# Optimizing Stroke Detection with Machine Learning Techniques

Neil Samant

[a]*Edison, 08820, NJ, neilsamant@gmail.com*

**Abstract**

This paper addresses the critical question of how various medical factors influence vulnerability to stroke in patients, aiming to develop an effective machine-learning model for stroke detection. Stroke remains a pervasive health concern, with one in six deaths from cardiovascular disease attributed to it in 2021. Every 40 seconds [1] in the United States, an individual experiences a stroke, emphasizing the urgent need for accurate prediction methods. Our study leveraged diverse machine learning models, necessitating dataset balancing prior to model evaluation. Subsequently, a comparison of metrics and fine-tuning of hyperparameters culminated in the selection of the K-Nearest Neighbors (KNN) as the optimal model. The KNN exhibited notable performance metrics, achieving an accuracy score of 80%, a recall score of 89.2%, an F1 score of 83.3%, and a precision score of 78.1%. These metrics signify the model's proficiency in correctly identifying stroke cases, its sensitivity to true positives, and its balance between precision and recall. This study underscores the potential of advanced machine learning models, particularly the KNN, in stroke detection, suggesting its viability for integration into medical consultations. However, the gravity of stroke as a life-threatening condition necessitates rigorous testing on larger and more recent datasets to ensure robustness and reliability in clinical applications. It should also be noted that changing the BMI, average glucose level, and age of the patient affected model performance the most.

## 1. Introduction

In the face of the escalating global burden of stroke, it is critical to understand which specific medical factors can affect stroke vulnerability in patients. For many years researchers have not been able to give a definite answer. Some medical experts claim that age is the most critical factor, while others claim it is prior history and genetics. This is why there needs to be an accurate way of predicting stroke based on certain factors, and it should also be determined which factors are the most important in the final result.

Understanding stroke vulnerability transcends conventional risk assessment, requiring a comprehensive exploration of both numerical metrics such as blood pressure and cholesterol levels and categorical variables like gender, smoking habits, and any prior history of heart disease. The output produced by the models takes the form of labels, either a 1 or a 0, where a 1 denotes if the patient is susceptible to a stroke, and a 0 indicates the opposite.

This research question involves supervised data, as the outputs already have labels of 1 and 0. The problem itself is a classification one, as we are going to determine whether the person (based on inputs) merits a "0" or "1" as their output label. The initial data is categorical, but for it to be compatible with the model, the data was changed by one-hot-encoding to numerical (integers).

## 2. Background and Related Works

In [2], data from the National Health and Nutrition Examination Survey (NHANES) was used with three different sampling methods (without data resampling, with data imputation, and with data resampling) to develop predictive models. They also used four machine learning classifiers and six performance measures to evaluate the performance of the models. The attributes used were slightly different, but there were some similarities and differences. Age and gender were present in our data, but many more advanced attributes were present here, such as albumin, creatinine (both from urine), WBC count, neutrophils, and blood cell counts. There are some advantages of this: more features would mean an increased chance of accuracy, but it could also lead to certain factors confounding each other.

In [3], the method was quite similar to what was used in this paper, using various machine learning algorithms to obtain good accuracy results. The main model used by the researchers in [3] was the AdaBoost model, which was not used in this work. The researchers claim that AdaBoost is a good model that is used for classification. They then proceed to use decision trees and a random forest algorithm. So, they have only used three models. However, the dataset is the exact same as the one used in this paper.

In [4], the research primarily focused on data mining techniques, with the common objective of predicting patients who could be at risk of developing a stroke. However, the classification algorithms used in this paper had no commonalities with what we used: three classification algorithms, namely C4.5, Jrip, and multilayers perceptron (MLP), are used on stroke patient data sets collected from National Guard hospitals in three different cities in Saudi Arabia. This also brings a key distinction between the type of data used: the data used in this paper was from a hospital and medical institution, similar to [2]. The advantage of this is obviously an increased chance of accuracy.

However, the disadvantage is that the results could not necessarily be generalized to the population outside Saudi Arabia.

## 3. Dataset

### *3.1. Key Information About the Dataset*

The dataset is from Kaggle, provided by user *fedesoriano* [5], and titled "Stroke Prediction Dataset". This dataset has 5110 rows and 12 columns. The first 11 columns correspond to features that have a weighted importance in the final output, which is in the 12th column. The first 11 columns, respectively, are "id", "gender", "age", "hypertension", "heart disease", "ever_married", "work_type", "Residence_type", "avg_glucose_level", "bmi", and "smoking_status". The 12th column is aptly titled "stroke", and only takes on binary values. The features of the dataset can be summarized in Table I below. Value types that include "(Binary)" indicate that they can only take on two values.

**Table I. Stroke Dataset Columns.**

| Attribute Name | Type of Values | Description | Possible Values |
|---|---|---|---|
| 1. **id** | Integer | A unique identification number for each person that has no bearing on the result. | Any random 4-5 digit positive number |
| 2. **gender** | String | The gender of the patient. | **"Male", "Female"** |
| 3. **age** | Integer | The age of the patient. | Any reasonable positive number |
| 4. **hypertension** | Integer (Binary) | Does the patient have hypertension? | **0:** no hypertension<br>**1:** hypertension |
| 5. **heart_disease** | Integer (Binary) | Does the patient have any heart disease? | **0:** no heart disease<br>**1:** heart disease |
| 6. **ever_married** | String (Binary) | Was the patient ever married? | **"No" or "Yes"** |
| 7. **work_type** | String | What type of job does the patient have? | **"Private", "Self-employed", "children", "Govt_job", "never_worked"** |
| 8. **Residence_type** | String (Binary) | Which type of area does the patient live in? | **"Rural" or "Urban"** |
| 9. **avg_glucose_level** | Float | What is the average glucose level (mg/dL) of the patient? | Any reasonable positive number |
| 10. **bmi** | Float | What is the body mass index (BMI) of the patient? | Any reasonable positive value, or "NaN" if not available |
| 11. **smoking_status** | String | What best describes the patient's smoking history? | **"formerly smoked", "never smoked", "smokes" or "Unknown" - denotes unavailability** |
| 12. **stroke** | Integer (Binary) | Did this patient have a stroke? | **0:** did not have a stroke<br>**1:** stroke |

### *3.2. Data Preprocessing and Handling Data Imbalances*

Firstly, it should be noted that the data had a heavy imbalance toward patients that did not have a stroke. To put this into perspective, out of the 5110 patient entries in the original dataset, only 249 had been listed as having a stroke. This would mean that if the model were trained on the original data, the model would only be able to accurately predict if someone could not have a stroke, and would mostly give zeros as the output labels. To fix this, the approach in [6] was used. They used undersampling and brought the number of samples with the output as "0" to equal 249. This would ensure that the model will be trained on an equal set of stroke and non-stroke patients. However, this does involve tampering with the original dataset, generally considered a bad practice as it may undermine the credibility of the model (being trained on complete real-world data). Eventually, the "resample" [7] function from "sklearn.utils" was used.

To prepare the data for model analysis, the first column (id) was dropped. This is simply a unique identifier that has no correlation to the output whatsoever. Then, null values had to be dealt with in the "bmi" column. This was done by simply filling those values with the mean of the existing "bmi" column, which would not change the column's numerical statistics. After this, categorical data had to be converted to numerical data to keep the dataset consistent.

### *3.3. Label Encoding*

As the machine is usually trained in numbers, the strings have to be converted into integers. There are five columns in the dataset that have string values. There was no predetermined key to encode the categorical columns. Rather, this was done by the "LabelEncoder" [7] function from "sklearn.preprocessing". The final encoding after LabelEncoder was used can be seen in Table II.

**Table II. Encoded Columns After the Use of LabelEncoder.**

| Column Name | Old Key | New Key |
|---|---|---|
| **gender** | **"Male", "Female"** | **Female:** 0<br>**Male:** 1 |
| **ever_married** | **"No", "Yes"** | **No:** 0<br>**Yes:** 1 |
| **work_type** | **"Private", "Self-employed", "children", "Govt_job", "never_worked"** | **"Govt_job":** 0<br>**"never_worked":** 1<br>**"private":** 2<br>**"Self-employed":** 3<br>**"children":** 4 |
| **Residence_type** | **"Rural", "Urban"** | **Rural:** 0<br>**Urban:** 1 |
| **smoking_status** | **"formerly smoked", "never smoked", "smokes" or "Unknown" - denotes unavailability** | **Unknown:** 0<br>**formerly smoked:** 1<br>**never smoked:** 2<br>**smokes:** 3 |

## 4. Methodology/Models

### *4.1. Splitting the Data*

The new dataset was split into train and test sets with 80% going into the training set and 20% going into the testing set. It should be noted that this is a common split used in the industry.

However, before this, the "stroke" column needed to be dropped from the new dataset and assigned the value "Y". The remaining dataset would be assigned the value "X". This is done for obvious reasons, to avoid the model being trained on the outputs as well. After this, the following classification models were used: Decision Tree, Logistic Regression, K-Nearest Neighbors, Random Forest, XG Boost, Support Vector Machine, and Naive Bayes Classification.

### *4.2. Classification Models*

**Decision Tree**

Decision trees can be used for both classification and regression problems. In the context of this problem, however, only the former is applicable. They work by recursively partitioning the dataset based on the most significant features, creating a tree-like structure of decision rules. At each node, the algorithm chooses the feature that best splits the data. The resulting tree can be interpreted, making decision trees valuable for understanding the decision-making process. [8]

**Logistic Regression**

Logistic regression is used for binary classification problems. It models the probability of the outcome belonging to a particular class using the logistic function. The model is trained by adjusting the weights assigned to each feature to maximize the likelihood of the observed outcomes. [9]

**K-Nearest Neighbors**

K-Nearest Neighbors, abbreviated as KNN, classifies a new data point based on the majority class of its k-nearest neighbors in the feature space. The choice of k influences the model's sensitivity to local variations; smaller k values lead to more flexible, potentially noisy predictions, while larger k values may smooth out local irregularities. [10]

**Random Forest**

Random Forest constructs many decision trees during training, each based on a random subset of features and data instances. Unlike a standalone decision tree, Random Forest provides more robust and accurate predictions, making it particularly effective in handling large datasets with high dimensionality. The ensemble nature of Random Forest also allows it to capture complex relationships in the data and reduces the risk of model bias associated with individual trees.

**XG Boost**

XG Boost (Extreme Gradient Boosting) builds a series of decision trees sequentially, with each tree correcting the errors of the previous ones. XG Boost incorporates regularization terms to control model complexity and prevent overfitting, and it uses a clever optimization technique for efficient computation. [11]

**Support Vector Machine**

Support Vector Machine (SVM) aims to maximize the margin between classes, with the margin defined as the distance between the hyperplane and the nearest data points of each class. It needs to be determined if this data is linearly separable. [12]

**Naive Bayes Classification**

This was the last model tried due to it being used in [4]. Naive Bayes is a probabilistic classifier based on Bayes' theorem. It assumes that features are conditionally independent given the class label, which is a naive assumption but often holds in practice. Naive Bayes calculates the probability of each class for a given set of features and predicts the class with the highest probability. It is computationally efficient and works well for text classification and other tasks with a large number of features.

## 5. Results and Discussions

### *5.1. Types of Metrics*

**Accuracy**

Accuracy reflects the overall correctness of a classification model by measuring the ratio of correctly predicted instances to the total number of instances. However, in scenarios with imbalanced datasets, where one class significantly outnumbers the other, a high accuracy may be misleading as the model might primarily predict the majority class. Accuracy does not account for the class distribution and can be disproportionately influenced by the dominance of the majority class, potentially resulting in a misleadingly high accuracy despite low performance in detecting the minority class. This was the case in the original dataset, where models trained on that mostly gave "0" as the output. [13]

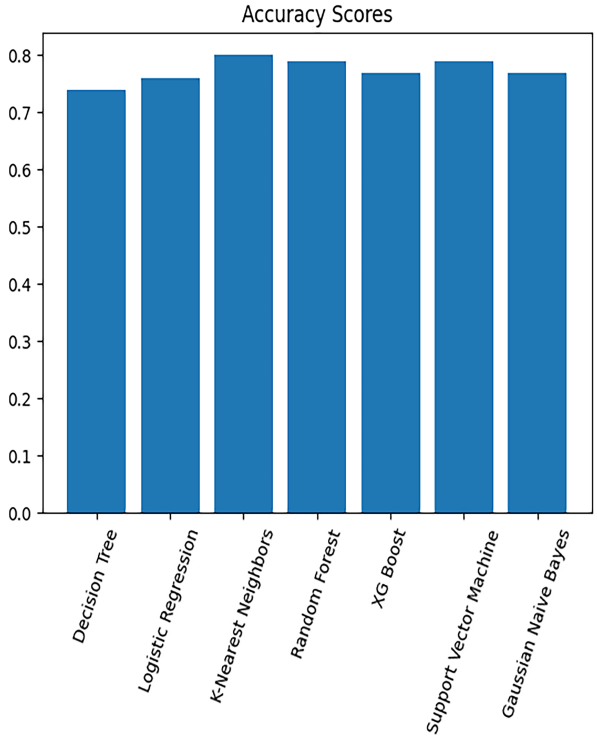


**Figure 1: The accuracy scores of each model.**

**Precision**

Precision evaluates the accuracy of the positive predictions made by the model, representing the ratio of true positives to the sum of true positives and false positives. It provides insights into how well the model identifies true positive instances. [13]

The equation for Precision can be denoted by:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

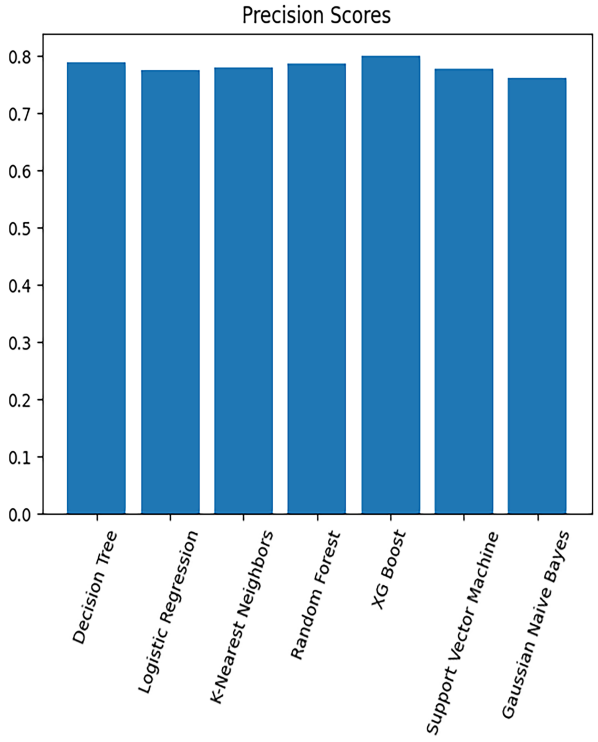


**Figure 2: The precision scores of each model.**

**Recall**

Recall, also known as sensitivity or true positive rate, assesses the model's ability to capture all positive instances. It is the ratio of true positives to the sum of true positives and false negatives, offering insights into the model's sensitivity to the positive class. [13]

The equation for Recall can be denoted by:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

**F1**

The F1 score is the harmonic mean of precision and recall, providing a balanced metric that considers both false positives and false negatives. It is particularly useful when there is an uneven class distribution, as it penalizes models that favor one class over the other. [13]

The equation for F1 can be denoted by:

$$F1 = 2 \cdot \frac{(Precision \cdot Recall)}{Precision + Recall}$$

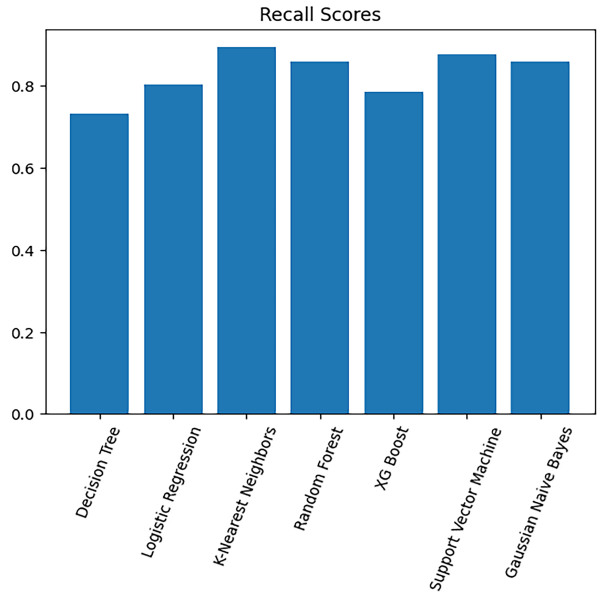


**Figure 3: The recall scores of each model.**

### *5.2. Individual Model Performance*

**Decision Tree**

The Decision Tree had an accuracy score of 74.0%, a precision score of 78.4%, a recall score of 73.2%, and an F1 score of 75.9%.

**Logistic Regression**

The Logistic Regression had an accuracy score of 76.0%, a precision score of 77.6%, a recall score of 80.3%, and an F1 score of 78.9%.

**K-Nearest Neighbors**

The K-Nearest Neighbors had an accuracy score of 80.0%, a precision score of 78.1%, a recall score of 89.3%, and an F1 score of 83.3%.

**Random Forest**

The Random Forest had an accuracy score of 79.0%, a precision score of 78.7%, a recall score of 85.7%, and an F1 score of 82.1%.

**XG Boost**

The XG Boost had an accuracy score of 77.0%, a precision score of 80.0%, a recall score of 78.6%, and an F1 score of 79.3%.

**Support Vector Machine**

The Support Vector Machine had an accuracy score of 79.0%, a precision score of 77.8%, a recall score of 87.5%, and an F1 score of 82.4%.

**Naive Bayes Classification**

The Naive Bayes Classification had an accuracy score of 77.0%, a precision score of 76.2%, a recall score of 85.7%, and an F1 score of 80.7%.

Overall, it can be concluded that the K-Nearest Neighbors model was the best-performing model. It had the highest accuracy, recall, and F1 scores, while it was only 1.9% less than the highest precision score.

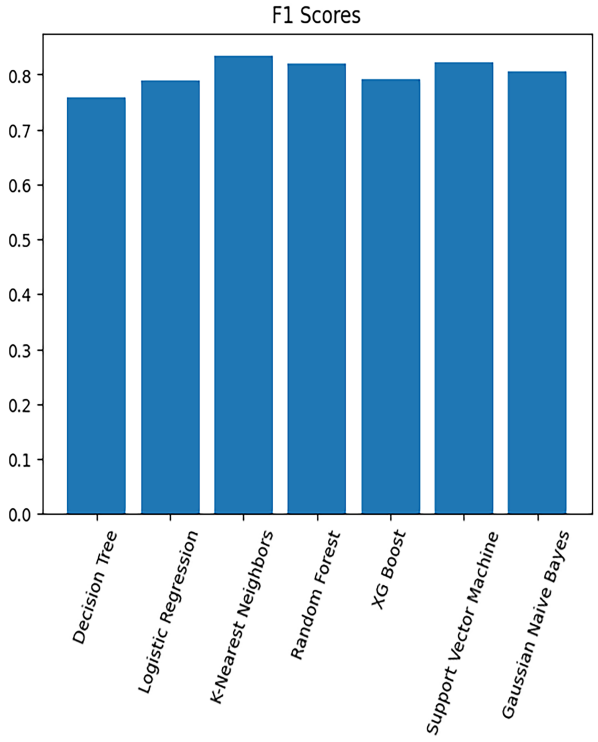


**Figure 4: The F1 scores of each model.**

### *5.3. Other Key Metrics*

**Receiver Operating Characteristic Curves**

Receiver Operating Characteristic Curves, also known as ROC curves, graphically represent the performance of a binary classification model [14] across various discrimination thresholds. The curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate, offering a visual tool for selecting an appropriate threshold based on the specific needs of the classification problem.

**Confusion Matrices**

A confusion matrix is a tabular representation of a classification model's predictions [15], detailing the counts of true positive, true negative, false positive, and false negative instances. They can be used to manually calculate precision, recall, and F1 scores.

As the confusion matrix shows, the K-Nearest neighbors model predicts 50 out of the 56 cases of stroke correctly. It also mistakenly predicts that 14 patients will have a stroke when their true label suggests they will not have a stroke. This is not as much of a concern for the 14 patients, but it is more of a problem for the 6 who were mistakenly given a "negative" result. Even the slightest error in the medical field can lead to the loss of lives.

This confusion matrix should be compared to the one in [3]. Other than the AdaBoost model, the second column was all zeros or extremely low numbers, which means that those models hardly predicted a stroke and mostly gave "0" as the output. That is a result of directly dealing with the imbalanced data set with minimal data preprocessing.

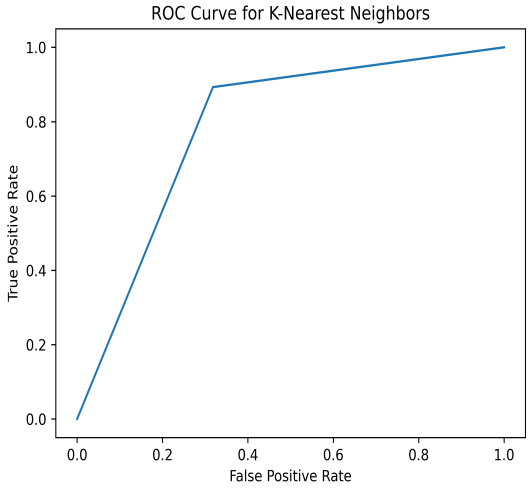


**Figure 5: The ROC Curve for the K-Nearest Neighbors Model.**

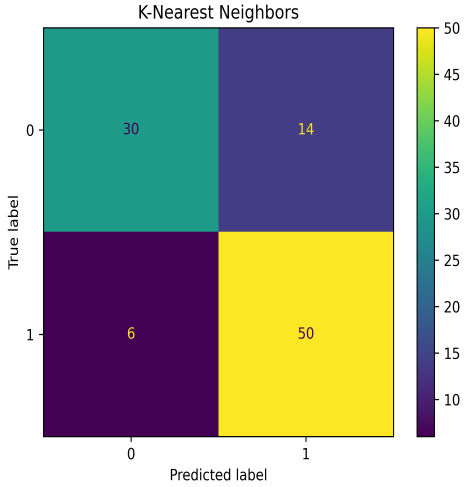


**Figure 6: The Confusion Matrix for the K-Nearest Neighbors model.**

## 6. Conclusions

Stroke is a critical medical condition that continues to affect millions of people worldwide. Optimal detection and prevention would lead to saving countless lives. Using a machine learning classification model can help in the early prediction of stroke and reduce its severe impact in the future. This paper shows the performance of various machine learning algorithms in successfully predicting stroke based on multiple attributes such as smoking status, heart disease, hypertension, and physical factors such as BMI and blood sugar levels.

For an imbalanced dataset such as this one, an accuracy score of 80% can be considered more than a satisfactory result, especially when dealing with a dataset of medical data. This category of data is known to have many anomalies and fluctuations. From the original dataset, all of the models recorded accuracy scores higher than 90%, but the recall, F1, and precision scores were quite poor, some even zero. The K-Nearest Neighbors model performed the best, with an accuracy score

of 80.0%, a precision score of 78.1%, a recall score of 89.3%, and an F1 score of 83.3%. This was achieved with the default hyperparameters.

In future research, fine-tuning hyperparameters and exploring alternative machine learning algorithms beyond the K-Nearest Neighbors model may further optimize performance metrics such as precision, recall, and F1 score. These efforts could contribute to developing a more robust and clinically applicable predictive tool for early stroke detection, thereby improving patient outcomes and saving lives.

**Acknowledgements**

This work was conducted through the Inspirit AI 1:1 Mentorship Program and Inspirit AI supported me by matching me with the mentor and helping me select a research project. Specifically, I would like to thank researcher Ye Wang from Inspirit AI for guiding me through this project and the specific techniques used for model and data analysis, Finally, I would like to thank user "fedesoranio" on kaggle.com for providing the dataset for fair use.

**References**

[1] "Stroke Facts," Centers for Disease Control and Prevention, https://www.cdc.gov/stroke/facts.htm.
[2] E. M. Alanazi, A. Abdou, and J. Luo, "Predicting risk of stroke from lab tests using Machine Learning Algorithms: Development and evaluation of Prediction Models," JMIR formative research, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8686476/.
[3] B. Imran, E. Wahyudi, A. Subki, S. Salman, and A. Yani. "Classification of stroke patients using data mining with adaboost, decision tree and random forest models." ILKOM Jurnal Ilmiah 14, no. 3 (2022): 218-228.
[4] O. Almadani, and R. Alshammari. "Prediction of stroke using data mining classification techniques." International Journal of Advanced Computer Science and Applications 9, no. 1 (2018).
[5] "fedesoranio", "Stroke Prediction Dataset," Kaggle, https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset.
[6] G. Sailasya, and G. L. Aruna Kumari. "Analyzing the performance of stroke prediction using ML classification algorithms." International Journal of Advanced Computer Science and Applications 12.6 (2021)
[7] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[8] "What is a decision tree with examples," What is A Decision Tree with Examples — EdrawMax Online, https://www.edrawmax.com/decision-tree/.
[9] M. Nandu, "Machine learning using logistic regression in python with code," Medium, https://blog.goodaudience.com/machine-learning-using-logistic-regression-in-python-with-code-ab3c7f5f3bed.
[10] A. Kumar, "K-Nearest Neighbors (KNN) python examples," Analytics Yogi, https://vitalflux.com/k-nearest-neighbors-explained-with-python-examples/.
[11] "XGBoost documentation," XGBoost Documentation - xgboost 2.0.2 documentation, https://xgboost.readthedocs.io/en/stable/.
[12] V. Kanade, "All you need to know about support vector machines," Spiceworks, https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/.
[13] K. P. Shung, "Accuracy, precision, recall or F1?," Medium, https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.
[14] A. Bhandari, "Guide to AUC ROC curve in machine learning: What is specificity?," Analytics Vidhya, https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/.
[15] S. Narkhede, "Understanding confusion matrix," Medium, https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.