Wine quality classification using logistic regression: A comparison of ordinal and non-ordinal methods

Musong Chen
School of Computing and Information Technology
University of Wollongong
CSCI933, SN:7424917, mc844@uowmail.edu.au

April 28, 2023

Abstract

In this report, we build a multinomial logistic regression model and an ordinal logistic regression model to classify wine quality. We evaluate the performance of both models using a subset of 5 data that were randomly selected before training from the wine quality dataset, our experiments indicate that the ordinal logistic regression model provides a more accurate classification of wine quality.

1 Introduction

Regression analysis is an important statistical tool for understanding and predicting the relationship between two or more variables. In this report, we will focus on the application of regression analysis to the Wine Quality Dataset.

First, we will provide a brief overview of the dataset structure and the steps taken in data preprocessing to ensure its suitability for regression analysis. Next, we will discuss the theory and properties of regression, including the L1, L2, and Elastic-net penalties, as well as logistic and ordinal logistic regression. In the experiments section, we will describe the data split and the experimental setup used in the study. We will then present the results of two experiments conducted on the Wine Quality Dataset.Finally, we will discuss the results of the experiments, draw conclusions, and offer suggestions for future research.

2 Wine quality dataset

2.1 Dataset structure

The dataset to be used in this report is the "Wine quality" dataset published by Cortez, Cerdeira, Almeida, Matos, and Reis (2009) for research purposes Géron (2019). This dataset contains 11 wine features, including fixed acidity, chlorides, and alcohol content etc, are all numerical data. The wine quality is classified as ranging from 0 to 10, which the larger number the better. On the other hand, the dataset contains 1599 samples which are desirable for training a simple model.

2.2 Data preprocessing

After checking the completeness of the data, we did not find any missing data, however, the data seems is slightly imbalanced since the ranges come in from decimal numbers to three-digit numbers which could cause inefficacy and bad result for model training. Therefore, we took care of this aspect by using the feature scaling method to normalise all the feature data which makes them in a similar scope from -3 to 3.

3 Theory and properties of regression

L1, L2, and Elastic-net penalties are regularization techniques applied to regression models to prevent overfitting and improve model generalization. These techniques add a penalty term to the objective function, which helps in reducing the magnitude of the coefficients.

3.1 L1, L2 and Elastic-net penalties

The **L1 penalty**, also known as **Lasso regularization**, adds the absolute values of the coefficients multiplied by a regularization parameter (λ) to the loss function. The L1 penalty encourages sparsity in the model, causing some of the coefficients to become exactly zero. This results in

feature selection and a simpler model. The equation for the L1 penalty is:

$$L_1(\beta) = \lambda \sum_{j=1}^{n} |\beta_j|$$

where β represents the coefficients, and j is the index of the coefficient.

The **L2 penalty**, also known as **Ridge regularization**, adds the squared values of the coefficients multiplied by a regularization parameter (λ) to the loss function. The L2 penalty reduces the magnitude of the coefficients without making them exactly zero. The equation for the L2 penalty is:

$$L_2(\beta) = \lambda \sum_{j=1}^{n} \beta_j^2$$

where β represents the coefficients, and j is the index of the coefficient.

Elastic-net regularization is a combination of L1 and L2 penalties. It adds a linear combination of the L1 and L2 penalties to the loss function. This method combines the advantages of both Lasso and Ridge regularization. Elastic-net has two regularization parameters: α and λ, where α is used to control the balance between L1 and L2 penalties. The equation for the Elastic-net penalty is:

$$ElasticNet(\beta) = \lambda \left[(1-\alpha) \sum_{j=1}^{n} \beta_j^2 + \alpha \sum_{j=1}^{n} |\beta_j| \right]$$

where β represents the coefficients, j is the index of the coefficient, λ is the regularization parameter, and α is the balance parameter between L1 and L2 penalties.

In summary, L1, L2, and Elastic-net penalties are regularization techniques that help to prevent overfitting and improve the generalization of regression models by adding penalty terms to the loss function based on the coefficients' magnitude.

3.2 Logistic regression

Logistic regression is a statistical method used to model the relationship between a binary dependent variable (outcome) and one or more independent variables (predictors). It estimates the probability of an outcome based on the input features by applying the logistic function (also known as the sigmoid function) to a linear combination of the predictor variables. The logistic function is given by:

$$g(x) = \frac{1}{1 + e^{-z}}$$

where z is a linear combination of predictor variables ($z = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$), g(x) is the probability of the positive class, and e is the exponential function.

The logistic regression model is fitted by maximizing the log-likelihood function, which can be represented as:

$$LL(\beta) = \sum_{i=1}^{n} [y_i \log(g(x_i)) + (1 - y_i) \log(1 - g(x_i))]$$

where $LL(\beta)$ is the log-likelihood, y_i is the binary outcome for observation i, x_i is the predictor variables for observation i, and $g(x_i)$ is the predicted probability for observation i.

3.3 Ordinal logistic regression

Ordinal logistic regression is an extension of logistic regression for ordinal categorical dependent variables. It models the relationship between an ordinal response variable and a set of predictor variables. In ordinal logistic regression, the assumption is that there is an underlying continuous latent variable (not observed) that determines the observed ordinal variable. The ordinal logistic regression model uses cumulative probabilities and cumulative logit functions, which are defined as:

For cumulative Probability

$$P(Y \leq k) \quad \text{for} \quad k = 1, 2, \ldots, K-1$$

For cumulative Logit:

$$\log \left[\frac{P(Y \leq k)}{P(Y > k)} \right] \quad \text{for} \quad k = 1, 2, \ldots, K-1$$

where Y is the ordinal dependent variable and K is the number of categories.

The ordinal logistic regression model can be represented as:

$$\log \left[\frac{P(Y \leq k)}{P(Y > k)} \right] = \alpha_k - (\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)$$

where α_k is the threshold for category k, and β_0, β_1, ..., β_n are the coefficients of the predictor variables.

3.4 Difference between those two Regression models

After talking about those two models, we would like to find out what excatly are the main difference between logistic regression and ordinal logistic regression, which is the nature of the dependent variable.

Logistic regression is always used for binary dependent variables, whereas ordinal logistic regression is used for ordinal categorical dependent variables. Moreover, ordinal logistic regression uses cumulative probabilities and cumulative logit functions, which account for the ordered structure of the response variable, while logistic regression uses the logistic function to estimate the probability of the positive class.

Overall, logistic regression and ordinal logistic regression are statistical methods used to model relationships between dependent and independent variables. Logistic regression is suitable for binary outcomes, while ordinal logistic regression is designed for ordinal categorical outcomes. The key difference between them lies in the treatment of the dependent variable and the functions used to model the relationship between the predictors and the outcome.

4 Experiments

Moving everything from theory to the practical aspect, it is always more exciting. In this experiment stage, we use two methods: multinomial logistic regression with scikit-learn and ordinal logistic regression with statsmodels. We first import the necessary libraries and perform data preprocessing, including splitting the data into training and test sets. Then, we train the two models, evaluate their performance, and compare their results. As mentioned before, we use the dataset containing different chemical properties of red wine samples and their respective quality ratings ranging from 0 to 10. The dataset is split into training and test sets and we will discuss this in detail later on.

The platform we used for this experiment is Anaconda for its great performance and compatibility with different libraries and components, where we successfully installed all the necessary libraries and components. Moreover, getting the dataset ready for use is another big deal Anaconda (n.d.).

4.1 Data split

We start by importing the dataset and separating the features X and the target variable y. The dataset contains 11 features (independent variables) and 1 target variable (wine quality). The shape of X is (1599, 11) and the shape of y is (1599,).

For reproducibility, we set a `random seed (42)` and shuffle the dataset. We then separate five samples for the holdout set and save the main dataset and holdout set as separate CSV files.

We import the two new datasets `main_dataset` and `holdout_set` then verify their shapes. The main dataset's shape is $(1594, 11)$ for X and $(1594,)$ for y. The holdout set's shape is $(5, 11)$ and $(5,)$ respectively. By the way, it is always encouraged to know your data if any change happened, that is why we have been checking the shape of X and y all the time.

We explore the dataset using various functions, such as `dataset.head()`, `dataset.info()`, `dataset.describe()`, and `dataset.hist()`. This allows us to gain insights into the data, including the distribution of each numerical attribute.

We also calculate the correlation matrix to examine the relationships between the features and the target variable (wine quality). The correlation matrix shows that alcohol has the highest positive correlation with wine quality, followed by volatile acidity, which has the highest negative correlation.

Since the data we are going to use contains 1599 samples and we need to randomly select 5 samples aside, we have come up with a few ideas on how to do so initially. The first one is to manually remove those five from the file and set them into a new file for later usage. But which five should we select and how to ensure randomness along the way come to be a big deal. Therefore, we think of another idea which is to write some Python code to do it.

We import the dataset in the first place right after logging on Jupyter Notebook, then we choose the `pandas` library to shuffle the dataset and extract the first five samples from the newly ordered dataset. We remove those five samples from the original dataset to create a holdout set for final evaluation and comparison for all the models we are going to build. The original dataset is saved as `'main_dataset.csv'`, which contains 1594 samples (subtracting 5 from 1599), and the holdout set is saved as `'holdout_set.csv'`, containing 5 samples.

Then, we import the main dataset and the holdout set separately for the model training later. At the same time, the main dataset is split into training and test sets using the `StratifiedShuffleSplit` method from scikit-learn. We set the test set size to 20%. After splitting, we verify the shapes of the training and test sets. We choose this

method to ensure that the proportion of each quality rating in the training and test sets is the same as the proportion in the entire dataset which could largely avoid bias when selecting samples.

4.2 Experimental setup

We choose to use set up Jupyter Notebooks for building machine learning models via Anaconda software for convenience and easy-to-use sake. Once installed, open Anaconda Navigator for managing the Anaconda environment that we are going to use and launching applications. To Create a new environment specific to this experiment by installing the necessary packages for our use. When all is done, we Launch Jupyter Notebook and create a new notebook ready to go. Jupyter (n.d.)

What's more, we have to solve the data imbalance problem that mentioned before, so, to standardize the features using the `StandardScaler` from scikit-learn. This step is crucial because the features have different scales, which might lead to poor performance in our models. This also ensures that all features are on the same scale, which helps the model to converge faster and achieve better performance.

4.3 Experiment 1

In this very first model, we focus on the Logistic Regression model using scikit-learn with its `GridSearchCV` and `LogisticRegression` from two classes. scikit-learn developers (2019)

We use `GridSearchCV` to perform hyperparameter tuning for the logistic regression model. We explore three different types of regularization: L1, L2, and Elastic Net. We also set the solver, maximum number of iterations, and other parameters accordingly. For example, In the parameter grid for the model, we use the following settings: $C \in 0.01, 0.1, 1, 10, 100$, *penalty* set to *elasticnet* or *l1* or *l2*, $l1_ratio \in 0.1, 0.5, 0.9$, *solver* set to *saga*, and $max_iter \in 10000, 20000, 30000.$, After training the logistic regression model using the best parameters from `GridSearchCV`, we obtain an accuracy of approximately **57%** on the test set.

Overall, our experiment demonstrates that a Logistic Regression model can be effectively used to predict wine quality based on its chemical properties.

The confusion matrix for the test set results can be visualized in Figure 1:

Additionally, we tested the model on the holdout set of 5 samples and obtained the following confusion matrix and

```
[[ 0  0  1  1  0  0]
 [ 2  0  8  1  0  0]
 [ 1  0 97 36  2  0]
 [ 0  0 45 73  9  0]
 [ 0  0  4 24 12  0]
 [ 0  0  0  2  1  0]]

accuracy_score:  0.5705329153605015
```

Figure 1: The 20% test set prediction result shown in a confusion matrix

accuracy with 40%, see Figure 2:

```
[[2 0]
 [3 0]]

accuracy_score:  0.4
```

Figure 2: The 5 samples prediction result shown in Confusion matrix

4.4 Experiment 2

We create the ordinal logistic regression model using the `OrderedModel` function with a `logit` distribution. The model is then trained on the training set using the `BFGS` optimization algorithm, with a maximum of 10000 iterations and a learning rate of 0.001. We apply L2 regularization and use the ADAM solver with a batch size of 32. The early stopping is enabled to prevent overfittingSeabold and Perktold (n.d.).

OrderedModel Results

Dep. Variable:	y	Log-Likelihood:	-1221.0
Model:	OrderedModel	AIC:	2474.
Method:	Maximum Likelihood	BIC:	2556.
Date:	Sun, 30 Apr 2023		
Time:	08:39:54		
No. Observations:	1275		
Df Residuals:	1259		
Df Model:	16		

	coef	std err	z	P>|z|	[0.025	0.975]
x1	0.0404	0.162	0.250	0.803	-0.277	0.358
x2	-0.5705	0.080	-7.156	0.000	-0.727	-0.414
x3	-0.0758	0.101	-0.752	0.452	-0.273	0.122
x4	0.0761	0.076	0.998	0.318	-0.073	0.225
x5	-0.2794	0.073	-3.849	0.000	-0.422	-0.137
x6	0.2338	0.080	2.905	0.004	0.076	0.392
x7	-0.4504	0.089	-5.047	0.000	-0.625	-0.275
x8	-0.0347	0.146	-0.237	0.813	-0.322	0.252
x9	-0.2240	0.105	-2.141	0.032	-0.429	-0.019
x10	0.4799	0.070	6.811	0.000	0.342	0.618
x11	0.9874	0.104	9.451	0.000	0.783	1.192
3/4	-5.9412	0.362	-16.398	0.000	-6.651	-5.231
4/5	0.6433	0.173	3.723	0.000	0.305	0.982
5/6	1.3158	0.044	30.014	0.000	1.230	1.402
6/7	1.0601	0.042	24.984	0.000	0.977	1.143
7/8	1.0919	0.088	12.380	0.000	0.919	1.265

Figure 3: The OrderedModel Results

After training, we evaluate the model on the testing set by calculating the predicted probabilities and determining the predicted class labels. The accuracy of the model is then calculated by comparing the predicted labels to the

true labels. The model achieves an accuracy of 'accuracy' on the testing set which is around **41%**. The model's performance is further evaluated by interpreting the results through the `.summary()` function, see Figure 3.

Lastly, we evaluate the model on the holdout set of 5 samples by calculating the predicted probabilities and predicted class labels. The model achieves an accuracy of 'accuracy' on the holdout set, which demonstrates its ability to generalize to new, unseen data. Unfortunately, the accuracy is 0 which we guess the reason is due to the limited amount of test data is sometimes can lead to misclassified for all.

4.5 Results

Both models achieved similar accuracy scores on the 20% test set for approximately **57%** for the multinomial logistic regression model and **41%** for the ordinal regression model with L2 penalty. On the holdout set, both multinomial logistic regression models achieved a better accuracy with **40%**, which it is **20%** for ordinal regression model.

Overall, the correlation matrix and visualizations helped us understand the relationships between the features and the target variable, as well as the distribution of the data. The experiment demonstrated the capabilities of both multinomial logistic regression and ordinal regression models for predicting wine quality.

5 Discussion

Although the models achieve reasonable accuracy on both the testing and holdout test sets, there may be room for improvement. To enhance the performance, we can consider the following strategies: OpenAI (2021)

1. Feature Selection (Feature Engineering): Our current model includes all the available features in the dataset. Some features might have a higher impact on wine quality than others. By applying feature selection techniques, we can potentially identify the most relevant features and improve the model's performance and interpretability.

2. Alternative Models: Logistic Regression is a relatively simple model. Other machine learning models, such as Support Vector Machines, Random Forests, or Neural Networks, might yield better performance. Performing a comparative analysis with these models can help identify the best approach for this problem.

3. Imbalanced Dataset: The dataset contains an unequal number of samples for each quality rating, which might affect the model's performance for underrepresented categories. Addressing class imbalance by using techniques like oversampling, undersampling, or generating synthetic samples can lead to a more robust model.

4. Ensemble Methods: Combining multiple models through ensemble methods such as bagging, boosting, or stacking can improve prediction accuracy and achieve better generalization.

5. Hyperparameter Tuning: In our experiment, we only considered a limited range of hyperparameters for tuning. A more exhaustive search, or using alternative methods like RandomizedSearchCV or Bayesian optimization, might lead to better model performance. Experiment with different hyperparameters, such as the learning rate, batch size, and regularization strength, to find the optimal combination for the model. Grid search or random search techniques can be used to explore various combinations efficiently.

6. Cross-validation: Instead of using a single training and testing split, perform k-fold cross-validation to obtain a more reliable estimation of the model's performance. This approach can help mitigate the impact of potential biases in the dataset.

By implementing these strategies, we believe we can further refine the model and enhance its predictive capabilities for wine quality assessment.

6 Conclusion

In conclusion, our experiment shows that a Logistic Regression model can be used to predict wine quality based on its chemical properties with reasonable accuracy. However, there is room for improvement by addressing the model's limitations and exploring alternative approaches. By further refining our model and considering other techniques, we can potentially develop a more accurate and robust solution for predicting wine quality.

References

Anaconda, I. (n.d.). *Anaconda documentation.* `https://docs.anaconda.com/`. (Accessed on: April 30, 2023)

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems* (2nd ed.). CA, USA: O'Reilly Media, Inc.

Jupyter, P. (n.d.). *Jupyter documentation.* `https://docs.jupyter.org/en/latest/`. (Accessed on: April 30, 2023)

OpenAI. (2021). *Chatgpt.* https://openai.com/. (Accessed on: April 30, 2023)

scikit-learn developers. (2019). *scikit-learn: Machine learning in python.* https://scikit-learn.org/0.21/documentation.html. (Accessed on: April 30, 2023)

Seabold, S., & Perktold, J. (n.d.). *Statistical models, hypothesis tests, and data exploration.* https://www.statsmodels.org/dev/. (Accessed on: April 30, 2023)