ID5059 Knowledge Discovery and Datamining
Assignment 2: Group 28

Introduction

This project aims to predict the outcomes of patients with cirrhosis. There are three possible outcomes for patients: C (alive), CL (alive with liver transplant), and D (patient died); these are measured after N_days. Data was collected on patient characteristics such as age, sex, attributes relevant to liver disease like laboratory test results, physical manifestations (like swelling and liver enlargement), their disease stage, and whether they received the trial drug or a placebo.

To achieve the main objective of this multiclass classification project, several tasks will be completed to understand the data and its structure, visualize relationships between features, deal with imbalances inherent to the data, and develop methods that handle missing information in the dataset. Then three models will be fitted, using both the original data and that which has been treated (after replacing missing values and dealing with class imbalances), to determine which strategies have the best results.

The commercial application of this type of classification task could include providing a screening service to patients with liver disease to predict their outcomes over time. For example, a patient who is being followed-up at a specialist clinic could input their current stage of disease, along with their lab results, to better understand their prognosis in the context of their disease. This could help clinicians tailor their treatment plans and offer better care to their patients.

Methods

Data exploration: The data has been previously split into training and test sets, consisting of 7905 rows and 20 columns for training and 5,271 rows and 19 columns for the test set, with the target column removed from the test set. The data is investigated using histograms to visualize the distribution of the data, a correlation matrix was plotted to visualize possible relationships between features and the target variable. Additionally, we used crosstabs and box plots to further investigate the association between categorical features, other features and the response variable.

Imputation methods: The data originally does not have any missing values, however, to explore methods of dealing with missing data, 20% of data was randomly deleted the following features: N_*days*, *Edema*, *Bilirubin* and *Stage*. This set of features was chosen as they contain a mixture of both numerical and categorical features. Next, three methods of imputation were applied. The first approach was a more basic method, which imputes values from a single dimension (i.e. the feature column). Using sklearn's simple imputer, the missing data was imputed using the "most common" strategy. The second approach, MICE (multiple imputation by chained equations) use the entire range of features to estimate the missing values. Each feature is treated as the response term and the rest are inputs, then a regressor is used to predict the missing value of the response feature. This is done iteratively, and each feature takes a turn being the response. Then a final set of imputed values is returned after the iterations are completed [1]. The last approach, KNN imputation, uses the k-nearest neighbors approach[2]. This uses Euclidean distance to find the nearest neighbors that have values for the variable, the missing value is input with the mean value from its nearest neighbors. The number of neighbors was set at 5 and the weights uniform.

Handling unbalanced data: Upon visualizing the target variable, each label accounted for: C: 62.8%, D: 33.7% and CL: 3.5%, a considerable imbalance between the groups. To resolve this issue, the main approach involved oversampling the data to give more weight to the minority classes, using two methods. The reason for choosing oversampling methods is because undersampling - the process of removing data from the majority class to balance classes - may potentially cause loss of valuable information that can help machine learning models identify

important patterns and trends for the majority class. The first method used was ADASYN (adaptive synthetic sampling). It adaptively generates minority data samples according to the distribution of the data. It measures the distribution of weights for different minority class examples depending on its difficulty of learning[3]. Essentially ADAYSN uses the k-nearest neighbours method to find neighborhoods for each of the minority examples. The dominance of the majority class is then calculated, and this determines the learning difficulty. It then focuses on creating more data for these harder to learn examples[4]. The second strategy, SMOTE is an oversampling method that deals with imbalanced classes by synthetically increasing samples from minority classes. The SMOTE method achieves this by finding its k-nearest neighbors in the feature space and creates a data point in between the two points.

Data cleaning/feature engineering: In the Data Preparation stage, we undertake some crucial steps to ensure the data is in a suitable format for analysis and modelling. The key processes involved in this stage are the following:

Feature Selection. In feature selection we re-analyze our correlations. Since there are no missing values, so correlations are our primary metric. In data exploration it is clear that there are no concerningly high inter-feature correlations and as such we are not concerned that there will be redundant information present in our model. This combined with the low quantity of total data suggests that keeping all features (barring *id* – a unique identifier) will provide our model with the most information on which to be trained.

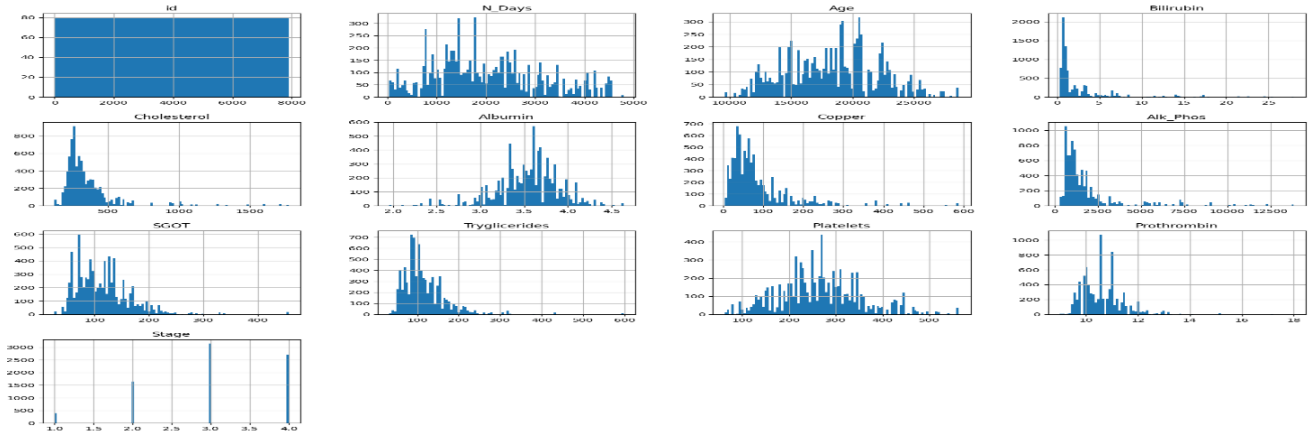


Figure 1: *Barplots – Addressing heavy tails in the data features.*

Feature Engineering. In this section, we address various aspects of feature engineering to enhance the quality of our dataset. The first step taken is fixing heavy tails which involves visualizing the distribution of features using histograms.

This helps identify features with heavy tails or a significant number of outliers. Then to reduce the impact of these outliers, we take the logarithm of the feature values. Additionally, some Prothrombin values have such extreme outliers we opt to remove these values altogether (this results in a loss of 15 rows – only 1.8% of data). Furthermore, we discretise continuous variables such as Age and Number of Days into categorical bins. Our intention here is to help in capturing non-linear relationships and reducing noise in the data. Age is now categorised into age groups (20-35, 35-45, 45-60, 60-80), while Number of Days is categorised into Year intervals (0-2, 2-5, 5-10, 10-15). Later we encode categorical variables to convert them into a format suitable for machine learning algorithms. This step involves creating dummy variables for each category within categorical features. We plot a correlation matrix heatmap to examine the relationships between the updated features in the dataset. In addition, by plotting barplots we explore the relationships between the newly created features and the target variable 'Status'. This helps us ensure that our binning strategy preserves meaningful relationships within the data.

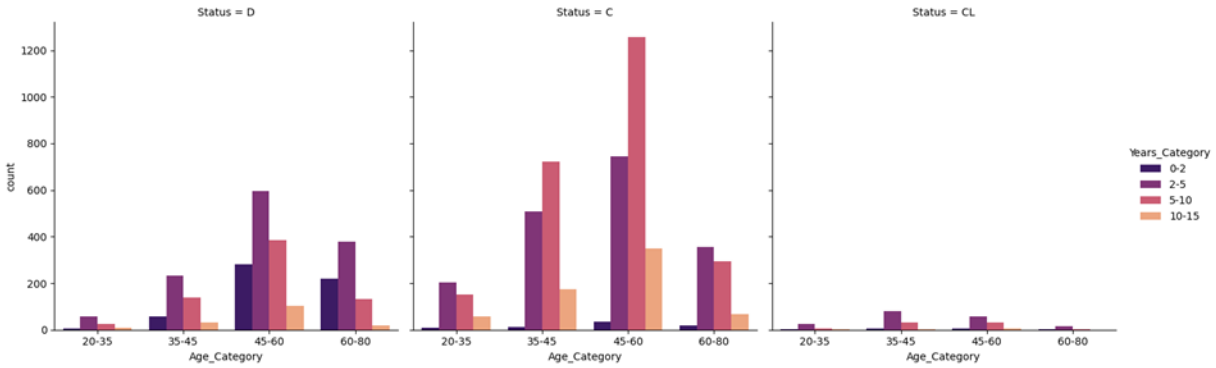


Figure 2: *Barplots – Feature Engineering. Binning strategy maintains relationships between both Age and Year in trial with Status.*

Pre-processing: Here a procedure for preparing our dataset is established. We define functions: df_strings_to_numbers(), split_and_drop_entire_DF(), log_adjust_outliers(), adjust_prothrombin(), Age_Categories(df), Year_Categories(df) with these we convert all string category values to numerical, split dataset into input/response, take log of values in heavy-tailed features, remove outlier rows based on Prothrombin value, engineer age categories and then engineer year categories. *The function Df_strings_to_numbers()* represents the first preprocessing step, as this needs to be done prior to resampling. The other functions are used after resampling, and for convenience are combined via the function processing_step_2(). Which outputs input data and our column of response variables. The final step in pre-processing is to apply our two pipelines full_input_preprocessor, full_response_preprocessor to input and data respectively. These pipelines simply apply the necessary encoding and scaling to our dataset. We employ OneHotEncoder()– so our models can receive unordered categorical data as an input, OrdinalEncoder() – so we can input ordered categorical data, StandardScaler() - scales numerical values. And we employ LabelEncoder() to ready our response data for the model. Here we select the features 'Stage', 'Years_Category', 'Age_Category' as ordinal and 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Drug' as nominal. While there is some ambiguity as to whether features like Edema are nominal or ordinal, it became clear through preliminary model training that this has very little effect on the performance of a model. Additionally, during preliminary model training it becomes clear that our models perform better when trained on the numerical data 'N_days' and 'Age' than on 'Years_Category' and 'Age_Category'. As such the engineering of these features is dropped from the pre-processing procedure.

Classification models: The following models were trained: Random Forest, Stochastic Gradient Descent (SGD), Support Vector Machine (SVM) and Neural Net.

The main method for evaluating these models will be log loss. This is a function of the sample prediction probability value. This means that the greater the probability of the sample being predicted correctly, the smaller the loss value[5]. This log loss metric follows a negative logarithmic curve, whereby the log loss slowly approaches zero as the probability of the sample being correctly classified approaches 100%. Other metrics will also be used to support analysis

Results

Data exploration: The most remarkable findings from data exploration were that the number of females patients (92.8%) in the dataset greatly outnumbered the number of males (7.2%). The distribution of some features was skewed with long tails: cholesterol, copper, alkaline phosphatase, SGOT, triglycerides, prothrombin. For stage, 40% corresponded to stage 3, 34% to stage 4, 21% stage 2, and 5% to stage 1.

Imputation methods: The three imputation methods, simple imputation (using most common variable for each feature), MICE, and KNN imputation were used to evaluate the performance of the Random Forest model. The

log loss for each of these methods is summarized in **Table 1**. Overall, there was not a significant difference between the imputed data and the original dataset. The simple imputation method slightly improved the log loss, which could be attributed to the fact that this inputs the missing data with the most common value for each feature, which makes the model more likely to predict over something that is more likely to be expected, in other words it would more accurately guess over data that is more predictable. The most frequently predicted classes were C and D, and the random forest model with imputed data performed poorly in predicting CL. Although the simple imputation method had the lowest log loss, either MICE or KNN imputation methods are more thorough methods of inputting missing values as they predict missing information based on the existing information in the entire dataset.

	Original Data	Hyperparameter Tuning	SMOTE	ADASYN	Simple Imputation	MICE	KNN Imputation
Random Forest	0.501	0.476	0.556	0.573	0.493	0.512	0.513
Neural Network	0.493		0.7599	07517			
SGD	0.789						
SVM	0.815						
Kaggle Submission	0.50077 (neural network with original dataset)						

Table 1*: Summary of log losses for trained models, imputation, and oversampling methods.*

Classification models: We train the three most common classifiers at first, Random Forest, Stochastic Gradient Decent, and Support Vector Machine (a polynomial kernel is adopted to incorporate the potential nonlinear relationship) with default parameters on data resampled by SMOTE. Using the One-Versus-Rest strategy and 5-fold cross-validation, validation accuracy was 81.4%, 68.3%, and 73.7% respectively. Note that no oversampled data is included in the validation set, only in the training set. Apart from traditional machine learning models, we also train a Neural Network model with four layers. The model aims to minimise the log loss function, which ended up with a validation accuracy of 76.2%. As a heuristic approach in the common multiclass classification tasks, Random Forest seems more promising than other chosen models.

Model Evaluation: Then, we investigate if the random forest overfits through visualising the precision-recall curves with predictions on the train and validation set. We found that the baseline model overfits as indicated by a poor prediction on the validation set compared with the training set, especially for imbalanced class CL. We then tuned the parameters of this model to see which combination leads to the best log loss score. The set of hyperparameters include, the minimum samples at splitting, the number of decision trees, the maximum tree depth and the minimum sample required to be a leaf. A justification of these parameters can be found in the supplied notebook. Overall, the tuned model gives a cross-validated prediction accuracy of 79.9%, a slightly lower but sensible result since overfitting is not fully addressed.

Finally, we apply ensemble learning as well by using a Voting Classifier to aggregate the tuned random forest with other mediocre models. A hard vote is adopted to choose the majority vote of the class since other models are not calibrated. However, a drop in accuracy to 78.2% compared to a single Random Forest classifier suggests there might be a huge discrepancy in predictions between models, leading to an overall deterioration of predictions. Therefore, we conclude Random Forest is a strong classifier in our case.

A feature importance plot from the random forest model is given in **Fig.3**. Bilirubin and Age are the first two most important features. To assess the model performance graphically, we use a confusion matrix and decision boundary plot. The decision boundary is plotted by first generating data points in the plane of the two most important features, i.e., Age and Bilirubin, and imputing all data in other dimensions with corresponding modes in the original data. Then we predict those data points with the trained Random Forest. In some sense, it phenomenologically resembles the Principal Component Analysis where data points are projected in lower dimensions along directions that maximise the total variance. PCA is not directly used since the first two principal components only capture 37.0% variance which is hardly representative of the original data. From the decision boundary plot, we can see the data points are barely linearly or nonlinearly separable, and this is consistent with the relatively inferior performance

of the Stochastic Gradient Descent Classifier based on the multinominal logistical regression and Support Vector Classifier which struggles to find the street that distinguishes the data points. In Fig.3, the diagonal grid colour in the confusion matrix and the distribution of regions overlayed by data points in decision boundary plots point out consistently that the prediction on C is more accurate than the other two statuses, as expected from the original data exploration.

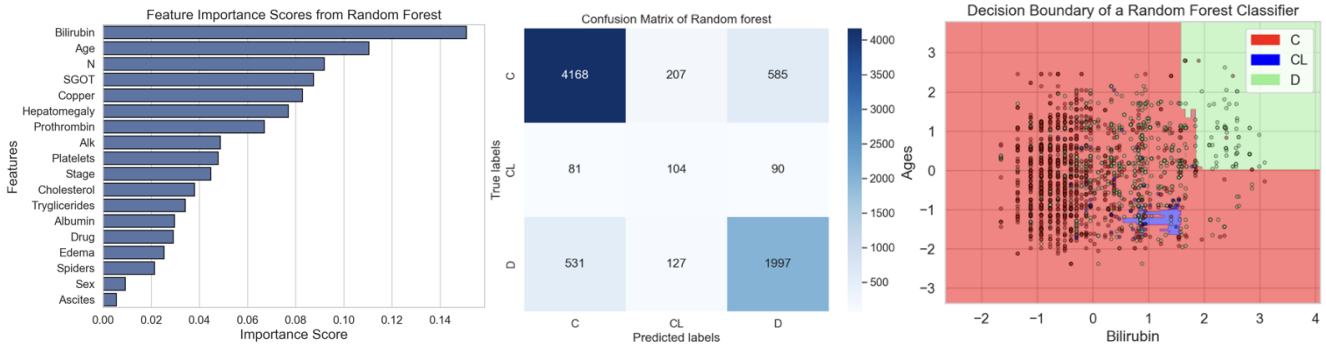


Figure 3 Left: *Feature importance plot from random forest.* ***Middle****: Confusion matrix on predictions.* ***Right****: Scatterplots of original data points projected in the plane with Age and Bilirubin as bases, overlayed by decision boundary from hyper-tuned random forest model.*

Handling unbalanced data: One issue arising from imbalanced class data is biased model training. In general, when a model is trained on imbalanced data, it tends to bias predicting the majority class, often meaning poor predictive capabilities on the minority classes. In real-world scenarios, this can have significant consequences, especially in applications where all classes are equally important. As mentioned previously, around 63% of the patients diagnosed with cirrhosis are still alive (C), 34% died (D), and 3% survived with a liver transplant (CL). For most models, we can see that they often predict C for all classes because of this imbalanced dataset. To solve this issue, we applied two oversampling methods, SMOTE and ADAYSN, to generate synthetic data for the minority class to even the distribution of classes. It is important to note that when generating synthetic data, this is only applied to the training set. In the context of 5-fold cross-validation, these methods should not be applied to the validation fold for each fold. This ensures that we are not evaluating the performance of the model on synthetic data as this generally provides an overestimation of how well the model is performing due to the synthetic data being similar to the original data.

As the initial investigation used SMOTE, we first trained our best model, the tuned Random Forest, with the original dataset. This obtained a log loss log loss score of 0.48, which is the lowest score obtained for any model. Both SMOTE and ADAYSN obtained a log loss score 0.55 and 0.57 respectively. The main difference however was that the model that used the original dataset never predicted the CL class for all samples. With the models that utilized SMOTE and ADAYSN, the CL class was now being correctly predicted 37.5% and 34.9% of the time respectively. This comes at a cost as the accuracy for the C and D class decreased from 92.0% and 71.0% to 84.1% and 76.2% with SMOTE and 82.2% and 77.6% for ADAYSN.

These oversampling methods were also trialed on the neural network to see if the same results would be obtained. Overall, the log loss metric also scored worse, but the trend for improving predictions for the CL class improved. Training the neural network on the original dataset showed that it was only able to predict CL 1.8% of the time, incorrectly predicting it as C and D 52.7% and 45.5% of the time respectively. Applying SMOTE and ADASYN, we can see that CL is accurately predicted 29.1% and 16.4% of the time respectively. This also comes at a cost as accuracy in predicting C and D decreased from 89.7% and 73.1% to 77.5% and 74.5% of the time respectively for SMOTE and to 74.9% and 78.1% with ADAYSN. This highlights that depending on whether it is important in accurately predicting CL, then applying these oversampling methods, preferably SMOTE, will improve the model's performance by mitigating these biases. However, if this distinction is not important and the goal is to

minimize the log loss metric, then the suggestion is to use the original dataset or to preprocess the data to group all CL data with C data and turn this problem into a binary classification problem. Another recommendation is that both these oversampling techniques obtained a better accuracy for CL if the data was transformed before preprocessing. However, care must be taken to ensure the data used for validating a model's performance does not contain synthetic data to mitigate overestimation.

Discussion

Some of the main limitations to this experiment would include the limited size of the dataset, along with its inherent imbalances, particularly in the outcomes. Although methods were explored to try to compensate for these limitations, the ideal way to overcome them is to have larger and more balanced data.

Returning to the business objective of this task, it is important to balance what is most important in a clinical perspective versus having a best performing model. One could argue that any of the three outcomes is equally valuable from a patient perspective, but perhaps clinicians are more interested in seeing how many patients with liver transplants survive, for example. Considering how costly and hard to come by a liver transplant might be, would it be useful to also know which of the patients who died had also received a liver transplant? Currently, all versions of the models are best at predicting the majority class C, but as mentioned before, it is important to consider which feature, if any, is most important for the stakeholders (client and patients).

If the possibility arose to expand the data and collect new samples, it would be important to include features which the models considered more significant to predicting the outcomes, such as: Bilirubin, Age, N_Days, Copper, and SGOT. This is also true for implementing the model as a product, if patients are going to be screened for their outcome, it is important for these features to be input for a better prediction. Additionally, if the models were to be optimized with new data from new patients, it might be useful to collect a new outcome category, DL, patients who died after a liver transplant. Further investigation can also explore the effectiveness of combining the explored imputation methods and oversampling techniques, in the search of finding an ideal balance of class predictions.

Conclusion

In summary, the cirrhosis patient data was initially explored, some of the features were adjusted to address their distributions, data was randomly removed to explore methods of imputing data. The data was then used to train four models, which were evaluated using cross validation. Log loss was calculated on each of the models, and Random Forest was chosen as the model for which its hyperparameters would be tuned. Class imbalances in the target variable were addressed using two oversampling methods, SMOTE and ADASYN, both of which increased log loss, but were more accurate for predicting the minority class CL. Overall, the best log loss was the Random Forest model with the original data (0.476), and across all models, the outcome most likely to be predicted was C (patient lived).

References

[1] *6.4. Imputation of missing values*. (n.d.). Scikit-Learn. Retrieved March 30, 2024, from https://scikit-learn/stable/modules/impute.html

[2] *Sklearn.impute.KNNImputer*. (n.d.). Scikit-Learn. Retrieved March 30, 2024, from https://scikit-learn/stable/modules/generated/sklearn.impute.KNNImputer.html

[3] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969

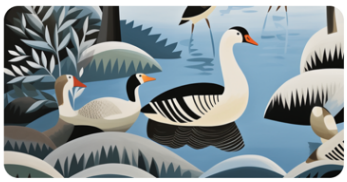
[4] Nian, R. (2019, December 13). An Introduction to ADASYN (with code!). *Medium*. https://medium.com/@ruinian/an-introduction-to-adasyn-with-code-1383a5ece7aa

[5] Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2022). A Comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science*, *9*(2), 187–212. https://doi.org/10.1007/s40745-020-00253-5

Appendix: Kaggle Submission

Multi-Class Prediction of Cirrhosis Outcomes

Playground Series - Season 3, Episode 26



Overview Data Code Models Discussion Leaderboard Rules Team Submissions

Leaderboard

Raw Data Refresh

YOUR RECENT SUBMISSION

ID5059-Group28.csv
Submitted by dyh799 · Submitted 2 minutes ago

Score: 0.50077
Public score: 0.51691

Jump to your leaderboard position