Automobile Dataset Analysis Report

- **Introduction**

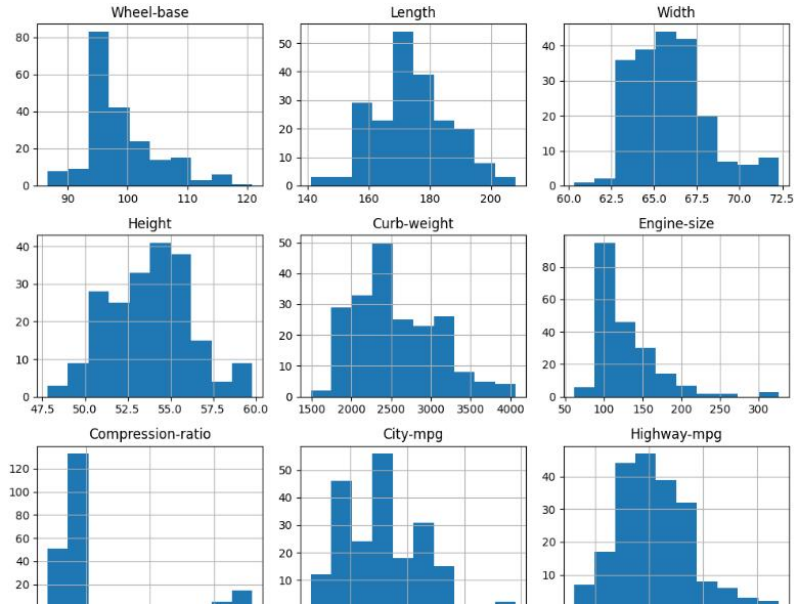
The Automobile Dataset is a multi-attribute dataset. It includes 25 features like numerical attributes (e.g., horsepower, length) and categorical attributes (e.g., fuel type, body style). The target variable ‘Symboling’ represents the car's risk rating. Through EDA analysis, I found some features and problems of the data set, including missing values, numerical characteristics skewed, outlier , imbalanced classes and strong correlation making it a good candidate for data preprocessing and advanced machine learning techniques.

- **Methodology**

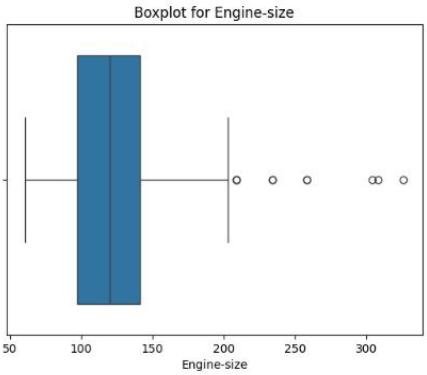
1.Data loading and parsing

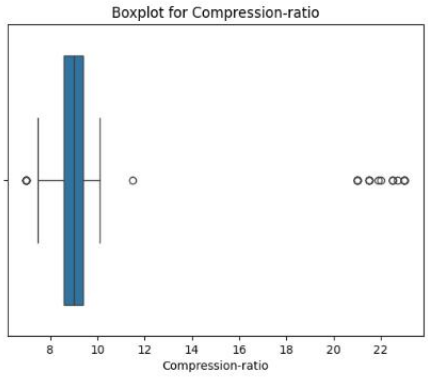
Load the data from the.dat file and parse it into Pandas DataFrame. Extract the column names and convert the data into the appropriate format by separating the metadata and data parts in the file. Code will data in the '? 'Replaced by the missing value tag' NaN ', after loading into the DataFrame. Separate the target column 'Symboling' into 'y' and the other columns as the feature 'X', and identify the numeric column and the classification column according to the data type. Finally, the code performs basic structural and statistical checks on the data set through head(), info(), and describe(), preparing it for subsequent analysis.

2. Performed EDA

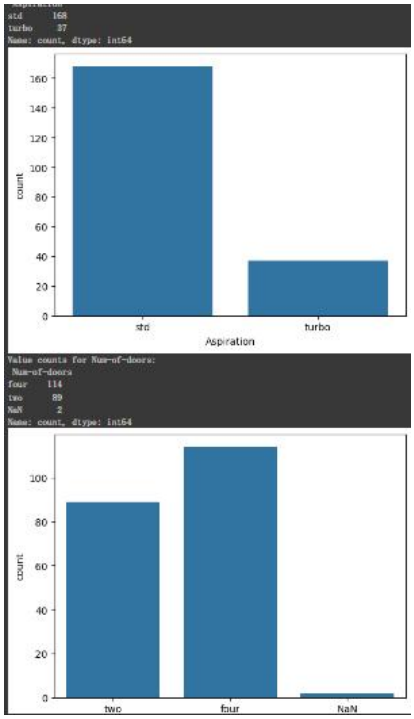


By using the distribution of histogram numerical features, quickly visualize the distribution of each numerical feature

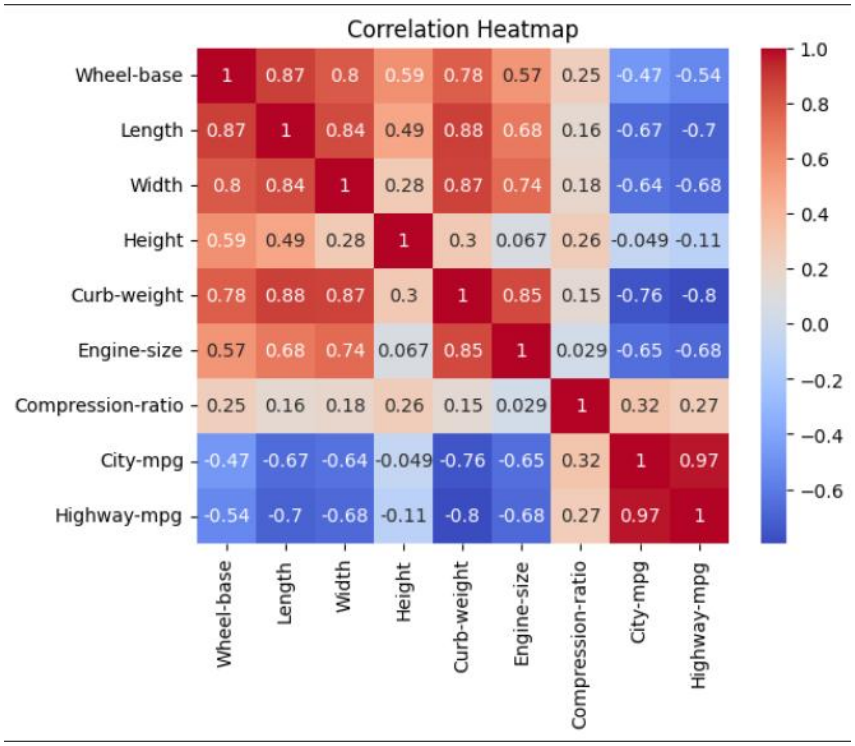




With a Boxplot, you can clearly see the distribution of each feature and its potential outliers.



By visualizing the distribution of each variable's value to understand the distribution of categories and spot data problems.



By using heatmap the linear correlation between features is visually demonstrated.

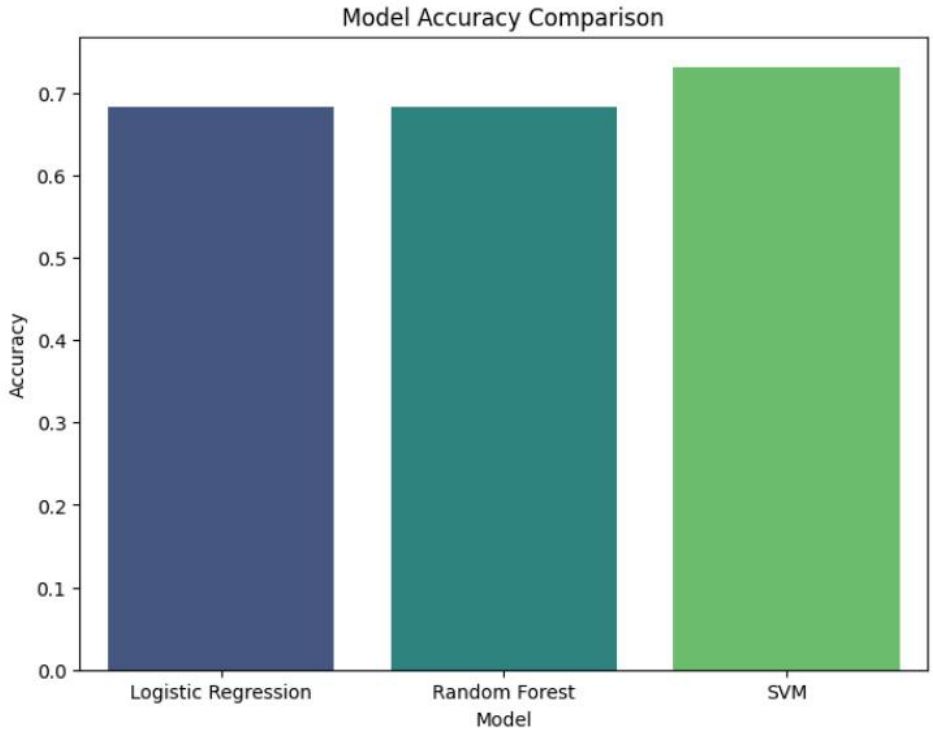
3. Preprocessed the data

The EDA analysis results show that there are missing values, numerical characteristics skewed, outlier , imbalanced classes and strong correlation between city-mpg and highway-map problems in the dataset. Therefore, I fill in the median for numerical features and standardize it, and fill in the most frequent values for classification features and perform unique thermal coding. Oversampling the feature matrix and target variables was performed using SMOTE to generate a training dataset with a more balanced class distribution, and the processed class distribution was printed to verify the balancing effect.

4. Model training

I use 'train_test_split' to split the data into a training set and a test set, 20% of which is used for testing, and I use 'stratify=y' to ensure a consistent distribution of categories for the target variables. Then, I define three models of logistic regression, random forest and support vector machine (SVM), which are stored in dictionaries to facilitate iterative training. Next, I train the models with the training set, evaluate performance on the test set, and save each model's accuracy, classification reports, and confusion matrix to the results dictionary for subsequent analysis and comparison.

5. Evaluated model performance



SVM: Best performance, highest accuracy, more than 70%.

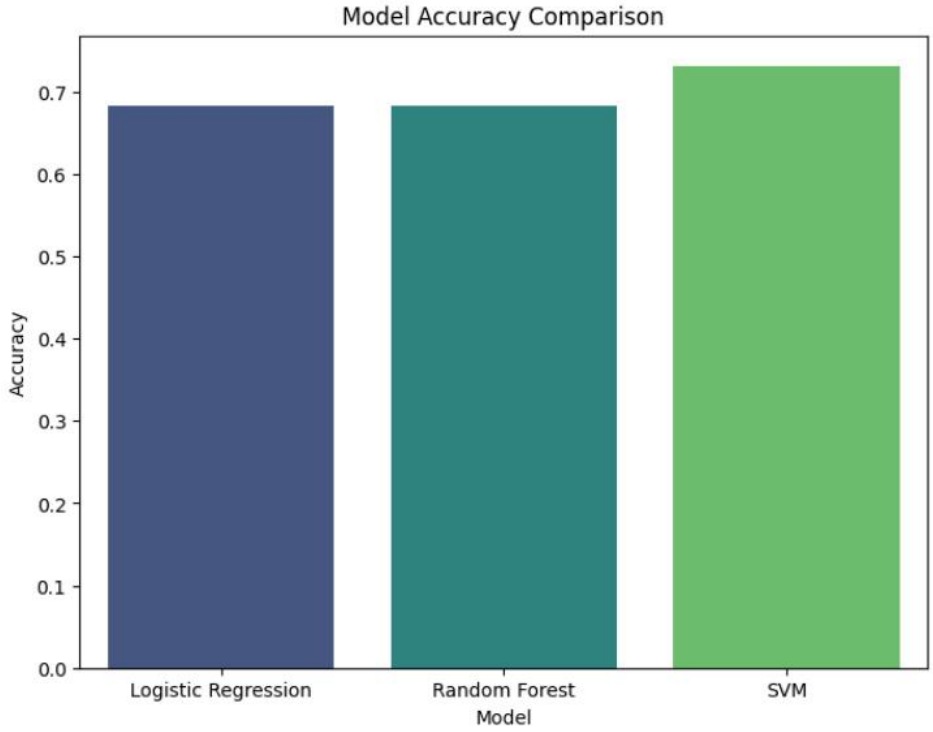
Random Forest: Slightly lower than SVM, but still close to 70%.

Logistic Regression: Relatively low performance, but not far from other models.

I use GridSearchCV for hyperparameter tuning of the random forest model. I define a parameter grid (param_grid) that includes multiple options for key hyperparameters, such as the number of decision trees (n_estimators), max_depth, and max_depth. And the minimum number of samples for dividing nodes (min_samples_split and min_samples_leaf). Through these combinations, I can explore the optimal configuration of model performance. Next, I use 3-fold cross-validation (cv=3) for hyperparameter search, accuracy as an evaluation metric, and multithreading (n_jobs=-1) to improve efficiency. After

completing the grid search, I extracted the best model (best_rf_model) and evaluated it on the test data set (X_test), calculated the tuned accuracy, and printed out the best hyperparameters and corresponding accuracy. Finally, I iterate through all the trained model results (saved in the Results dictionary), printing the accuracy, classification report, and confusion matrix for each model.

- **Results**



SVM: Best performance, highest accuracy, more than 70%.

Random Forest: Slightly lower than SVM, but still close to 70%.

Logistic Regression: Relatively low performance, but not far from other models.

The Random Forest model, after hyperparameter tuning, achieved the best performance with an accuracy of 73.17%.

- **Conclusion**

Through this project, I successfully classified automobile data and obtained some key gains: In the data preprocessing, I effectively dealt with the problems of missing values, outliers and strong correlations; Random forest and SVM perform best in model comparison. Hyperparameter tuning significantly improves the model performance. However, there are limitations to the project, such as the possible introduction of noise affecting generalization, limited treatment of strongly correlated features, and no attempt at more complex model optimization performance. These problems can be solved by more advanced feature dimensionality reduction techniques and model improvements.