UNIVERSITY of Prince Edward ISLAND

CS 4120 – MACHINE LEARNING, DATA MINING

FINAL REPORT

AUTHORS:

- **CHIZURUM EWELIKE**
- **OLUWATOMILAYO FALOSEYI**

GitHub: https://github.com/LAYO200/Wine-Quality-Prediction-models

WINE QUALITY- FINAL REPORT

Section 1.0: Introduction

Expert tasters often judge wine quality, but their evaluations can be subjective and time-consuming. We build machine learning models to provide a reproducible, data-driven evaluation of wine quality, aiming to support consistent production decisions and offer reliable indicators for consumers.

We address two tasks on the same dataset:

- **Classification:** predict High Quality (score ≥ 7) vs Low/Medium (< 7).
- **Regression:** predict the original quality score (0–10)

Classical baselines (Logistic/Linear Regression and Decision Trees) were implemented and evaluated at the midpoint. In this final stage we add neural networks, perform hyperparameter tuning, and compare classical vs NN performance using five required plots and two summary tables.

Section 2.0: Dataset, Tasks, and Metrics

We use the Wine Quality dataset from the UCI Machine Learning Repository (Cortez et al., 2009), containing 6,497 Portuguese Vinho Verde wine samples (1,599 red; 4,898 white). Each sample has 11 physicochemical features (e.g., acidity, residual sugar, pH, sulphates, alcohol) and a target quality score in the range of [0, 10] based on expert ratings.

Cleaning:

- Concatenated red and white subsets; added a categorical type column.
- Coerced all feature columns to numeric; dropped rows that became NaN after coercion.
- Cast quality to int; ensured features are float64 for stable MLflow schema.
- Derived binary label target_cls = 1 if quality ≥ 7, else 0.

This dual-task design enables a side-by-side comparison of linear and nonlinear approaches on the same dataset.

New engineered features:
For the final stage we added two simple ratio features to inject interaction structure:

1. Total acidity ratio = fixed acidity / volatile acidity.
2. SO_2–density ratio = total sulfur dioxide / density.

Both features are numeric and are treated like other continuous inputs. These are intended to help nonlinear models (trees and NNs) capture relationships between acidity, sulfur dioxide, and density more easily.

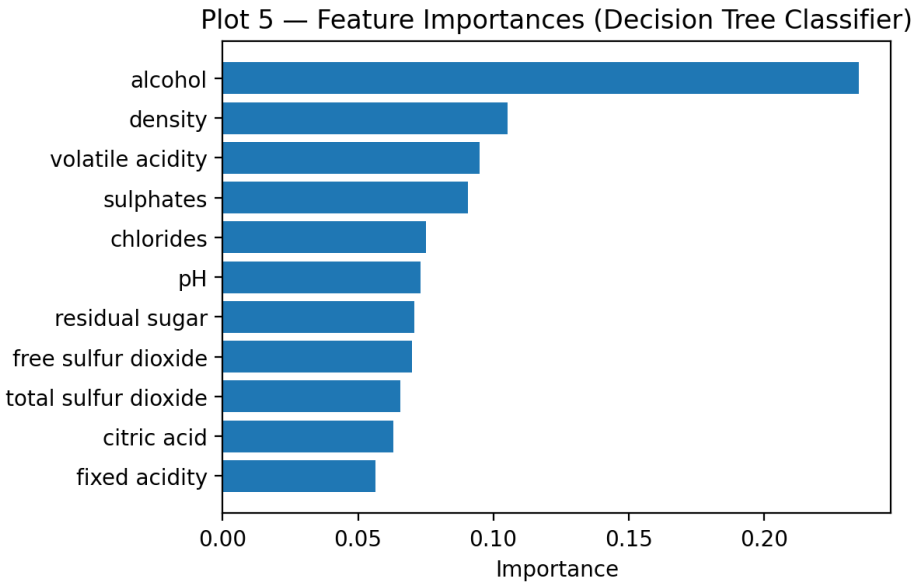
Section 3.0: Exploratory Data Analysis (EDA)

EDA was carried out in notebooks and partially summarized in the midpoint report. Key observations:

- The **class distribution** is skewed: roughly **81%** Low/Medium (0) vs **19%** High (1).
- Alcohol, sulphates, and lower density tend to correlate positively with quality, while volatile acidity is negatively related.
- Several features are moderately correlated with each other (e.g., free vs total sulfur dioxide).

In the final stage we focus on model-driven EDA:

- **Correlation heatmap (updated):** we overlay numeric correlation values on each cell so that relationships such as alcohol–quality and density–quality can be read quantitatively rather than just by color.



Interpretation: Shows that alcohol is by far the most important feature, followed by density, volatile acidity, and sulphates; fixed acidity and citric acid have relatively smaller importance. This aligns with the literature and helps interpret both classical and NN models.

These EDA results guided our model design: we expected nonlinear thresholds in alcohol and sulphates (favouring trees and NNs) and mostly monotonic relationships overall (explaining why linear models remain strong).

Section 4.0 Baselines and MLflow Tracking

We train two classical models for each task, implemented in src/train_baselines.py and reused in src/train_nn.py:

- **Classification baselines**
  - **Logistic Regression** with StandardScaler (linear decision boundary).
  - **Decision Tree Classifier** (nonlinear thresholds and interactions).
- **Regression baselines**

- **Linear Regression** with StandardScaler.
- **Decision Tree Regressor**.

Preprocessing is handled with a scikit-learn ColumnTransformer and Pipelines: numeric features are standardized, and the categorical type is one-hot encoded. This matches the plan in the proposal and midpoint and ensures that all models receive the same cleaned inputs.

MLflow tracking:

All experiments: classical and neural network, are tracked using **MLflow**:

- Each run logs:
  - Model type and hyperparameters,
  - Dataset split parameters (seed, val/test fractions, stratification target),
  - Evaluation metrics on validation and test sets,
  - Plots (figures 1–5) and comparison tables (Tables 1–2),
  - Serialized models with example inputs.

The tracking directory mlruns/ is committed to the repo so experiment history can be inspected and reproduced.

Section 5.0: Train–Validation–Test Split

We use the same split strategy as the midpoint to keep results comparable:

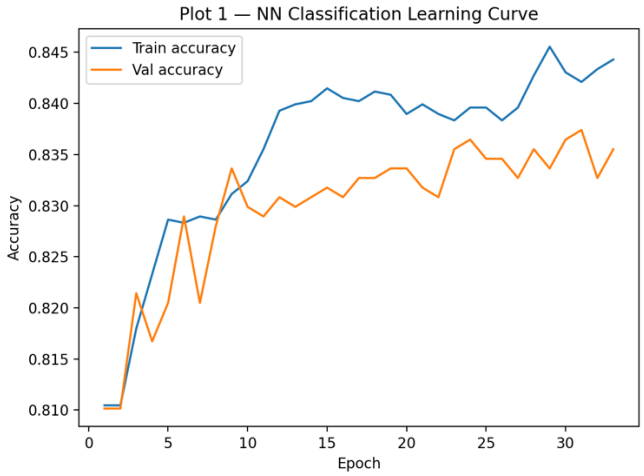
- **Initial split:**
  - 80% train, 20% test via train_test_split,
  - stratified on target_cls to maintain class imbalance in each split,
  - seed = **42** (fixed random seed).
- **Validation split:**
  - Within the 80% training set, we take **25%** as validation (so overall: 60% train, 20% validation, 20% test).
  - Again, we use stratified splitting on target_cls and the same seed.

This split is not time-based because the dataset has no temporal ordering; wines are not naturally indexed by date. Instead, a random stratified split is more appropriate and easier to reproduce. All models (classical and NN) use exactly the same train/val/test indices**,** avoiding any unfair advantage.

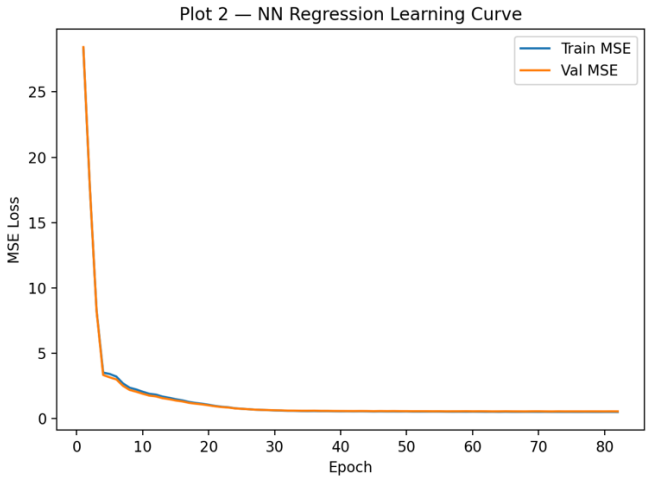
Section 6.0: Results and Discussion

This section summarizes the performance of classical baselines and the newly implemented neural networks, using the five required plots and two comparison tables. We focus on what worked**,** failure modes**,** and next steps**.**

Neural network learning behaviour:



Interpretation: Plots training and validation accuracy vs epoch for the MLPClassifier**.** Accuracy rises quickly in the first ~10 epochs and then stabilizes around 0.83–0.84. The validation curve closely tracks the training curve, suggesting that with dropout and early stopping the network does not strongly overfit.



Interpretation: **S**hows training and validation MSE vs epoch for the MLPRegressor. Both curves drop steeply at the beginning (from >25 down to <4) and then decay smoothly toward ~0.5–0.6, with very little gap between train and validation loss. This indicates that the regression NN fits the data well without memorizing noise

Classification performance and failure modes:

Table 1 – Classification: best classical vs neural network

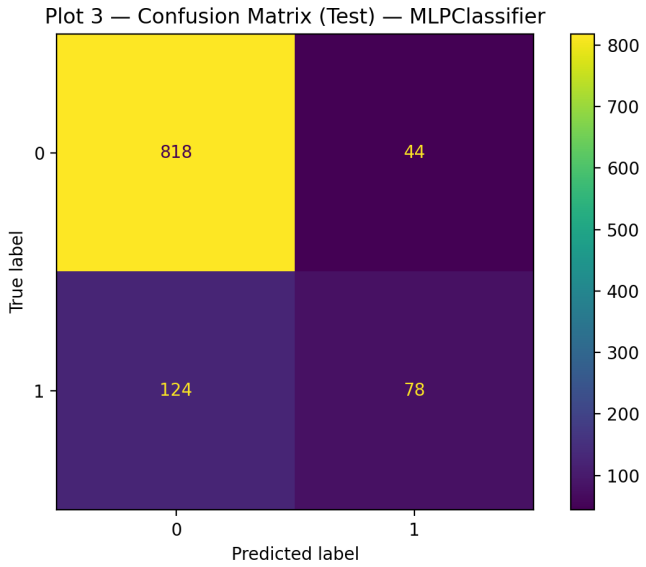
Model	Val Accuracy	Val F1	Test Accuracy	Test F1
Decision tree classifier	0.793233	0.471154	0.789474	0.469194
MLP Classifier	0.835526	0.471299	0.842105	0.481481

As in the midpoint report:

- The Decision Tree Classifier outperforms Logistic Regression in F1 despite slightly lower accuracy, confirming that it handles nonlinear thresholds and interactions (e.g., “alcohol above 10.5% and sulphates above X”).

In the final stage:

- The MLPClassifier matches the tree’s validation F1 and improves both validation and test accuracy**.**
- On the test set, it achieves the highest F1 (0.481 vs 0.469 for the tree), making it the best overall classifier on our chosen metrics.



Interpretation: Reports a confusion matrix with:

- TN = 818, FP = 44
- FN = 124, TP = 78

The model correctly identifies most low-quality wines and a moderate proportion of high-quality wines. The main failure mode is false negatives: good wines (label 1) predicted as low quality (label 0). These errors cluster near the 6/7 threshold, where the sensory scores are themselves borderline. This suggests that even with nonlinear modelling, perfect separation between strong 6s and weak 7s is not realistic.

Possible next steps for classification include:

- Adjusting the decision threshold or using class weights to reduce false negatives,
- Exploring tree ensembles (Random Forests, Gradient Boosted Trees) as strong baselines to compare with the MLP.

Regression performance and failure modes:

Table 2 – Regression: best classical vs neural network

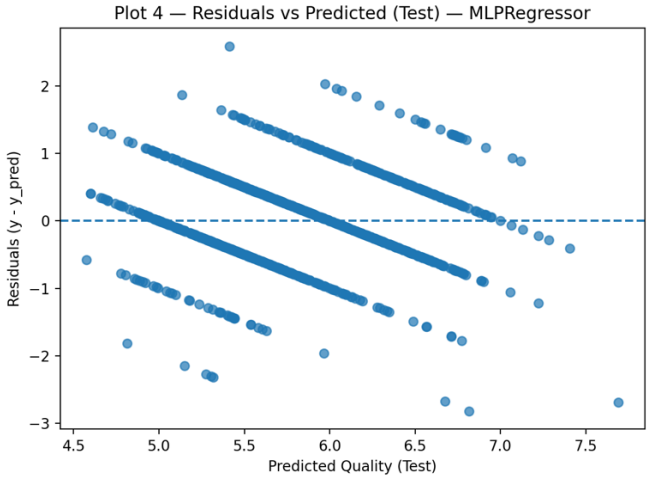
Model	Val Accuracy	Val F1	Test Accuracy	Test F1
LinearRegression	0.573437	0.735946	0.546002	0.701434
MLPRegressor	0.574711	0.734884	0.545765	0.688778

From the baselines:

- Linear Regression clearly beat the Decision Tree Regressor at midpoint and remains a strong model, indicating primarily monotonic relationships between features and quality.

In the final:

- The MLPRegressor has virtually identical MAE but slightly lower RMSE than Linear Regression on both validation and test sets, meaning it reduces larger errors while maintaining similar average error.



Interpretation: Plots residuals $y - \hat{y}$ against predicted quality. Residuals are roughly centered around zero with a gentle downward slope: the NN still underestimates some low-scoring wines and slightly overestimates some high-scoring ones, but the bias is less severe than for the linear model reported at midpoint. The spread of residuals does not explode at the extremes, suggesting reasonable generalization.

Next steps for regression could include:

- adding more interaction features or nonlinear terms,
- comparing the MLP against gradient-boosted tree regressors,
- modelling predictive uncertainty (e.g., prediction intervals) to reflect label noise.

What worked overall:

- Classical models already perform well: Decision Trees capture useful nonlinear structure for classification, and Linear Regression captures global trends for regression.
- Neural networks do not drastically outperform the baselines but provide small, consistent improvements**:**
    - F1 improves from 0.469 to 0.481 on the test set for classification.
    - RMSE drops from 0.701 to 0.689 for regression.
- Input standardization, dropout (p = 0.3), and early stopping (patience = 10) produced stable learning curves without major overfitting

Section 7.0: Neural Network Plan and Justification

The final model family for this project is a multilayer perceptron (MLP)**,** which is appropriate for tabular, numeric data such as physicochemical wine measurements.

Intended architecture family:

- **Family:** Fully connected MLPs (one for classification, one for regression).
- **Layers:**
    - Input layer: standardized numeric features + one-hot encoded wine type + engineered ratios.
    - Two hidden layers with 64 and 32 units and **ReLU** activations.
    - Output layer:
        - single logistic unit for classification (MLPClassifier),
        - single linear unit for regression (MLPRegressor).
- **Training:**
    - Optimizer: **Adam**, learning rate **1e-3**.
    - Loss: cross-entropy (classification) and MSE (regression).
    - Input dropout with probability 0.3 each epoch.
    - Early stopping on validation performance with patience = 10 out of at most 100 epochs.

Justification:
An MLP is a natural choice here because:

- The features are **numeric and tabular**, with no obvious spatial or sequential structure that would call for CNNs or RNNs.
- We expect **nonlinear interactions** (e.g., combinations of alcohol, acidity, sulphates, and SO_2) that go beyond what purely linear models capture.

- Compared to more complex architectures, MLPs are relatively lightweight, easy to tune, and integrate well with scikit-learn and MLflow.

The results confirm this design: MLPs slightly outperform our best classical models while remaining trainable on a standard laptop. Future extensions could combine MLPs with tree ensembles or apply similar architectures to related regression and classification tasks in food and beverage quality control.

Section 8.0: Conclusion

Using the UCI Wine Quality dataset, we built an end-to-end pipeline that cleans the data, performs EDA, trains classical baselines, and adds neural networks for both classification and regression. With careful preprocessing, stratified train–validation–test splits, and MLflow tracking, all experiments are fully reproducible from the final repository.

Classical models already provide strong baselines; the Decision Tree Classifier and Linear Regression achieve realistic performance and are easy to interpret. The neural network models (MLPClassifier and MLPRegressor) offer modest but consistent improvements in F1 and RMSE, confirming that there is some nonlinear structure that benefits from flexible function approximators without overwhelming the simplicity and interpretability of classical methods.