

Multi-Class Classification of Cars

Problem Statement: The problem is to build a model that can accurately predict the class of a car based on its features. The dataset is 'cars_class.csv', which is a multi-class classification dataset with 719 samples and 18 numerical features. The target variable is the class of the car which may be one of: 0 - bus, 1 - Opel Manta, 2 - Saab, 3 - Van.

Introduction: The classification of cars based on their features is an important task in the automotive industry. It allows car manufacturers to understand the market demand for different types of cars and to optimize their production accordingly. This report presents a multi-class classification project that aims to predict the class of a car, which may be one of four classes: 0 – bus, 1 – Opel Manta, 2 – Saab, 3 – Van using a dataset titled 'cars_class.csv'.

Dataset Details: The 'cars_class.csv' dataset is a multi-class classification dataset with 719 samples and 18 numerical features. The target variable is the class of the car which may be one of: 0 –bus, 1 – Opel Manta, 2 – Saab, 3 – Van. The features include Compactness, Circularity, Distance Circularity, Radius ratio, scatter ratio, and other relevant features whose combination enables the model to accurately predict the class of a car, whether it is a bus, Opel Manta, Saab, or Van.

The objective of this project is to develop a multi-class classification model that can accurately predict the class of a car based on these features.

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

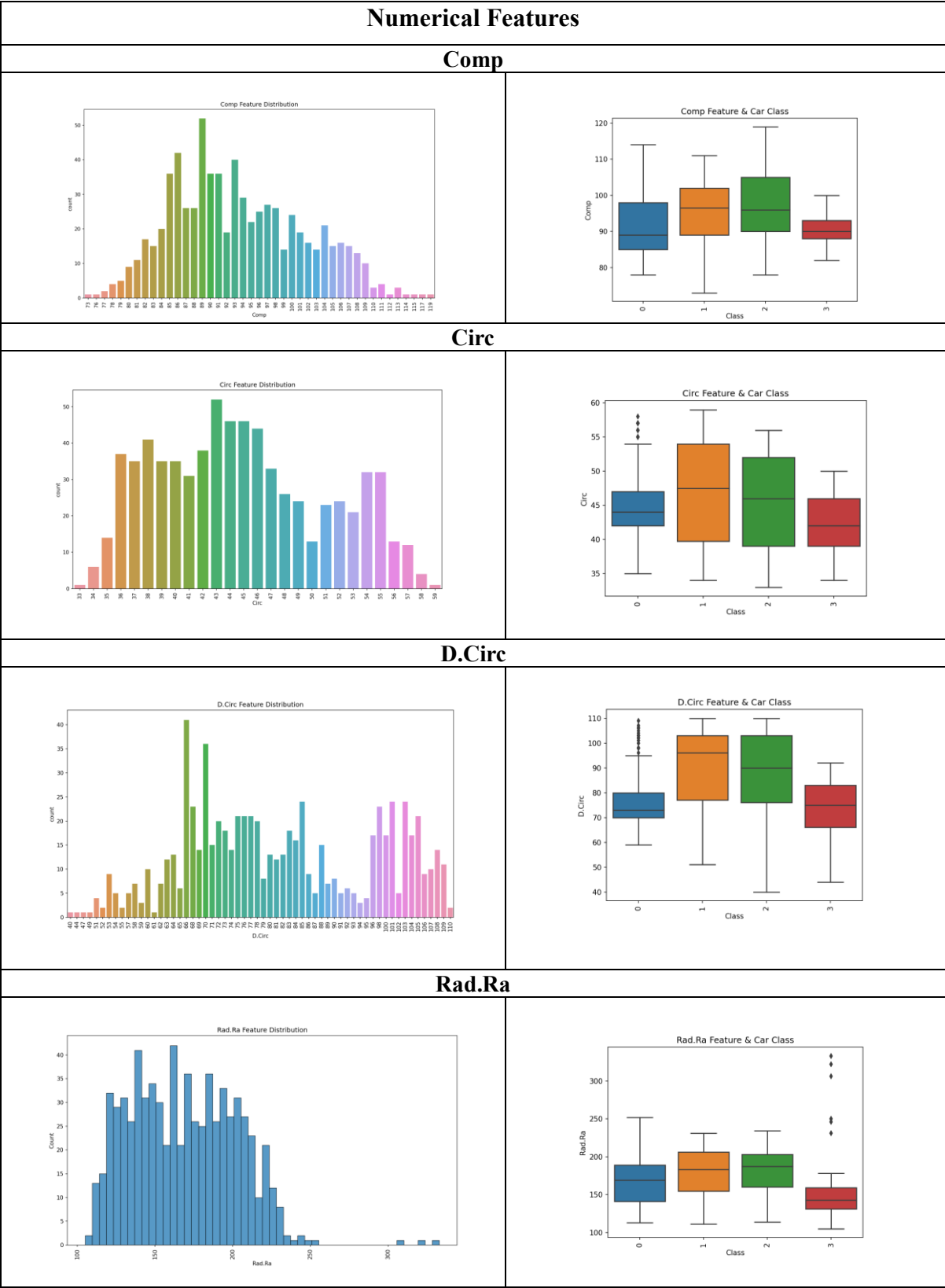
The project's success will be measured based on the accuracy of the model's predictions on a 20% holdout set of data. We will use metrics such as accuracy and F1-score to measure the performance of the model. We will also display the confusion matrix to understand how well the model is performing on each class.

Key Steps Involved:

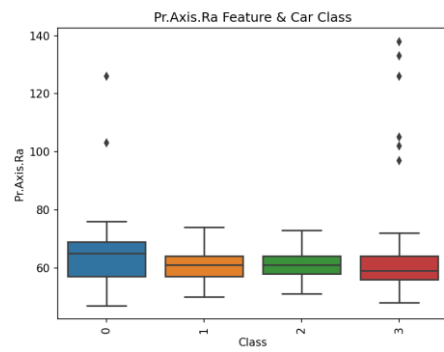
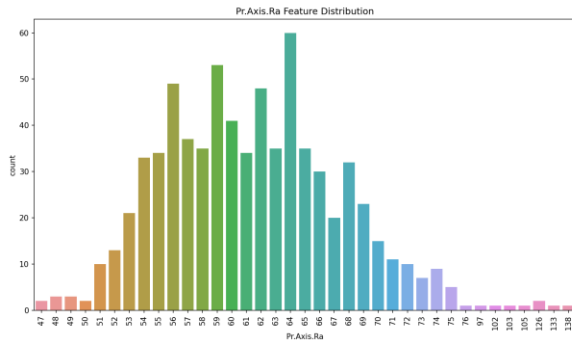
1. Loading the essential libraries and then loading and reading the dataset 'cars_class.csv'.
2. Splitting the dataset into train data and test data with 20% of the data as test data.
3. Exploratory Data Analysis

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, histplot, box plot and heatmap from seaborn library were used for visualizations in EDA to understand the distribution of data and the relationship between features and target.

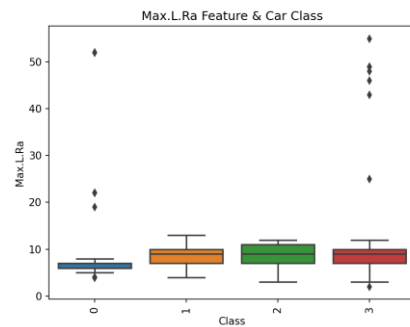
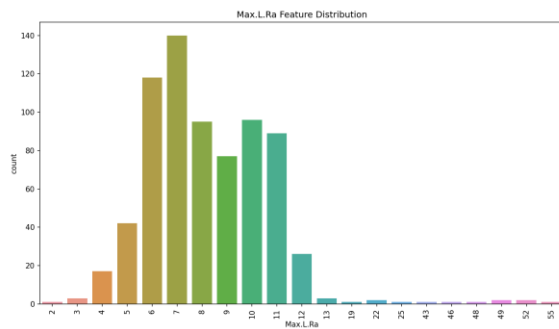
Below table represents Data visualization in EDA. Left column plots help visualize and understand the distribution of data in respective features, and right column plots help understand the relationship between features and target.



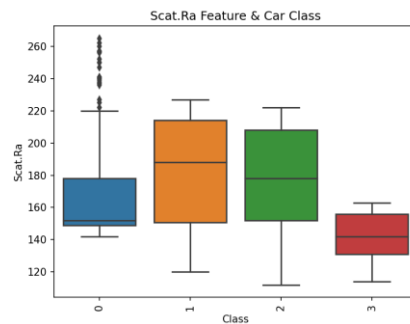
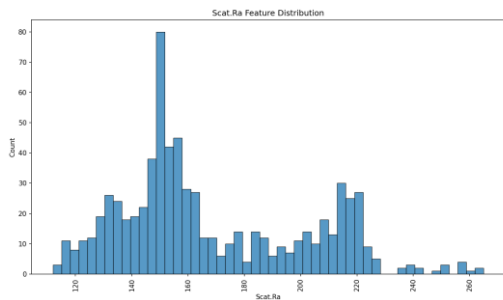
Pr.Axis.Ra



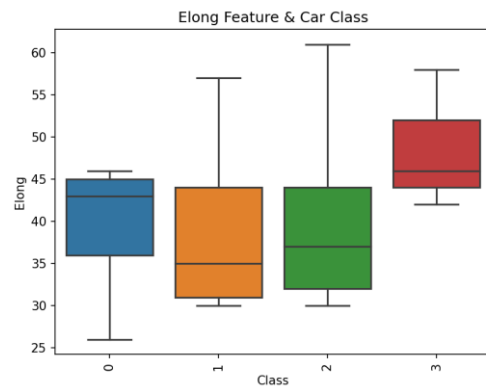
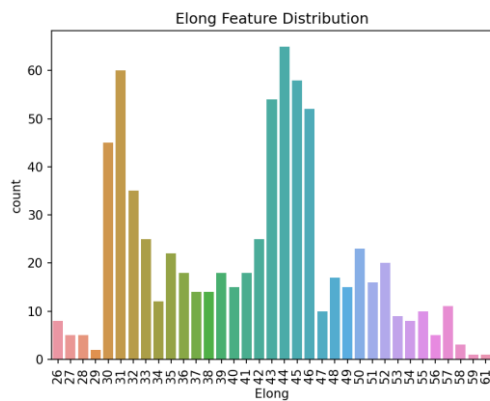
Max.L.Ra



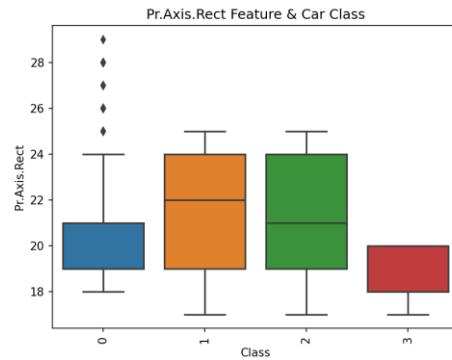
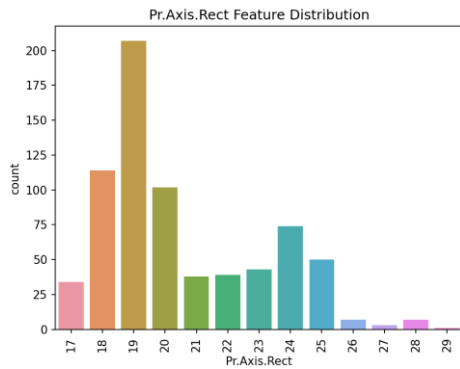
Scat.Ra



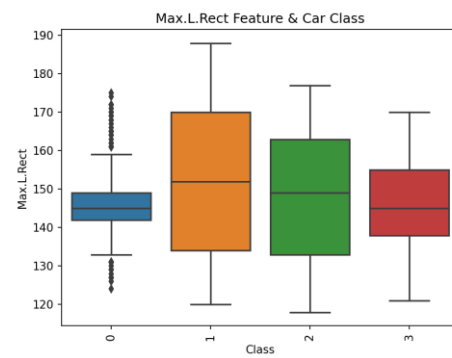
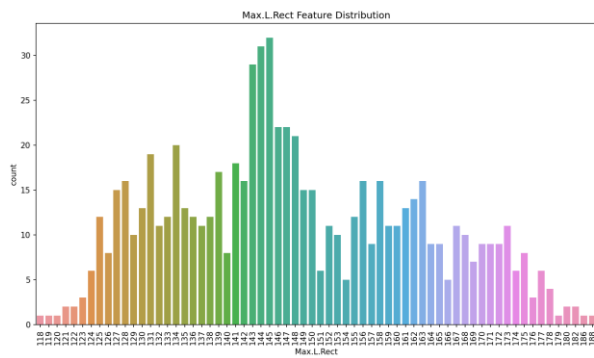
Elong



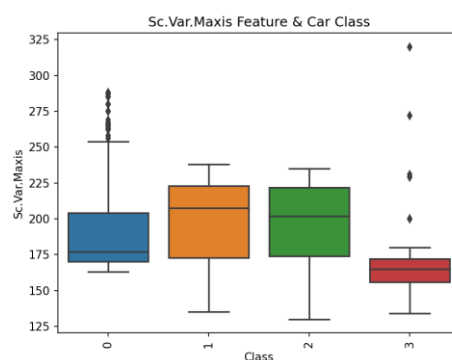
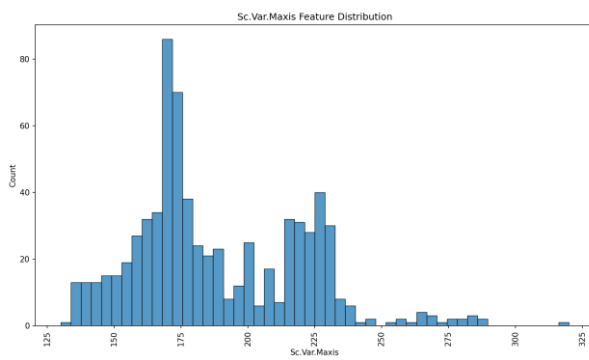
Pr.Axis.Rect



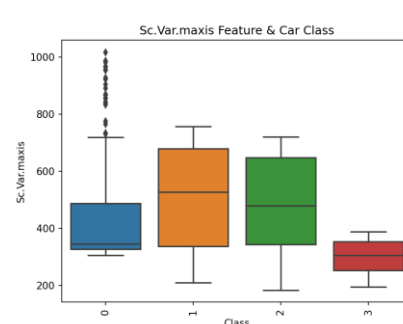
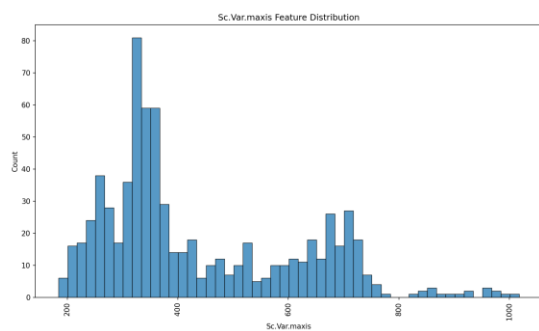
Max.L.Rect



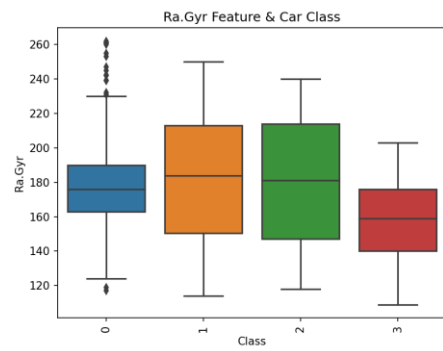
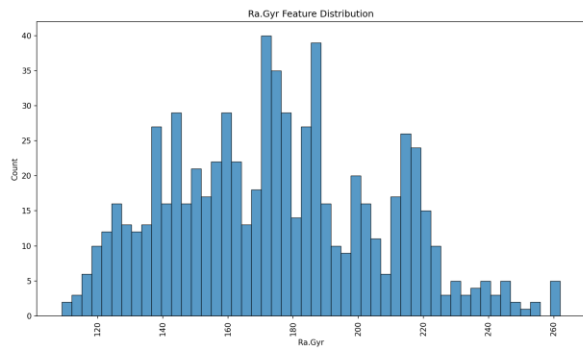
Sc.Var.Maxis



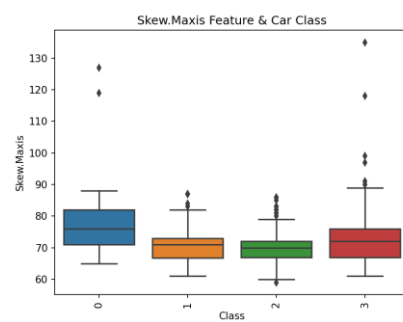
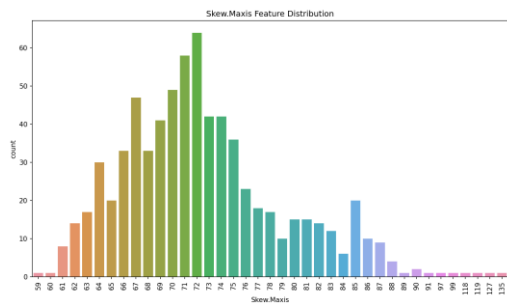
Sc.Var.maxis



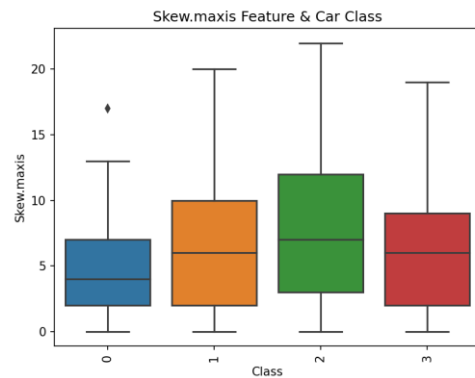
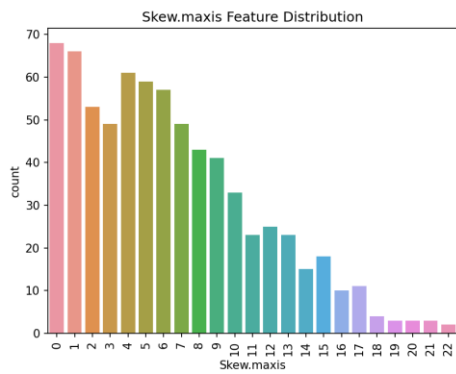
Ra.Gyr



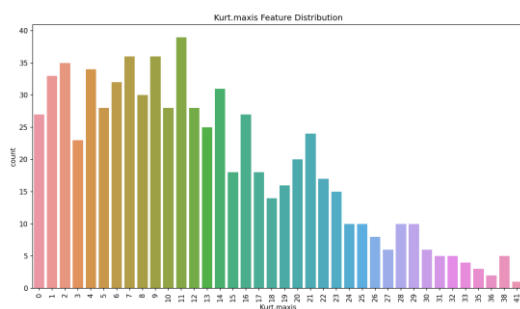
Skew.Maxis

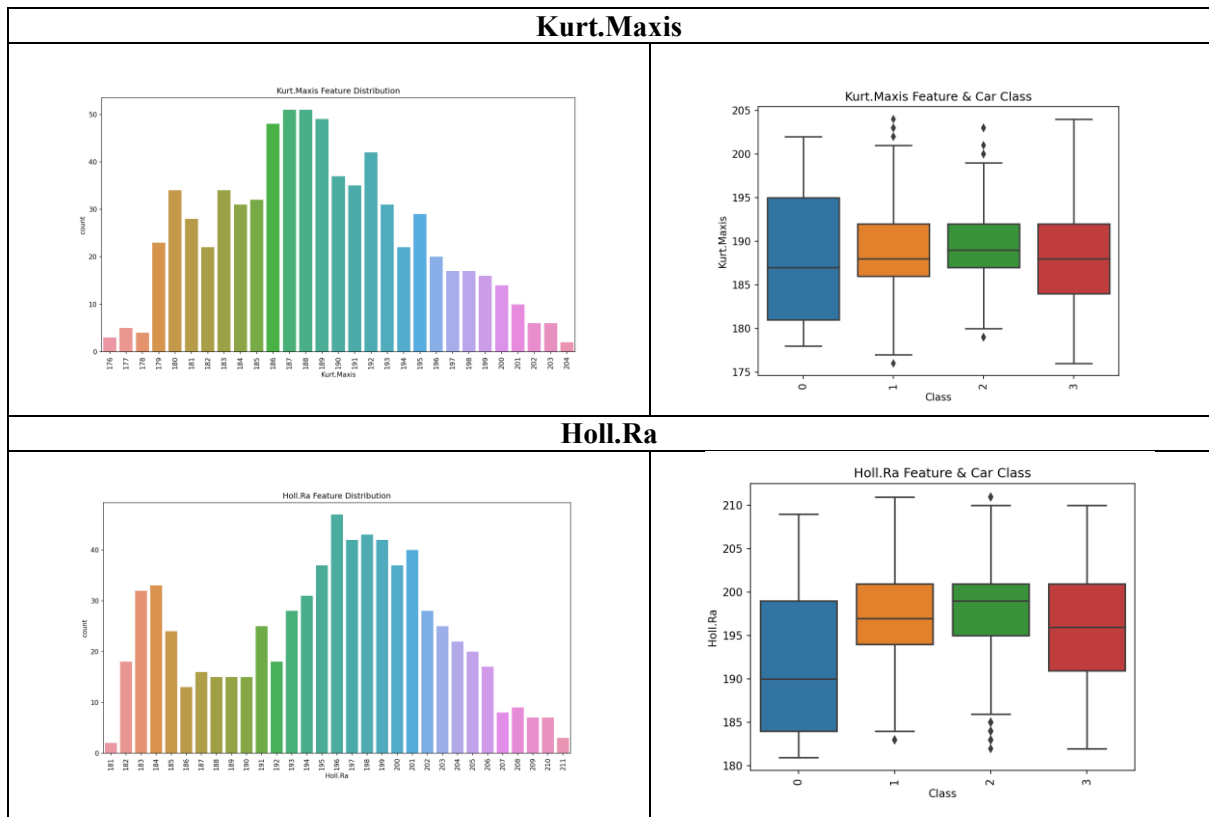


Skew.maxis



Kurt.maxis





By using countplot and histplot we are able to quickly visualize the frequency of each variable in a feature, which is useful in understanding the distribution of data. Using box plot visualization technique we are able to explore the relationship between features and the target.

Heatmap

Seaborn heatmap helps visualize the correlation between each feature. Using this it is easier to identify any highly correlated features that may need to be removed to avoid overfitting.

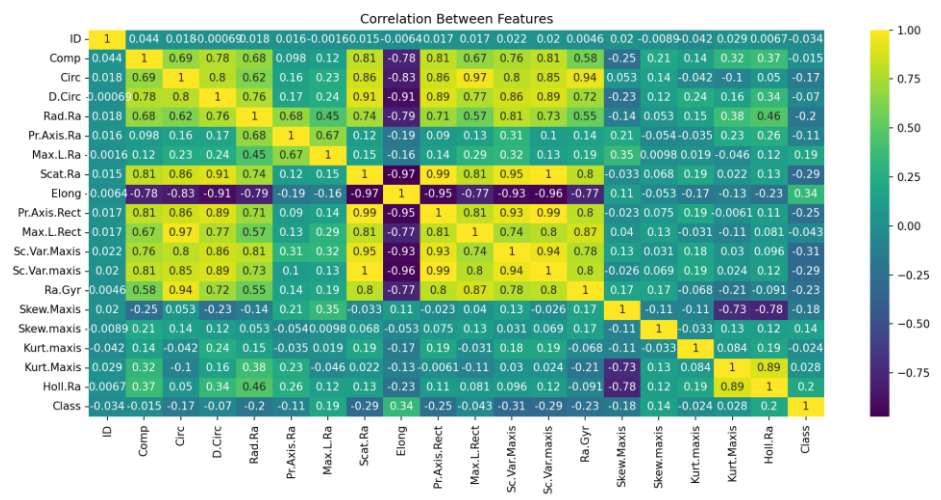
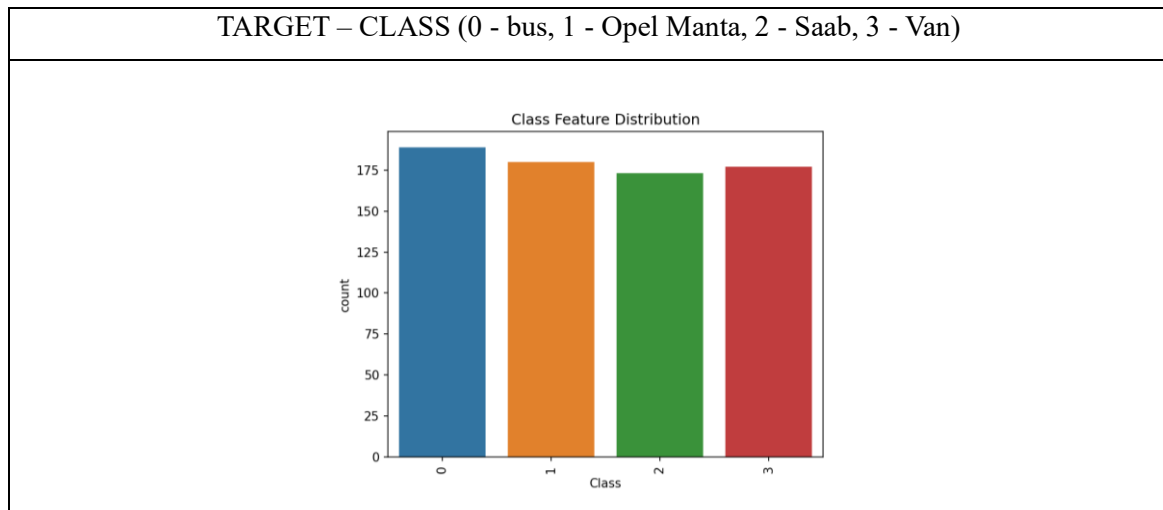


Fig. Heatmap to visualize Correlation

We can observe that Pr.Axis.Rect feature is highly correlated with Sc.Var.maxis and Scat.Ra features with a correlation of 0.99. Sc.Var.maxis and Scat.Ra features are correlated with a correlation of 1. Implies that any one feature among these three are enough for modelling and remaining two features can be eliminated.



Seaborn countplot of target column 'Class' shows that the class distribution in the dataset is relatively balanced, with each class having roughly the same number of samples.

4. Data Cleaning and Pre-processing:

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

a. Handling Missing Values:

Both Target Column and Feature Columns does not contain any missing values.

b. Handling Categorical Features:

All features are Numerical and require no encoding.

c. Feature scaling:

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

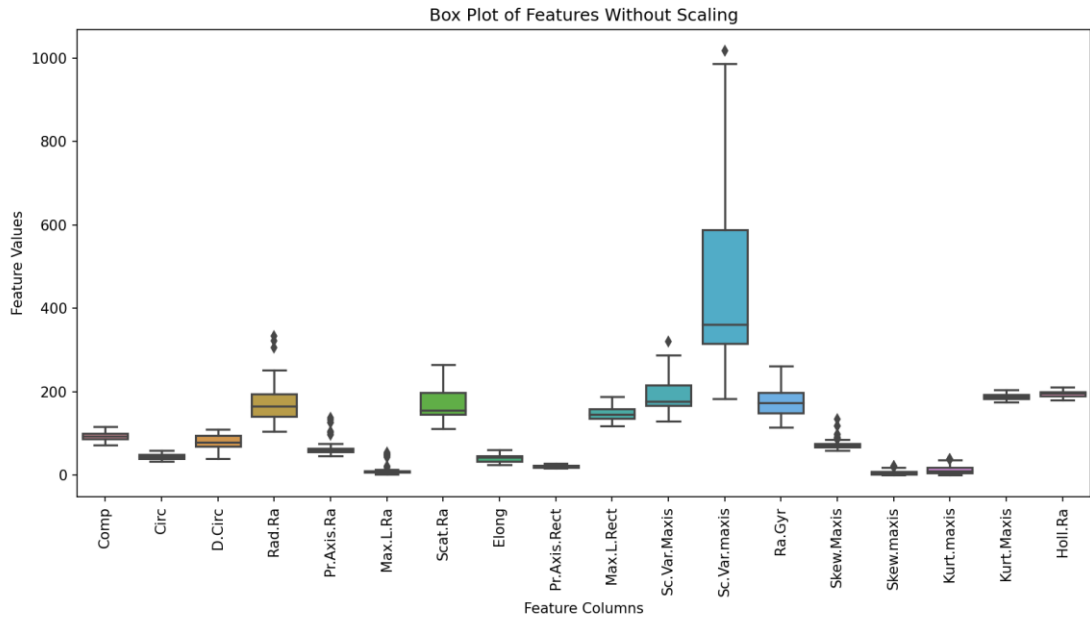


Fig. Box Plot of Feature Columns without Scaling

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:

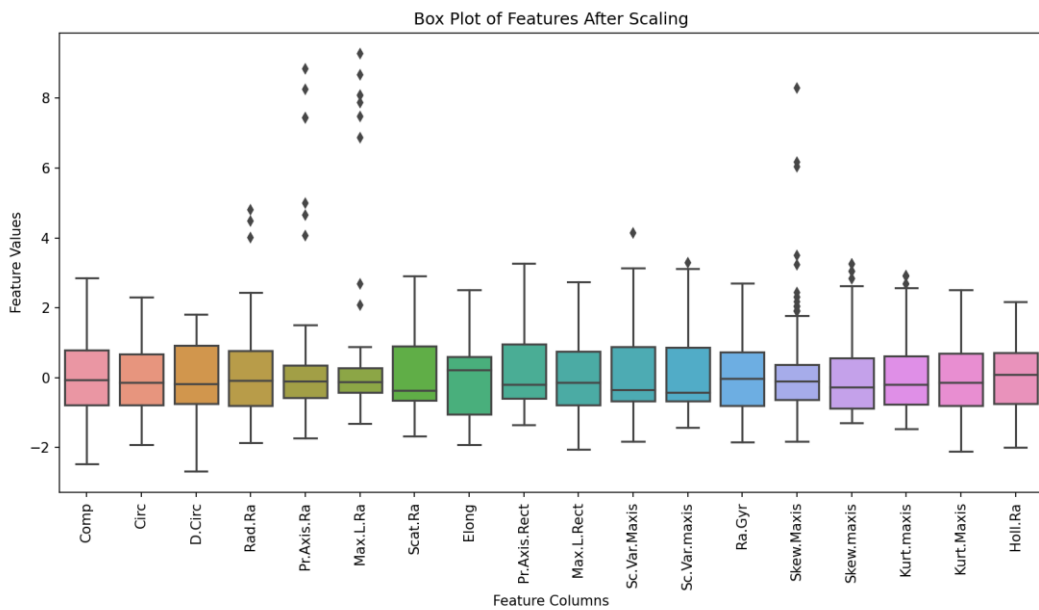


Fig. Box Plot of Feature Columns after Scaling

Comparison of Scores Before Scaling and After Scaling:

Table: Effect of Scaling on Model Scores

Model	Without Scaling		With Scaling	
	Train_Score	Test_Score	Train_Score	Test_Score
LogisticRegression	0.74087	0.722222	0.81913	0.777778
KNeighborsClassifier	0.787826	0.701389	0.822609	0.715278
SVC	0.493913	0.5	0.841739	0.75
DecisionTreeClassifier	0.74087	0.680556	0.74087	0.680556
RandomForestClassifier	0.775652	0.729167	0.784348	0.75
GradientBoostingClassifier	1	0.770833	1	0.756944

Overall, using Standard Scaler to scale a dataset with outliers has effectively improved machine learning model's performance in LogisticRegressor, KNeighborsClassifier, SVC and RandomForestClassifier.

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	
	Train_Score	Test_Score	Train_Score	Test_Score
LogisticRegression	0.81913	0.777778	0.818996	0.777778
KNeighborsClassifier	0.822609	0.715278	0.818996	0.722222
SVC	0.841739	0.75	0.849462	0.763889
DecisionTreeClassifier	0.74087	0.680556	0.729391	0.680556
RandomForestClassifier	0.784348	0.75	0.784946	0.722222
GradientBoostingClassifier	1	0.756944	1	0.777778

Based on the Scores Comparison, retaining outliers were considered relevant as no significant improvement in model scores were observed except SVC and GradientBoostingClassifier model.

5. Preliminary Model Selection for Tuning:

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

- i. LogisticRegression Model
- ii. SVC Model
- iii. RandomForestClassifier Model
- iv. GradientBoostingClassifier Model.

i. LogisticRegression Model:

Logistic Regression Model was manually tuned using Recursive Feature Elimination (RFE) to determine number of features required to avoid overfitting and improve performance. It was concluded through plot that 16 features give best results.



Fig. Scores variation with respect to number of features

Logistic Regression Model Score Comparison:

Model	Train Score	Test Score
LogisticRegression - all features	0.81913	0.77777
LogisticRegression - 16 features	0.80174	0.791667

Further tuning Regularization strength 'C' showed no improvement in Score.

Final Score of LogisticRegression Model:

Model	Train Score	Test Score
Logistic Regression	0.80174	0.791667

ii. SVC Model:

In SVC Model, feature selection was performed using PCA for SVC kernel 'rbf' and 'linear'. Further tuning was tried out using 'C' but no improvement was observed in both 'rbf' and 'linear' models. SVC model was also tuned for C without performing feature selection. Below table shows the scores obtained in each tuned SVC model.

SVC Model Score Comparison:

Model	Train Score	Test Score
SVC - all features, kernel = 'rbf'	0.841739	0.75
SVC - all features, kernel = 'linear'	0.843478	0.777778
SVC - 15 features, kernel = 'linear'	0.831304	0.78472
SVC (C=3, kernel=linear)	0.838260	0.8125

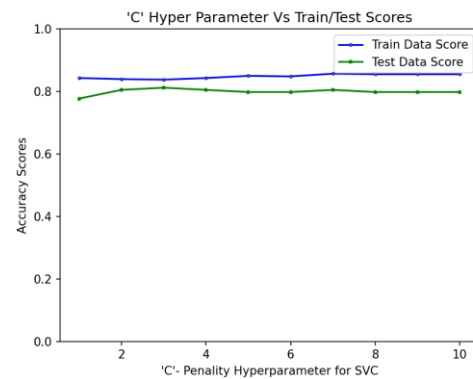


Fig. Tuning for feature selection using PCA in SVC model with kernel 'linear'.

Fig. Tuning for 'C' for SVC kernel 'linear'.

Final Score of SVC Model:

Model	Train Score	Test Score
SVC (C=3, kernel=linear)	0.838260	0.8125

iii. RandomForestClassifier Model:

To handle Overfitting and improve performance, the model was tuned for `n_estimators`, `max_depth` and `max_features`.

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 and improve performance.

Hyperparameter tuned model Score:

Model	Train Score	Test Score
RandomForestClassifier	0.994782	0.763888

To further handle Overfitting, the obtained model was tuned on `min_impurity_decrease`, `min_samples_split` and `ccp_alpha` but no significant improvement was observed.

RandomForestClassifier (n_estimators=100, max_depth=8, max_features=8,
min_impurity_decrease=0.012, random_state=0)

Final Score of RandomForestClassifier:

Model	Train Score	Test Score
RandomForestClassifier	0.801739	0.756944

iv. GradientBoostingClassifier Model:

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

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.

Hyperparameter tuned model Score:

Model	Train Score	Test Score
GradientBoostingClassifier	1.0	0.75694

To further handle overfitting, the obtained model was tuned for ccp_alpha manually. The obtained ccp_alpha slightly improved the performance of model and reduced Overfitting.

GradientBoostingClassifier(learning_rate=0.6, max_depth=6, max_features=12,
n_estimators=500, ccp_alpha=0.004, random_state=0)

Final Score of GradientBoostingClassifier:

Model	Train Score	Test Score
GradientBoostingClassifier	0.803478	0.736111

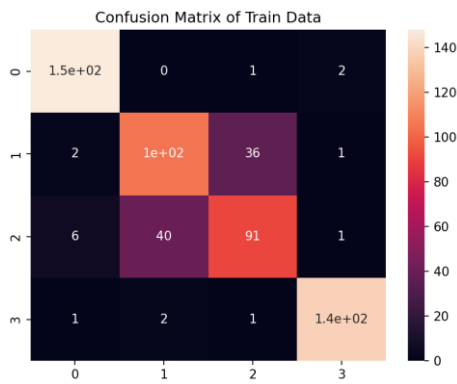

After tuning models using hyperparameter optimization techniques, the performance of each model on the training and testing data was compared. Among these models, the SVC model stood out as the best performer, with a train score of 0.8383 and a test score of 0.8125.

Additionally, the difference between the train and test scores was relatively small, only 0.0258. This indicates that the model is not overfitting to the training data and can generalize well to new, unseen data.

6. CLASSIFICATION MODEL and RESULTS:

Model: Space Vector Classification Model

Test Data Set Results:

Evaluation Metric	Train Score	Test Score
Accuracy	0.83826	0.8125
F1_Score	0.834041	0.807426
Confusion Matrix	 <p>Confusion Matrix of Train Data</p>	 <p>Confusion Matrix of Test Data</p>
CV Score	0.794676	0.762857

Importance of different features:

In the one-vs-one approach, the model is trained on every pair of classes. Since we have 4 classes, the model is trained on 6 pairs of classes: (0, 1), (0, 2), (0, 3), (1, 2), (1, 3), and (2, 3). Therefore 6 rows of coefficients, each row corresponding to one of the binary classifiers learned by the one-vs-one approach is given by 'coef_' method. The final multi-class prediction is made by choosing the class with the highest score.

The feature importance is represented by calculating the sum of the absolute values of the coefficients for each feature across all binary classifiers.

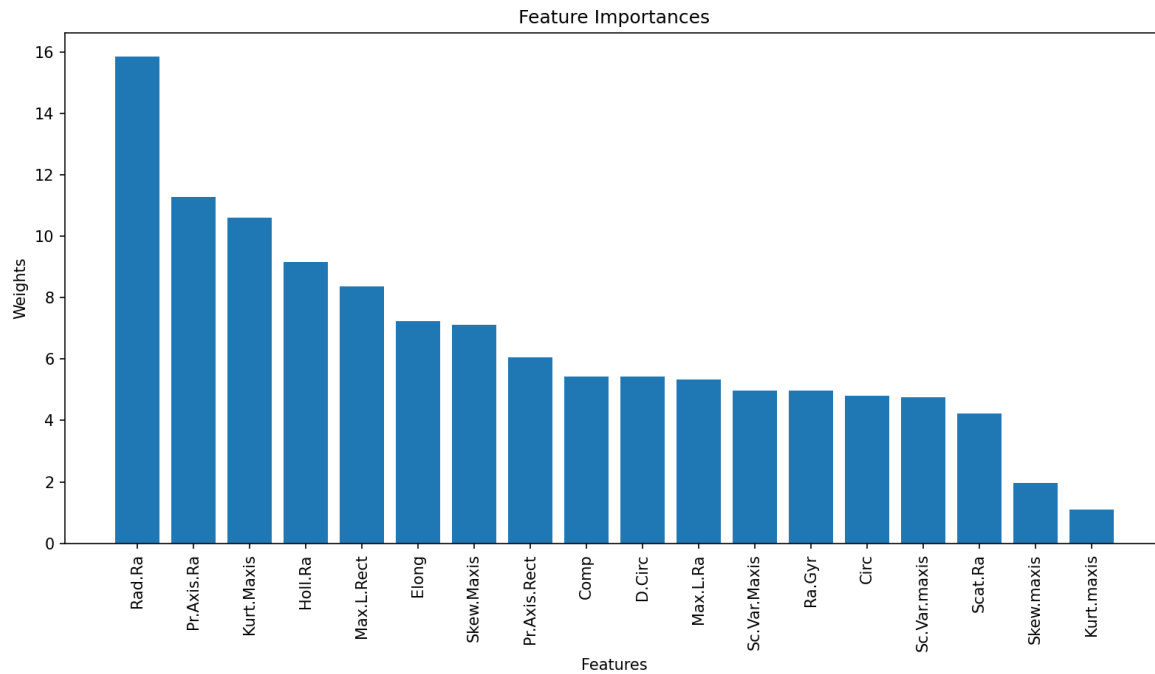


fig. Feature Importance plot

Based on these results, we can conclude that the Support Vector Classification 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 classification prediction project aimed to predict cars class using the 'cars_class.csv' dataset. We performed data pre-processing, exploratory data analysis, feature selection, and Classification modelling. The results showed that the Support Vector Classification Model could accurately predict the class of a car based on selected features with a score of 0.7628. This project has practical applications in the automotive industry and allows car manufacturers to understand the market demand for different types of cars and to optimize their production accordingly.