3x4: Testing 3 types of Classifiers on 4 Different Datasets

Abstract

This study investigates the performance of three supervised machine learning classifiers—Decision Tree, K-Nearest Neighbors (KNN), and Random Forest—across four distinct datasets, focusing on hyperparameter optimization and varying data splits. Each dataset was divided into training and testing splits of 20/80, 50/50, and 80/20, with hyperparameters tuned for each split using grid search and cross-validation. Three trials were conducted for each classifier on each split to ensure robust evaluation. The study examines the classifiers' accuracy and consistency across datasets and splits, with Random Forest demonstrating superior performance on average. This comprehensive evaluation highlights Random Forest's robustness and adaptability across diverse datasets and varying training sizes, providing insights into the impact of data splits and hyperparameter tuning on model performance.

1. Introduction

Supervised machine learning has emerged as a pivotal approach in artificial intelligence, enabling computers to learn from labeled data and make predictions or decisions based on new, unseen data. This machine learning method relies on a dataset containing input-output pairs, where the model learns to map the inputs to the correct outputs through various algorithms. The effectiveness of supervised learning is heavily influenced by the quality and quantity of labeled data available for training. However, obtaining precise labels can be time-consuming and costly, particularly in domains requiring domain expertise for annotation (Arachie & Huang, 2021).

In this study, we investigate the performance of three prominent supervised classifiers—Decision Tree, K-Nearest Neighbors (KNN), and Random Forest—across varying datasets and hyperparameter configurations. By focusing on hyperparameter optimization and analyzing different training-test splits, we aim to provide insights into how these factors influence model performance in supervised learning contexts.

2. Method

2.1. Learning Algorithms

Decision Trees (DT): I varied key hyperparameters, including max_depth, min_samples_split, and min_samples_leaf. The parameter grid includes max_depth values from 1 to 5, min_samples_split values of 2, 5, and 10, and min_samples_leaf values of 1, 2, and 4. I employed GridSearchCV with 3-fold cross-validation to identify the optimal hyperparameters for each split. The sklearn package was utilized for this classifier, which primarily implements the CART algorithm, using the entropy criterion for node splitting. Random_state was set to None for the classification, and shuffle was set to True.

KNN: I optimized the KNN classifier by varying the neighbors (n_neighbors) in a parameter grid ranging from 1 to 26. Before training, the feature data was scaled using MinMaxScaler to ensure that all features contributed equally to distance calculations. I employed GridSearchCV with 3-fold cross-validation to identify the optimal n_neighbors value, evaluating performance based on accuracy. When running the model, random_state was set to None, and I set the shuffle parameter to True for random sampling for the splits.

Random Forest: I varied key hyperparameters, including n_estimators, max_depth, min_samples_split, and max_features. The parameter grid consisted of n_estimators set to 50, 100, and 150; max_depth values of None, 10, 20, and 30; min_samples_split values of 2 and 5; and max_features values of 1, 8, and 20. I employed GridSearchCV with 3-fold cross-validation to identify the optimal hyperparameter configurations for each split. During classification, random_state was set to None, and shuffle was set to True.

2.2. Performance Metrics

The performance metric used to evaluate the performance of the three models was the classification accuracy for each training split, averaged over 3 trials to minimize noise. For each trial within the training splits, performance was reported using a classification report, which shows accuracy, precision, recall, f1-score, and weighted average of all the metrics.

2.3. Datasets: Data Preprocessing

Fertility:
The target variable (Y) for the fertility dataset was highly imbalanced, with significantly more positive labels than negative ones. This imbalance posed a challenge for the binary classification models, as it sometimes resulted in no negative predictions during training. This issue made it impossible to calculate a meaningful precision score. To address this problem, I applied the Synthetic Minority Over-sampling Technique (SMOTE) with random_state=42. This method generated synthetic data to balance the target labels, ensuring the models had a more equitable distribution of classes to train on.

Heart_disease:
This dataset was straightforward to clean, with only a few missing values that I dropped.

Car:
This dataset originally had only categorical features, so I used OrdinalEncoder to transform them into floats that were usable for analysis. For the 'doors' column, I assigned '5more' to just the float 5, and for the 'persons' column, I replaced the 'more' value with a float 6. There were no missing values in this dataset.

Breast_cancer:
For this dataset, I converted the target labels (M, B) to -1, and 1, respectively. I also had to drop a couple of columns for my analysis.

3. Experiment

For each classifier, I evaluated performance using three different train/test splits: 20/80, 50/50, and 80/20. I conducted 3-fold cross-validation on the training data within each split to identify the optimal hyperparameters. Originally, I did cross-validation once for every classifier, but I adjusted my method and decided to cross-validate parameters within each split for each classifier. This ended up yielding better results. Table 2 shows the parameters that scored the best for the classifiers for the heart_disease dataset. I chose not to include the parameters for every dataset in this paper for the sake of space, but they are shown in my code and reported for every classifier and dataset.

See Figures 1-3 for visualizations of the parameter testing for the heart_disease dataset (the rest of the visualizations are in the code provided). After determining the best hyperparameters, I ran three trials for each partition and reported the performance of the best-performing model. Table 1 summarizes the average classification accuracy achieved by each classifier across the four datasets.

Dataset	DT	KNN	RF
heart_disease	0.72	0.80	0.83
fertility	0.83	0.83	0.87
car	0.94	0.97	0.98
breast_cancer	0.92	0.96	0.96
Average	0.85	0.89	0.91

Table 1: Average classification accuracy over 3 trails for each classifier

Classifier	Training split	Best parameters
DT	20/80	{'max_depth': 1, 'min_samples_leaf': 1, 'min_samples_split': 2}
DT	50/50	{'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 2}
DT	80/20	{'max_depth': 2, 'min_samples_leaf': 1, 'min_samples_split': 2}
KNN	20/80	'n_neighbors': np.int64(13)
KNN	50/50	'n_neighbors': np.int64(11)
KNN	80/20	'n_neighbors': np.int64(13)
RF	20/80	{'max_depth': None, 'max_features': 1, 'min_samples_split': 2, 'n_estimators': 50}
RF	50/50	{'max_depth': 30, 'max_features': 1, 'min_samples_split': 2, 'n_estimators': 100}
RF	80/20	{'max_depth': 10, 'max_features': 1, 'min_samples_split': 2, 'n_estimators': 150}

Table 2: The best parameters found after hyperparameter testing for each training split for the heart_disease dataset.

Classifier: partition	heart_disease	fertility	car	breast_cancer
DT: 20% train	0.7157	0.725	0.9388	0.9013
DT: 50% train	0.7539	0.851	0.9595	0.931
DT: 80% train	0.6833	0.87	0.95	0.9327
KNN: 20% train	0.797	0.778	0.9677	0.9474
KNN: 50% train	0.805	0.845	0.96875	0.9614

KNN: 80% train	0.806	0.877	0.9739	0.9766
RF: 20% train	0.8235	0.87	0.9597	0.9561
RF: 50% train	0.83	0.886	0.9864	0.9567
RF: 80% train	0.85	0.944	0.995	0.9649

Table 3: Comparison of different partitions for each classifier and dataset.

4. Conclusion/Discussion

4.1. Drawbacks and Potential Improvements of the Experiment

One main drawback of this experiment was the significantly different data frame sizes. In the Caruna, Niculescu-Mizil paper, they selected the same number of cases for training for each dataset, which made their results more significant and generalizable. Because two of my chosen datasets were significantly smaller than the larger one, I wasn't able to experiment in this way and instead did the same partitions of the data for each classifier and dataset to introduce some generalizability. The disparity in sample sizes can affect the robustness and generalizability of the results, as smaller datasets may lead to overfitting or less reliable performance estimates compared to larger datasets. In future research, it would be beneficial to explore methods for balancing dataset sizes, such as resampling techniques or selecting datasets with comparable sizes to facilitate more meaningful comparisons.

Using the SMOTE technique on one of my datasets was acceptable for this experiment because I was only using it to test the classifier. However, this technique is risky when the reported results affect someone's well-being.

4.2. Some issues with classifier implementation

While testing hyperparameters for the Random Forest classifier, as outlined in the Caruana and Niculescu-Mizil paper, I noticed a significant increase in runtime—up to 3 minutes for parameter testing. To address this, I reduced the number of parameters being tested. Initially, I planned to evaluate the feature set [1, 2, 4, 6, 8, 12, 16, 20], but this configuration was taking too much time to run. Instead, I selected a smaller subset, [1, 8, 20], which still represented a diverse range of features while significantly reducing runtime.

Similarly, for the KNN, I tried to test 26 random values of k ranging from 1 to the size of the training set, like in the reference paper, but that was taking too long to run. As a solution, I tested 26 different k's in the form of np.arange(1,27). I would like to run a more diverse parameter testing with more k values to choose from, but it was too computationally expensive for this experiment.

4.3. Discussion

It’s worth noting that the car and breast_cancer datasets consistently achieved the highest classification accuracies across all classifiers and training splits, with performance exceeding 90% for all the problems. This superior performance can be attributed to the datasets’ inherent characteristics. Unlike the fertility and heart disease datasets, which include more abstract or imbalanced features, the car and breast_cancer datasets comprise well-defined, structured, and categorical attributes, making them highly predictable and easier for classifiers to distinguish between classes. Moreover, for the car dataset, preprocessing steps such as ordinal encoding ensured compatibility with machine learning algorithms without introducing significant noise or distortions.

Additionally, the car and breast_cancer datasets’ relatively large sizes and balanced class distributions likely contributed to the model’s ability to generalize effectively, even with smaller training splits. The lack of missing values and clean data structure further streamlined model training, minimizing overfitting risks and ensuring high test accuracy.

Another notable observation is the Random Forest classifier’s robust performance, especially on the car dataset. This can be explained by its ensemble nature, which integrates multiple decision trees to mitigate overfitting—a common challenge in datasets with clear patterns or low complexity. While KNN and Decision Tree also performed well, their simpler structures may have limited their scalability on datasets with more variability, such as fertility or heart disease.

4.3. Conclusion

The Random Forest classifier demonstrated the best overall performance across the four datasets, achieving an average classification accuracy of 91%, as shown in Table 1. This result highlights the robustness of ensemble methods, which tend to excel in scenarios with diverse and complex data distributions. The K-Nearest Neighbors (KNN) classifier performed better than expected, with an average accuracy of 89%, indicating that the simplicity of KNN can still yield competitive results under appropriate parameters. The Decision Tree (DT) classifier, while still effective, had the lowest average accuracy at 85%, reflecting its susceptibility to overfitting and its comparatively less sophisticated modeling capability. My results were consistent with the Caruana, Niculescu-Mizil paper, which also ranked RF as the best classifier, with KNN as second, and DT as third.

Interestingly, in the case of the DT classifier in the heart_disease dataset, the 80/20 train-test split did not perform as well as the 50/50 split. The same happened (marginally) in the car dataset. This can likely be attributed to a smaller training set in the 80/20 configuration, which may fail to capture enough variability in the data, leading to suboptimal model performance (it struggled to generalize). However, as evidenced in Table 3, the 80/20 split generally yielded the best performance overall. This trend suggests that having a larger training set typically provides the model with more data to learn from, improving its ability to generalize to unseen test samples. The exceptions to this trend may stem from certain datasets or classifiers requiring a more balanced train-test split to optimize performance.

These findings have significant implications for model generalizability to other datasets and real-world applications. The performance of the classifiers—Decision Tree, K-Nearest Neighbors, and Random Forest—varied across datasets, suggesting that their effectiveness depends on specific dataset characteristics. For instance, Random Forest consistently outperformed the other classifiers, likely due to its ability to manage complex feature interactions and mitigate overfitting through ensemble learning. These attributes make Random Forest a strong candidate for applications in domains with high-dimensional data or where feature interactions play a crucial role, such as healthcare or finance (Azar et al.).

5. Bonus Points

I think I deserve bonus points for this project because I used 4 datasets, and trained my 3 classifiers on each. I also got exceptional results for my RF for the breast_cancer and car datasets, and some of my datasets took a significant amount of data wrangling to figure out. I also performed parameter cross-validation on every single split for every classifier, not just every classifier. This gave me more accuracy on my tests and overall improved my results.

Figures

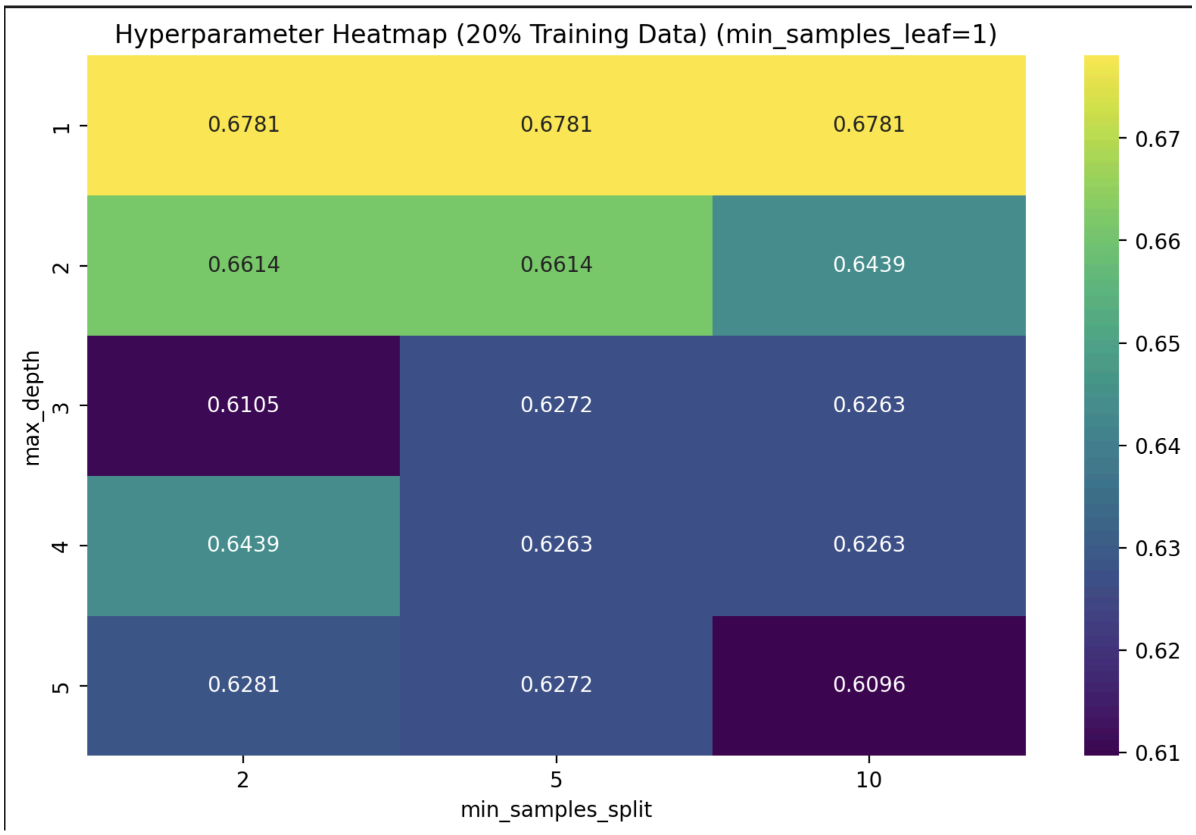

Fig 1: heatmap of DT hyperparameter training for the 20/80 training split for heart_disease dataset

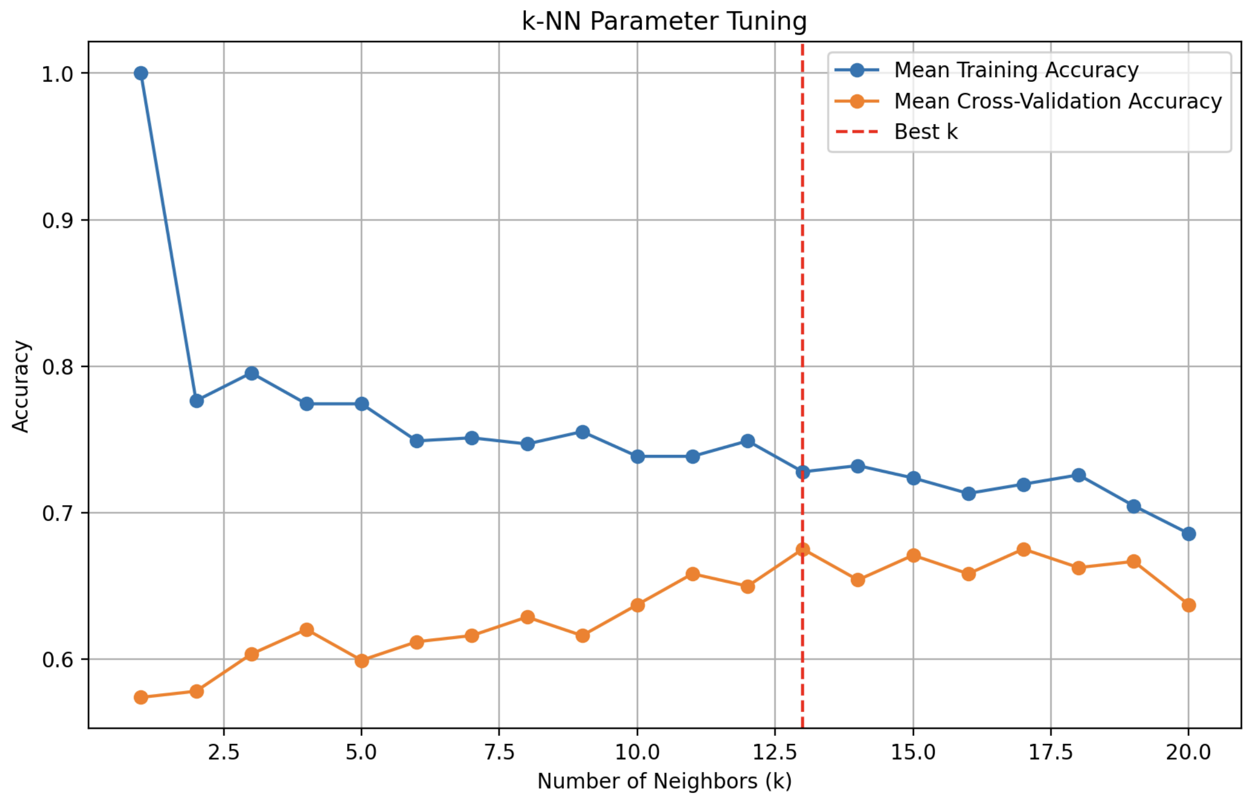

Fig 2: Line graph for KNN hyperparameter training for the 20/80 split in heart_disease dataset

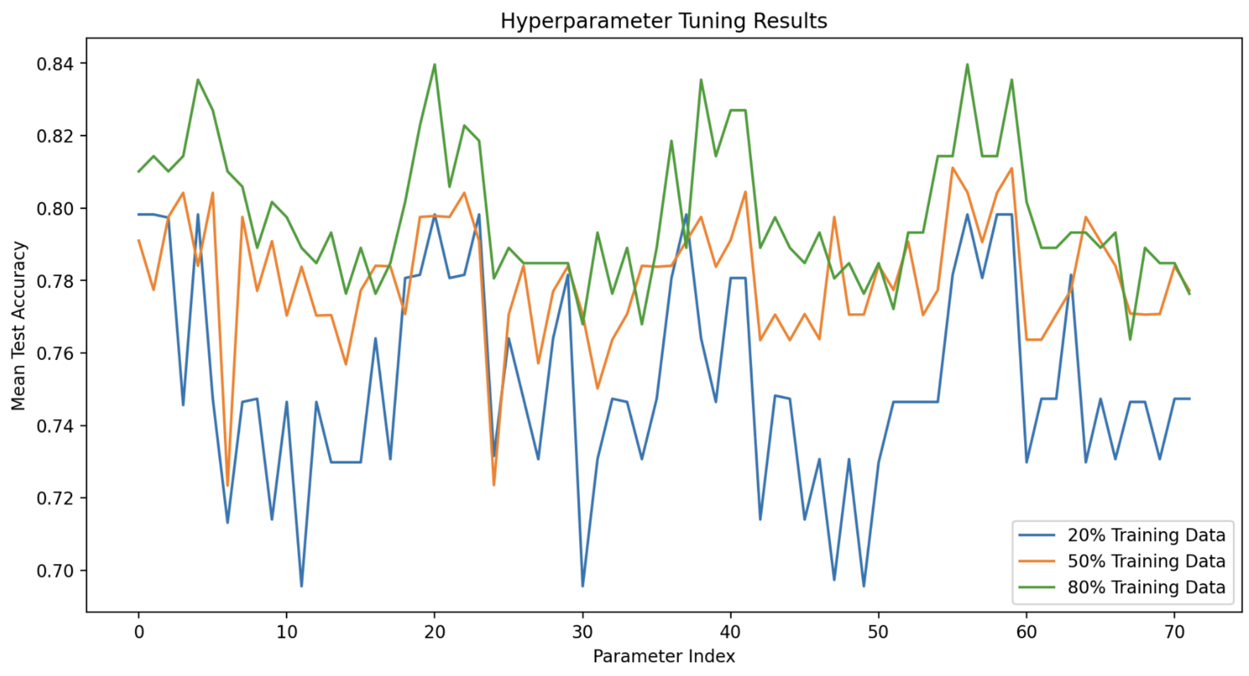

Fig 3: Stacked line graph for RF hyperparameter testing for all three splits in heart_disease dataset

6. References

1. Azar, A. T., Elshazly, H. I., Hassanien, A. E., & Elkorany, A. M. (2013, November 14). *A random forest classifier for lymph diseases*. Computer Methods and Programs in Biomedicine. https://www.sciencedirect.com/science/article/pii/S0169260713003751
2. Caruana, R., & Niculescu-Mizil, A. (2006, June 25). *An empirical comparison of supervised learning algorithms: Proceedings of the 23rd International Conference on Machine Learning*. ACM Other conferences. https://dl.acm.org/doi/10.1145/1143844.1143865
3. Caruana, R., & Niculescu-Mizil, A. (2006, June 25). *An empirical comparison of supervised learning algorithms: Proceedings of the 23rd International Conference on Machine Learning*. ACM Other conferences. https://dl.acm.org/doi/10.1145/1143844.1143865