Ten-task Iris classification capstone

Notes for SHS | Data Science | Topic 87 | 25 November 2026

Task 1: Define the project

Objective and required evidence: Three Iris classes; four numerical measurements; no target leakage.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 2: Load and validate

Objective and required evidence: load_iris(as_frame=True); 150 rows, 4 features, 50 each, no missing.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 3: Explore and visualize

Objective and required evidence: Real class counts and petal/scatter plots with centimetre units.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 4: Stratified split

Objective and required evidence: seed 87; 120 train, 30 test; counts 40 and 10 per class.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 5: Create a baseline

Objective and required evidence: DummyClassifier most_frequent; evaluate on the same held-out test.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 6: Train candidates

Objective and required evidence: Logistic Regression with scaler; max-depth-3 tree; 200-tree forest.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 7: Cross-validate

Objective and required evidence: Five stratified training folds; mean macro F1 used for selection.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 8: Evaluate final multiclass model

Objective and required evidence: Accuracy, macro F1, balanced accuracy, report and 3×3 matrix.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 9: Run binary extension

Objective and required evidence: Virginica vs other; positive proba, thresholds, precision, recall, ROC and PR.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.

Task 10: Write model card

Objective and required evidence: Provenance, versions, random state, false-classification risk, limits and reproducibility.

Python method: execute the corresponding notebook/source-code section; retain the exact printed result and source.

Common mistake: mixing training CV and final test, or claiming a fictional demonstration is actual model output. Extension: add a reasoned error analysis or check robustness.