Forecasting the Incidence of a Stroke using Machine Learning Models

Data Mining Project

Aliaksandr Zaman 109163

Aliaksandr Mazur 109081

Bogdan Yanovich 109072

Warsaw 2024

Table of contents

Introduction

Stroke is a severe medical condition with significant consequences, so making early detection and risk assessment is crucial for preventive interventions. In recent years, the integration of machine learning techniques in healthcare has gained significant attention, offering promising avenues for predicting and preventing various medical conditions. Machine learning has shown promise in predicting stroke risk based on various health indicators and demographic information. This project delves into the application of machine learning models to forecast the occurrence of strokes, a critical health concern with substantial implications for individuals and healthcare systems.

The goal of this project is thorough data exploration, feature manipulation, and the construction of multiple ML models to identify the most effective approach to predict the occurrence of a stroke. The project specifically explores the effectiveness of three distinct models - AdaBoost, Gradient Boosting, and Support Vector Machine - in predicting the likelihood of a stroke.

Data exploration

In our project, a dataset from the Electronic Health Record (EHR) was utilized, which was made available on the McKinsey & Company website and the Kaggle online platform. This dataset comprises 4987 observations, described by 10 independent variables and 1 dependent variable. Data analysis, machine learning model development, and model quality evaluation were conducted using Python. The table below provides a detailed description of each variable in the dataset.

Name of the variable	Explanation	Type	Values
gender	Gender of the patient	Character	“Male “ or “Female”
age	Age of the patient	Integer	From 1 to 82
hypertension	Does the patient suffer from hypertension	Binary	0 - no, 1 - yes
heart_disease	Does the patient have a heart disease	Binary	0 - no, 1 - yes
ever_married	Has the patient ever been married	Binary	0 - no, 1 - yes
work_type	Patient's place of work	Character	“Children”, “Govt_job”, “Neverworked”, “Private”, “Self-employed”
residence_type	Patient's place of residence	Character	“Rural”, “Urban”
avg_glucose_level	Average blood glucose levels	Numeric	From 55.1 to 272
bmi	Body mass index	Numeric	From 14 to 48.9
smoking_status	Does the patient smoke cigarettes	Character	“formerly smoked”, “never smoked”, “smokes”, “Unknown”
stroke	Does the patient have a stroke	Binary	0 - no, 1 - yes

Source: Own work.

The dataset contains 3 binary variables: heart disease (heart_disease), hypertension (hypertension), marital status (ever_married); 4 categorical variables: gender, work type, residence type, smoking status; one integer variable – age, and 2 floating-point variables: average glucose level in blood (avg_glucose_level) and body mass index (bmi). The dependent variable in the dataset is a binary stroke variable, where 1 indicates that after the examinations, it was found that the patient indeed had a stroke, and 0 indicates that the patient did not have a stroke. There are no missing data in the dataset.

To facilitate further analysis and correctly build machine learning models we've created dummy variables from variables gender, smoking_status, residence_type, work_type, and ever_married. Among the dummy variables, residence_type_Urban, ever_married_Yes, gender_Male, smoking_status_never_smoked, and work_type_Self-employed were removed to avoid multicollinearity, which is often a cause of incorrect predictions and difficulty in interpretation of the results.

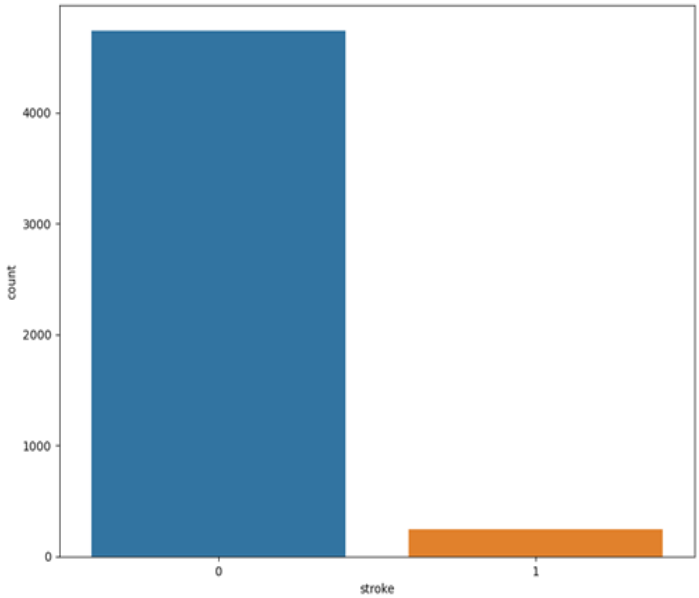
In the table below, descriptive statistics for the floating-point and integer variables have been calculated. The computed statistics include the mean, standard deviation, minimum, maximum, first quartile, median, and third quartile.

	Mean	Standard deviation	Minimum	Maximum	First quartile	Median	Third quartile
age	43.4	22.64	1	82	25	45	61
bmi	28.5	6.8	14	48.9	23.7	28.1	32.6
avg_glucose_level	106	45.1	55.1	271.7	77.23	91.85	113.86

Source: Own work.

The dependent variable "stroke" is of binary type: 0 indicates that the patient did not have a stroke, and 1 indicates that the patient had a stroke. Below is a distribution plot of the dependent variable:

Distribution of the Dependent Variable "stroke"

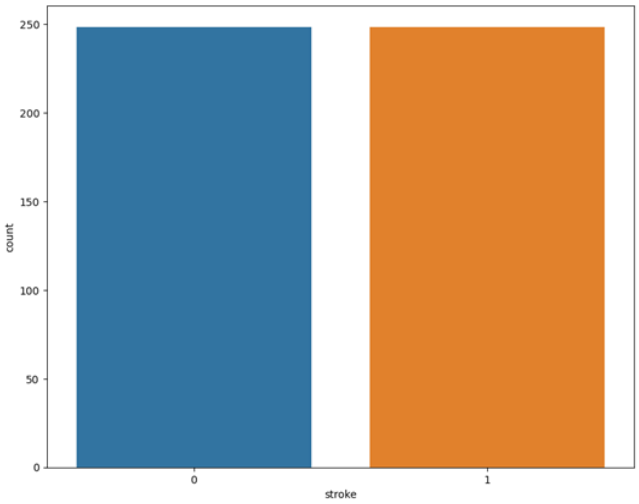


Source: Own work.

From the plot above, it can be inferred that the vast majority of patients who were presented to the hospital did not have a stroke (4733 observations, 95% of the dataset). Meanwhile, individuals who actually had a stroke constitute only 5% of the entire dataset (248 observations). The plot indicates that the dataset is imbalanced, which means there is a potential for inadequate detection of the positive class, i.e., the actual occurrence of a stroke.

To address the dataset imbalance, the undersampling technique was applied, involving the removal of a portion of observations belonging to the majority class. In this case, 4,491 observations belonging to the negative class (stroke = 0) were removed from the dataset. The modified dataset consists of 496 observations, with 248 observations belonging to the negative class, constituting 50% of the entire modified dataset. Below is a plot of the dependent variable "stroke" from the modified dataset.

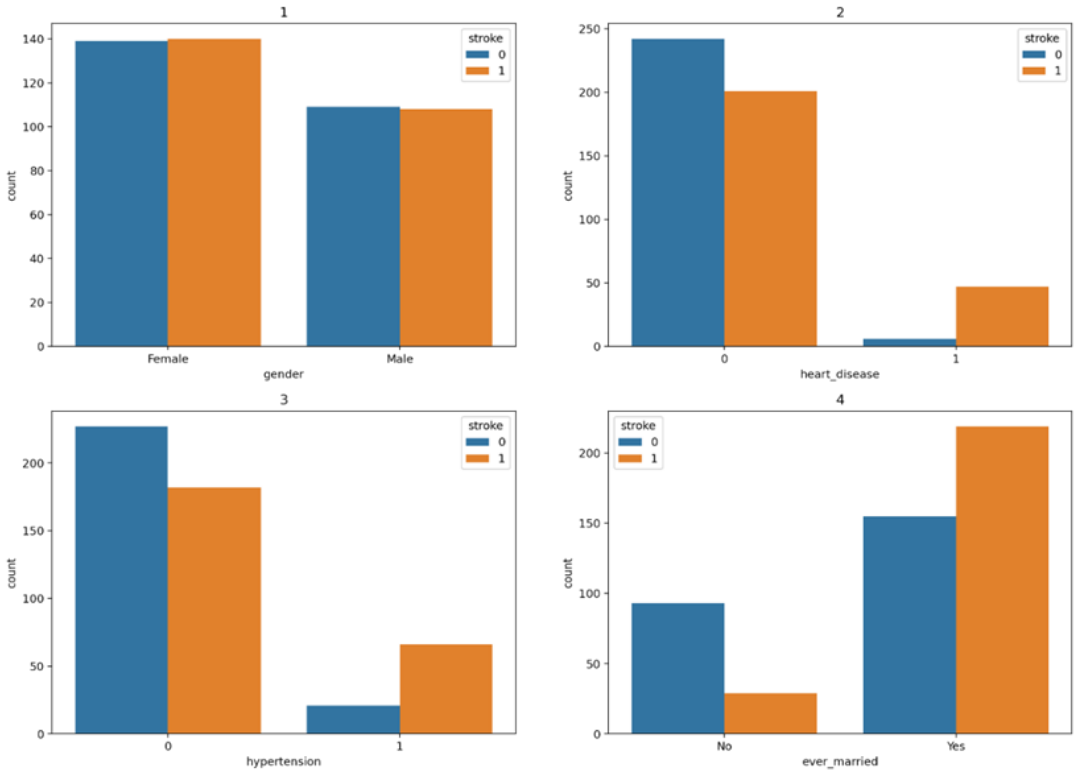
Distribution of the Dependent Variable "stroke" from the Modified Dataset



Source: Own work.

Further data analysis will be conducted on the modified dataset. The orange color corresponds to the variable "stroke" with a result of 1, indicating individuals who actually had a stroke, while the blue color represents individuals for whom medical examinations confirmed the absence of a stroke. Below are charts illustrating the relationships between the variables gender, heart_disease, hypertension, ever_married, and the variable stroke:

Dependency Plots between Explanatory Variables: gender, heart_disease, hypertension, ever_married, and the variable "stroke"



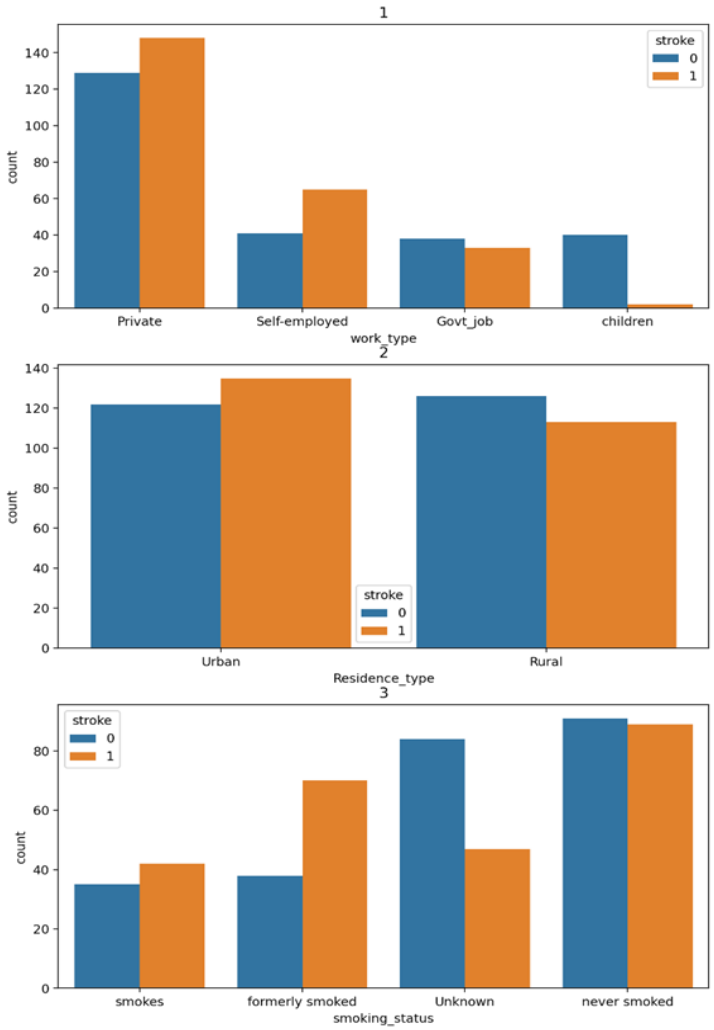
Source: Own work.

The following conclusions can be drawn from the above charts:

1. Every other patient seeking medical care had a stroke. It's worth noting that in the dataset, more women sought medical care than men.
2. The majority of patients who presented to the hospital did not have heart disease. Among patients who had a heart disease, a significant majority also had a stroke.
3. Most patients who sought medical care did not have hypertension. Among patients diagnosed with a stroke, a large portion had arterial hypertension.
4. The majority of patients seeking medical care were married. More than half of the patients who were married had a stroke.

Below are charts illustrating the relationships between the independent variables work_type, residence_type, smoking_status, and the variable stroke:

Dependency Plots between the Dependent Variable "stroke" and Explanatory Variables: work_type, residence_type, smoking_status



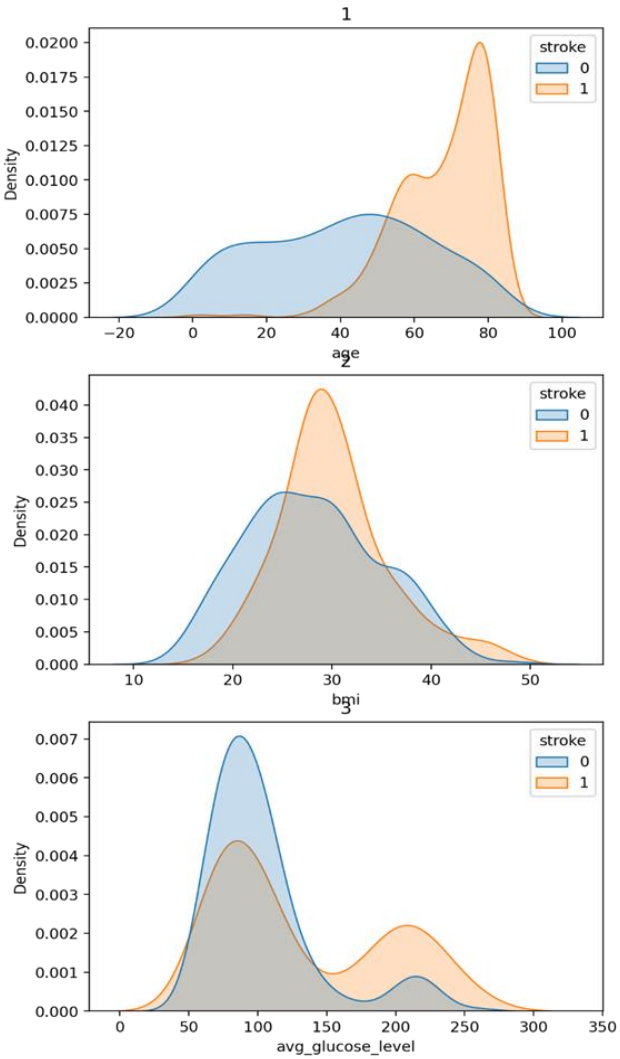
Source: Own work.

The following conclusions can be drawn from the charts above:

1. The majority of patients who presented to the hospital work in the private sector. Moreover, among all groups, the proportion of patients who had a stroke was highest in the private and self-employed groups.
2. Among patients living in urban areas, more than half had a stroke. Among patients living in rural areas, less than half had a stroke.
3. The majority of patients who sought medical care had never smoked. Among patients who currently smoke or have smoked in the past, more than half had a stroke.

For the variables age, bmi, and avg_glucose_level, kernel density estimation plots were created. This is a method for visualizing the distribution of data using a continuous probability density curve.

Dependency Plots between Explanatory Variables: age, bmi, avg_glucose_level, and the Dependent Variable "stroke"



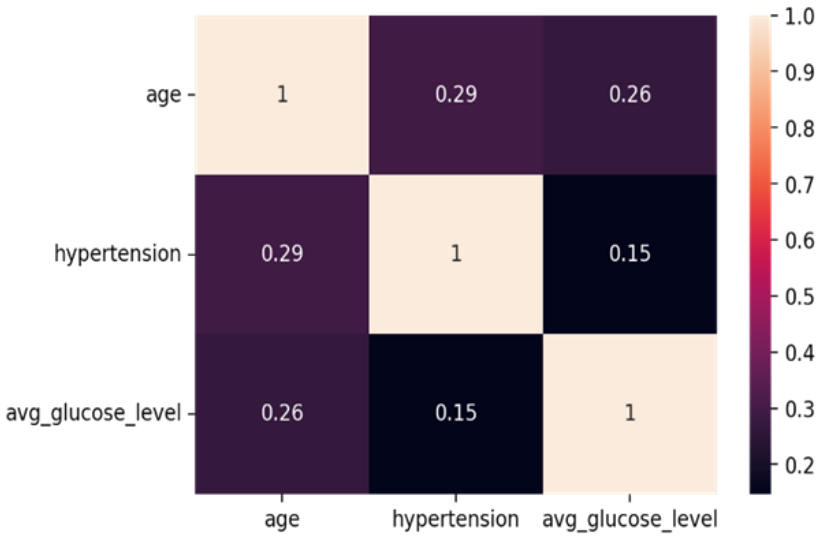
Source: Own work.

The following conclusions can be drawn from the above charts:

1. Almost all patients diagnosed with a stroke fall within the age range of 40 to 90 years, with the vast majority of stroke patients being over 70 years old.
2. The majority of patients who had a stroke were overweight (bmi > 25).
3. The significant majority of patients had a normal blood glucose level (between 60 mg/dL and 100 mg/dL). It is worth noting that among patients diagnosed with a stroke, a lot of people had a glucose level higher than 180 mg/dL. This means that for some patients higher than normal blood glucose level could be a significant factor in determining if they would have a stroke.

As part of the project, the correlation between continuous variables in the dataset was examined to understand the extent to which these variables are correlated.

Correlation between Continuous Variables



Source: Own work.

From the correlation matrix, it can be inferred that the variables are not highly correlated, and all correlations are positive. The highest correlation is between the variables age and hypertension (0.29). The second highest is between age and avg_glucose_level (0.26). This is likely related to the fact that the body's ability to produce insulin decreases with age, leading to an increase in blood glucose levels. The lowest correlation occurs between the variables hypertension and avg_glucose_level (0.15).

To predict the occurrence of a stroke in the project, three machine learning models were trained: AdaBoost, Gradient Boosting, and Support Vector Machine. During the training of each model, the GridSearchCV parameter optimization method was used. The dataset was divided into two parts: the training set, which includes 75% of the dataset, and the test set, which includes 25% of the dataset.

Machine Learning Models

In order to predict the occurrence of a stroke in the report, three machine learning models were trained: AdaBoost, Gradient Boosting, and Support Vector Machine. During the training of each model, the GridSearchCV method was used for parameter optimization. The data set was divided into two parts: a training set, which includes 75% of the dataset, and a test set, comprising 25% of the dataset.

GridSearchCV

One of the important factors influencing the performance of machine learning models is their hyperparameters. Unfortunately, there is no easy way to know the optimal values for hyperparameters in advance, so all possible values must be tested and the predictive power of different model versions compared. Doing this manually can be time-consuming and computationally expensive. GridSearchCV allows for automatic exploration of hyperparameter values for a model to determine their optimal values.

GridSearchCV tests all combinations of user-specified hyperparameters and evaluates the predictive power of the model for each hyperparameter combination using cross-validation, which is used to avoid overfitting of the predictive model, a common occurrence, especially when the dataset is limited. The GridSearchCV method is relatively easy to implement, but it's worth noting that this method can be computationally expensive, especially when a large range of hyperparameter values is provided by the user. In our work we used GridSearchCV to find the best hyperparameter values for each of the three models.

AdaBoost

AdaBoost, short for Adaptive Boosting, is a machine learning algorithm that belongs to the family of ensemble learning methods. The main idea behind AdaBoost is to combine the predictions of multiple weak learners (usually simple and weak classifiers) to create a strong classifier. The algorithm trains the first model by assigning equal weights to all observations. After training the initial AdaBoost model, it assigns greater weights to observations that were misclassified and proceeds to train the next model. The machine learning algorithm will continue to train new models until the lowest prediction error is achieved.

AdaBoost is known for its ability to adapt and improve the performance of weak learners by focusing on the most challenging instances in the dataset. It has applications in both classification and regression problems.

To build the best version of the AdaBoost model, the following hyperparameters were assigned for the GridSearchCV technique:

- learning_rate: [0.1, 0.5, 1] – this hyperparameter determines the contribution of each weak learner to the final prediction. It controls the step size at each iteration while moving toward a minimum of the loss function.
- n_estimators: [50, 100, 200, 300, 400] – this hyperparameter sets the number of weak learners (estimators) to be used in the ensemble. A higher number of estimators can lead to a more powerful model, but it also increases computational cost.
- algorithm: ['SAMME', 'SAMME.R'] – SAMME uses a step function to update the weights of misclassified samples, while SAMME.R uses a real-valued function for updating weights and introduces a renormalization factor to prevent numerical instability. The SAMME.R algorithm is usually faster, achieving a lower prediction error.

The AdaBoost model achieved the highest predictive power on the test set for the following hyperparameter values: learning_rate - 0.5, n_estimators – 50, algorithm - 'SAMME'.

Gradient Boosting

Gradient Boosting is another type of machine learning boosting model that we used in our report. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error. As well as Adaboost, Gradient Boosting adjusts the weights of observations, but instead of focusing on misclassified observations, it fits each new weak learner to the residuals (the differences between the actual and predicted values) of the combined ensemble. This allows Gradient Boosting to correct the errors made by the existing ensemble.

Gradient Boosting can be applied to various types of weak learners, commonly decision trees, but other types of models can be used as well. The flexibility to use different loss functions and weak learner types makes Gradient Boosting a powerful and versatile technique.

For the GridSearchCV technique, the following hyperparameter values were assigned:

- learning_rate: [0.1, 0.5, 1] – the weight applied to each estimator in the Gradient Boosting process,
- n_estimators: [1, 5, 10, 20, 40, 100] – the number of base estimators,
- max_depth: [3, 4, 5, 6] – this hyperparameter represents the maximum depth of each tree (weak learner) in the ensemble.

The Gradient Boosting model demonstrated the highest predictive power for the following hyperparameter values: max_depth – 3, learning_rate – 0.5, n_estimators – 40.

Support Vector Machines

The last model utilized in our report is the Support Vector Machine (SVM). It's a type of supervised learning algorithm used in machine learning to solve classification and regression tasks. The basic idea behind SVM is to find a hyperplane (a decision boundary that divides the data into two classes) in the N-dimensional space (N — the number of features) that best separates different classes in the feature space.

SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection.

The hyperparameters for the SVM model to the GridSearchCV technique were the following:

- C: [0.001, 0.01, 0.1, 0.5, 1] – regularization strength. This hyperparameter imposes a penalty for each misclassified data point. If C is small, the penalty for misclassified points is low. If C is large, the SVM attempts to minimize the number of misclassified observations at the expense of a higher penalty, resulting in a decision boundary with a smaller margin.
- kernel: ['linear', 'rbf'] – the type of kernel used in the algorithm.
- gamma: ['scale', 'auto'] – this hyperparameter determines the distance at which observations begin to influence the decision boundary. For example, low values of gamma imply that distant observations have an impact on the decision boundary. The smaller the gamma, the flatter the decision boundary will be.

The SVM model achieved the highest predictive power on the test set for the following hyperparameter values: C - 0.01, gamma – 'scale', kernel - 'linear'.

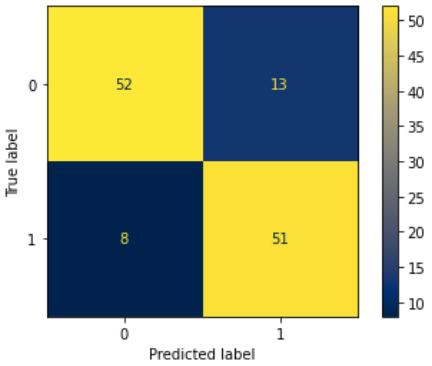
Model Performance Evaluation

After training machine learning models, it is necessary to assess the prediction quality of each model and choose the one with the highest predictive power. For this purpose, models were compared using statistics. The following statistics were utilized in the study along with their interpretations:

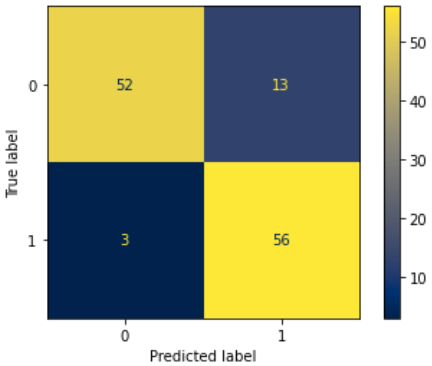
- Accuracy: The ratio of correctly classified observations to all observations.
- Precision: The ratio of correctly classified observations to the positive class to all observations classified as the positive class.
- Match Error Rate (MER): The ratio of incorrectly classified observations to all observations.
- Sensitivity: The ratio of correctly classified observations to the positive class to the sum of observations correctly classified to the positive class and observations incorrectly classified to the negative class.
- F1 Score: The harmonic mean between precision and sensitivity.
- Specificity: The ratio of correctly classified observations to the negative class to all observations classified as the negative class.

Within the report, confusion matrices were created for each model. Below are the confusion matrix plots for the AdaBoost, Gradient Boosting, and SVM models.

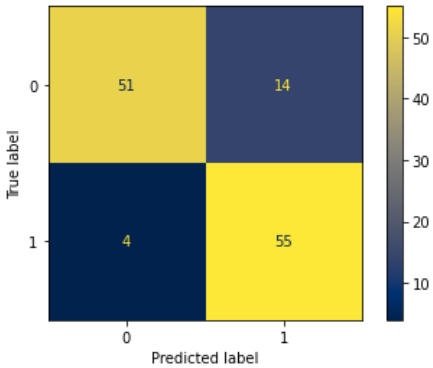
Gradient Boosting



AdaBoost



SVM



Source: Own work.

After training the models on the training set, predictions for the "stroke" variable were made on the test set, and performance metrics were calculated for each model. The table below presents the values of derivative statistics for each model.

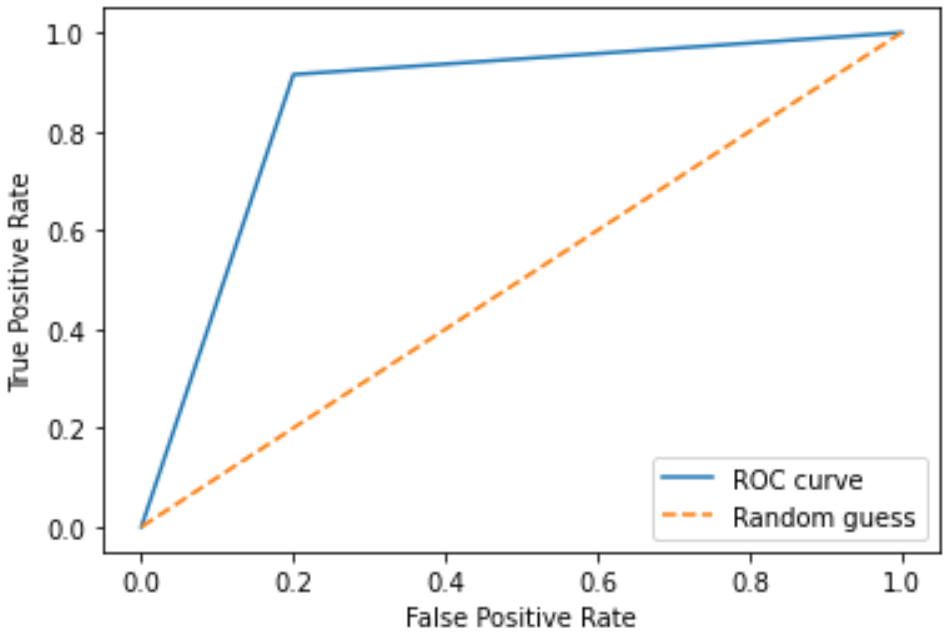
	AdaBoost	Gradient Boosting	SVM
Accuracy	**0.871**	0.831	0.855
Precision	**0.812**	0.797	0.80
MER	**0.129**	0.169	0.145
Sensitivity	**0.949**	0.864	0.91
F1 score	**0.875**	0.829	0.85
Specificity	**0.8**	**0.8**	**0.8**

Source: Own work.

From the above table, it can be inferred that the AdaBoost model achieved the highest levels of accuracy, precision, MER, sensitivity, and F1 score. Specificity has the lowest value for the AdaBoost, Gradient Boosting, and SVM models (0.8). In summary, based on the performance metrics, the AdaBoost model emerges as the best-performing model.

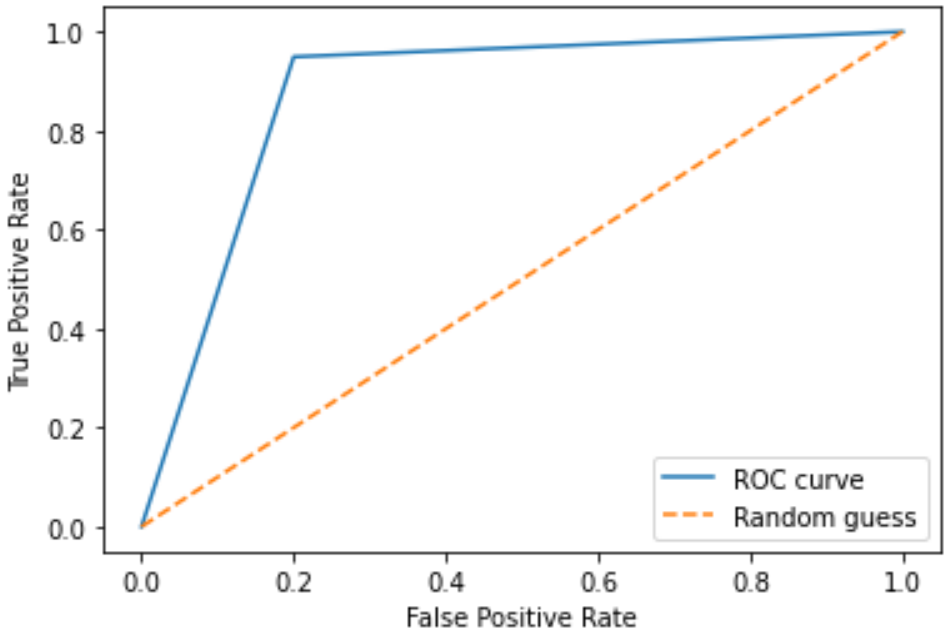
Another tool for evaluating classifier performance is the ROC curve, which illustrates how True Positive Rate changes with False Positive Rate. Below are the ROC curves presented for each model.

ROC curve for Gradient Boosting model



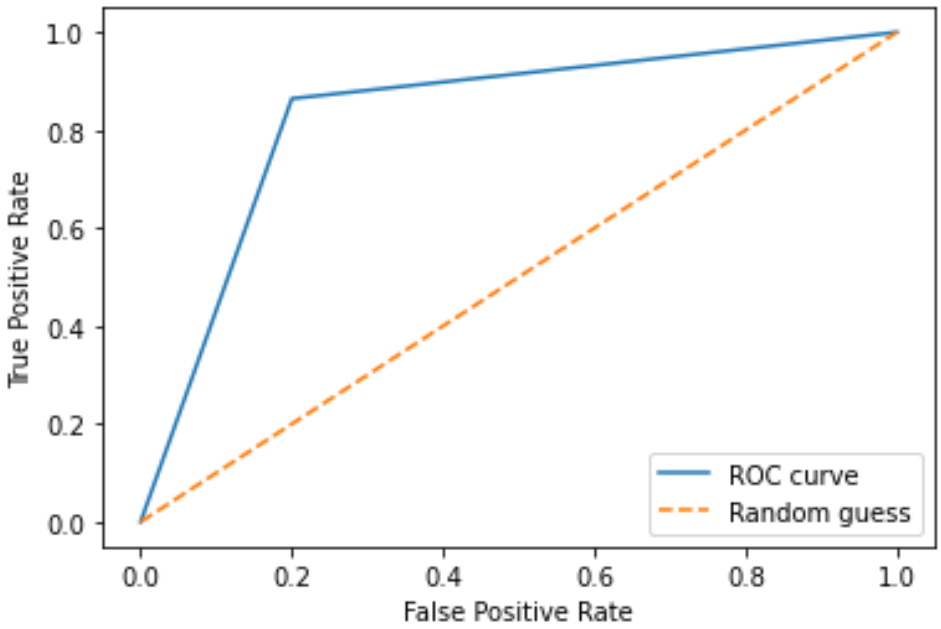
Source: Own work.

ROC curve for AdaBoost model



Source: Own work.

ROC curve for SVM model



Source: Own work.

The values of the AUC (Area Under the Curve) were calculated for each model to assess which model performs best in terms of the area under the ROC curve. The table below presents the AUC ROC statistics for each model.

AdaBoost	Gradient Boosting	SVM
0.875	0.731	0.858

Source: Own work.

According to the table above, the model with the highest Area Under the Curve is AdaBoost (0.875).

Summary

The report aimed to thoroughly explore the dataset and construct multiple ML models to predict the occurrence of a stroke. For this purpose, three machine learning models were built: AdaBoost, Gradient Boosting, and Support Vector Machines. After training the models on the training set, their predictive power was compared based on the confusion matrix, performance metrics, ROC curve, and area under the ROC curve.

In the vast majority of evaluation methods, the AdaBoost model performed well, achieving the highest performance metrics and having the largest area under the ROC curve. The Support Vector Machine (SVM) model also proved to be effective in predicting the occurrence of a stroke, while the Gradient Boosting model was the weakest in terms of predictive power.

It is worth noting that a modified dataset with undersampling was used in the study, consisting of 496 observations. Since undersampling was employed to address data imbalance by removing observations from the majority class, it could potentially result in reduced predictive power of the model on real-world data.