International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

ML-Powered Brain Stroke Prediction Identifying Key Risk Factors for Early Detection

Serra Aksoy

Institute of Computer Science, Ludwig Maximilian University of Munich (LMU), Oettingenstrasse 67, 80538 Munich, Germany

Abstract

This research takes a new direction by applying machine learning techniques to predict the likelihood of cerebrovascular accidents, or simply strokes, from an extensive dataset that was meticulously collected on the top-rated platform Kaggle. Through an extensive and thorough exploratory data analysis, we were able to reveal some of the significant risk factors accountable for these accidents, some of which are but not limited to age, hypertension, presence of cardiovascular disease, and average blood glucose level. In a bid to create effective prediction models, we utilized a series of sophisticated algorithms such as Logistic Regression, Decision Trees, and Random Forests, all of which collectively achieved a whopping accuracy of 95%. The findings demonstrate the remarkable potential of machine learning technology not only to predict strokes, but to identify and prevent them at an early level. This underscores the paramount importance of recognizing these top-risk factors in the framework of predictive modeling.

Keywords: Machine Learning, Brain Stroke Prediction, Logistic Regression, Decision Trees, Random Forest.

1. Introduction

A stroke in the brain occurs when the brain's blood flow is abruptly interrupted, and brain cells are injured. The two main forms are ischemic stroke, resulting from a blood clot, and hemorrhagic stroke, resulting from a ruptured blood vessel. Sudden weakness or numbness, confusion, speech impairment, and terrible headaches are symptoms. Strokes are medical emergencies requiring immediate care for effective treatment and minimizing long-term injury.

Machine learning (ML) is a subset of artificial intelligence (AI) that entails the development of algorithms and models that enable computers to learn from data and make decisions or predictions without being specifically programmed. It aims at coming up with models that can be generalizable from data. For instance, in this research, the machine learning model is trained to forecast the likelihood of an individual having a stroke without initially knowing if a stroke has occurred in the individual or not.

The ongoing improvement in stroke prediction techniques, ranging from the traditional statistical techniques to deep learning-based techniques, echoes the sustained prowess of artificial intelligence in medical diagnostic systems. Although initial ML techniques, including Random Forest (RF), Support Vector Machines (SVMs), and logistic regression, were vital in the prediction of stroke outcomes, recent research accentuates the enhanced precision and credibility that advanced deep learning models offer. These large dataset-trained models have achieved notable accuracy up to 98%, showing AI's potential

for early diagnosis, personalized treatment, and improved patient outcomes. Moving forward, multimodal data sources, explainable AI, and real-time clinical application will be essential to increase predictive accuracy even more and improved stroke care across the world.

2. Literature Review

Stroke is still a pressing and serious health issue around the world, always being the second leading cause of death in most nations and recognized as the third principal cause of disability worldwide, as was unearthed by the World Health Organization in a 2025 study [1,2]. In the initial stages of developing methods of outcome prediction following stroke, researchers depended heavily on conventional statistical approaches in addition to typical classical machine learning techniques. Some of the techniques that were used included RF [3,4], SVM [5], and logistic regression [6,7], which were widely used in analyzing data and making predictions in the field. The models that were developed primarily concentrated on the task of feature selection, which was based on structured clinical data, along with imaging biomarkers.

Aksoy et al. [8] presented a study utilizing deep learning, specifically the ConvNeXt Base model, to predict ischemic strokes using MRI scans. Recognizing the challenges of manual diagnosis due to high patient volumes, their model was trained on labeled medical imaging data to identify stroke-related patterns. The model achieved 84% accuracy, demonstrating its potential for early detection and improved clinical decision-making, ultimately facilitating faster and more effective stroke interventions.

Gupta et al. [9] used a Kaggle stroke dataset [8] that had 5110 patient records. Preprocessing of data included handling missing values, transforming categorical variables into numeric form using Label Encoding, and SMOTE technique-based class balancing of the dataset. The dataset was divided into 80:20 train and test ratios. Seven classifiers were experimented on, and the highest accuracy of 97% was achieved with RF.

Zubaidai et al. [10] also used a Kaggle dataset containing 5110 patients and 12 attributes. Preprocessing was done to deal with missing values, categorical, redundant features, and imbalance. They trained six models after splitting the data into an 80:20 ratio and found that Random Forest was better among the remaining five of them with 97% accuracy.

Dritsas et al. [11] performed stroke prediction in patients over 18 years old from a Kaggle dataset. They preprocessed the data and trained their system using nine algorithms. The best accuracy was achieved by the stacking algorithm with AUC of 98.9%, 98% accuracy, and 97.4% recall, precision, and F-measure.

3. Classification Model Performance Analysis

A confusion matrix is generally utilized for evaluating the performance of a classification model (Figure 1). It is presented in the form of a tabular format that recapitulates the predictions made by a model with the real labels of the dataset. The matrix includes four different elements. True Positive (TP) instances where the model accurately predicts the positive class. True Negative (TN) instances when the model accurately predicts the negative class. False Positive (FP) are the cases where the model incorrectly classifies the positive class. False Negative (FN) instances when the model incorrectly predicts the negative class.

The accuracy measure is commonly utilized as an evaluation measure to identify how effective the classification model has been. This measure is acquired by computing a ratio of correctly predicted instances, both true positives and true negatives, against all the instances present in the data. High

accuracy is a high value of accuracy (near 1 or 100%) indicates that a high percentage of the predictions made by the model are accurate. Low accuracy is a low accuracy value, a number which approaches 0, indicates that a high number of incorrect predictions have been made by the model. While accuracy is a straightforward and popular metric, it may be insufficient for datasets that are overwhelmed by class imbalances.

In settings where the classes are unevenly distributed, other metrics such as precision, recall, and the F1 score are employed to enable a more effective assessment. In the present analysis, even though a high accuracy rate was attained, the confusion matrix and other evaluation metrics showed that the model could not make believable predictions for stroke cases. Precision, recall, and F1 score are widely utilized metrics in classification problems, especially those involving imbalanced datasets. Precision is the proportion of true positive predictions, and all positives predicted, i.e., true positives and false positives. It tells us how correct the positive predictions of the model are. A high precision value indicates that the model's positive predictions are likely to be accurate.

Recall (Sensitivity or True Positive Rate) is calculated by dividing true positive predictions by the sum of actual positives, i.e., true positives and false negatives. This metric computes the model's capacity to recall all true positive instances. A high recall signifies that most actual positive instances have been labeled accurately.

F1 score is defined as the harmonic mean of precision and recall. This measure is a balance between the two measures, making it particularly useful for overall model evaluation. An F1 score indicates that the model achieves high precision and recall levels, thus guaranteeing accurate positive predictions while capturing a high percentage of true positive instances. These metrics give a better evaluation of the performance of classification models, particularly in situations where there are datasets with extreme class imbalances.

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# Accuracy Score
ac = accuracy_score(y_test, y_pred)
print(ac)

# Precision, Recall and F1 score
print(classification_report(y_test, y_pred))
```

Figure 1: Evaluation of Classification Model Performance

4. Material and Method

In this study, Brain Stroke Prediction dataset from Kaggle, which is a famous data science and machine learning community, was used [12]. This dataset contains 5110 patient data, and each record is characterized by 12 various clinical and demographic variables of importance while deciding on stroke risks. Among the dataset comprising various risk factors shortlisted in this study; hypertension, heart disease, age, and smoking status, all of which are proven to be significant drivers of the likelihood of a stroke. The specific features considered for analysis in this study are outlined as follows:

- Gender (Male, Female, Other)
- Age is a continuous variable.

- Hypertension (Binary: 0 = No, 1 = Yes)
- Heart Disease (Binary Variable: 0 for No and 1 for Yes)
- Ever Married (Categorical: Yes or No)
- Work Type (Categorical: Children, Govt_job, Never_worked, Private, or Self-employed)
- Type of Residence (Categorical Classification: Either Rural or Urban)
- Average Glucose Level (Continuous variable)
- BMI (Body Mass Index, Continuous variable)
- Smoking Status (Categorical: Never smoked, Formerly smoked, Smokes, or Unknown) Stroke (Binary: 0 = No Stroke, 1 = Stroke).

In order for the dataset to be ready for predictive modeling, a number of preprocessing steps were undertaken carefully. There were missing values in the feature BMI, which was imputed with the mean value to ensure consistency in the dataset. Categorical features such as Gender, Ever Married, Work Type, Residence Type, and Smoking Status were converted to numerical form using one-hot encoding. In this way, each unique category in these features can be expressed as a single binary vector, thus making the data interpretable to machine learning models. While selecting features, the ID column was dropped from the dataset as it had no useful information and made little contribution to the predictive modeling objective. In order to enhance the model's efficacy, continuous variables like Age, Average Glucose Level, and BMI were normalized with Min-Max Scaling so that all of the features were in the same range. The dataset was divided into 80% training and 20% testing, hence allowing for sufficient model training and evaluation and ensuring sufficient evaluation of the predictive model's performance.

The Random Forest Classifier served as the core predictive model because of its efficiency in dealing with imbalanced datasets and its capability in detecting non-linear relationships among risk factors of stroke. The model was trained on the preprocessed data with 500 trees, and the hyperparameters were tuned using grid search cross-validation to maximize accuracy. The model's performance was evaluated based on the evaluation metrics. Accuracy predicts overall classification performance. Accuracy is the ratio of stroke cases correctly identified to the total number of stroke cases estimated. Recall (Sensitivity) assesses how well the model identifies true cases of stroke. F1-Score is the harmonic average of recall and precision, which provides a balanced measure of the performance of the model. All calculations and experiments were performed using Python 3.8 and Scikit-learn and Pandas libraries for data preprocessing, training models, and measuring performance. The trained model for real-time stroke risk prediction is deployed using a Python implementation [13] (Figure 1), as shown below.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load dataset
df = pd.read_csv('stroke_data.csv')

# Handle missing values
df.fillna(df.mean(), inplace=True)

# Encode categorical variables
```

```
df = pd.get_dummies(df, drop_first=True)

# Split features and target
X = df.drop(columns=['stroke'])
y = df['stroke']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Figure 2: Brain Stroke Prediction Algorithm

Independent variables (features) are the input variables used for making predictions, and the dependent variable (target) is the outcome which the model would be trained to forecast. One row in the data set is a single observation, and one column is one distinct feature. In the present study, the outcome of stroke is utilized as the dependent variable, which is predicted using a set of independent variables such as age, body mass index (BMI), average glucose level, and other applicable clinical and demographic variables.
In order to prepare the data for training the models, the dataset was divided into dependent and independent variables (Figure 3). The feature matrix (X) was obtained by dropping the stroke column from the dataset, and the target variable (y) was just the stroke column.

```
X = stroke_df.drop('stroke', axis=1)
y = stroke_df['stroke']
```

Figure 3: Feature and Target Variable Separation in Stroke Prediction Dataset

Regression is a type of supervised learning that aims to predict a continuous numeric value. More precisely, it is used when the output variable is a real or floating-point number. Some examples of regression issues are predictions of house prices, temperature, or salary, since the target variable here is a continuous metric.
Classification, conversely, is yet another form of supervised learning whose aim is to predict the category or class to which a new data point will belong. The output variable in classification is discrete and indicates separate classes or categories. Some common examples are spam or non-spam classification of emails, image recognition issues, and medical diagnosis to ascertain if a patient has a particular disease. The choice of classifier is explained by the problem at hand. As the objective is to identify if an individual is likely to experience a stroke, the problem is a binary classification problem

(stroke or no stroke). The individuals are being divided into two groups based on some attributes or risk factors, so classification is the suitable technique for this problem.

Supervised learning refers to a process where an algorithm is trained using a labeled dataset. In this regard, each training example in the dataset is linked with its corresponding output or target variable. Since classification models are established using labeled target variables, classification and regression are both subcategories of supervised learning.

Conversely, unsupervised learning is the process of training an algorithm on a dataset with no target variables or explicit labels. The goal of the algorithm is to find patterns, structure, or relationships between the data without any defined outputs.

After following the prescribed procedure of machine learning, the model undergoes selection, training, and validation of its predictive ability. The model is then chosen for deployment if it shows better predictive ability.

In this study, certain classification models are developed, and the Random Forest Classifier is chosen because of its higher accuracy score and positive results represented in the confusion matrix. Prior to the model training phase, a critical issue in machine learning called imbalanced data needs to be handled.

Imbalanced data in classification problems happens when one class is greatly underrepresented. It biases the predictions of the model toward the majority class, lowers its generalizability for the minority class, and makes its performance hard to measure. In our case, the dataset is composed mainly of instances of the majority class (individuals without stroke), which results in exaggerated accuracy with poor performance in classifying the minority class (individuals with stroke). This is a common occurrence in numerous domains, especially in the medical field.

There are a few techniques to reverse the effects of imbalanced data, and only the selected technique is given here. SMOTE (Synthetic Minority Over-Sampling Technique) is applied to counter class imbalance by generating synthetic samples for the minority class. Artificial examples are created along the line segments between existing minority-class instances, which improve representation and balance the data set. Merits of SMOTE:

- Bias towards the majority class is reduced.
- Problems of underfitting are avoided by increasing instances for the minority class.

The application of SMOTE (Figure 4) enables balanced representation of stroke and non-stroke cases, resulting in better model performance. The data is first split into training and test sets. SMOTE is applied to generate synthetic instances, and then a further split of the dataset is performed to maintain balanced classes.

```
from imblearn.over_sampling import SMOTE

X = stroke_df.drop('stroke', axis=1)
y = stroke_df['stroke']

smote = SMOTE(sampling_strategy='minority')
X_sm, y_sm = smote.fit_resample(X, y)
print(y.value_counts())
print(y_sm.value_counts())

X_train, X_test, y_train, y_test = train_test_split(X_sm, y_sm, test_size=0.2, random_state=0,
stratify=y_sm)
print(y_train.value_counts())
print(y_test.value_counts())
```

Figure 4: SMOTE Implementation Code

Subsequent to dividing the dataset into two subsets, an instance of the SMOTE class is instantiated and fitted to the variables X and y, thereby equalizing the number of data points. This can be verified utilizing the value_counts() function. Following this, the dataset is partitioned into training and testing sets; however, in this instance, X_sm and y_sm are employed in place of X and y.

Random Forest Classification is comprised of an array of decision trees. Each one examines various facets of a problem, then votes to determine the most suitable solution. The random forest then tallies their votes to provide a more stable result than the use of a single tree, as is the case with decision tree classification (Figure 5).

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(random_state=0)
rf.fit(X_train, y_train)
```

Figure 5: Implementation of Random Forest Classifier

An instance of the Random Forest Classifier class is created first. A random state is specified to get reproducible results if the model is executed several times on the same data. Next, the model is trained by calling the fit function with X_train and y_train as parameters. In this way, the model is trained on the training data. Then predictions are made on the test data (Figure 6).

```
y_pred = dt.predict(X_test)
```

Figure 6: Prediction of Test Set Using a Decision Tree Classifier

5. Results

A confusion matrix was employed as a metric to examine and test the performance of the classification model that had been developed. The outcome of this examination reveals that the model accurately predicted a total of 911 cases that were "no stroke" and accurately determined 938 cases that were "stroke." Note, however, that there were a few misclassifications; specifically, 62 cases that were actually "no stroke" were misclassified, along with 34 cases that were supposed to be "stroke" but were misclassified. This resulted in the model making several false predictions. All told, the model achieved an excellent accuracy rate of 95.06%, a finding which firmly indicates that the classification task was carried out with a high level of efficacy and reliability. The high level of accuracy indicates that most of the predictions were made accurately for two classes.

In terms of Precision, for "No Stroke" (Class 0): 93.6% of the predicted cases were actually "no stroke." For "Stroke" (Class 1), 96.5% of the predicted cases were actually "stroke."

In terms of Recall, for "No Stroke" (Class 0): 96.4% of the actual "no stroke" cases were predicted accurately. For the case of "Stroke," which is labeled as Class 1, an impressive 93.8% of all actual instances of "stroke" were accurately identified and detected.

Directing our attention now to the F1 Score, we observe that for "No Stroke," which is labeled as Class 0, there was an extremely good balance of 95% between the two most significant measures of precision and recall. Conversely, for "Stroke" labeled as Class 1, the balance between recall and precision was recorded at an equally good 95.2%. The results indicate that the model was very good at distinguishing between cases labeled as "No Stroke" and those labeled as "Stroke." This was done with commendable

degrees of precision, recall, and F1-score. Therefore, these results indicate the high effectiveness of the model at accurately forecasting cases of stroke occurrences.

In order to make predictions in real time with the minimum amount of effort required and to facilitate the deployment of models as a web application, a decision was made to deploy the model with the assistance of Streamlit, which is one of the many frameworks that are popular for building web applications. Before deploying it, however, it was important that the model was saved so that it could be used at some time in the future; thus, the Pickle library was utilized for this. With the assistance of Pickle for model persistence, the model was successfully saved using the serialization process. This was carried out in write-binary mode, and the following commands were utilized to accomplish this:

```
import pickle

pickle_out = open('stroke_prediction_model.pkl', 'wb')
pickle.dump(rf, pickle_out)
pickle_out.close()
```

Figure 7: Prediction of Test Set Using a Decision Tree Classifier

Once the entire process was achieved successfully, the serialized model file named stroke_prediction_model.pkl was stored securely and maintained in the provided working directory. In addition to this, a Streamlit application, built on top of the Python programming language, was developed with the motive of providing an interactive interface to the users in order to calculate and predict their likelihood of having a stroke. The interactive application offers input fields for various related health parameters, which are then processed and computed by the trained model to generate meaningful predictions.

Subsequently, the application is initiated. A new Python file, designated as app.py, is established for this purpose (Figure 8).

```
import streamlit as st
import numpy as np
import pickle

def main():
    st.title("Stroke Prediction")

    # Display a formatted note about the input values
    st.markdown("""
    *Note:*
    - Gender: 1 for Male, 0 for Female
    - Work Type: 0 for Govn Job, 1 for Never worked, 2 for Private, 3 for Self-
employed, 4 for Children
    - Residence Type: 1 for Urban, 0 for Rural
    - Smoking Status: 0 for Unknown, 1 for Formerly smoked, 2 for Never smoked, 3 for
Smokes
    - For the remaining questions, 1 is Yes and 0 is No
    """)

    gender = st.number_input('Gender', min_value=0, max_value=1, step=1)
    age = st.number_input('Age', min_value=0)
    hypertension = st.number_input('Hypertension', min_value=0, max_value=1, step=1)
```

```
    heart_disease = st.number_input('Heart Disease', min_value=0, max_value=1, step=1)
    ever_married = st.number_input('Ever Married', min_value=0, max_value=1, step=1)
    work_type = st.number_input('Work Type', min_value=0, max_value=4, step=1)
    Residence_type = st.number_input('Residence Type', min_value=0, max_value=1,
step=1)
    avg_glucose_level = st.number_input('Average Glucose Level', min_value=0.0)
    bmi = st.number_input('BMI', min_value=0.0)
    smoking_status = st.number_input('Smoking Status', min_value=0, max_value=3,
step=1)

    user_input = np.array([gender, age, hypertension, heart_disease, ever_married,
work_type, Residence_type,
                           avg_glucose_level, bmi, smoking_status]).reshape(1, -1)

    pickle_in = open('stroke_prediction_model.pkl', 'rb')
    rf = pickle.load(pickle_in)

    if st.button('Predict'):
        prediction = rf.predict(user_input)

        if prediction[0] == 0:
            st.subheader('No Stroke Risk')
            st.write("Based on the provided information, it seems there is no
immediate risk of stroke. However, it's always advisable to consult with a healthcare
professional for a more accurate assessment.")
        else:
            st.subheader('Stroke Risk')
            st.write("Based on the provided information, there is a potential risk of
stroke. It is strongly recommended to consult with a healthcare professional for a
thorough evaluation and guidance on preventive measures.")

if __name__ == '__main__':
    main()
```

Figure 8: Stroke Risk Prediction Using Random Forest Model with Streamlit Interface

The critical step that needs to be performed at this particular stage of the process is the important task of deserializing the pre-trained model, and this is done effectively by using the well-known Pickle library. Following this important step, the model predicts an output from the given input, and a related message is accordingly returned to the user, which illustrates the outcome of this prediction in a brief and descriptive manner. An elaborate and detailed description with regard to the specific details and attributes of the Streamlit application created here is intentionally omitted from discussion. This is because the design and customization of the interface are left to the subjective preference and personal creativity of the implementer in question. The choice of using Streamlit as a framework for this application was highly influenced by its inherent simplicity and ease of use in facilitating the ease of creation of web applications. This is particularly so when one considers the numerous complexities usually involved in the creation and customization of a web interface by way of using HTML, which would otherwise require significantly more effort and time in achieving the same levels of customization and flexibility. In comparison to other frameworks, Streamlit provides an extremely easy and pleasant approach to creating web applications.

Another thing to keep in mind is as follows: to actually execute the application, you must navigate to the terminal in the PyCharm environment. There, you must enter the command streamlit run app.py to

initiate the process. Upon executing this command, the application will open directly in a web browser, and you will immediately see it and be able to interact with it. Additional customization of the application can also be achieved by going to its settings, where a variety of options exist to tailor the experience as you would like. Deployment options are also readily accessible, and you are able to generate a shareable URL that others may use. This generated URL can then be distributed to other users, thereby granting them easier access to interact with the deployed application that you have created.

6. Conclusion

In this study, advanced machine learning techniques are used to predict the likelihood of cerebrovascular incidents (strokes) based on a high-dimensional dataset available on the Kaggle database. By following a rigorous exploratory data analysis (EDA) procedure, we were able to successfully identify key risk factors for stroke incidents, including but not limited to age, hypertension, cardiovascular morbidities, and average glycemic levels. For constructing the predictive models, we utilized an assortment of supervised learning algorithms, viz. Logistic Regression, Decision Trees, and Random Forests, that achieved an ensemble accuracy of 95% as assessed by cross-validated performance measures like F1-score, ROC-AUC, and precision-recall curves. The Random Forest classifier, with enhanced feature importance rankings and less overfitting due to bootstrap aggregation, was pickled with Python's pickle module for persistence and subsequently incorporated into a Streamlit-based front end for real-time inference, where dynamic user input processing and probabilistic risk stratification were enabled. These results attest to the power of machine learning in not only attaining precise stroke prediction but also facilitating proactive recognition of vulnerable cohorts through explanation of significant feature interactions in the predictive model. This study highlights the value of feature-engineered, data-driven solutions in the optimization of clinical decision support systems, thus paving the way for early intervention practice in neurovascular well-being. Future studies can investigate the integration of multimodal biomarkers, including features derived from neuroimaging, with the application of novel ensemble architectures like gradient boosting or deep neural networks to further increase predictive accuracy and enhance model generalizability across demographic groups.

Multimodal neuroimaging and gradient boosting can be applied in future studies for better prediction. Data scientists and clinicians must collaborate to utilize machine learning in stroke prevention and treatment.

References

1. World Health Organization: The Top 10 Causes of Death Available online: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.
2. Samak, Z.A.; Clatworthy, P.; Mirmehdi, M. Automatic Prediction of Stroke Treatment Outcomes: Latest Advances and Perspectives. *Biomed. Eng. Lett.* **2025**, doi:10.1007/s13534-025-00462-y.
3. McKinley, R.; Häni, L.; Gralla, J.; El-Koussy, M.; Bauer, S.; Arnold, M.; Fischer, U.; Jung, S.; Mattmann, K.; Reyes, M.; et al. Fully Automated Stroke Tissue Estimation Using Random Forest Classifiers (FASTER). *J. Cereb. Blood Flow Metab.* **2017**, *37*, 2728–2741, doi:10.1177/0271678X16674221.
4. Böhme, L.; Madesta, F.; Sentker, T.; Werner, R. Combining Good Old Random Forest and DeepLabv3+ for ISLES 2018 CT-Based Stroke Segmentation. In *Brainlesion: Glioma, Multiple*

Sclerosis, Stroke and Traumatic Brain Injuries; Crimi, A., Bakas, S., Kuijf, H., Keyvan, F., Reyes, M., Van Walsum, T., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, 2019; Vol. 11383, pp. 335–342 ISBN 978-3-030-11722-1.

5. Maier, O.; Wilms, M.; Von Der Gablentz, J.; Krämer, U.; Handels, H. Ischemic Stroke Lesion Segmentation in Multi-Spectral MR Images with Support Vector Machine Classifiers.; Aylward, S., Hadjiiski, L.M., Eds.; San Diego, California, USA, March 24 2014; p. 903504.
6. Chawla, M.; Sharma, S.; Sivaswamy, J.; Kishore, L.T. A Method for Automatic Detection and Classification of Stroke from Brain CT Images. In Proceedings of the 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society; IEEE: Minneapolis, MN, September 2009; pp. 3581–3584.
7. Rekik, I.; Allassonnière, S.; Carpenter, T.K.; Wardlaw, J.M. Medical Image Analysis Methods in MR/CT-Imaged Acute-Subacute Ischemic Stroke Lesion: Segmentation, Prediction and Insights into Dynamic Evolution Simulation Models. A Critical Appraisal. *NeuroImage Clin.* **2012**, *1*, 164–178, doi:10.1016/j.nicl.2012.10.003.
8. Aksoy, S.; Demircioglu, P.; Bogrekci, I. Optimizing Stroke Classification with Pre-Trained Deep Learning Models. *J. Vasc. Dis.* **2024**, *3*, 480–494, doi:10.3390/jvd3040036.
9. Gupta, S.; Raheja, S. Stroke Prediction Using Machine Learning Methods. In Proceedings of the 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence); IEEE: Noida, India, January 27 2022; pp. 553–558.
10. Al-Zubaidi, H.; Dweik, M.; Al-Mousa, A. Stroke Prediction Using Machine Learning Classification Methods. In Proceedings of the 2022 International Arab Conference on Information Technology (ACIT); IEEE: Abu Dhabi, United Arab Emirates, November 22 2022; pp. 1–8.
11. Dritsas, E.; Trigka, M. Stroke Risk Prediction with Machine Learning Techniques. *Sensors* **2022**, *22*, 4670, doi:10.3390/s22134670.
12. Brain Stroke Prediction, Https://Medium.Com/@serurays/Brain-Stroke-Prediction-3df9243d38bc 2025.
13. Stroke Prediction Model, Https://Github.Com/Serurays/Stroke_Prediction_Model 2025.