Stroke Prediction Based on Machine Learning Models

Tianyue Huang, Ziming Song, Hongdao Meng
th3113, zs2815, hm3424
Dec., 2024

Abstract

Stroke is one of the leading causes of death globally, responsible for approximately 11% of total deaths. Early identification of individuals at high risk is essential to enable preventive measures. Our project aims to use Machine Learning methods that we learned in class and develop predictive models that accurately forecast stroke risk based on key clinical and demographic factors, including age, gender, and health conditions. Using a variety of machine learning models, from traditional techniques such as logistic regression, support vector machines, decision trees, and Naive Bayes to advanced architectures like Transformers, we explore different strategies for optimizing stroke prediction accuracy. Our work evaluates the models across several metrics, including recall, average precision, F1 score, accuracy, macro averages, and weighted averages, and highlights the potential of hybrid and deep learning models for risk assessment in healthcare.

1. Introduction

Stroke ranks as the second leading cause of death worldwide and remains a critical public health issue. According to the World Health Organization, stroke accounts for about 11% of all global deaths, underscoring the importance of timely prediction and intervention of stroke. Stroke prediction models can identify individuals at high risk and facilitate early interventions, which makes them play a significant role in healthcare. Traditional clinical methods often lack predictive precision achievable compared with machine learning methods, where complex relationships between risk factors can be more accurately identified.

Our project investigates the effectiveness of different machine learning models for stroke prediction. We use both traditional algorithms—such as logistic regression, support vector machines, decision trees, and Naive Bayes—and advanced deep learning architectures, including Transformer models, to classify stroke risk based on patient data. This study aims to determine which models perform best on stroke prediction, considering the inherent class imbalance in stroke datasets. Additionally, we assess the predictive capabilities of these models using comprehensive evaluation metrics, offering insights into their performance and applicability in stroke risk prediction.

2. Related Work

Stroke prediction represents a pivotal challenge in contemporary healthcare analytics, where machine learning and deep learning have shown unprecedented potential for risk prediction. By leveraging advanced computational techniques, researchers aim to design predictive models that can identify stroke risks with greater precision and earlier detection than traditional clinical approaches. This review part examines existing machine learning methodologies for stroke prediction, analyzing their strengths, limitations, and promising research trajectories.

2.1 Traditional Machine Learning Approaches

Traditional machine learning models have been widely used for stroke prediction due to their interpretability and computational efficiency. For instance, logistic regression has been a popular baseline method for stroke prediction, offering simplicity and explainability [1]. However, its linear nature limits its ability to capture complex interactions between medical features, such as diabetes, hypertension and cholesterol levels.

Similarly, researchers applied random forest models to predict strokes using patient health data, achieving an accuracy of 98.98% [2]. The experiment demonstrated the potential of ensemble learning in handling structured medical data but lacked consideration for imaging data, which is critical in stroke diagnosis.

SVM has also been explored for stroke prediction comparing different kernel methods to capture non-linear relationships [3]. Despite their mathematical rigor, SVM is sensitive to hyperparameter selection and probably struggles with high-dimensional datasets or class imbalances which are common in medical records.

2.2 Deep Learning for Stroke Prediction

Deep learning models, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have shown promising results in stroke prediction, especially when incorporating imaging data. A deep learning model leveraging MRI images successfully predicted ischemic stroke lesion size and location with an AUC of 0.92 [4]. This research highlighted CNNs' ability to process high-resolution image data, providing insights into lesion characteristics that traditional models can hardly achieve. Furthermore, RNNs have been applied on sequential health data, such as patient vitals and medication history, improving temporal pattern recognition in stroke prediction tasks [5].

Transformer-based architectures, initially designed for natural language processing, are increasingly being applied in medical area due to their good performance on modeling long-range dependencies. For example, a Transformer model was used to predict patient mortality risk based on electronic health records, outperforming traditional methods in both accuracy and recall [6], which demonstrates the versatility of attention mechanisms in handling heterogeneous medical data.

2.3 Hybrid Models

Hybrid approach combines deep learning with traditional machine learning, offering a balanced interpretability as well as predictive power. For example, researchers explored integrating CNN features extracted from image data with random forests models to predict stroke risk, achieving superior performance compared to standalone models [7]. Hybrid model not only enhance prediction accuracy but also allow multimodal data incorporation, such as demographics, lab results, and image features.

2.4 Challenges and Future Directions

While significant progress has been made, several challenges remain in stroke prediction:

1. **Class Imbalance**:

Stroke datasets often have an imbalanced distribution, with far fewer positive cases. Methods such as oversampling, cost-sensitive loss functions, and synthetic data generation (e.g., SMOTE) can address this issue [8].

2. **Multimodal Data Integration**:

   Incorporating diverse data types, such as imaging, genomics, and clinical records, poses challenges in data fusion but offers significant potential for improving prediction accuracy [9].

3. **Cross-Dataset Generalization**:

   Many models perform well on single datasets but fail to generalize across different populations or institutions. Techniques like domain adaptation and federated learning may help address this limitation [10].

3. ML Models

Our project explores comprehensive strategies for stroke risk prediction using traditional machine learning techniques as well as innovative Transformer model. We choose logistic regression model as baseline, and implement the following models respectively.

3.1 logistic regression (baseline)

A classifier is a binary classification model that predicts the probability for an outcome belonging to one of two categories. It minimizes the logistic loss and it can be trained via MLE(Maximum Likelihood) or MAP(maximum a posterior) estimation.
MLE:

$$arg\ max_{\theta} \sum^{m}_{i=1} logP(y_i | x_i, w)$$

MAP:

$$arg\ max_{\theta} \sum^{n}_{i=1} logP(y_i | x_i, w) + logP(w)$$

In this project, we aimed to evaluate the performance of logistic regression models with L1 and L2 regularization. The hyperparameter C was optimized using cross-validation to ensure effective regularization.

3.2 Support Vector Machines (SVM)

SVM is a supervised learning model often used for classification tasks. It works by finding the hyperplane that best separates the data into classes. The model aims to maximize the margin between the closest data points (support vectors) from each class while minimizing the empirical loss, which can lead to better generalization on unseen data.

The empirical loss in SVM is commonly represented by the hinge loss function, which penalizes misclassified points and those within the margin. The hinge loss function is defined as:

$$L(y_i, f(x_i)) = \max(0, 1 - y_i \cdot f(x_i))$$

where y_i is the true label of the data point x_i, and $f(x_i)$ is the predicted value. Minimizing this loss function ensures that the model not only classifies the training data correctly but also maximizes the margin between classes.

In our project, we use the sklearn.svm package to train SVM models. We apply both linear and non-linear kernels (poly, RBF, sigmoid) to train and test on the same dataset separately.

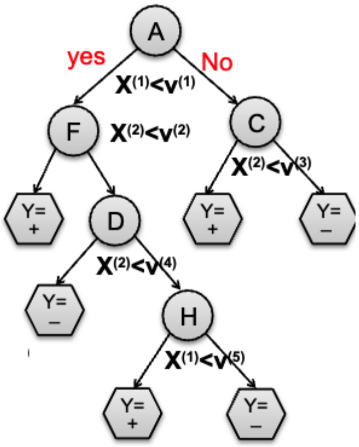
Hyperparameter Configuration:

- **Kernel:** We choose four kernel types and apply them separately:
  - **Linear:** Performs well for linearly separable data.
  - **Polynomial:** Performs well if relationships between classes have polynomial-like separations.
  - **RBF (Gaussian):** Performs well on complex, nonlinear data; useful if no clear linear boundary exists.
  - **Sigmoid:** Useful in cases where a nonlinear relationship between classes is suspected but complex boundary shapes like those provided by the RBF kernel are not needed.
- **Gamma (γ):** Relevant for RBF and polynomial kernels, gamma defines how far the influence of a single training example reaches. A low gamma means a larger influence, resulting in smoother decision boundaries; a high gamma creates more complex, localized decision boundaries. We use the default value in this experiment.

By experimenting with different kernels and hyperparameters, and by minimizing the empirical loss through the hinge loss function, we aim to find the SVM configuration that provides the best classification performance on our dataset.

3.3 Decision tree

A model uses tree-like structure to make decisions based on input features. Each internal node represents a decision based on a feature, while the leaf nodes represent the predicted outcomes. The model splits data recursively by selecting features that best divide the data according to metrics like Gini or entropy.



In our project, Decision tree classifiers were trained using the \`sklearn.tree.DecisionTreeClassifier\` class. Two impurity measures, Gini Impurity and Entropy, were evaluated to compare their effectiveness. The tree depths were varied from 1 to 6 levels to analyze the impact of model complexity on performance.

3.4 Naive Bayesian

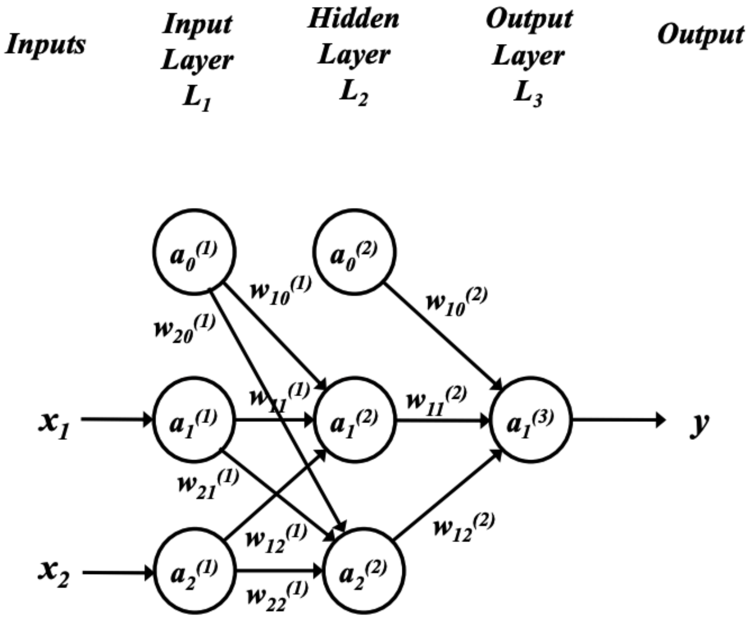
A classification algorithm that applies Bayes' theorem with the naive assumption that features are conditionally independent given the class.

$$C_{predicted} = arg\,max_C (P(C) \prod^{n}_{i=1} P(x_i|C))$$

In our project, Naive Bayesian are trained using package sklearn.naive_bayes. And we assume features are continuous and assume a normal (Gaussian) distribution, so we choose Gaussian Naive Bayes.

3.5 Neural Network (Transformer)

In this project, we utilize a neural network model inspired by the structure of the human brain. Neural networks consist of layers of interconnected nodes (neurons) that process data by learning patterns through weights and biases. Each layer transforms the input data using activation functions, enabling the network to model complex non-linear relationships.



3.5.1 Transformer Model

Specifically, we implement a Transformer model, inspired by the architecture introduced by Vaswani et al. (2017) in the seminal paper Attention Is All You Need [11]. Unlike traditional recurrent neural networks (RNNs) or long short-term memory networks (LSTMs), the Transformer architecture relies solely on self-attention mechanisms to capture dependencies across input features. This approach eliminates the sequential processing limitations of RNNs and LSTMs, enabling parallel computation and improved scalability.

Key Components of the Transformer Architecture:

Input Embedding

A linear embedding layer projects the input features into a high-dimensional space (256 dimensions). This step facilitates the model's ability to capture complex feature interactions and relationships. Positional encoding was omitted, as the input data in this binary classification task does not have a temporal or sequential structure.

Multi-Head Self-Attention

Multi-head self-attention is the core mechanism of the Transformer. Each attention head computes a weighted representation of the input by focusing on relevant parts of the data, as defined by the dot-product attention mechanism:

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Here, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ represent the query, key, and value matrices derived from the input. The scaling factor $\sqrt{d_k}$ prevents the attention scores from becoming too large. By using 8 attention heads, the Transformer can capture diverse feature interactions, enabling it to identify subtle patterns in the data.

Feedforward Networks

Each Transformer encoder layer contains a position-wise feedforward network (FFN) applied to each feature independently. The FFN consists of two linear transformations separated by a ReLU activation:

$$\text{FFN}(x)=\text{ReLU}(xW_1+b_1)W_2+b_2$$

This enhances the model's capacity to transform and learn higher-level representations.

Layer Normalization and Residual Connections

Layer normalization normalizes the output of each sublayer, improving gradient flow and training stability.Residual connections enable the flow of information across layers, preventing vanishing gradients in deep architectures [12].

Global Pooling and Output

The output embeddings of the final Transformer encoder layer are aggregated using global average pooling to produce a fixed-length representation. A fully connected layer maps the pooled representation to a single logit, which is then passed through a sigmoid activation for binary classification.

Hyperparameter Configuration

- **Embedding Dimension:** Set to 256 to provide a high-capacity feature space for learning complex interactions.
- **Transformer Layers:** Eight encoder layers were used to capture deep feature representations.
- **Attention Heads:** Eight attention heads were used to focus on multiple aspects of the input simultaneously.
- **Learning Rate:** Set to 0.0001, with a StepLR scheduler applied to decay the learning rate by a factor of 0.1 every 5 epochs.

- **Batch Size:** 32 samples per batch were used during training to balance computational efficiency and gradient stability.
- **Training Epochs:** The model was trained for 20 epochs.
- **Dropout:** A dropout rate of 0.1 was applied to prevent overfitting.

4. Experiment

4.1 Experiment Setting

4.1.1 Dataset

We choose Stroke Prediction Dataset with 11 clinical features for predicting stroke events (https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset)

Attribute Information

Attribute name	**Details**
id	unique identifier
gender	"Male", "Female" or "Other"
age	age of the patient
hypertension	0 if the patient doesn't have hypertension, 1 if the patient has hypertension
heart_disease	0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
ever_married	"No" or "Yes"
work_type	"children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
residence_type	"Rural" or "Urban"
avg_glucose_level	average glucose level in blood
bmi	body mass index
smoking_status	"formerly smoked", "never smoked", "smokes" or "Unknown"
stroke	1 if the patient had a stroke or 0 if not

4.1.2 Data processing

To ensure consistency across methods, all data underwent the following preprocessing steps:

- Normalization: Numerical features were standardized to zero mean and unit variance.
- Class Balancing: The RandomOverSampler was used to create a balanced dataset, ensuring equal representation of both classes. For the SVM and Naive Bayes models, we balance the dataset by sampling the majority and minority classes to ensure both classes have the same number of samples.
- Train-Test Split: The dataset was split into training (80%) and testing (20%) sets.

4.2 Accuracy/Error Measures

In our ML project, we utilize several widely-used evaluation metrics to comprehensively assess the performance of our models. These metrics include **Recall**, **Average Precision (AP)**, **F1 Score**, **Accuracy**, **Macro Average**, and **Weighted Average**, each of which measures the model's performance from different perspectives.

4.2.1 Recall

Recall measures the proportion of relevant labels that are correctly identified by the model. It focuses on minimizing false negatives, ensuring that most of the actual labels are captured. A higher recall value is better, because it means the model is successfully identifying more relevant labels. Nevertheless, high recall can sometimes come with a lower precision.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

4.2.2 Average Precision (AP)

AP evaluates the model’s precision and recall across all labels and measures the correctness of the ranking of labels. A higher AP indicates that the model can accurately recognize relevant labels in a multi-label context. Specifically, AP is helpful in understanding how well the model identifies positive instances when the data is imbalanced.

Given a list of precisions $P_1,P_2,\ldots,P_n$ at recall levels $R_1,R_2,\ldots,R_n$:

$$\text{AP} = \sum_n (R_n - R_{n-1}) \cdot P_n$$

Where:

- R_n and R_{n-1} are the recall levels at consecutive ranks
- P_n is the precision at rank n.

4.2.3 F1 Score

The F1 score is the harmonic mean of precision and recall, which provides a balance between the two metrics. It is particularly useful when we need to find an optimal trade-off between precision and recall. Higher F1 scores indicate a good balance and overall performance in both precision and recall, which is especially critical in imbalanced datasets where the minority class requires focused attention.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.2.4 Accuracy

Accuracy measures the proportion of correctly predicted instances among all instances in the dataset. While accuracy is an intuitive metric and provides a broad overview of model performance, it can be misleading in imbalanced datasets, as it may overrepresent the majority class's success.

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Instances}}$$

Where:

- Total Instances = TP + TN + FP + FN

4.2.5 Macro Average

Macro average computes the unweighted mean of precision, recall, and F1 scores across all classes. This metric treats all classes equally, regardless of their support (i.e., class size). It is especially useful when assessing the model’s performance on minority classes, as it ensures that their contribution is not overshadowed by the majority class.

For example, macro average F1 score is calculated as:

$$\text{Macro F1} = \frac{1}{C} \sum_{i=1}^{C} \text{F1}_i$$

Where:

- C is the number of classes,
- F1_i is the F1 score for each class i.

4.2.6 Weighted Average

Weighted average calculates the mean of precision, recall, and F1 scores across all classes, weighted by the number of true instances in each class. Unlike the macro average, this metric gives more importance to classes with larger support, making it suitable for evaluating overall performance while still accounting for class imbalances.

For example, weighted average F1 score is calculated as:

$$\text{Weighted F1} = \sum_{i=1}^{C} w_i \cdot \text{F1}_i$$

Where:

- C is the number of classes,
- w_i is the weight (proportion of instances) for class i,
- F1_i is the F1 score for each class i.

4.3 Results

We implement the five different models on the same dataset and methods we mentioned before. We compare the performance and list the result in the below table.

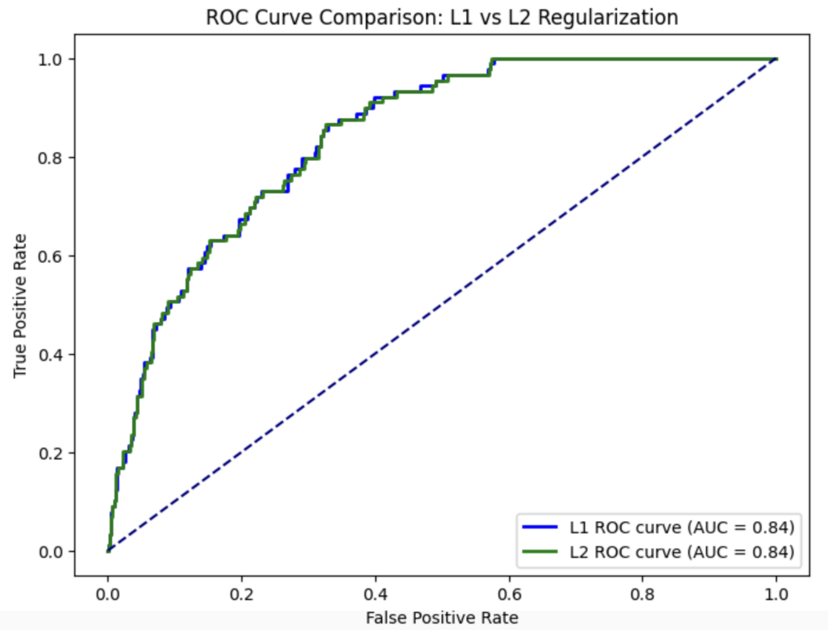
	logistic regression		**SVM**		**Decision tree**	**Naive Bayesian**	**Transformer**
Average Precision	L1	0.94	Linear	0.7487	0.9419	0.6056	0.9384
			Ploy	0.7410			
	L2	0.94	Rbf	0.7519			

				Sigmoid	0.6003			
Recall	Macro Avg	L1	0.50	Linear	0.73	0.57	0.62	0.71
				Ploy	0.74			
		L2	0.50	Rbf	0.73			
				Sigmoid	0.55			
	Weighted Avg	L1	0.94	Linear	0.73	0.90	0.62	0.77
				Ploy	0.74			
		L2	0.94	Rbf	0.73			
				Sigmoid	0.55			
F1 Score	Macro Avg	L1	0.49	Linear	0.73	0.57	0.57	0.54
				Ploy	0.74			
		L2	0.49	Rbf	0.72			
				Sigmoid	0.55			
	Weighted Avg	L1	0.91	Linear	0.73	0.90	0.57	0.83
				Ploy	0.74			
		L2	0.91	Rbf	0.72			
				Sigmoid	0.55			

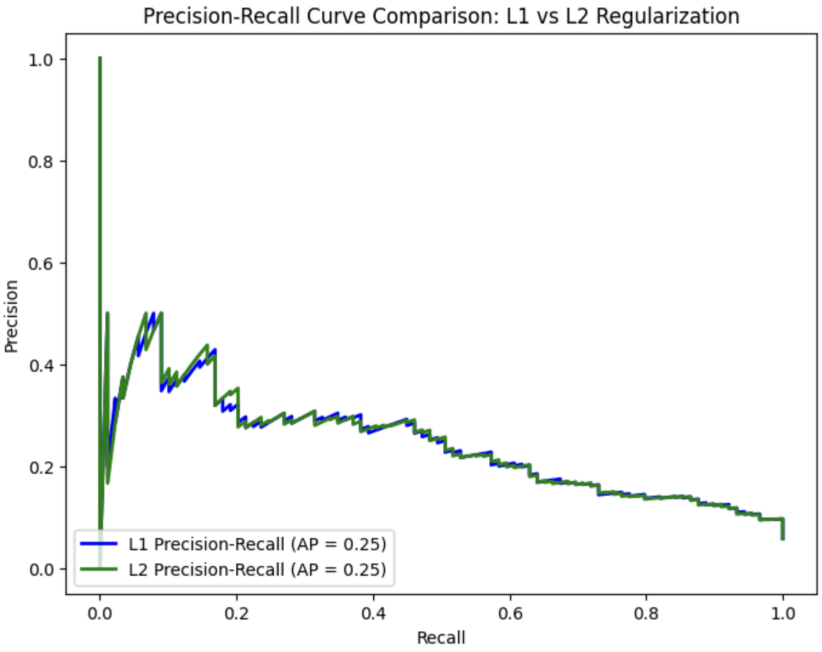
4.4 Analysis

4.4.1 logistic regression

Both L1 and L2 regularization achieved an accuracy score of 94.19% on the testing set.



The ROC curves for both models are nearly identical, with an Area Under the Curve (AUC) of 0.84, which suggests room for improvement, as the models are not achieving optimal class separability.



The average precision score of 0.25 for each. This low AP score. Combined with the PR curve's sharp decline, reinforces the observation that the models struggle to correctly predict the minority class. The low precision at higher recall levels suggests a trade-off where improving recall significantly reduces precision.

4.4.2 SVM

The SVM model is evaluated using four different kernels respectively: linear, polynomial (poly), radial basis function(rbf), and sigmoid. Among these, the poly and rbf kernels demonstrated the best overall performance, especially on the minority class (Class 1). The macro averages and weighted averages further prove the robust performance of the poly kernel, which maintains balanced metrics across both classes.

However, the sigmoid kernel struggled on this task with the lowest accuracy and poor F1 scores for both classes. The result shows that choice of kernel significantly impacts the SVM's ability to handle imbalanced datasets, with the poly kernel shown as the most effective for this task.

		Precision	**Recall**	**F1-Score**	**Support**
0	Linear	0.76	0.67	0.71	42
	Ploy	0.76	0.69	0.72	42
	Rbf	0.79	0.62	0.69	42
	Sigmoid	0.54	0.60	0.57	42
1	Linear	0.70	0.79	0.74	42
	Ploy	0.72	0.79	0.75	42
	Rbf	0.69	0.83	0.75	42
	Sigmoid	0.55	0.50	0.53	42
Accuracy	Linear			0.73	84
	Ploy			0.74	84
	Rbf			0.73	84
	Sigmoid			0.55	84
Macro avg	Linear	0.73	0.73	0.73	84
	Ploy	0.74	0.74	0.74	84
	Rbf	0.74	0.73	0.72	84
	Sigmoid	0.55	0.55	0.55	84
Weighted avg	Linear	0.73	0.73	0.73	84
	Ploy	0.74	0.74	0.74	84
	Rbf	0.74	0.73	0.72	84

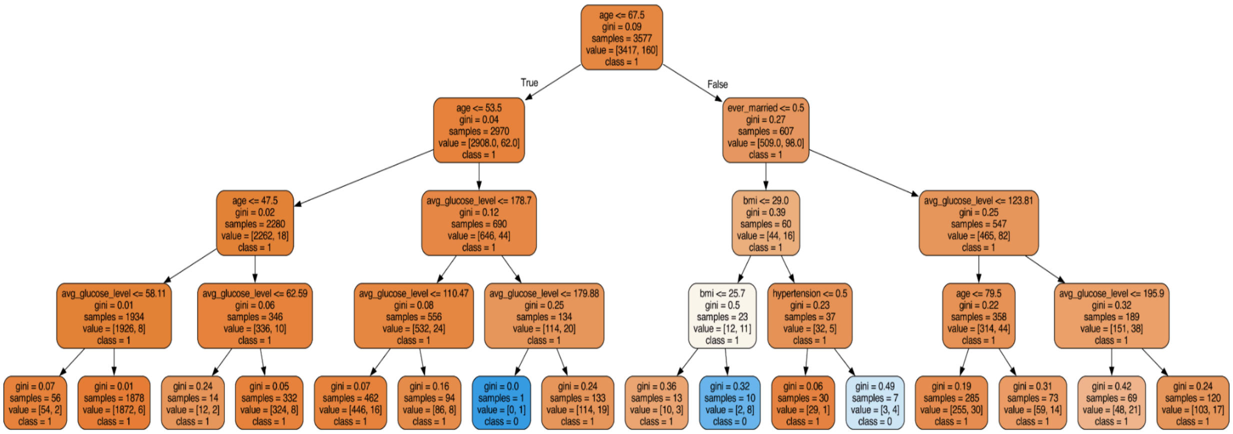
	Sigmoid	0.55	0.55	0.55	84

4.4.3 Decision tree

The model achieved an overall accuracy of 89.89%. The model performs well on the majority class, with high precision, recall and f1-score, which it still struggles with in the minority class.

	Precision	**Recall**	**F1-Score**	**Support**
0	0.95	0.94	0.95	1444
1	0.18	0.20	0.19	89
Accuracy			0.90	1533
Macro avg	0.56	0.57	0.57	1533
Weighted avg	0.91	0.90	0.90	1533

Tree depths were varied from 1 to 6 using both Gini Impurity and Entropy as impurity measures. The results suggest that a tree depth of 3 provides an optimal balance between performance and complexity. Increasing the tree depth beyond this point introduces a risk of overfitting, leading to a slight decline in accuracy.



4.4.4 Naive Bayesian

The Gaussian Naive Bayes model shows strong recall for class 1 (stroke cases) at **95%**, effectively identifying most stroke cases. However, its precision for class 1 is only **57%**, which indicates many false positives. For class 0 (non-stroke cases), the model achieves high precision (**86%**) but struggles with recall (**29%**), which leads to frequent misclassification of non-stroke cases as strokes. With an overall accuracy of **62%**, the model performs better on the minority class but struggles to balance precision and recall across both classes, as reflected in its low F1-scores for both classes.

	Precision	**Recall**	**F1-Score**	**Support**
0	0.86	0.29	0.43	42
1	0.57	0.95	0.71	42
Accuracy			0.62	84
Macro avg	0.71	0.62	0.57	84
Weighted avg	0.71	0.62	0.57	84

4.4.5 Transformer

The Transformer-based binary classifier achieved an accuracy of 93.84%, which indicates that the model can correctly classify the majority of samples in the dataset. Below is a detailed analysis of this result:

	Precision	**Recall**	**F1-Score**	**Support**
0	0.98	0.78	0.87	973
1	0.13	0.65	0.21	49
Accuracy			0.77	1022
Macro avg	0.55	0.71	0.54	1022
Weighted avg	0.94	0.77	0.83	1022

The Transformer-based binary classifier achieved a high accuracy of **93.84%**, which indicates strong overall performance. However, it struggled with class 1 (stroke cases), achieving a recall of **65%** but a low precision of **13%**, resulting in a poor F1-score of **0.21**. This suggests that while the model identifies many stroke cases, it also generates a high number of false positives. In contrast, the model performed exceptionally well on class 0 (non-stroke cases), with a precision of **98%** and an F1-score of **0.87**. These results highlight the model's bias toward the majority class, reflecting the impact of class imbalance. While the macro average F1-score of **0.54** suggests imbalanced performance across classes, further optimization and balancing techniques are required to improve minority class predictions.

5. Conclusion

5.1 Model Performance and Class Imbalance

Across all models, class imbalance significantly influenced performance, particularly for identifying stroke cases (the minority class). Logistic regression and SVM achieved competitive accuracy (94.19% and 93.84%, respectively), but both struggled with precision and recall for stroke cases. While Naive Bayesian models excelled in recall for stroke cases (95%), they suffered from low precision (57%), leading to frequent false positives. Similarly, decision trees and Transformers displayed strong performance on non-stroke cases but fell short in reliably identifying stroke cases, reflecting the overarching challenge of handling imbalanced datasets effectively.

5.2 Transformer Potential

The Transformer-based model demonstrated its ability to handle complex feature interactions with a high overall accuracy (93.84%) and strong performance for non-stroke cases (precision: 98%, F1-score: 0.87). However, it struggled with the minority class, achieving only 13% precision and an F1-score of 0.21 for stroke cases. This phenomenon emphasizes that while advanced models like Transformers can improve overall predictive power, they still need additional refinement—such as balancing techniques or domain-specific adjustments—to effectively handle predictions for minority classes.

5.3 Future Directions

To improve stroke prediction models, addressing class imbalance is critical. Techniques like oversampling, undersampling, and class-weighted loss functions should be prioritized. Additionally, simpler models like logistic regression remain reliable baselines, but advanced models such as Transformers offer promise if equipped with hyperparameter tuning and tailored feature engineering. Future efforts should focus on optimizing recall and precision for stroke cases, as early and accurate detection of strokes is crucial for implementing timely medical interventions and reducing mortality rates.

References

1. Hosseini, M. S., & Luo, J. (2017). *Stroke risk prediction using logistic regression and electronic health records*. Journal of Medical Systems, 41(10), 170.
2. Random Forest for Stroke Prediction (2020): https://jamanetwork.com/journals/jamanetworkopen/article-abstract/2762679
3. Zhang, Y., & Wallace, B. C. (2015). *A sensitivity analysis of SVMs in text classification*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(5), 1100-1111.
4. Deep Learning for Ischemic Stroke Lesion Prediction (2021): https://ieeexplore.ieee.org/document/10284685
5. Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzel, R. (2016). *Learning to diagnose with LSTM recurrent neural networks*. arXiv preprint arXiv:1511.03677.
6. Song, H., Rajan, D., Thiagarajan, J. J., & Spanias, A. (2018). *Attend and diagnose: Clinical time series analysis using attention models*. AAAI.
7. Suk, H. I., Lee, S. W., & Shen, D. (2017). *Deep ensemble learning of sparse regression models for brain disease diagnosis*. Medical Image Analysis, 37, 101-113.
8. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research, 16, 321-357.
9. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). *Recent advances in convolutional neural networks*. Pattern Recognition, 77, 354-377.
10. Li, W., Milletarì, F., Xu, D., & et al. (2019). *Privacy-preserving federated brain tumour segmentation*. Medical Image Analysis, 61, 101663.
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
12. Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.