

**Final-Term Project Report**

**Supervised Data Mining  
(Binary Classification)**

**NAME:** Yashwanth Reddy Boddireddy

**NJIT UCID:** yb244  
**Email Address:** yb244@njit.edu  
**Professor:** Yasser Abduallah  
CS 634 - 104 Data Mining

Contents

[Abstract: 3](#_Toc164625073)

[Introduction 3](#_Toc164625074)

[Project Workflow 3](#_Toc164625075)

[Data Collection 3](#_Toc164625076)

[Data Cleaning and Preparation 4](#_Toc164625077)

[Model Training 10](#_Toc164625078)

[Function for calculation of metrics using confusion matrix. 11](#_Toc164625079)

[Finding Best parameters for models using GridSearchCV. 11](#_Toc164625080)

[Function to create LSTM model. 12](#_Toc164625081)

[Model Training with Kfold. 12](#_Toc164625082)

[Model Evaluation. 13](#_Toc164625083)

[Calculating average metrics for each model after all folds 18](#_Toc164625084)

[Computing ROC curve and ROC area for each model. 19](#_Toc164625085)

[Results: 20](#_Toc164625086)

[Conclusion 21](#_Toc164625087)

[Recommendations: 21](#_Toc164625088)

[Steps to run the Program: 22](#_Toc164625089)

[STEP 1: Cloning the Repository 22](#_Toc164625090)

[STEP 2: Create Virtual Environment. 22](#_Toc164625091)

[STEP 3: Running the Notebook Locally 22](#_Toc164625092)

[Alternative for STEP 3: Running the Notebook on Google Colab 23](#_Toc164625093)

[References and Links: 23](#_Toc164625094)

# Abstract:

This project focuses on the prediction of stroke occurrence based on various health indicators and personal characteristics using a dataset from a healthcare study. The dataset includes 5,110 instances with features such as age, gender, type of work, residence type, average glucose level, body mass index (BMI), and smoking status. Using Random Forest, a deep learning model (e.g., LSTM), and another algorithm from a predefined list (e.g., Support Vector Machines), we aim to classify individuals based on their stroke risk. The models' performances are compared using accuracy metrics calculated through 10-fold cross-validation. The project highlights significant predictors of stroke and evaluates the effectiveness of different classification algorithms in predicting health outcomes.

# Introduction

Stroke remains one of the leading causes of death and long-term disability worldwide. Early prediction and identification of high-risk individuals can significantly enhance interventions and prevent severe outcomes. This project utilizes a healthcare dataset that encompasses critical attributes such as medical history, biometric data, and lifestyle factors, which are pivotal in assessing stroke risk. The dataset from the healthcare sector provides a comprehensive base for applying machine learning techniques to predict stroke occurrence. This study aims to implement and compare three distinct classification algorithms: Random Forest, a selected deep learning algorithm from the LSTM family, and a classical machine learning algorithm such as Support Vector Machines. The comparison focuses on their predictive accuracy and ability to handle imbalanced datasets typical of medical data, where stroke events are relatively rare compared to non-stroke instances. This initiative is part of a broader effort to integrate predictive analytics into healthcare to facilitate early diagnosis and personalized medicine approaches.

# Project Workflow

## Data Collection

The dataset utilized in this project was obtained from [Stroke Prediction Dataset (kaggle.com)](https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset), featuring data on 5,110 individuals. It includes various attributes like gender, age, hypertension, heart disease, marital status, work type, residence type, average glucose level, body mass index (BMI), smoking status, and stroke occurrence.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Data Cleaning and Preparation

The dataset was pre-processed to manage missing values, particularly in the BMI attribute. Categorical variables were encoded appropriately to facilitate analysis. Data normalization was performed on continuous variables such as age and average glucose level to standardize their ranges, enhancing the model's performance.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen with text

Description automatically generated

**Exploratory Data Analysis (EDA)**

An initial analysis was conducted to understand the distribution and relationship of variables. Visualizations such as histograms, box plots, and scatter plots were utilized to identify patterns and anomalies in the data, which informed the feature selection process.

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

A graph of different types of blood

Description automatically generated with medium confidence

**Age Distribution:** The first chart shows a fairly uniform distribution of ages, with a slight increase around the 50-60 age bracket. This could be indicative of a population with a wide range of ages represented in the data, which is less likely to be from a specific group (like school students or retirees) and more likely to be representative of a general population.

**Hypertension:** The distribution indicates that a large majority of the individuals in the dataset do not have hypertension, with only a small fraction having this condition.

**Heart Disease:** Similarly to hypertension, the chart for heart disease indicates that few individuals have heart disease compared to those who do not.

**Marital Status (Ever Married):** This chart suggests that most individuals in the dataset have been married at least once.

**Residence Type:** The data seems to be almost evenly split between two types of residence, possibly urban and rural.

**Average Glucose Level:** This histogram shows a right-skewed distribution, meaning most individuals have glucose levels on the lower side, but there is a long tail towards the higher glucose levels. The peak is around the 75-100 mg/dL range, which is considered normal.

**BMI:** This histogram also appears right skewed, with the majority of individuals having a BMI in the range considered normal or overweight (around 25-30), but fewer individuals in the underweight and obese categories.

**Stroke:** Like hypertension and heart disease, the vast majority of individuals have not experienced a stroke.

A black background with white text

Description automatically generated

A group of blue dots

Description automatically generated

**Average Glucose Level:** The distribution of average glucose levels is skewed right, with a peak in the normal range, but some individuals have extremely high levels. When compared to strokes, higher glucose levels appear to have more instances of strokes (orange dots).

**Age:** The age distribution is broad with a slight skew to the right, suggesting a population that includes elderly individuals. The scatter plots indicate a clear trend: the likelihood of stroke increases with age.

**BMI:** The BMI distribution is also right skewed, with most data points falling in the range considered normal or overweight. The relationship between BMI and stroke is not as clear from this plot, but there are some instances of strokes across all BMI values.

**Relationship Between Variables:**

**Age and Average Glucose Level:** There is no distinct pattern or correlation visible in the scatter plot between age and average glucose level. This suggests that within this dataset, there is no clear trend between age and glucose level.

**Age and BMI:** Again, there is no clear correlation between age and BMI from the scatter plot.

**Average Glucose Level and BMI:** The scatter plot does not show a clear correlation between average glucose levels and BMI.

**Stroke Occurrences:**

**Strokes and Average Glucose Level:** There is a noticeable cluster of strokes (orange dots) at higher glucose levels.

**Strokes and Age:** Strokes are more common in older age groups.

**Strokes and BMI:** Strokes are scattered across the range of BMI, indicating that within this dataset, BMI is not a clear-cut predictor for strokes.

A screen shot of a computer program

Description automatically generated

A graph with text and numbers

Description automatically generated with medium confidence

## Model Training

Three models were trained:

1. **Random Forest:** A robust ensemble technique that mitigates overfitting and is known for its high accuracy in classification tasks.
2. **Deep Learning Model (e.g., LSTM):** Chosen for its ability to model sequences, which is beneficial for data where temporal patterns might influence the outcome.
3. **Support Vector Machines (SVM):** Selected for its effectiveness in high-dimensional spaces and its ability to model non-linear decision boundaries through kernel functions.

Each model was subjected to 10-fold cross-validation to ensure the robustness of the findings and to mitigate the model’s performance variability.

### Function for calculation of metrics using confusion matrix.

A screen shot of a computer program

Description automatically generated

### Finding Best parameters for models using GridSearchCV.

A computer screen with text and images

Description automatically generated

A screen shot of a computer program

Description automatically generated

### Function to create LSTM model.

A computer screen with colorful text

Description automatically generated

### Model Training with Kfold.

A screen shot of a computer

Description automatically generated

A computer screen with white and green text

Description automatically generated

A computer screen shot of text

Description automatically generated

## Model Evaluation.

The models were evaluated using metrics derived from the confusion matrix, including True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Other metrics calculated included the False Positive Rate (FPR), False Negative Rate (FNR), True Skill Statistics (TSS), and Heidke Skill Score (HSS). These metrics were calculated manually to deepen understanding and ensure precise evaluation.

Iteration 1:

----- Metrics for all Algorithms in Iteration 1 -----

RF SVM LSTM

TP 2.00 5.00 9.00

TN 467.00 430.00 431.00

FP 16.00 53.00 52.00

FN 26.00 23.00 19.00

TPR 0.07 0.18 0.32

TNR 0.97 0.89 0.89

FPR 0.03 0.11 0.11

FNR 0.93 0.82 0.68

Precision 0.11 0.09 0.15

F1\_measure 0.09 0.12 0.20

Accuracy 0.92 0.85 0.86

Error\_rate 0.08 0.15 0.14

BACC 0.52 0.53 0.61

TSS 0.04 0.07 0.21

HSS 0.05 0.05 0.14

16/16 [==============================] - 0s 2ms/step

Iteration 2:

----- Metrics for all Algorithms in Iteration 2 -----

RF SVM LSTM

TP 7.00 8.00 7.00

TN 454.00 432.00 435.00

FP 23.00 45.00 42.00

FN 27.00 26.00 27.00

TPR 0.21 0.24 0.21

TNR 0.95 0.91 0.91

FPR 0.05 0.09 0.09

FNR 0.79 0.76 0.79

Precision 0.23 0.15 0.14

F1\_measure 0.22 0.18 0.17

Accuracy 0.90 0.86 0.86

Error\_rate 0.10 0.14 0.14

BACC 0.58 0.57 0.56

TSS 0.16 0.14 0.12

HSS 0.17 0.11 0.10

16/16 [==============================] - 1s 3ms/step

Iteration 3:

----- Metrics for all Algorithms in Iteration 3 -----

RF SVM LSTM

TP 4.00 11.00 9.00

TN 467.00 440.00 439.00

FP 17.00 44.00 45.00

FN 23.00 16.00 18.00

TPR 0.15 0.41 0.33

TNR 0.96 0.91 0.91

FPR 0.04 0.09 0.09

FNR 0.85 0.59 0.67

Precision 0.19 0.20 0.17

F1\_measure 0.17 0.27 0.22

Accuracy 0.92 0.88 0.88

Error\_rate 0.08 0.12 0.12

BACC 0.56 0.66 0.62

TSS 0.11 0.32 0.24

HSS 0.13 0.21 0.16

16/16 [==============================] - 1s 2ms/step

Iteration 4:

----- Metrics for all Algorithms in Iteration 4 -----

RF SVM LSTM

TP 2.00 4.00 7.00

TN 464.00 434.00 432.00

FP 22.00 52.00 54.00

FN 23.00 21.00 18.00

TPR 0.08 0.16 0.28

TNR 0.95 0.89 0.89

FPR 0.05 0.11 0.11

FNR 0.92 0.84 0.72

Precision 0.08 0.07 0.11

F1\_measure 0.08 0.10 0.16

Accuracy 0.91 0.86 0.86

Error\_rate 0.09 0.14 0.14

BACC 0.52 0.53 0.58

TSS 0.03 0.05 0.17

HSS 0.04 0.03 0.10

16/16 [==============================] - 1s 3ms/step

Iteration 5:

----- Metrics for all Algorithms in Iteration 5 -----

RF SVM LSTM

TP 2.00 4.00 5.00

TN 480.00 434.00 443.00

FP 16.00 62.00 53.00

FN 13.00 11.00 10.00

TPR 0.13 0.27 0.33

TNR 0.97 0.88 0.89

FPR 0.03 0.12 0.11

FNR 0.87 0.73 0.67

Precision 0.11 0.06 0.09

F1\_measure 0.12 0.10 0.14

Accuracy 0.94 0.86 0.88

Error\_rate 0.06 0.14 0.12

BACC 0.55 0.57 0.61

TSS 0.10 0.14 0.23

HSS 0.09 0.05 0.09

16/16 [==============================] - 1s 2ms/step

Iteration 6:

----- Metrics for all Algorithms in Iteration 6 -----

RF SVM LSTM

TP 2.00 3.00 5.00

TN 465.00 436.00 440.00

FP 22.00 51.00 47.00

FN 22.00 21.00 19.00

TPR 0.08 0.12 0.21

TNR 0.95 0.90 0.90

FPR 0.05 0.10 0.10

FNR 0.92 0.88 0.79

Precision 0.08 0.06 0.10

F1\_measure 0.08 0.08 0.13

Accuracy 0.91 0.86 0.87

Error\_rate 0.09 0.14 0.13

BACC 0.52 0.51 0.56

TSS 0.04 0.02 0.11

HSS 0.04 0.01 0.07

16/16 [==============================] - 1s 3ms/step

Iteration 7:

----- Metrics for all Algorithms in Iteration 7 -----

RF SVM LSTM

TP 3.00 6.00 8.00

TN 475.00 433.00 422.00

FP 16.00 58.00 69.00

FN 17.00 14.00 12.00

TPR 0.15 0.30 0.40

TNR 0.97 0.88 0.86

FPR 0.03 0.12 0.14

FNR 0.85 0.70 0.60

Precision 0.16 0.09 0.10

F1\_measure 0.15 0.14 0.16

Accuracy 0.94 0.86 0.84

Error\_rate 0.06 0.14 0.16

BACC 0.56 0.59 0.63

TSS 0.12 0.18 0.26

HSS 0.12 0.09 0.11

16/16 [==============================] - 1s 2ms/step

Iteration 8:

----- Metrics for all Algorithms in Iteration 8 -----

RF SVM LSTM

TP 7.00 9.00 10.00

TN 461.00 430.00 426.00

FP 17.00 48.00 52.00

FN 26.00 24.00 23.00

TPR 0.21 0.27 0.30

TNR 0.96 0.90 0.89

FPR 0.04 0.10 0.11

FNR 0.79 0.73 0.70

Precision 0.29 0.16 0.16

F1\_measure 0.25 0.20 0.21

Accuracy 0.92 0.86 0.85

Error\_rate 0.08 0.14 0.15

BACC 0.59 0.59 0.60

TSS 0.18 0.17 0.19

HSS 0.20 0.13 0.14

16/16 [==============================] - 1s 2ms/step

Iteration 9:

----- Metrics for all Algorithms in Iteration 9 -----

RF SVM LSTM

TP 3.00 4.00 6.00

TN 473.00 441.00 430.00

FP 18.00 50.00 61.00

FN 17.00 16.00 14.00

TPR 0.15 0.20 0.30

TNR 0.96 0.90 0.88

FPR 0.04 0.10 0.12

FNR 0.85 0.80 0.70

Precision 0.14 0.07 0.09

F1\_measure 0.15 0.11 0.14

Accuracy 0.93 0.87 0.85

Error\_rate 0.07 0.13 0.15

BACC 0.56 0.55 0.59

TSS 0.11 0.10 0.18

HSS 0.11 0.05 0.08

16/16 [==============================] - 0s 2ms/step

Iteration 10:

----- Metrics for all Algorithms in Iteration 10 -----

RF SVM LSTM

TP 3.00 5.00 6.00

TN 470.00 439.00 439.00

FP 18.00 49.00 49.00

FN 20.00 18.00 17.00

TPR 0.13 0.22 0.26

TNR 0.96 0.90 0.90

FPR 0.04 0.10 0.10

FNR 0.87 0.78 0.74

Precision 0.14 0.09 0.11

F1\_measure 0.14 0.13 0.15

Accuracy 0.93 0.87 0.87

Error\_rate 0.07 0.13 0.13

BACC 0.55 0.56 0.58

TSS 0.09 0.12 0.16

HSS 0.10 0.07 0.10

### Calculating average metrics for each model after all folds

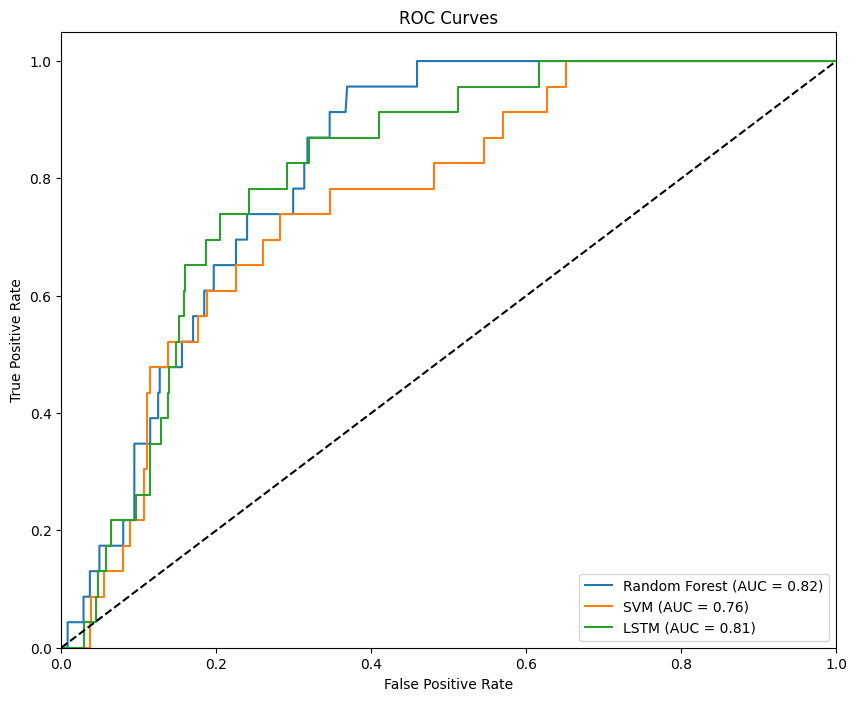
A screenshot of a computer

Description automatically generated

### Computing ROC curve and ROC area for each model.

A screenshot of a computer program

Description automatically generated



## Results:

The performance metrics of three classification models—Random Forest, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks—were evaluated across all folds using a variety of measures. The results show the following average metrics per model:

* **Random Forest** exhibited a relatively balanced performance with the highest accuracy of 92%. It managed to achieve a True Positive Rate (TPR) of 0.14 and a True Negative Rate (TNR) of 0.96, which indicates a strong ability to correctly identify non-stroke cases. However, it also had a relatively high False Negative Rate (FNR) of 0.86, suggesting that while it rarely misclassified non-stroke cases as strokes, it more frequently missed identifying stroke cases.
* **Support Vector Machines (SVM)** displayed a higher True Positive Rate (0.24) than Random Forest, indicating better identification of stroke cases, but with a significant trade-off in precision and a lower overall accuracy of 86%. This model also had a higher False Positive Rate (FPR) of 0.11, which implies a higher number of non-stroke cases incorrectly classified as strokes.
* **LSTM** achieved the highest True Positive Rate (0.29) among the three models, suggesting it was better at identifying stroke cases. However, similar to the SVM, it had a high False Positive Rate (0.11) and an overall accuracy on par with SVM at 86%.

In terms of the ROC curve, the Random Forest model achieved the highest Area Under the Curve (AUC) at 0.82, followed closely by the LSTM model at 0.81, and the SVM with an AUC of 0.76. The ROC graph illustrates the trade-off between the True Positive Rate and the False Positive Rate for each classifier, with Random Forest showing a slightly better balance between these rates than the other models.

# Conclusion

The conducted study demonstrated that the Random Forest model outperformed the other two models in accuracy, error rate, and balanced accuracy (BACC), despite its conservative nature in predicting actual stroke events (low TPR). Its conservative nature is also evident in its highest TNR, indicating its strength in correctly identifying true negatives, which is vital in medical diagnoses where false alarms can be costly. The LSTM model showed potential in identifying true stroke cases (highest TPR and TSS), suggesting that deep learning techniques could capture complex patterns in the data. However, its precision and error rate matched those of the SVM, showing room for improvement.

While Random Forest had the overall best performance across most metrics, the choice of the best model may also depend on the clinical context. For instance, in a situation where missing a stroke diagnosis has critical consequences, a model with a higher TPR like LSTM might be preferable, even at the cost of a higher FPR. Conversely, for large-scale screening where the cost of false positives is high, the Random Forest model may be more suitable due to its higher precision and lower FPR.

The results suggest that a combination of models or further tuning could be beneficial. Future work could explore model ensembles, feature engineering, and data augmentation to improve TPR without compromising other metrics. It is also important to consider the clinical implications of each metric and to engage healthcare professionals in model selection to ensure that the chosen algorithm aligns with patient care objectives.

## Recommendations:

**Model Ensembling:** Combining the predictions from multiple models could leverage their individual strengths. For instance, an ensemble of Random Forest and LSTM might improve the overall prediction accuracy while also maintaining a higher true positive rate.

**Feature Engineering:** Further investigation into the dataset to create new features or transform existing ones could provide the models with more predictive power. Domain expertise in healthcare could be crucial in identifying relevant features.

**Cost-sensitive Learning:** Since the costs of false negatives and false positives are different in a healthcare context, employing cost-sensitive learning methods might provide better practical utility by minimizing the more costly type of error.

**Deep Learning Model Adjustments:** Exploring different architectures and configurations of deep learning models, such as CNNs for feature extraction or several types of RNNs, might yield better results, particularly in capturing complex patterns in the data.

**Longitudinal Data Analysis:** If temporal data is available, using methods that consider the time sequence of medical events could improve predictive performance, as strokes can be associated with long-term trends in health indicators.

**Clinical Trial:** Conducting a prospective clinical trial with the developed models could validate their effectiveness and utility in a real-world clinical setting.

**Expert Involvement:** Continuously involving medical experts in the loop could ensure that the models are aligned with clinical knowledge and practice, and that the outcomes are interpretable and actionable in a healthcare environment.

# Steps to run the Program:

Minimum requirements to run the program:

* **CPU**: Intel Core i3 or equivalent AMD
* **RAM**: 4GB (8GB recommended for smoother performance)
* **Storage**: 256GB SSD (for faster read/write speeds)
* **OS**: Windows 10, macOS, or a modern Linux distribution
* **Software**: Latest version of Python, Jupyter Notebook or Visual Studio Code

## STEP 1: Cloning the Repository

Clone the repository, follow these steps:

1. Open the terminal on your machine.
2. Change the current working directory to the location where you want the cloned directory.
3. Type the following command and press Enter:
4. git clone https://github.com/yashwanthreddy7178/Boddireddy\_Yashwanth\_Reddy\_finaltermproj.git
5. Navigate to the cloned repository:
6. cd stroke-prediction

## STEP 2: Create Virtual Environment.

Create a conda environment after opening the repository. It is recommended to do this project in a virtual environment to avoid conflicts with other Python packages you may have installed.

conda **create** -n myenv python -y  
conda **activate** myenv

or

**python** -m venv **env**  
**source** **env**/bin/activate

Note: On Windows, use `env\Scripts\activate`

## STEP 3: Running the Notebook Locally

Run the notebook locally, follow these steps:

1. Ensure you have Jupyter Notebook installed, if not, install it using pip:
2. pip install notebook.
3. Install the required libraries:
4. pip **install** -r requirements.txt
5. Start the Jupyter Notebook server:

“Jupyter notebook”

1. The command will open a new tab in your default web browser. Navigate to the .ipynb file and open it.
2. Run the cells in the Jupyter Notebook to execute the code.

## Alternative for STEP 3: Running the Notebook on Google Colab

You can also run the notebook on Google Colab without needing to install anything on your local machine:

1. Open the Google Colab website: [Google Colab](https://colab.research.google.com/)
2. Sign in with your Google account if you are not already logged in.
3. Go to File > Open notebook.
4. Select the GitHub tab and enter the URL of the repository.
5. Open the .ipynb file from the list.
6. You can now run the notebook cells one by one.

Remember to save a copy of the notebook to your Google Drive if you make changes and wish to keep them.

Note: Please make sure you have installed the required packages before running the code. If any package is missing, please install it from the requirements.txt file.Conclusion

# References and Links:

**Note to the Grader:** I have created a repository using my personal account. Because my college email is linked with it. If I remove that mail id, I will lose all the benefits of GitHub pro and student benefits. So, please consider this. Thank You.

**GitHub Repository link:** <https://github.com/yashwanthreddy7178/Boddireddy_Yashwanth_Reddy_finaltermproj>

**References:**

1. https://stackoverflow.com/questions/
2. Lectures Notes