Unlocking the Secrets of Heart Transplant Success with Machine Learning

Trevor Leach
Department of Computer Science
College of Charleston
Charleston, SC
leachta@g.cofc.edu

***Abstract*—Machine learning (ML) models have been increasingly applied in healthcare to help predict medical outcomes, including the success of heart transplantation. The United Network for Organ Sharing (UNOS) provides a CSV dataset of over 40,000 heart transplants in the United States, which can be used to develop ML models for predicting transplant failure. The goal of the model developed in this project is to identify patients at risk of heart transplant failure within 365 days post-transplant to allow for earlier intervention and improved outcomes. This project makes use of Python's Scikit Learn, Pandas, and XGBoost libraries to build a binary classification model to predict whether a patient's transplant will fail. Algorithms, such as decision trees and support vector machines, are used to train the model to predict the outcome given features such as demographic information, medical history, and pre-transplant test results. The model's performance in predicting heart transplant failure is evaluated using metrics such as accuracy, precision, recall, and F1-score. The final model is expected to predict transplant failure, while minimizing false positive and false negative results and performs well on new data. The use of UNOS transplant data and ML models has the potential to greatly improve the outcomes of heart transplantation. By identifying patients at risk of transplant failure, healthcare providers can intervene earlier and provide necessary care to prevent failure. This can lead to better patient outcomes, improved use of resources, and a higher success rate for heart transplantation.**

I. Introduction

Heart transplantation is an invasive procedure done to replace a damaged or failing heart and an individual with a functioning heart of a deceased individual. In the last decade, over 31,000 heart transplants have taken place within the United States [1]. The average heart transplant incurs a total cost of $1.6 million, which includes $1 million for the hospital admission, $111,000 for the physician cost, and $270,000 for 180 days of post-transplant medical care [2]. This cost poses a significant weight on the patient, insurance companies, and the hospital.

In addition to the significant financial burden of heart transplants, there is also a number of individuals on the wait list to receive a heart transplant. As of 2019, there are 7,562 adults and 1,087 pediatric patients on the wait list [3]. 12% of the adult candidates have been on the wait list for over 2 years and 26% of pediatric candidates have been on the wait list for more than 90 days [3]. Additionally, 3,597 adult transplants and 509 pediatric transplants were performed in 2019 with first year mortality rates of 7.9% and 8.2%, respectively [3].

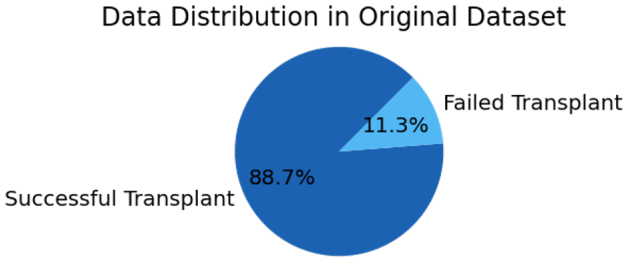


Fig. 1. Data distribution within the original heart transplant dataset.

Due to the number of candidates waiting for this life-saving procedure, it is important that available organs are allocated properly.

II. The Dataset

The data used in this project comes from the United Network for Organ Sharing (UNOS). UNOS is a federally contracted nonprofit that regulates all organ transplants within the United States. As part of this regulation, UNOS maintains large databases for all kinds of organ transplants and manages the matching of organ donors and recipients [4]. These datasets are available to any individual through the UNOS website and can be requested for individual use.

The heart transplant dataset contains over 500 features and over 40,000 objects, each representing a single heart transplant occurrence. The original data distribution is shown in Fig. 1. The heavily unbalanced classes in the data will present an issue during model training as only 11.3% of objects in the dataset are failed transplants.

III. Data Pre-Processing

The dataset from UNOS is a cleaned dataset as UNOS closely manages its integrity. Therefore, pre-processing for this project entailed small changes and filtering of the dataset for use in the model.

A. Determining Transplant Failure Time

The dataset does not include columns for transplant failure within the first year. Instead, it includes a GTIME column that indicates the graft lifespan in days and a GSTATUS column that indicates whether the graft has failed (1=Yes). Using

```
conditions = [
    ((df['GTIME'] <= 365) & (df['GSTATUS'] == 0)),
    ((df['GTIME'] <= 365) & (df['GSTATUS'] == 1)),
    ((df['GTIME'] > 365) | (df['GSTATUS'] == 0)),
]

choices = [
    np.nan,
    0,
    1
]

df['365DayOutcome'] = np.select(conditions, choices, default = np.nan)
```

Fig. 2. Code snippet to create a new column that indicates graft failure within 365 post-transplant.

these two columns, graft failure at the 365 day mark can be determined.

Using the Numpy Python library, a new column was created to include the graft outcome at the 365 day mark post-transplant. The following logic is used in Fig. 2 to create the new 365Outcome column:

- If GTIME is less than or equal to 365 and the GSTATUS if 0, the transplant is younger than a year old and has not failed, which will then be assigned an outcome status of NAN.
- If GTIME is less than or equal to 365 and the GSTATUS is 1, the graft failed within the first year and an outcome of 0 is assigned to that object.
- If the GTIME is greater than 365 or the GSTATUS is 0, the graft is either older than a year or has not failed and that object is assigned an outcome of 1 to indicate a successful transplant.

B. Feature Filtering

Due to the timeline for the project, physicians were consulted on which features they would expect to be the most important in predicting heart transplant outcomes. Those features are listed in the code snippet in Fig. 3.

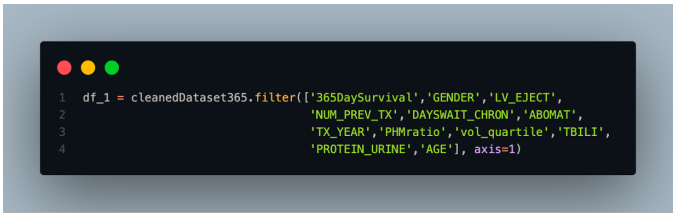
C. Feature Engineering

Three of the physician-specified features were objects that needed to be translated to other types prior to training the model. The features are as follows:

- GENDER: Original M/F data mapped to 0/1
- ABOMAT: Original ABO incompatible/ABO compatible/ABO identical mapped to 0/1/2
- PROTEIN_URINE: Original Y/N converted to boolean True/False

D. Drop Null Objects

To use a Random Forest algorithm, all features for each object must have a populated value. Therefore, null values must either be filled or dropped prior to model building. Since this is a substantial dataset, the null values were dropped rather than populated to maintain data purity.



```
df_1 = cleanedDataset365.filter(['365DaySurvival','GENDER','LV_EJECT',
                                 'NUM_PREV_TX','DAYSWAIT_CHRON','ABOMAT',
                                 'TX_YEAR','PHMratio','vol_quartile','TBILI',
                                 'PROTEIN_URINE','AGE'], axis=1)
```

Fig. 3. Code snippet to filter the columns physicians specified as being important to predicting heart transplant failure.

E. Synthetic Minority Oversampling Technique

To address the unbalanced classes in the dataset, the Synthetic Minority Oversampling Technique (SMOTE) was used to create synthetic data points of the minority class. An unbalanced dataset will result in a model with high accuracy and low specificity. Models trained on unbalanced datasets will be able to accurately classify the majority while showing poor specificity, or accuracy classifying the minority. In this context, models without SMOTE applied will not be able to accurately classify failed heart transplants sufficiently.

SMOTE works by sampling a minority data point, connecting that point to nearby minority data points via the k nearest neighbor algorithm, and introducing new synthetic data points along those lines [5]. This algorithm is used in many unbalanced dataset projects and is seen as a valid way to improve the model's performance. A critique of oversampling techniques is that they can produce ambiguous data points and blur the line between the classes.

F. Edited Nearest Neighbor

Studies suggest that the combination of an oversampling technique, such as SMOTE, with an undersampling technique, such as edited nearest neighbor (ENN), can be used to address the issue of ambiguous data points resulting from only using an oversampling technique. ENN works by finding the k nearest neighbors of a data point and evaluating their labels. If the majority of the neighbors share the same label as the data point in question, the data point is not considered noise and it left in the dataset. If the majority of neighbors are of the opposite class, that data point is removed from the dataset to make the distinction between classes more distinct [6].

SMOTE and ENN are applied concurrently using the imblearn Python library. The code in Fig. 4 shows the implementation. The X and y values are separated and passed into the SMOTEENN instance to fit and re-sample the data. The X and y that are returned will be equal in number and have gone through the undersampling technique to reduce ambiguity in the data. Once completed, the data are ready to be used for training the model.

IV. Methods and Models

A. XGBoost vs. Random Forest

The Random Forest algorithm implements multiple decision trees all built simultaneously on varying subsets of the data. The dataset goes through a bagging process to provide each

```python
from imblearn.combine import SMOTEENN

X = df_smote.drop(['365DaySurvival'], axis = 1)
y = df_smote['365DaySurvival']

smt = SMOTEENN(random_state=42)
X, y = smt.fit_resample(X, y)
```

Fig. 4. Code snippet for implementing the hybrid SMOTE/ENN algorithm to oversample and undersample the dataset and produce balanced classes.

tree with differing data to build from. At the end, the trees will be used together to predict classifications. Data is sent through all of the trees, each tree makes its prediction and casts a vote, and the overall predicted classification is made based on majority voting.

This project makes use of the XGBoost algorithm, a decision tree-based algorithm that implements gradient boosting to produce a strong classifier. The data is passed through the first weak classifier (decision tree), classification predictions are made, error is calculated, and the results are passed to the next weak classifier to correct the errors. This continues through a number of weak classifiers until the end of the algorithm at which point a strong classifier results. This process of building multiple models sequentially is called boosting [7]. As they both are based on decision trees, XGBoost and Random Forest perform similarly for this application. XGBoost also has benefits in terms of computational expense and GPU power.

B. Model 1: Physician-Specified Features, XGBoost, and no SMOTEENN

The first model was a preliminary approach that used the features physicians called out as important to their decision making. This model was trained with the XGBoost algorithm prior to balancing the data with SMOTEENN. After testing, the model produced the confusion matrix in Fig. 5 and accompanying performance metrics in Fig. 6.

These metrics show that this model performs well in multiple aspects but falls short in specificity. The model has 92.2% accuracy but only 47.6% specificity, showing that it can only accurately predict failed heart transplants 47.6% of the time. This shows the importance of a balanced dataset. Since this used the original unbalanced dataset, the model can predict that all patients will have a successful transplant and still show high accuracy rates.

C. Model 2: Physician-Specified Features, Random Forest, and no SMOTEENN

It is important to consider various algorithms when constructing a model as different algorithms can produce differing results. In the second model, the Random Forest algorithm was used to compare its performance with the XGBoost model. The second model was also run on the same features as Model 1 and prior to using SMOTEENN on the dataset.

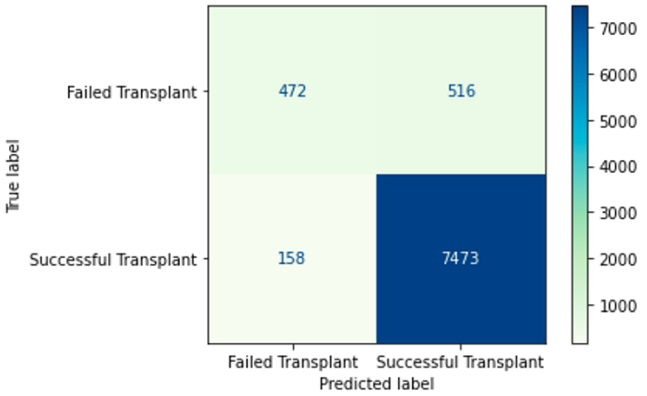


Fig. 5. Confusion matrix produced from Model 1.

Metric	Score
Accuracy	0.922
Precision	0.935
Recall	0.98
Specificity	0.476
F1 Score	0.957
AUC ROC	0.728

Fig. 6. Performance metrics for Model 1.

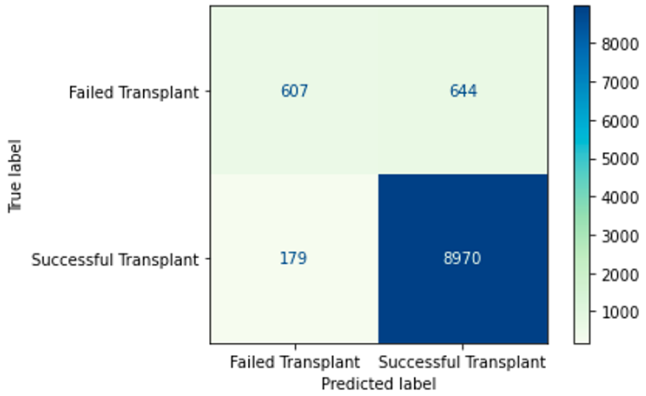


Fig. 7. Confusion matrix produced from Model 2.

Metric	Score
Accuracy	0.921
Precision	0.933
Recall	0.98
Specificity	0.485
F1 Score	0.956
AUC ROC	0.733

Fig. 8. Performance metrics for Model 2.

The confusion matrix in Fig. 7 and metrics in Fig. 8 show that the performance of the Random Forest model is very similar to the XGBoost model. Either algorithm could be used in this project to result in similar performance. For the following models, the XGBoost algorithm will be used.

D. Model 3: Physician-Specified Features, XGBoost, SMOTE, and no ENN

The specificity scores for the last two models were around 50%, indicating a significant lack in the models' ability to predict a failed transplant. To address this shortcoming, SMOTE was used to oversample the minority class and produce new data points to balance the classes. In this model, no undersampling technique was applied to reduce ambiguity in the data.

After building and training the XGBoost model, the confusion matrix in Fig. 9 was made, showing significantly better classification performance. The metrics in Fig. 10 also show that the specificity increased over 100% after using SMOTE to balance the dataset.

E. Model 4: Physician-Specified Features, XGBoost, and SMOTEENN

The creator of the SMOTE algorithm indicated that the use of an undersampling technique alongside SMOTE can result in a better data distribution for modeling. For this project, the ENN algorithm was selected for use as it had better performance metrics than Tomek techniques.

Including ENN continued to increase the model's accuracy and specificity. The increase of about 2% does not make this model significantly better than the previous, but it does show that the ENN increases model performance. Fig. 11 and Fig 12 summarize this model's performance.

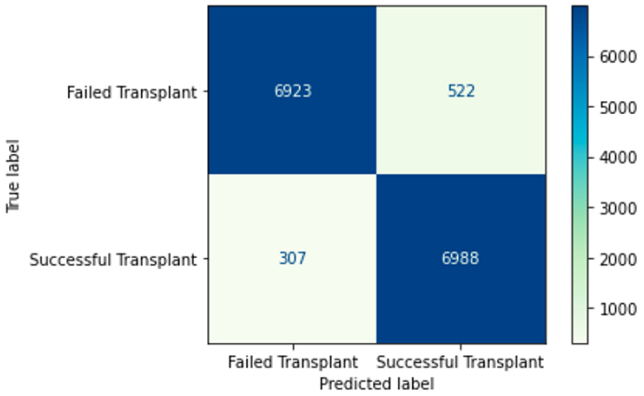


Fig. 9. Confusion matrix produced from Model 3.

Metric	Score
Accuracy	0.944
Precision	0.93
Recall	0.958
Specificity	0.93
F1 Score	0.944
AUC ROC	0.944

Fig. 10. Performance metrics for Model 3.

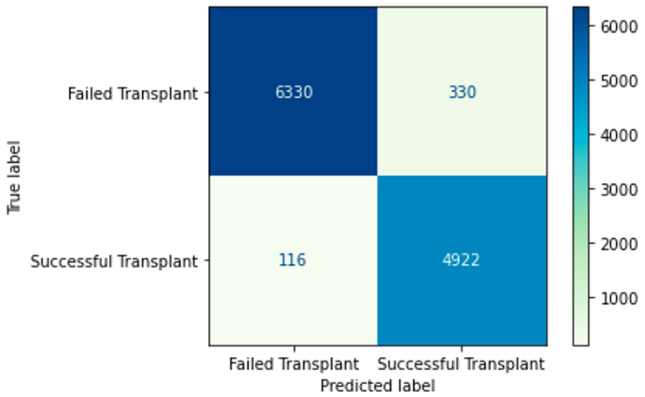


Fig. 11. Confusion matrix produced from Model 4.

Metric	Score
Accuracy	0.962
Precision	0.937
Recall	0.977
Specificity	0.95
F1 Score	0.957
AUC ROC	0.964

Fig. 12. Performance metrics for Model 4.

F. Model 5: Physician-Specified Features, XGBoost, SMOTEENN, and no TX_YEAR

One of the features physicians called out as important to be used in the model is the year the transplant took place (TX_YEAR). This feature is not a patient characteristic and could not be used for future predictors if there is little data for that year. For example, if a patient is having a transplant in January 2024, there would be little or no data on transplants in 2024 and therefore the prediction would be skewed. As this model is attempting to use patient characteristics and health as predictors for transplant failure, TX_YEAR does not logically make sense to include in the model.

This final model is identical to Model 4, but does not include TX_YEAR. The metrics show that the accuracy and specificity decreased slightly, but not to the point of resulting in a poor model. The confusion matrix in Fig. 13 and metrics in Fig. 14 show a high performing model.

V. Results and Interpretation

The final model using the XGBoost algorithm with SMOTEENN and using the physician-specified features except for TX_YEAR is the best performing model in this study. Although it does not have the highest performance indicators, it includes only patient-centric features and has the most valuable output of the models.

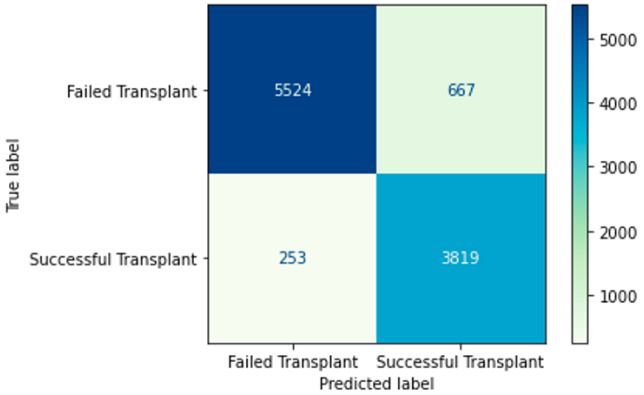


Fig. 13. Confusion matrix produced from Model 5.

Metric	Score
Accuracy	0.91
Precision	0.851
Recall	0.938
Specificity	0.892
F1 Score	0.892
AUC ROC	0.915

Fig. 14. Performance metrics for Model 5.

To ensure that the final model has not been overfit, a 10-fold cross validation was run to evaluate the critical performance metrics. Fig. 15 shows how each metric progressed through the cross validation, all eventually leading to high scores. The graph indicates that the model performs well and it not overfit to the data.

Additionally, the important features for the model are shown in Fig. 16 in descending order. This figure shows that the model can be simplified by removing the PROTEIN_URINE feature as it does not play a major role in classification. The other features do show significant weight in classification. A dictionary of the terms can be found in Table I.

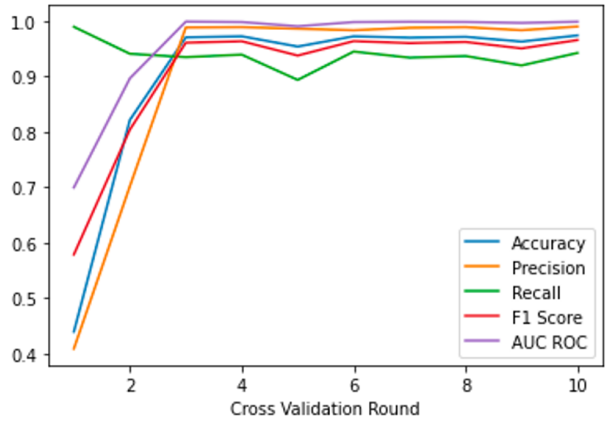


Fig. 15. Performance results from a 10-fold cross validation on Model 5.

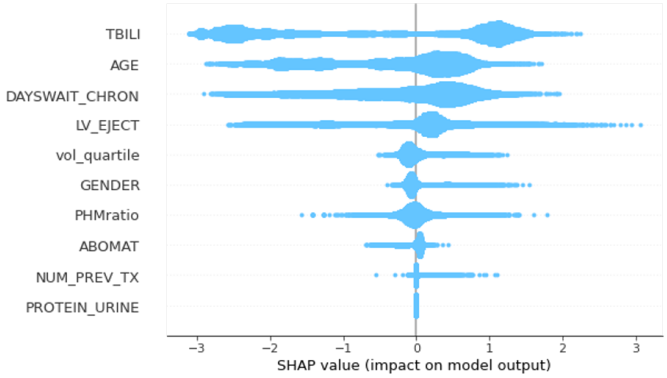


Fig. 16. Top features involved in classification for Model 5. The larger area indicates more impact on classification.

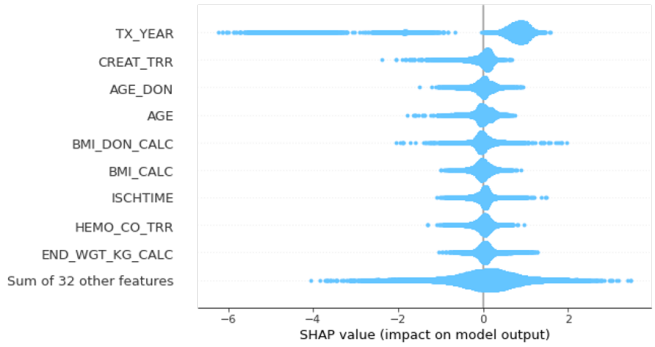


Fig. 17. Top features involved in classification for a model that includes all integer, float, and boolean features from the original dataset.

Since the feature subset used in Model 5 excluded a large number of features from the original dataset, it is beneficial to compare the top features for Model 5 and for a model that includes all integer, float, and boolean features. The overall top features for the entire dataset are shown in Fig. 17. When comparing Fig. 16 to Fig. 17, there are a number of features that are important in the overall dataset that were excluded from the subset used for Model 5. The differences in these figures show room for future work to include top features from Fig. 17 in a model to improve accuracy and specificity.

VI. Discussion

This project shows how effective machine learning can be in the healthcare industry. When combining clean data with trusted algorithms, accurate predictions can be made about whether a heart transplant will fail or succeed.

Future work with this project can involve the use of other top features in the model that were excluded from this model based on physician recommendations. The benefit of this work is that physicians can see that there are other critical components to their decision making that they may not have considered as strongly prior to this study. These results are actionable and can be used across the country.

Another point of future work is to only apply SMOTEENN to the training set and maintain the original test set so that the

ABOMAT	Donor-recipient blood matching
AGE	Age of the recipient
AGE_DON	Age of the donor
BMI_CALC	Recipient BMI
BMI_DON_CALC	Donor BMI
CREAT_TRR	Recipient serum creatinine
DAYSWAIT_CHRON	Total days on waiting list
END_WGT_KG_CALC	Recipient weight in kilograms
GENDER	Recipient gender
HEMO_CO_TRR	Recipient hemodynamics in L/min
ISCHTIME	Ischemic time in hours
LV_EJECT	Donor left ventricle ejection
NUM_PREV_TX	Number of previous transplants
PHMratio	Predicted heart mass ratio
PROTEIN_URINE	Donor protein in urine
TBILI	Recipient serum total bilirubin
TX_YEAR	Transplant year
vol_quartile	Heart volume

TABLE I
DICTIONARY OF FEATURE TERMS

model is only predicting true data. This could lead to a change in accuracy and specificity if the model has experienced overfitting to the synthetic data points.

VII. Conclusion

The results of this study show that balancing an unbalanced dataset using oversampling and undersampling techniques can produce a high performing and effective machine learning model. Additionally, XGBoost is an effective boosted algorithm that can produce results similar to Random Forest while conserving computational power.

For this domain in particular, these results show that we can accurately predict heart transplant failures and successes using machine learning. The UNOS dataset provides significant information that can lead to more effective allocation of organs.

Acknowledgment

Thank you to the Medical University of South Carolina and Roshan Mathi, MSDS for providing mentorship during this project and providing the data to be used.

References

[1] "Heart Transplant Sets All-Time Record in 2021." UNOS, 17 Feb. 2022, https://unos.org/news/in-focus/heart-transplant-all-time-record-2021/#:~text=2021%20is%20also%20the%20tenth,at%20hospitals%20across%20the%20country.

[2] "Finding Financial Support for Heart Transplants." Help Hope Live, 23 Feb. 2023, https://helphopelive.org/heart-financial-assistance/.

[3] Health Resources and Services Administration. (2019). *OPTN/SRTR 2019 Annual Data Report: Heart*. https://srtr.transplant.hrsa.gov/annual_reports/2019/Heart.aspx#HR_activity_adult_rem_reason_b64

[4] "What Is UNOS?: About United Network for Organ Sharing." UNOS, 21 Apr. 2022, https://unos.org/about/.

[5] Chawla, N. V., et al. "Smote: Synthetic Minority over-Sampling Technique." Journal of Artificial Intelligence Research, vol. 16, 2002, pp. 321–357, https://doi.org/10.1613/jair.953.

[6] Zhaozhao, X., et al. "A hybrid sampling algorithm combining M-SMOTE and ENN based on Random forest for medical imbalanced data." Journal of Biomedical Informatics, vol. 107, 2020.

[7] Mitchell, R and Frank, E. "Accelerating the XGBoost algorithm using GPU computing." PeerJ Computer Science, 2017.