THE ENERGY CONSUMPTION DASHBOARD REPORT

Designing the User Interface

The dashboard was developed to enable users to explore and model energy consumption data using an interactive linear regression approach. The primary objective was to create an intuitive and accessible interface that guides the user through data exploration, model building, diagnostics, and prediction. The dashboard is created in R Shiny with shinydashboard package with custom CSS for accessibility.

Inputs Chosen:

- Select Inputs for Variables:
    - Exploratory Data Analysis (EDA): Users choose X and Y variables (Temperature, Humidity, HVACUsage, LightingUsage, etc.) via dropdown menus to generate plots.
    - Model Building: A multi-select input allows users to choose predictors (such as Temperature, Humidity, SquareFootage, Occupancy, HVACUsage, LightingUsage, RenewableEnergy).
    - Prediction: Numeric inputs are provided for continuous variables and radio buttons are used for categorical variables (HVACUsage and LightingUsage). These radio buttons restrict input to valid values ("Off" or "On"), ensuring data consistency.
- Action Buttons:
    - "Generate Plot" in the EDA tab triggers reactive visualisations.
    - "Build Linear Model" in the Model Building tab creates the regression model from the selected predictors.
    - "Clear All Predictors" resets the selections.
    - "Predict Energy Consumption" in the Prediction tab uses the built, or default, model to provide predictions.

Outputs Chosen:

- Interactive Visualisations:
    - Plotly Outputs in the EDA tab provide dynamic scatter plots or histograms.
    - Model Summary: The built model's output is displayed in text format, showing coefficients, p-values, and diagnostic statistics.
    - Diagnostic Plots: Interactive residual and Q-Q plots are rendered to assess model assumptions.
    - Prediction Output: The predicted energy consumption is displayed based on user-provided inputs. Please note that a custom model needs to be

built first to have interactions with the predictions output, otherwise a default one would be used instead.

Reactivity

- Data Reactivity:
    - The dataset is loaded and processed reactively, ensuring that any change in the given CSV file updates all subsequent calculations.
- Model Reactivity:
    - When a user selects predictors and clicks “Build Linear Model,” a reactive value stores the new model.
    - A default model (using Temperature and Humidity) is built automatically if the user does not build a custom model, ensuring that the Prediction tab always functions.
- Dynamic Plot Generation:
    - Changing the variable selections in the EDA tab automatically updates the plots without reloading the entire page.
    - The diagnostics and prediction outputs react to any changes in the model or user inputs.

Interactivity and Design Choices

- User Interface Layout:
    - The dashboard is divided into clear tabs (Home, User Guide, EDA, Model Building, Diagnostics, Prediction) to separate tasks.
    - Custom CSS styling is used to create a consistent theme (dark blue header and sidebar with light content areas) that enhances readability and aesthetic appeal.
    - tags$style(HTML was also used to show adaptability. Although just the “rshiny-customisations.css” could be used to keep the code cleaner.
- Interactive Elements:
    - Plotly is employed for interactive visualisations, allowing users to zoom, hover, and explore data details.
    - Dropdowns, radio buttons, and action buttons guide user inputs, minimising errors and ensuring smooth operation.
- Reactivity and Feedback:
    - The app uses notifications to inform users when a custom model isn’t built, defaulting to a pre-set model.

- Debug outputs (printed in the console) help troubleshoot any issues with predictor names and new data consistency.

Insights Gained and Future Improvements

Insights Gained:

Working with Shiny's reactive side has significantly improved my understanding of dynamic data visualisation and model building in R. Additionally, integrating Plotly enabled the creation of interactive plots that greatly enhanced the overall user experience. Moreover, custom CSS styling played a crucial role in achieving a professional, accessible design that aligns with our branding guidelines.

Accessibility Checks:

The dashboard was designed with accessibility in mind. For example, the header image includes an alt attribute ("Sun-energy sphere icon") to ensure that screen readers can accurately describe the visual element. Custom CSS rules have been applied to guarantee that the header, sidebar, and content backgrounds provide sufficient contrast with the text, and the design has been verified using the shinya11y package to ensure that all interactive elements and plots meet accessibility standards. Media queries are used to adjust font sizes on smaller screens, ensuring that the dashboard remains readable across other platforms.

Potential Future Improvements:

Advanced EDA features could be achieved by including additional plot types, such as boxplots and bar charts for extra analysis, and by adding data filtering options. Furthermore, interactive legends and tooltips would help guide users more effectively through the data exploration process. Advanced model diagnostics might include extra measures, such as Cook's distance, to provide deeper insights into model performance. In terms of prediction enhancements, this tab could be extended to include confidence and prediction intervals, giving users a better understanding of uncertainty, and a textual interpretation could be provided, for example, "Based on your inputs, the model predicts an energy consumption of X kWh." Finally, accessibility improvements should continue to be a focus by refining ARIA attributes, and contrast ratios. Data should be continuously updated, perhaps with new columns with relevant variables to allow for a potentially stronger model. Overall, these steps will make the dashboard more intuitive for future use.