

REPORT ON THE PREDECTING POPULARITY OF CARS USING MODELS

PROBLEM STATEMENT:

Given features such as cost price, maintenance cost, safety ratio, number of doors and seats create a model which can predict the popularity of the car given a set of features. Plot necessary graphs and analyse the data and derive inferences.

APPROACH:

When I was given the data set the following were some of the questions which I asked myself-

1. What features of the given are most important and which are not?
2. How does each popularity vary with the features?
3. What models fit properly to this current scenario?
4. What was trending in the given population??

I began searching for the truth. Initially I plotted couple of bar graphs showing the variation of each popularity with respect to features: -

The **charts** have been included at the **END**

ANALYSIS AND OBSERVATION:

I went through each of the graphs for each popularity. I found the features which contributed the highest and which didn't contribute so much.

From the graphs I concluded that-

1. Features such as **number of doors** and **luggage boot size** **didn't contribute** much towards the popularity as the categories of number of doors and luggage boot size had the **equal probabilities** of occurring in a feature vector.
2. Features such as **number of seats, buying price, maintenance cost, safety rating** were the important features to focus more on.

:

For instance, we can conclude that a car which has a **safety rating-1** and **number of seats - 2, luggage space-1** and **cost category-4** is most likely to be popular(category-1).

Similarly, we can predict the popularity of the car by looking at the patterns of the feature vectors.

Using these statistics is potentially useful for the companies which desire to manufacture cars according to the needs of the population

I then wanted to verify this fact through algorithms such as -Recursive Feature Elimination(RFE), RandomForestClassifier(and measure the best features), Principal Component Analysis, Univariate Selection.

A short note on what above algorithm does:

1. Univariate Selection:

Statistical tests can be used to select those features that have the strongest relationship with the output variable.

The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

I used the chi squared (χ^2) statistical test for non-negative features to select 4 of the best features from the dataset.

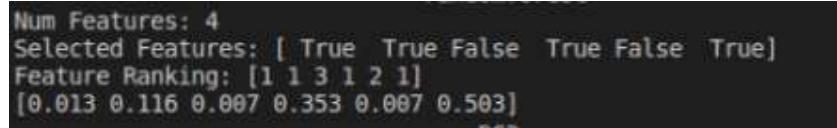
The output was  -

2. Recursive Feature Elimination(RFE)

The Recursive Feature Elimination (or RFE) works recursively removing attributes and building a model on those attributes that remain.

It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.


The example below uses RFE with the logistic regression algorithm to select the top 3 features.



3. Principle component analysis(PCA):

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form.

A property of PCA is that you can choose the number of dimensions or principal component in the transformed result.



From these I came to a conclusion about the important features that I noted were indeed right. Then I moved on to build a good model which will fit the data.

MODELLING THE DATA:

I had checked for data consistency earlier to plotting graphs and made sure that the data was in required format and was clean.

Initially I started off with Random Forest Classifier. I split the data into 2 parts – One for training (3/4th dataset) and remaining for testing (1/4th dataset).

The model was trained I noted that the accuracy was \approx 95-96 percent for the data. I then chose the 3/4th data randomly and trained and tested and the accuracy was almost the same.

Then I used decision trees to classify the data using the same procedure and found that the accuracy was in between 96.5 to 97.5 percent. The degree of randomness was set to 1.

I also tried using Nearest Neighbours, Support Vector Machine, Multi neural perceptron, Multi-class classification. These models had accuracy about 93.5 percent.

[illegible]

The above image is for decision trees

Reasons for choosing above algorithms:

I chose decision trees as it can predict accurately a class accurately over large number of features. A decision tree can help you weigh the likely consequences of one decision against another. Decision trees are relatively easy to understand when there are few decisions and outcomes included in the tree. These Help determine worst, best and expected values for different scenarios. In this case the scenarios are the patterns associated with each popularity class of cars. So its highly useful.

I used Random forest as –

It maintains accuracy even when a large proportion of the data are missing. And also they have lesser variance. Also there is less possibility of overfitting as it averages out/combines the result at each stage. But decision trees may overfit sometimes.

The other algorithms were tested initially out of curiosity and I just wanted to see their performance for this kind of data.

I concluded that the decision tree had the highest accuracy and Random forest can also be considered for better consistency.

CODE:

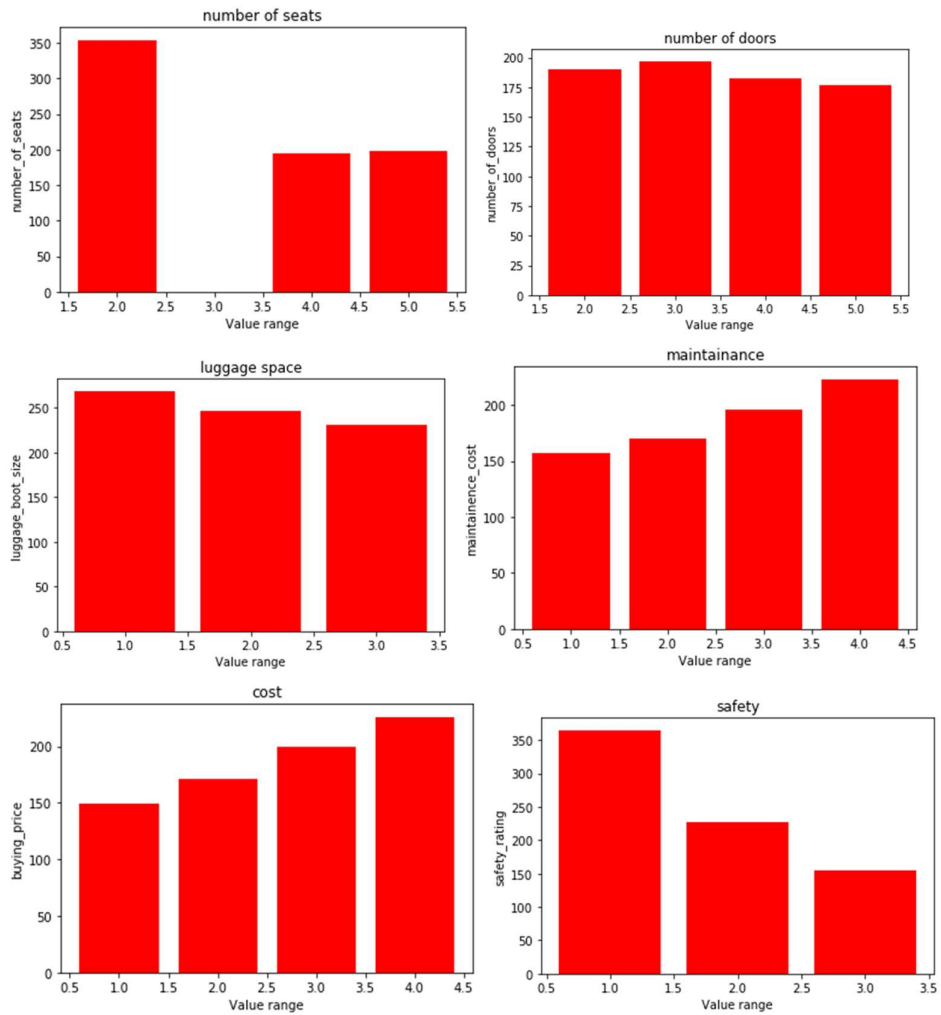
Actualdecisiontree.ipynb and actualrandomforest.ipynb are the files for the cars test.csv

To verify the accuracy run `decisiontree.ipynb` and `randomforest.ipynb`

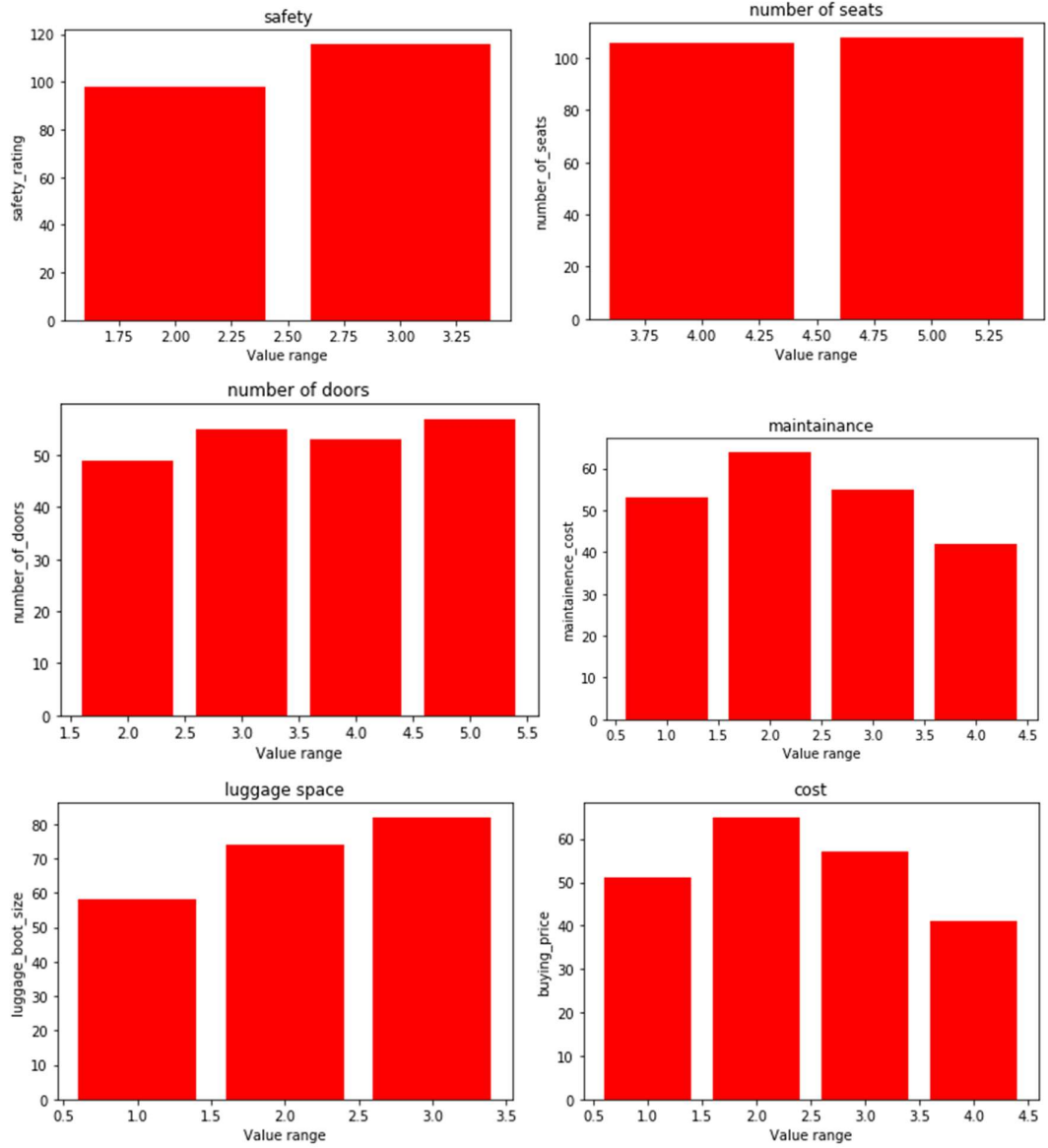
BAR CHARTS

POPULARITY-WISE

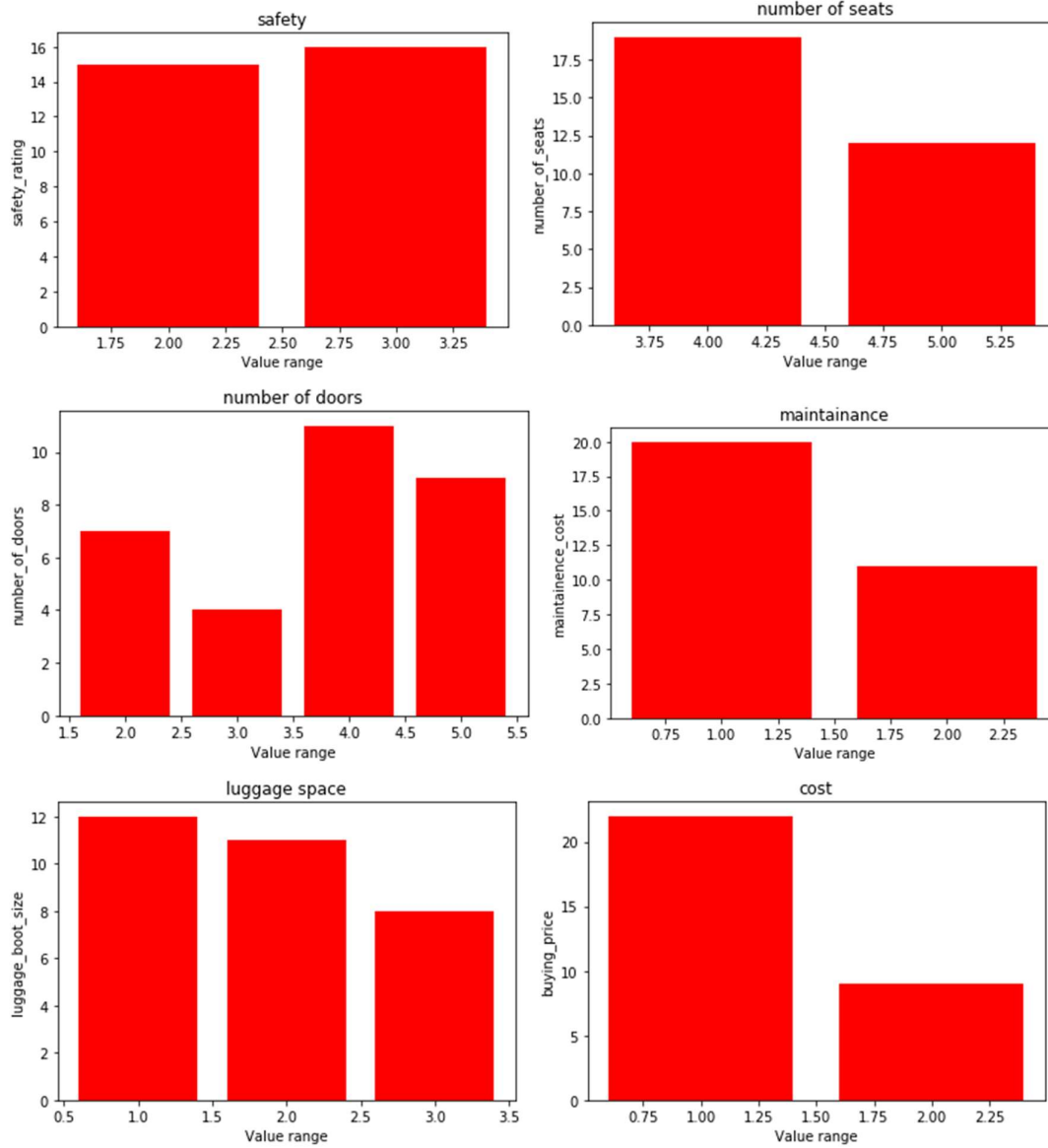
1. POPULARITY-1



POPULARITY -2



POPULARITY – 3



POPULARITY -4

