CSC 2515 Project: Heart Stroke Prediction

Bhagyashree Keswani, Dhavalkumar Patel, Xiule Fan, Nigel Petersen
University of Toronto

Abstract

With healthcare as a whole being an integral part of our society, the ability to accurately predict the prevalence of disease has become evermore important. In this work, we focused on the prediction of heart stroke, which is one of the most common causes of death in the world. Presented with the usual difficulties that come with disease prediction datasets, we explored several approaches to deal with the imbalanced label distribution in these datasets, incorporating a variety of standard machine-learning models. We found that resampling to increase balance during training significantly improved model performance. Unlike previous work, which balanced the distribution in the test set, we verified our model's performance with an imbalanced but more realistic test set.

1 Problem Statement and Objective

As the second most common cause of death worldwide [5], a stroke can affect every individual's health significantly. The capability of predicting whether or not an individual may have a stroke allows one to seek the necessary treatment earlier to lower their risk, and healthcare professionals can advise their patients on the possibility of a stroke sooner, if they are given this capability. In this work, we are interested in exploring various machine learning techniques to predict whether an individual will have a stroke given the dataset [1]. By comparing the performance of these different techniques, we can identify the most reliable solution to provide stroke prediction results to the public and healthcare professionals.

Compared to previous work, our algorithms were evaluated in a more realistic setting with imbalanced data distribution. Furthermore, we provided discussions on the proper metrics to use for classification problems with imbalanced data.

2 Summary of Prior Work

2.1 Stroke Prediction

An early attempt in stroke prediction [11] adopted a regression model to map a set of risk factors, as identified by statistical analysis on clinical data [13], to the probability for an individual to experience a stroke at a particular time. Other machine learning approaches have been applied to stroke prediction. A support vector machine (SVM) was designed in [10] to classify if an individual will experience a stroke within a given time span. Jeena and Kumar [9] also explored the use of an SVM for stroke prediction with further experiments with respect to different kernel functions used by the SVM. Dev et al. [4] first studied how each input feature relates to the output labels of stroke occurrences. They then selected the most relevant features as inputs to several machine learning models for stroke prediction. Additionally, one of the works found that stacking ensemble had the best performance after evaluating various machine learning techniques [6].

2.2 Imbalanced Medical datasets

Datasets related to disease prediction are often imbalanced with insufficient representation for the positive class [17]. The same is true for our dataset, where the positive class for stroke prediction constitutes approximately 2% of the total observations. In such cases, there is a high chance that a classifier may show significant bias towards the majority class. There are various methods employed in different papers to circumvent this issue. [6] uses the synthetic minority over-sampling technique (SMOTE) [3] for handling the imbalance. [2] describes how one-class classification algorithms are relevant when generic classifiers are unable to learn a class boundary for minority class. Another interesting work on imbalanced binary classification in disease prediction is described in [18]. They develop three different autoencoder-based classifiers as anomaly detectors for the data. In such cases, the algorithm treats the majority class as inliers and the minority (positive in our case) as outliers. The results of the best autoencoder model were comparable to most of the classifiers and could handle data imbalance well.

2.3 Data Distribution Shift

A common phenomenon in supervised learning is a shift in the distribution of data between training and testing. This shift can lead to inaccuracies in machine learning models, and larger issues consequently. Data distribution shift can mean a shift in the distribution of either the labels, or of the covariates, known as label shift and covariate shift, respectively, with the latter being a more commonly occurring phenomenon. One possible method for measuring a potential shift in the distribtution of the data is using the Jensen-Shannon (JS) divergence, which is a measure of the similarity between two probability distributions [8]. With resampling a heavily imbalanced datset, some information may be lost in the process, possibly presenting performance issues further on, and the JS divergence will be a useful tool to measure a potential shift in the distribution in such a setting.

3 Data Preprocessing and Preliminary Analysis

Among the 43,400 data samples in [1], some of the samples contain features with missing values. If these training samples with missing features were considered in our design, our algorithms would need to be able to process inputs with different dimensions. To avoid addressing this issue, we removed all of the samples with missing values in the dataset, which leads to 29,072 valid samples in the remaining dataset.

The features presented in the dataset can be separated into two categories: numerical features and categorical features. For example, the `age` feature is a numerical feature and it describes the exact age of an individual. The `gender` feature is a categorical feature and its values can be `male`, `female`, or `other`. To convert the values of the categorical features to numerical values, we applied one-hot encoding to these features to obtain a vector representation. For instance, after applying one-hot encoding to the `gender` feature, this feature is converted to a 3-dimensional vector. Only one of the elements in this vector is 1, while the rest are 0. Different values of the vector represent different genders. After performing one-hot encoding, the features of each sample are described as a vector with a dimension of 18.

Upon inspecting the distributions of each of the continuous features in Figure 1, the distributions of `bmi` and `average glucose level` are approximately the same for both positive and negative values of `stroke`, being unimodal and slightly skewed to the left, with `avg-glucose-level` approximately bimodal for positive observations. For the `age` feature, the distribution among positive `stroke` observations is unimodal and highly right-skewed, while relatively centered for the negative `stroke` observations, with the former an indicator that a higher age increases the chances of a stroke, as expected.

For the categorical features, observing pie plots of their proportions among both values of the response, the proportions are approximately equal for both values of the response, with approximately 30% of stroke-positive observations having positive values of `hypertension` and `heart disease`, suggesting both having a moderate influence on the response, as expected. Finally, in plotting a correlation heat map for all features, due to the extreme imbalance in the data set, none of the features have a moderate or strong correlation with the response, and there is no strong relationship between any pair of the features.

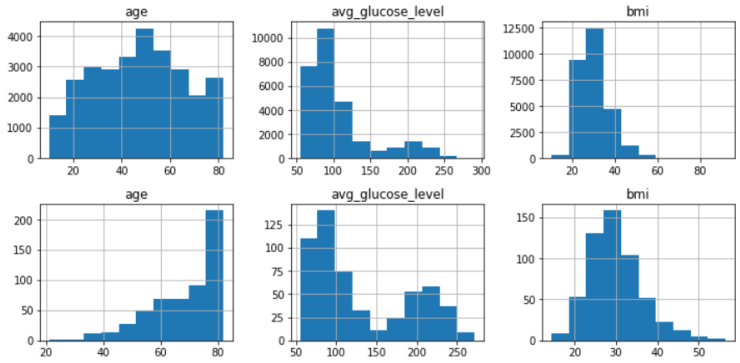


Figure 1: Sample distributions of continuous features for negative (top) and positive (bottom) values of the response.

In order to properly train, validate, and test our algorithms, we first saved 25% of the samples as the test set. The remaining 90% were separated such that 10% of them were used for validation while the rest were considered as the training set. After separating the dataset, there are 19,623, 2,181, and 7,268 samples in the training, validation, and test split, respectively.

Additionally, we performed two other feature engineering operations on the samples. The first operation is normalization of the numerical features. We computed the transformation needed to scale and translate the features in the training set such that they are all between 0 and 1. The same transformation with the parameters based on the training set was then applied to the validation and test sets. We also studied the correlation between different features in the training set. From this study, the features representing `male`, `self-employed`, and `never-smoked` were found to be highly correlated with the rest of the features, which implies that the first three features may carry redundant information. Therefore, these highly correlated features were removed from all of the samples. Consequently, the features of each sample are described by a vector with a dimension of 15.

Lastly, a model trained on the highly imbalanced dataset may have difficulty in making positive predictions. Furthermore, training on the complete training set with 19,623 samples is computationally demanding given the available computational resources. To remedy these problems, we first selected 8000 negative samples from the training set and then duplicated (oversampled) [14] the positive ones in the training set to obtain 2400 positive samples. The resampled positive and negative samples formed the new training set. Note that we did not alter the data distribution in the validation set or the test set, as opposed to [4]. We believe this strategy allows us to test our algorithms in more realistic scenarios where imbalanced distribution of stroke occurrences is more common. It is also worth mentioning that we explored undersampling [14] the negative samples and the more sophisticated SMOTE [3] re-sampling technique. Neither of them led to performance better than the proposed strategy.

4 Methodology

4.1 Machine Learning

We trained multiple machine learning models for heart stroke prediction. The brief descriptions of these models are provided below. The implementations of these models in the scikit-learn package [15] were used in this work. It is assumed each model described here is given a list of N training samples denoted by $\mathcal{D} = \left\{\left(x_{(1)}, t_{(1)}\right), \left(x_{(2)}, t_{(2)}\right), \ldots, \left(x_{(N)}, t_{(N)}\right)\right\}$, where $x_{(i)} \in \mathbb{R}^{15}$ denotes the input features and $t_{(i)} \in \{0, 1\}$ represents the label.

K-Nearest Neighbor The k-nearest neighbor classifier (KNN) [7] memorizes the training set $\mathcal{D}$. Given the features of a test sample $x \in \mathbb{R}^{15}$, the Euclidean distance between x and every training features $x_{(i)}$ is computed by $d_{(i)} = \left\|x_{(i)} - x\right\|_2$. The training samples corresponding to k smallest distances $d_{(i)}$ are retrieved from the training set. The prediction of x is determined by majority voting on the labels of these k retrieved samples.

Decision Tree The decision tree classifier follows a tree-like structure to solve a classification problem [7]. At each node of the tree, the training samples are split into two child nodes by comparing their features against a criterion. At the child node, the remaining samples are further separated according to another criterion. This process repeats until the tree reaches its preset maximum depth. At the terminal node of each tree branch, a class label is chosen by majority voting on the labels of the training samples at this node. The splitting criterion at each node is chosen to maximize the information gain of the split based on the training samples. During testing, the test sample x is placed into a particular terminal node by comparing it with the criteria selected previously. The class prediction of x is set to the label associated with this node.

Random Forest The random forest classifier is a type of bagging model [7]. Given the training dataset $\mathcal{D}$, we can generate B new datasets by sampling from $\mathcal{D}$ with replacement. A decision tree with a pre-determined maximum depth can be trained for each of the B datasets, which leads to an ensemble of B trees. This ensemble of trees is used to predict B labels for the test sample x. The final predicted label is obtained by majority voting on the B class predictions.

Support Vector Machine The original support vector machine (SVM) [7] seeks to separate the classes of the training samples in the feature space with a linear hyperplane by maximizing a margin. This margin is a region centered at the hyperplane, and no training features should fall within this margin. Since not all classes have distinct boundary in the feature space, a regularization term is often used in the training process to allow some features to overlap with the margin. Additionally, nonlinear kernel functions (e.g., polynomial, radial basis, and sigmoid functions) are sometimes applied to the input features to achieve nonlinear classification.

Adaptive Boosting The Adaptive Boosting (AdaBoost) algorithm [7] sequentially trains a number of classifiers. In each iteration of the algorithm, the classifier G_m is trained to minimize the classification error of the training samples. Note that the contribution of each sample to the error is weighted by a sample weight such that more weights are allocated to samples that were misclassified in the previous iterations. The weighted classification error rate is also used to compute the classifier coefficient α_m for G_m such that a larger α_m is assigned to a more accurate G_m. Lastly, the sample weights are updated to focus on misclassified training samples in the next iteration. After performing a total of M iterations to obtain M classifiers, the weighted sum of $\sum_{m=1}^{M} \alpha_m G_m(x)$ is used to predict the class of a test sample x.

Bagging This technique combines the results of different machine learning algorithms by computing a weighted majority vote or by computing the mean of different models' predictions on 'm' number of training points[7]. The latter case gives the predictor as $h(x) = \frac{1}{m}\sum_{m=1}^{M} h_i(x)$. Bagging has no effect on bias as the expectation of the predictor is unchanged, but variance reduces as m increases. Variance reduction is more significant when the correlation between predictions is small. In our case, the number of data points is large as compared to the number of features. There is a possibility of high variance in the dataset making bagging more relevant to our problem. Strictly speaking, a random forest is a specific type of bagging model that utilizes decision trees. However, other algorithms can also be adopted for bagging.

Multi-layer Perceptron A multi-layer perceptron (MLP) is a feed-forward neural network that contains fully-connected layers consisting of multiple hidden units(neurons) [7]. The input is followed by hidden layers where each hidden layer performs a computation given by $f(x) = \phi(Wx + b)$. The hidden layers are activated with activation functions like `sigmoid`, `tanh`, `ReLU`, and `LeakyReLU`. Backpropagation is employed to train the models. With a larger dataset, neural networks can capture the train information better due to their non-linear nature but require heavy tuning of hyperparameters. The number of hidden layers and neurons have been tuned for our problem to observe model performance with increasing complexity.

4.2 Training Procedure

We trained different configurations of the above models by varying a number of hyperparametes on the training set. For example, we varied the number of neighbors k for the KNN classifiers, the maximum depth for the decision trees, number of trees in the random forests, etc. Additionally, the

class weights were set to balanced whenever it was possible in order to better address our imbalanced dataset. After training different configurations, the hyperparameters leading to the best performance in terms of F1-score or accuracy score for each model type on the validation set were selected. Performance of the models based on these selected hyperparameters was verfied on the test set.

The best model configurations according to this training process are a KNN classifier with $k = 191$ neighbors, a decision tree with a maximum depth of 10, a random forest with $B = 160$ trees, an SVM classifier with a polynomial kernel and a regularization constant of 1, an AdaBoost model with $M = 1$ estimator and a learning rate of 0.01, and an MLP with two hidden layers of 32 neurons. Note that the base estimator selected for our AdaBoost model was a random forest since it provided better performance than a more commonly used decision tree did. Additionally, we also trained a bagging model composed of 10 random forests.

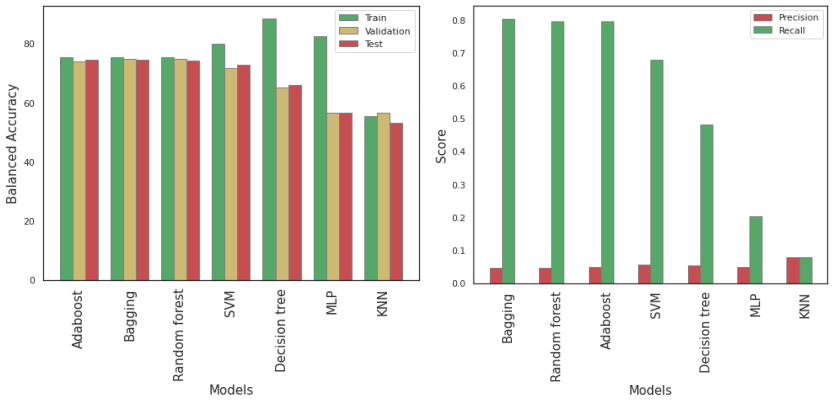
5 Results

During this study, we encountered one of the critical bottlenecks to developing robust machine learning models; a severely imbalanced dataset. Datasets related to disease prediction are often imbalanced with insufficient representation for the positive class. Three of the most popular methods to account for this imbalance were explored: re-sampling data, ensemble models, and balanced class weight. Here we summarize the performance of six machine learning models, and the applicability of multi-layer perceptron and autoencoder models.

Before implementing various techniques to account for imbalance, the entire machine learning pipeline was implemented without resampling and balanced class weight; the models performed extremely poorly with such approaches. Resampling and adjusting class weights are integral to classification problems with severe imbalances. The results of the test set with models trained using a 3:10 ratio of positive to negative class are reported in Table 1. The performance was evaluated using four metrics: `balanced accuracy`, `precision`, `recall`, and `roc-auc score`. The test set comprises 7131 negative-class (non-stroke) samples and 137 positive-class (stroke) samples, leading to a ratio of 19:1000. Since the dataset is imbalanced, focusing solely on the accuracy score would not be ideal as it does not reflect the actual performance of the model; For example, if the total number of negative samples is 90 while the total number of positive samples is 10. Even if the model misclassifies all the positive samples, the model's accuracy would still be 90% which is misleading. Instead, balanced accuracy was considered, which computes the mean of the specificity and sensitivity metrics.

It was noted that most of the models' performances on the train and the validation sets were quite comparable. Figure 2a presents the balanced accuracy for each data split. Comparing the validation scores (yellow), it can be highlighted that the ensemble methods had superior performance. The SVM, MLP, and Decision tree models had better training scores (green), but a drop in performance was observed for the validation set, leading to overfitting. A similar trend was observed for the test set as well. Among all the models, KNN fared poorly; this aligns with our expectations as the KNN algorithm assigns classes based on the closest point's class in the training set. For an imbalanced dataset, the performance of the KNN model degrades for the minority class. Bagging and Boosting models performed better as they incorporated predictions from several independent models.

It is also essential to examine the precision and recall scores to evaluate the performance. A high precision score means fewer false positives, whereas a high recall score refers to fewer false negatives. Since the positive (`stroke`) class is extremely important in this context, as a misclassification would not be ideal, an adequate focus was given to the recall score. Figure 2b represents the precision and recall scores of the test set. Ensemble models had the highest recall scores of close to 80%, while the precision scores were unsatisfactory ($\approx 5\%$). A similar trend was observed for the validation set. The recall scores were also similar for the training set; however, the precision scores were higher than those observed on the validation and test set which led us to further investigate the reason behind this strange behavior, and an interesting point was laid out: when the positive class is critical in a domain where imbalance exists, the precision score is not an ideal evaluation metric. To explain this statement, let's consider the following example where the negative and positive samples are 1200 and 100, respectively. The precision score is given by the ratio $\frac{TP}{TP+FP}$, where TP and FP are the number of true, and false positives, respectively. If the total number of false positives is more than or close to true positives, then the precision score would drop significantly, which can easily happen if



(a) Balanced Accuracy (b) Precision and Recall for Test set

Figure 2: Evaluation Results

the positive samples are rare. Instead, balanced accuracy that considers specificity and sensitivity (i.e., recall score) should be analyzed for medical datasets.

The Table [1] also includes the performance of similar machine learning models from the paper [16]. Except for the random forest and adaboost model, the authors reported better accuracy, recall, and precision scores. Precision scores were significantly higher than our models. To understand this difference in performance, their data preparation methodology was studied in-depth; The authors considered a subset of the data using an undersampling technique where the negative class was severely undersampled to match the positive class, which had 249 samples. 20% of the training set was used as a test set, leaving just 398 samples in the training set. This is an inappropriate method as it alters the distribution significantly. The model trained on such a sample would perform poorly when deployed in a natural setting as the distribution it would be exposed to would be drastically different from the training distribution. Overall, the models had close recall performance when trained and tested on different splits of the same dataset. However, precision scores were drastically different which is due to the balance in their training and testing set. This observation further bolsters our argument that precision score is not an efficient evaluation metric in an imbalanced classification where the positive class is the priority as they are heavily dependent on the ratio of positive and negative samples.

	Balanced Accuracy		**Precision**		**Recall**	
	Proposed	[16]	Proposed	[16]	Proposed	[16]
KNN	53.11	80	0.08	0.77	0.08	0.84
Decision Tree	66.09	66	0.06	0.78	0.48	0.78
Random Forest	74.33	73	0.05	0.72	0.79	0.74
SVM	72.89	80	0.06	0.79	0.67	0.84
Adaboost	74.48	73	0.05	0.72	0.79	0.78
Bagging	74.47	-	0.05	-	0.80	-
MLP	56.46	-	0.05	-	0.20	-

Table 1: Summary of Test-set classification results

The issues related to an imbalanced dataset will persist in the medical domain. Also, the dataset is a sample of the entire population and doesn't represent the entire spectrum of possible features for the positive class. Therefore, supervised machine learning models might not be applicable in all cases. Could a model trained with the just majority class serve better in this case? To answer this question, we explored a neural network-based autoencoder.

The Autoencoder was trained in an unsupervised setting using just the negative samples to reduce the reconstruction errors. It can be highlighted from the box plots in Figure 3 that the reconstruction errors for the non-stroke samples are lower when compared with the stroke class. In addition, the

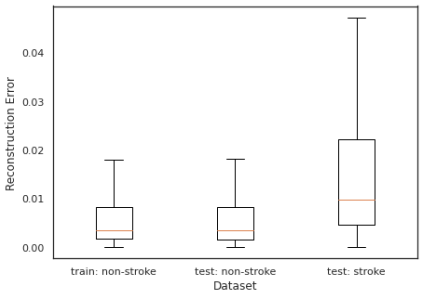
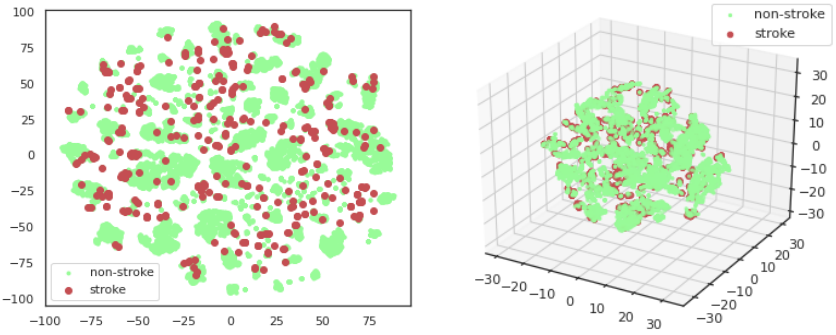


Figure 3: Reconstruction Error

variance of the errors observed in the latter case is more significant. This aligns with our expectation as the model was trained using the non-stroke class and has learned its representation well. However, when comparing the box plots, it can be pointed out that there is an overlap in errors between both classes; thus, with this method, it is challenging to determine a threshold for effective classification, and, extensive hyperparameters tuning is required to yield a robust classification model.

Another factor affecting the models' performance is the samples' separability. To visualize the high-dimensional features, t-distributed Stochastic Neighbour Embedding (t-SNE) [19] plots were used to project the training set features to a 2-dimensional space (Figure 4a) and 3-dimensional space (Figure 4b). From both of these figures, it is evident that the data points cannot be clustered effectively into two separate classes. This makes it difficult for the models to classify effectively.



(a) 2 components (b) 3 components

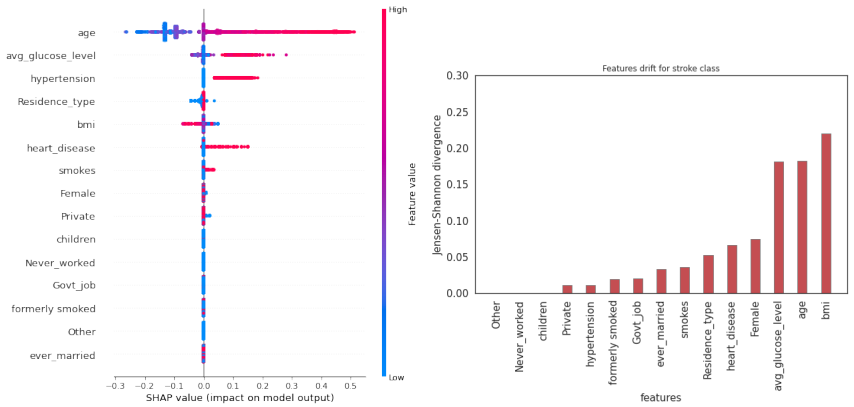
Figure 4: t-SNE visualisation on the train set

6 Model Explainability

This section aims to develop a discussion centered around the interpretations of the predictions. Out of all the features in the dataset, only a few contribute to the model's predictions. Figure 5a shows a SHAP (SHapley Additive exPlanations) [12] summary plot to explain one of the best-performing model's predictions (Bagging Classifier). The high values of the following features: `age`, `heart-disease`, `avg-glucose-level`, and `hypertension` have greater contribution toward the prediction of the positive (`stroke`) class. This inference aligns with the general belief that elder people with a history of hypertension, heart disease, and higher glucose level have a higher probability of experiencing heart stroke. With this interpretation, it can be stated that the models have learned to identify the stroke class.

Furthermore, the divergence of features in train and test data for both classes was determined. Jensen-Shannon(JS) Divergence concept was used to quantify the dissimilarity in the distributions. This metric is more valuable than Kullback-Leibler(KL)-divergence as it is a symmetric, normalized version of it; It can be calculated by the following equation for two distributions p and q [8]:

$$\text{JS}(p\|q) = \frac{\text{KL}(q\|m)}{2} + \frac{\text{KL}(q\|m)}{2} \qquad \text{where } m = \frac{p+q}{2}.$$



(a) SHAP Feature Impact Plot (b) JS Distance for Stroke class

Figure 5: Model Explainability

The performance of machine learning models degrades with shifts in feature distributions. Figure 5b represents the JS distance(i.e; square root of JS-Divergence) of each feature between the train and test set for the stroke class. It can be seen from this plot that three features had a moderate drift of 0.18: `age`, `heart-disease`, `avg-glucose-level`; from the figure 5a, it was noted that these features had higher contributions in the prediction for the stroke class. Stroke class is critical, and to maintain a similar performance as observed during the development, this drift should be monitored as a significant drift in the features would result in erroneous predictions.

7 Conclusion

In this work, we applied multiple machine learning algorithms on the task of heart stroke prediction using a publicly available dataset. The imbalanced class distribution in the dataset made the classification problem more difficult. Unlike previous work that modified the distribution in the test set, we evaluated our models with a test set based on the original imbalanced distribution, which is closer to a realistic scenario. In spite of the difficulty imposed by the imbalanced data, our models still achieved reasonable performance, especially with the help of data resampling and balanced weight techniques in the training stage. Moreover, our discussion and analysis on evaluation metrics revealed that some commonly used metrics (e.g., precision, accuracy) for classification problems may not be useful for imbalanced data distribution where the positive class is the priority. Instead, balanced accuracy based on specificity and sensitivity should be preferred. Furthermore, monitoring the data distribution for any drift during deployment is essential to the model's performance. To monitor the drift, the applicability of JS Divergence was explored to compare the train and the test distributions.

8 Contributions

The entire group was involved in problem formulation, literature review, and report writing. Dataset preparation and data visualization were performed by Xiule and Nigel. The machine learning algorithms were implemented and tuned by Xiule, Bhagyashree, and Dhaval. Performance evaluation was performed by Nigel and Dhaval. Discussion and analysis on the results were conducted by Dhaval and Bhagyashree.

References

[1] L. Amal. Heart stroke. https://www.kaggle.com/datasets/lirilkumaramal/heart-stroke, 2020. Accessed: Nov 13, 2022.

[2] J. Brownlee. One-class classification algorithms for imbalanced datasets. https://machinelearningmastery.com/one-class-classification-algorithms/, 2020. Accessed: Nov 12, 2022.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyera. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[4] S. Dev, H. Wang, C. S. Nwosu, N. Jain, B. Veeravalli, and D. John. A predictive analytics approach for stroke prediction using machine learning and neural networks. *Healthcare Analytics*, 2:100032, 2022.

[5] G. A. Donnan, M. Fisher, M. Macleod, and S. M. Davis. Stroke. *The Lancet*, 371(9624): 1612–1623, 2008.

[6] E. Dritsas and M. Trigka. Stroke risk prediction with machine learning techniques. *Sensors*, 22 (13), 2022.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Element of Statistical Learning*. Springer, 2009.

[8] Y. Huang. Kullback-leibler (kl) divergence and jensen-shannon divergence. https://yongchaohuang.github.io/2020-07-08-kl-divergence/, 2020. Accessed: Dec 12, 2022.

[9] R. S. Jeena and S. Kumar. Stroke prediction using SVM. In *Proc. International Conference on Control, Instrumentation, Communication and Computational Technologies*, pages 600–602, 2016.

[10] A. Khosla, Y. Cao, C. C.-Y. Lin, H.-K. Chiu, J. Hu, and H. Lee. An integrated machine learning approach to stroke prediction. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 183–192, 2010.

[11] T. Lumley, R. A. Kronmal, M. Cushman, T. A. Manolio, and S. Goldstein. A stroke prediction score in the elderly. *Journal of Clinical Epidemiology*, 55(2):129–136, 2002.

[12] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proc. International Conference on Neural Information Processing Systems*, page 4768–4777, 2017.

[13] T. A. Manolio, R. A. Kronmal, G. L. Burke, D. H. O'Leary, T. R. Price, and CHS Collaborative Research Group. Short-term predictors of incident stroke in older adults. *Stroke*, 27(9): 1479–1486, 1996.

[14] R. Mohammed, J. Rawashdeh, and M. Abdullah. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *International Conference on Information and Communication Systems*, pages 243–248, 2020.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] G. Sailasya and G. L. A. Kumari. Analyzing the performance of stroke prediction using ml classification algorithms. *International Journal of Advanced Computer Science and Applications*, 12(6), 2021.

[17] N. Seliya, A. A. Zadeh, and T. M. Khoshgoftaar. A literature review on one-class classification and its potential applications in big data. *Journal of Big Data*, 8(1):122, 2021.

[18] V.-I. Tomescu, G. Czibula, and Ştefan Niţică. A study on using deep autoencoders for imbalanced binary classification. In *Proc. International Conference Knowledge-Based and Intelligent Information & Engineering Systems*, pages 119–128, 2021.

[19] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.