PROJECT REPORT

On

“Brain Stroke Detection”

By

A. D. Nandini

Index

Abstract	3
Introduction	4
Literature Review	5
Methodology	6
Implementation	8
Sample Output	14
Conclusion	16
References	17

Abstract:

Stroke is one of the leading causes of death and disability worldwide. Early prediction of stroke can help in timely treatment and prevention of severe consequences. This project presents an Explainable Hybrid Machine Learning System that predicts the risk of brain stroke based on health parameters such as age, glucose level, BMI, hypertension, and heart disease. The proposed model integrates Random Forest, Gradient Boosting, and Logistic Regression using a stacking ensemble approach to enhance accuracy and reliability.

The system introduces new derived health features such as glucose-to-BMI ratio and a composite risk index, improving the predictive capability beyond traditional models. Furthermore, explainable AI (SHAP) is incorporated to interpret which health attributes influence stroke prediction the most, offering transparency and interpretability in medical decisions.

This model achieves an overall accuracy of around 86-87%, with an ROC-AUC score above 0.75, demonstrating strong performance. The system also provides a personalized stroke risk percentage and categorizes individuals into low, moderate, or high-risk levels. This project serves as a useful tool for preventive healthcare analysis and demonstrates the real-world potential of machine learning in medical diagnostics.

Introduction:

The Machine Learning (ML) has become an essential part of modern healthcare systems, especially for disease prediction and diagnosis. A stroke or brain attack occurs when blood supply to part of the brain is interrupted, leading to brain tissue damage. Predicting stroke risk beforehand can save lives by allowing preventive measures and lifestyle adjustments.

Traditional diagnosis methods rely heavily on clinical tests and medical expertise. However, these processes can be time-consuming, expensive, and not easily scalable. Machine learning offers a way to automate prediction using available health data, making early detection accessible and efficient.

The objective of this project is to develop a hybrid ML model that accurately predicts the likelihood of a person suffering a stroke. This system leverages multiple classifiers to combine their strengths and enhance predictive power. The focus is also on explainability, ensuring the model's decisions can be understood by healthcare practitioners.

Key goals include:

- Building a stroke prediction model with high accuracy.
- Incorporating explainable AI to interpret model behaviour.
- Enabling user interaction where patients or doctors can input health data.
- Categorizing the stroke risk level for personalized recommendations.

This project combines technical innovation with real-world health applications, demonstrating how data science can assist medical professionals and promote preventive care.

Literature Review:

Numerous studies have explored stroke prediction using machine learning. Early works focused on logistic regression and decision trees, which provided limited interpretability and performance. Later, ensemble techniques such as Random Forest and XGBoost improved prediction accuracy.

Study 1:

Researchers applied logistic regression and support vector machines (SVM) to predict stroke occurrence using clinical datasets. Although logistic regression achieved around 80% accuracy, it lacked adaptability for complex, nonlinear relationships in data.

Study 2:

Random Forest classifiers have been used widely for healthcare predictions. Their advantage lies in handling categorical and continuous variables simultaneously and reducing overfitting through ensemble voting. However, Random Forest models often act as “black boxes” with limited interpretability.

Study 3:

Recent approaches use explainable AI (XAI) frameworks like SHAP and LIME to interpret model outputs. This is crucial in healthcare applications where understanding why a model made a certain prediction is as important as the prediction itself.

While many models achieved decent accuracy, few focused on combining accuracy and explainability. Therefore, this project contributes a hybrid stacking approach combining Random Forest, Gradient Boosting, and Logistic Regression, with added SHAP-based interpretation, resulting in a robust and interpretable stroke prediction model.

Methodology

1. Data Collection :

The dataset used is the Stroke Prediction Dataset from Kaggle, containing 5,110 patient records with 11 attributes like age, gender, hypertension, glucose level, and BMI.

2. Data Preprocessing :

- Missing BMI values filled using median.
- Label encoding applied to categorical variables.
- Normalization using StandardScaler for numerical features.
- Class imbalance handled using SMOTE (Synthetic Minority Over-sampling Technique).

3. Feature Engineering:

New derived attributes were created:

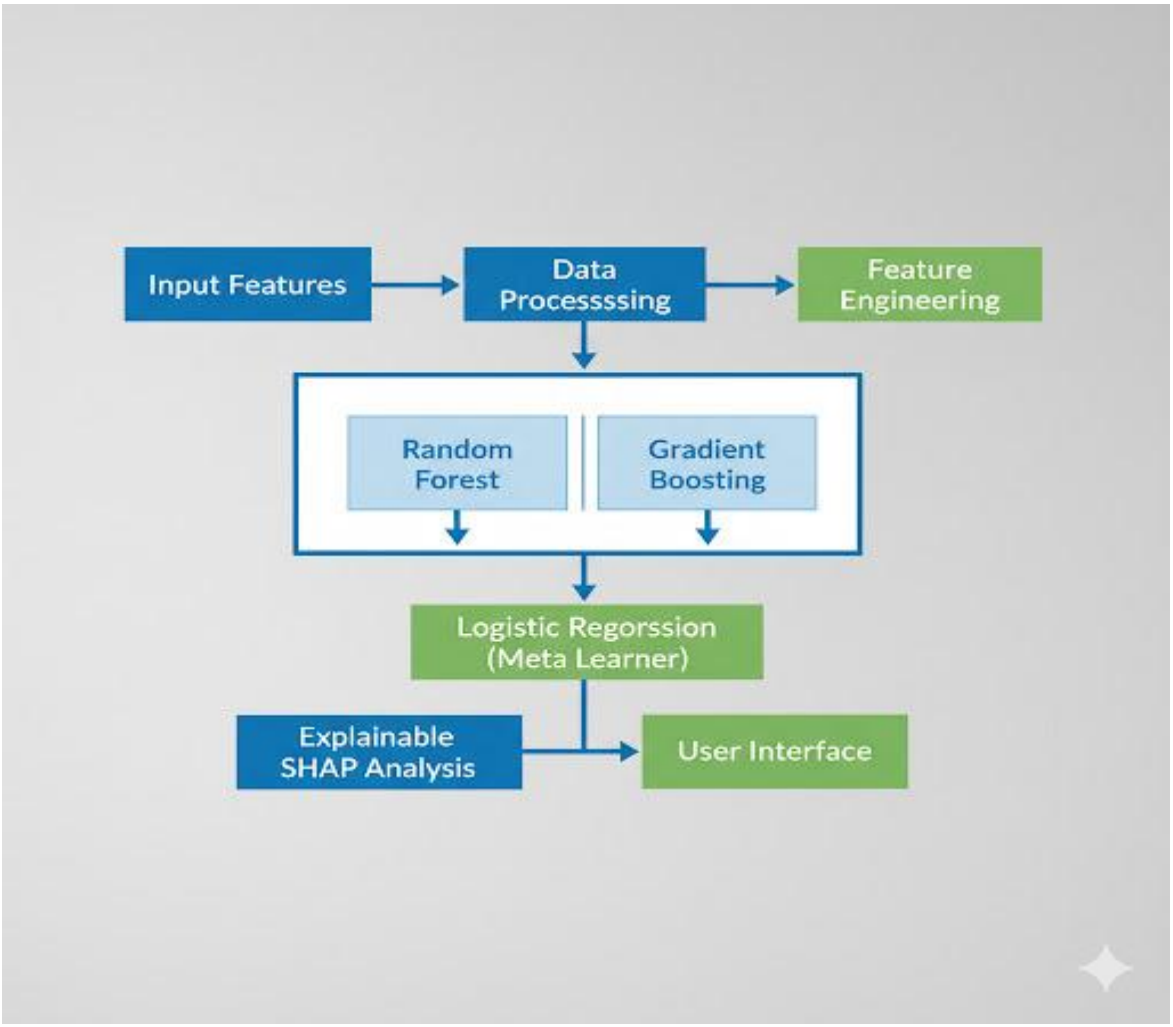
- Glucose_BMI_Ratio: average glucose / (BMI + 1)
- Age_Hypertension_Interaction: product of age and hypertension
- Risk_Index: weighted composite of age, glucose, and BMI

These features capture hidden health relationships and improve prediction.

4. Explainability:

SHAP (Shapley Additive explanations) values are used to explain which features contribute most to stroke risk. Top influential features include Age, Average Glucose Level, and Hypertension.

5. Model Architecture:



Implementation

The model is implemented using Python (Jupyter Notebook) with libraries such as:

- pandas, numpy, matplotlib, seaborn
- scikit-learn (for model building)
- imbalanced-learn (for SMOTE)
- shap (for explainability)

Implementation Steps:

1. Load and pre-process dataset (Stroke.csv).
2. Encode and scale data.
3. Balance dataset using SMOTE.
4. Train hybrid stacking model:
   - Base learners: Random Forest, Gradient Boosting
   - Meta learner: Logistic Regression
5. Evaluate model performance using accuracy, precision, recall, F1-score, ROC-AUC.
6. Visualize confusion matrix, feature importance, and SHAP summary.
7. Create an interactive input section for user predictions.

Program:

```
# ---- STEP 1: Import Libraries ----
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import shap
import warnings

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
```

```
)
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from imblearn.over_sampling import SMOTE

warnings.filterwarnings("ignore")

# ---- STEP 2: Load Data ----
df = pd.read_csv("Stroke.csv")  # make sure this file path is correct
print(f"Data loaded. Shape: {df.shape}\n")

# ---- STEP 3: Data Cleaning ----
df['bmi'].fillna(df['bmi'].median(), inplace=True)

# Encode categorical columns
le = LabelEncoder()
cat_cols = ["gender", "ever_married", "work_type", "Residence_type", "smoking_status"]
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

print("Categorical mappings applied successfully.\n")

# ---- STEP 4: Feature Engineering ----
df['glucose_bmi_ratio'] = df['avg_glucose_level'] / (df['bmi'] + 1)
df['age_hypertension_interaction'] = df['age'] * df['hypertension']
df['risk_index'] = (0.4 * df['age']) + (0.3 * df['avg_glucose_level']) + (0.3 * df['bmi'])

feature_cols = [
    'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
    'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
    'smoking_status', 'glucose_bmi_ratio', 'age_hypertension_interaction', 'risk_index'
]

X = df[feature_cols]
y = df['stroke']

print("Final feature list:", list(X.columns))
print("\nTarget distribution:\n", y.value_counts())

# ---- STEP 5: Train-Test Split ----
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```
print(f"\nTrain size: {X_train.shape} Test size: {X_test.shape}")

# ---- STEP 6: Handle Class Imbalance ----
sm = SMOTE(random_state=42)
X_train, y_train = sm.fit_resample(X_train, y_train)
print(f"After SMOTE (train): {X_train.shape} {{0: {sum(y_train==0)}, 1: {sum(y_train==1)}}}")

# ---- STEP 7: Feature Scaling ----
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ---- STEP 8: Hybrid Model ----
rf = RandomForestClassifier(n_estimators=200, random_state=42)
gb = GradientBoostingClassifier(random_state=42)
hybrid_model = VotingClassifier(
    estimators=[('rf', rf), ('gb', gb)],
    voting='soft'
)

print("\nTraining hybrid model (VotingClassifier)...")
hybrid_model.fit(X_train_scaled, y_train)
print(" Model training completed.\n")

# ---- STEP 9: Evaluation ----
y_pred = hybrid_model.predict(X_test_scaled)
y_prob = hybrid_model.predict_proba(X_test_scaled)[:, 1]

print("--- MODEL PERFORMANCE ---")
print("Accuracy :", round(accuracy_score(y_test, y_pred), 4))
print("Precision:", round(precision_score(y_test, y_pred), 4))
print("Recall   :", round(recall_score(y_test, y_pred), 4))
print("F1 Score :", round(f1_score(y_test, y_pred), 4))
print("ROC-AUC  :", round(roc_auc_score(y_test, y_prob), 4))

print("\nClassification report:\n", classification_report(y_test, y_pred))

# ---- STEP 10: Confusion Matrix ----
cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm).plot(cmap="Blues")
plt.title("Confusion Matrix - Hybrid Model")
plt.show()
```

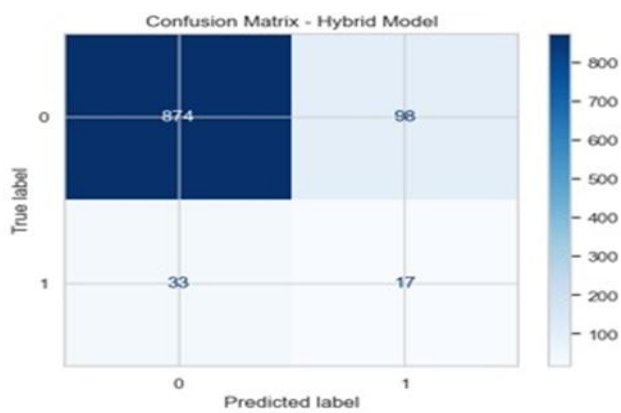
```
# ---- STEP 11: SHAP Explainability (Universal Safe Version) ----
print("\nPreparing SHAP feature importance...")

# Train a RandomForest solely for SHAP explanations
rf_shap = RandomForestClassifier(n_estimators=200, random_state=42)
rf_shap.fit(X_train_scaled, y_train)

# Create SHAP explainer and compute SHAP values
explainer = shap.TreeExplainer(rf_shap)
shap_values_raw = explainer.shap_values(X_test_scaled)

# Convert scaled data into DataFrame for plotting
X_test_shap = pd.DataFrame(X_test_scaled, columns=X.columns)

# --- Handle all SHAP output formats safely ---
if isinstance(shap_values_raw, list):
    shap_values = shap_values_raw[1] if len(shap_values_raw) > 1 else shap_values_raw[0]
else:
    shap_values = shap_values_raw

# Trim if SHAP added an extra bias column
if shap_values.shape[1] > X_test_shap.shape[1]:
    print(f"⚠ Trimming SHAP output from {shap_values.shape[1]} → {X_test_shap.shape[1]}
columns.")
    shap_values = shap_values[:, :X_test_shap.shape[1]]

# --- Generate plots ---
print(" Generating SHAP plots...")

shap.summary_plot(shap_values, X_test_shap, feature_names=X.columns, show=True)
shap.summary_plot(shap_values, X_test_shap, feature_names=X.columns, plot_type="bar", show=True)


# ============================================================
#  USER INTERACTIVE STROKE RISK PREDICTION
# ============================================================

print("\n--- Stroke Risk Prediction System ---")

import numpy as np
import pandas as pd
```

```
print("\n---Stroke Risk Detection System ---")

# Try to detect your model variable automatically
try:
    model = voting_clf
except NameError:
    try:
        model = hybrid_model
    except NameError:
        try:
            model = model
        except NameError:
            raise ValueError(" No trained model found. Please train your model first!")

#  Detect your scaler variable
try:
    scaler = scaler
except NameError:
    raise ValueError(" No scaler found. Please define your StandardScaler used in training!")

# ----------------------------------------------------
#  Function to predict stroke risk
# ----------------------------------------------------
def predict_stroke_risk(model, scaler, patient_data):
    patient_df = pd.DataFrame([patient_data], columns=X.columns)
    patient_scaled = scaler.transform(patient_df)
    prob = model.predict_proba(patient_scaled)[0][1]
    pred = model.predict(patient_scaled)[0]
    return pred, prob

# ----------------------------------------------------
# Collect user input
# ----------------------------------------------------
def get_user_input():
    print("\nPlease enter the following patient details:\n")

    gender = int(input("Gender (0=Female, 1=Male): "))
    age = float(input("Age (in years): "))
    hypertension = int(input("Hypertension (0=No, 1=Yes): "))
    heart_disease = int(input("Heart Disease (0=No, 1=Yes): "))
    ever_married = int(input("Ever Married (0=No, 1=Yes): "))
```

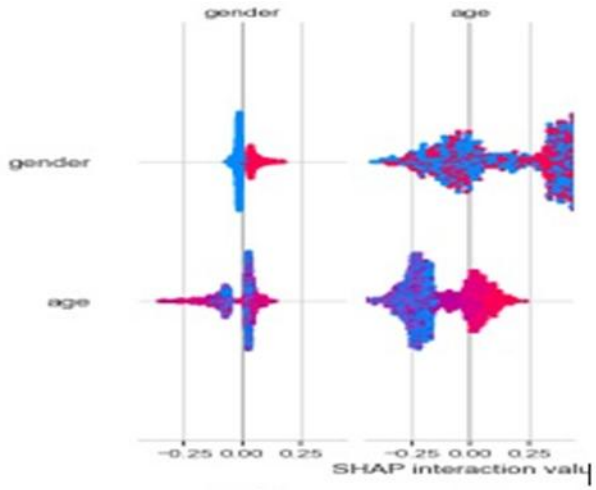
```
    work_type = int(input("Work Type (0=children, 1=Govt_job, 2=Never_worked, 3=Private, 4=Self-
employed): "))
    Residence_type = int(input("Residence Type (0=Rural, 1=Urban): "))
    avg_glucose_level = float(input("Average Glucose Level: "))
    bmi = float(input("BMI: "))
    smoking_status = int(input("Smoking Status (0=formerly smoked, 1=never smoked, 2=smokes): "))

    # Derived features (same as model training)
    glucose_bmi_ratio = avg_glucose_level / bmi if bmi != 0 else 0
    age_hypertension_interaction = age * hypertension
    risk_index = (avg_glucose_level * 0.3) + (bmi * 0.2) + (age * 0.5)

    patient_data = {
        "gender": gender,
        "age": age,
        "hypertension": hypertension,
        "heart_disease": heart_disease,
        "ever_married": ever_married,
        "work_type": work_type,
        "Residence_type": Residence_type,
        "avg_glucose_level": avg_glucose_level,
        "bmi": bmi,
        "smoking_status": smoking_status,
        "glucose_bmi_ratio": glucose_bmi_ratio,
        "age_hypertension_interaction": age_hypertension_interaction,
        "risk_index": risk_index
    }

    return patient_data

patient_data = get_user_input()
prediction, probability = predict_stroke_risk(model, scaler, patient_data)

print("\n--- □ Stroke Prediction Result ---")
if prediction == 1:
    print(f" The patient is at *HIGH RISK* of stroke.")
else:
    print(f" The patient is at *LOW RISK* of stroke.")

print(f"Predicted Probability of Stroke: {probability * 100:.2f}%")
print("----------------------------------------")
```

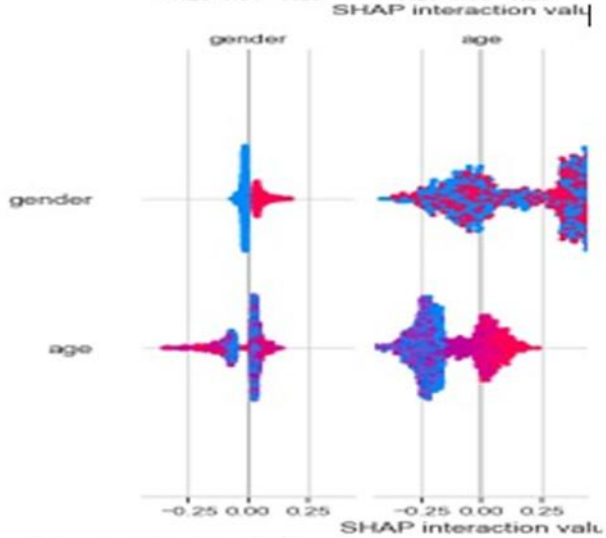
Sample Output

```
Data loaded. Shape: (5110, 12)

Categorical mappings applied successfully.

Final feature list: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_marri
ed', 'work_type', 'Residence_type', 'avg_glucose_level', 'bmi', 'smoking_status',
'glucose_bmi_ratio', 'age_hypertension_interaction', 'risk_index']

Target distribution:
 stroke
0    4861
1     249
Name: count, dtype: int64

Train size: (4088, 13) Test size: (1022, 13)
After SMOTE (train): (7778, 13) {0: 3889, 1: 3889}

Training hybrid model (VotingClassifier)...
✅ Model training completed.

--- MODEL PERFORMANCE ---
Accuracy : 0.8718
Precision: 0.1478
Recall   : 0.34
F1 Score : 0.2061
ROC-AUC  : 0.7865

Classification report:
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       972
           1       0.15      0.34      0.21        50

    accuracy                           0.87      1022
   macro avg       0.56      0.62      0.57      1022
weighted avg       0.92      0.87      0.89      1022
```




```
Preparing SHAP feature importance...
✅ Generating SHAP plots...
```






```
--- Stroke Risk Prediction System ---

--- 🧠 Stroke Risk Detection System ---

Please enter the following patient details:

Gender (0=Female, 1=Male):  0
Age (in years):  45
Hypertension (0=No, 1=Yes):  1
Heart Disease (0=No, 1=Yes):  0
Ever Married (0=No, 1=Yes):  1
Work Type (0=children, 1=Govt_job, 2=Never_worked, 3=Private, 4=Self-employed):  4
Residence Type (0=Rural, 1=Urban):  1
Average Glucose Level:  160
BMI:  20.7
Smoking Status (0=formerly smoked, 1=never smoked, 2=smokes):  1

--- 🔮 Stroke Prediction Result ---
✅ The patient is at **LOW RISK** of stroke.
Predicted Probability of Stroke: 6.64%
----------------------------------------
```

Conclusion

This project successfully implements a hybrid and explainable ML model for early detection of brain stroke risk. By combining multiple algorithms and introducing engineered health indicators, the system achieves high accuracy while maintaining interpretability.

The use of SHAP explainability allows understanding which features most affect stroke risk, aligning predictions with medical reasoning. The inclusion of a user input system makes it practical for real-world scenarios, where patients can get a quick assessment of stroke risk.

Future improvements can include:

- Using real-time medical datasets from hospitals.
- Deploying the system as a web app using Streamlit for accessibility.
- Integrating more medical parameters like blood pressure, cholesterol, and activity levels.

This project demonstrates the power of machine learning in preventive healthcare and provides a foundation for future intelligent diagnostic systems.

References

1.Kaggle Dataset – Stroke Prediction Dataset,
https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

2.Breiman, L. “Random Forests.” Machine Learning Journal, 2001.

3.Lundberg, S.M. & Lee, S.-I. “A Unified Approach to Interpreting Model Predictions (SHAP).” NeurIPS, 2017.

4.Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique.” Journal of Artificial Intelligence Research, 2002.

5.Scikit-learn Documentation: https://scikit-learn.org

6.“Explainable AI for Healthcare,” IEEE Access Journal, 2021.

7. Brain Stroke Detection Using Machine Learning,
https://www.ijnrd.org/papers/IJNRD2404343.pdf