CSC 466 Lab 3 Report: Classification Algorithms

Rachel Roggenkemper: rroggenk@calpoly.edu
Kirina Sirohi: kasirohi@calpoly.edu
Dr. Alexander Dekhtyar
CSC 466: Knowledge Discovery from Data

Abstract

In this lab, we implemented C4.5, Random Forest, and K-Nearest Neighbors to predict the class label for every observation in the training dataset using a cross-validation procedure. To evaluate our results, we tested our models by tuning our hyperparameters for each method. Overall, we discovered that our C4.5 algorithm performed the best in terms of accuracy metrics, while our K-Nearest Neighbors algorithm did the worst since it was difficult to test due to the amount of time it took to run.

Introduction

The classification is given a training dataset, the goal is to construct a classification/prediction function that correctly predicts the class label for every record in the training dataset. This is a form of supervised learning because the training set contains class labels. Thus we can compare (supervise) predictions of our classifier.

Implementation Details

1. **C4.5:**
   The C4.5 algorithm for decision tree induction takes in the training dataset and the list of attributes, and outputs the constructed decision tree. It is a recursive decision tree induction algorithm following three main steps: (1) termination conditions, (2) selection of the splitting attribute, and (3) tree construction.
2. **Random Forest:**
   Random Forests are an extension of bagging. A bagging technique resamples the training set with replacement, but keeps all attributes in the dataset ”active” for each resampled training set. 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.
3. **K-Nearest Neighbors:**
   K-Nearest Neighbors is a simple, but surprisingly robust lazy evaluation algorithm. The idea behind K-Nearest Neighbors is given a data point, using distance measures to find the k-nearest neighbors to that given point, and assigning the class of that point using the class of the plurality of the k-nearest neighbors.

Nursery Dataset

The Nursery dataset contains information about the decision-making process for recommending children for nursery care. Each data point contains a unique combination of values collected during an intake interview with parents, documenting family status and circumstances. The class variable is the recommendation the application resulted in. There are five levels of recommendations: not recommended, recommended, very recommended, priority, special priority (with recommended, strangely enough being an outlier class with only two instances in the dataset).

C4.5: The hyperparameter we were tuning was the threshold. The different values of the threshold we were testing were 0.2, 0.4, 0.3, and then 0.25. With a threshold of 0.2, this resulted in an accuracy of 88.2%. We then tested a threshold of 0.4, which resulted in a threshold of 70.7%. After, we tried a threshold of 0.3, which resulted in an accuracy of 79.7%. Lastly, we tested an accuracy of 0.25, which resulted in an accuracy of 87.8%. Although a threshold of 0.2 resulted in the best accuracy, the corresponding tree was too extensive and was overfitting the data. In lieu of that, we went with a threshold of 0.25. This led to a lower accuracy compared to a threshold of 0.20, but now the decision tree is much more fair.

Random Forests: The hyperparameters we were tuning for Random Forests were number of attributes, number of data points, and number of trees. We are utilizing cross validation with 10 folds. Some of the sets of hyperparameters tested included using 2 attributes, 20 data points, and 3 trees which resulted in an accuracy of 67.6%. Next we tested 4 attributes, 30 data points, and 5 trees which resulted in an accuracy of 78.6%. Lastly, we tested 4 attributes, 60 data points, and 5 trees which resulted in an accuracy of 41.1%. Thus, we went with 4 attributes, 30 data points, and 5 trees since it resulted in the highest accuracy.

K-Nearest Neighbors: The hyperparameter we were tuning was k, the number of nearest neighbors. Due to the sheer size of the dataset, running this algorithm took a massive amount of time (about 6 hours!), which limited our testing capabilities. Thus, due to the size of the dataset, we chose to run the program using a k = 50. This resulted in an accuracy of 81.2%.

Classification Algorithm	**C4.5**	**Random Forests**	**K-Nearest Neighbors**
Hyperparamters	Threshold = 0.25	Attributes = 4 Data Points = 30 Trees = 5	k = 50
Accuracy	87.8%	78.6%	81.2%

Thus, for the nursery dataset, C4.5 was the classification algorithm that performed the best out of C4.5, Random Forests, and K-Nearest Neighbors. This model performed the best when taking accuracy into account.

Iris Dataset

The Iris dataset is a dataset containing a few hundred records. Each record depicts physical dimensions of a specific iris flower from one of three different subspecies of iris: (1) Iris Setosa, (2) Iris Versicolor, or (3) Iris Virginica. There are four physical parameters measured for each flower: (1) sepal length in cm, (2) sepal width in cm, (3) petal length in cm, and (4) petal width in cm.

C4.5: The hyperparameter we were tuning was the threshold. The different values of the threshold we were testing were 0.2, 0.5, 0.25, and 0.1. These resulted in accuracies of 93.3% for thresholds of 0.2, 0.25, and 0.5, and an accuracy of 86.8% for a threshold of 0.1. Ultimately, we went with a threshold of 0.25 because it was a nice middle ground between our hyperparameters tested.

Random Forests: The hyperparameters we were tuning for Random Forests were number of attributes, number of data points, and number of trees. We are utilizing cross validation with 10 folds. Some of the sets of hyperparameters tested included 4 attributes, 30 data points, and 5 trees which resulted in an accuracy of 89.3%. Next we tested 2 attributes, 30 data points, and 8 trees which resulted in an accuracy of 81.3%. Lastly, we tested 3 attributes, 30 data points, and 8 trees which resulted in an accuracy of 87.3%. Thus, we went with 4 attributes, 30 data points, and 5 trees since it resulted in the highest accuracy.

K-Nearest Neighbors: The hyperparameter we were tuning was k, the number of nearest neighbors. The different values of the number of nearest neighbors we tested were 5, 6, 7, and 10. The respective accuracy levels were 95.3%, 93%, 94%, and 93.3%. Thus, since using 5 nearest neighbors resulted in the highest accuracy and made sense logically for this dataset, we went with that ultimately.

Classification Algorithm	**C4.5**	**Random Forests**	**K-Nearest Neighbors**
Hyperparamters	Threshold = 0.25	Attributes = 4 Data Points = 30 Trees = 5	k = 5
Accuracy	93.3%	89.3%	95.3%

In conclusion, for the iris dataset, K-Nearest Neighbors was the classification algorithm that performed the best out of C4.5, Random Forests, and K-Nearest Neighbors. This model performed the best when taking accuracy into account.

Wine Datasets

The Wine dataset contains information about the chemical properties of around 4900 red and white varieties of Portuguese Vinho Verde wine. Each wine is described by 11 numeric features (such as acidity, citric acid, residual sugar, etc..), followed by the class variable describing the quality of the wine. The class variable takes values 0, 1, …, 10. The goal is to predict from the chemistry of the wine its quality. Note that there are two separate datasets corresponding to red and white wine collections. Because red and white wines have very different chemical profiles, we will be treating these two datasets as independent and separate.

Red Wine

C4.5: The hyperparameter we were tuning was the threshold. The different values of the threshold we were testing were 0.25 and 0.4. Using a threshold of 0.25, we achieved an accuracy of 59.6%. Then, because we didn't want to try a lower threshold to avoid overfitting, we tested using a threshold of 0.4. This resulted in an accuracy of 60.9%. Since the accuracy didn't change much from 0.25 to 0.4, we felt like 0.4 was an optimized value.

Random Forests: The hyperparameters we were tuning for Random Forests were number of attributes, number of data points, and number of trees. We are utilizing cross validation with 10 folds. Due to the shear size of the dataset, running this algorithm took a massive amount of time, which limited our testing capabilities. Thus, due to the size of the dataset, we chose to run the program using 3 attributes, 30 data points, and 4 trees. This resulted in an accuracy of 49.9%.

K-Nearest Neighbors: The hyperparameter we were tuning was k, the number of nearest neighbors. Due to the shear size of the dataset, running this algorithm took a massive amount of time, which limited our testing capabilities. Thus, due to the size of the dataset, we chose to run the program using 50 nearest neighbors. This resulted in an accuracy of 51.2%.

Classification Algorithm	**C4.5**	**Random Forests**	**K-Nearest Neighbors**
Hyperparamters	Threshold = 0.4	Attributes = 3 Data Points = 30 Trees = 4	k = 50
Accuracy	60.9%	49.9%	51.2%

In the end, for the red wine dataset, C4.5 was the classification algorithm that performed the best out of C4.5, Random Forests, and K-Nearest Neighbors. This model performed the best when taking accuracy into account.

White Wine
C4.5: C4.5: The hyperparameter we were tuning was the threshold. The different values of the threshold we were testing were 0.25 and 0.4. Using a threshold of 0.25, we achieved an accuracy of 58.4%. Then, because we didn't want to try a lower threshold to avoid overfitting, we tested using a threshold of 0.4. This resulted in an accuracy of 60.5%. Since the accuracy didn't change much from 0.25 to 0.4, we felt like 0.4 was an optimized value.

Random Forests: The hyperparameters we were tuning for Random Forests were number of attributes, number of data points, and number of trees. We are utilizing cross validation with 10 folds. Due to the shear size of the dataset, running this algorithm took a massive amount of time, which limited our testing capabilities. Thus, due to the size of the dataset, we chose to run the program using 3 attributes, 50 data points, and 3 trees. This resulted in an accuracy of 39.7%. Next, we chose to run the program using 10 attributes, 30 data points, and 4 trees. This resulted in an accuracy of 41.3%.

K-Nearest Neighbors: The hyperparameter we were tuning was k, the number of nearest neighbors. Due to the shear size of the dataset, running this algorithm took a massive amount of time, which limited our testing capabilities. Thus, due to the size of the dataset, we chose to run the program using 50 nearest neighbors. This resulted in an accuracy of 46.4%.

Classification Algorithm	**C4.5**	**Random Forests**	**K-Nearest Neighbors**
Hyperparamters	Threshold = 0.4	Attributes = 10 Data Points = 30 Trees = 4	k = 50
Accuracy	60.5%	41.3%	46.4%

Ultimately, for the white wine dataset, C4.5 was the classification algorithm that performed the best out of C4.5, Random Forests, and K-Nearest Neighbors. This model performed the best when taking accuracy into account.

Conclusion
In summary, from our rigorous testing, we discovered that our C4.5 algorithm performed extremely well for the datasets we tested in terms of accuracy metrics. Although we tested as much as we were able to, our testing capabilities were limited due to the amount of time it took

to run the algorithms, especially K-Nearest Neighbors. However, overall, C4.5 still performed very well in terms of accuracy.