Trip advisor hotel reviews

Machine learning model to predict rating score for hotel service

Agenda

1. Exploratory data analysis
2. Data preparation
3. Model selection
4. Final thoughts

Exploratory data analysis

Exploratory data analysis

- This data set contains 20491 rows and 2 columns.
- The data set has no null values.
- The data set has no duplicated values.

Rating score has the following statistics:

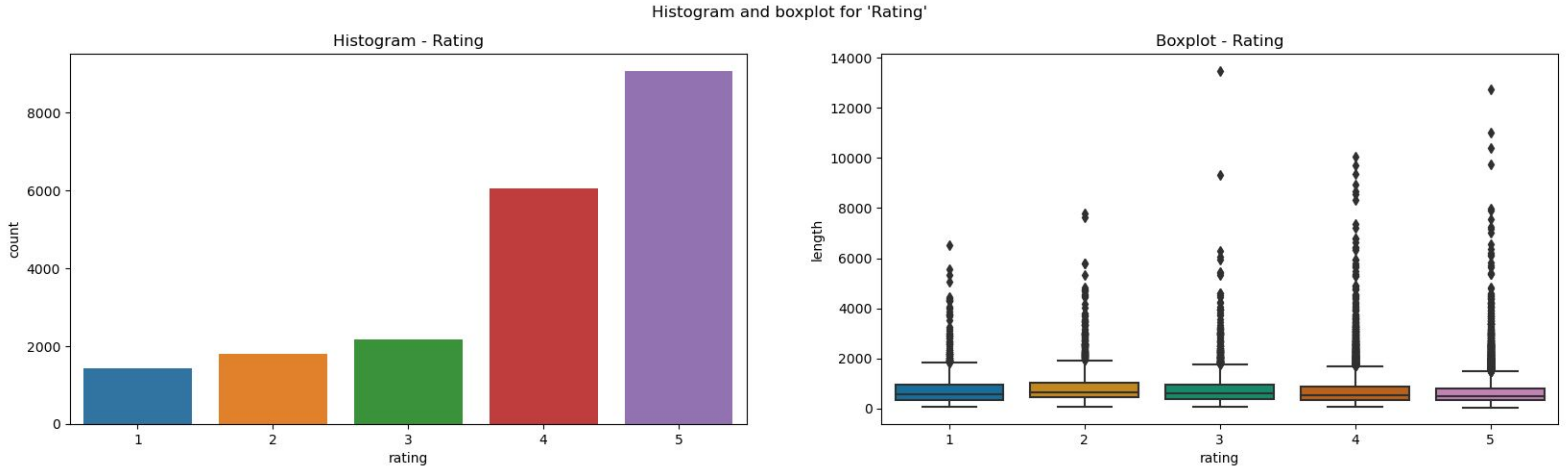
mean	3.95
std	1.23
min	1.00
25%	3.00
50%	4.00
75%	5.00
max	5.00

Rating has the following occurence

rating	percentage	value counts	cummulative
5	0.441853	9054	0.441853
4	0.294715	6039	0.736567
3	0.106583	2184	0.843151
2	0.087502	1793	0.930652
1	0.069348	1421	1.000000

- Which shows that 74% of the reviews has good rating scores (4 and 5).
- Target variable (score) is imbalanced.

Exploratory data analysis



Exploratory data analysis

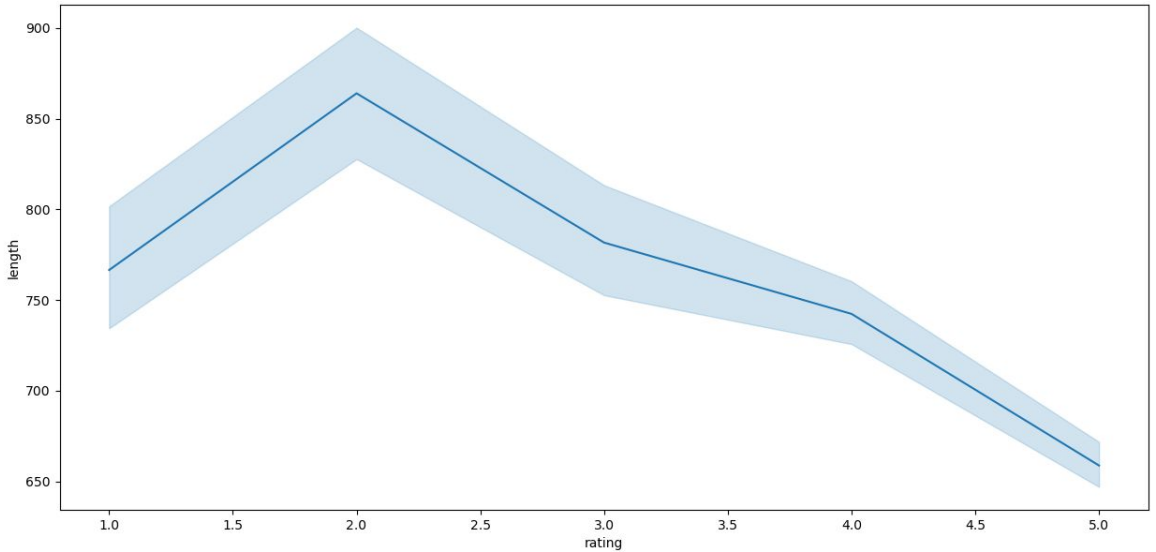
The length words in review sentences has the following statistics:

mean	722.0
std	689.0
min	41.0
25%	336.0
50%	534.0
75%	856.0
max	13498.0

In average, reviews has 722 words.

Exploratory data analysis

By checking the relationship between rating and length of sentences, is possible to conclude that there exists a negative correlation between length and rating. This is, given less words in a review better the rating.



Exploratory data analysis

In general, the top 20 of most used words in reviews are:

WORD	FREQUENCY
hotel	8447
room	8112
time	5522
resort	5487
day	5421
night	4285
nice	3836
got	3711
people	3671
n't	3581
restaurant	3567
beach	3547
good	3283
really	3252
problem	3132
place	3129
lot	3111
went	2934
thing	2916
food	2836

Data preparation

Data preparation

Cleaning data reduces the length of sentences in reviews. (Delete special characters)

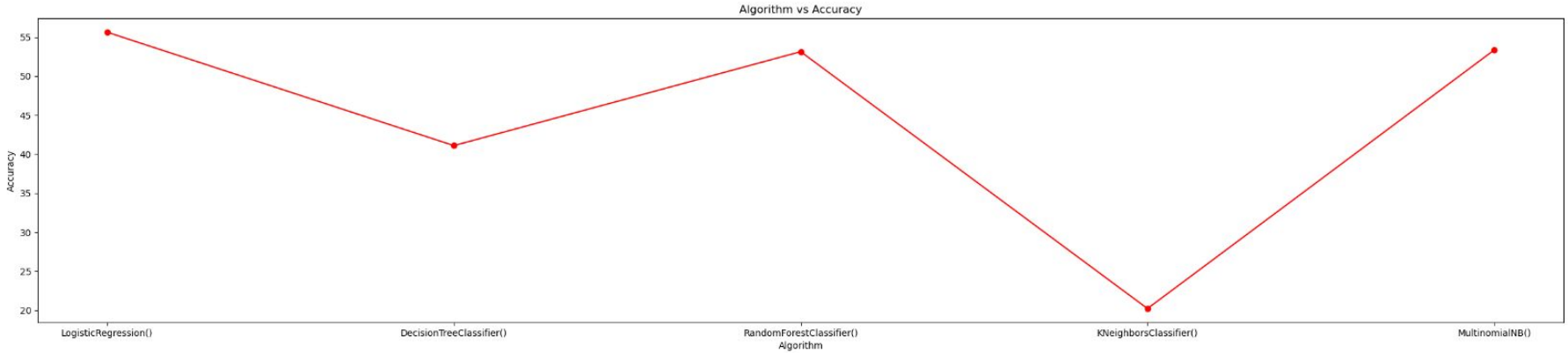
	rating	length	clean_length
count	20491.00	20491.0	20491.00
mean	3.95	721.9	597.84
std	1.23	689.1	564.26
min	1.00	41.0	31.00
25%	3.00	336.0	282.00
50%	4.00	534.0	444.00
75%	5.00	856.0	709.00
max	5.00	13498.0	11189.00

Model selection

Model selection

Compared classification algorithms:

```
LR = LogisticRegression()
DTR = DecisionTreeClassifier()
RFR = RandomForestClassifier()
KNR = KNeighborsClassifier()
NB = MultinomialNB()
```

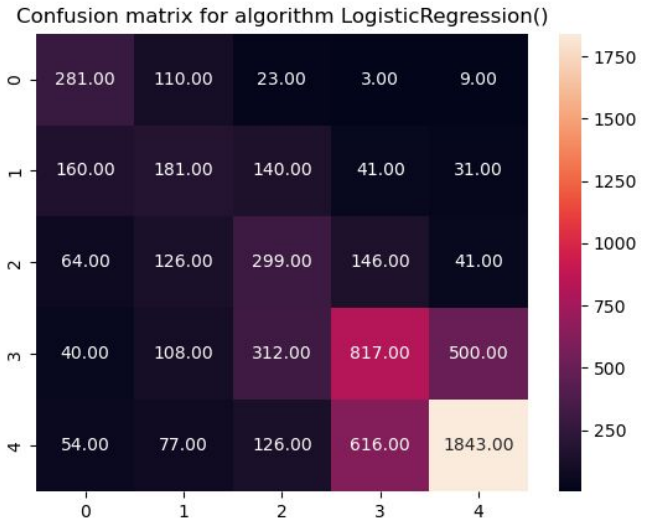


Model selection

```
************************************************************
************************************************************
Accuracy of LogisticRegression(): 0.55644111906311
              precision    recall  f1-score   support

           1       0.66      0.47      0.55       599
           2       0.33      0.30      0.31       602
           3       0.44      0.33      0.38       900
           4       0.46      0.50      0.48      1623
           5       0.68      0.76      0.72      2424

    accuracy                           0.56      6148
   macro avg       0.51      0.47      0.49      6148
weighted avg       0.55      0.56      0.55      6148
```



Model selection

Due that Logistic regression model has the better performance, it is going to be the model to be optimized. To do it, we used grid search technique.

With the results obtained from grid search, the best achieved performance has an accuracy of 55%.

```
Best: 0.626065 using {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.625981 (0.008078) with: {'C': 1000, 'penalty': 'l2', 'solver': 'newton-cg'}
0.625697 (0.007943) with: {'C': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
0.618208 (0.007552) with: {'C': 1000, 'penalty': 'l2', 'solver': 'liblinear'}
0.626044 (0.008022) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
0.626065 (0.007807) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.618218 (0.007497) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
0.625707 (0.007873) with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
0.625897 (0.008159) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.617671 (0.007853) with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.625360 (0.006799) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.625350 (0.006703) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.616062 (0.********************************************************* 'liblinear'}
0.609225 (0.********************************************************* 'newton-cg'}
0.609288 (0. Accuracy of LR_opt: 0.5507482108002603                    'lbfgs'}
0.601357 (0.              precision    recall  f1-score   support     'liblinear'}
0.567193 (0.                                                        : 'newton-cg'}
0.567151 (0.           1       0.64      0.48      0.55       564   : 'lbfgs'}
0.560597 (0.           2       0.35      0.31      0.32       623   : 'liblinear'}
               3       0.42      0.31      0.36       928
               4       0.45      0.49      0.47      1612
               5       0.68      0.76      0.72      2421

        accuracy                           0.55      6148
       macro avg       0.51      0.47      0.48      6148
    weighted avg       0.54      0.55      0.54      6148
```

Final thoughts

Final thoughts

- To improve performance, a suggestion is to reduce the classification classes. This is, not to predict the specific rating score but, for example, to predict the probability to recommend the hotel or to predict if the experience was good, bad or neutral.
- Another strategy to improve performance, is to use deep learning algorithms. For example, to use LTSM. However, this kind of algorithms needs high power of computation.

Regards!