Car Popularity Prediction

DATA DESCRIPTION

Our train dataset contained 1628 rows and 7 columns, and test data contained 100 rows and 6 columns.

Data to predict – popularity (of cars)

Predictors – buying_price

Maintainance_cost

number_of_doors

number_of_seats

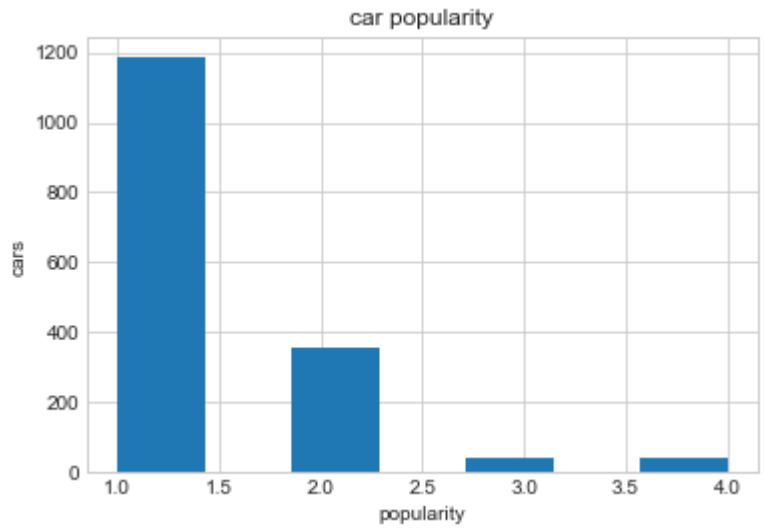
luggage_boot_size

safety_rating

ANALYSIS OF DATA

Checking data distribution –

Popularity –



So we see that maximum of our input data is of the popularity range 1 , and very few of the data are of popularity 3 or 4.

Looking at the stats of popularity also suggests the same –

```
In [101]: train.popularity.describe()
Out[101]:
count    1628.000000
mean        1.348280
std         0.654766
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max         4.000000
Name: popularity, dtype: float64
```

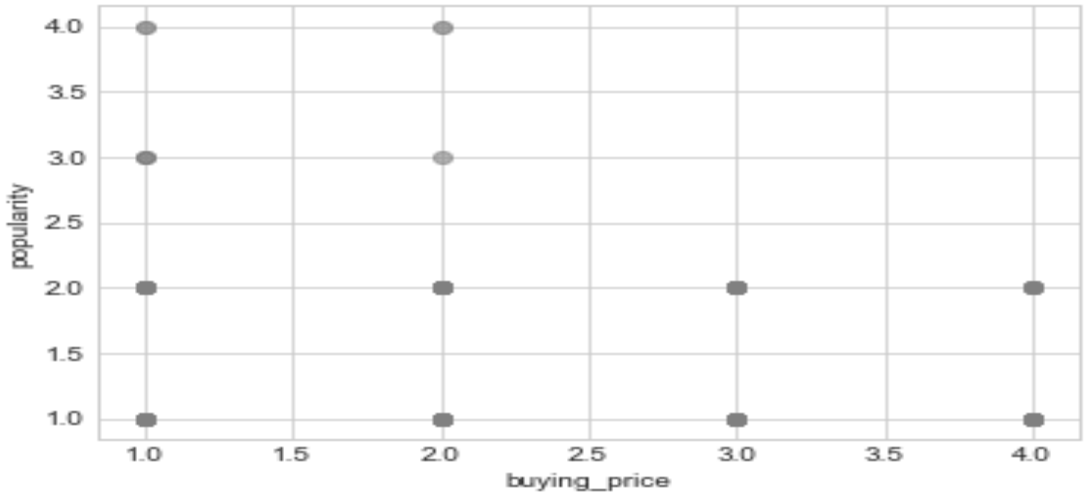
Looking at other data –

```
       buying_price  maintainence_cost  number_of_doors  number_of_seats  \
count   1628.000000        1628.000000      1628.000000      1628.000000
mean       2.532555           2.528256         3.493857         3.633292
std        1.109626           1.116920         1.120557         1.257815
min        1.000000           1.000000         2.000000         2.000000
25%        2.000000           2.000000         2.000000         2.000000
50%        3.000000           3.000000         3.000000         4.000000
75%        4.000000           4.000000         4.250000         5.000000
max        4.000000           4.000000         5.000000         5.000000

       luggage_boot_size  safety_rating   popularity
count        1628.000000    1628.000000  1628.000000
mean            1.987101       1.977887     1.348280
std             0.816520       0.819704     0.654766
min             1.000000       1.000000     1.000000
25%             1.000000       1.000000     1.000000
50%             2.000000       2.000000     1.000000
75%             3.000000       3.000000     2.000000
max             3.000000       3.000000     4.000000
```

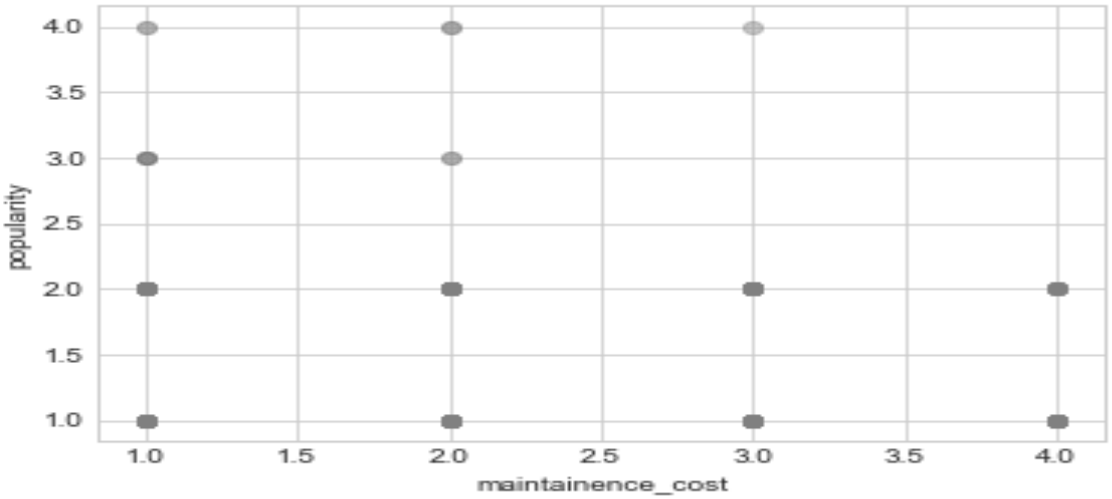
Checking relationship with popularity –

Buying_price



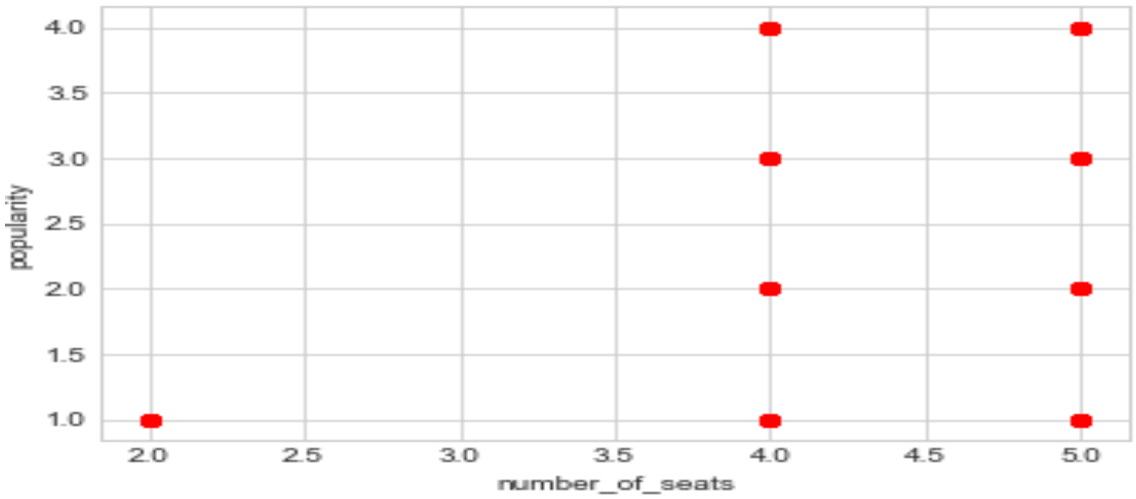
We see that the cars with higher price don't have that much popularity.

Maintainance_cost



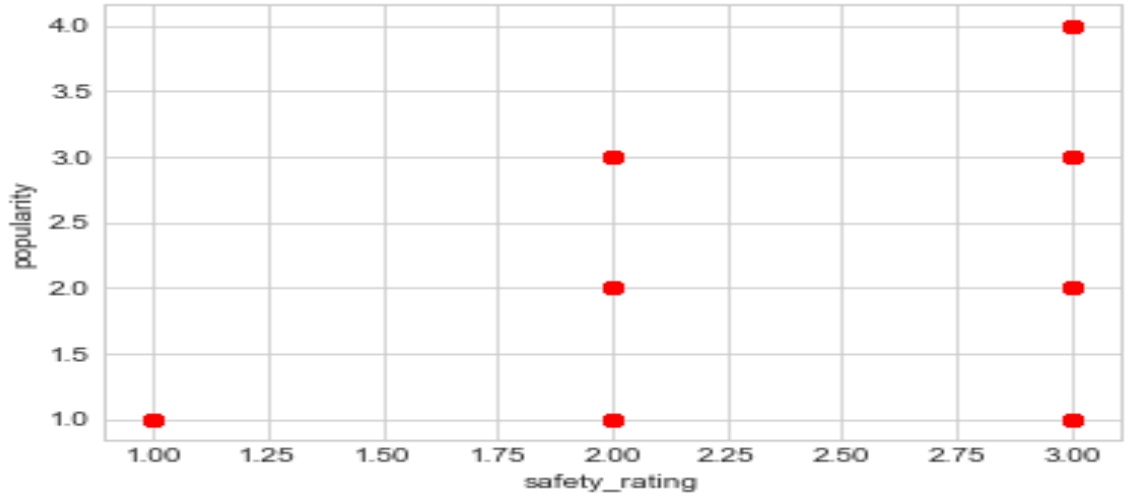
As expected maintainance cost also has the same kind of relationship with popularity.

Number_of_seats



As the number of seats can be 2,4 or 5 so we see that we don't have any erroneous data in here, and we see that the less the number of gates the less the popularity.

Safety_rating



As expected the less safe the car is, the less the popularity it has.

Number_of_doors and the **luggage_boot_size** have almost no effect with there values.

LOOKING INTO DATA

Checking for null values -

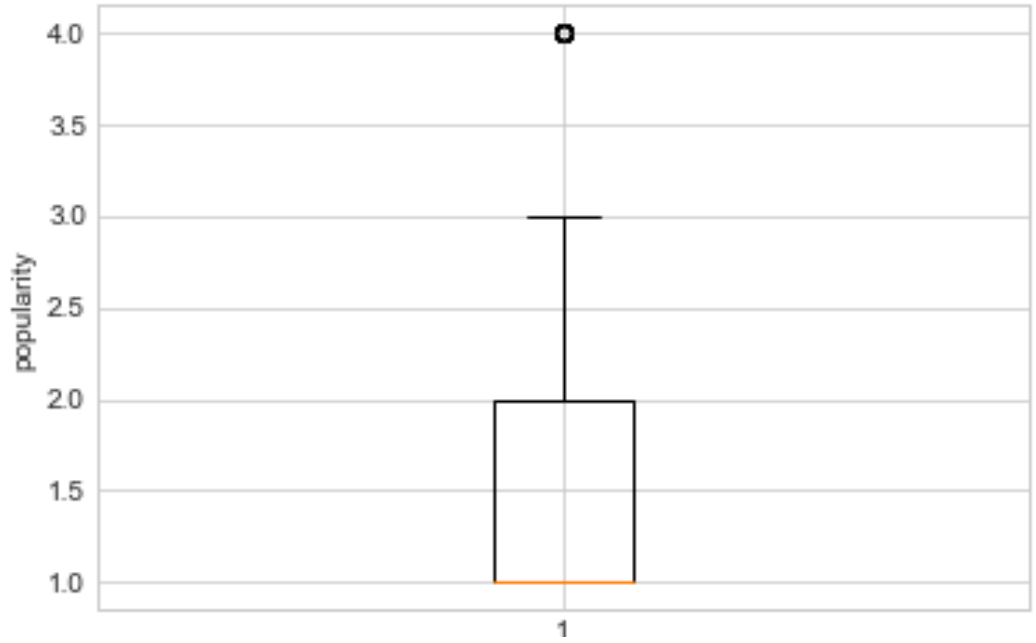
As we see into data , we check for null values – checking at the whole data we found no null values in any column. So no missing values in this train set.

Checking for errors –

As we see that the values can't be out of the range that is described for each and every figure, and for graph we noticed that dataset set does not contain any erroneous value.

Checking for outliers –

As we see the boxplot of the popularity, we figure out that nothing can be judged as outlier.



We see that the popularity 4 may seem like an outlier but it is not, rather it is a data of great importance with less number of occurences.

DATA PREPROCESSING

As we have done the analysis of features so now we will be focusing on other parts.

Scaling –

Here we see that each and every feature is in the range of 1-5 so we don't need to do any kind of scaling.

Encoding Features –

We see that in the dataset the features are in numeric form and have precedence amongst them(even when they are categorical) so no need of using one hot encoding.

MODEL DECISION

This problem seems to be a classification problem as we have 4 classes, this seems to be a **multi-class classification**.

Here we have got 6 features on the basis of which we have to select which which of the class out of the 4 options, this test data should fall into.

Looking at the data we can figure out that there would be a clear margin of separation between the points and here we have less number of training data so I chose **SVM(Support Vector Machine)** as the classifier for this dataset.

SVM classifier first plots all the support vector(points) and then creates a well defined plane to separate points of different classes. But as linear function to solve this problem , can sometimes not be that useful , so I used **rbf kernel** of the support vector machine to do this work for me.

I used scikit learn library of python to do this work, the features I selected to check which would give me the best results was C(penalty parameter) = 300 as

higher this value more away points will also be taken into consideration , and gamma = 1(default) as I didn't want my model to overfit.

Language Used

Python

Libraries used

matplotlib : https://matplotlib.org/api/pyplot_api.html

pandas : https://pandas.pydata.org/

sklearn SVM : http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html