GEARS OF JUDGEMENT: MACHINE LEARNING FOR CAR CONDITION CLASSIFICATION

Presented by Rohan Seetepalli

Overview

- Introduction
- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Analysis and Model fitting
- Real time Prediction
- Summary
- Future Scope

Introduction

Assessing the condition of a car is a crucial step in the automobile industry, especially for buyers, sellers, and dealerships aiming to make informed decisions. Traditional methods rely on manual inspections, which can be subjective and inconsistent.

This project addresses that challenge by developing a machine learning-based classification system that predicts a car's condition using features such as buying price, maintenance cost, seating capacity, boot space, and number of doors. The model categorizes vehicles into unacceptable, acceptable, good, or very good, and is further extended to perform real-time binary classification, simplifying results into acceptable and unacceptable classes for practical use.

Data Pre-processing

Dataset

This dataset has been obtained from the UCI Repository. It consists of 7 variables and 1726 records. The variables are described as follows:-

- ***Buying Price (Categorical):*** **Indicates the initial cost of the car (e.g., low, medium, high, very high).**
- ***Maintenance Cost (Categorical)*****: Represents the ongoing service expense category for the car.**
- ***Number of Doors (Categorical)*****: Specifies how many doors the car has (e.g., 2, 3, 4, 5 or more).**
- ***Seating Capacity (Categorical)*****: Denotes how many people the car can accommodate (e.g., 2, 4, 5).**
- ***Luggage Boot Size (Categorical)*****: Reflects the storage capacity of the car's trunk (e.g., small, medium, big).**
- ***Car Condition (Target Variable) (Categorical)*****: The class label indicating the overall condition — unacceptable, acceptable, good, or very good.**

```python
df= pd.read_csv("car.csv")
df
```

	buying	maint	doors	persons	lugboot	safety	class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc
...	...	...	...	...	...	...	...
1722	low	low	5more	more	med	med	good
1723	low	low	5more	more	med	high	vgood
1724	low	low	5more	more	big	low	unacc
1725	low	low	5more	more	big	med	good

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   buying   1727 non-null   object
 1   maint    1727 non-null   object
 2   doors    1727 non-null   object
 3   persons  1727 non-null   object
 4   lugboot  1727 non-null   object
 5   safety   1727 non-null   object
 6   class    1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Data Cleaning

<u>Data Cleaning</u> **: Data cleaning is the process of identifying and correcting or removing inaccurate, incomplete, or irrelevant data from a dataset .The process performed for data cleaning are as follows:**

- **Label Encoding the columns: Buying, maintenance, lugboot, safety and class**
- **Replacing '5more' with the numerical value '5' in the doors column.**
- **Replacing 'more' with the numerical value '5' in the persons column.**

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder

columns_to_encode= ['buying','maint','lugboot','safety','class']
le= LabelEncoder()
for col in columns_to_encode:
    df[col]=le.fit_transform(df[col])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   buying   1727 non-null   int64
 1   maint    1727 non-null   int64
 2   doors    1727 non-null   object
 3   persons  1727 non-null   object
 4   lugboot  1727 non-null   int64
 5   safety   1727 non-null   int64
 6   class    1727 non-null   int64
dtypes: int64(5), object(2)
memory usage: 94.6+ KB
```

```python
#Replace '5more' with 5
df['doors'] = df['doors'].replace('5more', 5)

# Optional: Convert the column to integer if needed
df['doors'] = df['doors'].astype(int)

# Display result
print(df)
```

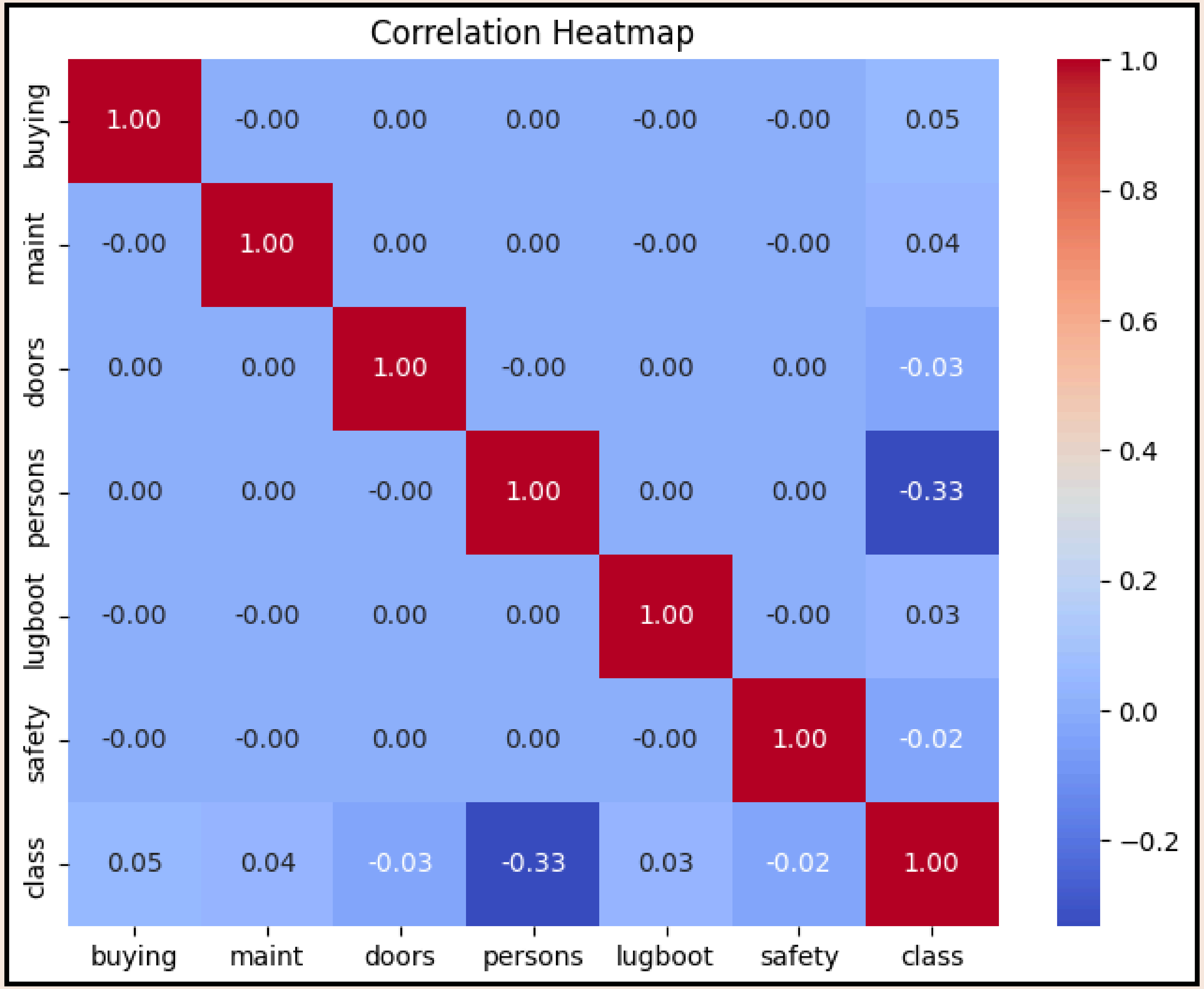
```
      buying  maint  doors persons  lugboot  safety  class
0          3      3      2       2        2       2      2
1          3      3      2       2        2       0      2
2          3      3      2       2        1       1      2
3          3      3      2       2        1       2      2
4          3      3      2       2        1       0      2
...      ...    ...    ...     ...      ...     ...    ...
1722       1      1      5    more        1       2      1
1723       1      1      5    more        1       0      3
1724       1      1      5    more        0       1      2
1725       1      1      5    more        0       2      1
1726       1      1      5    more        0       0      3

[1727 rows x 7 columns]
```

```python
df['persons'] = df['persons'].replace('more', 5)
df['persons'] = df['persons'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   buying   1727 non-null   int64
 1   maint    1727 non-null   int64
 2   doors    1727 non-null   int64
 3   persons  1727 non-null   int64
 4   lugboot  1727 non-null   int64
 5   safety   1727 non-null   int64
 6   class    1727 non-null   int64
dtypes: int64(7)
memory usage: 94.6 KB
```

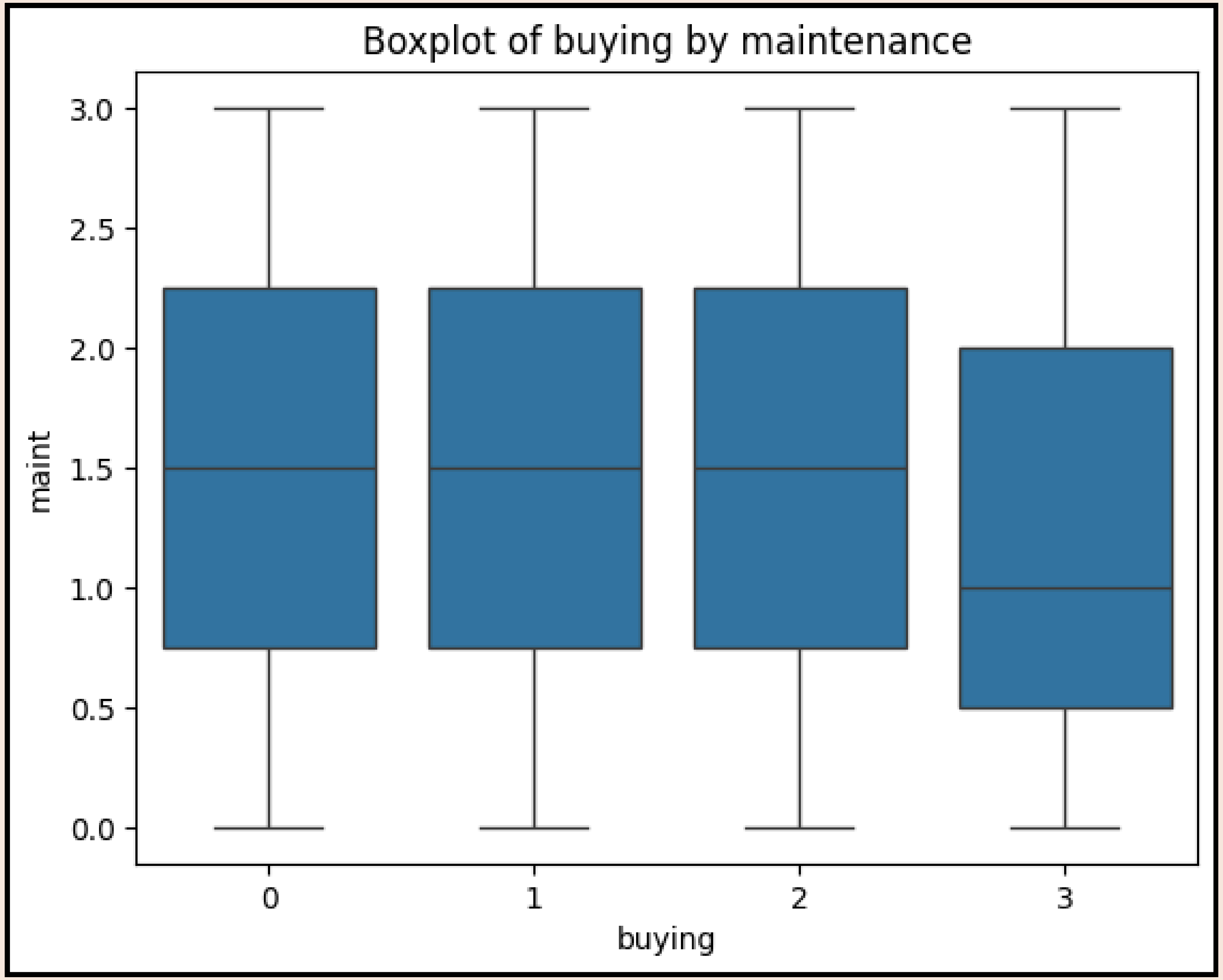
Exploratory Data Analysis (EDA)

Correlation Heatmap



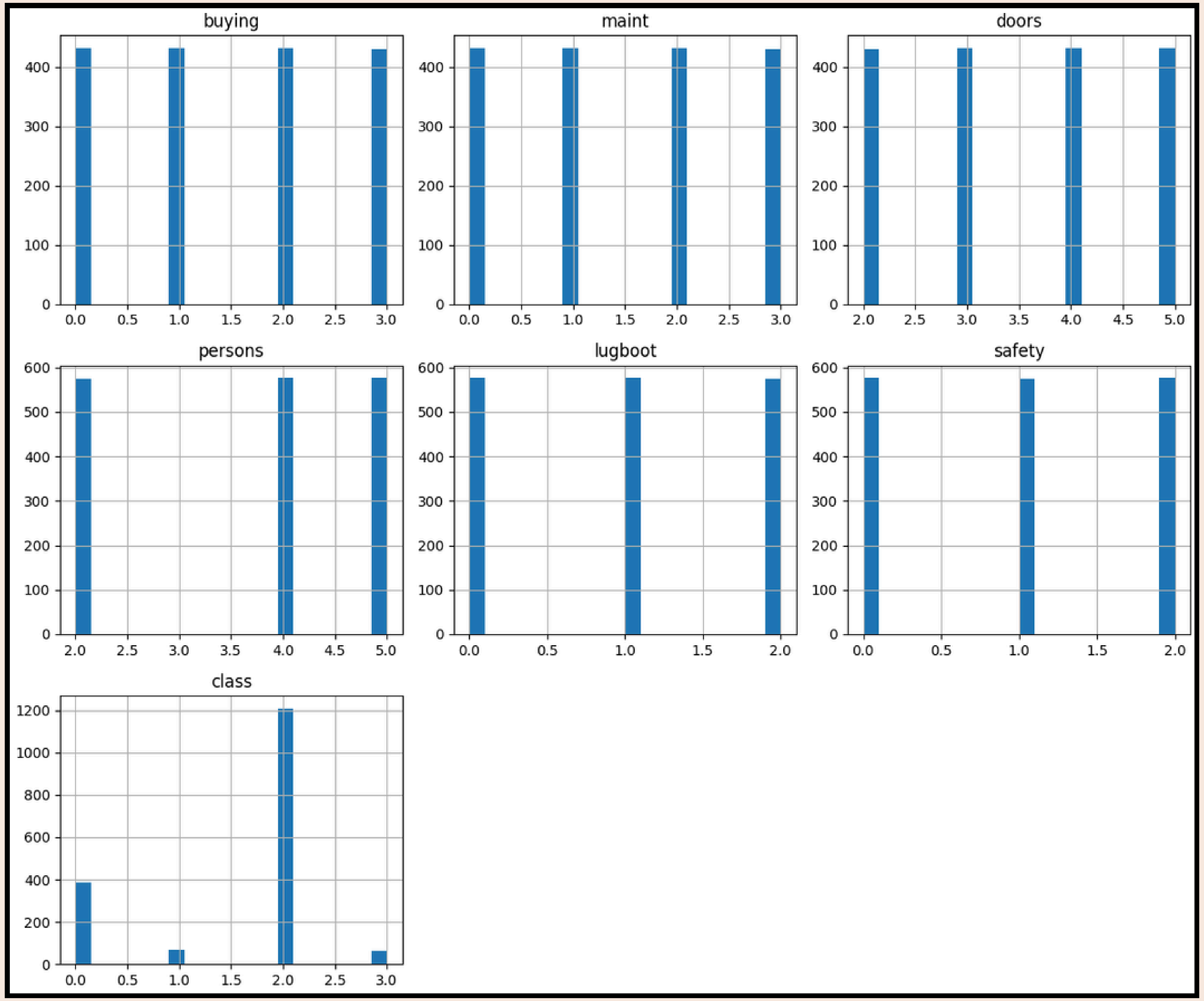
A graphical representation that shows the strength and direction of relationships between numerical variables using color-coded cells.

Box-plot



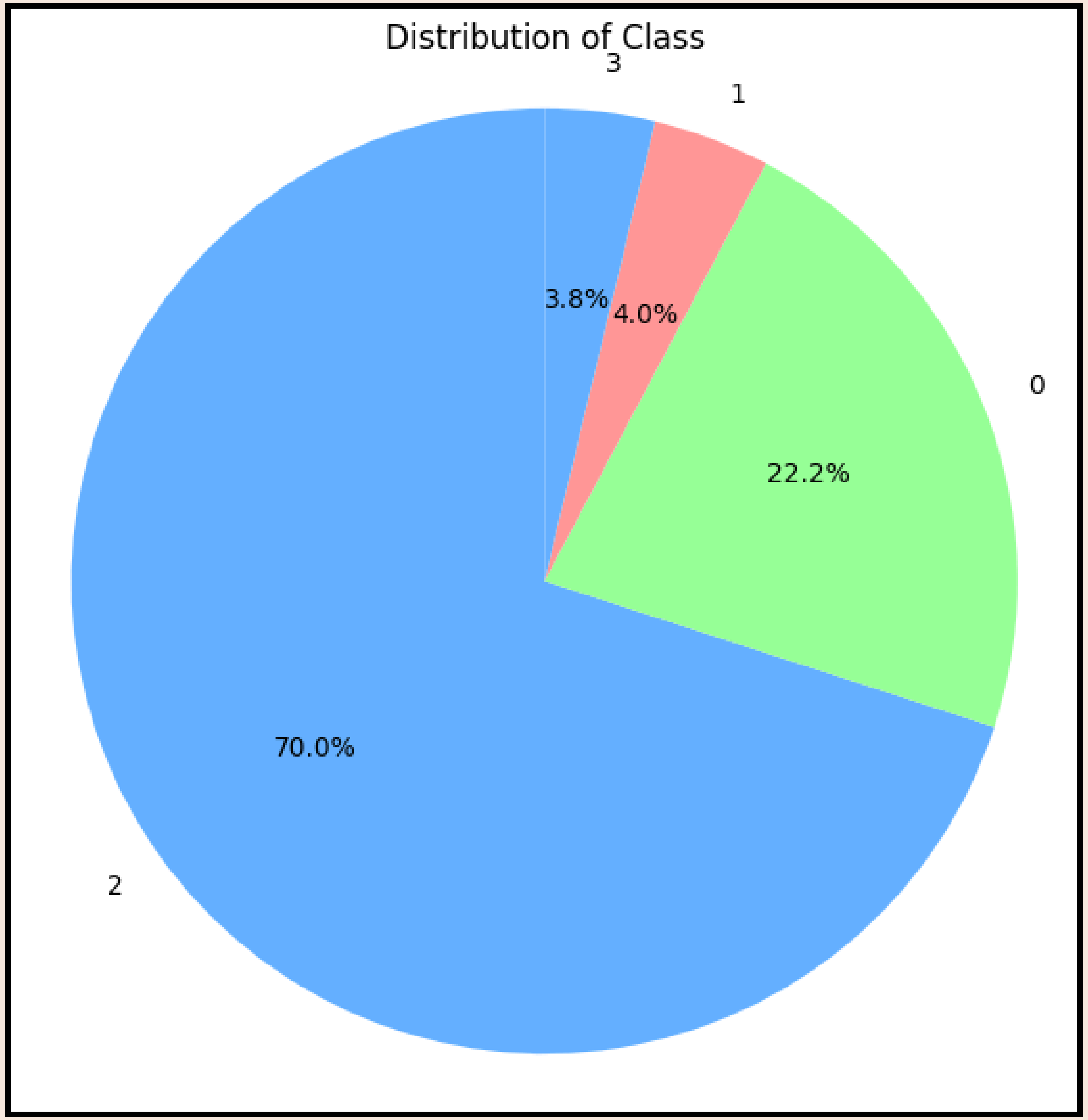
A graphical representation of the distribution of a numerical variable that displays the median, quartiles, and potential outliers, helping identify variability and skewness in the data.

Histogram



A bar graph that displays the distribution of a numerical variable by grouping data into ranges (bins).

Pie-plot



A circular chart that shows the proportion of different categories as slices of a whole.

Analysis

Machine Learning Algorithms

<u>K-Nearest Neighbors (KNN):</u> A simple, instance-based algorithm that classifies data points based on the majority class of their closest neighbors.
<u>Support Vector Machine (SVM)</u>: A powerful algorithm that finds the optimal hyperplane to separate classes by maximizing the margin between them.
<u>Decision Trees:</u> A flowchart-like model that splits data into branches based on feature values to make predictions.
<u>Random Forests:</u> An ensemble learning method that builds multiple decision trees and combines their outputs for more accurate and stable predictions.
<u>Bagging (Bootstrap Aggregating):</u> An ensemble technique that trains multiple models on random subsets of data and aggregates their predictions to reduce variance.
<u>AdaBoost (Adaptive Boosting):</u> A boosting method that combines weak learners sequentially, giving more focus to previously misclassified instances to improve accuracy.

Splits / Algorithms	80-20 SPLIT	70-30 SPLIT	75-25 SPLIT	60-40 SPLIT
KNN	96.2	95.3	95.1	95.8
SVM	90.1	89.3	90.9	90.7
DECISION TREES	84.1	84.9	86.5	85.3
RANDOM FORESTS	78	75.9	79.9	77.4
BAGGING	91.3	91.6	92.4	91.4
ADABOOST	84.3	84.9	86.5	82.2

Real-time Prediction

Real-time Prediction

The project was extended to support real-time prediction using user-inputted car attributes.

- Inputs include: buying price, maintenance cost, number of doors, seating capacity, and luggage boot size.
- The original multi-class classification (unacceptable, acceptable, good, vgood) was simplified to binary classes.
- The binary output labels are:
- Acceptable (includes acceptable, good, and vgood)
- Unacceptable (remains as is)
- This feature enables instant classification based on user-provided values.
- It enhances the project's practical applicability, especially for car buyers, sellers, or dealership evaluation tools.

```python
# Example input (must be in the same order and format as your original X):
# For instance: [buying, maint, doors, persons, lugboot, safety]
new_car = [3, 3, 5, 5, 2, 2]  # Include all 6 features, update values

result = real_time_prediction(new_car, rfe, best_bagging_model)
print("🚗 Real-Time Car Condition Prediction:", result)
```

```
🚗 Real-Time Car Condition Prediction: Acceptable
```

Summary

Summary

This project aimed to classify car conditions based on key attributes such as buying price, maintenance cost, number of doors, seating capacity, and luggage boot size using various machine learning algorithms. The initial model performed multi-class classification, which was later optimized into a binary classification system for real-time prediction. Among all models tested, the Bagging classifier of the 70-30 split was the most optimal and achieved the best accuracy, making it suitable for practical deployment.

Future Scope

Future Scope

Integrate real-world data from used car listings to enhance model robustness and accuracy.
Extend the model to include numerical features like mileage, engine size, or car age for deeper insight.
Deploy the system as a web or mobile app to assist car buyers and dealerships in instant condition assessment.
Implement model explainability tools like SHAP to improve trust and interpretability in predictions.

THANK YOU