This project demonstrates my knowledge of machine learning concepts, specifically Random

Forest Algorithm and Decision Trees. Codes are completed in Python (Spyder).

I used the following resources while conducting this project:

1. http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)

2. https://machinelearningmastery.com/implement-random-forest-scratch-python/

by Jason Browniee

3. https://chrisalbon.com/machine-learning/random\_forest\_classifier\_example\_scikit.html

4. https://www.youtube.com/watch?v=D\_2LkhMJcfY - Random Forest - Fun and Easy Machine

Learning

First I will start off by defining a few basic machine learning terms and concepts.

**Machine Learning:**

Machine learning is the process of making a computer do something without explicitly being programmed to do so.

**Supervised Learning:**

This describes the function F(x)=Y. Supervised learning requires an input variable (x) and an output variable (Y), achieving a basic mapped function. We want to approximate this function so when we have a new value of x, we can predict the output variable Y.

A training dataset would be the teacher, while the algorithm acts as the student. The algorithm creates predictions of the training dataset while the teacher overlooks and corrects.

**Classification**:

A problem that has a categorical output, such as “yes,” “no,” “red,” “blue.”

**Regression**:

A problem that has a real value output, or continuous output like “dollars” or “pounds.”

**Unsupervised Machine Learning:**

When we only have input data (x), following the equation above, and no corresponding output variables (Y).

**Clustering**: A problem which groups data according to similarity

**Association**: A problem where you find patterns or rules based on your data structure.

**Decision Tree:**

A decision tree is a type of supervised learning algorithm which functions for both categorical and continuous input variables. They are mostly used in classification problems. A tree influences classification and decision regression trees (CART). To determine each split of the tree, many algorithms are used. The four most commonly used algorithms to determine the split are: Gini Index, Chi-Square, Information Gain, and.

The **advantages** of a decision tree are: easy to follow, useful for data exploration, minimal data cleaning needed, it can handle both numerical and categorical data types, and it’s a non-parametric method. The **disadvantages** are: over-fitting, non-fit for continuous variables.

**Regression trees** are used when the dependent variable is continuous and **classification trees** are used when the dependent variable is categorical.

Now, we will explore the **Random Forest Algorithm** and its purpose.

Decision trees can utilize greedy selection, taking the best split point of each dataset. Hence these trees are susceptible to a high level of variance (when pruning not involved). To minimize the high variance, multiple trees with different samples can be created, forming correlated trees. However, we can force a difference in these decision trees by limiting its features or rows that the greedy algorithm evaluates at each split point while creating each tree. This method is called random forest algorithm. Multiple samples of a dataset are used, and a tree is produced for each one making its own split. Hence, this results in trees which are uncorrelated to one another, giving us a better and diverse performance and combined prediction.

The dataset that I will be using for this project is the **Sonar Dataset**, from the UCI Repository website. Here is a little background on the dataset: The dataset’s “The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock” (archives.uci.edu). It is a multivariate dataset with 208 instances or obseervations. The variables are continuous and in range of 0 to 1. “M” is the string for mine and “R” is the string for rock, which will be converted to integers 0 and 1.

I followed Browniee’s steps from machinelearningmastery.com to understand this process. Here are how the following algorithms will be formed:

The sample of input values are chosen randomly without replacement, so inputs are considered once when finding the split point with the lowest cost.

The CSV file I am working with is saved as **sonar.all-data.csv**.

**get\_split:** function which finds the split point with the lowest cost. It takes in a dataset and fixed input values to evaluate the input argument.

**test\_split:** helper function which splits the dataset by the **gini\_index()** algorithm and a candidate split point. This evaluates the cost associated of any split by the groups of rows made.

**features**: variable we use for the randomly created features by randomly selecting feature indices and adding them to a list. This features list is then enumerated to certain values in the training dataset and read as split points.

Here a is a run-through of how the code runs:

First, the dataset is imported. The string values are converted to numeric ones valued at 0 or 1. This is done by the helper functions **load\_csv()**, **str\_column\_to\_float()**, and **str\_column\_to\_int()**.

These functions prepare the dataset.

Next, we will build and evaluate k models and to approximate the model for new data. The K models will be generated to estimate the performance as the mean model error. The helper functions **cross\_validation\_split()**, **accuracy\_metric()** and **evaluate\_algorithm()** all use classification accuracy to evaluate each model.

Then, we will construct CART algorithms (which will adapt as bagging) as well as **test\_split()**, which splits datasets into groups. **gini\_index()** evaluates split point. The functions **to\_terminal()**, **split()** and **build\_tree()** are implemented to create the single decision tree predict() to predict with a decision tree; subsample() makes a subsample of training dataset and **bagging\_predict()** to make a prediction with a list of decision trees.

Finally, **random\_forest()** is created, which generates a list of decision trees from subsamples of the training dataset, and then uses that to make predictions.

Note: the **get\_split()** function is the key difference between random forest and bagged decision trees. The function makes a small adjustment to the way the trees are created (Browniee).

After running the code, here are our results:

Trees: 1

Scores: [56.09756097560976, 63.41463414634146, 60.97560975609756, 58.536585365853654, 73.17073170731707]

Mean Accuracy: 62.439%

Trees: 5

Scores: [70.73170731707317, 58.536585365853654, 85.36585365853658, 75.60975609756098, 63.41463414634146]

Mean Accuracy: 70.732%

Trees: 10

Scores: [82.92682926829268, 75.60975609756098, 97.5609756097561, 80.48780487804879, 68.29268292682927]

Mean Accuracy: 80.976%

So we can see that from Random Forest algorithm, for tree 1 we generate a 62% accuracy; for Trees 5 we generate a 71% accuracy; for Trees 10 we generate a 81% accuracy. Hence, the higher the number of trees, the higher the accuracy gets. This is the whole point of Random Forest: to create more accurate predictions by the use of more trees.