# Chapter 5

# A-Stacking and A-Bagging

## 5.1 Introduction

Ensemble learning is useful in overcoming the problems of single classifier systems, i.e. computational problems: when the learning process of a weak classifier is imperfect, statistical problems: when learning data is too small to capture the entire hypotheses space and representational problems: when the true target function cannot be found by any of the hypothesis from the hypotheses space [113]. One of the active areas of research in supervised learning has been to study methods for constructing good ensembles of classifiers [114].

Multiple classifier systems (MCS) sometimes referred to as a committee of classifiers or a mixture of experts have been exploited by various algorithms [72]. Bagging, boosting, stacking and random forest are the popular methods based on MCS paradigm. Multiple variants of these ensemble methods have been proposed and used in the past, such as Ubagging [115], AdaBoost [116, 117], AveBoost [118], conservative boosting [119], GA-stacking [120], cooperative ensemble learning system (CELS) [121], etc.

Stacking [122] and bagging [123] are two popular ensemble learning approaches applied in various real-world scenarios such as intrusion detection, spam classification, credit scoring etc. [124–129].

Stacking uses a meta-classifier to fuse the ensemble outputs, whereas voting, weighted majority voting etc. are the common ways to combine ensemble outputs in bagging. Also, the diversity in stacking is achieved by using heterogeneous classifiers on the same training set, whereas in bagging we try to gain diversity by using the same base classifier

on different training sets [130]. However, as these different training sets are bootstrapped from a single dataset, they are not entirely disjoint with each other, which results in low diversity [12]. Several modified versions of popular ensemble learning approaches have been proposed in the past [131–133], but to the best of our knowledge the adaptiveness of the algorithm towards the dataset has not been explored yet.

Although ensemble learning is well-known for this particular application, stacking has not been used for spoof fingerprint detection. We claim that for such applications, instead of straightforward usage of base classifiers, it is crucial to adapt to the features of the dataset and to adjust the learning model accordingly.

Christopher Merz [134] argues that having a disjoint set of classifiers is advantageous in the ensemble learning as it yields weakly correlated predictions. This motivated us to maintain the diversity of the ensemble by dividing the original training set into multiple subsets using clustering. In that way, we are able to generate a diverse set of classifiers by considering the features extracted from live and spoof fingerprint images of the dataset.

The models for fingerprint recognition are vulnerable to attacks by spoof fingerprints made of different moulds of substances like silicon, wood glue, latex, gelatin, etc. Therefore, it is required to perform liveness detection before fingerprint recognition to ensure that fabricated moulds are not used for authentication.

Local Binary Patterns (LBP) is an efficient way to determine the texture of an image by labelling each pixel with a binary value based on the thresholds on the neighbouring pixels [103, 135]. LBP considers the central pixel as the threshold and based on that it assigns the binary values to the neighbouring pixels. LBP value of the pixel is calculated by summing up the element-wise product of the binary values with their weights. LBP histograms are robust in terms of grayscale variations, making them suitable for spoof fingerprint detection, as they can easily incorporate fingerprints with skin distortions, different skin qualities, dry, moist or dirty skin.

### 5.1.1 Contributions

- We explore the behaviours of stacking and bagging with various base classifiers on spoof fingerprint detection problem.

- We emphasize that the learning algorithms must be adaptive towards the properties

inherent in the dataset.

- We establish that the diversity among the ensemble of classifiers can be achieved by performing clustering on the original training set and forming subsets of it.
- We propose adaptive models of stacking and bagging for spoof fingerprint detection and show their competitiveness on class balanced and imbalanced datasets.

## 5.2 Stacking

Stacking [122] is a learning approach based on ensemble learning which combines the predictions made by multiple base classifiers generated by using different learning algorithms $L_1, L_2, ...L_n$. These classifiers are trained on the same training data $D_{Train}$ containing examples in the form $s_i =< x_i, y_i >$, where $x_i$ is the input vector, and $y_i$ is the class label associated with it.

In the first phase, base classifiers $l_1, l_2, ...l_n$ make predictions for the query instance $x_q$. In the second phase, the meta-classifier $M$ combines the predictions made by base classifiers and predicts the final class label. A critical issue in stacking is the choice of meta-classifier. Comparative studies have been presented in the past [136] to analyse the performance of different learning algorithms as the meta-classifier. Logistic regression [137] is the current most popular learning scheme used in stacking as a meta-classifier.

The effort of using a meta-classifier is justified only when the performance of the ensemble of classifiers is better than the best individual base classifier [136]. Therefore, it is advised to consider the performance of the best individual base classifier to be the baseline for the performance of the ensemble.

### 5.2.1 A-Stacking

Algorithm 3 explains the methodology of A-Stacking and the conceptual model is illustrated in Figure 5.1. The first step is to perform clustering on the original training data $D_{Train}$ to form $n$ clusters $(c_1, c_2, ..., c_n)$ of instances. Instances belonging to the same cluster are similar to each other and dissimilar to the instances belonging to other clusters. By performing clustering, we are able to generate disjoint bags of instances. These bags of instances are used as individual training datasets. For each cluster $c_i$, we generate base

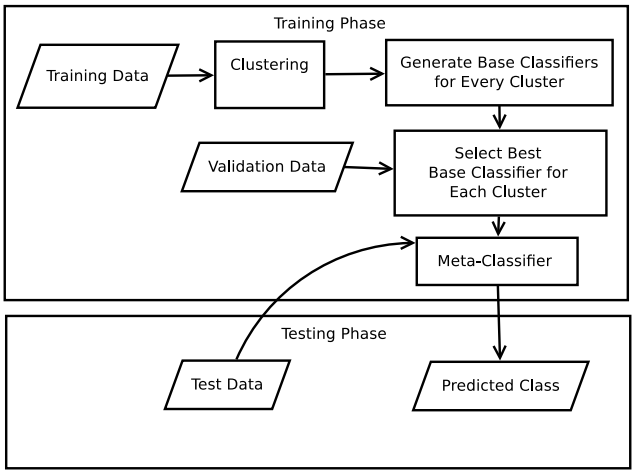


Figure 5.1: Conceptual model of A-Stacking.

classifiers $l_i^1, l_i^2, .., l_i^n$ by applying different classifiers $L_1, L_2, .., L_n$. These base classifiers are tested on the validation data $D_{valid}$ to select the best base classifier for each cluster. The individual accuracies of base classifiers are calculated using 10-fold cross-validation, and the best performing individual base classifier from each cluster is chosen and sent to the meta-classifier $M$. In the second phase, we combine the predictions made by the qualified base classifiers by using the meta-classifier. Therefore, for a query instance $x_q$ from the test data $D_{Test}$, first we consider predictions from the base classifiers, and later we combine these predictions using the meta-classifiers to get the final class label.

## 5.3 Bagging

Bagging [123] is a method of generating multiple versions of a base classifier by making bootstrap replicates of training data and using them to get an aggregated predictor. The performance of Bagging improves if used with an unstable learner, i.e. if the learner causes significant changes by perturbing the training set.

Let the size of the original training set $D_{Train}$ is $N$. Our task is to generate $n$ bags of size $N$ each by sampling $D_{Train}$ with replacement. These $n$ bags of instances may have duplicate instances, and the union of all these bags is a subset of $D_{Train}$. Each bag is used by the learning algorithm $L$ to generate a base classifier $l_i$. These $n$ base classifiers $l_1, l_2, ..., l_n$ are combined using majority voting to get the final predicted class label.

### 5.3.1 A-Bagging

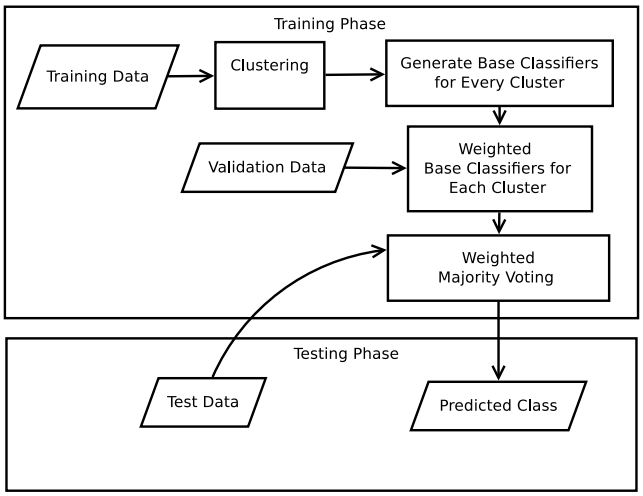


Figure 5.2: Conceptual model of A-Bagging.

Algorithm 4 explains the working mechanism of A-Bagging and the conceptual model is illustrated in Figure 5.2. We use the same idea as A-Stacking in A-Bagging. The first step is to use the original training data $D_{Train}$ for clustering to form $n$ clusters. These clusters act as $n$ bags of instances to be used for training and testing of individual base classifiers. In that way, we are able to generate disjoint bags of instances. The union of these bags of instances is the original training data $D_{Train}$. Therefore A-Bagging is different from traditional Bagging, which uses bootstrap samples of training data to train the base classifiers.

Later, these predictions are combined using weighted majority voting. The weights are assigned to the individual classifiers according to their performance on the validation data. Higher weight is assigned to the classifier having better performance. For classifying a query instance $x_q$, we associate the weight of the base classifier with its predicted class label. For the query instance $x_q$, the class label with the highest coefficient (i.e. the sum of weights of the base classifiers) is assigned. We use Equations 5.1 and 5.2 to assign weights to the classifiers.

$$w_y^x = \sum_{i=1}^{n} a_{iy}^x \tag{5.1}$$

where, $w_y^x$ is the total weight associated with the class label $y$ for an instance $x$. $n$ is the number of base classifiers and $a_{iy}^x$ is the accuracy of $i^{th}$ base classifier which has predicted the class label $y$ for instance $x$ on validation data. The final class label $y_f$ determined by

**Algorithm 3:** A-Stacking Algorithm.

**Data:**

training data $D_{Train}$,

test data $D_{Test}$,

validation data $D_{Valid}$,

a clustering algorithm $C$,

a set of classification algorithms $< L_1, L_2, .., L_n >$,

a meta-classifier $M$,

number of base classifiers $n$.

**Result:** ensemble classifier $Z$

1 $\{c_1, c_2, .., c_n\} \leftarrow C(D_{Train})$;

2 **for** *i= 1 to n* **do**

3 $< l_i^1, l_i^2, .., l_i^n > \leftarrow < L_1, L_2, .., L_n > (c_i)$;

4 Check the accuracies $< a_i^1, a_i^2, .., a_i^n >$ of $< l_i^1, l_i^2, .., l_i^n >$ on $D_{Valid}$;

5 Select the best performing base classifier $l_i$ from $< L_1, L_2, .., L_n >$ and send it to $M$;

6 Integrate the qualified base classifiers $< l_1, l_2, .., l_n >$ using the meta-classifier $M$;

7 **return** the ensemble classifier $Z$ integrated by $M$ to classify the instances in $D_{Test}$;

the weighted majority is given by Equation 5.2:

$$y_f^x = argmax_y(w_y^x) \tag{5.2}$$

The proposed models use the concept of multiple classifier systems by generating a set of base classifiers, pruning them based on their performance on validation data and integrating them using logistic regression or weighted majority voting. The idea of using multiple classifier systems is justified to reduce the false predictions made by a single classifier system that is untrained on the class of a given test instance.

The first part of the proposed models is to generate the ensemble of base classifiers. The ensemble is created by using the following components: (i) Training data $D_{Train} = (x_1, y_1), ....(x_n, y_n)$ contains $n$ training examples belonging to both live and spoof classes, where $x_i$ is a set of attributes generated from an image feature extraction algorithm (in

**Algorithm 4:** A-Bagging Algorithm.

**Data:**

training data $D_{Train}$,

test data $D_{Test}$,

validation data $D_{Valid}$,

a clustering algorithm $C$,

a classification algorithm $L$,

number of base classifiers $n$.

**Result:** ensemble classifier $Z$

1 $\{c_1, c_2, .., c_n\} \leftarrow C(D_{Train})$;

2 **for** *i= 1 to n* **do**

3 $l_i \leftarrow L(c_i)$;

4 Check the accuracy $a_i$ of $l_i$ on $D_{Valid}$;

5 **return** the ensemble $Z$ of $< l_1, l_2, .., l_n >$ to classify the instances in $D_{Test}$;

6 Classify instances in $D_{Test}$ using the weight co-efficients generated by Equation 5.2;

our case, LBP) and $y_i$ is the corresponding class label. (ii) A clustering algorithm $C$ is used to cluster the training examples based on the similarity inherently present in the data. The target is to create a group of clusters $c_1, ....., c_k$ where the examples belonging to one cluster possess similar values of attributes whereas the examples belonging to different clusters possess different values of the attributes defined by LBP feature. (iii) A classification algorithm $L$ (in A-Stacking $L = \{L_1, L2, .., L_n\}$) to train the base classifiers on each $c_i$. $L$ uses each $c_i$ to generate a base classifier $l_i$, which is used to make a decision individually. In this way, the decision boundary of each base classifier is different from others, resulting in an ensemble of diverse base classifiers. Later these decisions are integrated by logistic regression or weighted majority voting to decide for the whole ensemble. In this section, we present our experimental results and give a performance analysis of A-Stacking and A-Bagging.

## 5.4 Results and Discussion

### 5.4.1 Experimental Setup

We use python Weka wrapper to use clustering and classification functionalities of Weka [110]. All the original datasets have been randomized and divided into 80:20 ratio for training $D_{Train}$ and validation $D_{Valid}$, so that the validation set remains disjoint from the training set. We use Simple-kMeans [111] as our clustering algorithm which performs reasonably well on the chosen datasets with $k$=3. We encourage the readers to experiment with various values of $k$ and choose the optimal value empirically.

In this study, we compare A-Stacking and A-Bagging with the traditional stacking and bagging respectively with three base classifiers: SMO (Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier) [79], Random Forest (RF) [80] and Voted Perceptron (VP) [138]. These classifiers have been applied and proven to be efficient in various applications related to image classification. SVMs were used in fingerprint spoof detection in the recent literature [4, 5], whereas random forest and voted perceptron have also been a preferred choice of classifiers in spoof fingerprint and other image classification applications [139, 140]. We report the accuracy of the classifiers as well as their false positive rate and emphasize that both of these measures must be used simultaneously to evaluate the performance of the classifier. Accuracy (Acc) signifies the overall performance of the classifier on both classes; however, the false positive rate (FPR) is crucial in such applications because of the high misclassification cost. A spoof fingerprint being classified as the live fingerprint may have a considerable impact, therefore for a classifier to be used in a real-world domain, it is necessary to maintain high accuracy and low false positive rate.

For stacking, we use the above-mentioned three base classifiers along with logistic regression [137] as the meta-classifier. The meta-classifier is responsible for integrating the individual decisions of the base classifiers. As the meta-classifier brings overhead to the system, the effort is only justified when the overall accuracy of the ensemble is greater than the best performing base classifier SelectBest.

A variety of schemes have been proposed for combining the ensemble outputs [141], such as majority vote [114], weighted majority vote [114], average [114], weighted average [114], the Bayes approach [142], the probabilistic schemes [143], combination by a neural

network [142], etc. Majority voting has been a popular way of combining the ensemble outputs in bagging, whereas stacking requires the presence of a meta-classifier. In this study, for a fair comparison, we consider majority voting for both A-Bagging and bagging and logistic regression as a meta-classifier for both A-Stacking and stacking.

### 5.4.2 Datasets and Pre-processing

The description of the datasets used in this study is given in Table 3.1. We use livdet2011 dataset [1], which was used in fingerprint liveness detection competition 2011.

The image features need to be transformed into numeric attributes to perform classification on the fingerprint images. We do that by using LBP feature extraction on Weka. We use Binary Patterns Pyramid LBP from the imageFilter package of Weka[1]. It is a batch filter for extracting a pyramid of rotation-invariant local binary pattern histograms from images. Each local binary pattern represents an intensity pattern (e.g. an edge or a corner) around a point. A histogram of local binary patterns, therefore, encodes the larger-scale patterns that occur across regions of images. Local binary patterns are useful for texture and face recognition.

The fingerprint image features are transformed into 756 numeric attributes and one nominal class attribute making a high dimensional dataset.

### 5.4.3 Results

The results of the comparison between A-Stacking and stacking along with SelectBest on the class-balanced datasets measured by accuracy and false positive rate are described in Table 5.1.

It can be seen from Table 5.1 that the performance of stacking on DigitalPersona is lower than both A-Stacking and SelectBest. Therefore, the effort of using the meta-classifier in stacking is not justified. The performance of the proposed model is slightly lesser than its counterpart, but yet the performance is better than SelectBest, which supports the usage of meta-classifier. On average, A-Stacking performs well and yields competitive results to its counterparts.

The performance of A-Bagging is shown in Table 5.2. A-bagging performs well on

[1]https://github.com/mmayo888/ImageFilter

Table 5.1: Performance evaluation of A-Stacking on class-balanced datasets.

| **Dataset** | **Stacking (SMO+RF+VP)** | | **SelectBest** | | **A-Stacking (SMO+RF+VP)** | |
|---|---|---|---|---|---|---|
| | **Acc(%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** |
| **Biometrika** | 82.35 | 0.23 | 78.85(RF) | 0.25 | **82.55** | 0.23 |
| **DigitalPersona** | 82.85 | 0.23 | 85.30(SMO) | 0.18 | **85.35** | **0.18** |
| **ItalData** | 66.75 | 0.16 | 70.00(SMO) | 0.15 | **70.05** | **0.15** |
| **Sagem** | 86.24 | 0.10 | 84.23(SMO) | 0.14 | 85.11 | 0.10 |
| **Average** | 79.54 | 0.17 | 79.59 | 0.18 | **80.76** | **0.16** |
| **±Std Error** | 4.35 | 0.03 | 3.49 | 0.02 | **3.62** | 0.03 |

class-balanced datasets, but the performance is marginally below the rivals in some cases.

The performance of A-Stacking on imbalanced class datasets of Biometrika sensor is given in Table 5.3. It is evident that A-Stacking performs better than its counterparts in most of the cases, and it is always superior to the best individual base classifier SelectBest, which makes it more effective than conventional stacking.

The results of the comparison between A-Stacking and conventional stacking on class-imbalanced datasets of DigitalPersona sensor are given in Table 5.4. A-Stacking performs better than SelectBest in every case, however stacking is outperformed by SelectBest in three cases, which proves that the role of meta-classifier is fully justified in A-Stacking but not in stacking.

Table 5.5 shows the performance comparison on class-imbalanced datasets of ItalData sensor. It is evident that A-Stacking gives better performance than stacking in every case and its performance is always competitive with SelectBest, but it is never below the accuracy of SelectBest. It is worthwhile to note that SMO is the best individual base classifier in all the cases making it an excellent choice for spoof fingerprint detection.

Table 5.2: Performance evaluation of A-Bagging on class-balanced datasets.

| Dataset | Bagging (SMO) | | Bagging (RF) | | Bagging (VP) | | ABagging (SMO) | | ABagging (RF) | | ABagging (VP) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Acc (%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** | **Acc(%)** | **FPR(0-1)** |
| **Biometrika** | 79.10 | 0.29 | 80.15 | 0.23 | 78.05 | 0.33 | 78.60 | 0.31 | 78.85 | 0.25 | 77.15 | 0.29 |
| **DigitalPersona** | 85.25 | 0.17 | 81.75 | 0.26 | 78.10 | 0.34 | **85.30** | 0.18 | 79.75 | 0.27 | 77.70 | 0.34 |
| **ItalData** | 68.55 | 0.17 | 57.15 | 0.12 | 56.15 | 0.12 | **70.00** | **0.15** | **58.80** | 0.13 | 52.45 | 0.13 |
| **Sagem** | 85.36 | 0.12 | 83.05 | 0.16 | 77.70 | 0.16 | 84.23 | 0.14 | **83.10** | 0.16 | **78.97** | 0.20 |
| **Average** | 79.56 | 0.19 | 75.52 | 0.19 | 72.50 | 0.24 | 79.53 | 0.19 | 75.12 | 0.20 | 71.57 | 0.24 |
| **±Std Error** | 3.95 | 0.04 | 6.15 | 0.03 | 5.45 | 0.06 | **3.50** | 0.04 | **5.52** | 0.03 | 6.38 | **0.05** |

Table 5.3: Performance evaluation of A-Stacking on class imbalanced Biometrika datasets.

| Dataset | Stacking(SMO+RF+VP) | | SelectBest | | A-Stacking(SMO+RF+VP) | |
|---|---|---|---|---|---|---|
| | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** |
| **Live+Ecoflex+Gelatin** | 71.05 | 0.53 | 68.75(SMO) | 0.57 | 70.30 | 0.55 |
| **Live+Ecoflex+Latex** | 73.90 | 0.49 | 71.80(SMO) | 0.54 | **74.30** | **0.47** |
| **Live+Ecoflex+Silgum** | 74.80 | 0.44 | 71.45(SMO) | 0.51 | **74.95** | 0.44 |
| **Live+Ecoflex+WoodGlue** | 67.00 | 0.62 | 67.15(SMO) | 0.59 | **67.15** | **0.59** |
| **Live+Gelatin+Latex** | 79.25 | 0.35 | 76.25(SMO) | 0.41 | **79.50** | 0.35 |
| **Live+Gelatin+Silgum** | 78.85 | 0.33 | 75.80(SMO) | 0.39 | **80.20** | **0.30** |
| **Live+Gelatin+WoodGlue** | 73.60 | 0.46 | 72.25(SMO) | 0.48 | **73.70** | 0.46 |
| **Live+Latex+Silgum** | 77.05 | 0.39 | 73.60(SMO) | 0.47 | 76.55 | 0.41 |
| **Live+Latex+WoodGlue** | 69.80 | 0.55 | 70.95(SMO) | 0.50 | **71.15** | **0.52** |
| **Live+Silgum+WoodGlue** | 69.25 | 0.53 | 67.40(VP) | 0.56 | **69.35** | **0.50** |
| **Average** | 73.45 | 0.47 | 71.54 | 0.50 | **73.71** | **0.46** |
| **±Std Error** | 1.32 | 0.03 | 1.00 | 0.02 | 1.36 | 0.03 |

Table 5.4: Performance evaluation of A-Stacking on class imbalanced DigitalPersona datasets.

| **Dataset** | **Stacking(SMO+RF+VP)** | | **SelectBest** | | **A-Stacking(SMO+RF+VP)** | |
|---|---|---|---|---|---|---|
| | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** |
| **Live+Gelatin+Latex** | 71.70 | 0.52 | 69.65(SMO) | 0.54 | **72.15** | **0.49** |
| **Live+Gelatin+Playdoh** | 69.10 | 0.61 | 68.05(SMO) | 0.63 | 68.90 | 0.61 |
| **Live+Gelatin+Silicone** | 66.90 | 0.63 | 68.35(SMO) | 0.61 | **68.35** | **0.61** |
| **Live+Gelatin+WoodGlue** | 69.00 | 0.56 | 67.65(SMO) | 0.57 | **69.90** | **0.54** |
| **Live+Latex+Playdoh** | 72.90 | 0.50 | 71.20(SMO) | 0.55 | 72.60 | 0.51 |
| **Live+Latex+Silicone** | 71.90 | 0.51 | 71.80(SMO) | 0.52 | **72.80** | **0.49** |
| **Live+Latex+WoodGlue** | 65.95 | 0.63 | 70.00(SMO) | 0.54 | **70.05** | **0.54** |
| **Live+Playdoh+Silicone** | 67.75 | 0.61 | 67.80(SMO) | 0.61 | **67.80** | 0.61 |
| **Live+Playdoh+WoodGlue** | 75.95 | 0.44 | 75.40(SMO) | 0.44 | **76.25** | **0.43** |
| **Live+Silicone+WoodGlue** | 74.30 | 0.46 | 73.60(SMO) | 0.47 | 73.95 | **0.46** |
| **Average** | 70.54 | 0.55 | 70.35 | 0.55 | **71.27** | **0.53** |
| **±Std Error** | 1.05 | 0.02 | 0.83 | 0.02 | 0.86 | 0.02 |

Table 5.5: Performance evaluation of A-Stacking on class imbalanced ItalData datasets.

| **Dataset** | **Stacking(SMO+RF+VP)** | | **SelectBest** | | **A-Stacking(SMO+RF+VP)** | |
|---|---|---|---|---|---|---|
| | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** |
| **Live+Ecoflex+Gelatin** | 63.40 | 0.46 | 69.95(SMO) | 0.31 | **69.95** | **0.31** |
| **Live+Ecoflex+Latex** | 67.35 | 0.41 | 68.10(SMO) | 0.39 | **68.35** | 0.40 |
| **Live+Ecoflex+Silgum** | 60.00 | 0.44 | 61.90(SMO) | 0.40 | **61.90** | **0.40** |
| **Live+Ecoflex+WoodGlue** | 69.80 | 0.28 | 70.25(SMO) | 0.28 | **70.65** | 0.29 |
| **Live+Gelatin+Latex** | 68.35 | 0.28 | 69.95(SMO) | 0.25 | **69.95** | **0.25** |
| **Live+Gelatin+Silgum** | 59.40 | 0.42 | 64.35(SMO) | 0.27 | **64.35** | **0.27** |
| **Live+Gelatin+WoodGlue** | 67.95 | 0.31 | 70.20(SMO) | 0.24 | **70.20** | **0.24** |
| **Live+Latex+Silgum** | 63.60 | 0.38 | 66.05(SMO) | 0.32 | **66.05** | **0.32** |
| **Live+Latex+WoodGlue** | 70.80 | 0.33 | 71.45(SMO) | 0.30 | **71.45** | **0.30** |
| **Live+Silgum+WoodGlue** | 63.85 | 0.38 | 69.60(SMO) | 0.32 | **69.60** | **0.32** |
| **Average** | 65.45 | 0.37 | 68.18 | 0.31 | **68.24** | 0.31 |
| **±Std Error** | 1.26 | 0.02 | 0.98 | 0.02 | 0.99 | 0.02 |

Table 5.6: Performance evaluation of A-Stacking on class imbalanced Sagem datasets.

| **Dataset** | **Stacking(SMO+RF+VP)** | | **SelectBest** | | **A-Stacking(SMO+RF+VP)** | |
|---|---|---|---|---|---|---|
| | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** | **Accuracy(%)** | **FPR(0-1)** |
| **Live+Gelatin+Latex** | 76.91 | 0.38 | 79.76(SMO) | 0.30 | **79.76** | **0.30** |
| **Live+Gelatin+Playdoh** | 66.89 | 0.62 | 66.20(SMO) | 0.61 | 66.20 | **0.61** |
| **Live+Gelatin+Silicone** | 69.15 | 0.56 | 68.02(SMO) | 0.56 | **70.87** | **0.54** |
| **Live+Gelatin+WoodGlue** | 73.13 | 0.51 | 78.48(SMO) | 0.35 | **78.48** | **0.35** |
| **Live+Latex+Playdoh** | 80.10 | 0.32 | 80.84(SMO) | 0.29 | **81.18** | 0.29 |
| **Live+Latex+Silicone** | 76.62 | 0.40 | 75.63(SMO) | 0.40 | 76.12 | 0.40 |
| **Live+Latex+WoodGlue** | 68.95 | 0.58 | 69.84(SMO) | 0.56 | **69.89** | **0.55** |
| **Live+Playdoh+Silicone** | 66.55 | 0.59 | 67.63(SMO) | 0.59 | **67.68** | **0.59** |
| **Live+Playdoh+WoodGlue** | 75.04 | 0.43 | 76.71(SMO) | 0.38 | **76.71** | **0.38** |
| **Live+Silicone+WoodGlue** | 72.98 | 0.48 | 72.69(SMO) | 0.48 | 72.69 | 0.48 |
| **Average** | 72.63 | 0.49 | 73.58 | 0.45 | **73.96** | **0.45** |
| **±Std Error** | 1.45 | 0.03 | 1.71 | 0.04 | 1.65 | 0.04 |

The performance evaluation of A-Stacking on class-imbalanced Sagem datasets is given in Table 5.6. The performance is consistent and always better than SelectBest. The overall accuracy and false positive rate are better than its counterparts.

The performance of A-Bagging on class-imbalanced Biometrika datasets is evaluated in Table 5.7. We compare the accuracies and false positive rates of A-Bagging with traditional bagging on respective base classifiers. Here, the highlighted values show the better performance of A-bagging on its bagging counterpart. It is evident from Table 5.7 that A-Bagging performs better than bagging in most of the cases irrespective of the base classifier.

Table 5.8 shows the performance comparison between A-Bagging, and it's counterparts on class-imbalanced datasets of DigitalPersona sensor. Here, the performance is marginally lesser with SMO and VP as base classifiers in some cases, but the average performance is better with RF base classifier.

Table 5.9 shows the results on imbalanced-class datasets of ItalData sensor. The

Table 5.7: Performance evaluation of A-Bagging on class imbalanced Biometrika datasets.

| Dataset | Bagging (SMO) | | Bagging (RF) | | Bagging (VP) | | ABagging (SMO) | | ABagging (RF) | | ABagging (VP) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) |
| Live+Ecoflex+Gelatin | 67.25 | 0.61 | 62.30 | 0.74 | 60.65 | 0.75 | **68.75** | **0.57** | **64.65** | **0.69** | **64.50** | **0.67** |
| Live+Ecoflex+Latex | 69.15 | 0.58 | 67.20 | 0.64 | 67.10 | 0.63 | **71.80** | **0.54** | **68.80** | **0.60** | **68.20** | **0.61** |
| Live+Ecoflex+Silgum | 67.20 | 0.60 | 71.00 | 0.53 | 69.70 | 0.57 | **71.45** | **0.51** | **71.05** | **0.53** | **70.60** | **0.55** |
| Live+Ecoflex+WoodGlue | 67.45 | 0.59 | 56.10 | 0.87 | 61.65 | 0.75 | 67.15 | 0.59 | **57.60** | **0.84** | 56.05 | 0.86 |
| Live+Gelatin+Latex | 75.15 | 0.44 | 73.45 | 0.49 | 73.75 | 0.48 | **76.25** | **0.41** | **74.00** | **0.48** | **75.15** | **0.44** |
| Live+Gelatin+Silgum | 74.95 | 0.41 | 74.00 | 0.45 | 72.25 | 0.50 | **75.80** | **0.39** | **74.65** | **0.44** | **72.85** | **0.50** |
| Live+Gelatin+WoodGlue | 70.55 | 0.50 | 60.90 | 0.76 | 66.50 | 0.63 | **72.25** | **0.48** | 60.70 | **0.76** | 59.10 | 0.80 |
| Live+Latex+Silgum | 71.55 | 0.51 | 72.05 | 0.50 | 69.85 | 0.55 | **73.60** | **0.47** | **72.20** | **0.50** | **72.85** | **0.48** |
| Live+Latex+WoodGlue | 71.15 | 0.50 | 64.70 | 0.67 | 65.05 | 0.64 | 70.95 | **0.50** | 63.95 | 0.68 | **69.00** | **0.55** |
| Live+Silgum+WoodGlue | 66.25 | 0.59 | 65.95 | 0.60 | 60.55 | 0.72 | **67.05** | **0.58** | **66.70** | **0.59** | **67.40** | **0.56** |
| Average | 70.06 | 0.53 | 66.76 | 0.62 | 66.70 | 0.62 | **71.50** | **0.50** | **67.43** | **0.61** | **67.57** | **0.60** |
| ±Std Error | 1.00 | 0.02 | 1.87 | 0.04 | 1.49 | 0.03 | 1.01 | 0.02 | 1.81 | 0.04 | 1.94 | 0.04 |

Table 5.8: Performance evaluation of A-Bagging on class imbalanced DigitalPersona datasets.

| Dataset | Bagging (SMO) | | Bagging (RF) | | Bagging (VP) | | ABagging (SMO) | | ABagging (RF) | | ABagging (VP) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) |
| Live+Gelatin+Latex | 72.05 | 0.50 | 61.40 | 0.76 | 57.55 | 0.84 | 69.65 | 0.54 | **61.55** | **0.75** | 56.20 | 0.87 |
| Live+Gelatin+Playdoh | 68.15 | 0.63 | 65.75 | 0.68 | 66.25 | 0.66 | 68.05 | 0.63 | 65.45 | 0.68 | **67.40** | **0.64** |
| Live+Gelatin+Silicone | 68.95 | 0.59 | 62.40 | 0.73 | 65.45 | 0.68 | 68.35 | 0.61 | 62.35 | **0.72** | **67.70** | **0.61** |
| Live+Gelatin+WoodGlue | 69.10 | 0.55 | 53.05 | 0.93 | 60.90 | 0.76 | 67.65 | 0.57 | **60.95** | **0.76** | 52.45 | 0.94 |
| Live+Latex+Playdoh | 71.95 | 0.52 | 66.80 | 0.65 | 59.90 | 0.80 | 71.20 | 0.55 | **66.95** | **0.64** | **60.75** | **0.78** |
| Live+Latex+Silicone | 72.20 | 0.50 | 64.40 | 0.69 | 62.50 | 0.74 | 71.80 | 0.52 | **65.35** | **0.68** | **62.85** | **0.72** |
| Live+Latex+WoodGlue | 70.40 | 0.55 | 60.60 | 0.75 | 59.90 | 0.77 | 70.00 | **0.54** | **60.90** | **0.74** | 55.60 | 0.87 |
| Live+Playdoh+Silicone | 68.70 | 0.59 | 64.00 | 0.70 | 63.60 | 0.70 | 67.80 | 0.61 | 63.70 | 0.71 | **63.80** | 0.71 |
| Live+Playdoh+WoodGlue | 74.65 | 0.46 | 66.10 | 0.66 | 58.15 | 0.83 | **75.40** | **0.44** | **67.15** | **0.63** | 56.95 | 0.85 |
| Live+Silicone+WoodGlue | 73.50 | 0.45 | 64.60 | 0.69 | 58.40 | 0.82 | **73.60** | 0.47 | **65.00** | **0.68** | **61.10** | **0.76** |
| Average | 70.96 | 0.53 | 62.91 | 0.72 | 61.26 | 0.76 | 70.35 | 0.55 | **63.93** | **0.70** | 60.48 | 0.77 |
| ±Std Error | 0.70 | 0.02 | 1.27 | 0.02 | 0.97 | 0.02 | 0.83 | 0.02 | 0.75 | 0.01 | 1.62 | 0.03 |

Table 5.9: Performance evaluation of A-Bagging on class imbalanced ItalData datasets.

| Dataset | Bagging (SMO) | | Bagging (RF) | | Bagging (VP) | | ABagging (SMO) | | ABagging (RF) | | ABagging (VP) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) |
| **Live+Ecoflex+Gelatin** | 67.30 | 0.31 | 56.75 | 0.75 | 58.40 | 0.40 | **69.95** | **0.31** | **59.85** | **0.66** | 57.90 | **0.24** |
| **Live+Ecoflex+Latex** | 66.90 | 0.42 | 59.35 | 0.73 | 63.15 | 0.35 | **68.10** | **0.39** | **62.55** | **0.67** | 59.50 | **0.31** |
| **Live+Ecoflex+Silgum** | 62.65 | 0.39 | 56.25 | 0.73 | 56.00 | 0.30 | 61.90 | 0.39 | **58.20** | **0.67** | **57.40** | 0.45 |
| **Live+Ecoflex+WoodGlue** | 71.00 | 0.25 | 59.85 | 0.67 | 58.75 | 0.26 | 70.25 | 0.28 | **62.15** | **0.60** | **62.60** | **0.21** |
| **Live+Gelatin+Latex** | 70.70 | 0.25 | 64.00 | 0.58 | 63.00 | 0.33 | 69.95 | 0.25 | **66.40** | **0.53** | **66.35** | **0.22** |
| **Live+Gelatin+Silgum** | 67.55 | 0.25 | 57.85 | 0.63 | 60.80 | 0.41 | 64.35 | 0.27 | **59.30** | **0.58** | 58.15 | **0.32** |
| **Live+Gelatin+WoodGlue** | 72.50 | 0.23 | 62.10 | 0.59 | 60.20 | 0.24 | 70.20 | 0.24 | **62.90** | **0.56** | **61.95** | 0.25 |
| **Live+Latex+Silgum** | 64.70 | 0.38 | 61.30 | 0.63 | 61.20 | 0.23 | **66.05** | **0.32** | **61.75** | **0.57** | 55.10 | 0.37 |
| **Live+Latex+WoodGlue** | 71.35 | 0.29 | 64.00 | 0.58 | 60.00 | 0.24 | **71.45** | 0.30 | **66.35** | **0.52** | **69.10** | 0.29 |
| **Live+Silgum+WoodGlue** | 66.40 | 0.32 | 54.05 | 0.69 | 56.75 | 0.21 | **69.60** | **0.32** | **59.30** | **0.61** | **60.85** | 0.34 |
| **Average** | 68.10 | 0.31 | 59.55 | 0.66 | 59.82 | 0.30 | **68.18** | 0.31 | **61.87** | **0.60** | **60.89** | 0.30 |
| **±Std Error** | 1.01 | 0.02 | 1.06 | 0.02 | 0.75 | 0.02 | 0.98 | 0.02 | 0.90 | 0.02 | 1.35 | 0.02 |

performance is competitive with traditional bagging in most of the cases. However, there is a room for improvement in the false positive rate with SMO and VP base classifiers.

The performance of different bagging predictors on class-imbalanced datasets of Sagem sensor is given in Table 5.10. It is evident that A-Bagging produces competitive results both in terms of accuracy and false positive rate.

### 5.4.4 Discussion on Results

As we are proposing the adaptive versions of the ensemble learning algorithms, it is vital to consider an application where this adaptiveness is essential. We take spoof fingerprint detection problem to show the working mechanism of the proposed models. Livdet 2011 is a high dimensional dataset which makes it easier to provide a thorough analysis of the performance of the proposed models. In addition to that, it is important to show the models' behavior on class-imbalanced datasets. We accomplished this by taking the whole "live" class along with two subcategories of the "spoof" class.

As it is mentioned earlier, having an ensemble of weakly correlated base classifiers increases the diversity of the ensemble which results in better performance. We take this as our motivation and build classifiers on different subsets of training data. We perform clustering on the training data to form subsets of data. In this study, we considered

Table 5.10: Performance evaluation of A-Bagging on class imbalanced Sagem datasets.

| Dataset | Bagging (SMO) | | Bagging (RF) | | Bagging (VP) | | ABagging (SMO) | | ABagging (RF) | | ABagging (VP) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) | Acc(%) | FPR(0-1) |
| **Live+Gelatin+Latex** | 80.59 | 0.28 | 70.92 | 0.56 | 58.15 | 0.81 | 79.76 | 0.30 | 70.72 | **0.56** | **61.73** | **0.72** |
| **Live+Gelatin+Playdoh** | 68.56 | 0.57 | 62.72 | 0.73 | 65.91 | 0.66 | 66.20 | 0.61 | 61.59 | 0.75 | 65.07 | 0.67 |
| **Live+Gelatin+Silicone** | 69.49 | 0.54 | 66.99 | 0.64 | 67.19 | 0.59 | 68.02 | 0.56 | 66.01 | 0.65 | **67.42** | 0.61 |
| **Live+Gelatin+WoodGlue** | 77.55 | 0.35 | 66.25 | 0.67 | 63.26 | 0.69 | **78.48** | 0.35 | 65.61 | 0.69 | 62.42 | 0.72 |
| **Live+Latex+Playdoh** | 80.35 | 0.31 | 71.31 | 0.54 | 62.27 | 0.72 | **80.84** | **0.29** | **73.72** | **0.48** | 60.70 | 0.75 |
| **Live+Latex+Silicone** | 75.39 | 0.41 | 72.93 | 0.50 | 67.97 | 0.51 | **75.63** | **0.41** | **73.37** | **0.49** | 67.23 | 0.55 |
| **Live+Latex+WoodGlue** | 68.90 | 0.57 | 63.45 | 0.69 | 61.83 | 0.68 | **69.84** | **0.56** | 64.34 | **0.68** | 61.24 | **0.66** |
| **Live+Playdoh+Silicone** | 67.63 | 0.59 | 65.76 | 0.65 | 65.91 | 0.63 | **67.63** | **0.59** | 65.52 | **0.65** | **66.11** | **0.62** |
| **Live+Playdoh+WoodGlue** | 77.21 | 0.38 | 69.74 | 0.59 | 69.44 | 0.54 | 76.71 | **0.38** | 69.44 | 0.59 | 68.36 | 0.59 |
| **Live+Silicone+WoodGlue** | 72.29 | 0.48 | 70.57 | 0.56 | 65.76 | 0.57 | **72.69** | **0.48** | **70.67** | 0.56 | **65.81** | **0.56** |
| **Average** | 73.79 | 0.45 | 68.06 | 0.61 | 64.76 | 0.64 | 73.58 | 0.45 | **68.10** | **0.61** | 64.61 | 0.64 |
| **±Std Error** | 1.59 | 0.03 | 1.11 | 0.02 | 1.06 | 0.02 | 1.71 | 0.04 | 1.28 | 0.03 | 0.90 | 0.02 |

Simple-kMeans as the clustering algorithm, but the models are independent of the choice of the clustering algorithm. In A-Stacking, we use multiple base classifiers on the subsets of data, and the performance on both class-balanced and imbalanced datasets as given in Tables5.1,5.3,5.4,5.5,5.6 is competitive to its counterparts. We compared the results with the traditional stacking approach as well as with the best individual base classifier of the ensemble. As our results are always better than SelectBest, we justify the effort of using a meta-classifier. It is evident from our experimental results that traditional stacking often gives lower accuracy than the SelectBest, which is against the argument.

In A-Bagging, we worked with the same motivation, and here also, the results are competitive. We used a weighted majority voting scheme to combine the predictions made by the base classifier on different subsets of data. We compared the performance of A-Bagging with conventional bagging with corresponding base classifiers on class balanced and imbalanced spoof detection datasets and showed that the results are equally well on both types of datasets.

We claim that the proposed adaptive algorithms are generic and can be applied in various applications. The motivation is to draw attention towards the properties intrinsic to the datasets, which makes the proposed algorithms different from the traditional ensemble learning algorithms. Therefore, A-Stacking and A-Bagging are not restricted

to fingerprint spoof detection and can be exploited for applications where stacking and bagging have been used, i.e., network intrusion detection [124], credit scoring [144–146], bio-informatics [147], Nosiheptide fermentation product concentration prediction [148], wireless sensor network target localization [149], predicting high performance concrete compressive strength [150], bankruptcy prediction [151], etc.