Project 1: Classification Algorithms

Alif Al Hasan
Case Western Reserve University
axh1218@case.edu

Alexander Lott
Case Western Reserve University
asl139@case.edu

1 INTRODUCTION

In this project, we evaluated six classification algorithms on two datasets using 10-fold cross-validation. This report covers implementation details, performance metrics, and hyperparameter sensitivity for each algorithm.

Data Preprocessing (Common to All Algorithms): All algorithms used consistent preprocessing steps. **StandardScaler** (used by Alif) normalized features to zero mean and unit variance, while **MinMaxScaler** (used by Alex) scaled features to [0,1]. **LabelEncoder** converted categorical variables to numerical format, and data validation checked for missing values and inconsistencies.

2 ALGORITHM IMPLEMENTATIONS

2.1 K-Nearest Neighbors

2.1.1 Algorithm Workflow. K-Nearest Neighbors is a simple classification algorithm that assigns each unlabeled data point a class based on the classes of the k labeled points that are closest to it. The unlabeled point receives its new class label through a majority vote of of its k-nearest neighbors.

2.1.2 Parameter Selection.

- **k = 13, 27** - The number of nearest neighbors for Dataset 1 and Dataset 2, respectively. k-values from 1 to 30 were tested for each dataset in order to find the best values for the parameter
- **p = 2** - Default metric of euclidean distance chosen for simplicity
- **random_state = 42** - Ensures reproducibility

Table 1: K-Nearest Neighbors Performance Results

Dataset	Accuracy	Precision	Recall	F1-Score
Dataset 1	0.9701	0.9910	0.9336	0.9610
Dataset 2	0.7185	0.6670	0.3586	0.4603

2.1.3 Performance Results.

2.1.4 Analysis. K-Nearest Neighbors excelled on Dataset 1 with 97.18% accuracy and 99.13% precision. However, Dataset 2 results were significantly weaker, with one of the worst recall scores among all algorithms tested, indicating the dataset's class distributions are too complex for KNN's relatively simple approach.

2.1.5 Hyperparameter Sensitivity. The sensitivity analysis reveals that the number of k-nearest neighbors chosen for each dataset is crucial to ensuring optimal performance. k=27 appears to provide the optimal bias-variance tradeoff for Dataset 2, and it is worth noting that it is about twice as high as the optimal value of k=13 for Dataset 2 (which makes sense; Dataset 2 exhibits more complex class distributions and would thus require using more data to assign each points its proper class).

Table 2: KNN Hyperparameter Sensitivity on Dataset 2

Parameter	Accuracy	Effect
k=1	0.6516	Underfitting - too much regularization
k=15	0.7014	Moderate underfitting
k=27	**0.7208**	**Optimal balance**
k=28	0.7121	Slight overfitting begins
k=30	0.7186	Fitting unexpectedly improves

2.2 Decision Tree

2.2.1 Algorithm Workflow. This algorithm utilizes a splitting criterion of the user's choice to construct a decision tree from the training data. The decision tree can then be used to assign a class to a data point by traversing the tree according to the data point's characteristics until a leaf node is reached, at which point the data point is assigned the majority class of the leaf node's samples.

2.2.2 Parameter Selection.

- **Criterion: Gini** - Default purity measure for splitting the tree, chosen for simplicity
- **max_depth = 5, 3** - The maximum depth of the decision tree for Dataset 1 and Dataset 2, respectively. Values from 1 to 10, as well as the value "None", were tested for each dataset in order to find the parameter's optimal values
- **min_samples_split = 3, 2** - The minimum amount of samples required to split the tree for Dataset 1 and Dataset 2, respectively. Values from 2 to 12 were tested for each dataset in order to find the parameter's optimal values
- **min_samples_leaf = 3, 10** - The minimum amount of samples required to designate a node of the tree as a leaf node for Dataset 1 and Dataset 2, respectively. Values from 2 to 12 were tested for each dataset in order to find the parameter's optimal values
- **random_state = 42** - Ensures reproducibility

Table 3: Decision Tree Performance Results

Dataset	Accuracy	Precision	Recall	F1-Score
Dataset 1	0.9350	0.9279	0.9013	0.9135
Dataset 2	0.7358	0.6623	0.5175	0.5717

2.2.3 Performance Results.

2.2.4 Analysis. Decision Tree classification performed well on Dataset 1 (93.50% accuracy) and, surprisingly, on Dataset 2 (73.58%

accuracy), where it beat every other algorithm and almost overtook AdaBoost. Dataset 2's optimal tree was simpler (max_depth=3) than Dataset 1's (max_depth=5), despite Dataset 2's greater class distribution complexity; this demonstrates that Decision Tree classifiers may be a simple, yet effective way to process complex datasets.

Table 4: DT Hyperparameter Sensitivity on Dataset 2

Parameters	Accuracy	Effect
max_depth=1	0.6645	Moderate underfitting
max_depth=2	0.6968	Slight overfitting
max_depth=3	**0.7358**	**Optimal balance**
max_depth=6	0.6796	Moderate overfitting
max_depth=None	0.6687	Moderate overfitting

Parameters	Accuracy	Effect
min_samples_split=2	0.7358	Negligible underfitting
min_samples_split=3	**0.7358**	**Optimal balance**
min_samples_split=4	0.7358	Negligible overfitting
min_samples_split=5	0.7358	Negligible overfitting
min_samples_split=6	0.7358	Negligible overfitting

Parameters	Accuracy	Effect
min_samples_leaf=8	0.7336	Slight underfitting
min_samples_leaf=9	0.7314	Slight underfitting
min_samples_leaf=10	**0.7358**	**Optimal balance**
min_samples_leaf=11	0.7293	Slight overfitting
min_samples_leaf=12	0.7272	Slight overfitting

2.2.5 Hyperparameter Sensitivity. Sensitivity analysis shows that max_depth is the most critical of the three tuned parameters. It produced noticeable accuracy changes for Dataset 2, while adjusting min_samples_leaf caused only minor shifts. min_samples_split exhibited no meaningful impact on accuracy at the tested scale.

2.3 Naïve Bayes

2.3.1 Algorithm Workflow. Naïve Bayes classification is a probabilistic algorithm that assigns data points to the class with the highest posterior probability given the data points' features. The "Naïve" part comes from the algorithm's assumption that all of the data's features are independent of each other.

2.3.2 Parameter Selection.

- **var_smoothing = 1e-1, 1e-1** - for Dataset 1 and Dataset 2, respectively. Exponents from -9 to 1 were tested for each dataset in order to find the best values for the parameter
- **random_state = 42** - Ensures reproducibility

Table 5: Naïve Bayes Performance Results

Dataset	Accuracy	Precision	Recall	F1-Score
Dataset 1	0.9350	0.9536	0.8703	0.9088
Dataset 2	0.7122	0.5894	0.5515	0.5607

2.3.3 Performance Results.

2.3.4 Analysis. Naïve Bayes performs reasonably well on Dataset 1 but yields the worst recall score (87.03%) out of all of the algorithms. However, its performance on Dataset 2 is surprisingly good, yielding a relatively high accuracy score and a recall score (55.15%) that beats even that of AdaBoost. Considering that Dataset 2 has a more complex distribution of classes, these results indicate that probabilistic approaches like Naïve Bayes may be well-suited to handling such complexity.

Table 6: NB Hyperparameter Sensitivity on Dataset 2

Parameter	Accuracy	Effect
vs=1e-3	0.7077	Underfitting - too much regularization
vs=1e-2	0.7099	Moderate underfitting
vs=1e-1	**0.7163**	**Optimal balance**
vs=1e0	0.6861	Slight overfitting begins
vs=1e1	0.6537	Overfitting

2.3.5 Hyperparameter Sensitivity. Sensitivity analysis shows that var_smoothing is key for optimizing Naive Bayes classifiers. Small order-of-magnitude changes, especially when above optimal values, significantly affect accuracy. Since tested values grow exponentially without causing chaotic accuracy changes, minor perturbations to the parameter (not its magnitude) won't meaningfully impact results.

2.4 Support Vector Machine (SVM)

2.4.1 Algorithm Workflow. SVM finds the optimal hyperplane that maximizes the margin between classes. It transforms data into higher-dimensional space using kernel functions, then identifies the decision boundary with maximum distance from support vectors.

2.4.2 Parameter Selection.

- **Kernel: RBF** - Handles non-linear boundaries effectively without numerical instabilities.
- **C = 1.0** - Tested values [0.1, 0.5, 1.0, 5.0, 10.0], C=1.0 provided balanced training accuracy and generalization.
- **Gamma: 'scale'** - Auto-calculates as 1/(n_features × variance), adapting to dataset characteristics
- **random_state = 42** - Ensures reproducibility

2.4.3 Overfitting Prevention.

- Regularization parameter C=1.0 allows controlled misclassification
- Automatic gamma calculation prevents over-specialization

Table 7: SVM Performance Results

Dataset	Accuracy	Precision	Recall	F1-Score
Dataset 1	0.9719	0.9696	0.9576	0.9616
Dataset 2	0.7209	0.6267	0.4625	0.5186

2.4.4 Performance Results.

2.4.5 Analysis. SVM excelled on Dataset 1 (97.19% accuracy) due to well-separated classes and effective RBF kernel performance. Lower accuracy on Dataset 2 indicates more complex, overlapping class distributions. Feature scaling ensured equal contribution from all features.

Table 8: SVM Hyperparameter Sensitivity on Dataset 2

Parameter	Accuracy	Effect
C=0.1	0.6580	Underfitting - too much regularization
C=0.5	0.7186	Moderate underfitting
C=1.0	**0.7209**	**Optimal balance**
C=5.0	0.6947	Slight overfitting begins
C=10.0	0.6841	Overfitting

2.4.6 Hyperparameter Sensitivity. The sensitivity analysis reveals that SVM performance is highly dependent on proper C parameter selection. The optimal C=1.0 provides the best bias-variance tradeoff for Dataset 2.

2.5 AdaBoost

2.5.1 Algorithm Workflow. AdaBoost combines multiple weak learners through adaptive weighting. It starts with equal sample weights, trains a weak learner, calculates its error, adjusts sample weights (emphasizing misclassified examples), and repeats. Final predictions use weighted voting from all learners.

2.5.2 Parameter Selection.

- **Base Estimator: Decision Tree (max_depth=3)** - Shallow trees are ideal weak learners, complex enough to learn patterns while remaining "**weak**".
- **n_estimators = 4** - Tested [2, 3, 4, 8, 16], 4 estimators provided best results.
- **Learning Rate = 1.0** - Full contribution from each learner, lower rates required more estimators without performance gain.
- **Algorithm: SAMME.R** - Uses probability estimates for better performance.

2.5.3 Overfitting Prevention.

- Shallow base learners (depth=3) limit individual complexity
- Small ensemble size (4 estimators) prevents over-complexity

Table 9: AdaBoost Performance Results

Dataset	Accuracy	Precision	Recall	F1-Score
Dataset 1	0.9421	0.9378	0.9056	0.9206
Dataset 2	0.7359	0.6598	0.5188	0.5740

2.5.4 Performance Results.

2.5.5 Analysis. AdaBoost achieved the highest accuracy on Dataset 2 (73.59%), demonstrating strong adaptability to challenging patterns. The 4-estimator configuration provided optimal balance. Adaptive learning effectively focused on difficult examples, while ensemble diversity captured multiple data aspects.

Table 10: AdaBoost Hyperparameter Sensitivity on Dataset 2

Parameter	Accuracy	Effect
2 estimators	0.7207	Underfitting
3 estimators	0.7272	Slight underfitting
4 estimators	**0.7359**	**Optimal performance**
8 estimators	0.7013	Performance degradation begins
16 estimators	0.7013	Over-regularization

2.5.6 Hyperparameter Sensitivity. The analysis reveals an unusual pattern where performance peaks at just 4 estimators and then declines. This suggests that for the second dataset, a small focused ensemble works best, and additional estimators introduce unnecessary complexity that harms performance. But for the first dataset, increasing number of estimators improved overall performance.

2.6 Neural Network (MLP)

2.6.1 Algorithm Workflow. The Multi-Layer Perceptron uses forward propagation (input → hidden layers with ReLU activation → output with sigmoid), calculates Binary Cross-Entropy loss, performs backward propagation with gradient descent, and updates weights using Adam optimizer. Early stopping prevents overfitting.

2.6.2 Parameter Selection.

- **Architecture: (100, 50)** - Two hidden layers provide optimal capacity. 100 neurons capture feature interactions, 50 neurons refine representations
- **Activation: ReLU** - Efficient computation and mitigates vanishing gradients.
- **Optimizer: Adam** - Adaptive learning rates for efficient convergence.
- **Early Stopping: patience=10** - Stops if validation score doesn't improve for 10 epochs.
- **L2 Regularization: alpha=0.001** - Weight decay encourages simpler models.

2.6.3 Overfitting Prevention.

- Early stopping monitors validation performance
- L2 regularization penalizes large weights
- Adaptive learning rate prevents overshooting

Table 11: Neural Network Performance Results

Dataset	Accuracy	Precision	Recall	F1-Score
Dataset 1	0.9509	0.9833	0.8823	0.9279
Dataset 2	0.7034	0.6485	0.3875	0.4605

2.6.4 Performance Results.

2.6.5 Analysis. Neural Network achieved exceptional precision (98.33%) on Dataset 1, making it ideal when false positives are costly. Hierarchical learning effectively captured Dataset 1's patterns. Lower performance on Dataset 2, especially recall (38.75%), indicates difficulty with complex distributions.

Table 12: Neural Network Architecture Sensitivity on Dataset 2

Architecture	Accuracy	Effect
(32)	0.6061	Underfitting
(64)	0.6860	Slight underfitting
(64, 32)	0.6904	Good performance
(100, 50)	**0.7034**	**Optimal balance**
(128, 64)	0.7034	Equal accuracy but lower precision/recall
(64, 32, 16)	0.6774	Unnecessary complexity

Table 13: Neural Network Training Parameter Sensitivity (using (100, 50) architecture)

Parameter	Accuracy	Effect
SGD, learning_rate=0.001	0.6645	Poor convergence
Adam, no early stopping	0.6925	Overfitting
Adam, early stopping	**0.7034**	**Optimal**
Adam, alpha=0.0 (no L2)	0.7012	Slight overfitting
Adam, max_iter=500	0.7034	No improvement

2.6.6 Hyperparameter Sensitivity. The sensitivity analysis reveals that both architecture design and training parameters significantly impact neural network performance. The (100, 50) architecture with Adam optimizer and early stopping provides the best balance between model capacity and generalization for Dataset 2.

3 PERFORMANCE COMPARISON

Table 14: Performance Comparison on Dataset 1

Algorithm	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbors	0.9701	0.9910	0.9336	0.9610
Decision Tree	0.9350	0.9279	0.9013	0.9135
Naive Bayes	0.9350	0.9536	0.8703	0.9088
SVM	0.9719	0.9696	0.9576	0.9616
AdaBoost	0.9421	0.9378	0.9056	0.9206
Neural Network	0.9509	0.9833	0.8823	0.9279

Table 15: Performance Comparison on Dataset 2

Algorithm	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbors	0.7185	0.6670	0.3586	0.4603
Decision Tree	0.7358	0.6623	0.5175	0.5717
Naive Bayes	0.7122	0.5894	0.5515	0.5607
SVM	0.7209	0.6267	0.4625	0.5186
AdaBoost	0.7359	0.6598	0.5188	0.5740
Neural Network	0.7034	0.6485	0.3875	0.4605

3.1 Key Findings

- **KNN:** Exceptional Dataset 1 performance (97.01% accuracy, 99.10% precision) but worst Dataset 2 recall (35.86%). Optimal k doubled from 13 to 27 for complex boundaries.
- **Decision Tree:** Unimpressive performance for Dataset 1 (93.50%) but unexpectedly good performance for Dataset 2 (73.58%). Optimal tree depth for Dataset 2 (3) is roughly half the optimal depth for Dataset 1 (5).
- **Naive Bayes:** Best Dataset 2 recall (55.15%). Optimal var_smoothing=1e-1 prevented zero probabilities and handled class imbalance.
- **AdaBoost Superiority:** Best Dataset 2 performance (73.59%) with unconventional 4-estimator configuration showing strong adaptability.
- **SVM Consistency:** Robust cross-dataset performance (97.19% and 72.09%) with C=1.0 and RBF kernel.
- **Neural Network Trade-offs:** Excellent Dataset 1 precision (98.33%) but poor Dataset 2 recall (38.75%).
- **Dataset Impact:** Complex Dataset 2 favors ensemble, decision tree, and probabilistic methods over distance-based approaches.

3.2 Hyperparameter Sensitivity Summary

- **KNN:** Highly sensitive to k selection (k=27 for Dataset 2 vs. k=13 for Dataset 1).
- **Decision Tree:** max_depth most influential; min_samples_leaf has minor effect; min_samples_split shows negligible impact.
- **Naive Bayes:** Optimal var_smoothing=1e-1 balanced smoothing effectively.
- **AdaBoost:** Unusual optimal at 4 estimators (versus typical 50+); performance declined beyond this.
- **SVM:** Highly sensitive to C; under-regularization (C=0.1: 65.80%) and over-regularization (C=10.0: 68.41%) both degrade performance.
- **Neural Network:** (100, 50) architecture with Adam and early stopping proved optimal.

4 CONCLUSION

AdaBoost proved most robust on Dataset 2 (73.59%) with an unconventional 4-estimator configuration, while SVM showed impressive consistency across both datasets (97.19% and 72.09%). Neural Networks and KNN achieved exceptional precision on Dataset 1 but poor recall on Dataset 2. Naive Bayes achieved highest recall (55.15%) on Dataset 2, while Decision Trees exhibited disappointing performance on Dataset 1 but surprisingly strong performance (73.58%) when processing Dataset 2's more complex features.

Hyperparameter analyses revealed dataset-dependent optimal configurations: Dataset 2 required double the k-neighbors (k=27 vs. k=13) and a shorter binary tree (max_depth=3 vs. max_depth=5), while SVM performance degraded rapidly with improper C values. The results suggest that complex datasets favor ensemble methods (AdaBoost), well-tuned decision trees, and probabilistic approaches (Naïve Bayes), while well-separated data allows flexibility. Empirical validation and careful tuning are essential for optimal performance.