Analysis of Medical Data Using Machine Learning Methods

Final Project

NOY FLAT, MAYA VISHNEVSKY, TOM TIMIANKER

AUGUST 2021

Introduction

The goal of this project is to use machine learning models to help physicians detect and treat hospital-acquired bacterial infections. Using real patients' data acquired from MIMIC-III and eICU databases, we had to develop two models:

1. A classification model to determine if the infection is bacterial or not (we reference this as 'task A')
2. A classification model to determine if the administered antibiotics were an appropriate treatment to the infection or not ('task B')

In the following project description, we will demonstrate the methods we tried and the ones we chose to try and predict the abovementioned classifications.

Cohort Description

The datasets we used hold clinical data of patients admitted to the Beth Israel Deaconess Medical Center in Boston, Massachusetts (MIMIC-III DB)[1] and other critical care units throughout the continental United States (eICU DB)[2].

- The cohort for task A from MIMIC-III DB holds 3075 patients, of whom 161 had bacterial infection.
- The cohort for task A from eICU DB holds 126 patients, of whom 8 had bacterial infection.
- The cohort for task B (consists of data that originated from MIMIC-III DB only) holds 105 patients, of whom 22 were given an appropriate treatment.

The data we fetched from the DBs is:

- Lab tests data.
- Microbiology labs data - We fetched the microbiology culture sites for task A, and the culture sites and organism name for task B. This data could be problematic, because if the patient was infected and we used the culture that contains the bacteria this data will be illegal. That is why we used only data obtained 3 days before the infection diagnosis for task A, 2 day for task B. In the train set, this data was available for about 10% of the patients.
- Prescriptions given to the patient.
- Patient data - gender, age, weight, ethnicity.

Further explanation on the features we use in the model can be found in the `Methods - Data Preprocessing" section.

One significant characteristic of the cohort data is that the "negative" and "positive" classes are imbalanced. In each task, the number of "positive" samples is much lower than the number of "negative" samples. We discuss how we handled the unbalanced data in the "Models" section.

The following tables of characteristics compare the "negative" and the "positive" groups according to a selected number of out features using a statistical hypothesis test that tests whether the distribution of the feature is different in each of the groups.

Each of the continuous features were tested using the two-sample Kolmogorov–Smirnov test and we present the p-value for the null hypothesis that both of the samples are drawn from the same distribution.

For each of the discrete features we performed different tests according to the tasks to better suit it for the sample size.

For the binary features of task A in the MIMIC data which has a relatively large sample, we used the Chi-squared test and we present the p-value of the null hypothesis that there is not any dependency between the classes in the population.

For the binary features of task A in the eICU data and task B which has a relatively small sample, we used the Fisher's exact test and we present the p-value of the null hypothesis that is not any dependency between the classes in the population.

I should be mentioned that we examined the task A MIMIC data and the eICU data for task A as two separate data set.

Features from MIMIC in task A:

Feature	p-value	Hypothesis test
var_Glucose	0.001442	Two-sample Kolmogorov–Smirnov
3_days_max_Glucose	0.002637	Two-sample Kolmogorov–Smirnov
3_days_min_Potassium	0.00938	Two-sample Kolmogorov–Smirnov
drug_LORAZEPAM	0.011865	Chi-squared test
3_days_max_Hematocrit	0.040477	Two-sample Kolmogorov–Smirnov
drug_GLUCAGON	0.063417	Chi-squared test
age	0.179079	Two-sample Kolmogorov–Smirnov
drug_DOCUSATE	0.443854	Chi-squared test
3_days_avg_Hematocrit	0.456047	Two-sample Kolmogorov–Smirnov
drug_MIDAZOLAM	0.473935	Chi-squared test
ethnicity_Hispanic	0.611086	Chi-squared test
3_days_min_Chloride	0.955226	Two-sample Kolmogorov–Smirnov

Features from eICU in task A:

Feature	p-value	Hypothesis test
var_Glucose	0.032143	Two-sample Kolmogorov–Smirnov
drug_GLUCAGON	0.071429	Fisher's exact test
3_days_max_Hematocrit	0.186701	Two-sample Kolmogorov–Smirnov
3_days_avg_Hematocrit	0.19323	Two-sample Kolmogorov–Smirnov
3_days_min_Potassium	0.218366	Two-sample Kolmogorov–Smirnov
drug_LORAZEPAM	0.285714	Fisher's exact test
drug_DOCUSATE	0.357143	Fisher's exact test
drug_MIDAZOLAM	0.428571	Fisher's exact test
3_days_max_Glucose	0.456349	Two-sample Kolmogorov–Smirnov
3_days_min_Chloride	0.79002	Two-sample Kolmogorov–Smirnov
age	0.815394	Two-sample Kolmogorov–Smirnov
ethnicity_Hispanic	1	Fisher's exact test

Features from MIMIC in task B:

Feature	p-value	Hypothesis test
4_days_avg_INR(PT)	0.002393	Two-sample Kolmogorov–Smirnov
var_INR(PT)	0.002997	Two-sample Kolmogorov–Smirnov
var_Lymphocytes	0.008598	Two-sample Kolmogorov–Smirnov
var_PT	0.008912	Two-sample Kolmogorov–Smirnov
4_days_min_Calcium, Total	0.028561	Two-sample Kolmogorov–Smirnov
var_Hematocrit	0.029703	Two-sample Kolmogorov–Smirnov
var_MCH	0.035424	Two-sample Kolmogorov–Smirnov
4_days_avg_Platelet Count	0.039625	Two-sample Kolmogorov–Smirnov
drug_WARFARIN	0.628869	Fisher's exact test
drug_PROPOXYPHENE	1	Fisher's exact test

We can see that there is a connection between the low p-value we received in the above-mentioned tests to the features we chose in the feature selection part of our project.

Methods

Data Preprocessing

Our first step was grouping all the data Lab tests such that all of the patient's information will be presented in only one row. For each patient we used the maximum, minimum and average value of each lab test in the K days before the target time. We performed a research in order to decide on the best value of K by checking the correlation between the target and the K-related features on numerous values of K. We found that K = 3 for task A and K = 4 for task B produced the best results. Additionally, we wanted to get an insight from the entire set of lab tests that were made for each patient. We explored what can be the best way to achieve that and we decided on taking the variance of the lab tests for each patient, that will give information on whether the lab tests were significantly changed throughout the patient's hospital admission.

For microbiology data, prescriptions data and ethnicity we used one hot encoding. Those features were categorical and using one hot encoding they turned to multiple binary features.

Then we merge all this data to one table that contains a single row for each patient.

This process contains one difference between task A and task B: In task A, since there were two data sets to work with, we could only use features found in both databases. That is why some lab tests, microbiology culture sites and prescription are used only in task B.

For the data to be similar between the two data sets, we also modified the way the ethnicities were divided in MIMIC, because in the eICU DB the ethnicities were more generic.

The next step in the data preprocessing is imputation. Imputation is a very important step to have all the data needed for the model. We use KNN imputation, because we wanted the imputation to be based on the other data we have on the patient and this method is a well-known and showed good results in published studies.[3]

Finally, we scale the data to generally fit the normal distribution. The scaling happens independently on each feature. The standard score of a sample is calculated by removing the mean of the samples and dividing by their deviation.

Models

Throughout our working process, we checked numerous classifying models for both tasks, such as Gradient Boosting, AdaBoost, Logistic Regression, SVM, Random Forest and AdaCost. The AdaCost Classifier[4,5,6] is a variation of the AdaBoost classifier that differs from AdaBoost in the samples each estimator is trained on. The basic idea is giving different costs to different misclassification errors. At each round, the weight updating rule increases the weights of costly wrong classifications (in our case – false negative) more aggressively but decreases the weights of costly correct classifications (in our case – true positive) more conservatively. This corresponds to the fact we prefer a patient who's not suffering from bacterial infection will be treated with antibiotics rather than the opposite.

After we have evaluated their performances, we decided to focus on 2 decision tree based ensembled classifiers: Random Forest and AdaCost.

Our working process included dividing our decision making into three parts:

1. <u>Class balancing decision</u>: The classes are extremely unbalanced, which led us to search for possible solutions for class balancing.
   The options we explored were:
   a. **NCL** (Neighborhood Cleaning Rule)[7]: Compute k nearest neighbors for each sample. If the sample belongs to the majority class, and it is misclassified by its k nearest neighbors, it is removed. If it belongs to the minority class, and it is misclassified by its k nearest neighbors then the majority class examples among those neighbors are removed. We checked this method for k = 3, 5.
   b. **Naïve up sampling**: randomly sample entries from the minority group until we reach an equal number of minority and majority group's entries.
   c. **SMOTE** (Synthetic Minority Oversampling TEchnique)[8]: The minority class is over sampled by taking each minority class sample and introducing synthetic examples along the line segments joining its k minority class nearest neighbors. This approach effectively forces the decision region of the minority class to become more general. We checked this method for k = 1, 2, 3, and for sampling strategy = 0.1, 0.2, …, 1. The sampling strategy indicates the desired ratio between the classes or in other words: how many synthetic samples should be produced.
   d. **Down sampling**: we randomly down sample the majority class. The percentage of the samples we chose to delete was tuned in the process of balancing the data (ratio of negative samples that are kept = 0.6, 0.7, 0.8, 0.9).
   e. **Unify sampling options**:
      - **NCL & SMOTE** - NCL should be performed before SMOTE, because SMOTE changes the k nearest neighbors of the samples which might add noise to the down sampling process in NCL.
      - **down sampling & SMOTE** - Random down sampling should be performed before SMOTE because SMOTE gets an argument which indicates what is the desired label ratio and down sampling before SMOTE might bring us close to this ratio without introducing synthetic samples.

2. Feature selection decision: Deciding which one of the features is the most significant and can help us determine the predicted target.

   For each model we used two feature selection methods and chose the features that led to higher AUPRC. The first method was **recursive feature elimination**: we eliminated one tenth of the features each iteration until we got to the number of features that maximizes the AUPRC. We used a built-in feature selection function that is based on the feature importance attribute of each estimator.

   The second method we used is also based on the feature importance attribute of each estimator. We used **permutation importance** rather than directly using the feature importance because impurity-based feature importance can inflate the importance of numerical features and can favor features that lead to over-fitting. The permutation importance is calculated by shuffling numerous times each feature's samples and checking the effect it has on the AUPRC. Before evaluating the permutation importance, we deleted correlated features. This is a crucial step because given two correlated features the feature that is addressed second will not add new information, which will lead us to under-evaluate the importance of both features (because we're evaluating the importance repeatedly).

   To detect correlated features, we followed these steps:

   a. Calculate the absolute value of Spearman's rank correlation coefficient to assess monotonic relationships. When using k-folds validation, if in certain fold all the values are equal, i.e. the variation is zero, we deleted the feature.
   b. Perform hierarchical clustering on these values (i.e., use the correlation values as distances in the distance matrix).
   c. Inspect the hierarchical clustering dendrogram and manually choose a threshold which will be used to determine whether the 'distance' between two features is too small and one of them should be deleted.
      At first we chose threshold as the maximum value that applying it leads to AUPRC = max_auprc-0.15*std(max_auprc) where max_auprc is the AUPRC achieved so far (the output of choosing the resampling method). We chose it this way because deleting correlated features should not change the result – all (or a least most of) the information that could be lost is found in the one feature for each cluster we kept. Further information regarding choosing the threshold can be found under 'Results'.

3. Parameter tuning decision: Using the adequate parameters for each of the models can significantly change and improve our results. We used grid-search over a parameter grid with Cross-Validation. For both tasks we've decided to tune the parameters on 80% of the MIMIC samples and use the other 20% to check we didn't cause over fitting.

   In addition to trying different parameters values for each model we added a new parameter to the Random Forest classifier: **LHO-LOO parameter**[9]. LHO-LOO stands for leave half (of majority class observations) out leave one (minority class observation) out. For each decision tree in the forest, we randomly deleted samples according to this rule. After deleting the samples, we merged the rest of the majority and minority samples to generate one subsampling. The original reason to slightly under sample the minority group is to achieve stable feature selection; the selected features may change significantly due to the influence of a single observation. In our case this factor is chosen after the feature selection, but we believe that introducing slightly different minority samples to each tree might improve the results. Instead of deleting half of the samples as described in the referenced article, we tuned the fraction of chosen negative samples.

After tuning all parameters, we tried a new approach: rather than tuning this parameter and treating it as all other parameters we used this parameter to 'soften' the potentially harmful effect down sampling might have. We calculated the correct sampling strategy for SMOTE parameter that leads to same amount of synthetic samples as in the original resampling, and calculated the right LHO-LOO parameter that leads to same negative/positive ratio. This approach is favorable for task B, where we saw that down sampling leads to better results but also might delete valuable samples.

Another interesting parameter is the **ccp_alpha** parameter for the Random Forest classifier: This parameter helps handle the over fitting by pruning each tree using a version of the **C**ost **C**omplexity **P**runing algorithm, in which the "subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen"[10]. This method is important mainly in task A where we saw a severe over fit.

Evaluation Process

As mentioned in the previous section, we evaluated each decision in each of the optional models using repeated K-fold validation using the **average precision score** which *"summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:* $\sum_n (recall_n - recall_{n-1}) \cdot precision_n$*"*[11]. We chose this metric because the ROC curve tends to provide an overly optimistic result for class-imbalanced data, especially when the class-skew is severe. The AUPRC should be preferred because, over different prevalence values, it gives coherent results with the PPV (TP/TP+FP) and the NPV (TN/TN+FN)[12].

Each comparison between classifiers, methods, parameters and thresholds was based on repeated stratified 5-cross validation. At each iteration the samples were shuffled to create different folds. The goal of this evaluation method was to minimize the variation of the results. Unfortunately, even after repeating the validation process 5, 10 or even 50 times the variation was still quite large in comparison to the AUPRC: for model A the variation was ~0.0005 and for model B it was ~0.025. This topic will be further discussed at the result section. In addition to AUPRC and AUPRC variation we also calculated the over fitting that the examined model suffers from. The over fitting is the ratio between the AUPRC when the classifier predicts the training set's labels and the AUPRC when the classifier predicts the validation set's labels.

For task A we used only the MIMIC originated samples to determine the resampling method, the selected features, and the classifier's parameters. We used the eICU DB as an external test set to help us evaluate our performance. For task B we decided to use all the available samples due to the small number of samples we had.

Results

Chosen methods

Class balancing decision: For both classifiers and both tasks the chosen method was down sampling with SMOTE. The parameters chosen are shown in the table below. It should be noted the differences between some of the methods and the parameters values were not significant, and that the variance of each model led us to choose a method that is not necessarily the best.

	down	strategy	K neighbors	AUPRC	variance	over fit
Task A RF	0.6	0.5	3	0.076	0.0007	14.25
Task B RF	0.7	0.8	1	0.441	0.0267	2.69
Task A AC	0.8	0.4	2	0.086	0.0008	2.51
Task B AC	0.7	0.8	1	0.427	0.0327	2.56

RF is Random Forest, AC is AdaCost, down is the percentage of the negative class chosen in the down sampling process, strategy is the ratio between the classes (the down sampling is preformed first).

We can find an intuitive justification for the k neighbors parameter: There are not a lot of samples for task B, if the samples distribute similarly to task A that leads us to believe they are more sparse. Therefore, the distance between the positive examples might be bigger and synthesize samples between 'far' samples (the second nearest neighbor is not as close as in task A) give worse AUPRC.

Feature selection decision:

When applied on AdaCost classifier (task A) the permutation importances mean of only ~40/193 were not equal to zero. Although it seemed over 100 features aren't valuable for the estimator's learning, we saw that adding those features increases the AUPRC. A possible explanation for this matter is not choosing the threshold correctly, but it is more likely the fluctuations are caused by the random nature of the estimators (we expect that correlated features lead to lower but not zero importance).
After a few attempts to choose the threshold that does not decrease the AUPRC we decided to choose it in a trial-and-error method. To reduce the randomness and over fit of the feature selection we ran 5 iterations of 5-fold cross validation and checked how many times each feature had a permutation importance greater than 0. We chose another threshold that determines how many folds out of the 25 are required to choose a certain feature. We used the same method for the recursive feature elimination. We indeed saw that extracting features in this threshold-out-of-25-folds manner leads to higher AUPRC and lower over fit.
For Random Forest classifier (task A) no matter how high we set the threshold, most of the features had a non-zero permutation importance. Some of the features had negative permutation importance which means that substituting the feature with noise is better than the original feature. The other features had a very small positive value. Our conclusion was that this method doesn't suit this model. We didn't reach improvement with recursive elimination either, so we decided to check in the next step 2 options: recursively eliminated features and all features.
For task B we saw that deleting correlated features decreases the AUPRC to ~0.3, even if the threshold is set to zero. Deleting features with zero variation in the training set in a certain fold caused this effect. We hope that checking 25 folds and taking each feature that had permutation importance > 0 in at least PI-folds-threshold of them kept us from deleting valuable features.

	Correlation threshold	PI folds threshold	RE folds threshold	# features	AUPRC	variance	over fit
Task A RF	0.5	<14	<24	127(re)	0.0745(re)	0.0004(re)	14.35(re)

Task B RF	0	<15	<23	44(re)
				18(**pi**)
					0.542(re)
					0.571(pi)
						0.0425(re)
						0.0311(pi)
							2.18(re)
							1.93(pi)

Task A AC	0.5	<7	<9	48(pi)	0.0997(pi)	0.0006(pi)	1.96(pi)

Task B AC	0	<2	<11	52(re)
				38(pi)
					0.524(re)
					0.522(pi)
						0.0346(re)
						0.0351(pi)
							2.04(re)
							2.1(pi)

RF is Random Forest, AC is AdaCost. PI stands for permutation importance and RE for recursive elimination. The brackets indic ate the selected method.

Correlation threshold determines whether the correlation between two features is high enough to delete one of them.

Folds threshold (pi)/(re) is the number of folds (out of 25) the feature should be selected in to be selected to train the sub mitte d classifier.

Parameter tuning decision

We tuned both classifiers for both tasks and chose for each task the tuned classifier that gave better results: For task A we chose AdaCost classifier, for task B we chose Random Forest classifier. The AUPRCs on the validation sets were A – 0.09, B – 0.76. For A the results were worse than on untuned classifier, but we should keep in mind that the results are highly affected by the folds we chose randomly and by the random nature of the resampling and the classifiers.

Results on given folds

In this section we present the results on the folds we were given in the beginning of the semester. Those results are very sensitive to the inherent randomness.

Task A – MIMIC cross validation:

AUPRC – mean = 0.095, std = 0.03

AUROC – mean = 0.614, std = 0.06

Task A – MIMIC train, eICU test:

AUPRC – mean = 0.29

AUROC – mean = 0.69

Task B - MIMIC cross validation:

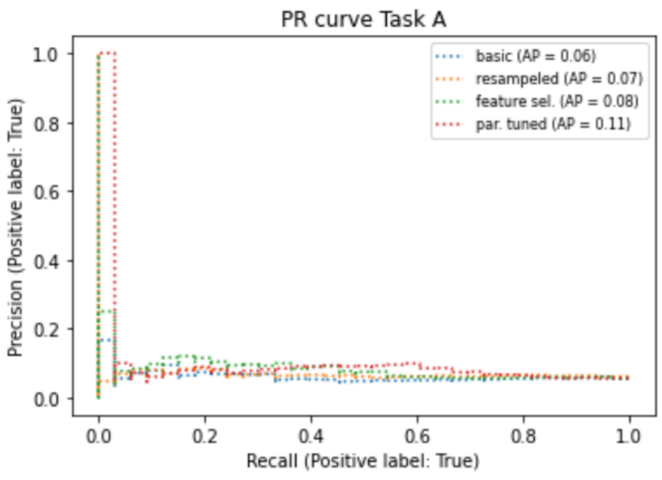
AUPRC – mean = 0.549, std = 0.21

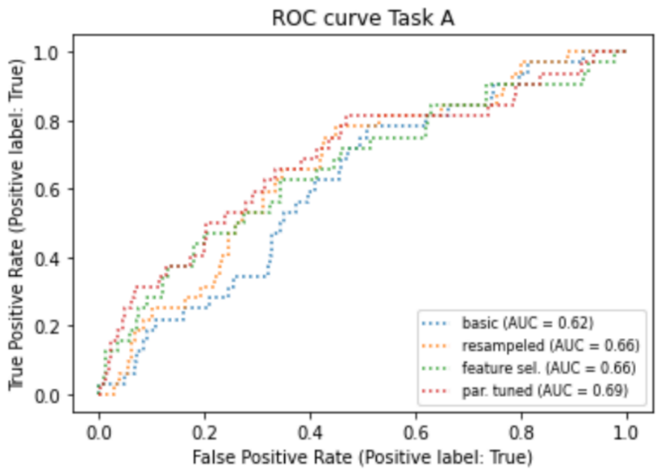
AUROC – mean = 0.775, std = 0.12

In the following graphs we can see the improvement after each step.

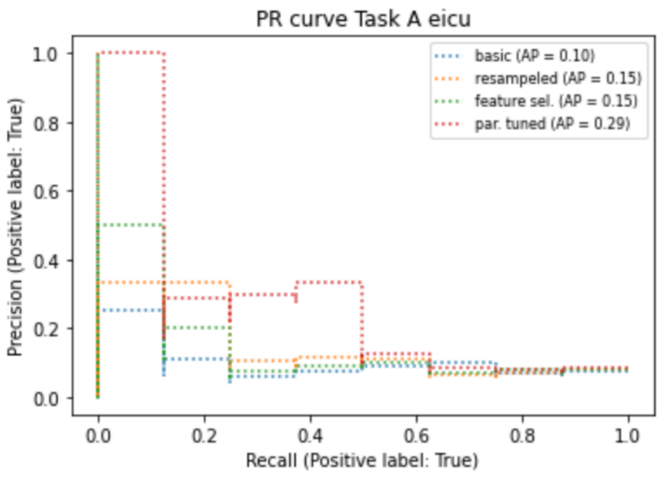
For each task we show only the chosen classifier after each step: First the basic classifier (blue), second the results after resampling the data (orange), then after resampling and selecting features (green) and finally the final classifier after resampling the data, selecting the features and tuning the classifier’s parameters (red).

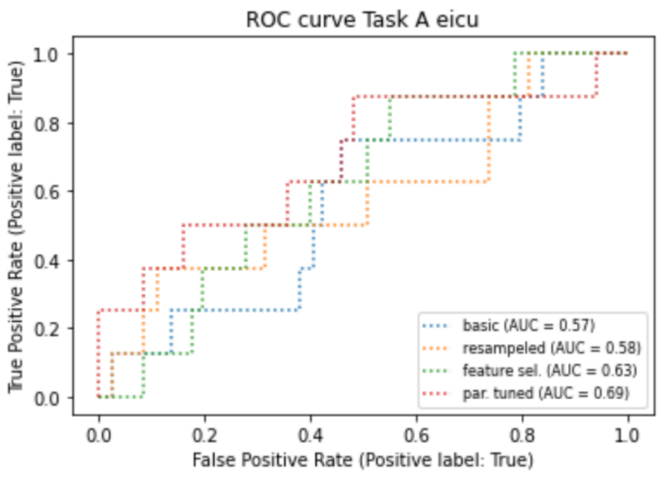
Task A – MIMIC train, MIMIC validation (one of the given folds):



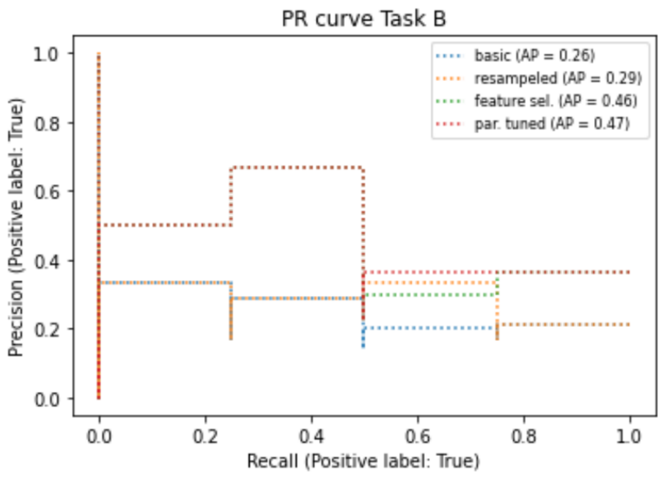


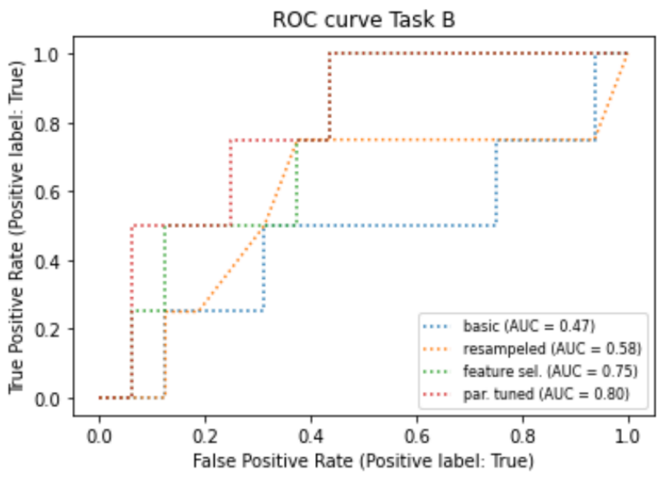
Task A – MIMIC train, eICU validation:





Task B – MIMIC train, MIMIC validation (one of the given folds):





Discussion

As emphasized through the models and results sections, the high variance of the models limits the accuracy of the chosen methods and parameters.

Another issue the pipelined decision-making raises is using the same data for all the stages. It might lead to bias in our evaluation scores: the decisions are made using all the data (averaged over different folds), so there is a leak of information from the one phase to another. The chosen methods of the previous phases of the pipeline are chosen based on the current phase's training set and validation set. We've decided that in the tradeoff between dividing the data between the different decisions (resampling method, feature selection, etc.) and giving the estimators a sufficient amount of data to tarin and evaluate on, we tend to prefer the latter.

Moreover, the order of the decision making has a great influence on the results and if we changed this order, it might have had an impact on the decisions we took and the results.

We have learned to value the importance of parameter tuning and we were encountered with many new algorithms and methods to handle class-imbalancing and handle with a small amount of data.

References

[1] Johnson, A., Pollard, T., Shen, L. et al. MIMIC-III, a freely accessible critical care database. Sci Data 3, 160035 (2016). https://doi.org/10.1038/sdata.2016.35

[2] Pollard, T., Johnson, A., Raffa, J. et al. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. Sci Data 5, 180178 (2018). https://doi.org/10.1038/sdata.2018.178

[3] Jerez, José M., et al. "Missing data imputation using statistical and machine learning methods in a real breast cancer problem." Artificial intelligence in medicine 50.2 (2010): 105-115.

[4] Fan, Wei, et al. "AdaCost: misclassification cost-sensitive boosting." Icml. Vol. 99. 1999.

[5] Sun, Yanmin, Andrew KC Wong, and Mohamed S. Kamel. "Classification of imbalanced data: A review." International journal of pattern recognition and artificial intelligence 23.04 (2009): 687-719.

[6] https://github.com/joelprince25/adacost

[7] Laurikkala, Jorma. "Improving identification of difficult small classes by balancing class distribution." Conference on Artificial Intelligence in Medicine in Europe. Springer, Berlin, Heidelberg, 2001.

[8] Chawla, Nitesh V. "Data mining for imbalanced datasets: An overview." Data mining and knowledge discovery handbook (2009): 875-886.

[9] Fu, Guang-Hui, et al. "Stable variable selection of class-imbalanced data with precision-recall criterion." Chemometrics and Intelligent Laboratory Systems 171 (2017): 241-250.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn-ensemble-randomforestclassifier

[11] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html#sklearn.metrics.average_precision_score

[12] Ozenne, Brice, Fabien Subtil, and Delphine Maucort-Boulch. "The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases." Journal of clinical epidemiology 68.8 (2015): 855-859.