**Abstract**
The ability to accurately classify and predict instances of data has become increasingly important with the rise of machine learning. In addition, the growing accessibility to public datasets has allowed for more researchers and computer scientists to train models for specific use cases. These various use cases will allow for increased efficiency and automation in many different fields such as healthcare, finance, and business. In this paper, we conduct experiments on three different supervised machine learning models like Random Forests, Decision Trees, and Neural Networks using four datasets. By analyzing and comparing the results of each model on these four data-sets, we analyzed which models would perform best in a variety of different situations.

**Introduction**

Machine learning has been in the forefront of Computer Science for quite a while now. In particular, Supervised Machine Learning is a way in which algorithms learn from labeled data-sets called a training set to be able to predict or classify new instances of information. For example, a supervised machine learning algorithm can learn to classify any one instance of a bird given that it had learned from pictures of many different birds. This process of recognizing or classifying instances of data can help automate processes requiring less human intervention. Furthermore, supervised machine learning can also be helpful where humans are not good at identifying something.

In this experiment, we are going to assess the accuracy, and recall on four different data-sets with both categorical and numerical attributes using three different supervised Machine Learning models. We are going to compare the accuracy of different models on different data-sets and find if one is statistically significant than the other model. This will allow us to know which model is better for a particular data-set.

For this experiment, we are going to implement Decision Trees, Random Forests, and Neural Networks using scikit-learn and train the different models using these different techniques. 75% of the data-set will be used for training, and 25% will be used for testing. The accuracy will be determined on the test set by comparing the true label versus the predicted label. Consequently,we will be able to measure the performance of these models in regard to each other.

After implementing the different supervised learning algorithms using sci-kit learn,we ran the algorithms on these different data-sets across 5 different seeds. All of the average accuracies across the five seeds for the different data-sets, and supervised algorithms were above 90%. One exception was the mnist_1000 dataset with the Decision Tree Algorithm. The average accuracy for this across the different five seeds was only 78.92%.

**Problem**

As stated earlier, supervised machine learning algorithms are trained to automate certain processes such as classification of a new instance of data based on the trained model. This trained model will help any new instances of data be predicted or classified. As mentioned before,supervised machine learning has many applications in numerous fields. One such example can be to predict if a person has heart disease given their health information. If the prediction model for this problem is accurate enough then it can potentially help physicians with helpful information and improved diagnosis. Relating to the benefits of supervised machine learning, in this experiment, we will try to assess four different data-sets , and their performance on Decision Trees, Random Forests, and Neural Networks.In doing this, we try to gain insights on which model performs better on what data-set, and the reasons this might be happening.

The data-sets we will be working with are “hypothyroid.csv”, “mnist_1000.csv”, “monks1.csv”, and “votes.csv”. The hypothyroid data-set is obtained from WEKA, and has instances of both nominal and categorical patient attributes. Along with the attribute data, each patient instance has a label indicating if they have hypothyroidism, and the kind of hypothyroidism if they do have it. The “monks1.csv” data-set obtained from the UCI data repository has certain physical characteristics as nominal attributes in the data-set, and the labels tell us if one instance is a monk or not. There is a simple rule for this data-set to identify a monk. An instance is a monk if the monk has a red jacket color or the monk’s head shape and body shape are the same. The “votes.csv” data-set also obtained from WEKA contains information about members of the U.S. Congress and if they were a democrat or a republican. The attributes in the data-set are the voting histories of the members relating to numerous political policies. Lastly, we will also be working on the “mnist_1000.csv” dataset, which has pixel grayscale intensities of each cell in a 28x28 image. This will help us label a digit from 0 to 9 for each instance according to the grayscale intensities of each pixel.

The summary statistics for each data-set is given below:

| | No. of Attributes | No. of possible labels | Proportion of each label |
|---|---|---|---|
| monks1.csv | 6 | 2 | yes: 0.5<br>no : 0.5 |
| hypothyroid.csv | 27 | 4 | negative: 0.922<br>compensated_hypothyroid: 0.0514<br>primary_hypothyroid:0.0252<br>secondary_hypothyroid:0.00053 |
| mnist_1000.csv | 784 | 10 | All labels 0-9 have proportion 0.1 |
| votes.csv | 15 | 2 | republican :0.386<br>democrat : 0.614 |

**Solution**

To execute the project, we will have to run the different supervised learning algorithms(Decision Trees, Random Forests, Neural Networks) on our four different data-sets.

Firstly, we have to prepare the data-set for training with these three supervised learning algorithms. Any data-set has to be shuffled before splitting it into training, and testing sets in order to prevent biases that may show up as a result of the ordering of the data-set.
Since these supervised machine learning algorithms can not work with raw categorical data or need some other algorithm implementation to work with categorical data, we are going to one hot encode the categorical columns of each data-set. One hot encoding is a technique that allows users to convert categorical data to numerical values, which allows a data-set to be implemented with a supervised learning algorithm that only works with numerical data.

Since the data-set is now shuffled and ready for implementation with the algorithms, we can now split the data into training and testing sets. For the purposes of this experiment, we will be using 75% for training and 25% for testing.

We train the training sets using scikit-learn – a pre-existing module API for machine learning – that allows easy development for machine learning models. The part of “fitting” refers to this part in scikit-learn which just means to run the algorithm on the training set. This part in the code is self-explanatory and readers are encouraged to check the documentation of scikit-learn for the in-depth analysis of the usage of it.

As we will be using Decision Trees, Random Forests, and Neural Networks for our experiment, it is important to get an idea of how each of these work.

The first learning approach we used in our experiment was decision trees. Decision trees are structured trees such that each node is an attribute, and each branch contains one of the values of the attribute. In addition, the leaf nodes of the trees represent the prediction value on a specific path from the root to a leaf. In order to build the tree, the algorithm uses recursion to grow the children along each path. First, it checks to see if there are still attributes not seen on a path. In the case every attribute is already in the path, it takes the most common label in the set of training instances to use as the leaf (prediction). If there are still attributes to be added, the algorithm will move on to the next check. The second step checks to see if all instances have the same label. If so, the algorithm will add a leaf node with the label as the prediction. The last step is the recursive else case that starts by finding the best attribute based on the highest calculated gain. It then either finds a best guess if there exists no subset of a unique value of an attribute or it recurses over the children with the new subset. After the tree is built, the prediction is made by feeding in instances and starting from the root node and going down the tree path that matches the test instance’s attributes. After it reaches a leaf node, it will return the prediction.

The next learning approach we used was random forests. The structure of random forests is based around training multiple trees on the initial training set. This allows for an additional layer

of specificity in the prediction process as it allows for each tree to learn from a subset of attributes after picking the best attribute. By training the trees on a subset, it allows for there to be less correlation between each tree. This leads to more consistent predictions and better generalization of the algorithm. After the trees are created, the prediction is made by passing an instance into each tree and returning the most common prediction among the forest.

The third learning approach we used was neural networks. The structure of the neural network is built around the number of neurons included in the network, threshold, and learning rate. After the test set is normalized, each instance is passed into the neural network, where the first step is calculating the out_k value for each neuron. After all out_k values are calculated, the out_o of the output neuron can then be calculated. The neural network compares the out_o value to the actual label, which gives an error value that can be used to calculate the feedback. By calculating the feedback, it allows the neural network to update the weight for each neuron based upon the impact it had on the error. After the neural network is trained on all test instances, the prediction is made by passing in instances and calculating the out_k and out_o values. If the out_o value is greater than or equal to the given threshold, the prediction label is a 1, else the program outputs a 0.

**Experimental Setup**

The first performance measure we used to evaluate the solutions was accuracy. Accuracy is calculated by dividing the number of correct predictions over the total number of predictions in the subset. This gives insight into what percentage of the predictions were correct. The second performance measure we used to evaluate the solutions was recall. Recall is calculated by dividing the number of correct labels over the number of correct labels plus the number of false negative predictions. For all three of our solutions, we used scikit-learn. In addition, we used a training percentage of 75% and test percentage of 25% for all three models. For the random seeds, we chose the following five seeds: 785, 39, 92, 15, and 1001. Lastly, we used the default scikit-learn hyperparameters for all of the solutions except two parameters. The first is max_iter of the neural network which we changed to 500, and the second was changing recall_score average parameter to None when calculating multilabel datasets (mnist_1000 and hypothyroid).

**Results**

**Average Accuracy Based on Five Random Seeds**

| Name of Dataset | Neural Network | Random Forests | Decision Trees |
|---|---|---|---|
| All Datasets | 0.9658 | 0.9670 | 0.9093 |
| monks1.csv | 1.0 | 0.9722 | 0.9074 |
| mnist_1000.csv | 0.9130 | 0.9452 | 0.7892 |
| hypothyroid.csv | 0.9790 | 0.9945 | 0.9936 |

| | | | |
|---|---|---|---|
| votes.csv | 0.9706 | 0.9560 | 0.9468 |

**Confidence Intervals Based on Five Random Seeds**

| Name of Dataset | Neural Network | Random Forests | Decision Trees |
|---|---|---|---|
| monks1.csv | [1.0, 1.0] | [0.9368, 1.008] | [0.8449, 0.9699] |
| mnist_1000.csv | [0.9004, 0.9256] | [0.9350, 0.9554] | [0.7709, 0.8075] |
| hypothyroid.csv | [0.9685, 0.9895] | [0.9891, 0.9999] | [0.9878, 0.9994] |
| votes.csv | [0.9344, 1.0068] | [0.9120, 1.0] | [0.8986, 0.9950] |

**Average Recall Based on Five Random Seeds (monks1.csv)**

| Label | Neural Network | Random Forests | Decision Trees |
|---|---|---|---|
| Yes | 1.0 | 0.945 | 0.893 |
| No | 1.0 | 1.0 | 0.920 |

Our results for monks1 show that all three approaches were statistically similar. After averaging the accuracy over the five random seeds and calculating the confidence intervals, we observe that there is overlap between neural networks and random forests as well as random forests and decision trees. This overlap means that when comparing all three at the same time, they are statistically similar. After analyzing these results, we most likely observe these results since all three algorithms are capable of binary classification by using the attributes of each instance to train the model. Overall, these results imply that all three approaches are useful in constructing a model to predict binary labels.

**Average Recall Based on Five Random Seeds (mnist_1000.csv)**

| Label | Neural Network | Random Forests | Decision Trees |
|---|---|---|---|
| zero | 0.950 | 0.970 | 0.876 |
| one | 0.972 | 0.980 | 0.921 |
| two | 0.914 | 0.929 | 0.747 |
| three | 0.914 | 0.926 | 0.740 |
| four | 0.900 | 0.944 | 0.802 |
| five | 0.873 | 0.959 | 0.713 |

| six | 0.945 | 0.965 | 0.816 |
|---|---|---|---|
| seven | 0.924 | 0.947 | 0.832 |
| eight | 0.856 | 0.908 | 0.700 |
| nine | 0.882 | 0.921 | 0.750 |

Our results for mnist_1000 show that the random forests were the most accurate approach. When comparing the confidence intervals over the five random seeds, random forests had the highest confidence interval of [0.9350, 0.9554] while having no overlap with the other two approaches. The second best approach was neural networks which had a confidence interval of [0.9004, 0.9256], while the decision trees performed the lowest with an interval of [0.7709, 0.8075]. After analyzing the confidence intervals of the three approaches, we conclude that the random forest's structure of learning from a subset of attributes rather than the full set of attributes gives it the advantage in this dataset. Since there are ten different labels that can be predicted, using subsets allows it to break up possible predictions into multiple different trees, before making the final prediction. Overall, these results display that random forests are more accurate than neural networks and decision trees when predicting for multiple different labels.

**Average Recall Based on Five Random Seeds (hypothyroid.csv)**

| Label | Neural Network | Random Forests | Decision Trees |
|---|---|---|---|
| negative | 0.991 | 0.996 | 0.997 |
| compensated_hypothyroid | 0.860 | 0.984 | 0.975 |
| primary_hypothyroid | 0.758 | 0.961 | 0.919 |
| secondary_hypothyroid | 0.0 | 0.0 | 0.0 |

Our results for hypothyroid show that all three approaches resulted in statistically similar accuracies. When comparing the confidence intervals, there was overlap between all three of the approaches. The only differences come when checking the recalls of each algorithm since the average recall for neural networks for the compensated_hypothyroid label and primary_hypothyroid are much lower than the recalls from using random forests and decision trees. The difference in recalls show that while the overall accuracies are similar for all three approaches, the neural network has less ability to predict positive cases of hypothyroid. As a result, we conclude that random forests and decision trees should be used when trying to predict for a dataset with a small number of positive cases. This is most likely due to the structure of the tree and forests which allow it to predict based on a single path and not rely on weights that were mostly trained on negative inputs. Overall, these results show the need to look at the recall values when determining which approach is best to use for a particular dataset.

**Average Recall Based on Five Random Seeds (votes.csv)**

| Label | Neural Network | Random Forests | Decision Trees |
|---|---|---|---|
| Republican | 0.951 | 0.946 | 0.917 |
| Democrat | 0.983 | 0.962 | 0.965 |

Our results for the votes dataset show that all three approaches performed statistically similar. When looking at the confidence intervals, we observed that all intervals had overlap between each other. When looking at how the dataset was constructed, we observed that the labels used binary classification like monks1, however there were only two attribute values (yes/no) for each attribute. This was the main difference between the datasets since monks1 had attributes with multiple different values. Overall, we conclude that all three approaches are a good fit for the votes dataset as none of them provide a significant statistical advantage over the other.

**Statistical Comparison after Analyzing Results**

| Approach | Statistically Outperformed | Statistically Similar | Statistically Worse |
|---|---|---|---|
| Neural Network | 0 | 3 | 1 |
| Random Forests | 1 | 3 | 0 |
| Decision Trees | 0 | 3 | 1 |

After analyzing the results from the four datasets, we can draw conclusions on which is best to use on certain types of datasets. When training a model to predict on a dataset with binary classification, all three models perform well. When looking at the results from monks1 and the results from votes, there was no statistically significant difference between the calculated confidence intervals. Therefore, we can use either of the three approaches and obtain similar results. In the case we are training a model to make predictions for a dataset with a small number of positive cases, the use of random forests or decision trees would give the most accurate results. The results of hypothyroid showed that while there is no statistical difference in accuracies, random forests and decision trees have a better chance of detecting positive cases. This is seen by observing the higher recall values of the decision trees and random forests. Lastly, when training a model to make predictions for a dataset with multiple different labels, using random forests should yield the most accurate predictions. The results of mnist_1000 showed a clear advantage of using random forests in comparison to neural networks or decision trees when trying to predict for many different labels.

**Conclusion and Future Work**

In this paper, we tested the accuracy and recall of three different machine learning algorithms on four different datasets. By using scikit-learn libraries, we trained and compared neural networks,

random forests, and decision trees by comparing the confidence intervals to find whether there was significant statistical difference between. Our results showed that in cases of binary classification, all three models resulted in similarly accurate predictions. We also observed that random forests outperformed the other two in multi label classification, and that random forests and decision trees were better at identifying positive cases in a dataset. All in all, this experiment serves as a good baseline for comparing machine learning algorithms on different types of datasets.

The development of new and refined models presents an opportunity for future research regarding the comparison of machine learning models. One area that could be researched is to compare the three models studied in this experiment with other models such as naive bayes. In addition, comparing variations of particular models such as convolutional neural networks to other types of neural networks would provide greater specificity when selecting the best model to use. Another area of future research could be using these models on more datasets which could further support and confirm the findings we found in this experiment. It would also provide us with more insight into which model applies best to certain use cases.