SML Project - Documentation

Tristan Leiter

November 15, 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/wi
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
is_white	Binary indicator for wine type (1 = White, 0 = Red)	int64
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

Chapter 4

Regularized Linear Models

4.1 Multinomial Logistic Regression (Lasso & Ridge)

Implementation: `glmnet` with `family = "multinomial"`

What is being optimized?

The model minimizes the **Multinomial Negative Log-Likelihood** (also known as Cross-Entropy Loss), combined with a regularization term. The objective function to minimize is:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(p_{ik}) + \lambda \cdot \text{Penalty}(\beta)$$

Where:

- N is the number of observations.
- K is the number of classes (wine quality 3, 4, ..., 9).
- y_{ik} is an indicator (1 if observation i is in class k, 0 otherwise).
- p_{ik} is the predicted probability that observation i is in class k.
- λ is the regularization parameter (tuned via cross-validation).
- **Lasso Penalty:** $\sum |\beta_j|$ (pushes coefficients to exactly zero).
- **Ridge Penalty:** $\sum \beta_j^2$ (shrinks coefficients towards zero).

How it works

- **Training:** The `glmnet` algorithm uses coordinate descent to find the coefficients (β) that minimize this joint loss function for a fixed λ.
- **Tuning:** We test multiple λ values. For each, we calculate the **Accuracy** on the validation set. We select the λ that **maximizes Validation Accuracy**.

Chapter 5

Decision Trees

5.1 General Decision Trees

A conventional decision tree (like CART) is a powerful model, but it has a major flaw: it's greedy and prone to overfitting.

High Variance: As a result, if you slightly change the training data (e.g., add or remove a few rows), you can get a completely different tree. This instability is called high variance.

5.2 Random Forest

Implementation: `randomForest` with a factor response variable.

Random Forest does not strictly "minimize" a global loss function like regression models. Instead, it minimizes **impurity** at every split in every tree. For classification, the standard impurity measure is the **Gini Index**:

$$Gini(t) = 1 - \sum_{k=1}^{K} (p_{tk})^2$$

Where p_{tk} is the proportion of training samples in node t that belong to class k.

- **Training:** Each individual tree tries to find splits that **maximize the decrease in Gini Impurity**. A split that perfectly separates classes has a Gini index of 0.
- **Tuning:** We tune the `mtry` hyperparameter. For each `mtry` value, we train a forest and evaluate it on the validation set. We select the `mtry` that **maximizes Validation Accuracy**.

5.3 Gradient Boosting

Implementation: `gbm` with `distribution = "multinomial"`

GBM explicitly minimizes the **Multinomial Negative Log-Likelihood** (Multinomial Deviance), the same core loss function as standard logistic regression, but it does so sequentially.

$$Loss = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(p_{ik})$$

- **Training:** GBM builds trees sequentially. Each new tree is trained to predict the **negative gradient** of this loss function with respect to the current model's predictions. Effectively, each new tree tries to correct the probabilistic errors made by the ensemble so far.
- **Tuning:** We use K-fold Cross-Validation to tune the number of trees (M) and interaction depth (J). We select the combination that **minimizes the CV Multinomial Deviance**.

Chapter 6

Additional Models

6.1 CatBoost

CatBoost's primary advantages come from two novel techniques that solve common problems in standard gradient boosting: **target leakage** and **prediction shift**.

Implementation: `catboost` with `loss_function = 'MultiClass'`

CatBoost minimizes the **MultiClass** loss, which is its implementation of the negative log-likelihood for multiclass problems (similar to GBM's deviance).

- **Training:** Like standard GBM, it builds trees sequentially to minimize this loss. However, it uses **Ordered Boosting** to combat prediction shift and **Ordered Target Statistics** to handle categorical features (though we didn't have any in this specific dataset after pre-processing).
- **Tuning:** We explicitly set the `eval_metric = 'Accuracy'`. During hyperparameter tuning, we select the best parameters (learning rate, depth, L2 regularization) that **maximize Validation Accuracy**.

Chapter 7

Results

Appendix A

Overview