SMART E-COMMERCE USING CUSTOMIZED ALGORITHMS

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at University of Virginia, Department of Computer Science.

By

LILIYA IVANOVA BESALEVA

DOCTOR OF PHILOSOPHY

Graduate advisor: Professor/Assoc. Chair Alfred C. Weaver

Committee chair: BP America Professor Jack Stankovic

Committee members: Associate Professor Worthy Martin, Assistant Professor Hongning Wang, Professor Larry Richards

University of Virginia, Department of Computer Science

Charlottesville, Virginia

September, 2017

Acknowledgements

I would like to thank my academic advisor Professor Alfred Weaver for his never-ending guidance, expertise and patience in the preparation of this work, and for being such a wonderful friend and mentor for me all these years. He graciously took me in as his graduate student, trained me to be my best self and helped me persevere in this long journey of pursuing my degree. I would, also, like to thank my family for their constant support in trying times, and their unconditional faith in my success.

I am very honored and truly grateful to have had the chance to work with BP America Professor Jack Stankovic, Assoc. Professor Worthy Martin, Professor Larry Richards and Assoc. Professor Hongning Wang as part of my committee. Their insightful feedback and guidance aided me in my efforts to dig deeper, and challenge myself in the exploration of this interesting topic.

Finally, I would like to express my gratitude to the University of Virginia and all of the wonderful staff members of the Computer Science Department for their help, kindness and generosity. From the kind, welcoming and altogether wonderful staff members – Brenda, Christine, Kim, Janet, Debby, Beth, Wendy, Essex, Scott, Barbara and so many more, to the truly phenomenal Department Chair – Harry Douglas Forsyth Professor Kevin Skadron, and all of the amazing professors at the University of Virginia who got me through these years – the biggest thank you from the bottom of my heart. You made UVa my true home in the United States, and I will never forget you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

SMART E-COMMERCE PERSONALIZATION USING CUSTOMIZED ALGORITHMS

By Liliya Ivanova McLean (Besaleva)

Submitted to the University of Virginia Department of Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

Applications for machine learning algorithms can be observed in numerous places in our modern lives. From medical diagnosis predictions to smarter ways of shopping online, big fast data is streaming in and being utilized constantly. Unfortunately, unusual instances of data, called imbalanced data, are still being ignored at large because of the inadequacies of analytical methods that are designed to handle homogenized data sets and to "smooth out" outliers. Consequently, rare use cases of significant importance remain neglected and lead to high-cost loses or even tragedies. In the past decade, a myriad of approaches handling this problem that range from data modifications to alterations of existing algorithms have appeared with varying success. Yet, the majority of them have major drawbacks when applied to different application domains because of the non-uniform nature of the applicable data. Within the vast domain of e-commerce, we are proposing an innovative approach for handling imbalanced data, which is a hybrid meta-classification method that will consist of a mixed solution of multimodal data formats and algorithmic adaptations for an optimal balance between prediction accuracy, sensitivity and specificity for multiclass imbalanced datasets. Our solution will be divided into two main phases serving different purposes. In phase one, we will classify the outliers with less accuracy for faster, more urgent situations,

which require immediate predictions that can withstand possible errors in the classification. In phase two, we will do a deeper analysis of the results and aim at precisely identifying high-cost multiclass imbalanced data with larger impact. The goal of this work is to provide a solution that improves the data usability, classification accuracy and resulting costs of analyzing massive data sets in e-commerce.

LEGEND

Chapter 1 introduces the issue of big data in e-commerce and how real life situations often lead to imbalanced use cases with a single or multiple sets of underrepresented data points. Section 1.1 describes the background of this dissertation and in particular it gives examples of e-commerce use cases, states the problem to which we are proposing a solution and lists our contributions. Section 1.2 reviews in detail scientific approaches to handling imbalanced data based on both major groups of alterations – algorithm generation and data modifications.

Chapter 2 presents a comprehensive review of classic and modern algorithms and data manipulation approaches to handling imbalanced data. Section 2.1 focuses on sampling-based methods covering both categories of under and over-sampling. Section 2.2 talks about cost-based methods in cost-sensitive SVM, decision trees and neural networks. Section 2.3 reviews active learning-based methods which is a semi-supervised approach for interactively querying the user. Next, in section 2.4, kernel-based methods are discussed. Finally, in section 2.5 we present the problem of multiclass imbalance problems.

Chapter 3 reviews bagging and boosting meta-algorithmic techniques which pool decisions from multiple classifiers. In section 3.1, standard bagging and boosting methodologies are discussed with examples of basic bagging and boosting algorithms. In contrast, sections 3.2 and 3.3 provide information about online and cost-sensitive advanced bagging and boosting techniques, such as AdaBoost (Adaptive Boosting Algorithm), UnderOverBagging and SMOTEBagging algorithms.

Chapter 4 discusses our solution approach, as well as some of the e-commerce specific challenges that are unique to this work. In this chapter, there is a strong focus on cost-sensitive offline and online solutions due to the particular usability of this approach within commercial applications.

Chapter 5 addresses the experimental setup and data utilized within this work. In

addition, here we are more deeply considering modern computational environments and provide justification for our chosen systems design.

Chapter 6 focuses on the achieved results for minority and majority classes. Section 6.1 discusses multiclass minority classes, while Section 6.2 is about multiclass majority classes. And finally, we unify both use cases in Section 6.3 where we review our success measures and achieved advancements.

Chapter 7 summarizes all discoveries of this work and sharpens the focus on the current achievements and areas of opportunities. And it finally concludes with a prediction of upcoming machine learning trends in e-commerce, as well as a reflection on probable future development of the current work.

Chapter 1

1. Introduction

Modern e-commerce has transformed from a one-size-fits-all experience into a tightly knit network of peer-reviewed, social media-shareable, trends-obsessed, customer-centric platforms for highly personalized shopping and selling.

Shoppers have begun expecting a 'smart' experience from their preferred online merchandisers that accurately represents social events and in-session behavior. Because mishandled e-commerce opportunities can cause significant financial losses to large businesses, we are exploring how to better personalize the shopping experience using machine learning (ML) techniques.

Example: Shopping. Ms. Jones is a regular in-store shopper at her favorite home improvements store BigBox. She often goes there to find little accent pieces for home decoration or supplies necessary for updating and fixing something in her house. Recently, BigBox extended its online catalogue to include items beyond its in-store assortment, so Ms. Jones decides to begin using their digital business as well. There she searches for throw pillows and matching sofa sets because she would like to update the look of her living room. In her search results for "throw pillows" in the BigBox website, the search ranking algorithm first returns throw pillows of different shades of turquoise with geometric patterns because in her previous in-store shopping trips, Ms. Jones has purchased paint and other decoration elements which would match very nicely with these colors and patterns. On the item pages of the items returned from Ms. Jones' search query, there are recommendations for other

decoration pieces, including sofa sets, which would be complimentary to the item at which she is currently looking. Finally, during her checkout process, additional matching items are recommended which, when purchased with the items in her cart, provide additional savings.

Example: Apparel Avatar. Ms. Brown loves shopping for apparel online because of the convenience it provides--but she does not feel confident that each item she selects will fit her well because she cannot try it on as she does in a physical store. Fortunately, the big retail store ClothesForAll has a new invention that helps people like Ms. Brown. In the store, there is a walk-through full body scanner, similar to the scanners at airports, which makes a 360-degree scan of Ms. Jones' body. Based on that scan, a unique online avatar representing Ms. Jones' measurements is created and linked to her store account. After using that avatar for a few shopping occasions, she begins to see two main changes in her shopping experience. First, when searching for a specific clothing article, she is first shown available clothing in her size that best suits her body type, and then other items follow. That change occurs because the store's website uses machine learning algorithms which, based on Ms. Brown's profile information, customize her search results. Second, when looking at a piece of clothing, she can choose to see it on her avatar, and therefore know what it would look like on her body. Additionally, based on the avatar measurements, personalized advertisements are automatically generated and shown.

Example: Medical Prognosis. "Welcome back, Ms. White, the doctor is now going to explain what she sees in your last set of images and suggest some strategies." Ms. White puts on an Oculus Rift head-mounted display that allows her to see the same 3D images as her doctor.

Dr. Black welcomes Ms. White and begins. "These three white dots you see are small cancers on your liver. We have used Machine Learning algorithms to analyze six thousand cases of liver cancer recorded over the past ten years, and so first we use that capability to predict, based upon past experience, what happens if these cancers go untreated." The video begins and the white dots grow dramatically in size until they wrap themselves around the spinal cord.

Dr. Black continues. "Now let's see what happens if we treat them with surgery. You can see that the dots disappear on the liver but more form in the lungs. These are new cancers

accidentally spread by the surgery itself. So now let's look at the combination of surgery and chemotherapy. The dots disappear from the liver, appear in the lung, and then disappear from the lung. This third choice is our recommendation based upon our computerized analysis of your specific situation."

Example: Target. Finally, a great real-life example for the power of machine learning-driven personalization is what the major retailer – Target has been doing for years. Target is famous in the retail industry for having employed statisticians and data scientists to use purchase behavior to identify shoppers who were pregnant and then market to them. Presumably, those statisticians and data scientists used data from Target's baby registry system to identify pregnancy-driven buying patterns. These patterns were then used to write algorithms that could identify pregnant shoppers and offer discounts or coupons that were likely to make that shopper more loyal to Target.

In contrast, a machine learning system would have taken a different approach. Rather than explicitly be designed to identify a pregnant shopper, it would have identified those patterns by itself based on common attributes and characteristics of the shoppers.

While there are certainly limitations, machine learning has the potential to optimize and automate common e-commerce systems, but it still has a long way to go before being able to handle all real-life limitations, such as insufficient or imbalance data.

1.1Problem Statement

Mass customization, the ability to create a highly-personalized experience for millions of individuals, is enabled by Machine Learning. ML applications have grown exponentially in accuracy, speed and applicability over the past five years. A major reason for this burgeoning interest in ML studies is the ever-growing volume and velocity of data captured at a significantly lower cost since the invention of cloud computing and the availability of cheaper computational power. We see applications of data classification and prediction techniques in virtually every aspect of our lives, including real-time areas such as shopper profiling, virtual

reality gaming, financial decision-making, fraud detection, medical prognosis, disaster control, pattern recognition, and national intelligence and security scenarios.

The core functionality necessary for developing these improved experiences is the ability to correctly classify and utilize copious amounts of data, in both real-time and offline settings in a cost-sensitive fashion. This data can originate from various sources and have different formatting, modality, consistency, frequency and reliability characteristics. Additionally, the velocity of data acquisition and increased performance demands are introducing new restrictions on the ML algorithms.

Generally speaking, the concept of classification includes decision-making in any context. However, the quality of the ML is limited by the quality of the training data. Consequently, the development of automated classification systems that replace handcrafted features with automatically selected appropriate feature sets for representation of big data is desirable and, if carefully engineered, will increase the accuracy of our predictions. Such systems solve some of the problems present in older studies such as the removal of subjectivity that comes from humans, and the increased computational capacity of the machines versus a manual approach.

When solving a set of classification problems to achieve optimal results we have to know the structure of the data which we will be processing. This is especially imperative in situations where the data set is not evenly distributed and a lot of data entries not belonging to the majority class obstruct the clarity of the results. One of the often overlooked but very significant problems in this field is the improper classification of imbalanced data [1]. Imbalanced data sets are a special classification problem where the class distribution is not uniform among the classes. Such data are considered imbalanced, even though it might be the most accurate representation of a real-life application because of the random nature of irregularly occurring events. Typically, they are composed of two classes: the majority (negative) class and the minority (positive) class [2]. From a learning perspective, imbalanced classes are of most interest when the number of outliers is small (which in turn makes classification exponentially more difficult). Conventional learning algorithms in machine

learning and data mining typically do not work well for imbalanced class problems since their objective is to minimize the overall error rate, which implicitly treats all misclassification costs equally. As a result, these algorithms may produce trivial results, typically classifying all test examples as negative. Additionally, it is often the case that the positive (minority) class is the one of greater interest. In such situations, a common mistake amongst classic classification techniques occurs while trying to minimize the percentage of erroneous class label predictions. In many case, there is an assumed equality of all miscellaneous errors. And yet, real-life practice has proven that some false classifications have far higher implications, hence higher cost, than other.

For example, a cancer patient normally treated with chemotherapy may nevertheless be in a 1-in-10,000 situation in which an autoimmune disease contraindicates that approach. Consider a high-traffic transit station, such as a big international airport when the algorithm predicts a false negative (e.g., traveler is believed not to carry a bomb when in fact he does). Even if the chance for that false negative is one in a million, the wrong labeling could lead to terrible consequences. Unfortunately, the standard learning approximation requires balanced training sets (i.e., positive and negative classes) because they are much easier to simulate and acquire. Therefore, only the majority of balanced data (referred to in the ML world as a negative class) is classified, while the imbalanced data (i.e., positive class) remains difficult to classify, and therefore largely ignored in scientific studies.

For this reason, the class imbalance problem has often been formulized within the cost-sensitive learning framework. The class imbalance problem gets more challenging in the context of learning from data streams, which unfortunately is required in many practical classification problems. For example, in e-commerce, customer dynamic features-based personalization data analysis [3], one challenge is that a customer's intent to convert (i.e., purchase) on a specific product is relatively rare. When comparing browsing activities (e.g., price/brand/quality comparisons or inspirational browsing), and the cost of missing a possible conversion is much higher than detecting. Meanwhile, in the online retail scenario, a customer's dynamic features data is usually collected incrementally over time, and the intent

detection system must respond to the signals in real time. Even though the entire data can be received from the start and the classifier can be trained in batch mode, it is still favorable if the model can adapt online to new data. Such data may come from the same subject but from a different time, or even from different subjects (e.g., similar customers). This is because the size of the entire data set can be too large to fit in memory or to train a classifier all at once. Moreover, since the positive examples (conversions) are relatively rare, each of them can convey very important information. However, in the batch learning scenario, the trained classifier is static and is not updated in the testing phase. Therefore, this problem should also be formulated within the incremental learning framework. The main challenge of such a problem is how to perform well even with very few positive examples at the early stage of model building, and perform better as more examples are available, motivating us to develop effective incremental learning algorithms dealing with class imbalance problem.

In this work, we are examining cost-sensitive anomaly detection classification algorithms that can be applied to in-house and third-party multimodal content to identify those that have been wrongly grouped and wrongly ranked. We also examine how they affect the unique customer experience by influencing the behavioral prediction models powering e-commerce personalization. With this solution, we are looking to achieve optimal balance between prediction accuracy, sensitivity and specificity for multiclass imbalanced datasets. For that reason, we are employing three meta-algorithmic approaches that combine several machine learning techniques into one predictive model to decrease the variance (bagging), decrease the bias (boosting) or improve the predictive force (stacking alias ensemble).

Every algorithm consists of two steps:

1. Producing a distribution of simple ML models on subsets of the original data.

2. Combining the distribution into one "aggregated" model.

We are going to use the following three meta-models for optimizing our data classification balance:

1. **Bagg**ing (**B**ootstrap **Agg**regation) is a way to decrease the variance of your prediction by generating additional data for training from your original dataset using combinations with repetitions to produce multisets of the same cardinality/size as your original data. By increasing the size of your training set you cannot improve the model's predictive force, but you can decrease the variance, narrowly tuning the prediction to expected outcome.

2. Boosting is a two-step approach, where one first uses subsets of the original data to produce a series of average performance models and then "boosts" their performance by combining them together using a cost function (such as majority vote). Unlike bagging, in classical boosting the subset creation is not random and depends upon the performance of the previous models: every new subset contains the elements that were (likely to be) misclassified by previous models.

3. Stacking is similar to boosting: you could also apply several models to your original data. The difference here is, however, that you do not have just an empirical formula for your weight function. Instead, you introduce a meta-level and use another model to estimate the input together with outputs of every model to estimate the weights or, in other words, to determine which models perform well and which model performs badly given these input data.

Our main contributions can be summarized as follows.

1. We demonstrate an online cost-sensitive ensemble learning framework, which generalizes a batch of widely used variance-decreasing bagging and bias- decreasing boosting-based cost-sensitive learning algorithms into its online version. With that we are enabling more accurate personalization experiences across the vast domain of e-commerce.

2. We analyze the consistency between the proposed algorithms and their batch mode counterparts, showing that under certain conditions, as the number of examples approaches infinity, the models generated by online cost-sensitive ensemble learning algorithms converge to that of batch cost-sensitive ensembles.

3. Three separate research areas (imbalanced data classification, cost-sensitive learning and incremental learning are bridged together in this paper to form an innovative, cost-effective, multimodal and multiclass classification technique that improves upon current state-of-the-art classifiers for imbalanced data. As meta-learning techniques, the proposed framework can convert any existing cost-insensitive online learning algorithm into cost-sensitive one.

4. With some straightforward modifications, the proposed algorithms can also deal with the concept drift problem in non-stationary environments.

The performance of the proposed real time cost-sensitive ensemble algorithms is evaluated on private, large, online retailer data sets, and a comprehensive comparison study of online and batch cost-sensitive ensemble learning algorithms is presented.

1.2 Handling imbalanced data

There are many recent studies on the topic of how to handle imbalanced data, and they range from data modifications to alterations of existing algorithms. On the other hand, many methods for learning from data streams are also available in the literature on incremental/online learning. Although these two issues have been extensively studied independently in the past decades, the research on simultaneously solving both problems is limited.

In this work, incremental learning is referred to as the general technique of learning from data streams, and online learning is characterized as responding immediately to a new instance and then discarding it. Except for SMOTE-based online learning algorithms, the positive examples are stored to create synthetic samples.

In addition, while imbalanced data has traditionally been studied as a two-class data classification problem (where there is one imbalanced class and one balanced class), in modern research a new understanding of the complexity of this problem has come into view. We now know that more often than not in real life applications, we have to deal with multiclass classification where more than a single class might be imbalanced in its own right. With that knowledge in mind, we have reviewed current state-of-the-art solutions and have proposed a multimodal and multiclass handling solution for the advancement of modern imbalanced data use cases within the application domain of e-commerce.

A) Data modifications

At the data level, there are several possible tactics to combat imbalanced data. Traditionally, the first natural approach is the collection of more data when possible. By expanding the data, more entries of the initially underrepresented type could appear, or even brand new data classes could be introduced, and to present a completely different modeling problem or a more even set with relatively similar class weights.

If access to more data is not possible, the next most common action is to change the performance metrics used to measure and showcase the classification quality. Typically, the standard metric used to determine the quality of a classifier is accuracy (number of correct classifications out of the examined set). But when we use only accuracy that could be very misleading with highly imbalanced classes because even with every entry classified as part of the majority class, the accuracy would come out as very high (100% minus the percentage of imbalanced entries). For a set of 1000 entries with 990 of them belonging to class A and 10 to class B, the accuracy of a standard classifier would be 99.9% which is deceivingly high and could lead to costly consequences. This problem is called an “accuracy paradox.” A simple

way to uncover the underlying data classes and not succumb for the accuracy paradox is to use confusion matrixes. A confusion matrix is the breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (what classes incorrect predictions were assigned).

	Predicted Positive	Predicted Negative
Actually Positive	True Positive	False Positive
Actually Negative	False Negative	True Negative

Table 1. Truth Table Confusion Matrix

Additionally, we can look at the *precision* (a measure of the classifier's exactness) and *recall* (a measure of the classifier's completeness) of the classification.

$$precision = \frac{true\ positives}{predicted\ positives} \quad \text{and} \quad recall = \frac{true\ positives}{actual\ positives}$$

A further step is to employ *Fmeasure* (Fscore) which is the weighted average of precision and recall. This metric is a good criterion if recall is regarded more important than precision.

$$Fmeasure = \frac{2 * precision * recall}{precision + recall}$$

These metrics come in handy in situations with highly skewed class balance as in the following example:

In a clinical study 286 women who are breast cancer survivors are examined. The data set used to identify their condition contains 9 attributes describing whether breast cancer

recurred within 5 years. Of the 286 women, 201 did not suffer a recurrence of breast cancer, whereas cancer did recur in the remaining 85.

We can use confusion matrixes to illustrate the possible naïve classification alternatives. Table 2 is indicative of a situation where we assume there were no recurrences of cancer, while table 3 shows the opposite situation – all 286 women were assumed to have recurring cancer within 5 years.

	Predicted Recurrence	Predicted No Recurrence
Actual Recurrence	0	0
Actual No Recurrence	85	201
	85	201

Table 2. "Assume No Recurrences" Confusion

	Predicted Recurrence	Predicted No Recurrence
Actual Recurrence	85	201
Actual No Recurrence	0	0
	85	201

Table 3. "Assume All Recurrences"

Finally, let's assume the existence of another more valuable classifier, classifier XYZ, which correctly predicts 10 recurrence events, as well as 188 no recurrence events.

	Predicted Recurrence	Predicted No Recurrence
Actual Recurrence	10	13
Actual No Recurrence	75	188
	85	201

Table 4. Classifier XYZ Confusion

Let

TP = true positives,

TN = true negatives,

PP = predicted positives, and

PN = predicted negatives.

Then **accuracy** is calculated as (TP + TN)/(PP + PN) and for all these use cases is:

- Assume No Recurrence model: accuracy = (0+201)/(85+201) = 0.703
- Assume All Recurrences model: accuracy = (85+0)/(85+201) = 0.297
- XYZ model: accuracy = (10+188)/(85+201) = 0.692

Precision is calculated as (TP/(TP + FN) = TP/PP) and results in:

- Assume No Recurrence model: precision = 0/(0+0) = 0
- Assume All Recurrences model: precision = 85/(85+201) = 0.30
- XYZ model: precision = 10/(10+13) = 0.43

Recall is calculated as (TN/(TN + FP) = TN / PN) and yields:

- Assume No Recurrence model: recall = 0/(0+85) = 0
- Assume All Recurrences model: recall = 85/(85+0) = 1
- XYZ model: recall = 10/(10+75) = 0.12

And the **Fmeasure** is calculated as (2*((precision*recall)/(precision + recall)) which results in:

- Assume No Recurrence model: Fmeasure = 2*((0*0)/0+0) = 0
- Assume All Recurrences model: Fmeasure = 2*((0.3*1)/0.3+1) = 0.46
- XYZ model: Fmeasure = 2*((0.43*12)/0.43+12) = 0.1

These results suggest that even though the Assume No Recurrence model has the highest accuracy, both the Assume All Recurrences and XYZ models are stronger prediction models, as they outperform the Assume No Recurrences model in every way.

Alternatively, another very popular measurement for imbalanced datasets are Receiver Operator Characteristics (ROC) and Precision-Recall (PR) curves. Like precision and recall, accuracy is divided into sensitivity and specificity and models can be chosen based on the balance thresholds of these values.

Both the ROC and PR curves visualize (see Figures 1, 2) the performance of the classifier over its entire operating range. The most widely used measure is the area under the curve (AUC). When dealing with highly skewed datasets there is a deep connection between ROC space and PR space because if a curve dominates in ROC space, it also dominates in PR space. Together they increase the confidence of prediction significantly.

The AUC can be used to compare the performance of two or more classifiers. Considering the AUC, we could make a comparison between a single selected threshold and the classifier's performance at that point or even the overall performance of one classifier versus another.

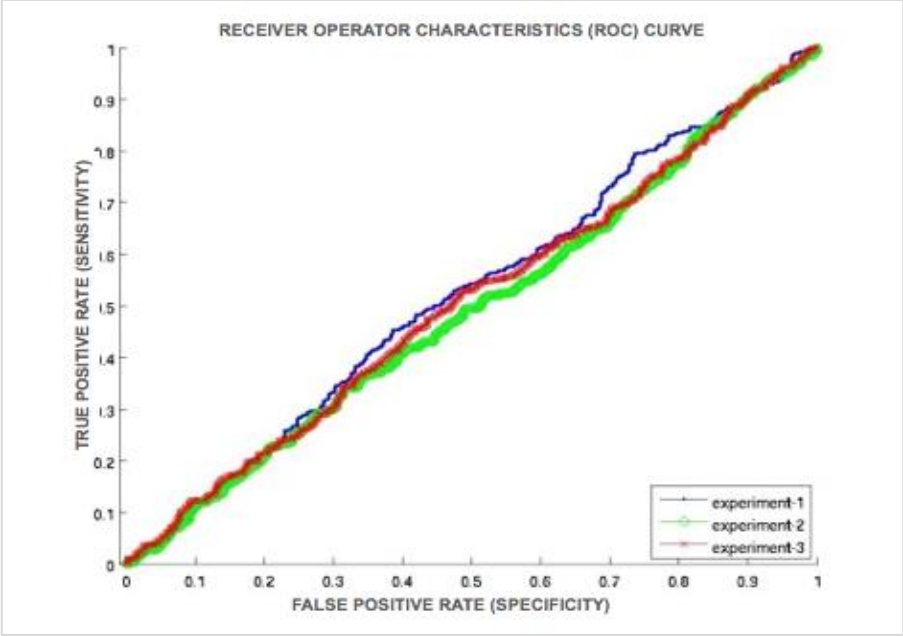


Fig 1. Example of ROC Curve Visualization

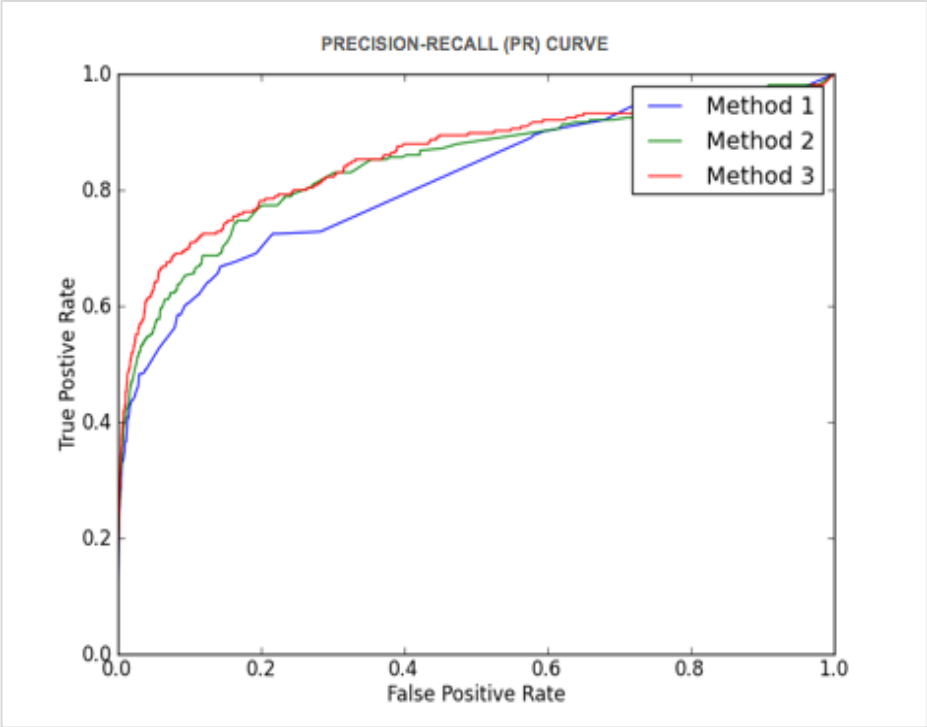


Fig 2. Example of Precision-Recall Curve Visualization

If the collection of more data, a change of measurements and a change of performance metrics still does not help, a more effective approach is to create a balanced set of classes by various forms of data re-sampling. You could add copies of instances from the under-represented class called over-sampling (or more formally, sampling with replacement), or you could delete instances from the over-represented class, called under-sampling. Some more comprehensive types of re-sampling include random with replacement, random under-sampling, directed over-sampling, directed under-sampling (both types of direct sampling are characterized by a non-random sample selection), over-sampling with informed generation of new samples, and combinations of the above techniques.

And finally, there are even more sophisticated data manipulation solutions via generation of synthetic samples. One factor about these approaches is that we could take into consideration misclassification costs when designing the algorithms such that the

misclassification cost of positive samples is higher than that of negative ones. A standard way to generate cost-sensitive synthetic samples is to analyze the original data set for common features of the data points with similar error costs and then to randomly sample the attributes from instances in the minority class. Such an approach can also be regarded as a class of meta-learning techniques which can convert cost-insensitive algorithms into cost-sensitive ones without modifying them. One advantage of this category of approaches is that it is independent of specific classifiers and therefore it is applicable to any existing cost-insensitive learning algorithm.

As part of this approach, there are systematic algorithms that could be used to generate synthetic samples. The most popular of such algorithms is called SMOTE (Synthetic Minority Oversampling Technique). It works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbs an instance of one attribute at a time by a random amount within the difference to the neighboring instances. This sampling technique can be further incorporated with ensemble learning algorithms [4] and has received much attention recently due to its better performance on imbalanced data sets [5].

Ensemble-based techniques for class imbalance problem inherit the good properties of ensemble learning algorithms, improving the generation ability of learning algorithms via bias or/and variance reduction, as well as achieving cost sensitivity by sampling techniques. In addition, traditional ensemble learning algorithms themselves have sampling step in each iteration. Therefore, little extra learning cost is added when embedding the re-sampling step to rebalance the data set. As for learning from data streams, which is of interest to us due to e-commerce big fast data (BFD), some algorithms are naturally incremental, or can be easily extended to incremental algorithms, including k-NN, naive Bayes classifier, binary linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), and so on.

B) Algorithm modifications

At the algorithmic level, solutions include adjusting the costs of the various classes to smooth out the class imbalance, adjusting the probabilistic estimate at the tree leaf (when working with decision trees), adjusting the decision threshold, and recognition-based (i.e., learning from one class) rather than discrimination-based (two class) learning.

One of the more interesting, highly sought after and useful algorithmic optimizations currently is responsible for the transition from manual selection of sets of features to an automatic generation via high-level abstractions with multiple processing layers, and it is called *deep learning*. Traditional data pipelines for e-commerce products use image processing feature descriptors (e.g. *SIFT* – Scale Invariant Feature Transform and *HoG* – Histogram of Oriented Gradients) dependent on manual selection of classification features to represent the characteristics of the objects. Even though they have demonstrated their representational power on a variety of visual search tasks, they are still unable to carry high-level concepts of objects. To address this problem, in recent years researchers proposed a neural network based deep learning architecture named Convolutional Neural Networks (CNNs).

In addition, the incremental/online versions of more sophisticated algorithms have been proposed in the literature, including but not limited to decision trees [6], random forests [7], [8] , multi-class LDA [9], [10], logistic regression [11], support vector machines [12], [13], and other kernel methods [14], [15]. Besides the base learning algorithms, the online versions of ensemble learning techniques, bagging and boosting, were also derived in [16] by approximating binominal distribution using a Poisson distribution.

In this work, we are employing the use of deep learning techniques, such as deep neural networks (DNN), convolutional neural networks (CNN), deep belief networks (DBN) and more, to enhance the performance of our innovative e-commerce-specific over-sampling technique that we call Over-sampling via Arbitrary Features Allocation (OAFA) and we describe in chapter IV – Approach and E-Commerce-Specific Challenges. The combination of our unique highly customized re-sampling algorithm with visual processing deep hierarchical

learning techniques creates a powerhouse methodology for accurate prediction of imbalanced multimodal e-commerce data.

Chapter 2

2. Literature Review

This section presents a comprehensive review of renowned and modern methods in the field of imbalanced data discovery and classification which could be applied as solutions within the world of e-commerce for optimized information categorization. We are later going to use these innovative approaches as a base for performance comparison to our customizable methodology in terms of prediction accuracy, sensitivity and specificity within the application domain of online business.

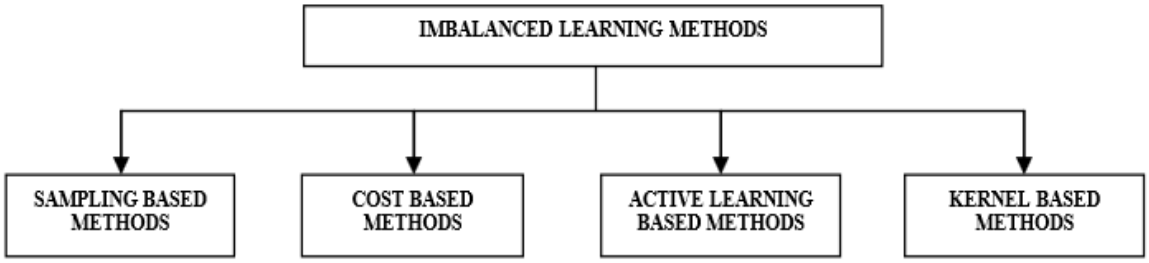


Fig. 3. Methods in Imbalanced Learning

2.1 Sampling-Based Methods

Significant work has been done to handle the imbalanced learning problem, and it can be categorized into four categories: sampling-based methods (this section), cost-based methods (section 2.2), active learning-based methods (section 2.3) and kernel-based methods (section 2.4). Figure 3 shows these categories. All these methods attempt to resolve the imbalanced learning problem with various levels of efficiency based on the application domain within which they are utilized.

In imbalanced learning sampling methods, the size of the classes is altered, i.e. it may increase the number of samples or it may reduce the samples. The method in which the number of samples get reduced is called –under-sampling, while the method which increases the number of samples is called over-sampling [17]. Figure 4 represents a hierarchy of sampling methods.

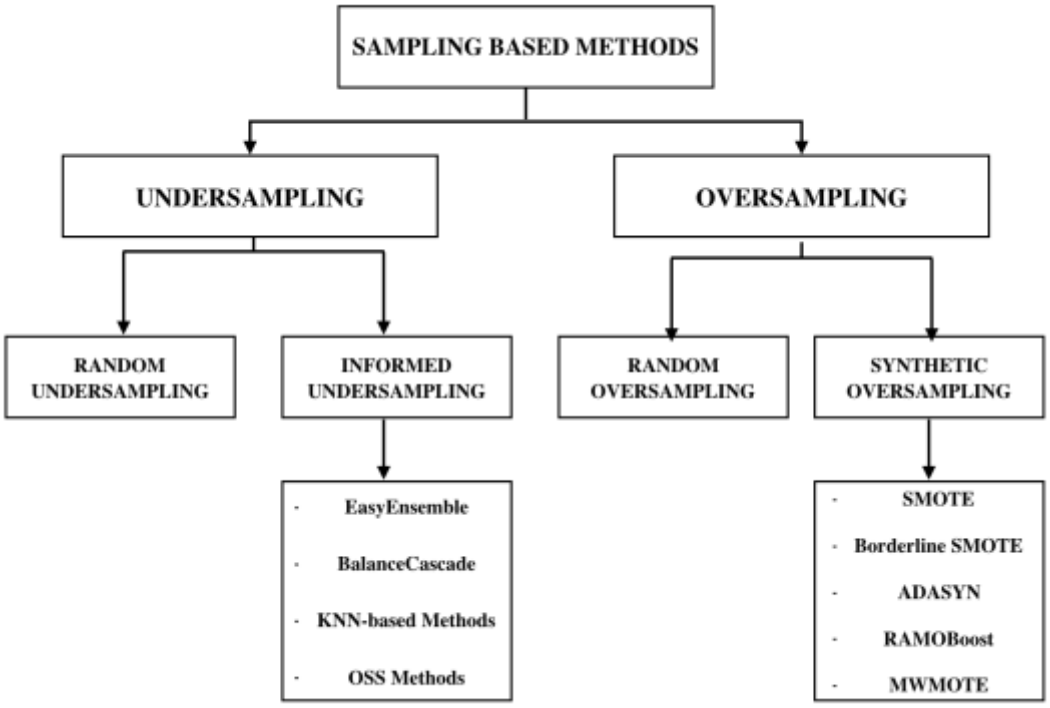


Fig 4. Taxonomy for Sampling-Based Methods

A) Under-sampling

Under-sampling methods reduces the number of samples from the majority class in order to create a balance between the majority and minority classes. When majority class samples are reduced randomly it is called random under-sampling. When samples are reduced on the basis of some statistical knowledge, then it is called informed under-sampling. The main drawback of random under-sampling is that it may miss some important concepts from the

majority class as it randomly removes the samples [18]. To overcome this problem many researchers propose various informed under sampling techniques as follows:

- EasyEnsemble

In the EasyEnsemble method, a majority class is divided into several subsets and the size of each subset is equal to the size of a minority class. Then for each subset, it develops a classifier using the whole minority class and the majority class subset. Results generated from all the classifiers are combined to get the final decision. To develop a classifier AdaBoost (a meta-algorithm for algorithmic performance optimizations) is used. The EasyEnsemble approach is shown in Figure 5. Because EasyEnsemble uses independent random sampling with replacement, it can be considered as an unsupervised learning algorithm [19].

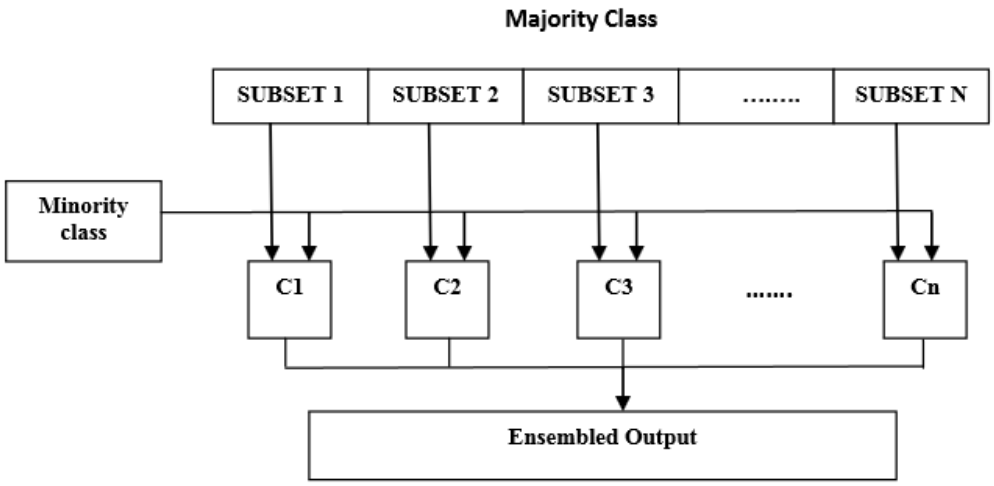


Fig 5. EasyEnsemble Approach

- BalanceCascade

This method follows the supervised learning approach [20]. The BalanceCascade method works by forming a subset of majority class which contains a number of samples equal to the number of minority class sample. When classifier C1 is trained using the majority class subset and the whole minority class, the samples from a majority subset which are correctly classified are removed. This newly generated sampled set of majority class is given as an input to classifier C2. The same procedure is iterated until a final classifier is reached. At every

classifier, the size of the majority subset gets reduced. In BalanceCascade there is a sequential dependency between classifiers. BalanceCascade differs from EasyEnsemble as it removes true majority samples in order to reduce redundancy.

- k-NN-based methods

To achieve under-sampling, a k-NN based approach has been invented that deals with imbalance data distribution. This k-NN based approach includes four different methods, namely NearMiss-1, NearMiss-2, NearMiss-3 and most distant method [21] In the NearMiss-1 method, majority samples whose average distances to three closest minority samples are the smallest are selected for the under-sampling. The NearMiss-2 method selects majority samples that are close to all minority samples. This method selects the samples based on their average distances to three farthest minority samples. The NearMiss-3 method guarantees every minority sample is surrounded by some majority samples. This method selects a given number of closest majority samples for each minority sample. In the Most-Distant method, majority samples whose average distances to three closest minority samples are the farthest are selected for the under-sampling. On the basis of experimental results researchers have suggested that NearMiss-2 methods performed well as compared to other methods.

- One-sided Selection method

This is another type of informed under-sampling method in which only the most representative majority samples are kept and the remaining samples are removed from the class [22] In order to choose the most representative samples, OSS first chooses one sample x randomly from the majority class. Then taking minority samples and x as a training set OSS uses the k-NN algorithm for classification of the remaining samples of majority class. After that, the correctly classified samples are removed from the majority class in order to eliminate redundancy. Therefore, the majority class will contain only incorrectly classified

samples and x. At the end, OSS removes borderline and noisy samples using data cleaning techniques.

To start the under-sampling process OSS selects only one majority sample randomly. Hence, the overall result will be dependent on that sample. Also, OSS does not consider the existence of sub-concepts within a majority class. To overcome this problem a novel method has been proposed called ClusterOSS. In this method, using clustering methods like k-means, clusters of majority classes are formed. Then samples located near the center of the cluster are selected for an under-sampling process.

There are two main differences between OSS and ClusterOSS. OSS uses only one majority sample to start under-sampling process, while ClusterOSS uses multiple samples. Another difference is that OSS starts under-sampling randomly, while in a ClusterOSS the number and instances of samples is decided in advance [23]. Experimental results suggest that the combination of ClusterOSS with random under-sampling will give better results.

B) Over-sampling

Over-sampling methods add the samples to the original imbalanced data set. There are two types of over-sampling: random over-sampling and synthetic over-sampling. In random over-sampling minority samples are randomly replicated, but this may lead to an over-fitting problem. In the synthetic over-sampling method, synthetic samples are generated from minority samples. There are various over-sampling methods described in the literature. They are as follows:

- SMOTE

N.V. Chawla et al. proposed a powerful method called Synthetic Minority Over-sampling Technique (SMOTE) which has shown great success in many applications [24]. Initially for each minority sample k-nearest neighbors are determined. Then a synthetic sample is

generated along the line segment joining minority sample and its nearest neighbor. Firstly, SMOTE takes the difference between minority sample and its nearest neighbor. This difference is then multiplied by a random number between 0 and 1, and added to original minority sample. In this way a synthetic sample is generated. SMOTE generates an equal number of synthetic samples for each minority sample.

To handle the imbalanced learning problem in big data a novel approach (called the Enhanced SMOTE algorithm) has been proposed. This algorithm works in two steps. In the first step, original data set is decomposed into subsets of binary classes using Binarization techniques such as OVA (over-versus-all) and OVO (one-versus-one). Then for each binary class SMOTE is applied. Random Forest is used to classify the data [25].

- Borderline-SMOTE

As SMOTE generates synthetic samples for each minority sample it may lead to over generalization [26]. The main objective of Borderline-SMOTE is to identify minority samples located near the decision boundary. Then these samples are used further for over-sampling. This method focuses on borderline samples because classifier may misclassify them. For this use case, H. Han et al. proposed two methods - borderline-SMOTE1 and borderline-SMOTE2 [27]. Both methods give better results on TP rate and F-value as compared to classic SMOTE.

- ADASYN

Haibo He and E.A. Garcia proposed a novel approach Adaptive Synthetic Sampling (ADASYN) to handle imbalanced data sets. In synthetic sample generation process, there is no need to consider all minority samples as there may be problem of overlapping [28] ADASYN uses the weighted distribution of minority samples. It assigns weight to a minority sample depending on the importance of the minority sample. Samples which are difficult to classify get higher weight than others. More samples are generated for the sample having a higher weight. ADASYN can be integrated with an ensemble based learning algorithm to get optimized results.

- RAMOBoost

Ranked Minority Over-sampling in Boosting (RAMOBoost) is a technique which systematically generates synthetic samples depending on sampling weights. It adjusts these weights of minority samples according to their distribution. This method works in two stages. In the first stage the decision boundary is shifted towards the samples which are difficult to learn from both majority and minority classes. In the second stage a ranked sampling probability distribution is used to generate synthetic samples. If RAMOBoost adopts techniques used in SMOTE-N method, then it can handle datasets having nominal features [29]

- MWMOTE

Existing synthetic over-sampling methods may have some insufficiencies and inappropriateness in many scenarios [30] In order to overcome these problems, a new method has been proposed called Majority Weighted Minority Oversampling Technique (MWMOTE). This method works in three steps. In the first step, the samples from the minority class which are difficult to learn and which contain more information are selected. In the second step, a selection weight is assigned to those selected samples. Most important samples get higher weight. In the last step, using selection weights this algorithm generates synthetic samples from selected minority samples. Many over-sampling methods use k-NN-based approach for sample generation process, but MWMOTE uses a clustering approach which gives better results than previous approaches. MWMOTE attempts to improve the sample selection and sample generation process very efficiently. In future MWMOTE can be extended for multiclass imbalances.

2.2 Cost-Based Methods

As sampling-based methods try to remove or add samples to balance between majority and minority classes, cost-based methods use a cost matrix in dealing with imbalanced

learning. This type of algorithm directly modifies the traditional algorithms to achieve cost sensitivity by taking different misclassification costs into consideration when designing the algorithms such that the misclassification cost of positive examples is higher than that of negative ones.

Cost matrix represents the cost associated with each misclassification. If any minority sample gets misclassified in a majority class, then its cost will be higher than misclassification of a majority class sample [31]

The goal of a classifier is to minimize the cost instead of classification error, and therefore the classification algorithms will bias towards the small class. For example, cost-sensitive SVM can be derived by kernel modification [32], biased penalty [33], or loss function modification [34]. For decision tree, the cost sensitivity can be introduced by probabilistic estimate calibration [35] or using different pruning methods [36] When dealing with decision trees, cost-sensitive methods can move the decision threshold, can apply pruning schemes based on cost-sensitivity or can consider cost-sensitivity in the spilt criterion [37]. For building cost-sensitive decision trees with missing values, a new splitting criterion has been proposed [38] which is based on tradeoffs between different cost units and the classification ability.

To introduce cost-sensitivity in neural networks, the output of the neural network must be cost-sensitive. The error minimization function should be adapted to get the expected cost. Based on cost-sensitivity some modifications should be applied to probabilistic estimate [39] To handle the multiclass imbalance problem, a new method based on an ensemble of cost-sensitive neural network has been proposed [40]. This new method optimizes the misclassification cost using evolutionary search technique.

2.3 Active Learning-Based Methods

In semi-supervised learning, there can be pool of data with labeled, as well as unlabeled, samples. To label the samples manually could be very expensive. To improve the classification

accuracy, active learning methods focus on acquiring labels for those unlabeled data samples. The active learner chooses the unlabeled samples which are closer to a decision boundary and with highest uncertainty. In the traditional approach, there exists a human annotator (oracle) who gives labels to unlabeled samples when the learner queries for labels. There are three main categories of this traditional approach: 1) pool-based active learning [41], stream-based active learning [42] and query construction based active learning [43] Another strategy is proposed for remote sensing image classification [44] in which a classifier assigns a rank to each unlabeled sample, and samples that are more important are selected. These selected samples are then labeled manually. A novel method is proposed to deal with noisy labels [45] There are two procedures used in this method, label integration and sample selection. In a label integration process, to get labels from multiple noisy labels a positive label threshold algorithm (PLAT) is used. Using the sample selection strategies, learning performance of PLAT can be improved.

2.4 Kernel-Based Methods

Along with sampling based methods and cost-sensitive methods, many researchers have worked on kernel based methods in order to deal with imbalanced data sets. In *"A Kernel-based Sampling to Train SVM with Imbalanced Data Set"* [46] a new over-sampling strategy based on kernel function has been proposed to train a support vector machine (SVM). First, it generates a synthetic sample from minority samples similar to SMOTE, then a pre-image of each synthetic sample is identified, and all these pre-images collectively append to the original minority set. To overcome the problem of generalization due to various over-sampling methods a novel approach of quasi-linear SVM and assembled SMOTE has been proposed [47] In this approach, using a minimum spanning tree, data is divided into a number of local linear partitions so that they are linearly separable. Then synthetic samples are

generated using assembled SMOTE. Finally, using a quasi-linear kernel function SVM classifies data efficiently.

2.5 Multiclass imbalance problem

In this work, we are focusing on developing a real-life data classification when heavily imbalanced data sets are in use. For a successful solution of this problem we first need to identify the types of data that we will be analyzing as defined by e-commerce constrains. Advancements in wireless communication techniques and the popularity of mobile devices (e.g., mobile phone, tablets, smart watches) contribute to a new aspect of online shopping: mobile commerce. This is a form of e-commerce done via native mobile apps and mobile browsers. While in 2014 mobile commerce shopping was still lagging traditional desktop shopping, in 2015 began the trend of mobile-first, and in 2016 for many large retailers, mobile traffic became their predominant channel and helped in acquiring their biggest number of new customers [48] Consequently, in our work we investigate and design technologies which can handle the different aspects of data clustering and categorization based on its origin (e.g., mobile commerce or traditional desktop commerce). One of the specifics of mobile commerce is the superior accuracy of localization data from GPS (e.g., within 15 feet) versus localization data from IP address (within 25 miles). As seen in our previous research of medical crowdsourcing-powered mobile applications, streaming data coming from varying locations and networks has to be clustered differently due to discrepancies in format [49]. Although this situation is often initially perceived as a two-class imbalance data classification (where there is one imbalanced class and one balanced class), it is actually a multiclass one.

A number of solutions have been proposed at the data and algorithm level to deal with class imbalance, the efforts of most major classification techniques so far are focused on two-class imbalance problems in the literature. Most existing imbalance learning techniques are only designed for and tested in two-class scenarios. They have been shown to be less effective or even cause a negative effect in dealing with multiclass tasks. Some methods are not directly applicable. Among limited solutions for multiclass imbalance problems, most

attention in the literature was devoted to class decomposition—converting a multiclass problem into a set of two-class sub-problems. Given a *c*-class problem (*c > 2*), a common decomposing scheme is to choose one class labeled as positive and to merge the others labeled as negative for forming a sub-problem. Each class becomes the positive class once, and thus, *c binary* classifiers are produced to give a final decision (known as one-against-all (OAA) or one-versus-others (OVO). However, it aggravates imbalanced distributions, and combining results from classifiers learned from different sub-problems can cause potential classification errors. It is desirable to develop a more effective and efficient method to handle multiclass imbalance problems.

Most existing solutions for multiclass imbalance problems use class decomposition schemes to handle multiclass and work with two-class imbalance techniques to handle each imbalanced binary subtask. For example, protein fold classification is a typical multiclass imbalance problem. Tan et al. used both OAA and OAO schemes to break down this problem and then built rule-based learners to improve the coverage of minority class examples. OAA and OAO are the two most popular schemes of class decomposition in the literature. Zhao et al. [50] used OAA to handle multiclass and under-sampling and SMOTE techniques to overcome the imbalance issue. Liao [51] investigated a variety of over-sampling and under-sampling techniques used with OAA for a geo tagging flaw classification problem. Chen et al. [52] proposed an algorithm using OAA to deal with multiclass and then applied some advanced sampling methods that decompose each binary problem further so as to rebalance the data. Fernandez [53] integrated OAO and SMOTE in their algorithm. Instead of applying data level methods, the algorithm developed by Alejo et al. [54] made the error function of neural networks cost-sensitive by incorporating the imbalance rates between classes to emphasize minority classes, after decomposing the problem through OAA. Generally speaking, class decomposition simplifies the problem. However, each individual classifier is trained without full data knowledge. It can cause classification ambiguity or uncovered data regions with respect to each type of decomposition.

Chapter 3

3. Review of Bagging and Boosting Techniques

In this chapter, we briefly review the standard bagging and boosting algorithms, as well as their online versions and cost-sensitive versions, which motivate the proposed online cost-sensitive ensemble framework.

3.1 Standard bagging and boosting

In recent years, ensemble learning algorithms have been the topic of much theoretical and experimental research. These algorithms provide methods for invoking a learning algorithm (the base learning algorithm) multiple times and for combining the resulting

hypotheses into an ensemble hypothesis (e.g., via a majority vote). Ensemble learning algorithms work by combining the outputs of multiple base learners. The basic idea here is to improve the generalization ability of individual classifiers by training them on different data sets, which is motivated by bias-variance trade-off [55]. The goal in using an ensemble of hypotheses is to be superior in some sense to the individual hypothesis generated by the base algorithm on the training instances. Averaging the outputs of base models tends to cancel the variance component or/and reduce the bias. On the other hand, to achieve good performance, the diversity, which is usually introduced by presenting different training data to different base models among classifiers, is an important characteristic [56]. Different approaches to averaging base models and obtaining desired diversity distinguish different ensemble methods. Two representative techniques among them are Bagging and Boosting (AdaBoost) [57].

Given a data set $S = \{(x1, y1), \ldots, (xN, yN)\}$ of size N, where $x_n \in X$, $y_n \in Y = \{0, 1\}$, M base models h_m, bagging constructs M classifiers with bootstrap replicas $\{Sm\}$ of S, where Sm is obtained by drawing examples from the original data set S with replacement, usually having the same number of examples as S. The diversity among the classifiers is introduced by independently constructing different subsets of the original data set. After constructing ensembles, the prediction of the class of a new example is given by majority voting. The pseudo-code of Bagging is shown in Alg. 1.

Algorithm 1 Bagging Algorithm

Input: S, M

1: **for** $m = 1, \ldots, M$ **do**
2: $S_m = Sample\_with\_replacement(S, N)$
3: Train a base learner $h_m \rightarrow Y$ using S_m
4: **end for**

Output: $H(x) = \arg\max_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$

AdaBoost is another widely-used ensemble learning algorithm. Unlike bagging, which treats all examples equally, AdaBoost focuses more on difficult examples. In particular, Adaboost sequentially constructs a series of base learners in such a way that examples that are misclassified by the current base learner h_m are given more weight in the training set for the following learner h_{m+1}, whereas the correctly classified examples are given less weight. More specifically, the weights of all examples are initially equal, and then examples misclassified by h_m are given half the total weight for the following learner h_{m+1}, and the correctly classified examples are given the remaining half of the total weights. The pseudo-code of AdaBoost is shown in Alg. 2.

It should be emphasized that by using the update rule in step 6 of Alg. 2, the normalization step is avoided. That is, the summation of D_{m+1} will remain one after each update without normalization, which is crucial to designing the online boosting algorithm [58] and the proposed online cost-sensitive boosting algorithms, since the normalization factor is unavailable during the online learning process. After update, the reweighted examples can be either directly used to train the next base learner, or first resampled according to the weights and then the unweighted samples are used to train the base learner. In this work, all boosting techniques are implemented by resampling. The reasons are three-fold: First, sampling-based boosting algorithms are consistent with bagging techniques. Second, they are also consistent with their online counterparts introduced later, which simulate sampling with replacement by using a Poisson distribution. Finally, reweighting on the training set is not applicable to all learning algorithms.

Algorithm 2 Boosting (AdaBoost) Algorithm

Input: S, M

1: Initialize $D_1(n) = \frac{1}{N}$ for all $n \in \{1, \ldots, N\}$
2: **for** $m = 1, \ldots, M$ **do**
3: Train a base learner $h_m \rightarrow Y$ using S with distribution D_m
4: $\epsilon_m = \sum_{n=1}^{N} D_m(n) I(h_m(x_n) \neq y_n)$
5: **for** $n = 1, \ldots, N$ **do**
6: $D_{m+1}(n) = D_m(n) \times \begin{cases} \frac{1}{2(1-\epsilon_m)}, & h_m(x_n) = y_n \\ \frac{1}{2\epsilon_m}, & h_m(x_n) \neq y_n \end{cases}$
7: **end for**
8: **end for**

Output: $H(x) = \arg\max_{y \in Y} \sum_{m=1}^{M} log(\frac{1-\epsilon_m}{\epsilon_m}) I(h_m(x) = y)$

3.2 Online Bagging and Boosting

The framework of online ensemble learning algorithms [59] is inspired by the observation that the binomial distribution Binomial(p,N) can be approximated by a Poisson distribution Poisson(λ) with λ = Np as N → ∞, where the probability of success p in the binomial distribution is equivalent to D(n) in bagging and boosting algorithms. For example, since D(n) = $\frac{1}{N}$ for all examples of bagging algorithms, the uniform sampling with replacement of bagging algorithms can be approximated by Poisson (1). For online boosting, λ can be computed by tracking the total weights of correctly classified and misclassified examples for each base learner (λ_m^{SC}, λ_m^{SW}). The online bagging and boosting algorithms are described in Alg. 3 and Alg. 4 respectively.

Algorithm 3 Online Bagging Algorithm

Input: S, M

for $n = 1, \ldots, N$ **do**
  for $m = 1, \ldots, M$ **do**
    Let $k \sim Poisson(1)$
    Do k times
      Train the base learner $h_m \rightarrow Y$ using (x_n, y_n)
  end for
end for

Output: $H(x) = \arg\max_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$

Algorithm 4 Online Boosting (AdaBoost) Algorithm

Input: S, M

Initialize $\lambda_m^{SC} = 0, \lambda_m^{SW} = 0$ for all $m \in \{1, \ldots, M\}$
for $n = 1, \ldots, N$ **do**
  Set $\lambda = 1$
  for $m = 1, \ldots, M$ **do**
    Let $k \sim Poisson(\lambda)$
    Do k times
      Train the base learner $h_m \rightarrow Y$ using (x_n, y_n)
    if $h_m(x_n) = y_n$ **then**
      $\lambda_m^{SC} \leftarrow \lambda_m^{SC} + \lambda, \epsilon_m \leftarrow \frac{\lambda_m^{SW}}{\lambda_m^{SC} + \lambda_m^{SW}}, \lambda \leftarrow \frac{\lambda}{2(1-\epsilon_m)}$
    else
      $\lambda_m^{SW} \leftarrow \lambda_m^{SW} + \lambda, \epsilon_m \leftarrow \frac{\lambda_m^{SW}}{\lambda_m^{SC} + \lambda_m^{SW}}, \lambda \leftarrow \frac{\lambda}{2\epsilon_m}$
    end if
  end for
end for

Output: $H(x) = \arg\max_{y \in Y} \sum_{m=1}^{M} log(\frac{1-\epsilon_m}{\epsilon_m}) I(h_m(x) = y)$

3.3 Cost-sensitive Bagging and Boosting

Cost-sensitive ensemble learning is a meta-technology that takes different misclassification costs into consideration via biased resampling/re-weighting methods before each iteration of bagging and boosting. As a result, cost-sensitive ensemble learning

algorithms can turn any cost-insensitive classifier into a cost-sensitive one with little extra learning cost, while preserving the good properties of standard ensemble learning algorithms. The main difference between these techniques lies in the choice of resampling mechanism. From this point, we briefly review the two most popular cost-sensitive ensemble learning algorithms, namely UnderOverBagging and SMOTEBagging [60]. One of the contributions of this work is the derivation of their online extensions from a purely static format. Given an imbalanced data set of N^+ minority class S^+ and N^- majority class S^-, one straightforward approach to implement bagging-based ensemble learning algorithms is to under-sample the majority class or over-sample the minority class, which gives UnderBagging and OverBagging respectively. UnderOverBagging is a uniform approach combining both UnderBagging and OverBagging. In addition, the resampling rate (*a%*) can be also varied over the bagging iterations, which further boosts the diversity among the base learners. Alg. 5 shows the pseudo-code of UnderOverBagging, from which it can be observed that the sampling method is gradually switched from under-sampling the majority class to over-sampling the minority class. The number of training examples for the first base learner is lower than the final one.

Algorithm 5 Batch UnderOverBagging Algorithm

Input: $S, M, C > 1$

1: **for** $m = 1, \ldots, M$ **do**
2: $a = \frac{100m}{M}$
3: $S^-_{train} = Sample\_with\_replacement(S^-, N^- a\%)$
4: $S^+_{train} = Sample\_with\_replacement(S^+, CN^+ a\%)$
5: $S_m = S^+_{train} + S^-_{train}$
6: Train a base learner $h_m \rightarrow Y$ using S_m
7: **end for**

Output: $H(x) = \arg\max_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$

In SMOTEBagging [61] (Alg. 6), the negative class is sampled with replacement at rate *100%* (i.e., *N* negative examples are generated).

Algorithm 6 Batch SMOTEBagging Algorithm

Input: $S, M, k, C > 1$

1: **for** $m = 1, \ldots, M$ **do**
2: $a = \frac{100m}{M}$
3: $S^-_{train} = Sample\_with\_replacement(S^-, N^-)$
4: $S^+_{train} = Sample\_with\_replacement(S^+, CN^+a\%)$
5: $S^+_S = SMOTE(S^+, C(1-a\%), k)$
6: $S_m = S^+_{train} + S^-_{train} + S^+_S$
7: Train a base learner $h_m \rightarrow Y$ using S_m
8: **end for**

Output: $H(x) = \arg\max_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$

At the same time, CN^+ positive examples are generated for each base learner, among which *a%* of them are created by resampling and the rest of the examples are created by the synthetic minority oversampling technique (SMOTE) shown in Alg. 7. The main idea of SMOTE is to generate more new synthetic examples by interpolating the positive examples. As a result, all of the base learners are trained on a more balanced and diverse data set.

Algorithm 7 Batch SMOTE Algorithm

Input: S, T, k

1: **for** $n = 1, \ldots, N$ **do**
2: **for** $i = 1, \ldots, T$ **do**
3: Randomly choose one of the k nearest neighbors of x_n, x'_n
4: Calculate the difference between x_n and x'_n
5: Generate a random number γ between 0 and 1
6: Create a synthetic instance:
$x^i_{S,n} = x_n + \gamma(x'_n - x_n)$
7: **end for**
8: **end for**

Output: TN synthetic instances $\{x^1_{S,1}, \ldots, x^T_{S,N}\}$

The diversity is further boosted by varying *a%* so that the ratios of bootstrap replicates and synthetic examples generated by SMOTE varies over the bagging iterations.

Chapter 4

4. Approach and e-Commerce-Specific Challenges

4.1Design Solution

As part of our investigation of reviewing current best practices and cutting edge methodologies, we concluded that for our e-commerce solution we need to find an innovative way of handling multi-modal imbalanced data in order to cover the complex domain of e-commerce. In addition, the imbalance problem becomes even more challenging in the context of learning from data streams, rather than from a static data set. In many

practical situations, we do not have the luxury of having a fully available data set in the beginning of the training and learning period for an algorithm, so we needed a solution which can be adaptable to the amount of data available at any given time. Consequently, we focused on designing an adaptable cost-sensitive ensemble classifier which can generalize a number of batch cost-sensitive algorithms applicable to multimodal data to their online streaming versions. Our solution needed to be implementable, reliable and accurate, and can be extended to other application domains with minor modifications. Due to the low implementation cost and high opportunity for result visualization and applicability within multiclass classification, we designed an ensemble classification algorithm using a custom over-sampling technique together with *K-means* and combined it with random under-sampling to offset any overfitting.

4.2 K-OAFA

It is known that SMOTE has two definitive weaknesses. First, the algorithm treats all insertion locations in the same way. Second, it blurs the boundaries of the majority and the minority classes. In order to overcome these downfalls, we developed an algorithm that inserts data items in the regional distribution and does not insert data items at the boundaries.

Our new over-sampling algorithm consists of three steps during which we combine synthetically generated minority classes based on feature-selection over-sampling and *K-means* supplemented over-sampling. We call this method K-means Over-sampling via Arbitrary Features Allocation (K-OAFA).

In the first step, which we call Over-sampling via Arbitrary Features Allocation (OAFA), we generate artificial instances for minority classes by allocating arbitrarily chosen features of the existing minority class to a new minority class.

In the second step, we return to the original data set to apply the K-means algorithm together with SMOTE. This is done to prevent interpolation generalization by first performing the clustering operation before the interpolation, and then performing the interpolation in

the clustering region. Consequently, the interpolation data is on the connection between the cluster core and the original data point. In this step, for the minority class, we first use the K-means algorithm for the clustering operation. After the cluster operation, the fixed K clusters are formed and the core of each cluster is recorded. The interpolation operation is performed for each cluster sample. The original sample point is then interpolated by the cluster center, which is used as the original sample point. The specific steps we followed are:

Step 1. Find the center of the minority class samples.

Step 2. Create a new minority class. The new synthetic samples are created as

$$p_j = x + rand\ (0,1) * (X_c - x), j = 1,2 \dots N,$$

where *rand (0, 1)* represents a random number in the interval (0, 1), p_j represents new synthetic samples, x represents the minority class, and X_c represents the center of the minority class.

Step 3. Replace the minority class of the original dataset with the new minority class.

In the *third step*, we combine the minority classes generated by OAFA and SMOTE *K-means* to maintain the arbitrary nature of the minority class features. Then the new dataset is put into the original dataset to get the final sample. Consequently, our solution does not require the distance metric on the feature space, which is hard to define when it is a mixture of numerical and categorical variables. Additionally, using classic visualization techniques on the dataset, all the imbalanced data points with similar features reveal themselves by overlapping on the image map. Therefore, this combination method increases the

performance of a classification rule especially when the features are independently affecting the response.

4.3 Imbalance e-commerce Ensemble Algorithm (ICE-A)

Ensemble classification learning is a machine learning technique. It uses a simple classification algorithm to get a number of different base classifiers that are combined in some way to receive a strong classifier. Through our research experiments we have discovered that in the field of e-commerce an effective way of achieving data class equilibrium in multiclass classifications is the addition of under-sampling. Therefore, after applying K-OAFA, we also carried out random under-sampling to form multiple weak classifiers. This method does not require additional memory for intermediate results storage. Finally, the multiple weak classifiers were integrated to form the final strong classifier. The algorithm description of Ensemble K-OAFA and Random Under-sampling is shown in Table 4 and consists of 5 distinct steps. We call this algorithm Imbalance e-Commerce Ensemble Algorithm (ICE-A).

Compared with existing imbalanced data classification methods, our ICE-A method uses the K-OAFA over-sampling technique to increase the size of the minority class and adjust the balance degree of the imbalanced dataset, to balance the data distribution. Under the condition of keeping the distribution of the whole dataset, the under-sampling is used to reduce the training data and reduce the size of the dataset, to reduce the training time of the model and improve the classification efficiency of the algorithm. At the same time, our algorithm trains the weak classifier in each iteration process and uses the ensemble learning method boosting technology to combine the classifier. According to the classification results, the samples are given new weights to generate multiple weak classifiers, and the final output results are then obtained by the weight of the weak classifier. Therefore, the algorithm could improve the classification efficiency and increase the classification accuracy of the minority class.

Input:

Dataset: $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$;

Base classifier: C;

Over-sampling rate: M;

Under-sampling rate: N;

Output: Strong classifier: $F(x)$;

Process：

Step 1: The weights of the samples were initialized: $W(i) = 1/n$

Step 2: The minority class P (Positive class) was sampled by K-OAFA to form a balanced data at rate M.

Step 3: The whole dataset was under-sampled randomly at rate N under the condition of keeping the data distribution. The dataset D′ was formed and its weight distribution was W.

Step 4: for k = 1 to K

1) Train weak classifier according to the training dataset D′ and its weight distribution W′, and calculate the weak hypothesis $h_t: X \times Y \rightarrow [0, 1]$

2) Calculate the pseudo-loss of h_t:

$$\varepsilon_k = \sum_{(i,y):y_i \neq y} D_k(i)(1 - f_k(x_i, y_i) + f_k(x_i, y))$$

3) Calculate the weight update parameters:

$$\beta_k = \frac{\varepsilon_k}{1 - \varepsilon_k};$$

$$\omega_k = \frac{1}{2} \cdot (1 - f_k(x_i, y) + f_k(x_i, y_i))$$

4) Update weight distribution W_t:

$$W_{k+1}(i) = W_{k+1}(i)\beta_k^{\omega_k}$$

5) Normalization processing:

$$W_{k+1}(i) = \frac{W_{k+1}(i)}{\sum_i W_{k+1}(i)}$$

Step 5: The final classifier obtained by K weighted voting:

$$F(x) = \sum_{k=1}^{K} \beta_k \cdot f(x, y)$$

Table 5. Ensemble K-OAFA and Random Undersampling Algorithm (ICE-A)

4.4 Online cost-sensitive ICE-A (ICE-ABoost)

This section addresses the problem of online cost-sensitive learning using ensembles. Offline learning algorithms take as input a set of training instances and output a hypothesis. In contrast, online learning algorithms take as input a single labelled training instance as well as a hypothesis and output an updated hypothesis. Thus, given a sequence of training instances an online algorithm will produce a sequence of hypotheses. Online learning algorithms are designed to reuse the previous hypothesis in various ways, allowing them to reduce update times to meet the constraints of online learning problems—these constraints are typically much tighter than for offline problems. The advantages of this hypothesis reuse are even more significant in an ensemble learning algorithm, since offline ensemble construction can be very expensive. To our knowledge, all previous empirical evaluations of ensemble methods have taken place in offline learning settings. In this dissertation, we investigate online variants of ensemble learning algorithms and demonstrate online performance gains similar to those seen in the previous offline evaluations. We also ensure that our online variants have efficient implementations that might be applied to online learning problems with significant resource constraints, such as large-scale e-commerce. Without this restriction, an offline algorithm can be used directly in the online setting at substantial resource cost.

We distinguish between sequential-generation and parallel-generation ensemble approaches, and give reasons to focus on parallel generation in this research. We then describe a generic online ensemble algorithm that allows for parallel generation. We show a boosting-style instantiation of this algorithm that we have implemented called online ICE-ABoost.

A) Offline Ensemble Generation via Boosting

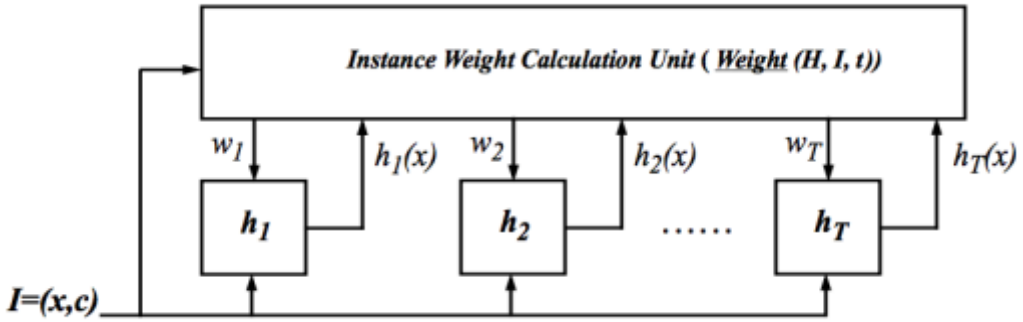
Boosting is an ensemble method that has received much attention and has been shown in several studies to outperform another popular ensemble method, bagging, in a number of offline domains [62]. We assume here that the base learning algorithms take into account a weight associated with each training instance, and attempts to return a learned hypothesis that is cost-sensitive and minimizes the weighted classification error. Some of the most commonly used boosting algorithms for offline problems generate hypotheses sequentially as follows. The first hypothesis is the result of presenting the set of training instances, all with weights of 1, to the base learning algorithm. Now assume the algorithm has already generated *T–1* hypotheses. Weights are then assigned to the training instances such that larger weights are associated with instances that the previous hypotheses performed poorly on (the "hard" instances). These weighted instances are then given to the base learning algorithm which outputs the T^{th} hypothesis. Boosting algorithms differ mainly in the ways weights are assigned to instances and the ways hypotheses are combined. The AdaBoost algorithm and the boost by majority algorithm [Freund, 1995] have been proven to be boosting algorithm in the theoretical sense. AdaBoost and ICE-A have been empirically compared and exhibit similar performance.

B) Online Ensemble Generation via Boosting

There are several avenues that could be explored when designing an online ensemble algorithm. A naive approach is to maintain a dataset of all observed instances and to invoke an offline algorithm to produce an ensemble from scratch when a new instance arrives. This approach is often impractical both in terms of space and update time for online settings with resource constraints. To help alleviate the space problem we could limit the size of the dataset by only storing and utilizing the most recent or most important instances. However, the resulting update time is still often impractical, particularly for boosting methods—when the training set used by a boosting algorithm is altered we potentially need to recalculate

weights and invoke the base learning algorithm for each of the *T* hypotheses from scratch. We propose an online ensemble algorithm that takes a parallel-generation approach, which means it generates and updates the ensemble members simultaneously. We say the algorithm takes a multiple-update approach if it updates more than one ensemble member for each training instance encountered, whereas a sequential approach only updates one member at a time until the completion of the generation of the previous member.

One reason for choosing a multiple-update parallel-generation approach is that the offline methods of boosting and bagging both have the property that a single training instance can contribute to the training of many ensemble members—we believe that achieving this property in the online setting is essential to obtaining rapid convergence to the desired target concept; this is particularly important in the presence of imbalanced data. Our empirical results described on Chapter 6 - Results provide evidence that our parallel-generation multiple-update ensembles converge more quickly than a sequential approach would. Sequential-generation algorithms also suffer additionally in the presence of imbalanced data because at any time most ensemble members are never going to be updated again—this patently requires adapting such algorithms with some kind of restart mechanism. Sequential methods also require a difficult-to-design method for determining when to stop updating an ensemble member in favor of starting on another member. To address these problems, we considered in this dissertation only algorithms taking the parallel-generation multiple update approach. We note that this approach interacts well with our motivating application in that multiple updates can easily be carried out simultaneously on a highly parallel implementation platforms.



Input:
ensemble $H = \langle (h_{1,\ldots}, h_T), (v_1, \ldots, v_T) \rangle$
new training instance $I = \langle x, c \rangle$
base online learning algorithm Learn (*instance, weight, hypothesis*)
voting weight update function Update-Vote (*ensemble, instance, hypothesis-number*)
instance weight function Weight (*ensemble, instance, hypothesis-number*)

1. for each $t \in \{1, 2, \ldots, T\}$, ;; *possibly executed in parallel*
2. do $\hat{v}_t$ = Update-Vote (H, I, t) ;; *the new voting weight of hypothesis t*
3. w_t = Weight (H, I, t) ;; *the weight of this instance for updating* h_t
4. $\hat{h}_t$ = Learn (I, w_t, h_t)

Output: new ensemble $\hat{H} = \langle (\hat{h}_{1\ldots}, \hat{h}_T), (\hat{v}_1, \ldots, \hat{v}_T) \rangle$

Table 6: Generic depiction of ICE-Aboost multiple-update online learning algorithm.

Above is the pseudo-code and graphical depiction of the online ensemble update procedure. The graphical depiction illustrates the following sequence of update events: 1) The training instance *I* is given to the base learners *h1, . . ., hT* and to the instance weight calculation unit. 2) The base learners provide their prediction *h1(x),..., hT(x)* of the class of the training instance to the instance weight calculation unit. 3) The instance weight calculation unit gives each base-learner an instance weight. 4) Finally, the base learners update their models according to the weight.

Table 6 shows the generic ICE-A multiple-update algorithm we use. The algorithm outputs an updated ensemble, taking as input an ensemble, a training instance, an online learning algorithm, and two functions *Update-Vote()* and *Weight()*. The function *Update-Vote()* is used to update the (*v1,...,vT*) vector of ensemble member voting weights (typically based on how each member performs on the new instance). The function *Weight()* is used for each ensemble member h_t to assign a weight w_t to the new instance for use in updating h_t. For each hypothesis h_t the algorithm performs the following steps:

- First, in line 2, a new scalar voting weight v_t is computed by the function *Update-Vote()*. For example, if *Update-Vote()* always returns the number one, the ensemble prediction will simply be the majority vote.
- In line 3, a scalar instance weight w_t is computed by *Weight()*. For example, in boosting *Weight()* would typically be a function of the number of mistakes made by previous hypotheses on the current instance, whereas in bagging *Weight()* would not depend on the ensemble members.
- Finally, in line 4, h_t is updated by *Learn()* using the training instance with the computed weight w_t. After each hypothesis and voting weight in the ensemble is updated in this manner (possibly in parallel), the resulting ensemble is returned.

The immediate research value is that ICE-A uses a memoryless instance of an ensemble algorithm, which results in an ensemble that outperforms single hypotheses in classification accuracy.

Chapter 5

5. Experimental Setup

For the classification method of balanced data, the classification accuracy is commonly used as an evaluation index. However, this evaluation index is the same for the cost of error classification of all kinds of samples, so the evaluation index is not reasonable in the imbalanced dataset. Typically, in imbalanced datasets, the positive class (Positive) represents a minority class, and the negative class (Negative) represents the majority class. The evaluation index of imbalanced data is generally based on the confusion matrix (see Table 7).

Category	Actual Positive Class	Actual Negative Class
Experimental Positive Class	True Positive	False Negative
Experimental Negative Class	False Positive	True Negative

Table 7. Confusion Matrix

The precision of reaction represents the ratio between the actual positive samples, and all experimental class samples.

$$precision = \frac{true\ positives}{predicted\ positives}$$

The recall, also known as specificity, represents the ratio between the actual true positive samples, which are classified as positive, and all the experimental positive class.

$$recall = \frac{true\ positives}{actual\ positives}$$

Recall, Precision and Fmeasure are the evaluation criteria for the positive class (minority class). In general, the Fmeasure is used as the evaluation criterion for the classification of imbalanced datasets.

$$Fmeasure = \frac{2 * precision * recall}{precision + recall}$$

Gmean is based on the correct classification rate of the minority class and the classification accuracy of the majority class, and it is usually used as a measure of the overall classification performance of the imbalanced dataset

$$Gmean = \sqrt{\frac{TN}{TN + FP} \times Recall}$$

Here, Fmeasure and Gmean were *selected* as the evaluation criteria to evaluate the performance of the algorithm on the imbalanced dataset.

5.1 Experimental Data

The experiment was carried out on a private large online retailer data sets as part of the authors' occupation and the proposed algorithms were compared with the most popular and prevalent existing algorithm, such as SVM, SMOTE and AdaBoost to make an effectiveness assessment.

In order to evaluate the ensemble K-OAFA algorithm for imbalanced data, 5 datasets, larger than five hundred thousand entries each, were selected to carry out the experiment shown in Table 8. Here we refer to specific characteristics of the customer shopping behavior and identifying features as attributes. Examples of attributes are name, age, preference for certain brands, categories, times of the year and many more.

Each set was divided into subsets of 10 to 100 and 100 to 1000 entries per set for experimental purposes. Sets Reco1 and Reco2 represented data used for delivering customer item recommendations, while sets P13N1, P13N2 and P13N3 were data representing customer segments (e.g. shoppers between 30-40 years of age with no children, etc.) and customer features (e.g. preferred shopping category, income bracket, frequency of purchases, etc.) used for personalized shopping experiences (e.g., replenishable lists, greetings, special discounts). In the selected datasets, the number of minority class samples and majority class samples is not balanced.

Dataset	Number of Data Samples	Number in Minority Class	Proportion of Data in Minority Class (%)	Number of attributes
Recos1	654,229	51,265	7.8%	115
Recos2	839,992	72,235	8.6%	203
P13N1	1,203,845	88,543	7.4%	344
P13N2	1,182,387	103,456	8.7%	120
P13N3	563,911	42,642	7.6%	319

Table 8. Experimental Datasets

In order to cancel the orders of magnitude difference between the dimensions of data and avoid a large prediction error caused by differences in input and output, a data normalization function was used here.

The experiment was carried out 100 times for two-class minority sets, and the average value was taken as the final result. The experimental results are shown in Table 9 and Table 10.

Dataset	Classification Algorithm			
	SVM	SMOTE	AdaBoost	K-OAFA
Recos1	0.510	0.697	0.531	0.763
Recos2	0.534	0.815	0.681	0.829
P13N1	0.634	0.628	0.634	0.652
P13N2	0.456	0.521	0.491	0.683
P13N3	0.732	0.894	0.864	0.934

Table 9. Experimental Result (Fmeasure)

Table 8 shows the Fmeasure comparison of the 4 algorithms in the five datasets. Table 10 shows the Gmean comparison of the four algorithms in the five datasets. As we can see from the Table 9 and Table 10, the SVM was used to classify the imbalanced datasets directly; Fmeasure and Gmean were relatively low. That was because it did not balance the imbalanced dataset. We can also see that the SMOTE algorithm has been carried out to balance the partial dataset, so Fmeasure and Gmean have been improved significantly.

As shown in Table 9 and Table 10, the AdaBoost used a number of classifiers for integration; the classification rate was higher than that of the SVM. However, the improvement was not very obvious in some data because the imbalance of the dataset was not handled. For example, in the P13N2 dataset the improvement of values of Fmeasure and Gmean was not very obvious.

Dataset	Classification Algorithm			
	SVM	SMOTE	AdaBoost	K-OAFA
Recos1	0.654	0.722	0.735	0.791
Recos2	0.731	0.851	0.901	0.917
P13N1	0.682	0.693	0.687	0.704
P13N2	0.535	0.635	0.546	0.702
P13N3	0.908	0.949	0.928	0.962

Table 10. Experimental Result (Gmean)

Because ICE-A utilized our K-OAFA algorithm to balance the imbalanced dataset and used the integrated technology to strengthen the classifier, therefore it resulted in having a better classification rate, and the Fmeasure and Gmean were higher than in the other algorithms. At the same time, compared with AdaBoost, the modeling time of ICE-A was reduced from 93.5s to 26.7s, as observed in the low-cost cloud heterogeneous computation environment used for these experiments. That happened because ICE-A performed under-sampling for the balanced dataset, reducing the size of the sample set and shortening the running time.

5.2 Computation Environment

Computational solutions range from cloud-based computing (e.g. Google Cloud, AWS) to an emerging revolution in high-speed, low-cost heterogeneous computational environments. Understanding how large-scale dynamic systems operate requires the integration of the many layers of customer-originating streaming information that high-throughput technologies are generating.

As an example, the amount of data from in-session behavior-based online experience personalization could collectively approach the petabyte scale for the raw information alone. The situation will soon be exacerbated by third-generation sequencing technologies that will enable us to scan entire clickstream data flow for massive businesses and other application areas in just minutes, and for less than US$100 [63]. To this should be added data from imaging technologies and other high-dimensional sensing methods. Although processing individual data dimensions is complex, the true challenge is in integrating the multiple sources of data. Mining such large high-dimensional data sets poses several hurdles for storage and analysis. Among the most pressing challenges are: data transfer, access control and management; standardization of data formats; and accurate modelling of clickstream systems by integrating data from multiple dimensions.

Solutions to integrating the new generation of large-scale data sets require approaches akin to those used in physics, climatology and other quantitative disciplines that have mastered the collection of large data sets. Cloud computing and heterogeneous computational environments are relatively recent inventions that address many of the limitations relating to data transfer, access control, data management, standardization of data formats and advanced model building.

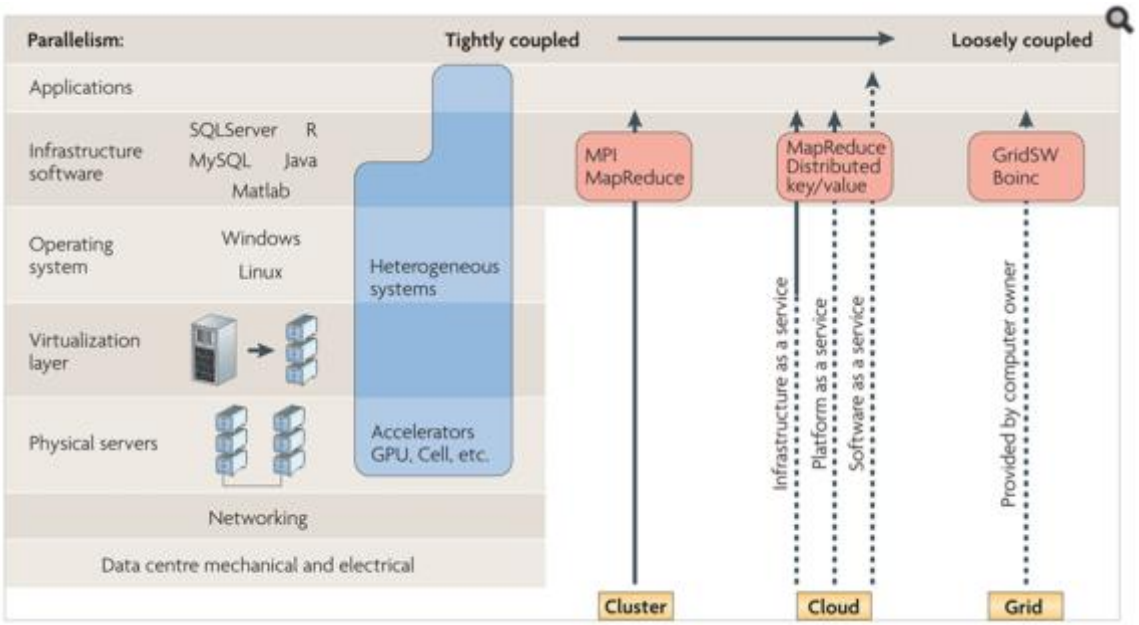


Fig. 6. Cluster, cloud, grid and heterogeneous computing hardware and software stacks

In order to understand the optimizations introduced via modern heterogeneous systems, we first need to fully understand their architecture [Fig. 6]. The hardware and software stacks comprise the different layers of a computational environment. At the lowest level of the stack is the physical structure that houses the hardware, with networking infrastructure coming next, followed by the physical computers or servers. Sitting on top of the physical hardware is the virtualization layer, and the operating system lies on top of that. Finally, there are the software infrastructure and application layers. The different types of computing can be differentiated by which of these layers are under the user's direct control (solid line) and which levels are provided by others, for example, the cloud provider and grid volunteer (dashed lines). Cloud and grid services are best suited for applications with loosely coupled, or coarse-grained, parallelism. Heterogeneous systems include specialized hardware accelerators, such as graphics processing units (GPUs). These accelerators are optimized for massive tightly coupled, or fine-grained, parallelism. However, the software that runs on

these accelerators differs from its general-purpose processor (GPP) counterparts, and often must be specifically written for a particular accelerator. MPI stands for message passing interface.

Compared to general purpose processors (GPPs), heterogeneous systems can deliver a tenfold increase or greater in peak arithmetic throughput for a few hundred US dollars. Cloud computing, on the other hand can make large-scale computational clusters readily available on a pay-as-you-need basis. But both approaches have trade-offs that result from trying to optimize for peak performance (heterogeneous systems) or low-cost and flexibility (cloud computing). It is important that we understand the advantages and disadvantages of these different computational platforms and the problems to which they are best suited. See Table 11.

Large-scale Computing Platform	Computing Architectures	Advantages	Disadvantages	Example Applications
Cluster computing	Multiple computers linked together, typically through a fast-local area network, that effectively function as a single computer	Cost-effective way to realize supercomputer performance	Requires a dedicated, specialized facility, hardware, system administrators and IT support	Parallel Data Mining Agents Bayesian network reconstruction Supervised and unsupervised learning
Cloud computing	Computing capability that abstracts the underlying hardware architectures (for example, servers, storage and networking), enabling convenient, on-demand network access to a shared pool of computing resources that can be readily provisioned and released (NIST Technical Report)	The virtualization technology used results in extreme flexibility; good for one-off HPC tasks, for which persistent resources are not necessary	Privacy concerns; less control over processes; bandwidth is limited as large data sets need to be moved to the cloud before processing	Searching sequence databases Aligning raw sequencing reads to previously recognized data patterns Detection of data outliers Most applications running on a cluster can be transferred to a cloud
Grid computing	A combination of loosely coupled networked computers from different administrative centers that work together on common computational tasks. Typified by	Ability to enlist large-scale computational resources at low or no cost (large-scale	Big data transfers are difficult or impossible; minimal control over underlying	Distributed supercomputing High throughput computing

	volunteer computing efforts (such as Folding@Home), which 'scavenge' spare computational cycles from volunteers' computers	volunteer-based efforts)	hardware, including availability	
				On-demand computing
Heterogeneous computing	Computers that integrate specialized accelerators — for example, GPUs or reconfigurable logic (FPGAs) — alongside GPPs	Cluster-scale computing for a fraction of the cost of a cluster; optimized for computationally intensive fine-grained parallelism; local control of data and processes	Significant expertise and programmer time required to implement applications; not generally available in cluster- and cloud-based services	Bayesian network learning
				Running computationally heavy deep neural networks (DNN), convolutional neural networks (CNN), deep belief networks (DBN) classifier jobs

Table 11. Advantages and disadvantages to computation architecture formats

Due to the combined computationally-intense parallelism capabilities and the relatively low cost and accessibility of cloud heterogeneous computation, we have selected to use a Google cloud-based environment for running all of our experiments for this work. Specifically, we used the powerful modeling framework TensorFlow, real-time streaming framework Storm and distributed data base Couchbase. All of these tools existed on Google Cloud Platform.

Chapter 6

6. Results

In this chapter, we review the results from the experiments performed with the aforementioned subsets between 10-100 and 100-1000 for all 5 datasets Reco1, Reco2, P13N1, P13N2 and P13N3. These analyses have the purpose of representing e-commerce use cases where certain data attributes belong to the majority (95% or more of the class entries) or to the minority (less than 5% of the class entries) class. By understanding the performance of our classifiers against best-in-class techniques, we are able to prove their superior applicability for the recognition of imbalanced multiclass data points. These techniques are then to be used in the classification of data points representing customer behavior patterns, which are the foundation of serving relevant personalized content.

We have divided the outcomes of our experiments into multiclass minority and majority class examples, and have outlined the most important aspects of their correlation and performance analysis. We are using 5 performance measures – subset origin, recall, precision, Fmeasure and Gmean, and are testing the performance of the state-of-art SMOTE, against our new methods K-OAFA and ICE-A.

6.1 Multiclass Minority Cases

In this section, we are conducting analysis and performance pattern recognition on multiclass minority cases. The number of minority classes is varied from 1 to 20, as that provides significant depth and can be used in DNNs. The impact of multi-minority on the performance of over-sampling and under-sampling techniques is illustrated and analyzed in depth.

A) Correlation Analysis:

Five performance measures and three ensemble training methods (e.g., SMOTE, K-OAFA, and ICE-A) permit 15 pairwise correlations with respect to the number of minority classes. They show whether multi-minority degrades the classification performance of the three ensemble training methods and which performance aspects are affected. The three single-class measures are recorded for the minority class that joins all the training sessions from 1 to 20. Table 12 summarizes the correlation coefficients for “10–100” and “100–1000” data groups.

All pairs present very strong negative correlations on both groups of small and large data sets. It implies a strong monotonic decreasing relationship between the measures and the number of minority classes. All of them are decreasing as more minority classes are added into the training data, regardless of the size of the training data and whether resampling is applied. In other words, multi-minority reduces the performance of these ensembles consistently, and data resampling seems not to be helpful. Next, we will investigate the performance degradation caused by multi-minority classes from the level of every single class.

Correlation				
10-100	**Recall**	**Precision**	**Fmeasure**	**Gmean**
SMOTE	-89	-94	-91	-97
K-OAFA	-88	-93	-91	-98
ICE-A	-93	-93	-93	-99
100-1000	**Recall**	**Precision**	**Fmeasure**	**Gmean**
SMOTE	-99	-99	-100	-100
K-OAFA	-99	-99	-99	-100
ICE-A	-99	-99	-100	-100

Table 12. Rank correlation coefficients (in percent) between the number of majority classes and four performance measures in three ensemble methods on "10-100" and "100-1000" data sets. Recall, precision, and Fmeasure are calculated for minority class

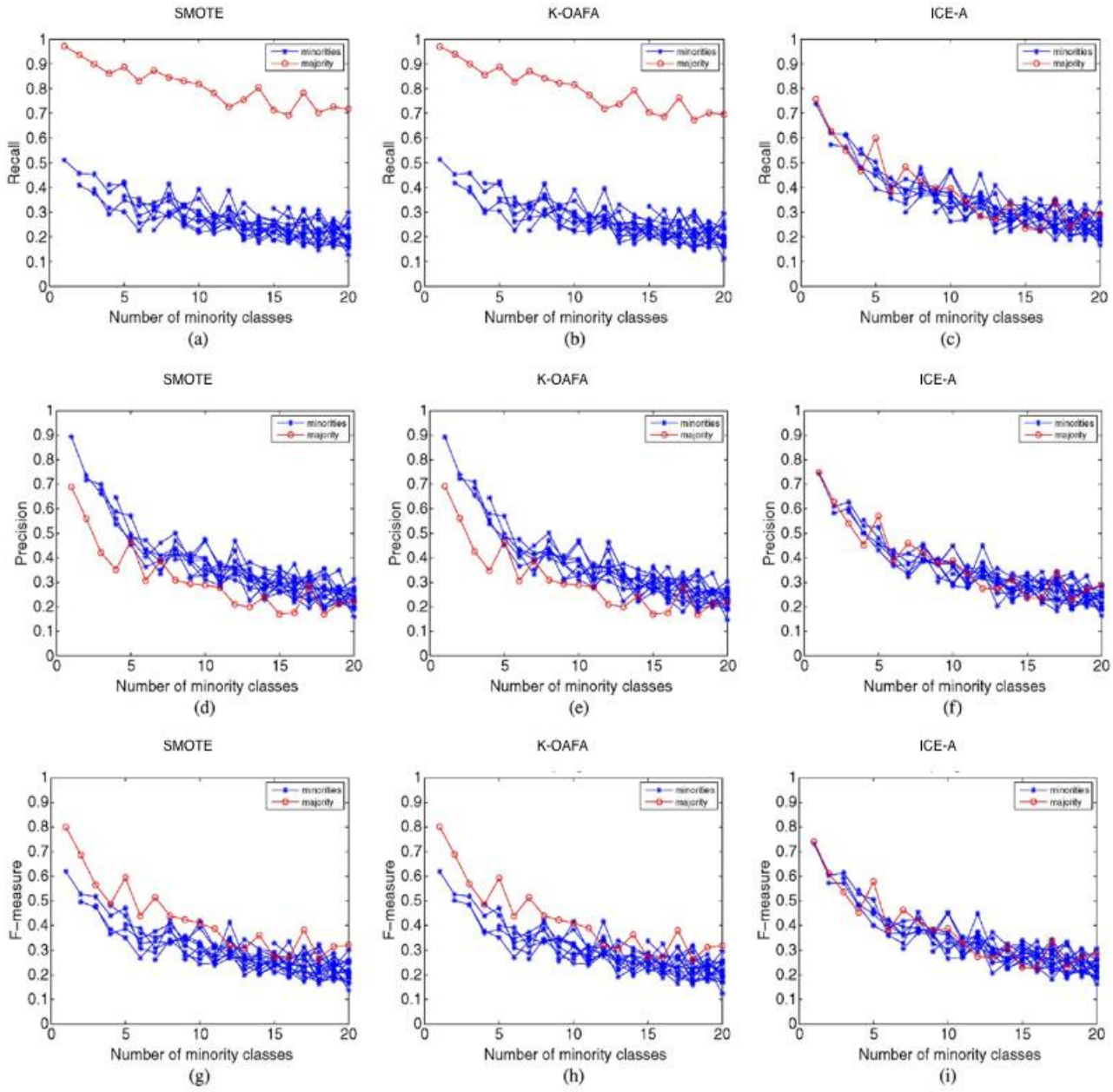


Fig 7. Single-class performance patterns among classes in multi-minority cases of "10–100" (x-axis: number of minority classes; y-axis: performance output). (a) Recall: SMOTE. (b) Recall: K-OAFA. (c) Recall: ICE-A. (d) Precision: SMOTE. (e) Precision: K-OAFA. (f) Precision: ICE-A. (g) Fmeasure: SMOTE. (h) Fmeasure: K-OAFA. (i) Fmeasure: ICE-A.

B) Performance Pattern Analysis:

We now focus on the "10–100" group of data sets and illustrate the changing tendencies of single-class measures for all classes as the class number increases. In Fig. 7, the presented pattern reveals detailed information about how the classification performance of each class is affected and the differences among ensemble methods and evaluated measures. All of the following pattern plots are scaled in the same range.

As seen in Fig. 7, every class's performance is decreasing. No evidence shows which class suffers from more performance degradation than other classes. The classification gets equally difficult on all classes. For each class, corresponding to one curve in the plot, the measure value drops faster at the first few steps, when the number of minority classes is approximately smaller than 10. As it gets larger, the rate of reduction slows down.

Among the three performance measures, the drop of precision [Fig. 8(d) and (e)] is more severe than that of recall [Fig. 7(a) and (b)] in SMOTE and K-OAFA. Precision is the main cause of the decrease in Fmeasure. The reason is that multi-minority increases the risk of predicting an example into a wrong class. As to recall, it seems that the difficulty of recognizing examples within each class is less affected by multi-minority as compared to precision because the proportion of each class of data in the whole data set is hardly changed by adding a small class. In ICE-A, each class is reduced to have a small size. Adding minority classes changes the proportion of each class significantly. It explains why ICE-A's recall [Fig. 7(c)] presents higher sensitivity to multi-minority than the recall produced by SMOTE and K-OAFA [Fig. 7(a) and (b)].

Among the three ensemble methods, SMOTE and K-OAFA have similar performance patterns, where the majority class obtains higher recall and Fmeasure than the minority classes, but lower precision values. Over-sampling does not alleviate the multiclass problem. Although over-sampling increases the quantity of minority class examples to make every class have the same size, the class distribution in data space is still imbalanced, which is dominated by the majority class. In ICE-A, undersampling counteracts the performance differences among classes. During the first few steps, ICE-A presents better recall and Fmeasure on

minority classes [Fig. 7(c) and (i)] than SMOTE and K-OAFA [Fig. 7(a), (b), (g), and (h)]. From this point of view, it seems that using undersampling might be a better choice. However, its advantage is weakened as more minority classes join the training. When the class number reaches 20, three ensemble algorithms have very similar minority-class performance. The reason could be that undersampling explicitly empties some space for recognizing minority classes by removing examples from the majority class region. When there is only one minority class, a classifier is very likely to assign the space to this class. When there are many minority classes, they have to share the same space. Hence, the effect of undersampling is reduced. Undersampling seems to be more sensitive to multi-minority. For this consideration, it would be better to expand the classification area for each minority class, instead of shrinking the majority class. To achieve this goal, advanced techniques are necessary to improve the classification generalization over minority classes.

Finally, it should be noted that SMOTE and K-OAFA return similar results when plotted out [Fig. 7(a), (b), (d), (e), (g) and (h)] due to the similar nature of their approach via over-sampling. The difference, not visible in these graphs, remains within the maintained knowledge of minority class boundaries existent in K-OAFA and not in SMOTE.

6.2 Multiclass Majority Cases

We proceed with the same analyses for the multi-majority data "10–100" and "100–1000." The number of majority classes is varied from 1 to 20. The impact of multi-majority is studied here.

A) Correlation Analysis:

Table 13 summarizes the correlation coefficients. Single-class performance measures are recorded for the only minority class of each data set. Similar to the multi-minority cases, strong negative correlations between five performance measures and the number of majority classes are observed in both groups of small and large data sets, which indicate a monotone

decreasing relationship. All three ensemble training methods suffer from performance reduction caused by "multi-majority."

10-100	**Recall**	**Precision**	**F-measure**	**G-mean**
SMOTE	-79	-85	-92	-97
K-OAFA	-84	-86	-92	-97
ICE-A	-92	-94	-95	-99
100-1000	**Recall**	**Precision**	**F-measure**	**G-mean**
SMOTE	-100	-96	-100	-100
K-OAFA	-100	-93	-99	-100
ICE-A	-99	-99	-100	-100

Table 13. Rank correlation coefficients (in percent) between the number of majority classes and four performance measures in three ensemble methods on "10-100" and "100-1000" data sets. Recall, precision, and Fmeasure are calculated for minority class.

B) Performance Pattern Analysis:

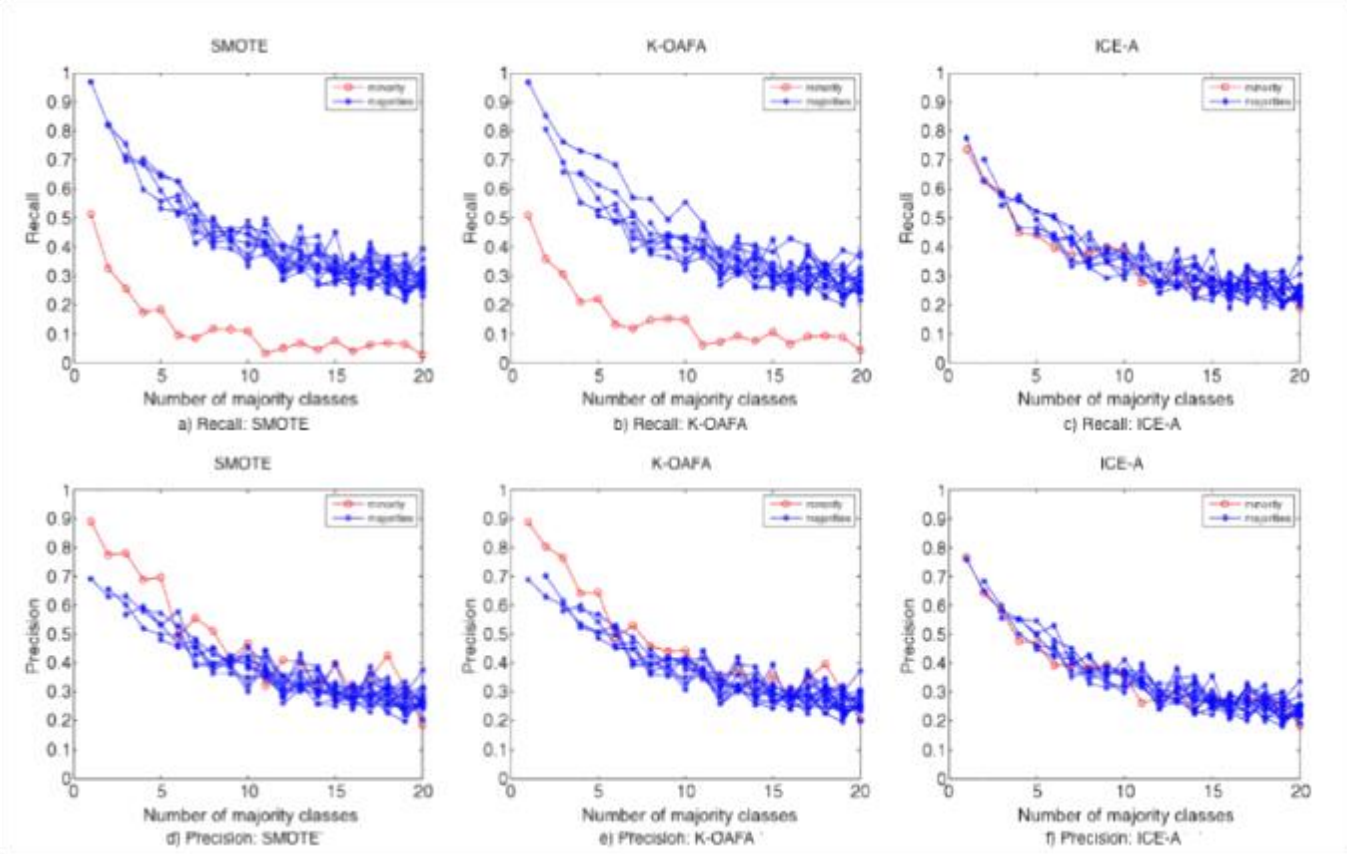
To gain more insight, we focus on the "10–100" group of data sets and present the changing tendencies of single-class measures for each class along with the increase of the number of majority classes in Fig. 8. All plots are in the same axis scale.

Among the classes in each plot, adding majority classes makes the recognition of examples of each class [i.e., recall presented in Fig. 8(a)–(c)] equally difficult. In SMOTE and K-OAFA, minority-class precision drops faster than that of the majority classes [Fig. 8(d) and

(e)] because the large quantity of new majority class examples overwhelms the minority class even more. Minority class examples are more likely to be misclassified than before compared to majority class examples.

All performance measures present a drastic decrease. Especially in recall plots of SMOTE and K-OAFA [Fig. 8(a) and (b)], more and more majority class examples take the recognition rate of the minority class down to nearly 0. For every existing majority class, adding more majority classes can make it appear to be in minority. Therefore, the recall of majority classes also shows a fast drop.

Among the three ensemble methods, ICE-A produces better minority-class Fmeasure than SMOTE and K-OAFA, but the recall of majority classes is sacrificed greatly. It causes the concern that using undersampling will lose too much data information when multiple majority classes exist and can lead to severe performance reduction in majority classes.



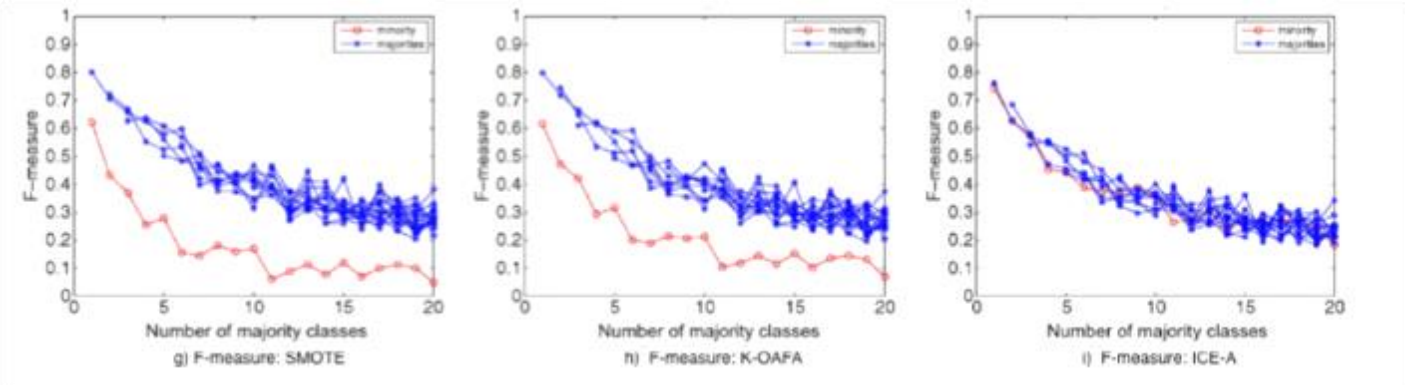


Fig 8 (Cont.). Single-class performance patterns among classes in multi-majority cases of "10–100" (x-axis: number of minority classes; y-axis: performance output). (a) Recall: SMOTE. (b) Recall: K-OAFA. (c) Recall: ICE-A. (d) Precision: SMOTE. (e) Precision: K-OAFA. (f) Precision: ICE-A. (g) F-measure: SMOTE. (h) F-measure: K-OAFA. (i) F-measure: ICE-A

Based on all of the observations in this section, we make the following conclusion: 1) As no new information is introduced into the minority class to facilitate the classification in SMOTE and K-OAFA, overfitting minority-class regions happens with low recall and high precision values when compared with those measures obtained from the majority classes. Over-sampling does not help for both multi-minority and multi-majority cases. 2) ICE-A performs the same under multi-minority and multi-majority cases due to undersampling. In the multi-minority case, ICE-A can be sensitive to the class number; in the multi-majority case, there is an elevated risk of sacrificing too much majority-class performance. 3) Between multi-minority and multi-majority, the multi-majority case seems to be more difficult than the multi-minority case. SMOTE and K-OAFA present much worse minority-class performance in Fig. 7(g) and (h) compared to Fig. 8(g) and (h). This is because adding majority class examples aggravates the imbalanced situation. 4) Between balanced and imbalanced data, multiclass leads to performance degradation in both scenarios. We believe that learning imbalanced data is much harder than learning balanced one, for the performance difference

between the types of classes shown in the performance pattern analysis and the particular performance requirement for minority classes, which would not happen in the balanced case. Therefore, because of different learning objectives and varying e-commerce use cases, different treatments should be considered.

6.3 Success measure

As this work is focused on advancing modern e-commerce imbalanced data-driven personalization, we have assigned our measures of success in accordance with what they represent in real-life online shopping. An important aspect of the applicability of our proposed methods is their cost efficiency, precision of prediction and human-friendly visualization capabilities for the easier utilization by professional data scientists.

For both K-OAFA and ICE-A, we have proven are two online and offline learning algorithms which either meet or beat the best-known algorithms for multiclass majority and minority use cases in terms of precision, and recall resulting in more plentiful and relevant item recommendations and personalization experiences.

In addition, as described in Chapter 4 – Approach and e-Commerce Specific Challenges, ICE-A is an online cost-effective method which reuses previous hypothesis resulting in reduced update times to meet the tight constraints of online learning problems. The advantages of hypothesis reuse are furthermore evident in ensemble algorithms due to the prohibitive cost associated with offline ensemble construction.

Consequently, one of the biggest advantages of ICE-A over other classifiers with similar application areas is the simple for implementation approach of data nodes visualization, faster results (an average improvement of 65 seconds) and pattern detection. This approach is not data class size dependent and could accommodate vast use cases, which, in the case of real life e-Commerce applications, is of vital importance. That allows the quick and intuitive detection of meaningful imbalanced entries, whereas other approaches require complex for

implementation analysis techniques. By performing our calculations via ICE_A with a velocity of 55 seconds for 1M data points, and 2 minutes for SMOTE, we are now presenting a solution which has a potential to improve business revenue by millions of dollars with its faster reaction time.

Industry studies have shown that customers bounce off of retail websites on average after 68 seconds if they cannot find their desired items. This creates a need for fast-responding model training solutions reflecting the latest customer signals in production.

CHAPTER 7

7. Conclusion

In this chapter, we analyze the outcome of our proposed solution for an improved classifier with an application domain within e-commerce. Based on the improved over-sampling algorithm and integration technology, we proposed an integrated classification algorithm for imbalanced data. First, the traditional SMOTE algorithm was improved to K-OAFA, reducing the defects of the SMOTE algorithm. Then, combined with the classifier ensemble technology, an integrated classification algorithm for imbalanced data named ICE-A was proposed. In ICE-A, K-OAFA was used to conduct over-sampling, and random under-sampling was carried out to reduce the problem scale and form a new dataset. In the new dataset, a number of weak classifiers were trained to generate, and integration techniques were used to integrate several weak classifiers to form the final strong classifier. The experiment was carried out on private large online retailer datasets, Fmeasure and Gmean were used as the evaluation indexes to evaluate the proposed algorithm. Consequently, the experiment results have proven the effectiveness of the new algorithm in dealing with multi-modal commerce data.

7.1 E-commerce advancements

Online retail personalization and item recommendations are possible via a complex system of algorithms supporting unique experiences throughout the customer shopping

journey. User behavior–based robust classifiers utilizing deep learning intent predictions are pivotal tools for modern customer segmentation and customization [64]. Contemporary research has shown that extracting features from high dimensional data during the pre-train phase improves prediction accuracy and precision, and results in higher integrity of personalization relevance [65]. As an added layer of complexity, in recent years, a multitude of personalization channels have emerged ranging from direct in website (via video, audio, banners, concept extraction, shipping cost and delivery times estimations, targeted merchandising, customized search results, and many more), to smart Customer Relationship Management (CRM) platforms in the form of personalized trigger and batch emails, as well as item recommendations in house and on social media. Therefore, modern solutions need to be adaptable to multi-modal data demands. Our hybrid ICE-A classifier follows the same model, and has shown substantial improvements over traditional classification techniques in working with multi-modal data.

7.2 Machine learning trends in e-commerce

In recent year, numerous personalization-associated trends are emerging across all of e-commerce and are quickly becoming the standard which most users are trained to expect. In this section, we review the current trends in e-commerce which require machine learning-driven performance to be scalable and cost-effective for actual application.

A) Smart chatbots

Good customer service often requires a conversation which is why chat works so well in e-commerce. When a shopper submits a question via chat, a customer service representative can answer and guide the shopper to a solution. Similarly, when a shopper posts a question or complaint on social media sites (e.g., Facebook, Twitter) a quick and helpful response makes a world of difference in that shopper's experience. Yet, all size businesses may find it

challenging to staff and maintain a team of customer service representatives large enough to monitor all chat and social media.

This is where one of the latest trends in personalized customer experiences has emerged in the form of *intelligent, learning chatbots* that can manage basic customer service questions and learn how to help customers in ways that are specific to a particular online store. These chatbots are able to take care of on-site chat sessions or social media tweets and posts.

Currently, many third-party companies are developing and offering learning chatbot solutions that even small online merchants can afford.

B) Product search

Personalized search results are still in an initial state of being designed as a regular customer experience but due to their wide-spread popularity and the omnipresence of product searching in retail, they are quickly gaining traction. As part of personalized search based on data science techniques, companies focus on intelligent ranking based on previous customer behavior patterns, query understanding and expansion, related queries (recommendations for other similar searches to the one you have just executed that might render you a better result), de-duping, image recognition and understanding, concept extraction, sentiment and trend analysis and entity recognition (i.e., customer identity stitching).

C) Targeted marketing

Product recommendations are among the most powerful form of on-site merchandising for online retailers. Learning product recommendation systems promise to dramatically improve conversion rates and customer satisfaction, but in order to continue improving their performance with the growing expectations of shoppers, these recommendations need to be targeted at the level of the specific customer rather than on general trends level.

Current product recommendation systems generally use a particular product's popularity to decide how and when to recommend it. But machine-learning recommendation systems may take a shopper's particular buying habits into consideration or compare product attributes like matching colors or "looks" to recommend. The system may even predict which recommendation will be the most likely to generate incremental sales.

D) Optimized pricing

In the near future, online retailers may be able to use learning algorithms to analyze and understand pricing trends, product demand, and customer behavior to determine the just-right price for a particular item, to maximize profit or achieve other ecommerce business goals.

Too often, online sellers become involved in a margin-slashing price war with competitors, particularly on marketplaces. But a learning price-management system may help retailers find the best price for each item it carries.

E) Fraud detection and prevention

For financial reasons, fraud detection and prevention tends to be more of an issue for relatively large ecommerce businesses than for small or even mid-sized retailers. Small

ecommerce business may not experience enough fraud to make it worthwhile to purchase fraud detection software.

If applicable to employ a fraud prevention solution, you can expect machine-learning solutions to become popular. These systems will look for fraud patterns in a particular e-commerce business's customer base. The key advantage is that a learning system will be almost unique to its e-commerce retailer. It will be looking at the trends that predict fraud in a very specific way. Ultimately, this could make the system much better at predicting fraud relative to a particular e-commerce business.

F) Improved business decisions

Machine learning algorithms may also contribute to ecommerce decision-making, including any of the following operations.

- Predicting product demand.
- Supply and demand analysis and forecast
- Wallet management and funding source optimization
- Various scheduling and optimal resource allocation
- Classifying products and identifying keywords.
- Managing marketing campaigns.
- Estimating shipping and packing costs.
- Improving customer segmentation.

In conclusion, we can say that due to the complex behavioral patterns associated with some of these applications, likely the future of machine learning in e-commerce will utilize

various deep learning techniques, such as deep neural networks (DNN) and recurrent neural networks (RNN). These classification methodologies allow for building multi-level learning networks which can predict undetectable in a scalable way by common human-powered logic patterns and future actions.

7.3 Future work

In this work, we focused on detection and understanding of customer behavior-based attributes as displayed in the shoppers' online retail journey. With this we are solving problems associated with personalized content and experiences for newly or previously acquired customers through various channels. This leaves us agnostic to use cases further removed from the individual customer around imbalanced data associated with traffic trends identification. Said trends could serve as an indicator to shifts in shopping preferences in omnichannel businesses (shopping in store vs. shopping online), brand popularity changes, micro and macroeconomics state, and many other market-level predictions.

One of the contemporary trends in traffic identification is data gravitation-based classification. Yet, just like in attribute detection and classification, imbalanced data is not yet well-handled in monitoring of the activities of internet applications despite its frequent occurrence. As this is a problem which is rarely considered by the research community at large, we feel it could be a good opportunity for expansion of this work. Specifically, we would be focusing on e-commerce-specific trends detection based on in-demand market research.

REFERENCES

1. K. Satyasree, J. Murthy, "An exhaustive literature review on class imbalance problem. International Journal of Emerging Trends and Technology Computer Science", pp. 109–118, 2013.

2. S. Wang, X. Yao, “Multiclass imbalance problems: Analysis and potential solutions.” Proceedings of IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), pp. 1119-1130, 2012.

3. Braik, William, "Real time streaming pattern detection for e-commerce." Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, 2016.

4. R. Polikar, “Ensemble Based Systems in Decision Making,” IEEE Circuits and Systems Magazine, vol. 6, no. 3, pp. 21–45, 2006

5. M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, “A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches,” IEEE Trans. Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 42, no. 4, pp. 463–484, 2012.

6. P. E. Utgoff, N. C. Berkman, and J. A. Clouse, “Decision Tree Induction Based on Efficient Tree Restructuring,” Machine Learning, vol. 29, no. 1, pp. 5–44, 1997.

7. A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, “Online Random Forests,” in Proc. Online Learning for Computer Vision Workshop, 2009, pp. 1393–1400.

8. M. Denil, D. Matheson, and N. de Freitas, “Consistency of Online Random Forests,” Proc. Int’l Conf. Machine Learning, 2013.

9. L.-P. Liu, Y. Jiang, and Z.-H. Zhou, “Least Square Incremental Linear Discriminant Analysis,” in Proc. Int’l Conf. Data Mining, pp. 298–306, 2009.

10. T.-K. Kim, S.-F. Wong, B. Stenger, J. Kittler, and R. Cipolla, "Incremental Linear Discriminant Analysis Using Sufficient Spanning Set Approximations," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 1–8, 2007.

11. J. Langford, L. Li, and T. Zhang, "Sparse Online Learning via Truncated Gradient," J. Machine Learning Research, vol. 10, pp. 777– 801, 2009.

12. P. Laskov, C. Gehl, S. Kruger, and K.-R. Muller, "Incremental Support Vector Learning: Analysis, Implementation and Applications," J. Machine Learning Research, vol. 7, pp. 1909–1936, 2006.

13. G. Cauwenberghs and T. Poggio, "Incremental and Decremental Support Vector Machine Learning," Advances in Neural Information Processing Systems, pp. 409–415, 2011.

14. J. Kivinen, A. J. Smola, and R. C. Williamson, "Online Learning with Kernels," IEEE Trans. Signal Processing, vol. 52, no. 8, pp. 2165–2176, 2004.

15. 25 S. C. Hoi, R. Jin, P. Zhao, and T. Yang, "Online Multiple Kernel Classification," Machine Learning, vol. 90, no. 2, pp. 289–316, 2013.

16. N. C. Oza and S. Russell, "Online Bagging and Boosting," in Proc. Artificial Intelligence and Statistics, pp. 105–112, 2001.

17. H. He, Self-Adaptive Systems for Machine Intelligence, ISBN: 978-0-470-34396-8, Hardcover, Wiley, 248 pages, 2011.

18. H. He and E.A. Garcia, "Learning from Imbalanced Data," IEEE Trans. Knowledge Data Eng., vol. 21, no. 9, pp. 1263-1284, 2009.

19. X.Y. Liu, J. Wu, and Z.H. Zhou, “Exploratory Under Sampling for Class Imbalance Learning,” Proc. Int’l Conf. Data Mining, pp. 965- 969, 2006.

20. X.Y. Liu, J. Wu, and Z.H. Zhou, “Exploratory Under Sampling for Class Imbalance Learning,” Proc. Int’l Conf. Data Mining, pp. 965- 969, 2006.

21. J. Zhang and I. Mani, “KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction,” Proc. Int’l Conf. Machine Learning, Workshop Learning from Imbalanced Data Sets, 2003.

22. M. Kubat and S. Matwin, “Addressing the Curse of Imbalanced Training Sets: One-Sided Selection,” Proc. Int’l Conf. Machine Learning, pp. 179-186, 1997.

23. Victor H. Barella, Eduardo p. Costa, and Andre C P L F Carvalho, “ClusterOSS: a new undersampling method for imbalanced learning” International Journal of Computer Applications (0975 – 8887) National Conference on Advances in Computing, 0975-8887, 2015.

24. N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, “SMOTE: Synthetic Minority over-sampling Technique,” J. Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

25. Reshma C. Bhagat and Sachin S. Patil, “Enhanced SMOTE Algorithm for Classification of Imbalanced Big- Data using Random Forest”, IEEE International Advance Computing Conference (IACC), 2015.

26. H. He and E.A. Garcia, “Learning from Imbalanced Data,” IEEE Trans. Knowledge Data Eng., vol. 21, no. 9, pp. 1263-1284, 2009.

27. H. Han, W.Y. Wang, and B.H. Mao, “Borderline- SMOTE: A New Over-sampling Method in Imbalanced Data Sets Learning,” Proc. Int’l Conf. Intelligent Computing, pp. 878-887, 2005.

28. H. He, Y. Bai, E.A. Garcia, and S. Li, “ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning,” Proc. Int’l Joint Conf. Neural Networks, pp. 1322-1328, 2008.

29. S. Chen, H. He, and E.A. Garcia, “RAMOBoost: Ranked Minority Over-sampling in Boosting,” IEEE Trans. Neural Networks, vol. 21, no. 20, pp. 1624-1642, 2010.

30. Sukarna Barua, Md. Monirul Islam, Xin Yao, “MWMOTE-Majority Weighted Minority Over-sampling Technique for imbalanced data set learning”, IEEE Trans. Knowledge and data engineering, vol. 26, no. 2, 2014.

31. H. He and E.A. Garcia, “Learning from Imbalanced Data,” IEEE Trans. Knowledge Data Eng., vol. 21, no. 9, pp. 1263-1284, 2009.

32. G. Wu and E. Y. Chang, “Adaptive Feature-Space Conformal Transformation for Imbalanced-Data Learning,” in Proc. Int’l Conf. Machine Learning, pp. 816–823, 2003.

33. Y. Lin, Y. Lee, and G. Wahba, “Support Vector Machines for Classification in Nonstandard Situations,” Machine Learning, vol. 46, no. 1-3, pp. 191–202, 2002.

34. H. Masnadi-Shirazi and N. Vasconcelos, “Risk Minimization, Probability Elicitation, and Cost-Sensitive SVMs,” in Proc. Int’l Conf. Machine Learning, pp. 759–766, 2010.

35. B. Zadrozny and C. Elkan, “Learning and Making Decisions When Costs and Probabilities Are Both Unknown,” in Proc. ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining, pp. 204–213, 2001.

36. C. Drummond and R. C. Holte, “Exploiting the Cost (In)Sensitivity of Decision Tree Splitting Criteria,” in Proc. Int’l Conf. Machine Learning, pp. 239–246, 2000.

37. He, Haibo. Self-adaptive systems for machine intelligence. John Wiley & Sons, 2011.

38. Xingyi LIU, “Cost-sensitive Decision Tree with Missing Values and Multiple Cost Scales”, Int’l Joint Conf. on Artificial Intelligence, 2009.

39. He, Haibo. Self-adaptive systems for machine intelligence. John Wiley & Sons, 2011.

40. Zhi-Hua Zhou and Xu-Ying Liu, “Training Cost- Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem”, IEEE Trans on knowledge and data engineering, vol. 18, no. 1, 2006.

41. B. Settles and M. Craven, “An analysis of active learning strategies for sequence labeling tasks,” in Proc. Conf. Empirical Methods NaturalLang. Process. (EMNLP), pp. 1070–1079, 2008.

42. S. Dasgupta, D. Hsu, and C. Monteleoni, “A general agnostic active learning algorithm,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 20. 2008, pp. 353–360

43. C. X. Ling and J. Du, “Active learning with direct query construction,” in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. DataMining (KDD), Las Vegas, NV, USA, pp. 480–487, 2008.

44. D.Tuia, F. Ratle, F. Pacifici, M.F. Kanevski and W.J. Emery, “Active Learning Methods for Remote Sensing Image Classification”, IEEE Trans. on Geo-science and Remote sensing, vol. 47, issue 7, 2009.

45. Jing Zhang, Xindong Wu and Victor S. Sheng, “Active Learning with Imbalanced MultipleNoisy Labeling”, IEEE Trans. on Cybernetics, vol. 45, no. 5, 2015.

46. Zeng, ZhiQiang, and ShunZhi Zhu. "A kernel-based sampling to train SVM with imbalanced data set." *Conference Anthology, IEEE*. IEEE, 2013.

47. Bo ZHOU, Cheng YANG, Haixiang GUO and Jinglu HU, “A Quasi-linear SVM Combined with Assembled SMOTE for Imbalanced Data Classification”, Int’l Joint Conf. on Neural Networks, 2013.

48. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics

49. L. I. Besaleva and A. C. Weaver, "CrowdHelp: A crowdsourcing application for improving disaster management," *2013 IEEE Global Humanitarian Technology Conference (GHTC)*, San Jose, CA, pp. 185-190, 2013.

50. L.J. Zhao, X.K. Diao, D.C. Yuan, W. Tang, Enhanced classification based on probabilistic extreme learning machine in wastewater treatment process, Procedia Engineering, Volume 15, pp. 5563-5567, 2011.

51. Liao, Shuai, et al. "Tag features for geo-aware image classification." IEEE Transactions on Multimedia 17.7, pp. 1058-1067, 2015.

52. Chen, Chao, Andy Liaw, and Leo Breiman, "Using random forest to learn imbalanced data." University of California, Berkeley 110, 2004.

53. Galar M., Fernández A., Barrenechea E., Herrera F. EUSBoost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recognition, pp. 3460–3471, 2013.

54. Alejo R., Sotoca J.M., García V., Valdovinos R.M. (2010) Cost-Sensitive Neural Networks and Editing Techniques for Imbalance Problems. In: Martínez-Trinidad J.F., Carrasco-Ochoa J.A., Kittler J. (eds) Advances in Pattern Recognition. MCPR 2010. Lecture Notes in Computer Science, vol 6256. Springer, Berlin, Heidelberg

55. C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006

56. L. I. Kuncheva and C. J. Whitaker, “Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy,” Machine learning, vol. 51, no. 2, pp. 181–207, 2003

57. Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” in J. Computer and System Sciences, vol. 55, 1997.

58. N. C. Oza and S. Russell, “Online Bagging and Boosting,” in Proc. Artificial Intelligence and Statistics, pp. 105–112, 2001.

59. N. C. Oza and S. Russell, “Online Bagging and Boosting,” in Proc. Artificial Intelligence and Statistics, N. C. Oza, “Online Ensemble Learning,” Ph.D. dissertation, University of California, Berkeley, pp. 105–112, 2001.

60. S. Wang and X. Yao, “Diversity Analysis on Imbalanced Data Sets by Using Ensemble Models,” in Proc. IEEE Symp. Computational Intelligence and Data Mining, pp. 324–331, 2009.

61. S. Wang and X. Yao, “Diversity Analysis on Imbalanced Data Sets by Using Ensemble Models,” in Proc. IEEE Symp. Computational Intelligence and Data Mining, pp. 324–331, 2009.

62. Freund & Schapire, 1996; Quinlan, 1996; Bauer & Kohavi, 1999; Dietterich, 2000

63. Flusberg BA, et al. Direct detection of DNA methylation during single-molecule, real-time sequencing. Nature Methods, vol. 7, pp. 461–465, 2010.

64. Oroojlooyjadid, Afshin, Lawrence Snyder, and Martin Takáč. "Applying Deep Learning to the Newsvendor Problem.", 2016.

65. Vieira, Armando. "Predicting online user behavior using deep learning algorithms.”, 2015.

PUBLICATIONS

- Besaleva L.I., Weaver A.C., “Imbalanced Data Classification via Ensemble Oversampling in e-Commerce,” 3rd IEEE International Conference on Big Data Intelligence and Computing (IEEE DataCom 2017), Nov 6-10, 2017 (accepted)
- Besaleva L.I., Weaver A.C., "Large Scale E-Commerce Applications Data Classification,” The 21st World Multi-Conference on Systemics, Cybernetics and Informatics, July 8-11, 2017
- Besaleva L.I., Weaver A.C., "Imbalanced Data in E-Commerce Networks,” IEEE, Intelligent Systems Conference, September 7-8, 2017
- Besaleva L.I., Weaver A.C., "Large Scale E-Commerce Applications Data Classification,” IEEE International Conference on Big Data Intelligence and Computing, November 6-10, 2017 (pending publication)
- Besaleva L.I., Weaver A.C., "Solutions for Imbalanced Data Analysis in Large-scale E-Commerce,” ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 13-17, 2016
- Besaleva L.I., Weaver A.C., "Applications of Social Networks and Crowdsourcing for Disaster Management Improvement,” IEEE Computer Magazine, May 2016
- Besaleva L.I., Weaver A.C., "CrowdHelp: m-Health Application for Emergency Response Improvement through Crowdsourced and Sensor-Detected Information,” 2014 Wireless Telecommunications Symposium, April 9-11, 2014

- Besaleva L.I., Weaver A.C., "Mobile Electronic Triaging for Emergency Response Improvement Through Crowdsourced and Sensor-Detected Information," 2013 ACM Wireless Health 2013, November 1-3, 2013
- Besaleva L.I., Weaver A.C., "CrowdHelp: Application for Improved Emergency Response through Crowdsourced Information," 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), UbiMi, September 8-12, 2013
- Weaver, A.C., Boyle, J.P., Besaleva, L.I., "Applications and Trust Issues When Crowdsourcing a Crisis," Computer Communications and Networks (ICCCN), 2012 21st International Conference on, vol., no., pp.1,5, July 30 - Aug. 2, 2012
- Textbook of IT Education Based on Invariants for high-school to junior college students, in collaboration with Prof. Sc.D. G. Totkov, UD Dr. R. Doneva