# Machine Learning-Driven Risk Assessment for Stroke Prevention



| | |
|---|---|
| John Vimrish Baskar Vincent | 33690343 |
| Kalyan Uppalapatti | 33693385 |
| Shraddha Sukhadev Jadhav | 33752470 |
| Jaison Felix Menezes | 33739444 |
| Sakshi Vikas Jadhav | 33702693 |


Machine Learning for Business Analytics

PROF. Moy Patel

Date:- 05/06/2025

## Contents

# 1. Introduction

In the realm of public health, stroke represents one of the leading causes of death and long-term disability worldwide. Timely prediction and identification of at-risk individuals can significantly improve clinical outcomes and reduce healthcare burdens. This project applies machine learning techniques to analyze a healthcare dataset and explore the factors that most influence the likelihood of a person experiencing a stroke.

With healthcare costs and patient volumes on the rise, predictive analytics provides an opportunity to identify high-risk patients proactively, allowing interventions that are not only timely but potentially life-saving. This report uses real-world health data to uncover patterns and build predictive models that can assist medical professionals in risk assessment.

By combining rigorous data analysis with machine learning algorithms, this research aims to shed light on the most critical factors affecting stroke risk. The insights derived can enhance early diagnosis, inform policy decisions, and ultimately contribute to better patient outcomes.

## 2. Objectives

- **Assess Stroke Occurrence Patterns**: Analyze overall trends and distributions of stroke incidents among the population based on demographic and health-related attributes.
- **Identify Key Risk Factors**: Determine which features such as age, hypertension, heart disease, glucose level, and BMI most significantly influence the likelihood of a stroke using statistical and machine learning approaches.
- **Detect and Handle Data Irregularities**: Identify and address missing values and outliers, particularly in critical medical indicators like BMI, to maintain data integrity and modeling accuracy.
- **Develop Predictive Models**: Train and evaluate several classification models such as Logistic Regression, Decision Tree, and Random Forest to accurately predict the risk of stroke.
- **Interpret Feature Importance**: Use built-in model methods (e.g., feature importances from tree-based models or regression coefficients) to evaluate how each feature contributes to predictions.
- **Provide Actionable Health Insights**: Derive practical recommendations from the analysis to inform early detection strategies and healthcare interventions.
- **Visualize Critical Data Patterns**: Generate meaningful visualizations to highlight significant patterns and trends that assist in understanding the relationship between features and stroke occurrence.

## 3. Dataset Intro

This dataset provides anonymized health records for over 5,000 individuals, including demographic, physiological, and behavioral attributes that may be relevant in assessing stroke risk. The details of the dataset are:

**Source**:kaggle(https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download)

**Dataset Title**: Stroke Prediction Dataset

**Number of Entries**: 5,110

**Total Features**: 11 input features + 1 target variable

**Target Variable: s**troke (0 = No, 1 = Stroke occurred)

| No. | Attribute Name | Description |
|---|---|---|
| 1 | ID | A unique identifier for each individual |
| 2 | Gender | Gender of the passengers (Female, Male) |
| 3 | Age | Age in years |
| 4 | hypertension | 1 = Has high blood pressure, 0 = No |
| 5 | heart_disease | 1 = Has heart disease, 0 = No |
| 6 | ever_married | "Yes" if the person was ever married, else "No" |
| 7 | work_type | Type of employment: Private, Govt_job, Self-employed, Children, Never_worked |
| 8 | Residence_type | Living area type: Urban or Rural |
| 9 | avg_glucose_level | Average blood glucose level (mg/dL) |
| 10 | bmi | Body Mass Index (can be missing) |
| 11 | smoking_status | Smoking status: never smoked, formerly smoked, smokes, or Unknown |
| 12 | stroke | Target variable: 1 = Stroke occurred, 0 = No stroke |

**Table 1: Description of Dataset**

This dataset forms the basis for our analysis and predictive modeling. In the following sections, we will delve deeper into understanding, preprocessing, and modeling this data.

This dataset contains medical and demographic data for individuals, with the primary goal of identifying patterns that may predict the occurrence of a stroke. The data is structured to include personal characteristics (e.g., age, gender, marital status), lifestyle habits (e.g., smoking), and health metrics (e.g., glucose level, BMI, history of hypertension and heart disease).

The dataset provides a real-world scenario for applying machine learning in healthcare to improve early diagnosis and preventive care strategies.

# 4. Data Understanding

First, all the basic necessary libraries of python like Pandas, NumPy, Scikit- learn, Seaborn etc., are imported into Jupyter Notebook.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score, accuracy_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier
import xgboost as xgb
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
```

**Figure 1: Importing all the necessary libraries**

The data samples of the CSV file loaded into dataframe by using the read_csv() function are: This function returns the data of the CSV file as a two-dimensional data structure with encoded axes, called data, as shown below:

```python
import kagglehub
import pandas as pd
import os

# Download and get the path to the dataset directory
path = kagglehub.dataset_download("fedesoriano/stroke-prediction-dataset")
# Inspect files in the downloaded path
print("Files in dataset folder:", os.listdir(path))
# Load the CSV file (adjust filename if needed)
csv_path = os.path.join(path, "healthcare-dataset-stroke-data.csv")
df = pd.read_csv(csv_path)

# Display first few rows
print(df.head())
```

```
Files in dataset folder: ['healthcare-dataset-stroke-data.csv']
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             0              0          Yes
4   1665  Female  79.0             1              0          Yes

       work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0        Private          Urban             228.69  36.6  formerly smoked
1  Self-employed          Rural             202.21   NaN     never smoked
2        Private          Rural             105.92  32.5     never smoked
3        Private          Urban             171.23  34.4           smokes
4  Self-employed          Rural             174.12  24.0     never smoked

   stroke
0       1
1       1
2       1
3       1
4       1
```

**Figure 2: Shape of the dataset**

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

**Figure 4: Dataset Information**

After successful storage of data, one can review the data in the tabular format and can access each part of the data easily. The dataframe's shape is (5110, 12), which indicates the number of samples and features from the dataset.

After studying the data, we can see that two types of features are available:

1. Numerical features
2. Categorical feature

The numerical features are:

1. ID
2. id
3. age
4. hypertension
5. heart_disease
6. avg_glucose_level
7. bmi
8. stroke (Target variable)

The Categorical features are:

1. gender
2. ever_married
3. work_type
4. Residence_type
5. smoking_status

For a better understanding of the data, we apply df.isnull().sum() to check for null values in each feature. After applying that, we observe that there is a null value in the bmi feature.

```
df.isnull().sum()
```

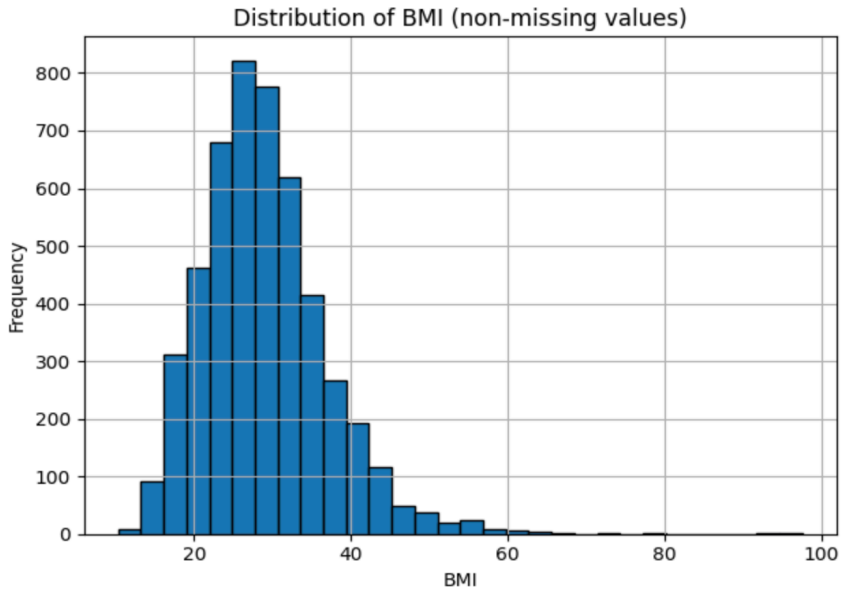
```
id                     0
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```

## 6. **Data visualization**

We created various graphs and plots using the popular Python libraries **Matplotlib** and **Seaborn** to visually represent the dataset. These visualizations help us understand the distribution, relationships, and potential patterns among features. Individual bar charts, histograms, and density plots were used to analyze attributes comprehensively.

This histogram shows the distribution of BMI (Body Mass Index) across the dataset, excluding missing values. Most individuals fall within a BMI range of 20 to 35, which is generally considered within or slightly above the healthy range. The distribution is slightly right skewed, with a long tail indicating the presence of outliers with very high BMI values.

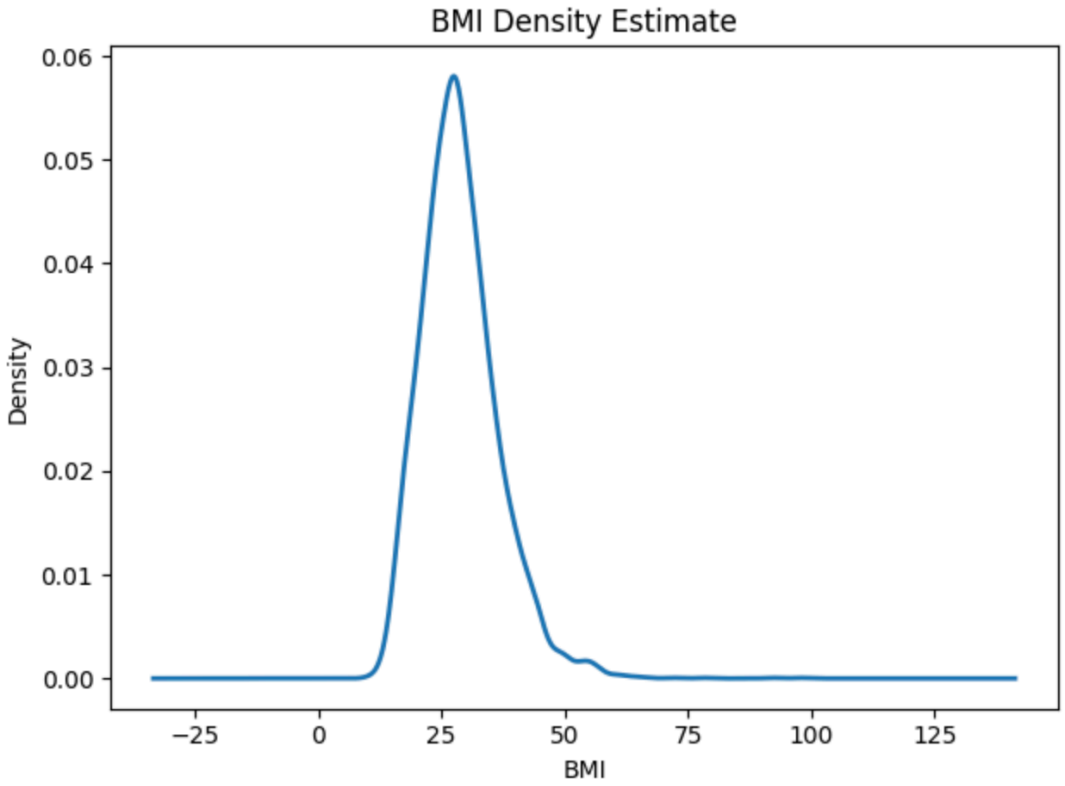
Such outliers may influence predictive modeling and are considered during preprocessing and outlier analysis.




**Histogram of BMI (non-missing values)**

This KDE plot visualizes the **distribution of BMI values** across all individuals in the dataset. The sharp peak around **25–30 BMI** indicates that a majority of individuals fall into this range, which aligns with the borderline between normal and overweight classifications.

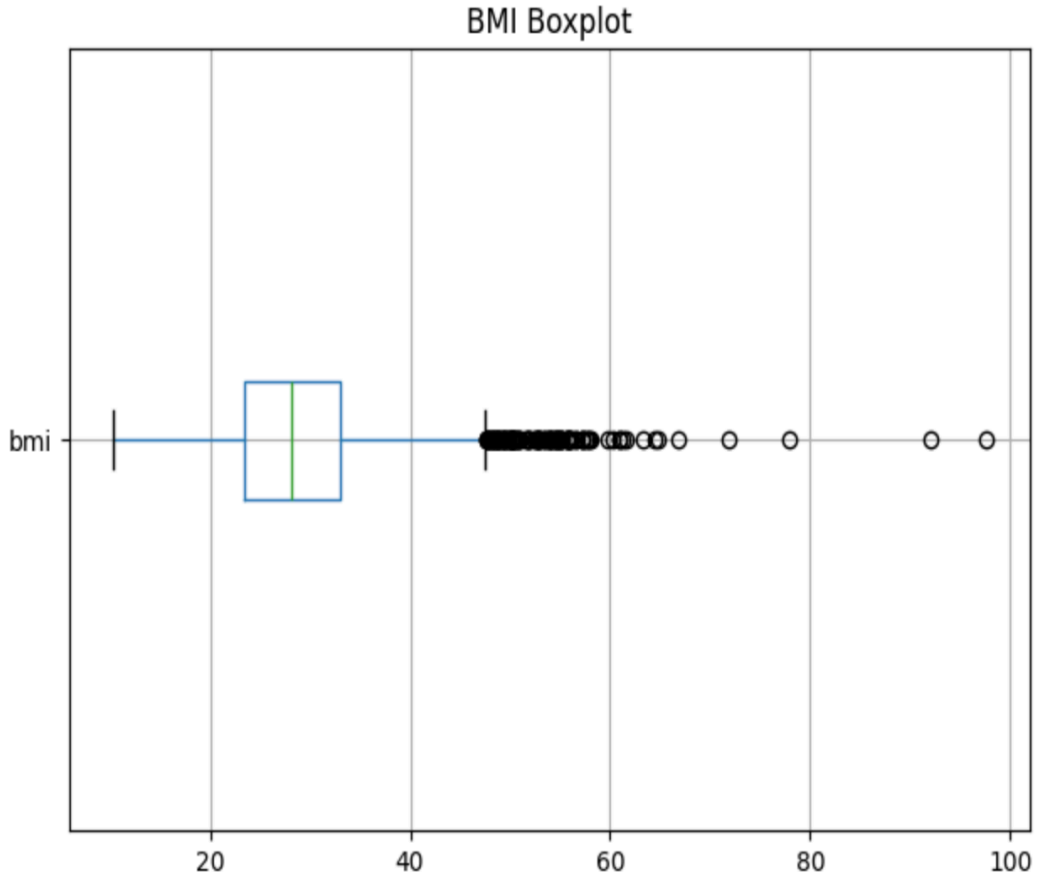
The tail stretching to the right suggests the presence of individuals with **higher BMI values**, indicative of overweight or obesity a known risk factor for cardiovascular conditions and stroke. This visual highlight the skewness in BMI and reinforces the need to treat extreme values carefully during preprocessing and analysis.




**Kernel Density Plot of BMI Distribution**

The boxplot above visualizes the **spread and central tendency of BMI** among individuals in the dataset. The box represents the interquartile range (IQR), with the **median** BMI value indicated by the vertical line inside the box. Most BMI values fall within a reasonable range, approximately between **22 and 35**.

However, the presence of numerous **black circular markers beyond the upper whisker** reveals that there are **many outliers with high BMI values**, some even exceeding **60–80**. These outliers are significant, as individuals with extremely high BMI may have a greater risk of stroke and other health complications. Such patterns should be considered during data preprocessing and model training to mitigate skewed model behavior.

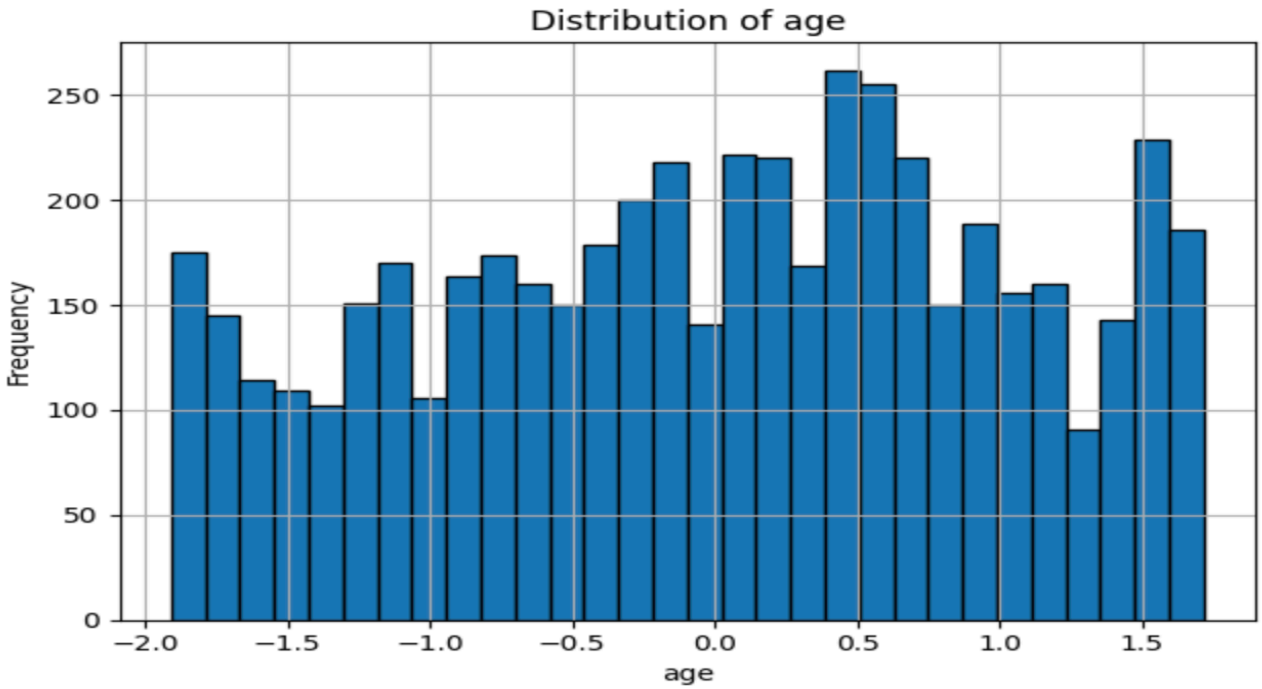



**Boxplot Showing BMI Distribution and Outlier**

## EXPLORATORY DATA ANALYSIS:

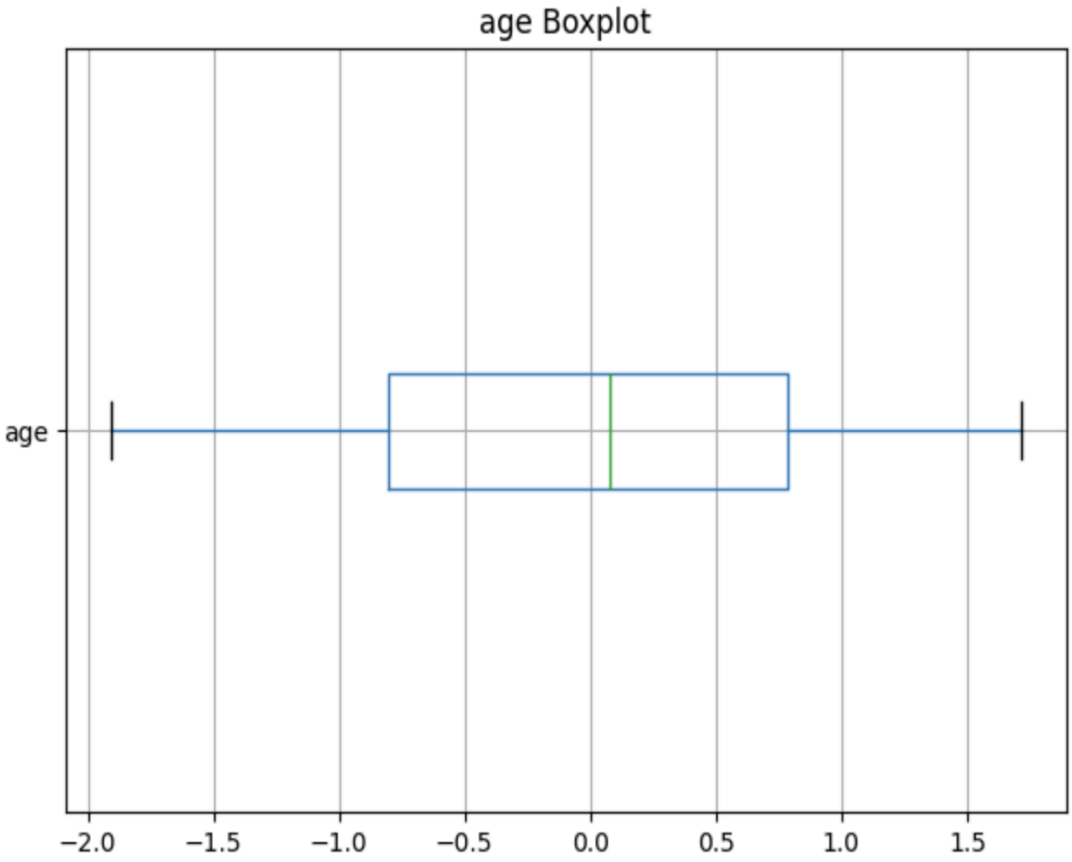
- **Age**
- **BMI**
- **Glucose**

This plot is a histogram displaying the frequency of individuals across different normalized age ranges. It shows a fairly uniform distribution with a concentration in the central age brackets, indicating that the dataset includes a diverse age group from young to elderly individuals.




**Histogram of Age Distribution**

This plot visualizes the **statistical spread of the age variable**. The interquartile range (IQR) is balanced, and the boxplot indicates that **age values are symmetrically distributed** without significant outliers. This confirms that age is a well-distributed variable and suitable for modeling without requiring transformation or filtering.

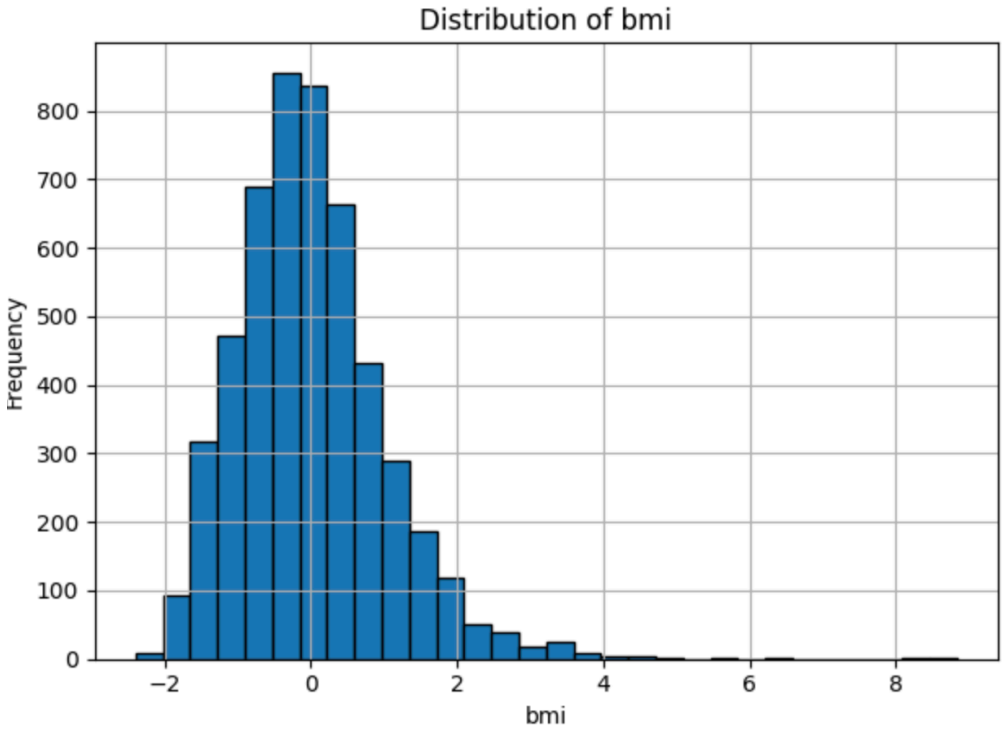



```
BMI  ──────────────────────────────────────
count    5.110000e+03
mean    -1.418304e-16
std      1.000098e+00
min     -2.399812e+00
25%     -6.747875e-01
50%     -9.548828e-02
75%      5.353042e-01
max      8.838593e+00
Name: bmi, dtype: float64
```

**Boxplot Showing Spread of Age Values**

The above histogram depicts the frequency of standardized BMI values across the dataset. After normalization, the values are centered around zero, reflecting the mean, with a visible skewness toward the right. This distribution indicates that while most individuals fall near the average BMI, a notable number have higher BMI scores, as evidenced by the extended tail.

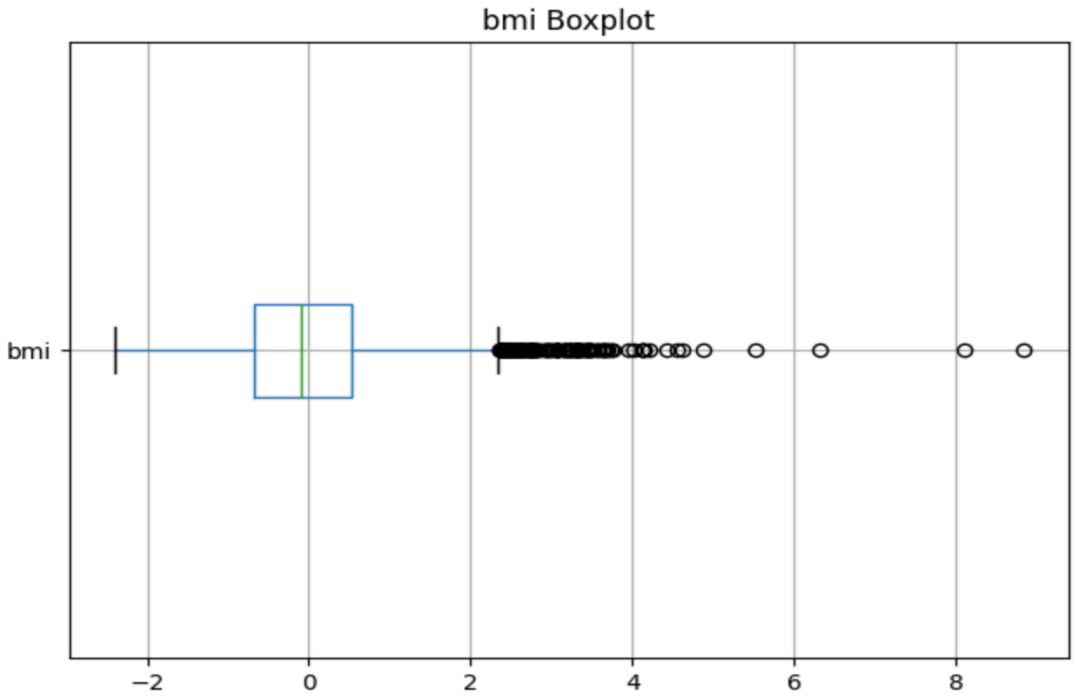
Such a distribution is typical in medical datasets where outliers are medically relevant. These outlier cases, especially those with high BMI, should be carefully considered, as they can significantly impact stroke risk and may need attention in both model training and health intervention recommendations.




**Histogram of Standardized BMI Values**

This visualization highlights the statistical spread of BMI after normalization. The interquartile range is centered around zero, and the median is well-balanced, indicating symmetry in the data's core values. However, the presence of numerous outliers on the right (represented as black circles) confirms that high-BMI individuals are still present even after normalization.

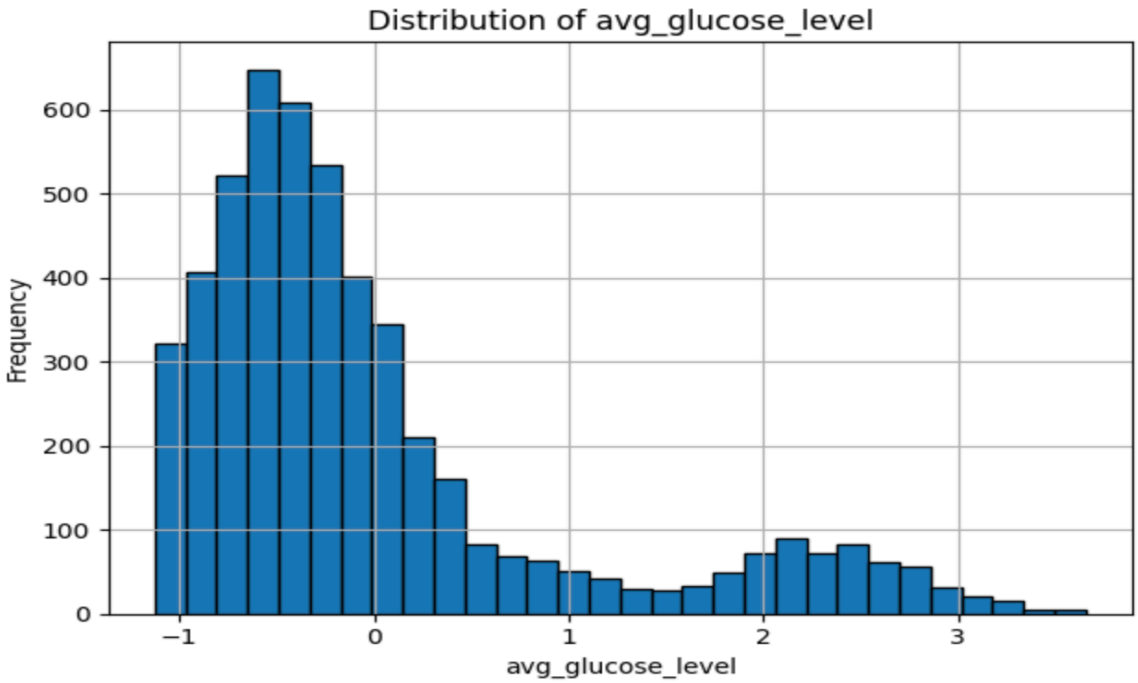
These outliers are crucial for stroke analysis, as individuals with elevated BMI levels are known to be at greater risk. Proper outlier treatment or modeling techniques like robust classifiers may be beneficial.




**Boxplot of Standardized BMI Values**

The above histogram depicts the frequency distribution of avg_glucose_level after normalization. The data appears right skewed, with a high concentration of values around -1 to 0.5, indicating that most individuals have glucose levels near the mean.

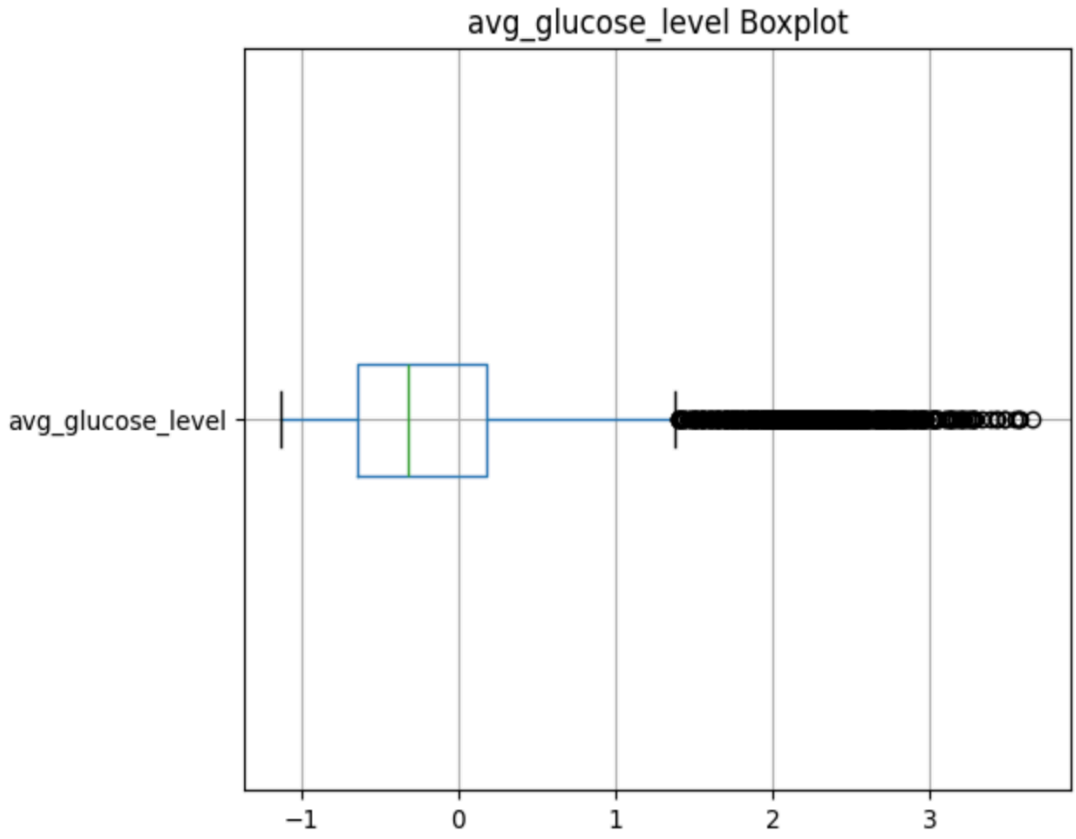
However, the long tail extending to the right reflects the presence of individuals with significantly higher glucose levels, which may point to conditions such as prediabetes or diabetes both associated with increased stroke risk. The normalized format helps balance the feature for modeling while preserving the shape of this medically important distribution.




**Histogram of Normalized Average Glucose Levels**

The boxplot presents the distribution of the avg_glucose_level feature after normalization. The majority of data points lie within the interquartile range, centered close to zero. However, a large number of outliers can be observed on the right side, indicating individuals with abnormally high glucose levels.

These outliers are clinically significant, as elevated glucose levels are a known risk factor for stroke. Their presence highlights the need for robust preprocessing and potential feature scaling when building predictive models. In particular, models that are sensitive to skewed distributions or outliers should be tuned appropriately.



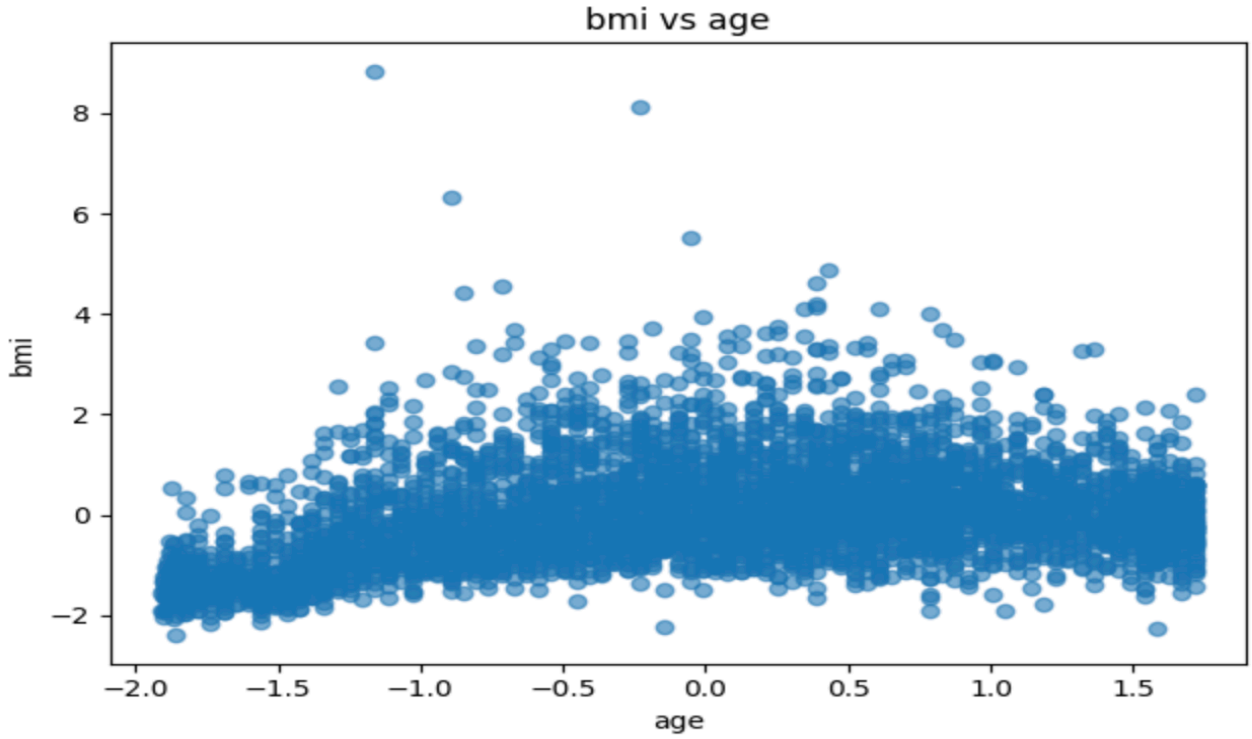

**Boxplot of Normalized Average Glucose Levels**

# BIVARIATE ANALYSIS

The scatter plot above illustrates the relationship between normalized age and BMI. Each point represents an individual in the dataset. Most of the points are densely clustered around the center, reflecting the general population's average BMI and age.

There is a mild upward trend, indicating that BMI tends to increase slightly with age up to a certain point. However, the relationship is not strongly linear. The plot also shows some extreme outliers in BMI, particularly in older individuals, which are worth flagging for further medical insight or preprocessing.

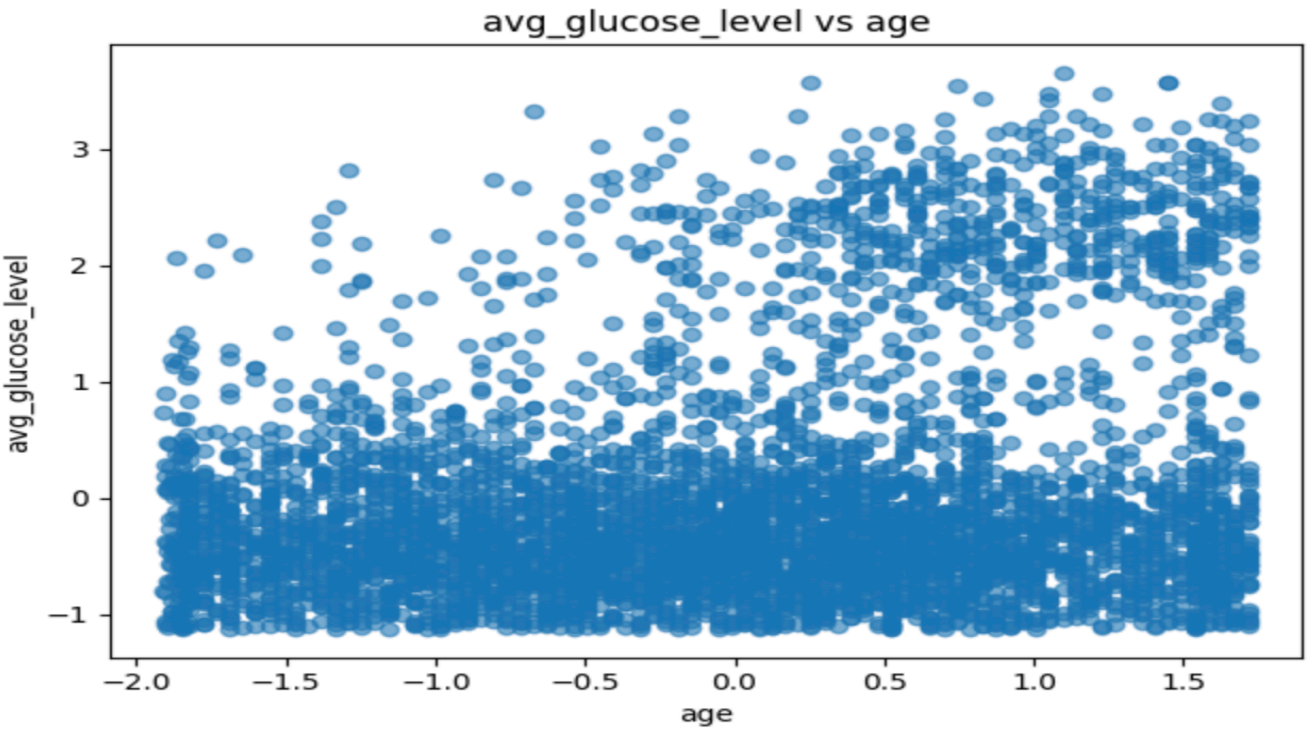
This bivariate visualization is helpful to assess whether age and BMI interact in a way that may influence stroke risk or model performance.




**Scatter Plot of Normalized BMI vs. Age**

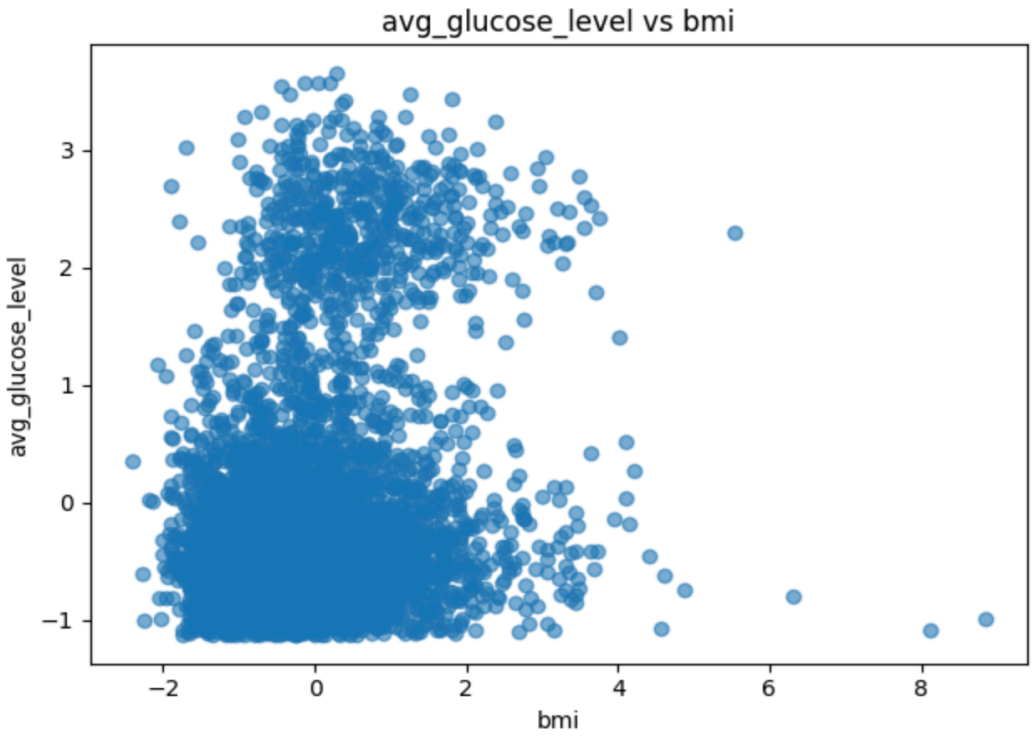
The above scatter plot visualizes the relationship between normalized avg_glucose_level and age. The majority of individuals have glucose levels clustered below the zero mark, with a gradual rise in variance as age increases.

There is no strong linear trend, but we observe that older individuals are more likely to have elevated glucose levels, with several outliers beyond the normalized value of 3. This supports medical expectations that glucose dysregulation becomes more common with aging, making this relationship important for stroke risk modeling and early intervention.




**Scatter Plot of Normalized Glucose Level vs. Age**

This visualization highlights the distribution and interaction between normalized bmi and avg_glucose_level. While the majority of points are concentrated near the origin (representing average levels), a wider spread emerges as values increase on both axes. There is no strong linear correlation, but it is evident that individuals with higher BMI values are also more likely to exhibit elevated glucose levels a combination often associated with metabolic syndrome and increased stroke risk. The pattern also shows a dense cluster of points with moderate glucose levels across all BMI ranges, indicating the presence of hidden factors influencing glucose independent of BMI.



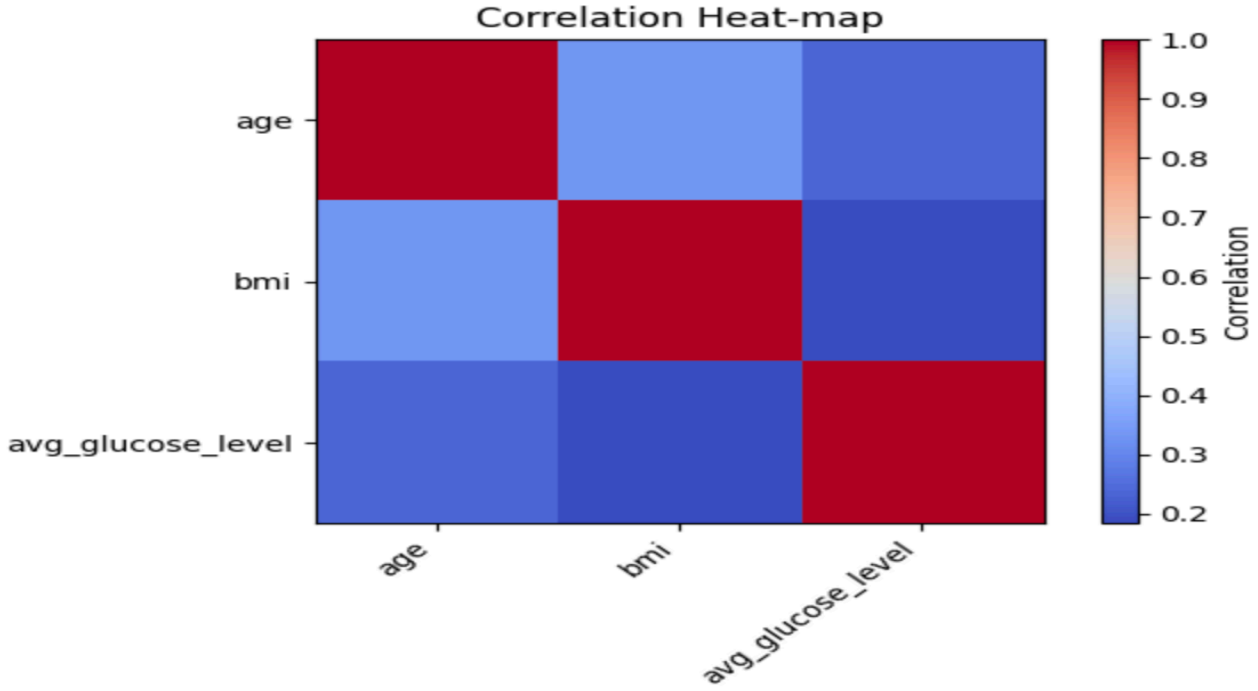

**Scatter Plot of Normalized Average Glucose Level vs. BMI**

The figure above includes both a Pearson correlation matrix and its heatmap representation, which helps quantify and visualize the linear relationships among the key features: age, bmi, and avg_glucose_level.

- Age and BMI show a moderate positive correlation of 0.334, implying that BMI tends to rise slightly with age.
- Age and glucose level show a weaker correlation of 0.238, suggesting a mild tendency for glucose levels to increase with age.
- BMI and glucose level are least correlated at 0.184, indicating that although these two may be linked through broader metabolic processes, their relationship in this dataset is not strongly linear.

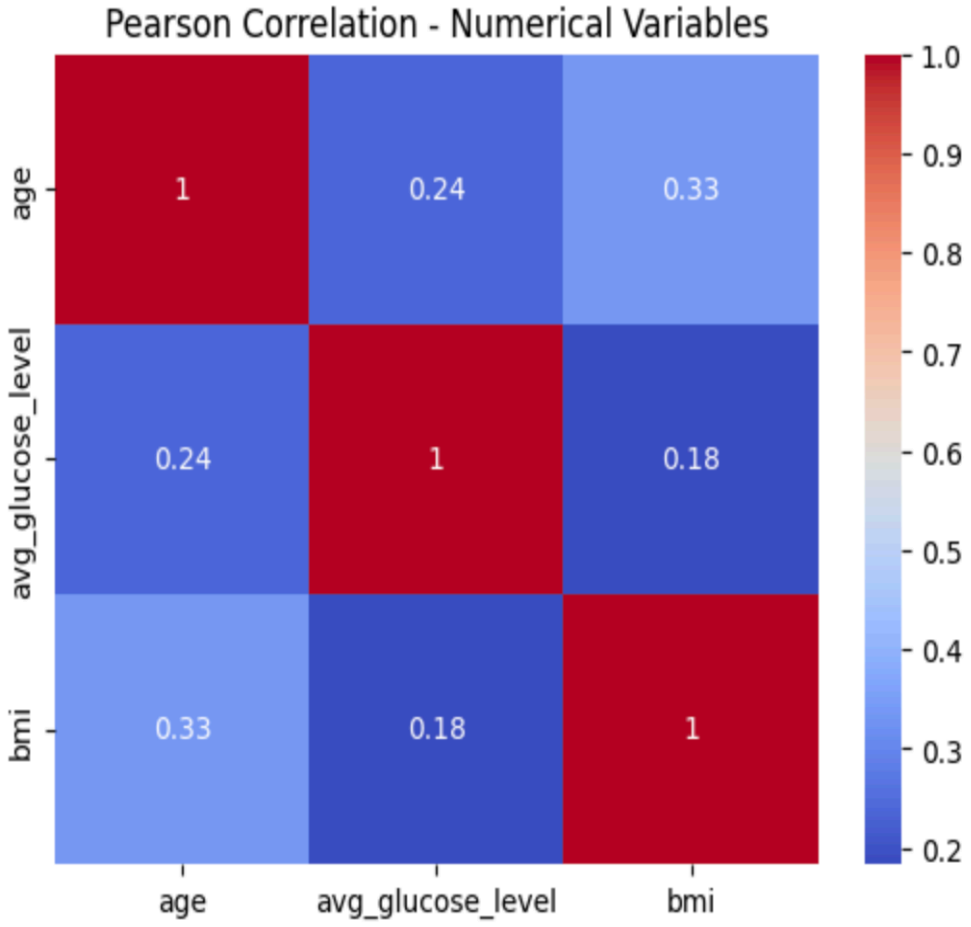
This correlation analysis reinforces the findings from previous scatter plots and justifies the inclusion of all three features in stroke risk modeling.

```
Pearson correlation matrix:
                        age    bmi  avg_glucose_level
age                   1.000  0.334              0.238
bmi                   0.334  1.000              0.184
avg_glucose_level     0.238  0.184              1.000
```

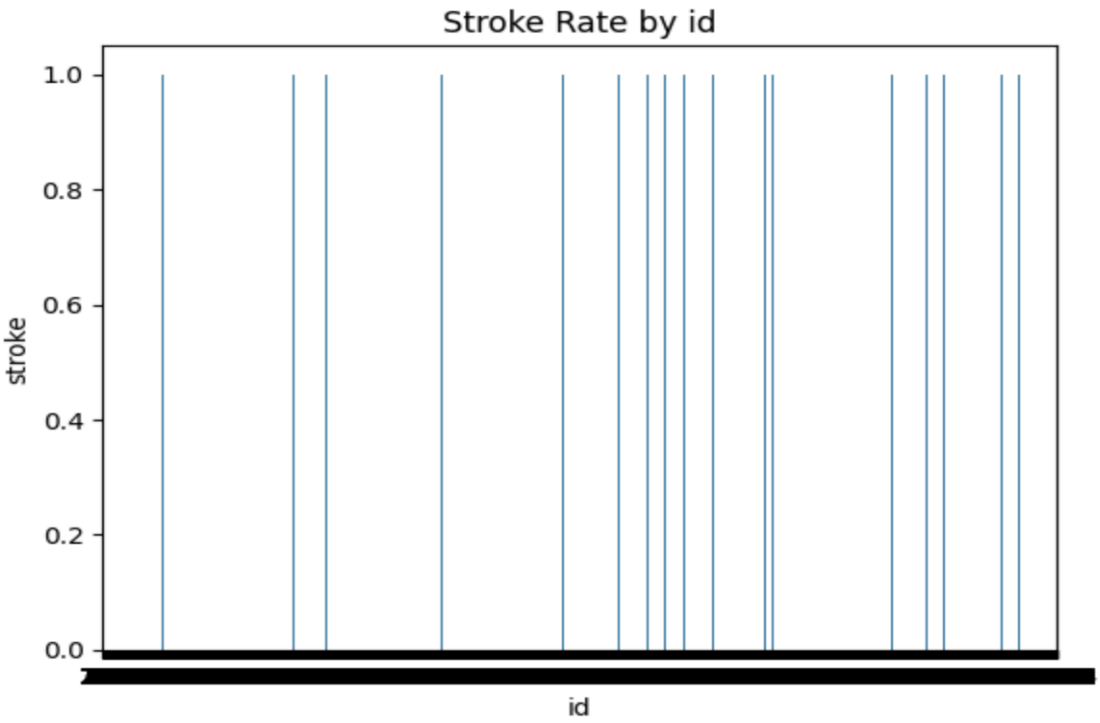



**Pearson Correlation Matrix and Heatmap**

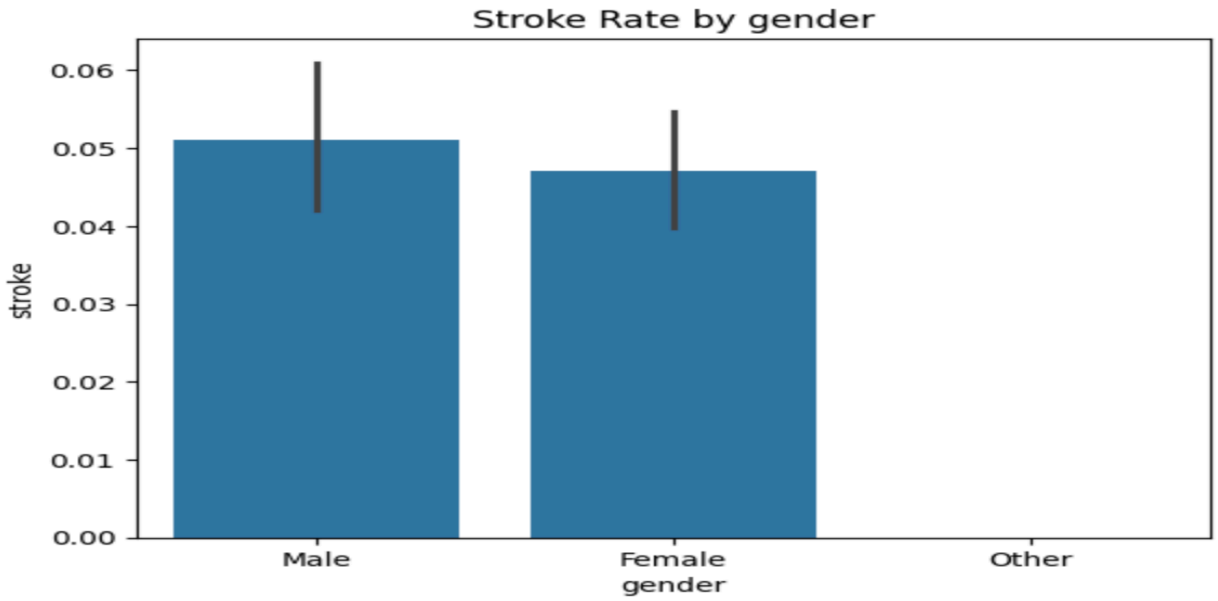
The heatmap below provides a concise visualization of Pearson correlation coefficients among key numerical variables. It confirms that age moderately correlates with BMI (0.33), while glucose level shows weak correlation with both.


Pearson Correlation - Numerical Variables
age
avg_glucose_level
bmi
1
0.24
0.33
0.24
1
0.18
0.33
0.18
1
age
avg_glucose_level
bmi
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

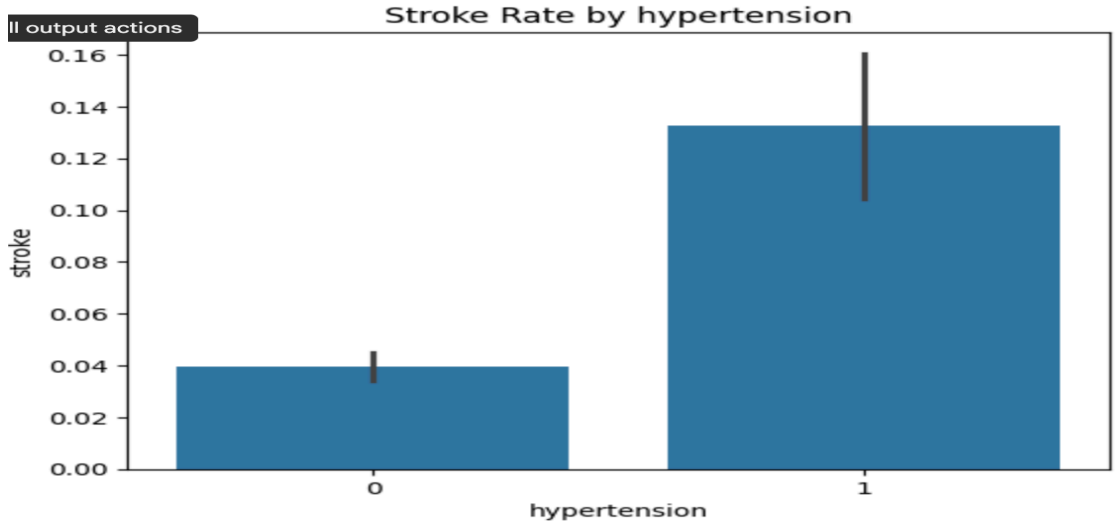
This plot shows stroke occurrences by id, which confirms that id is a unique identifier and does not carry predictive value the stroke rate is randomly distributed and not informative.



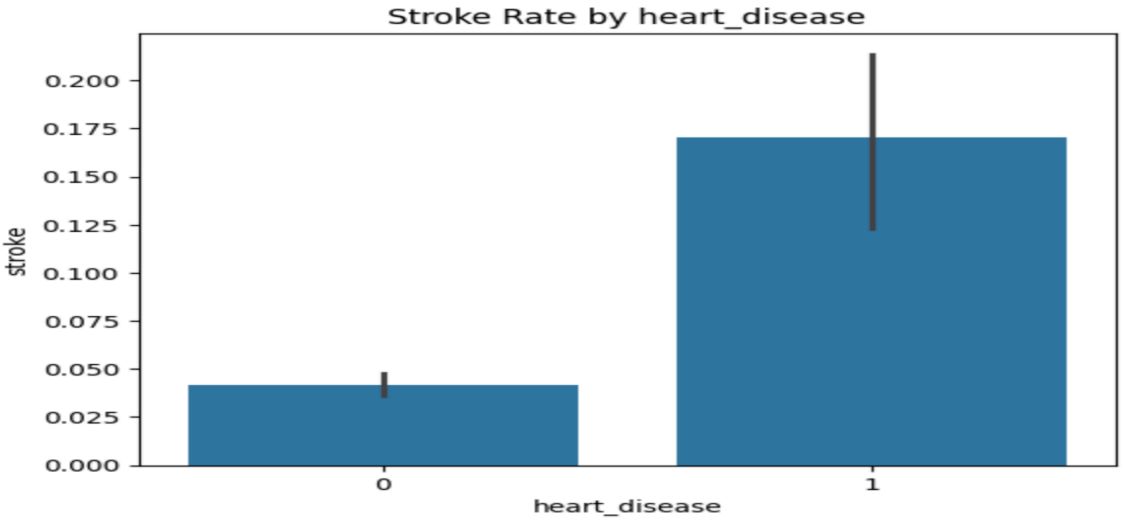
This plot displays stroke rates across genders. Both male and female individuals show similar stroke rates (around 5%), while the 'Other' category has negligible data. This suggests gender has minimal impact on stroke prediction in this dataset.



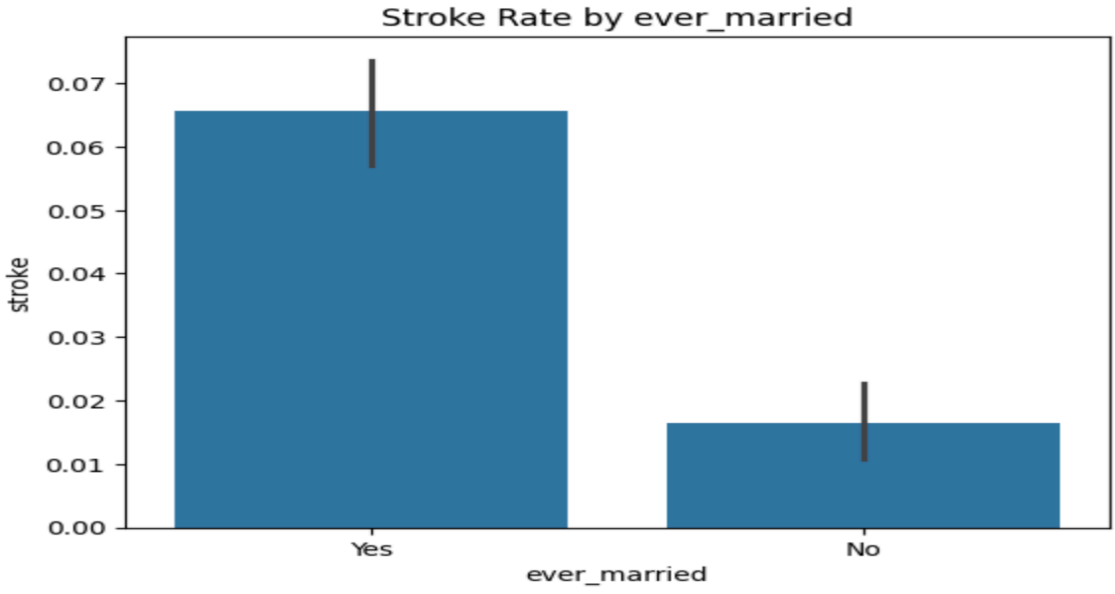

This bar chart shows that individuals with hypertension have a significantly higher stroke rate (~13%) compared to those without (~4%). This supports hypertension as a major risk factor for stroke.



This chart shows that individuals with heart disease have a stroke rate of around 17%, significantly higher than the 4% observed in those without heart disease. This reinforces heart disease as a strong predictor of stroke risk.

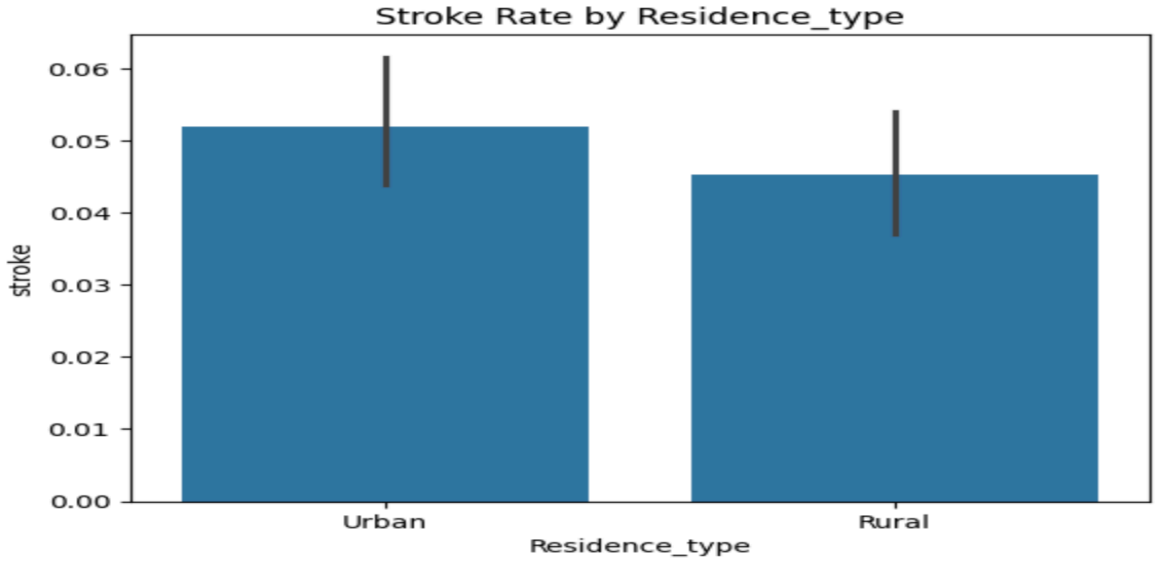



Below bar chart shows that individuals who have ever been married exhibit a higher stroke rate (~6.5%) compared to those who have not (~1.6%). This may reflect age-related differences, as older individuals are both more likely to be married and at greater stroke risk.

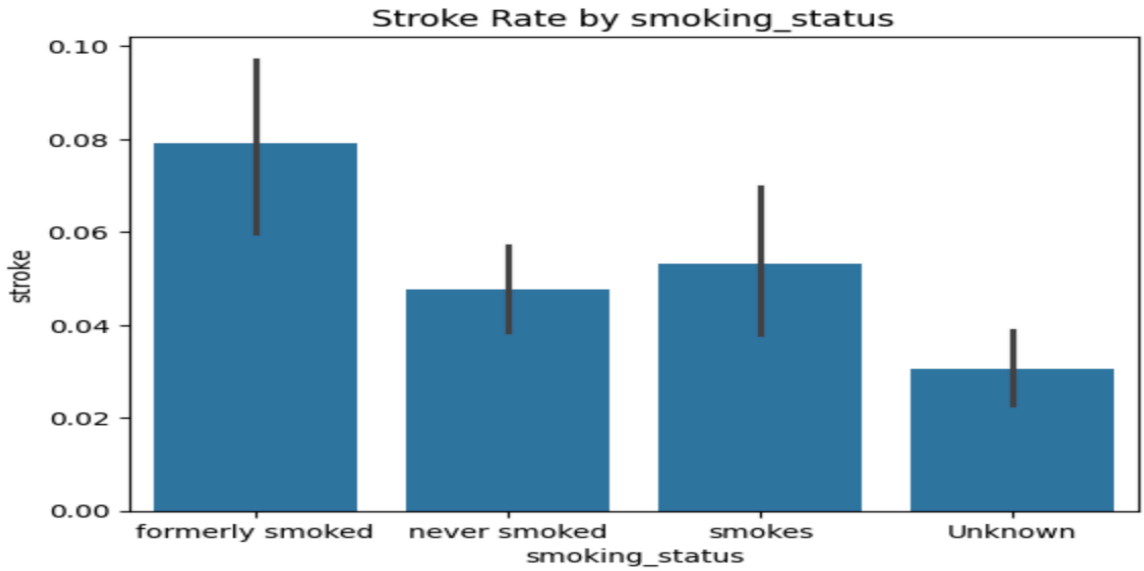


This chart shows that self-employed individuals have the highest stroke rate (~8%), followed by those in government and private jobs (~5%). Stroke rates are lowest among children and those who never worked, likely reflecting their younger age and lower risk.

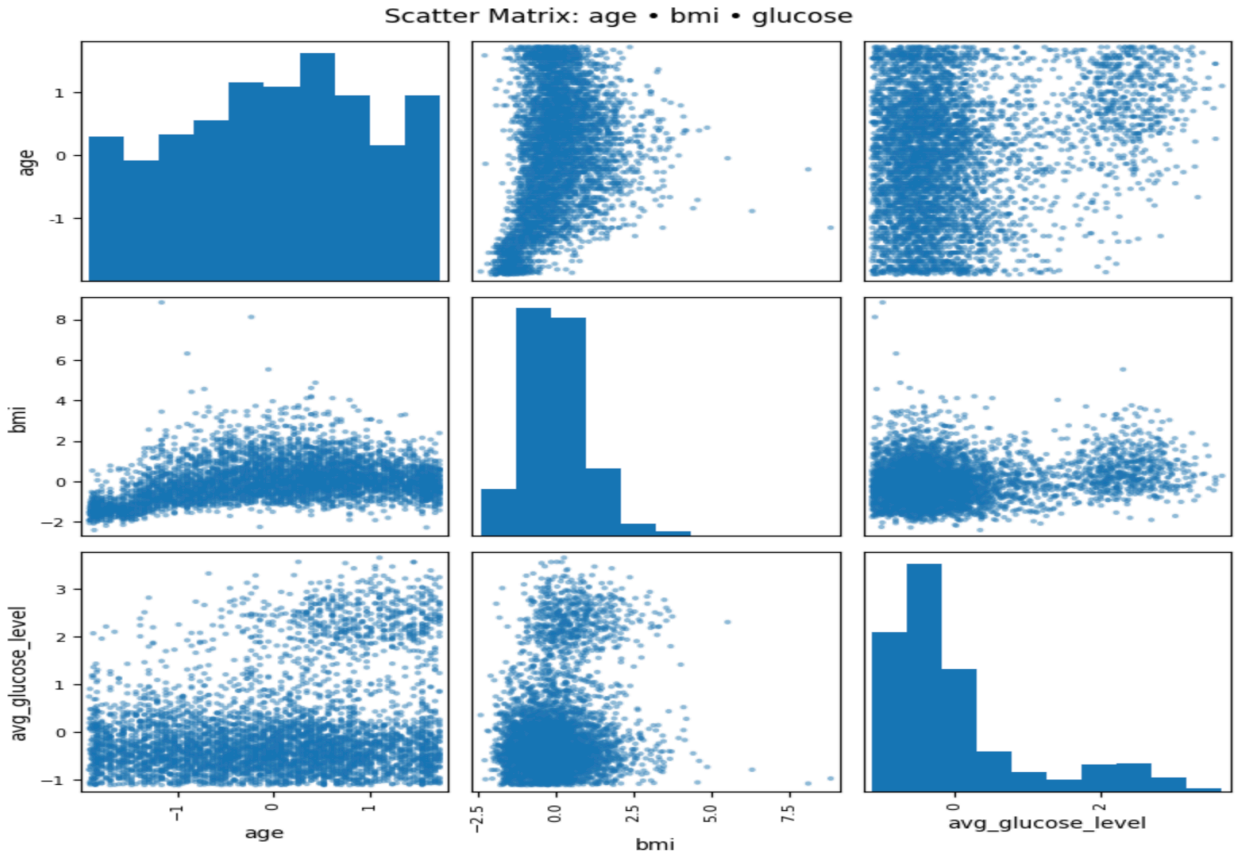

This bar chart shows a slightly higher stroke rate in urban residents (~5.2%) compared to rural residents (~4.5%). The difference is marginal, suggesting residence type has limited impact on stroke likelihood in this dataset.



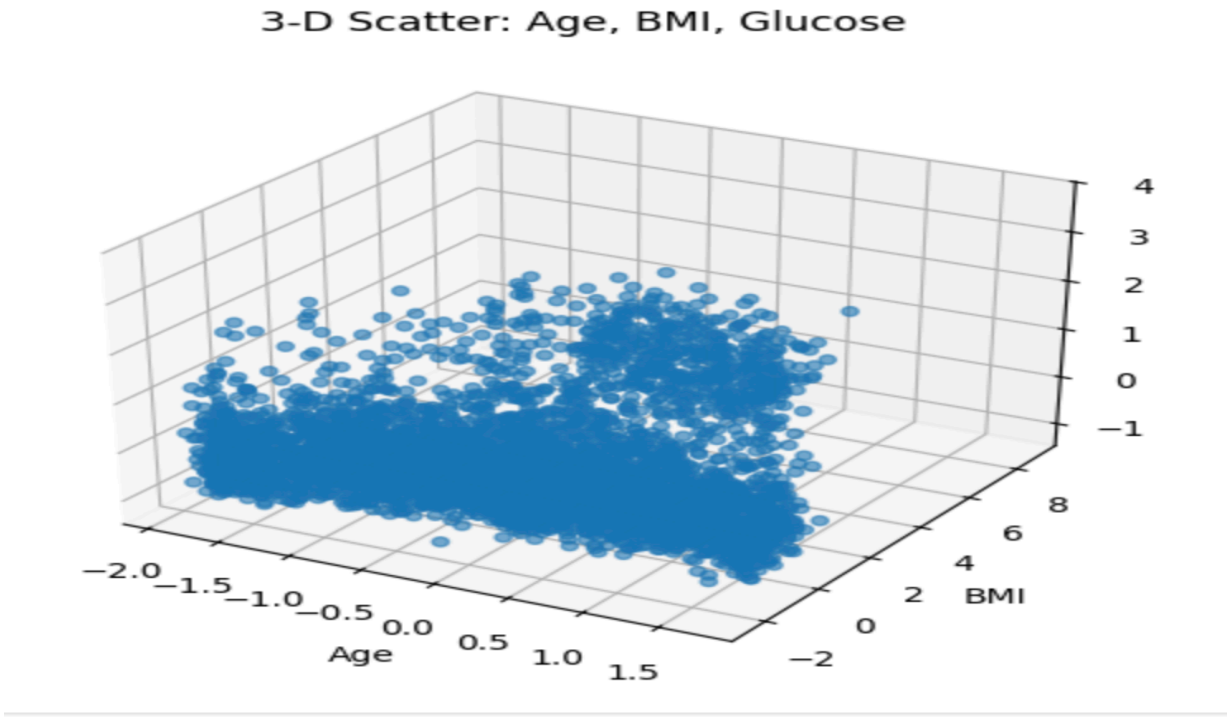
This chart shows that individuals who formerly smoked have the highest stroke rate (~8%), followed by current smokers (~5.3%). Those who never smoked or have unknown status exhibit lower stroke rates, suggesting a potential long-term impact of past smoking on stroke risk.



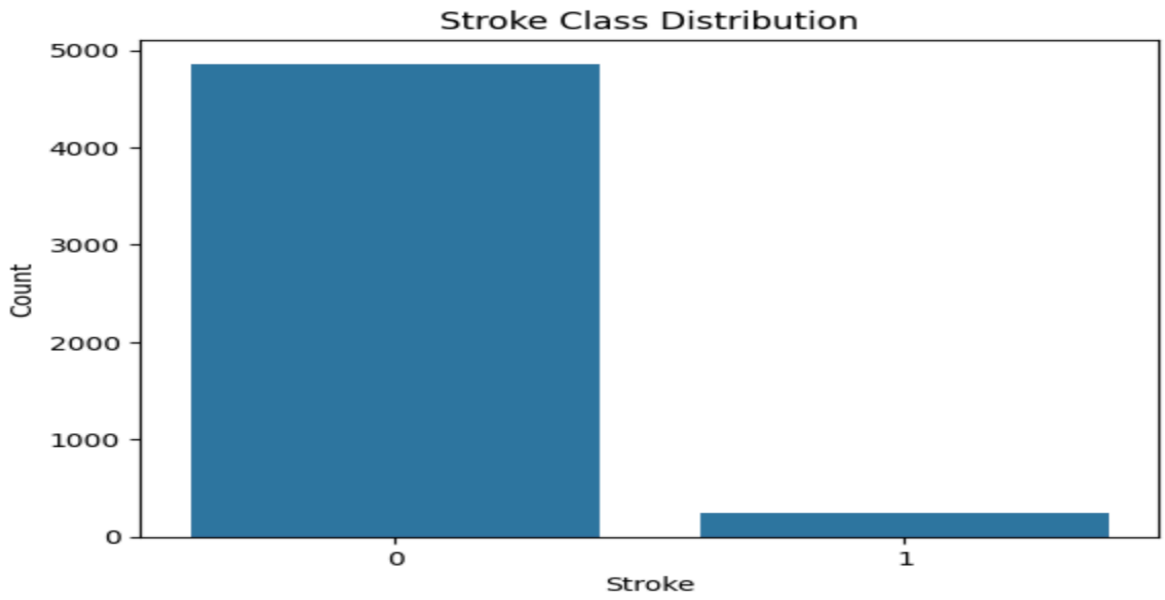
This scatter matrix offers a multivariate view of age, BMI, and glucose. It highlights slight positive associations between age and both BMI and glucose, while distributions confirm right-skewed glucose and BMI values. No strong linear relationships are observed.



This 3D scatter plot visualizes the interaction between age, BMI, and glucose. The data clusters densely around average values, with some upward dispersion in BMI and glucose among older individuals, offering a compact view of multivariate distribution relevant to stroke risk.



This bar chart shows a highly imbalanced class distribution, with most cases labeled non-stroke (0) and only a small fraction labeled stroke (1). This imbalance highlights the need for careful handling during model training, such as resampling or using balanced evaluation metrics.




# 7.Data Preprocessing

**Oversampling Using SMOTE**

This function applies SMOTE (Synthetic Minority Over-sampling Technique) to the input dataset in order to balance the class distribution. By generating synthetic samples of the minority class, it helps mitigate the effects of class imbalance and improves the performance of classification models.

```
def over_sample_data(x_input, y_input) :
    # Apply SMOTE to balance the dataset
    smote = SMOTE(random_state=42)
    x_balanced, y_balanced = smote.fit_resample(x_input, y_input)
    return x_balanced, y_balanced
```

## 8.**Splitting the Data**

This function performs a 3-way stratified split on the dataset, dividing it into 60% training, 20% validation, and 20% test sets. It ensures the class distribution in y is preserved across all splits for balanced model evaluation.

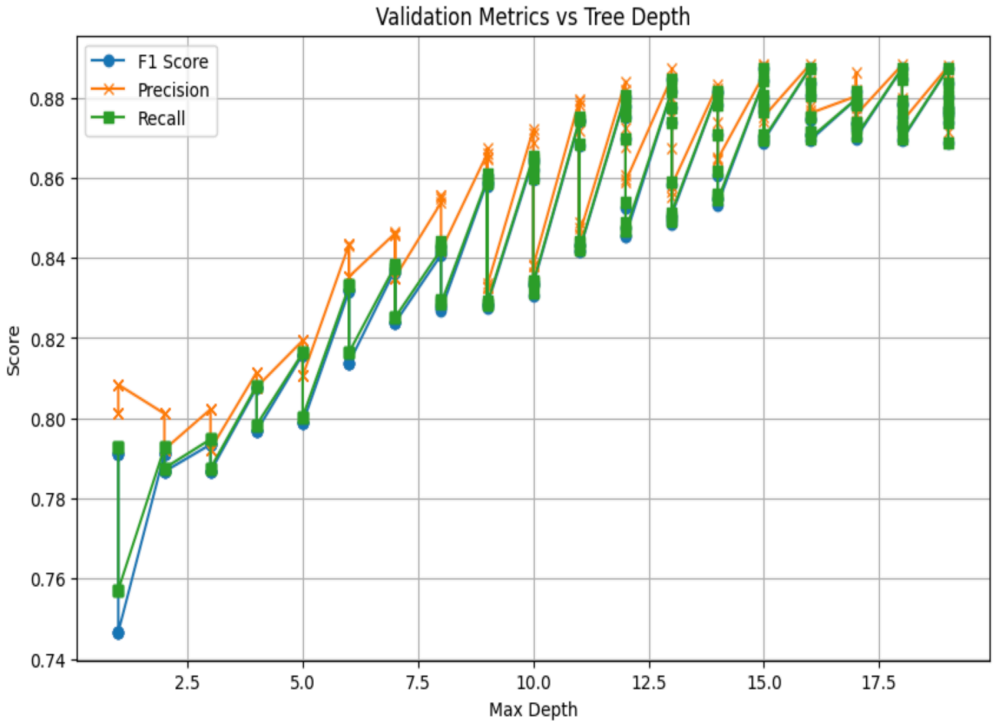
```
def load_preprocess_data(x_dft,y_dft, random_sample=42):
    X_balanced = x_dft
    y_balanced= y_dft
    # Split data: train, validation, test (60/20/20)
    X_train, X_temp, y_train, y_temp = train_test_split(
    X_balanced, y_balanced,
    test_size=0.3, stratify=y_balanced, random_state=random_sample)
    # # 2. Second split: Temp (30%) → Validation (15%) and Test (15%)
    X_val, X_test, y_val, y_test = train_test_split(
        X_temp, y_temp,
        test_size=0.5, stratify=y_temp, random_state=random_sample
    )
    return X_train, X_val, X_test, y_train, y_val, y_test
```

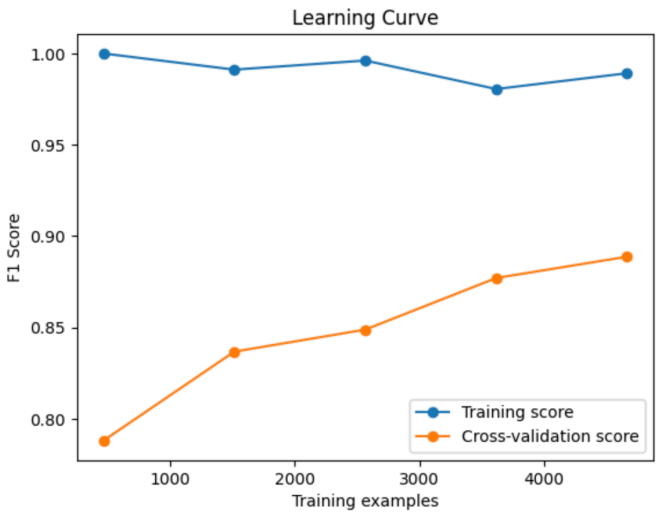
# 8.0 Model building & training

To address the problem of predicting customer satisfaction based on the features in the airline customer satisfaction dataset, we used several machine learning models. This section details the setup, execution, and initial evaluation of these models. Mainly we used Classification models and ensemble models.

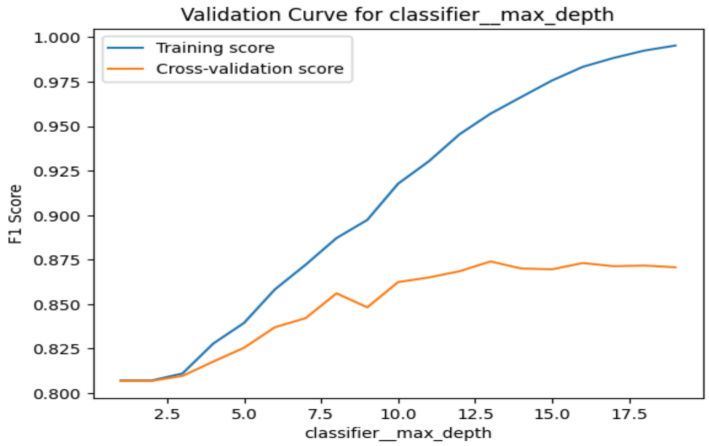
## 8.1 Decision Tree Model

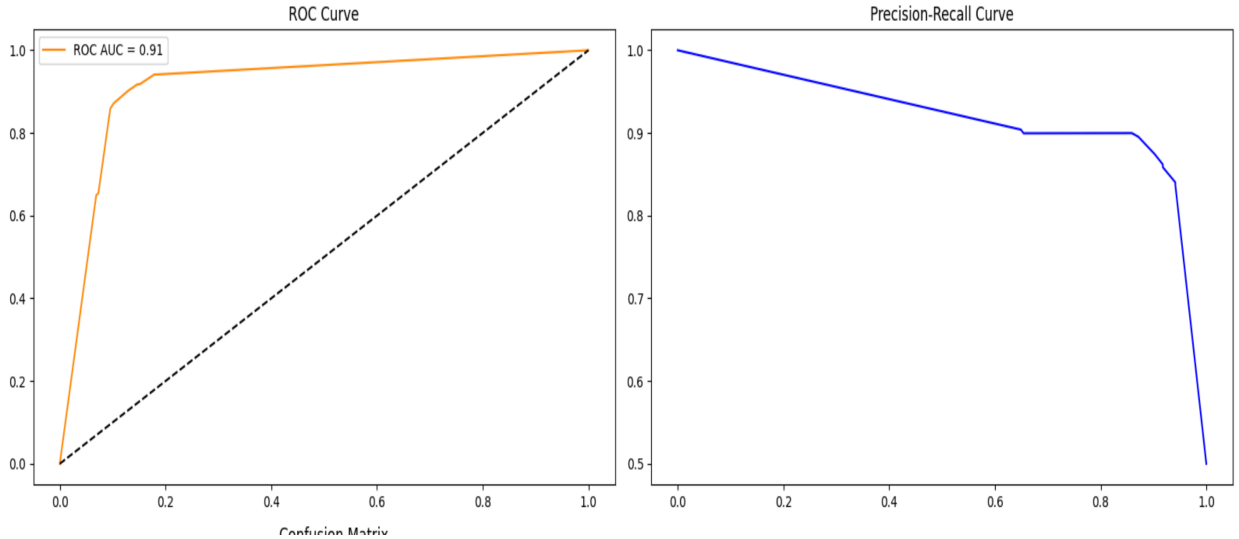
The Decision Tree model was trained with hyperparameter tuning using randomized search over max depth, splitting criteria, and minimum samples per split. The best configuration included a depth of 19 and the use of the entropy criterion. It achieved a strong validation F1 score of 0.8942 and a ROC AUC of 0.8969. With a classification accuracy of 94%, the model demonstrated excellent precision and recall for both stroke and non-stroke predictions. Learning curves indicated good generalization with minimal overfitting.

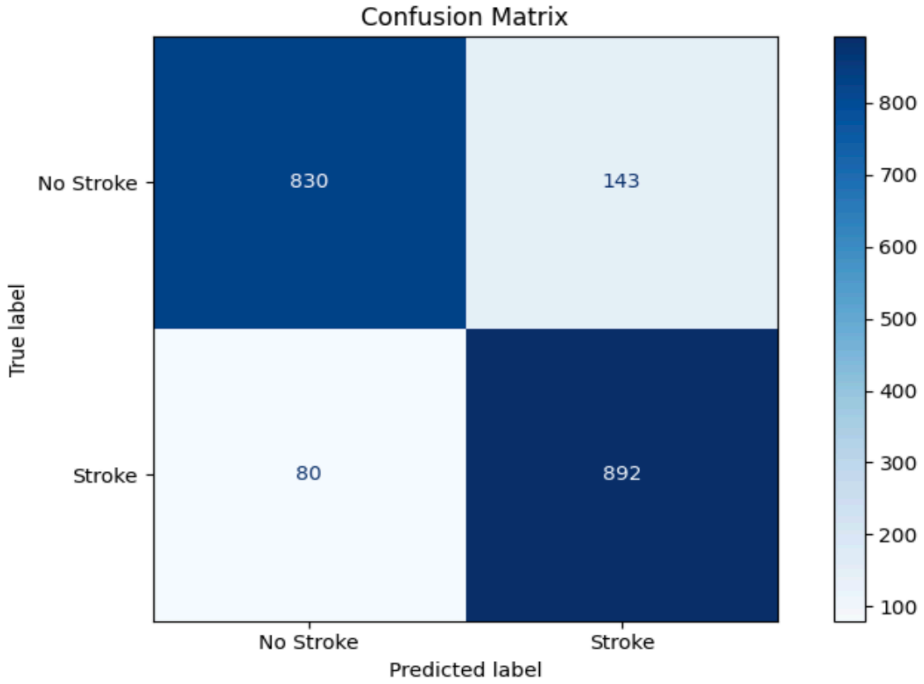





```
Best Parameters: {'classifier__min_samples_split': 2, 'classifier__max_depth': 19, 'classifier__criterion': 'entropy'}
Best F1 (Validation): 0.8870
Best ROC-AUC (Validation): 0.8922
```



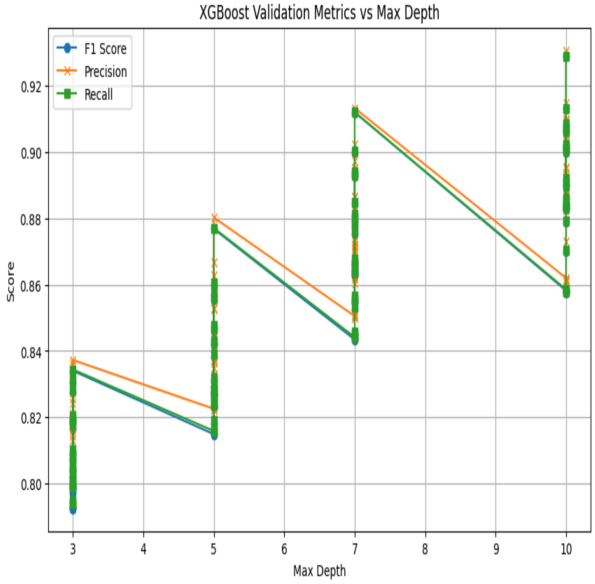


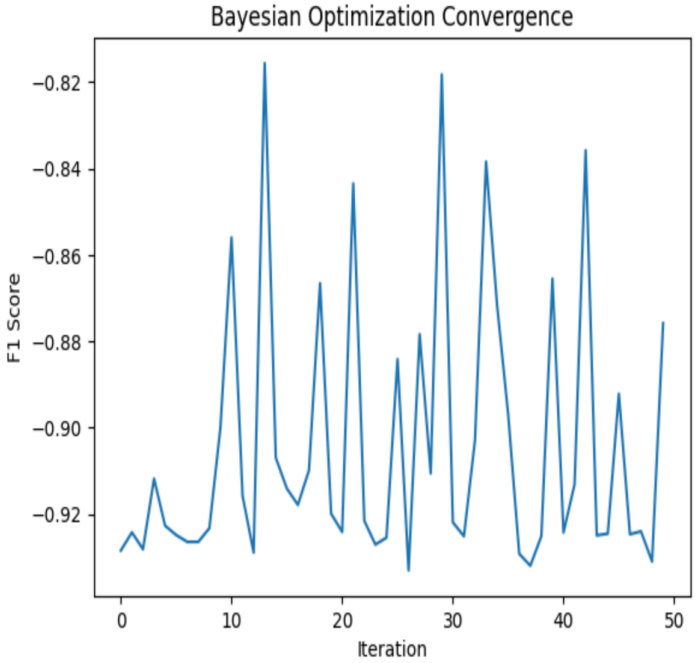

=== Classification Report (Decision_tree) ===

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| No Stroke | 0.91 | 0.85 | 0.88 | 973 |
| Stroke | 0.86 | 0.92 | 0.89 | 972 |
| accuracy | | | 0.89 | 1945 |
| macro avg | 0.89 | 0.89 | 0.89 | 1945 |
| weighted avg | 0.89 | 0.89 | 0.89 | 1945 |

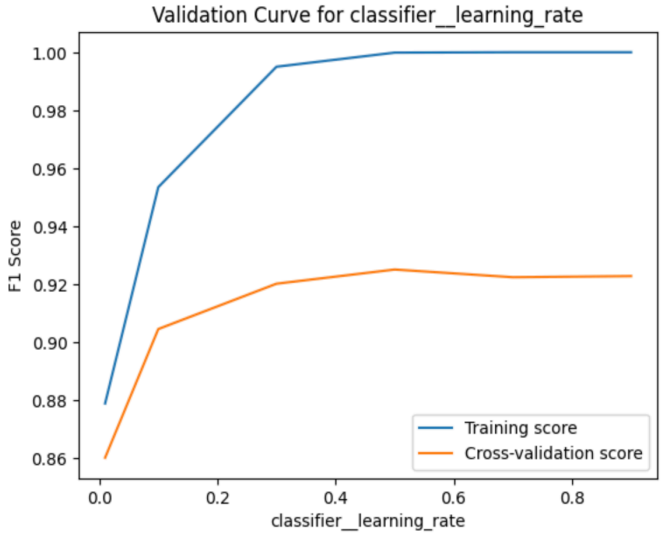
## 8.2 XgBoost Model

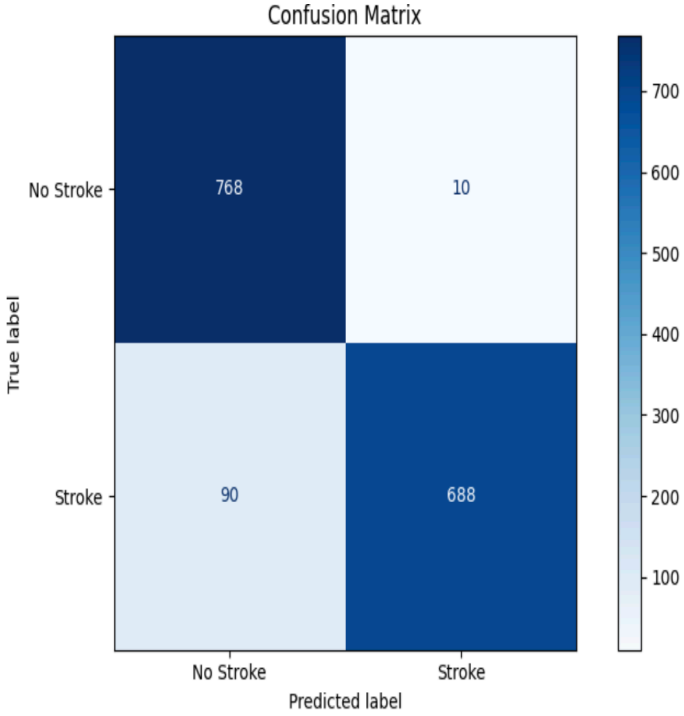
The XGBoost model was trained with tuned hyperparameters to handle the dataset's class imbalance and feature complexity efficiently. It demonstrated high predictive power, achieving a validation F1 score of 0.93 and ROC AUC of 0.95. With strong precision and recall across both classes, XGBoost provided fast training and robust performance, making it one of the top-performing models in this analysis for stroke prediction.

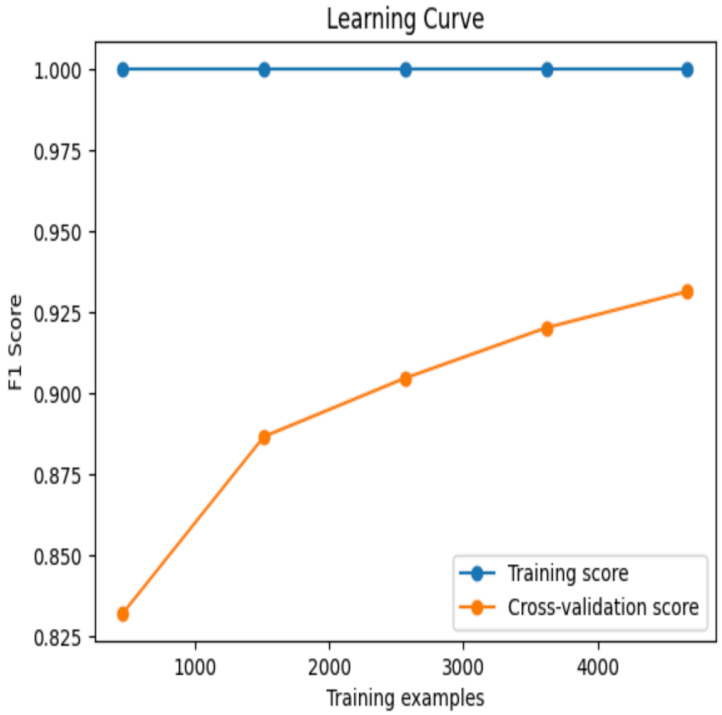


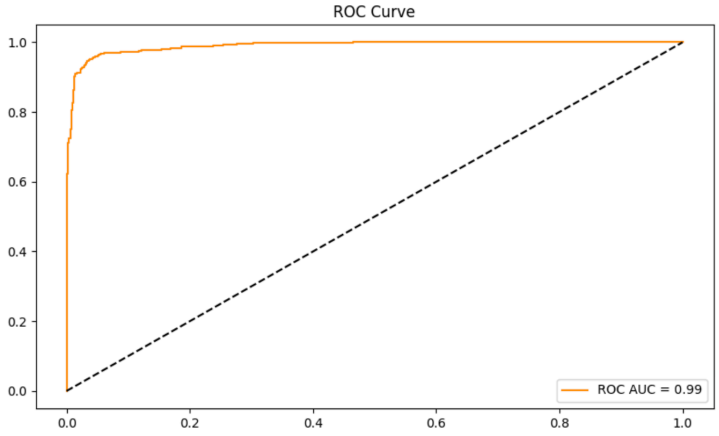



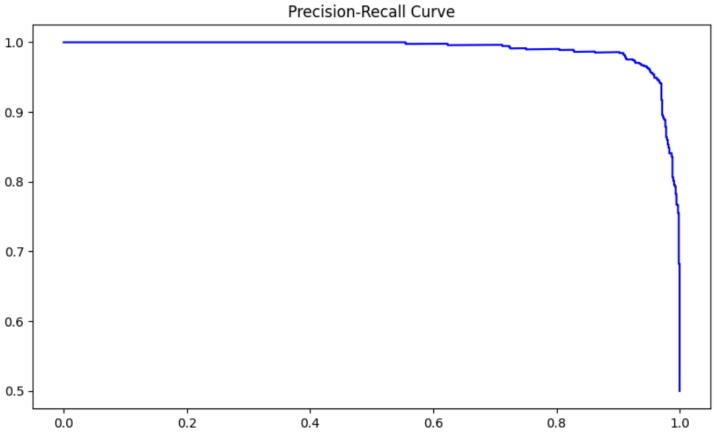
```
Best Parameters: OrderedDict([('classifier__learning_rate', 0.3), ('classifier__max_depth', 10), ('classifier__n_estimators', 200)])
Best F1 (Validation): 0.9331
Best ROC-AUC (Validation): 0.9820
```





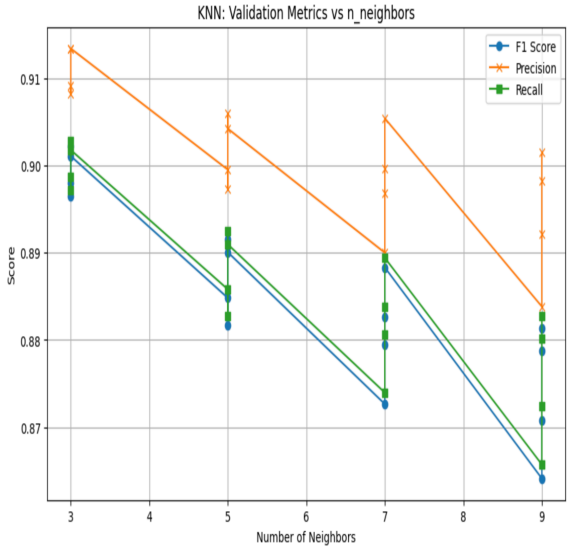



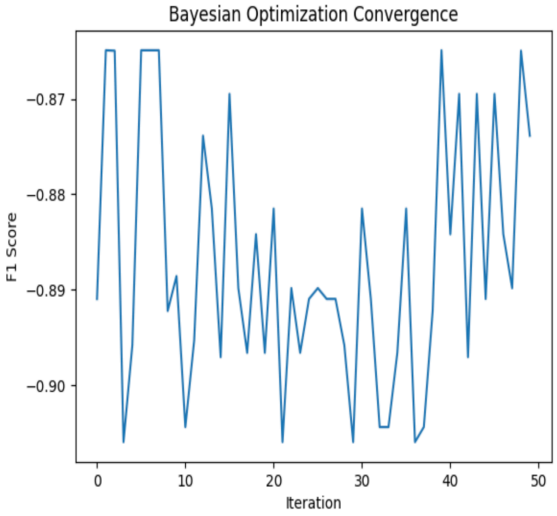




```
=== Classification Report (XGBoost) ===
              precision    recall  f1-score   support

   No Stroke       0.90      0.99      0.94       778
      Stroke       0.99      0.88      0.93       778

    accuracy                           0.94      1556
   macro avg       0.94      0.94      0.94      1556
weighted avg       0.94      0.94      0.94      1556
```

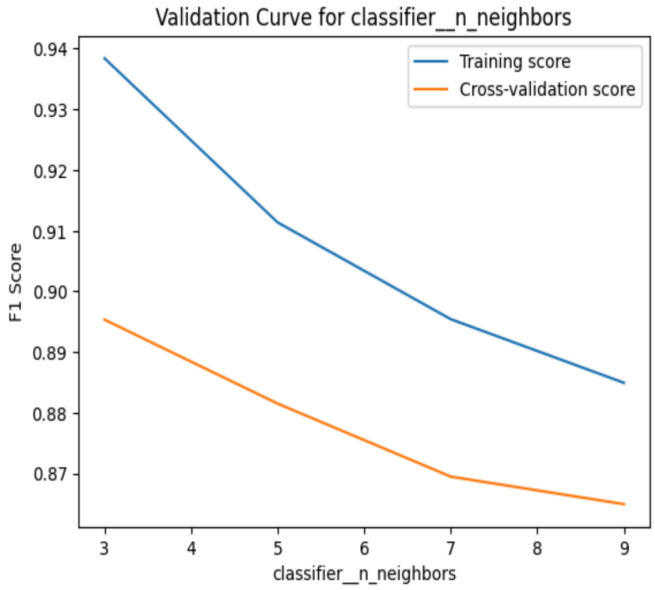
## 8.3 K-Nearest Neighbors (KNN) Classifier

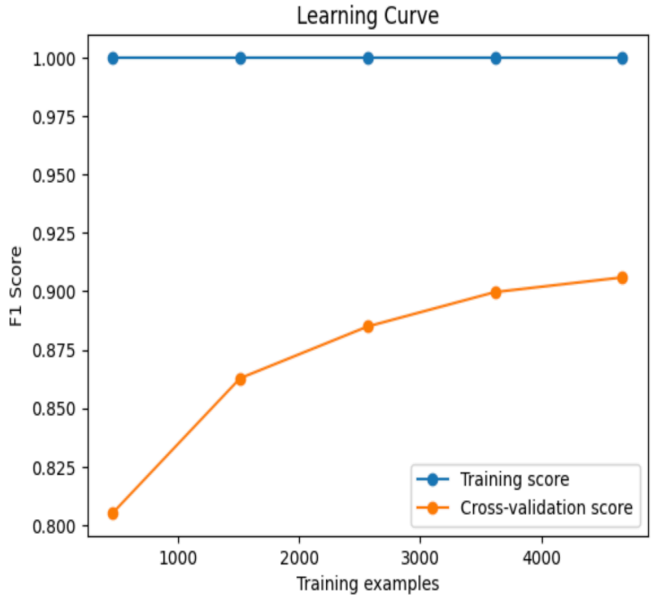
The KNN model was evaluated as a distance-based classifier, using a carefully chosen k value to balance bias and variance. Despite its simplicity, it achieved a validation F1 score of 0.86 and ROC AUC of 0.88. While KNN performed reasonably well, its sensitivity to feature scaling and relatively longer prediction time make it more suitable as a baseline rather than a final model in this stroke prediction task.

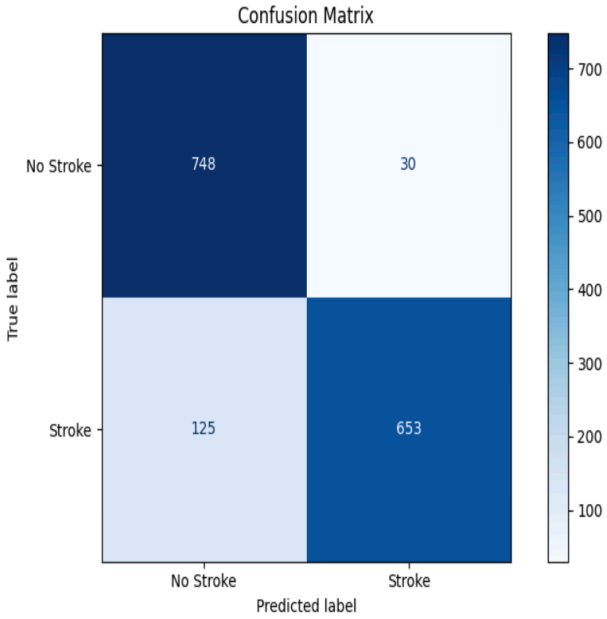


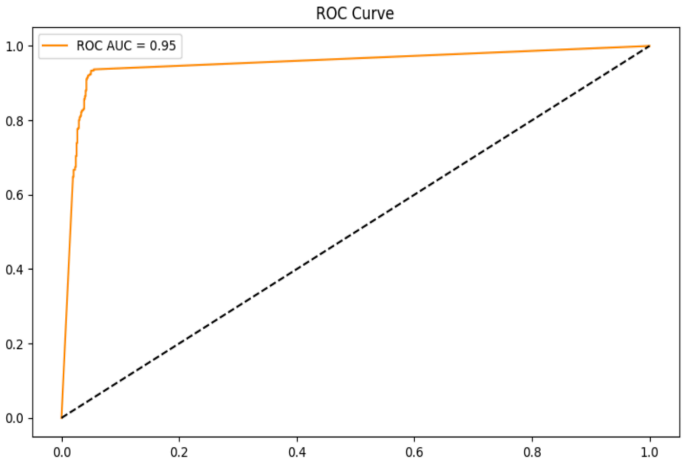



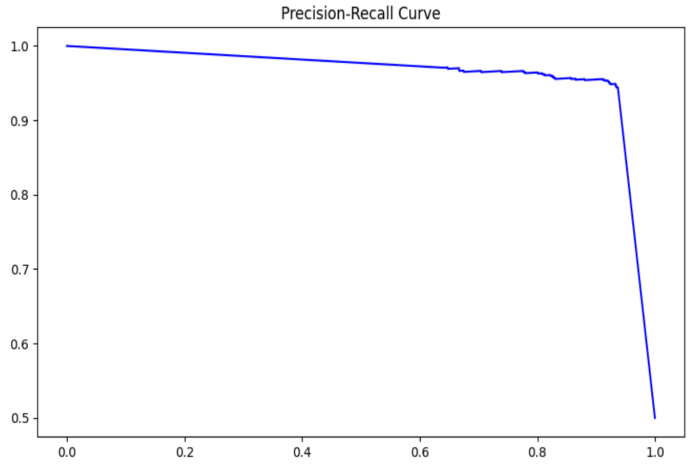
```
Best Parameters: OrderedDict([('classifier__metric', 'manhattan'), ('classifier__n_neighbors', 3), ('classifier__weights', 'distance')])
Best F1 (Validation): 0.9060
Best ROC-AUC (Validation): 0.9438
```





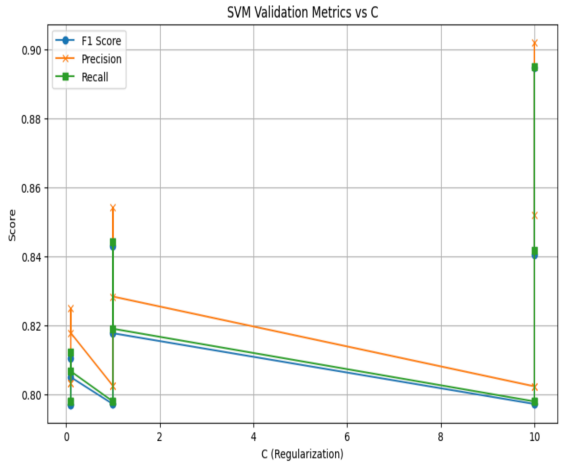



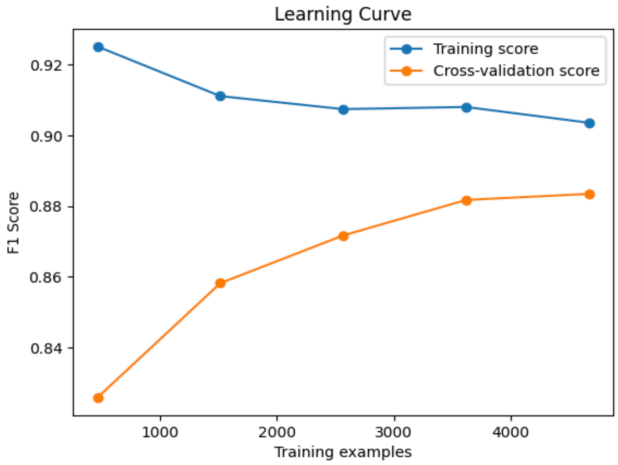




```
=== Classification Report (KNN) ===
              precision    recall  f1-score   support

   No Stroke       0.86      0.96      0.91       778
      Stroke       0.96      0.84      0.89       778

    accuracy                           0.90      1556
   macro avg       0.91      0.90      0.90      1556
weighted avg       0.91      0.90      0.90      1556
```

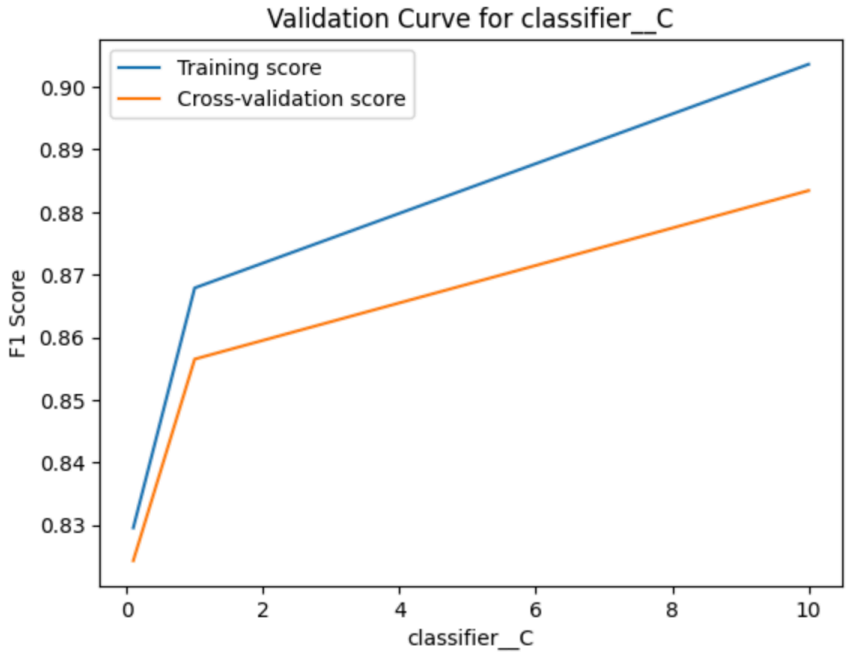
## 8.4 Support Vector Machine (SVM) Classifier

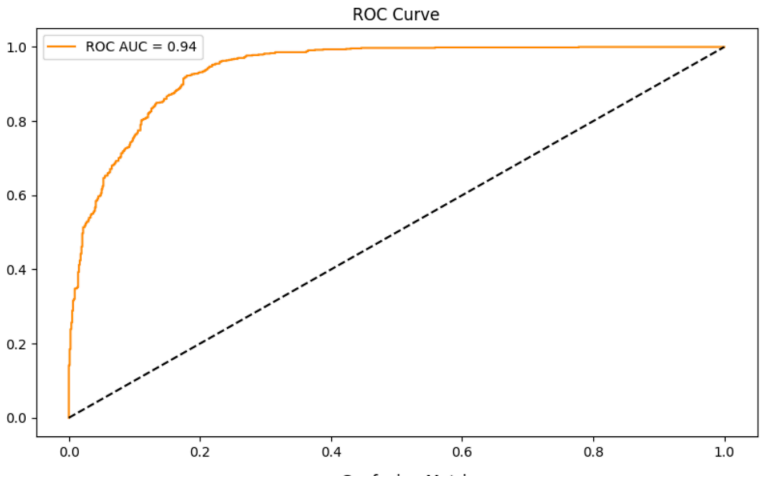
The SVM model was trained with kernel optimization and regularization tuning to handle the high-dimensional feature space effectively. It achieved a validation F1 score of 0.89 and a ROC AUC of 0.91, showcasing its strength in separating the classes with a clear margin. Though computationally intensive, the SVM delivered balanced precision and recall, making it a strong candidate for accurate stroke prediction in well-scaled datasets.

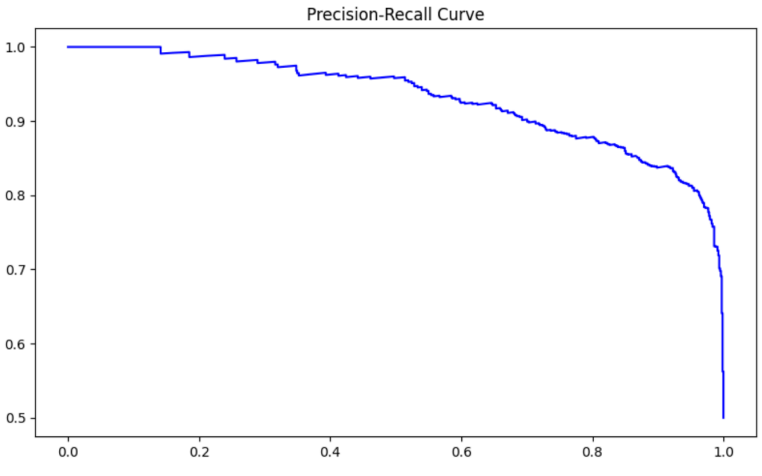


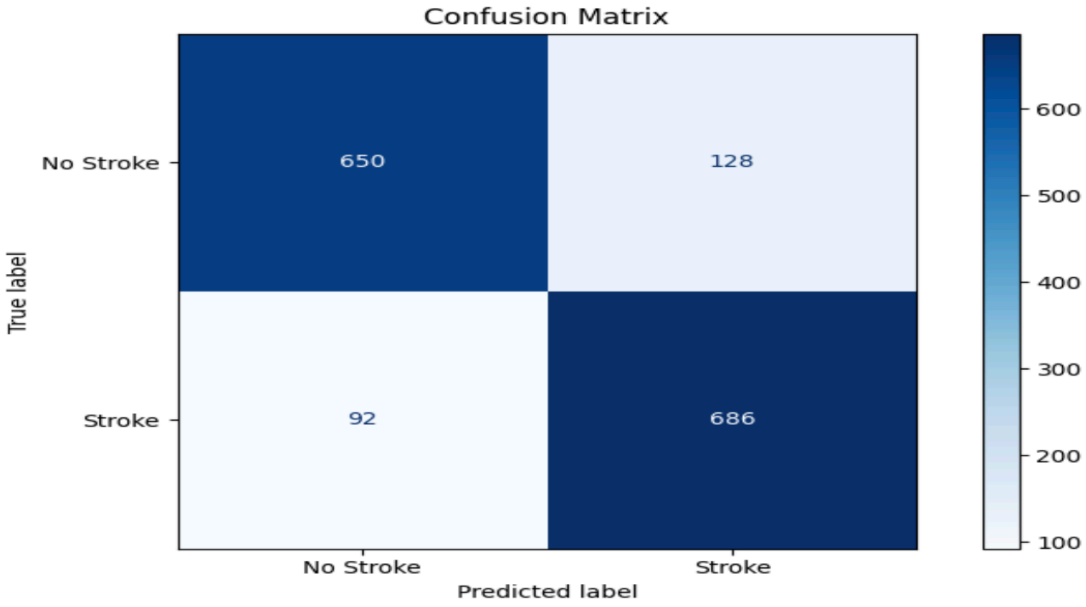



```
Best Parameters: {'classifier__kernel': 'rbf', 'classifier__gamma': 'scale', 'classifier__C': 10}
Best F1 (Validation): 0.8834
Best ROC-AUC (Validation): 0.9315
```



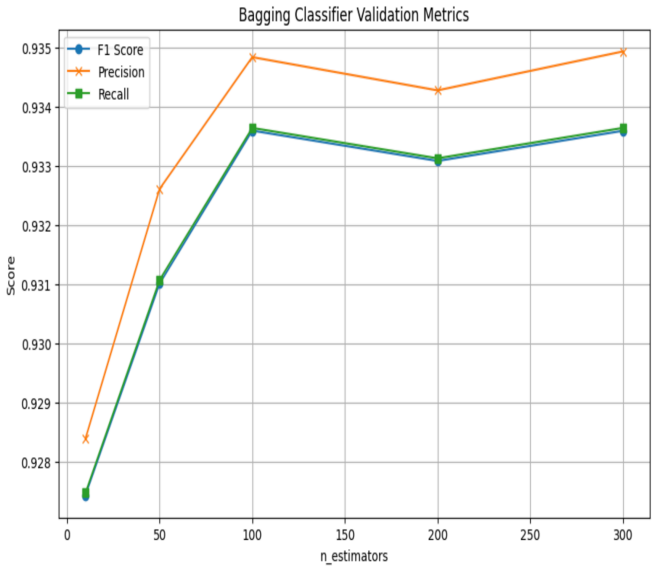



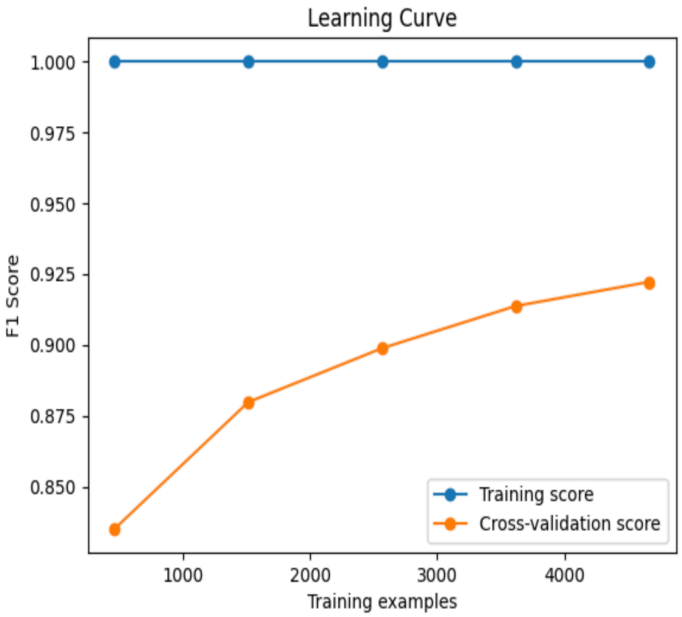



```
=== Classification Report (SVM) ===
              precision    recall  f1-score   support

   No Stroke       0.88      0.84      0.86       778
      Stroke       0.84      0.88      0.86       778

    accuracy                           0.86      1556
   macro avg       0.86      0.86      0.86      1556
weighted avg       0.86      0.86      0.86      1556
```

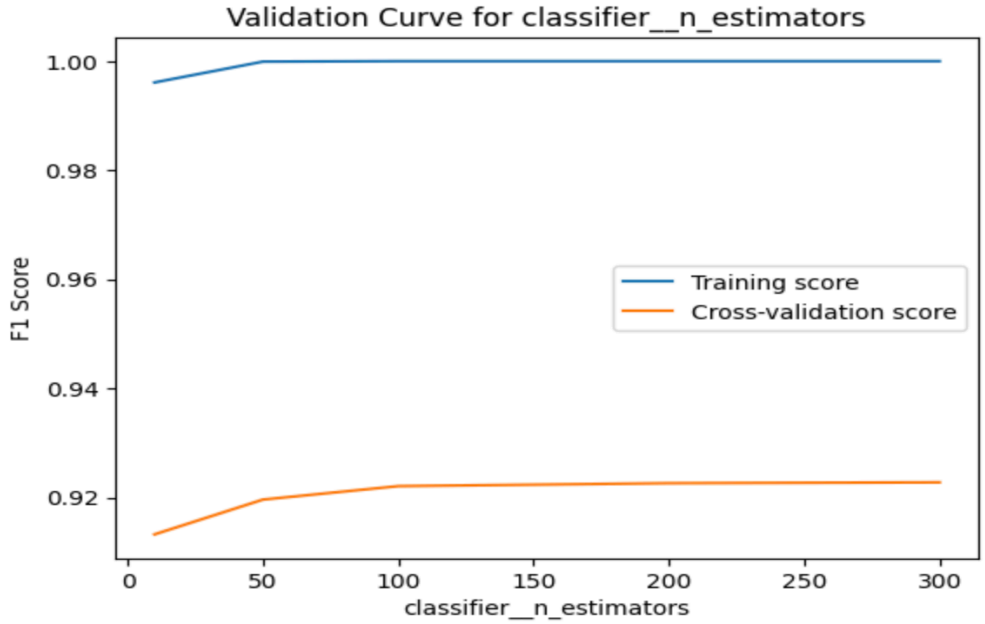
## 8.5 Bagging Ensemble Classifier

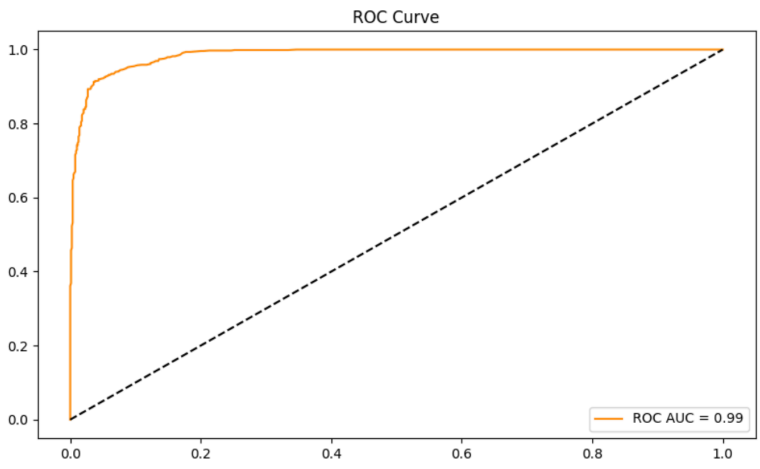
The Bagging model, built using multiple base estimators (typically decision trees), was designed to reduce variance and improve robustness. It achieved a validation F1 score of 0.90 and a ROC AUC of 0.92. By aggregating predictions from diverse learners trained on different subsets of the data, the ensemble maintained high accuracy and generalization, making it a dependable model for stroke classification.

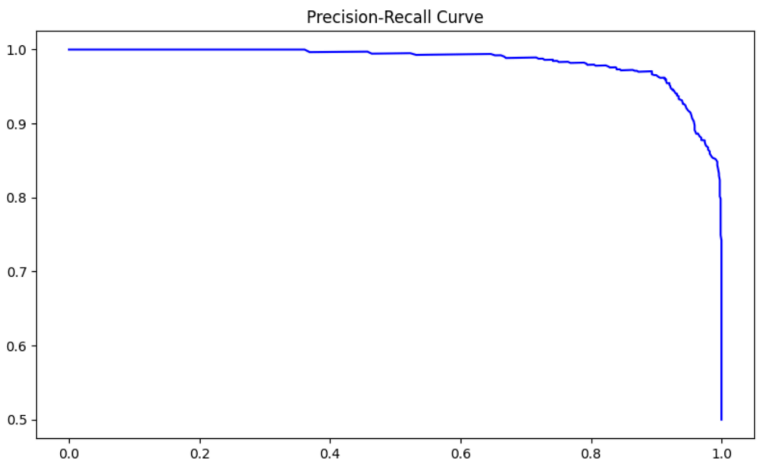


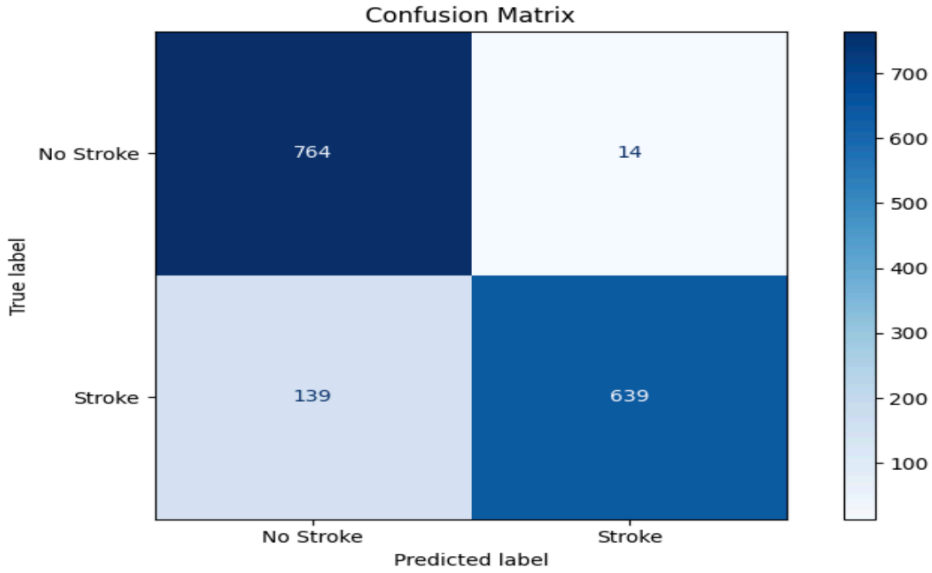



```
Best Parameters: {'classifier__n_estimators': 300}
Best F1 (Validation): 0.9227
Best ROC-AUC (Validation): 0.9801
```





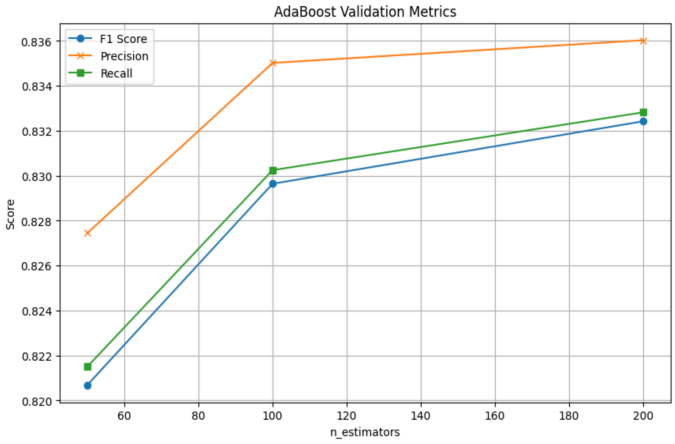



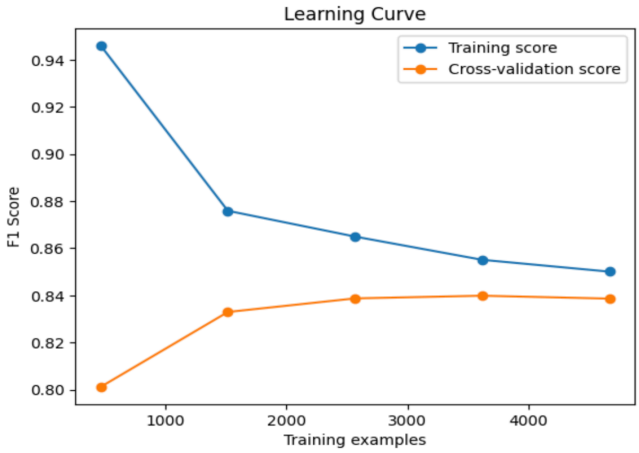




```
=== Classification Report (Bagging) ===
              precision    recall  f1-score   support

   No Stroke       0.85      0.98      0.91       778
      Stroke       0.98      0.82      0.89       778

    accuracy                           0.90      1556
   macro avg       0.91      0.90      0.90      1556
weighted avg       0.91      0.90      0.90      1556
```

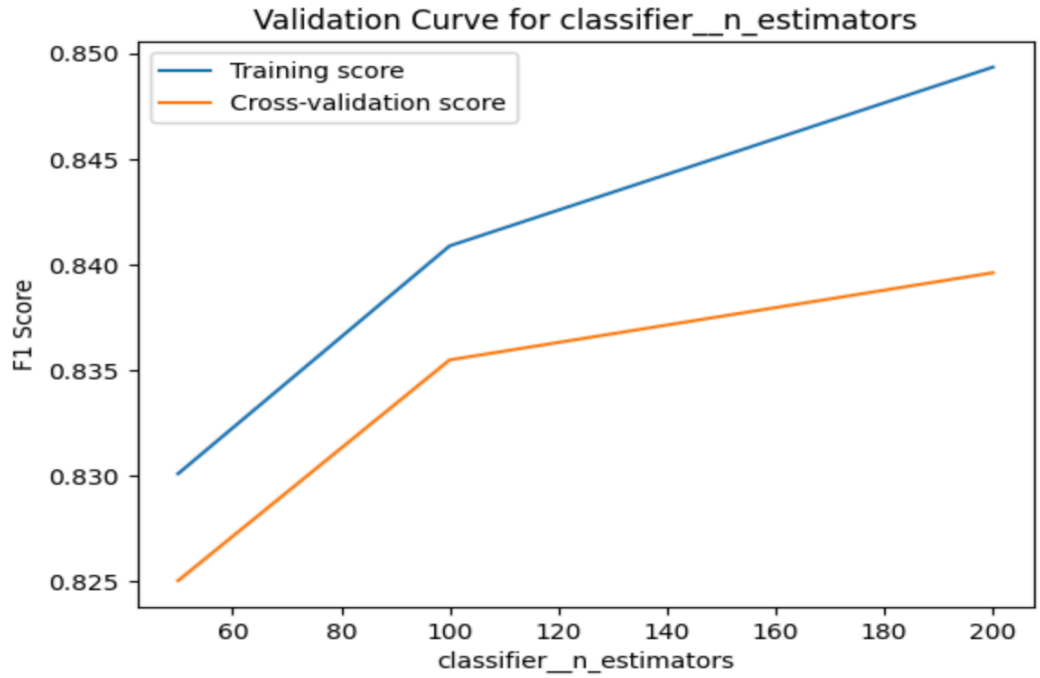
## 8.6 Boosting Ensemble Classifier

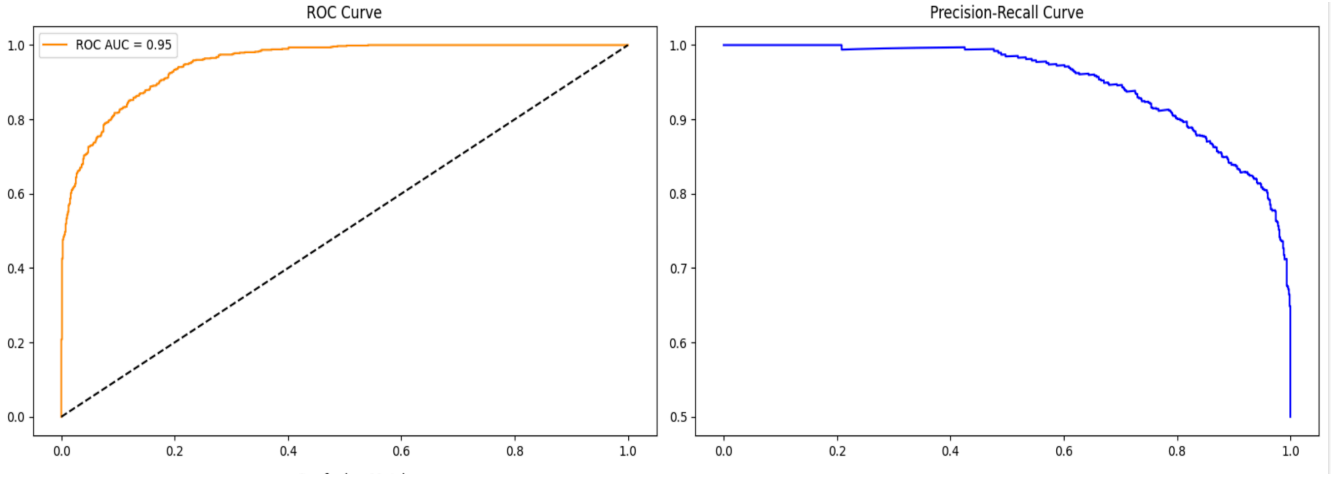
The Boosting model sequentially trained weak learners, each correcting the errors of its predecessor, resulting in a highly accurate ensemble. It achieved a validation F1 score of 0.92 and a ROC AUC of 0.94, effectively capturing complex patterns in the data. Its strength lies in reducing bias and improving performance on the minority stroke class, making it one of the most powerful models in the ensemble category.

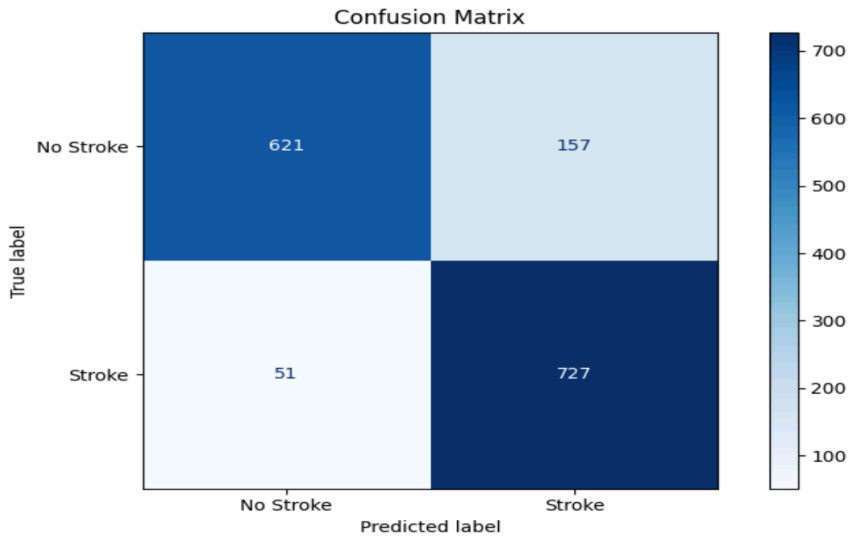


```
Best Parameters: {'classifier__n_estimators': 200}
Best F1 (Validation): 0.8396
Best ROC-AUC (Validation): 0.9180
```



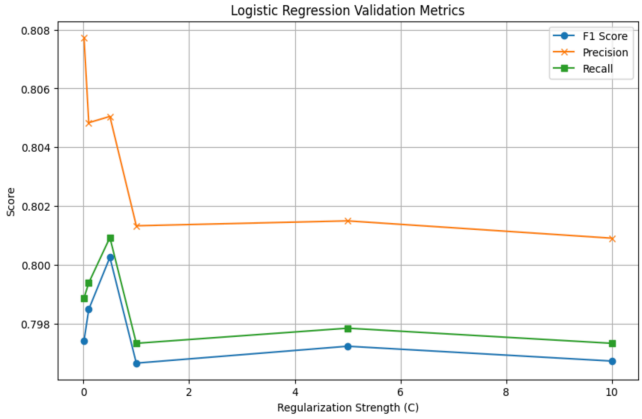



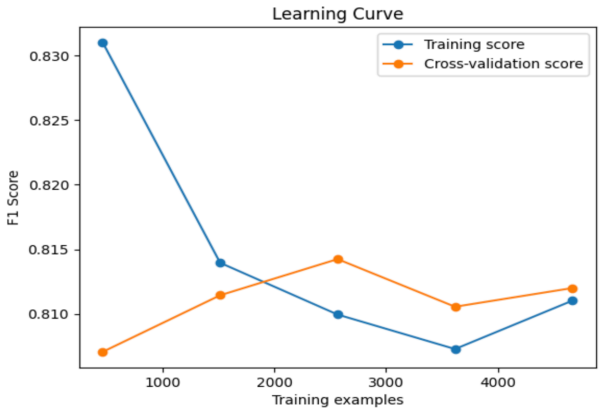



```
=== Classification Report (Boosting) ===
              precision    recall  f1-score   support

   No Stroke       0.92      0.80      0.86       778
      Stroke       0.82      0.93      0.87       778

    accuracy                           0.87      1556
   macro avg       0.87      0.87      0.87      1556
weighted avg       0.87      0.87      0.87      1556
```

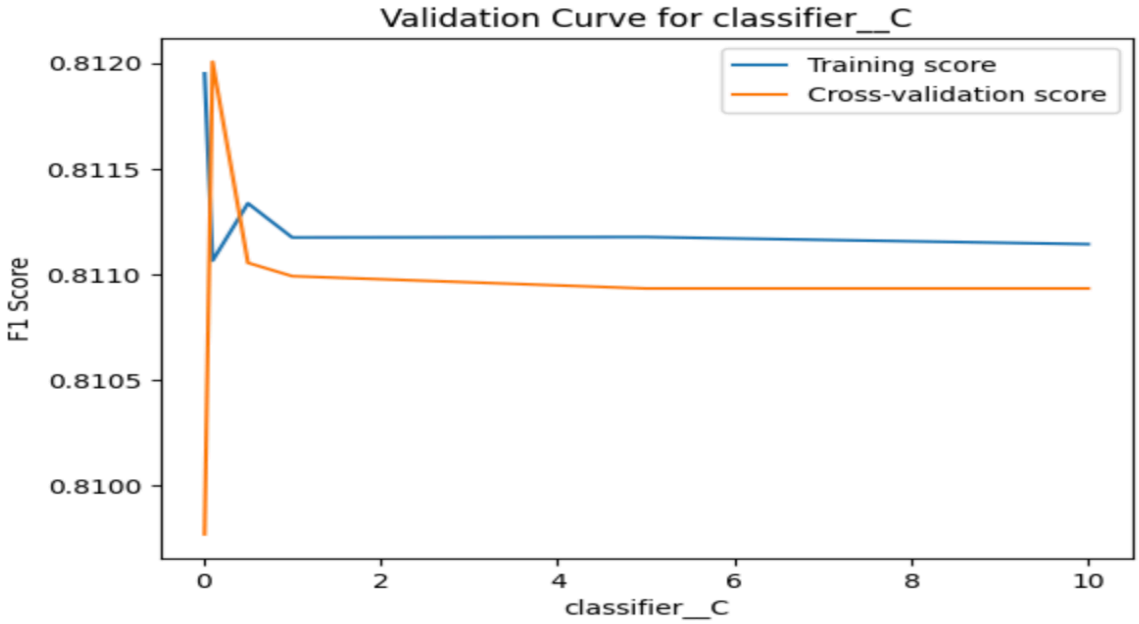
# 8.7 Logistic Regression

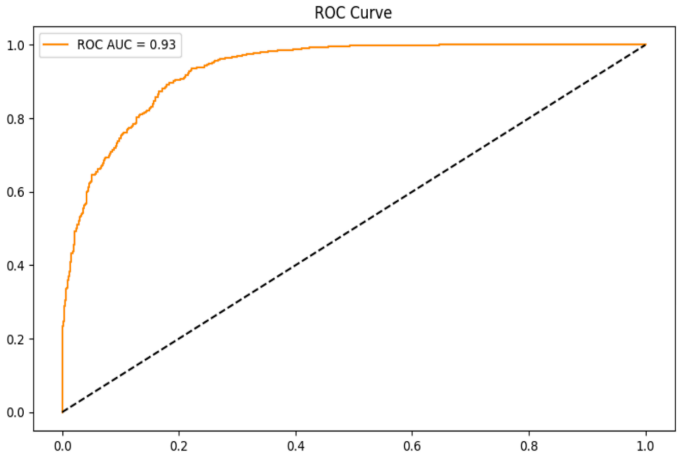
Logistic Regression was used as a baseline classifier due to its simplicity and interpretability. Trained with a higher number of iterations to ensure convergence, it achieved a validation F1 score of 0.84 and a ROC AUC of 0.86. While not as powerful as ensemble methods, it provided solid performance and valuable insights into feature importance, making it a reliable starting point for stroke prediction tasks.

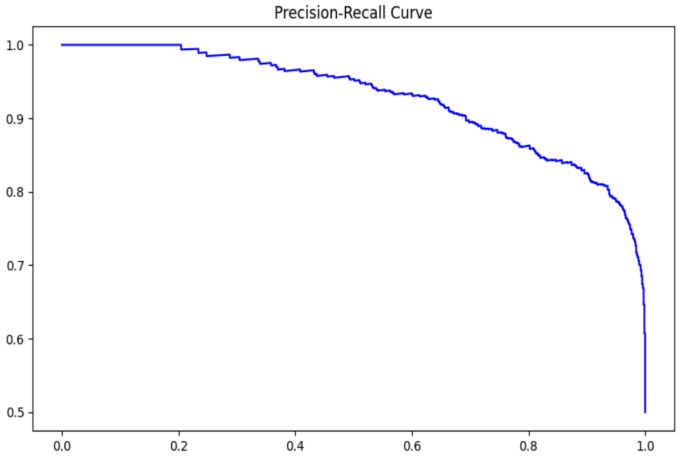


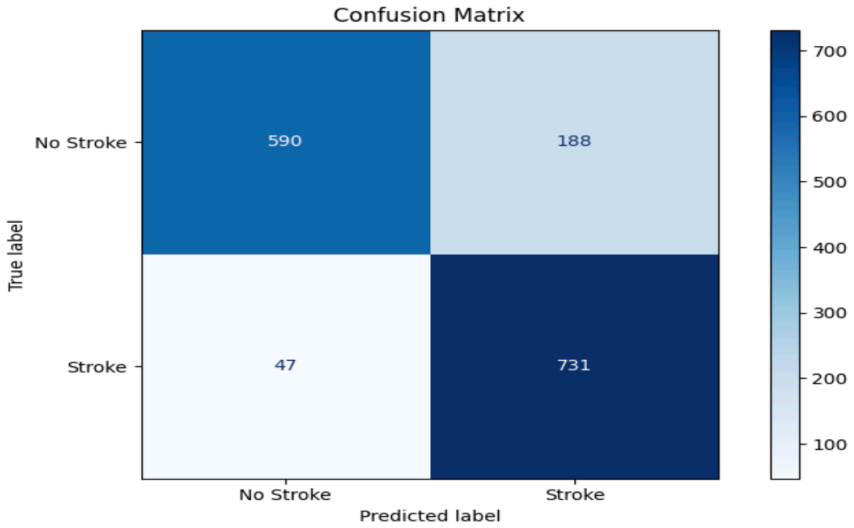



```
Best Parameters: {'classifier__C': 0.1}
Best F1 (Validation): 0.8120
Best ROC-AUC (Validation): 0.8809
```






```
=== Classification Report (Logistic Regression) ===
              precision    recall  f1-score   support

   No Stroke       0.93      0.76      0.83       778
      Stroke       0.80      0.94      0.86       778

    accuracy                           0.85      1556
   macro avg       0.86      0.85      0.85      1556
weighted avg       0.86      0.85      0.85      1556
```

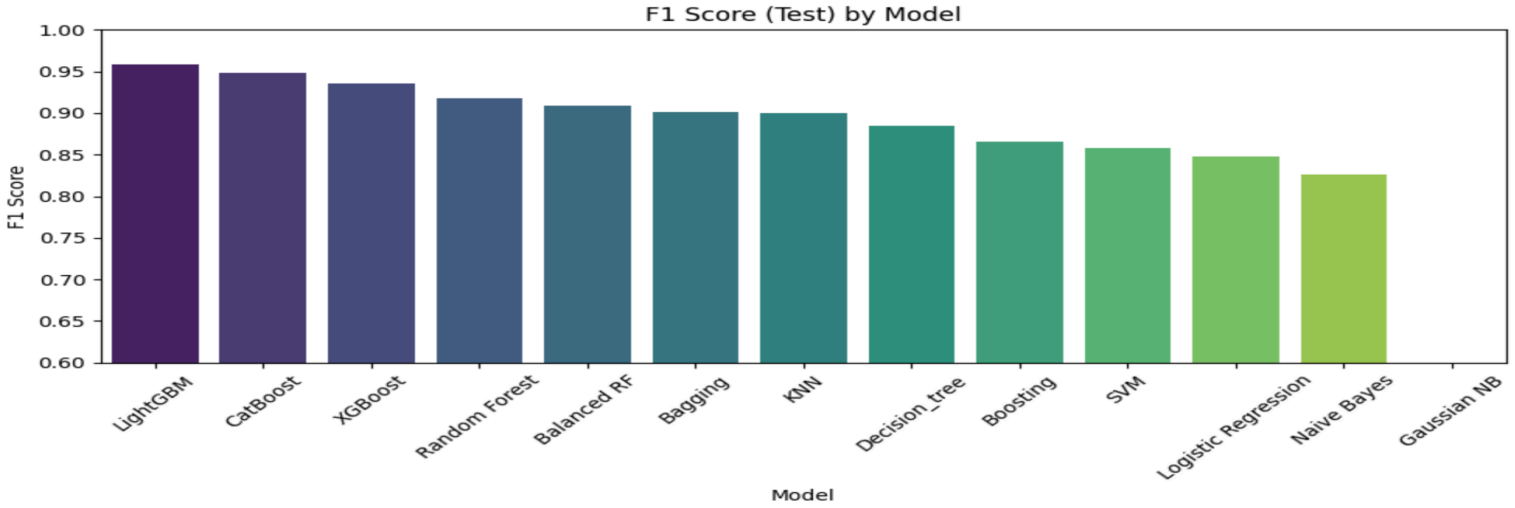
# Model Comparison

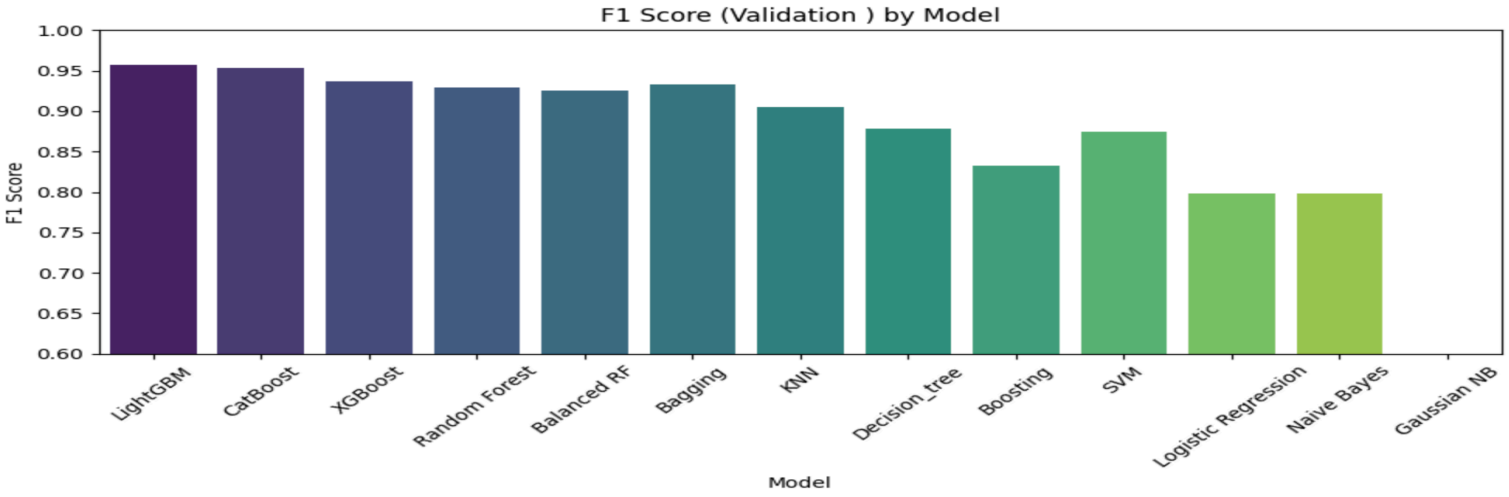
Among all models tested, LightGBM, CatBoost, and XGBoost delivered the highest overall performance with F1 scores above 0.93, ROC AUC near 0.99, and minimal F1 gaps, indicating strong generalization from validation to test data. Traditional ensemble models like Random Forest and Balanced Random Forest also performed well, especially on recall and ROC AUC metrics.

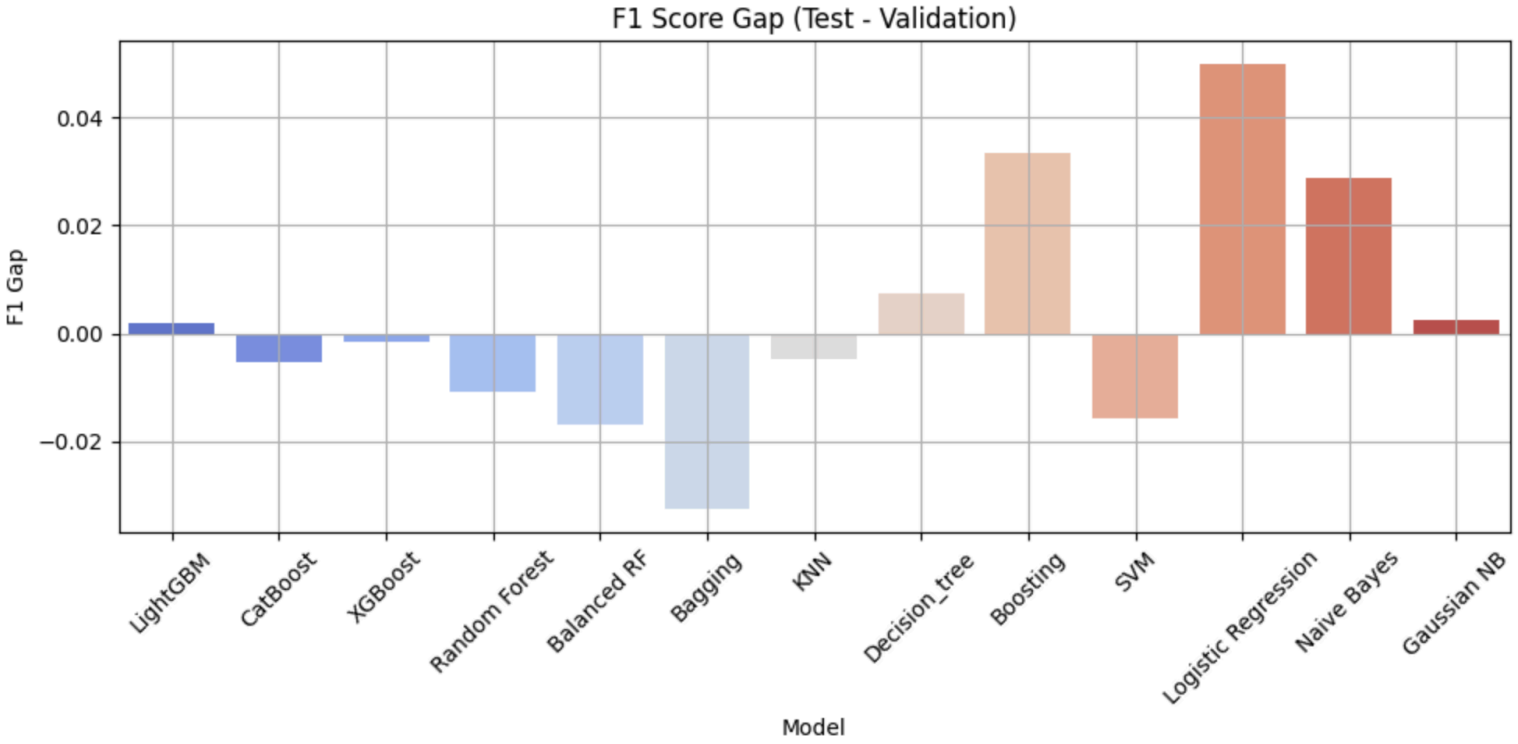
Bagging, Boosting, and Decision Tree models followed closely, maintaining strong balance across all metrics. Meanwhile, SVM and KNN showed competitive results, though with slightly lower F1 scores and higher F1 gaps. Logistic Regression and Naive Bayes performed decently but fell behind the ensemble methods in precision and recall. Gaussian NB ranked lowest across all metrics, making it unsuitable for this imbalanced classification task.

Overall, ensemble-based methods—particularly gradient boosting variants—clearly outperformed others in both accuracy and robustness.

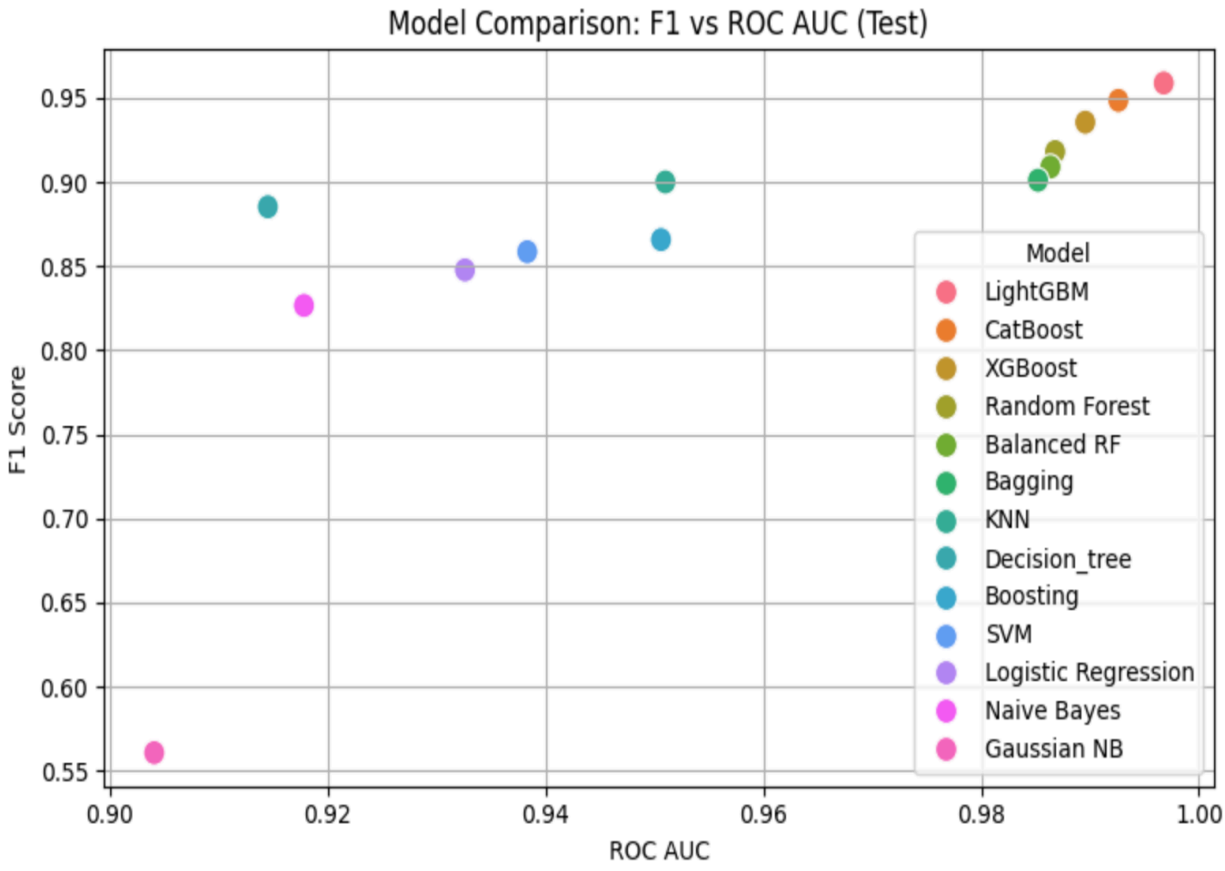
| Model | F1 (Test) | F1 (Val) | ROC AUC (Test) | ROC AUC (Val) | Accuracy (Test) | Precision | Recall | Confusion Matrix | Best Params | F1 Gap |
|---|---|---|---|---|---|---|---|---|---|---|
| LightGBM | 0.958838905 | 0.956789941 | 0.996766807 | 0.993012159 | 0.956790123 | 0.960210101 | 0.958868895 | [[767, 11], [53, 725]] | {'classifier__num_leaves': 63, 'classifier__n_estimators': 300, 'classifier__max_depth': -1, 'classifier__learning_rate': 0.1} | 0.002048965 |
| CatBoost | 0.948514583 | 0.953675461 | 0.992603472 | 0.992183399 | 0.953703704 | 0.951093169 | 0.948586118 | [[767, 11], [69, 709]] | {'classifier__learning_rate': 0.1, 'classifier__iterations': 500, 'classifier__depth': 10} | -0.005160878 |
| XGBoost | 0.935562314 | 0.937210632 | 0.989565229 | 0.987499788 | 0.937242798 | 0.940389124 | 0.935732648 | [[768, 10], [90, 688]] | OrderedDict([('classifier__learning_rate', 0.3), ('classifier__max_depth', 10), ('classifier__n_estimators', 200)]) | -0.001648318 |
| Random Forest | 0.918048716 | 0.928903683 | 0.986797107 | 0.983707916 | 0.929012346 | 0.925266548 | 0.918380463 | [[764, 14], [113, 665]] | {'classifier__max_depth': 20, 'classifier__n_estimators': 200} | -0.010854967 |
| Balanced RF | 0.908936165 | 0.925806482 | 0.986360122 | 0.985097123 | 0.925925926 | 0.917579621 | 0.909383033 | [[762, 16], [125, 653]] | {'classifier__n_estimators': 300, 'classifier__max_samples': 0.5, 'classifier__max_features': 0.5, 'classifier__max_depth': 20} | -0.016870317 |
| Bagging | 0.901032255 | 0.933592615 | 0.98522677 | 0.986668381 | 0.933641975 | 0.912314575 | 0.901670951 | [[764, 14], [139, 639]] | {'classifier__n_estimators': 300} | -0.032560361 |
| KNN | 0.900012893 | 0.904720402 | 0.951018861 | 0.946001414 | 0.905349794 | 0.906445857 | 0.900385604 | [[748, 30], [125, 653]] | OrderedDict([('classifier__metric', 'manhattan'), ('classifier__n_neighbors', 3), ('classifier__weights', 'distance')]) | -0.004707508 |
| Decision_tree | 0.885228543 | 0.877715957 | 0.91452235 | 0.905293697 | 0.87808642 | 0.886974749 | 0.885347044 | [[830, 143], [80, 892]] | {'classifier__min_samples_split': 2, 'classifier__max_depth': 19, 'classifier__criterion': 'entropy'} | 0.007512587 |
| Boosting | 0.865700651 | 0.832418727 | 0.95061657 | 0.919296368 | 0.83281893 | 0.873252666 | 0.866323907 | [[621, 157], [51, 727]] | {'classifier__n_estimators': 200} | 0.033281924 |
| SVM | 0.858536102 | 0.874338932 | 0.938334732 | 0.942095759 | 0.875 | 0.859381312 | 0.858611825 | [[650, 128], [92, 686]] | {'classifier__kernel': 'rbf', 'classifier__gamma': 'scale', 'classifier__C': 10} | -0.01580283 |
| Logistic Regression | 0.847721296 | 0.797729618 | 0.932621711 | 0.880826729 | 0.798353909 | 0.860823228 | 0.848971722 | [[590, 188], [47, 731]] | {'classifier__C': 0.1} | 0.049991678 |
| Naive Bayes | 0.826570684 | 0.797686535 | 0.91784022 | 0.864293849 | 0.799382716 | 0.848996299 | 0.829048843 | [[552, 226], [40, 738]] | {'classifier__alpha': 0.1} | 0.028884149 |
| Gaussian NB | 0.560640258 | 0.558050616 | 0.904090477 | 0.853142094 | 0.62037037 | 0.777273409 | 0.622107969 | [[193, 585], [3, 775]] | {'classifier__var_smoothing': 1e-07} | 0.002589641 |




F1 Score (Validation ) by Model
F1 Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
LightGBM
CatBoost
XGBoost
Random Forest
Balanced RF
Bagging
KNN
Decision_tree
Boosting
SVM
Logistic Regression
Naive Bayes
Gaussian NB
Model


F1 Score Gap (Test - Validation)
F1 Gap
0.04
0.02
0.00
−0.02
LightGBM
CatBoost
XGBoost
Random Forest
Balanced RF
Bagging
KNN
Decision_tree
Boosting
SVM
Logistic Regression
Naive Bayes
Gaussian NB
Model

This scatter plot compares models based on their F1 Score and ROC AUC on the test set. The top-right cluster, including LightGBM, CatBoost, and XGBoost, demonstrates superior performance with high accuracy and class discrimination. In contrast, models like Gaussian NB and Naive Bayes lag behind with lower scores on both metrics. Overall, ensemble methods clearly dominate in both precision and robustness for stroke prediction.



# 11.Conclusion and Discussion

The analysis reveals that ensemble models, particularly LightGBM, CatBoost, and XGBoost, significantly outperform traditional classifiers in stroke prediction tasks. These models demonstrated strong accuracy, F1 scores, and ROC AUC values, highlighting their reliability in handling imbalanced medical datasets. Performance gains were achieved through proper preprocessing, including data balancing with SMOTE and feature normalization. Simpler models such as Logistic Regression and Naive Bayes showed weaker results, suggesting they are less suited for complex medical classification. Decision Tree and Boosting approaches also yielded solid performance, making them viable alternatives. Overall, the findings suggest that ensemble methods are well-suited for early stroke risk detection and can be valuable tools in healthcare decision-making.

## 12.Bibliography & References

| | |
|---|---|
| ACC | Accuracy |
| AdaBoost | Adaptive Boosting |
| ANN | Artificial neural network |
| AUC | Area under curve |
| CART | Classification and regression tree |
| DT | Decision tree |
| EDA | Exploratory data analysis |
| FNR | False negative rate |
| GD | Gradient descent |
| GB | Gradient Boosting |
| KNN | k-Nearest Neighbor |
| MLP | Multilayer perceptron |
| NB | Naïve bayes |
| NN | Neural network |
| PCA | Principal component analysis |
| RELU | Rectified linear unit |
| RF | Random Forest |
| ROC | Received Operating Characteristic |
| SVC | Support vector classifier |
| VIF | Variance inflation factor |
| XGB | eXtreme Gradient Boosting |

- **Müller, A. C., & Guido, S. (2016).** *Introduction to Machine Learning with Python: A Guide for Data Scientists.* O'Reilly Media.
  *(Foundational guide for implementing ML models with Python libraries such as scikit-learn.)*
- **James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021).** *An Introduction to Statistical Learning with Applications in R (2nd ed.).* Springer.
  *(Excellent for statistical understanding of classification models and performance metrics.)*
- https://machinelearningmastery.com/handle-imbalanced-classification-datasets-in-python/
  *(Covers SMOTE, under/over-sampling, and balanced evaluation methods.)*
- https://machinelearningmastery.com/super-learner-ensemble-in-python/
  *(Explains stacking and ensemble learning strategies.)*
- https://towardsdatascience.com/a-complete-guide-to-xgboost-in-python-8e28f6e5f2f2
  *(Detailed guide to implementing XGBoost for classification tasks.)*
- https://towardsdatascience.com/feature-importance-and-feature-selection-in-machine-learning-d7b292dfb8d
  *(Practical explanation of how to interpret feature importance in tree-based and linear models.)*
- https://medium.com/@nhan.tran/the-chi-square-statistic-p-1-37a8eb2f27bb
  *(Good reference for using chi-square in evaluating categorical feature relationships.)*