Deep Stroke Detect: Optimizing Stroke Risk Prediction Using SMOTEENN and Transfer Learning

Sireesha Moturi[1], Sai Sarnya Tupakula[2], Nazeema Shaik[3]
Akhila Parella[4], Sandhya Diddi[5],
[1,2,3,4,5]*Department of CSE, Narasaraopeta Engineering College, Andhra Pradesh, India*
[1]sireeshamoturi@gmail.com, [2]tupakulasaranya123@gmail.com, [3]nazeemashaik852@gmail.com,
[4]akhilaparella@gmail.com, [5]diddisandhya120@gmail.com,

***Abstract*—Stroke remains a major global health concern, being one of the top causes of mortality and a leading contributor to long-term disability. This is the very reason why recognizing the signs of risk is critical to treatment—timing is everything. In this paper, we present a novel approach to assess the risk of stroke using deep learning. The system is applied to real-world healthcare datasets, which often exhibit a greater prevalence of non-stroke cases compared to stroke cases. To address the imbalance and enhance the data quality, we implemented several data preparation steps, SMOTEENN, and a cross-validated deep neural network model. We also applied transfer learning to mitigate the impact of scarce data on model performance. The model's accuracy was assessed using the common metrics of accuracy, precision, recall, F1-score, and ROC-AUC. The outcomes were quite encouraging, achieving 95.24% accuracy on average and 95.52% F1-score, confirming that the model is extremely accurate. Apart from the outcomes, this study attempts to solve significant real-world challenges such as the sparse nature of stroke cases, data sparsity, and selection of relevant health indicators. The findings indicate that deep learning approaches are very effective, provided there is thorough data preparation along with intelligent preprocessing.Feature selection can be an impactful method for predicting stroke risk— potentially helping doctors take action earlier and save lives.**

***Index Terms*—Stroke Prediction, Deep Learning, Transfer Learning, SMOTEENN, Imbalanced Dataset, Neural Networks, Stratified Cross-Validation.**

I. INTRODUCTION

A stroke is a dangerous medical condition that may result in death or permanent disability if not identified and managed in a timely manner. The World Health Organization (WHO) claims that a substantial number of people are affected by strokes each year. A good number of these individuals are either sadly succumbing to their medical conditions or are living with profound disabilities due to the strokes they experienced [1]. What's especially concerning is that a significant number of these cases might have been prevented if early warning signs had been detected in time [2]. Unfortunately, traditional screening tools are often too slow, costly, or difficult to access—particularly in remote or under-resourced communities—making timely diagnosis a serious challenge [3]. That's why we chose to focus on this problem. We believe that smart, early-warning systems powered by technology could save lives and improve health outcomes on a large scale.

To build such a system, we had to deal with the challenges that come with real-world healthcare data. Medical datasets are often unbalanced—meaning they contain far fewer stroke cases than non-stroke cases—and they can also include missing or incomplete information. To address these issues, we applied modern methods like SMOTEENN, deep neural networks, and transfer learning [4]While traditional machine learning models have shown some success in identifying diseases using structured medical records [5], [6], [7] they tend to struggle when faced with complex, noisy, or imbalanced real-world data. In contrast, deep learning and transfer learning techniques are better at understanding intricate patterns and can still perform well even when there's not a lot of labeled data available [4].Our solution, StrokeNet-X, is a hybrid deep learning model that's designed specifically for stroke prediction using real clinical data. The model incorporates advanced data preprocessing and applies SMOTEENN, which cleverly combines oversampling and data cleaning, to deal with the imbalance between stroke and non-stroke cases [4].What sets our approach apart is that we don't just stop at predicting whether a stroke might occur. We also look into post-stroke complications—factors that can significantly affect a patient's emotional and physical recovery [8], [9], [10]. Additionally, we explore how transfer learning helps improve the model's ability to reduce false negatives, which is especially important because failing to detect someone at risk of stroke could lead to severe consequences. Unlike many existing systems that tend to overlook these critical errors, our model emphasizes sensitivity—making sure that people who truly need help are identified early. We understand that real-world clinical data is rarely perfect, and that's why our model is built to handle missing values, unbalanced data, and other challenges that can otherwise reduce the reliability of predictions.

II. RELATED WORK

Tanaka et al. [11] utilized real-time sensor data from wearable devices, applying advanced preprocessing techniques like

979-8-3315-9524-1/25/$31.00 ©2025 IEEE

signal denoising, temporal feature engineering, and Z-score normalization. Collecting and processing real-time physiologic data trained an LSTM model which achieved a 92% precision surpassing GRU and came to show promise for wearable healthcare monitoring systems. This work showcases effective temporal data processing and its influence on model performance.

Park et al. [12] used an IoT stroke monitoring dataset for time-series predictive modeling and proposed a Bi-LSTM framework achieving 93% accuracy with increased recall over baseline models.

Ali et al. [13] explored the heavily cited Kaggle Stroke Dataset, incorporating feature selection techniques like mutual information gain. They achieved impressive results with the XGBoost classifier, attaining an accuracy of 94%. Notably, age and blood sugar levels were significant factors in estimating the likelihood of a stroke.

Nair et al. [9] developed a CNN-based model for stroke prediction using data collected from smartwatches. This demonstrated how CNNs can be used with data from wearable sensors, emphasizing the promise that exists for systems aimed at the detection of strokes on a portable basis. Their study provided basic understanding of feature-based stroke classification systems. Nevertheless, it faced challenges with data representation, struggling with imbalance, which SMOTE-type algorithms could address.

Jagannadham et al. [14] describes an application of deep learning in the detection of brain tumors by the application of neural convolutional networks. Their systems process MRI scans in order to detect the tumor boundaries with precision. This approach is useful in enhancing the accuracy of the diagnosis and the speed of the interpretation.

Seva et al. [15] developed a smart system to predict liver disease using a machine learning method called Random Forest. Since medical data is often unbalanced, they used SMOTE-ENN to even things out and improve the model's performance. This helped the system make more accurate and fair predictions, making it useful for real-world healthcare.

Reddy et al. [16] worked on the importance of structured data splits and data augmentation for achieving high accuracy in real-time applications. Inspired by these works, we applied a welldefined split strategy and added to improve model generalization 2 classes.

III. Methodology

The stroke prediction system we built follows a simple and well-organized process, as shown in Fig 1. It all starts with collecting real medical data from sources like Kaggle. Since this data often includes missing values or messy entries, we begin by cleaning it up—filling in gaps and removing anything unusual. Next, we focus on the most important health factors like age, blood sugar, BMI, and blood pressure, which are key to identifying stroke risk. Because stroke cases are much rarer than non-stroke ones, we use a smart technique called SMOTEENN to balance the data and make the model more accurate. After preparing the data, we feed it into a deep learning model that learns to spot patterns. To make the model even better, we apply transfer learning, which means we build on models that have already been trained for similar tasks. We test the model thoroughly using 10-fold cross-validation to ensure it performs well on different parts of the data. Finally, we evaluate its performance using metrics like accuracy, precision, and F1-score. The ultimate goal is to help doctors catch early signs of stroke, so patients can get the right care at the right time.

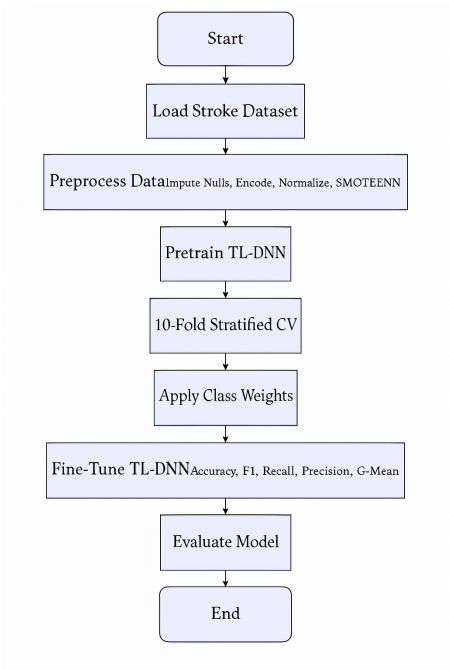

Fig. 1: Proposed Stroke Prediction TL-DNN Framework

A. Dataset Description

The dataset used in this project predicts if a person is likely to have a stroke. It includes details about each person's background, such as age and gender. It also covers daily habits like smoking and job type, health conditions like hypertension or heart disease, and some health measurements like BMI (Body Mass Index) and blood glucose level. The final column in the dataset, called stroke, indicates whether the person has had a stroke or not. All the columns used in the dataset as shown in the Table 1

Dataset [17] includes details of 43,400 people. Out of these, only 783 actually had a stroke, and the other 42,617 did not. This shows that most people in the dataset didn't have a stroke, which creates a problem called class imbalance. It means the data has very few positive cases, so the model might find it harder to learn how to detect stroke cases accurately.

```
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 43400 non-null  int64
 1   gender             43400 non-null  object
 2   age                43400 non-null  float64
 3   hypertension       43400 non-null  int64
 4   heart_disease      43400 non-null  int64
 5   ever_married       43400 non-null  object
 6   work_type          43400 non-null  object
 7   Residence_type     43400 non-null  object
 8   avg_glucose_level  43400 non-null  float64
 9   bmi                41938 non-null  float64
 10  smoking_status     30108 non-null  object
 11  stroke             43400 non-null  int64
dtypes: float64(3), int64(4), object(5)
```

TABLE I: Description of input variables

B. Data Preprocessing

The Fig.2 represents Filling in the Missing Information Some parts of the data are blank or not filled in. For missing BMI, we take the average BMI from all the other people and use that number to fill the blanks. For smoking info, we either mark it as "Unknown" or guess it based on other clues. Turning Words into Numbers Computers don't understand words the way we do. So, if we give them words like 'Male' or 'Self-employed', they get confused. We have to turn those words into numbers so the computer can make sense of them. So we convert: Words like gender into numbers: Gender was encoded using binary encoding(0:Male,1:Female). For things like job type—such as 'Private', 'Government Job', or 'Self-employed'—we change them into separate number codes. Private $\rightarrow$ [1, 0, 0] Self-employed $\rightarrow$ [0, 1, 0] Govt job $\rightarrow$ [0, 0, 1] This makes it easier for the computer to tell the differences between each type. To avoid that, we make all numbers the same size using something called standardization. All numerical features were standardized using the formula:

$$z = \frac{x - \mu}{\sigma}$$

where x is the feature value, μ is the mean, and σ is the standard deviation. The computer doesn't think big numbers are more important than small ones as shown in the Fig.3

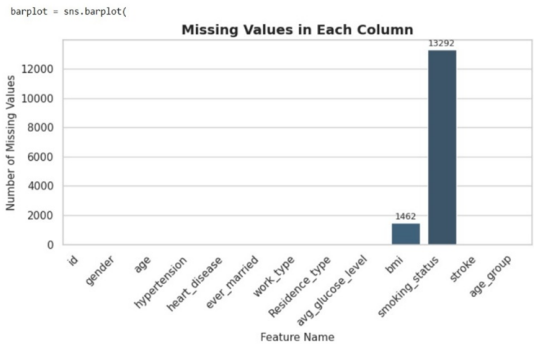

Fig. 2: Before missing values

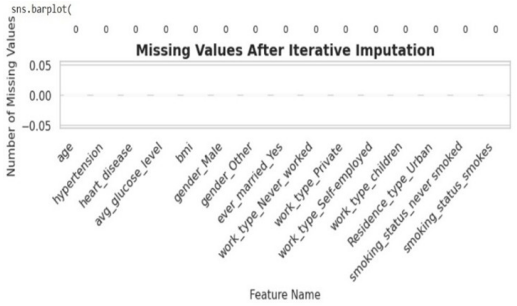

Fig. 3: After missing values

IV. MODEL ARCHITECTURE

The stroke prediction model is designed to understand how various aspects of a patient's health, like age, blood pressure, or medical history, might relate to their chances of having a stroke.

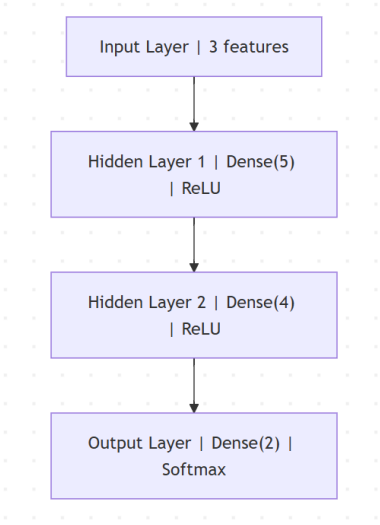

Fig. 4: DNN Architecture

The structure begins with an input layer, which then passes the information through two fully connected hidden layers — kind of like stepping stones that help the model learn and understand patterns more deeply as shown in Fig.4. The first hidden layer has 128 neurons, each acting like a little helper that looks at the input and tries to spot useful patterns. Together, they start breaking down the data to help the model understand what it means. They use the ReLU activation function, which helps the model focus on important patterns by turning off less useful signals and letting stronger ones pass through. This helps the model understand deeper and more complex patterns hidden in the data, almost like connecting the dots to see the bigger picture. To prevent the model from memorizing the training data too closely and to help it perform better on new, unseen data, a Dropout layer is added next. It randomly turns off 20% of the neurons during training, encouraging the model to learn more general and robust

patterns. The second hidden layer consists of 64 neurons, also using ReLU, and is followed by another Dropout layer with the same rate. These layers enable the model to extract meaningful representations from the data. The final output layer has a single neuron that acts like a decision-maker. This means it gives a result as a probability between 0 and 1—basically telling us how likely it is that a person might be at risk of having a stroke . The model is trained using the Binary Cross entropy loss function, which helps it measure how well it's making predictions between two classes—stroke or no stroke. To help the computer learn better, we use something called the Adam optimizer. It's like a smart helper that guides the computer, step by step, so it can find the right answers faster. This is really useful, especially when some things (like stroke cases) don't happen very often in the data.

V. TL-DNN Model

In medical data like this, there are a lot more people who didn't have a stroke compared to those who did. This uneven data can make it hard for the computer to learn how to spot stroke cases correctly, since there are so few of them. To fix this issue, we used a technique called SMOTEENN, which is a combination of two methods: SMOTE and ENN. SMOTE (Synthetic Minority Over-sampling Technique) helps by creating new, artificial stroke cases based on the existing ones. It does this by finding similar cases and generating new data points that are like them, which increases the number of stroke cases in the dataset. ENN (Edited Nearest Neighbors) helps clean up the data by removing incorrect or confusing records, especially from the majority class (non-stroke cases).For transfer learning, we used the pre-trained weights for the deep neural network on a massive healthcare data set and fine-tuned the last layers to perform better on the small stroke data set. It checks each data point and compares it to nearby ones—if it doesn't match well with them, it gets taken out This step helps get rid of noise or outliers that might confuse the model. In the below fig.5 by combining these two methods, SMOTEENN balances the dataset and cleans it up. This makes it more reliable for training. After applying SMOTEENN, the number of stroke and non-stroke samples becomes much closer. This helps the model learn more fairly from both stroke and non-stroke cases. As a result, it becomes better at spotting actual stroke cases and is less likely to miss them just because they are rare.

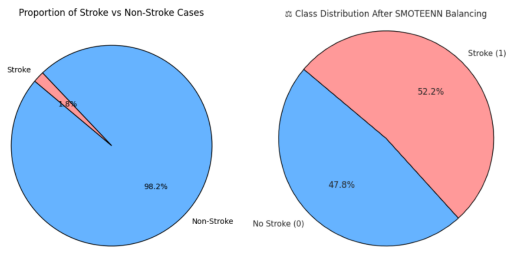

Fig. 5: class distribution after SMOTEENN

VI. Training Strategy

To make sure the computer works well for everyone—both people who might have a stroke and those who won't—we used a fair way to test it. It's called Stratified 10-Fold CrossValidation (don't worry, that's a big name for a simple idea). Here's how it works: We take all the data and split it into 10 equal parts. Each part has a fair mix of people with and without stroke. Then we test the computer 10 times, each time using a different part to see how well it does.

This helps us know if the computer is really good at its job, even when it sees new data it hasn't seen before. It's like giving the computer a practice test again and again to make sure it's ready for the real thing. In each round of training, we ran the Deep Neural Network (DNN) for 100 cycles (called epochs), using 32 samples at a time—this group size is known as the batch size. We used the Adam optimizer to help the model learn quickly and effectively, along with the Binary Crossentropy loss function, which is ideal for tasks where the goal is to make a clear yesor-no prediction—like whether someone is at risk of having a stroke.To prevent overfitting, we implemented EarlyStopping with a patience value of 10. It can training when the validation loss no longer improved. We also used ReduceLROnPlateau to lower the learning rate when performance stabilized, which helped with convergence.Important metrics like accuracy, precision, recall, and F1score across all 10 folds were used to validate the model. About 95 percent of the time, the model accurately predicted the result. Better yet, the model received a score of. Even better, the model scored an impressive 95.52% on the F1score, meaning it was great at telling apart stroke and nonstroke cases—and it did so without leaning too much toward either side. This shows strong performance and the ability to generalize across different data subsets.

VII. Experimental Setup

The experiments for stroke prediction were carried out on a regular home computer running Windows 10, powered by an Intel® Core™ i5 processor with 8 GB of RAM. Since no dedicated GPU was available, all computations were done on the CPU. To make the most of this setup, batch sizes and training parameters were carefully tuned to keep the process efficient. The dataset used included various clinical and lifestyle-related features relevant to stroke risk. Before training, the data was cleaned and preprocessed—missing values were filled in, categorical features were converted into numerical form, and all values were normalized. To balance the number of stroke and non-stroke cases, a technique called SMOTEENN was used, which both adds synthetic examples and removes noisy ones. The model itself was a deep neural network built using TensorFlow and Keras, trained using 10-fold cross-validation to ensure fair and reliable evaluation. Transfer learning was also applied by starting with a model that had already been trained on a related healthcare dataset.

VIII. RESULTS AND DISCUSSION

To check how well our model works, we used some important checks because the data we used had a lot more people without stroke than with stroke. That means we couldn't just look at how many times the model was right overall—we needed to look deeper. So, we used different ways to measure it, like G-Mean, False Positive Rate and False Negative Rate Out of all these, recall was the most important for us. That's because in stroke prediction, it's better to catch all the people who might have a stroke—even if some of them don't—than to miss someone who really needs help. If recall is high, it means our model is good at finding those people who could be in danger. We also used something called the ROC-AUC score, which helps us see how well the model can tell the difference between people who might have a stroke and those who probably won't. To make sure our model was doing well on all parts of the data, we split the data into 10 parts and tested the model on each one (this is called 10-fold cross-validation).The best fold shows as Confusion Matrix in fig.6. It did well each time, so we know the model is steady and trustworthy. Since the experiments were conducted on a CPU-only system, future work will evaluate performance on GPU-based clinical setups for real-time deployment.In the Table 2 TL-DNN Model shows the highest accuracy when compared to the other models as shown in the below.

TABLE 2: Performance Metrics of Improved TL-DNN Model

S.No	Model	Accuracy	G-Mean	FPR	FNR	AUC
1	CNN	77.3	78.1	26.0	17.6	77.0
2	DNN	77.3	77.7	25.0	19.4	77.0
3	ANN	80.1	80.3	21.7	17.7	80.0
4	TL-DNN	**95.2**	**95.0**	**7.3**	**2.5**	**98.6**

The stroke prediction model did a good job overall, performing well in all the tests we ran. Its recall score of 97. 22% means it is very good at finding patients who are at risk of stroke. Early diagnosis is important because it can help save lives. The model also has a high F1 score of 95. 52% and a precision of 93.87%, which shows it catches most stroke cases while keeping false alarms low, making its predictions more trustworthy. The ROC-AUC score of 0. 95 shows in the Fig.7 the model can clearly tell the difference between patients who may have a stroke and those who won't, even when the decision line changes. To make sure the results were reliable, the model went through 10-fold stratified cross-validation. This test showed the model performs well across different parts of the data. A G-Mean score of 95.13% shows that the model performs well on both stroke and non-stroke cases, handling them equally without favoring one over the other. Overall, these results suggest the model is ready for real use. It helps doctors find stroke risk early and improve how patients are treated.

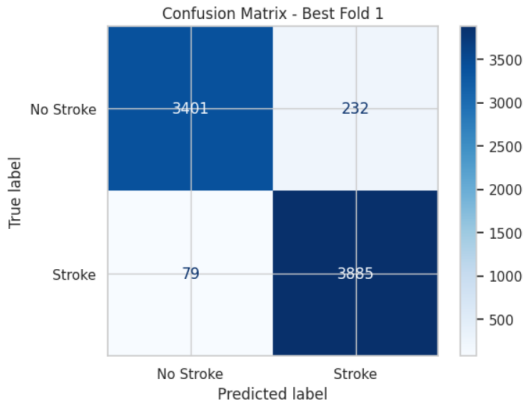

Fig. 6: Confusion Matrix

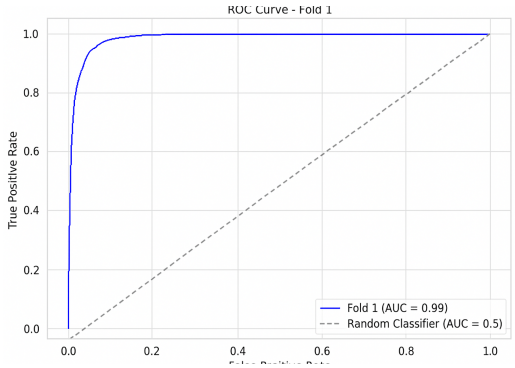

Fig. 7: ROC Curve

In Fig.8 the left graph shows accuracy over epochs, comparing training accuracy (blue line) and validation accuracy (green dashed line). Both lines show an upward trend, meaning the model's ability to correctly predict stroke cases improves with each epoch. The validation accuracy reaches above 97%, which indicates strong generalization to unseen data. Meanwhile, training accuracy steadily rises to around 91%, showing the model is learning effectively without overfitting.In Fig.8 the right graph presents loss over epochs, with training loss (red line) and validation loss (orange dashed line) both decreasing steadily. Lower loss means the model is making fewer mistakes. The validation loss drops significantly and remains below training loss, suggesting strong performance and a good fit to the validation data. Minor fluctuations in validation loss are normal but do not harm overall performance.In the Fig.9 the sample input and output have been displayed by using the TL-DNN model.

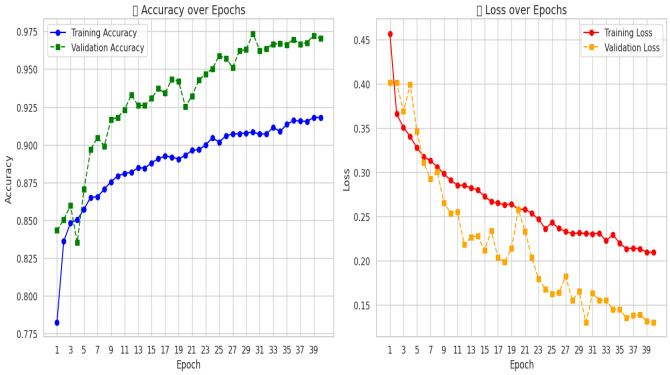

Fig. 8: Training and Validation Accuracy

```
sample_input = pd.DataFrame([{
    'gender': 'Male',
    'age': 3,
    'hypertension': 0,
    'heart_disease': 0,
    'ever_married': 'No',
    'work_type': 'children',
    'Residence_type': 'Rural',
    'avg_glucose_level': 95.12,
    'bmi': 18,
    'smoking_status': ''
}])
```

```
Predicted Probability of Stroke: 0.0000
Predicted Class: No Stroke
```

Fig. 9: Predicting Brain Stroke through TL-DNN Model

IX. Conclusion and Future Work

This study talks about a smart computer program that helps doctors find out if someone might get a stroke. A stroke is when the brain doesn't get enough blood, and it can be very dangerous. Sometimes, there aren't enough stroke cases in the data, so the computer might get confused. To fix this, the researchers used a special trick called SMOTEENN to help the computer learn better. This method gives the accuracy 95.28%.They also checked the computer's work many times using different pieces of the data. This made sure it worked well every time. The computer is very good at finding people who might get a stroke. It catches almost all the cases, so doctors don't miss anyone who needs help. It also does a great job telling who is safe and who might be in danger. In the end, this smart computer can help doctors find stroke signs early and take care of people better and that can save lives. While the results are encouraging, it is important to test the model on data from multiple hospitals and institutions to ensure its reliability in real-world clinical use. Although results are promising, validation on multi-institutional clinical datasets is essential to confirm the robustness of the proposed model

A. Looking at Brain Pictures

Scientists are working on smart systems that can look at brain scans (like X-rays or MRIs) and tell if someone might have a stroke soon. This can help doctors find problems much earlier.

B. Easy Explanations for Doctors

New stroke tools will be designed so that even doctors who don't use computers much can understand what the system is saying and why. This builds trust and helps in faster decisions.

C. Helping Doctors in Hospitals

Computers will be placed in hospitals to quickly check if any patient is in danger of stroke. They can help doctors save time and act fast.

References

[1] W. H. O. Dr Hanan Balkhy, "Stroke, cerebrovascular accident," 2024, https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.

[2] V. L. Feigin, B. Norrving, and G. A. Mensah, "Global burden of stroke," *Circulation Research*, vol. 120, no. 3, pp. 439–448, 2017.

[3] B. C. V. Campbell, "Ischaemic stroke: Pathophysiology and principles of treatment," *International Journal of Stroke*, vol. 14, no. 6, pp. 574–584, 2019.

[4] D. Mukherjee and A. Patil, "Early prediction of stroke using machine learning and deep learning models," *IJETER*, vol. 8, no. 5, pp. 1946–1951, 2020.

[5] I. Kavakiotis, "Machine learning and data mining methods in diabetes research," *Computational and Structural Biotechnology Journal*, vol. 15, pp. 104–116, 2017.

[6] A. Mirajkar, "Stroke prediction using machine learning techniques," *International Journal of Engineering Research Technology*, vol. 9, no. 6, pp. 382–386, 2020.

[7] S. Sunayna, S. Rao, and M. Sireesha, "Performance evaluation of machine learning algorithms to predict breast cancer," in *Computational Intelligence in Data Mining*, ser. Smart Innovation, Systems and Technologies, J. Nayak, H. Behera, B. Naik, S. Vimal, and D. Pelusi, Eds. Springer, Singapore, 2022, vol. 281, pp. 71–79.

[8] X. Liang, "Stroke prediction using deep learning," *BioMed Informatics*, vol. 20, no. 4, pp. 243–251, 2022.

[9] H. Nair, "Cnn-based stroke risk prediction using smartwatch-generated physiological data," *Sensors*, 2024.

[10] S. Mamidala, S. Moturi, S. Rao, J. Bolla, and K. Reddy, "Machine learning models for chronic renal disease prediction," in *Data Science and Applications. ICDSA 2023*, ser. Lecture Notes in Networks and Systems, S. Nanda, R. Yadav, A. Gandomi, and M. Saraswat, Eds. Springer, Singapore, 2024, vol. 819, pp. 291–301.

[11] Y. Tanaka, "Real-time stroke prediction using wearable sensor data and lstm networks," *IEEE Journal of Biomedical and Health Informatics*, 2024.

[12] M. Park, "Iot-enabled stroke monitoring and prediction using bi-lstm model," *IEEE Access*, 2024.

[13] S. Ali, "Stroke prediction using feature selection and xgboost on kaggle dataset," in *Proceedings of the International Conference on Health Informatics (ICHI)*, 2024.

[14] S. L. Jagannadham, K. L. Nadh, and M. Sireesha, "Brain tumour detection using cnn," in *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, 2021, pp. 734–739.

[15] A. Seva, S. N. Tirumala Rao, and M. Sireesha, "Prediction of liver disease with random forest classifier through smote-enn balancing," in *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*, Jabalpur, India, 2024, pp. 928–933.

[16] K. V. N. Reddy, "Automated traffi c sign recognition via cnn deep learning," in *IEEE IATMSI*, 2025.

[17] Kaggle, "Cerebral stroke prediction – imbalanced dataset," 2022, https://www.kaggle.com.