An Empirical Comparison of Supervised Classification Algorithms

By Gurpreet Dhillon

Abstract

There are many different supervised classification algorithms that use numerous ideologies to achieve the goal of classification. Some algorithms perform better than others on some datasets. There is no one size fits all algorithm that will magically work for every problem. Some algorithms may work for a given problem but may need additional tuning to solve the problem. This idea took inspiration from a research paper written by Rich Caruana and Alexandru Niculescu-Mizil called "An Empirical Comparison of Supervised Learning Algorithms." This paper looks to find optimal hyperparameter tuning for logistic regression, random forest and k-nearest neighbors to find the best solutions on three different datasets.

1. Introduction:

In supervised classification, we train our classifier with data that has categorical labels. During this training stage, we are attempting to understand the association between each observation and its classification label. Once training is completed, we feed new data into the classifier without classification labels. Afterwards, the classifier then uses the knowledge gained from the training stage to make a predicted classification for that new data. Finally, we are able to measure how well the classifier performed by comparing the predicted classifications to the actual classifications. This process is known as "the accuracy."

The goal of this paper is to find the optimal solutions for each classifier given the data the classifier is using. This will be done by tuning each classifiers hyperparameters with the goal of achieving the highest accuracy possible. To achieve this, we will iterate through each classifiers hyperparameters. Our goal is to find the best hyperparameters that will yield the highest accuracy. The inspiration for this

paper came from “An Empirical Comparison of Supervised Learning Algorithms”, a research paper written by Rich Caruana and Alexandru Niculescu-Mizil which compares different supervised learning algorithms.

In addition to the accuracy metric, we will also use some complementary metrics such as precision, recall and f1 score. Precision will identify every observation predicted to be positive that is actually positive. Recall will illustrate the proportion of every positive observation that is truly positive. The F1 score will help us to measure of correctness achieved in positive prediction. These metrics allows us to see if a classifier is dealing with an imbalanced dataset. An imbalanced dataset is when the data has an unproportional ratio of classes. This means that the dataset does not have enough observations for a given class. We will keep this in mind when we run our experiments.

2. Methods

We will run 3 trials for the 3 classifiers, 3 datasets and we will be splitting the data 3 different ways, 20/80, 50/50, 80/20 for the training and test sets. In total, there will be 81 experiments. We will report the averaged accuracies for the 3 trials done. Train, validation accuracies will be in jupyter notebook. Test accuracies will be reported in the experiment table below.

2.1. Algorithms

The different classifiers used are Logistic Regression, Random Forest, and K-Nearest Neighbors. Logistic regression uses a probabilistic approach to identify classes. Random Forest uses decision trees to identify classes. K-Nearest Neighbors uses distance measures to identify classes. All three use different ideologies to achieve the same goal of classification.

2.1.1. Hyperparameters

The hyperparameters that we will be taking into account for Logistic regression will be penalty and C. We will use the default penalty parameter of L2 norm. This is known as one of the options for the

loss functions of our classifier which is used to reduce variance. Another parameter includes C, the inverse of the regularization strength, C = 1/lambda. C will be of the range 10^-8 to 10^4 similar to the values used in the carura paper.

The hyperparameters that we will be using for Random Forest will be the number of trees, n_estimastors. N_estimastors will be of the range 1-50.

The hyperparameters that we will be utilising for K-Nearest Neighbors will be the number of neighbors, n_neighbors. N_neighbors will be of the range 1-10

2.1.2. Gridsearch with Cross Validation

We will use the gridsearch technique to efficiently iterate through all specified hyperparameters for a classifier. This is a brute force method takes into account the given dataset and finds the best combination between the hyperparameters as it will iterate through all the different permutations. When it finishes its iterations, it will tell us the best hyperparameters to use in our classifier. In addition, gridsearch will use cross validation and this tells us the quality of our predictions. Using the dataset and splitting into a training and validation set will allow us for more quality in our overall predictions.

2.4 Datasets

We will be using three different datasets from the UCI database that will allow us to perform binary classification. The first one is a breast cancer dataset with 32 features and 569 observations. The second one is an email spam dataset with 56 features and 4601 observations. The third is a musk dataset with 165 features and 476 observations.

3. Experiment:

	Dataset	Classifier	Accuracy	Precision	Recall	F1-Score
20/80 Split	Breast cancer	Logistic Regression	0.3728070175438596 4	0.64	0.37	0.21

50/50 Split	Breast cancer	Logistic Regression	0.8842105263157894	0.89	0.88	0.88
80/20 Split	Breast cancer	Logistic Regression	0.3684210526315789	0.14	0.37	0.20
20/80 Split	Breast cancer	Random Forest	0.902046783625731	0.88	0.88	0.88
50/50 Split	Breast cancer	Random Forest	0.9298245614035089	0.93	0.93	0.93
80/20 Split	Breast cancer	Random Forest	0.95321637426	0.96	0.96	0.96
20/80 Split	Breast cancer	K-Nearest Neighbors	0.9495614035087719	0.95	0.95	0.95
50/50 Split	Breast cancer	K-Nearest Neighbors	0.9333333333333333	0.93	0.93	0.93
80/20 Split	Breast cancer	K-Nearest Neighbors	0.9473684210526315	0.95	0.95	0.95

20/80 Split	Email spam	Logistic Regression	0.9185004074979625	0.92	0.92	0.92

50/50 Split	Email spam	Logistic Regression	0.9222077357670578	0.92	0.92	0.92
80/20 Split	Email spam	Logistic Regression	0.9305103148751357	0.93	0.93	0.93
20/80 Split	Email spam	Random Forest	0.9309064565788282	0.93	0.93	0.93
50/50 Split	Email spam	Random Forest	0.9439374185136897	0.94	0.94	0.94
80/20 Split	Email spam	Random Forest	0.95801664857	0.96	0.96	0.96
20/80 Split	Email spam	K-Nearest Neighbors	0.8878022276555284	0.89	0.89	0.89
50/50 Split	Email spam	K-Nearest Neighbors	0.9048239895697523	0.91	0.90	0.90
80/20 Split	Email spam	K-Nearest Neighbors	0.9077090119435396	0.91	0.91	0.91

20/80 Split	Musk	Logistic Regression	0.9399507482477742	0.94	0.94	0.94
50/50 Split	Musk	Logistic Regression	0.951500454683237	0.95	0.95	0.95

80/20 Split	Musk	Logistic Regression	0.953030303030303	0.95	0.95	0.95
20/80 Split	Musk	Random Forest	0.9465807918166319	0.95	0.95	0.94
50/50 Split	Musk	Random Forest	0.9657471961200363	0.97	0.97	0.96
80/20 Split	Musk	Random Forest	0.9734848484848485	0.97	0.97	0.97
20/80 Split	Musk	K-Nearest Neighbors	0.9471490812653912	0.95	0.95	0.95
50/50 Split	Musk	K-Nearest Neighbors	0.9618066080630494	0.96	0.96	0.96
80/20 Split	Musk	K-Nearest Neighbors	0.9681818181818181	0.97	0.97	0.97

3.1 Analysis

Logistic Regression:

For the breast cancer dataset, the logistic regression classifier does not seem to work well on most training testing splits. This seems to be a case where the dataset does not have enough information to make good predictions. In the 80/20 split, the best hyperparameter was a inverse regularization value of 0.046415888336127725 which implies that regularization was strong. In essence, this happens when there is noise in our data. The precision, recall and f1 scores for both the malignant and benign classes were low meaning that the observations meaning that there were too many false positives and false negatives. In the 50/50 split, the precision, recall and f1 score for the benign and malignant observations were higher compared to the 20/80 split. This means that there were far less false negatives and false positives. The best hyperparameter for this split was an inverse regularization value of 21.54434690031878. This is not as strong compared to the 80/20 split. In the 50/50 split, there seems to be less noise. The 80/20 split had poor

performance; this is similar to the 20/80 split and has the same regularization value. In this case, the best split would be the 50/50 split.

For the email spam dataset, logistic regression does well across all splits. In the 20/80, 50/50, 80/20 splits, the accuracies and F1 scores are high. In this dataset, there seems to be a good balance between the classes and it is able to predict the positive predictions correctly. There are not many false positives or false negatives which is good. The best hyperparameter for the 20/80 and 50/50 splits are a regularization value of 1. In the 80/20 split, the best hyperparameter inverse regularization value is 1000; this is the best achieved so far in the logistic regression experiment. This split had the lowest regularization of all the models. The best split in this case would be the 80/20 split.

For the Musk dataset, logistic regression also does well across all splits. In the 20/80, 50/50, 80/20 splits, the accuracies and F1 scores are high. This dataset also has a good balance between classes and the model is able to predict the positive predictions correctly. There are not many false positives or false negatives which is good. The best hyperparameter for the 20/80 split, is an inverse regularization value of 0.0001. The best hyperparameter for the 50/50 split, is an inverse regularization value of 0.002154434690031882. The best hyperparameter for the 50/50 split, is an inverse regularization value of 0.002154434690031882. The best split in this case would be the 80/20 split.

Random Forest

For the breast cancer dataset, the Random Forest classifier seems to work well on all splits. The accuracies, precision, recall and f1 scores are relatively high for all splits. The hyperparameters for each trail of a split varied. For example, in the 20/80 split, the n_estimators hyperparameter in the first trial was a 45, 33 for the second trial and 46 for the third. This variance happened on all the trails for each split. This can be attributed to the randomness in the process of the classifier.

For the email spam dataset, the Random Forest classifier seems to work well on all splits. The accuracies, precision, recall and f1 scores are relatively high for all splits. The hyperparameters for each trial of a split varied similar to the breast cancer dataset.

For the Musk dataset, the Random Forest classifier seems to work well on all splits. The accuracies, precision, recall and f1 scores are relatively high for all splits. The hyperparameters for each trail of a split varied like in the other two datasets.

K-Nearest Neighbors (needs data standardized)

For the breast cancer dataset, the K-Nearest Neighbors classifier seems to work well on all splits. K-Nearest Neighbors uses distance measures to identify classes, so splitting the data will not have much of an impact on the accuracies. The malignant observation will have a closer distance to each other than begien observations and vice versa. That being said the accuracies, precision, recall and f1 scores are relatively high for all splits. The best hyperparameter for the 20/80 split is a n_neighbor value of 7. This is probably due to not as many observations in the training set as the full dataset does not have that many observations and features to being with. The best hyperparameter for the 50/50 and 80/20 splits is a n_neighbor value of 3.

For the email spam dataset, the K-Nearest Neighbors classifier seems to work well on all splits. The accuracies, precision, recall and f1 scores are relatively high for all splits. The best hyperparameter for the 20/80 split is a n_neighbor value of 5. The best hyperparameter for the 50/50 split is a n_neighbor value of 7. The best hyperparameter for the 80/20 split is a n_neighbor value of 1.

For the musk dataset, For the email spam dataset, the K-Nearest Neighbors classifier seems to work well on all splits. The accuracies, precision, recall and f1 scores are relatively high for all splits. The best hyperparameter for the 20/80, 50/50, and 20/80 splits is a n_neighbor value of 2.

4 . Conclusion:

In conclusion, the performance of the three classifiers were evaluated on three different datasets using three different splits. Random forest and k-nearest neighbors performed well on all splits for all datasets as all their accuracies, precision, recall and f1 scores were relatively high. It is also important to note that k-nearest neighbors classifier needs the dataset standardized before using the classifier as all features need to be scaled properly. Once the data was standardized, the classifier gave better results. Logistic regression performed well on 2 out of the 3 datasets. Logistic regression had trouble with the cancer dataset and only performed well on the 50/50 split. On the other two splits, the precision scores revealed that there were too many false positives. In addition, the recall score suggested that it was also having trouble with the false negatives. This demonstrates that Logistic Regression is not optimal for this dataset. Out of these three classifiers, I would suggest the random forest algorithm as it will great results and less time to compute compared to k-nearest neighbors.

Acknowledgments

I would like to thank professor Zhuowen Tu for an informative and fun quarter. I would also like to thank all the TA's and IA's for being very helpful throughout the quarter.

References

[1] Caruana, R., Niculescu-Mizil A. (2006). "An Empirical Comparison of Supervised Learning Algorithms" ICML 2006: 161-168.

[2] Caruana, R., Karampatziakis, N., Yessenalina, A. (2008). "An Empirical Evaluation of Supervised Learning in High Dimensions" ICML 2006: 161-168.