Predicting wine quality using Machine Learning

I) Introduction

From its numerous applications in the real world, machine learning can also be used as a business strategy to improve the overall quality of products. When taking into account the past trials and errors in a certain field, we can try to predict what the customer will most likely appreciate.

This is especially true for a gastronomic beverage such as wine, as it is highly complex and a great number of parameters weigh in on quality fluctuations. Basing ourselves on data collected in 2009 in Portugal, we will use machine learning methods to try and make accurate predictions on which factors make a wine good. This problem has implications not only for consumers, but also for wine producers around the world which have not established their brand yet on the market.

The problem formulation part will explain the machine learning (ML) problem in detail, which will then be analysed in the methods sections, using ML methods to exploit the data. The results section will elaborate and contrast the results found, which will then be summarised in the conclusion.

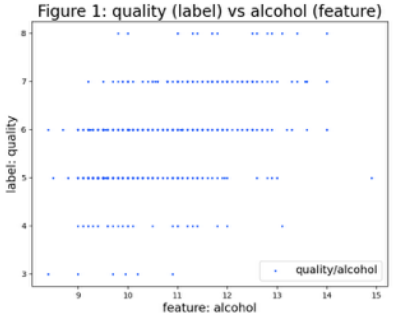
II) Problem Formulation

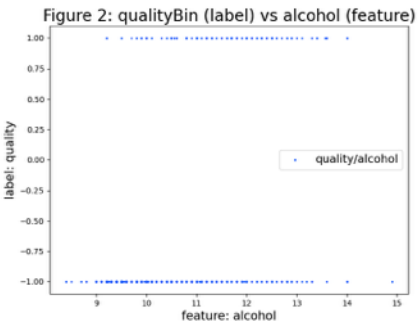
Problem: given the data in the dataset, how accurately can we predict the quality of a wine given multiple elements that make up the beverage?

The dataset was provided by the Minho University of Portugal (and was obtained on Kaggle [1]). It provides an accurate documentation of the quality of the Vinho Verde wine type evaluated in 2009. Indeed, it is made up of 1,143 data points with no missing features or labels. Included are various biological components of wines: acidity, sugar levels, pH, sulfates, chlorides or even alcohol level.

Our goal is to predict whether a wine with certain characteristics can be considered “good” or not, which leads us to believe a classification method should be used to analyse the data. However, since the wines’ quality is currently ranked from 0 to 10 (with minimum = 3 and maximum = 8) we need to add a column to the dataset which classifies quality in a binary way, either “good” or “not good”. To do so, we use a simple classifier function which assigns a value to each wine in this new column (called qualityBin), with wines having about 7 quality rating being assigned a 1, and -1 otherwise. The reason we use these values will be explained in the methods section. As for why we use 7 for gauging quality, it is an arbitrary value based on the quality range (3 to 8).

Since this problem will make use of multiple features as training data, it is difficult to represent the data in a graphical environment (which would require the use of hyperplane space). We can still however, visualise in Figure 1 how the label will be classified according to features (in this case alcohol). In Figure 2 we see how binary classification affects the graphical representation.





III) Methods

As mentioned previously, we will make use of a classification model as a machine learning method [2]. We, however, have a choice to make between using Logistic Regression or Support Vector Classification (SVC). To choose either one of them, we can try to apply both methods to the problem and see which one offers the best accuracy!

The dataset offers a range of 1,143 data points, none of which are missing. Most features span on a range from 0 to 100, as it symbolises a percentage of the wine's composition. It is important to note that the dataset is not very extensive, in that 1,143 data points will not allow us to yield very accurate results. Indeed, a smaller dataset is more sensible to outliers. We will explain later how we dealt with this issue. Furthermore, the column for the label (quality in binary form) is now constituted of 1s and -1s, with 1 representing a "good" wine and -1 a "not good" wine. The reason we use -1 and 1 here is not only for convention's sake, but also because the loss method for the SVC model (hinge loss) operates on labels which takes values -1 or 1, while the loss method for logistic regression (logistic loss) takes arbitrary values as long as they differ (0 and 1 for example).

There were 12 different features to choose from [3], however we decided to pick the ones which showed most correlation to the wine's quality, that it the ones that most impact taste (to avoid overfitting). Therefore, we decided to use as features: fixed acidity, citric acid, sugar, chlorides, density and pH. Some of these features affect the wine's acidity (fixed acidity, citric acid, pH), while others affect taste.

i) SVC

Before discussing whether this is the appropriate ML method to use in this example, it is crucial to understand how it works to understand why it could be useful.

When some data is classified (good/bad, healthy/sick,…), the reason to use machine learning is to determine the ideal position to place the delimiter between the categories (in other words where do we put the | in "good | bad"). In a one dimensional dataset (line), this would equate to putting a point on the line to separate the categories. In a two dimensional dataset (plane), we would use an affine line. In a three dimensional dataset (space), we would use a plane. From there on we use "hyperplanes", which we can not represent graphically.

For the SVC model [6] we will use the hinge loss. Not only is this function importable from sklearn [7] but it also makes sense to use this loss function. Indeed, we visualising the SVC in a two dimensional space, it is a line that separates both categories of the data. The hinge loss is then measured by applying a margin to both sides of this line. If an outlier falls in this margin, depending on it's proximity to the hypothesis, it will result in a loss proportional to this distance, as the loss function is linearly decreasing with coefficient -1.

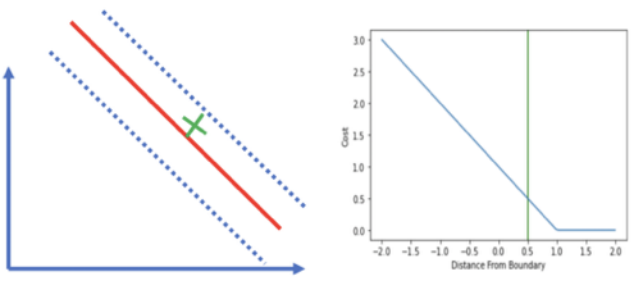


Figure 3: Hinge loss representation [4]

ii) Logistic Regression

Similar to SVC, Logistic Regression [8] actually uses the same hypothesis space (for SVC with a linear kernel). The representation of a LR model is in the form of an "S" shaped curve which here spans from -1 to 1.

What sets SVC and Logistic Regression apart are the loss functions we use to evaluate the quality of the hypotheses. Logistic loss [9] is represented by a curve (rather than a line for hinge loss).

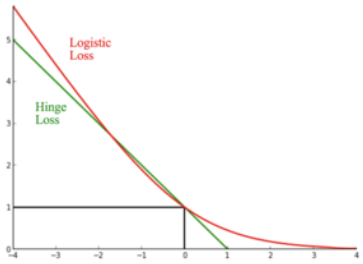


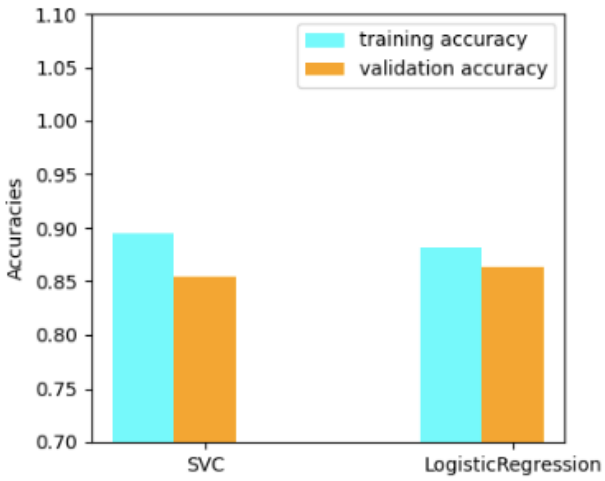
Figure 4: Logistic Loss vs Hinge loss [5]

iii) Decision

To know which method fits the model most, we need to specify how we analysed the data (all the code can be found as an appendix to this report).

When using function train_test_split from sklearn's model_selection, we are to define a test_size (0 to 1), which represents the percentage of the dataset we wish to use as training data, and which we use to validate the predictions. Since our dataset contains of 1,000 data points, we estimated it was not necessary to use k-fold cross validation to estimate the optimal amount of data to use to avoid under or overfitting. The test_size used was 0.8, as when trying with various test sizes, it yielded the best results in terms of high accuracy and low loss. Therefore, 80% of the set would be used for training, and 20% for verifying the hypotheses.

The final decision on which model to use was made after comparing the models' training and validation accuracies (similarly to assignment A4). On the histogram we see that the difference between the training accuracy and validation accuracy is lowest for the Logistic Regression model. Indeed, even if the training accuracy is lower than for the SVC model, The Logistic Regression seems to be more consistent.



IV) Results

As mentioned above, the final decision for the ML method used was based on the observation that the difference between training and validation accuracy was lowest when using Logistic Regression.

We may also want to look into the validation and training error of both these methods, to estimate their effectiveness.

For the SVC model, the log loss is most appropriate to estimate loss. As for Logistic Regression, hinge loss is the most appropriate. Comparing both the training and validation loss will allow us to further trust our model decision. It is important, however, to standardise the data when using Logistic Regression, as otherwise we will overfit it and it will lead to incoherent loss values. To do so, we will use L2 Regularisation, making use of sklearn's StandardScaler() method. After regularisation, we get:

Figure 6: SVC errors vs LogisticRegression errors

Figure 6 shows that the difference between training and validation loss is greater for the SVC model. This confirms ours hypothesis that the Logistic Regression model is more appropriate in this case.

The test set is essential in making a final decision on the choice of ML method. It is composed of data points neither in the training nor validation set. The choice of size is done similarly to the one for splitting the data into train / validation sets, using the train_test_split method from sklearn, and we found the optimal amount to be 0.2, or 20%.

Similarly to the previous section, we can evaluate the test error to estimate how good the chosen method is at dealing with new data. To do so, we apply both hinge loss and log loss to the new set.

Figure 7: Test error for both models

While we see that the LogisticRegression loss is superior for the test set, since the difference is so slight we assume that this still validates our model decision.

V) Conclusion

Estimating wine quality is thus one problem applicable to machine learning, and can be either considered through regression or classification models. Here, by placing a standard at 7 for quality, we could successfully analyse the data we found by classification. The model can be criticised, however, in that it does not contain many data points, and also contains outliers.

Furthermore, while the accuracy of our model can be considered high, so is the loss. This can be a factor of overfitting our model.

In the future, collecting more training data would be beneficial to improve the model. One can also speculate that a Random Forest Classifier model could also be appropriate for this problem. More biological research on how we gauge quality can also help us determine which features to use for training the data.

VI) References

[1] Wine quality dataset:
https://www.kaggle.com/yasserh/wine-quality-dataset

[2] Information about the machine learning models and loss functions (course book):
https://github.com/alexjungaalto/MachineLearningTheBasics/blob/master/MLBasicsBook.pdf

[3] Using multiple features:
https://datascience.stackexchange.com/questions/39034/how-to-train-ml-model-with-multiple-variables

[4] Hinge loss imagery:
https://programmathically.com/understanding-hinge-loss-and-the-svm-cost-function/

[5] Logistic loss imagery:
https://jhui.github.io/2017/01/15/Machine-learning-regression-and-logistic-regression/

[6] Sklearn's SVC method:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[7] Sklearn's Hinge loss:
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hinge_loss.html

[8] Sklearn's Logistic Regression method:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[9] Sklearn's Logistic loss:
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

[10] Help for data visualisation (assignment A4):
https://jupyter.cs.aalto.fi/user/guidikc1/notebooks/notebooks/ml2022/Assignment4/a4.ipynb

[11] Classifier function used to create qualityBin column:
https://stackoverflow.com/questions/15281320/classifying-a-series-to-a-new-column-in-pandas

VII) Appendix

[12] https://github.com/ChristianGuidikov/CS-C3240-ML-Project/blob/main/wine_rating.py