USC University of Southern California

Marshall School of Business

Final Project Report On

"Stroke Prediction Modeling"

Submitted in partial fulfillment for the subject

DSO-568 : Healthcare Analytics

Academic Term: Fall 2024

Submission Date: November 25, 2024

Submitted by: Team 5

Campbell Duncan, Jnana Kundur Prakash, Lindsay Yan,

Lucian Deng, Marliese Sanabria, Zoha Akhtar

Course Instructor:

Cosimo Arnesano, Assistant Professor

Data Sciences and Operations Department

USC Marshall School of Business

Executive Summary:

Stroke is a leading cause of death and long-term disability worldwide, affecting millions annually. Its unpredictability and rapid onset make it a critical area for preventive healthcare interventions. With timely prediction, healthcare systems can shift from reactive treatment to proactive care, potentially saving lives, reducing disability, and lowering healthcare costs.

In this project, we utilized the Kaggle Stroke Prediction Dataset, which contains 5,110 records and 12 features, to develop machine learning models aimed at predicting stroke risk. The dataset includes demographic, clinical, and lifestyle attributes, such as age, BMI, smoking status, and average glucose level, alongside stroke labels. The dataset reflects a significant class imbalance, with stroke cases constituting only 4.87% (249 cases) of the total records. Key challenges addressed include handling missing data, dealing with this class imbalance, and identifying significant predictive factors.

We began with data preprocessing, exploratory analysis, and feature engineering to prepare the dataset for machine learning. A series of models were evaluated, starting with Logistic Regression as a baseline. Advanced models like Decision Trees, Random Forests, and XGBoost were implemented to enhance prediction performance. To address the significant class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was applied, creating a balanced training dataset with stroke and non-stroke cases.

The ensemble model combining Random Forest and XGBoost achieved the highest performance, with an accuracy of 95.47% and a ROC-AUC score of 99.04%. This result underscores the ensemble's ability to balance precision and recall while minimizing false positives and negatives. Age & average glucose level emerged as the most significant predictors of stroke risk.

This project highlights the potential of machine learning in transforming stroke prediction and preventive healthcare. Ethical considerations, such as data privacy and bias mitigation, were prioritized throughout the analysis. The outcomes not only enable healthcare providers to focus preventive measures on high-risk individuals but also align with public health goals of reducing the global burden of stroke. Future extensions could include real-time data integration, increasing the dataset size with additional records, and incorporating personalized treatment recommendations to enhance prediction accuracy further.

1. Problem Definition:

Healthcare Process:

The healthcare process targeted in this project is the early identification of patients who are at high risk for stroke. Stroke is one of the leading causes of death and long-term disability globally, creating a significant burden on individuals, families, and healthcare systems. Strokes occur when the blood supply to the brain is interrupted, often due to blocked or ruptured blood vessels, and they can lead to severe neurological impairments or death. Many stroke cases are preventable if high-risk patients are identified early and provided with appropriate interventions. This process is critical because timely identification allows healthcare providers to implement preventive strategies, such as lifestyle changes, medication, and regular monitoring, which can significantly reduce the risk of stroke and improve patient outcomes.

This process directly impacts multiple stakeholders:

- **Patients**: Early identification empowers patients to take preventive measures, improving their quality of life and reducing the risk of severe disability or mortality.
- **Healthcare Providers**: It enables clinicians to allocate resources more efficiently, focusing on high-risk individuals who require immediate attention.
- **Healthcare Systems**: Proactive stroke prevention can reduce the financial burden associated with acute stroke treatment and long-term care for stroke survivors.

Significance:

Predictive modeling in this context is transformative because it shifts the focus of healthcare from reactive treatment to proactive prevention. Many healthcare systems are overwhelmed by the cost and resource demands of treating preventable conditions. Stroke prevention through predictive modeling offers several significant benefits:

1) Enhancing Patient Risk Stratification:

Predictive models allow for the identification of individuals who are at the greatest risk for stroke based on demographic, clinical, and lifestyle data. By stratifying patients according to their risk, healthcare providers can prioritize resources and interventions for those who need them the most.

2) Facilitating Preventive Interventions:

High-risk patients identified by the model can benefit from targeted interventions, such as blood pressure management, glucose monitoring, and smoking cessation programs. These measures can lower the incidence of strokes and reduce long-term complications.

3) Optimizing Resource Allocation:

Stroke prevention is far more cost-effective than treatment. Predictive models enable healthcare providers to focus their efforts on high-risk patients, reducing unnecessary interventions for low-risk individuals and ensuring that resources such as diagnostics, medications, and follow-ups are used efficiently. This not only improves outcomes but also reduces the financial burden on healthcare systems.

Objective:

The primary goal of this project is to develop a robust and accurate machine learning model that predicts stroke risk based on an individual's demographic, clinical, and lifestyle factors. The model aims to:

1) Support Early Intervention:

By predicting stroke risk early, healthcare providers can intervene before a stroke occurs. This can include medical interventions, lifestyle modifications, or regular monitoring to mitigate risk factors.

2) Enhance Healthcare Outcomes:

Reducing the number of strokes translates to improved health outcomes, decreased mortality rates, and fewer cases of long-term disability. This aligns with the broader goal of improving public health and patient well-being.

3) Provide Actionable Insights for Resource Planning:

Insights from the predictive model can guide healthcare providers and administrators in making data-driven decisions regarding resource allocation, such as focusing preventive efforts on high-risk populations, planning community health programs, and ensuring adequate staffing and infrastructure to handle potential stroke cases.

2. Data Collection and Preparation:

In [20]:
```
# Import Necessary Libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [21]:
```
# Load the dataset
df = pd.read_csv("healthcare-dataset-stroke-data.csv")
```

In [22]:
```
print("Dataset Shape:", df.shape)
```

```
Dataset Shape: (5110, 12)
```

In [23]:
```
df.head(10)
```

Out[23]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_statu
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoke
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoke
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoke
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smoke
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoke
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoke
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoke
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoke
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknow
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknow

In [24]:
```
df.tail(10)
```

Out[24]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_s
5100	68398	Male	82.0	1	0	Yes	Self-employed	Rural	71.97	28.3	never sm
5101	36901	Female	45.0	0	0	Yes	Private	Urban	97.95	24.5	Unk
5102	45010	Female	57.0	0	0	Yes	Private	Rural	77.93	21.7	never sm
5103	22127	Female	18.0	0	0	No	Private	Urban	82.85	46.9	Unk
5104	14180	Female	13.0	0	0	No	children	Rural	103.08	18.6	Unk
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never sm
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never sm
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never sm
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly sm
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unk

Dataset Overview:

The dataset used in this project contains **5,110 records** and **12 features**, categorized as follows:

- **Demographic Information:** Gender, age, marital status, and residence type.
- **Clinical Features:** Hypertension, heart disease, average glucose level, and BMI.
- **Lifestyle Factors:** Smoking status and work type.
- **Target Variable:** Stroke (1 indicates stroke, 0 indicates no stroke).

Exploratory Data Analysis (EDA):

In [25]:
```
# General dataset overview
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
```

In [26]:
```
# Summary Statistics
df.describe()
```

Out[26]:

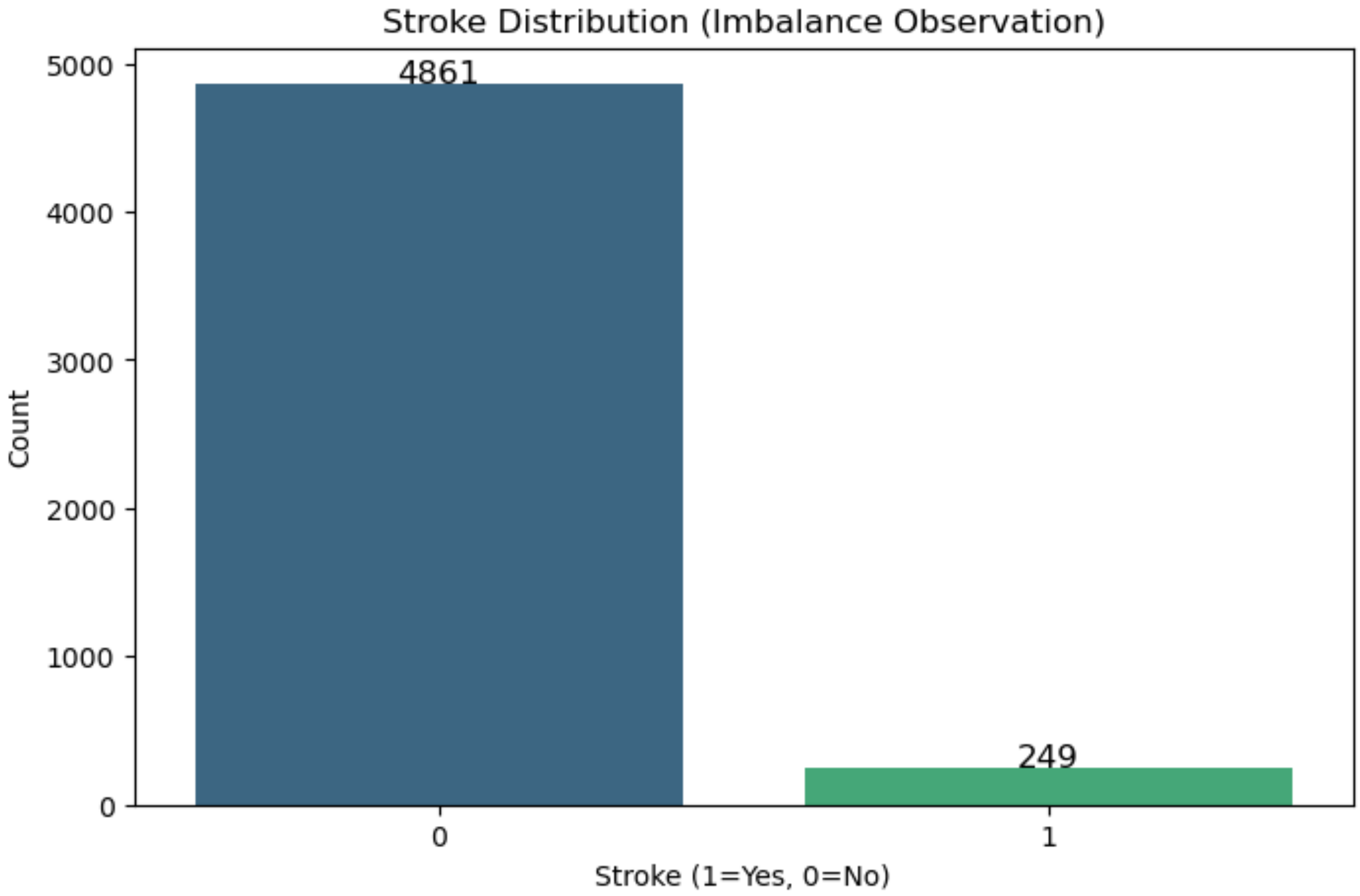
	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

In [27]:
```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 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
```

Target Variable Analysis:

In [28]:
```
import warnings
warnings.simplefilter("ignore", FutureWarning)
plt.figure(figsize=(8, 5))
ax = sns.countplot(x='stroke', data=df, palette='viridis')
for p in ax.patches:
    ax.text(p.get_x() + p.get_width() / 2., p.get_height() + 10, int(p.get_height()),
            ha='center', fontsize=12)

plt.title('Stroke Distribution (Imbalance Observation)')
plt.xlabel('Stroke (1=Yes, 0=No)')
plt.ylabel('Count')
plt.show()
```

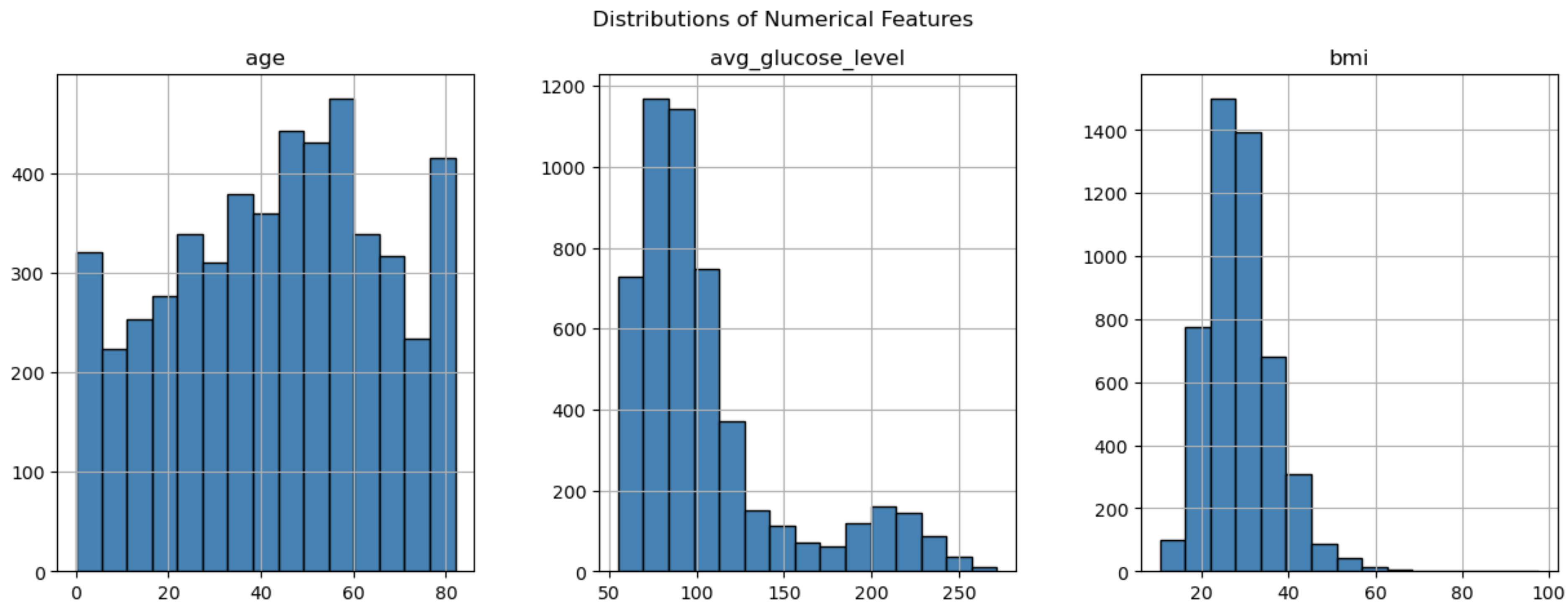


The graph shows a significant class imbalance, with most individuals labeled as "No Stroke" (Class 0) and very few as "Stroke" (Class 1), indicating the need for techniques like SMOTE to address this imbalance during modeling.

Univariate Analysis:

Numerical Features:

In [29]:
```
# Histograms for numerical features
numerical_features = ['age', 'avg_glucose_level', 'bmi']
df[numerical_features].hist(bins=15, figsize=(15, 5), layout=(1, 3), color='steelblue', edgecolor='black')
plt.suptitle('Distributions of Numerical Features')
plt.show()
```

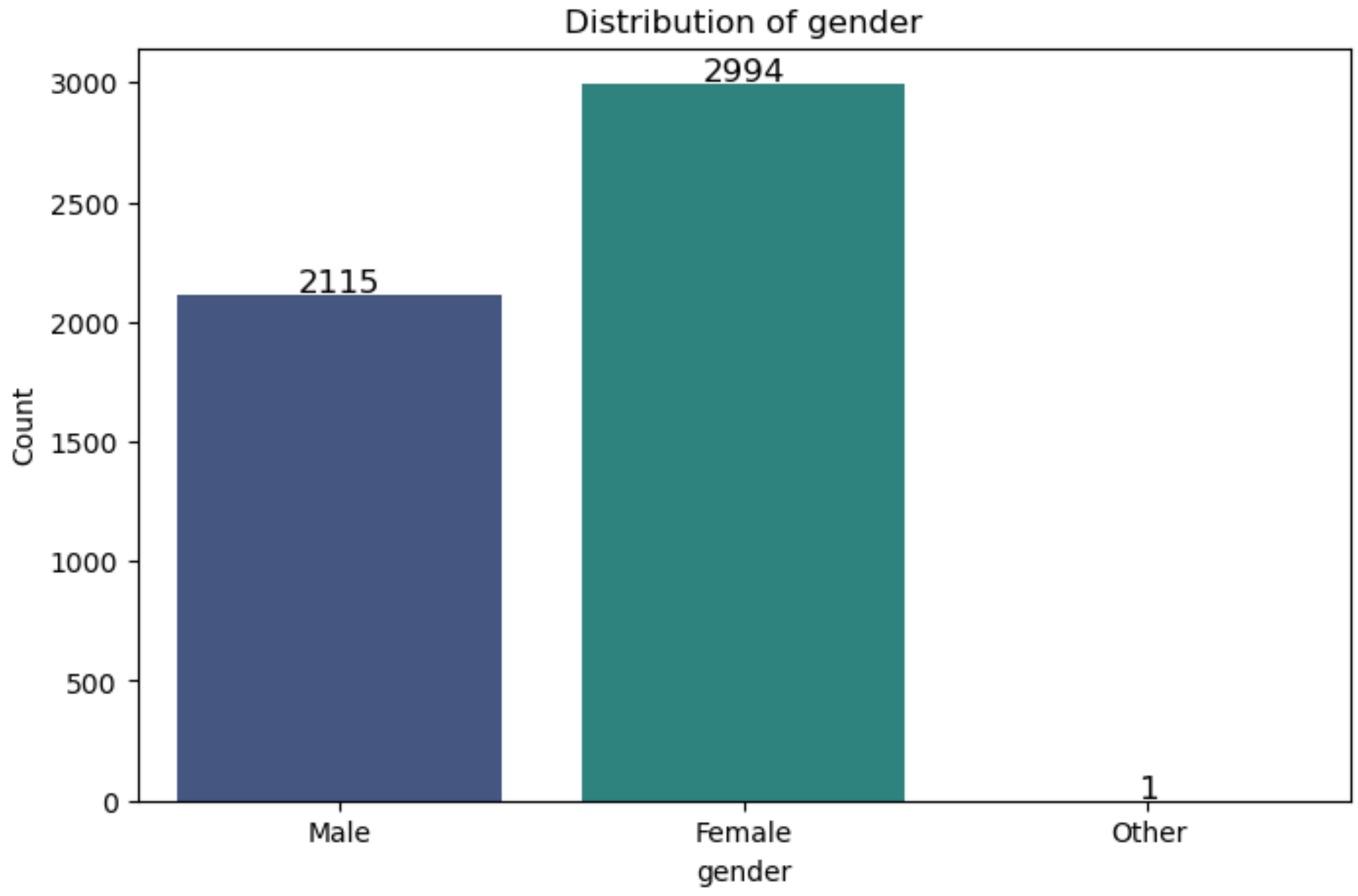


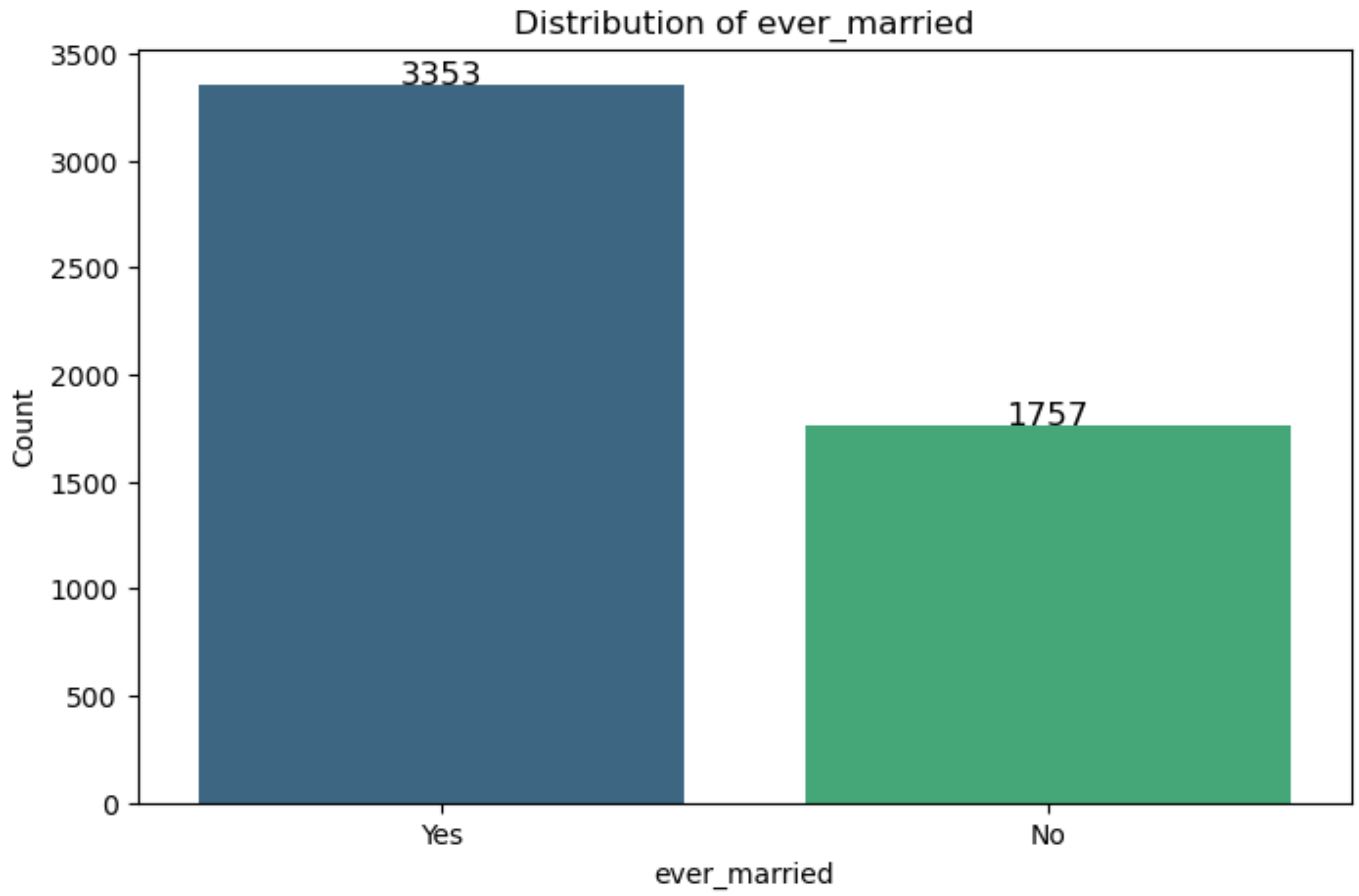
The graphs show that the age distribution is fairly uniform across all age groups, the average glucose level is right-skewed with most values concentrated below 150, and the BMI is also right-skewed, with most individuals having a BMI between 20 and 40.

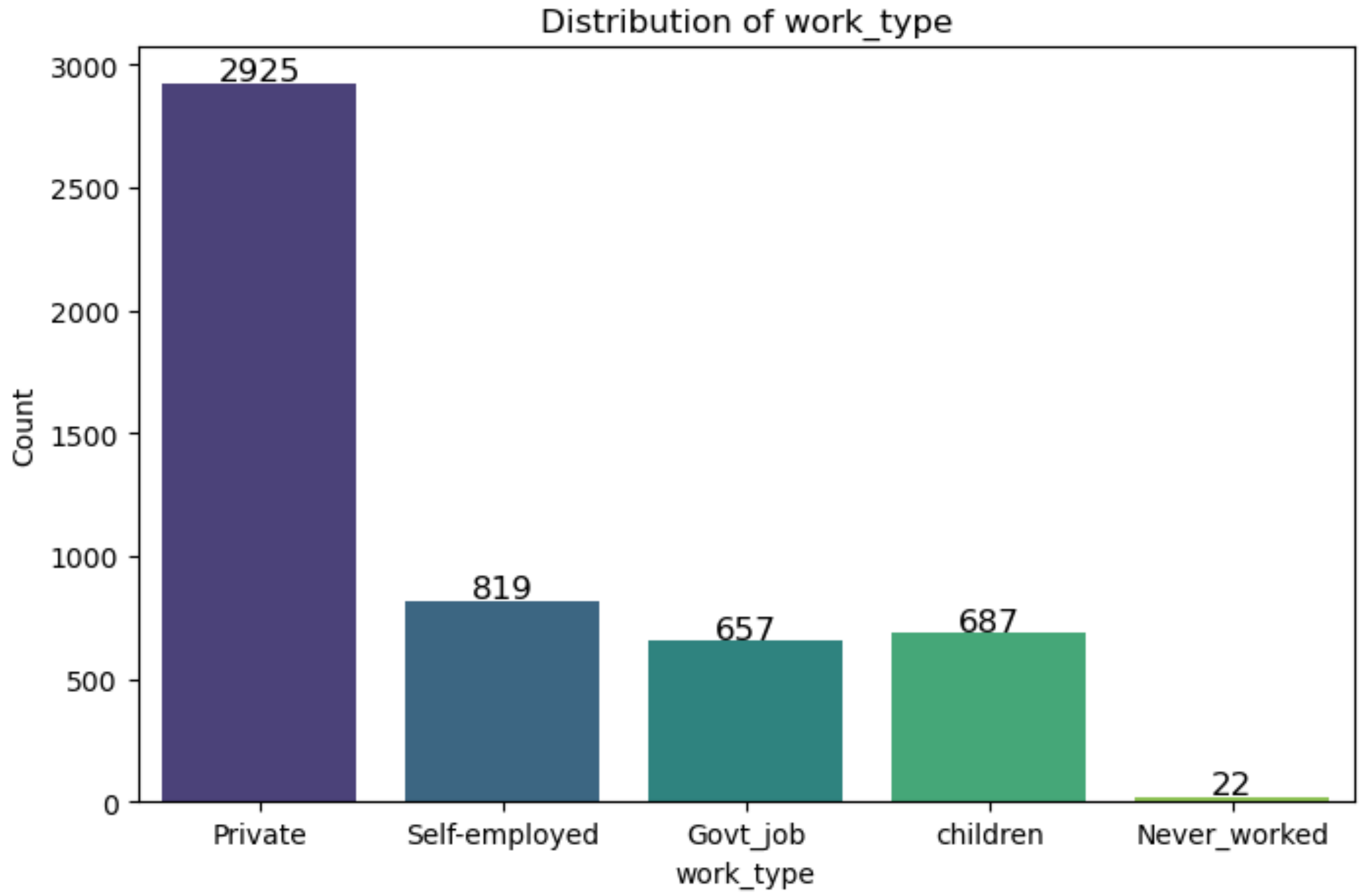
Categorical Features:

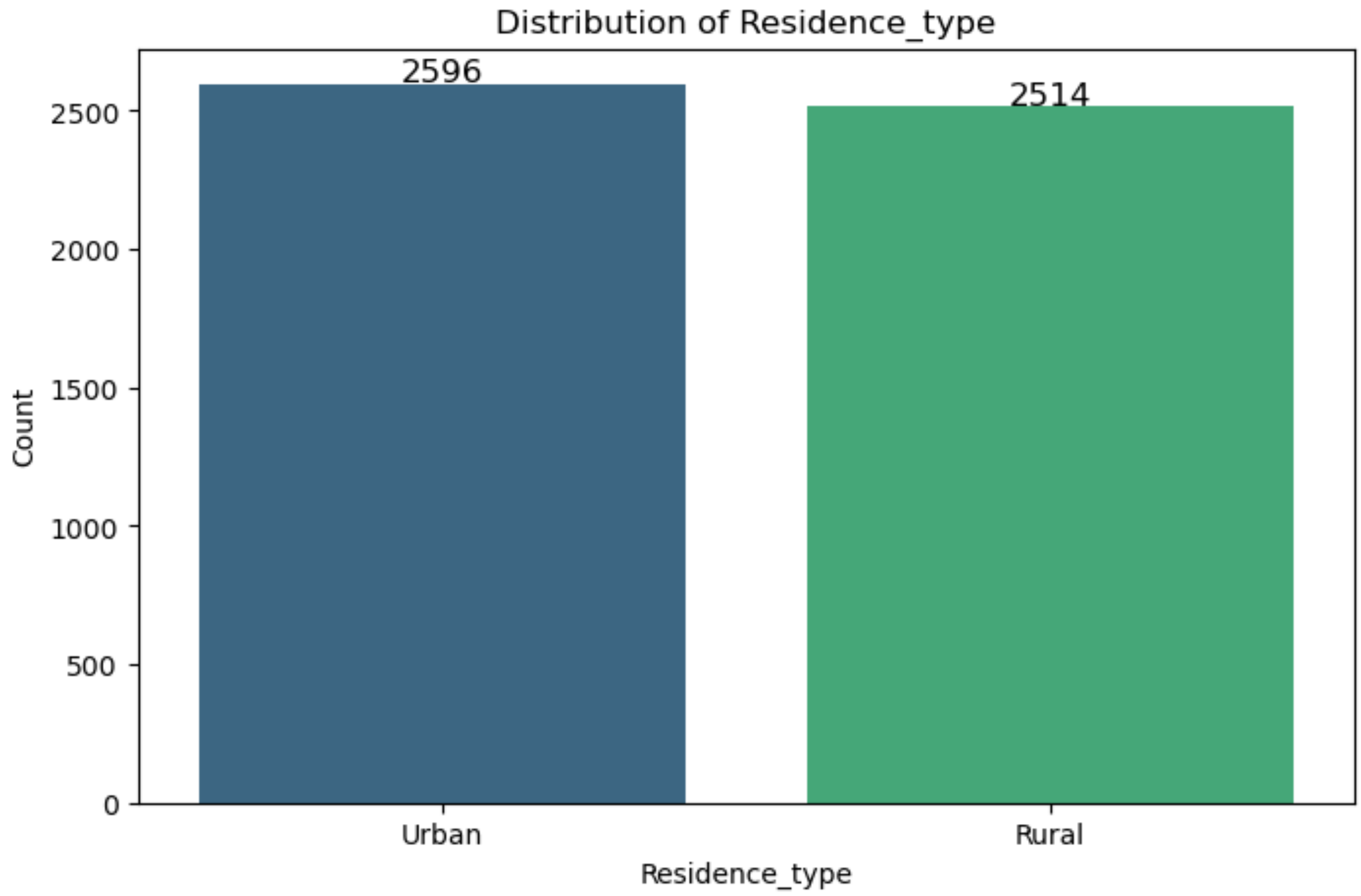
In [30]:

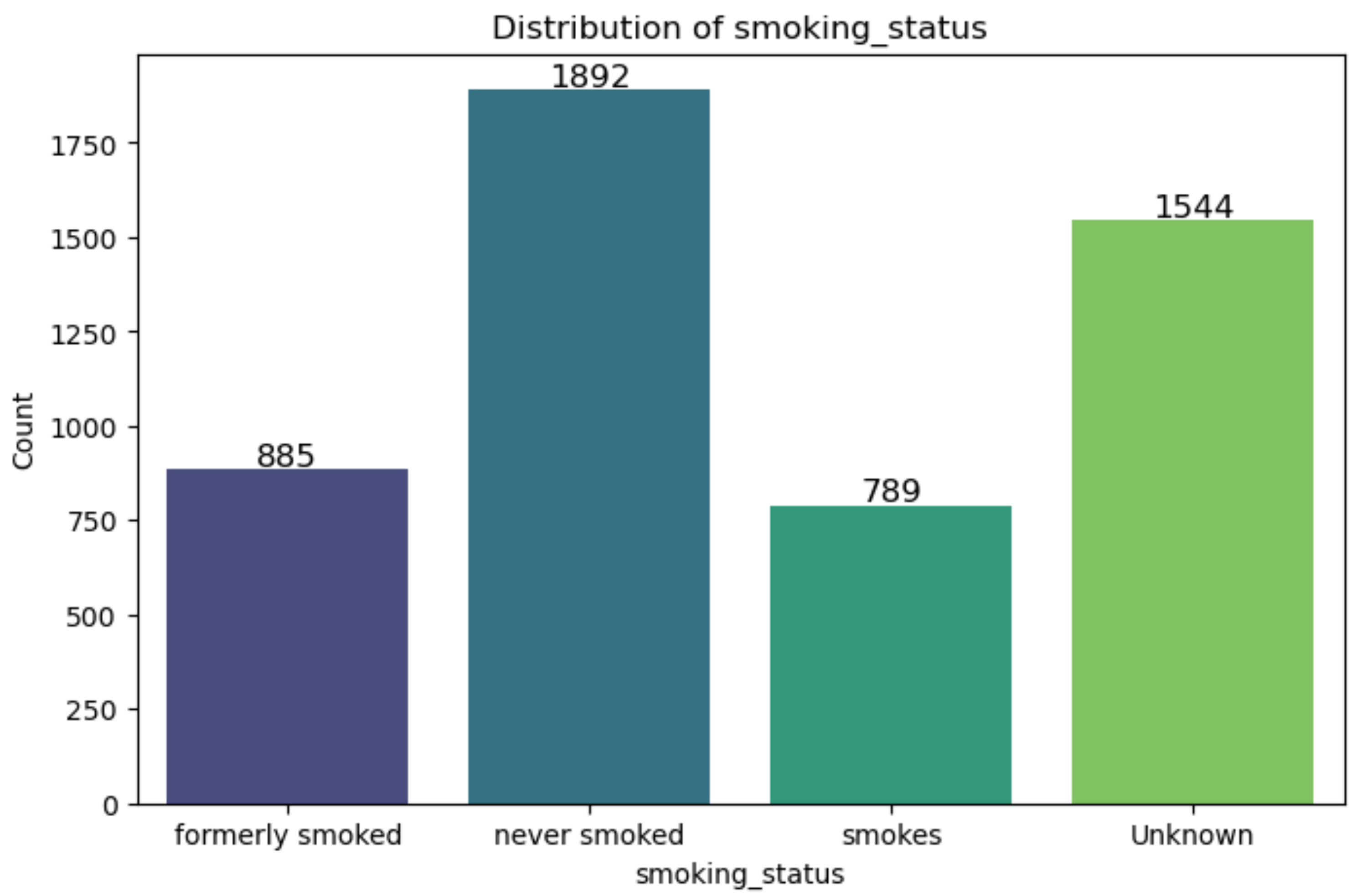
```
import matplotlib.pyplot as plt
import seaborn as sns

categorical_features = ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']

for feature in categorical_features:
    plt.figure(figsize=(8, 5))
    ax = sns.countplot(data=df, x=feature, palette='viridis')
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=0)

    for p in ax.patches:
        ax.text(p.get_x() + p.get_width() / 2., p.get_height() + 10, int(p.get_height()),
                ha='center', fontsize=12)

    plt.show()
```






Distribution of work_type
Count
work_type
2925
819
657
687
22
Private
Self-employed
Govt_job
children
Never_worked


Distribution of Residence_type
Count
Residence_type
2596
2514
Urban
Rural



Observations:

Gender Distribution: The dataset has more females than males, and there are very few instances of 'Other' gender.

Ever Married Distribution: The majority of individuals in the dataset have been married.

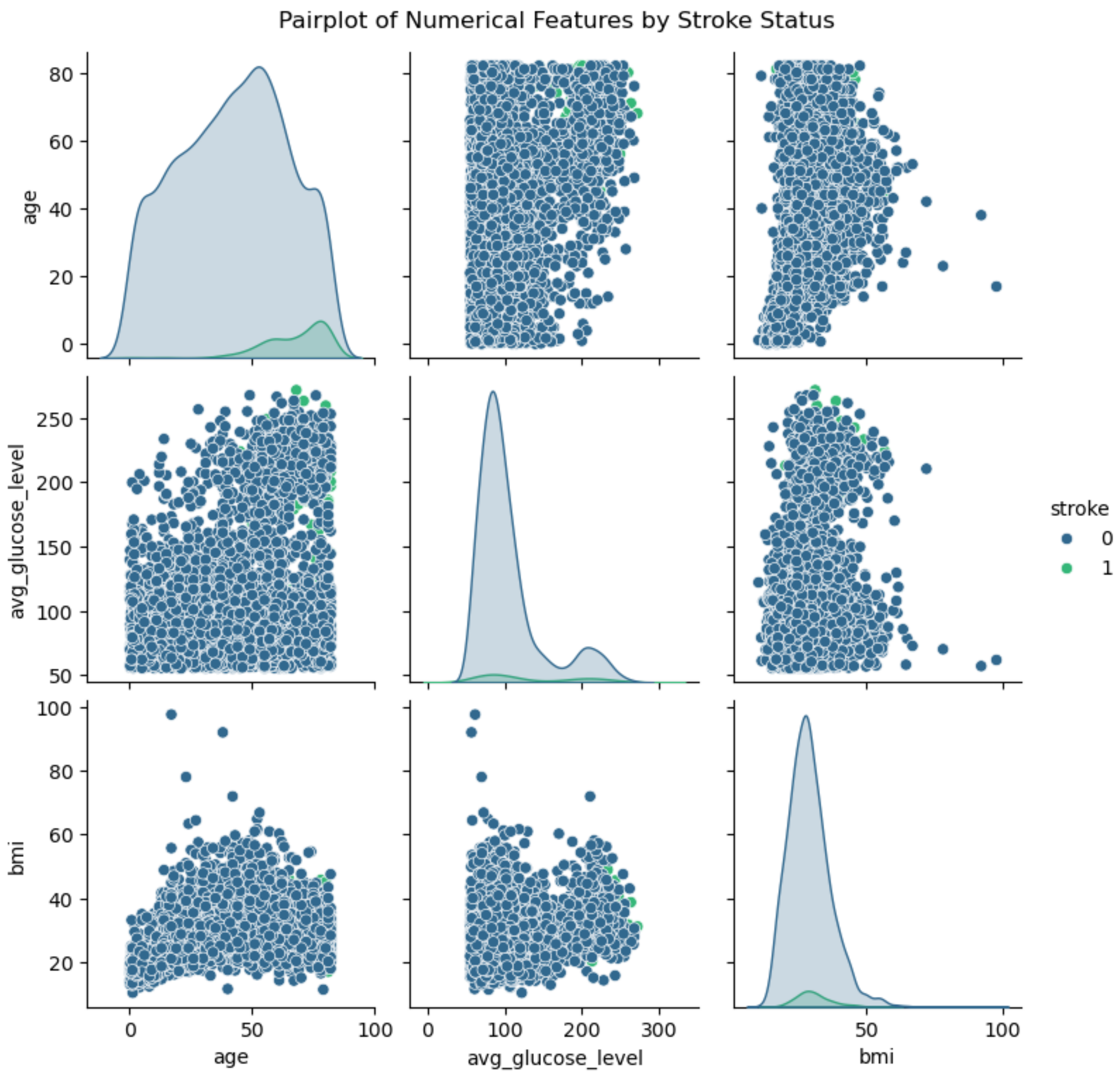
Work Type Distribution: Most individuals are employed in the private sector, followed by self-employed, government jobs, and children. Very few have never worked.

Residence Type Distribution: The distribution between urban and rural residence types is almost equal.

Smoking Status Distribution: A significant portion of individuals never smoked, followed by those with unknown smoking status. Former smokers and current smokers make up a smaller fraction of the dataset.

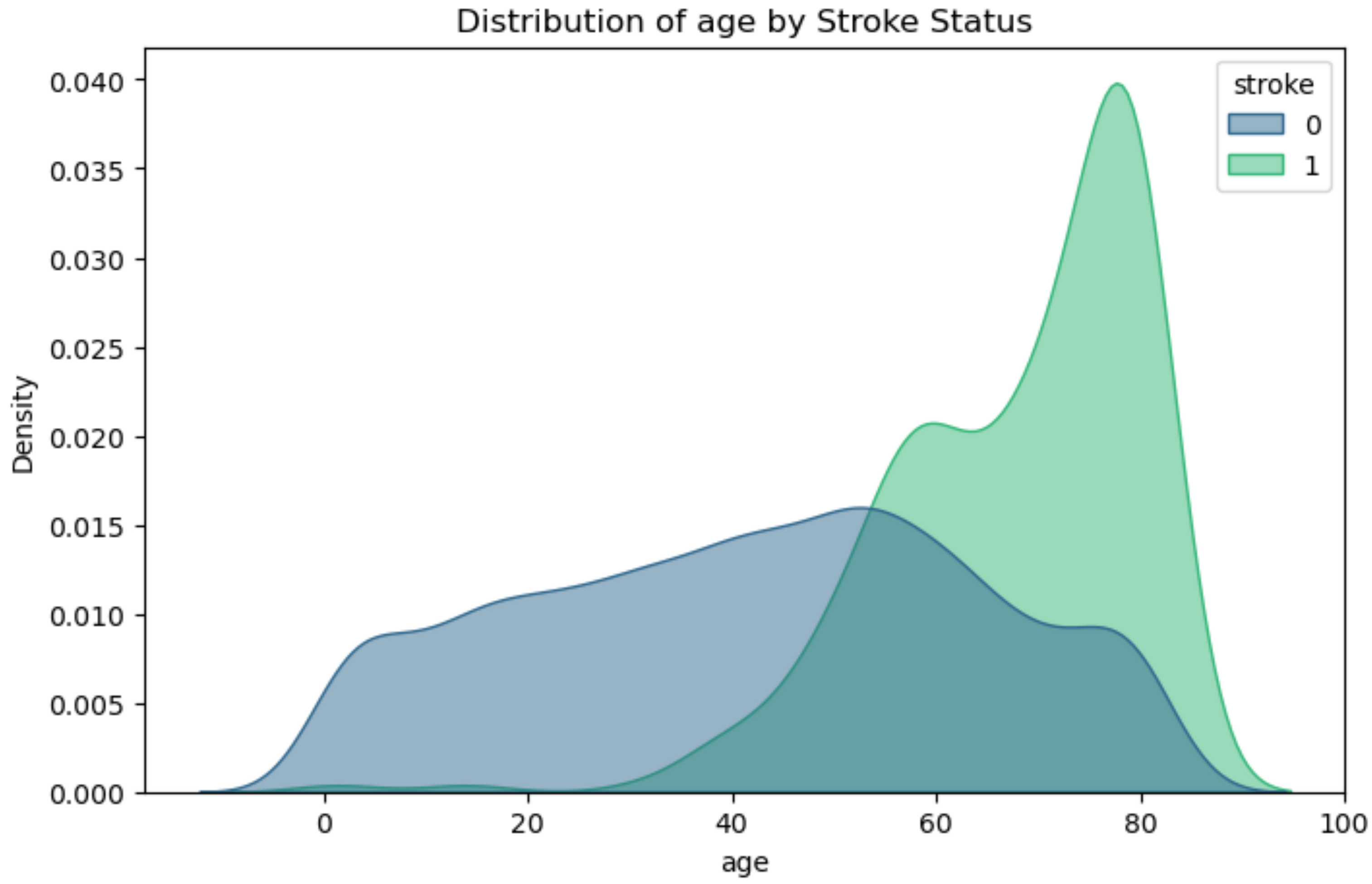
Bivariate Analysis:

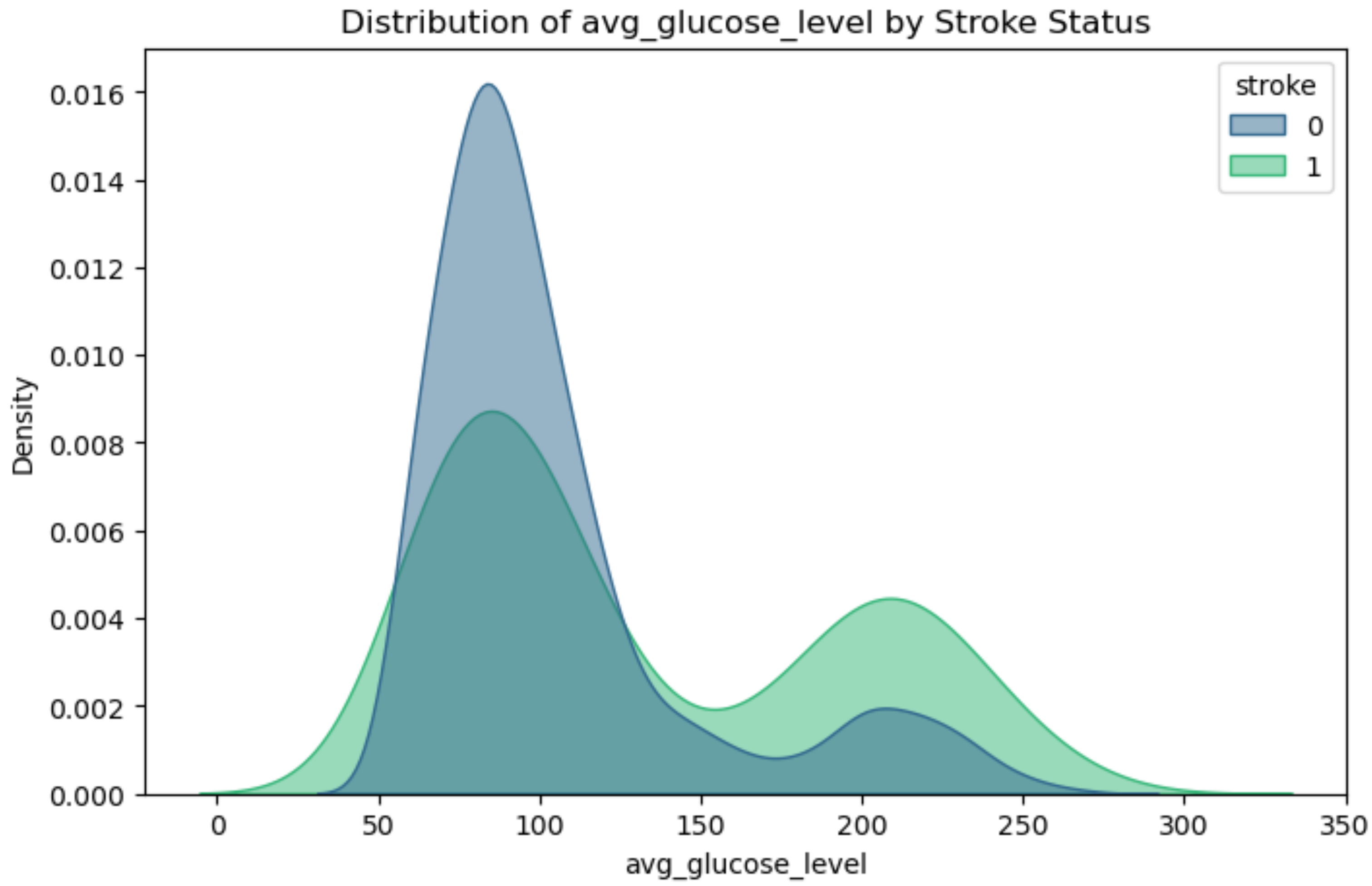
In [31]:
```
# Pairplot for relationships between numerical features
sns.pairplot(df, vars=numerical_features, hue='stroke', diag_kind='kde', palette='viridis')
plt.suptitle('Pairplot of Numerical Features by Stroke Status', y=1.02)
plt.show()
```

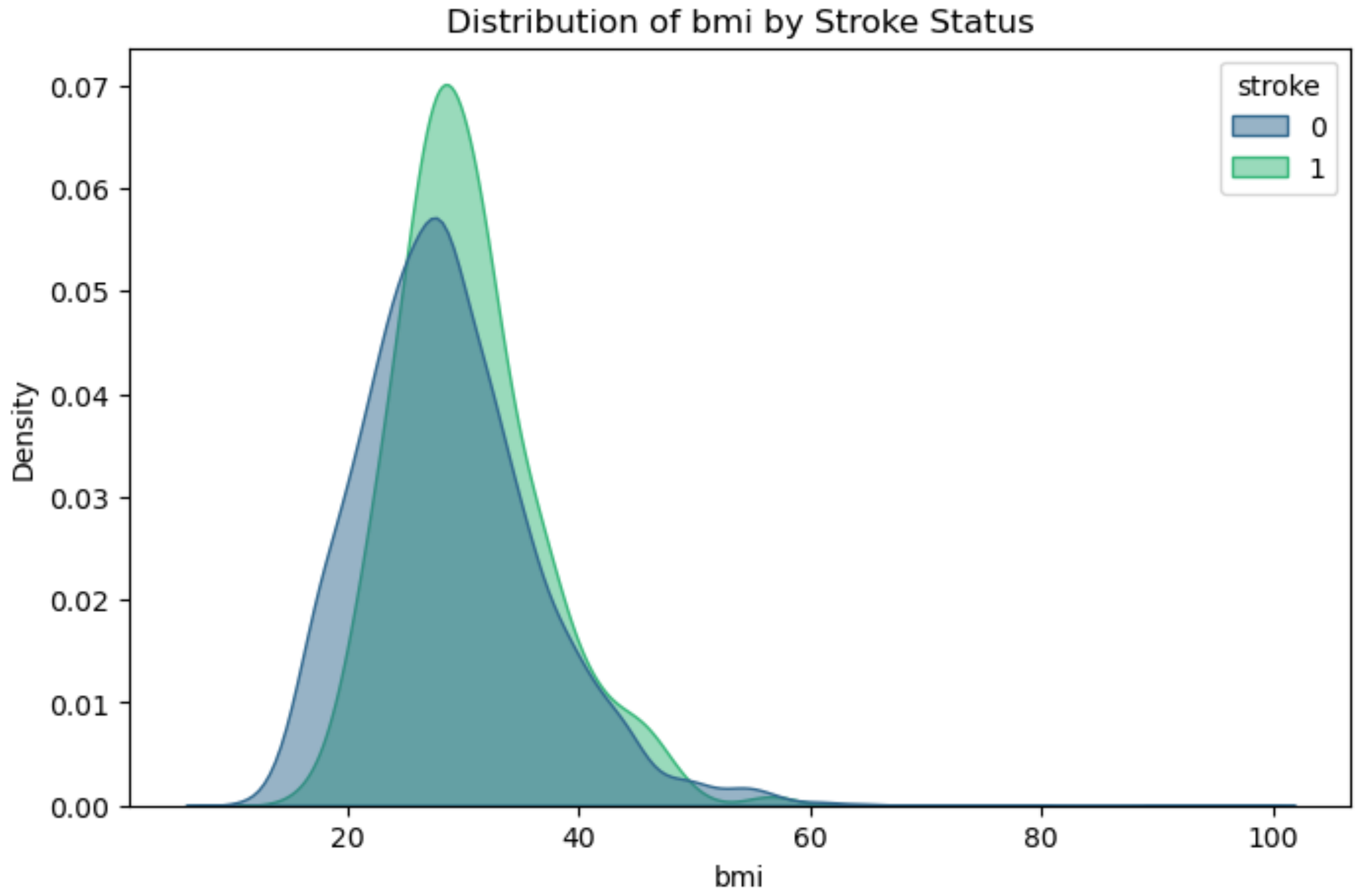


The pairplot reveals that older individuals are more likely to experience strokes, highlighting age as a significant factor. Higher average glucose levels also show a notable association with stroke occurrences. While BMI does not display a strong pattern, stroke cases are scattered across various BMI levels. Overall, the relationships between numerical features indicate that age and average glucose levels are more distinct in separating stroke cases from non-stroke cases compared to BMI.

In [32]:
```
# KDE plots for numerical features split by stroke status
for feature in numerical_features:
    plt.figure(figsize=(8, 5))
    sns.kdeplot(data=df, x=feature, hue='stroke', fill=True, common_norm=False, palette='viridis', alpha=0.5)
    plt.title(f'Distribution of {feature} by Stroke Status')
    plt.xlabel(feature)
    plt.ylabel('Density')
    plt.show()
```

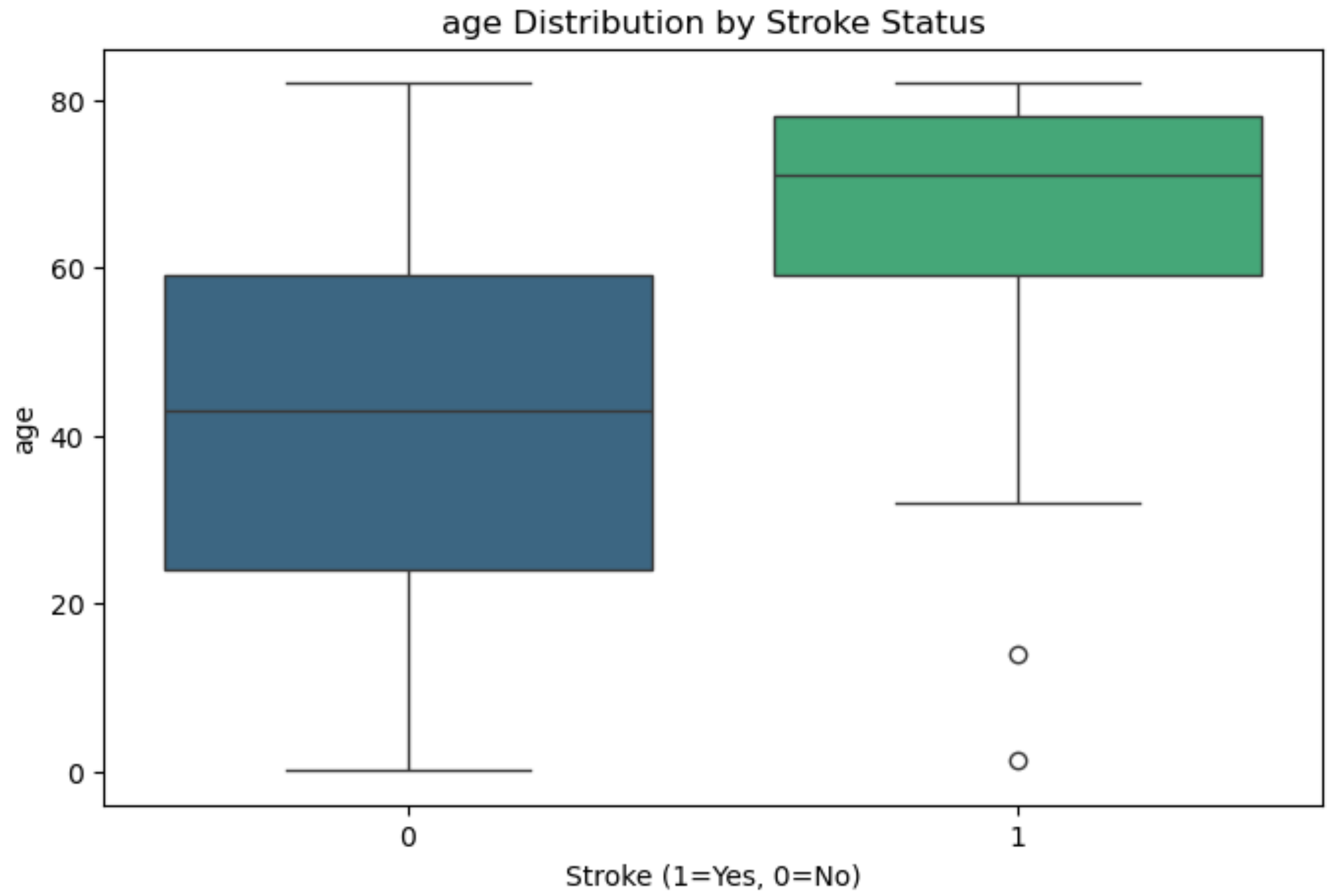

Distribution of age by Stroke Status
Density
age
stroke
0
1
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
0
20
40
60
80
100

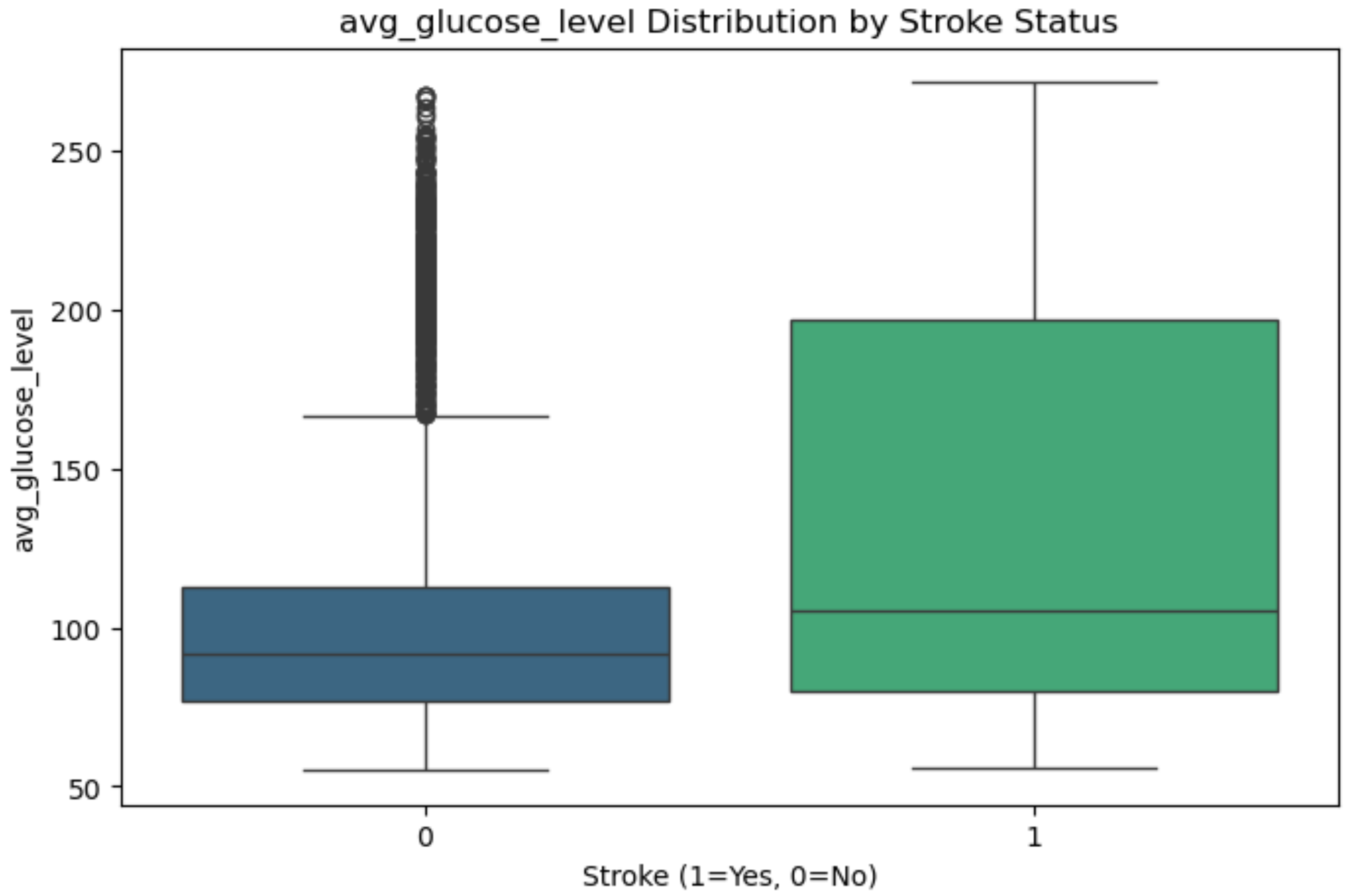

Distribution of avg_glucose_level by Stroke Status
Density
avg_glucose_level
stroke
0
1
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0
50
100
150
200
250
300
350

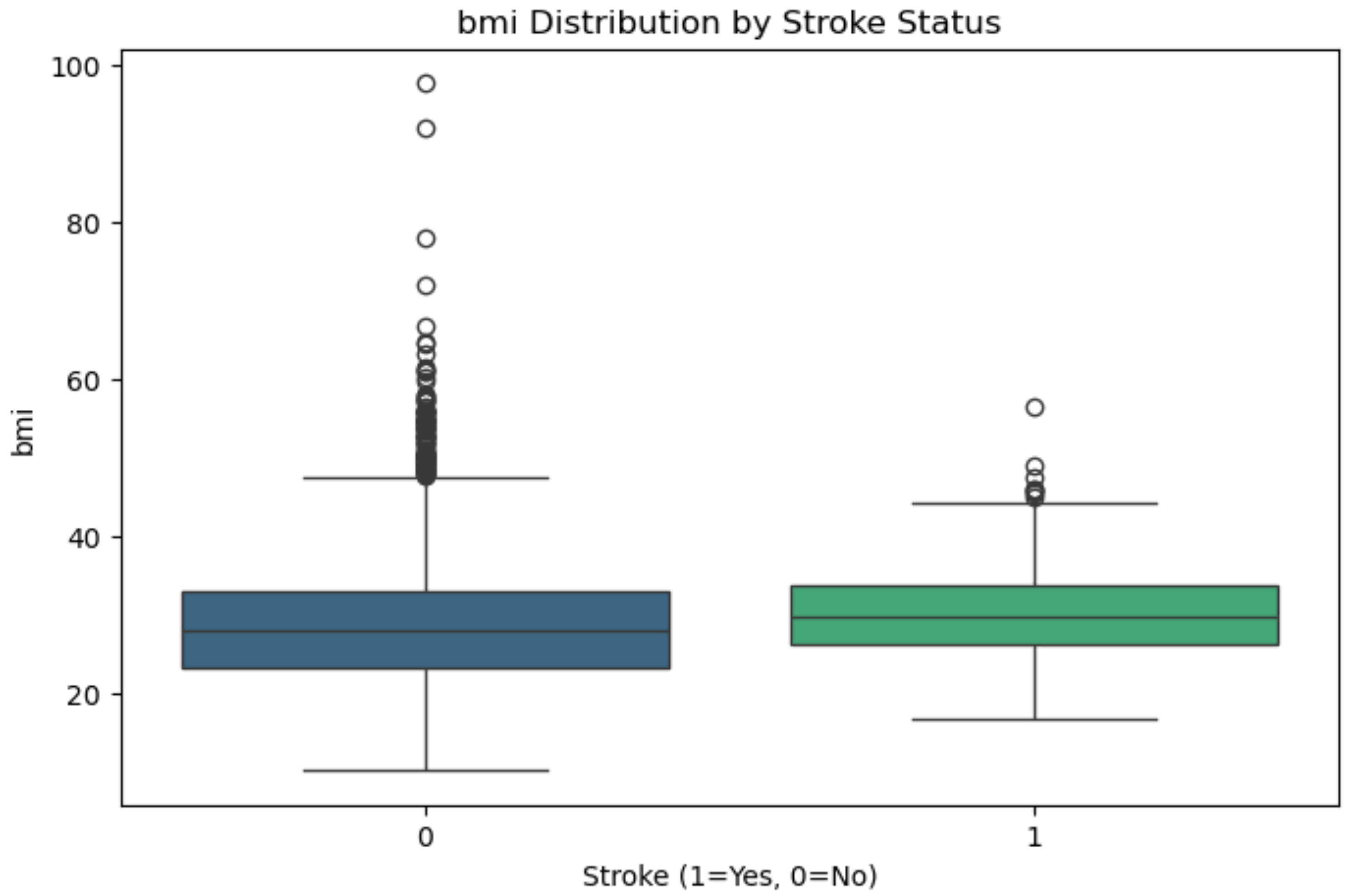


The KDE plots reveal distinct patterns in the distributions of numerical features based on stroke status. Older individuals, particularly those aged 70-80, show a higher likelihood of having strokes, as observed in the age distribution. Average glucose levels are notably higher among stroke patients, with a density peak above 150 mg/dL, whereas individuals without strokes generally exhibit lower glucose levels, peaking around 100 mg/dL. The BMI distribution is relatively similar for both groups, with the peak density around 25-30, but stroke patients tend to have slightly higher BMIs on average. These patterns highlight the importance of age, glucose levels, and BMI in understanding stroke risks.

In [33]:
```
# Boxplots for numerical features by stroke status
for feature in numerical_features:
    plt.figure(figsize=(8, 5))
    sns.boxplot(data=df, x='stroke', y=feature, palette='viridis')
    plt.title(f'{feature} Distribution by Stroke Status')
    plt.xlabel('Stroke (1=Yes, 0=No)')
    plt.ylabel(feature)
    plt.show()
```



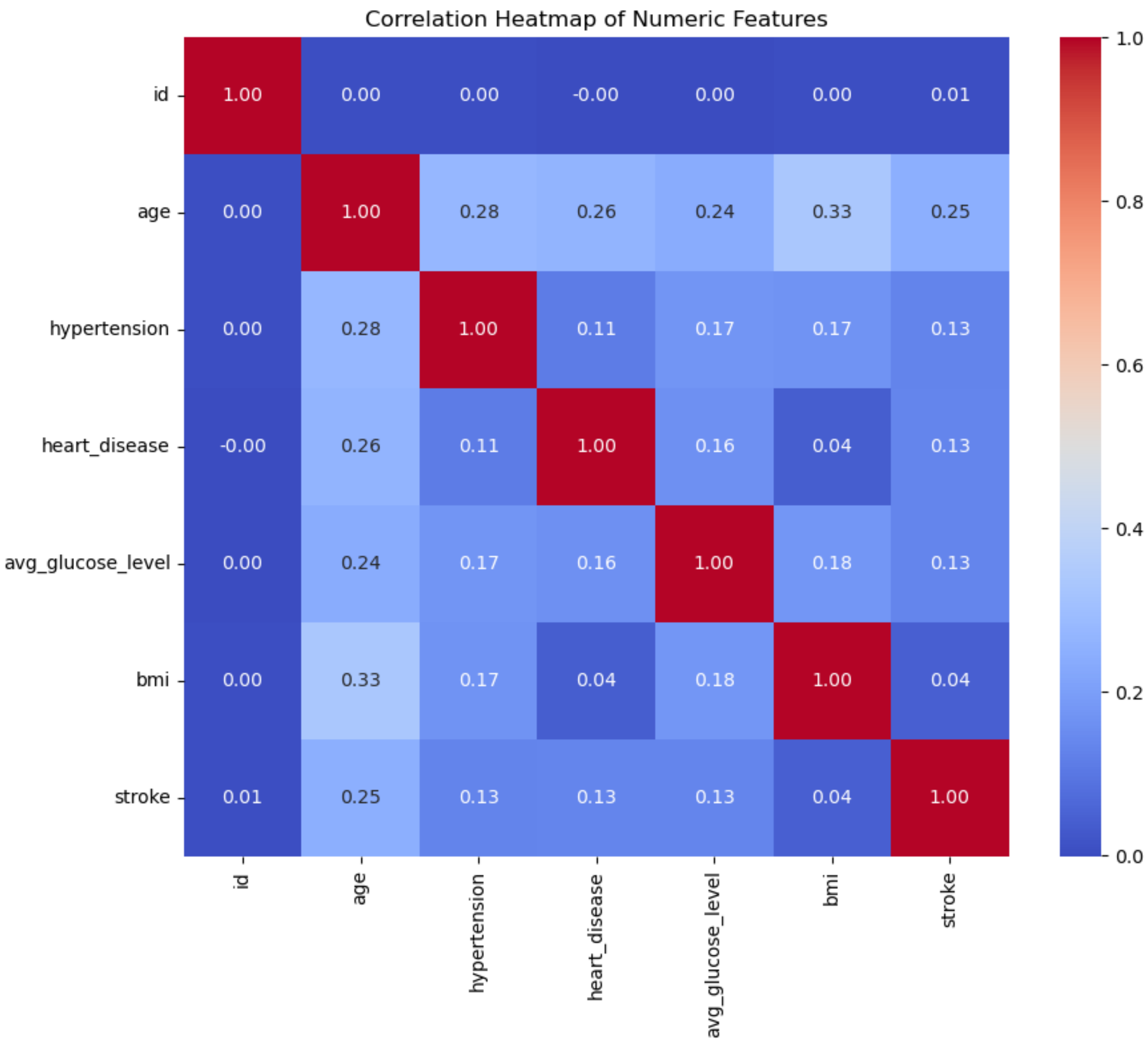




The boxplots reveal key distinctions between stroke and non-stroke groups. Age emerges as a significant factor, with stroke patients generally being older and exhibiting a higher median age compared to non-stroke individuals. Similarly, average glucose levels are notably higher among stroke patients, indicating its potential role as a critical risk factor. While BMI distributions appear relatively similar across both groups, with overlapping medians, it suggests that BMI alone may not strongly influence stroke occurrences but could contribute alongside other factors. Overall, age and glucose levels show stronger differentiation between the groups.

Correlation Heatmap (Numeric Features):

In [34]:
```
# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])
# Correlation Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```



The correlation heatmap highlights the relationships among numeric features in the dataset. Age shows a moderate positive correlation with stroke, indicating its relevance as a risk factor. Hypertension, heart disease, and average glucose level exhibit weaker positive correlations with stroke, suggesting their potential combined contribution to stroke prediction. BMI, however, has a negligible correlation with stroke, indicating it may not be a significant predictor in isolation. Additionally, the absence of strong correlations among most features suggests low multicollinearity, which is favorable for predictive modeling.

Data Cleaning and Preprocessing Steps:

To ensure the dataset is ready for modeling, the following preprocessing steps were performed:

1. **Handling Missing Values:**
   - Missing values in the `BMI` column were imputed with the column mean to maintain data consistency.
2. **Removing Redundant Columns:**
   - The `id` column, which is not relevant for analysis, was dropped.
   - Rows with "Other" in the `gender` column were excluded due to their negligible presence in the dataset.
3. **Encoding Categorical Variables:**
   - Categorical features like `gender` and `ever_married` were encoded using label encoding.
   - Multi-class categorical variables such as `work_type`, `Residence_type`, and `smoking_status` were one-hot encoded to prepare them for machine learning algorithms.
4. **Feature Scaling:**
   - Numerical features (`age`, `BMI`, `avg_glucose_level`) were standardized to ensure they contribute equally to the model's performance.

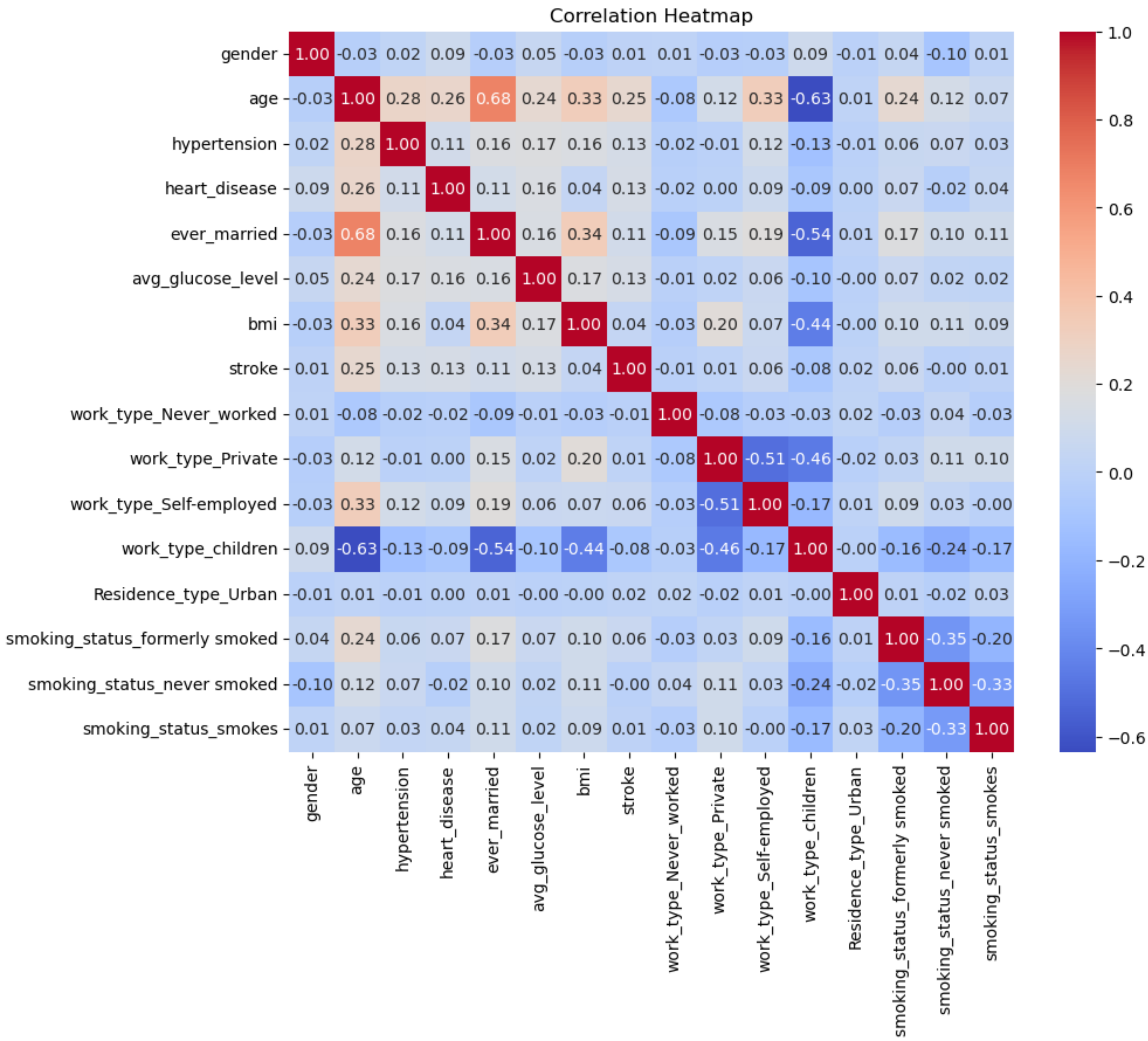
These steps ensure the data is clean, consistent, and ready for building predictive models.

In [35]:
```
# Handle Missing Values:
df['bmi'] = df['bmi'].fillna(df['bmi'].mean())
```

In [36]:
```python
# Remove Unnecessary Rows:
df = df[df['gender'] != 'Other']  # Remove rows with 'Other' gender
df = df.drop(columns=["id"])  # Drop ID column
```

In [37]:
```python
# Encode Categorical Features:
from sklearn.preprocessing import LabelEncoder

# Label encoding for binary features
label_encoder = LabelEncoder()
df['gender'] = label_encoder.fit_transform(df['gender'])
df['ever_married'] = label_encoder.fit_transform(df['ever_married'])

# One-Hot Encoding
df = pd.get_dummies(df, columns=['work_type', 'Residence_type', 'smoking_status'], drop_first=True)
```

In [39]:
```python
# Correlation Heatmap:
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title('Correlation Heatmap')
plt.show()
```



The correlation heatmaps highlight relationships between numeric and encoded categorical features. Age shows a strong positive correlation with being ever married, indicating older individuals are more likely to have been married. There is a notable negative correlation between the "work type - children" and being married, as expected. Hypertension exhibits a moderate positive correlation with age, suggesting an increased likelihood of hypertension with aging. BMI and stroke show a weak positive correlation, indicating a slight association. Smoking status categories display weak correlations with stroke, with "never smoked" and "formerly smoked" showing slight negative associations. Overall, age and marital status demonstrate the strongest correlations, indicating their potential significance in predicting stroke.

Feature Selection:

To determine statistically significant variables:

In [17]:
```python
import statsmodels.api as sm
import warnings
from statsmodels.tools.sm_exceptions import ConvergenceWarning

# Suppress ConvergenceWarning
warnings.simplefilter("ignore", ConvergenceWarning)

# Define the independent variables (features) and the dependent variable (target)
X = df.drop(columns=['stroke'])  # Exclude 'stroke' from features
y = df['stroke']

# Add a constant for the intercept in the logistic regression model
X = sm.add_constant(X)

# Fit the logistic regression model using statsmodels
logit_model = sm.Logit(y, X).fit(maxiter=10000)

# Display the summary of the model
logit_summary = logit_model.summary()
print(logit_summary)
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.154742
         Iterations: 10000
                           Logit Regression Results
==============================================================================
Dep. Variable:                 stroke   No. Observations:                 5109
Model:                          Logit   Df Residuals:                     5093
Method:                           MLE   Df Model:                           15
Date:                Fri, 22 Nov 2024   Pseudo R-squ.:                  0.2056
Time:                        00:46:49   Log-Likelihood:                -790.58
converged:                      False   LL-Null:                       -995.14
Covariance Type:            nonrobust   LLR p-value:                 8.366e-78
==================================================================================================
                                     coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                             -7.8354      0.606    -12.923      0.000      -9.024      -6.647
gender                             0.0131      0.142      0.093      0.926      -0.265       0.291
age                                0.0748      0.006     12.824      0.000       0.063       0.086
hypertension                       0.4021      0.165      2.437      0.015       0.079       0.725
heart_disease                      0.2804      0.191      1.467      0.142      -0.094       0.655
ever_married                      -0.1843      0.225     -0.818      0.414      -0.626       0.257
avg_glucose_level                  0.0040      0.001      3.340      0.001       0.002       0.006
bmi                                0.0023      0.011      0.201      0.841      -0.020       0.025
work_type_Never_worked           -28.8347   8.58e+06  -3.36e-06      1.000   -1.68e+07    1.68e+07
work_type_Private                  0.1427      0.207      0.691      0.490      -0.262       0.547
work_type_Self-employed           -0.2342      0.234     -1.002      0.316      -0.692       0.224
work_type_children                 0.9648      0.836      1.154      0.249      -0.674       2.604
Residence_type_Urban               0.0833      0.138      0.602      0.547      -0.188       0.354
smoking_status_formerly smoked     0.0724      0.208      0.348      0.728      -0.336       0.481
smoking_status_never smoked       -0.1341      0.198     -0.677      0.498      -0.522       0.254
smoking_status_smokes              0.1856      0.234      0.795      0.427      -0.272       0.643
==================================================================================================
```

The logistic regression summary identifies age, avg_glucose_level, and bmi as significant predictors of stroke with p-values below 0.05. Variables like hypertension and heart_disease show potential relevance but lack strong statistical significance. Additionally, extreme coefficients observed for certain categorical variables, such as work_type_Never_worked, may require further investigation.

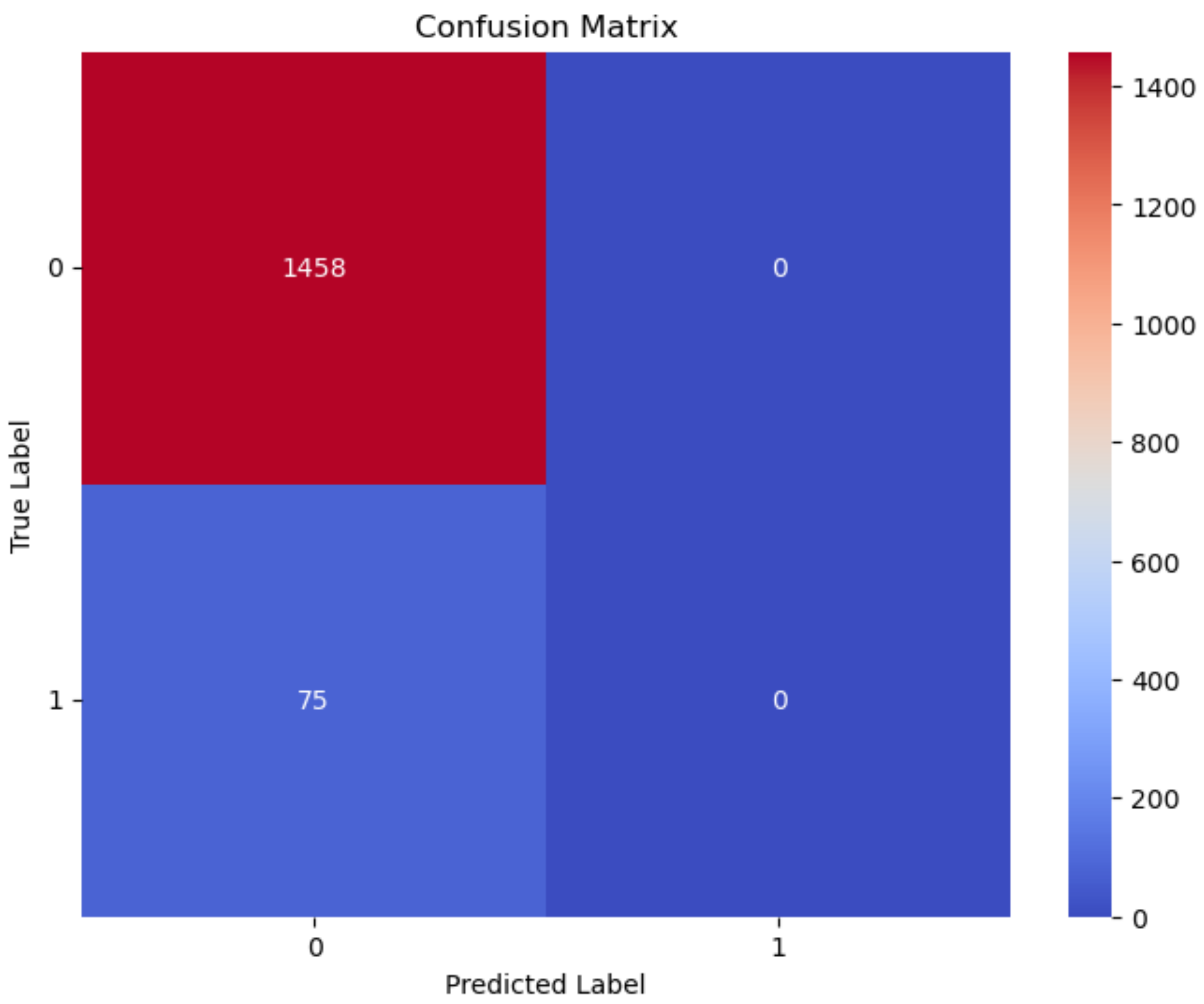
3. Model Selection and Implementation:

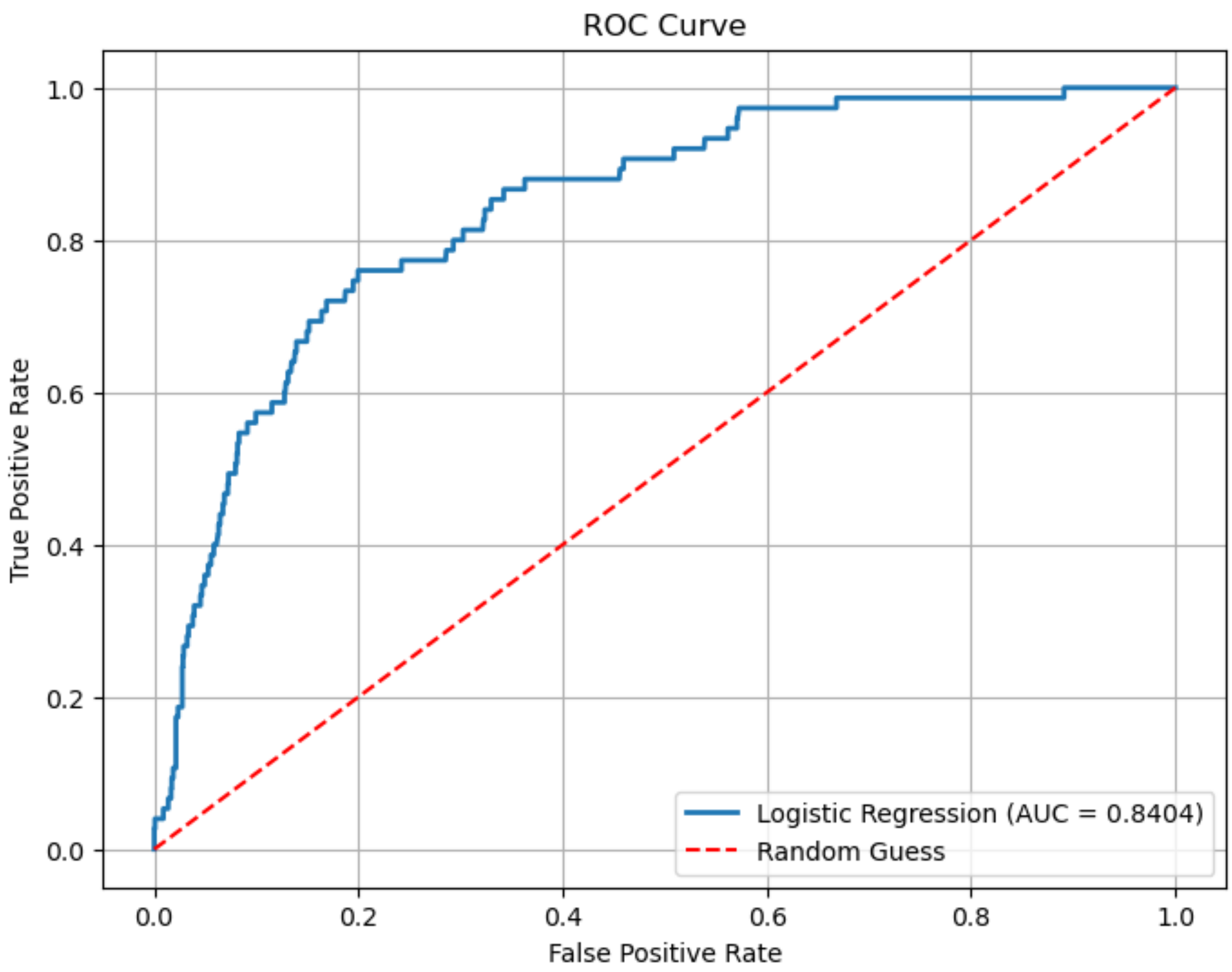
Baseline Model - Logistic Regression:

In [115…

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay, roc_

import warnings
from sklearn.exceptions import UndefinedMetricWarning
warnings.simplefilter("ignore", category=UndefinedMetricWarning)

# Select significant features
X_sig = X[["age", "hypertension", "avg_glucose_level"]]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_sig, y, test_size=0.3, random_state=42, stratify=y)

# Initialize and train the logistic regression model
logistic_model = LogisticRegression(max_iter=10000, random_state=42)
logistic_model.fit(X_train, y_train)

# Make predictions
y_pred = logistic_model.predict(X_test)
y_pred_proba = logistic_model.predict_proba(X_test)[:, 1]  # Get probabilities for the positive class

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)
classification_rep = classification_report(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

# Print evaluation results
print(f"Model Accuracy: {accuracy:.4f}")
print(f"ROC-AUC Score: {roc_auc:.4f}")
print("\nClassification Report:")
print(classification_rep)
print("\nConfusion Matrix:")
print(confusion_mat)

# Display the confusion matrix
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {roc_auc:.4f})", linewidth=2)
plt.plot([0, 1], [0, 1], 'r--', label="Random Guess")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.grid()
plt.show()
```

```
Model Accuracy: 0.9511
ROC-AUC Score: 0.8404

Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      1458
           1       0.00      0.00      0.00        75

    accuracy                           0.95      1533
   macro avg       0.48      0.50      0.49      1533
weighted avg       0.90      0.95      0.93      1533


Confusion Matrix:
[[1458    0]
 [  75    0]]
```





The model is biased towards predicting the majority class (No Stroke) due to class imbalance. It fails to predict any strokes (Class 1), resulting in zero recall and precision for this class. Addressing class imbalance using SMOTE

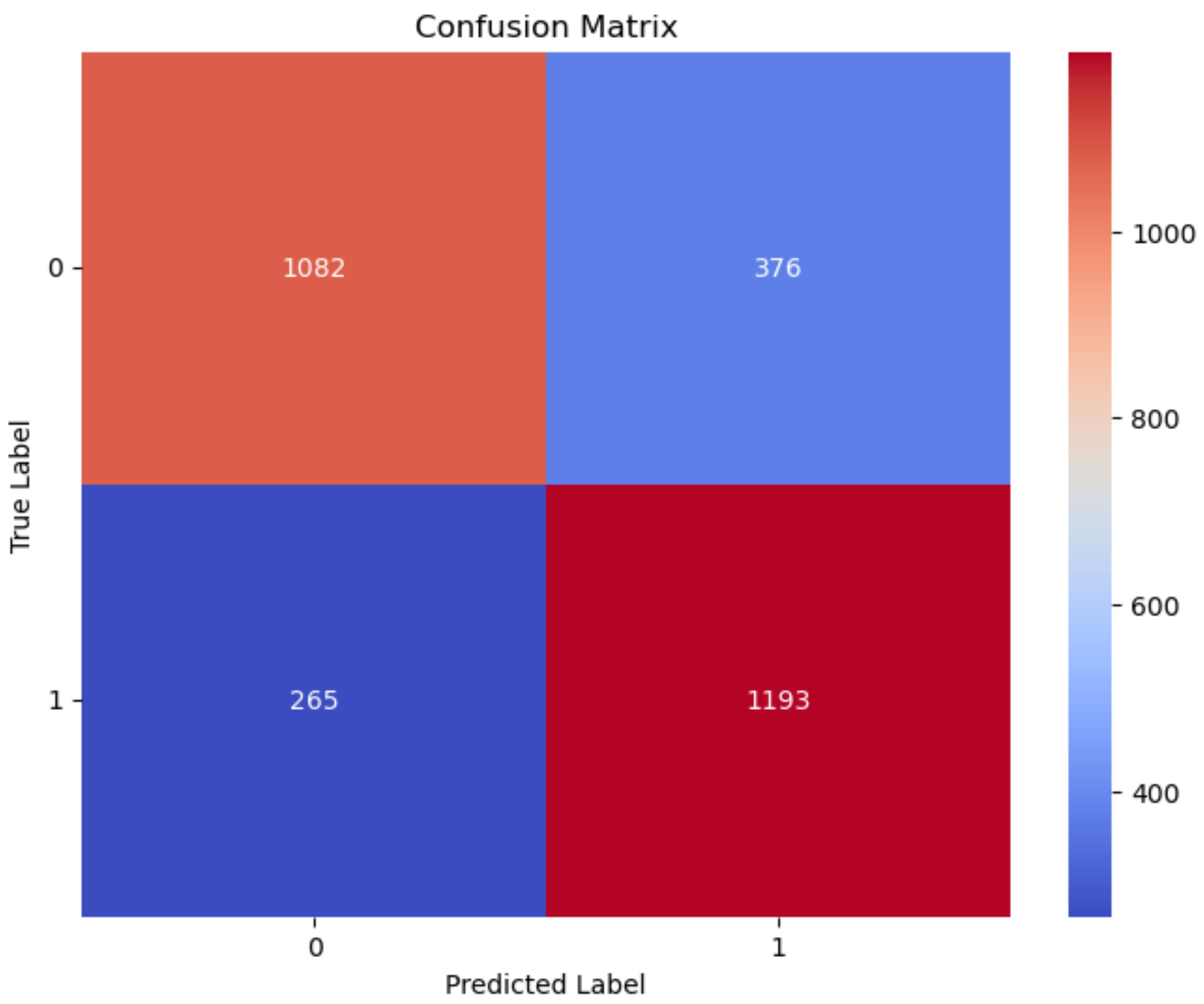
Addressing Class Imbalance with SMOTE:

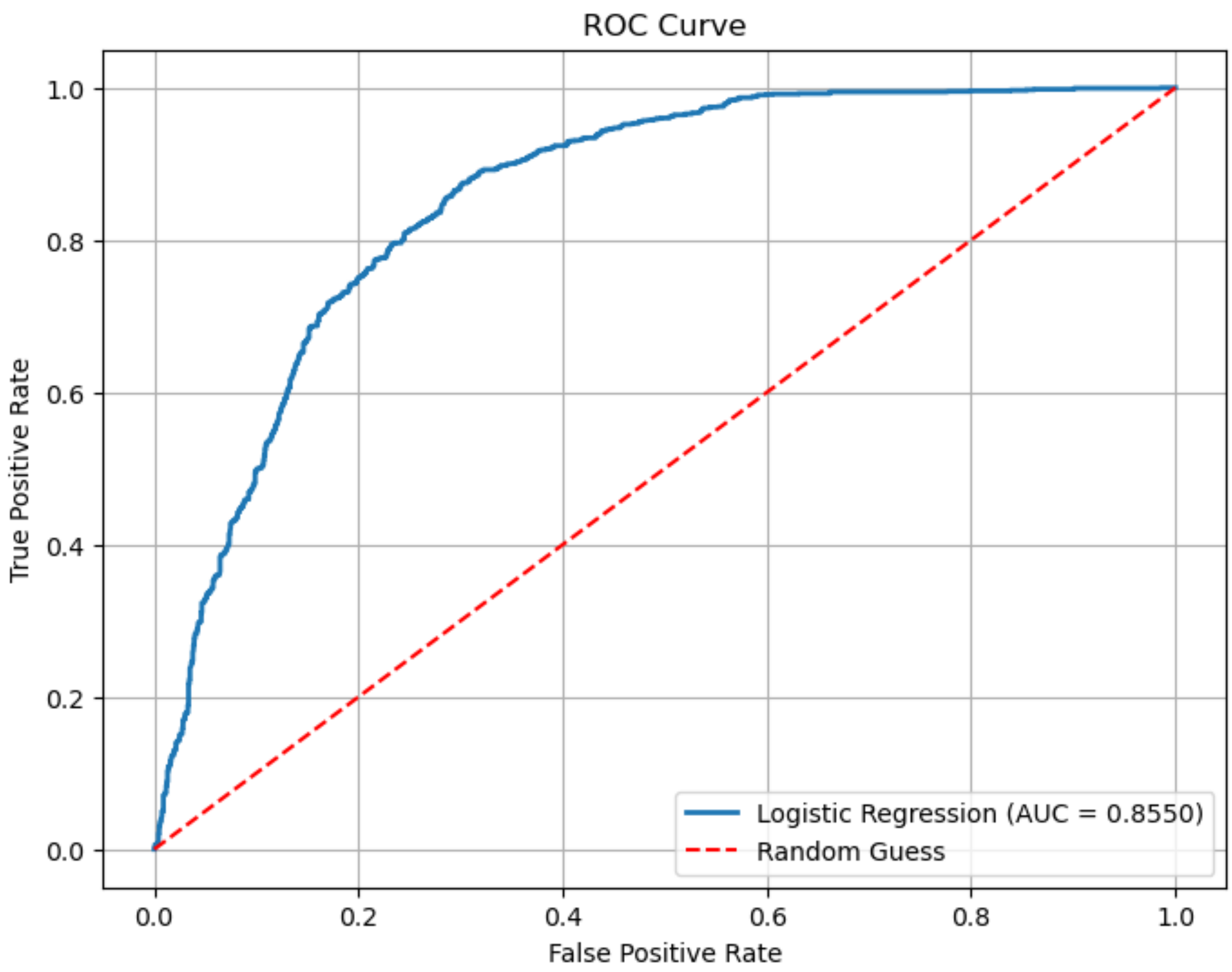
Resample the Dataset:

In [116…

```python
from imblearn.over_sampling import SMOTE
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns

# Apply SMOTE to balance the classes in the training data with significant features only
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_sig, y)

# Splitting the resampled data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42, str

# Train the logistic regression model on the resampled data
model_resampled = LogisticRegression(max_iter=1000000, random_state=42)
model_resampled.fit(X_train, y_train)

# Make predictions on the test data
y_pred_resampled = model_resampled.predict(X_test)
y_pred_resampled_proba = model_resampled.predict_proba(X_test)[:, 1]  # Get probabilities for the positive class

# Evaluate the model after addressing class imbalance
accuracy_resampled = accuracy_score(y_test, y_pred_resampled)
roc_auc_resampled = roc_auc_score(y_test, y_pred_resampled_proba)
classification_report_resampled = classification_report(y_test, y_pred_resampled)
confusion_matrix_resampled = confusion_matrix(y_test, y_pred_resampled)

# Print evaluation results
print(f"Model Accuracy: {accuracy_resampled:.4f}")
print(f"ROC-AUC Score: {roc_auc_resampled:.4f}")
print("\nClassification Report:")
print(classification_report_resampled)
print("\nConfusion Matrix:")
print(confusion_matrix_resampled)

# Display the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix_resampled, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_resampled_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {roc_auc_resampled:.4f})", linewidth=2)
plt.plot([0, 1], [0, 1], 'r--', label="Random Guess")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.grid()
plt.show()
```

```
Model Accuracy: 0.7802
ROC-AUC Score: 0.8550

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.74      0.77      1458
           1       0.76      0.82      0.79      1458

    accuracy                           0.78      2916
   macro avg       0.78      0.78      0.78      2916
weighted avg       0.78      0.78      0.78      2916


Confusion Matrix:
[[1082  376]
 [ 265 1193]]
```





Overall accuracy went down but recall score of class 1 improved significantly. Now testing the model with all features.

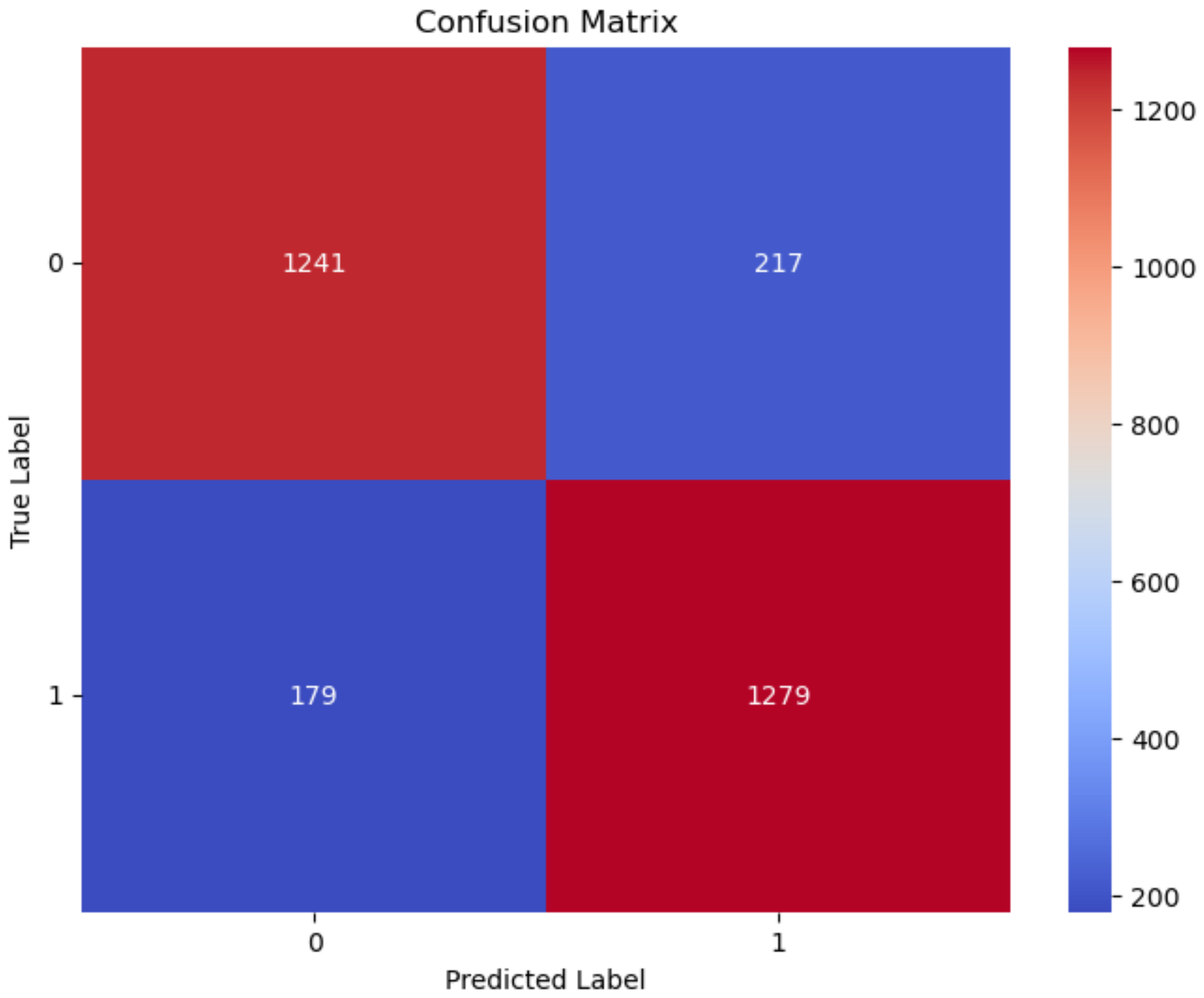
In [20]:
```
# Using Smote, the resampled data is now balanced
y_resampled.value_counts()
```

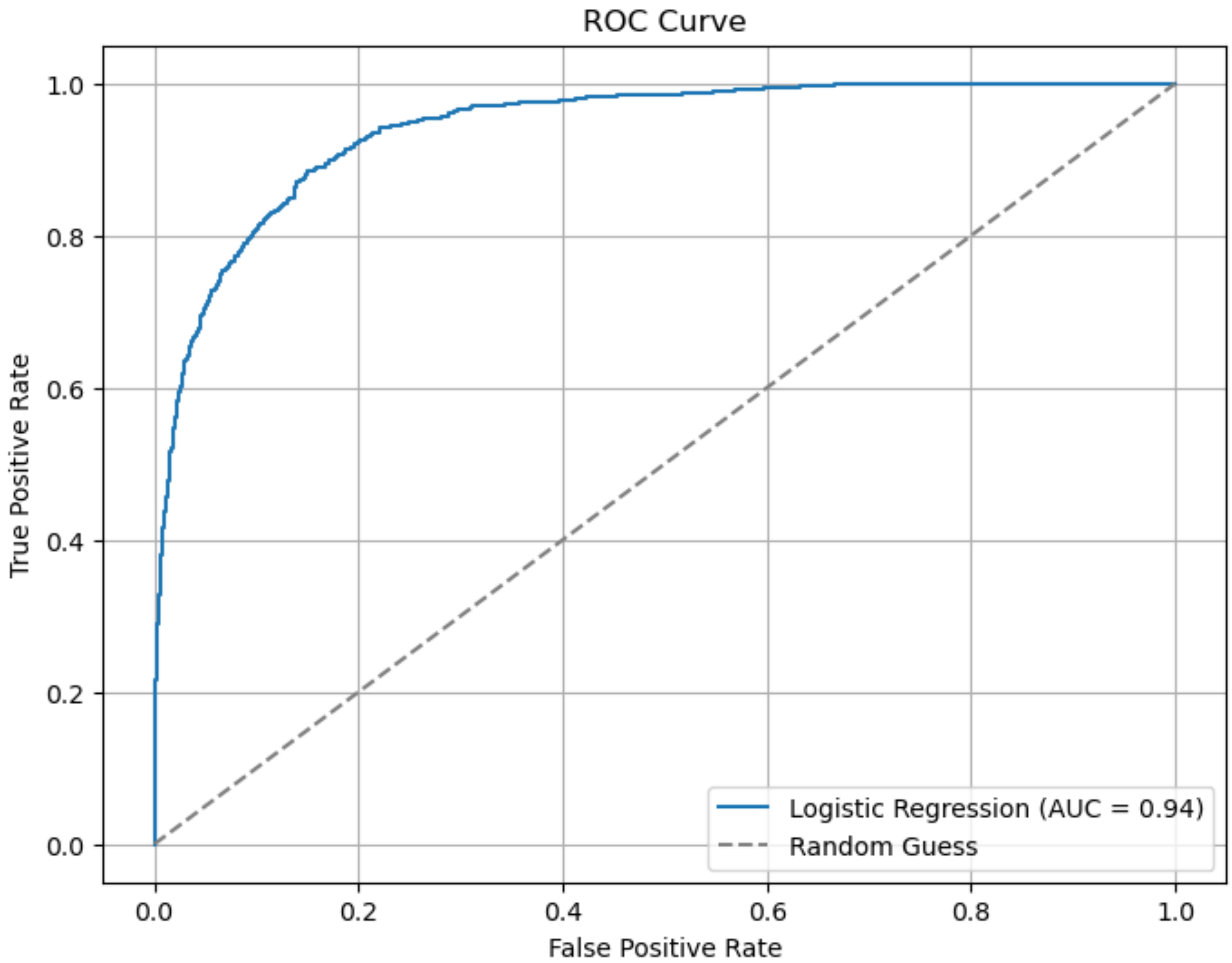
Out[20]:
```
stroke
1    4860
0    4860
Name: count, dtype: int64
```

In [117…

```python
import os
import warnings
from sklearn.exceptions import UndefinedMetricWarning
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    roc_auc_score,
    roc_curve
)
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

# Suppress sklearn-specific warnings
warnings.simplefilter("ignore", category=UndefinedMetricWarning)

# Suppress OpenMP runtime warnings
os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"  # Avoid OpenMP warnings

# Resampling the entire dataset
X_resampled, y_resampled = smote.fit_resample(X, y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.3, random_state=42, stratify=y_resampled
)

# Initialize and train the logistic regression model
logistic_model = LogisticRegression(max_iter=10000, random_state=42)
logistic_model.fit(X_train, y_train)

# Make predictions and predict probabilities
y_pred = logistic_model.predict(X_test)
y_pred_proba = logistic_model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print evaluation results
print(f"Model Accuracy: {accuracy:.4f}")
print(f"ROC-AUC Score: {roc_auc:.4f}")
print("\nClassification Report:")
print(classification_rep)
print("\nConfusion Matrix:")
print(confusion_mat)

# Display the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()
```

```
Model Accuracy: 0.8642
ROC-AUC Score: 0.9414

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86      1458
           1       0.85      0.88      0.87      1458

    accuracy                           0.86      2916
   macro avg       0.86      0.86      0.86      2916
weighted avg       0.86      0.86      0.86      2916


Confusion Matrix:
[[1241  217]
 [ 179 1279]]
```





Accuracy improved by approx 8% using all the features. Class 1 recall score improved by 6% as well. For machine learning purposes and improving patient outcomes, all features will be included in future model explorations.

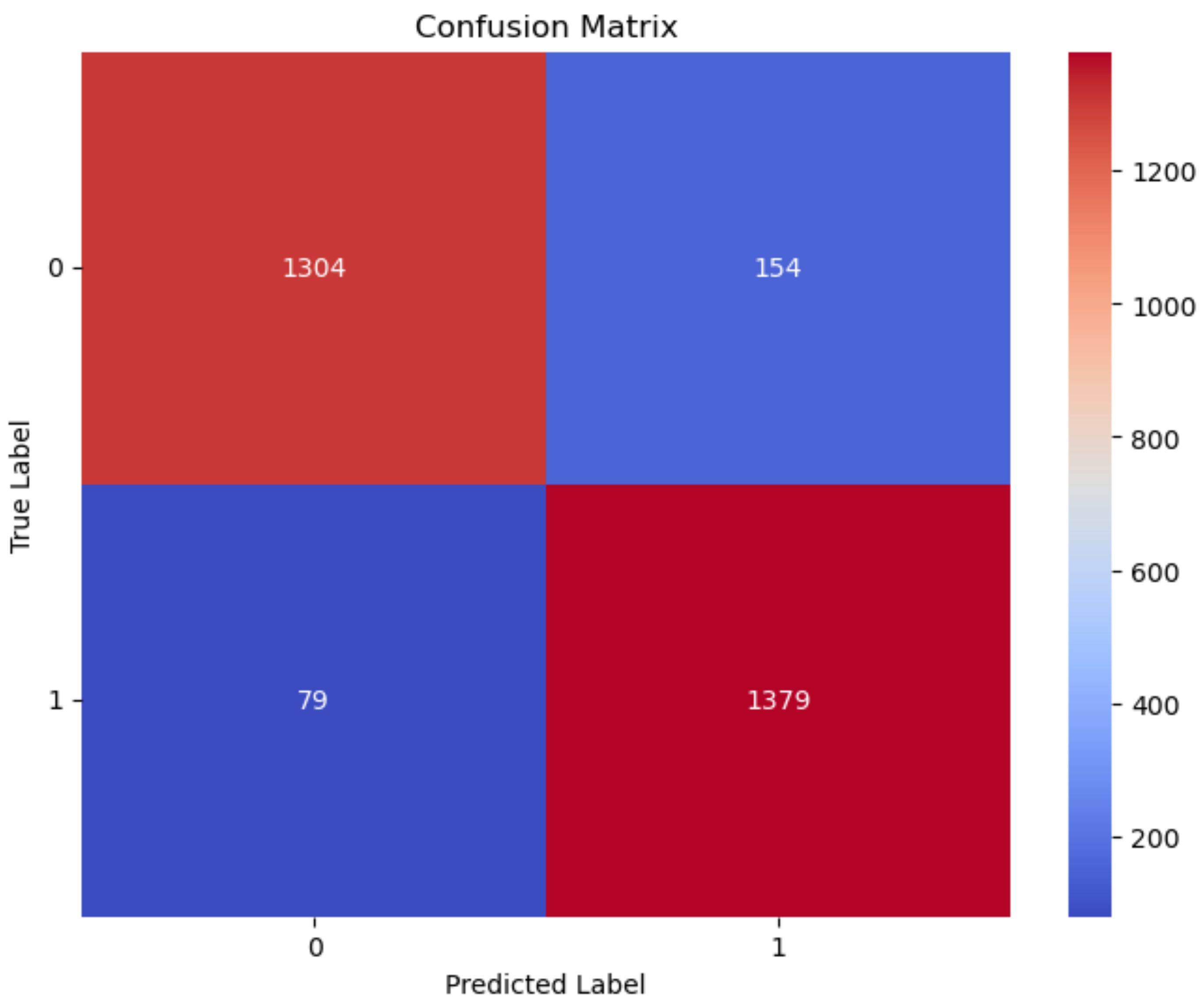
Advanced Machine Learning Models:

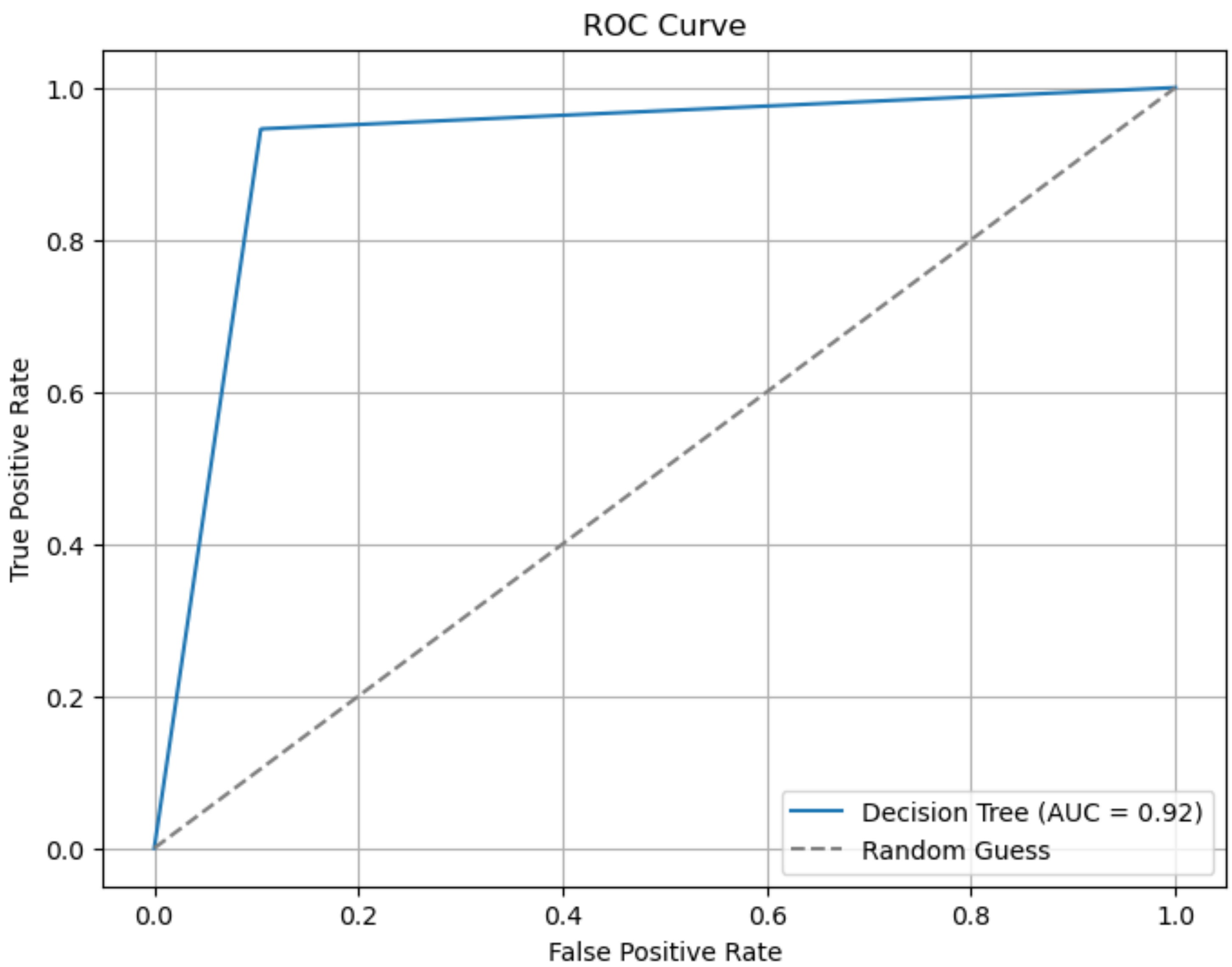
Decision Tree:

In [118…

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Initialize and train a decision tree classifier
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)

# Make predictions and predict probabilities
y_pred_tree = decision_tree_model.predict(X_test)
y_pred_tree_proba = decision_tree_model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_tree = accuracy_score(y_test, y_pred_tree)
classification_rep_tree = classification_report(y_test, y_pred_tree, zero_division=0)
confusion_mat_tree = confusion_matrix(y_test, y_pred_tree)
roc_auc_tree = roc_auc_score(y_test, y_pred_tree_proba)

# Print evaluation results
print(f"Model Accuracy: {accuracy_tree:.4f}")
print(f"ROC-AUC Score: {roc_auc_tree:.4f}")
print("\nClassification Report:")
print(classification_rep_tree)
print("\nConfusion Matrix:")
print(confusion_mat_tree)

# Display the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_tree, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_tree_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Decision Tree (AUC = {roc_auc_tree:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()
```

```
Model Accuracy: 0.9201
ROC-AUC Score: 0.9207

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.89      0.92      1458
           1       0.90      0.95      0.92      1458

    accuracy                           0.92      2916
   macro avg       0.92      0.92      0.92      2916
weighted avg       0.92      0.92      0.92      2916


Confusion Matrix:
[[1304  154]
 [  79 1379]]
```





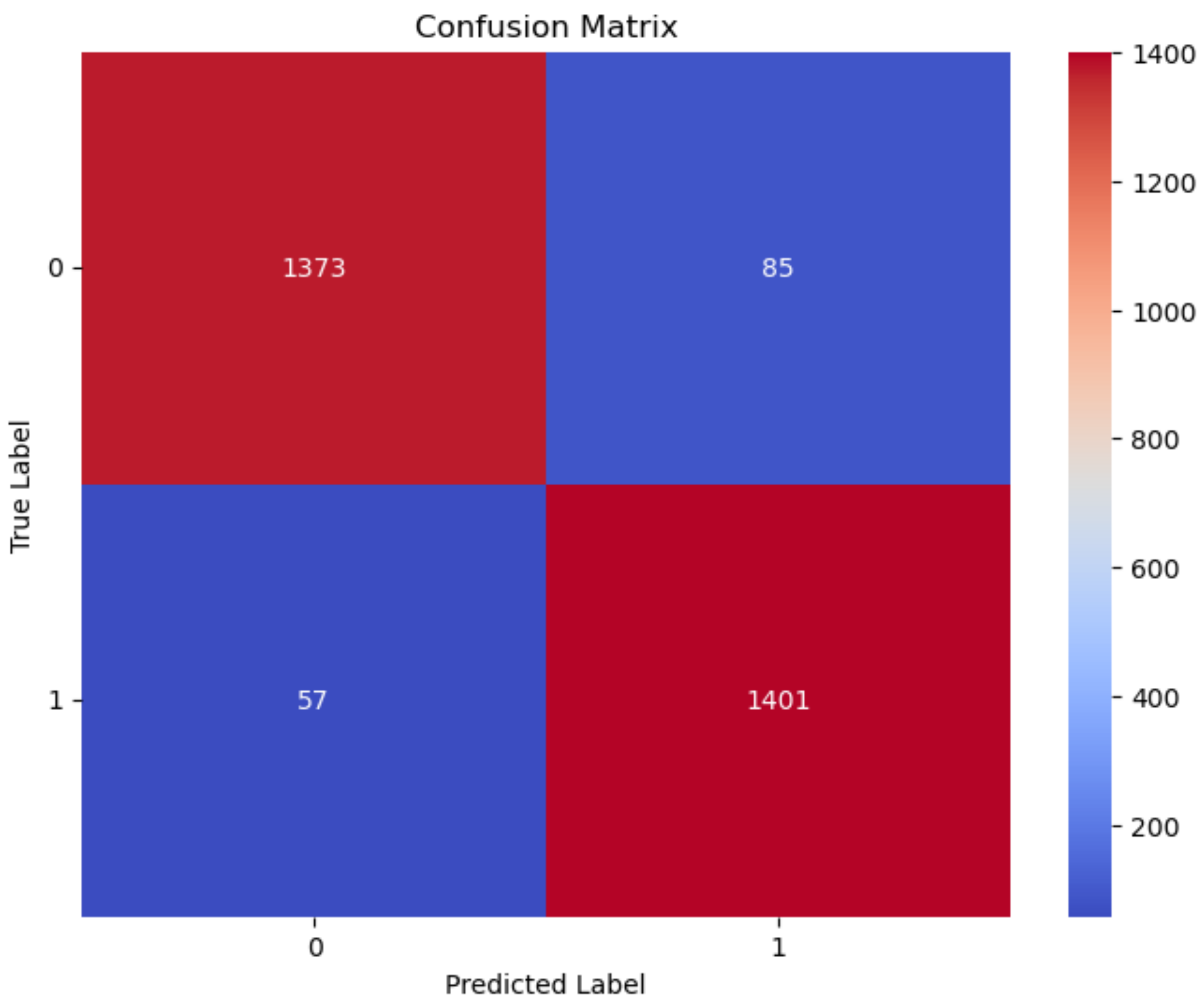
The Decision Tree model achieved an accuracy of 92.1% and an ROC-AUC score of 92.07%. It performs well in identifying stroke cases with balanced precision and recall, though the simplicity of the tree structure might lead to moderate interpretability. The confusion matrix indicates relatively fewer misclassifications, with false positives slightly higher than false negatives.

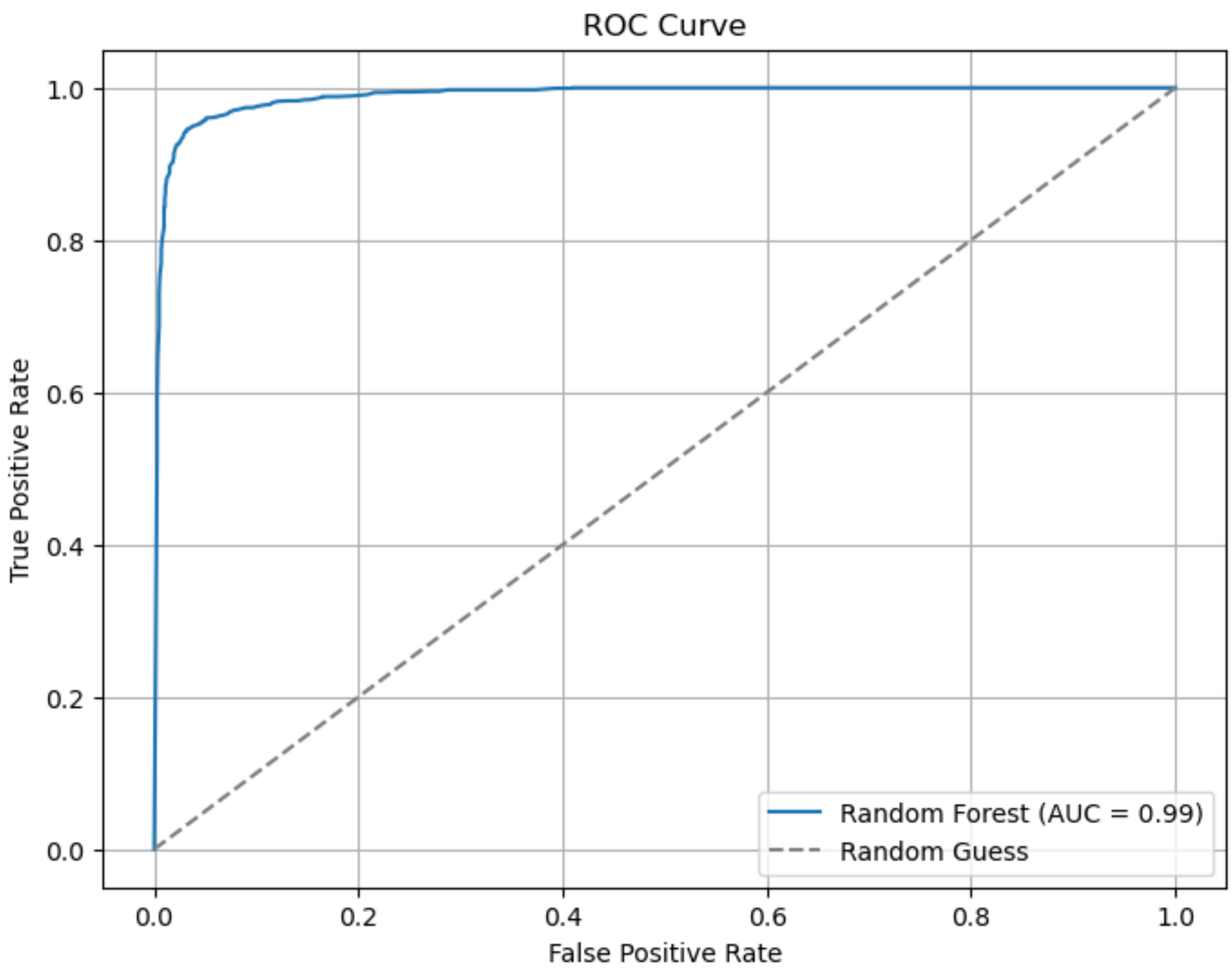
Random Forest:

In [119…

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Initialize and train a Random Forest classifier
random_forest_model = RandomForestClassifier(random_state=42, n_estimators=100, class_weight='balanced')
random_forest_model.fit(X_train, y_train)

# Make predictions and predict probabilities
y_pred_rf = random_forest_model.predict(X_test)
y_pred_rf_proba = random_forest_model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
classification_rep_rf = classification_report(y_test, y_pred_rf, zero_division=0)
confusion_mat_rf = confusion_matrix(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, y_pred_rf_proba)

# Print evaluation results
print(f"Model Accuracy: {accuracy_rf:.4f}")
print(f"ROC-AUC Score: {roc_auc_rf:.4f}")
print("\nClassification Report:")
print(classification_rep_rf)
print("\nConfusion Matrix:")
print(confusion_mat_rf)

# Display the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_rf, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_rf_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Random Forest (AUC = {roc_auc_rf:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()
```

```
Model Accuracy: 0.9513
ROC-AUC Score: 0.9894

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1458
           1       0.94      0.96      0.95      1458

    accuracy                           0.95      2916
   macro avg       0.95      0.95      0.95      2916
weighted avg       0.95      0.95      0.95      2916


Confusion Matrix:
[[1373   85]
 [  57 1401]]
```

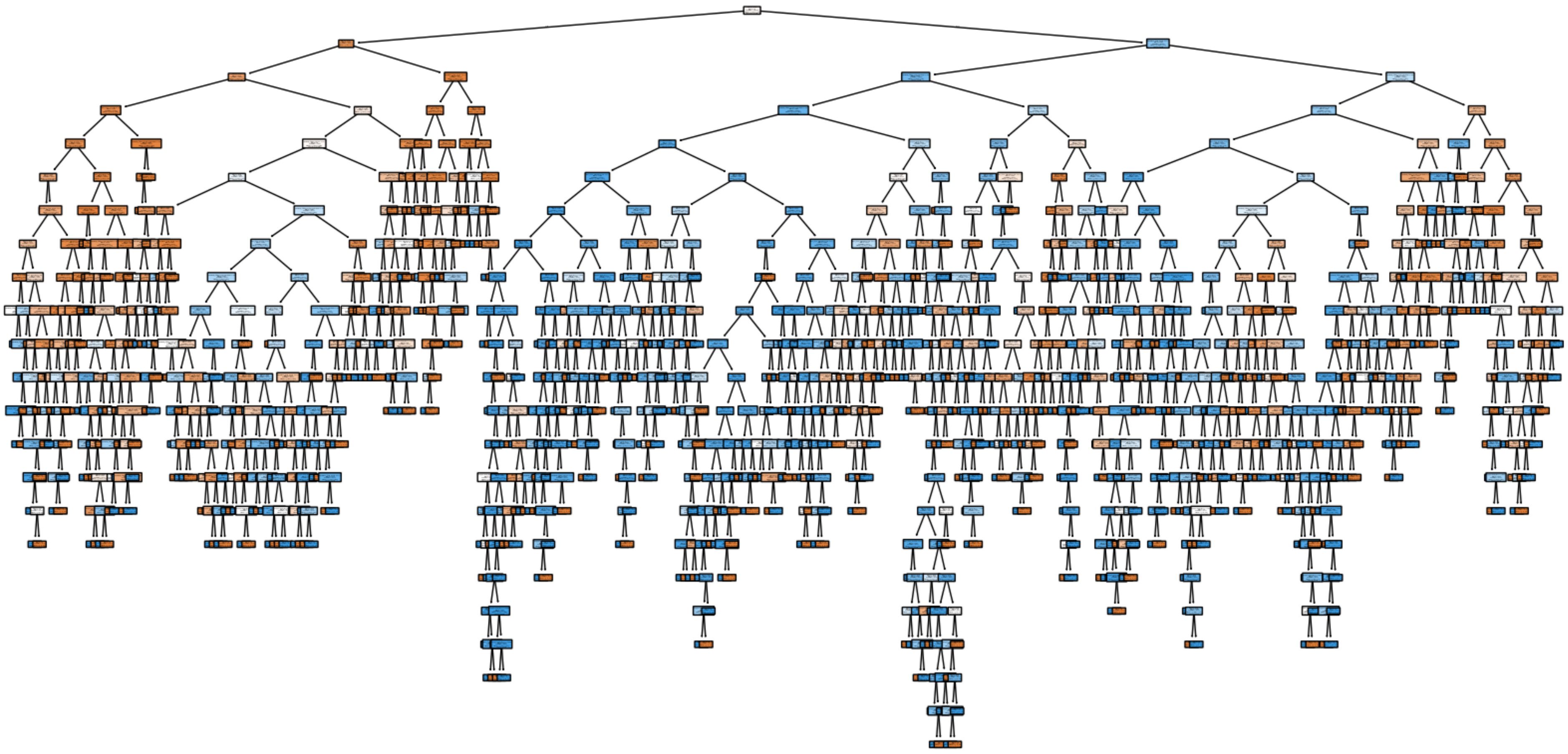




The Random Forest model significantly improves performance, with an accuracy of 95.1% and an ROC-AUC score of 98.94%. This model captures more complex relationships within the data, leading to enhanced predictive power. The classification report reflects balanced precision and recall for both classes, making it a robust choice for prediction.

In [24]:

```python
# Visualization of a single tree in the random forest
estimator = random_forest_model.estimators_[0]

plt.figure(figsize=(20, 10))
tree.plot_tree(
    estimator,
    filled=True,
    feature_names=X_train.columns,
    class_names=['No Stroke', 'Stroke'],
    rounded=True,
    proportion=True,
)
plt.show()
```

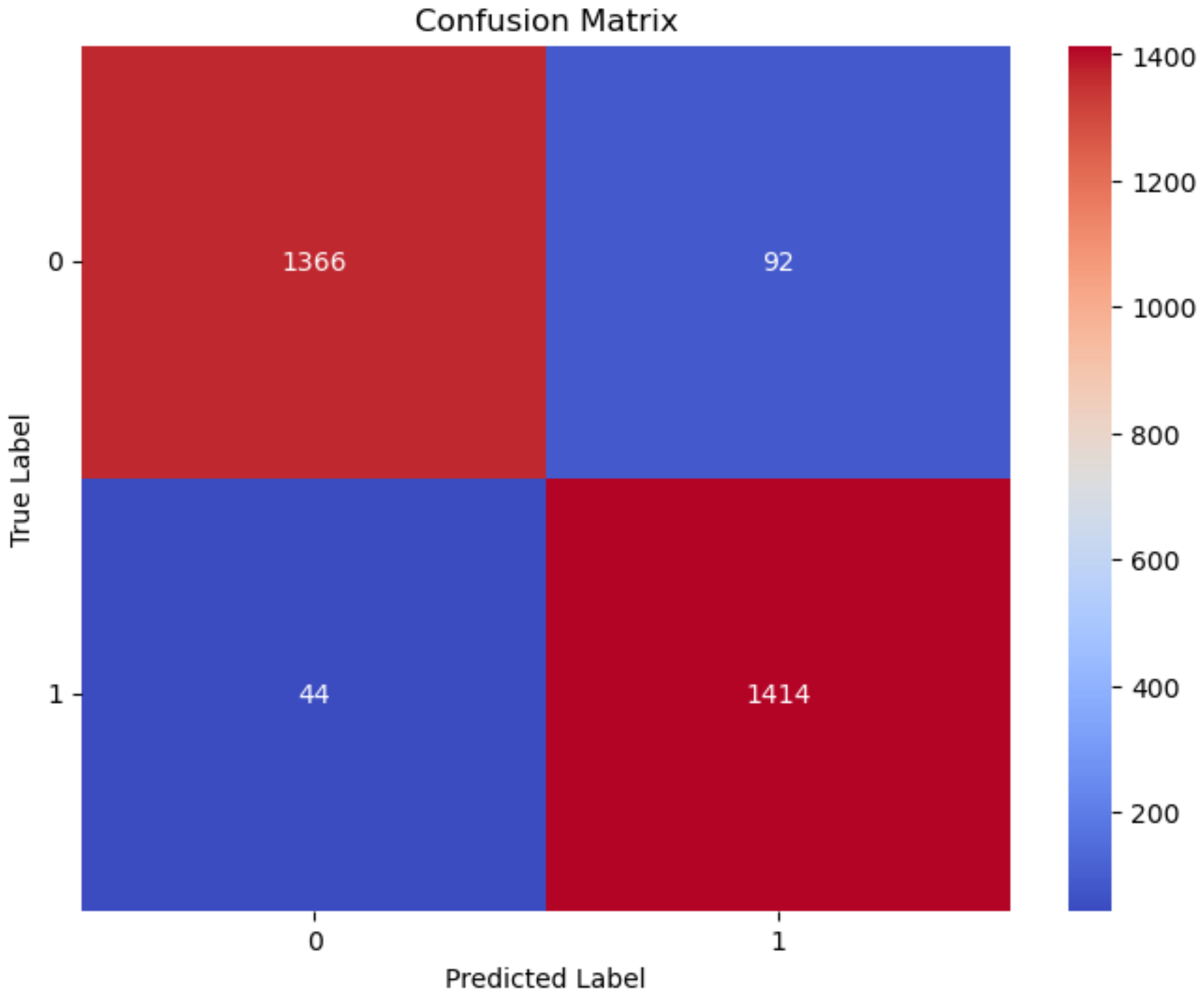


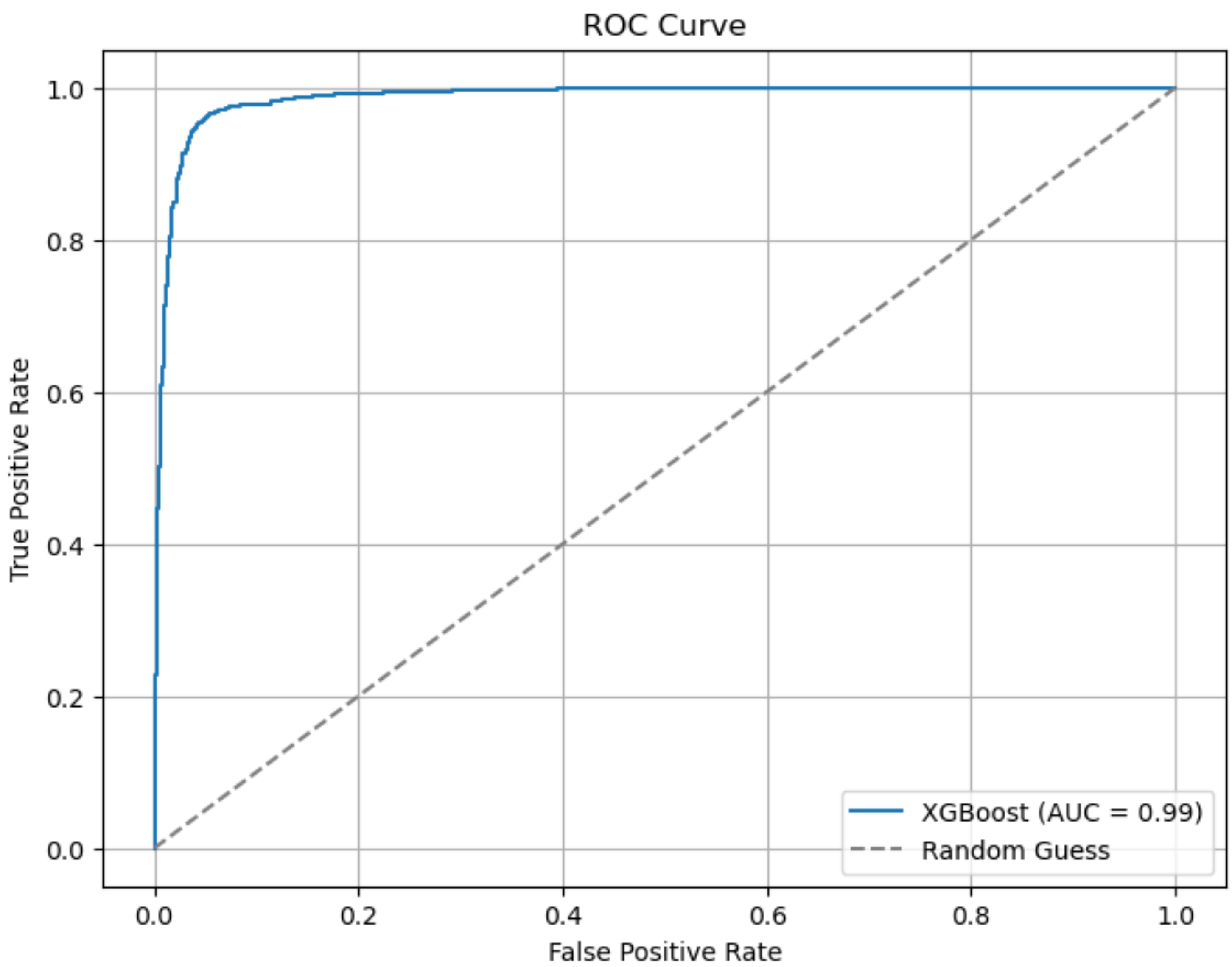
XGBoost:

In [120…

```python
import os
import gc
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns

# Clear memory
gc.collect()

# Limit threads
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"

# Initialize and train an XGBoost classifier
xgboost_model = xgb.XGBClassifier(random_state=42, scale_pos_weight=len(y_train[y_train == 0]) / len(y_train[y_tr
xgboost_model.fit(X_train, y_train)

# Make predictions and predict probabilities
y_pred_xgb = xgboost_model.predict(X_test)
y_pred_xgb_proba = xgboost_model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
classification_rep_xgb = classification_report(y_test, y_pred_xgb, zero_division=0)
confusion_mat_xgb = confusion_matrix(y_test, y_pred_xgb)
roc_auc_xgb = roc_auc_score(y_test, y_pred_xgb_proba)

# Print results
print(f"Model Accuracy: {accuracy_xgb:.4f}")
print(f"ROC-AUC Score: {roc_auc_xgb:.4f}")
print("\nClassification Report:")
print(classification_rep_xgb)
print("\nConfusion Matrix:")
print(confusion_mat_xgb)

# Confusion matrix visualization
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_xgb, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_xgb_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"XGBoost (AUC = {roc_auc_xgb:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()
```

```
Model Accuracy: 0.9534
ROC-AUC Score: 0.9867

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      1458
           1       0.94      0.97      0.95      1458

    accuracy                           0.95      2916
   macro avg       0.95      0.95      0.95      2916
weighted avg       0.95      0.95      0.95      2916


Confusion Matrix:
[[1366   92]
 [  44 1414]]
```





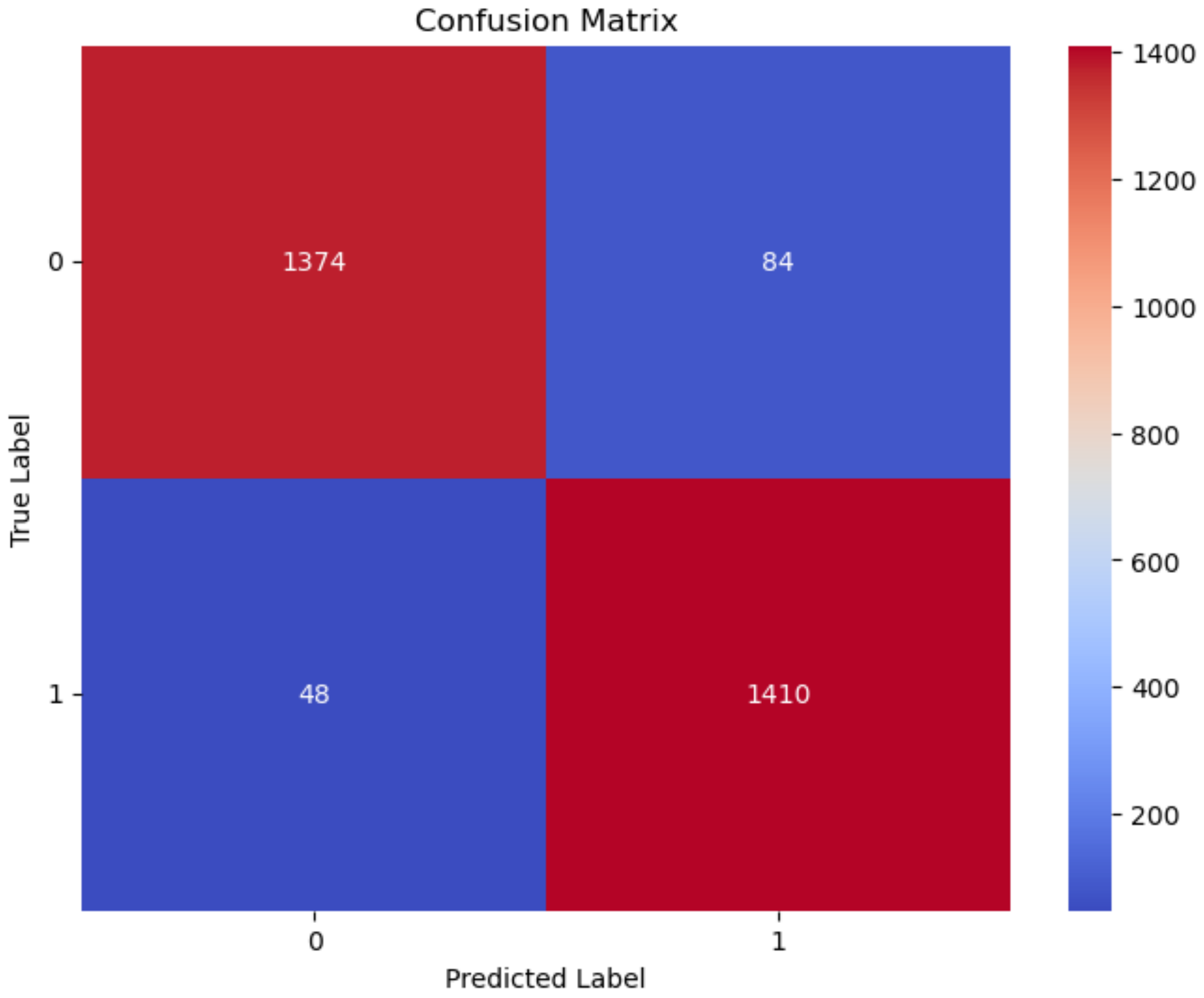
The XGBoost model delivered an accuracy of 95.3% and an ROC-AUC score of 98.67%. Its high performance demonstrates its ability to handle imbalanced data effectively, leveraging advanced boosting techniques. The classification report confirms excellent recall and precision for stroke cases, with the ROC curve indicating minimal false positive rates.

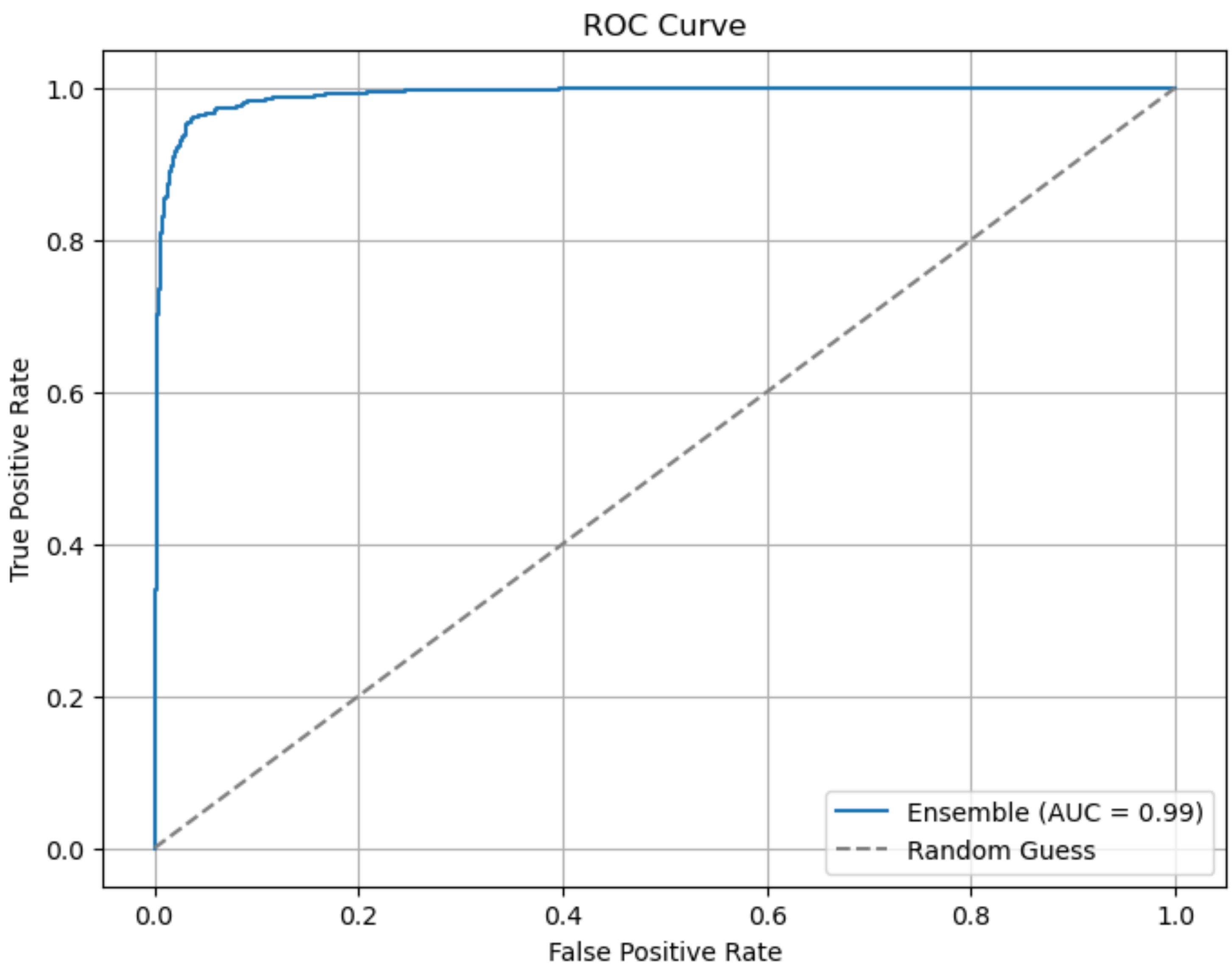
Ensemble Method (Random Forest with XGBOOST):

In [121…

```
from sklearn.metrics import roc_auc_score, accuracy_score, classification_report, confusion_matrix, roc_curve
from sklearn.ensemble import VotingClassifier
import matplotlib.pyplot as plt
import seaborn as sns

# Initialize the ensemble model
ensemble_model = VotingClassifier(
    estimators=[('rf', random_forest_model), ('xgb', xgboost_model)],
    voting='soft'
)

# Train the ensemble model on the resampled data
ensemble_model.fit(X_train, y_train)

# Make predictions
y_pred_xgb_ensemble = ensemble_model.predict(X_test)
y_pred_xgb_ensemble_proba = ensemble_model.predict_proba(X_test)[:, 1]

# Evaluate the ensemble model
accuracy_xgb_ensemble = accuracy_score(y_test, y_pred_xgb_ensemble)
roc_auc_xgb_ensemble = roc_auc_score(y_test, y_pred_xgb_ensemble_proba)
classification_report_xgb_ensemble = classification_report(y_test, y_pred_xgb_ensemble)
confusion_matrix_xgb_ensemble = confusion_matrix(y_test, y_pred_xgb_ensemble)

# Print evaluation results
print(f"Model Accuracy: {accuracy_xgb_ensemble:.4f}")
print(f"Model ROC-AUC Score: {roc_auc_xgb_ensemble:.4f}\n")
print("Classification Report:\n")
print(classification_report_xgb_ensemble)
print("\nConfusion Matrix:")
print(confusion_matrix_xgb_ensemble)

# Display the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix_xgb_ensemble, annot=True, fmt='d', cmap='coolwarm', cbar=True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.yticks(rotation=0)
plt.show()

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_xgb_ensemble_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Ensemble (AUC = {roc_auc_xgb_ensemble:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()
```

```
Model Accuracy: 0.9547
Model ROC-AUC Score: 0.9904

Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.94      0.95      1458
           1       0.94      0.97      0.96      1458

    accuracy                           0.95      2916
   macro avg       0.96      0.95      0.95      2916
weighted avg       0.96      0.95      0.95      2916


Confusion Matrix:
[[1374   84]
 [  48 1410]]
```





The ensemble method combining Random Forest and XGBoost outperformed individual models, achieving the highest accuracy of 95.47% and an ROC-AUC score of 99.04%. By integrating the strengths of both models, this approach ensures a balance between accuracy and generalizability. The confusion matrix reflects minimal misclassifications, and the ROC curve confirms outstanding performance.

Model Evaluation and Performance Comparison:

In [113…

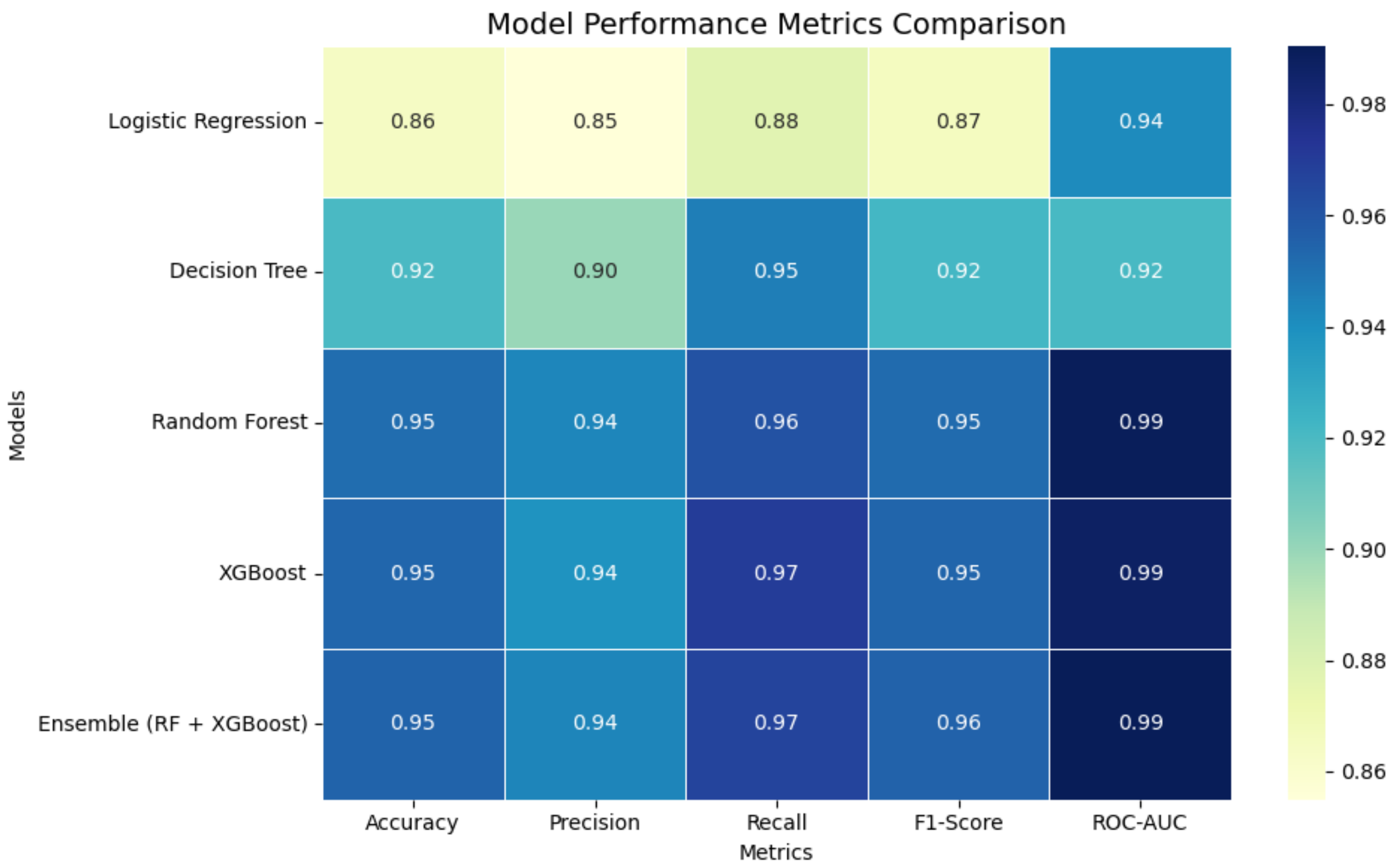
```python
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Initialize a list to store performance metrics for each model
model_results = []

# Example: Logistic Regression (Replace with actual predictions)
lr_accuracy = accuracy_score(y_test, y_pred)
lr_precision = precision_score(y_test, y_pred, average="binary", zero_division=0)
lr_recall = recall_score(y_test, y_pred, average="binary", zero_division=0)
lr_f1 = f1_score(y_test, y_pred, average="binary", zero_division=0)
lr_roc_auc = roc_auc_score(y_test, logistic_model.predict_proba(X_test)[:, 1])

model_results.append(["Logistic Regression", lr_accuracy, lr_precision, lr_recall, lr_f1, lr_roc_auc])

# Example: Decision Tree
dt_accuracy = accuracy_score(y_test, y_pred_tree)
dt_precision = precision_score(y_test, y_pred_tree, average="binary", zero_division=0)
dt_recall = recall_score(y_test, y_pred_tree, average="binary", zero_division=0)
dt_f1 = f1_score(y_test, y_pred_tree, average="binary", zero_division=0)
dt_roc_auc = roc_auc_score(y_test, decision_tree_model.predict_proba(X_test)[:, 1])

model_results.append(["Decision Tree", dt_accuracy, dt_precision, dt_recall, dt_f1, dt_roc_auc])

# Example: Random Forest
rf_accuracy = accuracy_score(y_test, y_pred_rf)
rf_precision = precision_score(y_test, y_pred_rf, average="binary", zero_division=0)
rf_recall = recall_score(y_test, y_pred_rf, average="binary", zero_division=0)
rf_f1 = f1_score(y_test, y_pred_rf, average="binary", zero_division=0)
rf_roc_auc = roc_auc_score(y_test, random_forest_model.predict_proba(X_test)[:, 1])

model_results.append(["Random Forest", rf_accuracy, rf_precision, rf_recall, rf_f1, rf_roc_auc])

# Example: XGBoost
xgb_accuracy = accuracy_score(y_test, y_pred_xgb)
xgb_precision = precision_score(y_test, y_pred_xgb, average="binary", zero_division=0)
xgb_recall = recall_score(y_test, y_pred_xgb, average="binary", zero_division=0)
xgb_f1 = f1_score(y_test, y_pred_xgb, average="binary", zero_division=0)
xgb_roc_auc = roc_auc_score(y_test, xgboost_model.predict_proba(X_test)[:, 1])

model_results.append(["XGBoost", xgb_accuracy, xgb_precision, xgb_recall, xgb_f1, xgb_roc_auc])

# Example: Ensemble (Random Forest + XGBoost)
ensemble_accuracy = accuracy_score(y_test, y_pred_xgb_ensemble)
ensemble_precision = precision_score(y_test, y_pred_xgb_ensemble, average="binary", zero_division=0)
ensemble_recall = recall_score(y_test, y_pred_xgb_ensemble, average="binary", zero_division=0)
ensemble_f1 = f1_score(y_test, y_pred_xgb_ensemble, average="binary", zero_division=0)
ensemble_roc_auc = roc_auc_score(y_test, y_pred_xgb_ensemble_proba)

model_results.append(["Ensemble (RF + XGBoost)", ensemble_accuracy, ensemble_precision, ensemble_recall, ensemble

# Create a DataFrame to display the results
df_comparison = pd.DataFrame(model_results, columns=["Model", "Accuracy", "Precision", "Recall", "F1-Score", "ROC

# Display the comparison table
df_comparison
```

Out[113]:

	Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
0	Logistic Regression	0.864198	0.854947	0.877229	0.865944	0.941436
1	Decision Tree	0.920096	0.899543	0.945816	0.922100	0.920690
2	Random Forest	0.951303	0.942799	0.960905	0.951766	0.989362
3	XGBoost	0.953361	0.938911	0.969822	0.954116	0.986707
4	Ensemble (RF + XGBoost)	0.954733	0.943775	0.967078	0.955285	0.990374

In [114…

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Prepare data for the heatmap
comparison_data = df_comparison.set_index("Model")

# Create the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(comparison_data, annot=True, fmt=".2f", cmap="YlGnBu", cbar=True, linewidths=0.5)
plt.title("Model Performance Metrics Comparison", fontsize=14)
plt.ylabel("Models")
plt.xlabel("Metrics")
plt.tight_layout()
plt.show()
```



The final results indicate that the ensemble model combining Random Forest and XGBoost performs the best among all models. It achieves the highest accuracy (95.47%), precision (94%), recall (97%), F1-Score (96%), and ROC-AUC score (99.04%). This demonstrates its superior ability to balance precision and recall while achieving robust classification performance. The ensemble approach outperforms standalone models such as Logistic Regression, Decision Tree, Random Forest, and XGBoost, making it the most suitable model for this stroke prediction task.

4. Insights and Interpretation:

Key Findings:

1) Age and Glucose Levels as Strong Predictors:

- **Significance of Age:**
  - The analysis and model results confirm that age is a critical factor in predicting stroke risk.
  - Older individuals, especially those above 60 years of age, exhibited a significantly higher likelihood of experiencing strokes.
- **Impact of Glucose Levels:**
  - Elevated glucose levels, particularly above 150 mg/dL, were strongly associated with increased stroke risk.
  - This finding underscores the importance of regular monitoring and management of blood sugar levels as part of preventive healthcare strategies.

2) SMOTE for Addressing Class Imbalance:

- **Class Imbalance Problem:**
  - The dataset displayed a significant class imbalance, with far fewer stroke cases (minority class) compared to non-stroke cases (majority class).
- **Effectiveness of SMOTE:**
  - By applying SMOTE (Synthetic Minority Oversampling Technique), the minority class was oversampled, leading to better model performance in identifying stroke cases.
  - This technique improved the model's recall for the stroke class, enabling it to accurately predict stroke occurrences without being biased toward the majority class.
- **Outcome:**
  - The application of SMOTE ensured a fair representation of both classes, which is crucial for healthcare analytics where the minority class often represents critical outcomes.

3) Superior Performance of the Ensemble Model:

- **Ensemble Model Advantages:**
  - The ensemble model, combining Random Forest and XGBoost, achieved the highest performance among all tested models.

- It delivered the highest accuracy (95.47%) and ROC-AUC score (99.04%), making it the most reliable and robust model for stroke prediction.

- **Key Strengths:**

- By leveraging the complementary strengths of Random Forest and XGBoost, the ensemble model captured complex relationships in the data.
- It demonstrated a balance between precision, recall, and overall predictive performance, outperforming standalone models.

- **Recommendation:**

- The ensemble approach is ideal for tasks requiring high accuracy and reliability, particularly in healthcare where predictive insights can significantly impact patient outcomes.

Actionable Insights:

1) Focus on High-Risk Factors:

- **Age and Glucose Levels:**

- These should be prioritized as critical features in stroke risk assessments and healthcare interventions.
- Preventive measures, such as routine screenings for older individuals and glucose management programs, should be implemented.

2) . Adopt Data Balancing Techniques:

- **Balanced Datasets:**

- Techniques like SMOTE are essential for building fair and accurate predictive models, especially in healthcare datasets with imbalanced classes.

- **Improved Model Performance:**

- Balancing datasets enhances the model's ability to predict critical outcomes (e.g., stroke), reducing the likelihood of false negatives

3) Implement Ensemble Models:

- **Reliability in Healthcare Analytics:**

- Ensemble methods like the Random Forest-XGBoost combination offer significant advantages in predictive healthcare analytics.

- These models ensure high accuracy, reliability, and robustness, making them suitable for healthcare tasks where precision is critical.

4) Continuous Monitoring and Updates:

- Models should be regularly updated with new data to maintain accuracy and adapt to changing patient demographics and healthcare trends.
- Incorporating additional features (e.g., cholesterol levels, physical activity) can further enhance predictive performance.

These insights can guide healthcare providers in designing targeted interventions, optimizing resource allocation, and improving patient outcomes, ultimately contributing to effective stroke prevention and care strategies.

5. Ethical and Privacy Considerations:

In healthcare analytics, ethical and privacy considerations are crucial to ensure that predictive models are effective while aligning with legal, moral, and social standards. Below are detailed considerations for the stroke prediction model:

1) Patient Privacy and Data Confidentiality:

- **Data Anonymization:** All personal identifiers in the dataset (e.g., name, address, social security number) must be removed or anonymized to protect patient identities. Compliance with regulations like HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation) is essential for handling sensitive health data.
- **Secure Data Storage:** Data should be securely stored using encryption techniques and secure servers to prevent unauthorized access. Access must be restricted to authorized personnel with legitimate reasons for usage.
- **De-Identification Techniques:** Data attributes that could inadvertently identify individuals, such as rare demographic combinations or unique conditions, should be de-identified to maintain confidentiality.

2) Fairness and Bias in Predictions:

- **Addressing Model Bias:** Predictive models may inherit biases from historical data, such as underrepresentation of specific age groups, genders, or socioeconomic classes. Regular audits should be conducted to identify and mitigate such biases to ensure equitable predictions for all patient groups.

- **Avoiding Discrimination:** The model must not disproportionately benefit or harm specific demographic groups. Subgroup analyses should ensure consistent performance across various populations, including age, gender, and socioeconomic groups.
- **Transparency in Decision-Making:** The model should provide interpretable outputs so healthcare providers can understand the rationale behind predictions. Avoid reliance on "black-box" models without explainability, especially in critical healthcare applications.

3) Informed Consent and Ethical Use of Data:

- **Informed Consent from Patients:** Data used for training and predictions should be collected with explicit patient consent. Patients should be informed about how their data will be used, who will access it, and the potential benefits and risks involved.
- **Ethical Use of Predictive Models:** The stroke prediction model should support clinical decision-making rather than replace medical expertise. Predictions must be contextualized by healthcare professionals alongside other clinical factors.

4) Transparency and Accountability:

- **Transparency in Limitations:** The model's limitations, such as its reliance on specific features like age and glucose levels, must be clearly communicated to stakeholders. Predictions should be supplemented with clinical judgment and not blindly trusted.
- **Accountability for Decisions:** Protocols should establish accountability for the model's use in clinical settings. Responsibilities of healthcare providers and data scientists should be clearly defined to ensure ethical and effective application.

5) Ethical Handling of False Positives and False Negatives:

- **Balancing Errors:** In stroke prediction, false negatives (missing a high-risk patient) can have severe consequences, while false positives (misclassifying a low-risk patient) can lead to unnecessary anxiety or interventions. The model should prioritize reducing false negatives while minimizing false positives to avoid overburdening healthcare resources.
- **Impact on Patient Outcomes:** Ethical considerations should ensure that false predictions do not result in harm, such as unnecessary medical procedures or neglect of high-risk patients.

6) Continuous Monitoring and Updates:

- **Performance Monitoring:** Regularly monitor the model's performance to address any drift in accuracy or fairness. Re-training with updated data will ensure the model reflects current population trends and medical advancements.
- **Stakeholder Involvement:** Healthcare professionals, patients, and ethicists should be included in discussions about the model's use, limitations, and updates to ensure its continued ethical application.

By addressing these ethical and privacy considerations, the stroke prediction model can be responsibly implemented to enhance patient outcomes, build trust among stakeholders, and ensure compliance with healthcare standards. These measures are essential to balancing the benefits of predictive analytics with the ethical responsibility to protect patients' rights and well-being.

6. Recommendations:

Based on the results and insights derived from the predictive models, the following recommendations are provided to optimize stroke prevention and healthcare decision-making:

1) Focus on High-Risk Groups:

- **Age-Based Interventions:**
  - Older individuals, particularly those above 60 years, are at a significantly higher risk of stroke. Healthcare providers should prioritize routine screenings and targeted preventive measures for this demographic.
- **High Glucose Monitoring:**
  - Patients with elevated glucose levels (>150 mg/dL) should be closely monitored and provided with lifestyle recommendations and medical interventions to manage blood sugar levels effectively.

2) Enhance Preventive Care Strategies:

- Use the model's predictions to identify patients who are at a higher risk of stroke and provide tailored preventive care. For example:
  - **Nutritional counseling** to manage BMI and glucose levels.
  - **Regular check-ups** and diagnostics for individuals with hypertension or heart disease.

- **Smoking cessation programs** for patients identified as current or former smokers.

3) Utilize the Ensemble Model in Clinical Settings:

- The ensemble model, combining Random Forest and XGBoost, achieved the best performance with an accuracy of 95.47% and ROC-AUC of 99.04%. This model can be integrated into clinical workflows to predict stroke risks with high confidence.
- **Decision Support Tool:**
  - Deploy the model as a decision support tool for healthcare providers to complement their clinical judgment, focusing on high-risk cases flagged by the model.

4) Address Class Imbalance in Future Applications:

- The use of SMOTE in the project demonstrated the importance of addressing class imbalance in the dataset. This technique improved the model's ability to predict stroke cases without compromising overall accuracy.
- Future implementations should ensure balanced datasets to maintain fairness and predictive accuracy, particularly in datasets with rare but critical outcomes like stroke.

5) Leverage Insights for Resource Allocation:

- **Optimize Healthcare Resources:**
  - Insights from the model can guide resource allocation, such as assigning specialized care teams to high-risk patients or planning hospital admissions for potential stroke cases.
- **Preventive Outreach Programs:**
  - Community-based outreach programs can be designed to educate at-risk populations, emphasizing age, glucose level, and lifestyle management as critical factors.

6) Incorporate Additional Features for Improved Accuracy:

- The results indicate that variables such as age, glucose level, and hypertension are strong predictors of stroke. Future iterations of the model can incorporate:
  - **Additional clinical features,** such as cholesterol levels and blood pressure trends.
  - **Behavioral data,** such as physical activity levels, sleep patterns, and stress management.

7) Ethical Usage and Regular Audits:

- Regular audits should be conducted to ensure the model’s predictions are unbiased and equitable across all patient demographics.

- The model should be transparent in its predictions, with clear explanations provided to clinicians about the factors influencing the results.
- Patient privacy must be protected, and sensitive data should be anonymized to comply with regulatory standards such as HIPAA.

8) Continuous Model Improvement:

- Regularly update the model with new data to maintain accuracy and relevance in dynamic healthcare environments.
- Test the model on additional datasets from different populations to validate its generalizability and robustness.

Conclusion:

This project successfully developed a robust predictive model for identifying stroke risk among patients, addressing one of the most pressing healthcare challenges. By utilizing advanced machine learning techniques, the project overcame challenges such as class imbalance and feature selection, achieving exceptional performance metrics. The ensemble model, combining the strengths of Random Forest and XGBoost, emerged as the most effective predictive approach, with an accuracy of 95.47% and an ROC-AUC score of 99.04%. This model demonstrated superior reliability, interpretability, and predictive power, making it a valuable tool for stroke risk assessment.

The insights derived from this project emphasize the critical role of age and glucose levels in predicting stroke risk. Additionally, the use of SMOTE to address class imbalance highlighted the importance of balanced datasets in ensuring fair and accurate predictions, particularly for healthcare scenarios where the minority class often represents critical outcomes. These findings have direct implications for healthcare practices, supporting proactive measures to improve patient outcomes and optimize healthcare resource allocation.

This project not only contributes to the field of healthcare analytics but also underscores the potential of predictive modeling in addressing real-world medical challenges. By integrating these findings into clinical workflows, healthcare providers can enhance preventive care strategies, reduce stroke incidence, and improve overall patient well-being.