Abstract

This project implements a novel ensemble method to improve the performance of binary classification. The proposed method in the base paper is a non-linear combination of base models and an application of adaptive selection of the most suitable model for each data instance. Ensemble methods, an important type of machine learning technique, have drawn a lot of attention in both academic research and practical applications, and they use multiple single models to construct a hybrid model. A hybrid model generally performs better compared to a single individual model. The proposed approach in the base paper is based on a hybrid model has been validated on Repeat Buyers Prediction dataset, and the experiment results show improvement on F1 score, compared to the best individual model. In addition, the proposed method is compared with two other commonly used ensemble methods (Averaging and Stacking) in terms of F1 score. The improvement over the base paper is the use of Hierarchical clustering to select the base model classifier.

1. Introduction

In supervised learning techniques, ensemble learning method is the technique that uses multiple single models to construct a hybrid model in order to achieve better performance compared to that of using a single model. A workflow for solving classification problems by applying ensemble methods consists of the following. First, raw data usually need to be pre-processed for initializing a training dataset, during which feature extraction and normalization are applied. Second, training sets for each individual single model are derived from the initialized dataset. Third, single models are trained from different training datasets or by different algorithms. Finally, all the single models are combined to construct the ensemble and then the final [hybrid] model is constructed based on the results of individual models and the hybrid model is further validated and tested.

The ensemble learning techniques could be categorized into two types: Type I techniques focus on deriving new training sets from the initial training set to train multiple different single models. Type II techniques focus on finding ways to blend the individual models. In the base paper, Type II techniques are used as method for improving predictive performance in binary classification problems. The most popular Type II ensemble techniques include bagging and boosting. The proposed method gives a nonlinear combination of base classifiers to give the final ensemble classifier.

The single classifiers used to train the data set are

- AdaBoost
- XGBoost
- Random Forest
- Logistic Regression

The base classifier models are selected using 5 different methods

- Root Mean Square Deviation
- Zero One Loss
- F1 Score
- Area Under Curve Values
- Hierarchical Clustering

A meta model is generated from the training data set using the predictions of the base classifiers and this meta model is trained and used to predict the best classifier for each of the test tuples. Then the performance is evaluated using F1 Score(2*Harmonic mean of precision and recall) and AUC values.(Area Under the Curve of the Receiver Operating Characteristic [ROC] curve).

2. Literature Survey

2.1 Background

Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. Under boosting classifiers AdaBoost and XGBoost are used.

AdaBoost Classifier

The AdaBoost algorithm of Freund and Schapire was the first practical boosting algorithm, and also remains one of the most widely used and studied with applications in numerous fields. AdaBoost is best used to boost the performance of decision trees on binary classification problems. It can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier.

AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (e.g., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner. A weak classifier is prepared on the training data using the weighted samples. Only binary classification problems are supported, so each decision stump makes one decision on one input variable and outputs a value for the first or second class value. A stage value is calculated for the trained model which provides a weighting for any predictions that the model makes. The effect of the stage weight is that more accurate models have more weight or contribution to the final prediction. The training weights are updated giving more weight to incorrectly predicted instances, and less weight to correctly predicted instances. Weak models are added sequentially, trained using the weighted training data. The process continues until a pre-set number of weak learners have been created (a user parameter) or no further improvement can be made on the training dataset. The prediction for the ensemble model is taken as the sum of the weighted predictions.

XGBoost Classifier

XGBoost is short for “Extreme Gradient Boosting”, is a classifier model which implements the gradient boosting algorithm. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. Gradient boosting can benefit from regularization methods to reduce overfitting.

Bagging and Random Forest

For bagging the Random Forest Classifier is used. Bagging tries to implement similar learners on small sample populations and then takes a mean of all the predictions.

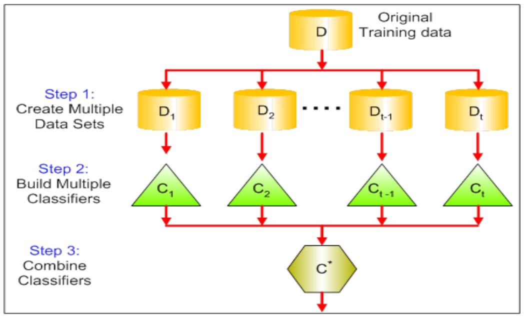

D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
....
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

Figure 2.1.1: Bagging Ensemble Method

Decision Trees are used to construct a Random Forest classifier. Decision trees alone have a very high variance. Random Forest is an extension of bagging that in addition to building trees based on multiple samples of the training data, it also constrains the features that can be used to build the trees, forcing trees to be different.

Logistic Regression

Logistic Regression is a regression model where the dependent variable is categorical. Logistic regression can be binomial, ordinal or multinomial. Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types, "0" and "1" (which may represent, for example, "dead" vs. "alive" or "win" vs. "loss").
Multinomial logistic regression deals with situations where the outcome can have three or more possible types (e.g., "disease A" vs. "disease B" vs. "disease C") that are not ordered. Ordinal logistic regression deals with dependent variables that are ordered.

Below is an example logistic regression equation:

$$y = e^\wedge(b0 + b1*x) / (1 + e^\wedge(b0 + b1*x))$$

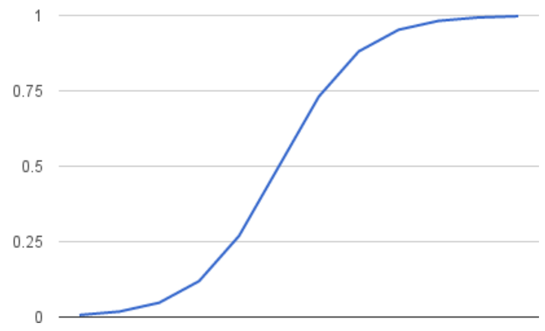


Figure 2.1.2: Sigmoid Function is used by logistic regression.

2.2 Related Works

The Repeat Buyers Prediction dataset is used in the base paper to construct and validate the proposed ensemble classifier. The dataset available does not have many features and hence feature engineering is applied to get many features. The works in [1] and [2] illustrate feature engineering on the Repeat Buyers Prediction Dataset Using the activity log section of the dataset, 9 features are engineered. These include:

- The ratio of total number of clicks to the total number of actions
- The ratio of total number of items added to cart to the total number of actions
- The ratio of total number of items purchased to the total number of actions
- The ratio of total number of items added to favorites to the total number of actions
- The number of distinct days of interaction of a customer with a merchant
- The number of distinct days on which items were purchased by a customer from a merchant
- The number of distinct items bought
- The number of distinct category of items bought
- The number of distinct brands of the items bought

Thus the above features and the age and gender are aggregated together to get the actual dataset which is used to train the classifiers.

The python documentation for scikit-learn is also referred and classifiers are implemented using the Python scikit-learn module [4]. The XGBoost module documentation is referred to implement the XGBoost classifier [3].

2.3 Problem Statement

To implement an ensemble method for binary classification which is adaptive to each of the testing data and uses the best classifier for each test data to predict its class label as proposed in the base paper and compare the performance of the proposed ensemble classifier with the single classifiers used to construct the ensemble. The dataset used is the Repeat Buyers Prediction dataset.

2.4 Objectives

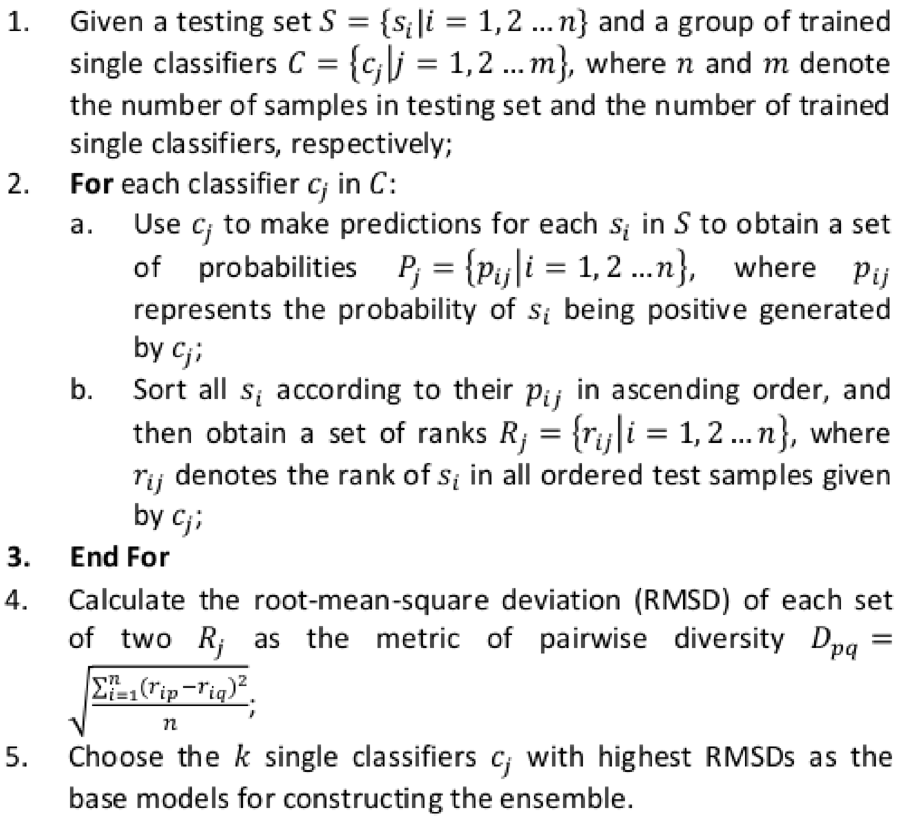
The ensemble model involves the selecting of base models from the single trained classifiers and use their predictions to train a meta model with the labels relabeled with the classifier names.

- Select base models that are based on the pairwise diversity of the classifiers
- Recognize the best base model for each data instance
- Predict each unknown instance using a suitable base model (best base model for that instance)

3. Proposed Methodology

- The preprocessing of the dataset is done using pandas
- Then feature engineering as described in the related works section is applied. Thus the final dataset has 11 features
- The feature engineered dataset is first sampled using stratified sampling method as the classes in the dataset are imbalanced (the label ‘1’ is sparse)
- Then individual classifiers, Adaboost, XGBoost, Random Forest and Logistic Regression are trained using the training set of the sampled dataset and their individual F1 scores and AUC values are noted
- The using the testing dataset, base models are selected using the following five methods:

I. Root Mean Square Deviation



1. Given a testing set $S = \{s_i | i = 1, 2 \dots n\}$ and a group of trained single classifiers $C = \{c_j | j = 1, 2 \dots m\}$, where n and m denote the number of samples in testing set and the number of trained single classifiers, respectively;
2. **For** each classifier c_j in C:
   a. Use c_j to make predictions for each s_i in S to obtain a set of probabilities $P_j = \{p_{ij} | i = 1, 2 \dots n\}$, where p_{ij} represents the probability of s_i being positive generated by c_j;
   b. Sort all s_i according to their p_{ij} in ascending order, and then obtain a set of ranks $R_j = \{r_{ij} | i = 1, 2 \dots n\}$, where r_{ij} denotes the rank of s_i in all ordered test samples given by c_j;
3. **End For**
4. Calculate the root-mean-square deviation (RMSD) of each set of two R_j as the metric of pairwise diversity $D_{pq} = \sqrt{\frac{\sum_{i=1}^{n}(r_{ip}-r_{iq})^2}{n}}$;
5. Choose the k single classifiers c_j with highest RMSDs as the base models for constructing the ensemble.

II. Zero-One Loss diversity based selection:

Here the base models are selected to be the single trained models with the highest and the least zero-one loss.

III. F1 score based selection:

Here k single trained classifiers with the highest F1 scores are selected as the base models.

IV. AUC based selection:

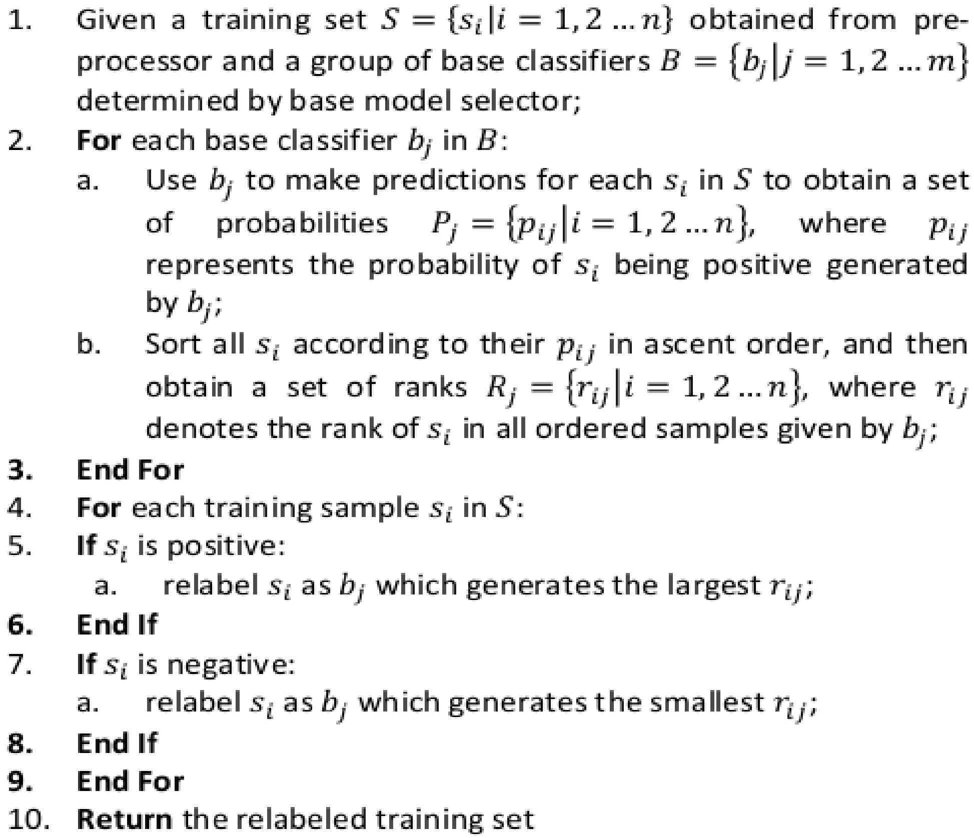
Here k single trained classifiers with the highest AUC values are selected as the base models.

V. Agglomerative Clustering:

Here the ranks of each single classifier calculated from the first part of the RMSD algorithm is given as the input to an agglomerative clustering model which clusters the most similar classifiers together and hence only one classifier can be selected from each cluster. The number of clusters for clustering equals the number of base model classifiers.

The base models are selected in such a way that the selected base models have high pairwise diversities.

- Relabeling of the train-set is done using the base classifiers selected in the previous step. The algorithm is shown below:



```
1.  Given a training set $S = \{s_i | i = 1, 2 \dots n\}$ obtained from pre-processor and a group of base classifiers $B = \{b_j | j = 1, 2 \dots m\}$ determined by base model selector;
2.  **For** each base classifier $b_j$ in $B$:
    a.  Use $b_j$ to make predictions for each $s_i$ in $S$ to obtain a set of probabilities $P_j = \{p_{ij} | i = 1, 2 \dots n\}$, where $p_{ij}$ represents the probability of $s_i$ being positive generated by $b_j$;
    b.  Sort all $s_i$ according to their $p_{ij}$ in ascent order, and then obtain a set of ranks $R_j = \{r_{ij} | i = 1, 2 \dots n\}$, where $r_{ij}$ denotes the rank of $s_i$ in all ordered samples given by $b_j$;
3.  **End For**
4.  **For** each training sample $s_i$ in $S$:
5.  **If** $s_i$ is positive:
      a.  relabel $s_i$ as $b_j$ which generates the largest $r_{ij}$;
6.  **End If**
7.  **If** $s_i$ is negative:
    a.  relabel $s_i$ as $b_j$ which generates the smallest $r_{ij}$;
8.  **End If**
9.  **End For**
10. **Return** the relabeled training set
```

- Meta-model training (Stacking Ensemble):
  A meta-dataset is created using the predictions of the base model classifiers for each train tuple. Hence the new dataset is of the shape: (number of train tuples X number of base classifiers). Then this meta-dataset with relabeled classes is trained using Logistic Regression. The test tuples are also modified as the train tuples and the meta model classifier (Logistic Regression) if used to predict the class labels (relabeled classes - classifiers).

- Final Prediction:
  Each test-tuple is predicted with the classifier, whose label is predicted by the meta model classifier.

- The F1 scores and AUC values are calculated for the proposed ensemble method and compared with the best base classifier.

4. Conclusion and Future Works

In this project, a novel ensemble method is implemented, which is capable of performing adaptive selection of the best base model for each unknown instance. The proposed method is validated on RBP dataset, and the performance of AUC and F 1 score was compared with those of other two existing ensemble techniques (Averaging and Stacking). The proposed ensemble method has an overall performance improvement in terms of F 1 score, which is considered a more valuable metric in practice. Moreover, a higher pairwise diversity of combined base models can lead to a further improvement toward the performance of ensemble model. The future works in this project include:

- Engineer more features from the RBP dataset to increase the accuracy of classifiers further.
- Implement other modes of hierarchical clustering and compare its performance with the results obtained.
- Use more single classifiers to get a more diverse ensemble.

5. References

[1] Liu, G, et al. “Report for Repeated Buyer Prediction Competition byTeam 9*TAR.” In Proceedings of the 1st International Workshop onSocial Influence Analysis Soclnf 2015.

[2] He, B, et al. “Repeat Buyers Prediction after Sales Promotion for Tmall Platform.” In Proceedings of the 1st International Workshop on Social Influence Analysis Soclnf 2015.

[3] xgboost.readthedocs.io/en/latest/

[4] scikit-learn.org/stable/

[5] machinelearningmastery.com/implementing-stacking-scratch-python/