricity demand ("Demanda programada PBF total").
- Resample the data to have daily frequency (as the dataset may have irregular timestamps).
- Rename the "value" column to "energy" for better readability.
3. **Data Visualization:**
- Plot the daily energy demand over time to visualize the trends and patterns.
- Analyze the data distribution, check for outliers, and calculate basic statistics like mean, standard deviation, skewness, and kurtosis.
- Perform a Shapiro-Wilk test to check for normality in the data.
4. **Feature Engineering and Time-Series Analysis:**
- Create additional time-related features, such as year, quarter, month, week, and day of the week, from the "datetime" column.
- Calculate rolling statistics (moving averages and standard deviations) over different time windows to analyze trends and seasonality.
5. **Seasonality Analysis:**
- Plot the moving averages over different time windows to identify seasonal patterns.
- Calculate the coefficient of variation (CV) for different quarters and months to analyze seasonality variations.
6. **Heteroscedasticity Analysis:**
- Plot the rolling standard deviations over different time windows to detect heteroscedasticity (variance changing over time).
7. **Linear Regression Modeling:**
- Split the dataset into training and testing sets, using "2014" to "2017" for training and "2018" for testing.

- Train a linear regression model on the training data and evaluate its performance on the testing data using Root Mean Squared Error (RMSE).

8. **Time-Series Cross-Validation:**

- Implement time-series cross-validation using TimeSeriesSplit with a window size of 2 years to avoid data leakage.
- Tune hyperparameters of the Random Forest Regressor using a parameter grid.
- Evaluate the performance of the model using RMSE for each fold and identify the best set of hyperparameters.

9. **Feature Importance Analysis:**

- Determine the importance of features using the best-performing Random Forest Regressor model.
- Plot the feature importances to identify the most significant features contributing to the electricity demand forecasting.

10. **Multi-Step Forecasting:**

- Perform multi-step forecasting for 1, 7, 14, and 30 periods ahead using the trained Random Forest Regressor model.
- Calculate Mean Absolute Percent Error (MAPE) for each forecasting period to assess the accuracy of predictions.

11. **Forecast Visualization:**

- Plot the 1-period ahead forecast along with actual values to visualize the forecasting performance.
- Plot scatter plots for different forecasting periods to compare actual and predicted values.

12. **Conclusion and Recommendations:**

- Summarize the findings and results of the electricity demand forecasting project.
- Discuss the accuracy and limitations of the model and potential areas for improvement.
- Provide recommendations on how the forecasting model can be utilized by power utilities for effective demand management.

Code Explanation :

*If this section is empty, the explanation is provided in the .ipynb file itself.

Step 1: Importing Libraries and Loading Data

- The code begins by importing necessary Python libraries such as pandas, datetime, numpy, matplotlib, seaborn, and scipy.stats. These libraries provide various functionalities for data manipulation, visualization, and statistical analysis.
- Next, the code loads the dataset from a CSV file called "data.csv" using the pandas library's read_csv function. It also parses the "datetime" column as datetime objects to work with time-related data effectively. The data is stored in a pandas DataFrame for further processing.

Step 2: Data Preprocessing and Filtering

- In this step, the code filters the dataset to keep only the relevant data related to electricity demand. It selects the rows where the "name" column has the value "Demanda programada PBF total".
- The code then resamples the data to have daily frequency, ensuring that all timestamps are aligned to daily intervals. This step handles any irregularities in the data.
- To improve readability, the code renames the "value" column to "energy".

Step 3: Data Visualization

- This step focuses on visualizing the energy demand data over time. It uses matplotlib and seaborn libraries to create line plots, distribution plots, and statistical plots to analyze and understand the data's characteristics.
- The code plots the daily energy demand as a time series to observe trends and patterns.
- It calculates basic statistics like mean, standard deviation, skewness, and kurtosis to understand the data's distribution and identify any potential outliers.
- The code performs the Shapiro-Wilk test to check if the data follows a normal distribution or not.

Step 4: Feature Engineering and Time-Series Analysis

- Feature engineering involves creating additional time-related features from the "datetime" column. This step includes extracting the year, quarter, month, week, and day of the week from the datetime objects. These features will help capture the seasonality and trend patterns in the data.
- The code calculates rolling statistics, such as moving averages and standard deviations, over different time windows. Rolling statistics provide insights into trends and seasonality variations in the data.

Step 5: Seasonality Analysis

- The code plots the moving averages over different time windows to identify any seasonal patterns in the data.
- It calculates the coefficient of variation (CV) for different quarters and months. The CV helps assess the variability of demand within each season.

Step 6: Heteroscedasticity Analysis

- Heteroscedasticity refers to the changing variance of the data over time. The code plots the rolling standard deviations over different time windows to detect any heteroscedasticity in the energy demand.

Step 7: Linear Regression Modeling

- This step involves splitting the dataset into training and testing sets. The training set contains data from "2014" to "2017," and the testing set contains data from "2018."
- The code trains a linear regression model on the training data to predict energy demand. It then evaluates the model's performance on the testing data using Root Mean Squared Error (RMSE) as the evaluation metric.

Step 8: Time-Series Cross-Validation

- Time-Series Cross-Validation is performed using TimeSeriesSplit with a window size of 2 years to avoid data leakage. Data is split into multiple folds, and the Random Forest Regressor is trained and evaluated on each fold with different hyperparameters.
- The code tunes hyperparameters of the Random Forest Regressor using a parameter grid to find the best set of hyperparameters.

Step 9: Feature Importance Analysis

- The code determines the importance of features using the best-performing Random Forest Regressor model. Feature importances indicate which features contribute most significantly to the electricity demand forecasting.
- It plots the feature importances to visualize the relative importance of different features.

Step 10: Multi-Step Forecasting

- Multi-step forecasting is performed for 1, 7, 14, and 30 periods ahead using the trained Random Forest Regressor model.
- Mean Absolute Percent Error (MAPE) is calculated for each forecasting period to assess the accuracy of the predictions.

Step 11: Forecast Visualization

- The code plots the 1-period ahead forecast along with actual values to visualize the forecasting performance.
- Scatter plots are plotted for different forecasting periods to compare actual and predicted values.

Step 12: Conclusion and Recommendations

- In this final step, the code concludes the electricity demand forecasting project and summarizes the findings and results.
- It discusses the accuracy and limitations of the model and provides recommendations on how the forecasting model can be utilized by power utilities for effective demand management.

Future Work :

Electricity demand forecasting is a critical task for power utilities to efficiently manage their resources and plan for future requirements. While the current project has provided a foundation for demand forecasting, there are several avenues for further improvement and exploration. Below is a detailed plan for future work:

Step 1: Data Collection and Preprocessing

- Collect more granular and diverse data: Acquire additional data sources, such as weather data, economic indicators, holidays, and special events, to capture external factors that influence electricity demand.

Step 2: Advanced Feature Engineering

- Explore advanced feature engineering techniques: Investigate time-series decomposition methods like Seasonal and Trend decomposition using LOESS (STL) to extract the seasonal, trend, and residual components explicitly.

Step 3: Time-Series Analysis

- Explore advanced time-series analysis methods: Implement time-series forecasting techniques like ARIMA, SARIMA, and Prophet to compare their performance with the current Random Forest Regressor model.

Step 4: Model Selection and Hyperparameter Tuning

- Experiment with different models: Try out various machine learning algorithms, including Gradient Boosting Machines, Long Short-Term Memory (LSTM) networks, and Recurrent Neural Networks (RNNs), to identify the best-performing model for the specific electricity demand dataset.

Step 5: Ensembling Techniques

- Implement ensemble models: Explore ensemble techniques like stacking and blending to combine the predictions of multiple models, enhancing forecasting accuracy and robustness.

Step 6: Online Learning and Continuous Improvement

- Set up an automated pipeline: Implement an automated data pipeline to regularly update the model with new data and retrain it periodically to capture changing demand patterns.

Step 7: Error Analysis and Uncertainty Estimation

- Analyze forecasting errors: Perform in-depth error analysis to identify patterns in prediction inaccuracies and take corrective actions.
- Estimate prediction uncertainties: Implement uncertainty estimation techniques, such as bootstrapping and Monte Carlo simulations, to quantify uncertainty in the forecasted values.

Step 8: Real-Time Forecasting

- Develop real-time forecasting capability: Build a system to generate real-time electricity demand forecasts based on the latest available data, enabling better decision-making for demand management.

Step 9: Integration with Demand Response

- Integrate forecasting with demand response programs: Use the forecasts to optimize demand response strategies, incentivizing consumers to adjust their electricity consumption during peak periods.

Step 10: Visualization and Reporting

- Develop interactive dashboards: Create user-friendly dashboards that provide visualizations and insights for stakeholders, making it easier to interpret the forecasting results.
- Generate automated reports: Generate automated reports summarizing the forecasting performance and key insights for management and regulatory purposes.

Step-by-Step Guide to Implement Future Work:

1. **Data Collection:** Gather additional data sources related to weather, holidays, and economic indicators.
2. **Data Preprocessing:** Clean and preprocess the new data to align it with the existing dataset.

3. **Advanced Feature Engineering:** Implement time-series decomposition techniques like STL to extract components from the data.
4. **Time-Series Analysis:** Implement ARIMA, SARIMA, and Prophet models for time-series forecasting.
5. **Model Selection and Hyperparameter Tuning**: Experiment with various machine learning algorithms and tune their hyperparameters using cross-validation.
6. **Ensembling Techniques:** Combine the predictions of multiple models using stacking or blending.
7. **Online Learning and Continuous Improvement:** Set up an automated pipeline for regular model updates and retraining with new data.
8. **Error Analysis and Uncertainty Estimation**: Analyze prediction errors and estimate uncertainty in the forecasts.
9. **Real-Time Forecasting:** Develop a system for generating real-time forecasts based on the latest data.
10. **Integration with Demand Response:** Integrate forecasting with demand response programs to optimize electricity consumption.
11. **Visualization and Reporting:** Create interactive dashboards and automated reports for stakeholders.

Concept Explanation :

Alright, let's dive into the magical world of the Random Forest algorithm! ✨

Imagine you are in a forest , and you want to predict whether a fruit you've just found is an apple or an orange . But you're not an expert in fruit identification, and there are so many different types of fruits and trees around! ♂

Luckily, you have some friends who are really good at this game. Each of them specializes in identifying a particular fruit or tree. One friend is the "Apple Expert" , another is the "Orange Guru" , and so on.

Now, you're faced with a tough decision. Whose opinion should you trust the most? Or should you take the average of all their opinions? ♀

Here's where the magic of Random Forest comes in!

The Random Forest algorithm works just like this forest scenario. It's a group of decision trees , each specializing in making predictions for a specific part of the data. But instead of asking just one friend (i.e., one decision tree), it asks a whole bunch of them!

Let's say we have a dataset with information about fruits, like color, size, and shape. Each decision tree in the Random Forest gets to look at a random subset of this data and make its own prediction about whether the fruit is an apple or an orange.

Now, the beauty of this algorithm lies in its wisdom of crowds ♂. Each decision tree is like an opinionated friend, but together, they form a powerful ensemble. They vote on the most popular choice: whether the fruit is an apple or an orange.

So, instead of relying on just one friend's potentially biased opinion, we let all the decision trees vote on the final prediction! The majority wins! And this process leads to a more reliable and accurate prediction.

But wait, there's more! The Random Forest algorithm is not only smart but also clever! It avoids overfitting, which is like wearing oversized shoes that don't fit. Overfitting happens when a model is too specific to the training data and can't handle new data it hasn't seen before.

To prevent overfitting, the Random Forest algorithm adds an extra sprinkle of randomness . Each decision tree only sees a random subset of the features (like color, size, or shape). So, they don't become too fixated on just one feature, making them more adaptable to different fruits.

By combining the opinions of many decision trees and their clever avoidance of overfitting, the Random Forest algorithm becomes a powerful fruit identifier . It can handle a wide variety of fruits with ease, making it one of the most popular algorithms in the machine learning forest! ♀

So next time you're in the forest of data, wondering if that fruit you found is an apple or an orange, remember the magical Random Forest and its ensemble of decision trees! ✨

Exercise Questions :

Question 1: What is the main objective of the Electricity Demand Forecasting project?

Answer 1: The main objective of the Electricity Demand Forecasting project is to predict the future electricity demand in the Spanish market using historical data.

Question 2: Why is the dataset preprocessed and what are the major steps involved in the preprocessing phase?

Answer 2: The dataset is preprocessed to clean and transform the data into a suitable format for analysis. The major steps involved in preprocessing are removing irrelevant columns, handling missing values, converting date-time data, and resampling the data to a daily frequency.

Question 3: How is the data visualized to gain insights into the energy demand patterns?

Answer 3: The data is visualized using various plots such as line plots, bar plots, box plots, and scatter plots to understand the trends, seasonality, and distribution of energy demand over time.

Question 4: What is the significance of feature engineering in this project?

Answer 4: Feature engineering is crucial as it involves creating relevant features from the existing data that can improve the model's predictive performance. It includes lag features, moving averages, and dummy variables for months and weekdays.

Question 5: Explain the concept of the Random Forest algorithm used for forecasting energy demand.

Answer 5: The Random Forest algorithm is an ensemble of decision trees. It aggregates the predictions of multiple decision trees to make a final prediction. Each tree receives a random subset of the data and features to reduce overfitting and improve accuracy.

Question 6: How does the algorithm handle the issue of overfitting in the model?

Answer 6: The Random Forest algorithm tackles overfitting by using random subsets of data and features for each decision tree. This ensures that no single tree becomes too specific to the training data, resulting in a more generalized and robust model.

Question 7: What is the purpose of evaluating the model using TimeSeriesSplit cross-validation?

Answer 7: TimeSeriesSplit cross-validation is used to evaluate the model's performance on time-ordered data. It splits the dataset into multiple training and validation sets, simulating real-world forecasting scenarios.

Question 8: How are the most important features identified in the Random Forest model?

Answer 8: The feature importances are calculated based on how much each feature contributes to the model's performance. The higher the importance score, the more critical the feature is in making accurate predictions.

Question 9: What are the metrics used to evaluate the forecasting accuracy of the model?

Answer 9: The Root Mean Squared Error (RMSE) and Mean Absolute Percent Error (MAPE) are the metrics used to evaluate the forecasting accuracy. RMSE measures the prediction error, while MAPE measures the percentage error.

Question 10: What are the potential future enhancements for this Electricity Demand Forecasting project?

Answer 10: Potential future enhancements include trying other forecasting algorithms like ARIMA or LSTM, incorporating external factors like weather data, and implementing more advanced feature engineering techniques to capture more complex patterns in the data. Additionally, optimizing hyperparameters and exploring ensemble methods can further improve the model's performance.