Documentation for Car Price Prediction Application

Overview

This documentation outlines the process and functionality of a Streamlit application designed to predict car prices based on various features. The application incorporates data processing, machine learning model training, evaluation, and a user-friendly interface for real-time predictions.

Features

- Upload multiple city datasets in Excel format.
- Automatic data processing to handle missing values and standardize data formats.
- Train and evaluate multiple machine learning models.
- Hyperparameter tuning for improved model performance.
- User interface for inputting features and receiving price predictions.

1. Data Processing

Function: process_data(city_files)

This function handles the data processing steps:

- **Input**: A list of uploaded Excel files containing city datasets.
- **Output**: A processed Pandas DataFrame.

Steps:

1. **Reading Datasets**: Uses pd.read_excel() to read each Excel file. If reading fails, an error message is displayed.
2. **Concatenating Datasets**: Valid datasets are combined into a single DataFrame.
3. **Handling Missing Values**:
    - Numerical columns: Fill missing values with the column mean.
    - Categorical columns: Fill missing values with the mode (most frequent value).
4. **Standardizing Data Formats**: For example, the distance column is cleaned to remove units (e.g., "kms").
5. **Encoding Categorical Variables**: Uses one-hot encoding to convert categorical features into numerical format.

2. Model Training

Function: train_model(X_train, y_train)

This function trains multiple regression models and selects the best performing one.

- **Input**: Features (X_train) and target variable (y_train).
- **Output**: The best trained model after hyperparameter tuning.

Steps:

1. **Model Selection**: Evaluates multiple models (Linear Regression, Decision Tree, Random Forest, Gradient Boosting) using cross-validation.
2. **Hyperparameter Tuning**: Uses GridSearchCV to optimize parameters for the Random Forest model.

3. Model Evaluation

Function: evaluate_model(model, X_test, y_test)

This function evaluates the performance of the trained model.

- **Input**: The trained model, test features (X_test), and test target variable (y_test).
- **Output**: A dictionary containing evaluation metrics.

Metrics:

- **MAE**: Mean Absolute Error.
- **MSE**: Mean Squared Error.
- **R^2**: Coefficient of determination.

4. Streamlit Application

Function: main()

The main function runs the Streamlit application.

Steps:

1. **Title and File Upload**: Displays the title and allows users to upload city datasets.
2. **Data Validation**: Ensures that all uploaded files are in the correct format (Excel).
3. **Process Data**: Calls process_data() to clean and prepare the dataset.
4. **Train-Test Split**: Splits the processed data into training and testing sets.
5. **Train Model**: Calls train_model() to train the best model.
6. **Evaluate Model**: Calls evaluate_model() to display model performance metrics.
7. **User Input for Prediction**: Allows users to input features for price prediction.
8. **Prediction**: When the user clicks the predict button, the model predicts the car price based on the provided inputs.

5. Saving the Model

The trained model is saved to a file (model.pkl) using Python's pickle module for later use.

Usage Instructions

1. **Run the Application**: Execute the script in a Python environment that supports Streamlit.
2. **Upload Datasets**: Use the file uploader to select and upload your city datasets in Excel format.

3. **View Model Performance**: After processing and training, review the displayed performance metrics.
4. **Make Predictions**: Input the required features and click the predict button to receive the estimated car price.

Requirements

- Python 3.x
- Streamlit
- Pandas
- NumPy
- Scikit-learn
- OpenPyXL

Conclusion

This application provides an end-to-end solution for predicting car prices using machine learning techniques. It effectively processes data, trains models, evaluates performance, and offers a user-friendly interface for predictions.