UNIVERSITY OF ARKANSAS

Data Science

Foundations of Classification Algorithms

Jonatan Alcala, Kenney Maloney (Group 18)

Table of Contents

Preprocess

Handled missing values

- Replaced **?** with **"Unknown"** (kept rows)

Dropped redundancy

- Removed **education** (duplicate of **education-num**)

Encoded Categorical Features

Standardized numeric features

- Scaled values (age, fnlwgt, capital-gain, etc.) for balanced training

Split & saved data

- Train/test split with stratification
- Saved preprocessed train, test, validation sets

Perceptron

3-Fold Stratified CV

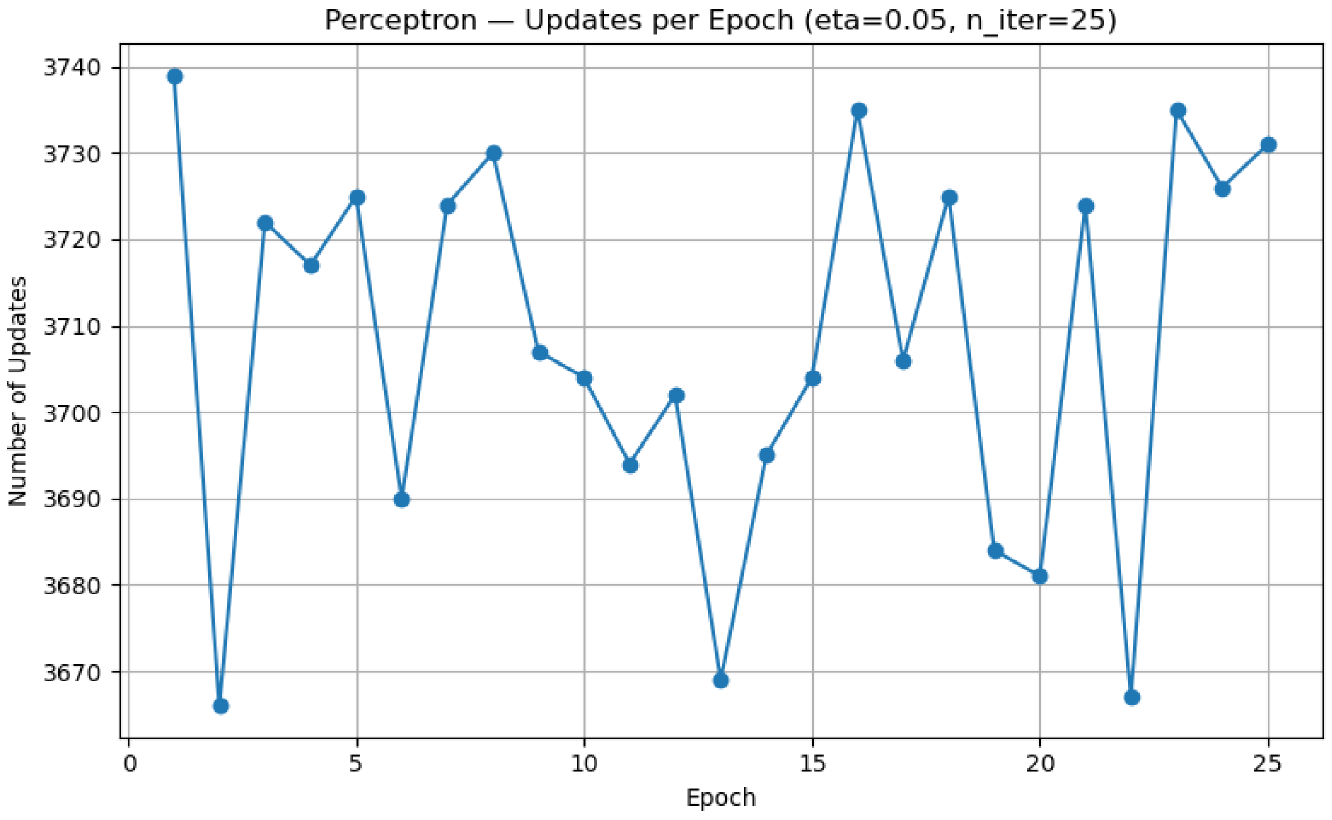
- Training grid: η ∈ {0.001, 0.005, 0.01, 0.05}, n_iter ∈ {25, 50, 100, 150}

Selected Hyperparameters

- η = 0.05, n_iter = 25
- Mean CV acc ≈ 82.7%

Results (Test)

- Accuracy: 78.75%



Adaline

3-Fold Stratified CV

Best Configurations

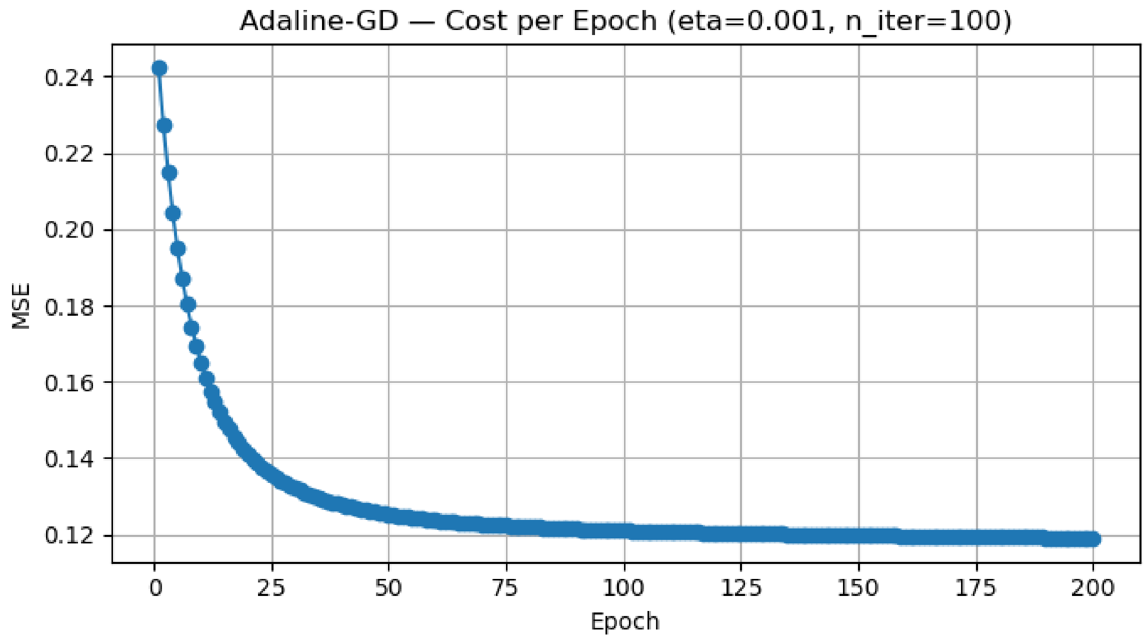
- GD: η=0.01, n_iter=200
- SGD: η=0.001, n_iter=100

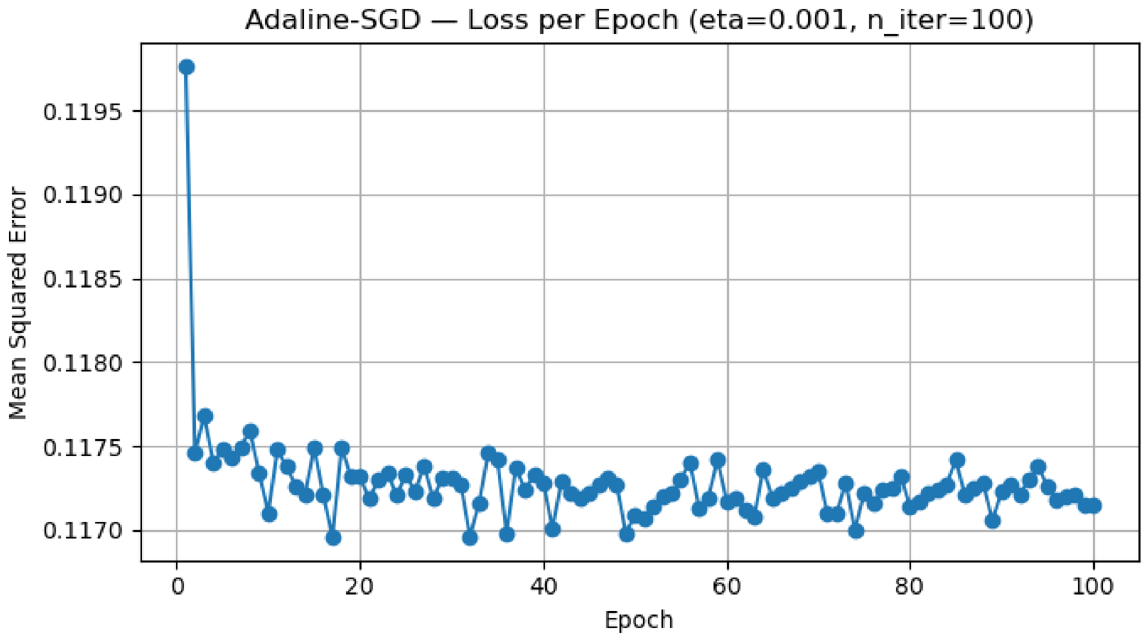
CV → Test

- GD: CV 83.44% | Test 83.10%
- SGD: CV 84.34% | Test 83.21%

Convergence

- GD: Smooth, monotonic MSE decrease → stable batch updates.
- SGD: Noisy but low MSE; faster passes; regularizing effect.





Scikit-Learn Perceptron & Adaline

Perceptron (GridSearchCV, 5-fold Stratified)

- Best configurations: elasticnet, alpha=1e-4, eta0=0.1, max_iter=1000, early_stopping=True
- Scores: CV ≈ 81.92% → Test ≈ 76.76% (some overfitting)

Adaline via SGDRegressor (squared error)

- Best configurations: alpha=1e-3, eta0=1e-4, penalty=None, max_iter=1000
- Score: Test ≈ 83.44%

Takeaways

- Perceptron: quick baseline but CV → Test drop
- Adaline: more numerically sensitive, but best Test ≈ 83% +

Logistic Regression

Before Hypertuning:

- Accuracy = 85 %
- Precision:
  - 0 = 88%
  - 1 = 73%
- Recall:
  - 0 = 93%
  - 1 = 58%

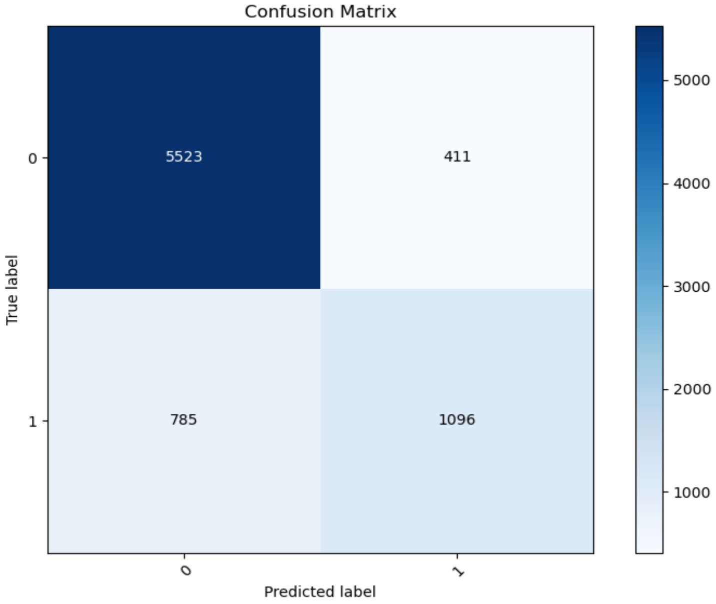
Selected Hyperparameters

- C = 0.615848211066026

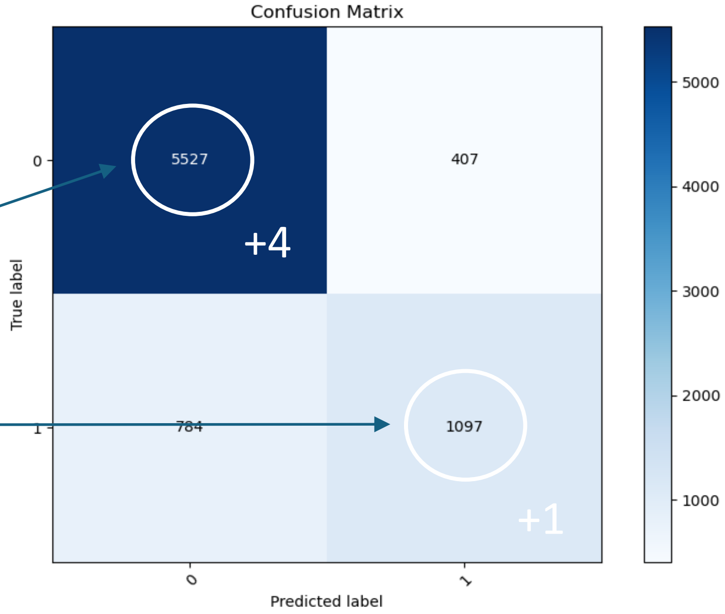
After Hypertuning:

- Accuracy = 84.8%
- Precision and Recall remain unchanged
  - Note difference in true positives and true negatives from the first model

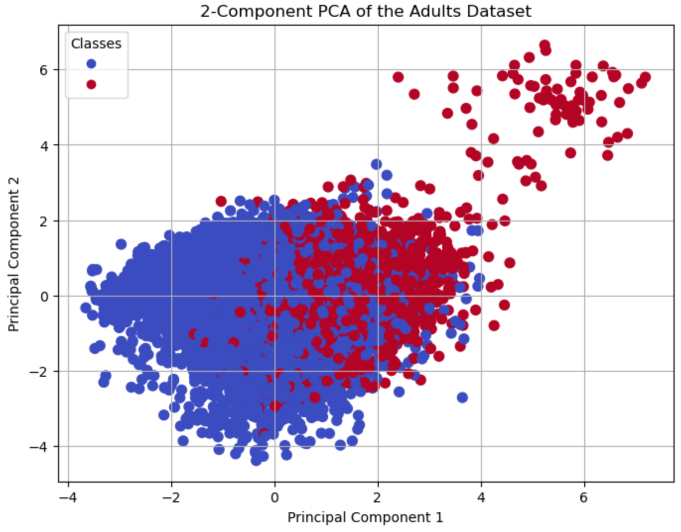
BEFORE HYPERTUNING

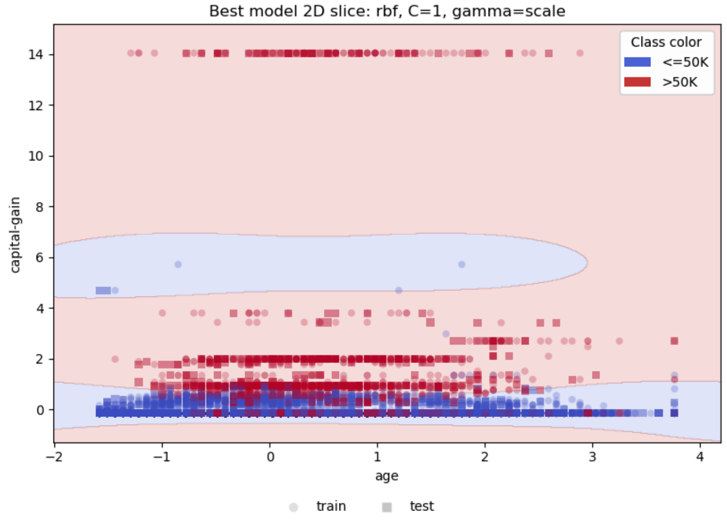


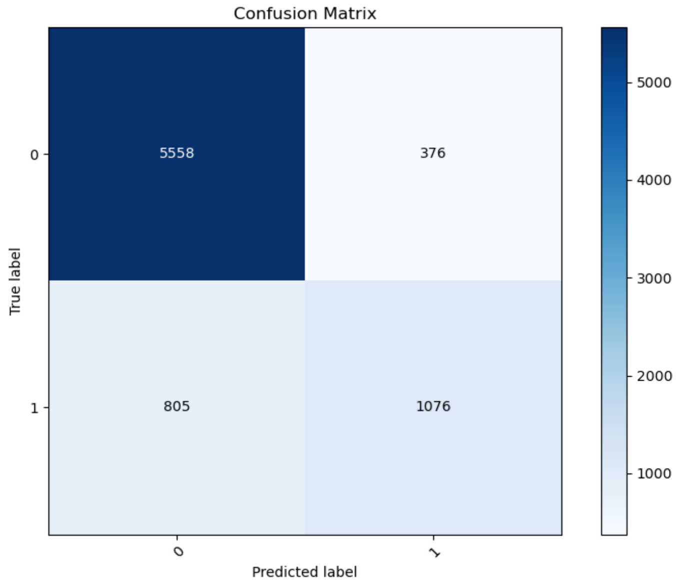
AFTER HYPERTUNING



Support Vector Machine







PCA:

- Not linearly separable with two PC's
- PC1 is the most informative
- Skipped by GridSearchCV

Best Model:

- **Kernel**: RFB
- **C:** 1
- **Gamma:** Scale
- **CV Accuracy:** 85.7%
- **Test Accuracy:** 84.9%

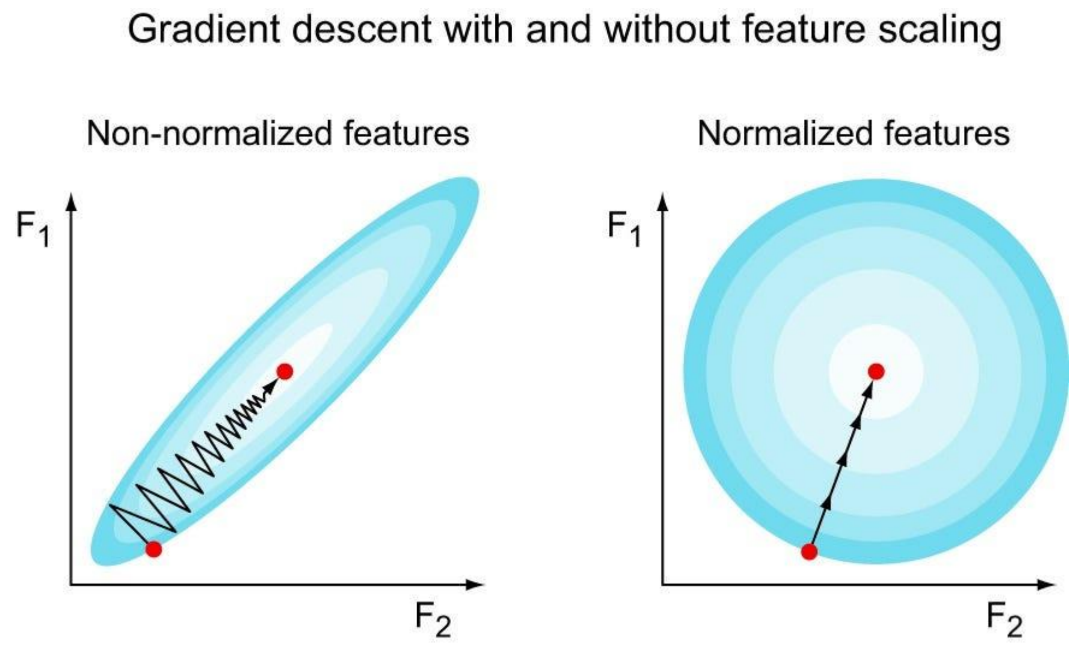
Confusion Matrix:

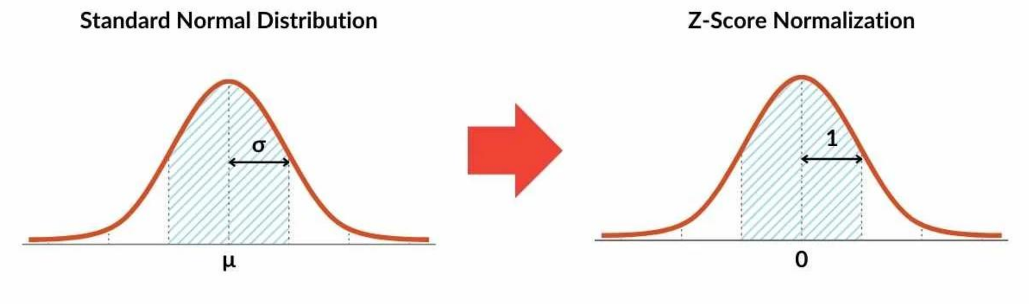
- Precision and recall remain like the logistic regression
- Slight shifts in true & false negatives

Reflection: Feature Scaling

Why is important for gradient-based algorithms?

- **Faster, stabler convergence:** one learning rate works; fewer epochs
- **Less zig-zagging:** scaling rounds the loss contours, so steps point to the minimum
- **Equal feature influence:** big-range features (e.g., capital-gain) don’t dominate small ones (e.g., age)
- **Fair regularization:** L1/L2 penalties act comparably across coefficients



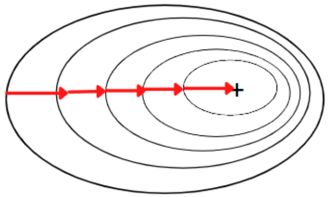


Reflection: Gradient Descent Variants

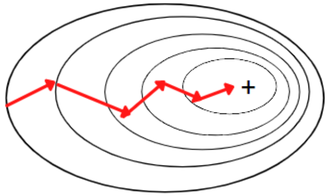
Batch vs Stochastic Gradient Descent

- **BGD vs SGD:** smooth & costly full-batch steps vs. noisy & fast single-sample steps
- **BGD** computes the gradient over *entire dataset* for each step
- **SGD** computes the gradient using a *single example*

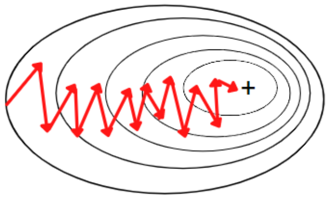
Batch Gradient Descent

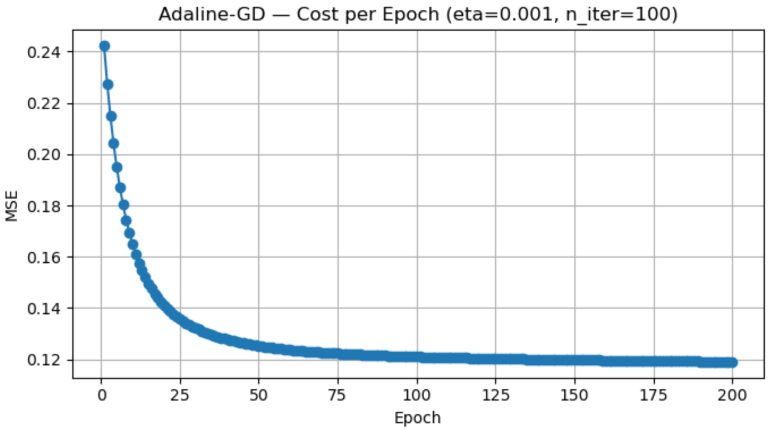


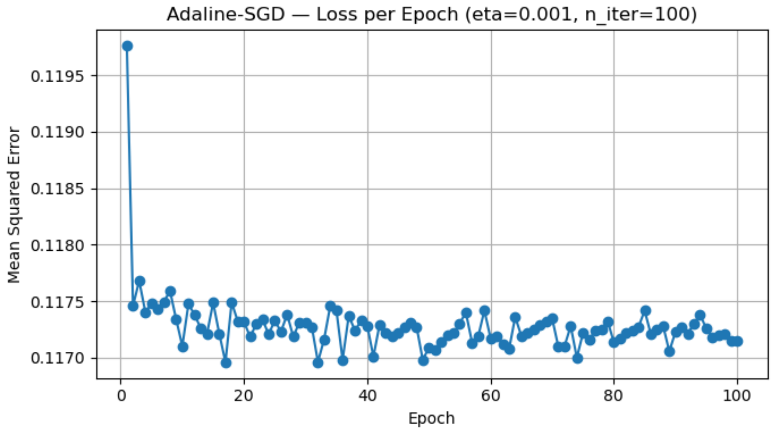
Mini-Batch Gradient Descent



Stochastic Gradient Descent







Reflection: Scikit-learn vs Book Implementations

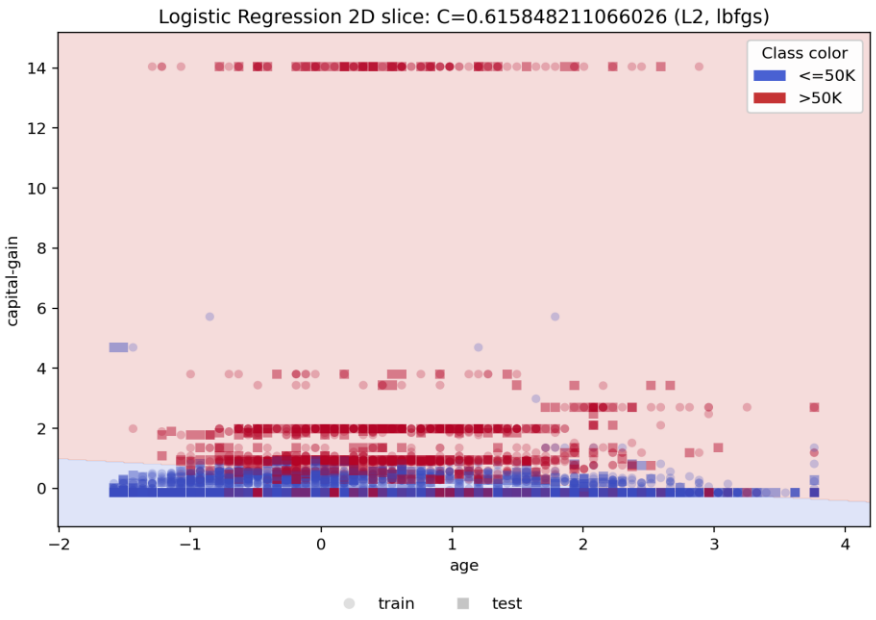
Why does scikit-learn outperform book code (Perceptron & Adaline)?

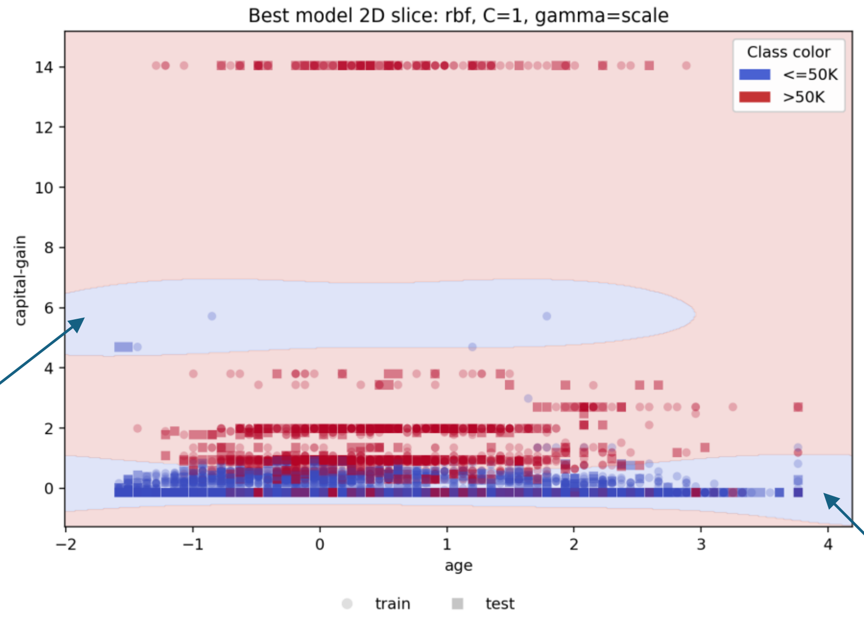
- **Compiled core (Cython):** SGD inner loops run in C (sgd_fast.pyx) → no Python-loop overhead
- **Stronger Optimizer:** LR schedules, L1/L2/ElasticNet, shuffling, early stopping, class weights, averaging
- **Space Aware Math:** accepts CSR; uses efficient safe_sparse_dot → big win on one-hot features
- **Vectorized & parallel:** BLAS + optional OpenMP for fast wall-clock convergence

*for list of references look at *Research & Concepts.md* in **docs/** folder

Reflection: Decision Boundaries

Comparing Logistic Regression & SVM





Not present in logistic

Broader tail

<u>Logistic</u>

- Can only be a linear decision boundary
  - Shows how this data doesn't fit the assumption of being linearly separable because of the overlap

<u>SVM</u>

- Fit for a radial basis function so it's more flexible than the logistic regression
- Allows for nonlinearity

Reflection: Regularization

Preventing Overfitting in Machine Learning

- Helps reduce the model's tendency to memorize the noise/outliers (overfitting)
- Makes the model less complex
  - **For Logistic/SVM:** As C decreases, the model smooths out decision boundaries and improves generalizations to potentially match better with unseen data
  - **For Adaline/Perceptron:** Depending on the regularization applied, penalties are assigned, and weights are driven lower to slow down growth. This also helps its ability to predict on unseen data.

Reflection: Impact of the *C* Parameter

Logistic Regression and Linear SVC: 0.01, 1.0, 100.0

Effect of C values:

- **Small C:** Creates a wider margin, allows some misclassifications, and keeps the model simpler.
- **Medium C:** Balances margin size with classification accuracy.
- **Large C:** Focuses on minimizing misclassification, often creating more complex boundaries that risk overfitting.

Impact on our models:

- **Logistic Regression:** Best accuracy (84.72%) at **C = 1**, suggesting the model captures patterns while tolerating some noise.
- **Linear SVC:** Best accuracy (84.71%) at **C = 100**, meaning it fits the data more tightly, but may be overfitting.

Overall: The choice of C controls model sensitivity and complexity.