Tribhuvan University

Institute of Science and Technology

Comparative Analysis on Ensemble Learning: Bagging and Boosting

Dissertation

Submitted to

Central Department of Computer Science and Information Technology

Kirtipur, Kathmandu, Nepal

In Partial Fulfillment of the Requirements

For the Master's Degree in Computer Science and Information Technology

By

Bhoj Raj Adhikari

T.U Registration No: 5-2-33-20-2011

T.U. Examination Roll No: 305/073

January, 2020

Supervisor

Mr. Bikash Balami

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and no sources other than listed here have been used in this work.

_____

Bhoj Raj Adhikari

Date: January, 2020

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that this dissertation is prepared under my supervision by **Mr. Bhoj Raj Adhikari** entitled **"Comparative Analysis on Ensemble Learning: Bagging and Boosting"** be accepted as in fulfilling partial requirement for the completion of Master's Degree of Science in Computer Science and Information Technology.

_____

Asst. Prof. Bikash Balami

Central Department of Computer Science and Information Technology

Tribhuvan University, Kathmandu Nepal

(Supervisor)

Date: January, 2020

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that, we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

Asst. Prof. Bikash Balami	**Asst. Prof. Nawaraj Paudel**
Central Department of Computer Science &	**Head of Department**
Information Technology	Central Department of Computer Science &
Tribhuvan University	Information Technology
(Supervisor)	Tribhuvan University

External Examiner	Internal Examiner

Acknowledgement

Firstly, I would like to express my sincere gratitude to my respected teacher as well as my research supervisor, Mr. Bikash Balami, Assistant Professor, Central Department of Computer Science & Information Technology (CDCSIT), Tribhuvan University for boosting my morale with his valuable suggestions and strong guidelines throughout this research work in tackling with various difficulties. Without his assistance and dedicated involvement in every step throughout the process, this research work would have never been accomplished.

I would also like to express warm thanks to respected Head of Department Asst. Prof. Nawaraj Paudel for providing me favorable environment in conducting the research. I am sincerely grateful to entire group of Professors, Lecturers of the Department for their valued inspiration and encouragement.

I would like to express my sincere thanks to my colleagues Mr. Bikash Regmi, Kshitiz Bhatt, Sushil Banstola and all my well-wishers who directly and indirectly helped me to complete this work.

Last but not least I must express my very profound gratitude to Sister Boma Adhikari and my parents for providing me with unfailing support and continuous encouragement through the process of writing this thesis.

Bhoj Raj Adhikari

CDCSIT, TU

Abstract

Combine the prediction from multiple models to improve the overall performance of model is an ultimate task of Ensemble learning. Bagging and Boosting are two widely used ensemble learning techniques works based on numbers of classifiers combination to aggregate prediction. Performance of single classifier has limitation due to noise, bias and variance in dataset. By applying divide and conquer approach on ensemble methods helps to minimize those limitation which ultimately leads to performance improvement. Bagging is a bootstrap aggregation while boosting attempts to fit a sequence of weak learner's models to build a strong classifier. The performance of bagging and boosting has been analyzed on the basis of Accuracy, Precision, Recall and F1-Measures for Adult dataset with and without noise. The Gaussian noise distribution has used for noise addition on dataset due to CLT. The results show that on the basis of Accuracy, Recall, F1-Measures boosting outperforms bagging whereas in terms of Precision, bagging has better result.

Keywords: Ensemble Learning, Bagging, Boosting, Gaussian Noise

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CLT	Central Limit Theorem
FN	False Negative
FP	False Positive
IDE	Integrated Development Environment
OCR	Optical Character Recognition
SL	Strong Learner
SMT	Statistical Machine Translation
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
WL	Weak learner

Chapter 1:

Introduction

An ensemble is the art of combining a diverse set of individual learners/models together to improvise the stability and predictive power of the model. Ensemble learning techniques attempt to make the performance of the predictive models better by improving their accuracy. Ensemble basically trains a large number of models and then combines the predictions to come to a conclusion. The method of combining the classifiers depend upon the choice of models trained. Training a bunch of models and taking their result by using combining schema is a principal approach of Ensemble learning. This approach allows the production of better predictive performance compared to a single model.

It is a procedure where multiple learner modules are applied on a dataset to extract multiple predictions, which are then combined into one composite prediction. The learning process is commonly broken down into two tasks as constructing a set of base learners from the training data and combining some or all of these models to form a unified prediction model. Ensemble methods attempt to improve forecasting bias and reducing variance by providing critical boost to forecasting abilities and decision-making accuracy. Where, Bias is a source of error in a model that causes it to over-generalize and underfit the data whereas variance is sensitivity to noise in the data that causes a model to over-fit. It is useful when there is uncertainty in choosing the best prediction model and when it is critical to avoid large prediction errors [16].

Multiple classifier systems, also called ensemble learning have proven themselves to be very effective and extremely versatile in a broad spectrum of problem domains and real-world applications. Originally developed to reduce the variance thereby improving the accuracy of an automated decision-making system, ensemble systems have since been successfully used to address a variety of machine learning problems [13]. Combining outputs from multiple classifiers, known as ensemble learning, is one of the standard and most important techniques for improving classification accuracy in machine learning [3].

1.1 Bagging Classifier

Bagging refers to bootstrap aggregation i.e. repeating the sample with replacement and perform aggregation of results to be precise, which is a general-purpose methodology to reduce the variance of models. It is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions either by voting or by averaging to form a final prediction.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N data from the original training dataset, where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Bagging reduces overfitting (variance) by averaging or voting, however this leads to an increase in bias which is compensated by the reduction in variance though [8].

1.2 Boosting Classifier

Boosting is an ensemble modeling technique which attempts to build a strong classifier from the number of weak classifiers. The main principle of boosting is to fit a sequence of weak learner's models that are only slightly better than random guessing i.e. it is able to convert weak learners to strong learners. First, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training dataset is predicted correctly or the maximum numbers of models are added. Boosting being a sequential process, each subsequent model attempts to correct the errors of the previous model.

It is a machine learning method based on the idea that a combination of weak learner can perform better than any of the simple classifiers alone. A weak learner (WL) is a learning algorithm capable of producing classifiers with probability of error strictly less than that of random guessing (0.5, in the binary case) where strong learner (SL) is able to yield classifiers with arbitrarily small error probability [6]. The algorithm process accordingly:

Step 1: Base Learning combines each distribution and applies equal weight to them.

Step 2: If any prediction occurs during the first base learning algorithm then pay high attention to that prediction error.

Step 3: Repeat step 2 until the limit of Base Learning algorithm has been reached or high accuracy.

Step 4: Finally, it combines the entire weak learner to create one strong prediction true.

1.3 Problem Definition

Due to instability of single classifier it does not preforms well for all dataset. Generally, Single classifier performs better only when having large number of data. For neural learning technique, such models give good accuracy only with good parameters like number of hidden layers, activation function, and number of nodes in each layer etc. The advantage of using Ensemble model over single classifier is to improve the accuracy of algorithm as well as it is able to provide high stability for all type of dataset. The choice of Bagging and Boosting classifier over the rest is that it works based on combining schema.

1.4 Objectives of Thesis

The main objective of thesis is

- To analyze the performance and to determine the accuracy level of Bagging and Boosting algorithm.
- To determine the accuracy level of both the algorithms with and without noised dataset based on different number of base classifiers.

1.5 Limitation of Thesis

Although Ensemble learning classifier are Bagging, Boosting and stacking; the comparative analysis has been performed on only Bagging and Boosting classifier. It is due to the fact that these classifier techniques perform based on homogeneous base classifier. Whereas stacking classifier performs as heterogeneous base classifier. I.e. it works on different base classifier so the exact classifier numbers cannot be predicted. So, comparative analysis of all classifier altogether is not suitable.

1.6 Thesis Organization

The organization of this thesis is as follows:

Chapter 1 describes the introduction, problem statement and objectives.

Chapter 2 describes the literature review of the existing researchers related to Ensemble Learning: Bagging and Boosting.

Chapter 3 describes the algorithm and methodology of the Bagging and Boosting algorithms

Chapter 4 contains the implementation overview of the algorithms with result analysis.

Chapter 5 concludes the conclusion of thesis works.

Chapter 2:

Background Study and Literature Review

2.1 Background Study

In 1990, the first boosting procedure was proposed by Schapire [15], where the key result is that weak and strong learnability are equivalent, in the sense that strong learning can be performed by combining WLs. Bagging is application of bootstrap procedure for having high-variance machine learning algorithm.it is a predictors for generating an aggregated predictor.

2.2 Literature Review

In [2], testing on real and simulated datasets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy. The vital element in the instability of the prediction method is basically using single classifier. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

In [12], the bagging and boosting with neural network and decision trees algorithm as base classifier was evaluated. The result shows that Bagging is almost always more accurate than a single classifier, it is sometimes much less accurate than Boosting. Meanwhile, Boosting can create ensembles that are less accurate than a single classifier – especially when using neural networks.

In [5], the test and training error curves in an optical character recognition (OCR) problem as both a function of training set size and computational cost using neural-based ensemble technique was implemented.

In [1], the purpose of the study is to improve the understanding of why and when these algorithms, which use perturbation, reweighting, and combination techniques, affect classification error. By providing a bias and variance decomposition of the error shows how different methods and variants influence these two terms. The result shows that Bagging reduced variance of unstable methods, while boosting methods reduced both the bias and variance of unstable methods but increased the variance for Naive-Bayes, which was very stable. The voting methods lead large and significant reductions in the mean-squared errors.

In [14], the revision of the classifier selection methodology and evaluates the practical applicability of diversity measures in the context of combining classifiers by majority voting. A number of search algorithms are proposed and adjusted to work properly with a number of selection criteria including majority voting error and various diversity measures. The algorithms used a binary vector of classifier incidences, indicating exclusion (0) or inclusion (1) of the classifier in the combination, as a representation of the selection solution. Furthermore, a diversifying operator was applied to the populations of solutions, which prevented duplication of the same combinations found as a result of the search algorithms. The majority voting has then been applied to the best combinations returned by the algorithms and provided the basis for the assessment of different diversity measures used as selection criteria. The result shows that the better the correlation between the measure (selection criterion) and the combiner performance, the higher the performance of the selected combinations with optimal results.

In [10], to improve the limited classification performance of the real SVM, SVM ensemble with bagging or boosting has been proposed. In bagging, each individual SVM is trained independently using the randomly chosen training samples via a bootstrap technique. In boosting, each individual SVM is trained using the training samples chosen according to the sample's probability distribution that is updated in proportional to the errorless of the sample. Various simulation results for the IRIS data classification and the hand-written digit recognition, and the fraud detection shows that the proposed SVM ensemble with bagging or boosting outperforms a single SVM in terms of classification accuracy greatly.

In [17], this paper has examined techniques for combining simple ensemble learning approaches with the aim of exploring the relationship between ensemble member diversity and ensemble error. The results strongly support the proposition that combining effective ensemble learning strategies is conducive to reducing test error.

In [11], the evaluation of both neural networks and decision tree classification algorithms has been studied. The result shows that even though Bagging almost always produces a better classifier than any of its individual component classifiers and is relatively impervious to overfitting, it does not generalize any better than a baseline neural-network ensemble method. Also result shows that Boosting is a powerful

technique that can usually produce better ensembles than Bagging although it is more susceptible to noise and can quickly overfit a dataset.

In [4], the effectiveness of randomization, bagging, and boosting for improving the performance of the decision-tree algorithm was proposed. The experiment shows that in situations with little or no classification noise, randomization is competitive with bagging but not as accurate as boosting. In situations with substantial classification noise, bagging is much better than boosting, and sometimes better than randomization.

In [7], the demonstration of AdaBoost in many settings to improve the performance of a learning algorithm was implemented. When starting with relatively simple classifiers, the improvement can often lead to a composite classifier that outperforms more complex “one-shot” learning algorithm. For non-binary classification problems, boosting simple classifiers can only be done effectively if the more sophisticated pseudo-loss is used. Boosting combined with a complex algorithm may give the greatest improvement in performance when there is a reasonably large amount of data available.

In [18], to address the issue of building a strong translation system using a group of weak translation systems generated from a single SMT (statistical machine translation) engine using a Bagging/Boosting-based approach was proposed. The experimental result shows that this approach is very useful in improving the translation accuracy of three state-of-the-art SMT systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based system.

Chapter 3:

Methodology

3.1 Bagging Classifier

Bagging is one of the Ensemble construction techniques which is also known as Bootstrap Aggregation. Bootstrap is a sampling technique in which selects the "n" observations out of a population of "n" observations with replacement. It is the foundation of bagging technique. The selection process of training tuples among dataset is random. i.e., each observation can be chosen from the original population so that each observation is equally likely to be selected in each iteration of the bootstrapping process.

After the bootstrapped samples are formed, separate models are trained with the bootstrapped samples. In real experiments, the bootstrapped samples are drawn from the training set, and the sub-models are tested using the testing set. The final output prediction is combined across the projections of all the sub-models [8].

Algorithm 1: Bagging

Inputs:

Training data $S = \{x_i, y_i\}, i = 1, 2, \dots, N, y_i \in \{\omega_1, \omega_2, \dots, \omega_C\}$;

Supervised learning algorithm (Base Classifier);

Integer T specifying ensemble size;

Percent R to create bootstrapped training data.

Do *t = 1, ..., T*

1. Take a bootstrapped replica S_t *by randomly drawing R% of S.*

2. Call Base Classifier with S_t *and receive the hypothesis (classifier)* h_t.

3. Add h_t *to the ensemble,* $\varepsilon \leftarrow \varepsilon \cup h_t$.

End

Ensemble Combination: *Simple Majority Voting - Given unlabeled instance x*

1. Evaluate the ensemble $\mathcal{E} = \{h_1, h_2, \dots, h_T\}$ *on x.*

2. Let $v_{t,c} = 1$ if h_t chooses class ω_c , and 0 otherwise.

3. Obtain total vote received by each class:

$$V_c = \sum_{t=1}^{T} v_{t,c} \text{ , } c = 1, 2, \ldots, C$$

Output: *Class with the highest V_c*

Process Flow Diagram:

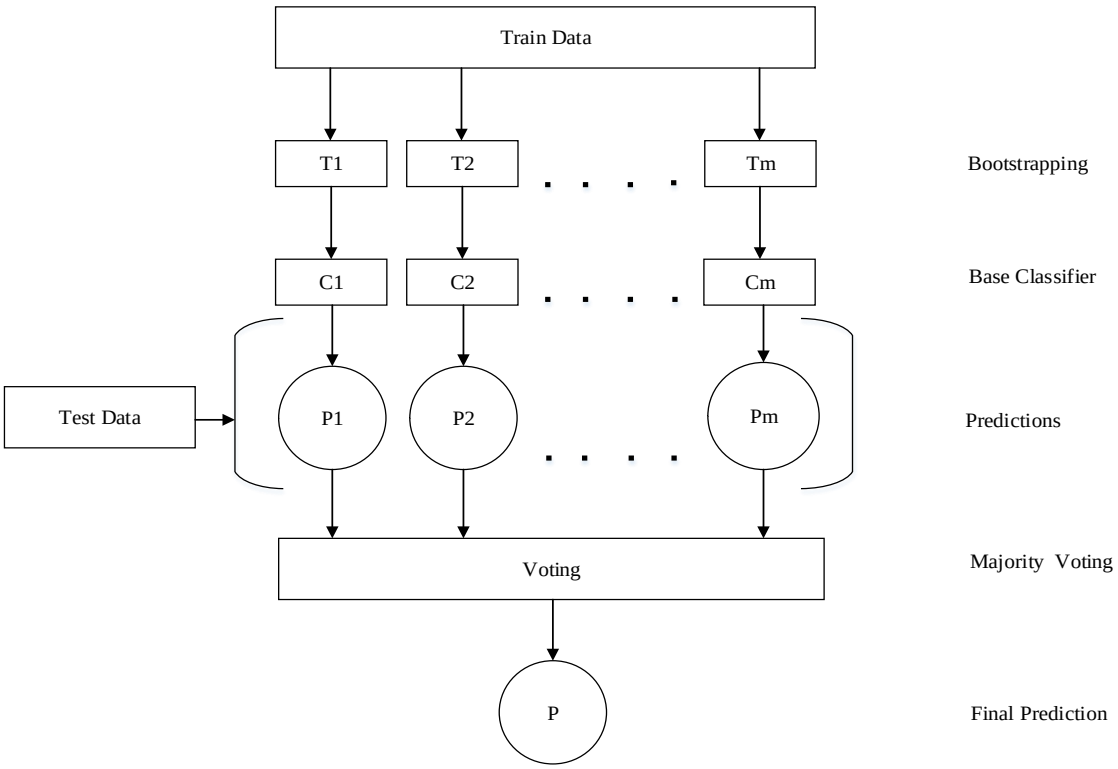

Figure 1: Bagging Processing

3.2 Boosting Classifier

Boosting is a form of sequential learning technique. The algorithm works by training a model with the entire training set, and subsequent models are constructed by fitting the residual error values of the initial model. It attempts to give higher weight to those observations that were poorly estimated by the previous model. Once the sequence of the models is created the predictions made by models are weighted by their accuracy scores and the results are combined to create a final estimation [6].

Algorithm 2: Boosting

Inputs:

Training data S = $\{x_i, y_i\}, i = 1, 2, \ldots, N, y_i \in \{\omega_1, \omega_2, \ldots, \omega_C\}$;

Supervised learning algorithm (Base Classifier);

Integer T specifying ensemble size.

Initialize:

$D_1(i) = 1/N.$

Do *t = 1, ..., T*

1. Draw training subset S_t *from the distribution* D_t.

2. Train Base Classifier on S_t *, receive hypothesis* $h_t: X \longrightarrow Y$

3. Calculate the error of h_t *:*

$$\epsilon_t = \sum_{i=1}^{N} D_t(i) \; [\, h_t(x_i) \neq y_i]$$

4. Calculate weight of h_t*:*

$$\alpha_t = ln \frac{1-\epsilon_t}{\epsilon_t}$$

5. Update sampling distribution:

$$D_{t+1}(i) = \frac{D_t(i).exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$$

Z_t *is a normalization factor, such that* $\sum_{i=1}^{N} D_{t+1}(i) = 1$

End

Weighted Majority Voting: *Given unlabeled instance z, obtain total vote received by each class*

$$V_c = \sum_{t=1, h_t(z)=\omega_C}^{T} \alpha_t \; , c = 1, 2, \ldots, C$$

Output:

Class with the highest V_c.

Process Flow Diagram:

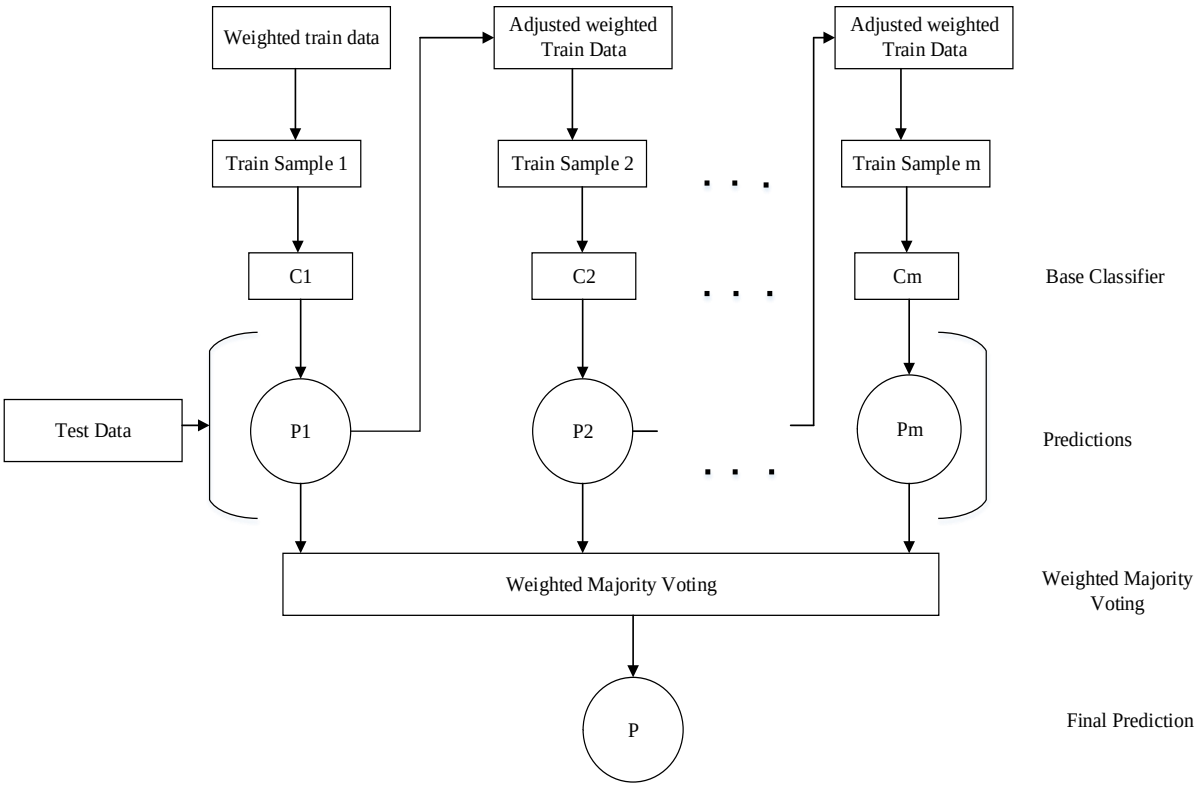

Figure 2: Boosting Processing

3.3 Implementation Method

Three strategies need to be chosen to building an effective ensemble system. The three pillars of ensemble systems are:

- Data sampling/selection;
- Training member classifiers; and
- Combining classifiers.

3.3.1 Bagging and Boosting Classifier Implementation

Decision Tree Algorithm as Base Classifier used for Bagging and Boosting classifier.

i. Data preprocessing
ii. Apply Bagging classifier described in sec.5.1 using Ten-fold cross-validation. Or
iii. Apply Boosting Classifier described in sec.5.2 using Ten-fold cross-validation.

3.4 Data Collection

For this study a labeled dataset as Adult dataset[1] (see sampled dataset in Appendix A): standard datasets for machine learning available at UCI machine learning repository is taken. And dataset Extraction was done by Barry Becker from the 1994 Census database. The dataset is designed to determine whether a person makes over 50K a year by using a class label as >50K & <=50K. It contains 14 independent and one dependent variable as Age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country and income respectively with 48842 instances.

3.5 Data Preprocessing

Before data feeding to model transformation should be applied on dataset according to algorithm and needs. Whenever the data is gathered from different sources it is collected in raw format which might not be feasible for the analysis. Data Preprocessing is a technique that is used to convert the raw data into a clean dataset. Dataset should be transformed in such a way that more than one Machine Learning algorithms are executed in a dataset and best among them is chosen.

The quality of data and the useful information that can be derived from data preprocessing has direct impact over the ability of model to learn. So before feeding the dataset it has to be preprocessed accordingly as handling Null values, Standardization, Handling categorical variable, Noise addition.

3.6 Noise Addition:

To implement both the algorithm in a noise is added in the dataset by using the Gaussian noise distribution method. It is a statistical noise having probability density function equal to that of the normal distribution, also known as the Gaussian distribution [19].

$$f(x;\ \mu,\ \sigma^2)\ =\ \frac{1}{\sigma\sqrt{2\pi}}\,e^{-\frac{1}{2}\left((x-\mu)/\,\sigma\right)^2}$$

[1] https://archive.ics.uci.edu/ml/datasets/Adult

Where μ is the mean or expectation, σ^2 is the variance.

The distribution with μ=0 and σ^2=1 is called the standard normal distribution or the unit normal distribution. It is widely used distribution function because of the CLT (Central Limit Theorem).

3.7 Performance Evaluation Measures

The analysis of both the algorithms has been measured with various parameters as Accuracy, Precision, F1-measure and Recall. This ultimately helps to choose the better one among both algorithms. All the parameter values are computed based on Confusion matrix as true positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). TP and TN are the correctly predicted positive values and correctly predicted negative values respectively. FP and FN are the total number of data when actual class is No and predicted class is Yes and when actual class is Yes and predicted class is No respectively.

Accuracy:

It is a ratio of correctly predicted observation to the total observations. It is the most intuitive quality performance measures for the effectiveness of the machine learning model [9]. Higher the accuracy betters the model.

Mathematically it is calculated as:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision:

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It highlights the correct positive predictions out of all the positive predictions. High precision indicates low false positive rate. Higher the precision better the results and measured [9] as:

$$precission = \frac{TP}{TP + FP}$$

Recall (Sensitivity):

Recall is the ratio of correctly predicted positive observations to the all observations in actual class. Recall highlights the sensitivity of the algorithm. It is the ability of the classifier to find all the positive samples.

Mathematically [9] calculated as:

$$recall = \frac{TP}{TP + FN}$$

F1-Measure:

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It tells how precise the classifier is i.e. how many instances it classifies correctly, and how robust it is i.e. it does not miss a significant number of instances. With high precision but low recall, the classifier is extremely accurate, but it misses a significant number of instances that are difficult to classify [9]. It is calculated as:

$$F1 - measure = 2 \times \frac{precision \times recall}{precision + recall}$$

Chapter 4:

Implementation and Analysis

4.1 Implementation Tools

4.1.1 Python Programming Language

Python is a multi-paradigm, easy to learn and powerful programming language. Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991. It lets system more quickly, integrated and effective. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available. The Python interpreter is easily extended with new functions and data types implemented in C, C++ or other languages callable from C. Python is also suitable as an extension language for customizable applications. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

4.1.2 Anaconda

Anaconda is a free, multiplatform and open-source distribution of the Python and R programming languages for scientific computing as data science, machine learning applications, large-scale data processing, predictive analytics, etc., that aims to simplify package management and deployment. Package versions are managed by the package management system Conda. It analyses the current environment including everything currently installed and together with any version limitations specified.

With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling individual data scientists. It has more than 1,500 Python/R data science packages. It Manage libraries, dependencies, environments with Conda and also develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano.

4.1.3 Spyder IDE

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. it is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software. Beyond its many built-in features, Spyder's abilities can be extended even further via its plugin system and API. It can also be used as a PyQt5 extension library, allowing you to build upon its functionality and embed its components.

4.1.4 Sicikit-learn

Scikit-learn are a Python module integrating classic machine learning algorithms in the tightly-knit scientific Python world (numpy, spicy, matplotlib). It aims to provide simple and efficient solutions to learning problems accessible to everybody and reusable in various contexts.

4.1.5 Category Encoders

It encodes categorical variables into numeric with different techniques. While ordinal, one-hot and hashing encoders have similar equivalents in the existing scikit-learn version. It is featured with dataframes, column configuration regardless of input type, compatible with sklearn.

4.1.6 Matplotlib

Matplotlib is a Python 2D plotting library. It can be used in Python scripts, IPython shells, Jupyter notebook, Spyder and web application server's etc. it is used for plots generation, histograms, power spectra, bar charts, scatterplots, etc.

4.2 Testing Environment

The following hardware and software configuration was used to implement this thesis.

Hardware Specification:

- System : intel(R) Core(TM) i5-3210M CPU @2.50GHz
- Hard Disk : 500 GB.
- RAM : 8 GB.

Software Specification:

- Operating system : Windows 10 Pro
- Coding Language : Python
- Tools : Spyder 3.7

Table 1: Performance Measures of Bagging and Boosting based on no. of Classifier without Noise

No. of Base Classifier	Accuracy		F1-Measure		Precision		Recall	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
1	80.934	76.149	87.411	86.453	87.841	76.149	86.990	100.000
2	78.759	81.923	85.133	88.840	91.059	83.782	79.942	94.565
3	82.912	84.091	88.861	90.020	88.150	86.123	89.591	94.298
4	81.818	84.257	87.758	90.176	89.995	85.860	85.639	94.960
5	83.262	84.318	89.114	90.201	88.210	85.998	90.051	94.847
6	82.672	84.760	88.473	90.418	89.555	86.690	87.424	94.493
7	83.600	84.828	89.387	90.478	88.051	86.591	90.769	94.741
8	83.194	85.012	88.920	90.632	89.216	86.429	88.630	95.275
9	83.956	84.982	89.627	90.572	88.198	86.751	91.118	94.777
10	83.753	84.840	89.329	90.393	89.308	87.301	89.357	93.743
11	84.281	84.871	89.842	90.366	88.393	87.652	91.356	93.271
12	83.814	84.883	89.417	90.384	88.993	87.574	89.851	93.399
13	84.355	85.098	89.894	90.539	88.381	87.577	91.469	93.724
14	83.937	85.111	89.514	90.550	88.958	87.574	90.091	93.750
15	84.502	85.215	89.989	90.620	88.502	87.615	91.533	93.846
16	83.888	85.258	89.502	90.645	88.742	87.662	90.281	93.849
17	84.490	85.276	90.011	90.669	88.272	87.575	91.831	94.001
18	84.453	85.258	89.886	90.654	88.992	87.585	90.810	93.960
19	84.607	85.301	90.097	90.696	88.245	87.501	92.035	94.144
20	84.380	85.221	89.859	90.638	88.809	87.501	90.940	94.024
21	84.711	85.178	90.145	90.604	88.464	87.535	91.898	93.912
22	84.435	85.172	89.910	90.599	88.720	87.535	91.140	93.904
23	84.785	85.129	90.203	90.565	88.412	87.559	92.082	93.799
24	84.810	85.160	90.151	90.579	88.994	87.619	91.344	93.759
25	84.576	85.233	90.070	90.614	88.277	87.763	91.947	93.670
26	84.459	85.258	89.931	90.612	88.684	87.912	91.226	93.499
27	84.840	85.301	90.259	90.645	88.323	87.896	92.298	93.589
28	84.521	85.412	89.985	90.729	88.621	87.856	91.399	93.807
29	84.484	85.418	90.012	90.722	88.236	87.955	91.871	93.678
30	84.453	85.461	89.944	90.744	88.565	88.024	91.375	93.646

Table 2: Performance Measures of Bagging and Boosting based on no. of Classifier with Noise

No. of Base Classifier	Accuracy		F1-Measure		Precision		Recall	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
1	77.008	76.021	84.851	86.376	84.995	76.021	84.709	100.000
2	73.906	78.747	81.357	86.132	89.021	85.451	74.910	86.861
3	79.069	79.177	86.366	86.159	85.542	87.072	87.210	85.291
4	78.160	81.331	85.146	88.582	88.129	82.795	82.363	95.266
5	80.371	80.666	87.280	87.641	86.013	85.227	88.588	90.457
6	79.649	81.089	86.391	88.059	87.858	84.617	84.977	92.054
7	80.832	81.327	87.637	88.304	85.964	84.268	89.381	92.907
8	80.398	81.150	87.037	87.949	87.505	85.529	86.577	90.695
9	81.607	81.323	88.161	88.021	86.316	85.884	90.094	90.347
10	80.962	81.384	87.499	87.942	87.369	86.640	87.633	89.328
11	81.561	81.611	88.147	88.155	86.195	86.357	90.195	90.073
12	81.503	82.191	87.923	88.793	87.280	85.117	88.583	92.815
13	81.803	82.594	88.323	88.970	86.228	85.854	90.529	92.336
14	81.714	82.605	88.087	88.927	87.259	86.162	88.938	91.881
15	82.191	82.590	88.588	88.916	86.366	86.163	90.934	91.857
16	81.876	82.594	88.220	88.915	87.184	86.187	89.285	91.826
17	82.237	82.578	88.606	88.897	86.460	86.225	90.868	91.745
18	81.738	82.628	88.149	88.941	86.987	86.181	89.347	91.888
19	82.156	82.670	88.566	88.969	86.342	86.207	90.915	91.922
20	82.144	82.832	88.422	89.036	87.186	86.535	89.701	91.690
21	82.164	82.836	88.577	89.059	86.316	86.411	90.970	91.887
22	82.079	82.797	88.402	89.040	87.010	86.343	89.842	91.918
23	82.313	82.747	88.677	89.008	86.375	86.308	91.108	91.887
24	82.229	82.893	88.510	89.100	87.026	86.409	90.053	91.973
25	82.436	83.004	88.753	89.168	86.466	86.494	91.171	92.018
26	82.233	83.031	88.543	89.165	86.843	86.646	90.317	91.840
27	82.594	83.123	88.870	89.224	86.465	86.700	91.419	91.907
28	82.432	83.127	88.663	89.219	87.026	86.750	90.369	91.841
29	82.432	83.127	88.751	89.213	86.472	86.792	91.161	91.781
30	82.275	83.108	88.600	89.216	86.682	86.675	90.609	91.917

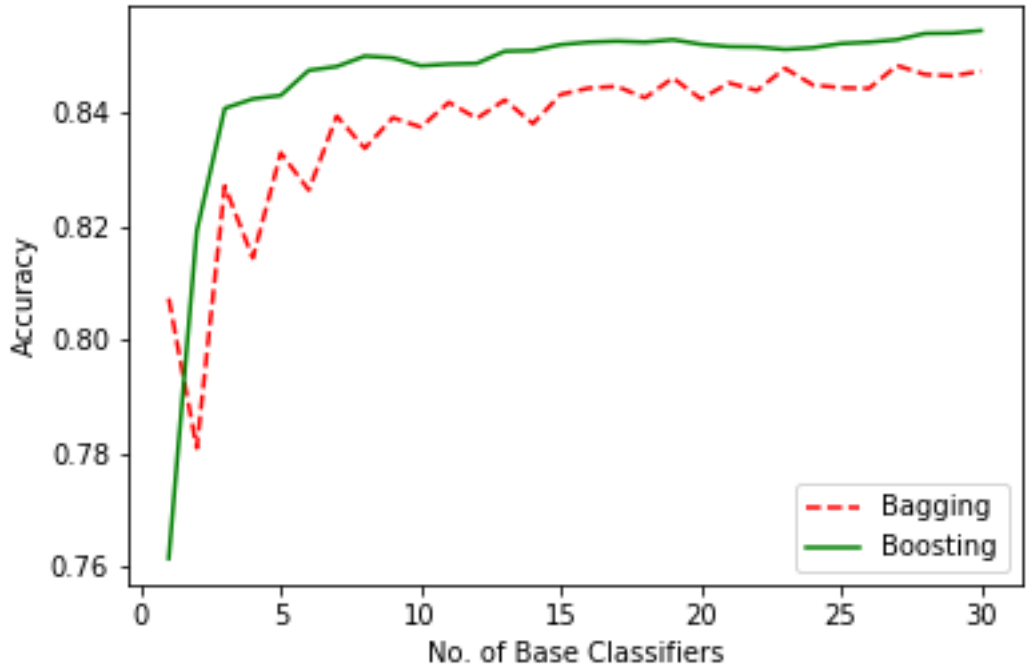

Figure 3: Accuracy Curve of Bagging and Boosting based on no. of base classifier without Noise

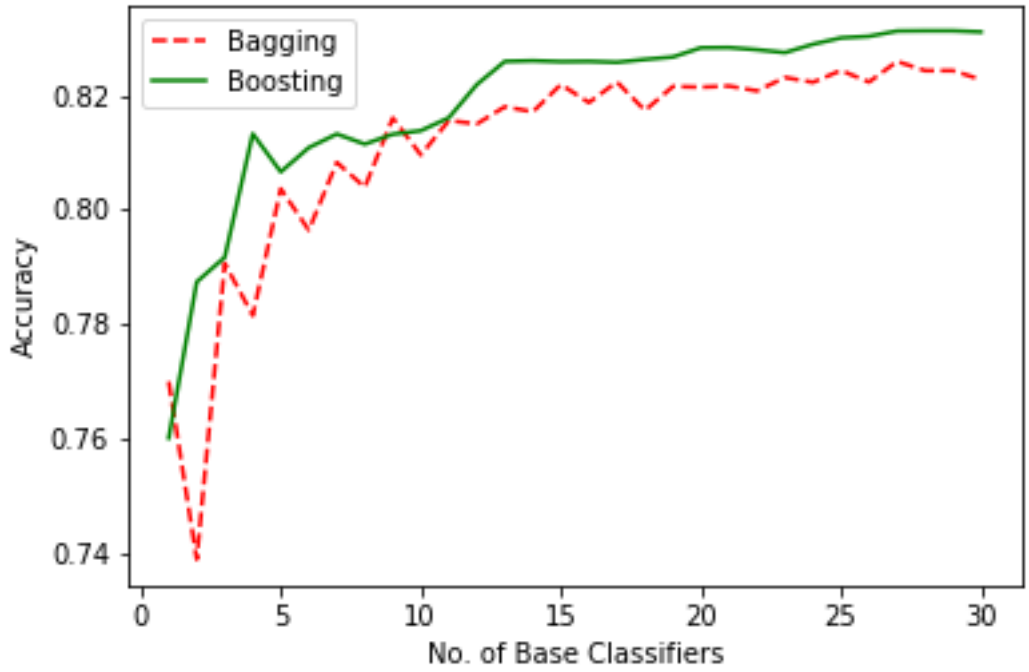

Figure 4: Accuracy Curve of Bagging and Boosting based on no. of base classifier with Noise

Table 3: Performance Measures of Bagging and Boosting based on Data Size without Noise

Data Size (%)	Accuracy		F1-Measures		Precision		Recall	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
10	82.035	84.214	88.079	89.938	88.342	86.861	87.859	93.276
20	83.737	85.012	89.401	90.515	89.169	87.780	89.651	93.449
30	84.101	84.920	89.585	90.405	89.452	87.763	89.728	93.235
40	83.684	84.390	89.327	90.107	89.122	87.137	89.543	93.327
50	83.354	84.840	89.087	90.393	88.869	87.301	89.315	93.743
60	83.584	84.889	89.213	90.482	89.091	86.746	89.346	94.582
70	84.174	85.280	89.615	90.710	89.327	87.094	89.917	94.644
80	84.383	85.231	89.742	90.669	89.633	87.230	89.857	94.407
90	84.210	85.244	89.626	90.682	89.545	87.210	89.712	94.449

Table 4: Performance measures of Bagging and Boosting based on Data Size with Noise

Data Size (%)	Accuracy		F1-Measures		Precision		Recall	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
10	79.362	81.174	86.471	87.787	85.789	86.197	87.233	89.527
20	80.098	81.203	87.058	88.248	86.725	84.635	87.429	92.245
30	79.955	80.252	86.914	87.283	86.566	85.745	87.276	88.896
40	79.638	81.304	86.666	88.229	86.543	84.832	86.791	91.947
50	80.178	80.878	87.007	87.734	86.810	85.727	87.209	89.876
60	80.052	80.743	86.881	87.477	86.815	86.456	86.953	88.533
70	81.068	81.603	87.552	88.115	87.406	86.504	87.703	89.802
80	81.308	81.384	87.752	87.942	87.419	86.640	88.094	89.328
90	81.231	81.842	87.671	88.330	87.567	86.369	87.780	90.457

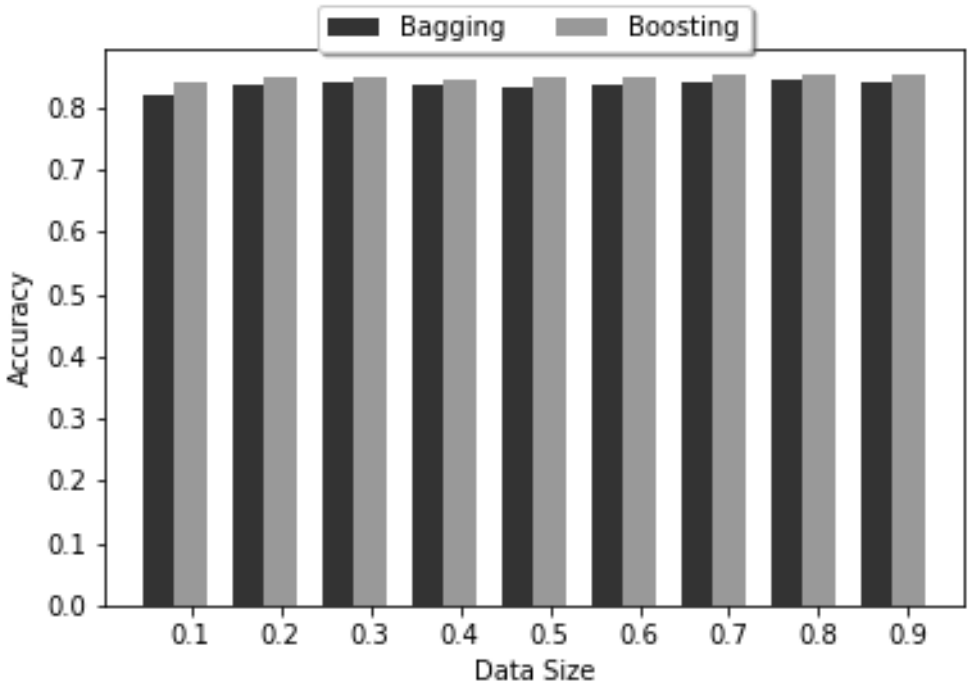

Figure 5: Accuracy bar graph of Bagging and Boosting based on data size without Noise

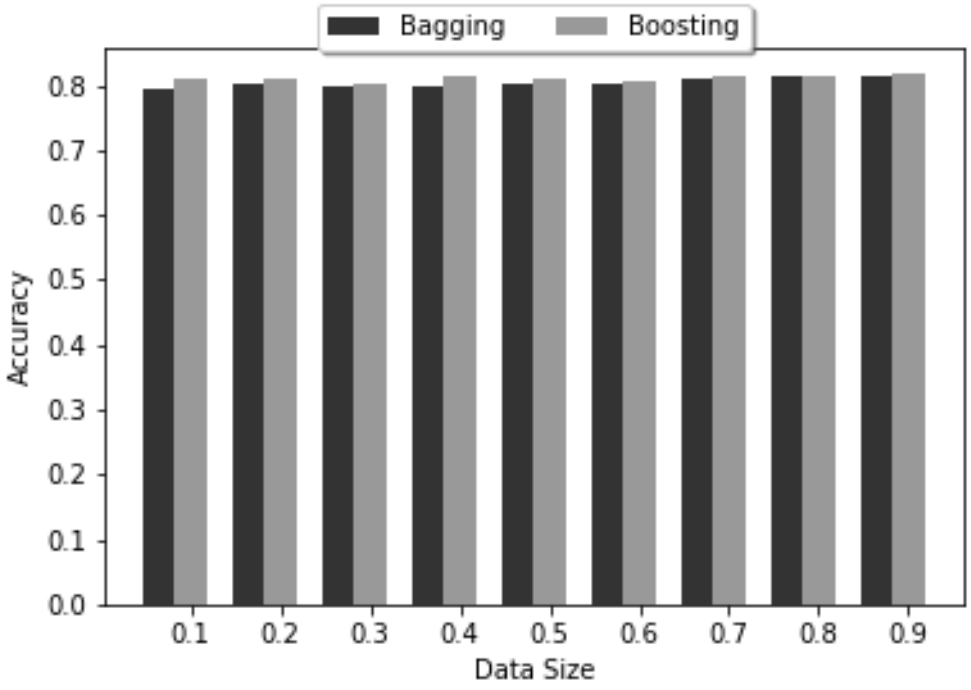

Figure 6: Accuracy bar graph of Bagging and Boosting based on data size with Noise

4.3 Results Analysis

The result of bagging and boosting algorithms has been analyzed by Accuracy, F1-Measure, Precision, and Recall parameters value with Adult dataset.

From table 1 for dataset without noise, accuracy percentage of bagging ranges from *78.7 - 84.8* and boosting ranges from *76.1 - 85.4*. Also the percentage of Precision, Recall value and F1-measure of bagging ranges from *87.8 - 91.0, 79.9 - 92.3* and *85.1 - 90.2* respectively. And for boosting the results are *76.1 - 88.0, 93.2 - 100,* and *86.4 - 90.7* accordingly. Results show that Accuracy, Recall and F1-measure have greater value for boosting. In terms of Precision bagging has higher value.

From Table 2 for dataset with noise, Accuracy, Recall and F1-measure of boosting have higher percentage ranges from *76.0 - 83.1, 85.2 - 100* and *86.1 - 89.2* respectively. Whereas bagging has higher precision value ranges from *84.9 - 89.0* in percentage.

From Table 3, with various data size without noised dataset the boosting has higher value of the Accuracy, Recall and F1-measure in terms of percentages ranges from *84.2 - 85.3, 93.2 - 94.6* and *89.9 - 90.6* respectively. Also the precision has higher value for the bagging ranges from *88.3 - 89.6* percentages.

From Table 4, with noised dataset the percentage values of accuracy, recall and F1-Measure of boosting has higher value ranges from *80.2 - 81.8, 88.5 - 92.2* and *87.2 - 88.3* respectively. Also the Precision has higher value for the bagging ranges from *85.7 - 87.5* percentages.

Chapter 5:

Conclusion and Future Recommendation

5.1 Conclusion

An ensemble learning algorithm that is bagging algorithm and boosting algorithm have been studied and implemented as well. The Adult dataset is chosen for the study. The algorithms are implemented and analyzed with different performance evaluation parameters to determine better algorithm.

From the obtained results, it can be concluded that in case of number of classifiers (>10) boosting algorithm outperforms over bagging algorithm in terms of accuracy, F1-measure, and recall for dataset with and without noise. It is due to the fact that boosting gives high preferences to misclassified data to reduce error iteratively by assigning higher weight for those classified data. Although in terms of precision value bagging is better. On the basis of different data size the obtained results of accuracy, F1-measure, and recall for dataset having noise and without noise shows that boosting performs better than that of bagging. Whereas in case of precision bagging outperforms than boosting.

5.2 Future Recommendation

In this work, decision tree algorithm is used as base classifier in bagging and weak learner in boosting. Different machine learning classification algorithm such as Naïve Bayes Classifier, CART (Classification And Regression Trees) etc. can be used to ensemble such base/weak classifiers. Also Parameter tuning like k-fold cross validation can be used in near future.

REFERENCES

[1] Bauer, E. and Kohavi, R., An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants, *Machine Learning,* vol. 36, pp. 105–139, 1999.

[2] Breiman, L., Bagging Predictors, *Machine Learning*, vol. 24, pp. 123–140, 1996.

[3] Bryll, R., Gutierrez-Osuna, R. and Queka, F., Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets, *Pattern Recognition*, vol. 36, pp. 1291-1302, 2003.

[4] Dietterich, T. G., An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, *Machine Learning*, vol.40, pp. 139–157, 2000.

[5] Drucker, H., Cortes, C., Jackel, L. D. and Vapnik, V., Boosting and Other Ensemble Methods, *Neural Computation*, vol. 6, pp. 1289-1301, 1994.

[6] Ferreira, A. J. and Figueiredo, M. A. T., Boosting Algorithms: A Review of Methods, Theory, and Applications, *Springer*, pp. 35-85, 2012.

[7] Freund, Y. and Schapire, R. E., Experiments with a New Boosting Algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996.

[8] geeksforgeeks.org, ML Bagging Classifier [Online].

Available: https://www.geeksforgeeks.org/ml-bagging-classifier/. [Accessed: 12th December 2019].

[9] Goutte, C. and Gaussier, E., A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation, *Lecture Notes in Computer Science*, vol. 3408, pp. 345-359, April 2005.

[10] Kim, H.-C., Pang, S., Je, H.-M., Kim, D. and Bang, S. Y., Constructing support vector machine ensemble, *Pattern Recognition*, vol. 36, no. 12, pp. 2757–2767, December 2003.

[11] Maclin, R. and Opitz, D., An Empirical Evaluation of Bagging and Boosting, *The Fourteenth National Conference on Artificial Intelligence*, Providence, Rhode Island, 1997.

[12] Opitz, D. and Maclin, R., Popular Ensemble Methods: An Empirical Study, *Journal of Artificial Intelligence Research,* 1999, vol. 11, pp. 169-198.

[13] Polikar, R., Ensemble Learning, *Springer*, pp. 1-34, 2012.

[14] Ruta, D. and Gabrys, B., Classifier selection for majority voting, *Information Fusion,* vol. 6, pp. 63–81, 2005.

[15] Schapire, R. E., The Strength of Weak Learnability, *Machine Learning*, vol. 5, pp. 197-227, 1990.

[16] Steinki, O. and Mohammad, Z., Introduction To Ensemble Learning [pdf]. Available: https://pdfs.semanticscholar.org/96bc/9c947bee74e6adf6c6ae9a2aece93596d350.pdf. [Accessed: 10th December 2019].

[17] Webb, G. I. and Zheng, Z., Multistrategy Ensemble Learning: Reducing Error by Combining Ensemble Learning Techniques, *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 980-991, August 2004.

[18] Xiao, T., Zhu, J. and Liu, T., Bagging and Boosting statistical machine translation systems, *Artificial Intelligence*, vol. 195, pp. 496–527, 2013.

[19] Zahir, A. H. and Mahmood, A. A., Real-time white noise generation using the TMS320C6713 DSP starter kit, *Int. J. Reasoning-based Intelligent Systems*, vol. 4, no. 4, pp. 214-220, 2012.

Appendix A

Sample Dataset

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K

Appendix B

Source Code

```
import numpy as np

import pandas as pd

from sklearn import model_selection

import category_encoders as ce

from sklearn.ensemble import BaggingClassifier

from sklearn import tree

from sklearn.ensemble import AdaBoostClassifier

from sklearn.model_selection import train_test_split, cross_validate


split=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]

seed=7

kfold = model_selection.KFold(n_splits=10, random_state=seed)

for s in split:

  model1 = BaggingClassifier(tree.DecisionTreeClassifier(random_state=1),
n_estimators=10)

    results1 = cross_validate(model1, X_train, y_train, cv=kfold, scoring=['accuracy',
'precision', 'recall', 'f1'])

  model2 = AdaBoostClassifier(random_state=1, n_estimators=10)

    results2 = cross_validate(model2, X_train, y_train, cv=kfold,scoring=['accuracy',
'precision', 'recall', 'f1'])


n_est=range(1, 31)

for bc in n_est:

model3 = BaggingClassifier(tree.DecisionTreeClassifier(random_state=1),
n_estimators=bc)

    results3 = cross_validate(model3, X_train, y_train, cv=kfold, scoring=['accuracy',
'precision', 'recall', 'f1'])
```

```
    model4 = AdaBoostClassifier(random_state=1, n_estimators=bc)

    results4 = cross_validate(model4, X_train, y_train, cv=kfold, scoring=['accuracy',
'precision', 'recall', 'f1'])
```