AI/ML Task Report

Muhammad Huzaifa

Task #1 – Car Price Predictions

Approach:

1. **Data Loading & Analysis:**
    - Loaded the dataset and displayed initial rows to understand the structure.
    - Checked for data types and missing values using .info() and .isnull().sum().
    - Some Price values had ‘Call for price’, which I dropped.
2. **Data Cleaning:**
    - Plotted the distribution of car prices i.e. shown below:

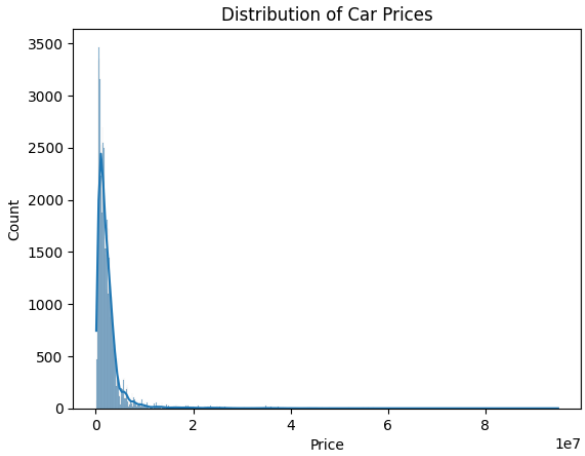

Figure 1. Shows most cars being in the price range of <1.5 Mil

    - Handled extreme values in 'CC' by replacing values outside the range 600-5000 with the median.
    - Filled missing 'Version' values with the mode of their respective 'Make' and 'Model' group.
3. **Feature Engineering:**
    - Created a new feature 'Vehicle_age' by subtracting 'Make_Year' from the current year.
    - Encoded categorical features using LabelEncoder for correlation calculation.

- Using correlation matrix, only kept the relevant features.

4. **Preprocessing Pipelines:**
    - Defined separate pipelines for numerical and categorical data:
        - Numerical: Imputation with mean and scaling.
        - Categorical: Imputation with most frequent value and one-hot encoding.
    - Combined these into a ColumnTransformer.
5. **Model Training and Evaluation:**
    - Defined and trained pipelines for three models: Linear Regression, Random Forest, and Decision Tree.
    - Evaluated models using RMSE, R^2 score, and MAPE.
    - Plotted actual vs. predicted car prices i.e. shown below:

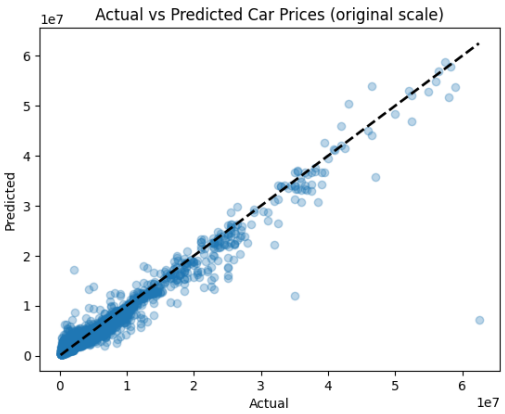

Figure 2. Showing fairly good performance of the RandomForestRegressor

6. **Model Export:**
    - Exported the best model (Random Forest) for future use.

Challenges Faced:

- Significant effort was needed to impute missing 'Version' values accurately.
- Addressing outliers in engine capacity ('CC') to avoid skewing the model predictions.
- Identifying the most relevant features for model training.
- Finding the best hyperparameters for desired model performance.

Task #2 – Movie Recommendation System

Approach

1. **Data Loading & Analysis**:
    - Loaded the dataset using pandas.
    - Checked for any missing values.
    - Checked all available datatypes.
    - Identified unique rating values and their counts.
    - Plotted histogram of unique ratings i.e. shown below:

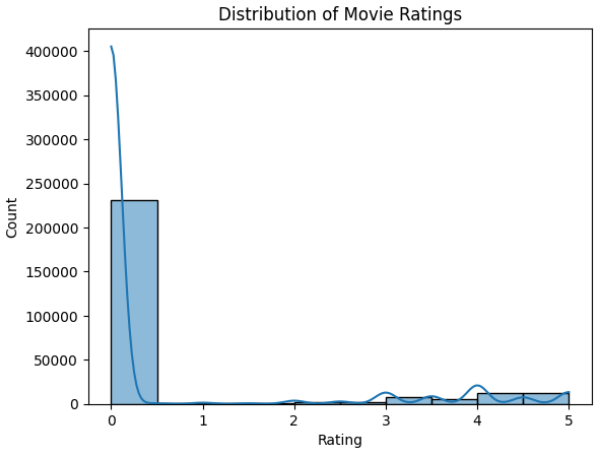

Figure 3. Shows a very high frequency of empty ratings.

2. **Model Building**:
    - Used the K-Nearest Neighbors (KNN) algorithm with cosine similarity to build a recommendation model.
    - Cosine similarity used because it performs well in case of sparse matrix, like the one in our case.
    - K value can be changed in accordance with number of movie recommendations needed.
3. **Recommendations**:
    - Developed a function to recommend movies based on a given title.
    - Demonstrated the function with two example movies, which showed correct working.

4. **Model Saving**:
    - Saved the trained KNN model using joblib.

Challenges Faced

- Had to flatten the dataframe in order to correctly visualize the histogram showing ratings.
- Deciding on the model to be used and the similarity metric required some research to come up with a good solution.