CSC 466 Lab 3 Report: Supervised Learning: Classification Algorithms

Martin Solomon Hsu, mshsu@calpoly.edu
Lana Mai Huynh, lmhuynh@calpoly.edu

Abstract

In this lab, we utilize three different classification algorithms, C4.5, Random Forest, and K-Nearest Neighbors, to try and correctly predict the class label for every record in a training dataset. To evaluate the performance of our different classification algorithms, we rigorously tested our models by tuning our hyperparameters through many iterations on three datasets of different sizes and consisting of different types of attributes. Ultimately, we found that our C4.5 algorithm consistently performed extremely well in terms of accuracy metrics and runtime.

I. Introduction

Classification is the process of classifying something according to shared qualities or characteristics. This is a type of supervised learning because the training set contains class labels, so we can “supervise” predictions of our classifier.

C4.5

The C4.5 algorithm is a recursive decision tree induction algorithm which consists of three main steps: (1) termination conditions, (2) selection of the splitting attribute, and (3) tree construction.

Random Forest

Random forests build a collection of decision trees, where each decision tree is built based on a subset of a training set and a subset of attributes.

K-Nearest Neighbors

The KNN algorithm, a classifier that utilizes lazy evaluation, operates by finding the *k* nearest neighbors to a given data point, and it takes the majority vote to classify the data point.

II. Classifier Algorithm Applications

Heart Dataset

The heart dataset consists of 918 observations, where each row is a person. There are 11 numerical or categorical attributes in which we try to classify heart disease.

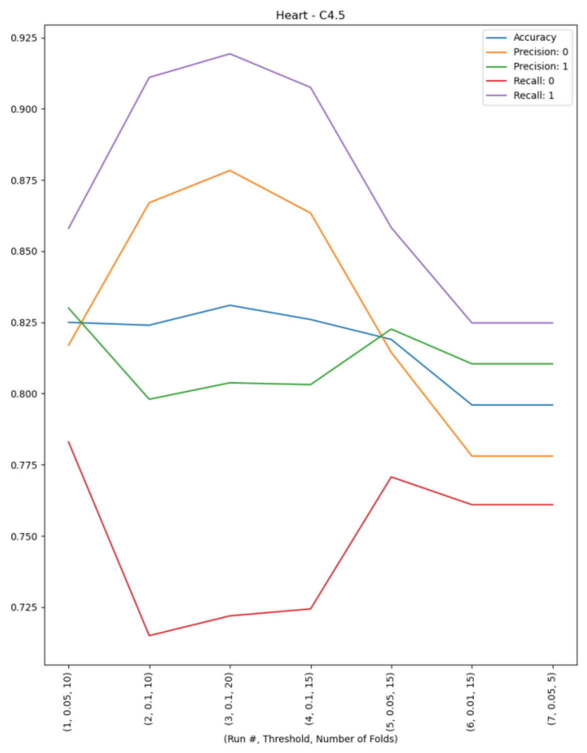
C4.5
For the C4.5 algorithm, we started out with a threshold of 0.05 and 10 folds. The accuracy of 0.825 was satisfactory and our precision and recall values were high. We then increased and decreased the threshold and fold values for six more iterations, but our first set of hyperparameters ended up being the most optimal values.

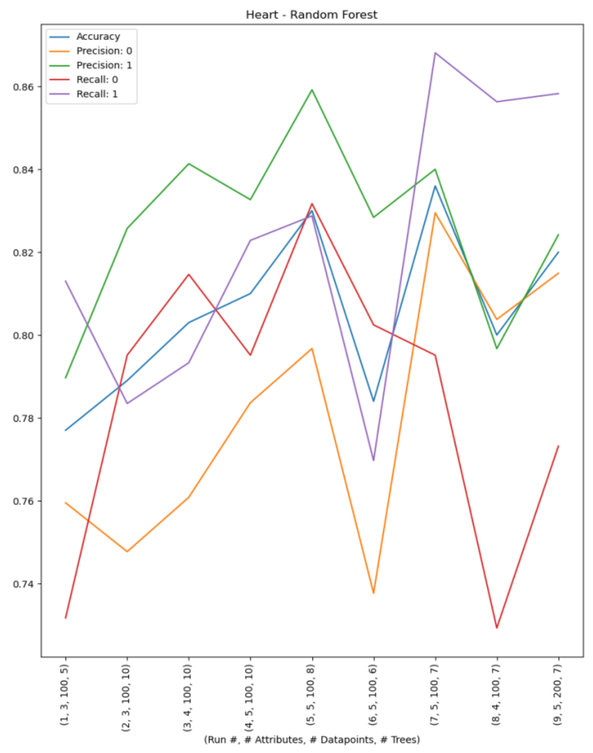
Random Forest
For the random forest algorithm, we started out with 3 attributes, 100 data points, and 5 trees. We adjusted all three hyperparameters, but we noticed that the most influential ones were the number of attributes and trees. After 9 iterations, we concluded that the optimal values of hyperparameters are 5 attributes, 100 data points, and 7 trees.

KNN
For the *k*-nearest neighbors algorithm, we tested out *k* values of 2, 3, and 4. Although there are only 2 class values, whether the person has heart disease or not, a *k* value of 3 produced the best accuracy, precision, and recall values.

Comparative Study

Figure 2.1: Heart Dataset Metrics by Classification Algorithm





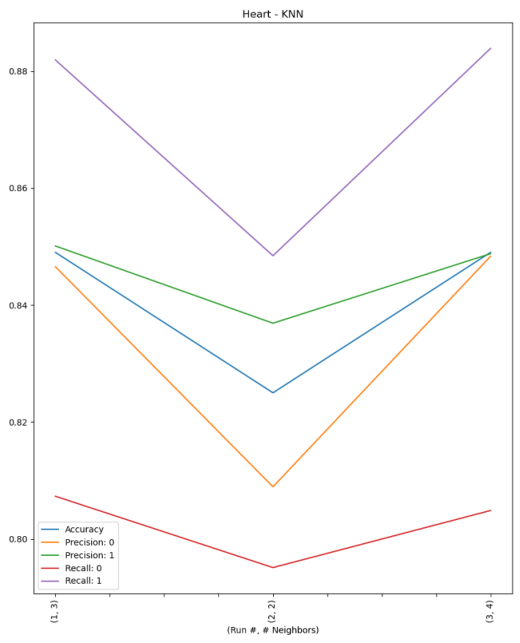


Table 2.1. Classification algorithms comparison with optimal hyperparameters on Heart Dataset

Classification Algorithm	C4.5	Random Forest	KNN

Hyperparameters	Threshold = 0.05
	Validation Folds = 10
		Attributes = 5
		Data points = 100
		Trees = 7
			k = 3

Accuracy	0.825	0.836	0.849
Precision			
No disease (0)	0.817	0.83	0.847
Disease (1)	0.83	0.84	0.85
Recall			
No disease (0)	0.783	0.795	0.807
Disease (1)	0.858	0.868	0.882
Runtime (seconds)	0.32	0.19	1.43

For the heart dataset, the KNN classification algorithm performed the best. The model gave us the best accuracy, precision, and recall values. Although the runtime is the worst out of all three algorithms, it is only by an increase of one second which is almost negligible in context of using our model in the real world.

Mushroom Dataset
The mushroom dataset consists of 8,124 observations, where each row is a mushroom. There are 22 categorical attributes in which we try to classify the type of mushroom.

C4.5
For the C4.5 algorithm, we started out with a threshold of 0.05 and 10 folds. To our surprise, we achieved perfect accuracy, precision, and recall. We reduced the threshold value and number of folds, expecting our validation metrics to worsen; however we still achieved perfect accuracy, precision, and recall. We concluded that a threshold of 0.01 and 5 folds is optimal for this dataset because there is a small computation cost in tandem with a smaller risk of overfitting.

Random Forest
For the random forest algorithm, we started out with 5 attributes, 100 data points, and 10 trees. We achieved a high accuracy of 0.96, and precision and recall values were excellent. From here, we adjusted the number of attributes and trees for three more iterations to see how they would affect our validation metrics. We concluded with 5 attributes, 100 data points, and 7 trees, which gave us 0.975 accuracy and excellent precision and recall values. It surprised us that we only needed 5 out of 22 attributes to build a great model.

KNN

For the *k*-nearest neighbors algorithm, we first tested out a *k* value of 2, since there are only 2 classified values: e and p. We achieved perfect accuracy, precision, and recall. We also tested out a *k* value of 3, which also produced perfect accuracy, precision, and recall. Ultimately, we chose a *k* value of 2 because it makes the most sense in the context of our data.

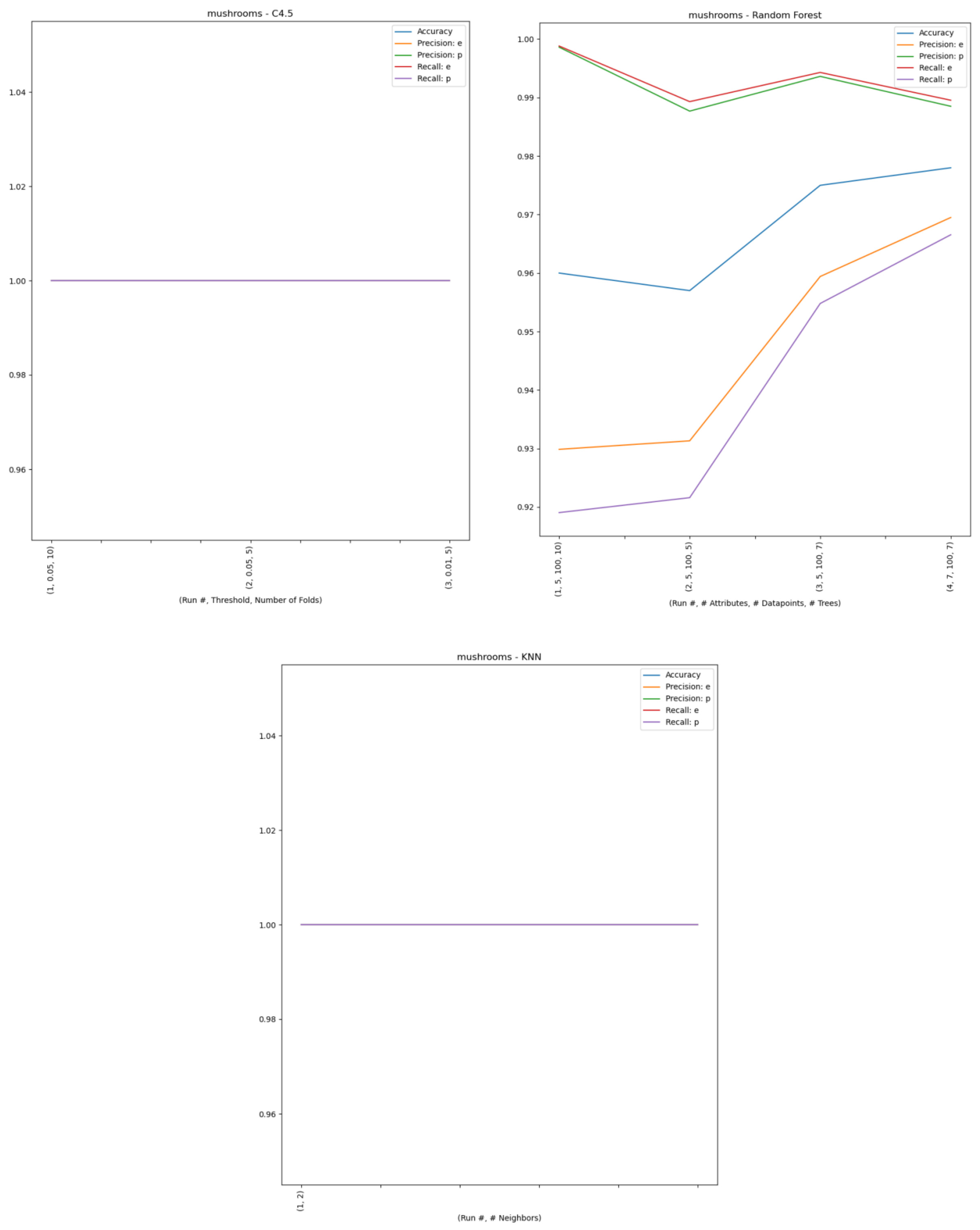


Figure 2.2: Mushroom Dataset Metrics by Classification Algorithm

Table 2.2. Classification algorithms comparison with optimal hyperparameters on Mushroom Dataset

Classification Algorithm	C4.5	Random Forest	KNN

Hyperparameters	Threshold = 0.01
	Validation Folds = 5
		Attributes = 5
		Data points = 100
		Trees = 7
			k = 2

Accuracy	1	0.975	2
Precision			
Mushroom e	1	0.959	1
Mushroom p	1	0.994	1
Recall			
Mushroom e	1	0.994	1
Mushroom p	1	0.955	1
Runtime (seconds)	0.22	1.1	14.17

For the mushroom dataset, all three models performed very well. C4.5 and KNN gave us perfect accuracy, precision, and recall, suggesting that this data was easy to classify. However, the C4.5 classification algorithm had the best runtime, so it is our preferred algorithm of choice.

Iris Dataset

The mushroom dataset consists of 150 observations, where each row is an iris flower. There are 4 numerical attributes in which we try to classify the type of iris.

C4.5

For the C4.5 algorithm, we started out with a threshold of 0.05 and 10 folds. We achieved a high accuracy of 0.953, and relatively high precision and recall values as well. Similarly to the other datasets, we adjusted the threshold and number of folds to reduce our computational cost, while attempting to keep our accuracy, precision, and recall values high. In the end, we optimized our model with a 0.05 threshold and 2 folds, giving us an accuracy of 0.96 and relatively high precision and recall values. It surprised us that our validation metrics stayed consistent, even when drastically dropping the number of folds to 2.

Random Forest

For the random forest algorithm, we started out with 4 attributes, 50 data points, and 10 trees. We initialized with a smaller number of data points, 50, because we have a much smaller dataset than the previous ones that we used. We achieved a high accuracy of 0.953, and precision and recall values were excellent. We adjusted our hyperparameters, but our validation metrics did not change that much. After three more iterations, we decided that the optimal values of hyperparameters were our original values of 4 attributes, 50 data points, and 10 trees.

KNN

For the *k*-nearest neighbors algorithm, we tested out *k* values of 2, 3, and 4. Although there are only 2 class values, whether the person has heart disease or not, a *k* value of 3 produced the best accuracy, precision, and recall values. Surprisingly, all *k* values produced the same accuracy values. However, a *k* value of 3 produced the best precision and recall values, which makes the most sense in this context because there are three different types of irises that we are trying to classify.

Figure 2.3: Iris Dataset Metrics by Classification Algorithm

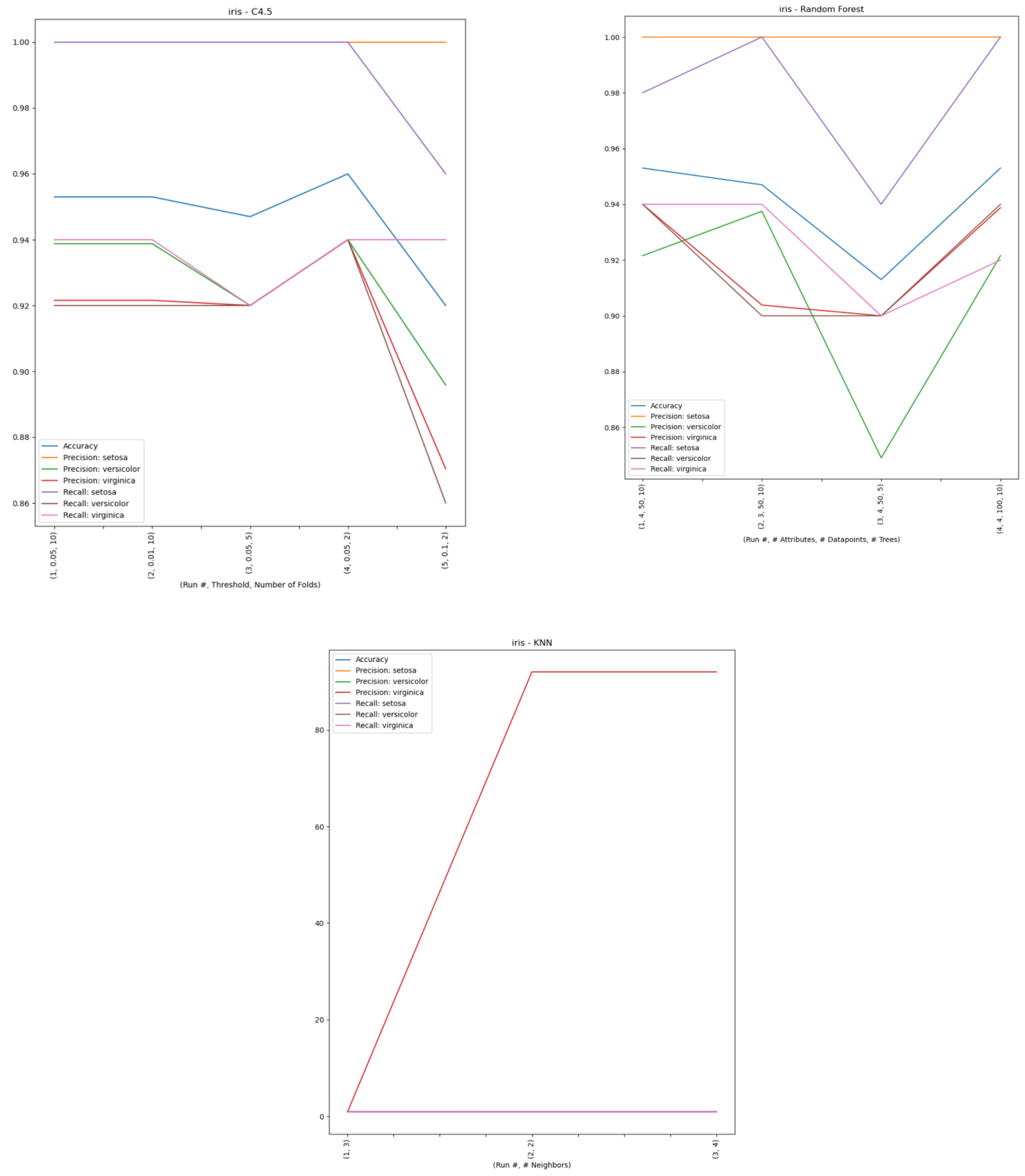


Table 2.3. Classification algorithms comparison with optimal hyperparameters on Iris Dataset

Classification Algorithm	C4.5	Random Forest	KNN

Hyperparameters	Threshold = 0.05
	Validation Folds = 2
		Attributes = 4
		Data points = 50
		Trees = 10
			k = 3

Accuracy	0.96	0.953	0.947
Precision			
Setosa	1	1	1
Versicolor	0.94	0.922	0.904
Virginica	0.94	0.94	0.939
Recall			
Setosa	1	0.98	0.98
Versicolor	0.94	0.94	0.94
Virginica	0.94	0.94	0.92
Runtime (seconds)	0.18	0.18	0.93

For the iris dataset, all three models performed very well. However, the C4.5 classification algorithm had slightly better accuracy, precision, recall, and runtime values, so that would be our algorithm of choice.

III. Conclusion

Overall, our C4.5 classification algorithm seemed to perform extremely well on all three datasets in terms of accuracy metrics and runtime, computed by utilizing *k*-fold cross-validation. C4.5 was our algorithm of choice for two out of the three datasets. KNN performed slightly better on the heart dataset, but C4.5 was not too far off. The heart dataset consists of both numerical and categorical attributes which suggests that we should do more investigation on why this is the case.