**Car Price Prediction Project Documentation**

**1. Project Overview**

This project involves building a machine learning application to predict the prices of used cars based on various attributes. The application uses historical car sales data from multiple cities to train a model. The trained model is deployed in a Streamlit web application that allows users to input car details and view predicted prices.

**2. Data Collection and Loading**

* **Data Sources**: Data was collected from multiple Excel files corresponding to different cities (bangalore, chennai, delhi, hyderabad, jaipur, kolkata).
* **Loading Process**: Data is loaded using pd.read\_excel() for each city, adding a 'City' column to identify the origin. All city data frames are concatenated into a single combined\_df DataFrame.

**3. Feature Extraction and Cleaning**

* **Function**: extract\_features(row)
  + This function parses JSON-like string fields (new\_car\_detail, new\_car\_overview, new\_car\_specs) to extract:
    - **Car Attributes**: Fuel\_Type, Body\_Type, Transmission, KM\_Driven, Model\_Year, Price, Engine, Brand, Model, Registration\_Year.
* **Data Cleaning**:
  + Converts KM\_Driven and Engine to numerical formats by removing commas and units.
  + Parses Price values, converting units (Lakh, Crore, etc.) to a common format in Lakh.
  + Fills missing values:
    - Registration\_Year with Model\_Year.
    - Price with the median price.
  + Drops rows with missing Engine values.
  + Removes duplicate rows.

**4. Data Preprocessing**

* **Target Variable**: Price is set as the target variable, while all other fields are used as features (X).
* **Categorical Columns**: Fuel\_Type, Body\_Type, Transmission, Brand, City.
* **Numerical Columns**: KM\_Driven, Engine, Registration\_Year, Model\_Year.
* **Column Transformer**: Combines StandardScaler for numerical columns and OneHotEncoder (with handle\_unknown='ignore') for categorical columns.

**5. Model Training and Evaluation**

* **Train-Test Split**: Data is split into training and testing sets with an 80-20 ratio.
* **Models Used**:
  + LinearRegression
  + RandomForestRegressor
  + DecisionTreeRegressor
  + GradientBoostingRegressor
* **Evaluation Metrics**: Each model is evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and R² score on the test set.
* **Hyperparameter Tuning**:
  + Performed on RandomForestRegressor using GridSearchCV.
  + Tuned parameters: n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf, max\_features.
* **Results**:
  + Displays the best parameters and R² score for the optimized Random Forest model.
  + Saves the best model as best\_rf\_model.pkl.

**6. Model Deployment with Streamlit**

* **User Interface**:
  + **Sidebar Inputs**: Allows users to input car details, including Fuel\_Type, Body\_Type, Transmission, KM\_Driven, Model\_Year, Engine, Brand, City, and Registration\_Year.
  + **Body Type Images**: Displayed based on the selected body type using body\_type\_images dictionary mapping.
* **Prediction**:
  + When the "Predict Price" button is clicked, user input is transformed using the pre-trained preprocessor, and the model predicts the car price.
* **Output**: Displays the predicted price in Lakh.

**7. Saving and Loading Models and Preprocessors**

* **Pickling**: The best model and preprocessor are saved using pickle for reuse in the Streamlit app.
* **Loading in Streamlit**: The best\_rf\_model.pkl and preprocessor.pkl files are loaded to transform new inputs and predict prices.

**8. Potential Improvements**

* **Data Enhancements**: More granular information could improve prediction accuracy, such as specific car features (e.g., safety ratings, mileage).
* **Model Tuning**: Further tuning of hyperparameters and experimenting with other regression models could yield improved accuracy.
* **App Improvements**: Adding more image options and additional filtering for users (e.g., specifying mileage) would improve user interaction.