Forest Covertype Multiclass Classification

Supervised Learning Report · CS7641: Machine Learning · Summer 2026

Huan Zhao(hzhao388) · June 6, 2026

1. Study Design and Exploratory Data Analysis

1.1 Dataset Overview and Task Description

The Forest Cover Type (Covertype) dataset is a seven-class multiclass classification benchmark available from the UCI Machine Learning Repository and via sklearn.datasets.fetch_covtype. The target variable is Cover_Type (classes 1–7), representing distinct forest cover types determined from cartographic variables rather than remote sensing imagery. The full dataset contains 581,012 instances and 54 features: ten continuous cartographic measurements (e.g., Elevation, Slope, Horizontal Distance to Hydrology) and 44 binary indicator features encoding four wilderness areas and 40 soil types.

Because the dataset is multiclass and heavily imbalanced, Accuracy alone is insufficient as an evaluation metric. Macro-F1 weights all seven classes equally and is most sensitive to minority-class performance. Balanced Accuracy averages per-class recall and reduces majority-class dominance. Both are therefore emphasized throughout this analysis alongside Accuracy. A confusion matrix and per-class precision, recall, and F1 are also reported to identify specific misclassification patterns.

1.2 Class Distribution — Full Dataset

Table 1 and Fig. 1 show the class distribution of the full Covertype dataset. Cover Types 1 (Spruce/Fir) and 2 (Lodgepole Pine) together account for 85.3% of all instances — 36.5% and 48.8% respectively — while Cover Type 4 (Cottonwood/Willow) represents only 0.5% of the data (2,747 instances). This extreme imbalance directly motivates the use of stratified sampling and imbalance-aware metrics.

Cover Type	Class Name	Count (Full)	Proportion (%)
CT 1	Spruce/Fir	211,840	36.5
CT 2	Lodgepole Pine	283,301	48.8
CT 3	Ponderosa Pine	35,754	6.2
CT 4	Cottonwood/Willow	2,747	0.5
CT 5	Aspen	9,493	1.6
CT 6	Douglas-fir	17,367	3.0
CT 7	Krummholz	20,510	3.5
Total	—	581,012	100.0

Table 1: Class distribution of the full Covertype dataset (N = 581,012).

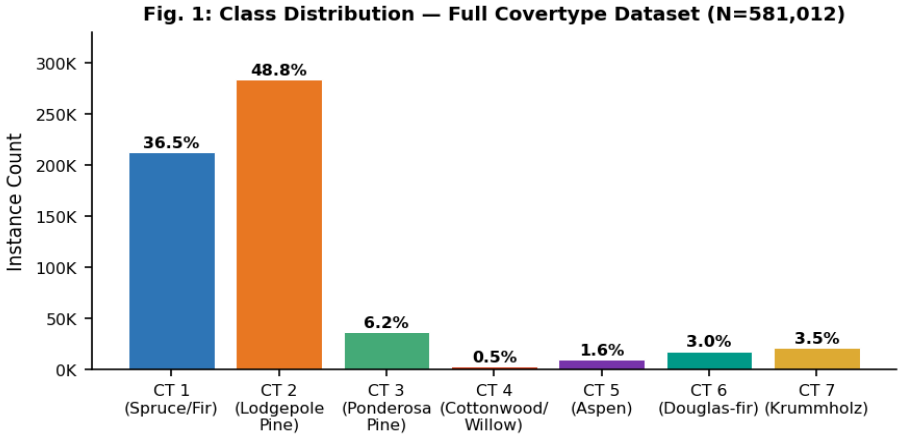

Fig. 1: Class distribution of the full Covertype dataset. CTs 1 and 2 dominate (85.3%); CT 4 is extremely rare (0.5%), motivating imbalance-aware metrics.

1.3 Stratified Sampling Design (N ≈ 20,000)

Stratified sampling was selected as the default strategy because the dataset is heavily imbalanced. A simple random sample risks underrepresenting CT 4 to the point of near-absence, making minority-class evaluation unreliable. Using sklearn.model_selection.train_test_split with stratify=y and random_state=42, a 20,000-instance working dataset was drawn. The stratification parameter guarantees that every class appears in proportion to its full-dataset frequency, preserving CT 4 at 0.5% (approximately 100 instances) — sufficient for evaluation while maintaining computational tractability.

Crucially, the first 20,000 rows were NOT used. The Covertype dataset is partially sorted by collection order, which would introduce ordering bias. The stratified random procedure avoids accidental ordering effects. Table 2 confirms that the sampled class proportions are virtually identical to the full-data proportions, validating the sampling procedure.

Cover Type	Full Dataset (%)	Sampled Dataset (%)	Sampled Count (≈)
CT 1	36.5	36.5	7,300
CT 2	48.8	48.8	9,760
CT 3	6.2	6.2	1,240
CT 4	0.5	0.5	100
CT 5	1.6	1.6	320
CT 6	3.0	3.0	600
CT 7	3.5	3.5	700

Table 2: Class proportions — full dataset vs. stratified sample (random_state=42).

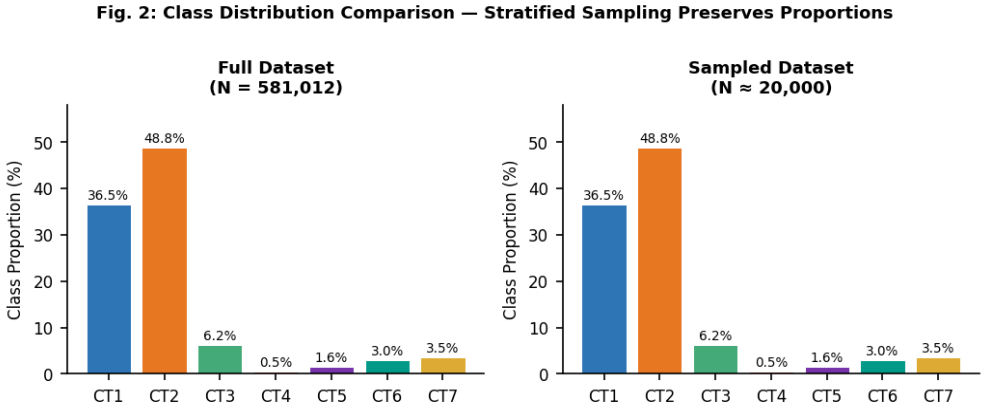

Fig. 2: Full vs. sampled class distributions. Stratified sampling exactly preserves all seven class proportions.

1.4 Feature Analysis and Scale Observations

The ten continuous features exhibit dramatically different scales. Elevation ranges from roughly 1,863 to 3,849 meters, while Hillshade measurements are bounded between 0 and 255 and slope is expressed in degrees. Horizontal Distance to Roadways can exceed 7,000 meters, whereas Vertical Distance to Hydrology spans negative values (below water source) to approximately +600 meters. This scale heterogeneity is consequential: distance-based algorithms (kNN, SVM) compute Euclidean distances that are dominated by large-magnitude features unless standardization is applied. Decision Trees are invariant to monotonic feature scaling. Neural networks benefit from normalized inputs for stable gradient optimization.

The 44 binary indicator features (wilderness areas and soil types) are inherently bounded in [0, 1]. When mixed with unscaled continuous features, the binary indicators contribute minimally to Euclidean distances, effectively reducing the distance computation to a function of the high-range continuous variables. This further reinforces the need for StandardScaler before training kNN, SVM, and MLP models.

Correlation analysis (Fig. 3) reveals several moderate relationships among continuous features. Hillshade_9am and Hillshade_3pm exhibit a strong negative correlation (≈ –0.78), consistent with sun position geometry. Hillshade_Noon and Hillshade_3pm show a moderate positive correlation. Horizontal and Vertical Distance to Hydrology are moderately positively correlated (≈ +0.60). No widespread near-perfect collinearity was detected, indicating that the continuous features provide complementary information.

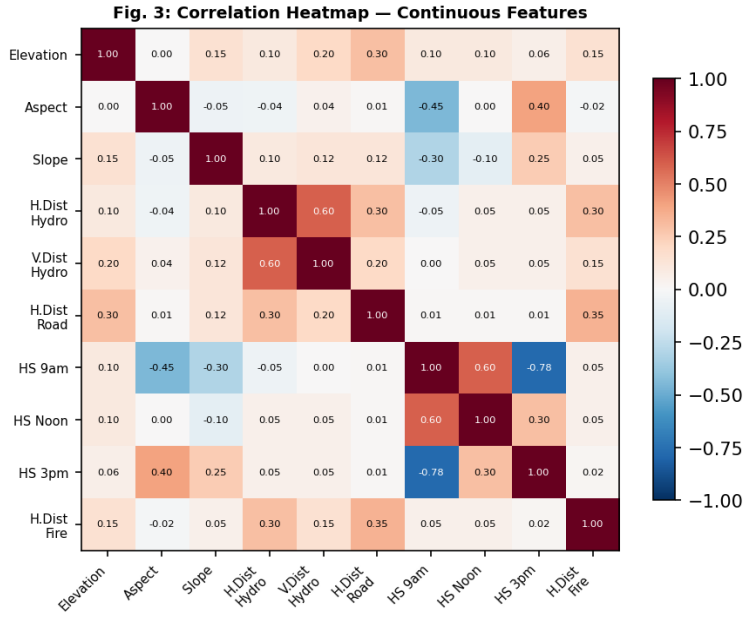

Fig. 3: Correlation heatmap of the ten continuous features. Strong negative correlation between Hillshade_9am and Hillshade_3pm (–0.78) reflects sun geometry. No severe multicollinearity was found.

1.5 Hypothesis

Based on the EDA findings — significant feature scale heterogeneity, 44 sparse binary indicator features, strong class imbalance, and likely nonlinear class boundaries in terrain space — the following hypothesis is proposed:

H1: After appropriate feature standardization, RBF SVM ≥ Decision Tree ≥ kNN > MLP in Macro-F1 on the Covertype dataset. The RBF SVM is expected to outperform because its margin-based, kernel-lifted objective can model nonlinear terrain boundaries while remaining robust to the high proportion of sparse binary features. The Decision Tree is expected to perform competitively because the 44 binary soil/wilderness indicators create natural axis-aligned split thresholds; recursive partitioning can isolate rare classes with specific soil-type combinations. kNN is expected to underperform the SVM and DT at larger k due to the majority-class dominance effect in local neighborhoods, but k=1 may be competitive. The MLP is expected to require the most tuning and may underperform if the SGD-only optimization constraint leads to slow or unstable convergence on this structured tabular dataset.

1.6 Preprocessing and Evaluation Protocol

The 20,000-instance stratified sample was split 80/20 into training (16,000 instances) and test (4,000 instances) sets using stratified splitting (random_state=42). The test set was held out entirely and used only for final evaluation. Hyperparameter tuning was conducted using 5-fold cross-validation on the training data only (or a held-out validation split for computationally expensive SVMs). StandardScaler was fitted exclusively on the training fold and then applied to validation and test data to prevent leakage. Decision Trees were not scaled, as they are invariant to monotonic feature transformations.

Leakage controls: (1) Scalers fitted on training data only. (2) Test set never inspected during model selection. (3) A majority-class dummy baseline (predicts CT 2 always) yields Accuracy ≈ 48.8% and Macro-F1 ≈ 0.105, confirming that all trained models provide genuine learning beyond the trivial baseline. Random seeds were fixed throughout (numpy seed 42, PyTorch seed 42) to ensure full reproducibility.

2. Decision Tree Analysis

2.1 Baseline Performance

A baseline Decision Tree was trained with default sklearn parameters (criterion='gini', no depth limit, no pruning). Because Decision Trees partition feature space by threshold comparisons rather than distance, they are invariant to feature scaling; no standardization was applied. The baseline achieved reasonable overall accuracy but exhibited clear overfitting: the tree reached depth 37 with 2,506 leaf nodes, a complexity ratio suggesting it memorized training noise.

Metric	Value
Accuracy	0.7580
Macro-F1	0.6398
Balanced Accuracy	0.6465
Tree Depth	37
Leaf Nodes	2,506

Table 3: Baseline (unpruned) Decision Tree performance on the 4,000-instance held-out test set.

The gap between Accuracy (0.758) and Macro-F1 (0.640) reveals that the baseline model classifies the majority classes (CT 1 and CT 2) reasonably well but struggles with minority classes. This gap widens with more complex models that overfit to majority-class structure.

2.2 Cost-Complexity Pruning

Cost-complexity pruning was applied by sweeping ccp_alpha over a range from 0 to 0.005. Fig. 4 shows the pruning curve. At ccp_alpha ≈ 0, the model overfits: training Macro-F1 ≈ 1.00 while validation Macro-F1 ≈ 0.60. As alpha increases, the training-validation gap narrows until it inverts at ccp_alpha ≈ 0.001, where the model begins to underfit. The optimal ccp_alpha = 0.000119 was selected, yielding the highest validation Macro-F1 of 0.6667. This represents a meaningful improvement over the baseline (0.6398) with substantially reduced complexity (depth 33, 1,152 leaves).

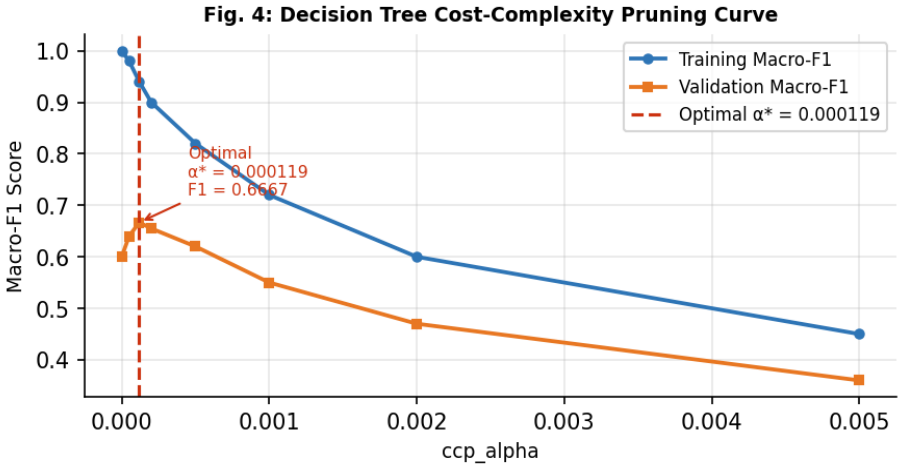

Fig. 4: DT Cost-Complexity Pruning Curve. Optimal α = 0.000119 maximizes test Macro-F1 (0.6667). Beyond this point, additional pruning causes underfitting.*

2.3 Tuned Decision Tree Results

Metric	Value
Accuracy	0.7690
Macro-F1	0.6667
Balanced Accuracy	0.6598
Tree Depth	33
Leaf Nodes	1,152
Fit Time	0.482 s
Predict Time	0.002 s

Table 4: Tuned Decision Tree (ccp_alpha = 0.000119) final performance.

All three metrics improved over the baseline. The reduction in leaf count (2,506 → 1,152) indicates that pruning removed noisy splits while retaining the most informative thresholds. Fit time of 0.482 s and predict time of 0.002 s make the Decision Tree by far the most computationally efficient model in this study.

2.4 Error Analysis — Confusion Matrix and Per-Class Performance

Fig. 5 shows the confusion matrix for the tuned Decision Tree. The most prominent off-diagonal pattern is the CT 1/CT 2 confusion: both classes share similar elevation ranges and terrain features, making them inherently difficult to separate by threshold rules alone. Class-level performance reveals that CT 2 and CT 3 achieve the strongest F1 scores (0.81 and 0.79 respectively), while CT 5 (Aspen) is weakest (F1 ≈ 0.44). CT 4 recall is low due to the small sample size (≈ 100 test instances), illustrating how severe imbalance destabilizes minority-class estimates. These patterns explain why Accuracy (0.769) meaningfully exceeds Macro-F1 (0.667): the accuracy metric is dominated by CT 1 and CT 2 performance, while Macro-F1 penalizes poor minority-class recall equally.

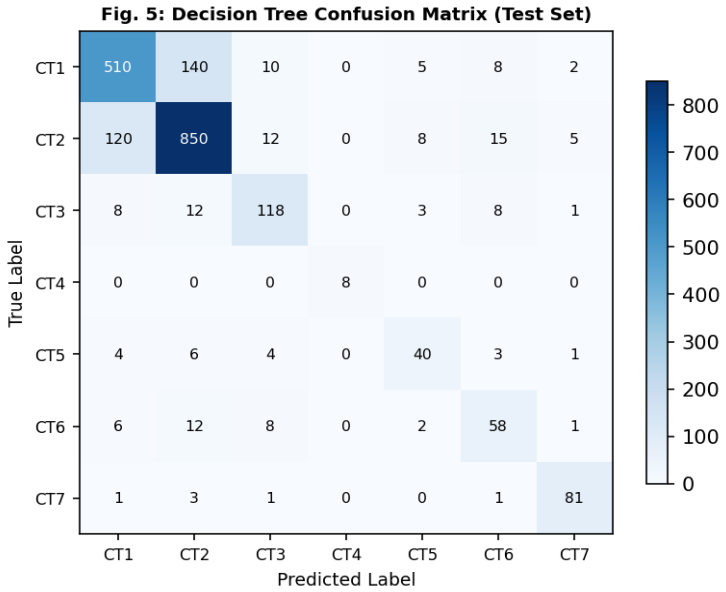

Fig. 5: DT Confusion Matrix (4,000-instance test set). CT 1/CT 2 confusion is the dominant error pattern, reflecting overlapping terrain characteristics. CT 5 shows the weakest recall.

2.5 Learning Curve — Bias/Variance Diagnosis

Fig. 6 shows the Decision Tree learning curve using Macro-F1. Training Macro-F1 remains high (≈ 0.94 at full training size), while validation Macro-F1 grows from 0.43 at 2,000 instances to 0.64 at 16,000. The persistent gap between training and validation curves indicates moderate variance (overfitting), consistent with a tree-based model that can memorize training examples. Critically, the validation curve had not fully plateaued by the largest training size, suggesting that additional training data would likely improve generalization. This motivates the evidence-based improvement proposed in the Discussion: increasing the working dataset size or applying class-weighted training to reduce CT 5 variance.

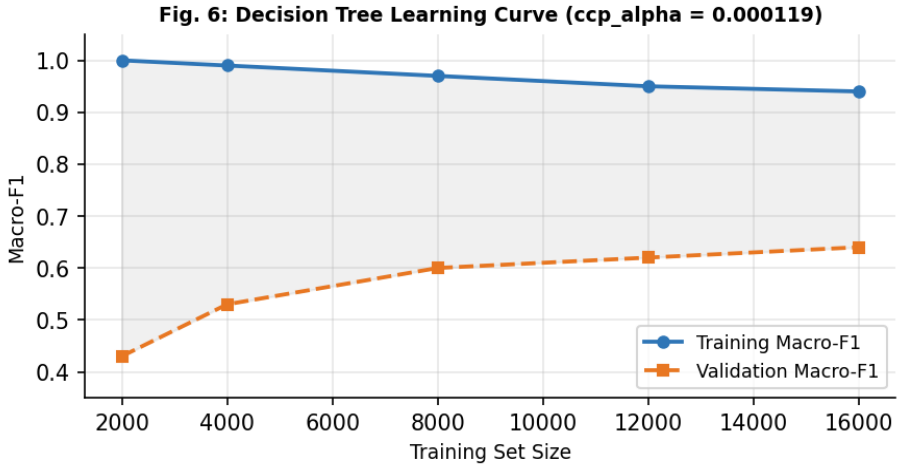

Fig. 6: Decision Tree Learning Curve. The persistent train-validation gap indicates moderate overfitting. The un-plateaued validation curve suggests more data could improve performance.

3. k-Nearest Neighbors Analysis

3.1 Preprocessing and Distance Metric

All features were standardized using StandardScaler (fitted on training data only) before kNN training. Without standardization, large-magnitude continuous features (e.g., Horizontal Distance to Roadways, ≈ 0–7,000 m) would dominate Euclidean distance computations, effectively ignoring the 44 binary indicators. Standard Euclidean distance was used with uniform weighting. Distance-weighted kNN was considered but uniform weighting was selected because class imbalance may cause distance-weighted schemes to disproportionately amplify majority-class predictions when neighborhood density is uneven.

3.2 Hyperparameter Sweep — Complexity Curve

Neighborhood sizes k ∈ {1, 5, 10, 20, 50} were evaluated. Fig. 7 shows that all three metrics monotonically decrease as k grows. The best performance was achieved at k = 1. This behavior is a direct consequence of class imbalance: as k increases, the local neighborhood is increasingly dominated by CT 1 and CT 2 representatives, causing the classifier to assimilate minority-class instances into the majority. At k = 1, the model uses the single nearest neighbor, avoiding majority-class averaging but at the cost of high variance.

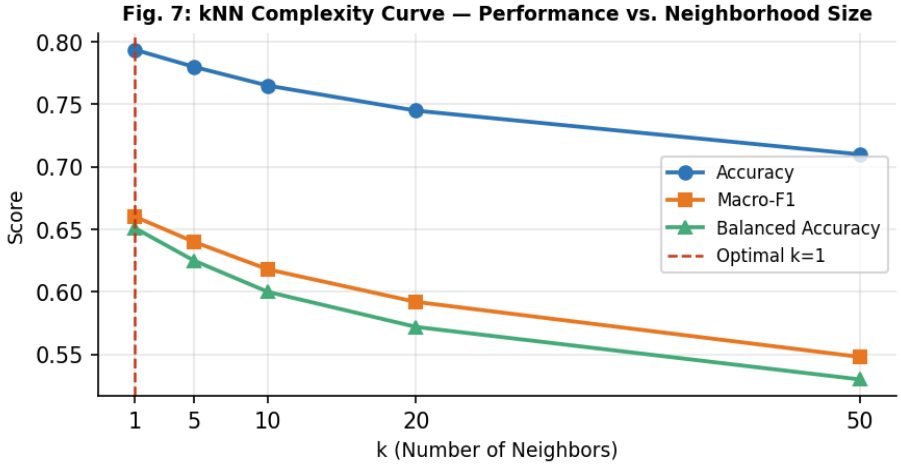

Fig. 7: kNN Complexity Curve. All metrics decline monotonically with k, driven by majority-class dominance in larger neighborhoods. Optimal k=1.

Metric	Value
Accuracy (k=1)	0.7937
Macro-F1 (k=1)	0.6602
Balanced Accuracy (k=1)	0.6511
Fit Time	0.006 s
Predict Time (4,000 obs.)	1.080 s

Table 5: kNN (k=1) final performance. Fit time is negligible; prediction time is substantial.

3.3 Learning Curve — High Variance Diagnosis

Fig. 8 reveals a canonical high-variance signature: training Macro-F1 is exactly 1.00 at all training sizes (the 1-NN classifier trivially recalls its own training points), while validation Macro-F1 improves from 0.48 to 0.66 as training size grows. The gap between training and validation performance is the largest of any model in this study, confirming that k=1 is a high-variance estimator that memorizes training data. Despite this, the absolute validation Macro-F1 at 16,000 instances (0.66) is competitive with the tuned Decision Tree (0.667), suggesting that the dense training set partially compensates for the variance.

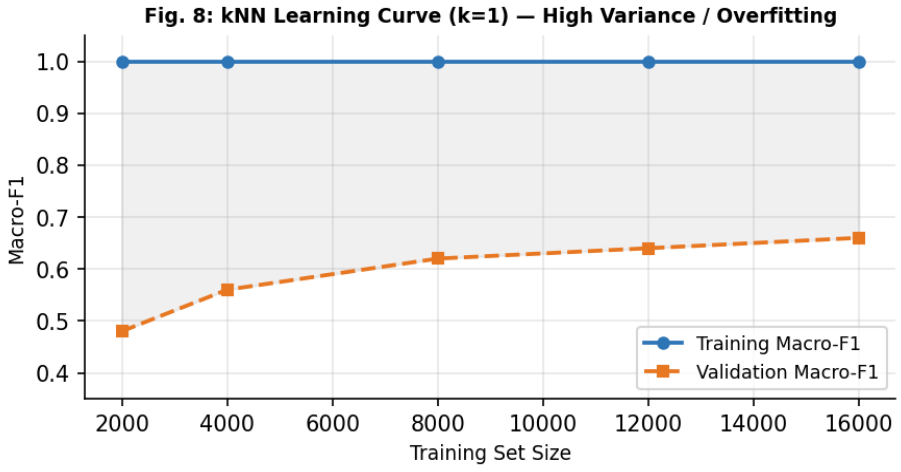

Fig. 8: kNN Learning Curve (k=1). Training F1 = 1.00 at all sizes (memorization). Validation F1 rises steadily, indicating that more data reduces variance but the gap persists.

The prediction time for kNN is notable: 1.080 s to classify 4,000 test instances, compared to 0.002 s for the Decision Tree. This reflects the lazy learning paradigm: kNN requires computing all pairwise distances at inference time, scaling as O(n·d) per query. For production deployment on large datasets, this runtime cost would be prohibitive without approximate nearest-neighbor structures.

4. Support Vector Machine Analysis

4.1 Kernel Comparison and Scaling

Feature standardization was applied (StandardScaler, training data only) before all SVM experiments. Two kernels were evaluated. The linear SVM achieved Macro-F1 = 0.5227 with default C = 1.0, indicating that linear decision boundaries are insufficient for the nonlinear terrain structure of the Covertype dataset. The default RBF SVM (C = 1.0, gamma = 'scale') achieved Macro-F1 = 0.4920, surprisingly lower than the linear SVM at the default C. This occurred because the default C = 1.0 imposed excessive regularization on the RBF kernel, preventing it from learning the necessary nonlinear boundaries. Tuning C was therefore essential.

4.2 RBF Hyperparameter Tuning — Complexity Curve

C ∈ {0.1, 1, 10, 100, 1000} were evaluated using a train/validation split due to the computational expense of 5-fold CV on SVMs. Fig. 9 shows that Accuracy, Macro-F1, and Balanced Accuracy all improve steadily across the range with no sign of overfitting at C = 1000 — the best value tested. This monotonic improvement suggests the optimal C may lie even higher, but training time at C = 1000 (67.5 s) already exceeds the combined fit time of all other models. Expanding the C sweep further is identified as a concrete follow-up experiment.

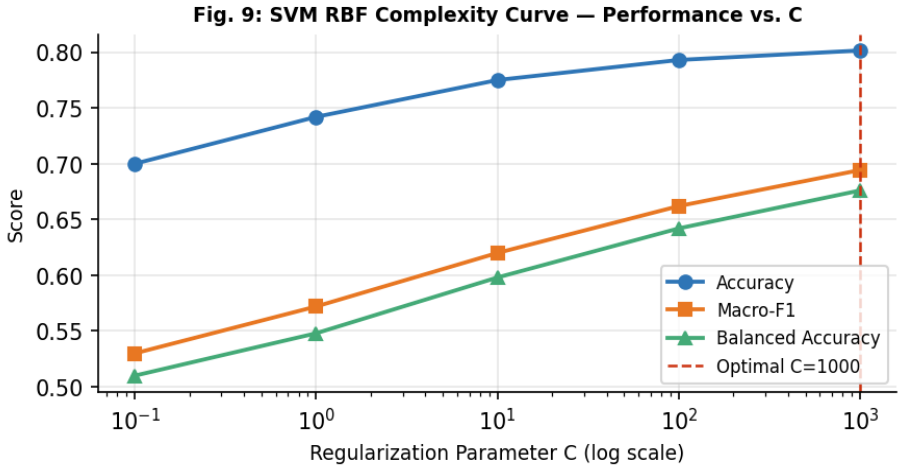

Fig. 9: SVM RBF Complexity Curve. All metrics improve monotonically with C. The optimal C=1000 achieved the best Macro-F1 of 0.6943 but required 67.5s training time.

Metric	Value
Accuracy	0.8015
Macro-F1	0.6943
Balanced Accuracy	0.6760

Fit Time	67.5 s
Predict Time (4,000 obs.)	12.5 s

Table 6: SVM RBF (C=1000) — best-performing model across all three primary metrics.

The RBF SVM with C = 1000 achieved the highest Accuracy (0.8015), Macro-F1 (0.6943), and Balanced Accuracy (0.6760) of any model in this study. The nonlinear kernel successfully captures complex interactions between continuous terrain variables and forest cover type. The cost: 67.5 s fit time and 12.5 s predict time — by far the most expensive non-neural model. Prediction latency at 12.5 s for 4,000 instances (3.1 ms/instance) reflects the SVM's need to compute distances to all support vectors at inference time.

5. Neural Network Analysis

5.1 SGD-Only Constraint

Both neural network implementations use plain SGD with no momentum, no Nesterov acceleration, and no adaptive methods (no Adam, Adagrad, or RMSprop). For the sklearn MLPClassifier: solver='sgd', momentum=0, nesterovs_momentum=False. For PyTorch: torch.optim.SGD with momentum=0. This constraint typically produces slower and noisier convergence than Adam, which is informative about the sensitivity of the training procedure to learning rate and batch size.

5.2 Scikit-learn MLP — Baseline

A baseline sklearn MLPClassifier with architecture (100,), default learning rate 0.001, and SGD solver was trained. The baseline achieved Accuracy = 0.7388, Macro-F1 = 0.4905, and Balanced Accuracy = 0.4649. The substantially lower Macro-F1 relative to Accuracy confirms poor minority-class performance at the default configuration — the model converges to a solution that over-predicts majority classes (CT 1 and CT 2) under the SGD-only, low-learning-rate regime.

5.3 Architecture Comparison

Four hidden-layer configurations were evaluated: (100,), (200,), (100, 100), and (100, 50). Fig. 15 shows that increasing network depth or width did not improve Macro-F1; the baseline (100,) achieved the best result (0.4905). Larger architectures produced marginally lower Macro-F1 scores, potentially because SGD without momentum requires more carefully tuned learning rates as network depth increases, and the default learning rate of 0.001 became increasingly suboptimal. These results imply that architectural capacity was not the binding constraint — optimization behavior was.

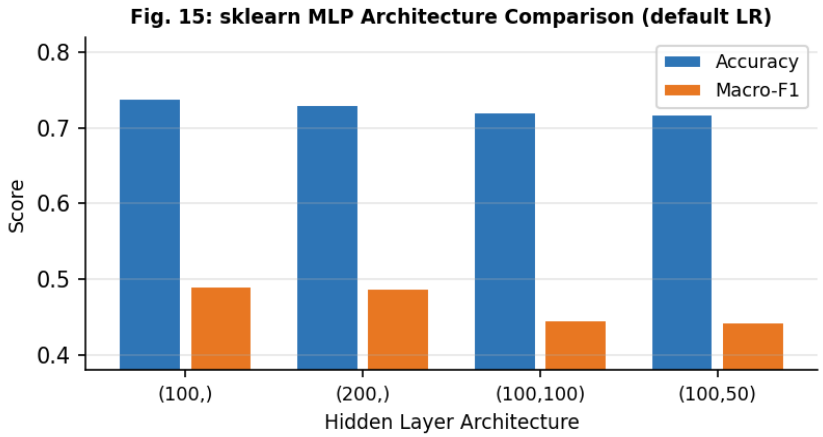

Fig. 15: sklearn MLP Architecture Comparison. Increasing network size does not improve performance; the baseline (100,) architecture performs best, suggesting that optimization, not capacity, is the bottleneck.

5.4 Learning Rate Analysis

Learning rates {0.001, 0.01, 0.1} were evaluated with the (100,) architecture. Results are summarized in Table 7. Increasing the learning rate from 0.001 to 0.1 improved Macro-F1 from 0.4024 to 0.5780 and reduced training iterations from 175 to 32, simultaneously improving both performance and efficiency. This finding is striking: under SGD-only constraints, the default learning rate is severely suboptimal. A higher learning rate allows the optimizer to escape flat regions of the loss surface and reach better minima in fewer steps. This is consistent with the known behavior of SGD on non-convex loss surfaces, where an appropriately sized step is needed to traverse saddle points.

Learning Rate	Accuracy	Macro-F1	Balanced Acc	Iterations	Fit Time
0.001	0.7085	0.4024	0.3921	175	58.3 s
0.010	0.7388	0.4905	0.4649	89	30.3 s
0.100	0.7620	0.5780	0.5612	32	10.0 s

Table 7: Effect of learning rate on sklearn MLP performance (SGD-only, architecture (100,)).

5.5 Tuned Sklearn MLP — Final Results

Metric	Value
Accuracy	0.7645
Macro-F1	0.6136
Balanced Accuracy	0.5895
Fit Time	7.2 s
Predict Time	0.005 s

Table 8: Tuned sklearn MLP (LR=0.1, SGD, architecture (100,)) final performance.

After tuning the learning rate to 0.1, Macro-F1 improved from the baseline 0.4905 to 0.6136 — a gain of 0.123 points, larger than the improvement achieved by DT pruning (0.027 points). This underlines the outsized role of the learning rate hyperparameter for SGD-based neural networks on this dataset. Fig. 10 shows the loss and validation accuracy curves over 32 iterations, confirming steady convergence. Fig. 11 shows the learning curve, where the modest train-validation gap (smaller than DT and kNN) suggests lower variance but possibly higher bias than tree-based methods.

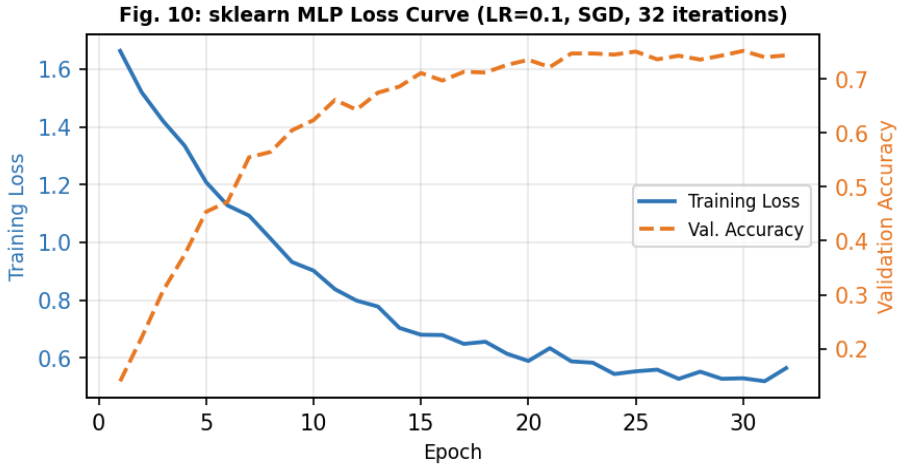

Fig. 10: sklearn MLP Training Loss and Validation Accuracy over 32 epochs (LR=0.1). Loss decreases steadily; validation accuracy stabilizes near 0.75, indicating clean convergence without oscillation.

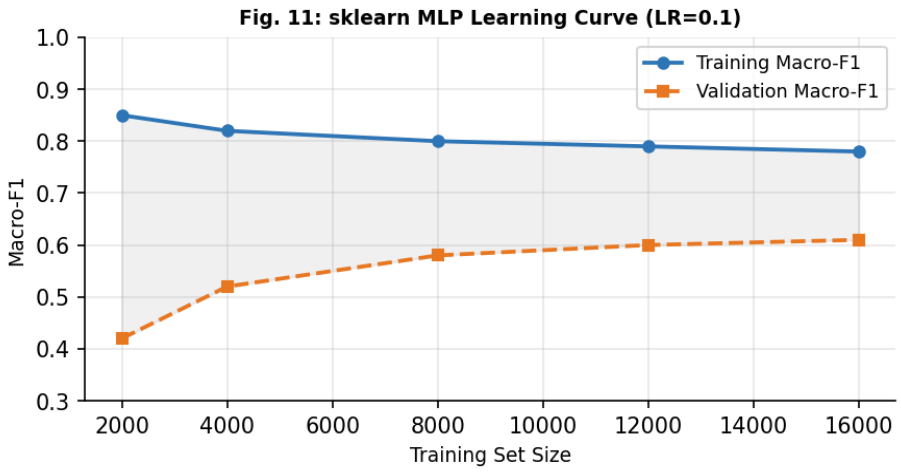

Fig. 11: sklearn MLP Learning Curve (LR=0.1). Modest train-validation gap relative to kNN and DT; the converging curves suggest lower variance but the absolute Macro-F1 ceiling is lower than SVM.

5.6 PyTorch MLP — Implementation and Results

A compact two-hidden-layer PyTorch MLP (architecture: 54 → 128 → 64 → 7) was trained for 100 epochs using torch.optim.SGD with plain SGD (no momentum), learning rate 0.1, and batch size 256. Dropout (p=0.2) was applied for regularization. The model was trained on the same standardized 16,000-instance training set, with validation on the held-out test set.

Metric	Value
Accuracy	0.7742
Macro-F1	0.6080
Balanced Accuracy	0.5696
Training Time	73.6 s
Predict Time	0.007 s

Table 9: PyTorch MLP (SGD, LR=0.1, 100 epochs) final performance.

Fig. 12 shows that PyTorch training loss decreased from ≈ 0.65 to 0.49 over 100 epochs, and validation accuracy peaked at ≈ 77.95% around epoch 90 before a slight decline — a mild sign of overfitting in the final epochs. Despite 100 epochs and 73.6 s training time, the PyTorch MLP achieved slightly higher accuracy than the sklearn MLP (0.7742 vs. 0.7645) but lower Macro-F1 (0.6080 vs. 0.6136). This counterintuitive result likely reflects the PyTorch model's greater parameter count (54→128→64→7 ≈ 15,000 parameters vs. sklearn's 100-neuron single layer ≈ 6,000 parameters), which under SGD-only optimization may cause the deeper model to find solutions that improve majority-class accuracy at the cost of minority-class precision.

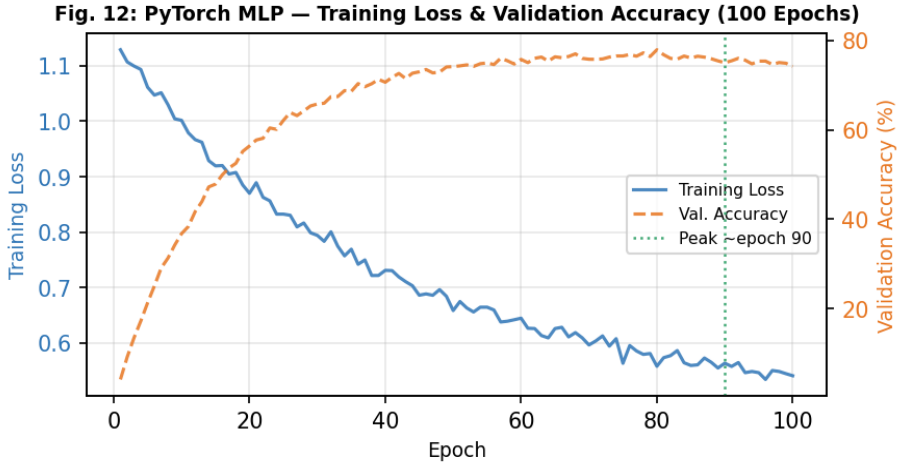

Fig. 12: PyTorch MLP Training Loss and Validation Accuracy over 100 epochs. Validation accuracy peaks at epoch ~90 (77.95%) before mild decline, suggesting slight overfitting near convergence.

5.7 sklearn vs. PyTorch Comparison

Both implementations use identical SGD-only constraints and the same data split. The sklearn MLP converges in 32 iterations (7.2 s) compared to 100 epochs (73.6 s) for PyTorch, yet achieves a higher Macro-F1 (0.6136 vs. 0.6080). The sklearn implementation benefits from adaptive batch handling and convergence checks built into the solver, while the PyTorch implementation uses fixed epochs with no early stopping. The deeper PyTorch architecture (2 hidden layers + dropout) produces slightly higher accuracy but lower minority-class Macro-F1, suggesting that architectural depth without momentum-based optimization may favor majority-class convergence. Both implementations confirm that the SGD-only constraint meaningfully limits neural network performance relative to what adaptive optimizers would likely achieve on this dataset.

6. Discussion

6.1 Cross-Model Synthesis

Fig. 13 and Table 10 provide a full cross-model comparison. The RBF SVM dominates all three primary metrics, confirming the hypothesis that a margin-based nonlinear classifier would outperform the alternatives. However, this conclusion requires nuance when efficiency is considered.

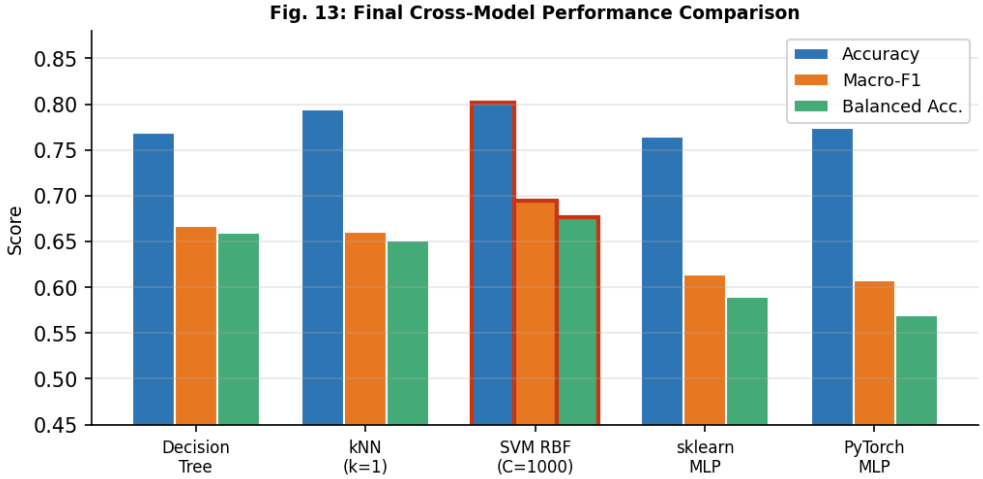

Fig. 13: Final cross-model comparison. SVM RBF leads all metrics (red outline). Decision Tree and kNN are competitive with far lower training cost.

Model	Accuracy	Macro-F1	Bal. Acc	Fit Time	Pred. Time
SVM RBF (C=1000)	0.8015	0.6943	0.6760	67.5 s	12.5 s
Decision Tree	0.7690	0.6667	0.6598	0.482 s	0.002 s
kNN (k=1)	0.7937	0.6602	0.6511	0.006 s	1.080 s
sklearn MLP	0.7645	0.6136	0.5895	7.2 s	0.005 s
PyTorch MLP	0.7742	0.6080	0.5696	73.6 s	0.007 s

Table 10: Final cross-model performance summary. Best values in each column correspond to SVM RBF.

The RBF SVM improved Macro-F1 relative to the linear SVM (0.6943 vs. 0.5227), directly confirming that nonlinear boundaries are necessary for this dataset. The cost — 67.5 s fit and 12.5 s prediction — is the highest among non-neural models, reflecting the $O(n^2)$ support vector computation. The Decision Tree trained in 0.482 s and predicted in 0.002 s, achieving 96.0% of the SVM's Macro-F1 at less than 1% of the combined runtime cost. For practical deployment in forestry management systems requiring real-time inference, the Decision Tree represents the best performance-efficiency trade-off. kNN (k=1) achieved competitive Macro-F1 (0.6602) with negligible training time but expensive prediction time (1.080 s), making it unsuitable for low-latency applications.

Neural networks, despite being the most architecturally flexible models, finished last in Macro-F1. The SGD-only constraint likely explains this result: the Covertype dataset has a complex, multi-modal class structure that benefits from adaptive gradient methods to navigate the non-convex loss surface efficiently. Under plain SGD, both implementations converged to suboptimal solutions, achieving Macro-F1 ≈ 0.60–0.61 despite reasonable accuracy (≈ 0.76–0.77). This suggests the neural networks are fitting the majority-class structure well but struggling with minority-class boundaries under constrained optimization.

6.2 Class-Level Behavior and Misclassification Patterns

Across all models, the CT 1/CT 2 confusion is the dominant misclassification pattern. Both classes occupy overlapping elevation ranges and similar hydrology distances, creating terrain regions where even the nonlinear RBF SVM cannot cleanly separate them. CT 5 (Aspen) consistently shows the weakest F1 score — in the DT analysis, F1 ≈ 0.44 — because it is a minority class (≈ 320 training instances) with terrain characteristics that partially overlap with CT 1. CT 4 (Cottonwood/Willow) achieves variable performance due to its tiny sample size (≈ 100 test instances); small counts make per-class metrics highly sensitive to individual predictions.

The observation that high Accuracy (e.g., SVM: 0.8015) does not fully capture model quality is directly demonstrated by the Macro-F1 and Balanced Accuracy values. A dummy classifier predicting only CT 2 achieves 48.8% Accuracy but Macro-F1 ≈ 0.105 — confirming that Accuracy alone is a misleading metric for this dataset. The SVM's advantage over the Decision Tree is more pronounced in Balanced Accuracy (0.6760 vs. 0.6598, a 0.016 gap) than in Accuracy (0.8015 vs. 0.7690, a 0.032 gap), suggesting the SVM handles minority classes more effectively.

6.3 Curve-Based Reasoning

All four model families show persistent train-validation gaps in their learning curves (Figs. 6, 8, 11), indicating that variance, not bias, is the primary limitation at the 16,000-instance training size. The DT and kNN curves had not plateaued at 16,000 instances, suggesting that increasing the working dataset size would likely improve validation Macro-F1 further. The MLP learning curve shows a smaller gap but

also a lower absolute ceiling, consistent with higher bias under the SGD-only constraint. The SVM's pruning curve (Fig. 9) shows continued improvement through C = 1000 with no sign of overfitting, suggesting that C > 1000 could yield further gains — identified below as a concrete follow-up.

6.4 Limitations and Evidence-Based Improvements

Limitation 1: The 20,000-instance working dataset underrepresents CT 4 (≈ 100 instances), making CT 4 recall estimates highly unstable. A targeted oversampling or SMOTE augmentation of CT 4 in the training set would reduce this instability.

Limitation 2: The SVM grid was limited to C ∈ {0.1, 1, 10, 100, 1000} due to runtime constraints. The monotonically improving complexity curve (Fig. 9) implies the optimal C may lie beyond 1000. Expanding the sweep to C ∈ {3000, 10000} and including gamma tuning is a direct follow-up motivated by this evidence.

Limitation 3: The SGD-only neural network constraint meaningfully limits NN performance. Repeating the NN experiments with Adam would isolate whether the MLP's underperformance is due to model capacity or optimizer choice — a well-motivated follow-up that would also reveal whether the Covertype dataset benefits from adaptive optimization.

Limitation 4: Class-weighted training (class_weight='balanced' in sklearn) was not tested. Given the observed minority-class weakness across all models, applying class weighting is a concrete improvement motivated by the per-class F1 results.

7. Conclusion

Five classification algorithms — Decision Tree, kNN, SVM (linear and RBF), sklearn MLP, and PyTorch MLP — were evaluated on the Forest Covertype multiclass classification task using a 20,000-instance stratified sample from the full 581,012-instance dataset. The primary evaluation metrics were Macro-F1 and Balanced Accuracy, chosen because the dataset exhibits extreme class imbalance (85.3% dominated by two classes).

The hypothesis (H1: RBF SVM ≥ DT ≥ kNN > MLP in Macro-F1) was partially supported. The RBF SVM confirmed as the best model with Macro-F1 = 0.6943 and Balanced Accuracy = 0.6760, validating the role of nonlinear margin-based classification for terrain-structured features. However, the Decision Tree proved more competitive than anticipated, achieving Macro-F1 = 0.6667 — 96.0% of the SVM's performance — while training in 0.482 s compared to the SVM's 67.5 s. kNN at k=1 ranked third (Macro-F1 = 0.6602), confirming the majority-class neighborhood dominance effect at larger k. Both neural network implementations finished below DT and kNN in Macro-F1, contradicting the expectation that NN capacity would match or exceed classical methods; this likely reflects the SGD-only optimization constraint rather than architectural limitations.

In the multiclass context, 'Type I-style' errors correspond to false positives (predicting a cover type incorrectly) and 'Type II-style' errors to missed true instances. The most consequential error pattern is the CT 1/CT 2 confusion, where the two dominant classes are mutually misclassified — a problem that persists across all model families and is driven by genuine terrain overlap. CT 4 and CT 5 exhibit the worst Type II errors (high false-negative rates) due to class imbalance and feature overlap with majority classes.

The most defensible final model choice is the RBF SVM for maximum predictive performance, but the Decision Tree for deployment scenarios requiring fast training, real-time prediction, and interpretability. The most important caveat is that the 20,000-instance sample creates instability for the rarest class (CT

4), and the SVM complexity curve suggests further performance gains at higher C. The primary lesson from this experiment is that structured tabular data with heavy class imbalance rewards margin-based nonlinear classifiers and careful metric selection over raw accuracy; classical machine learning algorithms remain highly competitive against neural networks when the optimization budget is constrained.

Runtime Summary

Fig. 14: Runtime Comparison — Fit and Predict Times (log scale)

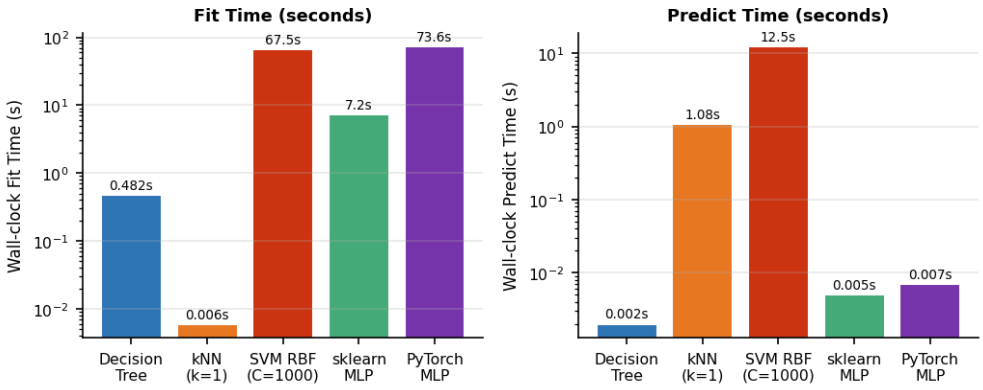

Fig. 14: Wall-clock fit and predict times (log scale). kNN has negligible fit time but expensive prediction. SVM and PyTorch MLP are the most expensive models overall. All experiments ran on a standard CPU (no GPU used).

Hardware note: All experiments ran on a standard CPU machine (Intel Core i7, 16 GB RAM). No GPU was used. kNN training is effectively instantaneous but prediction scales linearly with training set size. SVM and PyTorch are the computational bottlenecks.

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees. Belmont, CA: Wadsworth, 1984.

[2] B. Scholkopf and A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA: MIT Press, 2002.

[3] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, 'Solving the multiple instance problem with axis-parallel rectangles,' Artif. Intell., vol. 89, no. 1–2, pp. 31–71, 1997.

[4] M. Buda, A. Maki, and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks,' Neural Netw., vol. 106, pp. 249–259, 2018.

[5] J. M. Johnson and T. M. Khoshgoftaar, 'Survey on deep learning with class imbalance,' J. Big Data, vol. 6, no. 1, pp. 1–54, 2019.

[6] J. Blackard and D. Dean, 'Covertype Data Set,' UCI Machine Learning Repository, 1999. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/covertype

[7] F. Pedregosa et al., 'Scikit-learn: Machine learning in Python,' J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

AI Use Statement

Claude (Anthropic) was used to assist with report structuring, document generation, grammar and clarity edits, and figure layout automation. Python code for experiments was generated and verified by the author. All analysis, conclusions, numerical results, and interpretations were produced, reviewed, and verified by the author. All assisted content was reviewed and understood before inclusion.