Deliverable 2

Problem Statement

The problem at hand is to make a classifier that takes as an input the description of a wine and tries to output a price category. The final goal is to have a user input the description of a wine and the software output a price category for this wine. The categories are as follow: $0-20, $20-40, $40-80, $80-150, $150-300, $300-1000, >$1000.

Data Preprocessing

For this problem, I use Kaggle 130K wine review dataset (https://www.kaggle.com/zynicide/wine-reviews). This dataset contains 130 000 review of wines across the world, along with the price, grape variety, country, region, and name of reviewer. First, I created a dataset containing only the "description" and "price" columns. Then, I removed rows where the price was not a number because the model could not use these entry points. I removed all punctuation marks, because they would generate empty tokens later on, when lemmatizing. I removed stopwords because they don't contain useful information about the wine. As a last preprocessing step, I lemmatized the words to keep only the root, as it can help the performance of models.

Machine Learning Model

As of now, I have tried three models: A Naïve Bayes classifier, a Random Forest Classifier and a Logistic Regression Classifier, all from the sklearn library. I used 90% of the set for training and 10% for test, but I have gotten similar accuracy of results using only 70% for training. I used the accuracy_score method from sklearn to evaluate the performance of my models. Although I will discuss preliminary results in the next section, I suspect that the RFC is overfitting the data, because the train set accuracy is of 99% (which is quite high, compared to the two others).

My biggest challenge in using these models was the preprocessing part, on which I spent most of my time before submitting this deliverable. Once the data was lemmatized, it took me little time to try at least the 3 models mentioned above. Another issue I faced was that, at first, I was getting an accuracy of less than 50%, meaning my model was so bad that it guessed more wrong answers than right answers. I thought that this might be because I had too many categories. Before I reduce the number of categories, I will investigate which categories are confusing for the model. This is discussed later with the confusion matrix.

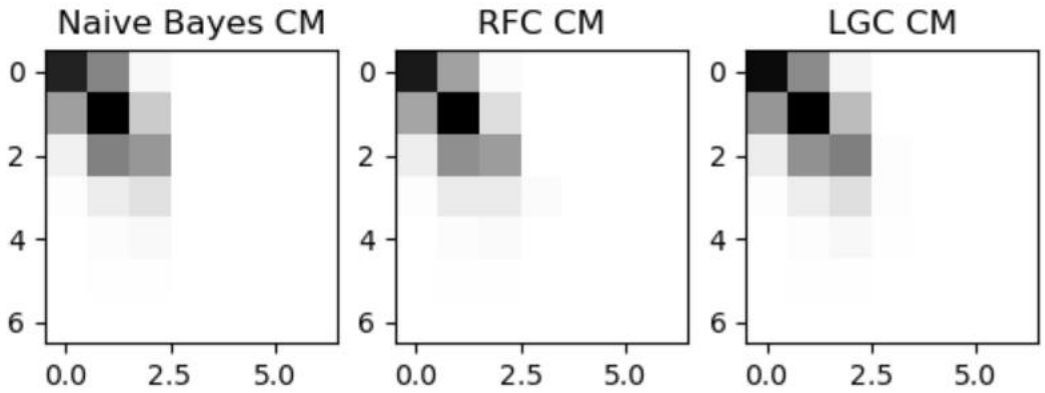
As a last note on the models, they have the advantage that they are quick to train. Once the data is preprocessed, it is easy to split the data into two brand new train and test sets and train the models again on the test set.

Preliminary results

Below is a screenshot of the accuracy for the test and train sets with the 3 models I implemented:

```
The test accuracy of the Naive Bayes model is 54.513142668209625%
The train set accuracy of the Naive Bayes model is 56.89815112466361%
The test accuracy of the RFC model is 59.0510828236072%
The train set accuracy of the RFC model is 99.9889783884567%
The test accuracy of the Logistic Regression model is 55.769548685733184%
The train set accuracy of the Logistic Regression model is 60.70795484813137
```

The accuracy of the models could overall be improved. I used accuracy score because I wanted a quick way to tell how my models were performing, and I have also used a confusion matrix to have an idea of which categories were problematic for the model. Below are the confusion matrices (graphs and numbers) for the Naïve Bayes, the RFC and the Logistic Regression:




```
Naive Bayes confusion matrix:
[[2504 1385   85    0    0    0    0]
 [1098 2905  591    0    0    0    0]
 [ 161 1431 1185    1    0    0    0]
 [  33  214  344    1    0    0    0]
 [   6   44   70    0    0    0    0]
 [   1   16   20    0    0    0    0]
 [   0    3    0    0    0    0    0]]
RFC confusion matrix:
[[2778 1137   59    0    0    0    0]
 [1096 3094  404    0    0    0    0]
 [ 210 1364 1204    0    0    0    0]
 [  36  245  253   58    0    0    0]
 [   6   45   61    0    8    0    0]
 [   3   19   13    0    0    2    0]
 [   0    3    0    0    0    0    0]]
LGC confusion matrix:
[[2611 1242  117    4    0    0    0]
 [1130 2732  724    8    0    0    0]
 [ 201 1178 1375   24    0    0    0]
 [  26  186  351   29    0    0    0]
 [   5   25   78   11    0    1    0]
 [   1   13   16    7    0    0    0]
 [   0    2    1    0    0    0    0]]
```

One thing we can see from the numbers in the confusion matrices is that the model has trouble differentiating wines that are in the $0-20 and the $20-40 range. Even, the 40-80 range is problematic. Knowing this will help guide my efforts in the weeks to come.

Next Steps

With what we have seen so far, I have a lot of things to improve on the model. First, I will add bias to the models while training. This might improve the accuracy, especially on the RFC model, which has a very high accuracy on the training set. Instead of TF-IDF, I will try using word embedding with a Word2Vec model and test if the results are different.

I will also try Support Vector Machines and Neural networks to see if they offer better performance right away than the models I have used so far. I would be really interested in seeing my model perform on more granular classes, so my main focus will be to search for what distinguishes the classes in order to help the model perform better.