In [30]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


#This means that the target variable is STROKE which has only two classes as 0 - No Stroke and 1 -

data = pd.read_csv('stroke-dataset.csv')

# Display the first few and last few rows of the dataset.
data
```

Out[30]:

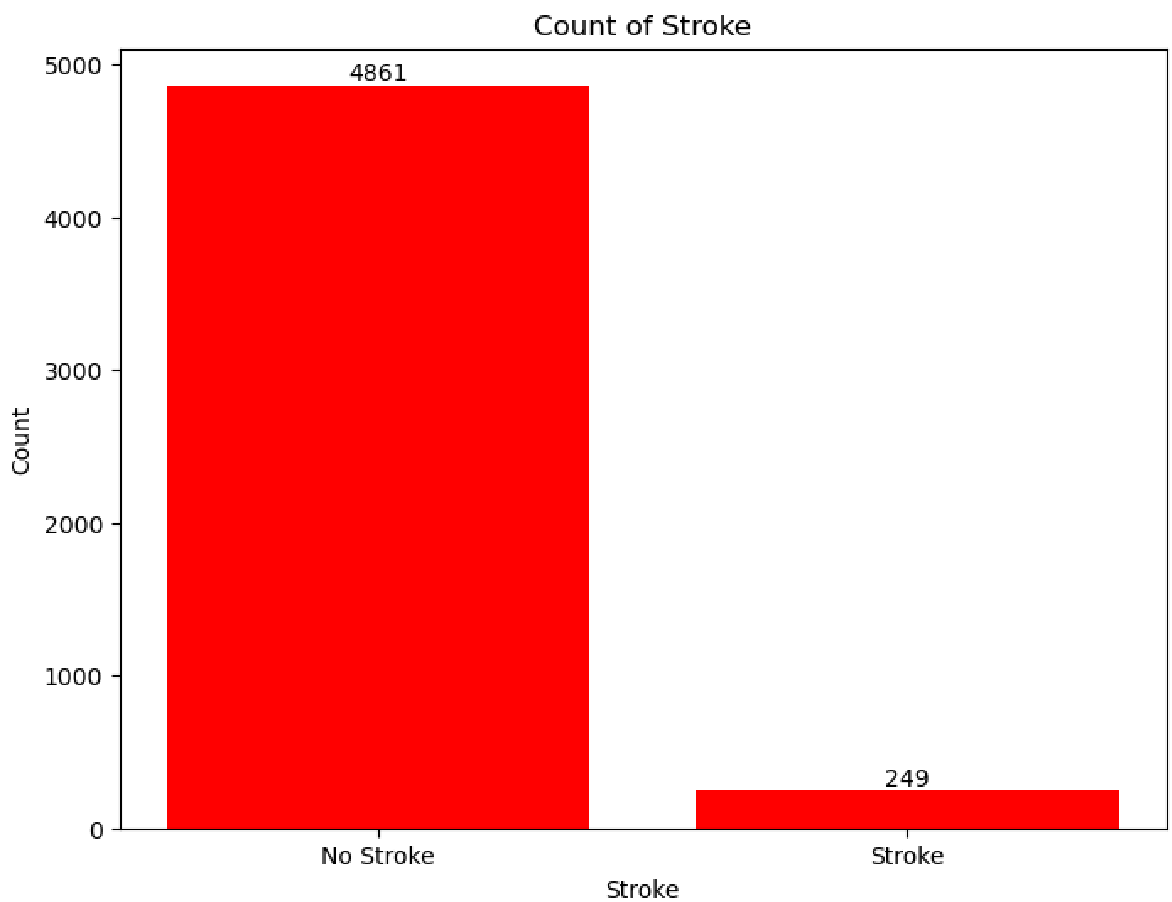
	id	gender	age	hypertension	heart_disease	ever_married	work_type	residence_type	avg_glucose_level	bmi
0	67	Female	17.0	0	0	No	Private	Urban	92.97	NaN
1	77	Female	13.0	0	0	No	children	Rural	85.81	18.6
2	84	Male	55.0	0	0	Yes	Private	Urban	89.17	31.5
3	91	Female	42.0	0	0	No	Private	Urban	98.53	18.5
4	99	Female	31.0	0	0	No	Private	Urban	108.89	52.3
...	...	...	...	...	...	...	...	...	...	...
5105	72911	Female	57.0	1	0	Yes	Private	Rural	129.54	60.9
5106	72914	Female	19.0	0	0	No	Private	Urban	90.57	24.2
5107	72915	Female	45.0	0	0	Yes	Private	Urban	172.33	45.3
5108	72918	Female	53.0	1	0	Yes	Private	Urban	62.55	30.3
5109	72940	Female	2.0	0	0	No	children	Urban	102.92	17.6

5110 rows × 12 columns

In [61]:
```
print(data.shape)

print(data.columns)
print(data.dtypes)
```

```
(5110, 12)
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
id                     int64
gender                object
age                  float64
hypertension           int64
heart_disease          int64
ever_married          object
work_type             object
residence_type        object
avg_glucose_level    float64
bmi                  float64
smoking_status        object
stroke                 int64
dtype: object
```

In [111]:
```
#df.plot(x='stroke', y='age', kind='bar')
import matplotlib.pyplot as plt

stroke_counts = data['stroke'].value_counts()

plt.figure(figsize=(8, 6))
bars = plt.bar(stroke_counts.index, stroke_counts.values, color='red')

plt.xlabel('Stroke')
plt.ylabel('Count')
plt.title('Count of Stroke')

plt.xticks([0, 1], ['No Stroke', 'Stroke'])

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='center', colo

plt.show()
```



In [62]:
```
#Without Solving Class Imbalance problem

X = data[['heart_disease', 'hypertension', 'avg_glucose_level', 'age']]
y = data[ 'stroke']

X_train, X_test, y_train, y_test= train_test_split(X, y, test_size=0.2, random_state=40)

log_reg = make_pipeline(StandardScaler(), LogisticRegression())

log_reg.fit(X_train, y_train)

y_pred = log_reg.predict (X_test)

print('Accuracy:', accuracy_score(y_test, y_pred))
print('Confusion Matrix: \n', confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9579256360078278
Confusion Matrix:
 [[979   0]
 [ 43   0]]
```

In [64]:
```
pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.11.0-py3-none-any.whl (235 kB)
     -------------------------------------- 235.6/235.6 kB 4.8 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\bhave\anaconda3\lib\site-packages
(from imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\bhave\anaconda3\lib\site-packages
(from imbalanced-learn->imblearn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\bhave\anaconda3\lib\site-packages (from
imbalanced-learn->imblearn) (1.24.2)
Collecting joblib>=1.1.1
  Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
     -------------------------------------- 302.2/302.2 kB 9.1 MB/s eta 0:00:00
Requirement already satisfied: scipy>=1.5.0 in c:\users\bhave\anaconda3\lib\site-packages (from i
mbalanced-learn->imblearn) (1.9.1)
Installing collected packages: joblib, imbalanced-learn, imblearn
  Attempting uninstall: joblib
    Found existing installation: joblib 1.1.0
    Uninstalling joblib-1.1.0:
      Successfully uninstalled joblib-1.1.0
Successfully installed imbalanced-learn-0.11.0 imblearn-0.0 joblib-1.3.2
Note: you may need to restart the kernel to use updated packages.
```

In [76]:
```
missing_values = df.isna().sum()
print(missing_values)
df = df.dropna()
df.count()
```

```
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                  0
smoking_status       0
stroke               0
dtype: int64
```

Out[76]:
```
id                   4909
gender               4909
age                  4909
hypertension         4909
heart_disease        4909
ever_married         4909
work_type            4909
residence_type       4909
avg_glucose_level    4909
bmi                  4909
smoking_status       4909
stroke               4909
dtype: int64
```

To Resolve the issue of “Class Imbalance” I prformed - Synthetic Minority Over-sampling Technique (SMOTE) to create synthetic samples for the minority class. SMOTE generates new samples that are combinations of existing ones, helping balance the class distribution.

In [92]:
```
from imblearn.over_sampling import SMOTE

df_encoded = pd.get_dummies(df, columns=['gender', 'ever_married', 'work_type', 'residence_type',

X = df_encoded.drop('stroke', axis=1)
y = df_encoded['stroke']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=43)

# Apply SMOTE to balance classes
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

model = LogisticRegression()
model.fit(X_resampled, y_resampled)

y_pred = model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

print(f'Accuracy: {accuracy}\n')

print(f'Classification Report:\n{report}')
```

```
Confusion Matrix:
[[886  53]
 [ 27  16]]
Accuracy: 0.9185336048879837

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.96       939
           1       0.23      0.37      0.29        43

    accuracy                           0.92       982
   macro avg       0.60      0.66      0.62       982
weighted avg       0.94      0.92      0.93       982


C:\Users\bhave\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarn
ing: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

```
Confusion Matrix

True Negatives (TN): 886
False Positives (FP): 53
False Negatives (FN): 27
```

```
True Positives (TP): 16
```

In [108]:

```
from imblearn.over_sampling import SMOTE

df_encoded = pd.get_dummies(df, columns=['gender', 'ever_married', 'work_type', 'residence_type',
df_encoded = pd.concat([df_encoded, df[['age', 'hypertension', 'heart_disease', 'avg_glucose_level

X = df_encoded.drop('stroke', axis=1)
y = df_encoded['stroke']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=43)

# Apply SMOTE to balance classes
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

model = LogisticRegression()
model.fit(X_resampled, y_resampled)

y_pred = model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

print(f'Accuracy: {accuracy}\n')

print(f'Classification Report:\n{report}')
```

```
Confusion Matrix:
[[1322   84]
 [  50   17]]
Accuracy: 0.9090291921249152

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1406
           1       0.17      0.25      0.20        67

    accuracy                           0.91      1473
   macro avg       0.57      0.60      0.58      1473
weighted avg       0.93      0.91      0.92      1473


C:\Users\bhave\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarn
ing: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

```
Confustion Matrix:

True Negatives (TN): 1322
False Positives (FP): 84
```

```
False Negatives (FN): 50
True Positives (TP): 17

The model is performing well in correctly identifying cases of "No Stroke", as indicated by the
high number of True Negatives (1322) and the relatively low number of False Positives (84).

However, the model is struggling with identifying cases of "Stroke", as indicated by the low
number of True Positives (17) and the moderate number of False Negatives (50). So there is room
for improvement in detecting strokes.
```