Predicting Car Prices using Regression Analysis

Problem Statement: The price of a car is determined by various factors such as make, engine-type, horsepower, peak-rpm, fuel type, and so on. In this project, we aim to develop a regression model to predict the price of cars based on 25 different features provided in the ‘cars_price.csv’ data set.

Introduction: The price of a car is a crucial factor that determines the purchasing decision of buyers. In the automotive industry, accurate estimation of car prices is essential for car dealerships, insurance companies, and other businesses. This report presents a regression analysis project that aims to predict car prices using a dataset titled 'cars_price.csv.'

Dataset Details: The ‘cars_price.csv’ dataset is a regression dataset with 206 samples and 25 features. The target variable for the dataset is the price of the car. The features include make, engine-type, horsepower, peak-rpm, fuel type, and other relevant factors that affect the price of a car.

The objective of this project is to develop a regression model that can accurately predict the price of a car based on these features.

The project will involve data cleaning, exploratory data analysis, feature selection, and regression modelling.

The project's success will be measured based on the accuracy of the model's predictions on a 20% holdout set of data. We will use metrics such as mean squared error (MSE), mean absolute error (MAE), and R-squared to evaluate the performance of the model.

Key Steps Involved:

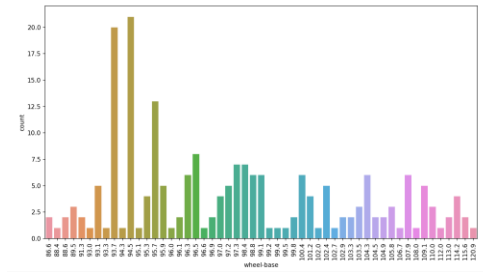
1. Loading the essential libraries and then loading and reading the dataset ‘cars_price.csv’.
2. Splitting the dataset into train data and test data with 20% of the data as test data.
3. Exploratory Data Analysis

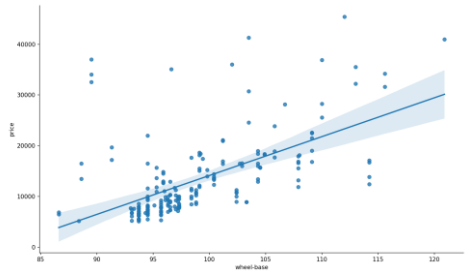
   Exploratory Data Analysis (EDA) is a crucial step in the data analysis process that involves understanding the data, identifying patterns and trends, and preparing the data for modelling. Countplot, lmplot, histplot and box plot from seaborn library were used for visualizations in EDA to understand the distribution of data and the relationship between features and target.

   Below table represents Data visualization in EDA. Left column plots help visualize and understand the distribution of data in respective features, and right column plots help understand the trend and relationship between features and target.

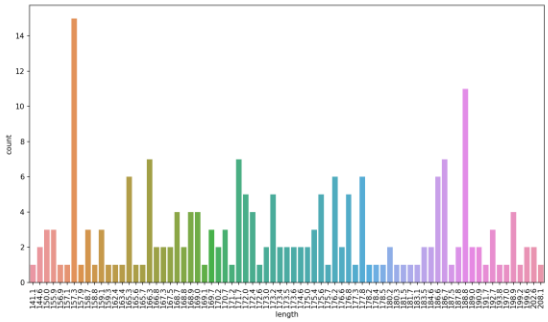
Numerical Features

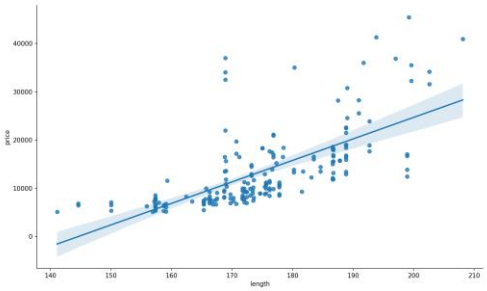
wheel-base



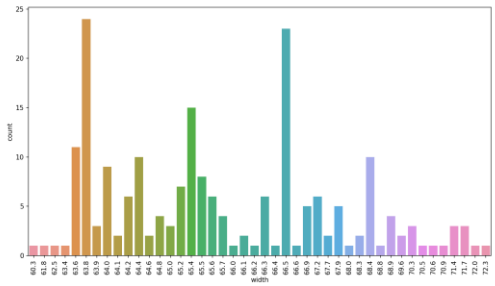


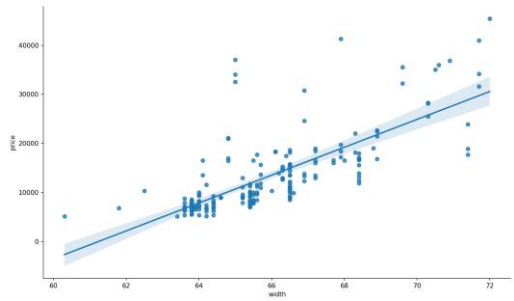
Length



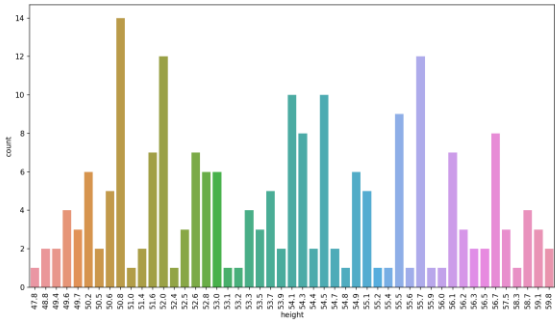


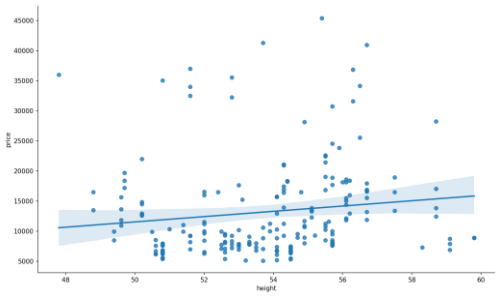
Width



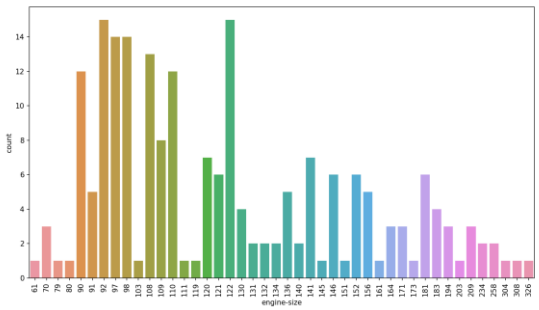


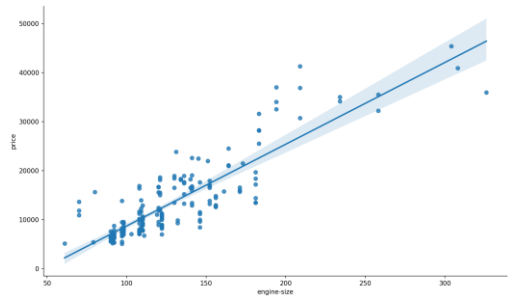
Height



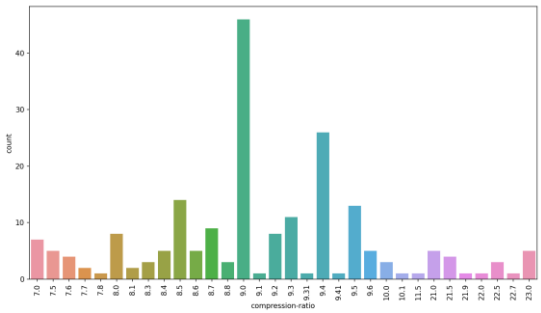


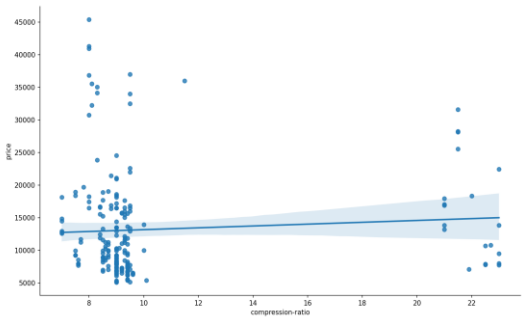
engine-size



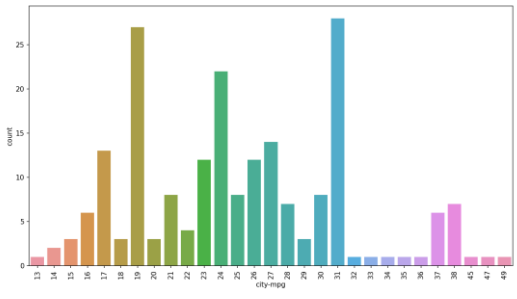


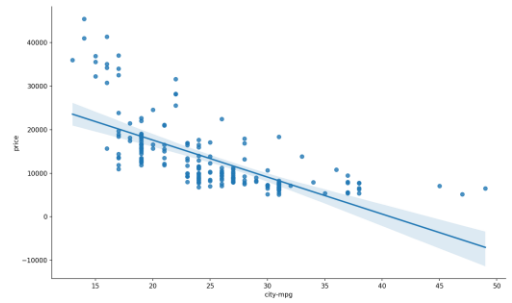
compression-ratio



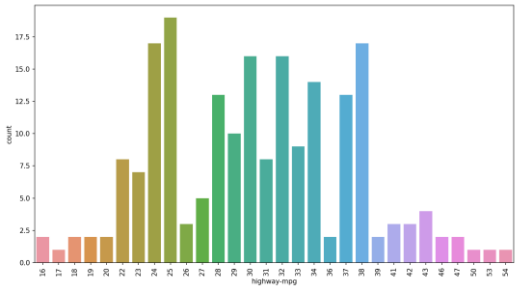


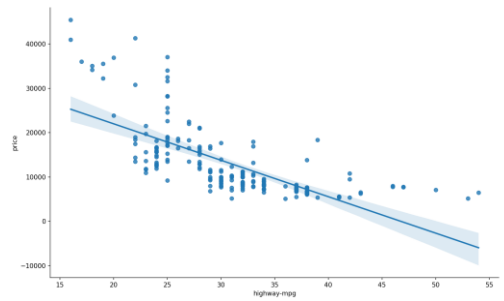
city-mpg



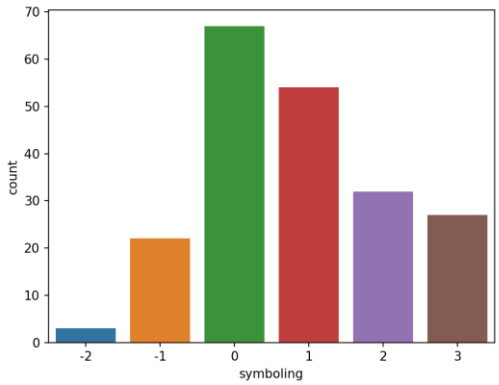


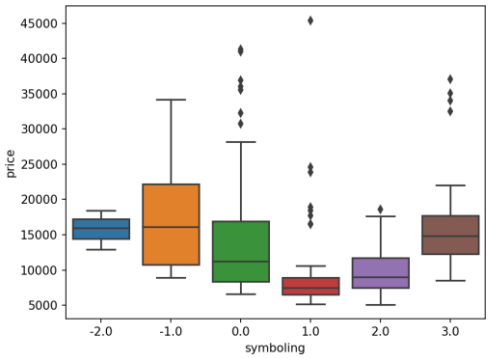
highway-mpg





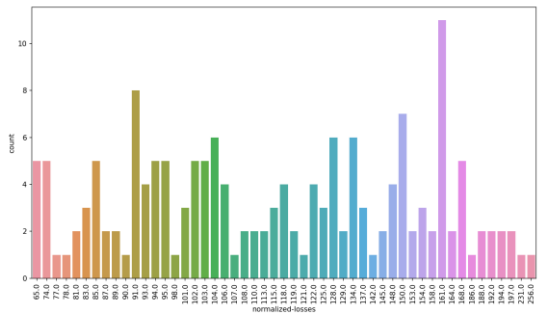
Symbolling

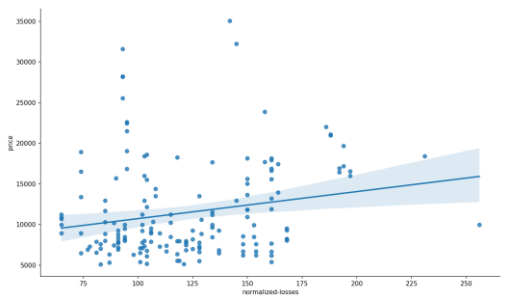




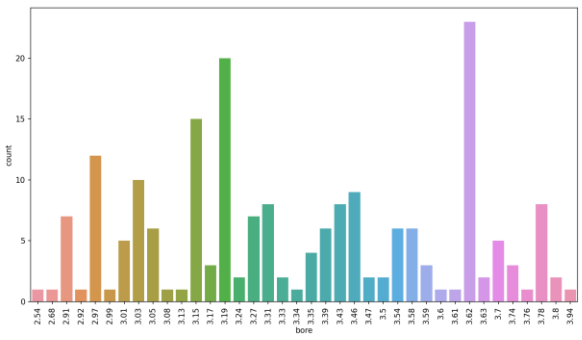
Feature columns with Missing Values- Dropped Missing Values

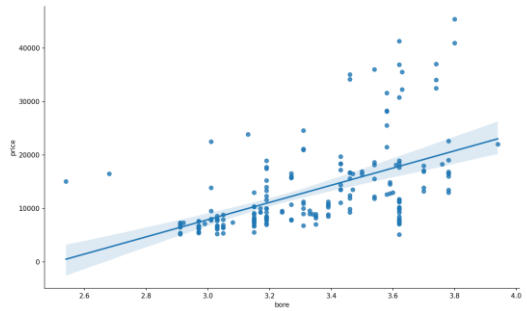
normalized-losses



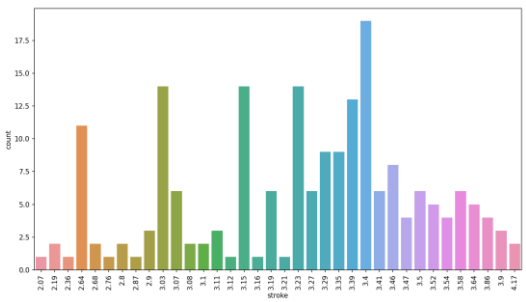


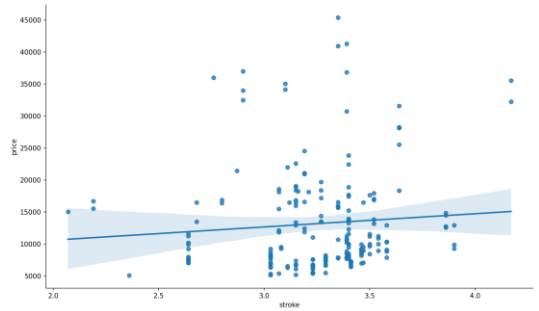
Bore



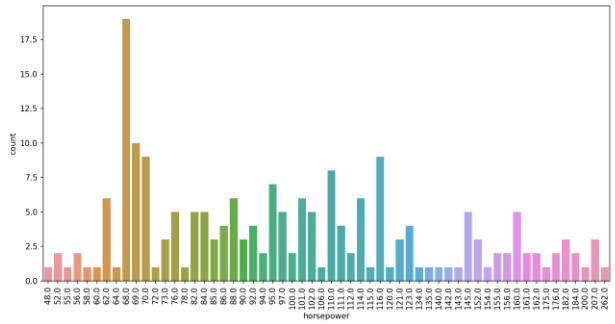


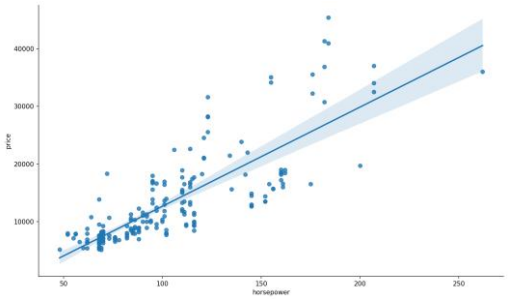
Stroke



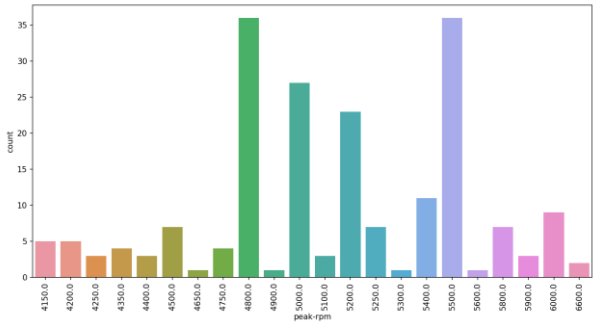


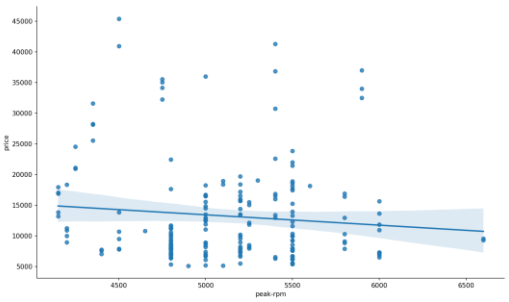
Horsepower



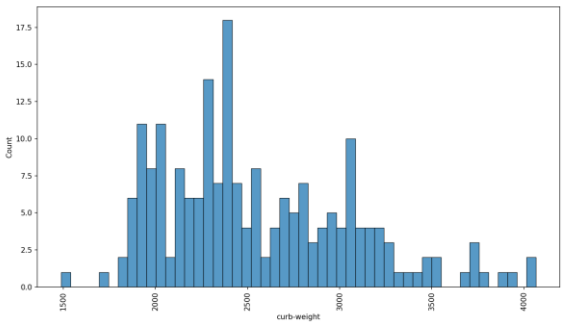


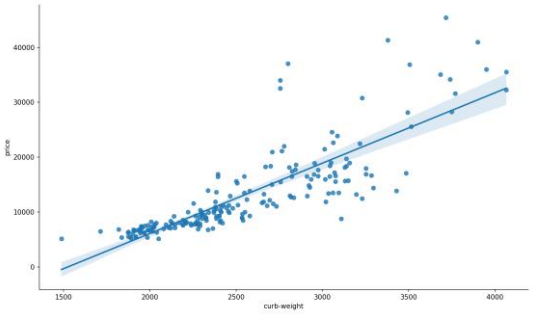
peak-rpm





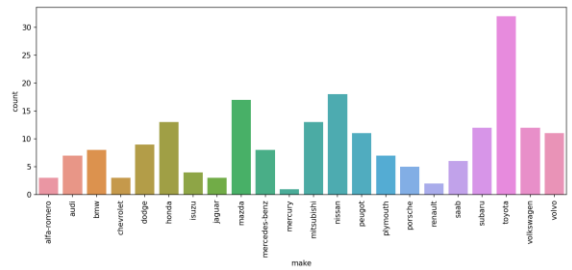
curb-weight

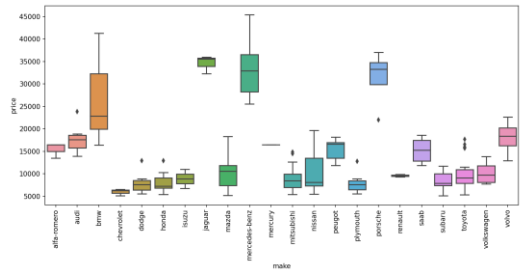




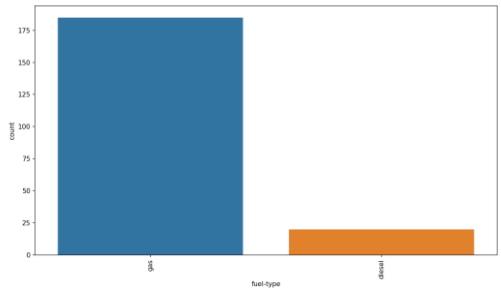
Categorical Columns

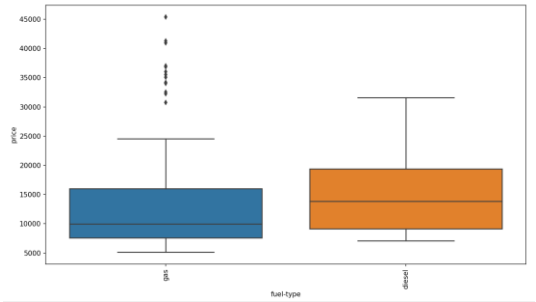
Make



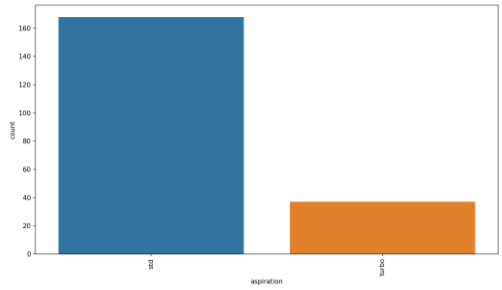


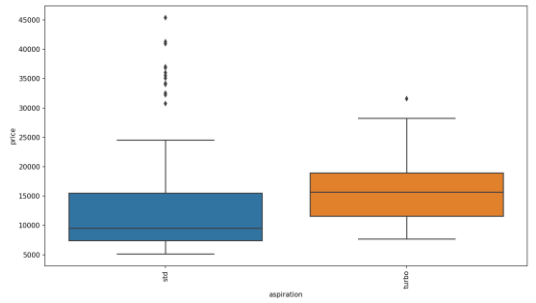
fuel type



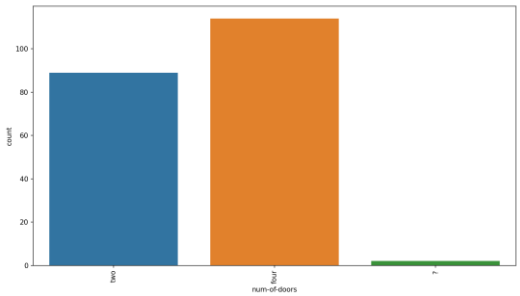


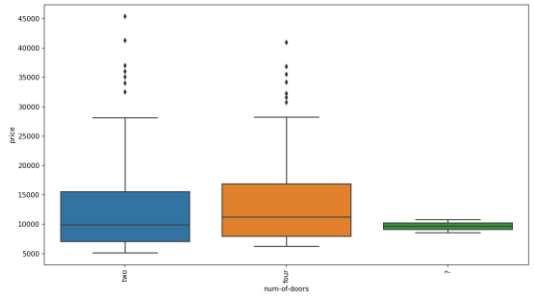
Aspiration



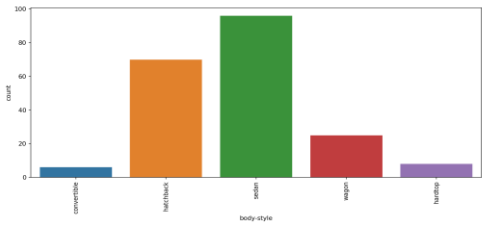


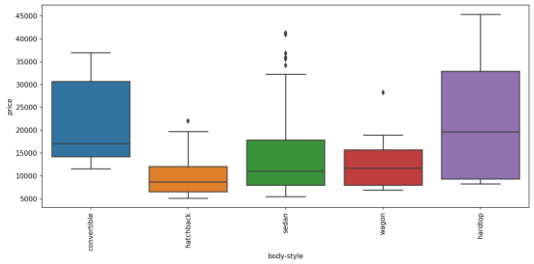
num-of-doors



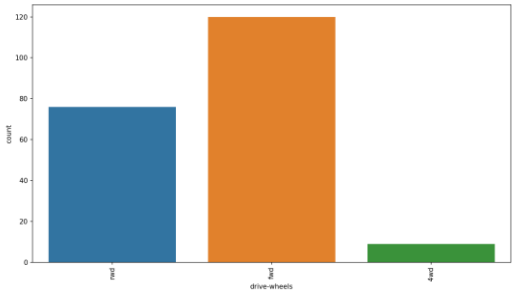


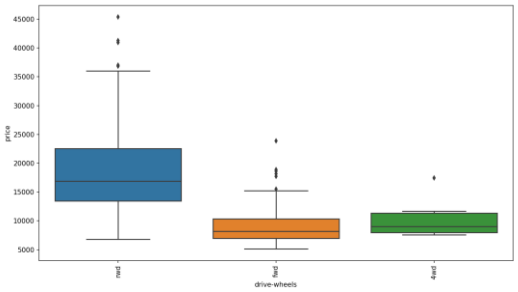
body-style



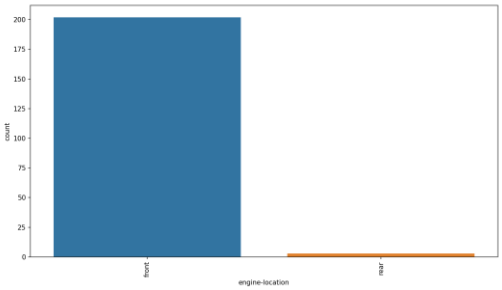


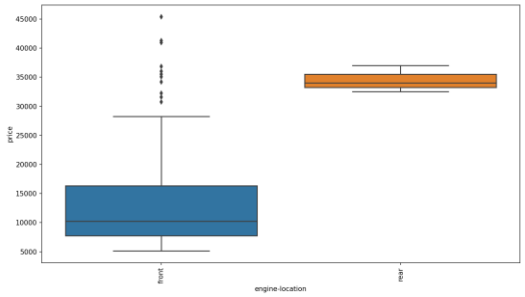
drive-wheels



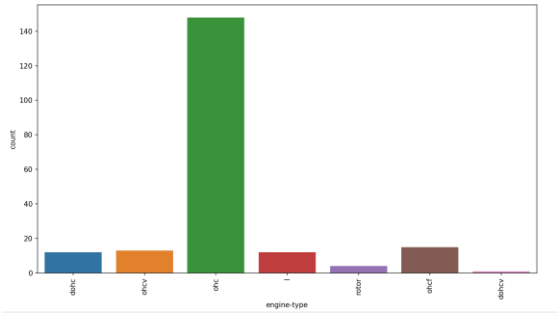


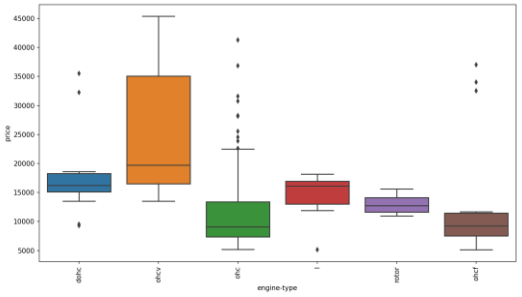
engine-location



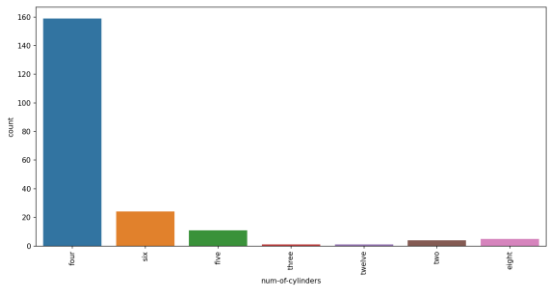


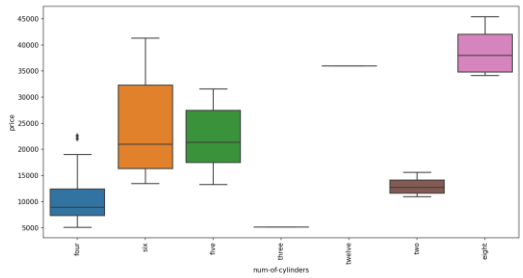
engine-type



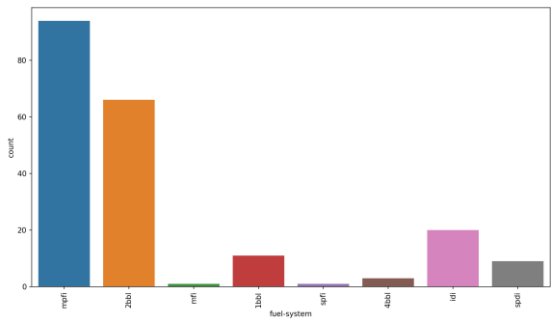


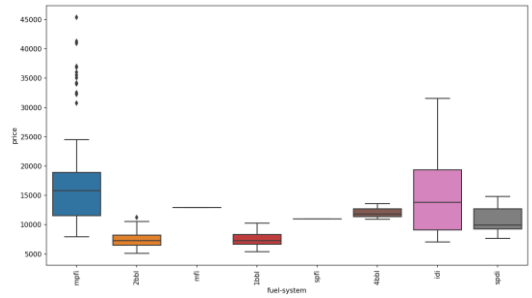
num-of-cylinders



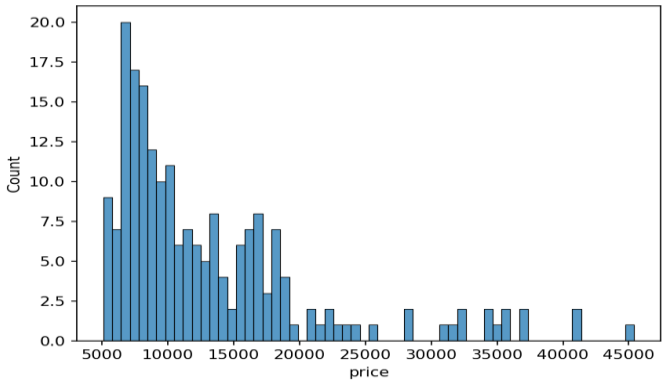


fuel-system





Target - price



By using countplot, we are able to quickly visualize the frequency of each category in a categorical feature, which is useful in identifying patterns and trends. Using lmplot and box plot visualization technique we are able to explore the relationship between features and the target.

By using lmplot, we could easily identify whether a feature is positively or negatively correlated with the target, and whether the relationship is linear or non-linear. The results showed that the price of a car was positively correlated with features such as engine size, horsepower, wheel-base, width, length and curb-weight, and negatively correlated with city-mpg and highway-mpg.

4. **Data Cleaning and Pre-processing:**

The initial step in this project was to clean and pre-process the data.

a. Handling Missing Values:

Target Column:

The Target Column 'price' of dataset contained missing values was removed before train test split.

Feature Columns:

The dataset features normalized-losses, aspiration, horsepower, peak-rpm, num-of-doors, bore, stroke of feature Matrix contained missing values.

The decision to drop the missing values was made as it was deemed irrelevant to impute them, and attempts to impute them using the most frequent value resulted in a reduced score for the model.

Table: Effect of Handling Missing data on Model Scores

Model	Drop Missing datapoint		Replacing with most frequent data	
Model	Train_Score	Test_Score	Train_Score	Test_Score
LinearRegression	0.92223	0.902829	0.918532	0.757724
SGDRegressor	0.918197	0.912002	0.905968	0.801276
KNeighborsRegressor	0.711252	0.825188	0.794963	0.801167
SVR	-0.14267	-0.174043	-0.130303	-0.0625002
DecisionTreeRegressor	0.999198	0.787793	0.999557	0.674152
RandomForestRegressor	0.975879	0.895005	0.990625	0.855181
GradientBoostingRegressor	0.994682	0.924518	0.995231	0.894185

b. Handling Categorical Columns:

Target Column 'price':

Both Test and Train Data to target was Type cast to Integer.

Feature Columns:

Numerical Features:

symboling, wheel-base, length, width, height, curb-weight, engine-size, compression-ratio, city-mpg, highway-mpg.

Categorical Features:

Type cast to numerical features: normalized-losses, bore, stroke, horsepower, peak-rpm

Columns dropped: body-style, engine-location, fuel-system

Few columns were dropped as the impact of encoding on the model's performance was negligible.

Encoded Features:

Label Encoded: 'make', 'num-of-cylinders'

One Hot Encoded**:** 'fuel-type', 'aspiration', 'num-of-doors', 'drive-wheels', 'engine-type'.

Categorical columns with few unique categories were One-Hot-Encoded. Features with considerable unique categories with low count were label Encoded to avoid error during un-stratified train test split. Labels in Label Encoding were assigned based on Value count order of data.

The decision to remove or retain a categorical feature column in a regression dataset was based on its relevance to the target variable and the impact of encoding on the model's performance.

c. Feature scaling:

Box plots are a useful tool for visualizing the distribution of a dataset and identifying outliers.

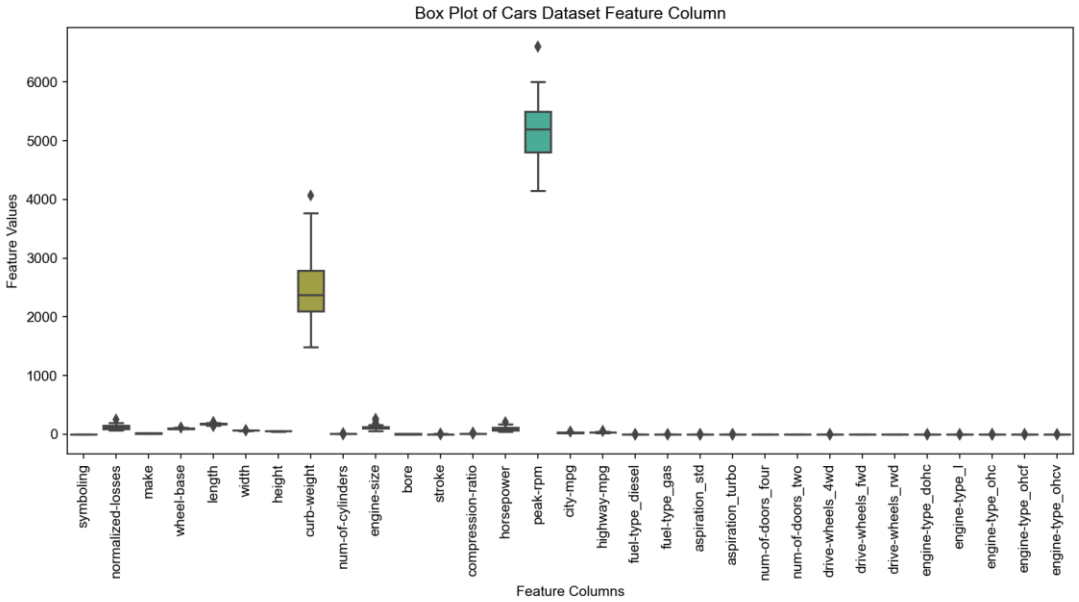


Fig. Box Plot of Feature Columns without Scaling

Observing the Box plot, the decision to scale the data to improve its performance in a machine learning model was made. Since the Box plot has shown that outliers are present in the dataset, Standard Scaler has been chosen for scaling.

Box Plot after Scaling:

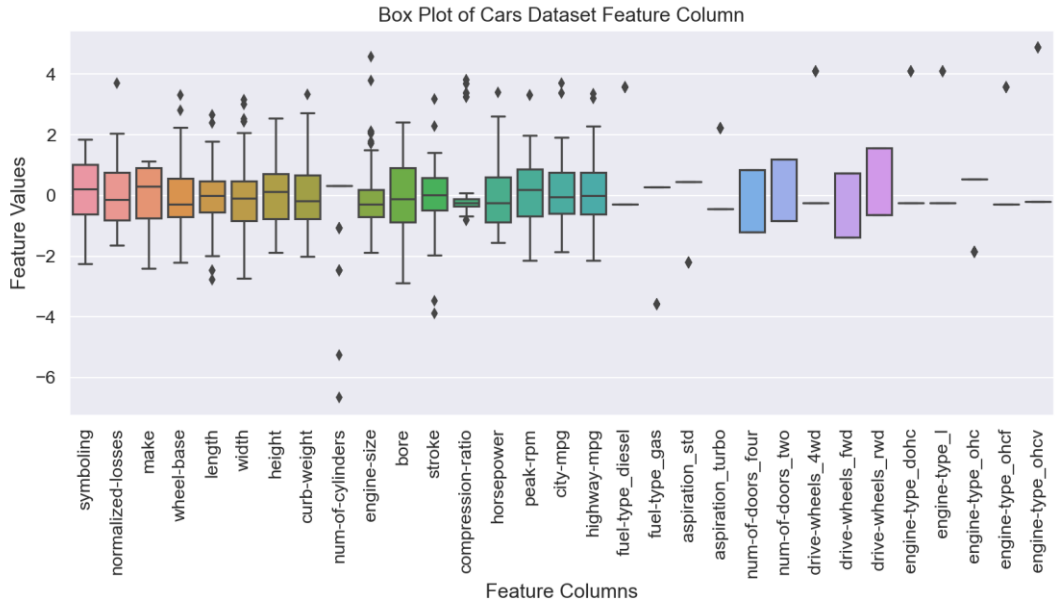


Fig. Box Plot of Feature Columns after Scaling

Comparison of Scores Before Scaling and After Scaling:

Table: Effect of Scaling on Model Scores

Model	Without Scaling		With Scaling	
Model	Train_Score	Test_Score	Train_Score	Test_Score
LinearRegression	0.92223	0.902829	0.92223	0.902829
SGDRegressor	-9.66593e+22	-1.08215e+23	0.914945	0.911043
KNeighborsRegressor	0.777484	0.763132	0.711252	0.825188
SVR	-0.14483	-0.176499	-0.14267	-0.174043
DecisionTreeRegressor	0.999198	0.831743	0.999198	0.761315
RandomForestRegressor	0.9826	0.908977	0.979859	0.916444
GradientBoostingRegressor	0.994682	0.924626	0.994682	0.927175

Overall, using Standard Scaler to scale a dataset with outliers has effectively improved machine learning model's performance in SGDRegressor and DecisonTreeRegressor. Linear Regressor and Gradient Boosting Regressor seem to have no impact. KNeighborsRegressor test score results greater than train score.

d. Handling Outliers:

The decision to retain or remove outliers were made by comparing the score with and without Outliers.
The data points with a Z-score greater than 3 or less than -3 were considered outliers and removed.

Table: Effect of Outliers on Model Scores

Model	With Outliers		Removing Outliers	
Model	Train_Score	Test_Score	Train_Score	Test_Score
LinearRegression	0.92223	0.902829	0.882981	-0.546655
SGDRegressor	0.914945	0.911043	0.871901	-2.47089
KNeighborsRegressor	0.711252	0.825188	0.723828	0.740428
SVR	-0.14267	-0.174043	-0.130563	-0.315231
DecisionTreeRegressor	0.999198	0.761315	0.997672	0.808632
RandomForestRegressor	0.979859	0.916444	0.964302	0.823366
GradientBoostingRegressor	0.994682	0.927175	0.994911	0.885576

Based on the Scores Comparison, retaining outliers were considered relevant for achieving good performance in LinearRegression Model, SGDRegressor Model, RandomForestRegressor, GradientBoostingRegressor.

5. Preliminary Model Selection for Tuning:

Models with good scores were decided to tune for obtaining best results.

- **i.** LinearRegression Model
- **ii.** SGDRegressor Model
- **iii.** RandomForestRegressor Model
- **iv.** GradientBoostingRegressor Model.

i. LinearRegression Model:

Final Score of LinearRegression Model:

Model	Train Score	Test Score
LinearRegression	0.92223	0.902829

ii. SGDRegressor Model:

In SGDRegressor Model, penalty is set to 'l1' i.e. Lasso Regression due to its inherent feature selection capability. Lasso Regression can drive some of the coefficients to exactly zero, effectively performing feature selection and removing irrelevant features from the model.

Final Score of SGDRegressor Model :

Model	Train Score	Test Score
SGDRegressor	0.918933	0.911646

iii. RandomForestRegressor Model:

To handle Overfitting, the model was tuned for max_depth and n_estimators.

To find the optimal values for these hyperparameters, a grid search was performed using the GridSearchCV function in scikit-learn. Optimal hyperparameters were found and implemented in the model to prevent overfitting.

```
RandomForestRegressor(max_depth=12, n_estimators=600)
```

Model	Train Score	Test Score
RandomForestRegressor	0.981444	0.910745

To further handle Overfitting, feature selection was done by manually tuning the hyperparameter max_features. The obtained optimal max_features value '8' was implemented in the model.

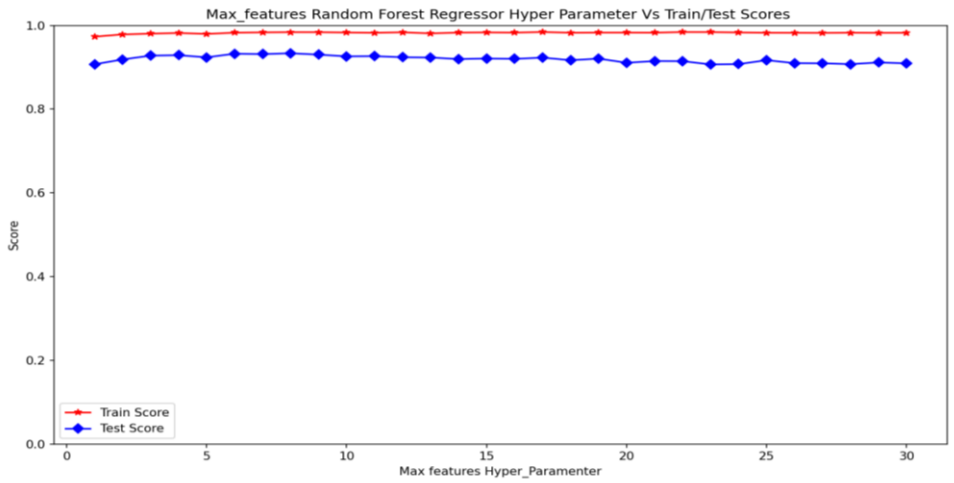


Fig. R2_Score Comparision for feature Selection

```
RandomForestRegressor(max_depth=12, n_estimators=600,
                      max_features=8)
```

Final Score of RandomForestRegressor:

Model	Train Score	Test Score
RandomForestRegressor	0.982472	0.932743

iv. GradientBoostingRegressor Model:

To handle Overfitting, the model was tuned for n_estimators, max_depth, and learning_rate.

To find the optimal values for these hyperparameters, a random search was performed using the RandomizedSearchCV function in scikit-learn. RandomizedSearchCV was opted over GridSearchCV for reducing the execution time as GradientBoosting is a series operation. Optimal hyperparameters were found and implemented in the model to prevent overfitting.

Best Parameter Values: {'n_estimators': 10, 'max_depth': 2, 'learning_rate': 0.3}

Model	Train Score	Test Score
GradientBoostingRegressor	0.957144	0.900358

To further handle Overfitting, feature selection was done by manually tuning the hyperparameter max_features. The obtained optimal max_features value ‘13’ was implemented in the model.

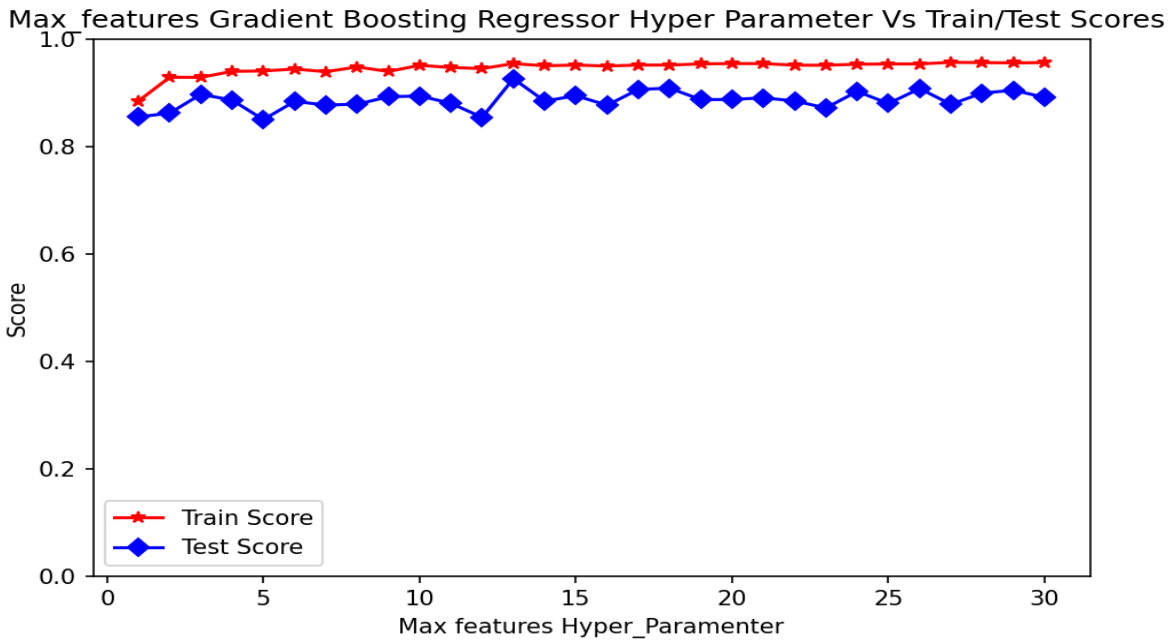


Fig. R2_Score Comparision for feature Selection

```
GradientBoostingRegressor(n_estimators=10, learning_rate=0.3,
                          max_depth=2, random_state=0, max_features=13)
```

Final Score of GradientBoostingRegressor Model:

Model	Train Score	Test Score
GradientBoostingRegressor	0.955284	0.927159

After tuning models using hyperparameter optimization techniques, the performance of each model on the training and testing data was compared. Among these models, the Gradient Boosting Regressor stood out as the best performer, with a train score of 0.955284 and a test score of 0.927159. Additionally, the difference between the train and test scores was relatively small, only 0.028. This indicates that the model is not overfitting to the training data and can generalize well to new, unseen data.

6. REGRESSION MODEL and RESULTS:

Model: Gradient Boosting Regressor

Test Data Set Results:

Evaluation Metric	Train Score	Test Score
R2_Score	0.955284	0.927158
MAE	958.268	1149.608
MSE	1566685.104	2318123.212
CV Score	0.860519	0.789144

Importance of different features:

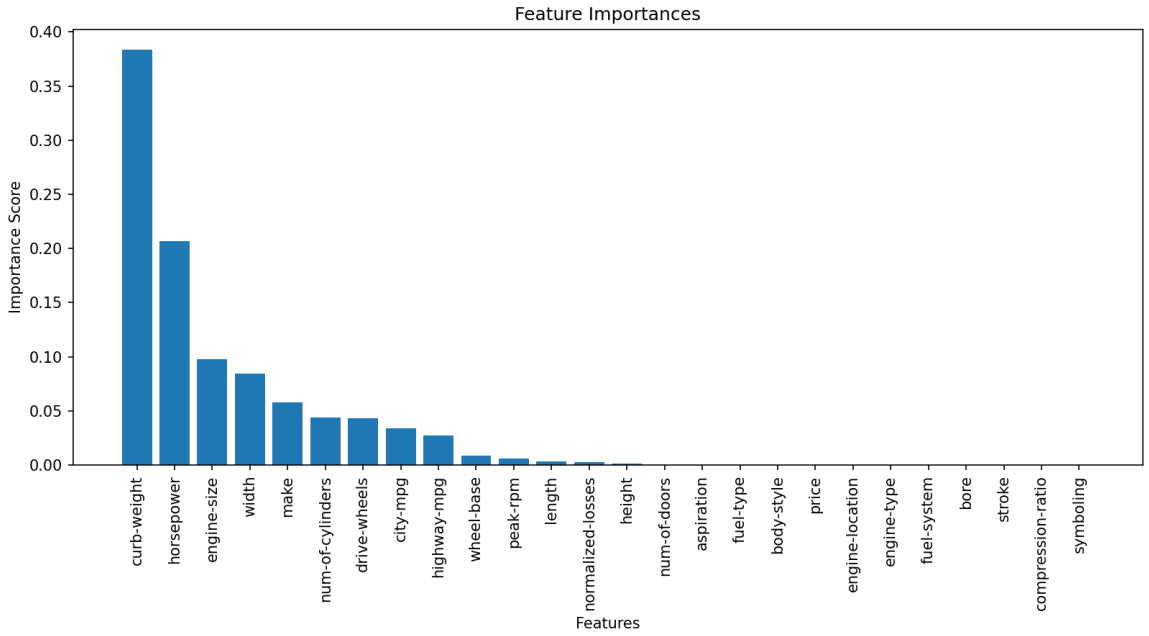


fig. Feature Importance plot

Based on these results, we can conclude that the Gradient Boosting Regressor model is the preferred model for this particular dataset and predictive task. It has demonstrated strong predictive performance on both the training and testing data, and the difference between the two scores is small, indicating good generalization ability.

CONCLUSION:

The regression analysis project aimed to predict car prices using the 'cars_price.csv' dataset. We performed data cleaning and pre-processing, exploratory data analysis, feature selection, and regression modelling. The results showed that the Gradient Boosting Regressor Model could accurately predict the price of a car based on selected features with a score of 0.7891. This project has practical applications in the automotive industry and can be useful for businesses and consumers looking to estimate the value of a car accurately.