SML Project - Documentation

Tristan Leiter

November 9, 2025

Contents

Chapter 1

Introduction

1.1 Introduction

Chapter 2

Exploratory Data Analysis

2.1 Overview

2.1.1 Description of the dataset

For this SML project, we are using the wine-quality dataset, obtained from Kaggle (https://www.kaggle.com/datasets/taweilo/wir quality-dataset-balanced-classification). It contains 21'000 observations with 11 features and 1 dependent variable, the wine quality. Wine quality is an ordinal variable. It represents a ranking from 3 to 9, with 9 being the highest score and better than all scores below it.

The dependent variable y is nicely balanced as it has the same distribution across all quality variables. Each quality variable contains 3'000 observations (14.3% of all observations). Subsequently, we keep quality as a numeric value and set this up as a regression task instead of a classification approach. This is done in order to allow for better inference (assuming the true quality would be 5, a prediction of 9 would be much worse than predicting a 4).

2.1.2 Feature description

Table 2.1: Dataset Variable Descriptions

Variable	Description	Type
fixed_acidity	Fixed acids (tartaric, malic, citric)	float64
volatile_acidity	Volatile acids (primarily acetic)	float64
citric_acid	Citric acid content	float64
residual_sugar	Remaining sugar after fermentation	float64
chlorides	Salt content	float64
free_sulfur_dioxide	Free SO2 (preservative)	float64
total_sulfur_dioxide	Total SO2 (bound + free)	float64
density	Density (related to alcohol/sugar)	float64
pH	pH level (acidity indicator)	float64
sulphates	Sulphates additive amount	float64
alcohol	Alcohol percentage	float64
quality	Target: Quality score (3-9)	int64

2.1.3 Feature bimodality

It can be observed that many features show a bimodal distribution (two different modes). We can clearly see the two distinct peaks in the histogram. This indicates that we observe a mixture of two different underlying distinct data generation processes. For example, due to the dataset including red and white wines, which usually have different, distinct properties.

For our further process, it essentially implies a hidden categorical feature (wine type: red or white wine) that is not explicitly given in the dataset. It can be argued that decision trees should handle this case rather wll as they can easily make a split to seperate the two populations whilst linear models might struggle.

2.1.4 Feature standardization

The variances differ by several orders of magnitude (total sulfur dioxide vs density, pH value,...). Standardization will be essential in order for our regularized linear models to work well and to ensure that all features contribute equally to the model. Yet, we also sort of loose some variation in the feature space.

2.1.5 Multicollinearity

Based on the correlation plot, multicollinearity is present in our dataset. Some features (free sulfur dioxide vs total sulfur dioxide or also density vs residual sugar) are highly positively correlated. In this case, we can expect the coefficients of linear models to be unstable as the model cannot properly differentiate the linear effect of each feature. Also, a small change in the training data could flip a coefficient from positive to negative, making interpretation very difficult. In addition, redundant features can lead to an overfitting of the model as we start to fit noise in the training data instead of the true signal.

Tree based models should generally work well in this environment as they will simply pick one of the correlated features to split on and ignore the others. It does not hurt their predictive performance. Yet, it will be difficult for interpreting the importance of each feature.

2.1.6 Splitting the dataset

We split the dataset in three subsets, the training set (50% of the data), a validation set (25% of the data), and finally the test set with the remaining 25%. Since the dependent variable is balanced, we do not require additional tools, like stratified sampling.

Chapter 3

Loss-function

3.1 Root Mean Squared Error (RMSE)

For this project, we decided to use root mean squared error (RMSE) as the key loss metric for both model validation and final performance assessment in the test set. We perform the following minimization:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Where:

* y_i is the actual quality score (e.g., 6). * $\hat{y}_i$ is the predicted score (e.g., 5.8). * N is the number of observations.

Whilst wine quality is an ordinal variable, a standard multi-class classification problem (using e.g. accuracy) is suboptimal. Firstly, accuracy would ignore ordinality. Predicting a quality of 3 for a true quality of 8 is penalized the same as predicting a 7, even though 7 is much better than a 3.

RMSE also treats the target as a continous variable. If our model predicts a 59 for a true 6, it has a very small error, whereas 5.1 would result in a bigger error. This rewards models that get close to the true value, even though we can not observe a continous value (rather a discrete one).

RMSE is also easier to interprete as the target variable. An RMSE of 0.7 would mean, on average, that our model's prediction are about 0.7 quality points away from the actual score.

3.2 RMSE vs MSE

We chose RMSE, because it scales back to the original units of our target variable. MSE is difficult to interpret by itself. An RMSE of 0.7 means, "on average, our model's prediction is about 0.7 points away from the true wine quality rating." We use a metric that matches the scale of the actual data.

Chapter 4

Regularized Linear Models

4.1 Ridge

Ridge regression is a regularized linear model that addresses multicollinearity and overfitting by adding an L_2 penalty to the ordinary least squares loss function. This penalty shrinks the coefficients towards zero but never exactly to zero, retaining all features in the final model.

The optimization problem for Ridge regression is:

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Where $\lambda \geq 0$ is a tuning parameter that controls the strength of the shrinkage.

4.2 Lasso

Lasso (Least Absolute Shrinkage and Selection Operator) is another regularized linear model, but it uses an L_1 penalty. Unlike Ridge, the L_1 penalty can shrink some coefficients exactly to zero, effectively performing automatic feature selection and producing sparse models.

The optimization problem for Lasso regression is:

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Chapter 5

Decision Trees

5.1 Random Forest

Random Forest is an ensemble learning method that builds a large collection of de-correlated decision trees. It uses bagging (training each tree on a random bootstrap sample of the data) and feature randomness (considering only a random subset of features for each split). The final prediction for regression is the average of the predictions from all individual trees.

For regression, each individual tree T_b is built by recursively splitting regions to minimize the Mean Squared Error (MSE) in each resulting node:

$$\text{Minimize} \sum_{R_j} \sum_{x_i \in R_j} (y_i - \hat{c}_j)^2$$

Where $\hat{c}_j$ is the average value of the target variable for observations falling into region R_j.

5.2 Gradient Boosting

Gradient Boosting Machines (GBM) also build an ensemble of decision trees, but they do so sequentially rather than in parallel. Each new tree is trained to predict the pseudo-residuals (errors) of the previous ensemble of trees, effectively correcting the mistakes of its predecessors.

At each step m, a new tree $h_m(x)$ is fitted to the negative gradient of the loss function (which, for squared error loss, is just the residual):

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x)$$

Where $F_m(x)$ is the updated model, ν is the learning rate (shrinkage) parameter, and $h_m(x)$ is the new tree that minimizes the loss:

$$h_m = \arg\min_h \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + h(x_i))$$

For our regression task, we use the squared error loss function $L(y, F(x)) = (y - F(x))^2$.

Chapter 6

Additional Models

6.1 CatBoost

Chapter 7

Results

7.1 CatBoost

Appendix A

Overview