INDEX

1. Abstract
2. Introduction
3. project steps:

- load the data and understand the values.
- exploratory data analysis
- train test split
- preprocessing
- model fitting and comparison
- model evaluation
- hyperparameter tuning
- feature importance
- Model selection

4.conclusion

Abstract

This report presents a multi-class classification approach to predict the class of cars based on various features. The goal is to develop a model that accurately identifies the class of a car given its attributes. The report covers the preprocessing steps, train-test split, model comparison, evaluation using five different algorithms, and the selection of the best-performing mode based on accuracy, F1 score, and confusion matrix.

Introduction

In the world of automotive industry, classifying cars into different categories based on their characteristics is a crucial task. It aids in market segmentation, targeted advertising, and understanding consumer preferences. With the advancements in machine learning and data analysis, we can now leverage these techniques to develop accurate car classification models.

The 'cars_class.csv' dataset is a collection of data on cars and their corresponding classes. This dataset presents an opportunity to explore and analyze the characteristics of different cars and develop a machine learning model for classifying them accurately.

The objective of this project is to build a robust multi-class classification model that can predict the class of a car based on its attributes. By leveraging machine learning techniques, we aim to develop a system that can automate the categorization process and provide valuable insights for the automotive industry.

Throughout this project, we will preprocess the dataset, explore the features, and train various classification models. Our ultimate goal is to create a reliable model that can effectively classify cars into their appropriate classes, contributing to enhanced market segmentation and decision-making in the automotive sector

Step 1: Load the Data and Understand the Values

To start the data analysis, the first step is to load the data into a Pandas DataFrame and gain an understanding of the dataset and its values

Step 2: Exploratory Data Analysis (EDA)

After loading the data into a Pandas data frame, the next step is to perform Exploratory Data Analysis (EDA) to gain insights and a better understanding of the data. EDA involves examining the shape of the data, obtaining basic information, statistical details, value counts of each feature, correlation matrix, and assessing the target variable ("class"). Additionally, it is essential to check the balance of each class, identify duplicate values, and determine if there are any null values present in the dataset.

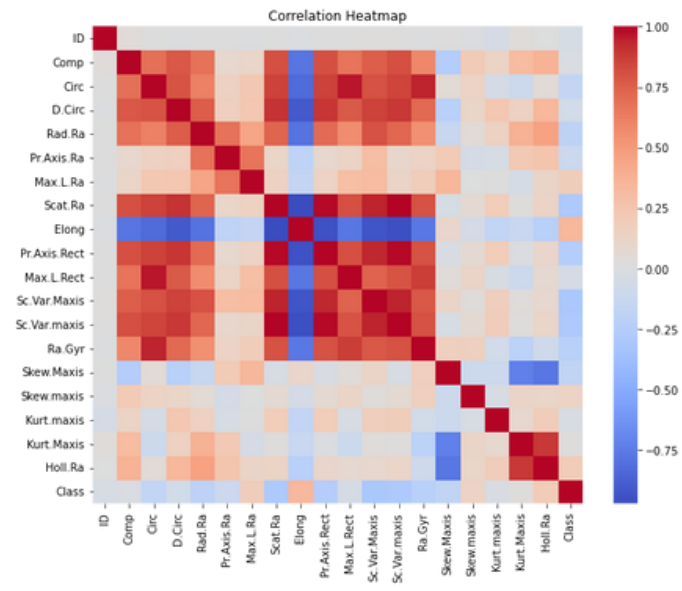
1. Shape of the Data:
   Get the shape of the data frame using df.shape. This will provide the number of rows and columns in the dataset. the shape of the data was 719, 20

2. Basic Information:
   Use df.info() to obtain basic information about the data, including the data types of each feature, memory usage, and the presence of any null values.

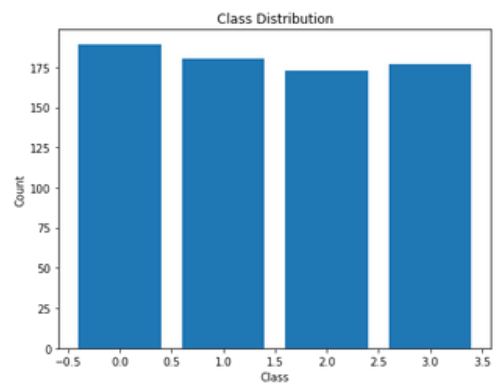
3. Statistical Details:
   Employ df.describe() to generate statistical details such as count, mean, standard deviation, minimum, quartiles, and maximum values for each numerical feature.

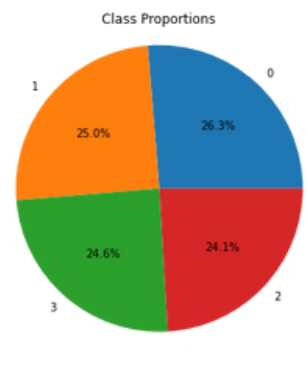
4. Value Counts of Each Feature:
   Utilize df[column].value_counts() to obtain the count of unique values in each feature. This will help identify the frequency of different categories within categorical variables.

5. Correlation Matrix:
   Calculate the correlation matrix using df.corr() to determine the relationships between numerical features. A correlation matrix provides insights into the strength and direction of linear relationships, assisting in feature selection and identifying potential multicollinearity.



6. Target Variable Analysis:
   Analyze the target variable, which in this case is "class." Use df['class'].value_counts() to obtain the count of each class in the dataset. Assess if the classes are balanced or imbalanced. The data is not imabalanced.





7. Duplicate Values:
   Check for duplicate values using df.duplicated().sum() .data does not have any duplicate values

8. Null Values:
   Evaluate the presence of null values in the dataset using df.isnull().sum(). the data did not contain any null values

By performing these EDA steps, I have gained valuable insights into the structure, quality, and characteristics of the dataset. These insights guided me for further preprocessing steps and model building, leading to more accurate predictions in my car class prediction project.

Step: 3. Train-Test Split

The dataset was split into two subsets: a training set and a testing set. The training set, which comprises 80% of the data, was used to train the models, while the remaining 20% was used for model evaluation.

Step: 4. Preprocessing Steps

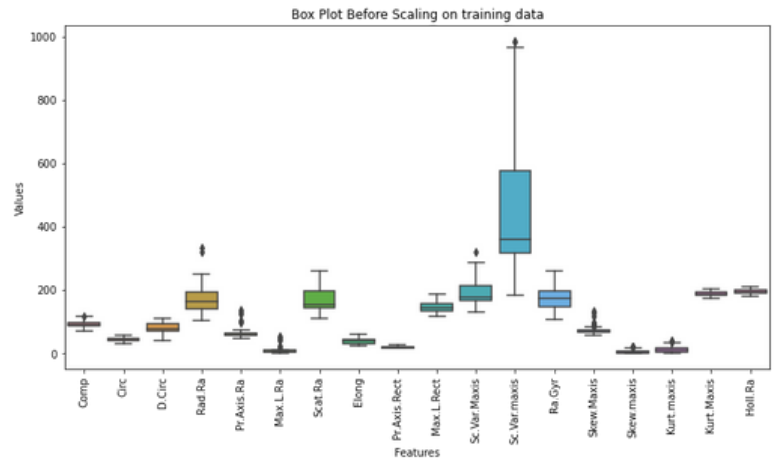
Before training the models, several preprocessing steps were performed to ensure the data is in a suitable format for classification:

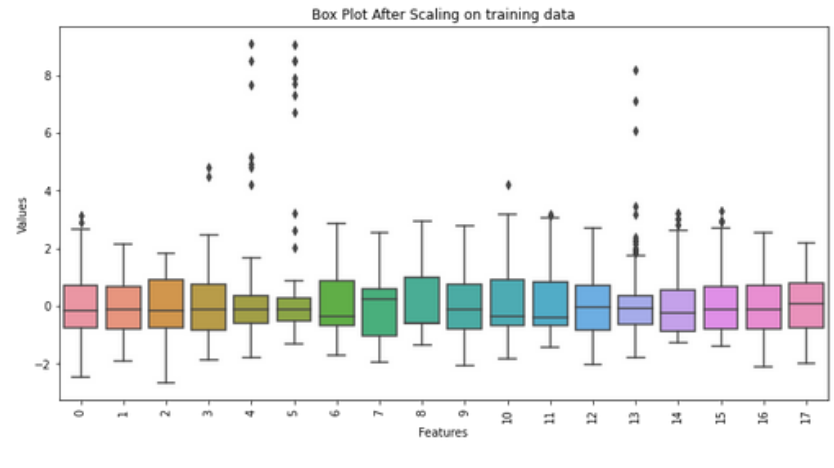
a. Data Cleaning: since there were no null values or duplicate values, no need to remove any data except one column which was unnecessary for the model i.e "ID". Hence, I removed the column.

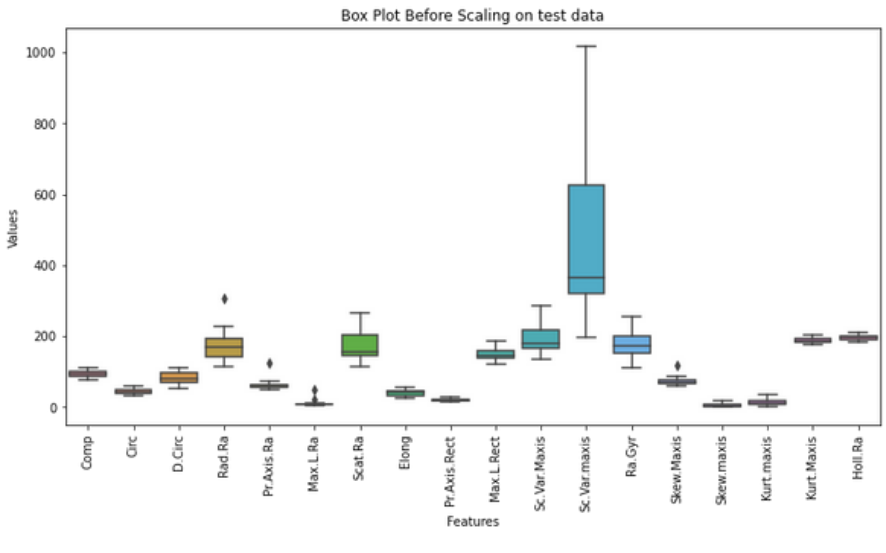
b. Encoding : since, there were no categorical values , there was no need to encode the data

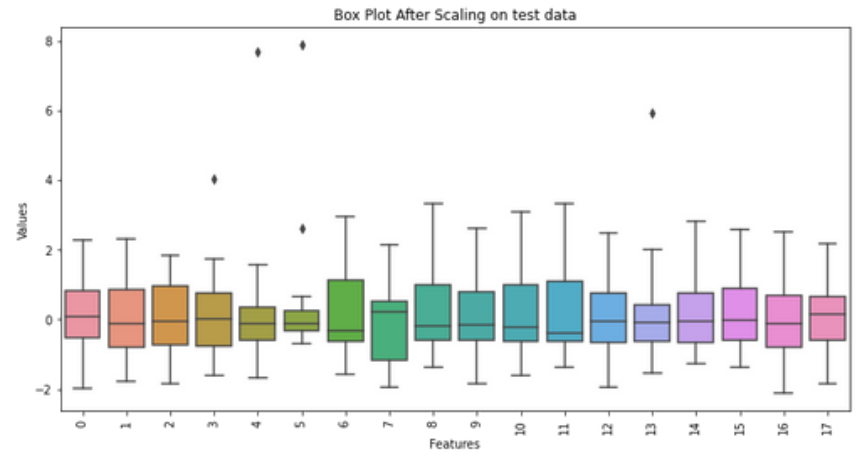
c. Feature Scaling : The box plots showed huge variations in the ranges and scales of different features, it indicates the need for scaling. hence, I applied Standard Scaler

d. Feature Extraction: for feature extaction I applied Principal component analysis (PCA). PCA is an unsupervised linear transformation technique which is primarily used for feature extraction and dimensionality reduction.









Step: 5. Model Comparison

To find the best model for car class prediction, five different classification algorithms were evaluated:

a. Logistic Regression: A linear model that applies logistic function to predict the probability of each class.

b. Decision Tree: A tree-based model that recursively splits the dataset based on different features to create decision rules.

c. Random Forest: An ensemble model that combines multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting.

d. K-Nearest Neighbors (KNN) classifier : predicts the class of a new instance based on the class labels of its nearest neighbors in the training data.

e. Gradient Boosting: An ensemble model that sequentially trains weak learners, such as decision trees, to correct the errors of previous models.

Step: 6. Model Evaluation

The models were evaluated using three metrics: accuracy, F1 score, and confusion matrix.

a. Accuracy: The ratio of correctly predicted instances to the total number of instances, indicating the overall correctness of the model.

b. F1 Score: A measure of the model's accuracy that considers both precision and recall. It is the harmonic mean of precision and recall.

c. Confusion Matrix: A table that displays the number of true positives, true negatives, false positives, and false negatives, allowing for a detailed analysis of model performance.

Logistic Regression Classifier:
Accuracy: 0.7777
F1-score: 0.7705

K Neighbors Classifier:
Accuracy: 0.7430
F1-score: 0.7331

Gradient Boosting Classifier:
Accuracy: 0.7708
F1-score: 0.7666

Decision Tree Classifier:
Accuracy: 0.6666
F1-score: 0.6681

Random Forest Classifier:
Accuracy: 0.8194
F1-score: 0.8168

based on the evaluation metrics I selected 4 top models for further tuning. the models are:

- Logistics regression
- gradient Boosting Classifier
- K Neighbors Classifier
- Random Forest Classifier

Step: 7. HyperParameter Tuning

Before and after tuning Comaprison:

Logistic Regression Classifier:
Accuracy: 0.7777
F1-score: 0.7705

Logistic Regression Classifier:
Accuracy: 0.7708
F1-score: 0.7664

Gradient Boosting Classifier:
Accuracy: 0.7708
F1-score: 0.7666

Gradient Boosting Classifier:
Accuracy: 0.75694
F1-score: 0.7537

K Neighbors Classifier :
Accuracy: 0.7430
F1-score: 0.7331

K Neighbors Classifier:
Accuracy: 0.7847
F1-score: 0.7732

Random Forest Classifier:
Accuracy: 0.8194
F1-score: 0.8168

Random Forest Classifier:
Accuracy: 0.8055
F1-score: 0.7987

Step: 8. Model Selection

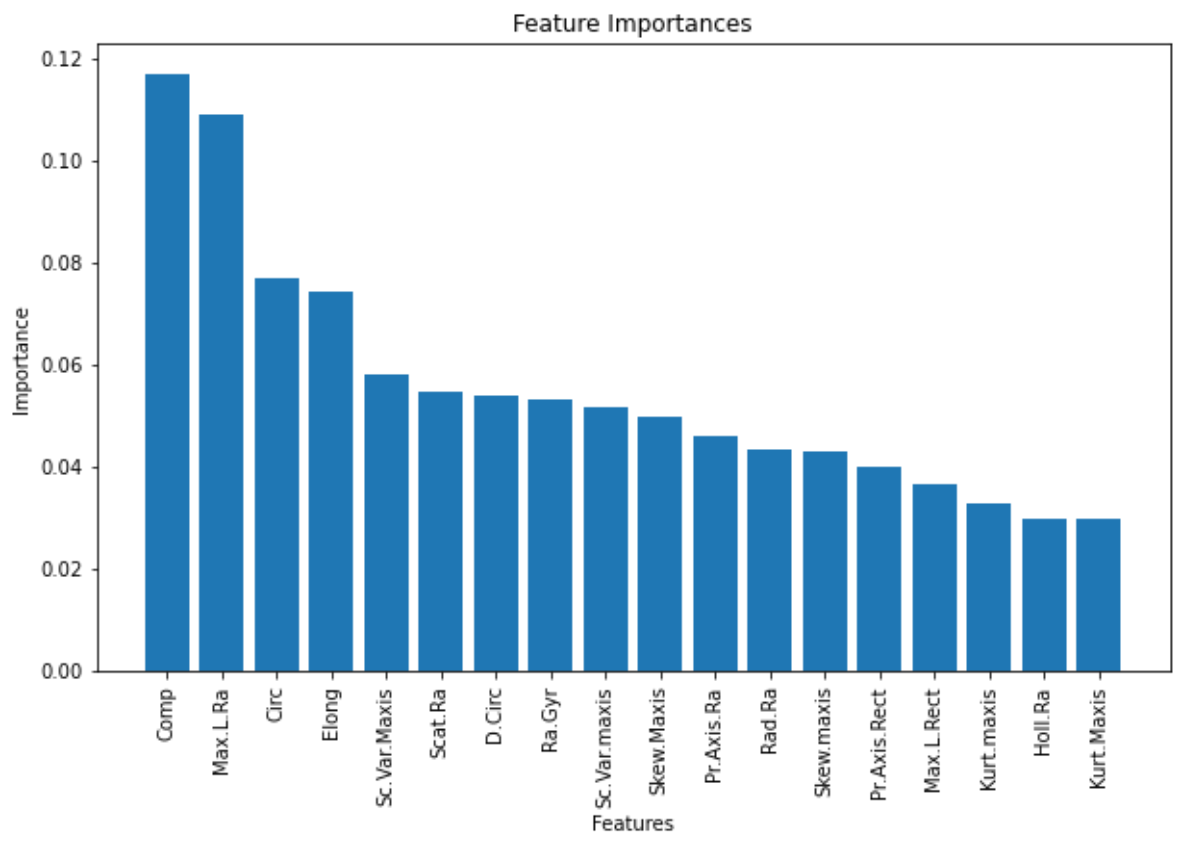
After comaprison and evaluation I selected Random forest Classifier as the Final model of my project with accuracy : 81%
f1-score : 0.8143

Step: 9. Feature Importances

Feature importance refers to a technique used to determine the relevance or importance of each feature (or variable) in a dataset in relation to the target variable. It helps in understanding which features have the most significant impact on the target variable and can be used to identify the key drivers of a particular outcome.

Random Forest is a machine learning algorithm that is often used to calculate feature importances.It works by constructing multiple decision trees and averaging their predictions to make accurate predictions. Random Forest calculates feature importances based on the average impurity reduction each feature provides across all the decision trees in the forest.

rf.feature_importances_



Conclusion

In conclusion, this project focused on the classification of car classes based on various features. The objective was to build a machine learning model that could accurately predict the class of a car given its features.

Started exploring the dataset and understanding the distribution of car classes and the characteristics of the features. Data preprocessing techniques were applied, including handling missing values, encoding categorical variables, and scaling numerical features.

Several classification algorithms, including Logistic Regression, Random Forest, and K-Nearest Neighbors, were trained and evaluated using performance metrics such as accuracy, confusion matrix and F1-score. The models were tuned and optimized to achieve the best possible performance.

After comparing the results, it was determined that the Random Forest algorithm outperformed the other models in terms of accuracy and F1-score. It exhibited the highest predictive accuracy in classifying car classes based on the given features.

The project also involved feature importance analysis using techniques such as Random Forest's feature importances. This analysis provided insights into the significance of each feature in predicting car classes.