Customer review rating prediction

By Simantini Patil

Machine Learning:

In this section I applied Machine learning models to the dataset to predict customer review rating score.

Hyperparameter tuning:

After feature creation, it was observed that different features in the data set had values in different ranges. For example, in 'delayed' column the value ranged from (-189.0, 146.0) and for product_value the value ranged from 2 - 13444.That means some column were more weighted compared to other.

Differences in variable ranges could potentially affect negatively to the performance of an algorithm so I used scikit-learn's Normalizer to normalize the data.

After normalizing the data between [0,1] I took the log values of the data. This introduced a few -inf and NaN values which I later on replaced with 0.

The data was still widely distributed so I set a lower and upper bound on feature values using clip function.

Also, the categorical columns were encoded using Labelencoder.

Dataset was then split into test and train datasets which was later using in modeling.

Modeling:

To predict the review rating score, I had to select a classifier that performs the best, given the features. I trained the model using three classifiers: Logistic Regression, Random Forest and Gradient Boosting, and chose the one with the best accuracy.

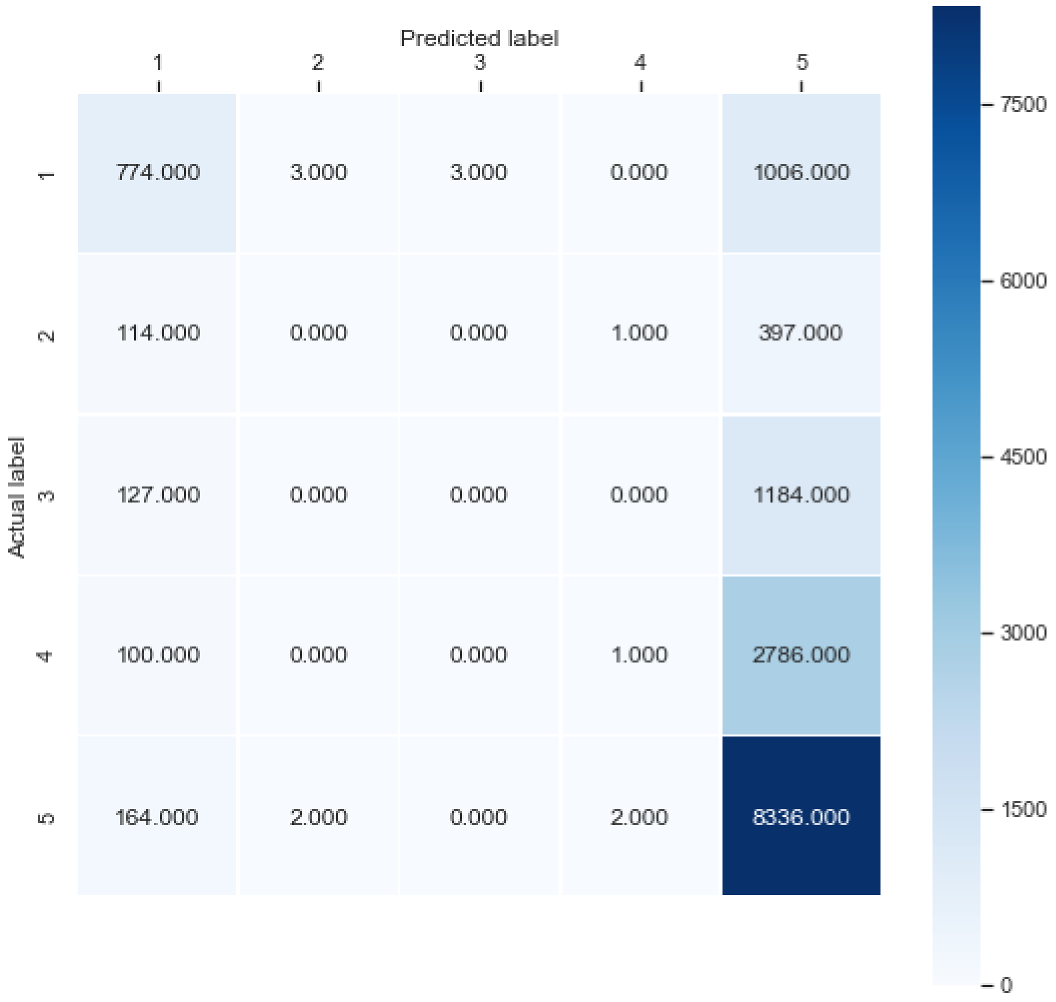
Accuracy score per model:
Random Forest Classifier : 0.5526
Logistic Regression: 0.59326
Gradient Boosting Classifier: 0.6074

GradientBoostingClassifier performed best with 60% accuracy so I chose Gradient Boosting Classifier as my model of choice.

To improve the model accuracy, I tried hyperparameter tuning using Grid Search Cross Validation. Fortunately, like always, scikit-learn has the tools available to us that reduces the amount of code to a bare minimum.

The score achieved with GSCV was 0.6088 which was not a significant improvement.

Later to summarize the performance of the algorithm, I calculated a confusion matrix and plotted it as a heatmap as below.



Review score 5 was classified most accurately than other review scores.

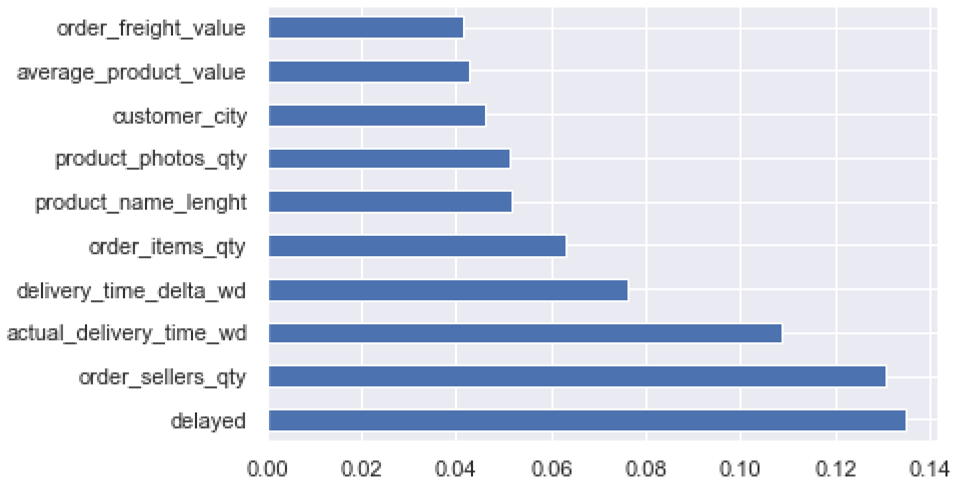
Later I calculated precision and recall:

```
             precision    recall  f1-score   support

          1       0.61      0.43      0.51      1786
          2       0.00      0.00      0.00       512
          3       0.00      0.00      0.00      1311
          4       0.25      0.00      0.00      2887
          5       0.61      0.98      0.75      8504

avg / total       0.46      0.61      0.49     15000
```

To describe the features and their relative importance to a model, I used scikit-learn's feature_importances_ attribute and plotted the results on a graph as below



From above we can say 'delayed' is the most important feature.

Conclusion:

The comparison between the three models in score classification is interesting. Although Random Forest is a complex model and it should be able of capturing more complex patterns, it does a really poor job in comparison with more simple and linear approaches such as Logistic Regression (55% vs 59% respectively in test set).

GradientBoostingClassifier performed best with 60% accuracy.

Another interesting point of the score classification can be seen in the confusion matrix. Precision and recall is higher in both extreme scores: 1 and 5. The possible explanation to this could be that the customers usually are satisfied (score 5) or not satisfied(1) so the number of reviews with score 1 and 5 are comparatively more than 2,3 and 4.