Robustness of Machine Learning Classification Models to Imbalanced In-Hospital Mortality Data

Group 6, INST0060: Foundations of Machine Learning 24/25

SN: 24040933 SN: 23100556 SN: 24106277 SN: 24056647 SN: 24049461

***Abstract*—Class imbalanced datasets present a significant challenge for classification tasks in machine learning, often leading to biased model performance and skewed evaluation metrics. This report explores the effects of training and testing machine learning models under varying degrees of class imbalance in a dataset for predicting in-hospital patient mortality. Three classification models—Logistic Regression, k-Nearest Neighbor, and Random Forest Classifier—are evaluated using the area under the Receiver Operating Characteristic curve metric to provide insights into how robust the performance of these models are to dataset class imbalances. Findings indicate that models trained and tested on more balanced datasets perform better. Additionally, the models' sensitivity to class imbalance varied. Logistic Regression exhibited the most significant performance drop under shifting imbalance conditions, while k-Nearest Neighbor and Random Forest Classifier showed greater adaptability across all datasets.**

***Index Terms*—Machine Learning, Class Imbalance, Classification, Receiver Operating Characteristic, Area Under the Curve, Logistic Regression, k-Nearest Neighbors, Random Forest Classifier, Binary Classification, Healthcare Data Analytics, Imbalanced Dataset, Supervised Learning, Model Robustness.**

I. Introduction

Class imbalance in datasets presents a common challenge for classification tasks in machine learning (ML) tasks. An over- or under-representation of one class of data can affect the performance of ML models in a number of ways, such as skewing a model's classification accuracy, which in turn can impact our understanding of how well a model is performing on a dataset [1].

This project seeks to evaluate the impact of training three models - Logistic Regression (classification) (LR), k-Nearest Neighbor (kNN), and Random Forest Classifier (RFC) - under one degree of class imbalance and testing at another degree of class imbalance. This experiment can provide insights into two interesting research questions:

1) How does class imbalance between training and testing datasets impact the true positive rate (TPR) against the false positive rate (FPR) of a model? In this project, the metric used to evaluate each model's performance in this respect is the Area Under the Receiver Operating Characteristic curve (AUC).
2) Do different types of models deal better with class imbalance between training datasets, and if so, why?

The master dataset used in the project is *Patient Survival Prediction* [2], which contains predictors of all-cause in-hospital mortality for admitted patients. The data represent a binary classification problem: the patient survived (hospital_death = 0) as the majority class and the patient died (hospital_death = 1) as the minority class. Developing an accurate classification model to predict in-hospital patient mortality has real-life implications for improvement of care in hospitals around the world to lower patient mortality rates.

Three degrees of class imbalance were created in three derived datasets with the following class imbalance ratios (majority class : minority class): 90:10, 70:30 and 50:50. Each model was trained on a training set from each derived dataset, and then tested on all testing sets of the derived datasets. For example, models trained on a derived training set of 90:10 class imbalance are tested on a 90:10 imbalanced derived test set for a baseline performance measurement, then on derived test sets of 70:30 and 50:50 class imbalances for comparison.

II. Discussion

This report outlines the experimental design used to investigate the robustness of the chosen ML models' ability to accurately classify data to changes in class imbalance. It includes an overview of: the models, performance metric selection, validation approaches, model and hyper-parameter selection, data pre-processing steps, the experiment design, and the results and findings. The results indicate that models generally perform best when trained and tested on datasets with similar imbalances. Further, non-linear models performed better than the LR model in nearly all cases, demonstrating their suitability for classification of the high-dimensional imbalanced dataset used. Further, non-linear models trained on moderate class imbalance (70:30) demonstrated just as good generalization as balanced data on different class imbalanced sets. This is important for real-world classification tasks as real-life data is most likely to have some class imbalance, meaning we can be more confident in these models' performance outside of experimental conditions. Future research should continue to explore the role of model selection, data pre-processing techniques for mixed categorical and numerical data, and the development of more sophisticated techniques to address class imbalance in ML, as further investigation was limited due to the project requirements.

III. Background

Models, Algorithms and Data Representations

Below is a description of the three models used and their underlying principles:

A. Logistic Regression

LR is a linear model designed for binary classification [3]. It predicts the probability $P(y = 1|x)$ using the sigmoid function [4]:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

Here, w represents the weight vector, x is the input feature vector, and b is the bias term. The objective function is the **binary cross-entropy loss** [5]:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

LR assumes linear separability in the feature space and is optimized using techniques such as gradient descent [3]

B. K-Nearest Neighbor

KNN is a non-parametric, instance-based learning algorithm that classifies a data point based on the majority label of its k-nearest neighbors in the feature space [6]. The distance metric, typically Euclidean distance, is used to identify the nearest neighbors [7]:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{m} (x_{i,k} - x_{j,k})^2}$$

The algorithm does not involve training per se; instead, the computational effort lies in searching for the nearest neighbors during inference. The choice of k is critical for balancing bias and variance.

C. Random Forest Classifier

RFC is an ensemble learning method that combines multiple decision trees through bagging. Each tree is trained on a bootstrap sample of the data, and features are randomly selected at each split to reduce over-fitting. The final prediction is obtained by majority voting among the individual trees for classification [8].

The loss minimized by each decision tree is the **Gini impurity** for classification [9]:

$$G = \sum_{i=1}^{C} p_i (1 - p_i)$$

where p_i is the proportion of samples belonging to class i, and C is the total number of classes. RFC leverages feature randomness and averaging to improve robustness and generalization.

Performance Metrics

We used the **AUC** metric to evaluate the performance of the models. The AUC measures the ability of the model to distinguish between classes by calculating the area under the **Receiver Operating Characteristic (ROC)** curve. The ROC curve represents the trade-off between the TPR (sensitivity) and the FPR (1-specificity) across various threshold levels. A higher AUC indicates better performance, with values ranging from 0.5 (random guessing) to 1.0 (perfect classifier).

Validation Approaches

We adopted a two-step validation strategy:

1) **K-Fold Cross-Validation:** The dataset was divided into k-folds (e.g., $k = 5$), where the model was trained on $k - 1$ folds and validated on the remaining fold. This process was repeated k times, ensuring each fold was used as a validation set exactly once. The results were averaged to estimate the model's generalization ability.
2) **Final Train-Test Split:** After tuning hyperparameters using cross-validation, the dataset was split into separate training (70%) and testing sets (30%). The model was retrained on the training set, and its performance was evaluated on the unseen testing set using AUC.

This combined approach ensured robust model evaluation and minimized the risk of over-fitting.

Dataset

The master dataset contains 83 unique features used to predict in-hospital patient death, and is made up of 91713 unique data points. Most features consist numerical data, with a minority being categorical data.

The class imbalance of the master dataset is approximately 91.4% the majority class, and 8.6% the minority class.

Predictors of in-hospital mortality for patients have not been well characterized. It would be useful develop a prediction model for in-hospital mortality among admitted patients, and to determine which features are the have the largest influence on in-hospital mortality. These insights could be used by management in hospitals and attached healthcare services to put in place measures to monitor and reduce risks more strongly correlated with higher in-hospital mortality rates.

IV. Method

Research aims

This project examines how training three models under one degree of class imbalance and testing under another affects classification performance. The models include a linear approach (LR) and nonlinear approaches (kNN, and RFC), reflecting different classification premises. LR assumes a linear decision boundary, KNN relies on local instance-based learning, and RFC captures complex patterns with an ensemble of decision trees.

Two key research questions guide this work. The first investigates how varying class imbalances between training and testing datasets impact the trade-off between TPR and

FPR, measured via AUC. Performance is compared for the same model under different imbalance conditions, not across models.

The second question aims to compare how well linear and nonlinear models handle class-imbalanced data, again using AUC to evaluate robustness under identical imbalance scenarios.

The findings are relevant to predicting in-hospital patient mortality, where class imbalance (survivors vs. deaths) varies across patient cohorts or causes of hospitalization.

Pre-processing

Pre-processing focused on creating derived datasets from the master dataset, ensuring class ratios were controlled as the experimental variable. The master dataset was converted into a data frame, inspected for duplicates, and missing values were imputed (mean for numeric features, mode for categorical) [10]. Identifier columns (e.g., encounter_id, patient_id) were removed, as they were irrelevant for predicting patient death.

The master dataset was stratified randomly into three subsets (each one-sixth of the original size) to meet project time constraints while retaining the original class imbalance [11]. Binary encoding was applied to categorical features for over-sampling and model compatibility [12], [13]. Synthetic Minority Over-sampling Technique for Nominal and Continuous (SMOTE-NC) was used to generate datasets with class ratios of 90:10 (Data1), 70:30 (Data2), and 50:50 (Data3), oversampling only the minority class to maintain the majority class size and keep class imbalance as the sole experimental factor [14], [15]. Under-sampling of the majority class was avoided to preserve potential patterns in the original data [16].

Each derived dataset was split into training and testing sets (70:30) via stratified random sampling to maintain class imbalance. Cross-validation was used as the validation strategy, eliminating the need for a separate validation set. This pipeline of cleaning, subset creation, encoding, synthetic oversampling, and splitting ensured datasets were suitable for investigating the effects of class imbalance in the experiments.

Model and hyper-parameter selection

A. Logistic Regression

LR is a well-established model for binary classification tasks, where the goal is to predict the probability of a binary outcome (in this case, hospital_death). The decision boundary in LR is determined by fitting a linear model to the data, making it computationally efficient and interpretable, while providing strong performance in cases where the relationship between features and the target is approximately linear [3]. The hyperparameter **C** controls the trade-off between fitting the model well on the training data and keeping the model weights small to prevent over-fitting. The **solver** hyperparameter defines the algorithm used to optimize the model. Two solvers were considered: **liblinear** and **saga**. Each solver has different characteristics, and the choice of solver may depend on factors like the size of the dataset and the number of features. The hyperparameter **Max Iter** sets the maximum number of iterations for the solver to converge. A higher value ensures that the solver has more time to find the optimal model parameters but can also increase computational time.

B. k-Nearest Neighbors Classifier

KNN is an instance-based classification model chosen for its ability to handle non-linear relationships and imbalanced data. By comparing distances between samples, kNN captures local patterns while adapting to global data distributions [6]. The hyperparameter **n_neighbors**, defining the number of neighbors used for classification, was tuned to explore the trade-off between local sensitivity and model stability; smaller values emphasize local patterns but risk over-fitting, while larger values improve generalization. The **weights** parameter, tested with uniform and distance-based options, evaluates whether closer neighbors contribute more effectively to predictions. Finally, the **distance metric** was optimized by comparing Euclidean, Manhattan, and Minkowski distances. The Minkowski metric serves as a generalization parameter, capable of adapting to diverse data characteristics by adjusting its parameters, encompassing both Euclidean and Manhattan as special cases. This flexibility allows kNN to effectively handle datasets with varying numerical and categorical features.

C. Random Forest Classifier

RFC is a robust model for this project because it combines multiple decision trees, reducing over-fitting while maintaining high accuracy through ensemble learning [8]. Its ability to handle non-linear relationships and high-dimensional data makes it well-suited for capturing complex patterns under varying class imbalances. The hyperparameter **n-estimators**, which determines the number of trees in the forest, was chosen to balance computational efficiency and model stability, as a larger number of trees generally improves performance but increases training time. The **max-depth** hyperparameter controls the depth of each tree, preventing over-fitting by limiting how granularly the model learns from the data. These two hyperparameters were selected because they significantly influence the model's bias-variance trade-off and overall predictive performance.

Evaluation metric selection

The ROC curve and AUC provide a threshold-independent evaluation of model performance, making them ideal for this project [17]. AUC summarizes the model's ability to distinguish between classes in a single value (0.5 for random guessing, 1.0 for perfect classification), offering a clear benchmark for comparison. Unlike precision and recall, AUC captures performance across all thresholds and avoids biases from class imbalance by focusing on TPR and FPR. This ensures robust, interpretable insights into the model's trade-offs between false positives and false negatives.

V. Results

A. Logistic Regression Model Performance

Thee hyperparameters of the model (C, solver, and max_iter) were optimized using manual grid search with 5-

fold cross-validation. The best hyper parameter combinations for LR across all datasets sorted by mean AUC are shown on Table I), note they all have the same solver (liblinear) and max_iter (1000), and dataset2's C is 1.0 which is the only difference with dataset1 and dataset3.

TABLE I: Best hyperparameter combinations for LR across all datasets (sorted by mean AUC)

Dataset	C	solver	max_iter	Mean AUC
Dataset1	10	liblinear	1000	0.8534
Dataset2	1.0	liblinear	1000	0.8912
Dataset3	10	liblinear	1000	0.9057

1) Logistic Regression Model Evaluation:

a) Tests on Data1 (90:10) - Figure 1: The LR model trained on Data1 achieved a test AUC of 0.8615. In comparison, models trained on Data2 and Data3 achieved slightly lower AUCs of 0.8572 and 0.8453 respectively. This suggests that the LR model trained on Data1 performs better on the test dataset that has the same degree of class imbalance, and that training models on less imbalanced data decreased performance on extreme imbalanced data.

b) Tests on Data2 (70:30) - Figure 2: The model trained on Data3 achieved the highest AUC of LR tests with 0.8998. Meanwhile, models trained on Data1 and Data2 achieved AUCs of 0.8823 and 0.8558 respectively. The moderate imbalance and balanced data in Data2 and Data3 help the model learn class boundaries more effectively while maintaining good generalization.

c) Tests on Data3 (50:50) - Figure 3: The LR model trained on Data3 achieved a test AUC of 0.9047 on Data3, slightly higher than the Data2-trained model (0.8997) but better than the Data1-trained model (0.8778). Small differences in AUC scores overall indicate that model still is robust to varying degrees of imbalance.

2) Logistic Regression Findings: The degree of class imbalance has a noticeable influences the LR model's performance when the minority class is less represented. When testing on Data1, all models performance was lower than for other datasets. Additionally, Data1 trained models performed the worse on all experiments. Models trained on moderate (70:30) and balanced datasets similarly across all experiments for the LR model, with only slight differences in performance.

B. kNN Model Performance

Grid search was performed to optimize hyperparameters, Table II summarizes the top 5 configurations for each dataset. The results demonstrate the impact of hyperparameter selection and class distribution on model performance.

1) kNN Model Evaluation:

a) Tests on Data1 (90:10) - Figure 4: The kNN model trained on Data1 performs poorly on its own dataset (AUC = 0.7613), reflecting the limitations of extreme class imbalance, which biases predictions toward the majority class. In contrast,

TABLE II: Best hyperparameter combinations for kNN across all datasets (sorted by mean AUC)

Dataset	n_neighbors	weights	metric	Mean AUC
Dataset1	10	distance	manhattan	**0.7772**
Dataset2	5	distance	manhattan	**0.9568**
Dataset3	10	distance	manhattan	**0.9682**

the model trained on Data3 achieves the highest AUC (0.9926) on Data1, demonstrating that balanced training data provides strong generalization for highly imbalanced test datasets. The Data2-trained model also performs extremely well (AUC = 0.9903), indicating that moderate imbalance during training enhances generalization.

b) Tests on Data2 (70:30) - Figure 5: The Data1-trained model performs very well on Data2 (AUC = 0.9566), as the moderately imbalanced nature of Data2 mitigates some over-fitting caused by the extreme imbalance in Data1. The Data3-trained model performs the strongest on Data2 (AUC = 0.9900), while the Data2-trained model also achieves a high AUC (0.9652).

c) Tests on Data3 (50:50) - Figure 6: The Data1-trained model maintained a strong performance on Data3 (AUC = 0.9537), indicating that kNN's reliance on local neighborhoods allows it to generalize reasonably well even when trained on highly imbalanced data. The Data2-trained model achieves the best performance on Data3 (AUC = 0.9893), outperforming the Data3-trained model (AUC = 0.9732). This demonstrates that moderate imbalance during training offers greater robustness and generalization compared to balanced training when testing on diverse datasets.

2) kNN Findings: The model trained on Data2 performed the best on Data3, and nearly matched (with difference of 0.0023) Data3-trained model performance on Data1, highlighting the generalization provided by moderate imbalance. Training on Data1 displayed lower performance across all experiments, likely due to over-fitting to the extreme class imbalance, while training on Data3 performed best across best on Data2 and Data1. These results emphasize that kNN's reliance on local neighborhood structures makes it sensitive to training data distribution, with moderate imbalance and balanced data proving to be similarly effective for generalization.

C. Random Forest Classifier Model Performance

The RFC algorithm was optimized using grid search for each dataset. Table III summarizes the top 5 hyperparameter combinations for each dataset, ranked by their cross-validation AUC scores. The highest-performing hyperparameters for each dataset are highlighted in bold.

TABLE III: Best hyperparameter combinations for Random Forest Classifier across all datasets (sorted by mean AUC)

Dataset	n_estimators	max_depth	Mean AUC
Dataset1	200	15	**0.8920**
Dataset2	200	20	**0.9745**
Dataset3	200	20	**0.9907**

Fig. 1: ROC Curves for LR model tested on Data1

Fig. 2: ROC Curves for LR model tested on Data2

Fig. 3: ROC Curves for LR model tested on Data3

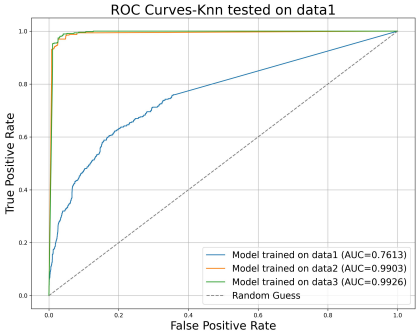

Fig. 4: ROC Curves for kNN model tested on Data1

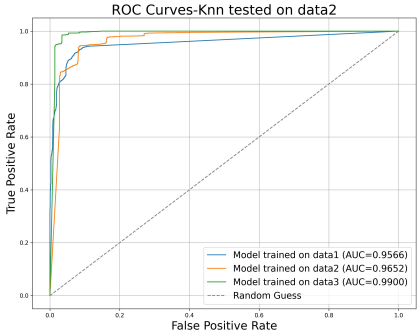

Fig. 5: ROC Curves for kNN model tested on Data2

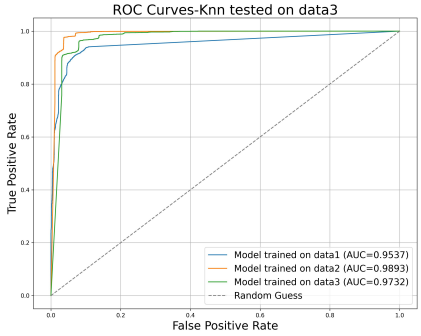

Fig. 6: ROC Curves for kNN model tested on Data3

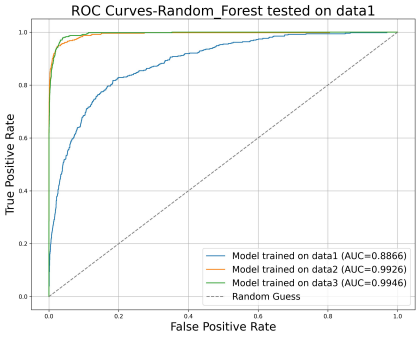

Fig. 7: ROC Curves for RFC model tested on Data1

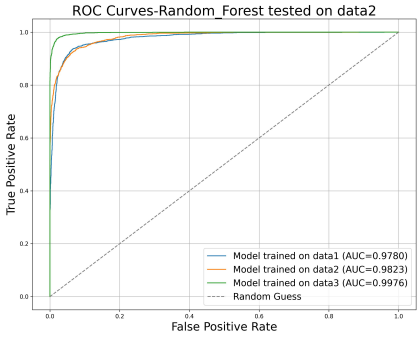

Fig. 8: ROC Curves for RFC model tested on Data2

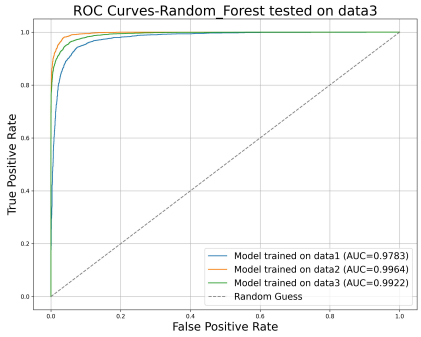

Fig. 9: ROC Curves for RFC model tested on Data3

RFC Model Evaluation:

a) Data1 (90:10) - Figure 7: Testing - The RFC model trained on Data1 achieved a test AUC of 0.8866. Models trained on Data2 and Data3, when tested on Data1, greatly outperformed the Data1 trained model, achieving AUCs of 0.9926 and 0.9946, respectively, suggesting superior generalization due to the higher diversity in balanced or moderately imbalanced datasets.

Training - The best configuration, **n_estimators = 200, max_depth = 15** (AUC = 0.8922), balanced tree depth with sufficient estimators to capture data patterns effectively. The lowest-performing settings, **n_estimators = 100, max_depth = 20** (AUC = 0.8854), likely over-fit due to excessive depth.

b) Data2 (70:30) - Figure 8: Testing - Achieving near perfect classification, the model trained on Data3 achieved the highest AUC of 0.9976 when tested on Data2. Models trained on Data1 and Data2 also performed extremely well on Data2, achieving AUCs of 0.9780 and 0.98.23 respectively, indicating robustness to moderate imbalance.

Training - The best hyperparameters, **n_estimators = 200, max_depth = 20** (AUC = 0.9744), leveraged deeper trees to generalize across classes effectively. The worst configuration, **n_estimators = 100, max_depth = 10** (AUC = 0.9607), likely lacked sufficient complexity to handle class distributions

effectively.

c) Data3 (50:50) - Figure 9: Testing - The RFC model trained on Data3 achieved a test AUC of 0.9922. Interestingly, the model trained on Data2 outperformed training on Data3, achieving an AUC of 0.9964, mimicking the pattern seen for the kNN model. This in turn underscores the RFC model's capacity to handle class imbalances.

Training - The optimal hyperparameters, **n_estimators = 200, max_depth = 20** (AUC = 0.9907), effectively utilized deeper trees to capture intricate patterns. Conversely, **n_estimators = 100, max_depth = 10** (AUC = 0.9786) underperformed, potentially due to insufficient tree depth and estimators.

D. Model Comparison

TABLE IV: Model Training and Testing Time Comparison

Model	Runtime (seconds)
Logistic Regression	606.00
k-Nearest Neighbors	50.00
Random Forest Classifier	458.51

The kNN (Figures 4-6) and RFC models (Figures 7-9) exhibit steep ROC curves. This is due to their non-parametric nature, with kNN relying on local neighborhood structures and RFC leveraging ensemble decision trees to capture complex global interactions. In contrast, the LR model (Figures 1-3) shows more rounded ROC curves, reflecting its simpler linear assumptions, which make it less adaptable to non-linear or complex data distributions.

Models trained on Data1 nearly always under-performed compared to models trained on Data2 and Data3. This discrepancy highlights the challenges posed by extreme class imbalance in Data1, where a 90:10 imbalance leads to overfitting on the majority class.

Interestingly, non-linear models trained on Data2 and Data3 only had very minor differences (at most 0.0248) in their AUC metrics when tested on differently imbalanced data. This suggests Data2 allows for generalization across different class imbalances just as well the balanced data. This is important for real-life applications of these models as real-life data is most likely to have some class imbalance, meaning we can be more confident in these models' performance outside of experimental situation like this project.

As shown in Table IV, kNN completes tasks significantly faster with a runtime of 50.00 seconds, compared to RFC's 458.51 seconds and LR's 606.00 seconds. Despite its speed, kNN has slightly lower accuracy than RFC, which achieves better results due to its ensemble structure, making it particularly effective on imbalanced datasets.

For general-purpose tasks, kNN's efficiency, with its significantly shorter runtime of 50.00 seconds (Table IV), makes it a strong choice. However, since the dataset used in this study is a medical dataset, accuracy becomes the most critical evaluation criterion. In such contexts, the RFC model, with its higher accuracy achieved through its ensemble learning mechanism, would be the preferred choice. Although its runtime is longer (458.51 seconds), the ability to capture complex patterns in imbalanced datasets makes RFC more suitable for medical applications, where precision and reliability are paramount.

REFERENCES

[1] Shaza M. Abd Elrahman and Ajith Abraham, "A Review of Class Imbalance Problem", *JNIC*, vol. 1, p. 9, Oct. 2013.

[2] M. Agarwal, "Patient Survival Prediction," *Kaggle*, Dataset, 2021. [Online]. Available: https://doi.org/10.34740/KAGGLE/DSV/2972359

[3] D. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. New York, Etc.: John Wiley and Sons, Cop, 2013.

[4] X. Zou, Y. Hu, Z. Tian and K. Shen, "Logistic Regression Model Optimization and Case Analysis," *2019 IEEE 7th International Conference on Computer Science and Network Technology* (ICCSNT), Dalian, China, 2019, pp. 135-139, doi: 10.1109/ICCSNT47585.2019.8962457.

[5] Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling," in IEEE Access, vol. 8, pp. 4806-4813, 2020, doi: 10.1109/ACCESS.2019.2962617.

[6] L. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009, doi: https://doi.org/10.4249/scholarpedia.1883.

[7] K. Chomboon, P. Chujai, P. Teerarassamee, K. Kerdprasop, and N. Kerdprasop, "An empirical study of distance metrics for k-nearest neighbor algorithm." *Proceedings of the 3rd international conference on industrial application engineering*. Vol. 2. 2015.

[8] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," ISPRS *Journal of Photogrammetry and Remote Sensing*, vol. 114, no. 114, pp. 24–31, Apr. 2016, doi: https://doi.org/10.1016/j.isprsjprs.2016.01.011.

[9] L. Jiang, B. Zhang, Q. Ni, X. Sun and P. Dong, "Prediction of SNP Sequences via Gini Impurity Based Gradient Boosting Method," in IEEE Access, vol. 7, pp. 12647-12657, 2019, doi: 10.1109/ACCESS.2019.2893269.

[10] "ML — Handling Missing Values," *GeeksforGeeks*, 2024. [Online]. Available: https://www.geeksforgeeks.org/ml-handling-missing-values/

[11] "Stratified Sampling in Pandas," *GeeksforGeeks*, 2021. [Online]. Available: https://www.geeksforgeeks.org/stratified-sampling-in-pandas/

[12] W. McGinnis, "Binary," *Category Encoders 2.6.4 documentation*, 2022. [Online]. Available: https://contrib.scikit-learn.org/category_encoders/binary.html#category_encoders.binary.BinaryEncoder

[13] "Feature Encoding Techniques - Machine Learning," *GeeksforGeeks*, 2022. [Online]. https://www.geeksforgeeks.org/feature-encoding-techniques-machine-learning/.

[14] "SMOTENC," *imbalanced-learn documentation version 0.12.4*, 2024. [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTENC.html

[15] "SMOTE for Imbalanced Classification with Python," *GeeksforGeeks*, 2024. [Online]. Available: https://www.geeksforgeeks.org/smote-for-imbalanced-classification-with-python/#smotenc-nominal-continuous

[16] J. Brownlee, "SMOTE for Imbalanced Classification with Python," *Machine Learning Mastery*, 2021. [Online]. Available: https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

[17] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern recognition* 30.7, pp. 1145-1159, 1997.