Leveraging the human-machine interface for semi-supervised learning

Doreen Atta Aikins (doreena@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr Michelle Lochner
African Institute for Mathematical Sciences (AIMS), South Africa

23 May 2019

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Humans have the ability to assign instances of objects into groups based on similarity (i.e. clustering), often with almost no misclassification. However, with a big collection of such instances, grouping instances by humans may be economically expensive and time consuming. This work investigates machine learning techniques, semi-supervised learning methods in particular, for automated grouping of instances. Semi-supervised learning has attracted the interests of many researchers and various methods have been proposed in the literature. In this work, we apply HDBSCAN[1] to cluster (and then classify) unlabelled instances. To test our method, we consider the random forest classifier on a benchmark, trained with already labelled instances; both methods give almost the same performance. This indicates that grouping of unlabelled instances can be indeed be automated.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Doreen Atta Aikins, 23 May 2019

[1] Hierarchical Density Based Spatial Clustering of Application with Noise.

Contents

1. Introduction

The world is undergoing a data revolution and astronomy is no exception. The community will be experiencing big data opportunities with the two upcoming large telescopes: Large Synoptic Survey Telescope (LSST) and Square Kilometre Array (SKA). The discovery potential of LSST and SKA will be large and as such traditional analysis will not give room for the community to fully unravel their potential. LSST is under construction in Chile and is estimated to detect about 10 million objects in the sky each night (Collaboration et al., 2009). SKA is being built in South Africa and Australia and is estimated to detect around 100 petabytes of celestial objects each day[1].

In the era of all these impressive advancement in technology, the storage and collection process of data has become easier than ever before but a realistic processing of such data remain an important question to address. Extracting information or discovering patterns in big data—by human—is computationally time-consuming. We aim to use machine learning in this regards.

Machine learning covers various tasks namely: supervised, unsupervised and reinforcement learnings. When supervised and unsupervised learnings are considered as a task, one says semi-supervised learning. This work considers semi-supervised learning. Machine learning has attracted the interests of many researchers and methods have been proposed in the literature. For instance, clustering is a technique that involves discovering relevant hidden patterns and structures in dataset. Specifically, it is the segmentation of unlabelled instances into similar groups (Han et al., 2011). Human in the Loop (HITL) is another method that leverages humans and the machine intelligence to create models. In other words, HITL is a framework that involves human responses into a learning process of a machine to improve the overall performance. The commonly used HITL is active learning. Active learning is a query system used to find labels for unlabelled instances at minimum cost (Settles, 2009).

In order to achieve our aim, we will use the clustering algorithm, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) to train a classifier from training data. The choice of the HDBSCAN is motivated by the fact that algorithm can locate dense regions in our dataset and also robust to noise (Campello et al., 2015). Following this, we find the number of training points, evaluate the performance of HDBSCAN classifier and produce a learning curve given the number of training points.

Also, we will use one of the supervised machine learning algorithm, random forest as a benchmark for comparison. Random forest has been shown to outperforms other classifiers in many problems (Ramadevi et al., 2015). Random forest is an ensemble of decision trees, where each single tree follows a set of rules to predict the class of an instance (Breiman, 2001). The prediction of random forest is defined as averaging or major voting of the predictions of the individual trees. We train random forest model on the labelled sample and record the accuracy. Our results show that random forest classifier gains from experience and improves on its accuracy as the size of the training set increases. Also, the HDBSCAN classifier improves on its prediction accuracy given fewer training points.

The remaining chapters of this project is organised as follows. In Chapter 2, we review some fundamental machine learning terminologies. Chapter 3 describes the two machine learning algorithms considered in this work. Briefly we explain ensembles methods and in more details discuss Random forest algorithm. Chapter 3 introduces HDBSCAN and Chapter 4 summarizes our results and discussions for the two algorithms. Finally, we conclude and make future suggestions in Chapter 5.

[1] adopted from https://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=215,accessed on 13th May,2019

2. Machine Learning Overview

Machine learning is in every way affecting the world in which we live in many diverse ways, from search engines like Google to self-controlling cars. It is an application of statistical and computational methods to create algorithms that model data and make accurate predictions on unseen data without being explicitly programmed (Russell and Norvig, 2016). In other words, it is analyzing processed data and using it to make predictions, identifying whether the predictions were correct and if otherwise, learning from the mistakes to improve on future predictions.

Data consists of explanatory variables (features) and sometimes a response variable (target). Suppose we have $X_1, X_2, \ldots, X_p$ features, a target Y and there exit an undetermined association or function f between these features and the target. Then

$$Y = f(X_1, X_2, \ldots, X_p) + \epsilon,$$

where ϵ is an random error term with mean 0. The function f cannot be easily be determined and so one may use machine learning techniques to get an estimate $\hat{f}$ which is given by

$$\hat{Y} = \hat{f}(X_1, X_2, \ldots, X_p).$$

Given a set of N training data $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n)\}$ such that $\mathbf{x}_i$ is the feature vector and y_i be the outcome for each data point. Under supervised learning, our aim is to find a function $\hat{f}$ from the hypothesis space, H (a space containing possible functions) such that the error between f and $\hat{f}$ tends to zero for any observation.

2.1 Parametric and Non-Parametric Methods

The aim of any machine learning project is to employ algorithms in order to estimate the unknown function f. These algorithms can be grouped into (Han et al., 2011):

2.1.1 Parametric Methods. In a parametric model, we may assume the structure of the function which consists of a fixed number of parameters. This method involves two steps:

1) select a functional form of the function. For example, we may assume f is a linear mapping in X. That is $f(X) = \beta_o + \beta_1 X_1 + \cdots + \beta_p X_p$.

2) estimate f. Here, we may generate a predictive model by estimating the coefficients (in our case, β_i-values).

This type of method is easy to understand, compute the parameters and interpret results. One major problem associated with this method is the likelihood of a mismatch. The chosen functional form of f could be a poor fit. The actual unknown function may be totally different from the selected model which in that case the approach will produce poor results.

2.1.2 Non-Parametric Methods. With non-parametric methods, there is no prior knowledge of the functional form of the function of f. Algorithms learn the functional form from data. The learner requires large training data in order to accurately approximate f. The disadvantage of non-parametric method is that it is time-consuming as it does not boil down to just finding f but often estimating more parameters.

2.2 Model Flexibility versus Interpretability

Generally, there is a trade-off between flexibility and interpretability. As per what problem is tackled, one may have to choose between either of the concept. If our main concerned is inference, then one may possible use simple methods as they are very understandable and easy to interpret. For example, a linear model is a less flexible method that can be a good choice for finding the relation between data features X_i's and output Y.

On the other hand, if prediction is the primary interest and interpretability is not a concern, then one may use a complex model that is very flexible in learning almost any kind of feature variable relationship. Although having more flexibility may seem to be desirable but this is not always the case since the selection of the model may lead to overfitting.

2.3 Explaining Different Machine Learning Algorithms

There are many machine learning techniques which enable computers to learn from data. Depending on the nature of the problem they maybe grouped into (Russell and Norvig, 2016):

2.3.1 Supervised Learning. Is a type of task which involves learning from labelled data. That means, it is a system where both variables X_i and labels Y_i are provided. Algorithms under this system can be further categorized into regression and classification. Regression involves predicting continuous output whereas classification is the process of predicting qualitative output. Examples of supervised learning algorithms are linear regression, random forests and support vector machines.

2.3.2 Unsupervised Learning. For unsupervised learning, data is unlabelled. Unsupervised model trains on dataset to discover information and make decisions on the unlabelled data. This is mostly achieved through clustering with a particular distant measure. Examples include the k-means algorithm and hierarchical clustering.

2.3.3 Semi-supervised Learning. Is a learning task that combines both supervised and unsupervised techniques. In other words, algorithms learn from dataset that consists of label and unlabelled data, mainly unlabelled. For example, consider a dataset with some portion of it of being labelled and the remaining portion unlabelled, one may used the label part to make a pretty good guess for the unlabelled data. Following this, a model can be built to make predictions on unseen data.

2.3.4 Reinforcement Learning. Is a kind of machine learning algorithm in which an agent (machine) learns from the interactions with its surroundings. This is achieved by allowing the agents to take a series of actions that are accompanied by rewards. The learning system is positively rewarded when it performs the right task whilst it is reprimanded for an incorrect action. This type of technique aims to find a model that maximizes its rewards.

2.4 Component of Machine Learning Algorithms

There are several machine learning algorithms and they are made up of three components (Domingos, 2012).

2.4.1 Representation. The ability of a model to learn accurately depends on how data is represented. When information is made computationally friendly, representation is basically selecting the hypothesis space and choosing which model it can possibly learn. Examples includes decision trees, random forest, k-nearest neighbor and many more. We will briefly discuss decision tree and random forest but will throw more light on them in the next chapter.

Decision tree learning (Alpaydin, 2009) is a hierarchical non-parameter model that uses a tree to learn from data to make predictions. Thus, given a test data, the labels are derived by passing the data points down from the root to a decision node (that contains the label of prediction).

Random Forest (Breiman, 2001) is a method that combines randomized decision trees to created a predictive model. The predictive model is obtained by averaging (majority voting) the outcomes of the trees for a regression (classification) problem respectively.

2.4.2 Evaluation. Evaluation functions (also known as scoring functions) are an essential part of an algorithm. They assess the performance of all possible models in order to distinguish the good from the bad candidates. The best model is one with the smallest value of the scoring function. We will discuss the most commonly used scoring functions.

1. First and Foremost, the mean absolute error (MAE) measures the average difference between the estimated values $\hat{y}_i$ and target values y_i. Mathematically, it can be represented as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{2.4.1}$$

2. The mean square error (MSE) is defined as the average of squared difference between the actual outputs y_i and estimated values $\hat{y}_i$. Mathematically, it can be expressed as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{2.4.2}$$

   The advantage of using MSE is that it is easier to compute it derivative.

3. The misclassification error (E_{rr}) is the average of incorrect classification. It is given by

$$\text{E}_{\text{rr}} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \tag{2.4.3}$$

   where y_i and $\hat{y}_i$ are the actual outputs and estimated values respectively and $I(y_i \neq \hat{y}_i)$ is an indicator variable that show 0 (correctly classified) if $y_i = \hat{y}_i$ and 1 otherwise.

4. Accuracy is a fraction of correct predictions over the total number of observations.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predicted observations}}. \tag{2.4.4}$$

5. The confusion matrix is another measure metric used to evaluate the performance of classifiers. Considering a binary system, a confusion matrix can be explained through a table with four different components of predicted and true values. It is composed of true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

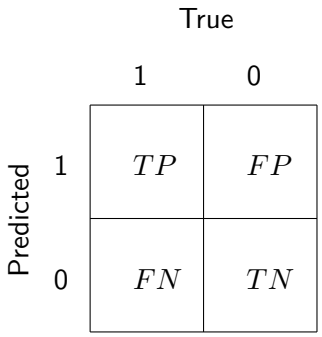


Figure 2.1: *A confusion matrix with two-class data; positive (1) and negative (0).*

Components of Matrix:

- TP is that instance where the classifier predicts positive and this tallies with the true class (positive). Thus the model is classifying correctly.
- TN also refer to that instance where the model makes a correct prediction of the negative value.
- FP refers to the situation where the classifier makes a wrong prediction of the negative value. That is it classifies negative as positive.
- FN represents the case where the classifier predicts negative instead of positive value.

These components are used to calculate the accuracy, recall, precision false negative and false positive rate (Fawcett, 2006).

i. As stated in Equation (2.4.4), the accuracy for the binary classification is as follows

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}. \tag{2.4.5}$$

ii. Recall is used to measure the proportion of positive that are correctly classified as positive. It is therefore calculated as

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.4.6}$$

iii. Precision is fraction of the correctly predicted positive over the total predicted positive.

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.4.7}$$

iv. False Positive Rate is used to measure the proportion of all negatives that are incorrectly identified as positive.

$$\text{False Positive Rate} = \frac{FP}{FP + TN}. \tag{2.4.8}$$

v. False Negative Rate is used to measure the proportion of all positive that are incorrectly identified as negative.

$$\text{False Negative Rate} = \frac{FN}{TP + FN}. \tag{2.4.9}$$

2.4.3 Optimization. There is a need for a search process to help select the best cost function from alternatives. Optimization algorithms help find the optimal parameter values that minimize loss on data. Examples of optimization algorithms are greedy search, gradient descent, linear programming and others.

Gradient descent (also called steepest descent) is a commonly used optimization technique. It is given by (Chong and Zak, 2013)

$$\mathbf{x_{n+1}} = \mathbf{x_n} - \gamma \Delta \mathbf{F}(\mathbf{x_n}), \tag{2.4.10}$$

where γ is a weighting factor (learning rate) that controls the steps size, $\mathbf{F}$ is the cost function and $\mathbf{x_i}$ is the position of $\mathbf{x}$ at i-th iteration. Computing the gradient with smaller weighting factor is time-consuming but it is often preferred to larger ones since the algorithm will be very accurate in finding the minimum point.

2.5 The Learning Process.

Any learning task comprises the following procedures (Brownlee, 2019):

1. Problem identification

   Defining and explaining the problem is the first crucial step to be considered. This involves describing and stating clearly the challenges and exploring potential strategies for solving them.

2. Collection of data

   Accumulating useful data that is in with line with the problem statement is very important since the performance of any model depends on the number and quality of the dataset it uses.

3. Data preparation

   This stage needs human expertise to clean the data in order to make it as good as possible for the machine learning project. Cleaning may involve figuring out missing values, deleting or correcting incorrect data values, conversion of data types, dealing with data errors and others.

4. Model Selection

   There are several machine learning algorithms for different tasks. We choose the model that is suitable for the problem statement and data.

5. Training

   During this stage, the preprocessed data is fed to the selected model. Before the data is used to build the model it is splitted into training and validation set. To throw light on the mentioned datasets (James et al., 2013):

   i. A training set is used for learning, that is to find the parameters of a model.

   ii. Validation set also known as a hold-out set is used to measure the unbiasedness of the model after training.

   The model together with the training dataset is trained using any machine learning algorithm.

6. Evaluation

   After training, the model is evaluated on the validation set to measure its performance. One or several metrics can be used to evaluate model performance. Some examples of evaluation metrics are mention in Section 2.4.2.

8. Parameter tuning

   This step involves tweaking the hyperparameters to achieve the best performance of the model prior to its performances in the evaluation stage.

7. Prediction

   Finally, a testing set is fed to the model with optimal parameters to make a prediction of the targets

   i. Testing set is an unseen data sample used to evaluate the performance of the final model.

The steps outlined depict that machine learning projects have a smooth process flow and so researchers have it is easy to successfully establish and complete a given work. But this is not the case, as they something encountered some problems such as curse of dimensionality, bias-variance trade-off and many more.

2.6 Curse of Dimensionality

The curse of dimensionality is one of the challenges in machine learning. It occurs when one is dealing with data in a high dimensional space. The more dimensions or features one add, it becomes tedious for the model to learn since the growth of data points increases at an exponential rate and as results become highly sparse. For example, this project uses a dataset of resolution images $8 \times 8 = 64$ pixels that is 64 dimensions.

2.7 Tuning Parameters

Another concern in machine learning projects is hyperparameter tuning. Any machine learning task seeks to find a learner that minimizes a loss function. Let $\mathcal{M}$ be a learner given by $\mathcal{M} = \mathcal{A}(\mathbf{X}^{tr},\ \lambda)$, where $\mathbf{X}^{tr}$ is the training set and λ is a set of hyperparameter. Then $\mathcal{M}$ minimizes a predefined loss function $\mathcal{L}(\mathcal{X}^{te};\ \mathcal{M})$ on a validation dataset $\mathcal{X}^{te}$.

Therefore, by (Claesen and De Moor, 2015), tuning parameters is equivalent to finding a set of hyperparameters λ^* that produces the best model $\mathcal{M}^*$ with the lowest loss value .

$$\lambda^* = \underset{\lambda}{\text{argmin}} \mathcal{L}(\mathcal{X}^{te};\ \mathcal{A}(\mathcal{X}^{tr};\ \lambda)). \tag{2.7.1}$$

The selection process can be done using an exhaustive search (grid search) that uses all the possible combination of parameters in a given set or a random search where parameters are selected randomly from the subset of the given set. Some examples of hyperparameters are the number of trees in a random forest network introduced in Section 2.4.1, the activation function in a neural network, the number of clusters to use in k-nearest neighbors and so on.

2.8 Bias-Variance Trade-off

A learning method must be consistency in generalizing a training set as well as a test set. If a model has 98% accuracy on training dataset and performs badly on a test set having an accuracy of 30% when it could have done better. Then the phenomena overfitting has occurred. The model has learnt the random noise in the training data.

The concept of overfitting can be understood from the prediction test error which can be subdivided into variance and bias (James et al., 2013).

Variance is the amount of change in the prediction of estimate $\hat{f}$ when a different training dataset is considered. That means the training dataset can influence the choice of the functional form of a model with regards to the quantity of parameters. A low variance implies changes in the training dataset results in small changes in the prediction of the target function whereas a high variance suggests changes in the dataset will lead to large changes in the estimate of the function. Generally, parametric models have low variance because they are less flexible in contrast with non-parametric model that have high variance.

Bias is the error due to assuming certain conditions of the model in order to make learning easy. The less assumptions made about the functional form of the model the smaller the bias error and vice versa. Thus, non-parametric model have low bias as compare to parametric model which have high bias.

As has been noted, non-parametric model usually have high variance but a low bias whilst a parametric model have low variance but high bias. The ideal situation for a learner would be having low bias and low variance. This is very difficult to achieve in practice because of the trade-off between these two error terms. That is, as one decreases the bias it will lead to an increase in variance and vice versa. However, there are some methods to help find the optimal bias or variance for a model. They include cross-validation, ensemble learning, dimensionality reduction and so on.

2.8.1 Cross-Validation. The problem of not having enough training data to fit a model prevent the use of the train and test split approach. Cross-Validation (CV) is an essential technique where a model uses all the data during training phase and indirectly for testing. In other words, CV is one of the resampling method used to either assess or select a machine learning project. It is used to calculate the test error of a model so that one may be able to measure its performance or to choose the appropriate level of flexibility (James et al., 2013).

One commonly used CV is k-fold cross-validation. This method involves randomly splitting the training set into k-groups (folds) each of equal size. In a classification setting, the first group is used as the held-out dataset and the remaining set of examples are used to fit the model. Then prediction can be made for the reserved dataset in order to estimate the misclassification error (or MSE for regression). The process is repeated for the second group until the k-th group ensuring that all groups have been used as a validation set. The k-fold CV is expressed as

$$\mathrm{CV}_{(\mathrm{k})} = \frac{1}{\mathrm{k}} \sum_{i=1}^{\mathrm{k}} \mathrm{E}_{rr_i}. \tag{2.8.1}$$

The advantage of CV is that it addresses the problem of high variability in the test error rate under the validation set approach stated in Section 2.5. It also solves the issue of overestimated test error rate as a large portion of the training set is used as compared to the validation set technique.

To summarize this chapter, we have introduced some machine learning concepts. Also, we discussed some main machine learning issues such as overfitting and curse of dimensionality. In the upcoming chapter, we present the random forest and HDBSCAN algorithms.

3. Machine Learning Algorithms

In this chapter, we will first give a brief explanation of some ensemble methods and later focus more on random forest. The second part of this chapter will dicuss clustering anaylsis in particular HDBSCAN.

Diversity is a good virtue that has been embraced by many computational communities. This concept encompasses different information from various sources to help reach a final decision. Not only has it proven worthy in many real-life applications, the field machine learning is not an exception. The idea of different types of models coming together as one to make predictions have come in handy in the machine learning community and other fields such as medicine. This process is known as ensembles learning. A study by (Ramadevi et al., 2015) shows that an ensemble (random forest) outperforms other machine learning algorithms such as decision trees, logistic regression and support vector machine in the analysis of breast cancer dataset. They concluded that it is to be preferred because it runs efficiently on all the four dataset used in this experiment. We will discuss some commonly used ensemble techniques before we talk about random forest.

3.1 Ensemble Methods

Ensemble methods are basically techniques used to find a suitable hypothesis that fit well on test data by relying on the contributions from several different models. Ensembles have contributed greatly in solving several machine learning problem such as feature selection, reducing variance and bias (thereby yielding better results), among others. The following are some types of ensembles:

3.1.1 Boosting. Is using learners that are weakly correlated but their performance is better than random guessing to build one final learner that is highly correlated with the target. Simply put, it is the conversion of weak learners into strong learners. Adaptive Boosting (AdaBoost) is one of the most widely used algorithms for boosting.

With AdaBoost (Freund et al., 1996), weak classifiers are sequentially trained on the weighted training dataset. The algorithm initially starts with fitting a model on equally weighted data points. Subsequently, the weights are adjusted (giving increasing values to wrongly classified data and smaller values for the right classification). The newly updated weighted dataset is used to train another model and the process is repeated until we reach a better model.

3.1.2 Stacking. Is an approach that involves fitting individual models on a training dataset and using their predictions to generate a new training set for a meta-classifier.

3.1.3 Bagging or Bootstrap Aggregation. (Breiman, 1996) is also one of the ensemble methods that fit models on bootstrap instances (randomly sampling from the training data with replacement) to produce the prediction either by averaging or majority voting of the results from the single model. That is, given training data D, we randomly generate a bootstrap sample D_i and train model f_i on a D_i. Then the process is repeated for m times in each case a subsample is obtained with a given model. Therefore, with a test set x

$$\text{Bagging} = \frac{1}{m}\sum_{i=1}^{m} f_i(x).$$

3.1.4 Random forest. (Breiman, 2001) is an ensemble technique that uses bootstrap instance to create a lot of decorrelated decision trees to perform either a classification or regression task. In bootstrap aggregation, the entire features are used to build the required number of trees which may results in correlation among trees. Random forest improves upon the limitation of the bagging method by using a random subsample of the features from the full space at each time of split in a tree. Thus reducing variance and increasing the predictive power of the model. A decision tree structure can be likened to a real tree with roots, branches and leaves yet it is drawn upside down. In other words, the leaves are at the bottom and the root is at the starting point. A decision tree is useful for solving both regression and classification problems. An example of a tree is illustrated below.

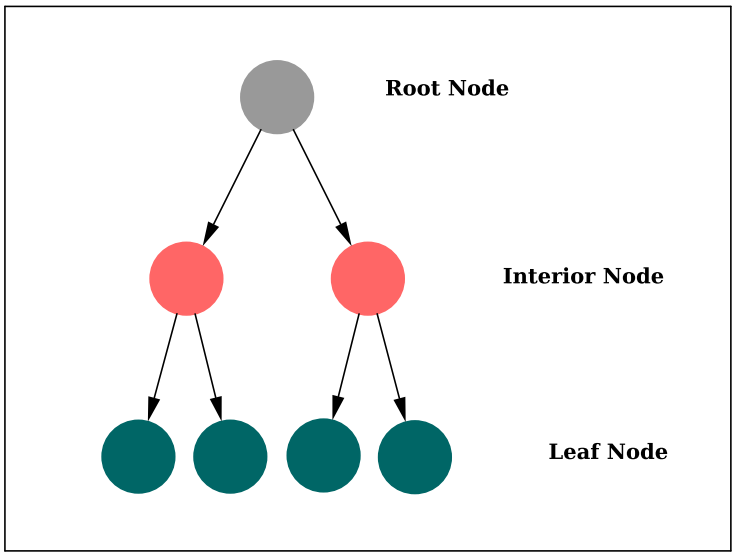


Figure 3.1: *An example of a decision tree. A tree is made of the root node, interior nodes and leaf nodes. The root node is the fundamental part of a tree serving as the building blocks. The interior nodes are intermediary nodes that have edges moving towards and away from them. Lastly, the leaf nodes (decision or terminal nodes) are the last nodes on the tree that have not links out them.*

A classification or regression tree builds a learner mainly by recursively partitioning the feature space (Strobl et al., 2009). That means, a tree begins with a the most contributing feature as the root node. Then a test is apply to the root node and is further divided into interior nodes. The partitioning process is repeated until a leaf node (constituting instances with similar response) is obtained in order to ensure homogeneity (pureness). Pureness can be defined as a state where a node contains only the same kind of observations whilst a node with different observations can be described as impureness. The tree-method uses the concepts of information gain (IG) to split on a feature at each stage. The feature with the maximum IG will be selected splitting node. Also, before and after each split IG is used to ascertain the level of impureness among features.

Given training data, $D = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n)\}$ such that $\mathbf{x_i}$ is the feature vector and y_i be the outcome for each data point.

Then IG is the difference between impurity before and after split.

$$\mathrm{IG} = \mathrm{H}(S) - \sum_{j=1}^{k} \frac{|S_j|}{|S|} \mathrm{H}(S_j), \tag{3.1.1}$$

where H is the impurity measure, S is a subset of instances from the D, S_j are the instances that belong to class j and $|S|$ and $|S_j|$ be the cardinality of S and S_j respectively.

There are different impurity measures for either a classification or regression task. In a regression tree analysis, the tree uses MSE (stated in Equation 2.4.2) as measure of impureness. Likewise in a classification tree analysis, there are various measures that can be apply to the trees. Some of the measures are as follows (Friedman et al., 2001):

Let p_i be the proportion of instances belonging to class i. Then

1. Misclassification impurity is defined as

$$\text{Misclassification impurity} = 1 - \max\{p_i\}. \tag{3.1.2}$$

   A feature node with homogeneous element has misclassification error of zero because the max $\{p_i\} = 1$.

2. Gini index is also defined as

$$\text{Gini index} = 1 - \sum_{i=1}^{k} p_i^2. \tag{3.1.3}$$

   Similarly, a feature node with all its observations from a single class has gini index of zero since $p_i = 1$ and other p_i's $= 0$.

3. Cross entropy (Shannon entropy) is defined as

$$\text{Cross entropy} = -\sum_{i=1}^{k} p_i \log_2 p_i. \tag{3.1.4}$$

   A pure node has cross entropy value of zero because $1 \times \log_2(1) = 0$.

   All the above measures have values always between 0 and 1. Gini index and misclassification error have one indistinguishable property. That is both indices of impurity measure have the same maximum value. Suppose we have a feature with m equal number of instances for each class. Then the proportion of instances belonging to class i is $\frac{1}{m}$. Hence the gini-index has maximum value of

$$1 - m\left(\frac{1}{m}\right)^2 = 1 - \frac{1}{m}$$

   and that of misclassification is

$$1 - \max\left\{\frac{1}{m}\right\} = 1 - \frac{1}{m}.$$

   Unlike gini index and misclassification error, Shannon entropy reaches its maximum value at

$$-m\left(\frac{1}{m}\right)\log_2\left(\frac{1}{m}\right) = \log_2 m.$$

From the iterative partitioning process in the creation of a tree, a node may be a leaf node when all instances in the training set are in the same category (purity). However, if a tree is grown fully to achieved the lowest impurity value for each leaf node overfitting may sometimes occur. Thus, a tree

needs to stop growing at some point. The algorithm keeps partitioning the nodes until a stopping criterion is meet (Han et al., 2011). Examples of some mostly used stopping rules are where there are no more features for further partitioning, setting the maximum tree depth, when your impurity values is less than a certain threshold and when a node contains less than minimum samples threshold and so on. Below is a framework of random forest.

Algorithm 1 Random Forest Algorithm

procedure RANDOMFOREST(D, P) ▷ D is a set of training set and P is feature space
  $T \leftarrow \emptyset$
  for $b = 1, ..., B$ **do** ▷ B is the number of trees in the forest
    $D_i \leftarrow$ a boostrap sample of size i from D
    $T_b \leftarrow$ Grow a tree from m selected features from P
    $T \leftarrow T \cup \{T_b\}$
  return T ▷ T is the ensemble

Random forest combines all the tree build by averaging in the case of regression and plural voting for classification analysis. Hence, a single meta-classifier is T is given as

1. Classification: mode of all decision trees $T_1, \ldots, T_B$.
2. Regression: $T = \frac{1}{B} \sum_{b=1}^{B} T_b$.

With random forest, the number of trees(B) and the number of features (m) to split on are important parameters. Usually, (Friedman et al., 2001) recommends $\lfloor m \rfloor$ number of features should be used in each split for a classification problem whereas $\lfloor \frac{m}{3} \rfloor$ for regression. A good way of knowing which tree to select depends on its performance on unseen data. Regarding random forest, there is no need to get a separate test data as we may recall that trees are built on bootstrap subset and thus the remaining training data that weren't chosen can serve as a validation set for tuning parameters and model assessment. These discarded dataset is known as Out-of bag (OOB). Computing the OOB error helps to determine the optimal number of trees for the final model.

Generally, knowing how accurate your ensemble is might not be the only thing of interest. The performance of features is equally an importance information to ascertain. As non-predictive features may be a hindrance to the performance of the classifier and so early detection of these variables will make learning simple and faster when they are omitted from an analysis. Random forest makes available the prediction strength of each feature by using the OOB sample. When a single tree is made, the OOB sample is used to validate the tree and the accuracy result is recorded. Then values of a feature (variable of interest) is randomly scrambled in the OOB sample, keeping the others untouched and again the prediction accuracy is computed. There is a decrease in the accuracy results due to the scrambled data. Finally, the mean of the decreased accuracies over the tress is calculated and used as the performance measure of that particular variable in the forest (Friedman et al., 2001).

We now present the second part of this chapter.

3.2 Clustering Analysis

The act of discovering multiple groups from a set of data such that points of a particular group have high similarity yet are dissimilar to other points in a different group is known as clustering. It is an important data analysis tool that has various applications in other fields like bioinformatic, biology and medicine and many more. In machine learning, clustering is an unsupervised technique that take the form of learning by observation. In other words, it finds patterns and hidden insights in data that have class labels not present. This kind of learning entails dividing the unlabelled datasets into subsets (clusters) such that elements of a subset are similar but dissimilar to other elements in another subset.

As has been noted, the primary aim of any clustering algorithm is to have high intra-cluster similarity (similarity between data points in one cluster) and low inter-cluster similarity (similarity of clusters). The similarity between two data points can be defined in many different ways depending on the clustering method. Some methods use distances as a measure of similarity. Aside dependency on distances, some methods also use the notion of density to defined similarity. The numerous clustering methods can be categorized into partitioning methods, hierarchical methods, density-based method and grid-based methods (Han et al., 2011).

3.2.1 Partitioning Methods. These methods of clustering assume the number of clusters to be formed even before the process is started. The method starts by partitioning n number of data points into k (predetermined) clusters ensuring that there is at least one point in each cluster. After that, it improves the partitioning by using distance measure as a criteria to move points around to finally achieve a good cluster where points are highly related and points in different clusters are different. One of the commonly used partitioning method is k-means.

3.2.2 Hierarchical Method. Unlike the partitioning method, there is no assumption about the number of clusters in hierarchical classification. The hierarchical clustering techniques may be divided into (Everitt et al., 2011):

1. Agglomerative

   This method is the bottom-up approach. At first, every data point forms its own cluster. Subsequently, pairs of clusters are merged if they are similar. The merging process is repeated till a single large cluster is produced. The final cluster becomes the root node of the hierarchical tree.

2. Divisive

   This method is the opposite of the agglomerative. It is also known as the top-down approach. Initially, the algorithm start with all the points in a large cluster and recursively splits the cluster into smaller ones. The splitting process goes on until every point is in its own cluster.

These method can be represented by a hierarchy or tree of clusters called dendrogram. Previously mentioned, in agglomerative method there is a need for a similar measure between clusters (inter-cluster similarity). There are several distance measures namely single linkage method, complete linkage, average linkage and so on. We will discuss the single linkage method.

The single linkage method is a widely used distance measure between clusters. The distance is determined by the nearest pair of elements in each cluster. Thus, the method is also called the nearest-neighbor clustering. Formally, let C_i and C_j be clusters. Then the single linkage is (Han et al., 2011)

$$\text{dist}_{min}(C_i, C_j) = min\{|p_i - p_j| : p_i \in C_i, p_j \in C_j\}. \tag{3.2.1}$$

3.2.3 Density-based method. One limitation of partitioning and hierarchical methods is the inability to filter out noise in the data in order to achieve clusters of arbitrary shapes. Unlike these methods, density-based methods employs the notion of density (number of data points) to find clusters of arbitrary shapes such as 'S' and oval shapes and many more. This is mainly achieved by separating sparse region from dense portion of your data space (Everitt et al., 2011). Examples are Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Ordering Points to Identify the Clustering Structure (OPTICS) and so on.

3.2.4 Heirarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). Is an extension of the DBSCAN by converting the algorithm into a single-linkage hierarchical clustering and later extracting the flat clustering (McInnes et al., 2017). We will briefly describe the DBSCAN algorithm and later discuss HDBSCAN.

DBSCAN algorithm has two user defined parameters ϵ and m_{pts} where ϵ is a radius of a neighbourhood and m_{pts} is the number of objects in your neighbourhood. The main strategy of the algorithm is to connect core points (Definition 3.2.5) and their neighbours to formed a cluster. Formally, let $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of data points where $\mathbf{x}_i$ is a feature attribute and $d(\mathbf{x}_i, \mathbf{x}_j)$ be defined as distance between pairs of objects in D.

3.2.5 Definition. Core and Noise points (Campello et al.). Let $\mathbf{x}_i \in D$, $|.|$ be the cardinality and $\mathsf{N}_\epsilon(\mathbf{x}_i) = \{\mathbf{x} \in D : d(\mathbf{x}, \mathbf{x}_i) \leq \epsilon\}$. If $|\mathsf{N}_\epsilon(\mathbf{x}_i)| \geq m_{\text{pts}}$. Then $\mathbf{x}_i$ is known as a core point with respect to (w.r.t) ϵ . Otherwise, $\mathbf{x}_i$ is a noise point.

3.2.6 Definition. ϵ-Reachable (Campello et al.). Two core points $\mathbf{x}_i$ and $\mathbf{x}_j$ are ϵ-Reachable w.r.t ϵ and m_{pts} if $\mathbf{x}_i \in \mathsf{N}_\epsilon(\mathbf{x}_j)$ and $\mathbf{x}_j \in \mathsf{N}_\epsilon(\mathbf{x}_i)$.

3.2.7 Definition. Density-Connected (Campello et al.). Two core points $\mathbf{x}_i$ and $\mathbf{x}_j$ are density-connected w.r.t ϵ and m_{pts} if they are

1. directly ϵ-Reachable. That is $\mathbf{x}_i \in \mathsf{N}_\epsilon(\mathbf{x}_j)$ and $\mathbf{x}_j \in \mathsf{N}_\epsilon(\mathbf{x}_i)$.
2. transitively ϵ-Reachable. That is, if $\mathbf{x}_i$ and $\mathbf{x}_k$ are ϵ-Reachable and $\mathbf{x}_k$ and $\mathbf{x}_j$ are ϵ-Reachable. Then $\mathbf{x}_i$ and $\mathbf{x}_j$ are ϵ-Reachable.

3.2.8 Definition. Cluster (Campello et al.). A cluster C w.r.t. ϵ and m_{pts} is largest subsample of D such that for $\mathbf{x}_i$, $\mathbf{x}_j \in C$ are density-connected.

3.2.9 Definition. Core distance (Campello et al., 2015). The core distance of a point $\mathbf{x}_i$ w.r.t to m_{pts}, $d_{\text{core}}(\mathbf{x}_i)$ is the distance from $\mathbf{x}_i$ to its m_{pts} nearest neighbor ($\mathbf{x}_i$ inclusive).

3.2.10 Definition. Mutual Reachability Distance (Campello et al., 2015). Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be members of D. The mutual reachability distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ w.r.t m_{pts} is defined as $d_{\text{mreach}}(\mathbf{x}_i, \mathbf{x}_j) = \max\{d_{\text{core}}(\mathbf{x}_i),\ d_{\text{core}}(\mathbf{x}_j),\ d(\mathbf{x}_i, \mathbf{x}_j)\}$.

3.2.11 Definition. Mutual Reachability Graph (Campello et al., 2015). The mutual reachability graph, G_{pts} is a complete graph in which every pair of distinct vertices (data points in D) is connected by an edge which is equal to the mutual reachability distance of the pair of vertices.

3.2.12 Definition. (Minimum Spanning Tree, (Zhong, 2013)). A spanning tree is a subgraph of a weighted undirected graph, G that contains all the vertices of G. A minimum spanning tree (MST) is the spanning tree with the smallest total weight.

The DBSCAN algorithm (Han et al., 2011) begins with marking all the data points in D as 'unvisited'.

Then randomly selects one of these points $(\mathbf{x}_i)$ and determines whether this point is a core or noise point. $\mathbf{x}_i$ is declared as a core point when it satisfies the condition in Definition (3.2.5). Otherwise, it is marked as a noise point and visited. If it is a core point, the neighbours of $\mathbf{x}_i$ are placed in a candidate set N and a cluster C is formed for $\mathbf{x}_i$. Then points in N that are not members of any existing cluster are added to C. After that, mark the unvisited points in N as visited and check if they are core points. If they are core points then their neighbours are again added to the set N. The method keeps adding points from N to C until there is no more points to add (that is N is empty). Finally, the cluster is complete. The procedure is repeated to find other clusters for the remaining unvisited points in the data set. Also according to DBSCAN algorithm (Campello et al.), clusters can be seen as the connected components of a graph where the vertices are the data points in D and vertices are connected by an edge if and only if they are ϵ-Reachable.

HDBSCAN algorithm computes a hierarchical single linkage clustering on a mutual reachable distance space. The approach of computing a single linkage clustering on the reachable distance space would not be able to determine whether an isolated point is a core or noise point at a given the level of hierarchy but the HDBSCAN algorithm captures this information (Campello et al.). The algorithm initially starts by calculating the core distances of each data point and later builds MST of G_{pts} via any efficient algorithm. Following this, MST is extended with edge connecting a vertex to itself (self-loops), where the weight of these self-loop corresponds to the core distance of the vertex. The reason for introducing self edges is to help capture the level in the hierarchy below it a point is declared as noise. Edges in the MST extended are removed in decreasing order of weights. In situations when there is a tide, edges of the same weights are removed at the same time. The outcomes is the HDBSCAN hierarchy. Below is the pseduo-code for HDBSCAN (Campello et al., 2015).

Algorithm 2 HDBSCAN Algorithm

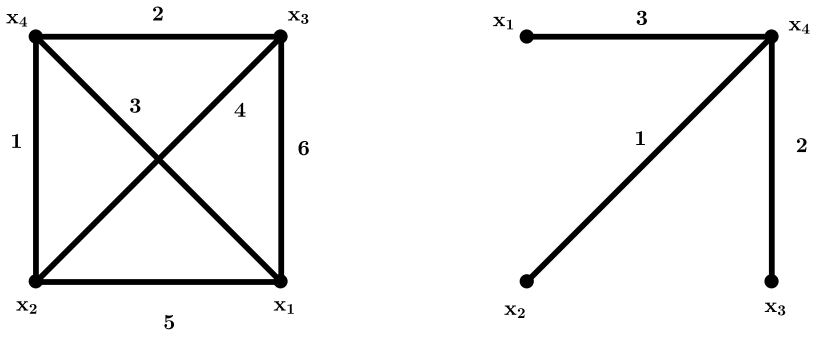
1: For each data object in D, calculate its core distance.
2: From the G_{pts}, compute an MST.
3: Compute MST$_{ext}$, by adding self loops to each vertice of MST with weight equals to its core distance.
4: Extract the HDBSCAN hierarchy as a dendrogram from MST$_{ext}$.
5: For the root node of the hierarchy tree assign all objects the same label (one cluster).
6: In decreasing order of weights, repetitively delete all edges from MST$_{ext}$. In situation where there is a tie, edges must be deleted at the same time.
7: Before each removal, fix the current level of the hierarchical tree to be equal the weight of the deleted edge(s).
8: After each removal, assign classes to the connected component(s) of the MST$_{ext}$. To get the next level of the tree, assign label to a component if it has more than one edge, otherwise label it noise.

Let us consider a simple illustration:

3.2.13 Example. Let $X = \{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4\}$ and G_{pts} be the mutal reachability. We build a MST graph via prim algorithm. We briefly describe the steps involved in prim's algorithm (McInnes et al., 2017).

1. Initialize a tree by randomly selecting a vertex from G_{pts}.
2. Find all the edges that connect the current tree to vertices not yet in the tree, add the lowest weight.
3. Repeat step 2 until all vertices are in the tree. That means we obtain a MST.

Now we construct the MST graph and extend it to obtain the MST_{ext}.



(a) G_{pts} **(b)** MST_{ext}

Figure 3.2: (a) *The mutal reachability graph with weights equal to the mutual reachability distances*; (b) *The extended MST where the core distance of* x_1, x_2, x_3, x_4 *is 2, 0.5, 1.5 and 0.8 respectively. For the sake of clarity self loops are excluded.*

The following results (shown in Table 3.1) were obtained after applying HDBSCAN to the dataset X. At level l= 4, there is a single cluster containing all the data points. Below this level, cluster $\mathbf{C_1}$ splits into $\mathbf{C_2}$ and $\mathbf{C_3}$. At level $l = 2$, cluster $\mathbf{C_3}$ splits into $\mathbf{C_4}$ and $\mathbf{C_5}$ and gradually they disappear. In the case of $\mathbf{C_2}$ it completely disappear from $l = 2$.

l	4	3	2	1.5	1	0.8	0.5
x_1	$\mathbf{C}_1$	$\mathbf{C}_2$	0	0	0	0	0
x_2	$\mathbf{C}_1$	$\mathbf{C}_3$	$\mathbf{C}_4$	$\mathbf{C}_4$	$\mathbf{C}_4$	$\mathbf{C}_4$	0
x_3	$\mathbf{C}_1$	$\mathbf{C}_3$	$\mathbf{C}_5$	0	0	0	0
x_4	$\mathbf{C}_1$	$\mathbf{C}_3$	$\mathbf{C}_4$	$\mathbf{C}_4$	$\mathbf{C}_4$	0	0

Table 3.1: *The HDBSCAN hierarchy for* D. $\mathbf{C}_i$ *means the data point belongs to cluster* i *at the hierarchical level* l *and 0 refers to a noise point.*

Clustering in a low dimensional space looks easy but often machine learning projects deal with high dimensional dataset. That is one has data that contains a lot of features. Being able to explore and visualize relationship between these features is one of the challenges in the field. This is where dimensionality reduction techniques are used. Dimensionality reduction is the process of reducing the number of features in the dataset so that one can explore relationship between fewer features and easily be able to visualize (Raschka, 2015).

3.2.14 T-Distributed Stochastic Neighbouring Entities (t-SNE). Is one of the dimensionality reduction tool used for data exploration and visualization. It convert higher dimensional data into lower dimensional feature space (one, two, or three -dimensional) in order to find patterns (clusters) in the data. The conversion ensures that relevant feature structures are preserved in the low dimensional space (Maaten and Hinton, 2008). Visualization of the dataset can be displayed via scatter plot or histogram.

To conclude this chapter, we have explained the two machine learning algorithms to be used in this project. Now, let us present our results.

4. Semi-supervised Learning Application

As stated in the aims of this study, we should present the results of random forest and the HDBSCAN algorithm. We demonstrate how combing clustering and active learning become powerful tools for exploring data and allowing classification at a minimum cost.

4.0.1 Dataset. In this project, for our analysis we consider the famous handwritten digits from Mixed National Institute of Standards and Technology (MNIST). The dataset consists of 1797 labelled digits, each image is 8×8 pixels in size with integer from range 0 to 16. There are ten classes. That is 0,..., 9. Below are some digit examples:

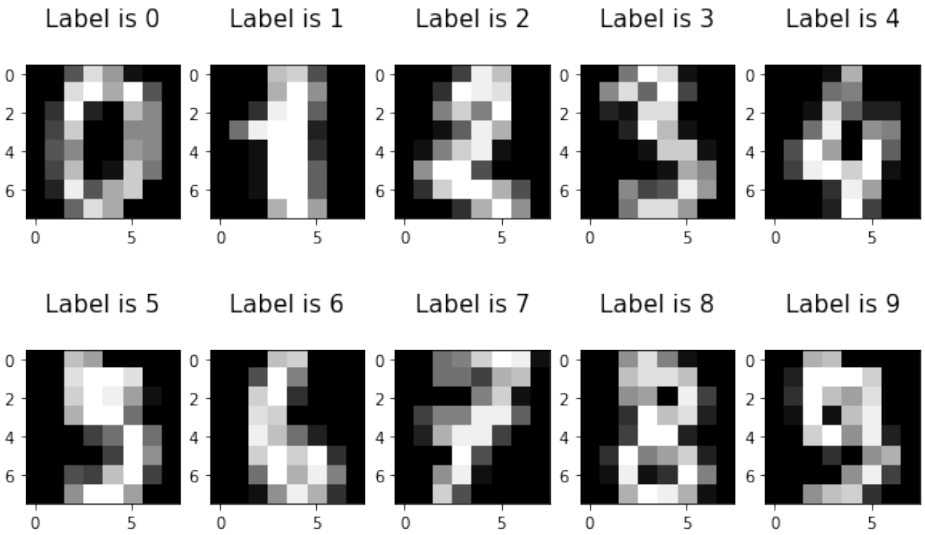


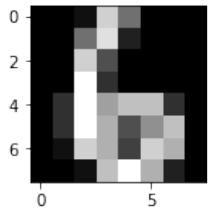
Figure 4.1: *Some examples from the MNIST database.*

4.0.2 Method. For the random forest results, the following method was adopted to obtain Figure 4.3.

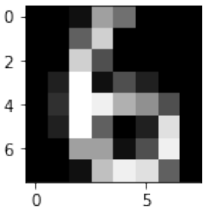
1. We divided our data set into training and test set.
2. After we took 10 instances from the training data and estimate a random forest model. In other words, we took 10 representatives corresponding to the ten target classes and apply the random forest algorithm in order to generate the hypothesis.
3. Following this, we measured the accuracy on the test data. That means we find the fraction of the examples in the test set that are correctly classified.
4. Then we randomly get 10 examples from the remaining training each trail and add it to the current sample to form a new set and repeat step 2 to 4 until we exhaust the training set. Samples in this procedure are selected without replacement and the test set stays the same across all trials.
5. Finally, we get a learning curve for the random forest classifier. That is a plot of the accuracy as the size of the training set increases.

Now let us consider a situation, when our dataset is unlabelled. The question then becomes how do we get labels? In figure 4.2, we explore with three samples from our dataset and calculate the euclidean distances between these three samples. We obtained that the distance between sample 1 and 2 and

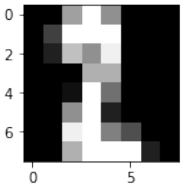
that of sample 1 and 3 are 17 and 47 respectively. The exploration gives us a clue that distances play a major role and thus using a clustering method such as HDBSCAN that has a distance as a similarity measure will help us to find grouping in our dataset.



(a) *Sample 1*



(b) *Sample 2*



(c) *Sample 3*

Figure 4.2: *Three instances from our dataset for exploration.*

Following this, we first cluster our dataset with the HDBSCAN algorithm . Later, we set out to find the number of training points. The number of training points in this part of study refers to the number of user defined labels. To be able to accomplish this idea we rely on feedback loops between the computer and the user since the learned model cannot tell us the labels of the clusters found. That is, each time there is a split in the dendrogram we ask a user for a label for that cluster. The following procedure was used to find classes for the clusters at specific λ values:

1. At a specific λ value, the centroid point of the cluster is obtained. The centroid point of the cluster is a good representative of the cluster since it is the average of all the data points.
2. The centroid point is visualized and a user called out a label (in our case a digits number). This labelled is assigned to that particular cluster.

Finally, the performance of the HDBSCAN classifier was evaluated given the training points. This can be seen in Figure 4.7. The results from the two experiments are present below. In both experiments, the best parameters were selected in order to maximize accuracy.

4.1 Random Forest Results

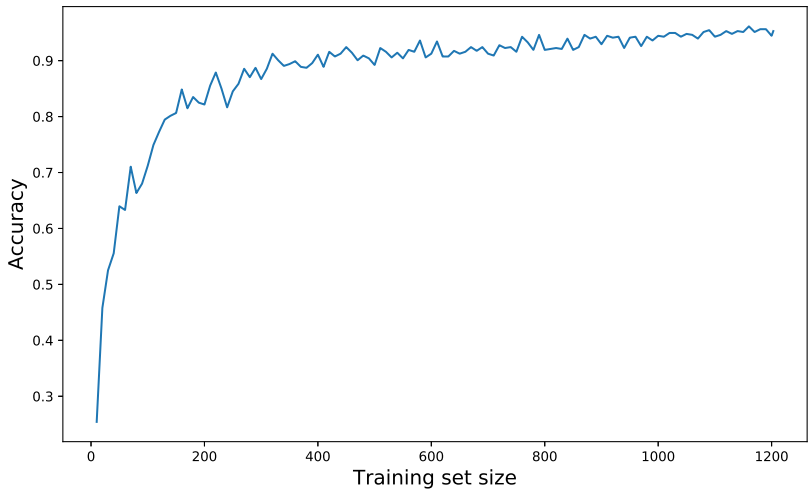


Figure 4.3: *The learning curve for random forest algorithm on 1200 random examples.*

4.2 HDBSCAN Results

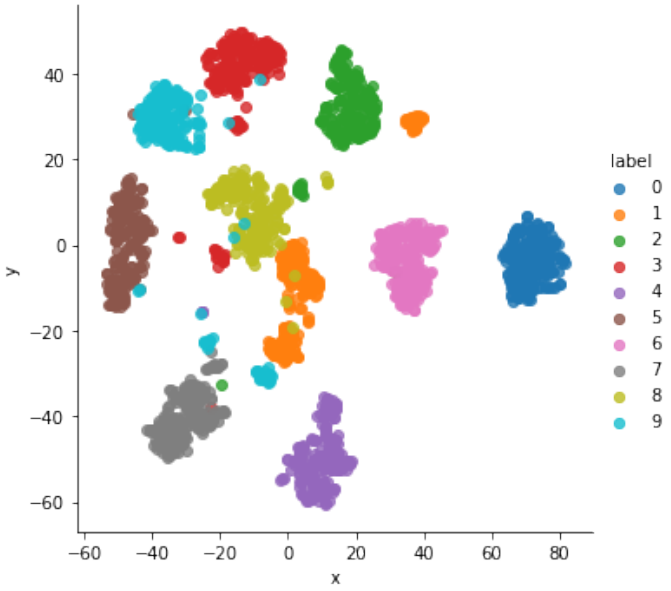


Figure 4.4: *Visualization of MNIST dataset by t-SNE* .

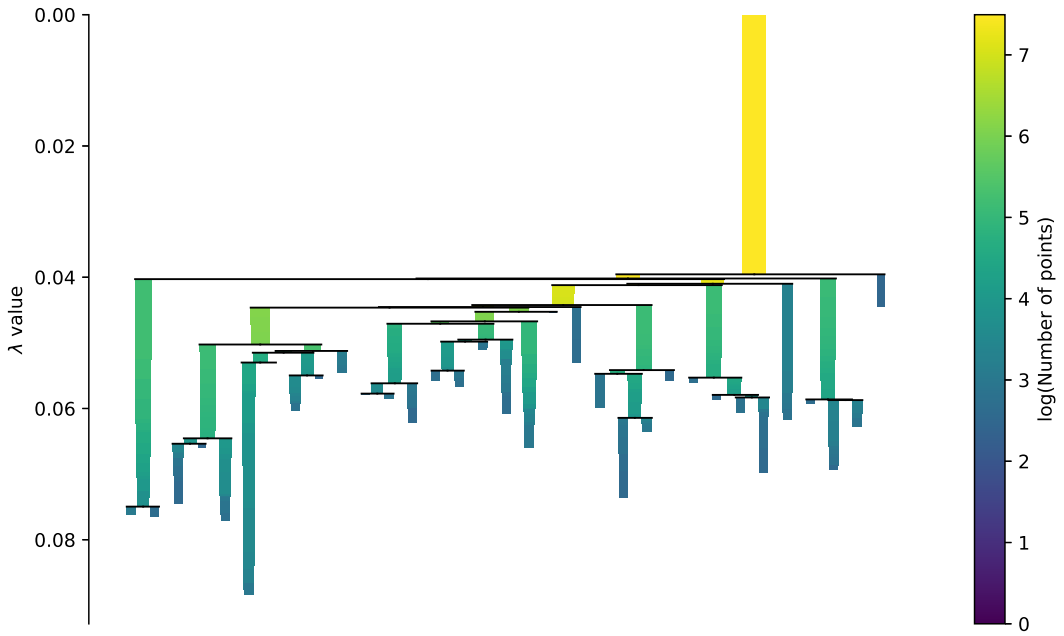
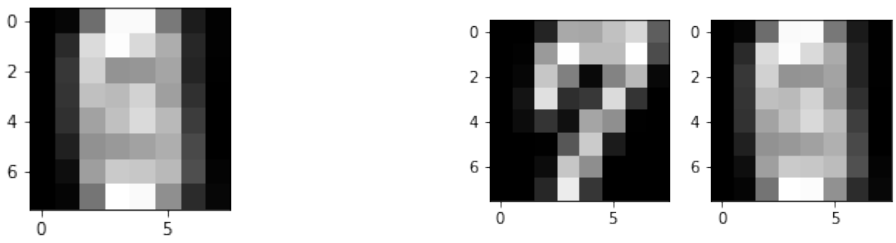


Figure 4.5: *Dendrogram of the HDBSCAN algorithm. The y-axis represent the hierarchy level and the width of the lines is the number of points in cluster at each level.*



(a) $\lambda = 0.03$ (b) $\lambda = 0.03956$

Figure 4.6: (a) *The centroid of the cluster at* $\lambda = 0.03$ *and the user defined label is digit 5* ; (b) *The centroid of the clusters at* $\lambda = 0.03956$ *and the given labels are 7 and 5.*

Table 4.1 shows all the user defined labels moving down the dendrogram. After obtaining all the labels, we went ahead to check the performance of our classifier at each level in the hierarchy. The results can be seen in Figure 4.7.

λ	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}
0.03	5																
0.03956	7	5															
0.04019	7	5	7														
0.04029	0	7	5	7													
0.041	0	7	1	5	7												
0.04120	0	7	1	2	5	7											
0.04424	4	0	7	1	2	5	7										
0.04454	4	0	9	1	2	5	7										
0.04463	4	0	9	1	2	5	6	7									
0.04527	4	0	3	9	1	2	5	6	7								
0.04673	4	0	9	1	2	5	6	3	7								
0.04709	4	0	9	9	1	2	5	6	3	7							
0.04951	5	4	0	9	9	1	2	5	6	3	7						
0.04981	5	4	0	9	9	1	2	5	5	6	3	7					
0.05025	5	4	0	9	9	1	2	5	6	5	1	3	7				
0.05123	5	8	4	0	9	9	1	2	5	6	1	3	7				
0.05150	1	5	8	4	0	9	9	1	2	5	6	1	3	7			
0.0530	1	5	8	4	0	9	1	2	5	6	1	3	7				
0.05415	1	5	8	4	0	9	1	2	5	4	6	1	3	7			
0.05423	1	5	5	8	4	0	9	1	2	5	4	6	1	3	7		
0.05472	1	5	5	4	0	4	9	1	2	5	4	6	1	3	7		
0.05496	1	5	5	4	0	4	9	1	2	5	4	6	1	1	3	7	
0.05530	1	5	5	4	0	4	2	9	1	2	5	4	6	1	1	3	7
0.05617	1	5	4	0	4	9	9	1	2	5	6	1	3	7			
0.05774	9	1	5	4	0	4	9	9	1	2	6	1	3	7			
0.05793	1	5	4	0	4	2	9	1	2	6	1	9	3	7			
0.05832	1	5	4	0	4	2	2	9	1	2	6	1	9	3	7		
0.05862	1	5	4	0	4	7	2	2	9	1	2	6	1	3	7		
0.05872	1	5	4	0	4	7	2	9	1	2	6	7	1	3	7		
0.06143	4	0	4	9	1	6	7	1	2	3	7						
0.06455	0	4	6	7	1	2	3										
0.06537	0	6	4	6	7	1	6	2	3								
0.07495	6	0	1	0													

Table 4.1: *The user-defined labels at each hierarchical level (λ), where C_i is the cluster found at that level. Note that the cluster indices are not of a particular order.*

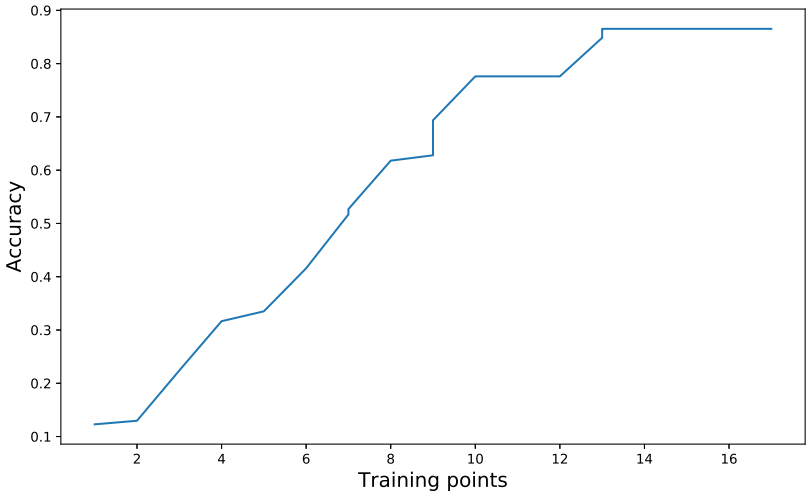


Figure 4.7: *Accuracy of the HDBSCAN classifier as the training points increases.*

4.3 Discussion

In the above section, we have shown the results obtained. Now, we are going to throw more light on the results.

In our first experiment, we explore the question of how much data is needed to learn the random forest model. From Figure 4.3, we noticed that with very little training data the accuracy is very low. However, as more training points were added the accuracy increased. That means the model is getting better by mastering and discovering patterns in the data. Here, we can conclude that about 200 data points are sufficient to build a good random forest estimator with accuracy of about 82 %. Following this, we perform our next experiment.

The HDBSCAN algorithm uses the top-down clustering approach. From Figure 4.5, at λ levels below 0.039, we have all the data points in a single cluster. Then moving the down the dendrogram at $\lambda = 0.4$, the top-most cluster splits into two clusters. This procedure keeps repeating until there is no more splits of clusters. In order to generate a training set for the classifier, we made use of the information provided in the dendrogram. At the root node of the dendrogram, we only acquire one training point. Subsequently, we obtain more training data points when the λ value was increased and this can be seen in Table 4.1.

As has been pointed out in Section 3.2.4, at each level of the hierarchy tree the HDBSCAN algorithm is able to identify points that are being declared noise. To address this problem, a noise point is assigned the user defined label before it was declared noise. From Figure 4.7, we noticed that the HDBSCAN classifier did not perform well when one training point was made available. This is mostly likely due to the fact that every object in the single cluster is given one label. The method improved as the number of labelled points increased, we record an accuracy of about 86% with few training points (about 13 of them) . Therefore, semi-supervised method (HDBSCAN classifier) shows improved performance over random forest method.

After discussing the results, we present a possible real world application of our method and overall conclusion in the subsequent chapter.

4.3.1 Possible Real World Application of our Method. With each passing year and the rise in technology, we contribute to making massive data in our daily activities. A very interesting area is social media in particular twitter where people from all walks of life interacts together on daily basis. Data produced from this platform is unlabelled text documents and the world can take advantage of these documents (tweets) to extract useful information. For example, let us consider having collected 10,000 tweets. Manual classification of these text documents requires that human label each document with particular label. Having to read all of these tweets is tedious, expensive and time consuming. A better way is to segment these enormous tweets into clusters and later find the class of the groupings.

Before one can cluster text documents, Natural language Processing techniques can be used to clean and processed the data into arrays. After which, we propose using the HDBSCAN algorithm to extract the hierarchical tree for further analysis. The concept of using only the centroid may not be very sufficient given the data under consideration. So we propose that each time there is a split of cluster, we select the central tweet and its k-th nearest (k is a parameter to be given) neighbors. These selected tweets consist of words from which a 'user' will infer a label. The cluster and label procedure continues until there is no more split of clusters. For instance, let us assume that cluster A is one of the cluster found after clustering the 10,000 tweets. For illustration purpose, we define the centroid of the cluster A to be t^* and set k$=$ 3. The table below shows the four tweets documents that will be considered.

Cluster A			
t^*	t_1	t_2	t_3
doctor	food	patient	disease
cancer	vitamin	medical	chronic
medicine	pms	pain	treatment
disease	weight	history	kidney

Table 4.2: *An assumed cluster from 10,000 tweets and t_i refers to the i-th tweet document.*

From Table 4.2, all the four tweets documents in cluster A points out the idea of health and so the user can classify the whole cluster to be containing health tweets documents. Similar deduction can be made for the other clusters found in the tree. In this way, one could reduced the human cost involved in analysing 10,000 tweets and get the documents categorized at a faster rate.

5. Conclusion

The aim of this project was to use semi-supervised learning system to address the challenges related to identification and classification of large datasets. Two machine learning algorithms: random forest (the baseline model) and HDBSCAN were considered.

The random forest classifier was trained on labelled instances and its performance on test set was evaluated. The HDBSCAN classifier was then applied to cluster (and later classify) unlabelled instances. The performance of the HDBSCAN was also evaluated. We found that both random forest and HDBSCAN classifiers did well in making predictions but the latter showed an improved performance with few training points.

From this study, we observed that using clustering and active learning techniques produces a reliable classification model at minimum cost. In addition, the involvement of human factor in an automated system is much more effective to build training sets very quickly.

In the future, we would consider pertinent scenario such as running out of clusters even before we obtain our small labelled dataset and think of approaches to address such cases.

Acknowledgements

Firstly, I would like to thank God for all the strength, wisdom and knowledge he has bestowed on me to enable me to complete this research work successfully. I can do all things through Christ who strengthens me. My greatest and sincerest gratitude to Dr Michelle Lochner, my academic advisor for all the constructive criticisms through the course of this research. This would not have been possible without your guidance and trust in me. I am also immensely grateful to Felicien Jordan Masakuna, Dr Kenneth Dadedzi and all my academic tutors for the insight and comments throughout this journey.

Another heartfelt gratitude to the Aikins family for all their encouragement, sacrifices and support. They have always been there for me through thick and thin. To all my colleagues and friends, it has been an awesome journey here at AIMS and am glad to have completed this journey with you all.

References

Alpaydin, E. *Introduction to Machine Learning*. MIT press, 2009.

Breiman, L. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

Breiman, L. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Brownlee, J. Machine Learning Mastery, 2019. URL https://machinelearningmastery.com/start-here/.

Campello, R. J., Moulavi, D., and Sander, J. Density-Based Clustering Based on Hierarchical Density Estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Campello, R. J. G. B., Moulavi, D., Zimek, A., and Sander, J. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data*, 10(1):5:1–5:51, July 2015. ISSN 1556-4681. doi: $10.1145/2733381$. URL http://doi.acm.org/10.1145/2733381.

Chong, E. K. and Zak, S. H. *An Introduction to Optimization*, volume 76. John Wiley & Sons, 2013.

Claesen, M. and De Moor, B. Hyperparameter Search In Machine Learning. *arXiv preprint arXiv:1502.02127*, 2015.

Collaboration, L. S. et al. LSST Science Book, version 2.0. *ArXiv e-prints*, 2009.

Domingos, P. M. A Few Useful Things to Know about Machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

Everitt, B. S., Landau, S., Leese, M., and Stahl, D. Miscellaneous Clustering Methods. *Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd, Chichester, UK*, 2011.

Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

Freund, Y., Schapire, R. E., et al. Experiments with a New Boosting Algorithm. In *Thirteenth International Conference on ML*, volume 96, pages 148–156. Citeseer, 1996.

Friedman, J., Hastie, T., and Tibshirani, R. *The Elements of Statistical Learning*, volume 1. Springer series in statistics New York, 2001.

Han, J., Kamber, M., and Pei, J. Data Mining Concepts and Techniques third edition. *Morgan Kaufmann*, 2011.

James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning with Applications in R*. Springer, 2013.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning research*, 9 (Nov):2579–2605, 2008.

McInnes, L., Healy, J., and Astels, S. hdbscan: Hierarchical Density Based Clustering. *The Journal of Open Source Software*, 2(11), mar 2017. doi: $10.21105/\text{joss}.00205$. URL https://doi.org/10.21105%2Fjoss.00205.

Ramadevi, G. N., Rani, K. U., and Lavanya, D. Evaluation of Classifiers Performance using Resampling on Breast cancer Data. *International Journal of Scientific & Engineering Research*, 6(2), 2015.

Raschka, S. *Python Machine Learning*. Packt Publishing Ltd, 2015.

Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited, 2016.

Settles, B. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

Strobl, C., Malley, J., and Tutz, G. An Introduction to Recursive Partitioning: Rationale, Application, and Characteristics of Classification and Regression Trees, Bagging, and Random Forests. *Psychological methods*, 14(4):323, 2009.

Zhong, C. *Improvements on Graph-based Clustering Methods*. Citeseer, 2013.