

Dataset Selection →

1. [IRIS](#): Small(150 Entries)
2. [Wine Quality](#): Medium(6497 Entries)
3. [Adult](#): Large(48842 Entries)

Preprocessing Summary→

1. IRIS →

- a. No missing values
- b. Features Standardized
- c. Labels Encoded (0,1,2)
- d. There were 4 numerical features with 3 classes

2. Wine Quality →

- a. There were 12 numerical features with Binary classification(0 = Bad wine quality <6 , 1 = Good wine >=6)
- b. Features Standardized
- c. Converted quality scores to binary classification
- d. Class Distribution (3290 Good, 1907 Bad)

3. Adult →

- a. There were 13 features with binary classification (0 = Income <= 50k , 1 = Income > 50k)
- b. Removed Missing Values
- c. Encoded 8 Categorical Features
- d. Used pre-existing train-test split from UCI

Model Training and Tuning→

Methodology :

1. Data: Preprocessed standardised numeric arrays
2. Split : Iris/Wine(80:20) , Adult(Original split)
3. Cross-Validation: GridSearchCV with cv = 5 , scoring = 'f1_weighted'
4. Metrics on test set: Accuracy , Precision , Recall , F1 , Confusion Matrix

Best Hyperparameters (Tested Configuration Selected by CV) :

1. Iris
 - a. GaussianNB: var_smoothing=1e-09
 - b. DecisionTree: max_depth=5, min_samples_split=2, min_samples_leaf=1, criterion=gini

- c. MLP: hidden_layer_sizes=(100,), activation=relu, learning_rate_init=0.001, max_iter=500
2. Wine
 - a. GaussianNB: var_smoothing=1e-08
 - b. DecisionTree: max_depth=15, min_samples_split=5, min_samples_leaf=2, criterion=entropy
 - c. MLP: hidden_layer_sizes=(100,50), activation=relu, learning_rate_init=0.001, max_iter=1000
3. Adult
 - a. GaussianNB: var_smoothing=1e-08
 - b. DecisionTree: max_depth=None, min_samples_split=10, min_samples_leaf=4, criterion=gini
 - c. MLP: hidden_layer_sizes=(100,100), activation=relu, learning_rate_init=0.001, max_iter=1000

Test Set Performance:

1. IRIS(multi-class)
 - a. NB: Acc 0.967, F1_weighted 0.966
 - b. DT: Acc 0.900, F1_weighted 0.899
 - c. MLP: Acc 0.800, F1_weighted 0.796
2. Wine(binary)
 - a. NB: Acc 0.682, F1_weighted 0.676
 - b. DT: Acc 0.775, F1_weighted 0.772
 - c. MLP: Acc 0.776, F1_weighted 0.774
3. Adult(binary)
 - a. NB: Acc 0.804, F1_weighted 0.788
 - b. DT: Acc 0.856, F1_weighted 0.844
 - c. MLP: Acc 0.851, F1_weighted 0.839

Confusion Matrices and MLP curves are saved under results/plots/

Observations

- Iris: Simple Gaussian assumptions work well; small data favours NB. Overfitting hurt DT beyond shallow depth; MLP underperformed likely due to small size.
- Wine: Non-linear interactions benefited DT and MLP; NB was limited by feature correlations.
- Adult: The Larger dataset helped tree depth and MLP capacity; NB struggled with mixed encoded categorical distributions.
- MLP training curves: Faster convergence on Iris (risk of overfit); smoother improvement on Wine; longer plateau phases on Adult before stabilisation.

In Context Learning→

Prompt Design

1. I used a Python script to generate a prompt for giving to the LLM, which basically includes some common text and uses some metrics to get test data and training data from the dataset
2. The dataset is present in the ICL/data_block folder, which was used for training, and the prompts are present in the ICL/prompts

Result:

- Iris (n=30): Accuracy 0.833, Macro-F1 0.832, Weighted-F1 0.832
- Wine (n=150): Accuracy 0.493, Macro-F1 0.450, Weighted-F1 0.491
- Adult (n=200): Accuracy 0.605, Macro-F1 0.564, Weighted-F1 0.628

Interpretation:

- Iris: Strong few-shot performance; the LLM captures class boundaries with only 6 labelled examples.
- Wine: Near-chance accuracy for a 2-class task; indicates sensitivity to feature scaling/format and few-shot selection.
- Adult: Moderate performance above naive majority, but below typical traditional ML baselines on this dataset.

Benefits of ICL vs traditional ML:

- Zero training time: immediate reuse of the same model across datasets.
- Flexible adaptation: adjust few-shot examples to steer decision boundaries.
- Good for rapid prototyping and low-resource labelling scenarios.

Limitations:

- Output formatting fragility (extra tokens inflate or reduce line count).
- Sensitivity to the choice/order of a few-shot examples.
- Scaling issues: larger feature counts or noisier data often underperform tuned classical models.
- Cost: Repeated large prompts can be more expensive than a single, trained, lightweight model.

Evaluation Metrics and Comparisons→

Best result:

1. Iris (small, clean, low-dimensional)

- Best model: Naive Bayes
- Test: Acc 0.967, F1-weighted 0.967
- Observation: The Gaussian assumption works very well on this small, well-separated dataset.

2. Wine (medium, noisy, numeric)

- Best model: MLP
- Test: Acc 0.776, F1-weighted 0.777
- Observation: Non-linear decision boundaries help; MLP slightly outperforms the Decision Tree.

3. Adult (large, mixed features originally; encoded + scaled)

- Best model: Decision Tree
- Test: Acc 0.856, F1-weighted 0.849
- Observation: With this preprocessing, a tuned tree outperforms MLP; Naive Bayes lags behind on mixed/encoded distributions.

Cross Model Comparison:

1. Accuracy/F1

- Classical models outperform ICL on Wine and Adult by a strong margin.
- On Iris, ICL is reasonable but still behind a simple Naive Bayes baseline.

2. Training time

- Classical: Naive Bayes \ll Decision Tree $<$ MLP (Adult MLP is slowest).
- ICL: No training, but repeated API calls per test batch; cost/time scale with test size.

3. Computational cost

- Classical: One-time CPU training; cheap inference.
- ICL: Ongoing API cost and latency per evaluation; batching mitigates this but does not remove it.

4. Interpretability

Decision Tree \gg Naive Bayes $>$ MLP $>$ ICL.

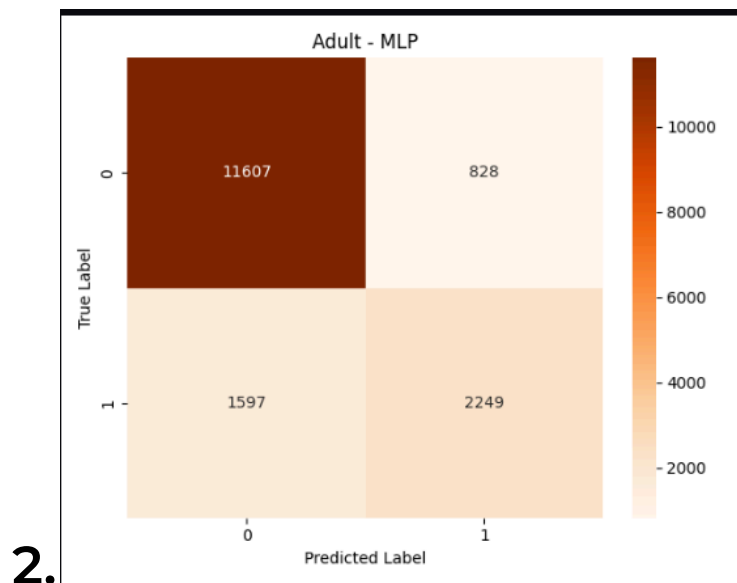
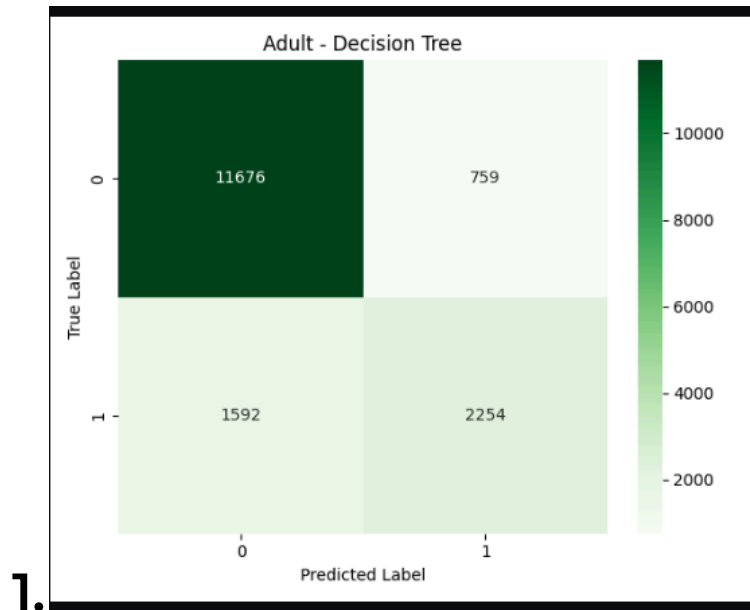
5. Robustness/generalisation

- Classical models (with proper preprocessing) are stable and reproducible.
- ICL is sensitive to prompt format, few-shot examples, and output cleanliness

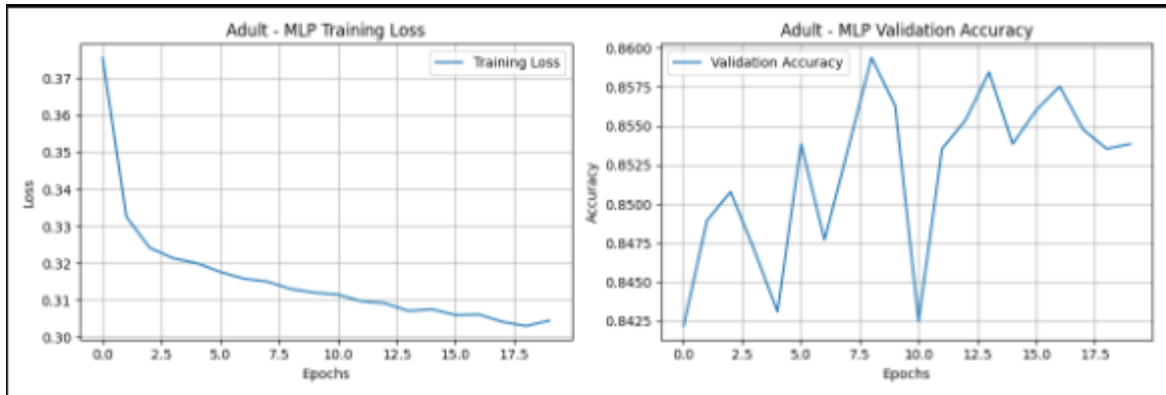
6. Ease of implementation

- Classical: Straightforward with scikit-learn; grid search/metrics tooling is mature.
- ICL: Quick to prototype (no training), but careful prompt design and output parsing are required to get reliable numbers.

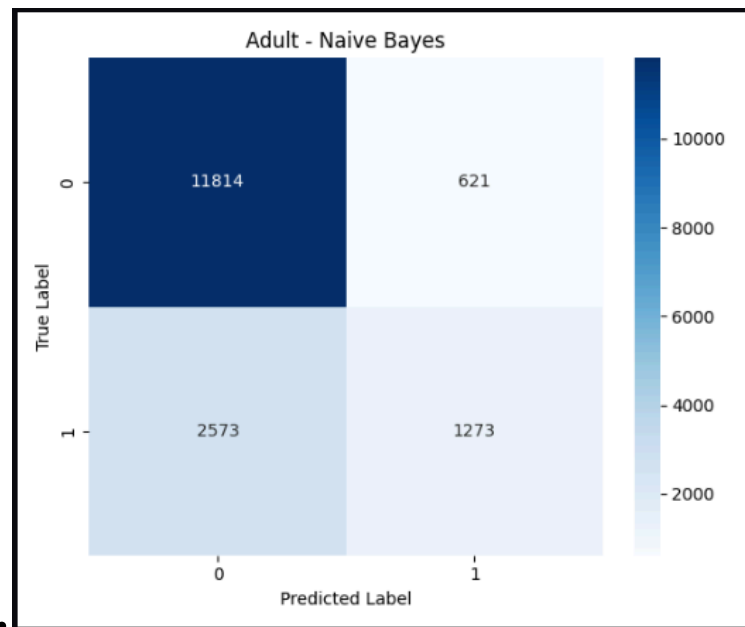
PLOTS



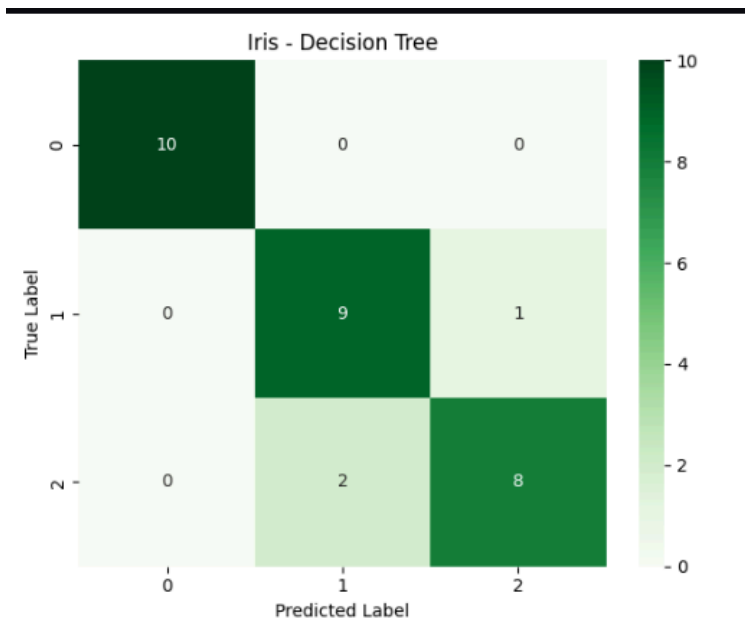
3.



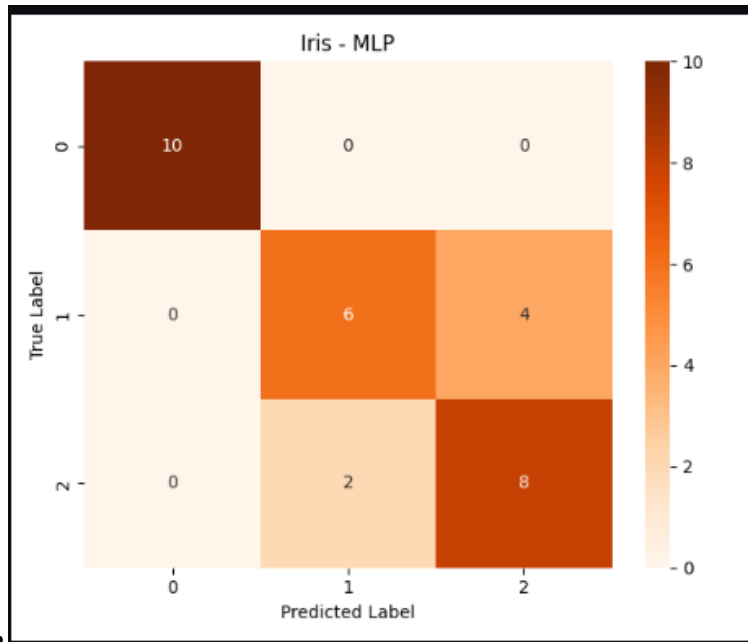
4.



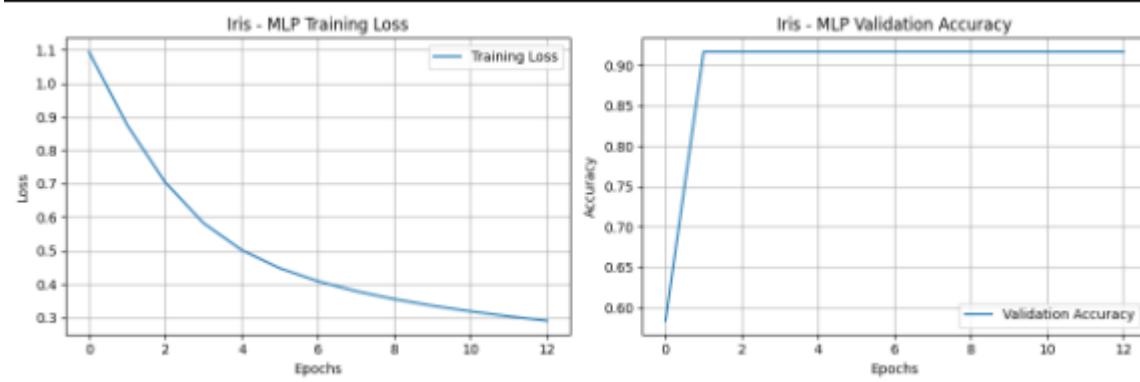
5.



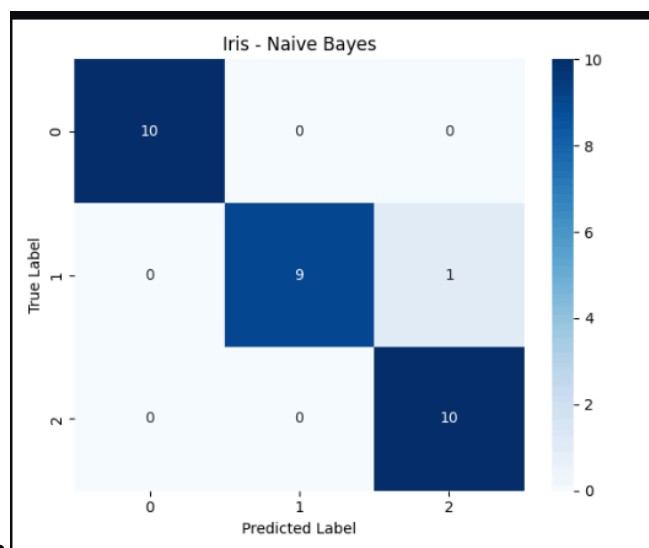
6.



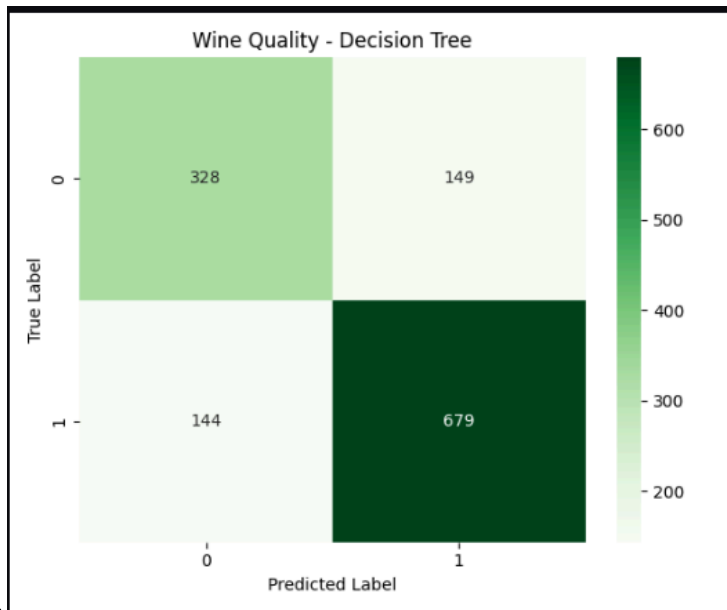
7.



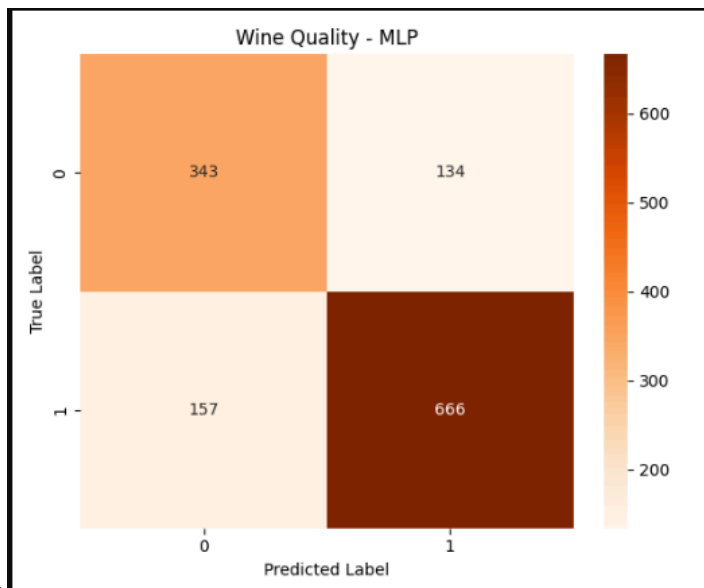
8.



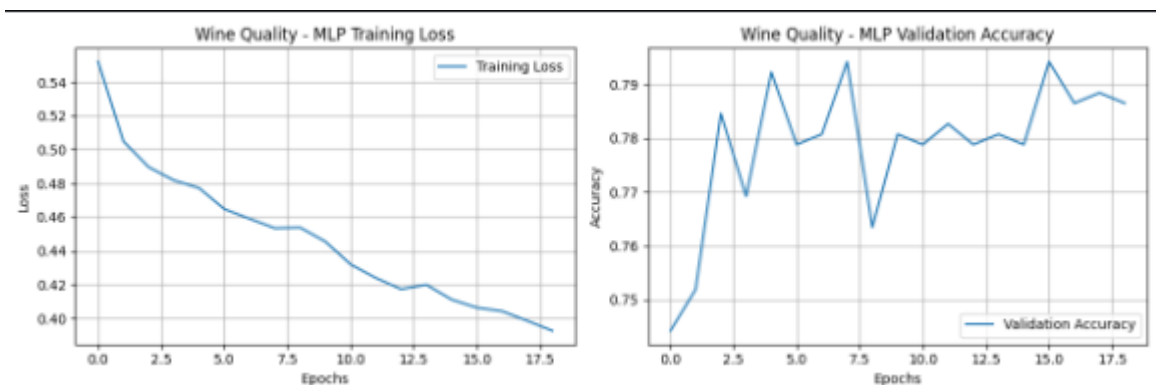
9.



10.



11.



12.

