EV Range Prediction Project

1. Dataset Description

The project is based on the ‘`electric_vehicles_spec_2025.csv`’ dataset, which contains comprehensive technical specifications for various electric vehicle models. This dataset provides a comprehensive collection of specifications and performance metrics for modern electric vehicles (EVs). It is designed to support researchers, analysts, students, and developers working on data science, machine learning, automotive market research, sustainability studies, or electric vehicle adoption analysis. Each row in the dataset represents a specific electric vehicle model with a rich set of attributes covering:

Core Attributes:

- **Brand and Model:** Manufacturer and specific nameplate of the EV.
- **Car Body Type:** Classification such as hatchback, SUV, sedan, etc.
- **Segment:** Vehicle segment (e.g., compact, midsize, executive).

Battery & Range:

- **Battery Capacity (kWh):** The gross energy capacity of the battery.
- **Number of Cells and Battery Type:** Technical battery information, where available.
- **Efficiency (Wh/km):** Power consumption rate of the vehicle.
- **Range (km):** Estimated driving range on a full charge.

Charging Details:

- **Fast Charging Power (kW):** Maximum supported DC fast-charging power.
- **Fast Charge Port Type:** Connector standard (e.g., CCS, CHAdeMO).

Performance:

- **Top Speed (km/h):** Maximum speed of the vehicle.
- **0–100 km/h Acceleration (s):** Time to reach 100 km/h from a standstill.
- **Torque (Nm):** Maximum torque output, where available.

Practical Specs:

- **Towing Capacity (kg):** Ability to tow loads, provided where applicable.
- **Cargo Volume (L):** Luggage space, sometimes approximate or expressed in alternative units.
- **Seats:** Total seating capacity.

Dimensions:

- **Length, Width, Height (mm):** Physical footprint of the vehicle.

Technical Information:

- **Drivetrain:** Powertrain configuration (e.g., AWD, RWD, FWD).
- **Source URL:** Reference link for each car (used in scraping).

2. Data Cleaning and Transformation Steps

The data preparation phase involved addressing missing values, creating a new feature, and scaling the final input features.

Data Cleaning and Imputation

- The initial dataset contained null values, notably in `number_of_cells`, `torque_nm`, and `towing_capacity_kg`.
- The project included steps for addressing these null values and converting object-type columns like `cargo_volume_l` to numeric types.
- Mean imputation was used in columns which had large number of null values whereas the ones with fewer null values were dropped.
- In heavily skewed distributions, winsorization was used to handle the outliers.

Feature Engineering

- A new feature, **`volume_m`**, was engineered by calculating the physical volume of the vehicle using its dimensions: `(length_mm * width_mm * height_mm) / 10^9`. This helped us to use one data column instead of three.
- The column '`cargo_volume_l`' was initially of object datatype. It was converted into numeric type and stored in a new column.

Feature Selection and Scaling

Out of all of the features, only those features that can have an effect on the range of the vehicle are selected. From the numerical features, we will only choose those which have high correlation with range.

- **Final Predictors:** The deployed model in `app.py` was built using a subset of five continuous features selected for their strong predictive power and availability in the application interface:
    1. `top_speed_kmh`
    2. `battery_capacity_kWh`
    3. `torque_nm`
    4. `acceleration_0_100_s`
    5. `fast_charging_power_kw_dc`
- **Scaling:** The chosen features were scaled using the **`StandardScaler`** from scikit-learn. This technique centers the data around a mean of zero with a unit standard deviation.
- **Persistence:** The **fitted** `StandardScaler` object was saved as `scaler.pkl` using the `pickle` library to ensure that live application inputs are scaled using the identical parameters learned from the training data.

3. EDA Insights (Exploratory Data Analysis)

Many different charts and plots are used for EDA showing relationships as well as the distributions of the features. The libraries like matplotlib and seaborn are used for visualizations. Below are some of the findings of the analysis:

- **Drive type Distribution:** AWD vehicles are most abundant at 40.13% followed by FWD and RWD at 32.56% and 27.31% respectively.
- **Brand counts:** Mercedes Benz has the highest number of models at 42 whereas a lot of companies only have 1 model like Omoda, Subaru, Honda and many more.
- **Segment count:** Medium segment vehicles are most abundant which constitutes of 19% whereas luxury and sport segment are the lowest.
- **Car body type:** A majority of the electric vehicles are SUV and just a handful are Coupe and cabriolet types.
- **Top speed and range:** Out of the three drivetrain types, AWD gives higher top speed. The average range as well as max range is greater than those of FWD and RWD. In terms of range, AWD and RWD have similar metrics like average and max range. FWD gives significantly lesser range than the rest.
- **Correlation:** Range is highly correlated to top speed, battery capacity, torque, acceleration time and fast charging power.

4. Model Development, Evaluation, and Deployment Process

Model Development and Evaluation

1. **Model Selection:** The notebook file explored multiple regression models like linear regression, ridge regression, lasso regression, decision tree regression and many more. Out of all these models, **Linear Regression** and **Random Forest Regressor** turned out to have the best metrics. The process involved evaluating models based on metrics like:
    - **R2 Score:** Measures the proportion of the variance in the dependent variable that is predictable from the independent variables.
    - **Adjusted R2 Score:** A modified R2 that accounts for the number of predictors.
    - **Overfitting Gap:** The difference between the R2 on the training set and the test set, used to identify overfitting.
2. **Final Model Selection:** After the initial model performance analysis, hyperparameter tuning was done on the two best models and those models were tested again with best parameters to see which one performs better. The Random Forest Regressor was seen to be the best model after final model performance metrics and it was used in deployment in the '`app.py`' file.
3. **Model Persistence:** The final model object (`best_model`) was saved using `pickle` as either '`/RF_regressor.pkl'` and then loaded by the app using the path defined in '`secrets.toml'`.

Deployment Process

The model is deployed as a user-friendly interactive web application using **Streamlit**.

Step	Mechanism / Code	Description
Resource Loading	`load_resources()` with `@st.cache_resource`	The function loads the saved model (`RF_regressor.pkl`) and the `scaler.pkl` only once, improving app performance. File paths are managed via `secrets.toml`.
Input Interface	`st.sidebar.slider`	The application provides five interactive sliders for the user to input the required features (`top_speed_kmh`, `battery_capacity_kWh`, `torque_nm`, `acceleration_0_100_s`, and `fast_charging_power_kw_dc`).
Prediction Pipeline	`scaler.transform(input_df)` and `model.predict(...)`	The live user inputs are converted into a DataFrame. This DataFrame is then correctly scaled using the loaded **`scaler's .transform()`** method. The model generates the prediction, which is displayed using a Streamlit metric.
Output Display	`st.metric`	The predicted range in kilometers is displayed prominently on the main page of the application.

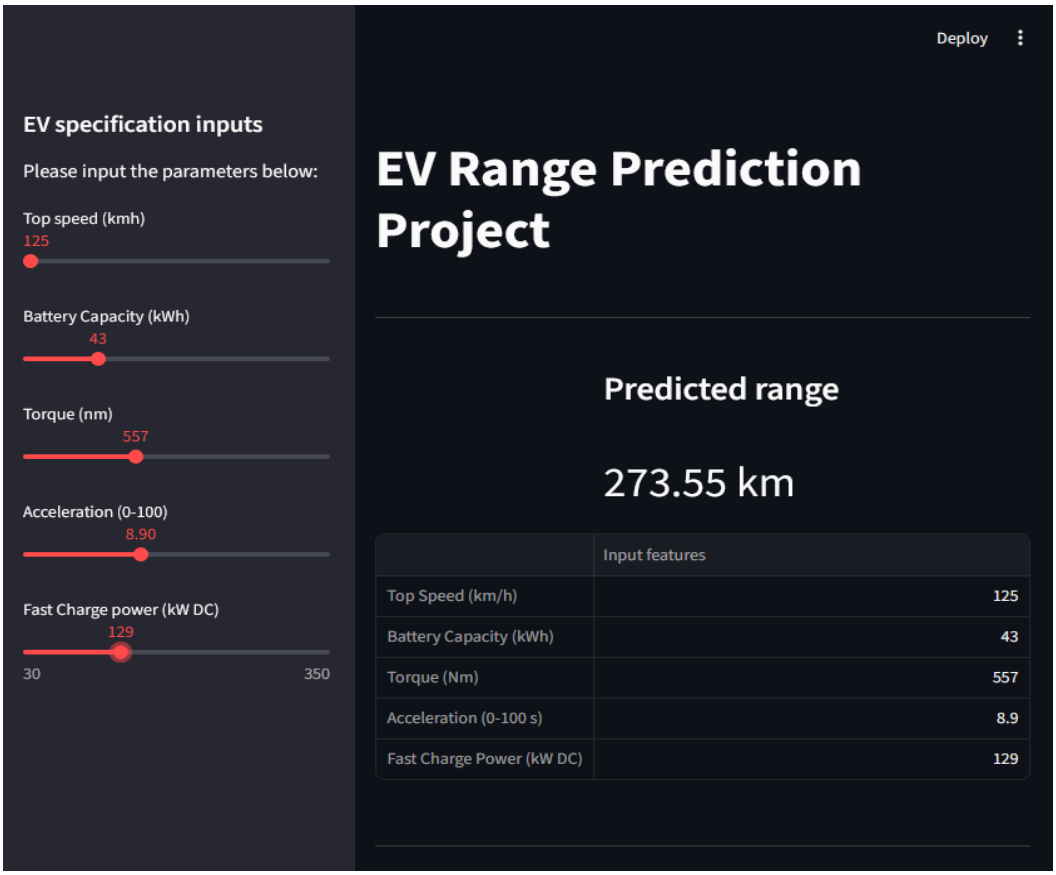

Figure 1: Streamlit app interface