CS 559: Machine Learning: Fundamentals and Applications
Fall 2025 Project: Company Bankruptcy Prediction

1 Objective

The performance of logistic regression applied to the **original train data** is presented in Table 1. While the prediction of the target is highly accurate according to the standard metric, identifying the companies that will file for bankruptcy ($y = 1$) is very poor due to the imbalance of the data (see Figure 1).

data set	FF(FT)	TT(TF)	standard accuracy	Eq (1) accuracy
train	5570(39)	5(193)	0.960	0.025
test	983(7)	0(22)	0.971	0

Table 1: Logistic Regression Result Table without any modification. The second and third columns present the number of companies' true status (y) and predicted status ($\hat{h}$) as FF - ($y = 0, \hat{h} = 0$), FT - ($y = 0, \hat{h} = 1$), TT - ($y = 1, \hat{h} = 1$), and TF - ($y = 1, \hat{h} = 0$). The accuracy score in the fourth column is evaluated as (FF+TT)/(FF+FT+TT+TF). The accuracy score in the fifth column is evaluated using Equation (1).

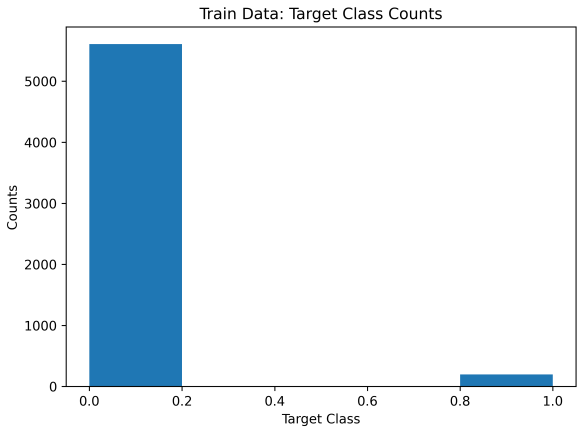

Figure 1: Train Data Target Distribution

The modeling objective is to identify the companies that will file for bankruptcy. Hereafter, the **accuracy score** (acc) is defined as how successfully the model identifies the company that files for bankruptcy as follows

$$acc = \frac{TT}{TF + TT}. \tag{1}$$

We add the constraint that **the number of test examples predicted as bankrupt must be less than 20% of the total number of test examples.** Without this **prediction sparsity constraint**, a constant prediction of True (Bankrupt) for all examples would result in perfect accuracy according to Equation (1). **Failing this constraint results in a test accuracy score of zero.**

As you can see from the comparison of the standard accuracy score with the Equation (1) accuracy score given in Table 1, the standard accuracy score is useless for this project. Note that the standard accuracy score is the default score metric used by Scikit-learn classification models. You should avoid displaying it at all since it might give you the false impression that your model is highly accurate. Instead, always display the Equation (1) accuracy score.

1.1 Class Competition

This project is a class competition, and teams will be ranked using the following metrics:

$$Rank = 0.3(acc_{train}) + 0.4(acc_{test}) + 0.3\left(\frac{50 - N_{features}}{50}\right)$$

where

- acc_{train} is the train accuracy score based on TT and TF values in the last row of Table 3.
- acc_{test} is the accuracy score of generalization obtained from Section 4.
- $N_{features}$ is the number of features used in the model, which appears in the last row of Table 3.

The rank will be converted to the top percentile in the competition and will be used for a portion of the individual's project grade (see Section 5.3).

2 Data Description

The train and test data sets have 5807 rows and 1012 rows, respectively. The total number of given features is 95. The project aims to train a model that identifies whether a company will file for bankruptcy (the target column is "Bankrupt?") or not. The test data set does not have the target column. Each team will submit the predicted classes for the 1012 test companies to the submission file.

3 Model

Since the data is highly imbalanced, logistic regression could not capture the crucial features for companies with $y = 1$. We need a different approach for this type of problem. The new required method is to use an unsupervised clustering technique to group together companies in similar conditions, and then create a distinct classification model for each cluster subgroup. This is a classic **divide and conquer** approach, where we decompose a given problem into (hopefully simpler) sub-problems, solve each sub-problem, and then combine the results to solve the given problem. For a team project like this one, it has the added benefit of clearly separating team responsibilities from individual responsibilities. Each member of your team will be responsible for creating the model for at least one cluster, and this will determine a large part of their individual grade. The decomposition (clustering) phase and the result combination (generalization) phase are team responsibilities that play a large part in determining your team rank, which also contributes to each individual grade.

It would be a mistake to think that the goal of clustering is to assign each team member an equal amount of work, which would imply that you should strive for each cluster to have about the same size. Rather, the goal of clustering is to create a set of sub-problems which are each easier to solve than the original problem. In other words, you want it to be relatively easy to solve bankruptcy prediction independently on each subgroup. This might mean that the subgroups have very different sizes, but there are still ways to divide up the work fairly between all team members. For example, you have the option of assigning more than one subgroup model to a given team member as long as each member models at least one subgroup. Furthermore, the individual grade of a team member is based on their **best-peforming** subgroup. Thus, if a member is assigned a relatively difficult subgroup, then it does not affect their score as long as they are also assigned an easier subgroup. Since your clustering strategy has a strong effect on both your team and individual scores, it is worth taking the time to experiment with different strategies.

3.1 Training Data Process for Clustering

95 features are considered too many features for clustering analysis, so your train data process should greatly reduce this number, say to no more than 50 features, in order to increase clustering efficiency. Be sure to drop the "Bankrupt?" and "Index" columns, but note that they will be used later. There are unlimited ways and numbers to extract, select, drop, and engineer the remaining features.

This reduced number of features is not necessarily the same as the $N_{features}$ number from Table 3 that is used to determine your team rank as stated in Section 1.1. In fact, $N_{features}$ should be significantly less than the number of features used for cluster analysis since $N_{features}$ averages the number of features used in the subgroup models.

Perform any other train data processing, such as standardization, transforming to Gaussian, etc., that is appropriate for your chosen clustering technique. Combine all preprocessing steps into a Python object such as a function, class, or dictionary and save that object to file using the `joblib` library. This will allow the generalization process, which is described in Section 4 and implemented in its own notebook file, to invoke the same preprocessing steps on test data without needing to repeat the train data process. It simply loads your saved file and applies the object to the test data. **Note:** This requires all members of your team to be using the same version of the `joblib` library. An easy way to do that is to have all members of your team upgrade to the latest version of `joblib` at the start of the project.

3.2 Company Characterization

Choose an unsupervised learning clustering technique. You should plan to experiment with different techniques to see which one provides the best result according to the following requirements.

1. Cluster the training data from 3.1 into k-many subgroups where k can be up to twice the team size but not less than the team size. For example, if a team has three members, the cluster groups can be between 3 and 6. This allows each team member to create the prediction model for at least one subgroup. If you find that a subgroup has no bankrupted companies among the train data, then do not assign it to a team member, but instead assign it a constant prediction model ($\hat{h} = 0$). If this does not leave enough subgroups for each team member to be assigned at least one, then increment k by one, and try again until each team member is assigned at least one subgroup. **Note:** It is not necessarily a bad thing to have one or more subgroups with no bankrupted companies since this helps to achieve the divide-and-conquer goal of making simple sub-problems. On the other hand, having too many such subgroups could force your k value to be large, which tends to cause overfitting. Experimentation is necessary to evaluate the tradeoff.
2. Report the number of companies and the number and proportion of bankrupted companies in each subgroup.
3. Identify unique or helpful characteristics in each subgroup. The characteristics must be stated in terms of the problem domain. Use visualization techniques to present identified characteristics. You should find these characteristics useful when designing the prediction model for the subgroup.
4. Keep the cluster IDs. They will be used in Section 3.3.1.
5. Save the original train data for each cluster to a separate CSV file so your teammates can use it to train their subgroup models.

3.3 Bulding Training Models

It can be broken into two steps. The first step, which is a team responsibility, is to predict the subgroup (the cluster-ID from Section 3.2) that a newly-observed company will likely belong to, and the second step, which is an individual member responsibility, is to classify whether the company will file for bankruptcy in each subgroup. Figure 2 displays a summary of the model.

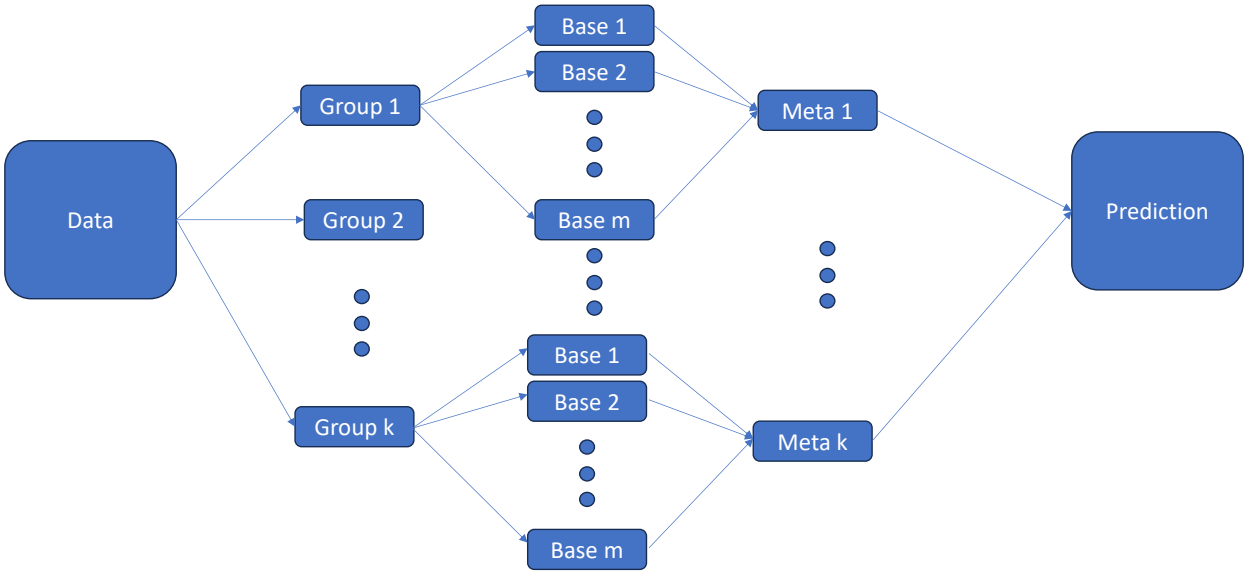

Figure 2: Stacking Method Training Model Diagram. The diagram shows the k-many subgroups, m-many base models in each subgroup, and k-many meta-models where each meta-model is a combined model of m-base models in each subgroup.

3.3.1 Cluster-ID Prediction

As a team effort, build a classification model that predicts the subgroup a test company will likely belong to using any supervised learning algorithm. You may use any classification model available from Scikit-learn, not just the ones we studied in this course. Identify the features that play important roles in this prediction. Save the fitted model to file using `joblib`. It will be used in Section 4.

3.3.2 Subgroup Bankruptcy Prediction

Each individual member must build a stacking model (with cross-validation) that predicts whether a company in their assigned subgroup will file for bankruptcy.

1. There must be three or more base models. Any classification model from Scikit-learn is allowed. The set of base models can vary between subgroups.

2. Each subgroup must use the same features for all base models. However, features can be different between subgroups. In fact, each subgroup may use a distinct train data process in order to select, extract, or engineer a reduced feature set specifically for that subgroup. Report the number of features used for the subgroup in the last column of Table 3.

3. Combine the subgroup-specific preprocessing steps and the fitted stacking model into a single object, and save it to file using `joblib`. It will be used in Section 4.

4 Generalization

Transform the test data set as the training set using the preprocessing steps saved in Section 3.1. Predict each test company's cluster ID using the fitted model saved in Section 3.3.1. Then, predict each test company's bankruptcy status using the preprocessing steps and fitted stacking model saved in Section 3.3.2. Paste the result to the submission file (Table 2). The instructor will evaluate the accuracy score using the predicted class the team submitted.

Index	Bankrupt?
0	1
1	0
⋮	⋮
1011	0

Table 2: Submission File Example

5 Timeline, Submission, Grade Scheme

The submission of the project is due on December 9 by 11:59 PM. Each team must choose one member to do the submission. **Multiple submissions from the same team will not be accepted.**

5.1 Timeline

The ideal timeline of the project is broken down as follows.

1. Train Model: The train data set will be released on Tuesday, 11/11, so the training data preparation and building of a training model can be started.
2. Generalization: **The test data will be released on the first week of Project Weeks (11/18)**. The generalization should be one-time work and should not take a long time.
3. The completed project, including video presentation and files must be submitted by 12/9.

5.2 Submission

Here are the requirements for submission. Each team will submit a single compressed **zip** file. The submission will require the following files inside the zip file. Name the file as *GroupNumber_Project.zip*.

1. Sections 3.1, 3.2, and 3.3.1: A single team notebook file: Clean and summarize the work. Use Markdown for comments and explanations. Name the file as *GroupNumber_TrainingData.ipynb*.
2. Section 3.3.2: A distinct notebook file from each individual team member. Show the work and summarize the work. Name the file as *MemberName_SubgroupNumber.ipynb*. If any member worked on multiple subgroups, submit a distinct file for each subgroup.
3. Section 4: Submit a team notebook file and the submission file as shown in Table 2. Name the files as *GroupNumber_Generalization.ipynb* and *GroupNumber_Generalization.csv*, respectively.
4. Result Summarization Report: Summarize the results as a team in a DOCX file. Name the file as *GroupNumber_Results.docx*.
   - Section 3.2: Report the number of subgroups, the number of $y = 0$ and $y = 1$, and summarize characteristics, distributions, or properties of companies in each subgroup.
   - Section 3.3.2: Construct a table as shown in Table 3. In addition to that, construct a confusion table for base models and the meta-model for each subgroup.

Subgroup ID	Name of Student	Number of Companies	Number of Bankrupt	Stacking model TT	Stacking model TF	$N_{features}$
1	John	1009	131	118	13	34
2	Kim	417	22	20	2	6
3	Constant	997	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
k	Name	xx	xx	xx	xx	xx
team		5807	198	a	$b = 198 - a$	c

Table 3: Result Table Example: Column 1: The subgroup ID from Section 3.2, column 2: the name of the member who worked on the subgroup, or 'Constant' denoting a subgroup with no bankruptcies, column 3: the number of train examples in the subgroup from Section 3.2, column 4: the number of bankrupt train examples in the subgroup from Section 3.2, column 5: the number of bankrupt train examples correctly classified by the stacking model, column 6: the number of bankrupt train examples incorrectly classified by the stacking model, column 7: the number of features the member used in the stacking model. As a team score, the last row gives the sum of each column except the last column, which is the weighted average of $N_{features}$, where the weight is (Number of Companies)/5807.

- Video Presentation: Record a video presentation to describe the workflow, models, and results. You can either include the video file in the zip file or provide a link in the summarization document.

** All reported results must be consistent with notebook files. Make sure to use **random.seed()** or **random_state=** whenever needed so the same results are reproduced each time the notebook runs. Each notebook file must be submitted with required results displayed. Any non-consistent results will be marked 0.

5.3 Grade Scheme

This is how the project will be graded. Although this is a team project, each member will be graded separately, as shown below

$$Score = 0.2(Rank) + 0.4(acc_{Table3}) + 0.4\left(\frac{50 - N_{features,Table3}}{50}\right) \tag{2}$$

where $Rank$ is defined in Section 1.1, acc_{Table3} is the accuracy score of the individual's stacking model in Section 3.3.2*, and $N_{features,Table3}$ is the number of features the individual used in the stacking model in Section 3.3.2. The accuracy scores and number of features reported in Table 3 will be used. The accuracy metrics in Equation 1 will be used for the individual's grade.

It is extremely important that the TT, TF, and $N_{features}$ values are reported correctly in Table 3 so that individual scores are computed fairly for all students. **Failure to report these values correctly is considered equivalent to cheating and will be penalized accordingly.** The values must be reported relative to the **original train data** for each cluster as determined in Section 3.2 rather than, for example, the smaller set of train data that might result after you do a train-test split on the original train data. An easy way to satisfy this requirement is the following. After you're satisfied that your subgroup stacking model has been properly trained, where the training is allowed to include a train-test split or other preprocessing that alters the number of train examples, you then run the original train data for your subgroup through your model, in other words, predict bankruptcy on the original train data for your subgroup, and report the resulting TT and TF numbers in Table 3. As a quick check on the validity of Table 3, the sum of column 5 plus the sum of column 6 must equal 198.

Note*: If a member trained the stacking model for more than one subgroup, the best-performed model in terms of accuracy will be used in the grading metric.
Note: The group performance (*Rank*) will weigh 20% of the project grade, and the remaining 80% of the grade will be evaluated from the individual's performance.
Note: This is a team project. If any member of a team is not responsive and does not return emails, texts, or calls, it is the team's responsibility to contact the instructor. If this member was reported twice or more, the student will be out of the project and receive a zero. Please be responsive to team members.

6 Approved Libraries

For this project, you may use any Python libraries mentioned above, as well as the following.

1. Any library allowed in any homework assignment in this course.
2. Any library from Scikit-learn.
3. The `scikit-learn-contrib` package `imbalanced-learn`. You might find this helpful in the training for a cluster subgroup that remains heavily imbalanced, but beware that it tends to cause overfitting. You should only use it as a last resort. Note that you are not allowed to model a heavily imbalanced subgroup with a constant predictor. You must build a stacking model for it even if it has just one bankrupt company out of more than a thousand. However, you **are** allowed to create a stacking model that overfits, but again, you should only do this as a last resort.
4. Any library from the `statsmodels` module.