EE 660 Course Project Report
Student: Lijing Yang
Student E-mail: lijingy@usc.edu
Student In Group: 1
Project Type: (1) Design a system based on real-world data
Instructor: Keith Jenkins
Date: Dec/02/2022

Client Prediction of Bank Subscription Term Deposits Using Supervised and Semi-Supervised Machine Learning Approaches

Abstract:

For banks, a customer's term deposit is a significant cash flow. With a constant cash flow, the bank can expand the market and investment further. The primary purpose of this project is to develop a customer classification system using machine learning, which aims to distinguish from personal information whether they will subscribe to a term deposit. This project uses real customer data from a bank in Portugal. The original data set has 20 inputs and one output with a total of 45211 data [1]. The project will first use preprocessing methods such as extracting features and balancing data to reduce the dataset to nine low-correlation features to ensure the model is accurate. Then the system will use eight supervised learning methods, a total of eight models, and two semi-supervised learning methods, a total of nine models, to analyze the data and explore the performance of supervised and semi-supervised learning for the data under different amounts of labeled data. The result shows that Supervised Linear Discriminant Analysis and Gradient Boosting Classifier have the most robust performance with AUC=0.95 when there is a large amount of labeled data as the training set. The Semi-Supervised Self-Learning K-Nearest Neighbor and Semi-Supervised Label Propagation Classification have the best performance in Semi-Supervised Learning with AUC=0.93. When there is a small amount of labeled data as the training set, only supervised SVM achieves higher performance than semi-supervised SVM, and other models are supervised better than semi-supervised.

Introduction:

Problem Type, Statement and Goals:

How to mine valuable information from messy data? In the past, it may have required years of training for experienced data analysts to discover useful information from thousands of data. Nevertheless, as more and more industries need data analysis, how to use computers to make fast and accurate judgments on large amounts of data has become an important subject. In this project, I got a copy from a Portuguese banking institution's direct marketing campaigns (phone calls) to judge whether the client would subscribe to a term deposit. In this data set, there are 20 features from customers, and one output is whether to subscribe to a term deposit. The banks are essential to determine whether customers will subscribe to a term deposit in time [1]. Because the bank's continuous cash flow represents a steady income stream, the timely discovery of potential customers can significantly increase the bank's income. In the past, such analysis had to be analyzed by experienced analysts. However, in this project, I will use commonly used machine learning algorithms such as K-Nearest Neighbor Classification (KNN), Logistic Regression,

Gaussian Naive Bayesian Classification, Support Vector Machine (SVM), Decision Tree, Random Forest, Gradient Boosting Classification, Linear Discriminant Analysis Classification (LDA) and other methods try to classify this data and analyze model performance such as ROC curve and ACU.

However, with the increase in data, new problems arise. In many cases, computers must have labeled data to learn from it. However, the original data is often unlabeled, so a lot of people and time must be wasted in the data preprocessing stage to classify the output values manually. The purpose of introducing machine learning for classification is to reduce the proportion of human decision-making in the analysis. However, we need to increase people's decision rate during preprocessing, which contradicts our philosophy. Therefore, in the second half of the project, I will also choose representative models and use semi-supervised learning to test the performance of supervised learning and semi-supervised learning under different amounts of labeled data to explore which classifier performs by using the ROC curve and ACU to find out which classifier is best for this classification problem.

Our Prior and Related Work: None

Overview of Our Approach:

Main topic:

- The trivial baseline: Randomly select one of the two classes (a red dotted line with AUC=0.5 in the figure)
- The non-trivial baseline: K-Nearest Neighbor Classification (represented by the orange line on the graph)

Extension:

- The baseline system: Randomly select one of the two classes (a red dotted line with AUC=0.5 in the figure)

Implementation:

Data Set:

The dataset I chose comes from the UCI Bank Marketing Data Set, which has 20 features extracted from customers and one output. Its features include [1]:

1. age (numeric)
2. job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
3. marital (categorical: 'divorced', 'married', 'single', 'unknown')
4. education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
5. default: has credit in default? (categorical: 'no', 'yes', 'unknown')
6. housing: has housing loan? (categorical: 'no', 'yes', 'unknown')
7. loan: has personal loan? (categorical: 'no', 'yes', 'unknown')
8. contact: contact communication type (categorical: 'cellular', 'telephone')
9. month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10. day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11. duration: last contact duration, in seconds (numeric).
12. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

14. previous: number of contacts performed before this campaign and for this client (numeric)
15. poutcome: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')
16. emp.var.rate: employment variation rate - quarterly indicator (numeric)
17. cons.price.idx: consumer price index - monthly indicator (numeric)
18. cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19. euribor3m: euribor 3 month rate - daily indicator (numeric)
20. nr.employed: number of employees - quarterly indicator (numeric)

According to the hint from the dataset website, the feature 'duration' highly affects the output target because if the duration is 0, then y must be 'no.' Moreover, y is known after the call. Therefore, this feature does not apply to this classification project, so this feature is dropped. Use DataFrame's built-in formula df.describe() to generate a statistical table to make the data distribution more intuitive, as shown in the following figure:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
count	41176.00000	41176	41176	41176	41176	41176	41176	41176	41176	41176	41176.000000	41176.000000	41176.000000	41176	41176.000000	41176.000000	41176.000000	41176.000000	41176.000000	41176
unique	NaN	11	3	7	2	2	2	2	10	5	NaN	NaN	NaN	3	NaN	NaN	NaN	NaN	NaN	2
top	NaN	admin.	married	university.degree	no	yes	no	cellular	may	thu	NaN	NaN	NaN	nonexistent	NaN	NaN	NaN	NaN	NaN	no
freq	NaN	10749	25001	13894	41173	22561	34928	26135	13767	8618	NaN	NaN	NaN	35551	NaN	NaN	NaN	NaN	NaN	36537
mean	40.02380	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.567879	962.464810	0.173013	NaN	0.081922	93.575720	-40.502863	3.621293	5167.034870	NaN
std	10.42068	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.770318	186.937102	0.494964	NaN	1.570883	0.578839	4.627860	1.734437	72.251364	NaN
min	17.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.000000	0.000000	NaN	-3.400000	92.201000	-50.800000	0.634000	4963.600000	NaN
25%	32.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	999.000000	0.000000	NaN	-1.800000	93.075000	-42.700000	1.344000	5099.100000	NaN
50%	38.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.000000	999.000000	0.000000	NaN	1.100000	93.749000	-41.800000	4.857000	5191.000000	NaN
75%	47.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.000000	999.000000	0.000000	NaN	1.400000	93.994000	-36.400000	4.961000	5228.100000	NaN
max	98.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	56.000000	999.000000	7.000000	NaN	1.400000	94.767000	-26.900000	5.045000	5228.100000	NaN

Fig.1: Statistical Analysis of Features

It can be seen from Fig.1 that the distribution of output results is very unsatisfactory because 'yes' and 'no' are extremely unbalanced. Therefore, the built-in formula df.hist() of DataFrame can be used to draw histograms further to explore the distribution of each feature and output, as shown in Fig.2.

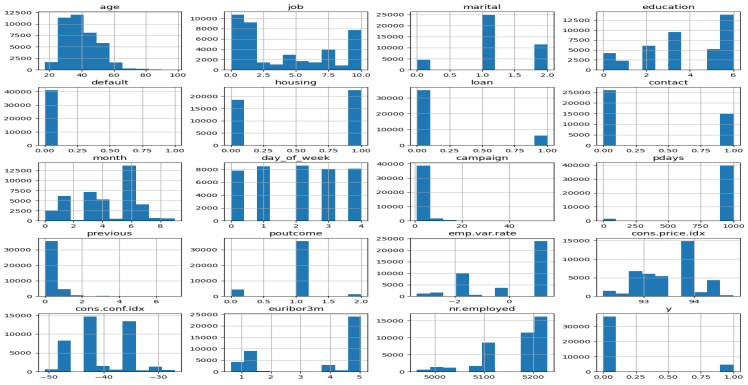

Fig.2: Feature histogram distribution

Dataset Methodology:

A total of 48,713 labeled data sets were generated after preprocessing, but the amount of data is too much for simple classification problems [1]. Therefore, a relatively large data set was selected as the test set in this project, and 50% of the entire data set was selected. Then 10,000 of the remaining 50% of the data sets are randomly selected to generate a new data set as the training set for all models. Because this project is divided into supervised learning and semi-supervised learning, the generated training set of 10,000 labeled data needs to be further divided to generate a labeled and unlabeled data training set. In this project, 5000 labeled data were selected as training sets one, and 5000 unlabeled data were selected as training set two for analyzing the performance of supervised and semi-supervised learning with high labeled data volume. To compare the performance changes of supervised learning and semi-supervised learning in the low amount of

labeled data, the labeled data training set of 7, 15, 30, 100, 500, 2000 data and 9993, 9985, 9970, 9900, 9500, 8000 data The unlabeled training set of is used.

To ensure that models such as supervised and semi-supervised decision trees, random forests, and gradient boosting can obtain the most suitable parameters for this data set, the training set not only acts as a training set but also as validation sets to help the model identify the best parameters. GridSearchCV() is used to compare the 5-fold cross-validation of different parameters to confirm which model parameter has the best performance.

In addition to the training set, the test set is also an essential data set because it is the core that reflects the quality of the model. The test set will separate into features and outputs. After the model is trained, it will use the features of the test set to make predictions and compare the model prediction results with the actual output and judge the quality of the model by calculating the exact score and AUC score and drawing the ROC curve

Preprocessing, Feature Extraction, Dimensionality Adjustment:

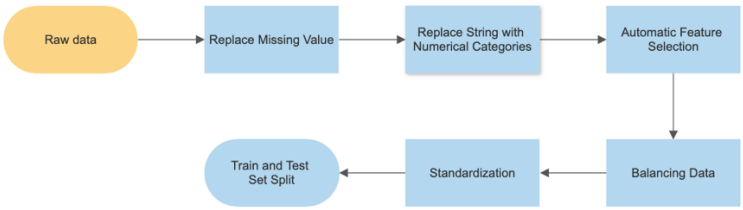

Fig.3 Preprocessing flow chart

Replace Missing Value:

After importing the data set, keep one data point and delete the duplicate data of the entire data set. According to the UCI database's description, it is claimed that this is a data set with 20 features and 41188 data points without missing values [1]. However, during the preprocessing of the data set, it was found that there were missing values in this data set. Here two methods of dealing with missing values are used. The first is to remove the entire data point containing missing values directly. The number of missing values in the respective features is rare, about 1%-3%. However, there would be about 10,000 data losses if all of them were deleted, which is 25% of the entire data set, leading to distortion of the data set, so this method was eventually abandoned. However, since there are not many missing values in each feature, the method chosen here is to use the SimpleImputer() formula to replace the missing values with the data with the highest frequency in each feature.

Replace String with Numerical Categories:

Machine learning algorithms cannot use string category features. Therefore, in this step, it needs to use sklearn.preprocessing.LabelEncoder() to convert the original string category into a digital category.

Automatic Feature Selection:

Fig.2 shows that many features correlate, such as euribor3m and nr.employed. By drawing the heatmap of the correlation, it is found that there are features with very high correlation, as shown in Fig.3. So at this step, the correlation-based automatic feature selection API featurewiz() with correlation Limit = 0.8 is used to select the most suitable features. Featurewiz is an open-source feature selection API whose purpose is to select the best features by SULOV algorithm and

Recursive XGBoost. Featurewiz will first use SULOV to find features with low correlation and high mutual information, then Recursive XGBoost will find the best features among the remaining features [2]. After applying featurewiz(), the number of features is reduced from 20 to 9. The new correlation heatmap does look a lot 'clean' as shown in Fig.4

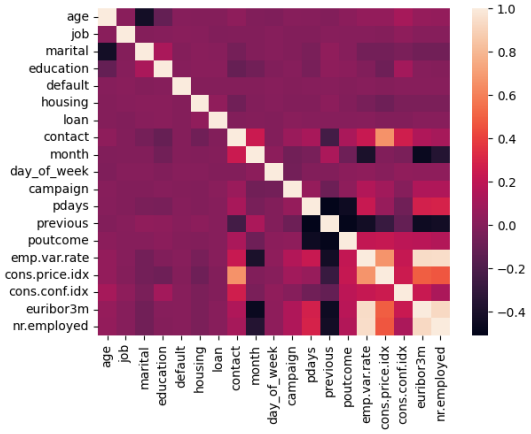

Fig.4 Correlation Heatmap Before Select

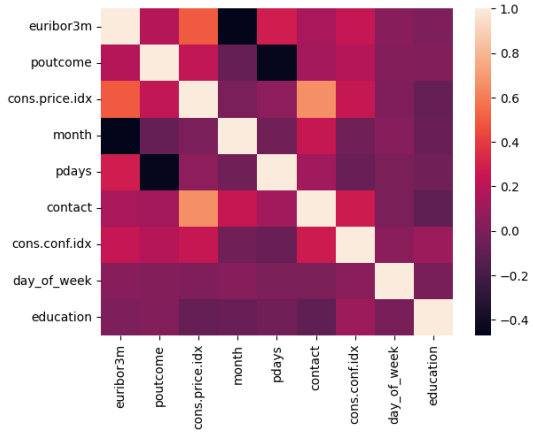

Fig.5 Correlation Heatmap After Select

Balancing Data:

The data set is highly unbalanced by observing the histogram of the Y value in Fig.2. To get almost the balanced training and test sets for each output, imblearn.combine.SMOTEENN() was used to reduce the number of 'No' and increases the number of 'Yes'. SMOTEENN is a combination of Synthetic Minority Oversampling TEchnique (SMOTE) and Edited Nearest Neighbors (ENN), an algorithm that can handle imbalanced data using both oversampling and undersampling [3]. SMOTE is mainly responsible for oversampling. Its principle is to draw a line between different points through K-nearest neighbors in the feature space and generate a new sample at a point along the line [4]. ENN is mainly responsible for downsampling, as Raden said: "the ENN method works by finding the K-nearest neighbor of each observation first, then check whether the majority class from the observation's k-nearest neighbor is the same as the observation's class or not. If the majority class of the observation's K-nearest neighbor and the observation class is different, then the observation and its K-nearest neighbor are deleted from the dataset [3].” Therefore, SMOTEENN can increase the number of 'no' labels through the SMOTE and delete ungroupable data through the ENN algorithm, as shown in Fig.5, which will significantly improve the model's performance when using this data set.

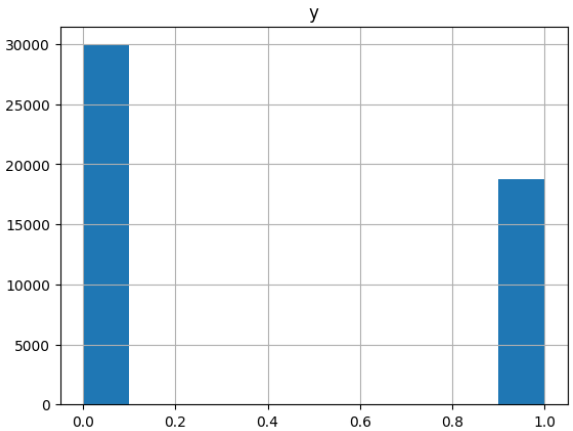

Fig.6 Dataset Label Distribute After Balancing

Numerical Value Standardization:

Because features involve continuous numerical data, it is necessary to standardize all continuous numerical data to ensure that the data can be used by models that calculate distance, such as K nearest neighbors and SVM. Then the preprocessing part ends here, and the original data set is ready for the test and training set split.

Training Process:

This project used multiply supervised and semi-supervised methods to increase the diversity of experiments. Such as the most basic K-nearest neighbors, logistic regression, naive Bayesian classifiers; and advanced classifiers such as decision trees, random forests, support vector machines, gradient boosting, and Linear Discriminant Analysis (LDA) and their corresponding self-training semi-supervised classifiers and label propagation classifiers. The reason for choosing these models is that they cover almost from the beginning to the end of the EE660. And these models are now widely used in academia and industry, so they are selected here as comparison models. The overfitting problem is also a very worthwhile consideration during the experiment. Because once the model is overfitted, the model's performance will start to decline, which cannot be tolerated in a project that focuses on improving model performance. However, thanks to the deletion of the outlier by SMOTEENN and the proportion of labeled data points being tiny, it is unlikely to be overfitting. It is observed that the model still maintains a very high train and test set accuracy under the maximum number of 5000 labeled data, and the two values are nearly the same. This proves that there is no overfitting even when the training set has the most significant number of labeled data.

* If no parameter is mentioned in the text, it is the default parameter.

K-Nearest Neighbor Classifier (KNN):

KNN may be the first classification method that everyone comes into machine learning. And in this project, KNN's ROC curve is used as a non-trivial baseline for comparison with other models. The way it distinguishes categories is straightforward, mainly by observing which category n nearby points belong to, then it belongs to this category. Here, n=5 is chosen. The reason for choosing this model is that it is simple and widely used, and it serves as a non-trivial baseline in this project.

Logistic regression:

Logistic regression is also a very classic entry classifier. Compared with KNN, logistic regression is more "mathematical". The principle is to obtain a value through linear regression and pass it through the Sigmoid function to get an output with a minimum value of 0 and a maximum value of one. This 0-1 output can determine the probability of its belonging to the category.

Naive Bayes classifier:

The Naive Bayesian classifier, like logistic regression and KNN, is also a primary entry classifier. However, it is wholly based on the probability theory algorithm, so it is also selected. The basic idea is to calculate the posterior probability through Bayesian theory by prior probability, usually used in spam filtering.

Support Vector Machines (SVM):

The principle of support vector machines is to find a hyperplane in N-dimensional space that can classify data points. There are many hyperplanes in two categories, but SVM only focuses on the plane with the most significant margin. The maximum margin makes the hyperplane exactly between the two classes, so more confidence can be gained when making data point predictions [5]. SVM is widely used in classification tasks, and its corresponding S3VM is also very famous, so it is selected as a comparison model here. The parameters used in this project are probability=True, gamma="auto" so that gamma is set to 1/n_features, and a built-in 5-fold cross-validation can be obtained to improve model performance further.

Decision tree:

Unlike all the above methods, decision trees implement classification by formulating rules. It is the first time the new algorithm was introduced in EE660 after EE559. The principle is to start with each feature, find a value in a feature that can divide the data into two categories with the smallest cost, and then continue this action until the classification is completed. From the description, if the depth of the decision tree is not artificially set, there is a high possibility of overfitting. So, 5-fold cross-validation is used to select that parameter among two classification methods [best', 'random'], two calculation loss methods ["gini","entropy"], and a decision tree depth of 1-20.

Random Forest:

Random forest is equivalent to an advanced version of decision tree. The principle is a bunch of decision trees; each decision tree will return a category with the most votes as the result of the random forest. The random forest is better than the decision tree because the classification result may be wrong. But as more and more classifiers are added, some classifiers may be inaccurate while others are correct. So, if all the decision trees don't make mistakes in the same direction, the random forest will have better accuracy [6]. Also, to find the most suitable parameters, 5-fold cross-validation is used here in two classification methods [best', 'random'], two calculation loss methods ["gini","entropy"], and [5, 10,20,40] n_estimators select the best performing parameters.

Gradient boosting:

The gradient boosting classifier is also a classifier involved in EE660. All the classifiers currently strong leaner in the above are capable of learning, so severe overfitting may occur. Gradient boosting classifiers are algorithms that aggregate multiple predictors and data training from a lower-accuracy classifier to a higher-accuracy classifier, often performing better than a single model. Avgif mentioned that in gradient boosting, each predictor tries to reduce the previous predictor's error. it does not simply fit a classifier on the data through each iteration, but the residual generated by the previous classifier fits a new classifier on the difference [7]. 5-fold cross-validation is also used here, and the best-performing parameters are selected in the maximum depth of 1-10.

Linear Discriminant Analysis (LDA):

LDA, like PCA, is usually used to reduce the dimension of the data during preprocessing. Its purpose is mainly to project the features in the high-dimension space to the low-dimension to minimize the resources used. Its principle is to map all data points to a line, but this line needs to minimize the variance and maximize the distance between categories. Compared with logistic

regression, LDA has better stability and performance when there are fewer data points and lower system complexity [8]. It is therefore chosen here as a purely probabilistic method classification model in advanced algorithms.

Model Selection and Comparison of Results:

In this project, only a few models, such as decision trees, random forests, and gradient boosting methods, must choose the appropriate parameters to generate the model. The technique used here is 5-fold cross-validation to determine the best parameter by finding the highest f1 score of each parameter.

Final Results and Interpretation:

ROC curves and AUCs from 8 models were obtained after running all supervised learning algorithms, the selected four models, Logistic regression, SVM, Random Forest, and Gradient Boosting, as shown in Fig.7, Fig.9, Fig.11, and Fig.13. Among all 8 models, KNN is 0.95, logistic regression is 0.90, naive Bayesian is 0.88, SVM is 0.92, the decision tree is 0.95, random forest is 0.98, gradient boosting is 0.98, and LDA is 0.90. This result is far beyond my expectation. I chose KNN as the baseline for all models because it is a simple algorithm, and I have minimal expectations for it to perform well. But now it is the simplest model and has unexceptionally high performance. And random forest and gradient boosting surpass other models to become the most performant models, which aligns with my expectations because they both use 5-fold cross-validation to select the most appropriate parameters. And the random forest is based on a large-scale decision tree; gradient boosting is based on learning through accumulating many weak learners. Judging on the time complexity of learning is much higher than other algorithms, so they should be the first. And LDA, naive Bayes, and logistic regression are also reasonable as the worst models in this project. Because most of them are mathematical or based on probability theory, although it saves calculation time, purely using probability theory in model will cause a lot of unexpected mistakes. Among all the models, KNN and gradient boosting have higher performance because decision trees and random forests take at least ten minutes to run a result, but KNN and gradient boosting can be completed within one minute with the same performance. Based on the principle that the time consumed is also an indicator of performance, KNN and gradient boosting are better. But it is easy to see that 5000 labeled data points are too many, resulting in the accuracy of supervised learning already having a good performance. So, in the following comparative experiment, I will reduce the number of labeled data to obtain supervised and semi-supervised learning to compare different amounts of labeled data.

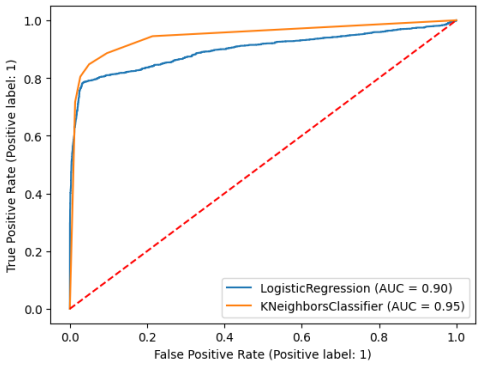

Fig.7 Logistic Regression ROC Curve

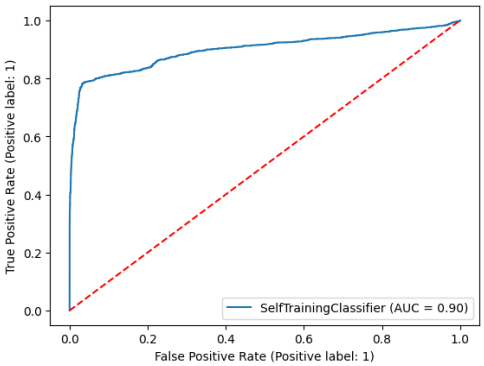

Fig.8 Self-Train Logistic Regression ROC Curve

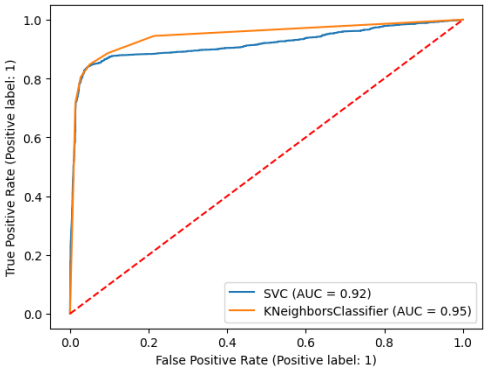

Fig.9 SVM ROC Curve

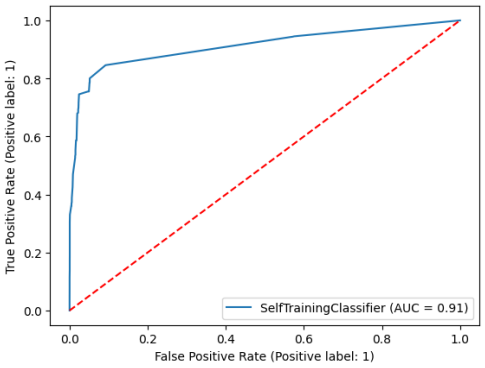

Fig.10 Self-Train Random Forest ROC curve

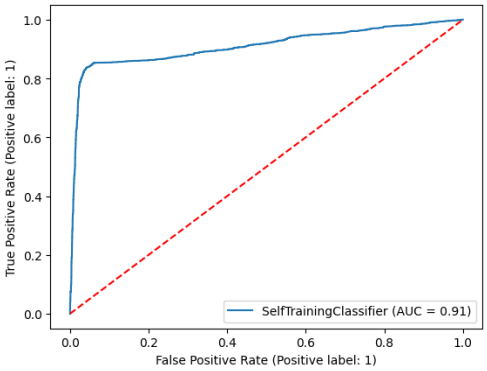

Fig.11 Self-Train SVM ROC Curve

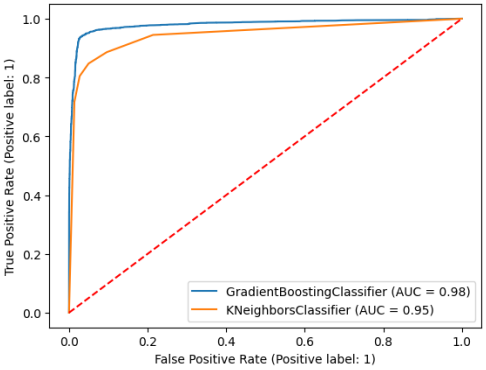

Fig.12 Gradient Boosting ROC Curve

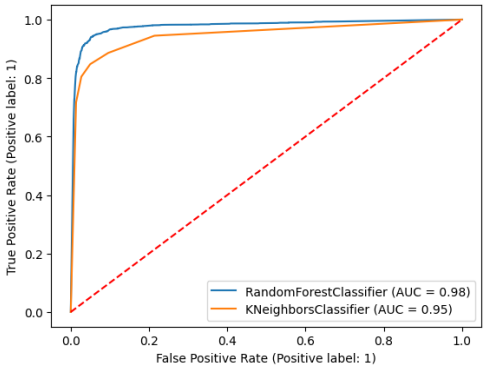

Fig.13 Random Forest ROC Curve

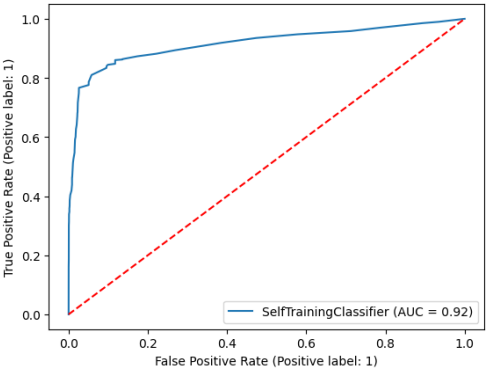

Fig.14 Self-Train Gradient Boosting ROC Curve

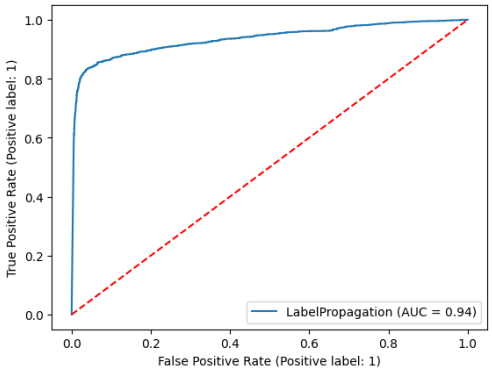

Fig.15 Semi-Supervised Label Propagation ROC Curve

Implementation for the extension:

In this project, two semi-supervised learning methods are used: self-training semi-supervised learning and label propagation semi-supervised learning (SSL). The labeled data used here is the same training set as the supervised learning. The only difference is the unlabeled training set, but how to make the unlabeled training set has been explained in detail in the third part.

Self-Training Semi-Supervised Learning:

Self-training semi-supervised learning first uses a basic supervised learning model as the base model. This base model is then trained on labeled data and used to predict labels for unlabeled datasets. Then the data points with a higher probability of correct classification are filtered out from the unlabeled dataset and given a soft label [9]. This process allows the original labeled dataset and the soft-labeled unlabeled dataset to generate a larger labeled dataset. This updated training set is then used to train basic supervised learning again. Because the previous steps used eight basic supervised learning methods, their corresponding self-training semi-supervised learning models are still used in this step.

Label Propagation Semi-Supervised Learning:

Propagation semi-supervised learning is a graph-based algorithm that inserts many labeled data in an unlabeled data plane. Each point is then linked to the other to generate an interconnected network. And assign a weight to each connection line, which will be distributed according to the shorter the connection distance, the greater the weight. Finally, walk through all the unlabeled points to find the probability distribution of reaching the labeled points. After walking all the lines, assign labels to the unlabeled points according to the probability distribution found on the way [10].

Final Results and Interpretation for the extension:

In this step, I drew a total of 9 ROC curves, eight correspond to supervised learning, and one is to label propagation semi-supervised learning. The selected five models, SSL Logistic regression, SSL SVM, SSL Random Forest, SSL Gradient Boosting, and Mark Propagation, as shown in Fig.8, Fig.10, Fig.12, Fig.14, and Fig.15. Among all nine models, SSL KNN is 0.94, SSL logistic regression is 0.90, SSL Naive Bayesian is 0.87, SSL SVM is 0.91, the SSL decision tree is 0.84, SSL random forest is 0.91, SSL gradient boosting is 0.92, SSL LDA is 0.89, and marker propagation is 0.94. This result is far beyond my expectation because, theoretically, when you get more points, the SSL performance should be much higher than SL, at least not lower. Nevertheless, it is found that the AUC of each semi-supervised learning is 0.01 lower than the corresponding SL. However, they both use the same labeled data; therefore, the SL part of the SSL self-train should be the same. So, the model's performance degrades due to the use of unlabeled data. This is mainly because the original model has some errors when predicting the label of unlabeled data. Therefore, during the second training, the model's performance will reduce. The principle of label propagation is very similar to KNN, so the AUC scores of the two are the same. But if we now reduce the original labeled data to try and see what happens. Here I chose 7, 15, 30, 100, 500, and 2000 labeled data and chose KNN, SSL KNN, SVM, SSL SVM, Random Forest, SSL Random Forest, and Label Propagation as comparison models for different numbers of labels. The ROC curve generated under the data is shown in the following:

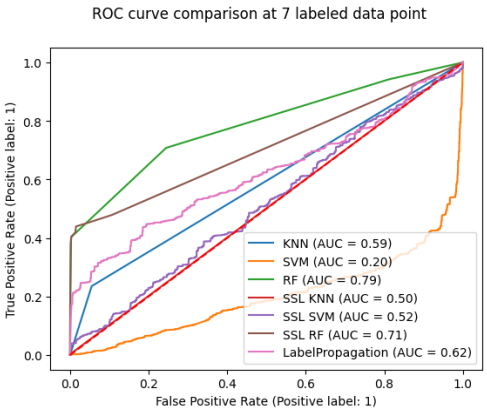

Fig.16 ROC Curve for 7 Labeled Data

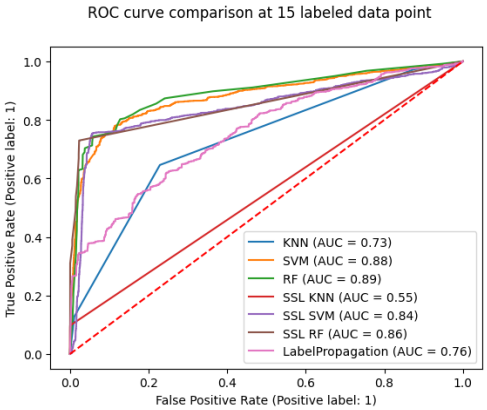

Fig.18 ROC Curve for 15 Labeled Data

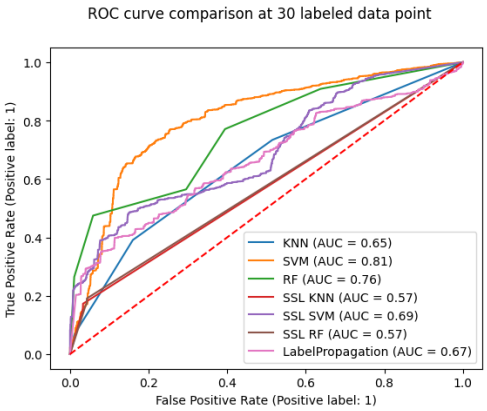

Fig.20 ROC Curve for 30 Labeled Data

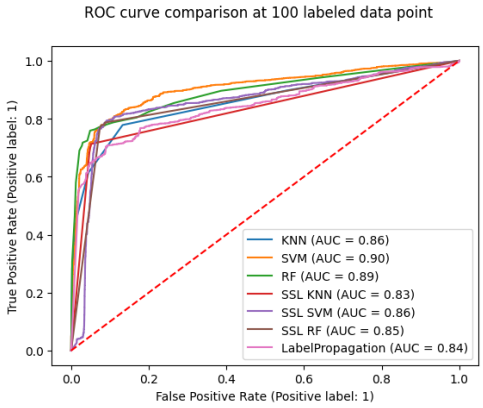

Fig.17 ROC Curve for 100 Labeled Data

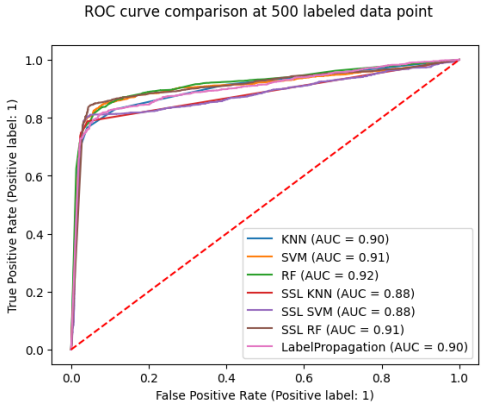

Fig.19 ROC Curve for 500 Labeled Data

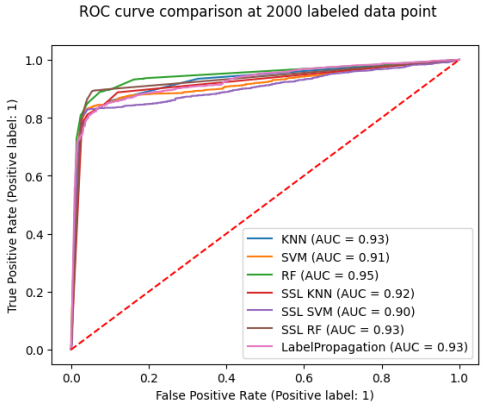

Fig.21 ROC Curve for 2000 Labeled Data

It can be seen from Fig.16 that when there are only seven labeled data, KNN, SSL KNN, and SSL SVM are very close to the red baseline in the middle, so they randomly select a class. But we can also see that the result of SVM at this time is even worse than random, which means that SSL SVM does improve performance when the labeled data is limited. But in other self-training models, SSL is still worse than SL. However, all SL models outperformed SSL in Fig.18 with only 15 labeled data. Then continuing the increase in the number of labels, the scattered ROC curve is gradually approaching, and each model is approaching one progressively, but SSL is still weaker than SL. Therefore, this experiment found that SSL SVM performs better than the original SVM when the labeled data is limited. But for other models, SSL cannot provide a higher prediction

when using a small amount of labeled data, and the performance of SSL and SL is nearly the same when using a large scale of labeled data.

Summary and conclusions:

This project used more than 17 models, and many basic models were found to have problems with self-training and semi-supervised learning. For example, in theory, semi-supervised learning will significantly improve the performance of supervised learning when there are few labeled data. However, in the actual experiment, only semi-supervised SVM has higher performance than supervised SVM in low-number labels. In contrast, the performance of most classifiers is supervised higher than that of semi-supervised. This is because the number of labels is tiny, so the error is further exacerbated during self-training. Therefore, all semi-supervised learning has poor performance compared with supervised learning. The label propagation is also poorly performed because the data set does not have a clear boundary dividing data points into two classes. If there is a clear boundary, those algorithms that use distance as the classification standard will become more accurate, such as semi-supervised KNN, semi-supervised SVM, and label propagation. An exciting direction to extend this project is to use both self-training and label propagation semi-supervised methods. For example, training a semi-supervised SVM with few data points will likely get an unsatisfactory hyperplane. But the new method can first select some unlabeled data and propagate nearby labels to it, then use these points to do semi-supervised SVM. Finally, it may get better results after several iterations.

References

[1] UCI Machine Learning Repository: Bank Marketing Data Set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. [Accessed: 09-Dec-2022].

[2] D. David, “Automatic feature selection in Python: An essential guide,” HackerNoon, 20-Jul-2021. [Online]. Available: https://hackernoon.com/automatic-feature-selection-in-python-an-essential-guide-uv3e37mk. [Accessed: 09-Dec-2022].

[3] R. A. A. Viadinugroho, “Imbalanced classification in Python: Smote-enn method,” 30-Sep-2021. [Online]. Available: https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50. [Accessed: 09-Dec-2022].

[4] J. Brownlee, “Smote for imbalanced classification with python,” 16-Mar-2021. [Online]. Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. [Accessed: 09-Dec-2022].

[5] R. Gandhi, “Support Vector Machine - introduction to machine learning algorithms,” 05-Jul-2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed: 09-Dec-2022].

[6] T. Yiu, “Understanding random forest,” 29-Sep-2021. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed: 09-Dec-2022].

[7] V. Aliyev, “Gradient boosting classification explained through python,” 07-Oct-2020. [Online]. Available: https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d. [Accessed: 09-Dec-2022].

[8] P. Sarkar, “What is LDA: Linear Discriminant Analysis for Machine Learning,” 01-Dec-2022. [Online]. Available: https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning. [Accessed: 09-Dec-2022].

[9] D. Steen, “A gentle introduction to self-training and semi-supervised learning,” 31-Aug-2020. [Online]. Available: https://towardsdatascience.com/a-gentle-introduction-to-self-training-and-semi-supervised-learning-ceee73178b38. [Accessed: 09-Dec-2022].

[10] S. Dobilas, “Semi-supervised learning - how to assign labels with label Propagation Algorithm,” 05-Feb-2022. [Online]. Available: https://towardsdatascience.com/semi-supervised-learning-how-to-assign-labels-with-label-propagation-algorithm-9f1683f4d0eb. [Accessed: 09-Dec-2022].