

Task 2 – Spark ML Classifier

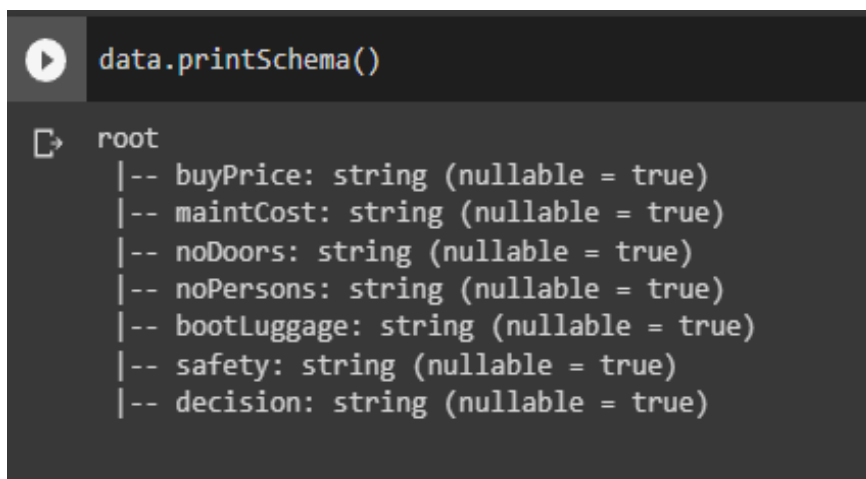
2.1 Introduction

In this task, the aim is to build a classifier using Spark ML that can predict car acceptability based on various attributes. The classifier can help potential buyers and sellers understand the desirability of a car in the market.

2.2 Data Analysis:

The dataset includes 1,728 instances with six distinct attributes, including buying price, maintenance cost, number of doors, capacity, luggage boot size, and safety value.

The schema of the dataset is as follows:

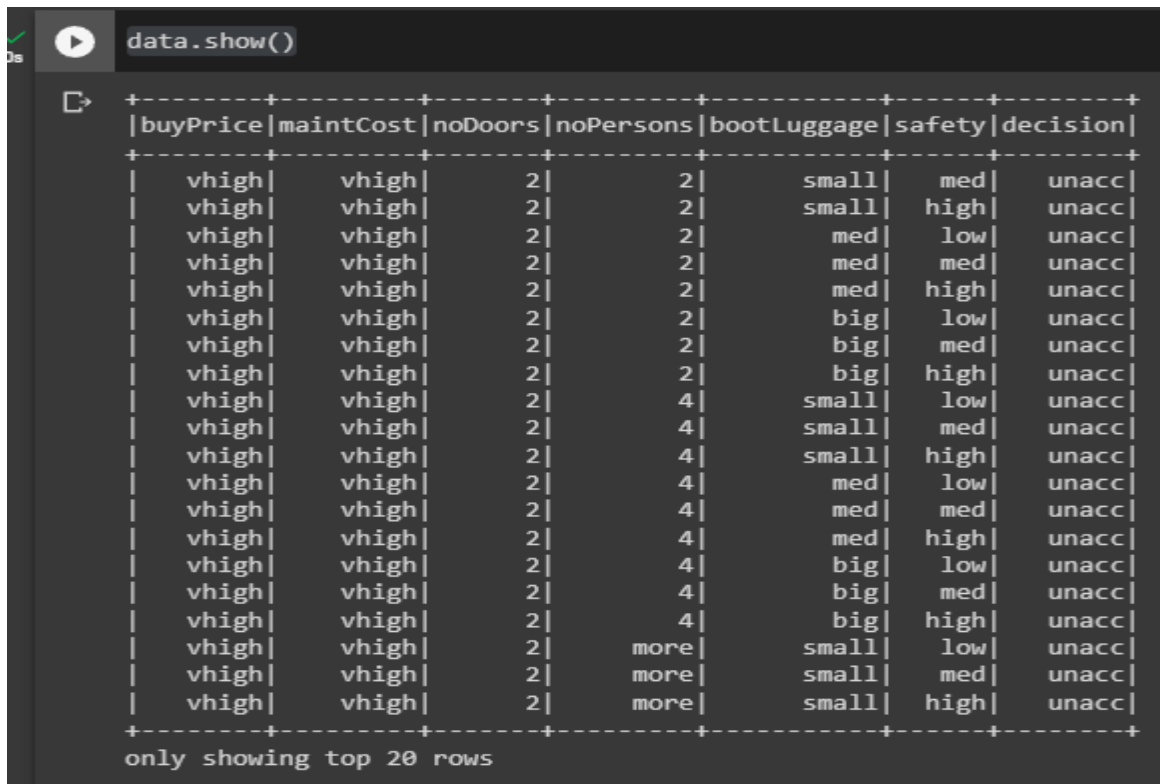
A screenshot of a code editor with a dark background. At the top, there is a play button icon followed by the text `data.printSchema()`. Below this, there is a folder icon followed by the text `root`. Under `root`, there are seven lines of text, each representing a field in the schema: `-- buyPrice: string (nullable = true)`, `-- maintCost: string (nullable = true)`, `-- noDoors: string (nullable = true)`, `-- noPersons: string (nullable = true)`, `-- bootLuggage: string (nullable = true)`, `-- safety: string (nullable = true)`, and `-- decision: string (nullable = true)`.

```
data.printSchema()

root
 |-- buyPrice: string (nullable = true)
 |-- maintCost: string (nullable = true)
 |-- noDoors: string (nullable = true)
 |-- noPersons: string (nullable = true)
 |-- bootLuggage: string (nullable = true)
 |-- safety: string (nullable = true)
 |-- decision: string (nullable = true)
```

Figure 1 - Dataset Schema

A preview of the dataset:



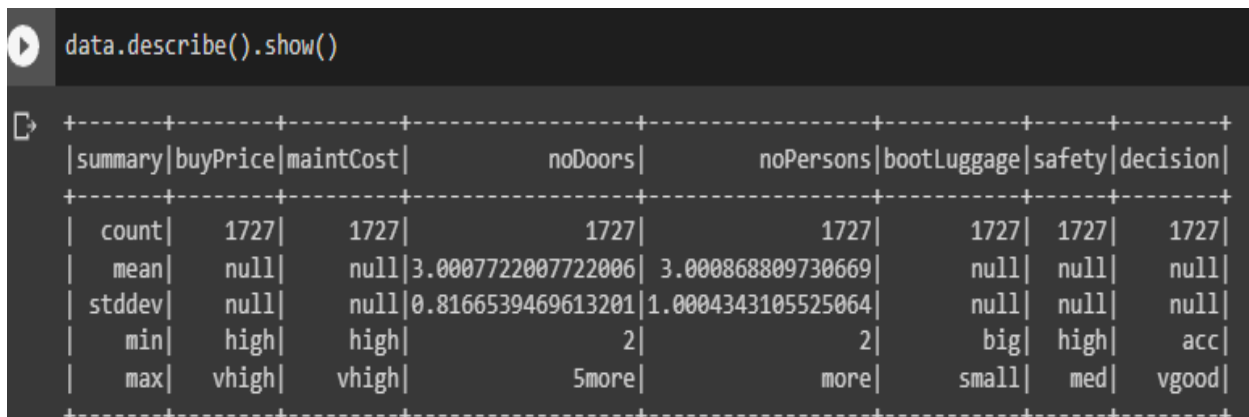
```
data.show()
```

buyPrice	maintCost	noDoors	noPersons	bootLuggage	safety	decision
vhigh	vhigh	2	2	small	med	unacc
vhigh	vhigh	2	2	small	high	unacc
vhigh	vhigh	2	2	med	low	unacc
vhigh	vhigh	2	2	med	med	unacc
vhigh	vhigh	2	2	med	high	unacc
vhigh	vhigh	2	2	big	low	unacc
vhigh	vhigh	2	2	big	med	unacc
vhigh	vhigh	2	2	big	high	unacc
vhigh	vhigh	2	4	small	low	unacc
vhigh	vhigh	2	4	small	med	unacc
vhigh	vhigh	2	4	small	high	unacc
vhigh	vhigh	2	4	med	low	unacc
vhigh	vhigh	2	4	med	med	unacc
vhigh	vhigh	2	4	med	high	unacc
vhigh	vhigh	2	4	big	low	unacc
vhigh	vhigh	2	4	big	med	unacc
vhigh	vhigh	2	4	big	high	unacc
vhigh	vhigh	2	more	small	low	unacc
vhigh	vhigh	2	more	small	med	unacc
vhigh	vhigh	2	more	small	high	unacc

only showing top 20 rows

Figure 2 - Dataset Preview

The summary statistics for each attribute are:



```
data.describe().show()
```

summary	buyPrice	maintCost	noDoors	noPersons	bootLuggage	safety	decision
count	1727	1727	1727	1727	1727	1727	1727
mean	null	null	3.0007722007722006	3.000868809730669	null	null	null
stddev	null	null	0.8166539469613201	1.0004343105525064	null	null	null
min	high	high	2	2	big	high	acc
max	vhigh	vhigh	5more	more	small	med	vgood

Figure 3 -Data Summary Statistics

The number of unique values in each categorical column:

```
▶ for col_name in data.columns:
    print(f"{col_name}: {data.select(col_name).distinct().count()} unique values")

↳ buyPrice: 4 unique values
   maintCost: 4 unique values
   noDoors: 4 unique values
   noPersons: 3 unique values
   bootLuggage: 3 unique values
   safety: 3 unique values
   decision: 4 unique values
```

Figure 4 - Column Unique Values

The dataset has no missing values:

```
▶ # Check for missing values in the dataset
  for col_name in data.columns:
      print(f"{col_name}: {data.filter(col(col_name).isNull()).count()} missing values")

↳ buyPrice: 0 missing values
   maintCost: 0 missing values
   noDoors: 0 missing values
   noPersons: 0 missing values
   bootLuggage: 0 missing values
   safety: 0 missing values
   decision: 0 missing values
```

Figure 5 - Missing Values in the Dataset

Target Variable Distribution:

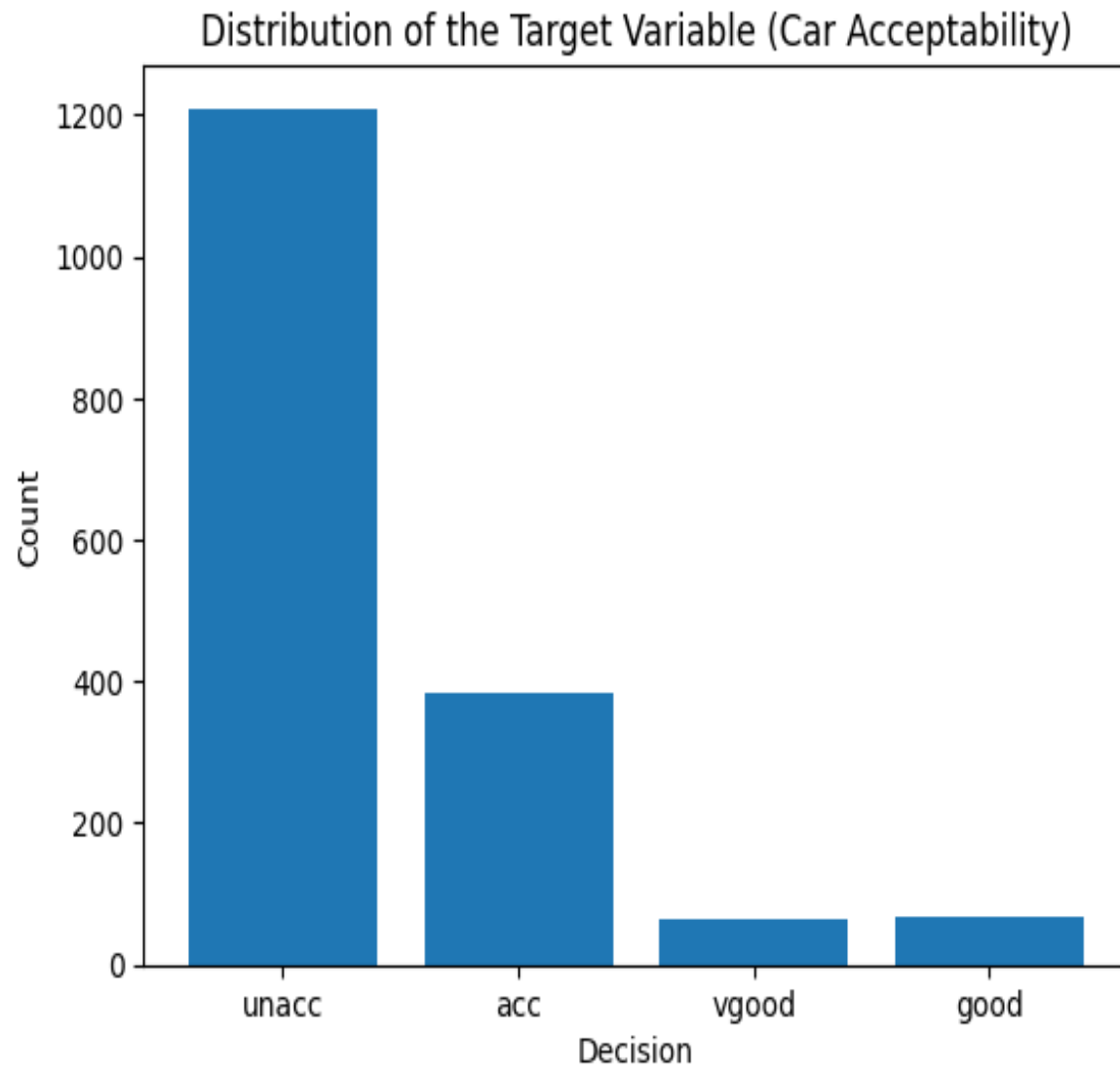


Figure 6 - Target Variable Distribution

The results revealed that the dataset is clearly dominated by unacceptable cars.

Analyzing the distribution of each feature in the dataset:

i) Buy price

Medium buying level cars are more acceptable than expensive cars, while expensive cars are the most unacceptable and low-grade cars are better than expensive cars.

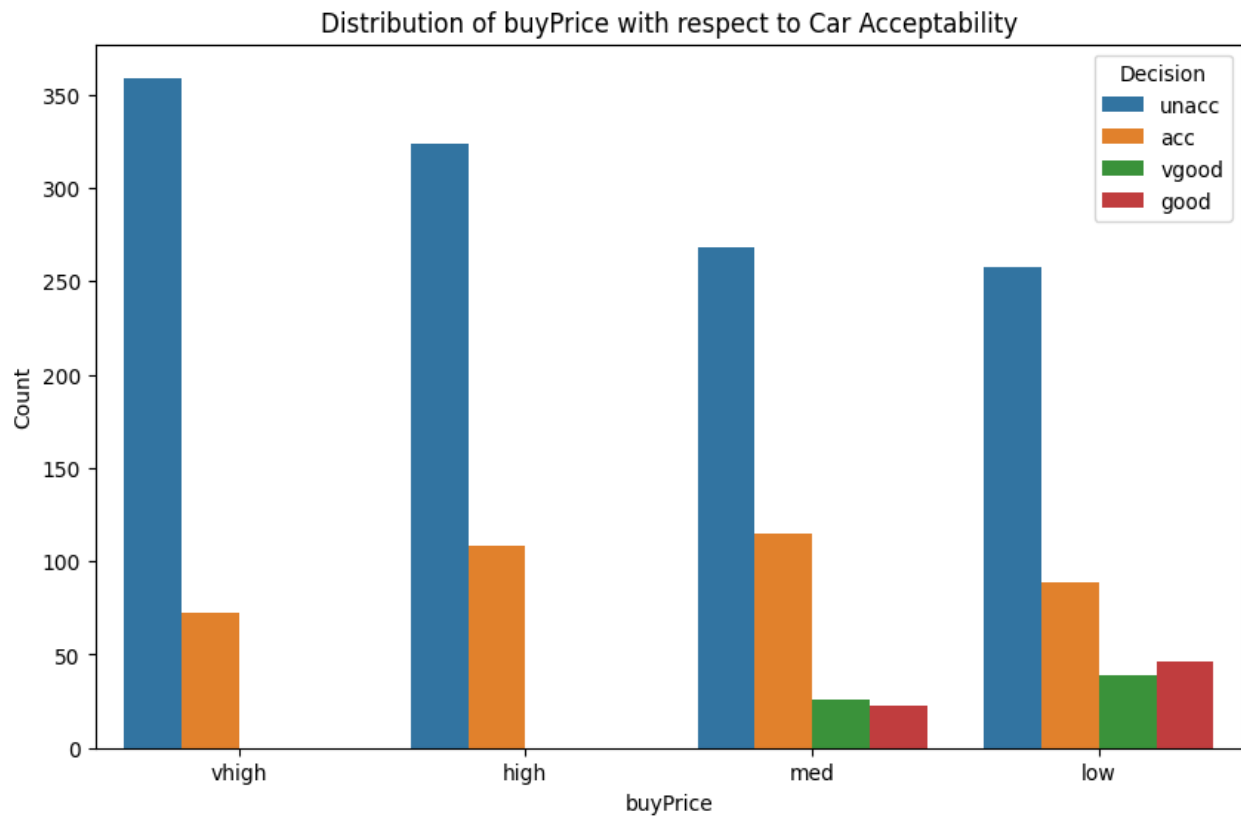


Figure 7 - Buy Price

ii) Maintenance Cost

The higher the price of a car, the higher the maintenance cost, as it is linked to the price of the car.

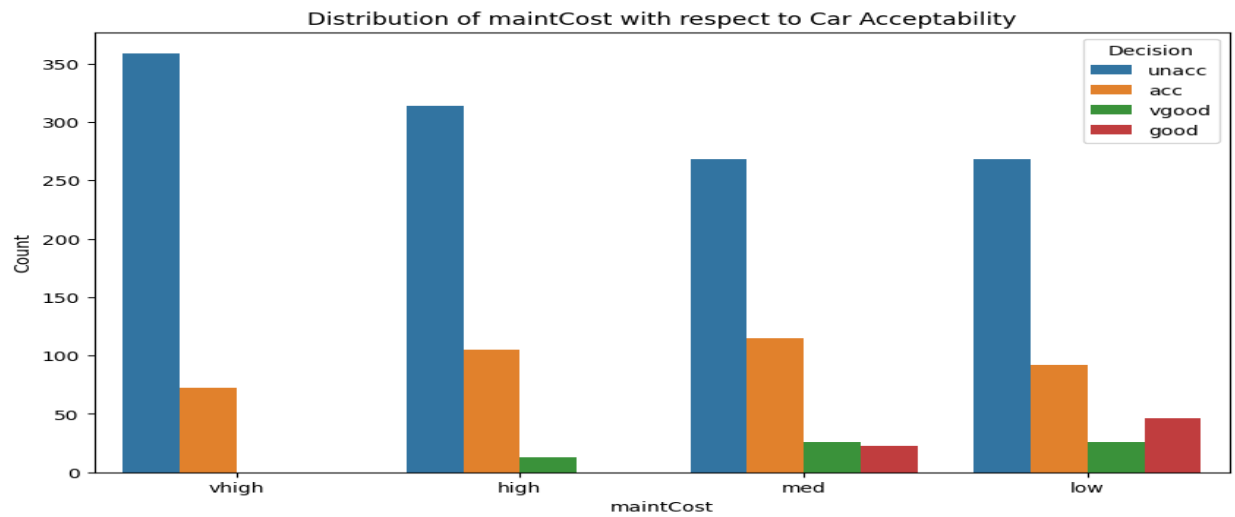


Figure 8 - Maintenance Cost

iii) Number of doors

The number of doors is equivalently distributed between 3,4 to 5 or more cars, and 2 door cars are not preferred.

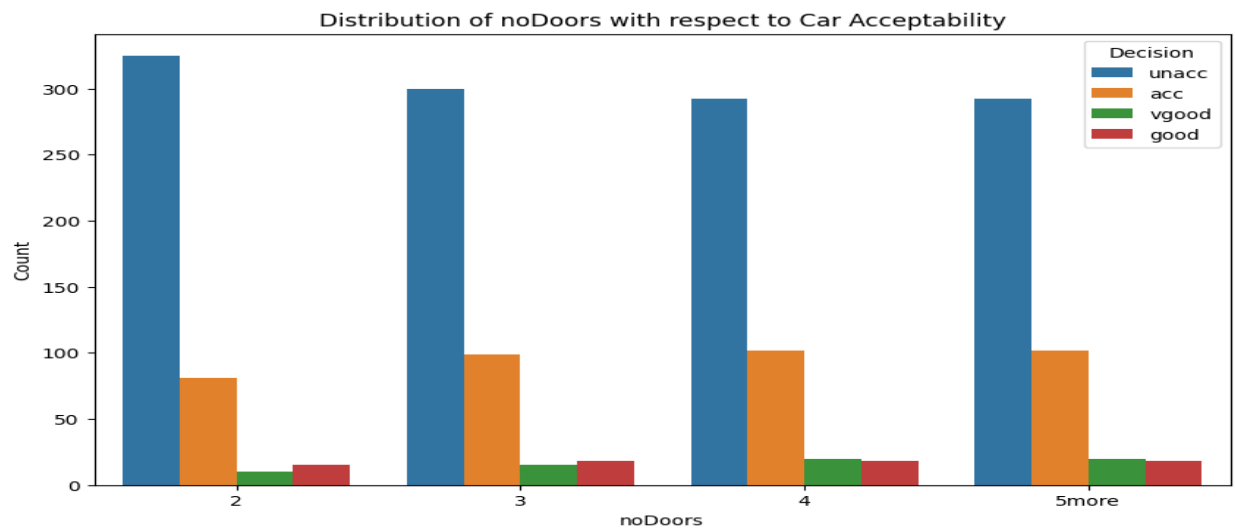


Figure 9 - Number of doors

iv) Number of Persons

Four and more person cars are most acceptable and a single user prefers cars with 2-person accommodation.

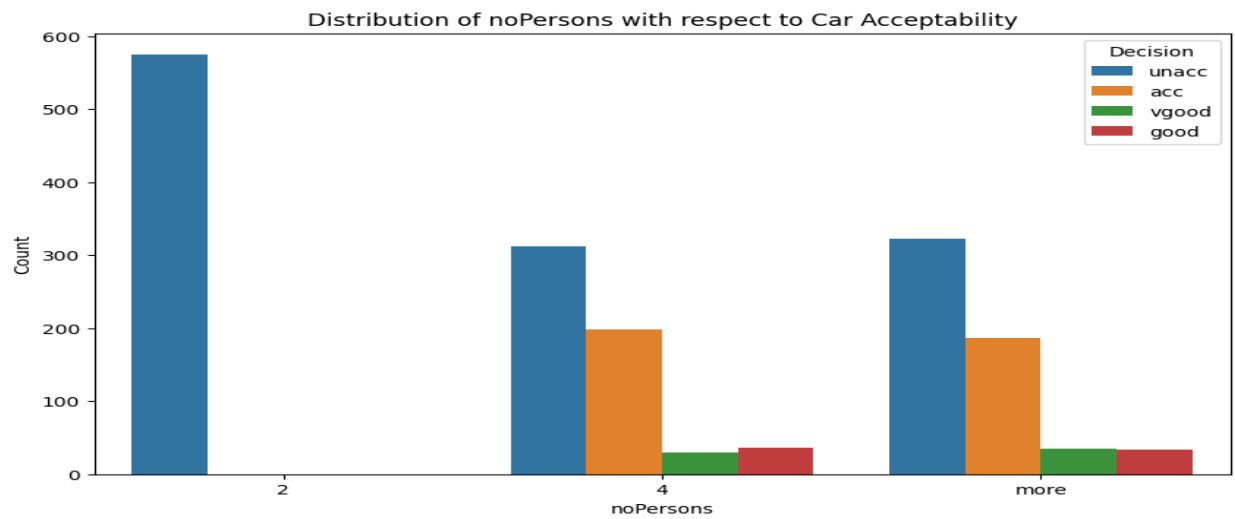


Figure 10 - Number of Persons

v) Luggage Boot Size

The big boot sized luggage cars are the most acceptable.

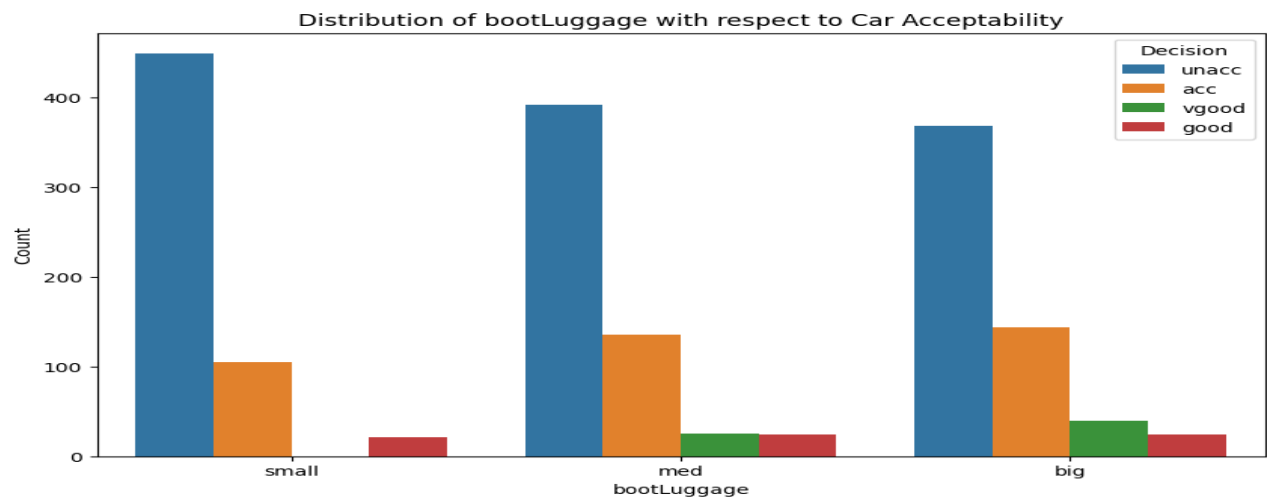


Figure 11- Luggage Boot Size

vi) Safety

The high safety vehicles are most preferred.

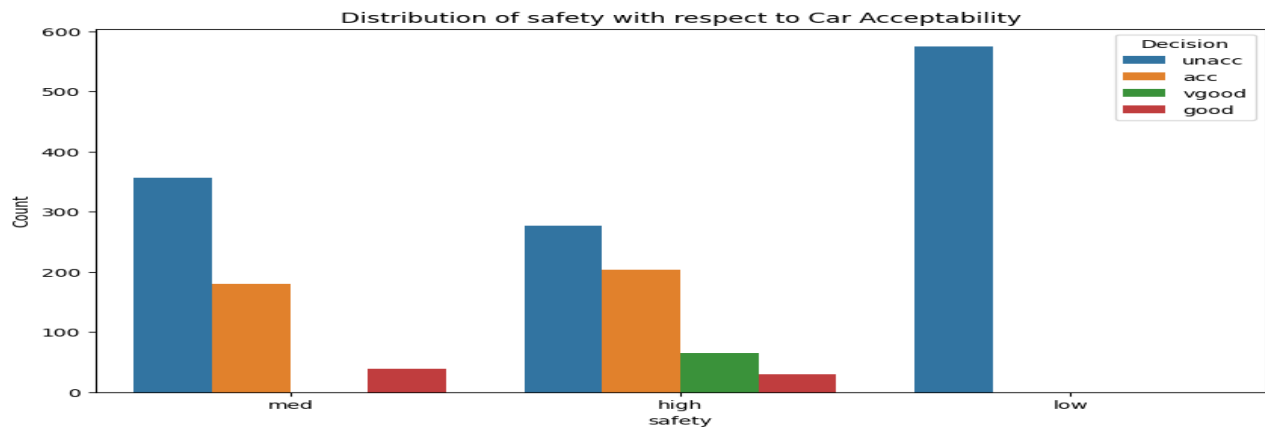


Figure 12 - Safety

2.3 Data Preparation

The data underwent preprocessing, which involved indexing the categorical features using the StringIndexer method provided by Spark ML. This step was crucial as most machine learning algorithms require numerical inputs. The following figure displays a sample of the transformed dataset, including both the original categorical values and their corresponding indexed values.

```
# Show a sample of the transformed data with the indexed values
selected_columns = [col_name for col_name in data.columns if "_index" in col_name] + categorical_columns
sample_data = data.select(selected_columns)
sample_data.show(n=10)
```

buyPrice_index	maintCost_index	noDoors_index	noPersons_index	bootLuggage_index	safety_index	class_index	buyPrice	maintCost	noDoors	noPersons	bootLuggage	safety
3.0	3.0	3.0	2.0	2.0	1.0	0.0	vhigh	vhigh	2	2	small	med
3.0	3.0	3.0	2.0	2.0	0.0	0.0	vhigh	vhigh	2	2	small	high
3.0	3.0	3.0	2.0	1.0	2.0	0.0	vhigh	vhigh	2	2	med	low
3.0	3.0	3.0	2.0	1.0	1.0	0.0	vhigh	vhigh	2	2	med	med
3.0	3.0	3.0	2.0	1.0	0.0	0.0	vhigh	vhigh	2	2	med	high
3.0	3.0	3.0	2.0	0.0	2.0	0.0	vhigh	vhigh	2	2	big	low
3.0	3.0	3.0	2.0	0.0	1.0	0.0	vhigh	vhigh	2	2	big	med
3.0	3.0	3.0	2.0	0.0	0.0	0.0	vhigh	vhigh	2	2	big	high
3.0	3.0	3.0	0.0	2.0	2.0	0.0	vhigh	vhigh	2	4	small	low
3.0	3.0	3.0	0.0	2.0	1.0	0.0	vhigh	vhigh	2	4	small	med

only showing top 10 rows

Figure 13 - Transformed Dataset

2.4 Model Training and Evaluation

The dataset was split into training and testing sets, with an 80-20 ratio. Two machine learning models, a Decision Tree and a Random Forest classifier, were built and evaluated using classification reports.

Table 1 - Decision Tree Classification Report

Decision Tree classifier				
Results:				
<ul style="list-style-type: none">The accuracy score is 0.88, indicating that the model correctly classified 88% of the test instances.The F1-score for the unacceptable class is 0.95, indicating a good balance between precision and recall for this class.The F1-score for the acceptable class is 0.77, which is lower than that of the unacceptable class but still reasonable.The F1-score for the good and very good classes is 0.00 and 0.77, respectively, showing that the model struggles to correctly classify the good class.				
<pre>[29] print("Decision Tree Classification Report:\n", dt_report)</pre>				
Decision Tree Classification Report:				
	precision	recall	f1-score	support
acc	0.73	0.81	0.77	72
good	0.00	0.00	0.00	11
unacc	0.94	0.95	0.95	227
vgood	0.71	0.86	0.77	14
accuracy			0.88	324
macro avg	0.60	0.65	0.62	324
weighted avg	0.85	0.88	0.87	324

Table 2 - Random Forrest Classification Report

Random Forest classifier

Results:

- The accuracy score is 0.90, indicating that the model correctly classified 90% of the test instances, slightly outperforming the Decision Tree classifier.
- The F1-score for the unacceptable class is 0.96, indicating a good balance between precision and recall for this class, similar to the Decision Tree classifier.
- The F1-score for the acceptable class is 0.80, which is higher than that of the Decision Tree classifier, showing an improvement in classification for this class.
- The F1-score for the good and very good classes is 0.00 and 0.92, respectively, showing that the model still struggles to correctly classify the good class but performs better for the very good class than the Decision Tree classifier.



```
print("Random Forest Classification Report:\n", rf_report)
```



Random Forest Classification Report:

	precision	recall	f1-score	support
acc	0.72	0.90	0.80	72
good	0.00	0.00	0.00	11
unacc	0.97	0.95	0.96	227
vgood	1.00	0.86	0.92	14
accuracy			0.90	324
macro avg	0.67	0.68	0.67	324
weighted avg	0.88	0.90	0.89	324

2.5 Confusion Matrix Analysis

The confusion matrices for both the Decision Tree and Random Forest classifiers provide a detailed view of the performance of each model in terms of correct and incorrect predictions for each class.

Table 3 - Decision Tree Confusion Matrix

Decision Tree classifier

Analysis:

- The majority of the instances are correctly classified, with 215 out of 227 unacceptable instances, 58 out of 72 acceptable instances, and 12 out of 14 very good instances predicted accurately.
- However, the model struggles to classify the good class, with none of the 11 instances correctly predicted.
- There are also some misclassifications between the unacceptable and acceptable classes, with 13 acceptable instances predicted as unacceptable and 11 unacceptable instances predicted as acceptable.



Table 4 - Random Forrest Confusion Matrix

Random Forest classifier

Analysis:

- The model shows a slight improvement in the classification of the acceptable class, with 65 out of 72 instances correctly predicted, and a comparable performance for the unacceptable class, with 215 out of 227 instances correctly predicted.
- The very good class is also predicted with a similar accuracy as the Decision Tree classifier, with 12 out of 14 instances correctly predicted.
- However, the Random Forest classifier also struggles to classify the good class, with none of the 11 instances correctly predicted.
- Misclassifications between the unacceptable and acceptable classes are slightly reduced compared to the Decision Tree classifier, with 7 acceptable instances predicted as unacceptable and 12 unacceptable instances predicted as acceptable.



Both classifiers perform well in classifying the unacceptable, acceptable, and very good classes, but struggle to correctly classify the good class, suggesting further improvements could be made.

2.6 Feature Importance Analysis

An analysis of the feature importance in the Random Forest model revealed that safety and the number of persons a car can carry were the most critical factors determining car acceptability. In contrast, the number of doors appeared to have no significant influence on the classification.

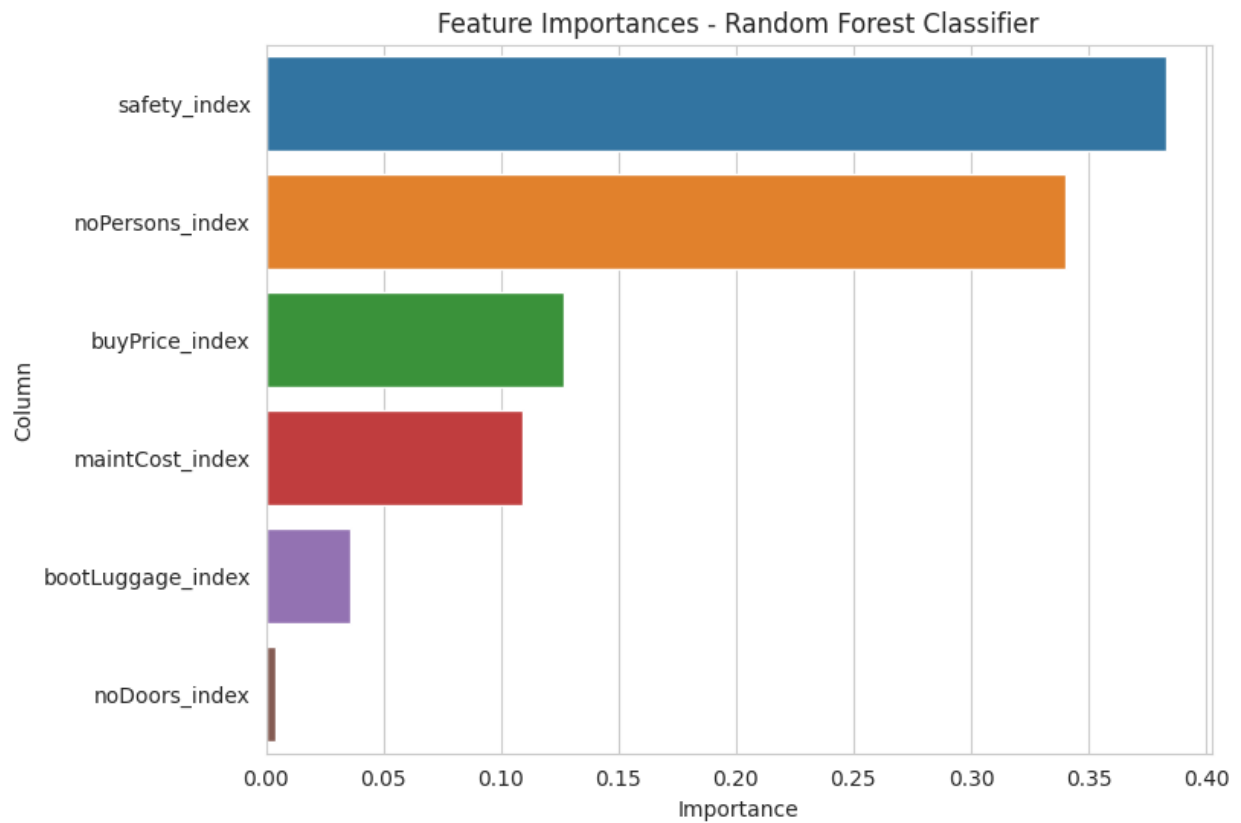


Figure 14 - Feature Importance Analysis

2.7 Justification for Selected Algorithms

Table 5 - Algorithm Justifications

Criteria	Description
Handling categorical data	Both Decision Trees and Random Forests can naturally handle categorical data, which is a common feature of the car evaluation dataset. This makes it easier to preprocess the data and apply these algorithms without the need for extensive feature engineering.
Interpretability	Decision Trees offer excellent interpretability, allowing you to visualize the decision-making process and understand the importance of individual features. Random Forests, as an ensemble of Decision Trees, also provide feature importance rankings, which can help you interpret the model's decision-making process.
Avoiding overfitting	Decision Trees are prone to overfitting, especially with noisy or complex datasets. Random Forests, as an ensemble method, can address this issue by averaging the predictions from multiple trees, leading to a more robust and accurate model.
Efficiency	Decision Trees and Random Forests are relatively efficient algorithms, making them suitable for working with moderate-sized datasets. This ensures that the model training and evaluation process is not overly time-consuming.
Model performance	Both Decision Trees and Random Forests are known for their good performance on various classification tasks, including those with mixed data types (categorical and numerical) like the car evaluation dataset.

2.8 Conclusion

The Decision Tree and Random Forest models performed well in the car classification task, with the latter demonstrating marginally superior performance in terms of accuracy. Potential improvements to be considered for future experimentation include data preprocessing, model parameters tuning, model evaluation, handling imbalanced classes, and trying additional models such as Gradient Boosted Trees, Logistic Regression, or Support Vector Machines. Data preprocessing can try using OneHotEncoder instead of StringIndexer, model parameters should be tuned, model evaluation can use Stratified K-Fold Cross Validation, and handling imbalanced classes can use SMOTE (Synthetic Minority Oversampling Technique).