Comparing Supervised Algorithms on the Metric of Accuracy Across Datasets and Training and Test Splits

Oishani Bandopadhyay

1. Abstract

This report presents an evaluation of five different classifier algorithms, Logistic Regression, KNN, Random Forest, Decision Tree, and SVM, across 4 datasets. The metric used for comparison is test accuracy and cross validation means accuracy. Comparisons across algorithms, train and test splits, and datasets and problems are presented.

2. Introduction

An empirical comparison of machine learning models is a valuable exercise in understanding the progress of machine learning research. The paper by Caruana and Niculescu-Mizil, 'An Empirical Comparison of Supervised Learning Algorithms', serves as a reference for this report, and is referred to as the reference paper (Caruana, n.d.).

This paper discusses 5 classifier models, also discussed in this reference paper, in detail (Caruana, n.d.): Logistic Regression, KNN, Random Forest, Decision Tree, and SVM. They are evaluated for test accuracy and cross evaluation means accuracy, to compare the performance of each algorithm on 4 datasets of varying sizes. While the reference paper uses several performance metrics, the focus on test accuracy and just these 5 algorithms helps to understand this metric in greater depth and comprehensively evaluate these algorithms (Caruana, n.d.).

Based on the findings in this reference paper, including averages across performance metrics, the overall performance of each of these 5 classifiers would be (Caruana, n.d.):

1. Random Forest (RF)
2. Support Vector Machine (SVM)
3. K-Nearest Neighbor (KNN)
4. Decision Tree (DT)
5. Logistic Regression (LR)

The findings in this report differ slightly from those in the reference paper, with the following general order of accuracies (Caruana, n.d.):

1. Random Forest (RF)
2. Logistic Regression (LR)
3. Support Vector Machine (SVM)
4. K-Nearest Neighbor (KNN)
5. Decision Tree (DT)

These results mostly align with the findings of the reference paper, and the differences are explained in detail under the Results section in Experiment (Caruana, n.d.).

3. Method

Since not only classifiers, but also variations in accuracy across different train/test ratios and multiple datasets were measured, there was a defined procedure to collect information from each run of the code.

First, the machine learning algorithms, described in greater detail below, were

implemented using functions. The functions were written to clearly output the accuracies across 3 different train/test splits:

- 20% training, 80% testing
- 50% training, 50% testing
- 80% training, 20% testing

Then, 4 datasets were selected from the UCI Machine Learning Repository, and cleaned to make sure they did not include any Null values. The cleaned data was then loaded into pandas dataframes and manipulated further. Some of the dataframes had categorical variables, which were converted to numerical variables using Label Encoding for simplicity. The function fit_transform was used for all categorical columns (if any), resulting in entirely numerical data.

The target variables were selected depending on the problem, and, if required, converted into binary variables so that the task for each algorithm would consistently be a binary classification task.

Once the datasets were cleaned, each of the algorithm functions were run on the datasets, and the X and y values were determined by which columns would be used and features and which column was to be predicted as the target variable. Functions written to optimize the hyperparameters of each of the variables were run before running the model functions, to give optimized results for each model.

Each model was run on each dataset thrice, in 3 trials. The averages of the training, testing and cross-validation accuracies across these 3 trials were reported and noted down for comparison. These scores are tabulated in the code, and test accuracy and cross validation mean accuracy are tabulated across datasets and splits.

Overall, 5 different classifiers were used to evaluate differences in the 3 accuracy metrics described above. Accuracy was measured using the accuracy_score function in sklearn, which computes subset accuracy: the set of labels predicted for a sample that exactly match the corresponding set of labels in the true target. The resulting accuracies across each of the 5 classifiers is presented in Tables 9 and 10.

The following learning algorithms were used, due to their varying levels of complexity and efficiency on datasets of different sizes and number of features:

Logistic Regression

This is one of the most efficient algorithms used, and performs very well on binary classification tasks such as the ones in this report, by using probabilistic calculations.

KNN Classification

The k-nearest neighbors algorithm is slightly more computationally intensive, and works well for recommendation systems. It has higher accuracy in cases where similarity between data points is significant, by classifying data based on its k-nearest neighbors in feature space.

Random Forest

This is a significantly versatile but computationally intensive algorithm useful in classification taks, especially multi-dimensional data. It works by building multiple decision trees, thus reducing overfitting.

Decision Tree

This algorithm is more simplistic and works well on smaller datasets for classification problems by using recursion and class labels made based on features.

SVM

The support vector machine is the most computationally intensive algorithm tested in this report, which performs well on classification tasks in high-dimensional spaces by mapping data into

higher dimensions and finding the hyperplane where data points are best separated.

Training and Testing Errors During and After Cross-Validation With Respect to Hyperparameters

Training and testing errors were initially measured during cross-validation. Multiple hyperparameters were tested for each of the models to get a general idea of ideal hyperparameters for each model on each dataset.

To capture results concisely and accurately, functions were written to use GridSearchCV to find the optimal hyperparameters for a given dataset and a given model. This was then used to run each model with its ideal hyperparameters, and the cross validation mean accuracy for each of the datasets is reported in Tables 5-8 .

The hyperparameters that were optimized for each algorithm are listed as follows:

Logistic Regression

- C (regularization inverse)
- max_iter (maximum iterations for solvers to converge)
- penalty (None, L2 or L1 penalty)
- solver (algorithms used in optimization problem)

KNN:

- n_neighbors (numer of neighbors)
- weights (weight function used)

Random Forest:

- max_depth (maximum depth of tree)
- min_samples_leaf (minimum number of samples required to be at leaf node)
- min_samples_split (minimum number of samples to split an internal node)
- n_estimators (numer of trees in forest)

Decision Tree:

- criterion (function to measure quality of a split)
- max_depth (maximum depth of tree)
- min_samples_leaf (minimum number of samples required to be at leaf node)
- min_samples_split (minimum number of samples to split an internal node)

SVM:

- C (regularization parameter)
- gamma (kernel coefficient)
- kernel (specified type of kernel to be used)

4. Experiment

Defining Algorithm and Optimization Functions

Using similar functions allowed consistent reporting of results and easier tabulation of the accuracies. Functions were defined one by one, using the sklearn libraries. They were easy to call for each of the datasets one by one, and worked without issues. Random Forest and SVM were very computationally intensive, resulting in some modifications such as default hyperparameters being used for SVM. Besides taking long to run and timing out, there were no major issues.

Cleaning and Preprocessing Data

Cleaning the data was fairly straightforward, since most of the datasets did not have any null values.

Some datasets had far more categorical variables than others which needed to be encoded, scaled, and then normalized. These were subprocesses added to the preprocessing of each dataset before running the algorithms. This process of scaling the data was needed for the Student Performance, Online Shoppers, and Student Dropout datasets. The Student

Performance dataset had a lot of categorical features which were more challenging to encode.

Running Algorithms on Datasets

Some of the datasets were harder to work with than others for varying reasons - the Online Shoppers dataset was very large and took a long time to run the SVM and Random Forest algorithms. Due to several timeouts, the hyperparamater optimization functions had to be modified. The Student Performance dataset did not get particularly high accuracies, which might have been due to the high number of catgorical features, the encoding process being somewhat simplistic, and the dataset being small. The Online Shoppers dataset also did not get very high accuracies, perhaps due to the unique choice of the target variable as 'Weekend', or the inability to optimize parameters for SVM due to how computationally intensive the optimization process was. The wine dataset, although the smallest one used, achieved great accuracy across each of the models, and shows a clear trend in terms of accuracy compared across the models. The student dropout dataset also achieved relatively high accuracies, probably due to its medium size and number of features.

Results

In terms of reporting the results, each of them were printed across the Jupyter Notebook. At the end of running the file, the results were compiled as the accuracies into Tables 1-10 for easier reference and discussion.

Datasets and Problems

The 4 datasets used varied in size and number of features. This explains the variation in accuracy results across the datasets.

Student Performance

648 data points, 34 features

The Student Performance dataset was a medium-sized dataset with several features, which might explain the lower accuracies across algorithms in predicting the target variable G3. There were 2 columns highly correlated with G3, G1 and G2, which would have been very helpful for making predictions, but the results would have been biased, and therefore not as useful.

Interestingly, the highest test accuracy, 75.69%, was from the 50-50 split using Random Forest, closely followed by Logistic Regression at 75.08%. This was probably due to overfitting the complex data in the 80/20 split. Random Forest had the highest testing accuracies overall, which was consistent with the findings of the reference paper (Caruana, n.d.). Logistic Regression did surprisingly well on this dataset, probably due to the small size of the dataset and the large number of categorical features. Decision Tree performed poorly, with 66.15% test accuracy, as expected according to the reference paper (Caruana, n.d.).

Table 1. Student Performance Test Accuracy

Algorithm	80-20 Split	50-50 Split	20-80 Split
Random Forest	72.31%	75.69%	69.42%
Logistic Regression	71.54%	75.08%	70.19%
SVM	70.00%	74.77%	69.04%
KNN	66.15%	68.92%	63.27%
Decision Tree	66.15%	66.15%	56.15%

Wine

177 data points, 14 features

This was the simplest dataset with the best classification results, due to the clean data, relatively small number of features, and all the features being numerical. Class was the target variable, converted into a binary variable for prediction.

With the 80-20 split, all 4 algorithms besides KNN reached 100% test accuracy, including Decision Tree as well. This was a good

way to ensure that the models were being optimized and running correctly for the datasets, since this was the simplest dataset of the 4 total.

Table 2. Wine Test Accuracy

Algorithm	80-20 Split	50-50 Split	20-80 Split
SVM	100.00%	98.88%	98.60%
Random Forest	100.00%	97.75%	95.10%
Logistic Regression	100.00%	97.75%	100.00%
Decision Tree	100.00%	89.89%	91.61%
KNN	97.22%	93.26%	96.50%

Online Shoppers

12,329 data points, 18 features

This was the largest dataset used, which led to some complications in classification. The target variable used was 'Weekend', where True and False were codified for the binary classification problem. There were 3 categorical variables that were converted to numerical using label encoding.

On this dataset, results were observed initially without scaling the numerical data, a procedure followed for all the datasets to improve accuracy. Then, the same algorithms and splits were run without scaling. All scaling was done using StandardScaler(). KNN and SVM had slightly higher accuracy after scaling the data, which was expected. The scaling did not make a significant difference in accuracy for Logistic Regression, Random Forest, or Decision Tree.

One complication that arose with this dataset was the size. Due to the large size of the dataset, it was not computationally feasible to optimize the hyperparameters for SVM within a reasonable time using GridSearchCV(), which was used for optimization in all the other datasets. So, default hyperparamters for SVM were used given the size of the dataset (kernel: rbf or radial basis function, C: 1.0, gamma: scale).

SVM outperformed the other algorithms on the 50/50 split with a 77.36% testing accuracy, closely followed by Random Forest with a 77.05% accuracy. These results were consistent with the findings of the reference paper, as accuracy dipped for Logistic Regression, KNN, and Decision Tree (Caruana, n.d.).

An interesting phenomenon happened with the splits, the 50/50 split and 20/80 splits generally had higher test accuracy than the 80/20 split. To investigate this further, the cross-validation mean accuracies were inspected, and they generally had higher accuracy for more training data. The 50/50 split and 20/80 split had marginally higher cross validation mean accuracy for Logistic Regression, KNN, and SVM than the 80/20 split.

This might be due to the higher simplicity of these models compared to tree-based models, as well as the increased size of test data helping the accuracy of these models in the 50/50 split. Random Forest and Decision Tree both had highest accuracy for the 80/20 split, probably due to these models being more sensitive to the size of training data. So, the differences in test accuracy across the splits is probably explained by the complexity and size of the data and variations in the test size leading to falsely higher accuracies in the 20/80 split for the Random Forest.

Table 3. Online Shoppers Test Accuracy

Algorithm	80-20 Split	50-50 Split	20-80 Split
SVM	76.32%	77.36%	77.31%
Random Forest	76.80%	77.05%	77.18%
Logistic Regression	74.86%	76.06%	76.90%
KNN	74.45%	75.20%	75.05%
Decision Tree	68.65%	69.99%	68.54%

Student Dropout

4,423 data points, 37 features

This was a fairly large dataset, and had higher accuracies compared to the other 2 larger datasets used. The target variable was mapped to 1 for Graduate or Enrolled, and 2 for Dropout, to make it a binary classification problem. Multiple

categorical variables were encoded using Label Encoding.

Logistic Regression gave the best results on this dataset, with an 87.23% accuracy on the 20/80 split and an 86.33% accuracy on the 80/20 split. This was followed by SVM, Random Forest, KNN, and Decision Tree. As mentioned in the reference paper, logistic regression can sometimes be the best model for a particular metric, in spite of being generally less accurate on average (Caruana, n.d.). In this case, the dataset might have had low complexity, and also the probabilistic measure of a binary outcome, which are both handled very well using Logistic Regression. The order of the other 4 algorithms is fairly consistent with the findings of the reference paper as well (Caruana, n.d.).

Table 4. Student Dropout Test Accuracy

Algorithm	80-20 Split	50-50 Split	20-80 Split
Logistic Regression	86.33%	86.71%	**87.23%**
SVM	85.65%	85.17%	85.06%
Random Forest	85.42%	86.30%	86.13%
KNN	81.92%	82.59%	82.18%
Decision Tree	81.92%	84.36%	81.16%

Trials

Overall, the 3 trials did not produce any variations across the accuracies. Each of the algorithms was run thrice on each of the datasets, and verified to have the same results for each of the trials. So, the average across all 3 trials was reported with the same results. This ensured no accidental results.

Training and Test Splits compared with Cross Validation Mean Accuracy

Overall, the training and test splits resulted in a fair amount of variation in accuracy across the datasets. While the 20/80 split was low for the Student Performance and Wine datasets, the Online Shoppers and Dropout datasets had surprisingly high test accuracies. The 50/50 split had surprisingly high accuracies across several of the models and datasets.

Some of the algorithms, especially when run on the Online Shoppers and Dropout datasets, had slightly lower test accuracies on the 80/20 data split than the 50/50 data split, which is unusual, but may be explained by the size of the datasets and certain tendencies of the algorithms. It may be due to the larger testing size resulting in some lucky overfitting of the datasets, or capturing relationships that are more simplistic.

Also, to run these models in a computationally efficient way, smaller datasets were used, which might have resulted in more fluctuations in the measurement of testing accuracy. Since the target variables were often slightly modified to turn them into binary variables, it is also possible that the accuracy metric for evaluation predicted the majority class with a higher frequency. If the datasets are imbalanced, this can lead to seemingly higher testing accuracy for the smaller training splits.

To evaluate accuracy better, cross-validation mean accuracy was also calculated for each of the datasets according to each of the splits. The following tables represent the cross-validation mean accuracy for each of the datasets, with the algorithms as rows.

In some of the datasets, accuracy increases with more training data, which is expected and matches the reference paper's findings, as algorithms improve more with more training data (Caruana, n.d.). The 50/50 split continues to have great results for the cross-validation mean accuracy in some of the datasets.

A useful thing to note is that Random Forest, which is generally the most accurate, always increases in accuracy in the 80/20 split, demonstrating the algorithm's improvement with more data clearly. This also helps explain the differences in test accuracy as reflections of

imbalanced datasets, or due to the other problems mentioned above.

Table 5. Student Performance Cross-Validation Mean Accuracy

Algorithm	20-80 Split	50-50 Split	80-20 Split
Random Forest	62.06	72.24	76.49
Logistic Regression	72.95	72.86	73.79
SVM	68.25	72.84	73.21
KNN	68.25	66.96	70.51
Decision Tree	61.32	65.11	69.75

Table 6. Wine Cross-Validation Mean Accuracy

Algorithm	20-80 Split	50-50 Split	80-20 Split
Random Forest	100	97.78	98.57
Logistic Regression	94.29	100	99.31
SVM	100	100	99.31
KNN	100	98.89	97.17
Decision Tree	82.86	93.27	92.24

Table 7. Online Shoppers Cross-Validation Mean Accuracy

Algorithm	20-80 Split	50-50 Split	80-20 Split
Random Forest	76.28	77.97	78.11
Logistic Regression	76.12	77.42	77.21
SVM	76.03	78.3	78.2
KNN	74.49	76.59	76.24
Decision Tree	68.69	70.4	71.21

Table 8. Student Dropout Cross-Validation Mean Accuracy

Algorithm	20-80 Split	50-50 Split	80-20 Split
Random Forest	87.79	87.66	87.6
Logistic Regression	87.78	88.29	87.88
SVM	84.96	86.89	87.17
KNN	83.26	84.13	84.06
Decision Tree	84.17	85.8	87.26

Performance of Classifiers

Over each of the 4 datasets, the Classifiers were ranked by test accuracy and cross validation means accuracy in Table 9 and Table 10.

They show that Random Forest has the highest accuracy overall, and this is quite consistent across datasets, using either metric. This aligns with the findings of the reference paper, where Random Forest generally has high accuracy given the algorithm's complexity and ability to improve significantly with more training data (Caruana, n.d.).

This is followed by Logistic Regression, which is closely followed by SVM. Logistic Regression performed surprisingly well on both metrics, especially on the Dropout dataset. The nature of the data being more simplistic, size of datasets being relatively small, and metric of accuracy being prediction of binary outcomes could explain this difference from the reference paper (Caruana, n.d.).

SVM performed very well, with accuracies almost the same as Logistic Regression. The high accuracy performance of SVM is accurate to the reference paper's finding s(Caruana, n.d.). KNN performs worse than SVM, but still fairly well, as seen in both the reference paper and the evaluations here (Caruana, n.d.). Decision Trees consistently perform the worst with either accuracy metric across datasets (an exception being the test accuracy for the Wine dataset, where overfitting may have been a factor). This is also similar to the findings of the reference paper, and is reflected in Tables 9 and 10 (Caruana, n.d.).

Table 9. Overall Testing Accuracy Measured Across 50/50 Train/Test Split Averaged Across 4 Datasets

Algorithm	Student Performance	Wine	Online Shoppers	Dropout	Average Accuracy	Rank
Random Forest	72.31	100	76.8	85.42	83.6325	1
Logistic Regression	71.54	100	74.86	86.33	83.1825	2
SVM	70	100	76.32	85.65	82.9925	3
KNN	66.15	97.22	74.45	81.92	79.935	4
Decision Tree	66.15	100	68.65	81.92	79.18	5

Table 10. Overall Cross Validation Mean Accuracy Averaged Across 4 Datasets

Algorithm	Student Performance	Wine	Online Shoppers	Dropout	Average Accuracy	Rank
Random Forest	76.49	98.57	78.11	87.6	85.1925	1
Logistic Regression	73.79	99.31	77.21	87.88	84.5475	2
SVM	73.21	99.31	78.2	87.17	84.4725	3
KNN	70.51	97.17	76.24	84.06	81.995	4
Decision Tree	69.75	92.24	71.21	87.26	80.115	5

5. Conclusion

Overall, the findings were fairly consistent with the discussion in the reference paper (Caruana, n.d.). The effort to empirically evaluate these models using test accuracy and cross validated mean accuracy resulted in interesting findings. The main divergence from the reference paper was the surprisingly good performance of the Logistic Regression model, which has been explained in detail in the previous section (Caruana, n.d.).

Another interesting finding was the training and testing splits causing differences in accuracy counterintuitively, with more training data sometimes resulting in lower testing accuracy. This is also explained in the previous section.

The overall success of the Random Forest algorithm across datasets demonstrates that there can be some consistency in terms of model performance, and this closely follows the high ranking of the Random Forest algorithm in the reference paper (Caruana, n.d.). The relatively poor performance of the Decision Tree algorithm also follows the ranking from the reference paper, due to their lack of complexity and tendency to overfit (Caruana, n.d.).

Overall, there is variation in which algorithm performs best for each problem in each dataset, especially seen in Tables 1-4. This demonstrates one of the key points made in the reference paper - there is no truly one-size-fits-all model (Caruana, n.d.). Hence, the evaluation of these supervised machine learning models across a broad range of datasets and other variations, such as metrics of evaluation and training and test splits is a valuable exercise.

To improve these evaluations further, multiple metrics could be used. Larger datasets would probably result in findings closer to the reference paper (Caruana, n.d.). More preprocessing of the datasets to address any biases or imbalances would also be helpful. Differences in encoding and more careful evaluation of optimal hyperparameters would also be helpful. Also, due to limitations on computational power, some of the algorithms could not be optimized appropriately, which could be addressed in future research.

Lastly, as research in machine learning continues to progress, there is great value to such empirical evaluations of a wide range of models. More models will continue to emerge, and evaluating their general performance as well as their specific areas of high performance is beneficial to solving any data science problem.

6. References

1. Caruana, R. (n.d.). An Empirical Comparison of Supervised Learning Algorithms. https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

2. UCI Machine Learning Repository http://archive.ics.uci.edu/