國立成功大學

資訊管理研究所碩士班

碩士論文

混合資料層面與演算法層面技術之集成分類方法

An Ensemble Algorithm with Data-level and Algorithm-level Approaches

研究生：蔡函璇

指導教授：翁慈宗博士

中華民國 111 年 5 月

國立成功大學

碩士論文

混合資料層面與演算法層面技術之集成分類方法
An Ensemble Algorithm with Data-level and Algorithm-level Approaches

研究生：蔡函璇

本論文業經審查及口試合格特此證明

論文考試委員：翁慈宗、利德江、蔡青志

指導教授：翁慈宗

系(所)主管：翁慈宗

（單位主管是否簽章授權由各院、系（所、學位學程）自訂）

中 華 民 國 111 年 5 月 28 日

摘要

不平衡資料中的少數類別往往是大多數研究關心的議題。過去在以訓練單一模型分類不平衡資料的方法，可分為資料層面與演算法層面，但都有著各自的侷限。而結合多個模型的集成方法上，有著基本模型每一個類別正確率都大於 0.5 的條件，若使用一般的學習方法來處理不平衡資料，容易因為少數類別正確率過低而不適合使用集成方法，因此本研究提出一個新的集成方法來解決此問題。過去也已有許多使用集成方法解決不平衡資料的學習問題，本研究是借鑑過去單一模型處理不平衡資料的方法，結合資料層面與演算法層面中三種不同產生基本模型的策略：過取樣、欠取樣與成本敏感方法，依比例來集成三種策略的基本模型成為集成模型，而三種策略有各自的優缺點，期望互補不同策略優缺點，滿足集成方法多樣性越高越好的條件，以達到更好的學習效果。本研究使用 40 個資料集進行實驗，結果也顯示在多數測度上本研究方法的分類性能有顯著優於單一策略的方法，分類性能也與近年提出的 CBWKELM 方法相當，且訓練時間平均只需 CBWKELM 方法不到一成的時間。

關鍵字：集成方法、不平衡資料、過取樣、欠取樣、成本敏感方法

An Ensemble Algorithm with Data-level and Algorithm-level Approaches

Han-Hsuan, Tsai
Tzu-Tsung, Wong
Institute of Information Management
National Cheng Kung University

SUMMARY

Analysts pay their attention on the correct prediction of the instances belonging to the minority class in an imbalanced data set. There are two common approaches for inducing single model to classify imbalanced data: data-level approach and algorithm-level approach. One requirement of ensemble learning, a technique that combines the predictions of multiple base models, is the accuracy of each base model must be greater than 0.5. Since the accuracy on the minority class is generally less than 0.5 for a base model, ensemble learning cannot be applied on classifying imbalanced data directly. This research aims to propose an ensemble algorithm that combines base models induced from oversampling and undersampling strategies belonging to data-level approach and cost-sensitive strategy belonging to algorithm-level approach. The proportions of the base models for the three strategies are set as parameters. The idea of the proposed method is that the base models induced by different strategies should be more diverse. The proposed method is tested on 40 imbalance data sets, and the experimental results show it significantly outperform the algorithm with only single strategy, regardless of the evaluation metric. In comparing with the state-of-the-art algorithm CBWKELM, the algorithm proposed in this study is 10 times faster and achieves an insignificant higher performance.

Keywords: Cost-sensitive strategy, ensemble algorithm, imbalance data, oversampling, undersampling.

INTRODUCTION

In class imbalance problem, conventional classification methods tends to bias toward the majority class, and the output model will misclassify samples of the minority class, which most research is concerned about. Ensemble learning is the popular technique

used to improve the performance of the output model. Instead of using only single base model for prediction, the ensemble model combines multiple base models to make the final result.. However, the accuracy of each class of each base model needs to be greater than 0.5. Otherwise, it is not proper to use on ensemble learning. It is a problem when we use conventional classification algorithm to build a base model, because the accuracy of minority class is most likely to be less than 0.5. The common solution of this is to combine ensemble learning with the modification approaches on single model training: data-level approach or algorithm-level approach, but each approach has its limitation. And this research uses data-level approach and algorithm approach together to increase diversity among base models and make an effective ensemble algorithm for imbalance data.

MATERIAL AND METHODS

In our ensemble algorithm, we use data-level approach and algorithm-level approach to train base models respectively. The concept of data-level approach is to modify class distribution of data, so that it can convert imbalance data to balance data. We use two common methods of this approach: oversampling method and undersampling method. Oversampling method increases the number of minority class, and undersampling method, decreases the number of majority class. The concept of algorithm-level approach is to modify algorithm in order to improve accuracy of minority class. One of popular methods of this approach is cost-sensitive learning, which minority class has more cost than majority class. The idea of our method is to follow the rule that more diversity will improve the performance of ensemble model. We use three strategies to train the base model: oversampling strategy, undersampling strategy and cost-senstive strategy, which are from above two approaches. In addition, we use different proportion to combine base models of these strategies, and only test 13 proportions in our best proportion search algorithm. For convenience, we use bagging method to generate each base model. Therefore, we select SMOTEBagging method as oversampling strategy, underBagging method as undersampling strategy, and CostSensitiveBagging method as cost-sensitive strategy. And our method is to name MSBagging (Mixed Strategy Bagging). In the experiments, we compare our method with 3 single strategy methods, and the state-of-the-art method, CBWKELM, total 5 methods to compare their performance and training time.

RESULTS AND DISCUSSION

We select 40 datasets in the experiments, and three common metrics to evaluate performance: F1, G-mean, and AUC. And we use F1 as the metric to find the best parameter conbination of each method. And then we also use Wilcoxon signed rank test to do the test of significance. As to the result of F1 metric, the first place of average F1 is our MSBagging method, the second is SMOTEBagging, and the third is CBWKELM. After the test of signficance, our MSBagging is significantly better than 3 single strategy methods, but not significantly better than CBWKELM. Although CBWKELM lose on average F1, but it is the first place in some datasets. As to the result of G-mean metric, the first place of average G-mean is MSBagging, and the second is CBWKELM. Its difference on average G-mean is less than difference on average F1, so these are not much different on G-mean. After the test of significance, MSBagging is significantly better than 3 single strategy methods, but not significantly different from CBWKELM. As to the result of AUC metric, the first place of average AUC is UnderBagging, and the second is MSBagging. Since the two methods are only slightly different on average AUC, so they are not much different. After the test of significance on AUC, it shows that the two methods are not significantly different, but MSBagging is significantly better than the other 3 methods. In the result of training time, under all parameter combinations, the fastest method is CostsensitiveBagging, the second is UnderBagging, the third is SMOTEBagging, the fourth is MSBagging, and the slowest method is CBWKELM. On single parameter combination, the third place is MSBagging, the fourth place is SMOTEBagging, because MSBagging is to mix three strategies, so the training time is among three single-strategy methods. Regardless of single or all parameter combinations, the training time of CBWKELM needs 10 more times than MSBagging.

CONCLUSION

Overall, the result of performance in 2 out of 3 metrics, MSBagging performance is significantly better than other 3 single strategy methods. But the performance between MSBagging and CBWKELM has no significant difference. As to training time, our MSBagging only spends 1.26 times than SMOTEBagging which is the most slowest among 3 single strategy methods, and spends less than 10% trainning time of CBKWELM. So MSBagging is effective by mixing three strategies. In the future, our method can be combined with pruning technique to improve the composition of the ensemble model. Or give the different weight of each base model, the better performance model has more weight, to improve the final performance on prediction.

誌謝

時光荏苒，轉瞬間已到了碩士畢業的尾聲。原本對研究懵懵懂懂的我，也完成了自己的碩士論文。首先要感謝的就是我的指導教授，翁慈宗教授。碩一的時候讓我們好好的修課，碩二就專心在論文上，在我迷惘不知道方向的時候，指點我可以進行的方向。又在我很容易將一件事說的很複雜時，老師依然很有耐心的去聽與理解我所說的意思，並給予回饋。老師教導我很多東西，改善了我對研究的思考方式。我很幸運，能夠正好遇到如此好的指導教授。也感謝我論文的審查委員，所給我的寶貴意見，完善我最終的論文。另外我很感謝我哥哥，即使回來的時間不長，但陪我思考研究的方向，又跟我談論做研究的大道理，等我學懂更多的東西，就可以來跟你進行更多的探討。也感謝我的家人默默的支持，在我繁忙時能夠體諒我，一路到我畢業，謝謝你們。還有感謝一同奮鬥論文的夥伴，雖然我們只有 meeting 時候才會到齊，但在聊天之外也交換一些想法，得到了不少收穫，理解別人所做的論文真是很有趣的事，也恭喜我們都完成各自的論文。最後感謝我在資訊所的朋友，在你們那裡打工的日子確實豐富了我碩士的生活，祝你們論文順利。感謝大家，讓我在碩士期間成長，這兩年過著充實的生活，我會帶著這些寶貴的經歷，朝著未來繼續前進。

蔡函璇　謹致於

國立成功大學資訊管理研究所碩士班

中國民國 111 年 7 月

目錄

摘要 I
目錄 VI
表目錄 VIII
圖目錄 IX
第一章 緒論 1
1.1 研究背景與動機 1
1.2 研究目的 2
1.3 論文架構 3
第二章 文獻探討 4
2.1 不平衡資料處理－單一模型 4
2.1.1 資料層面方法 4
2.1.2 演算法層面方法 5
2.2 集成方法 7
2.2.1 集成方法定義與常見方法 7
2.2.2 集成方法於不平衡資料集上 8
2.3 模型衡量測度 11
2.4 小結 14
第三章 研究方法 15
3.1 研究方法流程 15
3.2 決定各策略生成模型的比例與數量 17
3.3 訓練基本模型 20
3.3.1 以過取樣方法訓練基本模型 20
3.3.2 以欠取樣方法訓練基本模型 22

3.3.3 以成本敏感方法訓練基本模型.....23
3.4 生成基本模型的分類方法.....24
3.4.1 決策樹.....24
3.4.2 成本敏感決策樹.....26
3.5 模型評估方式.....26
3.5.1 與近年方法比較之對象.....27
3.5.2 評估測度.....27
第四章 實證研究.....28
4.1 資料集介紹.....28
4.2 MSBagging 不同參數組合比較.....30
4.3 與其他方法進行比較.....33
4.4 訓練時間比較.....42
4.5 小結.....45
第五章 結論與建議.....46
5.1 結論.....46
5.2 未來展望.....47
參考文獻.....48
附錄.....51

表目錄

表 2.1、基本模型間多樣性範例.....7
表 2.2、混淆矩陣.....11
表 3.1、符號表.....16
表 4.1、資料集資訊摘要.....29
表 4.2、MSBagging 的參數範圍.....31
表 4.3、MSBagging 最佳參數組合結果.....32
表 4.4、比較對象的參數範圍.....34
表 4.5、各方法最佳參數組合.....35
表 4.6、各方法的 F1 結果.....37
表 4.7、各方法 G-mean 結果.....38
表 4.8、各方法 AUC 結果.....39
表 4.9、依據 F1 做 Wilcoxon 符號檢定的結果.....41
表 4.10、依據 G-mean 做 Wilcoxon 符號檢定的結果.....41
表 4.11、依據 AUC 做 Wilcoxon 符號檢定的結果.....42
表 4.12、訓練時間(以 MSBagging 為基準的比值).....43
表 4.13、平均單一參數組合時間比值(以 MSBagging 為基準).....44

圖目錄

圖 2.1、ROC 曲線與 AUC 數值來源示意圖 13
圖 3.1、研究方法流程圖 17
圖 3.2、三種策略 1:1:1 集成範例圖 18
圖 3.3、三種策略 1.5:1:1 集成範例圖 18
圖 3.4、三種策略 1:1.5:1 集成範例圖 19
圖 3.5、三種策略 1:1:1.5 集成範例圖 19
圖 3.6、SMOTEBagging 演算法步驟 22
圖 3.7、UnderBagging 演算法步驟 23
圖 3.8、CostSensitiveBagging 演算法步驟 24
圖 3.9、決策樹示意圖 25

第一章　緒論

科技日新月異的時代，資料取得與保存更加地容易，因此資料量隨時間快速的增長，資料內部涵蓋的各式各樣的資訊，以各種形式儲存。如何能夠有效的從中取得有用的資訊，就是資料探勘領域想要解決的課題。而分類是資料探勘內的一條分支，能根據已知類別的資料進行學習，來準確分類未知類別的資料。但在面對千百種資料類型，如何各個擊破，以良好的方法學習如何正確的分類，是一門歷久不衰的學問。

1.1 研究背景與動機

少數案例往往是大多數研究關心的議題，如：罕見疾病、垃圾郵件、材料缺陷、不良品偵測。在資料分布中，這種至少一個類別樣本數量遠遠少於其他類別樣本數量，稱為不平衡資料(imbalance data)。而多數學習方法通常不考慮資料不平衡的現象，導致學習的結果都傾向預測樣本數量較多的類別，以達到最高的準確率，因此許多研究都試圖來克服這個問題。

過往對於訓練單一模型，不平衡資料的處理方法可分為兩種：(1) 資料層面方法(data-level approach)、(2) 演算法層面方法(algorithm-level approach) (Galar et al., 2012; Guo et al., 2017)。資料層面是通過一些手法改變資料的分布，使得少數類別樣本數量與多數類別樣本的數量相當，去除不平衡的現象，但通常改變資料分布的方法會有所侷限性。演算法層面是調整學習的演算法，使演算法重視少數資料，這方面流行的是成本敏感(cost-sensitive)方法，給少數類別資料較重的錯誤分類成本，但成本需要靠經驗來訂定。

集成方法(ensemble method)也是近年來廣泛應用的方法，以數個訓練出來的模型共同進行預測，因此不必每一個模型正確率(accuracy)都高，各模型間可

以互補不足的部分，但在集成方法使用上有兩個條件：(1) 訓練出來的模型多樣性(diversity)越高越好、(2) 模型對每一個類別的正確率大於 0.5 (Dietterich, 2000)。若是使用一般常用於平衡資料集的分類方法來處理不平衡資料，訓練出的模型會因為少數類別正確率小於 0.5，而不適用集成方法。即使使用不平衡資料處理方法訓練模型，也需要各模型之間的多樣性高，也就是讓模型之間對樣本分類的意見越多樣，才能夠展現集成方法的優勢。

1.2 研究目的

因為集成方法的概念並非針對不平衡資料所設計的，因此需要對集成方法進行調整，而目前已經很多使用集成方法來解決不平衡資料問題，與傳統不平衡資料處理方法相同，大多改良的部分可歸納為兩個部分：(1) 資料層面方法、(2) 演算法層面方法 (Guo et al., 2017)。

資料層面方法通過改變資料分布，將其化為多個平衡資料集進行訓練，因為成為平衡資料集就可以達成每個類別正確率大於 0.5 的條件，並且因為多個平衡資料集之間的樣本差異，使得訓練出來的模型之間產生更多的多樣性。演算法層面方法與單一模型處理不平衡資料的方式相似，使用成本敏感方法，為少數類別增加分類錯誤成本，避免直接調整資料分布 (Guo et al., 2017)。

而在集成方法上生成模型的分類方法則可以採用一種或多種方法並用。採用單一種分類方法則依靠不同平衡資料集來產生多樣性，並且根據資料特性的不同，使用不同分類方法會有不同的分類效果；而多種方法並用則可以互補不同分類方法的優缺點達到更高的多樣性，進而有更好的學習效果。

因此本研究希望透過並用資料層面與演算法層面的方法，以此產生更高的多樣性，來建構出一個分類方法有效地解決不平衡資料的學習問題，並使用合理的測度來檢視所提出分類方法的有效性，使得最終所提出的方法是能夠有效

地解決不平衡資料的學習問題。

1.3 論文架構

本研究分為五章：第一章說明本研究的研究背景、動機以及研究目的；第二章探討過去文獻對於不平衡資料的處理方法與搭配集成方法所做的改良；第三章敘述本研究所提出分類方法的細部流程與評估模型的方式；第四章會使用實際的資料集來對提出的分類方法進行驗證，並與近年提出的方法做比較；第五章則會對於本研究的結果進行總結，與提出未來可以進一步研究的方向。

第二章　文獻探討

本研究針對二元不平衡資料進行研究，也就是專注在只有兩個類別的不平衡資料上。在這一章會探討過去文獻針對不平衡資料與集成方法的相關研究。在 2.1 節中探討訓練單一模型時，對不平衡資料採取的策略；2.2 節則會介紹集成方法的概念與其常見的方法，再帶到其對於不平衡資料所採取的調整策略；2.3 節中講述常使用在不平衡資料上衡量模型好壞的測度；最後，2.4 節對於此章做個小結。

2.1 不平衡資料處理－單一模型

不平衡資料定義為一個類別樣本數量遠小於另一個類別的樣本數量，因此在訓練上，一般分類方法容易趨向多數類別樣本來達到正確率最大化，造成少數類別被忽視，但在很多案例中少數類別才是人們真正關心議題。在過去解決不平衡資料的文獻上，以訓練單一模型來說，可分為兩種層面的方法：(1) 資料層面方法、(2) 演算法層面方法 (Galar et al., 2012; Guo et al., 2017)。

2.1.1 資料層面方法

此類方法主要流行是重取樣(resampling)的方法，透過將不平衡資料集類別平衡化，成為平衡資料集再進行訓練。此類方法又可以分成三種：

(1) 欠取樣(undersampling)：透過刪除多數類別樣本來達成平衡資料集的方法。但欠取樣方法可能會導致丟失包含重要訊息的樣本(Raghuwanshi & Shukla, 2018)，導致訓練效果不如預期。

(2) 過取樣(oversampling)：透過合成少數類別樣本的方式來平衡化資料集。此種技術最廣泛使用與衍生的是 SMOTE (Synthetic Minority Oversampling Technique) (Chawla et al., 2002)，透過在兩鄰近少數類別樣

本間插值來生成新的少數類別樣本。通常採取過取樣方法會導致過擬合(overfitting) (Raghuwanshi & Shukla, 2018)，但在只有少量少數類別樣本的情形下，表現會比欠取樣方法好(Guo et al., 2017)。

(3) 混合取樣(hybrid-sampling)：結合過取樣與欠取樣的方式來將資料集平衡化。

此外，也有少數使用特徵選取(feature selection)或特徵提取(feature extraction)的方式來減少因為資料集不平衡所帶來的影響(Guo et al., 2017)。特徵選取會去除一些較無關的屬性，減少少數類別被誤判的風險；而特徵提取方式，則將原本的樣本特徵，透過函數映射成一個新的特徵，再進行訓練，可做強化少數類別樣本特徵，減少多數類別樣本特徵的一個手法。

2.1.2 演算法層面方法

此類方法以成本敏感方法最為流行，在演算法層面對少數類別加入較多的分類錯誤成本，相當於給予少數類別樣本較多的權重，來讓訓練出來的模型趨向於少數類別，提高少數類別分類的正確率。但對於成本多寡大多需要視資料集狀況或是依賴領域的專家決定，才能選擇適當的類別成本來訓練模型(Guo et al., 2017)。此類演算法大多為現有的演算法改良，使之能夠考量分類錯誤的成本。以下介紹幾種常使用的演算法，說明其在不平衡資料上所會遇到的困難，與調整為成本敏感演算法版本的方式：

(1) SVM (Support Vector Machine)：SVM 主要是透過建立目標式來找到一個超平面(hyperplane)，能盡可能分割多數類別樣本與少數類別樣本，並最大化與邊界樣本點的距離。此超平面可投影成一條分隔多數類別樣本與少數類別樣本的決策邊界線(decision boundary)，邊界線的一邊會將待分類樣本預測為多數類別，另一邊則預測為少數類別。在不平衡資料

上，因為多數類別樣本數過多，導致在所得出的決策邊界線會向少數類別靠近，而容易傾向將待分類樣本預測為多數類別(Sun et al., 2009)。在 Iranmehr et al. (2019) 所提出成本敏感 SVM 方法，會在目標式上加入類別分類錯誤成本，並為少數類別訂定較大的分類錯誤成本，使決策邊界線給予少數類別更大的空間，來提高少數類別的預測正確率。

(2) 極限學習機(Extreme Learning Machine, ELM)：極限學習機是單隱藏層前饋神經網路，相較其他神經網路方法，以訓練快速為特點，以透過建立目標式，來找出能達到高正確率的神經網路權重。因為沒有考慮到類別不平衡的情況，因此在不平衡資料，所訓練出來的極限學習機模型會傾向多數類別來讓提高正確率(Zong et al., 2013)。Zong et al. (2013)所提出的權重式極限學習機(Weighted Extreme Learning Machine)，即是使用成本敏感方法，在目標式加入類別分類錯誤成本，找出使總分類錯誤成本最小的神經網路權重，來藉此提高少數類別的正確率。

(3) 決策樹(Decision Tree)：決策樹是以樹狀結構表達分類判斷過程，以屬性值作為分支準則將樣本集合分成多個更小的集合，待分類樣本歷經多次分支準則後到達的葉節點(leaf node)，其對應的類別標籤即為預測的結果。建構決策樹時，每次決定分支準則會先計算樣本集合所包含的資訊量，並找到一個能減少最多資訊量的分支準則，重複動作找尋分支準則的動作，直到無法分支或是樣本皆屬於相同的類別。在不平衡資料上，會需要較多的分支準則才能預測為少數類別，因此預測為少數類別的條件過於嚴格；若有執行剪枝(pruning)來減少決策樹的過擬合時，則少數類別部分的分支容易被裁減掉(Sun et al., 2009)。Ting (2002) 提出成本敏感決策樹的方法，將分支準則改以最小化分類錯誤成本的方式挑選，使得所找到的分支準則能夠重視少數類別樣本，進而提高決策樹對

於少數類別的預測正確率。

2.2 集成方法

解決不平衡資料問題大多都會使用集成方法。在 2.2.1 節會先介紹集成方法概念與常見的方法；2.2.2 節描述集成方法在不平衡資料上所遇到的問題，與過去文獻的解決方法。

2.2.1 集成方法定義與常見方法

集成方法於近年來廣泛應用，此方法概念是由多個訓練出來的模型，稱為基本模型(base model)來共同決定，並且使用上有兩個條件(Dietterich, 2000)：

(1) 各模型對每個類別的預測正確率大於 0.5：各模型不可只針對某些類別正確率較高而其它類別正確率過低，至少每個類別都要預測正確超過一半的數量。

(2) 各模型間的多樣性要高：各模型對於資料分類意見比較多樣，如表 2.1 所示，每個模型分類為類別 α 的樣本都不盡相同，分類為類別 β 的樣本也都不同。若是各模型對樣本預測的結果越不同，越有可能集成正確率更高的模型。

若符合這兩個條件則使用集成方法可以帶來比單一模型更高的正確率，因此成為提高正確率最常使用的方法之一。

表 2.1、基本模型間多樣性範例

模型分類結果	樣本 1	樣本 2	樣本 3	樣本 4	樣本 5
基本模型 1	α	β	α	α	β
基本模型 2	β	β	β	α	β
基本模型 3	α	β	α	β	β

目前集成方法流行的有三種方法：

(1) 袋裝法(Bagging) (Breiman, 1996)：此方法是在原始資料集多次重複抽樣，形成多個資料集，資料集個別進行訓練，每次生成資料集間的關係是獨立的，因此可以以平行運算的方式訓練模型。

(2) 提升法(Boosting) (Freund, 1995)：此方法在每一次訓練模型後，記錄下分類錯誤的樣本，分類錯誤的樣本在下一輪訓練模型時，就比較容易被挑中加入訓練，最後各模型以正確率較高的模型權重較大來進行整合。因為每一個模型的預測結果會影響下一輪訓練的資料集，而無法以平行運算進行每個模型的訓練。此方法最廣泛使用的方法是 AdaBoost (Guo et al., 2017)。

(3) 堆疊法(Stacking) (Wolpert, 1992)：此方法是主要是對每個模型預測出來的模型結果稱為詮釋資料(metadata)，將詮釋資料作為新的資料集進行訓練，訓練出一個模型來整合前面各個模型的意見。

集成方法也可以再做一次集成，在 Liu et al. (2009) 提出透過袋裝法訓練多個 AdaBoost 模型再集成的稱為 EasyEnsemble；以提升法方法訓練多個 AdaBoost 模型再集成稱為 BalanceCascade。

2.2.2 集成方法於不平衡資料集上

不平衡資料集上使用一般傳統的方法訓練，為求快速達到最高正確率而容易傾向多數類別，因此導致少數類別正確率小於 0.5，就不適合使用集成方法，而且不論是使用袋裝法、提升法或是堆疊法都會發生，勢必得在方法上做調整。與單一模型處理不平衡資料的方法相似，調整的方法主要可分為兩類(Guo et al., 2017)：

(1) 資料層面方法之搭配

相當於集成方法與不平衡資料處理上的資料層面方法做結合。從 Guo et al. (2017) 對於集成方法的整理可知，文獻上多數是採用與提升法結合的方式來建構應用在不平衡資料上的集成方法。欠取樣方法中，隨機欠取樣(Random Undersampling)是以隨機選擇要去除的多數類別樣本的方式來平衡化資料集，與袋裝法結合成為 RUSBagging，又稱 UnderBagging (Wang & Yao, 2009)，與提升法結合就成為 RUSBoost (Seiffert et al., 2010)。另外也有以不同的挑選標準進行欠取樣，如 Hou et al. (2019)提出的方法是以密度的挑選標準，減少密度較高的多數類別區域以達成平衡資料集的集成方法。

因為集成方法可以結合多個模型，因此在欠取樣方法中，有將多數類別樣本分割多個集合，並與少數類別樣本合併再各別訓練成模型。如有將多數類別樣本直接依照少數類別樣本個數分割成多個樣本集合，再個別與少數類別合併，以形成平衡資料集來訓練出基本模型(Sun et al., 2015; Wang et al., 2017)；有基於分群(clustering)的集成方法，將多數類別樣本分割成多個群體再與少數類別合併做訓練，此方法有將多數類別分群成平衡資料集的分類方法(Ng et al., 2020; Zhao et al., 2016)，也有保留不平衡狀態，再後續用成本敏感方法來訓練的分類方法(Choudhary & Shukla, 2021)。

過取樣方法中，使用 SMOTE 方法與袋裝法結合成為 SMOTEBagging (Wang & Yao, 2009)，與提升法結合就變為 SMOTEBoost (Chawla et al., 2003)。也有以不同合成樣本的方式，如 Razavi-Far et al. (2021)提出的方法是以填充遺失值的方式來合成樣本。

此外，有用新測度做為判斷資料集是否平衡的標準來執行欠取樣或是過取樣，如 Tang and He (2017)提出的方法會以資料集多數類別與少數類別樣本的分布來判斷資料集是否平衡，在依此測度來平衡化資料集再做訓練。在實際案例上則有採用特徵提取與集成方法結合的方式來訓練模型(Li

et al., 2021)。

(2) 演算法層面的調整

在調整演算法層面的方法上，與單一模型處理不平衡資料的概念相同，流行成本敏感的分類方法，一樣是給予少數類別更多的分類錯誤成本，使訓練結果能重視少數類別。而在集成方法上，因為是集成多個模型，因此可直接採用成本敏感的分類方法，來生成基本模型，進而避開不平衡資料所帶來的問題(Guo et al., 2017)。

除了直接採用成本敏感的分類方法外，也有將 AdaBoost 演算法稍作調整，加入類別成本的考量，對少數類別分類錯誤的成本更重，因此趨向選擇分類錯誤的少數類別樣本加入訓練(Sun et al., 2007)。

而在不平衡資料中，集成模型中選擇用來生成基本模型的方法，從 Guo et al. (2017)統計常用的順序為 SVM、神經網路(Neural Networks)、單純貝氏分類器(Naïve Bayesian Classifier)、基於規則的分類方法、基於決策樹的方法，以這幾種最常使用。這些常使用的方法有各自的優缺點，在實務上需要挑選能夠適合的方法來產生基本模型，例如：SVM 雖然穩定且精確，但對於遺失值敏感，且難以在大型資料集上做訓練；而決策樹則可以很好的處理含有遺失值的資料，但可能在小型資料集上的表現較差(Guo et al., 2017)。

另外，一般集成方法會使用一些手法來提升正確率，這些手法在不平衡資料上也有效果，例如多樣性提升方法或是動態選擇方法(dynamic selection method)：

(1) 多樣性提升方法

像是使用異質結構，也就是用不同分類方法來生成基本模型，而使用相同分類方法生成基本模型稱為同質結構。在 Zefrehi and Altincay (2020)

的實驗結果中，異質比同質結構要好。在 Díez-Pastor et al. (2015) 中，集成方法配合使用特徵映射(feature mapping)與隨機預言機(random oracle)的多樣性提升方法在不平衡資料上取得不錯的結果。

(2) 動態選擇方法

動態選擇是依據待分類樣本從已有的訓練模型選擇一到多個適合的來進行預測。此方法可以化為集成規則上的調整，在分類樣本時才去評判待分類樣本與集成方法各模型中的匹配距離，越匹配的模型就權重越大，越不匹配就權重越小，依此來進行權重投票(Wang et al., 2017)。

目前也已有許多研究將集成模型應用在不平衡資料集上，如：乳癌復發辨識(Yang et al., 2021)、信用風險識別(Li et al., 2021)、軟體缺陷偵測(Chen et al., 2018)等等。

2.3 模型衡量測度

模型評估上若使用傳統的預測正確率，無法呈現是否有傾向預測多數類別的情形。為了在不平衡資料集上辨別模型的好壞，會將二元類別與實際類別和預測類別情況分為四個部分統計個數，這個結果稱為混淆矩陣(confusion matrix)，如表 2.2 所示：

表 2.2、混淆矩陣

		預測類別	
		少數類別	多數類別
實際類別	少數類別	TP (True Positive)	FN (False Negative)
	多數類別	FP (False Positive)	TN (True Negative)

由此可延伸出幾種常見的測度：

(1) F-measure

$$F-measure(\beta)=\frac{1+\beta^2}{\frac{\beta^2}{召回率}+\frac{1}{精確度}}=\frac{(1+\beta^2)\times 召回率\times 精確度}{召回率+\beta^2\times 精確度}$$

$$F1=F-measure(1)=\frac{2}{\frac{1}{召回率}+\frac{1}{精確度}}=\frac{2\times 召回率\times 精確度}{召回率+精確度}$$

$$召回率(Recall)=\frac{TP}{TP+FN}$$

$$精確度(Precision)=\frac{TP}{TP+FP}$$

召回率代表實際為少數類別的樣本中正確預測為少數類別的比率，精確度為預測為少數類別的樣本中實際也為少數類別的比率，F-measure 為模型分類性能考慮這兩個指標，調和平均這兩個數值，期望分類性能較好的模型這兩個數值都高，並以 β 調整這兩個指標的相對權重(Fernández et al., 2018)。β 小於 1 就會為加重精確度的權重，β 大於 1 則加重召回率的權重，若 β 等於 1，則召回率與精確度的權重相同，即為常使用的 F1 測度。

(2) G-mean

$$G-mean=\sqrt{TPR\times TNR}$$

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

$$TNR=\frac{TN}{TN+FP}$$

TPR 代表少數類別正確率，TNR 代表多數類別正確率，G-mean 平衡這兩個指標，取兩者的幾何平均數，若是只有多數類別樣本正確率高但少數類別樣本正確率低是無法取得較好的 G-mean 值，因此 G-mean 期望分類性能

較好的模型是此二者都高的。G-mean 也有另一種的計算方式，使用召回率與精確度來取幾何平均數，屬於 F1 的變體(Fernández et al., 2018)。

(3) AUC (Area Under Curve)

圖 2.1 為 AUC 數值來源示意圖，TPR 是少數類別預測正確率，FPR 是多數類別預測錯誤率，AUC 是由 TPR 為 Y 軸，FPR 為 X 軸，計算若是由不同的預測機率臨界值來決斷最終為多數類別或少數類別。左下角的起點為全部預測為多數類別樣本，到右上角為全部預測為少數類別樣本，一個樣本從預測為多數類別改為少數類別只會增加 TPR 或 FPR 其中一個數值，依此繪製出的曲線，稱為 ROC 曲線(Receiver Operating Characteristic curve)，AUC 即為 ROC 曲線下方的面積值。AUC 與其他測度不同，不預設 0.5 作為預測閥值，而是當 AUC 等於 1 時，左上角的點即為能夠完美分隔少數與多數類別的預測閥值，因此 AUC 期望分類性能較好的模型是 ROC 曲線能夠越靠近左上角，分隔少數與多數類別的預測閥值越明顯則分類性能越好(Fernández et al., 2018)。

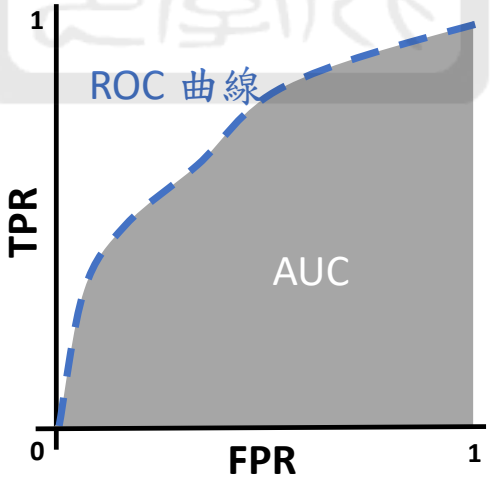

圖 2.1、ROC 曲線與 AUC 數值來源示意圖

$$FPR = \frac{FP}{TN + FP}$$

在 Guo et al. (2017) 整理在不平衡資料上前四大項會使用的測度順序為 AUC、正確率或錯誤率(Error rate)、G-mean、F-measure。

2.4 小結

綜合上述文獻可知，集成方法是能有效提升正確率的方法，並且提升集成方法學習結果的技術，也適用於不平衡資料。而目前解決不平衡最通用的方法，屬於資料層面方法與演算法層面方法，因此集成方法搭配這兩種層面都能夠有效的解決不平衡資料的學習問題。並且使用異質的方法可以有效地增加多樣性，進而提升集成模型的分類性能。目前已有文獻以先使用資料層面方法處理資料後，再使用演算法層面的方法訓練模型，但尚未有文獻將資料層面方法與演算法層面方法兩種不同層面各別產生基本模型來集成，因此本研究希望並用資料層面與演算法層面方法，以此建構一個異質的集成分類方法，來有效地解決不平衡資料的學習問題。

第三章　研究方法

從第二章文獻探討中可以知道，在過去解決二元不平衡資料集分類問題的文獻中，並沒有研究嘗試以資料層面方法與演算法層面方法各別產生基本模型，來進行異質集成。因此本研究以此想法建構一個分類方法，混合資料層面與演算法層面的三種策略，並將方法命名為 MSBagging (Mixed Strategy Bagging)，在本章對方法的流程做詳細說明。

本章在 3.1 節先說明本研究提出的研究方法流程；3.2 節講述如何決定依比例集成三種策略中的基本模型；3.3 節說明本研究三種訓練基本模型的策略細節；3.4 節則介紹所選擇來生成基本模型的分類方法；3.5 節介紹本研究模型評估的方式。表 3.1 定義與說明本章會使用到的符號。

3.1 研究方法流程

圖 3.1 為本研究提出的 MSBagging 的研究方法流程，先將二元不平衡資料集劃分成訓練集與測試集，訓練集將會使用三種不同的策略產生多個基本模型：過取樣方法、欠取樣方法與成本敏感方法。這三種涵蓋資料層面與演算法層面方法，這三種策略會被決定以某個比例來分配所要訓練的模型個數，再使用各別策略訓練出各自數量的基本模型，依此集成得出最終的集成模型。最終的集成模型以測試集來做效能評估。

表 3.1、符號表

符號	說明
$\boldsymbol{D}$	資料集
$\boldsymbol{D_{Train}}$	從資料集劃分出的訓練集
$\boldsymbol{D_{SubTrain}}$	從訓練集中選取的子訓練集，用來訓練模型
$\boldsymbol{Minority_{Train}}$	訓練集中少數類別樣本集合
$\boldsymbol{Majority_{Train}}$	訓練集中多數類別樣本集合
$\boldsymbol{M}$	基本模型
$\boldsymbol{N_M}$	生成目標基本模型數
$\boldsymbol{k}$	kNN 方法的參數，決定要取得多少個最鄰近點
$\boldsymbol{U}$	基本模型集合
$\boldsymbol{x}$	一個樣本
$\boldsymbol{x_c}$	SMOTE 方法中選取的中心樣本
$\boldsymbol{x_{near}}$	SMOTE 方法所選取與 x_c 鄰近的樣本
$\boldsymbol{x_{new}}$	合成的新樣本
$\boldsymbol{N_{gen}}$	合成樣本目標數量
$\boldsymbol{a}$%	Boostrap 方法中抽樣的比例
$\boldsymbol{b}$%	SMOTEBagging 方法用於建造子訓練集的合成樣本比例
$\boldsymbol{N_C}$	資料集中類別數量
$\boldsymbol{S}$	樣本集合
$\boldsymbol{S_i}$	從 S 中分割的第 i 個樣本子集合
$\boldsymbol{p(i\|S)}$	樣本集合 S 中，類別 i 的比例
$\boldsymbol{\mathrm{cost}(i,j)}$	預測為類別 i，但實際為類別 j 的分類錯誤成本
$\boldsymbol{C_{Min}}$	表示少數類別
$\boldsymbol{C_{Maj}}$	表示多數類別

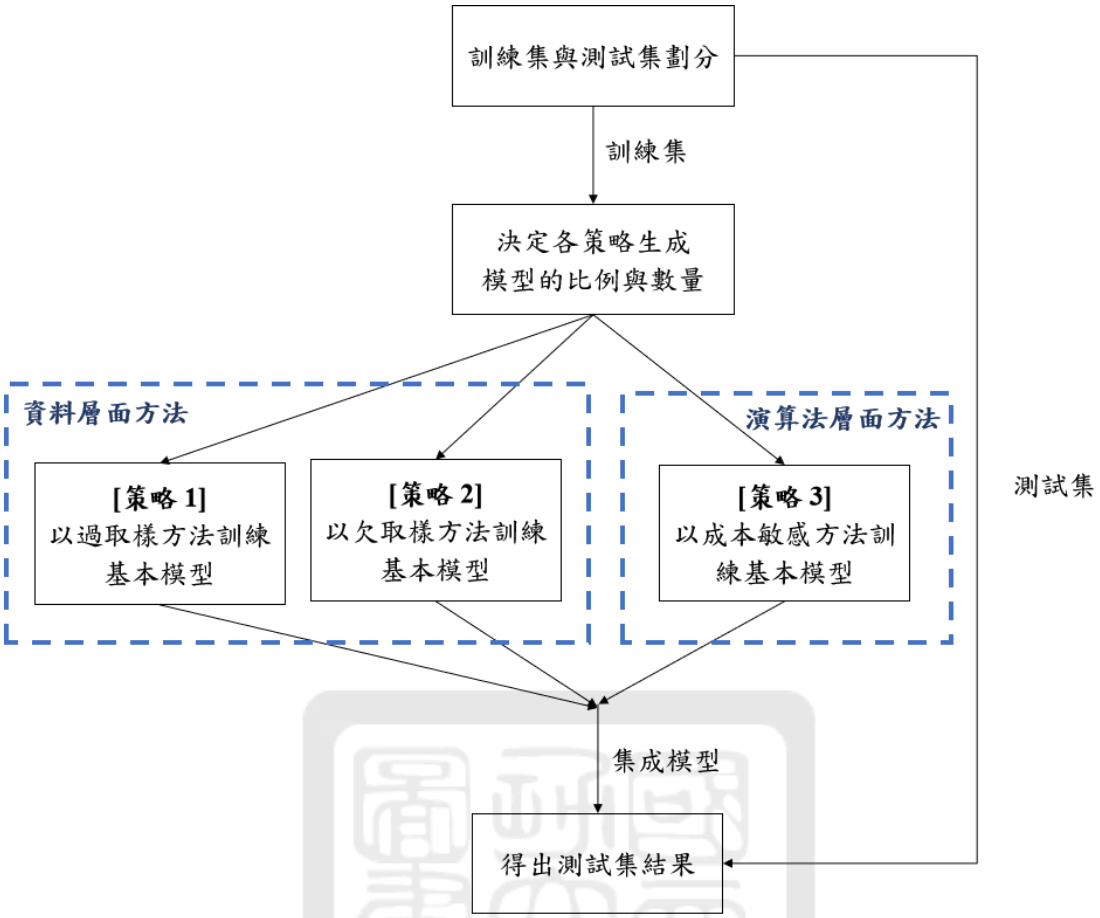

圖 3.1、研究方法流程圖

3.2 決定各策略生成模型的比例與數量

本研究方法會嘗試選取不同比例，讓過取樣、欠取樣與成本敏感三種策略生成特定數量的基本模型來進行集成，以達到較好的預測結果。而特定數量集成可以分為基底模型數與集成比例兩個參數，集成比例以過取樣、欠取樣與成本敏感的順序表示各策略訓練模型數量的比重，乘上基底模型數即為實際訓練的模型數量。以下給出集成比例會嘗試的範例，並給出嘗試的理由：

(1) 三種策略等比例集成：每一種策略可能都具有各自的特色以及足夠多的多樣性，相同比例足以達到更好的結果。以圖 3.2 為例，三種策略以 1:1:1 的集成比例，設定基底模型數為 20，則每一種策略都生成 20 個基

本模型，最後集成為有 60 個基本模型集成模型。

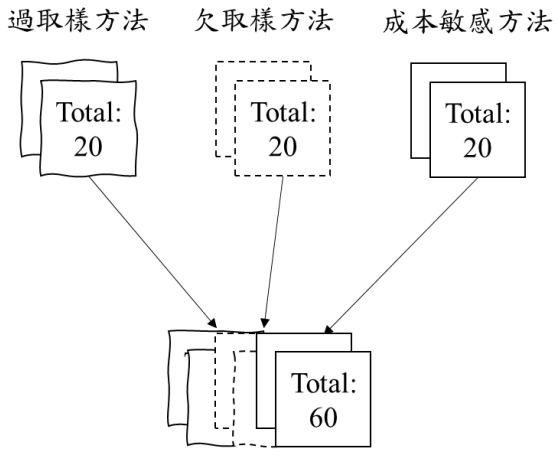

圖 3.2、三種策略 1:1:1 集成範例圖

(2) 三種策略以過取樣比重較大來集成：在第二章提過，過取樣有著過擬合的風險，增加過取樣模型的比重，可能達到更高的預測正確率。以圖 3.3 為例，以 1.5:1:1 的集成比例，設定基底模型數為 20，則過取樣基本模型 30 個，其餘 20 個，最終集成為 70 個基本模型的集成模型。

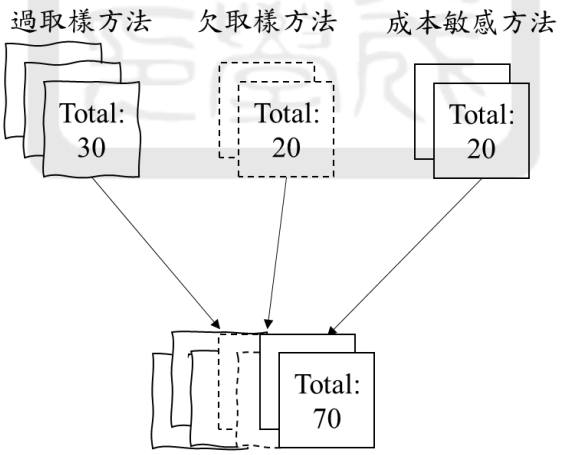

圖 3.3、三種策略 1.5:1:1 集成範例圖

(3) 三種策略以欠取樣比重較大來集成：在第二章提過，欠取樣有著遺失重要資訊的風險，反而因此有更多的多樣性，以更高的多樣性比重可能達到更好的集成結果。以圖 3.4 為例，使用 1:1.5:1 的集成比例，設定基底

模型數為 20，則欠取樣為 30 個模型，其餘 20 個模型，集成為有 70 個基本模型的集成模型。

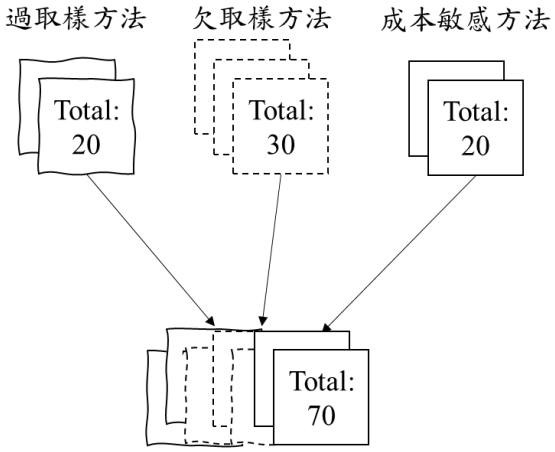

圖 3.4、三種策略 1:1.5:1 集成範例圖

(4) 三種策略以成本敏感比重較大來集成：在第二章提過，過取樣與欠取樣會有過擬合或是遺失重要資訊的風險，因此以較為穩定的成本敏感方法比重較多，可能帶來較好又穩定的結果。以圖 3.5 為例，使用 1:1:1.5 的集成比例，設定基底模型數為 20，則成本敏感為 30 個模型，其餘 20 個模型，集成為有 70 個基本模型的集成模型。

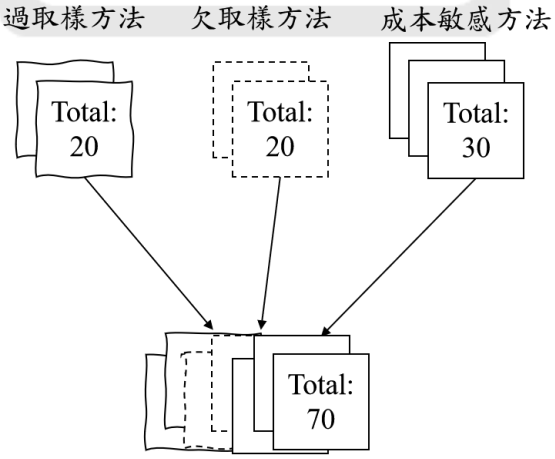

圖 3.5、三種策略 1:1:1.5 集成範例圖

並且可以有多個策略比重大於 1，因此本研究方法的集成比例可以有無限多種，為避免此情形，本研究限制每個策略的比重從 $\{1, 1.25, 1.5, 1.75, 2\}$ 五種中選取。若三種策略都試驗 5 種會讓組合後有 125 種可能，再乘上基底模型數量會測試的個數，則會讓測試的組合數過多，而花費較長的時間搜尋最佳參數組合，因此為了進一步縮小搜尋數量，本研究調整搜尋方法，以找出較佳的集成比例。從 1:1:1 等比例開始，先調整第一部份過取樣策略五種不同的比重，從五種可能中選擇最佳的過取樣比重；再調整第二部份欠取樣策略，多嘗試除了 1 以外的四種比重，再從五種不同比重選取最佳的欠取樣策略比重；最後調整第三部份成本敏感策略，多嘗試除了 1 以外的四種比重，再從五種不同比重選取最佳的成本敏感比重。相比搜尋 125 種可能組合來找到最佳集成比例，依此方法只要測試 $5 + 4 + 4 = 13$ 種組合，就可得到較佳的集成比例，大幅縮小搜尋所花費的時間。

3.3 訓練基本模型

本研究為了方便依比例集成、減少訓練時間並且保有基本模型之間的多樣性，在三種不同生成基本模型的策略上，會各別與袋裝法做結合來產生基本模型。接下來對所使用的三種訓練基本模型的策略，每一個部份再做細部說明。

3.3.1 以過取樣方法訓練基本模型

如第二章所述，過取樣使用合成少數類別樣本的方法平衡化資料集，而過取樣方法以 SMOTE 為經典方法，本研究也使用 SMOTE 方法進行樣本合成。SMOTE 方法是先從少數類別樣本選取一個樣本 x_c，以 x_c 為中心再透過 kNN 得到 k 個鄰近點，從鄰近點中任選一個樣本 x_{near}，新樣本 x_{new} 每一個屬性以在 x_c 與 x_{near} 之間隨機選取一點來生成，以此重複運作生成到目標的合成樣本數量 N_{gen}。

上述為 SMOTE 原始概念，也只適用於連續屬性，而對於離散屬性原始 SMOTE 論文也有 SMOTE-NC (Nonimal and Continuous)方法來處理，其對於連續屬性一樣以 SMOTE 來處理，而離散屬性則以鄰近點中佔多數的屬性值來合成。考慮到資料集含有離散屬性的可能，本研究預設使用 SMOTE-NC 方法來代表 SMOTE 方法。

SMOTE 與袋裝法的結合方法為 SMOTEBagging (Wang & Yao, 2009)，原本為多類別版本，因本研究針對二元類別，而改成二元類別版本，圖 3.6 為本研究所使用的演算法步驟，以下說明有使用到兩個函式：

(1) Bootstrap$(D, a\%)$：從資料集 D 中，執行可放回重複抽樣，抽出共 $|D| \times a\%$ 個樣本，回傳抽出的樣本集合。

(2) SMOTE$\left(k, D, N_{gen}\right)$：從資料集 D 中，執行 SMOTE 步驟中以 k 為參數的 kNN 找到鄰近的 k 個樣本然後合成樣本，回傳所生成的N_{gen}個合成樣本。

SMOTEBagging 方法是在每一次從訓練集 D_{Train} 產生平衡的子訓練集 $D_{balanced}$ 時，將多數類別樣本以可放回重複抽樣抽出相同數量，來增加子訓練集之間多數類別樣本的不同程度；少數類別樣本則由兩個來源混合至與多數類別數量相當的樣本數：(1) $b\%$ 從少數類別樣本中執行可放回重複抽樣、(2) $1 - b\%$ 使用 SMOTE 合成少數類別樣本。這兩個來源可以依照設定不同 b 數值方式組合，來增加子訓練集之間少數類別樣本的不同程度，進而增加所訓練出來的基本模型之間的多樣性。以此重複運作生成子資料集與訓練出基本模型 M，直到生成目標 N_M 個基本模型。

Algorithm 1: SMOTEBagging(D_{Train}, k, N_M)

Input:

D_{Train}：訓練集

k：SMOTE 使用 kNN 的參數

N_M：模型數量

Output:

U：模型集合

初始化模型集合 $U = \{\}$；

將資料集 D_{Train} 區分為多數類別集合 $Majority_{Train}$ 與少數類別集合 $Minority_{Train}$；

for $i = 1$ **to** N_M **do**

  設定合成樣本比例 b；

  $Majority_{SubTrain} = Bootstrap(Majority_{Train}, 100\%)$；

  少數類別重複抽樣比率 $a_{Min} = \frac{|Majoirty_{Train}|}{|Minoirty_{Train}|} \times b\%$；

  合成樣本個數 $N_{gen} = |Majority_{Train}| \times (1 - b\%)$；

  $Minority_{SubTrain} = Bootstrap(Minoirty_{Train}, a_{Min}) \cup SMOTE(k, Minority_{Train}, N_{gen})$；

  平衡子訓練集 $D_{SubTrain} = Majority_{SubTrain} \cup Minority_{SubTrain}$；

  基本模型 $M = TrainingMethod(D_{SubTrain})$；

  $U = U \cup \{M\}$；

end

圖 3.6、SMOTEBagging 演算法步驟

3.3.2 以欠取樣方法訓練基本模型

如第二章所述，欠取樣是以減少多數類別樣本來平衡化資料集，與袋裝法集成有稱為 UnderBagging (Wang & Yao, 2009)的方法，原本為多類別版本，針對本研究改為二元類別版本，圖 3.7 為本研究所使用的演算法步驟，其中所用到的 Bootstrap 函式，已在 3.3.1 節提到過。

UnderBagging 方法是每次以訓練集 D_{Train} 生成子訓練集 $D_{SubTrain}$ 時，少數類別樣本以可放回抽出相同數量，來增加子訓練集之間少數類別樣本的不同程度；多數類別以可放回抽出與少數類別數量相當的樣本數，兩者合併成平衡的子訓練集 $D_{SubTrain}$ 來訓練基本模型 M，重複運作直到生成目標 N_M 個基本模型。

Algorithm 2: UnderBagging(D_{Train}, N_M)

Input:
D_{Train}：訓練集
N_M：模型數量
Output:
U：模型集合

初始化模型集合 $U = \{\}$；
將資料集 D_{Train} 區分為多數類別集合 $Majority_{Train}$ 與少數類別集合 $Minority_{Train}$；
for $i = 1$ **to** N_M **do**
    $Minority_{SubTrain} = Bootstrap(Minority_{Train}, 100\%)$；
    多數類別重複抽樣比率 $a_{Maj} = \frac{|Minority_{Train}|}{|Majority_{Train}|}$；
    $Majority_{SubTrain} = Bootstrap(Majoirty_{Train}, a_{Maj})$；
    平衡子訓練集 $D_{SubTrain} = Majority_{SubTrain} \cup Minority_{SubTrain}$；
    基本模型 $M = TrainingMethod(D_{SubTrain})$；
    $U = U \cup \{M\}$；
end

圖 3.7、UnderBagging 演算法步驟

3.3.3 以成本敏感方法訓練基本模型

成本敏感方法是以設定樣本分類錯誤成本並使分類錯誤成本最小化。圖 3.8 為演算法步驟，其中所用到的 Bootstrap 函式，已在 3.3.1 節提到過。本研究每次以來先從訓練集 D_{Train} 可放回重複抽樣抽出 $|D_{Train}|$ 個樣本作為子訓練集 $D_{SubTrain}$ 來增加子訓練集之間樣本的不同程度，再以成本敏感分類方法 $CostSensitiveMethod$ 配合預先設定的樣本權重 W，樣本權重相當於樣本分類錯誤成本，依此來訓練出一個基本模型 M，重複運作直到生成目標 N_M 個基本模型。

Algorithm 3: CostSensitiveBagging(D_{Train}, N_M, W)

Input:

D_{Train}：訓練集

N_M：模型數量

W：樣本權重 (分類錯誤成本)

Output:

U：模型集合

初始化模型集合 $U = \{\}$；

for $i = 1$ **to** N_M **do**

  $D_{SubTrain} = Bootstrap(D_{Train}, 100\%)$；

  基本模型 $M = CostSensitiveMethod(D_{SubTrain}, W)$；

  $U = U \cup \{M\}$

end

圖 3.8、CostSensitiveBagging 演算法步驟

3.4 生成基本模型的分類方法

在生成單一的基本模型的分類方法上，本研究在資料層面方法使用一般分類方法(無分類錯誤成本方法)，在演算法層面方法使用成本敏感的分類方法(有分類錯誤成本方法)，來訓練出基本模型。

本研究中，使用決策樹來生成基本模型，決策樹具有不穩定的特性，因此可以建構較具多樣性。本研究在資料集層面方法使用經典的決策樹方法；在演算法層面中則是使用成本敏感版本的決策樹(Ting, 2002)。

3.4.1 決策樹

決策樹以樹狀結構表達分類判斷過程，由根節點(Root)開始，對一個屬性選取分支準則向下長，若其中一支要繼續分支就再選一個屬性分支準則，繼續向下延伸，此非根節點的分支節點稱為內部節點；若不再分支，則節點代表最終的分類結果，稱為葉節點。

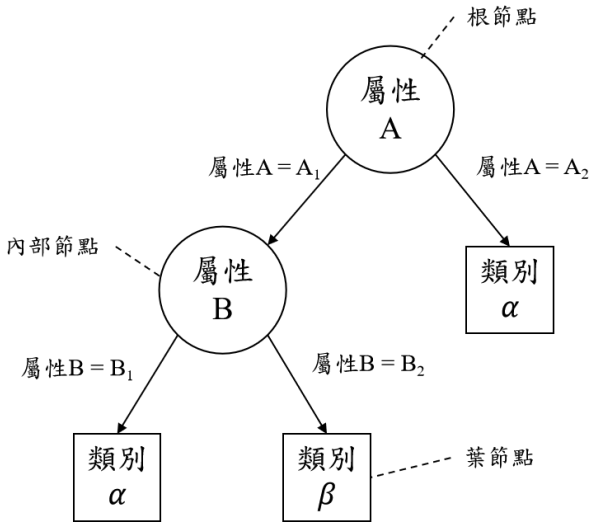

圖 3.9、決策樹示意圖

以選取屬性分支準則方式的不同，而有各種不同決策樹方法。在各類的決策樹方法中，本研究選擇了運算速度較快的 CART (Classification and Regression Trees) 演算法，其在分支準則上以 Gini 不純度(impurity)為基礎，再計算出信息增益(Gain)作為選擇分支準則的指標：

$$\text{Gini}(S) = 1 - \sum_{i=1}^{N_C} p(i|S)^2$$

$$\text{Gain}(A,S) = \text{Gini}(S) - \sum_{j=1}^{m} \frac{|S_j|}{|S|} \cdot \text{Gini}(S)$$

S 為資料集合，其中包含 N_C 個類別，$p(i|S)^2$ 代表類別 i 在 S 中的比例，而 *Gini* 用來表示 S 集合類別分布的不純度，不純度越高則類別分布較雜不好分類，越低就表示類別分布越單純較好分類。而 Gain 用來表示當使用屬性 A 做為分支標準時，分支後的 $S_1 \sim S_m$ 分支子集合，去除多少 S 的類別分布雜訊，Gain 越大代表去除雜訊越多，屬性 A 越適合做為分支準則，反之則不適合。

剪枝是決策樹中去除過擬合的手法，可增加決策樹穩定性，為保留決策樹的不穩定性，在本研究不進行剪枝。

3.4.2 成本敏感決策樹

本研究所選擇成本敏感版本的決策樹方法，是對每一個樣本的分類錯誤成本設定為類別成本，以分類錯誤成本最小化來進行每一輪分支的屬性挑選，使得最終決策樹是趨向少數類別。為保持決策樹不穩定性，剪枝在成本敏感決策樹一樣不使用。成本敏感版本上，首先要設定成本矩陣 $\text{cost}(i,j)$：

$$\text{cost}(i,j)\begin{cases}=0 & \text{if } i=j\\ >0 & \text{if } i\neq j\end{cases}$$

$\text{cost}(i,j)$ 代表被預測為類別 i 的樣本，真實為類別 j 的分類錯誤成本。若 $i=j$ 表示沒有分類錯誤，分類錯誤成本為 0；若是 $i\neq j$，則有大於 0 的分類錯誤成本，一般以類別樣本數的比例來設定，如多數類別 C_{Maj} 與少數類別 C_{Min} 比例為 5:1，則多數類別樣本預測錯誤成少數類別的成本 $\text{cost}(Min,Maj)$ 為 1，而少數類別樣本預測錯誤成多數類別的成本 $\text{cost}(Maj,Min)$ 為 5。接下來以最小化分類錯誤總成本 z，來決定分支準則：

$$minimize \quad z=\sum_{i=1}^{|D|}\text{cost}\big(\text{predictClass}(x_i),\text{actualClass}(x_i)\big)$$

x_i 為資料集 D 中的樣本，而分類錯誤總成本 z 即為每一個樣本分類錯誤成本的總和。以此公式即可在每一個節點選擇總成本最小的屬性作為分支準則，來建構成本敏感的決策樹。

3.5 模型評估方式

為了證明本研究方法的有效性，先探討本研究方法搜尋最佳參數組合的結果，再與單純使用每一種策略比較，並與近年提出的集成方法進行比較。本節分為兩部分，先介紹所選擇進行比較的近年集成方法，再說明所選擇用來評估模型分類性能的測度。

3.5.1 與近年方法比較之對象

要比較的近年集成方法選擇 Choudhary and Shukla (2021)提出的 CBWKELM 方法，CBWKELM 會先以 FCM (Fuzzy C-Means)方法得到每個樣本對每個群集的隸屬程度，利用 TH 作為閥值，隸屬程度大於 TH 值的樣本加入對應的群集，若一樣本對多個群集隸屬程度大於 TH，就會被加入多個群集，因此多個群集之間可能有重複的樣本，然後一個群集作為一個子訓練集以訓練出一個權重式極限學習機的模型，此權重式極限學習機方法是利用設定一個參數 C 作為真實與預測結果誤差在成本目標式的比重，與一個參數 σ 來生成一個高斯核心(Gaussian Kernel)，套用成本目標式最小化後的公式即可計算最佳解。

從上述可知 CBWKELM 是使用分群(FCM 方法)這種資料層面方法帶來模型之間多樣性，而後使用成本敏感方法(權重式極限學習機)進行訓練，而本研究是資料層面與成本敏感方法各別訓練模型後再結合，因此選擇此方法作為本研究的比較對象。

3.5.2 評估測度

在測度方面，在第二章提過不平衡資料常用的前四種測度為 AUC、正確率、G-mean、F-measure，但正確率並不適合用來評估不平衡資料，因此選擇較常用的 AUC、G-mean、F-measure 三種測度，納入本研究實驗的評估測度，以佐證本研究方法的有效性。本研究 F-measure 採用常用的 F1 測度；而 G-mean 是使用與精確率與召回率相乘再開根號的版本，來評估模型的分類性能：

$$G-mean=\sqrt{召回率\times 精確度}$$

第四章　實證研究

本章對第三章所提出的方法做實證研究，並使用 Python 3.9 來進行實作。首先 4.1 節中介紹實驗所使用的資料集；4.2 節來說明本研究的 MSBagging 方法所找到的最佳參數組合；4.3 節則是將 MSBagging 方法與其他方法進行比較；4.4 節對比較所有參與實驗的方法所花費的訓練時間；4.5 節對於本章所做的實驗做個小結。

4.1 資料集介紹

表 4.1 列出本研究實驗所使用資料集的相關資訊，IR (imbalanced ratio)表示二元資料集的類別不平衡程度：

$$\text{IR} = \frac{\text{多數類別樣本數}}{\text{少數類別樣本數}}$$

IR 數值越大即多數類別樣本所占比例越大，資料集的類別不平衡程度就越高，訓練模型時受到多數類別樣本的影響就越大；反之，IR 數值越接近 1，則資料集的類別不平衡程度就越低，訓練模型時就不容易被多數類別樣本所影響。

本研究資料集從 KEEL (Alcala-Fdez et al., 2011)、UCI (Dua & Graff, 2019)、Zenodo (Lemaitre, 2016)挑選 40 個二元資料集，IR 範圍介於 8.60 ~ 42.01，樣本數從 336 筆到 20000 筆。所選的資料集大多數原本並不是二元資料集，是從多類別資料集中選擇一部分類別作為少數類別，一部分作為多數類別，以此轉換成一個二元資料集，每個資料集所挑擇少數類別與多數類別的方式也呈現在表 4.1 中。

在本研究的實驗中，為了有效利用所有可訓練的資料，因此會以 5 等分交叉驗證法(5-fold cross validation)將資料集劃分五等分，每次取四等分作為訓練

表 4.1、資料集資訊摘要

資料集名稱	從原始資料集挑選的類別 少數類別 v.s. 多數類別	樣本數	IR	來源
abalone1	7 v.s. 其餘類別	4177	9.68	Zenodo
abalone2	17 v.s. 7,8,9,10	2338	39.31	KEEL
abalone3	18 v.s. 9	731	16.40	KEEL
car1	good v.s. 其餘類別	1728	24.04	KEEL
car2	vgood v.s. 其餘類別	1728	25.58	KEEL
coil_2000	minority v.s. 其餘類別	9822	15.76	Zenodo
drug_comp	Mushrooms = CL4,CL5,CL6 v.s. 其餘類別	1885	10.86	UCI
dry_bean	BOMBAY v.s. 其餘類別	13611	25.07	UCI
ecoli	imU v.s. 其餘類別	336	8.60	Zenodo
flare1	M-class > 0 v.s. M-class = 0	1389	19.43	Zenodo
flare2	F v.s. 其餘類別	1066	23.79	KEEL
jsbach	E_M v.s. 其餘類別	5665	18.20	UCI
kr-vs-k-zero	one v.s. draw	2901	26.63	KEEL
led7digit	1 v.s. 0,2,4,5,6,7,8,9	443	10.97	KEEL
letter_img	Z v.s. 其餘類別	20000	26.25	Zenodo
mammography	minority v.s. 其餘類別	11183	42.01	Zenodo
oil	minority v.s. 其餘類別	937	21.85	Zenodo
optical_digits	8 v.s. 其餘類別	5620	9.14	Zenodo
ozone_level	原始即為二類別	2536	33.74	Zenodo
page_blocks	1 v.s. 其餘類別	5473	8.77	UCI
pen_digits	5 v.s. 其餘類別	10992	9.42	Zenodo
satimage	4 v.s. 其餘類別	6435	9.28	Zenodo
seismic_bumps	原始即為二類別	2584	14.20	UCI
sensor_readings_24	Slight-Left-Turn v.s. 其餘類別	5456	15.63	UCI
shuttle	2,3,5,6,7 v.s. 其餘類別	14500	15.72	UCI
sick_euthyroid	sick euthyroid v.s. 其餘類別	3163	9.80	Zenodo
spectrometer	>=44 v.s. 其餘數值	531	10.80	Zenodo
steel_faults	Pastry v.s. 其餘類別	1941	11.28	UCI
us_crime	>0.65 v.s. 其餘數值	1994	12.29	Zenodo
vowel	0 v.s. 其餘類別	988	9.98	KEEL
wilt	原始即為二類別	4839	17.54	UCI
winequality-red	4 v.s. 其餘類別	1599	29.17	KEEL
yeast1	ML8 v.s. 其餘類別	2417	12.58	Zenodo
yeast2	MIT, CYT, ME3, EXC v.s. ME1,VAC,POX, ERL	1004	9.14	KEEL
yeast3	VAC, POX v.s. MIT, ME1, ME3, ERL	506	9.12	KEEL
yeast4	ME2 v.s. MIT, ME3, EXC, VAC, ERL	528	9.35	KEEL
yeast5	CYT v.s. ME2	514	9.08	KEEL
yeast6	ME2 v.s. 其餘類別	1484	28.10	KEEL
yeast7	ME1 v.s. 其餘類別	1484	32.73	KEEL
yeast8	EXC v.s 其餘類別	1484	41.40	KEEL

集，而另外一等分作為測試集，重複 5 次，使得每個樣本都會做為測試資料一次，以此來計算模型最終的分類性能。而關於資料集預處理的部分，因為會於 4.3 節中比較的 CBWKELM 方法，其論文(Choudhary & Shukla, 2021)實驗部分有將資料集以最大最小值正規化(min-max normalization)方法做預處理，透過下面的公式將資料集樣本屬性值轉換成介於 -1 到 1 之間的數值：

$$x' = \left(\frac{x - min_n}{max_n - min_n}\right) \times 2 - 1$$

此公式 min_n、max_n 分別為資料集樣本第 n 個屬性的最小與最大值，以此將某樣本第 n 個屬性原始值 x 轉換成新屬性值 x'。經試驗過 CBWKELM 在正規化後的資料集上分類性能表現較好，因此為了配合 CBWKELM，本研究實驗的資料集統一會先以最大最小正規化方法做預處理。

4.2 MSBagging 不同參數組合比較

本節是要試驗本研究提出的 MSBagging，在不同資料集上找出的最佳參數組合是否有某種規則，而基底模型數與集成比例則是 MSBagging 主要參數調整的對象，因此本實驗主要是探討是否基底模型數與集成比例有某種規則可以使 MSBagging 達到較佳訓練結果。

表 4.2 為 MSBagging 的參數範圍，在本實驗中，固定過取樣策略所使用的 kNN 的參數 k 值為 5，成本敏感策略中樣本分類錯誤成本設為最常使用的設定方式，將少數類別樣本分類錯誤成本設為訓練集 IR 值，多數類別樣本分類錯誤成本設為 1。而集成比例的格式以過取樣策略比重、欠取樣策略比重、成本敏感策略比重的順序表示，每個比重可能值為 1、1.25、1.5、1.75 或 2，但不是所有比重組合的集成比例都在本實驗的搜尋最佳參數組合的範圍內，本實驗會以特定規則來搜尋最佳的集成比例，搜尋集成比例的方式已在 3.2 節中說明，每次都會搜尋 13 種集成比例；而基底模型數則為方便計算以 8 為間隔從 8、16、

24、32、40 與 48 共六種數值進行實驗，因此對於 MSBagging 會搜尋 $6 \times 13 = 78$ 種參數組合，最後再以 F1 測度作為本實驗搜尋最佳參數組合的依據。

表 4.2、MSBagging 的參數範圍

方法參數	參數範圍
kNN 的 k 值	固定為 5
樣本分類錯誤成本	▪ 少數類別樣本：訓練集 IR 值 ▪ 多數類別樣本：1
基底模型數	$\{8, 16, 24, 32, 40, 48\}$
集成比例	▪ 比例＝過取樣：欠取樣：成本敏感 ▪ 各比重範圍＝$\{1, 1.25, 1.5, 1.75, 2\}$ ▪ 動態搜尋範圍，已於 3.2 節中說明

表 4.3 為本實驗的結果，列出每一個資料集所找到的最佳 F1 值與參數組合，而集成比例畫雙底線的是標示有比重大於 1 的數值。從結果的基底模型數量來看，基底模型數平均是 23.8，且頻率最高的是 16 有 14 個，次高的是 32 有 9 個，可知並非基底模型數愈高就能帶來比較好的訓練結果。而在集成比例中，可知每一種策略都有可能達到最大的比重 2，且沒有任何一個策略的比重有統一較多或較少的現象，也就是說沒有不必要的或一定需要的策略，不同資料集所偏好的策略不同，因此偏好的集成比例也不同。將兩個參數搭配來看，並沒有明顯的能看出基底模型數與集成比例的關聯，因此並沒有一個明顯的規則可以說明如何設定 MSBagging 的基底模型數與集成比例能夠帶來較好的訓練結果，會因為資料集不同而偏好的參數組合也有所不同。

表 4.3、MSBagging 最佳參數組合結果

資料集名稱	最佳參數組合	
	基底模型數	集成比例 (過取樣：欠取樣：成本敏感)
abalone1	40	1 : 2 : 1
abalone2	32	1.25 : 1 : 1
abalone3	8	1.75 : 1 : 1
car1	16	1.25 : 1 : 1.25
car2	16	1.5 : 1 : 2
coil_2000	16	1 : 2 : 1
drug_comp	32	2 : 1.25 : 1
dry_bean	16	1 : 1.75 : 1
ecoli	32	1 : 1.25 : 1
flare1	40	1 : 2 : 1
flare2	32	1 : 2 : 1
jsbach	8	1.75 : 1 : 1
kr-vs-k-zero	16	1.75 : 1.25 : 1
led7digit	8	2 : 1.25 : 1
letter_img	24	1.75 : 1 : 1
mammography	32	1.25 : 1 : 1.75
oil	40	1 : 2 : 1
optical_digits	16	1.75 : 2 : 1
ozone_level	32	1.25 : 1.5 : 1
page_blocks	32	1.25 : 1 : 1.75
pen_digits	16	1 : 1 : 1
satimage	24	2 : 1 : 1.25
seismic_bumps	24	1 : 1.75 : 1
sensor_readings_24	16	1.25 : 2 : 1
shuttle	16	1 : 1 : 1.5
sick_euthyroid	48	1 : 1 : 2
spectrometer	40	2 : 1 : 1
steel_faults	48	1.5 : 1 : 1.5
us_crime	24	1.75 : 1.75 : 1
vowel	8	1.25 : 1 : 1.75
wilt	40	1.25 : 1 : 1.5
winequality-red	8	1 : 1.5 : 1
yeast1	16	1 : 2 : 1
yeast2	32	1 : 1 : 1.5
yeast3	8	1.75 : 1.75 : 1
yeast4	16	1 : 2 : 1
yeast5	16	2 : 1 : 1
yeast6	16	2 : 1.25 : 1
yeast7	16	1.75 : 1.25 : 1
yeast8	32	1.5 : 1 : 2

4.3 與其他方法進行比較

從 4.2 節的實驗結果可知，本研究所提出的 MSBagging 方法中沒有明顯能得到較佳結果的參數組合方式，因此在本節實驗中與其他方法做比較時，會針對資料集各別挑選每個方法最佳的參數組合，來進行模型分類性能的比較。而比較對象包含本研究方法的三種策略：過取樣策略的 SMOTEBagging 方法、欠取樣策略的 UnderBagging 方法、成本敏感策略的 CostSensitiveBagging 方法，與 CBWKELM (Choudhary & Shukla, 2021)方法。

表 4.4 為本實驗的參數範圍，MSBagging 的參數範圍與表 4.2 相同，因此過取樣策略的 kNN 的 k 值與成本敏感策略的樣本分類錯誤成本也與 MSBagging 一致。但因 MSBagging 方法的模型總數不固定，且模型總數的多寡有可能影響到訓練結果，因此將三種策略的方法也可調整模型總數，再從中挑選表現較好的模型。模型總數的範圍因為 MSBagging 每個策略最少的模型數為 8，因此選擇以 8 做為最低的模型總數；間隔數量則是希望加大間隔以減少實驗的參數組合數，又避免因為間隔過大導致錯過更高分類性能的模型，因此選擇以 6 作為模型總數的間隔；參數上限則是因在 4.2 節的實驗結果中，MSBagging 最佳的參數組合中模型總數皆少於 200 個，MSBagging 模型總數的可由表 4.4 的基底模型數與集成比例計算而得，為方便與其他方法比較而放置於表 4.5 中，由此認為模型總數上限設至 200 多足以找到最佳分類性能的集成模型，最後模型總數範圍是以 6 為間隔從 8 到 212 共 35 種來進行實驗。

最後一個比較對象是實作 Choudhary and Shukla (2021)提出的 CBWKELM 方法，已經在 3.5.1 節大略介紹過此方法的訓練方式。CBWKELM 方法論文內還分成以軟投票法(Soft Voting)與最大投票法(Max Voting)兩種不同預測方法的模型，本實驗每個方法都以最大投票法預測，因此 CBWKELM 方法也只比較以

最大投票法預測類別的部分。並且 CBWKELM 方法論文內搜尋的參數組合數過多，為了減少訓練時間，因此本實驗簡化其參數的搜尋範圍，將論文內對 $log_2(\sigma)$ 的搜尋從 20 種減少到 4 種、對 $log_2(C)$ 從 35 種減少到 7 種、對 TH 從 9 種減少到 4 種，因此本實驗對於 CBWKELM 方法會搜尋 $4 \times 7 \times 4 = 112$ 種參數組合。

表 4.4、比較對象的參數範圍

方法	方法參數	參數範圍
MSBagging (本研究方法)	同表 4.2	
SMOTEBagging (過取樣策略)	kNN 的 k 值	固定為 5
	模型總數	$8 + 6i,\ i = 0, 1, 2, \ldots, 34$
UnderBagging (欠取樣策略)	模型總數	$8 + 6i,\ i = 0, 1, 2, \ldots, 34$
CostSensitiveBagging (成本敏感策略)	樣本分類錯誤成本	▪ 少數類別樣本：訓練集 IR 值 ▪ 多數類別樣本：1
	模型總數	$8 + 6i,\ i = 0, 1, 2, \ldots, 34$
CBWKELM	$log_2(\sigma)$	$\{-6, 0, 6, 12\}$
	$log_2(C)$	$\{0, 8, 16, 24, 32, 40, 48\}$
	TH	$\{0.1, 0.2, 0.3, 0.4\}$

搜尋最佳參數組合的方式則是與 4.2 節的實驗相同，所有方法皆以 F1 測度為依據來得到的最佳參數組合，最佳參數組合的模型再以 3.5.2 節提過會使用 F1、G-mean、AUC 三種測度來評估不同方法之間分類性能的差異。

表 4.5 中呈現各方法以 F1 為搜尋測度的最佳參數組合，而 MSBagging 的最佳參數組合是使用 4.2 節的實驗結果，已在表 4.3 中呈現過，但為方便比較

表 4.5、各方法最佳參數組合

資料集名稱	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM		
	模型總數	模型總數	模型總數	模型總數	$log_2(\sigma)$	$log_2(C)$	TH
abalone1	160	14	38	8	0	16	0.3
abalone2	104	56	134	8	6	16	0.4
abalone3	30	26	188	158	6	24	0.1
car1	56	86	116	20	0	16	0.1
car2	72	14	80	20	0	16	0.1
coil_2000	64	8	206	14	6	16	0.3
drug_comp	136	152	164	20	0	8	0.1
dry_bean	60	8	20	14	0	24	0.1
ecoli	104	110	38	50	0	8	0.2
flare1	160	8	38	134	12	0	0.3
flare2	128	8	50	74	12	8	0.3
jsbach	30	8	8	20	12	16	0.1
kr-vs-k-zero	64	32	32	56	0	16	0.1
led7digit	34	14	44	8	0	8	0.1
letter_img	90	194	176	62	0	24	0.4
mammography	128	176	86	104	0	0	0.3
oil	160	8	26	14	6	24	0.1
optical_digits	76	68	158	62	0	16	0.1
ozone_level	120	50	68	20	0	8	0.1
page_blocks	128	212	170	20	0	24	0.1
pen_digits	48	26	170	74	0	16	0.1
satimage	102	134	212	26	0	16	0.2
seismic_bumps	90	44	182	152	6	16	0.4
sensor_readings_24	68	8	14	20	0	16	0.1
shuttle	56	14	26	26	0	0	0.1
sick_euthyroid	192	206	74	86	0	16	0.3
spectrometer	160	32	188	68	0	16	0.1
steel_faults	192	38	8	20	6	24	0.3
us_crime	108	122	158	80	6	24	0.2
vowel	32	20	32	56	0	16	0.1
wilt	150	92	14	140	0	24	0.1
winequality-red	28	8	104	8	0	16	0.3
yeast1	64	8	68	8	6	8	0.4
yeast2	112	134	104	32	0	8	0.3
yeast3	36	20	158	62	0	0	0.1
yeast4	64	182	110	110	0	8	0.3
yeast5	64	20	44	86	0	8	0.3
yeast6	68	62	26	86	0	16	0.3
yeast7	64	8	26	32	0	16	0.1
yeast8	144	50	50	98	0	16	0.3

模型總數量，因此表 4.5 呈現的是 MSBagging 的模型總數。若看平均模型總數，MSBagging 平均 93.65 個、SMOTEBagging 平均 62 個、UnderBagging 平均 90.2 個、CostSensitiveBagging 平均 53.9 個，單一策略中以 UnderBagging 平均模型總數最多，CostSensitiveBagging 平均模型總數最少。UnderBagging 之所以需要較多模型總數，可能是因其減少多數類別樣本訓練數量，而需要更多模型來充分訓練多數類別樣本，以更好的進行預測；SMOTEBagging 與 CostSensitiveBagging 因無需去除樣本，而可以用較少的模型數量就取得較佳的結果；MSBagging 應是因為混合三種策略而使得最終模型總數較多；而 CBWKELM 的模型總數是以資料集分布而定，與參數無關，因此無法與其他方法比較。整體而言，每個方法的最佳參數組合並沒有一個可以歸納的規律，都因資料集不同而有所不同。

表 4.6、表 4.7、表 4.8 分別為各方法以 F1、G-mean、AUC 為測度的數值。從表 4.6 可知，平均 F1 為本研究的 MSBagging 為最高，SMOTEBagging 其次，CBWKELM 第三，但 CBWKELM 與 SMOTEBagging 相差不大，兩者在 F1 表現上是相當的。以 MSBagging 來說，因 F1 為挑選最佳參數組合的測度，可看出相較於單一策略，MSBagging 明顯能帶來較好的分類性能，並且有 22 個資料集 MSBagging 表現能夠超過使用單一策略的方法，在資料集 oil 與單一策略方法相比能提升至少 0.0969，因此 MSBagging 以混合三種策略的方式有機會表現超越單一策略方法許多。而 CBWKELM 則是在某些資料集上能獲得更高的 F1 數值，如在資料集 abalone3 上 CBWKELM 與其他方法相比提升至少 0.1322，因此 CBWKELM 有機會在資料集表現上更好，但以平均來看，與 MSBagging 有些差距。而表 4.7 中，平均 G-mean 也是以 MSBagging 勝出，CBWKELM 則排第二，但兩者差距比較小。即使並非以 G-mean 來挑選最佳的參數組合，MSBagging 依然在 19 個資料集上有超越所以單一策略的方法，但與

表 4.6、各方法的 F1 結果

資料集名稱	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM
abalone1	0.3890	0.3531	**0.4003**	0.2414	0.3911
abalone2	**0.4262**	0.4068	0.2462	0.3133	0.3776
abalone3	0.4255	0.4286	0.3432	0.3396	**0.5577**
car1	0.8609	0.8028	0.6479	**0.8873**	0.8467
car2	**0.9924**	0.9846	0.8228	0.9846	**0.9924**
coil_2000	0.1794	0.1122	**0.2263**	0.1228	0.1994
drug_comp	0.3046	0.2865	**0.3417**	0.1189	0.3392
dry_bean	0.9990	0.9962	0.9981	0.9981	**1.0000**
ecoli	**0.6914**	0.6857	0.6337	0.6349	0.6835
flare1	0.2368	0.1732	**0.2370**	0.2018	0.2264
flare2	0.2550	0.1842	**0.2966**	0.1429	0.2931
jsbach	0.8376	**0.8403**	0.7442	0.8000	0.7986
kr-vs-k-zero	**0.9498**	0.9479	0.7554	0.9372	0.8996
led7digit	0.7848	**0.8052**	0.6813	0.7654	0.6966
letter_img	0.9588	0.9633	0.8337	0.9533	**0.9843**
mammography	**0.7183**	0.7054	0.4357	0.6730	0.5320
oil	**0.5476**	0.4507	0.3646	0.4333	0.5128
optical_digits	0.9323	0.9201	0.9121	0.9135	**0.9845**
ozone_level	**0.3649**	0.2857	0.2533	0.2105	0.3436
page_blocks	**0.9873**	0.9855	0.9807	0.9861	0.9694
pen_digits	0.9833	0.9823	0.9801	0.9759	**0.9967**
satimage	0.6802	0.6756	0.6156	0.6410	**0.7171**
seismic_bumps	0.2514	0.1946	0.2774	0.1224	**0.2956**
sensor_readings_24	**0.9924**	0.9908	0.9805	0.9878	0.8936
shuttle	**0.9994**	0.9971	0.9869	0.9988	0.9740
sick_euthyroid	0.8878	0.8821	0.8440	**0.8885**	0.6380
spectrometer	0.7765	0.7901	0.6903	0.7692	**0.8837**
steel_faults	**0.6417**	0.6276	0.5541	0.5136	0.5825
us_crime	0.5301	0.5036	0.4925	0.4464	**0.5343**
vowel	0.9255	0.9312	0.8502	0.9231	**1.0000**
wilt	**0.8697**	0.8515	0.7635	0.8577	0.8000
winequality-red	0.1798	0.1136	0.1778	0.0345	**0.1987**
yeast1	0.0404	0.0755	0.1735	0.0106	**0.1749**
yeast2	**0.8400**	0.8229	0.7854	0.8085	0.7757
yeast3	0.4000	0.3704	0.4000	0.3235	**0.4211**
yeast4	0.5849	0.5591	0.5211	0.5000	**0.6032**
yeast5	**0.7523**	0.7477	0.7407	0.7312	0.7339
yeast6	**0.4444**	0.4270	0.3030	0.3235	0.3575
yeast7	0.7917	**0.8000**	0.6000	0.7381	0.6441
yeast8	0.6176	**0.6250**	0.3000	0.5902	0.3623
平均	**0.6508**	0.6321	0.5798	0.5961	0.6304

表 4.7、各方法 G-mean 結果

資料集名稱	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM
abalone1	0.3999	0.3531	**0.4563**	0.2512	0.4389
abalone2	**0.4267**	0.4068	0.3489	0.3414	0.3845
abalone3	0.4280	0.4286	0.3971	0.4187	**0.5683**
car1	0.8641	0.8031	0.6922	**0.8877**	0.8467
car2	**0.9924**	0.9846	0.8360	0.9846	**0.9924**
coil_2000	0.1795	0.1172	**0.2798**	0.1265	0.2779
drug_comp	0.3104	0.2887	**0.4034**	0.1711	0.3774
dry_bean	0.9990	0.9962	0.9981	0.9981	**1.0000**
ecoli	**0.6978**	0.6857	0.6658	0.6389	0.6880
flare1	0.2589	0.1737	**0.3223**	0.2178	0.2926
flare2	0.2814	0.1858	**0.3842**	0.1507	0.3771
jsbach	0.8391	**0.8404**	0.7584	0.8034	0.7987
kr-vs-k-zero	**0.9506**	0.9479	0.7791	0.9373	0.9027
led7digit	0.7864	**0.8058**	0.6935	0.7683	0.7067
letter_img	0.9589	0.9634	0.8436	0.9536	**0.9843**
mammography	**0.7186**	0.7054	0.5034	0.6919	0.5535
oil	**0.5478**	0.4562	0.4448	0.4658	0.5374
optical_digits	0.9331	0.9222	0.9124	0.9162	**0.9846**
ozone_level	0.3649	0.3104	0.3436	0.2495	**0.3678**
page_blocks	**0.9873**	0.9855	0.9808	0.9861	0.9695
pen_digits	0.9833	0.9823	0.9801	0.9760	**0.9967**
satimage	0.6803	0.6764	0.6409	0.6509	**0.7171**
seismic_bumps	0.2520	0.1966	0.3324	0.1805	**0.3385**
sensor_readings_24	**0.9924**	0.9908	0.9805	0.9878	0.8936
shuttle	**0.9994**	0.9971	0.9870	0.9988	0.9743
sick_euthyroid	0.8878	0.8821	0.8486	**0.8887**	0.6435
spectrometer	0.7778	0.7950	0.7050	0.7785	**0.8847**
steel_faults	**0.6418**	0.6293	0.5786	0.5277	0.5928
us_crime	0.5326	0.5052	0.5473	0.4660	**0.5584**
vowel	0.9264	0.9323	0.8576	0.9231	**1.0000**
wilt	**0.8705**	0.8534	0.7782	0.8586	0.8054
winequality-red	0.1831	0.1161	**0.2376**	0.0614	0.2081
yeast1	0.0670	0.0804	0.1974	0.0226	**0.2509**
yeast2	**0.8400**	0.8233	0.7890	0.8097	0.7779
yeast3	0.4025	0.3810	0.4278	0.3667	**0.4481**
yeast4	0.5853	0.5618	0.5431	0.5119	**0.6144**
yeast5	0.7538	0.7485	**0.7639**	0.7346	0.7355
yeast6	**0.4451**	0.4316	0.4018	0.3736	0.3961
yeast7	0.7944	**0.8002**	0.6462	0.7389	0.6659
yeast8	0.6179	**0.6278**	0.3948	0.5967	0.4164
平均	**0.6540**	0.6343	0.6170	0.6103	0.6492

表 4.8、各方法 AUC 結果

資料集名稱	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM
abalone1	**0.8509**	0.8182	0.8502	0.7403	0.8119
abalone2	0.9209	0.9104	**0.9267**	0.7811	0.8670
abalone3	0.8652	0.8425	0.8400	0.8243	**0.9556**
car1	0.9975	0.9940	0.9968	**0.9982**	0.9198
car2	**1.0000**	0.9999	0.9996	0.9999	0.9997
coil_2000	0.7236	0.6341	**0.7393**	0.6459	0.6724
drug_comp	0.8144	0.8041	**0.8206**	0.7829	0.8174
dry_bean	1.0000	1.0000	**1.0000**	1.0000	**1.0000**
ecoli	**0.9477**	0.9054	0.9321	0.9168	0.9051
flare1	0.7748	0.6932	**0.7956**	0.7230	0.6957
flare2	0.8637	0.7639	**0.8903**	0.7893	0.8318
jsbach	0.9817	0.9592	**0.9826**	0.9606	0.9797
kr-vs-k-zero	0.9997	**0.9998**	0.9989	0.9996	0.9877
led7digit	0.9285	0.9081	**0.9303**	0.9217	0.8947
letter_img	0.9995	**0.9996**	0.9992	0.9989	0.9902
mammography	0.9460	0.9399	**0.9506**	0.9008	0.8877
oil	**0.9354**	0.8344	0.9225	0.8537	0.9079
optical_digits	0.9970	**0.9977**	0.9963	0.9974	0.9880
ozone_level	0.8925	0.8671	**0.8980**	0.8015	0.7472
page_blocks	**0.9924**	0.9912	0.9905	0.9841	0.9392
pen_digits	**0.9997**	0.9986	0.9996	0.9992	0.9984
satimage	0.9569	**0.9586**	0.9547	0.9479	0.8942
seismic_bumps	0.7372	0.7000	0.7511	0.6974	**0.7527**
sensor_readings_24	0.9999	0.9987	**1.0000**	0.9987	0.9867
shuttle	**1.0000**	1.0000	1.0000	1.0000	0.9844
sick_euthyroid	**0.9867**	0.9854	0.9846	0.9822	0.8758
spectrometer	0.9707	0.9696	0.9634	**0.9725**	0.9383
steel_faults	**0.9452**	0.9329	0.9154	0.9173	0.9268
us_crime	**0.9151**	0.9086	0.9147	0.9015	0.8331
vowel	0.9972	0.9977	0.9956	0.9871	**1.0000**
wilt	0.9922	0.9886	0.9873	0.9904	**0.9939**
winequality-red	**0.7483**	0.6556	0.7300	0.5928	0.6534
yeast1	0.5802	0.4988	**0.6163**	0.5560	0.5937
yeast2	0.9527	0.9517	0.9531	**0.9632**	0.9025
yeast3	0.7696	0.7049	**0.8017**	0.7412	0.7691
yeast4	0.8928	**0.8972**	0.8942	0.8739	0.8236
yeast5	**0.9865**	0.9843	0.9853	0.9855	0.8965
yeast6	0.8927	0.8816	**0.9089**	0.8777	0.7895
yeast7	**0.9894**	0.9521	0.9872	0.9413	0.9156
yeast8	0.9108	0.8766	**0.9148**	0.8746	0.8487
平均	0.9164	0.8926	**0.9179**	0.8855	0.8794

CBWKELM 僅有些微的差距，因此兩者在 G-mean 表現上是差不多的。表 4.8 中可以發現，雖然在前兩個測度的表現上，UnderBagging 相較於其他方法來說表現並不好，但在平均 AUC 中反而以 UnderBagging 最高，不過 MSBagging 也在平均 AUC 上有著 0.9164 的好表現，只以 0.0015 的微小差距輸給 UnderBagging 而已，因此 MSBagging 在平均 AUC 的表現也是非常不錯的，表現與 UnderBagging 相當，反而在 F1 與 G-mean 平均表現較好的 CBWKELM，在平均 AUC 中表現比較差一點。

再以 Wilcoxon 符號排序檢定(Wilcoxon signed rank test)來檢驗是否 MSBagging 有顯著與其他方法的分類性能有差距。表 4.9、表 4.10、表 4.11 別呈現以 F1、G-mean、AUC 為依據的檢定結果。Wilcoxon 是符號排序檢定將兩比較對象的測度值相減後，將此差距值忽視正負後進行排序，再分別加總差距值為正的名次與為負的名次，取最小的作為檢定統計量，而後查表可得 p-value 值以進行檢定。表 4.9、表 4.10、表 4.11 中的 W_+ 為檢定流程中，MSBagging 比另一個方法要好的名次和，W_- 為另一個方法比 MSBagging 要好的名次和，依據 W_+、W_- 選較小的值再查表可得出 p-value 值，然後以 $\alpha = 0.05$ 來檢定是否拒絕 MSBagging 與另一方法分類性能有差距的假設。

從表 4.9 中可知在 F1 中 MSBagging 比只用單一策略的 SMOTEBagging、UnderBagging 與 CostSensitiveBagging 要好，這可在表 4.6 中看出來，反而是在表 4.6 與 MSBagging 平均有些差距的 CBWKELM，在檢定結果中 MSBagging 並沒有顯著比 CBWKELM 要好，表示 MSBagging 與 CBWKELM 在 F1 上表現差不多。從表 4.10 中可知，G-mean 中 MSBagging 是有顯著比其他三種只用單一策略的 SMOTEBagging、UnderBagging、CostSensitiveBagging 中要好，但無顯著比 CBWKELM 方法要好，與表 4.7 所觀察到的結果一致。

表 4.11 中可知，以 AUC 來說，MSBagging 除了 UnderBagging 外，有顯著比其他方法要好，MSBagging 在 AUC 上是與 UnderBagging 分類性能差不多的，與表 4.8 觀察到的結果一致。

整體而言，本研究提出的 MSBagging，在多數測度上會比只使用單一策略的 SMOTEBagging、UnderBagging、CostSenstiveBagging 要好，而 CBWKELM 則是在多數測度上與本研究的 MSBagging 分類性能相當。

表 4.9、依據 F1 做 Wilcoxon 符號檢定的結果

F1 比較對象	W_+	W_-	**p-value**	**檢定結果** $(\alpha = 0.05)$
MSBagging v.s. SMOTEBagging	674	146	**0.0004**	**拒絕**
MSBagging v.s. UnderBagging	690	90	**0.0000**	**拒絕**
MSBagging v.s. CostSensitiveBagging	802	18	**0.0000**	**拒絕**
MSBagging v.s. CBWKELM	474	306	0.2420	不拒絕

表 4.10、依據 G-mean 做 Wilcoxon 符號檢定的結果

G-mean 比較對象	W_+	W_-	**p-value**	**檢定結果** $(\alpha = 0.05)$
MSBagging v.s. SMOTEBagging	691	129	**0.0002**	**拒絕**
MSBagging v.s. UnderBagging	592	228	**0.0143**	**拒絕**
MSBagging v.s. CostSensitiveBagging	797	23	**0.0000**	**拒絕**
MSBagging v.s. CBWKELM	392	388	0.9761	不拒絕

表 4.11、依據 AUC 做 Wilcoxon 符號檢定的結果

AUC 比較對象	W_+	W_-	**p-value**	**檢定結果 ($\alpha = 0.05$)**
MSBagging v.s. SMOTEBagging	765	55	**0.0000**	**拒絕**
MSBagging v.s. UnderBagging	415	405	0.9442	不拒絕
MSBagging v.s. CostSensitiveBagging	770	50	**0.0000**	**拒絕**
MSBagging v.s. CBWKELM	745	75	**0.0000**	**拒絕**

4.4 訓練時間比較

4.3 節中比較本研究提出的 MSBagging 與其他方法的分類性能差異，在本節中要比較它們訓練模型所花費的時間差異。本節是要分析在得到 4.3 節數據結果下，各方法模型所需要的訓練時間。以 MSBagging 來說搜尋了 78 種參數組合，SMOTEBagging、UnderBagging、CostSensitiveBagging 搜尋了 35 種參數組合，CBWKELM 搜尋了 112 種參數組合，訓練時間即是加總每個搜尋的參數組合訓練模型所花費的時間。以 MSBagging 為例，會加總共 78 個不同參數組合訓練模型所花費的時間，以此來進行各方法訓練時間的比較。

訓練時間的結果從 CPU 為 Intel(R) Xeon(R) Gold 5217、64GB 記憶體的電腦上取得，實際的訓練時間放在**附錄 一**。為更容易說明，表 4.12 將訓練時間以 MSBagging 作為基準值，以比值的方式呈現訓練時間。由表中可知執行最快的是 CostSensitiveBagging，因為 CostSensitiveBagging 只需要調整樣本分類錯誤成本；UnderBagging 排第二，因為相對於 CostSensitiveBagging 需要花費減少多數類別樣本來平衡資料集的時間；SMOTEBagging 排第三，因為相對於 CostSensitiveBagging 需要產生合成少數類別樣本比 UnderBagging 所需減少多數類別樣本的時間更多一些；MSBagging 排第四，可能因其搜尋參數組合數較多，相較於最快的 CostSensitiveBagging 平均慢五倍的時間，但相較單一策略中

表 4.12、訓練時間(以 MSBagging 為基準的比值)

資料集名稱	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM
abalone1	1.0000	0.7209	**0.1882**	0.2011	1.8296
abalone2	1.0000	0.5595	0.2925	**0.2238**	3.1895
abalone3	1.0000	0.4284	0.5786	**0.3815**	1.2066
car1	1.0000	0.4979	0.6014	**0.3945**	10.7820
car2	1.0000	0.2585	0.3117	**0.2005**	5.4338
coil_2000	1.0000	1.2727	**0.0889**	0.2304	36.9344
drug_comp	1.0000	0.9452	**0.2485**	0.2564	1.2821
dry_bean	1.0000	0.8523	0.3891	**0.3347**	164.6373
ecoli	1.0000	0.5443	0.5443	**0.2221**	0.6567
flare1	1.0000	0.6357	0.3469	**0.1537**	2.3496
flare2	1.0000	0.4993	0.3927	**0.2468**	2.9542
jsbach	1.0000	1.3075	0.0281	**0.0268**	2.9628
kr-vs-k-zero	1.0000	0.4913	0.4915	**0.3125**	23.3283
led7digit	1.0000	0.6041	0.5945	**0.3848**	1.8565
letter_img	1.0000	0.8467	**0.0983**	0.1619	322.7632
mammography	1.0000	0.7264	0.2452	**0.1799**	24.2563
oil	1.0000	0.6832	0.2396	**0.1795**	2.5715
optical_digits	1.0000	1.5807	**0.1438**	0.1551	11.3570
ozone_level	1.0000	1.6833	**0.0594**	0.1582	2.0833
page_blocks	1.0000	0.5470	**0.2764**	0.4378	16.6360
pen_digits	1.0000	1.8774	**0.1706**	0.2101	87.1727
satimage	1.0000	1.0232	**0.0489**	0.1182	4.1832
seismic_bumps	1.0000	1.8048	**0.0326**	0.0392	0.2453
sensor_readings_24	1.0000	1.0128	**0.2430**	0.2902	39.6816
shuttle	1.0000	0.2202	0.0657	**0.0426**	6.9455
sick_euthyroid	1.0000	0.7082	0.1764	**0.1045**	3.9612
spectrometer	1.0000	0.7271	**0.1455**	0.1622	1.2796
steel_faults	1.0000	1.0794	**0.2382**	0.3221	2.9137
us_crime	1.0000	1.2667	**0.0875**	0.1378	1.9304
vowel	1.0000	0.5594	0.4759	**0.3147**	2.1657
wilt	1.0000	0.8266	0.3891	**0.2609**	9.7488
winequality-red	1.0000	0.4392	0.1694	**0.1254**	9.2575
yeast1	1.0000	1.4639	**0.0658**	0.2162	1.0967
yeast2	1.0000	0.4265	0.3919	**0.1816**	6.1517
yeast3	1.0000	0.4133	0.3725	**0.1785**	3.5594
yeast4	1.0000	0.4936	0.3464	**0.2267**	4.6259
yeast5	1.0000	0.4940	0.4999	**0.2235**	2.8306
yeast6	1.0000	0.4877	0.3497	**0.1912**	14.2421
yeast7	1.0000	0.5388	0.6851	**0.2090**	10.2281
yeast8	1.0000	0.2800	0.3326	**0.0763**	9.6162
平均	1.0000	0.7957	0.2862	**0.2118**	21.5227

平均最慢的 SMOTEBagging 只花費其平均 1.26 倍的時間，且即使搜尋參數組合數是其他單一策略方法的 2.2 倍，但在有些資料集上甚至能比 SMOTEBagging 快；最慢的是 CBWKELM，除了因為搜尋的參數組合數最多以外，其方法會隨著樣本數量的增加而讓訓練時間大幅增加，因此在 20000 筆資料集 letter_img 上需要耗費 322.7 倍 MSBagging 的訓練時間，因此並不適合應用在樣本數較大的資料集上，但若是在較小的資料集上，如同資料集 ecoli 只有 336 筆，就有可能比 MSBagging 還快，可知 CBWKELM 訓練時間的浮動範圍非常大。

在表 4.13 中以表 4.12 的時間平均比值除以每個方法搜尋的參數組合數，再以 MSBagging 為基準來呈現出平均單一參數組合需要的時間比值。結果一樣以只需調整樣本分類錯誤成本的 CostSensitiveBagging 最快，UnderBagging 則因為減少多數類別樣本的時間而排第二，SMOTEBagging 因為需要產生合成樣本的時間而排第四，而本研究的 MSBagging 方法因為混合三種策略因此時間是三種的中和，比 SMOTEBagging 要快而排第三，而 CBWKELM 最慢，單一參數組合要花上 MSBagging 方法約 15 倍的時間。

表 4.13、平均單一參數組合時間比值(以 MSBagging 為基準)

	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM
時間平均比值	1.0000	0.7957	0.2862	0.2118	21.5227
搜尋參數組合數	78	35	35	35	112
平均單一參數組合時間比值	1.0000	1.7733	0.6378	**0.4720**	14.9890

若能夠進一步減少 MSBagging 搜尋數量，則有可能會比 SMOTEBagging 還快，即使不調整搜尋數量，平均也只花 SMOTEBagging 方法 1.26 倍的時間，分類性能從 4.3 節中可知與 CBWKELM 相當，且 MSBagging 只需要花 CBWKELM 方法平均 4.65%的訓練時間即可達到 CBWKELM 的效果，因此 MSBagging 搜尋

78 種參數組合而增加的訓練時間所達到的分類性能是相當值得的。

4.5 小結

在 4.2 節的實驗可知本研究的 MSBagging 方法無法找到任何一個明顯的參數組合規則可以直接提升模型訓練效果，因資料集不同而最佳的參數組合也不同。在 4.3 節中與其他方法比較，可知在 F1 與 G-mean 的測度下，MSBagging 的分類性能有顯著比其他三個單一策略的方法要好，並與 CBWKELM 方法分類性能相當；在 AUC 測度下，雖然 MSBagging 的分類性能沒有顯著比欠取樣策略的 UnderBagging 好，但也與 UnderBagging 相當，並且分類性能有顯著比過取樣策略的 SMOTEBagging、成本敏感策略的 CostSensitiveBagging 與 CBWKELM 要好。在 4.4 節各方法訓練時間的比較上，可知 MSBagging 比單一策略的方法都長，比單一策略訓練時間最長的 SMOTEBagging 方法 1.26 倍的時間，但若只看單一參數組合而言，MSBagging 能比 SMOTEBagging 要快，而 CBWKELM 不管是搜尋多個組合或是單一組合上平均要花費 MSBagging 方法 10 多倍的時間。

因此本研究的 MSBagging 在多數測度上分類性能比單一策略的方法要好，與 CBWKELM 的分類性能相當，訓練時間上雖然 MSBagging 比單一策略方法要花更多時間訓練，但比 CBWKELM 快上許多，平均只花費 CBWKELM 方法不到一成的時間就能達到 CBWKELM 相當的分類性能，MSBagging 以混合三種策略所取得的分類性能是值得的。

第五章　結論與建議

本研究是以混合涵蓋資料層面與演算法層面的三種策略：過取樣策略、欠取樣策略、成本敏感策略，以此提升多樣性的方式來建構一個集成分類方法，在本章對於前面的實驗結果進行統整，並提出未來可以繼續延伸的方向。

5.1 結論

一般的學習方法在不平衡資料上，容易訓練出傾向多數類別而忽視少數類別的模型，而無法在實務上使用，也因為少數類別正確率小於 0.5 的基本模型不滿足集成方法的基本使用條件，而不適用集成方法。而過去文獻將集成方法在不平衡資料上都是搭配資料層面的方法或是演算法層面的方法，但尚未有文獻以並用資料層面與演算法層面的方式來進行異質集成，因此本研究則是混合資料層面與演算法層面的三種策略，將此方法命名為 MSBagging，以比例來集成三種策略的基本模型，並找尋最佳的比例，以解決不平衡資料的學習問題。

本研究使用 40 個資料集進行實驗，先找尋是否能找到挑選參數組合的方式來讓 MSBagging 達到較佳的訓練結果，但結果都因為資料集不同而最佳的參數組合也不同。而後將 MSBagging 與三種單一策略的 SMOTEBagging、UnderBagging、CostsensitiveBagging 和近年提出的 CBWKELM 方法進行比較，比較五種方法的分類性能與訓練時間，分類性能部分使用常用的 F1、G-mean、AUC 三種來進行評估。

在實驗中，MSBagging 總共搜尋 78 種參數組合，而三種策略搜尋 35 種參數組合，CBWKELM 則是搜尋 112 種參數組合，以 F1 作為挑選最佳參數組合的測度。分類性能的結果顯示在多數測度上本研究的 MSBagging 有顯著比三者的單一策略要佳，與 CBWKELM 則是分類性能相當。所有參數組合的訓練時間上，以 CostSensitiveBagging 最快、UnderBagging 第二、SMOTEBagging 第

三、MSBagging 第四、CBWKELM 最後一名；而在單一參數組合的訓練時間上，一、二、最後一名不變，第三名則變成 MSBagging，第四名為 SMOTEBagging，因為 MSBagging 混合三種策略，因此單一參數組合的訓練時間在三種策略之間。但不管是所有參數組合或是單一參數組合上，MSBagging 只需花費 CBWKELM 不到一成的時間，而分類性能與 CBWKELM 相當，因此 MSBagging 以混合三種策略的方式所取得的分類性能是值得的。

5.2 未來展望

本研究方法主要是以比例來混合三種策略的方法，且實證此方法的有效性，並未加入其他進一步提升模型表現的方式，未來可以嘗試結合來進一步提升模型分類性能的表現。例如，未來則可以考慮加入剪裁模型數量的動作，每個策略各自以能提升分類性能為目標來剪裁部分的模型數量，來進一步提升混合三種策略的分類性能。或是可以以不同的方式來集成所有基本模型的意見，像是為基本模型改以權重的方式來進行整合，每個策略的基本模型可以有不同的權重，來調整出分類性能較佳的集成模型。

參考文獻

Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S., Sanchez, L., & Herrera, F. (2011). KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*(2-3), 255-287.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123-140.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321-357.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. *Proceedings of 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 107-119. Cavtat-Dubrovnik, Croatia.

Chen, L., Fang, B., Shang, Z., & Tang, Y. (2018). Tackling class overlap and imbalance problems in software defect prediction. *Software Quality Journal*, *26*(1), 97-125.

Choudhary, R., & Shukla, S. (2021). A clustering based ensemble of weighted kernelized extreme learning machine for class imbalance learning. *Expert Systems with Applications*, *164*, Article 114041.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Proceedings of 1st International Workshop on Multiple Classifier Systems*, 1-15. Cagliari, Italy.

Dua, D., & Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, Irvine, School of Information and Computer Sciences.

Díez-Pastor, J. F., Rodríguez, J. J., García-Osorio, C. I., & Kuncheva, L. I. (2015). Diversity techniques improve the performance of the best imbalance learning ensembles. *Information Sciences*, *325*, 98-117.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Springer Nature Switzerland AG.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, *121*(2), 256-285.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A

review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, *42*(4), 463-484.

Guo, H. X., Li, Y. J., Shang, J., Gu, M. Y., Huang, Y. Y., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220-239.

Hou, Y., Li, L., Li, B., & Liu, J. (2019). An anti-noise ensemble algorithm for imbalance classification. *Intelligent Data Analysis*, *23*(6), 1205-1217.

Iranmehr, A., Masnadi-Shirazi, H., & Vasconcelos, N. (2019). Cost-sensitive support vector machines. *Neurocomputing*, *343*, 50-64.

Lemaitre, G., Nogueira, F., Aridas, C. K., & Oliveira, D. V. R. (2016). Imbalanced dataset for benchmarking [Data set]. Zenodo. https://doi.org/10.5281/zenodo.61452

Li, H. X., Feng, A., Lin, B., Su, H. C., Liu, Z. X., Duan, X. L., Pu, H. B., & Wang, Y. F. (2021). A novel method for credit scoring based on feature transformation and ensemble model. *PeerJ Computer Science*, 18, Article e579.

Liu, X. Y., Wu, J. X., & Zhou, Z. H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, *39*(2), 539-550.

Ng, W., Xu, S., Zhang, J., Tian, X., Rong, T., & Kwong, S. (2020). Hashing-based undersampling ensemble for imbalanced pattern classification problems. *IEEE Transactions on Cybernetics*, *52*(2), 1-11.

Raghuwanshi, B. S., & Shukla, S. (2018). UnderBagging based reduced kernelized weighted extreme learning machine for class imbalance learning. *Engineering Applications of Artificial Intelligence*, *74*, 252-270.

Razavi-Far, R., Farajzadeh-Zanajni, M., Wang, B. Y., Saif, M., & Chakrabarti, S. (2021). Imputation-based ensemble techniques for class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, *33*(5), 1988-2001.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans*, *40*(1), 185-197.

Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, *40*(12), 3358-3378.

Sun, Y. M., Wong, A. K. C., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(4), 687-719.

Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., & Zhou, Y. (2015). A novel ensemble method for classifying imbalanced data. *Pattern Recognition*, *48*(5), 1623-1637.

Tang, B., & He, H. B. (2017). GIR-based ensemble sampling approaches for imbalanced learning. *Pattern Recognition*, *71*, 306-319.

Ting, K. M. (2002). An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, *14*(3), 659-665.

Wang, L., Zhao, L., Gui, G., Zheng, B. Y., & Huang, R. C. (2017). Adaptive ensemble method based on spatial characteristics for classifying imbalanced data. *Scientific Programming*, *2017*, Article 3704525.

Wang, S., & Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. *Proceedings of 2009 IEEE Symposium on Computational Intelligence and Data Mining*, 324-331. Nashville, TN, USA.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241-259.

Yang, P. T., Wu, W. S., Wu, C. C., Shih, Y. N., Hsieh, C. H., & Hsu, J. L. (2021). Breast cancer recurrence prediction with ensemble methods and cost-sensitive learning. *Open Medicine*, *16*(1), 754-768.

Zefrehi, H. G., & Altincay, H. (2020). Imbalance learning using heterogeneous ensembles. *Expert Systems with Applications*, *142*, Article 113005.

Zhao, Y., Shrivastava, A. K., & Tsui, K. L. (2016). Imbalanced classification by learning hidden data structure. *IIE Transactions*, *48*(7), 614-628.

Zong, W., Huang, G.-B., & Chen, Y. (2013). Weighted extreme learning machine for imbalance learning. *Neurocomputing*, *101*, 229-242.

附錄 一、各方法的訓練時間(單位：秒)

資料集名稱	MS Bagging	SMOTE Bagging	Under Bagging	CostSensitive Bagging	CBWKELM
abalone1	1948.0	1404.2	**366.6**	391.7	3564.0
abalone2	988.9	553.3	289.2	**221.3**	3154.0
abalone3	706.2	302.5	408.6	**269.4**	852.0
car1	693.4	345.2	417.0	**273.6**	7476.2
car2	1311.6	339.0	408.8	**263.0**	7126.7
coil_2000	12905.8	16425.8	**1146.9**	2973.2	476669.9
drug_comp	1542.2	1457.7	**383.2**	395.5	1977.3
dry_bean	3101.7	2643.6	1206.9	**1038.2**	510652.5
ecoli	586.8	319.4	319.4	**130.3**	385.3
flare1	928.7	590.3	322.2	**142.7**	2182.1
flare2	609.4	304.3	239.3	**150.4**	1800.3
jsbach	19901.3	26020.5	559.4	**534.2**	58962.6
kr-vs-k-zero	803.9	394.9	395.1	**251.2**	18753.2
led7digit	479.8	289.8	285.2	**184.6**	890.7
letter_img	6704.9	5677.0	**658.8**	1085.2	2164081.6
mammography	3101.9	2253.2	760.6	**558.1**	75240.6
oil	1421.7	971.4	340.7	**255.2**	3656.0
optical_digits	5805.2	9176.2	**834.9**	900.1	65930.0
ozone_level	6330.1	10655.2	**376.2**	1001.2	13187.6
page_blocks	2729.9	1493.2	**754.6**	1195.3	45414.7
pen_digits	4793.1	8998.5	**817.5**	1007.0	417829.8
satimage	11721.2	11992.5	**572.9**	1385.1	49032.3
seismic_bumps	12305.0	22207.6	**401.1**	482.4	3018.7
sensor_readings_24	2283.5	2312.7	**555.0**	662.6	90613.9
shuttle	19272.5	4243.9	1265.6	**821.8**	133856.4
sick_euthyroid	2683.5	1900.5	473.3	**280.5**	10630.0
spectrometer	1408.6	1024.2	**204.9**	228.5	1802.4
steel_faults	1630.8	1760.3	**388.5**	525.2	4751.8
us_crime	7277.8	9218.8	**636.8**	1002.6	14048.8
vowel	698.4	390.7	332.4	**219.8**	1512.5
wilt	2102.9	1738.3	818.2	**548.6**	20500.4
winequality-red	1417.7	622.7	240.2	**177.8**	13124.2
yeast1	12313.7	18025.6	**810.6**	2661.7	13504.5
yeast2	861.8	367.6	337.7	**156.5**	5301.2
yeast3	756.9	312.8	282.0	**135.1**	2694.0
yeast4	566.4	279.5	196.2	**128.4**	2619.9
yeast5	568.6	280.9	284.2	**127.1**	1609.4
yeast6	867.8	423.3	303.5	**165.9**	12359.6
yeast7	747.4	402.7	512.1	**156.2**	7644.1
yeast8	1548.2	433.5	514.9	**118.2**	14888.1