**Model is Trained using following versions**

click 8.1.7

Flask 3.0.3

Flask-Cors 4.0.1

gunicorn 22.0.0

itsdangerous 2.2.0

Jinja2 3.1.4

MarkupSafe 2.1.5

numpy 2.0.1

pandas 2.2.2

pickle-mixin 1.0.2

python-dateutil 2.9.0

pytz 2024.1

scikit-learn 1.5.1

scipy 1.14.0

six 1.16.0

threadpoolctl 3.5.0

Werkzeug 3.0.3

jupyter notebook 7.2.1

Matplotlib 3.9.1

Seaborn 0.13.2

**Dataset :** quikr\_car.csv [Kaggle link : <https://www.kaggle.com/datasets/laxminarayanasahu/quikr-car> ]

**NOTE : If you want to change model just change .pkl file in application.py**

**Work Flow :**

**User Accesses the Web App**:

* The user opens their web browser and navigates to the CarValue.AI application.
* They are greeted with a clean and user-friendly interface where they can input their car’s details.

**User Inputs Car Details**:

* The user selects the car’s company (e.g., Toyota, Honda) from a dropdown menu.
* They then choose the specific car model (e.g., Corolla, Civic) from another dropdown.
* The user selects the year the car was purchased and its fuel type (e.g., Petrol, Diesel).
* Lastly, they enter the total kilometers the car has been driven.

**Form Submission**:

* After filling in all the details, the user clicks a button to submit the form.
* Behind the scenes, JavaScript gathers all the information the user has entered.
* Instead of reloading the page, JavaScript sends this information directly to the server using AJAX.

**Data Sent to the Server**:

* The data is sent to the Flask application (which runs on the server).
* The Flask app receives the data and begins processing it.

**Data Formatting**:

* The Flask application organizes the user’s input into a format that the machine learning model understands.
* It arranges the details into a table-like structure with columns representing the car’s name, company, year, kilometers driven, and fuel type.

**Loading the Machine Learning Model**:

* The Flask app loads the pre-trained Random Forest model from a file (RandomForestModel.pkl).
* This model has been trained on a large dataset of used car prices and is capable of predicting the resale value of a car based on the user’s input.

**Making a Prediction**:

* The formatted user data is passed into the Random Forest model.
* The model analyzes the input data and predicts what the car’s resale value would be in the current market.

**Sending the Prediction Back to the User**:

* The Flask app receives the predicted price from the model.
* It then sends this predicted price back to the frontend (the user’s browser) through the AJAX response.

**Displaying the Predicted Value**:

* JavaScript on the web page receives the predicted car value from the Flask app.
* Without refreshing the page, JavaScript updates the web page to show the user the predicted price.
* The user sees the car’s estimated resale value displayed on the screen.

**User Interaction Ends**:

* The user can either use the predicted value to make decisions about selling their car or modify their input to see how changes affect the car’s value.
* The entire process is smooth, fast, and interactive, with no need to reload the page or wait for long periods.