Chapter 4

Ensemble Learning Methods

4.1 Introduction

Supervised learning algorithms are effective in most application domains, however they have limitations. A single learning model may miss out on some local regions of the feature space, impacting overall performance. Ensemble learning techniques can be helpful in this regard as they bring together a diverse set of learners, ensuring that even if one misses a region of the feature space other members in the ensemble may be able to learn the pattern. In other words, ensemble learning takes into account the opinions of a number of learners to obtain the final decision [149]. It builds on traditional machine learning and ensures overall performance improvement by overcoming the individual drawbacks of learners. Conventionally, only one model is used to learn characteristics in the given data; the end performance of such a model may often be average. The primary notion in ensemble learning is that a group of decision makers is more reliable than a single decision maker. In literature, such grouping of decision makers is known by different names such as Blending [150], Ensemble of Classifiers [151], Committee of experts [152], Perturb and Combine [153], among many more. Based on whether existing knowledge is used or not, ensemble learning algorithms are categorized as supervised and unsupervised. Another category of ensemble learning is meta ensemble learning where an ensemble of ensembles is built to further avoid biases of individual ensembles. The purpose is to empirically investigate the performance of different ensemble methods in conjunction with hyper-parameter tuning of the predictive models. The aim is to study the performance of ensemble learning methods namely Bagging, Boosting, Bagging-Boosting and Stacking using different benchmark datasets. It is based on a data-centric supervised ensemble framework comprising of different engines each with its own functionality.

4.1.1 Motivation

Ensemble learning is known to be a powerful technique which relies on aggregating the decisions given by several learners. The ability and versatile nature of ensemble learning methods specially, bagging, boosting and stacking to enhance model performance has garnered tremendous attention from the research community. The aim is to study the performance of each individual learner and their generalization capabilities in the context of security datasets. The base learners are selected from diverse backgrounds in order to eliminate the biasness of the individual learners. The proposed framework is designed by exploiting the consistently performing learning algorithms to achieve best possible classification performance.

4.1.2 Contribution

In this chapter, an empirical evaluation of several benchmark security datasets using well known ensemble methods such as Bagging, Boosting, Bagging-Boosting, Stacking and a combination learner is carried out. Along with the evaluation, an analysis on how hyper-parameter tuning can play a big role in achieving high accuracy in case of most of the datasets is also done.

4.2 Background

This section presents in depth on what is ensemble learning, how to construct an ensemble from scratch, types of ensemble learning along with some theoretical basics of ensemble learning.

4.2.1 Ensemble Learning

An ensemble learning technique for a given task consists of a diverse set of learning algorithms, whose findings are combined in some manner to better the performance of an individual learner. The general idea behind such techniques is that the decision taken by a committee of models is likely to be better than the decision taken by a single model. This is because, each model in the committee may have learned some distinct inherent properties in the data which other models may have missed. So, each model may contribute some unique knowledge, impacting the overall performance positively. Therefore, by way of explanation, ensemble learning can be thought of as an opportunity to improve the decision-making process by choosing a diverse group of superior to average learners rather than an individual learner which may not give its best possible performance for a given task. This usually indicates that the generalizing power of a group of models is usually better than the generalizing power of an individual model. Figure 4.1 describes a typical framework of ensemble learning, where the outputs of different learning model are combined using a combination function.

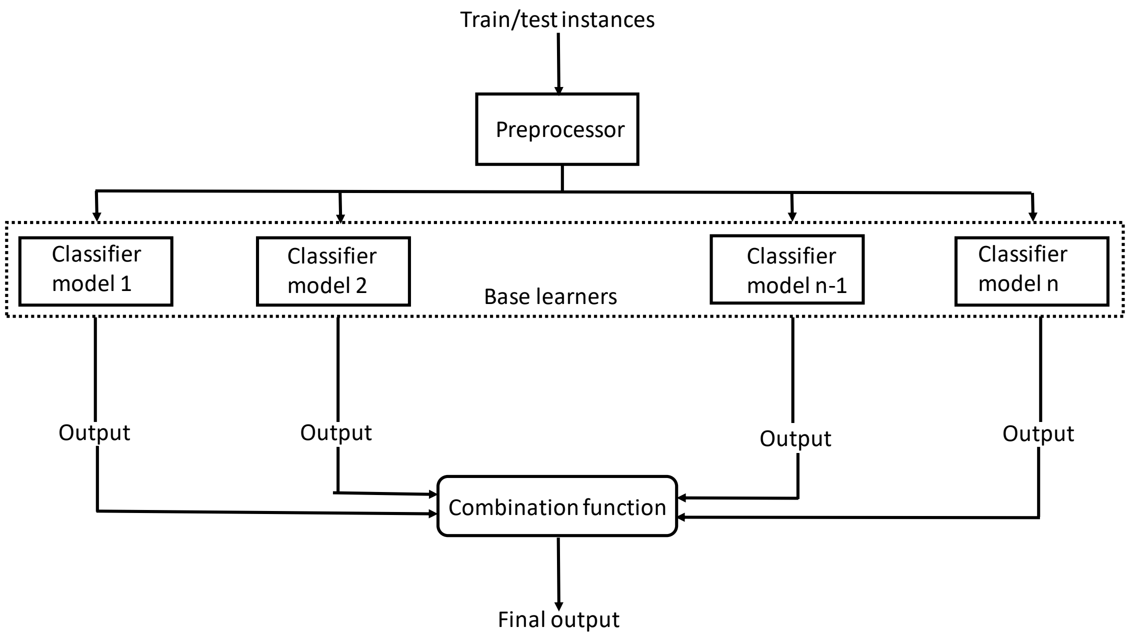

Figure 4.1: A Typical Ensemble Learning Framework

4.2.1.1 Types of Ensemble Learning

Depending on how a model learns, ensemble learning techniques can be of four types, supervised ensemble, unsupervised ensemble, semi-supervised ensemble and meta-ensemble. The first three kinds differ from each other in the manner of how they utilize the available knowledge. While supervised ensemble learning techniques make use of existing knowledge to classify new instances into respective classes (categories), unsupervised ensemble learning techniques first generate several groups with the help of clustering algorithms and then combine these groups towards generation of a best possible prediction using an appropriate consensus

function [154], [155], [156],[157]. On the contrary, semi-supervised ensemble techniques work with partial knowledge towards introduction of new knowledge or informative data to the existing distribution in the process to expand the training set [158],[159],[160],[161]. Meta-ensemble on the other hand, is an ensemble of ensembles created to avoid the biases of individual ensembles [162],[163],[164],[165],[166]. It was first introduced by Dettling [167].
Ensemble learning can be accomplished at different levels, such as attribute (feature) level [168][169] or decision level [170][171] based on the outputs given by the individual machine learning techniques. For example, at feature level, the feature ranks given by the individual feature selection algorithms can be combined using combination rules to get an aggregated list of best possible relevant features. The same learning models can be trained on random full-featured subsets of a dataset or versions of the same dataset with sampled subsets of features. The learning models are then tested on previously unseen samples. The final prediction of the models can be combined using different combination rules. The main goal is to increase the generalization capability of an ensemble, such that the final predictions are highly accurate.

4.2.1.2 Base Learners

Several individual learners participate to form an ensemble. Such learners, usually from distinct families, are referred to as base learners. In a supervised framework, these base learners take as input a set of labeled samples (training data) and learn significant characteristics from the data in order to classify the samples with appropriate labels. The output of the base learners are combined to obtain the final prediction of the ensemble. It is worth noting that the output of the individual classifier models can be either class labels or continuous outputs or even class probabilities. Depending on the type of output given by the base learners, the combination function is chosen. For example, majority voting or weighted majority voting works when combining class labels, whereas sum rule or mean rule works when combining continuous value outputs[123].

Definition 4.1 (Base learner)**.** Individual learners that constitute an ensemble are called the base learners.

4.2.1.3 Combination Function

A combination function tries to aggregate the outputs of learning models taking part in the ensemble process. The outputs to be combined can be in class label form, or continuous value outputs. Depending on the type of output of the base learners, the combination function needs to be chosen.

Definition 4.2 (Combination Function)**.** A function which combines the individual results of base learners to get the final output of the ensemble is called the combination function.

Class label Combination: To combine the class label outputs of learning models two simple strategies that can be used are: *Majority voting* and *Weighted majority voting.*

(a) *Majority voting*: Here, the ensemble chooses the class label with the most number of votes. That is, all the learning models have equal opportunities to choose their respective class label (these are counted as votes) and the label with most number of votes is chosen as the final decision of the ensemble. For example, let's assume there are 10 learning models and 6 learning models say that a particular instance say X_1 belongs to class A and 4 models assign it to class B, then the final output will be class A for the instance X_1 (i.e. go with the decision opted by the majority). This technique basically, can have three different cases [123].
Case 1: All learning models agree unanimously without any conflicts into predicting a single class label.
Case 2: At least one more than half the number of learning models agree into predicting a class label.
Case 3: Class label with the highest amount votes is chosen the winner.
Although very popular, majority voting technique has its own limitations because it may so happen that certain learning models are more suitable for a given task compared to others. In that case, weights must be assigned to the learning models.

(b) *Weighted majority voting:* Here, weights are assigned to the learning models when predicting a class label for a given test instance. This is because, out of all the participating learning models in the ensemble, some models may be more suitable compared to others. The suitable members of the ensemble are assigned higher weights than others. Lets assume that a learning model l_t takes a decision $d_{t,j}$ and chooses class c_j for an instance X_i. So, $d_{t,j} = 1$, if c_j is chosen otherwise 0. Subsequently, all the learning models are assigned weights based on their performances such that learning model l_t has weight w_t. For a class c_j the total weight can be given by the sum total of the product of individual weights of the learning models and respective decisions given by them. The final output is the class with the highest weighted vote. Therefore, the final output is class J according to the assumption if the Equation 4.1 holds true .

$$\sum_{t=1}^{T} w_t d_{t,J}(X_i) = max_{j=1}^{C} \sum_{t=1}^{T} w_t d_{t,j} \tag{4.1}$$

An elaborate description on how to combine continuous valued outputs is given in Appendix A.

4.2.1.4 Requirements of Base Classifier Selection

There are two primary requirements on how to select the base classifiers(learners).

(a) *Diverse nature:* If one of the classifiers overlook an inherent pattern in the data, there is a high probability that a similar classifier will also repeat similar mistakes. Such patterns may correspond to a distinguishing or interesting characteristic, in which case the classifiers will perform poorly. This is why, the classifiers must be chosen from a diverse set of families [172]. This will also make sure that the outputs of individual models are less correlated with each other.

(b) *Accuracy in the classifier performance:* The chosen base learners should be high performers in terms of accuracy individually or at least perform better than a random learner. If there are c classes in a dataset, a random classifier will have an average accuracy of $\frac{1}{c}$. On the other hand, a combination of weak learners, each performing worse than a random learner, may not give the desired quality outcomes.

4.2.1.5 Ensemble Methods

Over the years, several ensemble construction approaches have been developed. Some prominent ones are described next.

1. *Bagging* [173]: From the original dataset, samples are chosen at random with replacement to obtain different versions of the same dataset. It is known as a bootstrap technique and the samples are called bootstrap samples. A learning algorithm is then trained with these different sets of samples in parallel to obtain classifier models which assign class labels to the samples. The combination scheme used is majority voting, that is if majority of the base learners predict a sample to be of a particular class, then the corresponding label is assigned to it. Figure 4.2 illustrates an example of Bagging.

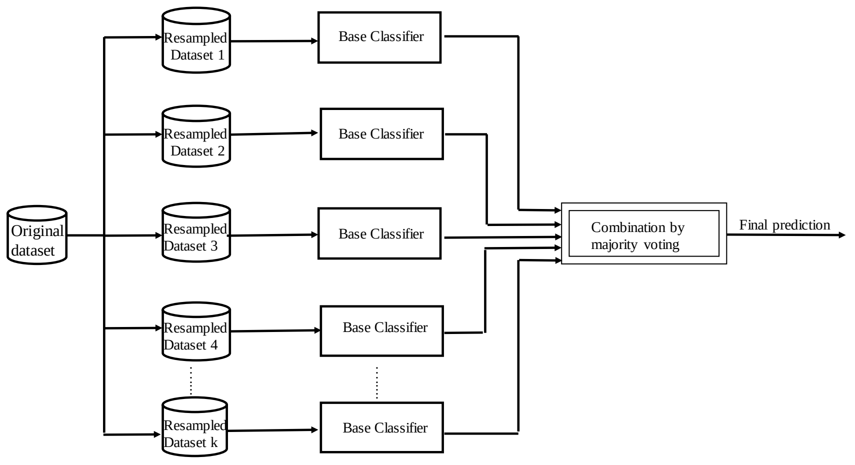

Figure 4.2: Bagging Example

Definition 4.3 (Bagging)**.** The process of constructing bootstrapped samples from the original dataset and giving these as input to base classifiers whose outputs are combined by majority voting is known as Bagging.

2. *Boosting* [174]: Boosting is an ensemble approach which works sequentially in iterations. It focuses on the misclassified instances in every iteration. In the first iteration, all the instances are assigned equal weights. In subsequent iterations, the misclassified samples of the previous iteration are given more weight, or are said to be boosted. Such a mechanism works well with weak learners as several weak learners can be combined to solve some hard task. For Boosting to work, the weak learners must perform better than random learners. *Adaboost*[175] is a popular Boosting model with which other learning

models such as Decision trees or Naive Bayes as a base learner can be used. Figure 4.3 illustrates an example of Boosting.

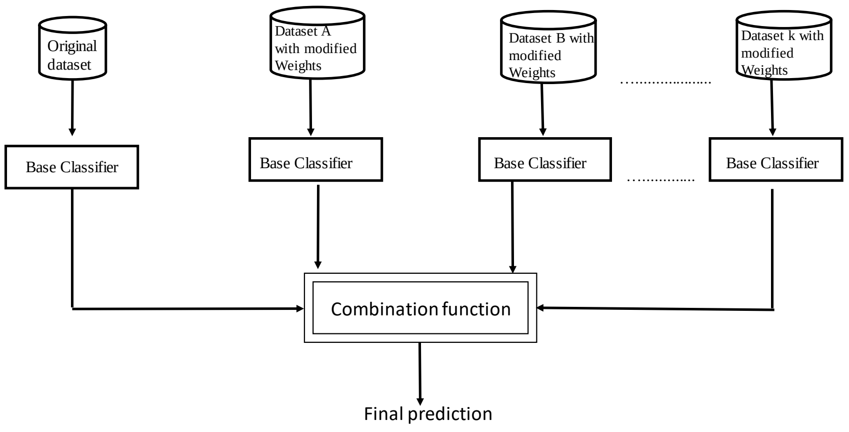

Figure 4.3: Boosting Example

Definition 4.4 (Boosting)**.** The process of building a sequence of classifiers that work together such that misclassified instances are reweighted to boost attention on them in each iteration is called Boosting.

3. *Stacking (also known as Stacked Generalization)* [176]: It is an ensemble approach where learning takes place in two levels. The layer-1 base classifiers are trained with a level-0 bootstrapped dataset. Their outputs are used as input by the next level meta-learner. As the name suggests, one layer of dataset and classifier is stacked over another layer of dataset and meta-classifier. It is called a meta-classifier because it learns from the behavior of set of classifiers above it. Figure 4.4 illustrates an example of Stacking.

   Definition 4.5 (Stacking (or Stacked Generalization))**.** The process of stacking one learning layer on top of the other such that the output of one layer is given as input to the next level meta learner is known as Stacking (or Stacked Generalization).

4.3 Data-centric Supervised Ensemble Proposed Framework

This section discusses in detail the proposed Data-centric Supervised Ensemble Framework for the empirical study. The framework consists of four engines, namely

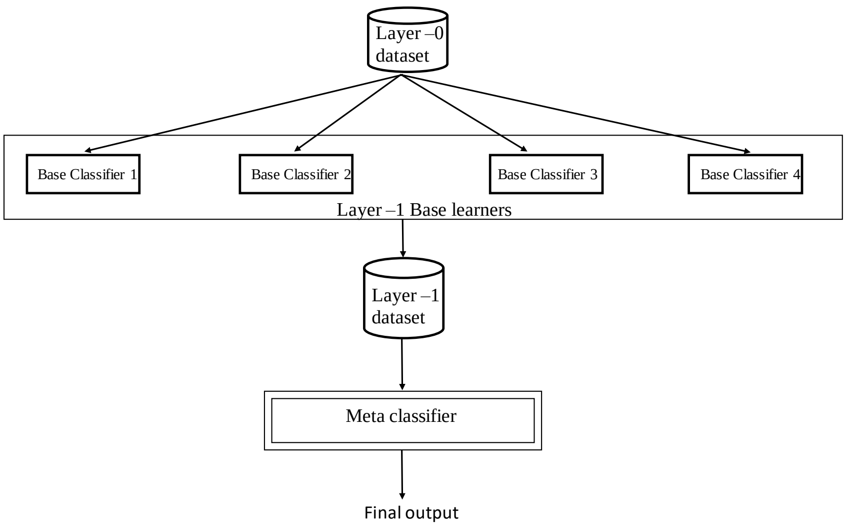

Figure 4.4: Stacking Example

preprocessing engine, parameter tuning engine, ensemble engine, and performance analysis engine. The data-centric framework illustrated in Figure 4.5 explains the methodology of the empirical study. A detailed description of the participating engines in the framework is provided below with necessary diagrams.

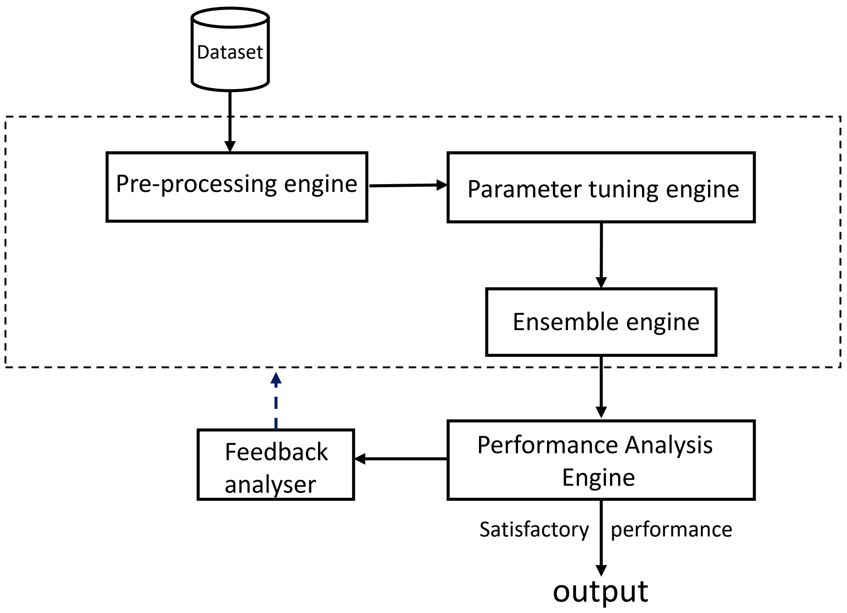

Figure 4.5: Data-centric Supervised Ensemble Framework

4.3.1 Preprocessing Engine

It is the first engine in the framework that consists of three sub-modules as shown in Figure 4.6. The first sub-module checks for the presence of any missing values. If missing values are not present, the next sub-module is executed. If missing values are however present, then they are either removed or appropriate missing value estimation techniques are applied. Missing values can be estimated in a number of ways as mentioned in [177]. Popular methods include k-NN Impute [178] and Row averaging [177]. The second sub-module helps in removing zero variance features, i.e. the features with no interesting patterns. Meaning, there may exist some features in a dataset whose values are same (for example: a column with all 0s or all 1s) all throughout. Such features do not contribute in any way for improving the performance of the predictor/classifier. In fact, the existence of such features will only downgrade the performance of the classifier. These features are therefore, removed in the pre-processing step as they do not contribute any knowledge in the prediction task and will in turn lead to overfitting the model. The third sub-module helps in normalizing each dataset. A suitable Normalization technique ensures that all the values are in the same scale [179]. For example, two features may have different range of values, one from say 0 to 100 and one from -1 to 1. In such cases, both these two features must be normalized and brought under the same range (may be from 0 to 1). However, before applying normalization technique, all the non-numeric data must first be converted to numeric data. The normalization step is absolutely necessary, particularly for predictors such as k-nearest neighbors which rely on distance based measures for prediction. This is because, feature values may not lie in the same range and hence, need to be explicitly brought into uniform scale of 0 to 1. In this work, *Min-Max* normalization technique [180] is used.

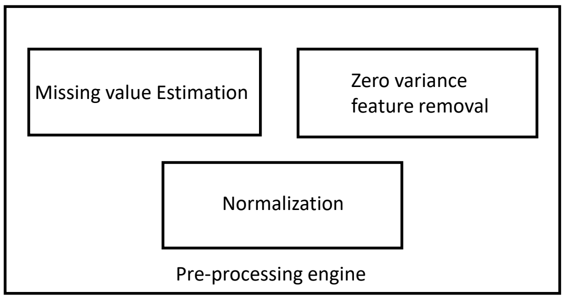

Figure 4.6: Preprocessing Engine

4.3.2 Parameter Tuning Engine

Parameter tuning engine is also known as the model selection engine. Model tuning or model selection is done to select appropriate set of hyper-parameters for each

model, and is specific to each dataset. This is the reason why the framework is a data-centric framework. For selecting the best possible set of hyper-parameters for a model, a grid is constructed with a list of potential values for each hyper-parameter. Each potential value of one hyper-parameter is checked with each potential value of every other hyper-parameter as data is fit into the model, and the technique is called GridSearch (searching the grid for best possible values). In this manner, in a permutation fashion the best possible set of hyper-parameter for each model is obtained. Here, the best possible set of hyper-parameter values for a model are evaluated in terms of accuracy, precision and recall. It is important here to note that, not all hyper-parameters of a model are tuned. Only crucial hyper-parameters which plays a potential role in determining the performance of a model is tuned. Figure 4.7 illustrates the working of this engine.
GridSearch is an additional step in fine tuning a model. Each model has default set of values for each hyper-parameter but most of the times model tuning the classifier gives a better result. This means, it may so happen for a classifier and for a specific dataset that model selection gives poorer results compared to the default hyper-parameter values of the model. In that case, the default values are chosen. After best possible parameter values are obtained, next the ensemble methods of Bagging, Boosting, Bagging combined with Boosting and Stacking are taken up.

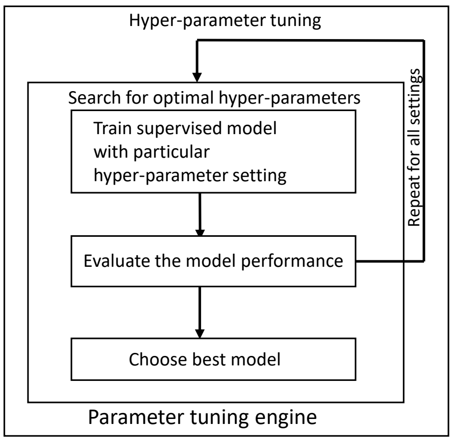
Hyper-parameter tuning
Search for optimal hyper-parameters
Train supervised model with particular hyper-parameter setting
Evaluate the model performance
Repeat for all settings
Choose best model
Parameter tuning engine

Figure 4.7: Parameter Tuning Engine

4.3.3 Ensemble Engine

The ensemble engine consists of methods such as Bagging, Boosting, Bagging-Boosting and Stacking (or Stacked Generalization) as shown in Figure 4.8. For

Bagging five inducers namely, k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Decision Trees (DT), Logistic Regression (LR) and Naive Bayes (NB) are used. For Boosting, AdaBoost, Extreme Gradient Boosting (XGB), Light Gradient Boosting (LGB), Hist Gradient Boosting (HGB) and Gradient Boosting (GB) are used. For Bagging-Boosting ensemble method, the bagging technique is combined with each of the boosting inducers separately. For Stacking however, the best inducers obtained from Bagging, Boosting and Bagging-Boosting methods are considered. The best inducers are then stacked one on top of the other to obtain the results. It is important to note that for each ensemble method employed, a consensus function (majority voting) is also applied to all the base inducers to obtain a final result.

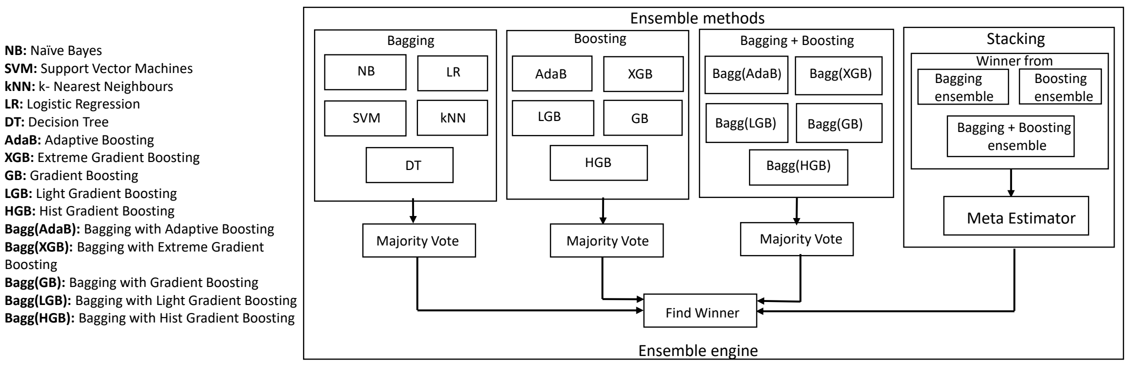
NB: Naïve Bayes
SVM: Support Vector Machines
kNN: k- Nearest Neighbours
LR: Logistic Regression
DT: Decision Tree
AdaB: Adaptive Boosting
XGB: Extreme Gradient Boosting
GB: Gradient Boosting
LGB: Light Gradient Boosting
HGB: Hist Gradient Boosting
Bagg(AdaB): Bagging with Adaptive Boosting
Bagg(XGB): Bagging with Extreme Gradient Boosting
Bagg(GB): Bagging with Gradient Boosting
Bagg(LGB): Bagging with Light Gradient Boosting
Bagg(HGB): Bagging with Hist Gradient Boosting
Ensemble methods
Bagging
NB
LR
SVM
kNN
DT
Boosting
AdaB
XGB
LGB
GB
HGB
Bagging + Boosting
Bagg(AdaB)
Bagg(XGB)
Bagg(LGB)
Bagg(GB)
Bagg(HGB)
Stacking
Winner from
Bagging ensemble
Boosting ensemble
Bagging + Boosting ensemble
Meta Estimator
Majority Vote
Majority Vote
Majority Vote
Find Winner
Ensemble engine

Figure 4.8: Ensemble Engine

4.3.4 Performance Analysis Engine

This engine is responsible for analysing the results obtained from the Ensemble Engine. Results if not satisfactory is forwarded to the Feedback Analyser. The Feedback Analyser then scrutinizes the output obtained from each of the engine in the process.

4.4 Experimental Results

In this section, the experiment results are reported on 6 datasets, and information on each of the datasets can be found in Table 4.1. For the ransomware multiclass dataset which contains a total of 12 classes (11 ransomware families and 1 normal family) all the results are shown in a class specific manner and separately from the rest of the datasets. This is because class specific data analysis helps to find intrinsic characteristics with respect to each class.

Table 4.1: Description of Datasets Used

Dataset name	#instances	#features	#classes
Android Dataset 1 [142]	2140	241	2
Android Dataset 2 [142]	1110	347	2
Phishing [144]	11,055	31	2
Kitsune Network Attack Dataset [145]	7,64,137	116	2
SWaT [139]	4,44,496	51	2
Ransomware Multiclass dataset [143]	1524	30,967	12

4.4.1 Bagging Results

Figures 4.9 and 4.10 presents the results for Bagging ensemble method for all the datasets in terms of Accuracy, Precision and Recall. In Appendix B.1 the hyper-parameter tuned values for the Bagging ensemble method with respect to all the classifiers considering all the datasets is enlisted. It is important here to note that for the Naive Bayes (Gaussian) model, there aren't any hyper-parameters to be tuned so GridSearch is not carried out for the model. With respect to Figure 4.9 it can be seen that Decision Tree (DT) when taken in a Bagging ensemble emerges as the winner (highest accuracy). Again, with respect to Figure 4.10, Support Vector Machine (SVM) clearly dominates, as out of the 11 classes it emerges as winner for a majority of 6 times.

4.4.2 Boosting Results

Figure 4.11 and 4.12 illustrates the results for Boosting ensemble method for all the datasets in terms of Accuracy, Precision and Recall. The hyper-parameter tuned values for Boosting ensemble method are enlisted in Appendix B.2 with respect to all classifiers considering all the datasets. With respect to both Figure 4.11 and Figure 4.12 it can be seen that Gradient Boosting outperforms other classifiers as it wins for majority of the datasets (in terms of highest accuracy).

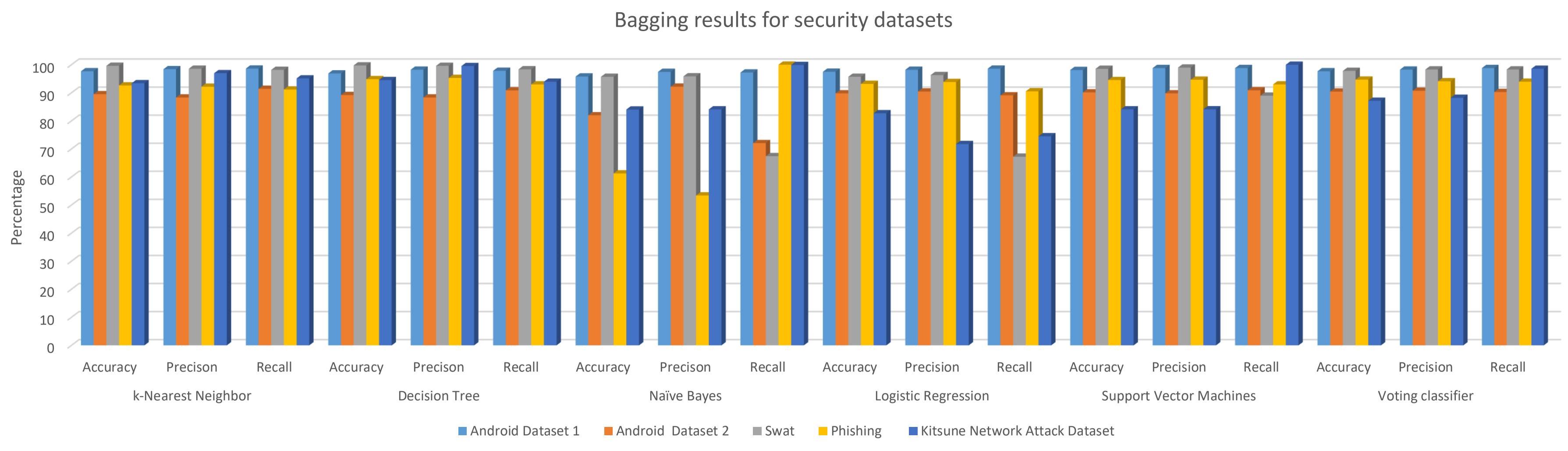

Figure 4.9: Bagging Results for Security Datasets

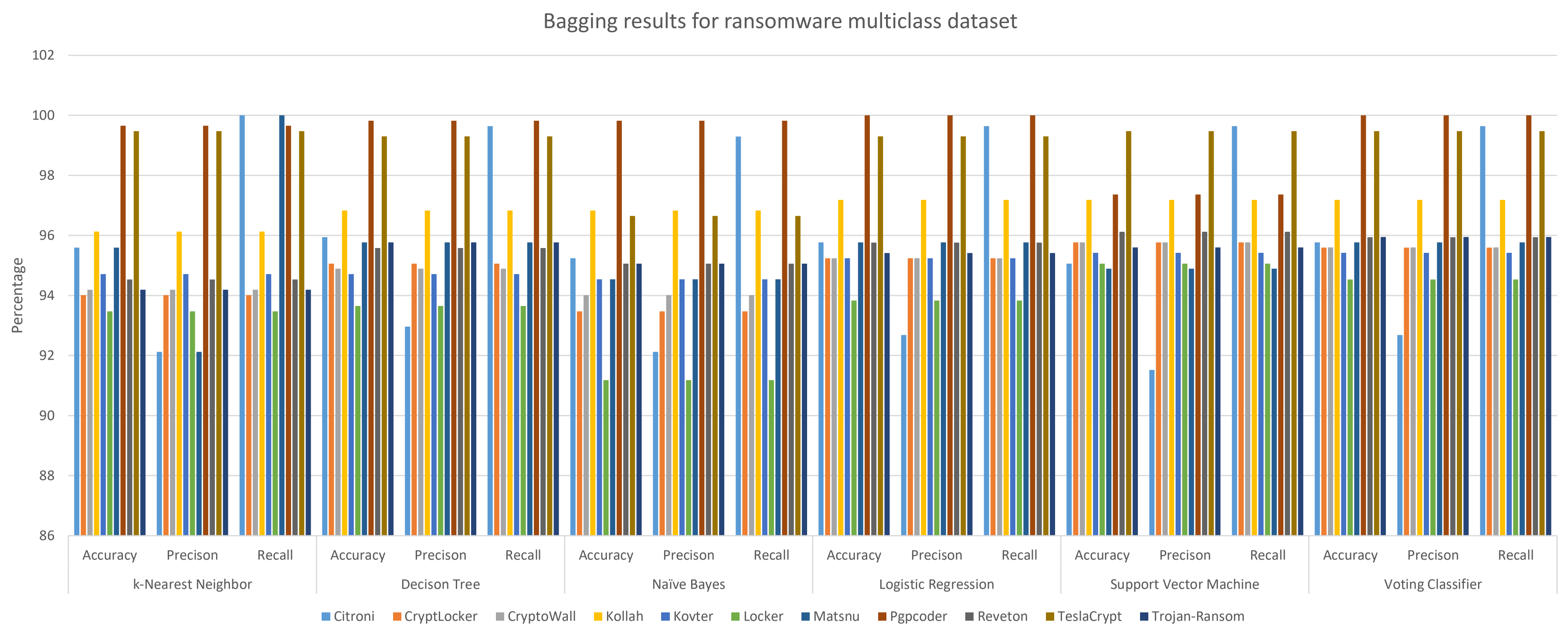

Figure 4.10: Bagging Results for Ransomware Multiclass Dataset

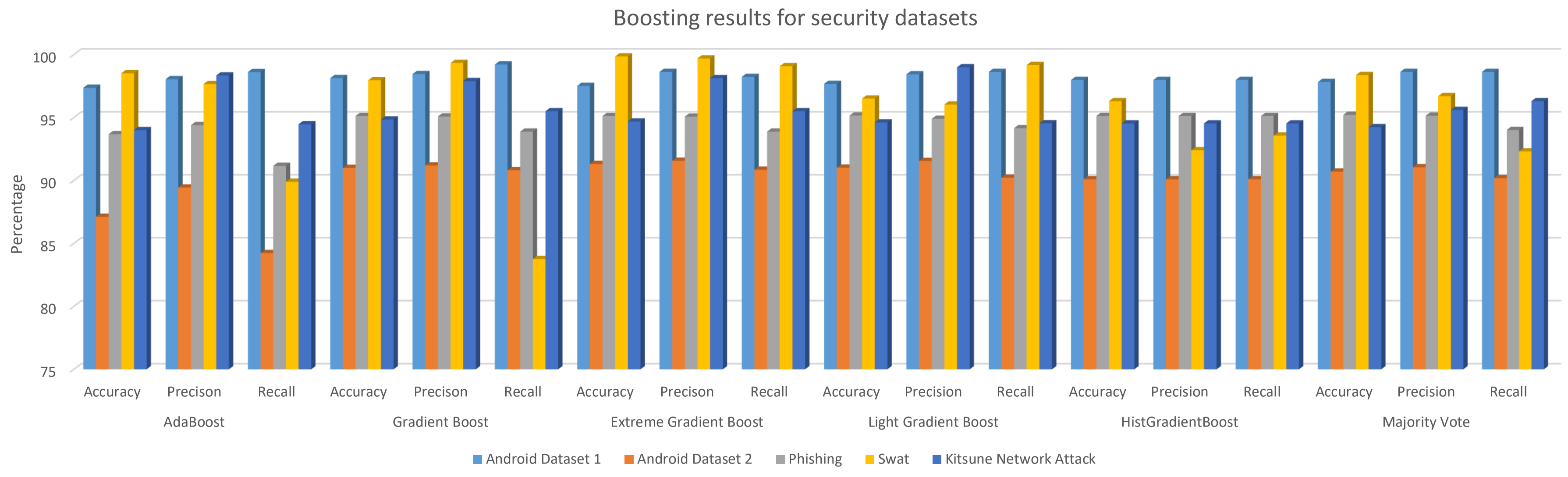

Figure 4.11: Boosting Results for Security Datasets

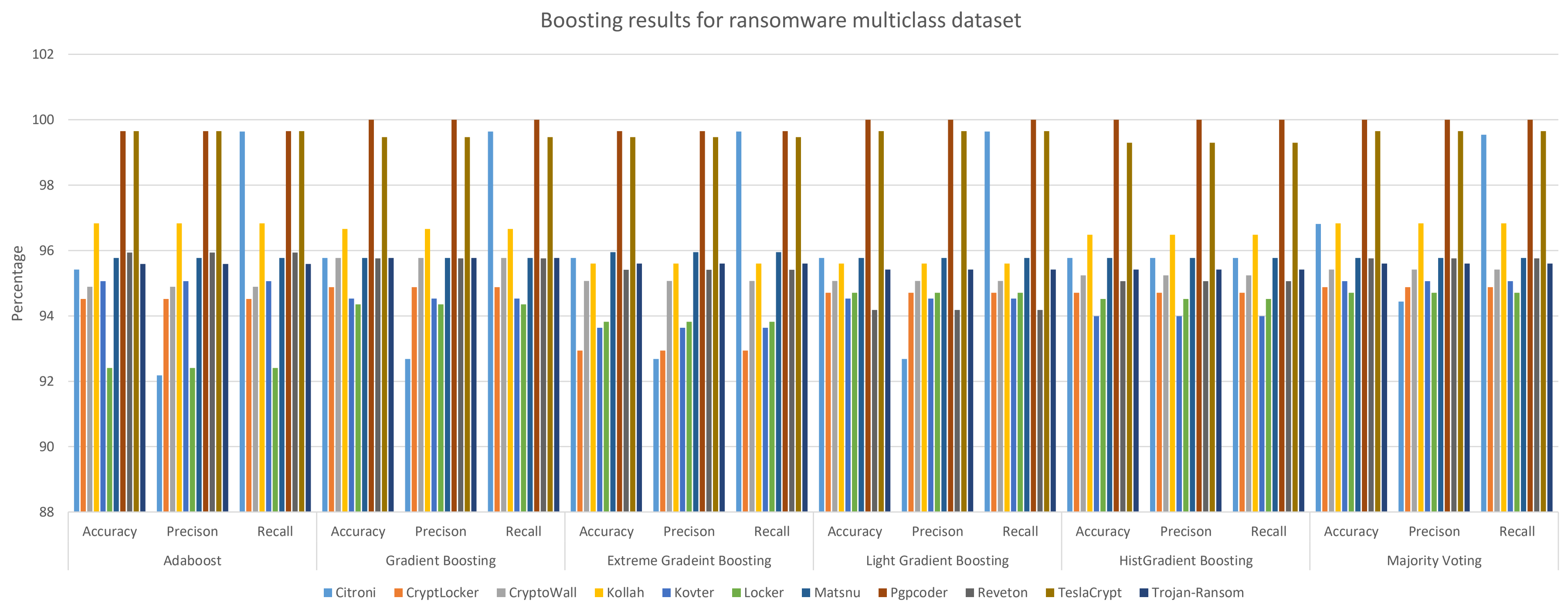

Figure 4.12: Boosting Results for Ransomware Multiclass Dataset

4.4.3 Bagging-Boosting Results

Figure 4.13 and 4.14 presents the illustrations for Bagging-Boosting combination in terms of Accuracy, Precision and Recall. All the Boosting classifiers are enveloped inside the Bagging ensemble method. Here, with respect to Figure 4.13, it can be seen that Extreme Gradient Boosting (XGB) emerges as the winner (in terms of accuracy). However, for Figure 4.14, Gradient Boosting wins for a total of 7 times overall and emerges as the sole winner.

4.4.4 Stacking Results

Table 4.2 and Table 4.3 show the Level 1 estimators used in the Stacking ensemble method for all datasets. In this level, a total of three models are used. Each of these three models are the winner from Bagging, Boosting and Bagging-Boosting ensemble method. As already mentioned earlier, in Level 2 a Random Forest Classifier is used. Illustrations for the same are shown in Figure 4.15 and 4.16.

Table 4.2: Stacking Ensemble Results

Ensemble Method: Stacking					
	Dataset name	Level 1 estimators	Accuracy	Precison	Recall
Security datasets	Android Dataset 1	Support Vector Machine Gradient Boosting Bagg(Gradient Boosting)	97.67	98.64	98.22
	Android Dataset 2	Support Vector Machine Extreme Gradient Boosting Bagg(Hist Gradient Boosting)	89.5	90.4	89.6
	SWaT	Decision Tree Extreme Gradient Boosting Bagg(Extreme Gradient Boosting)	96.5	97.3	96.8
	Phishing	Decision Tree Hist Gradient Boosting Bagging(Light Gradient Boosting)	94.75	94.72	92.93
	Kitsune Network Attack	Decision Tree Gradient Boosting Bagg(Extreme Gradient Boosting)	95.5	94.3	92.8

4.4.5 Observations

A few points have been observed during this study.

- Naive Bayes classifier works well when the data is not normalized.

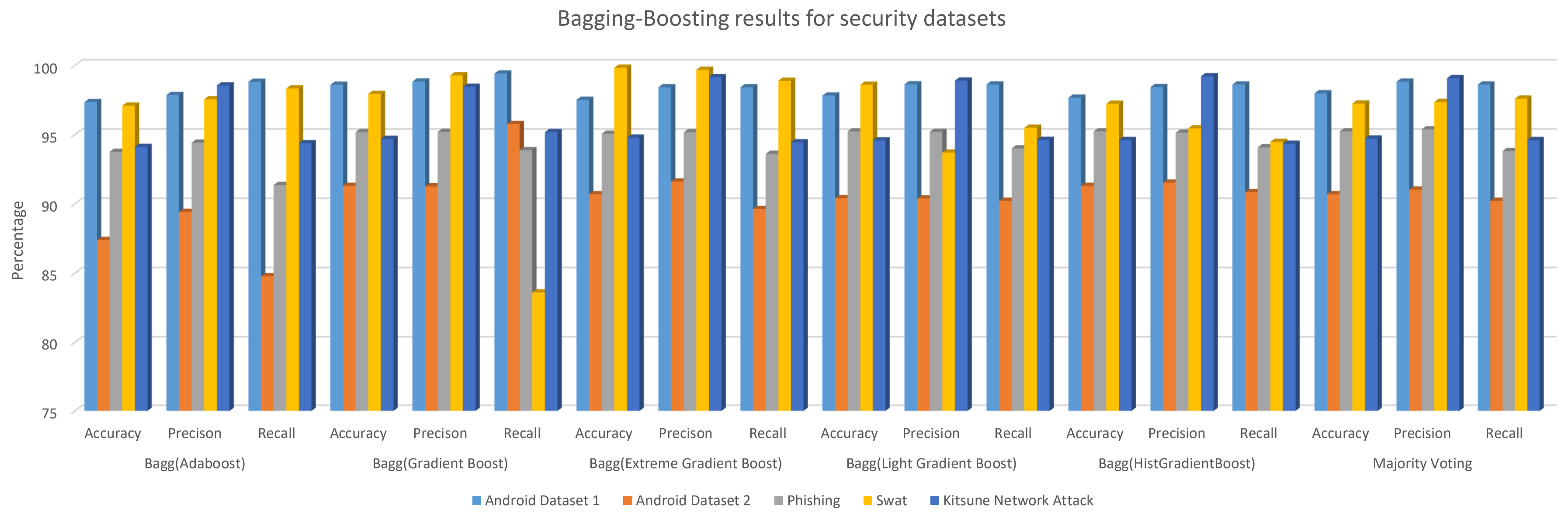

Figure 4.13: Bagging-Boosting results for Security Datasets

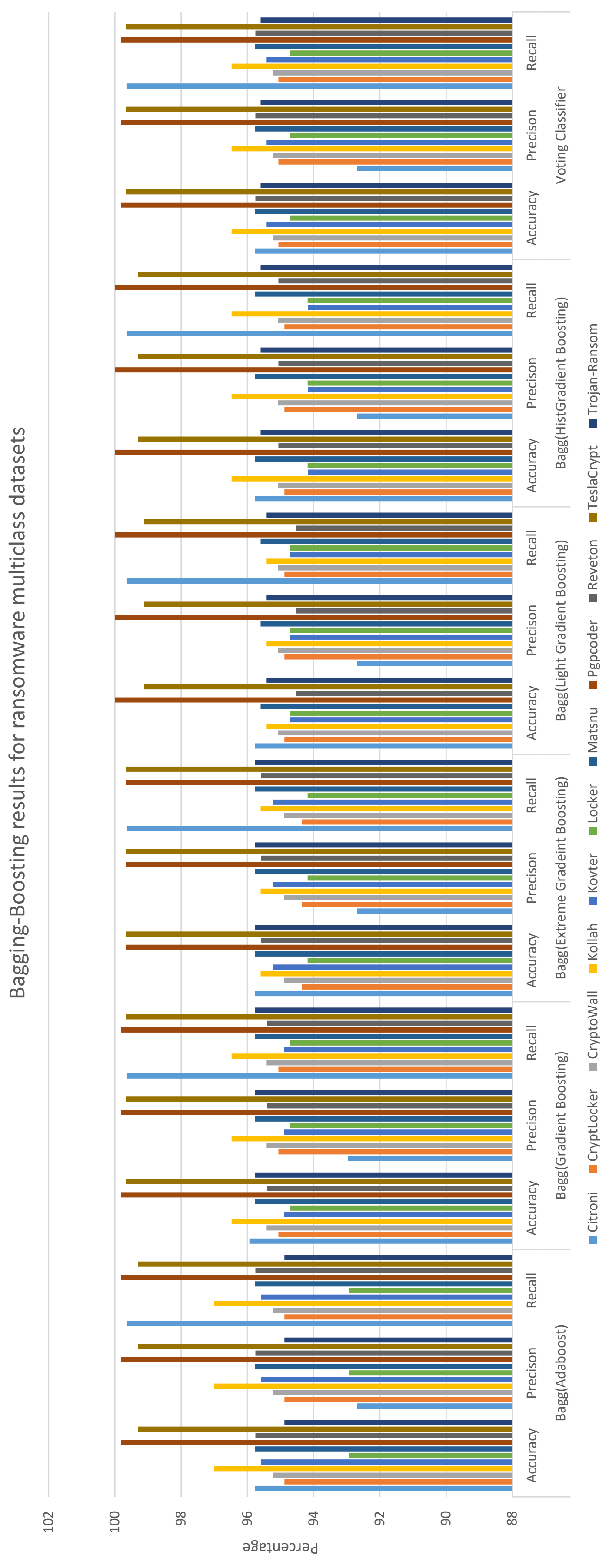

Figure 4.14: Bagging-Boosting results for Ransomware Multiclass Datasets

Table 4.3: Stacking Ensemble Results for Ransomware Multiclass Dataset

	Stacking Results			
	Level 1 classifier	Accuracy	Precison	Recall
Citroni	Decison Tree Gradient Boosting Bagg(Gradient Boosting)	95.94	92.68	99.64
CryptLocker	Support Vector Machine Gradient Boosting Bagg(Gradient Boosting)	95.06	95.59	95.42
CryptoWall	Support Vector Machine Gradient Boosting Bagg(Gradient Boosting)	95.6	95.42	95.42
Kollah	Support Vector Machine Gradient Boosting Bagg(AdaBoost)	97.01	96.83	97.01
Kovter	Support Vector Machine Adaboost Bagg(Adaboost)	94.17	95.94	94.88
Locker	Support Vector Machine Light Gradient Boosting Bagg(Gradient Boosting)	94.71	94.71	94.71
Matsnu	Decison Tree Extreme Gradient Boosting Bagg(AdaBoost)	95.95	95.95	95.95
Pgpcoder	k-Nearest Neighbors Gradient Boosting Bagg(Hist Gradient Boosting)	100	100	100
Reveton	Support Vector Machine Light Gradient Boosting Bagg(Adaboost)	96.29	95.58	95.58
TeslaCrypt	k-Nearest Neighbors Adaboost Bagg(Gradient Boosting)	99.65	99.65	99.65
Trojan-Ransom	Decison Tree Gradient Boosting Bagg(Gradient Boosting)	95.07	95.24	95.42

- Not all set of hyper-parameters need to be tuned, grid search in general takes a lot of time for execution in CPU. In an ideal world, with all the time and computational resource, all hyper-parameters can be fitted into a grid. But since, resources are limited hence, potential hyper-parameters with more impactful role only needs to be tuned for each model.

- Boosting takes more time to execute because of its sequential nature of execution compared to Bagging.

- For Bagging ensemble methods, Support Vector Machine (SVM) emerged as a clear winner for almost all the datasets.

- For Boosting, Gradient Boosting (GB) performs well in most of the datasets

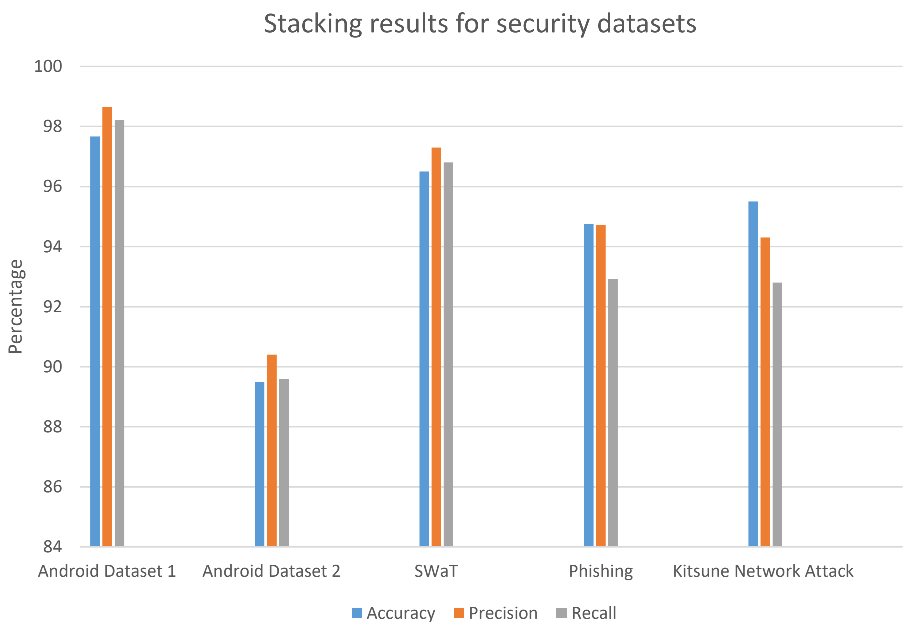

Figure 4.15: Stacking Results for Security Datasets

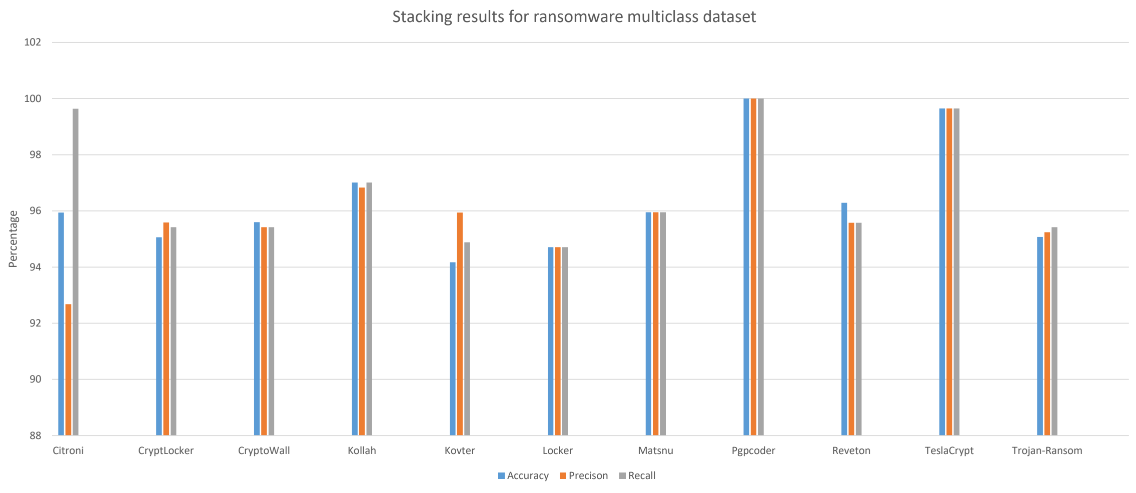

Figure 4.16: Stacking Results for Ransomware Multiclass Datasets

considered.

- For Bagging-Boosting however, Extreme Gradient Boosting (XGB) emerged as the winner.

4.4.6 Discussion

Ensemble learning is a relatively well-studied area in machine learning. Ensembles are flexible in structure and composition. An ensemble can be combined with other approaches in a seamless manner with some effort. It can be used not only with traditional machine learning methods like supervised and unsupervised learning but also with more advanced concepts like deep learning and transfer learning. An ensemble for supervised learning can be constructed at different levels, for example at sample level, or feature level or even at output level. Most of the research in the literature is focused on building ensembles at feature level and at output level. How well an algorithm works on previously unseen samples depends on the quality of the learning process. The quality of learning depends on a number of issues such as the choice of values of parameters and hyper-parameters of the learning model, the quality of data used, and preprocessing techniques employed, to name a few.

The next chapter discusses a traditional feature selection method named MICC-UD for the detection of XSS attacks. The method is based on a mutual information and correlation based score.