Exploring Multilayer Perceptron Architectures on the Wine Quality Dataset

Student Name: Peyyala Naveen kumar
Student ID: 23084508
GitHub: https://github.com/Naveen220609/mlp-wine-quality-architecture-tutorial
Dataset : winequality-red.csv

---

1. Introduction

What is a neural network?

Imagine a system that learns from examples, similar to how your brain learns. A **neural network** is a computer model inspired by how the brain works. It takes information (like features of wine) and passes it through layers of processing, learning patterns to make predictions (like deciding if wine is good or bad).

A **Multilayer Perceptron (MLP)** is one type of neural network. It has three main parts:

1. **Input layer**: receives the information (11 wine measurements)
2. **Hidden layer(s)**: processes the information (the "learning" happens here)
3. **Output layer**: gives the final answer (good or not good)

Why does architecture matter?

Think of "architecture" as the design of the network—how many hidden layers and how many neurons (processing units) in each layer. A network with more neurons and layers can learn more complex patterns, but it can also "memorise" the training data instead of learning true patterns, which is bad for predicting new wines.

This tutorial answers a simple question: **Is a bigger, deeper network always better? Or is a simpler, smaller network sometimes enough?**

---

2. The Wine Quality Dataset

What data are we using?

We used the **red Wine Quality dataset** from UCI Machine Learning Repository, which contains **1,599 real red wines** from Portugal. For each wine, we have **11 measurements** of its chemistry:

- Fixed acidity, volatile acidity, citric acid
- Residual sugar, chlorides
- Free sulfur dioxide, total sulfur dioxide
- Density, pH
- Sulphates, alcohol content

A human expert then tasted each wine and gave it a score from 3 to 8.

Our classification task

We simplified the task to a **yes/no question**: "Is this a **good wine** or **not good wine**?"

- **Good wine** = quality score 7 or 8 (217 wines, about 13.6%)
- **Not good wine** = quality score 3 to 6 (1,382 wines, about 86.4%)

This is an **imbalanced dataset**: most wines are in the "not good" category. This is realistic because truly excellent wines are rare.

How was the data split?

We divided the data into two groups:

- **Training set**: 1,279 wines (80%) – used to teach the network
- **Test set**: 320 wines (20%) – used to check if the network learned well

We made sure both groups have the same proportion of good and bad wines.

Why scale the features?

The wine measurements have very different sizes. Alcohol is around 8–15, but chlorides are often below 0.1. If we don't scale, the network gets confused and trains poorly. **Scaling** means adjusting all measurements to the same range (mean 0, standard deviation 1) so the network can learn fairly from all features.

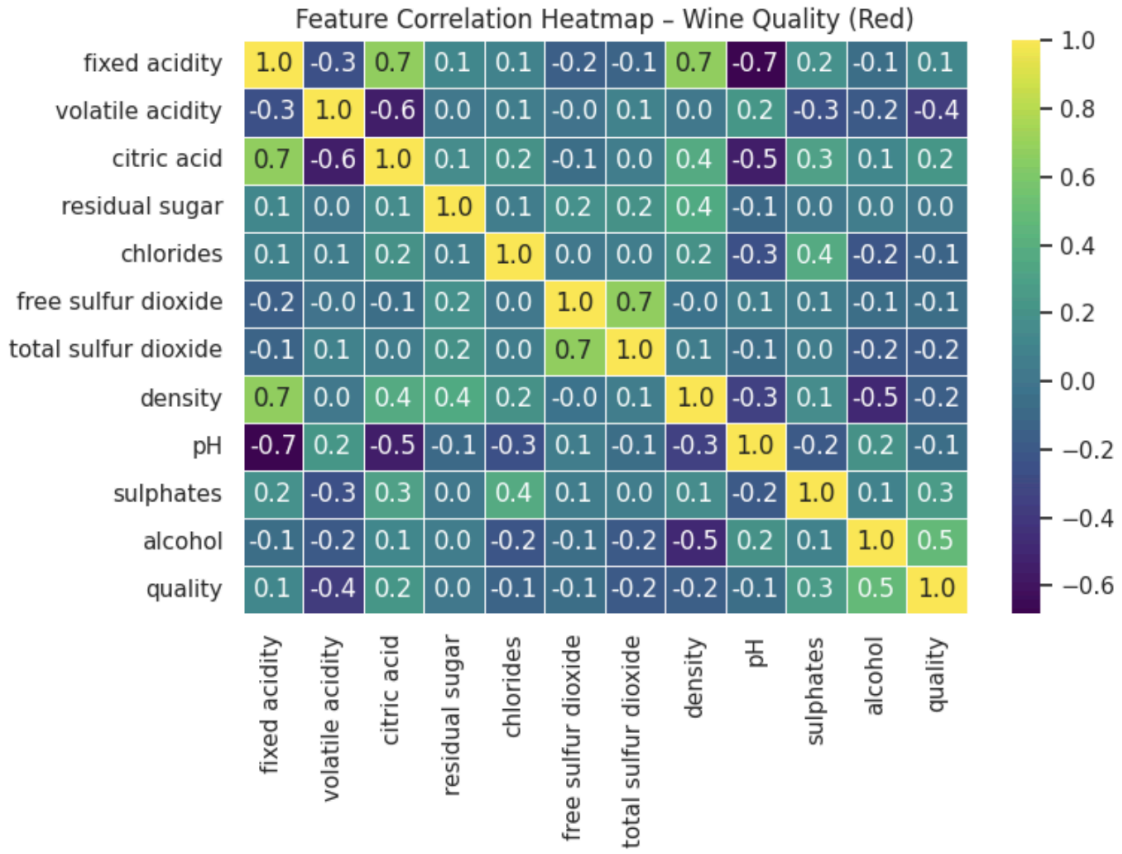

Figure 1: Feature Correlation Heatmap

Heatmap showing how wine chemistry features relate to each other and to quality. Brighter yellow = stronger correlation. Notice alcohol and sulphates have warmer colours with quality (positive), while volatile acidity is darker (negative).

Feature Insights: What the Network Learns

Which features matter most?

The correlation heatmap (Figure 1) shows which wine properties best predict quality:

- **Strong positive**: Alcohol content & Sulphates (wines with more alcohol and sulphates tend to be rated higher)
- **Strong negative**: Volatile acidity (wines with more vinegar-like smell tend to be rated lower)
- **Moderate**: Density, pH, citric acid

The MLP learns to use these relationships. Higher alcohol + higher sulphates = likely good wine. Higher volatile acidity = likely bad wine.

---

3. Simple Neural Networks: The Baseline

What is the baseline model?

We start with a very simple network as a baseline to compare against. This baseline has:

- **1 hidden layer** with **50 neurons**
- **ReLU activation** – a function that says "if input is negative, make it 0; if positive, keep it" (helps the network learn non-linear patterns)
- **Adam optimizer** – a method for updating the network's weights during training (like a learning algorithm)
- **Alpha = 0.0001** – a regularisation parameter (prevents the network from memorising instead of learning)

What are these terms?

ReLU (Rectified Linear Unit): A simple rule applied after each neuron. If the neuron's output is negative, ReLU turns it to 0. If it's positive, ReLU keeps it. This helps the network learn curved, non-linear relationships. Think of it like a light switch: off (0) or on (positive value).

Adam optimizer: A method for teaching the network. During training, the network adjusts its internal "weights" (connection strengths) to improve predictions. Adam is like a smart teacher that adapts the learning speed as training progresses.

Alpha (regularisation strength): A number that penalises the network if it gets too complex.

- Higher alpha = more penalty = simpler network (less overfitting).
- Lower alpha = less penalty = network can be more complex (more overfitting risk).

We used alpha = 0.0001 (weak penalty) for baseline and alpha = 0.001 (stronger penalty) for the best model.

Baseline results

The baseline network achieved:

- **Test accuracy: 0.925 (92.5%)**
- **Correctly identified 269 out of 277 "not good" wines**
- **Correctly identified 27 out of 43 "good" wines**

This is already very good! It's much better than always guessing "not good" (which would be right 86.4% of the time but never catching any good wines).

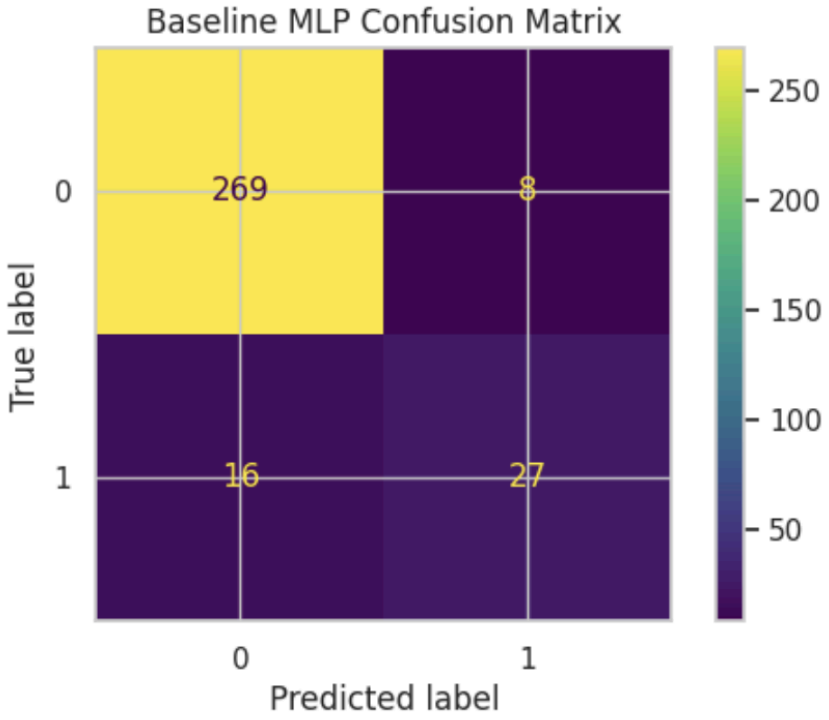

Figure 2: Baseline Model Confusion Matrix

Confusion matrix for the baseline MLP. Numbers show how many wines were correctly or incorrectly classified. Larger numbers along the diagonal (top-left and bottom-right) mean better predictions.

Baseline MLP – Test Metrics

Class	Description	Precision	Recall	F1-score	Support
0	Not good wine	0.94	0.97	0.96	277
1	Good wine	0.77	0.63	0.69	43
	Accuracy			**0.93**	320
	Macro average	0.86	0.80	0.82	320
	Weighted avg	0.92	0.93	0.92	320

Figure 3: Classification report

The F1-score combines precision and recall into one number by taking their harmonic mean,

$$F1 = 2 \times \frac{precision + recall}{precision \times recall}$$

, so it is high only when both are high.

---

4. Deeper and Wider Networks: The Experiments

The question we ask

The baseline with 50 neurons in 1 layer works well. But what if we:

- Make it **wider** (add more neurons to the layer)?
- Make it **deeper** (add more hidden layers)?

Does that improve predictions? Or does it just memorise the training data without learning true patterns?

The architectures we tested

We trained 8 different network architectures:

Architecture	Layers	Neurons per layer	Alpha
(50,)	1	50	0.0001
(50,)	1	50	0.001
(100,)	1	100	0.0001
(100,)	1	100	0.001 ✓ **Best**
(100, 100)	2	100, 100	0.0001
(100, 100)	2	100, 100	0.001
(100, 100, 100)	3	100, 100, 100	0.0001
(100, 100, 100)	3	100, 100, 100	0.001

Each architecture was trained and tested 3 times with different data splits (called **cross-validation**). This gives a fair estimate of how well each architecture generalises.

Results: What did we find?

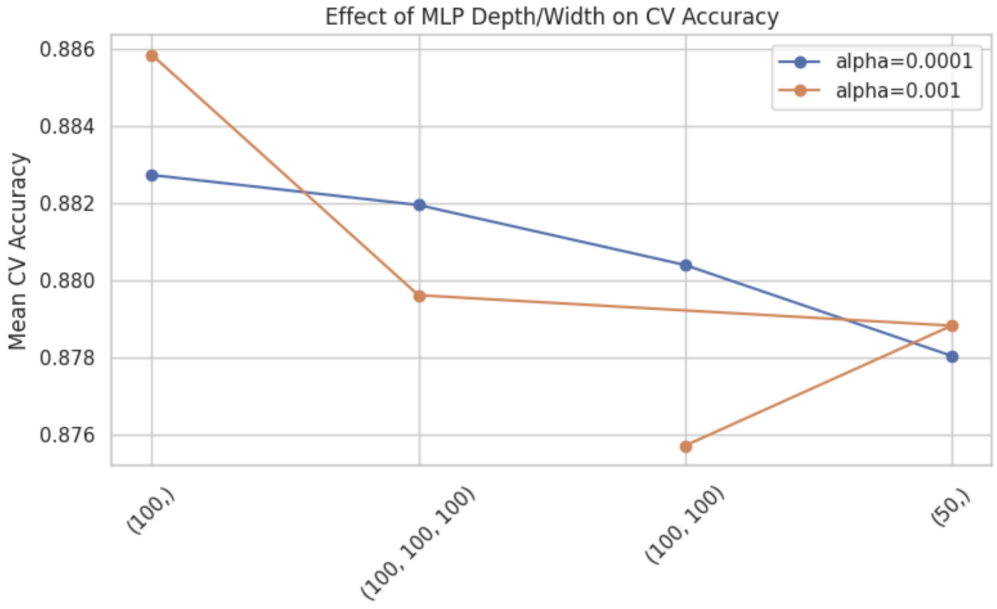

Figure 4: Comparison of Cross-Validation Accuracy

How accuracy changes as we make networks deeper or wider. Each line represents a different regularisation strength (alpha). Notice the single-layer 100-neuron model performs best.

Key findings:

1. **Wider is good**: Changing from 50 to 100 neurons improved accuracy from ~0.878 to ~0.886.
2. **Deeper is not always better**: Adding more layers (100,100) and (100,100,100) actually **lowered** accuracy back to ~0.876–0.882.
3. **Regularisation matters**: Using a stronger penalty (alpha=0.001) helped the simpler network learn better.

Why does deeper not always help?

With only 1,599 wines for training, deeper networks have too much "capacity" (too many adjustable weights). They learn to perfectly memorise the training data instead of learning true patterns. This is called **overfitting**.

We can see this in the cross-validation results:

- Deeper networks get almost **100% accuracy on training data** (they memorised it)
- But only ~87–88% on validation data (they forgot what they learned on new data)
- Simpler networks: **~98% on training**, **~88–89% on validation** (better balance)

---

6. Best Model vs Baseline: Side-by-Side Comparison

Which model won?

After cross-validation, the best model was: **Single layer, 100 neurons, alpha = 0.001**

Let's compare it to the baseline:

Metric	Baseline (50,)	Best Model (100,)
Cross-validation accuracy	—	0.8859
Test accuracy	0.9250	0.9156
Good wines correctly found	27 / 43 (63%)	27 / 43 (63%)
Bad wines correctly found	269 / 277 (97%)	266 / 277 (96%)

Surprising result: The baseline achieved slightly **higher test accuracy** (0.925 vs 0.916) But the best-model performed better on **cross-validation** (0.886).

This teaches an important lesson: **Cross-validation and test accuracy don't always perfectly match.** One model may look best during training but perform differently on fresh data. This is normal and why we report both metrics.

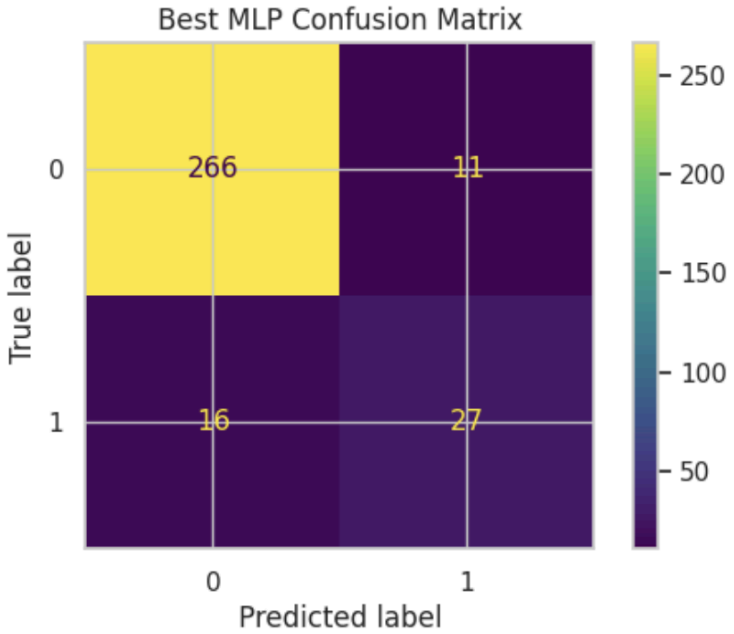

Figure 5: Best Model Confusion Matrix

Confusion matrix for the best MLP (100 neurons, alpha=0.001). Compare with Figure 2 to see differences from baseline.

Best MLP (100 neurons, alpha = 0.001) – Test Metrics

Class	Description	Precision	Recall	F1-score	Support
0	Not good wine	0.94	0.96	0.95	277
1	Good wine	0.71	0.63	0.67	43
	Accuracy			0.92	320
	Macro average	0.83	0.79	0.81	320
	Weighted avg	0.91	0.92	0.91	320

Figure 6: Classification report

---

7. Learning Curves: Does more data help?

What is a learning curve?

A learning curve shows how a model improves as we give it more training examples. If validation accuracy keeps going up, more data would help. If it flattens, we've hit the limits of this model design.

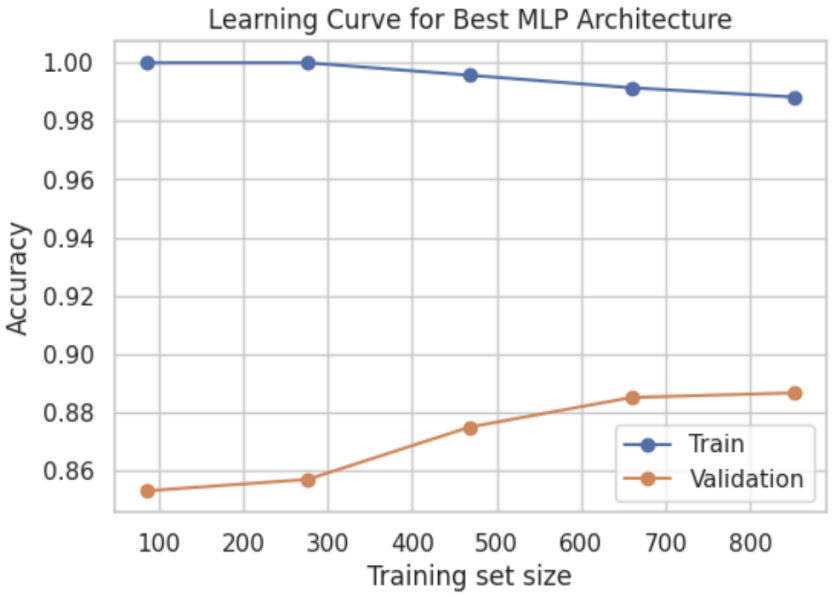

Figure 7: Learning Curve for Best Model

How the best model performs as we increase training data size. Training accuracy stays near 100% while validation accuracy grows from ~85% to ~89%, showing there's still room for improvement with more data.

What does this mean?

- The training accuracy stays near 100% (the network memorises the training data)
- The validation accuracy climbs from 85% → 89% as more data is used
- **The validation curve hasn't flattened**, meaning more labelled wines would probably help

This suggests that collecting more data (if possible) might improve the model more than making it deeper or wider.

Simple Recommendations

If you were building an MLP to predict wine quality, here's what we learned:

Situation	What to do
Starting out	Use 1 hidden layer, 50–100 neurons. Start simple.
If accuracy is low	Add more neurons to the layer (make it wider), not more layers (depth).
If you suspect overfitting (training near 100%, test much lower)	Increase alpha (stronger regularisation).
If you have lots of data	You can afford deeper networks; try 2–3 layers.
If you have little data (like 1,599 samples)	Keep it shallow (1–2 layers). Deep networks will memorise.

---

8. Discussion: Why Does This Matter?

Real-world applications

Wine quality prediction is low-risk, but the same **MLP techniques** are used in serious domains:

- **Medical diagnosis**: Predicting if a patient has a disease
- **Credit scoring**: Deciding loan approval
- **Fraud detection**: Spotting suspicious transactions

In these cases, making the right choice about network depth and width is crucial:

- **Too simple** (shallow, narrow): Model misses real patterns → wrong diagnoses
- **Too complex** (deep, wide): Model memorises noise and biases → unfair or overconfident decisions

Ethical note

If you use an MLP to predict wine quality for wine rating or pricing, be aware:

- The model may amplify biases in historical data (e.g., if expensive wines were over-represented in training)
- A 91.5% accurate model still gets 1 out of 11 wines wrong
- Simple, interpretable models may be preferable to complex ones if humans need to explain decisions to customers or regulators

---

9. Conclusion

This tutorial showed that on the Wine Quality dataset, a **simple MLP with one hidden layer of 100 neurons performs best**. Making the network deeper (adding 2 or 3 layers) did not improve accuracy and actually hurt cross-validation performance, likely due to overfitting. This teaches a valuable lesson: **more capacity is not always better**. Model design should match the size and complexity of the data.

The practical takeaway: **Start simple, test carefully with cross-validation, and only add complexity if it demonstrably helps.**

---

10. References

- UCI Machine Learning Repository. (2009). "Wine Quality Data Set." = “winequality-red.csv” Available at https://archive.ics.uci.edu/dataset/186/wine+quality
- Cortez, P., Cerdeira, A., Alves, F., et al. (2009). "Modeling wine preferences by data mining from physicochemical properties." *Decision Support Systems*, 47(4), 547–553.
- Scikit-learn developers. (2024). "MLPClassifier — scikit-learn documentation." Available at https://scikit-learn.org/stable/modules/neural_network_supervised.html
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
