Conversational Used Car Price Predictor

CS702 - Computing Lab

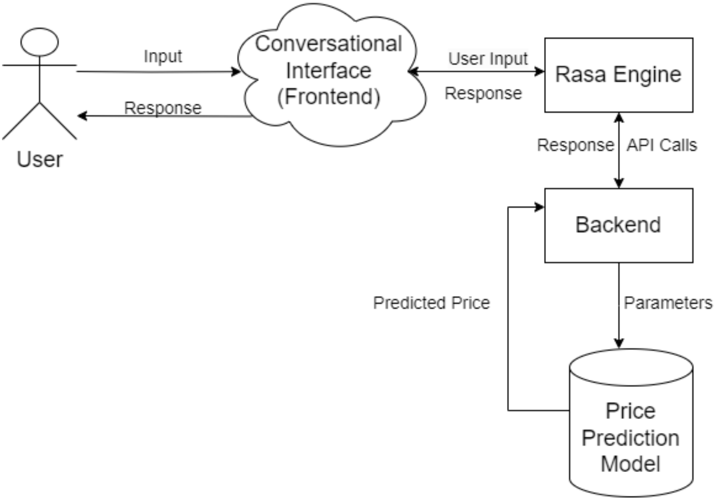
ABHIJITH C   ANAND M K

Department of Computer Science and Engineering
NITK Surathkal

Introduction

- This project focuses on developing a **Conversational Used Car Price Predictor**, integrating a chatbot interface with a machine learning model.
- **Goal:** To allow users to interact through natural conversation rather than filling out traditional forms to predict used car prices.
- The chatbot will collect necessary car details (brand, model, year, mileage, etc.) step by step through an intuitive and engaging interface.
- A machine learning model will use the gathered data to predict the price of the used car, ensuring accurate and reliable predictions.
- The chatbot also handles additional queries, such as explaining how the price was calculated or what factors affect the car's value.

System Architecture



,

System Flow Explanation

- The user interacts with the system via a chat interface in the frontend, initiating a conversation to predict the price of their used car.
- The user inputs (car details such as make, model, year, mileage, etc.) are processed by the **Rasa Engine**, which handles NLP and conversation management.
- The Rasa Engine extracts relevant parameters from the conversation and sends them to the backend, where the **Price Prediction Model** calculates the car's estimated price based on the input features.
- **SHAP** (SHapley Additive exPlanations) is applied to explain the contribution of each feature (e.g., mileage, car age, etc.) to the final price prediction, identifying the most impactful factor.
- The backend returns both the predicted price and the explanation of the most significant contributing factor, which are displayed to the user through the chat interface.

Project Overview: Progress and Next Steps

- **Milestones Achieved:**
  - Data collection and preprocessing for the car price prediction model.
  - Model development using **Random Forest** and SHAP calculation for feature contribution analysis.
  - Backend API to return the predicted price and the maximum contributing feature.
  - Development of a sample chatbot to collect car parameters from the user.
- **Upcoming Work:**
  - Improve the Rasa chatbot by adding more test data.
  - Validation of user input parameters in the Rasa chatbot.
  - Handle general questions in Rasa.
  - Integration of the backend with Rasa.
  - Frontend integration with Rasa for seamless user interaction.

Data Preprocessing Steps (Part 1)

- **Finding the Dataset:**
  - Dataset sourced from Kaggle: `cardekho_dataset` in 2023.
- **Dropping Unnecessary Columns:**
  - Removed columns such as `"Unnamed: 0"` (irrelevant index) and `"car_name"` (redundant feature).
- **Converting Vehicle Age to Year of Manufacture:**
  - Calculated `year_of_manufacture` using the formula:

$$\texttt{year\_of\_manufacture} = 2023 - \texttt{vehicle\_age}$$

  - Dropped the `vehicle_age` column.
- **Filtering Popular Car Models:**
  - Filtered the dataset to retain models with more than 300 entries, ensuring sufficient data for model training.

Data Preprocessing Steps (Part 2)

- **Feature (X) and Target (y) Definition:**
  - Defined `X` (features) by dropping `selling_price` and `seller_type`.
  - Defined `y` as the `selling_price` (target variable).
- **Categorical Columns:**
  - Identified categorical features like `fuel_type`, `transmission_type`, `brand`, and `model` that need encoding.
- **Data Splitting:**
  - Split the dataset into 80% training and 20% test sets using `train_test_split()`, ensuring reproducibility with `random_state=42`.
- **Preprocessing Pipeline (One-Hot Encoding):**
  - Used `ColumnTransformer` to apply `OneHotEncoder` to categorical columns, converting them into binary variables for model compatibility.

Model Training Process (Part 1)

- **Model Selection:**
  - A **Random Forest Regressor** was chosen for car price prediction.
  - The model is robust, handles non-linear data well, and is less prone to overfitting.
- **Pipeline Setup:**
  - A pipeline was created to seamlessly combine preprocessing and model training.
  - Ensures that the same transformations are applied during both training and testing.
- **Hyperparameter Tuning:**
  - **RandomizedSearchCV** was employed to efficiently search for the best hyperparameters.
  - This method randomly samples a specified number of hyperparameter combinations from the defined parameter grid.

Model Training Process (Part 2)

- **Hyperparameter Tuning (continued):**
  - Parameters tuned included:
    - `n_estimators`: Number of trees in the forest (100, 200, 300).
    - `max_depth`: Maximum tree depth (None, 10, 20, 30).
    - `min_samples_split`: Minimum samples to split (2, 5, 10).
    - `min_samples_leaf`: Minimum samples at each leaf (1, 2, 4).
  - **n_iter=10**: Specifies the number of different combinations to try.
- **Cross-validation:**
  - **5-fold cross-validation** was used to evaluate model performance.
  - This process repeats 5 times, with each fold serving as the test set once.
  - Final performance is averaged over the 5 iterations for robust evaluation.
- **Results:**
  - The best model was selected based on the **R^2 Score: 0.925** on the test set.
  - The best model was saved for future predictions.

SHAP Value Calculation

- **SHAP Value Calculation:**
  - SHAP (SHapley Additive exPlanations) is a method used to explain the output of machine learning models by assigning each feature a contribution value for a particular prediction.
  - An `Explainer` object is created for the trained Random Forest model using SHAP.
  - SHAP values are computed for the transformed input data, representing how much each feature pushed the prediction higher or lower than the average model output.
- **Analyzing SHAP Values:**
  - SHAP values are organized into a DataFrame, providing a clear breakdown of each feature's contribution to the predicted price.
  - Positive SHAP values indicate features that push the predicted price higher, while negative values indicate features that pull the price lower.

Identifying Influential Features

- **Identifying the Most Influential Feature:**
  - The feature with the highest SHAP value is identified as the most influential factor in determining the predicted price for that specific car.
  - This feature typically has the largest impact on driving the predicted price up or down.
- **Percentage Contribution Calculation:**
  - The SHAP value of the most influential feature is divided by the predicted price to calculate its percentage contribution.
  - This provides a more intuitive understanding of how much influence the top feature has on the final predicted price.
- **Feature Attribution:**
  - The system outputs both the predicted price and the feature with the highest contribution, along with the percentage of its influence.
  - This gives users a clear explanation of why the price is what it is, based on the input features.

Backend API Creation

- **Framework Used:** Flask
  - A lightweight web framework for Python, used to create web applications.
- **Endpoints:**
  - `/predict_price`:
    - Accepts car attributes as query parameters.
    - Returns the predicted selling price of the car.
  - `/max_contribution`:
    - Accepts the same car attributes as query parameters.
    - Returns the highest contributing feature to the predicted price and its percentage contribution.
- **Input Handling:**
  - Retrieves input data via query parameters (GET API).
- **Response Format:**
  - Outputs predictions and contributions as JSON, facilitating easy integration with frontend applications.

Upcoming Work

- Improve the Rasa chatbot by adding more test data to enhance its understanding and responsiveness.
- Validate user input parameters to ensure accurate and reliable data collection for predictions.
- Handle general questions within the chatbot to improve user engagement and satisfaction.
- Integrate the backend with Rasa for real-time access to the price prediction model.
- Implement frontend integration with Rasa for a seamless user interaction experience.

References

- Rasa Technologies, “Rasa Documentation,” *Rasa*, 2023. [Online]. Available: `https://rasa.com/docs/`
- Armin Ronacher, “Flask Documentation,” *Flask*, 2023. [Online]. Available: `https://flask.palletsprojects.com/`
- SHAP Documentation, “SHAP (SHapley Additive exPlanations),” 2023. [Online]. Available: `https://shap.readthedocs.io/en/latest/`
- Leo Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, 2001, pp. 5-32. [Online]. Available: `https://link.springer.com/article/10.1023/A:1010933404324`
- Scikit-learn, “Random Forests,” 2023. [Online]. Available: `https://scikit-learn.org/stable/modules/ensemble.html-rand`

Thank You!