Brain Stroke Prediction

Importing Libraries

In [2]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt #creating static, interactive, and animated visualizations in Python
import seaborn as sns #data visualizations
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.utils.validation import check_is_fitted

import warnings
warnings.filterwarnings('ignore')
```

Viewing the data in pandas

In [3]:

```
#The dataframe is read from the csv file
df=pd.read_csv('brain_stroke.csv')
df.head(10)
```

Out[3]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucos
0	Male	67.0	0	1	Yes	Private	Urban	
1	Male	80.0	0	1	Yes	Private	Rural	
2	Female	49.0	0	0	Yes	Private	Urban	
3	Female	79.0	1	0	Yes	Self-employed	Rural	
4	Male	81.0	0	0	Yes	Private	Urban	
5	Male	74.0	1	1	Yes	Private	Rural	
6	Female	69.0	0	0	No	Private	Urban	
7	Female	78.0	0	0	Yes	Private	Urban	
8	Female	81.0	1	0	Yes	Private	Rural	
9	Female	61.0	0	1	Yes	Govt_job	Rural	

In [4]:

```
df.tail(5)
```

Out[4]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_gl
4976	Male	41.0	0	0	No	Private	Rural	
4977	Male	40.0	0	0	Yes	Private	Urban	
4978	Female	45.0	1	0	Yes	Govt_job	Rural	
4979	Male	40.0	0	0	Yes	Private	Rural	
4980	Female	80.0	1	0	Yes	Private	Urban	

In [5]:

```
#Count of number of records and columns
df.shape
```

Out[5]:

```
(4981, 11)
```

In [6]:

```
# Provides the data type of all attributes and the number of NOT NULL values count is obtained
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4981 non-null   object 
 1   age                4981 non-null   float64
 2   hypertension       4981 non-null   int64  
 3   heart_disease      4981 non-null   int64  
 4   ever_married       4981 non-null   object 
 5   work_type          4981 non-null   object 
 6   Residence_type     4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi                4981 non-null   float64
 9   smoking_status     4981 non-null   object 
 10  stroke             4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

In [7]:

```
df.describe()
```

Out[7]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000
mean	43.419859	0.096165	0.055210	105.943562	28.498173	0.049789
std	22.662755	0.294848	0.228412	45.075373	6.790464	0.217531
min	0.080000	0.000000	0.000000	55.120000	14.000000	0.000000
25%	25.000000	0.000000	0.000000	77.230000	23.700000	0.000000
50%	45.000000	0.000000	0.000000	91.850000	28.100000	0.000000
75%	61.000000	0.000000	0.000000	113.860000	32.600000	0.000000
max	82.000000	1.000000	1.000000	271.740000	48.900000	1.000000

In [8]:

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

Out[8]:

```
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
```

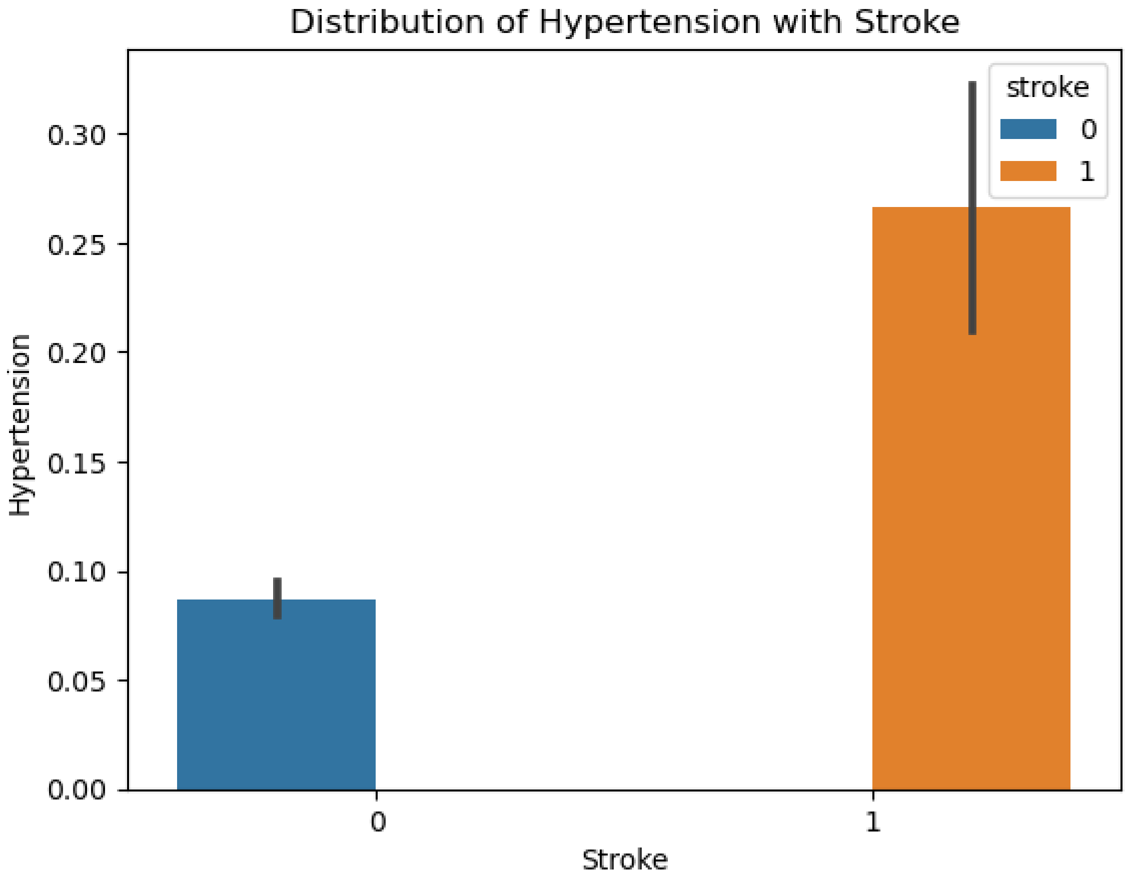
In [9]:

```
# Checking the values in the stroke column
print(df["stroke"].value_counts())
```

```
0    4733
1     248
Name: stroke, dtype: int64
```

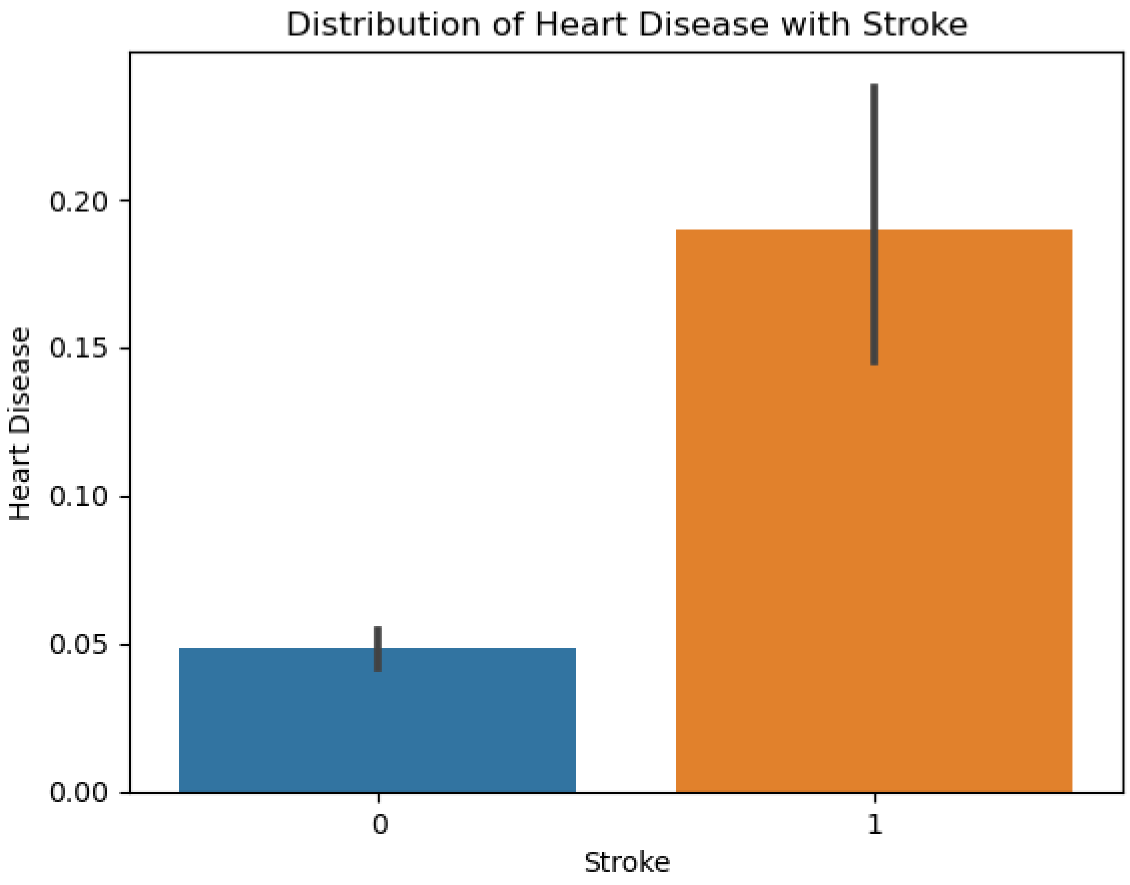
In [11]:

```
sns.barplot(x="stroke", y='hypertension', data=df, hue="stroke")
#label axis
plt.xlabel("Stroke")
plt.ylabel("Hypertension")
plt.title("Distribution of Hypertension with Stroke");
```



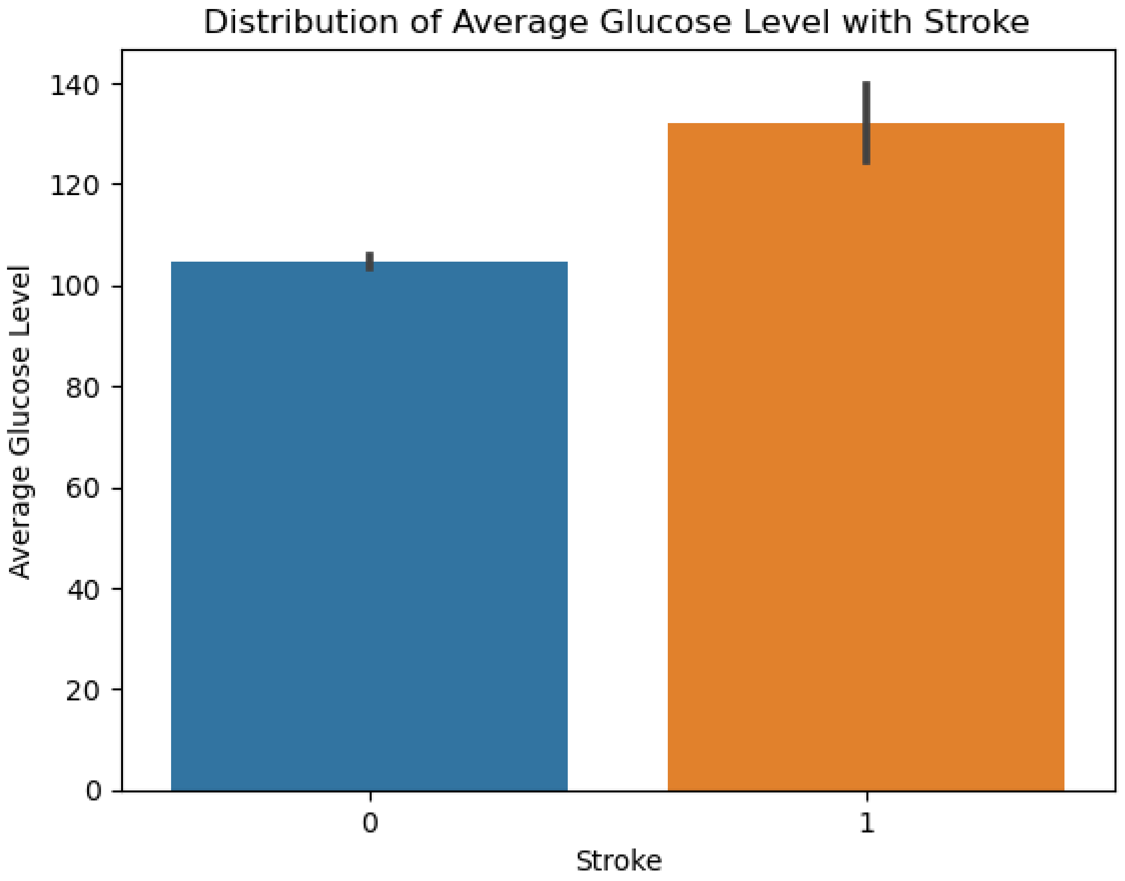
In [12]:

```
sns.barplot(x="stroke", y='heart_disease', data=df)
#label axis
plt.xlabel("Stroke")
plt.ylabel("Heart Disease")
plt.title("Distribution of Heart Disease with Stroke");
```



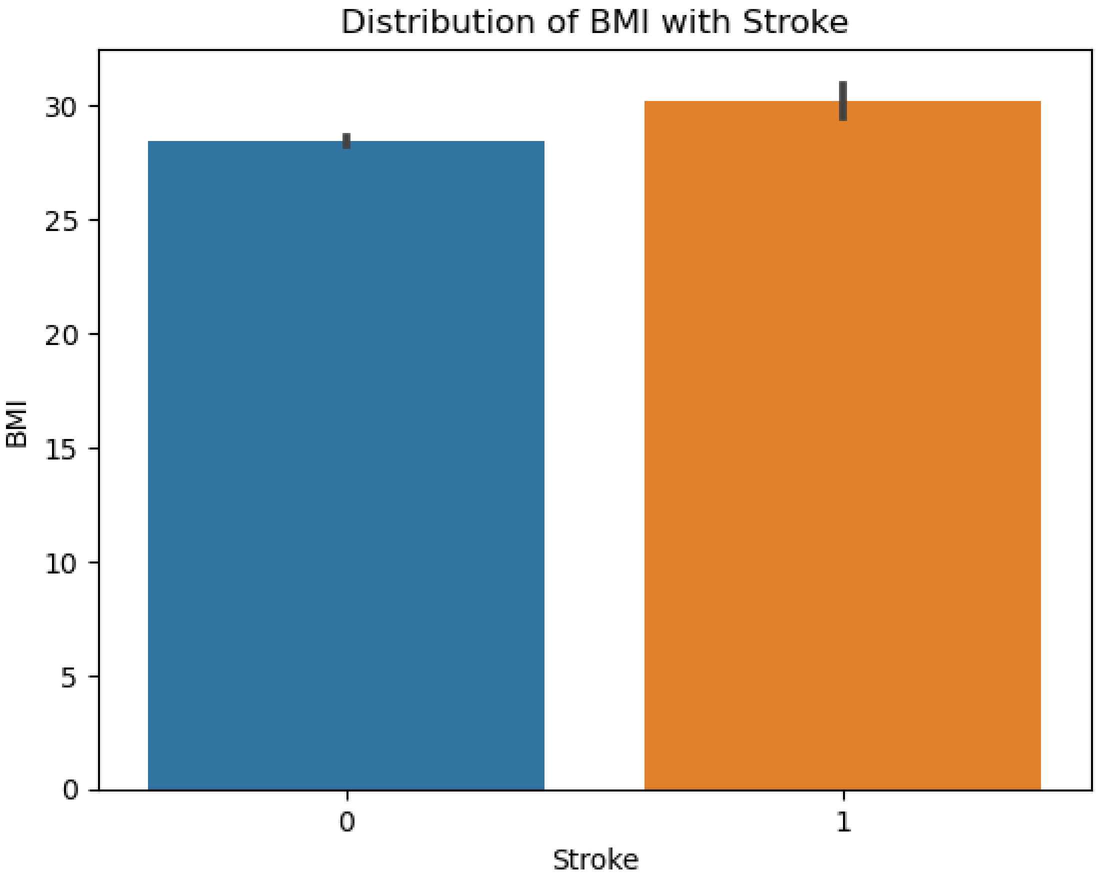
In [13]:

```
sns.barplot(x="stroke", y='avg_glucose_level', data=df)
#label axis
plt.xlabel("Stroke")
plt.ylabel("Average Glucose Level")
plt.title("Distribution of Average Glucose Level with Stroke");
```



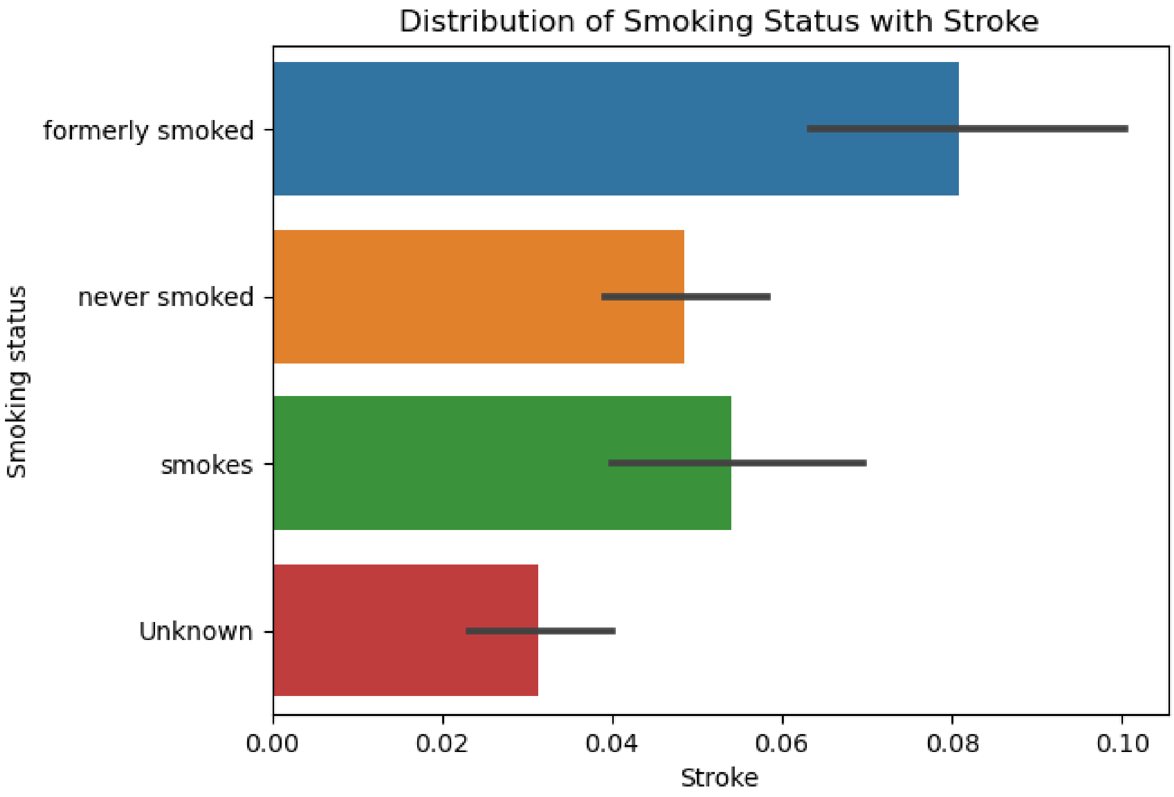
In [14]:

```
sns.barplot(x="stroke", y='bmi', data=df)
#label axis
plt.xlabel("Stroke")
plt.ylabel("BMI")
plt.title("Distribution of BMI with Stroke");
```



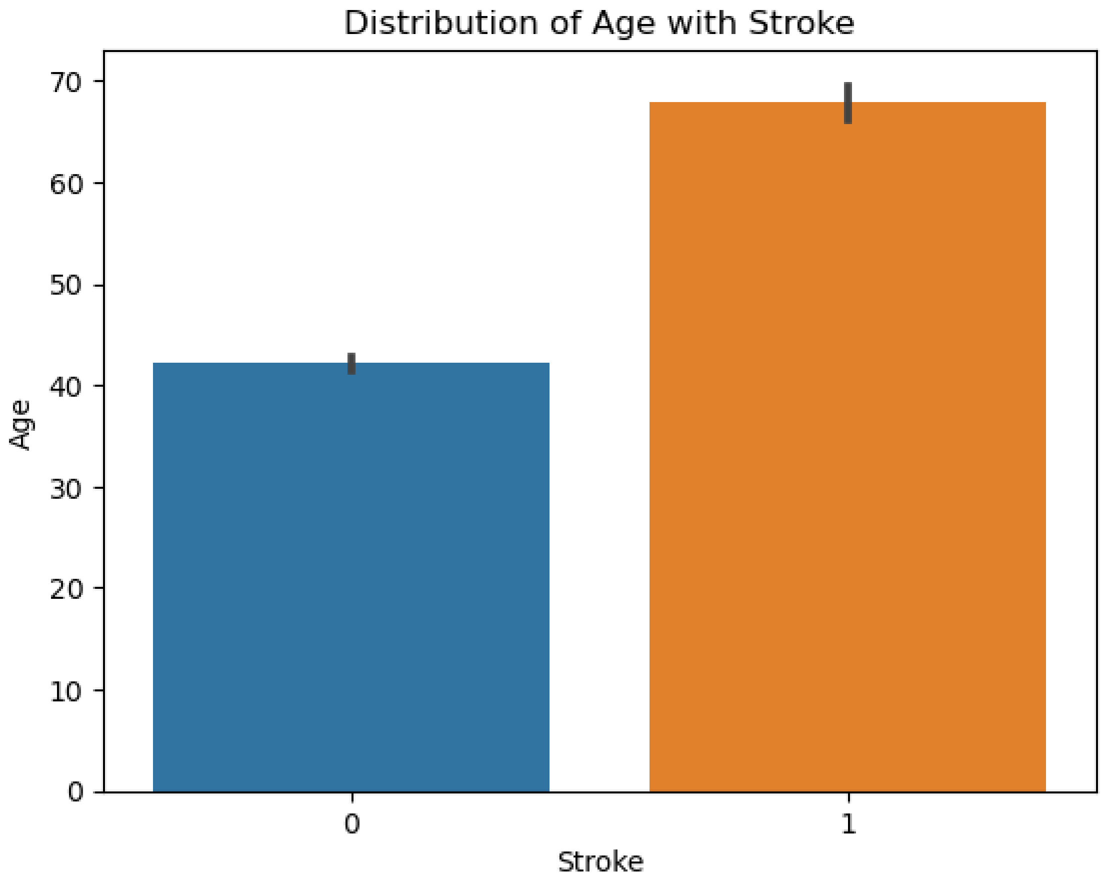
In [15]:

```
sns.barplot(x="stroke", y='smoking_status', data=df)
#label axis
plt.xlabel("Stroke")
plt.ylabel("Smoking status")
plt.title("Distribution of Smoking Status with Stroke");
```



In [16]:

```
sns.barplot(x="stroke", y='age', data=df)
#label axis
plt.xlabel("Stroke")
plt.ylabel("Age")
plt.title("Distribution of Age with Stroke");
```



In [17]:

```
print(df["smoking_status"].value_counts())
```

```
never smoked       1838
Unknown            1500
formerly smoked     867
smokes              776
Name: smoking_status, dtype: int64
```

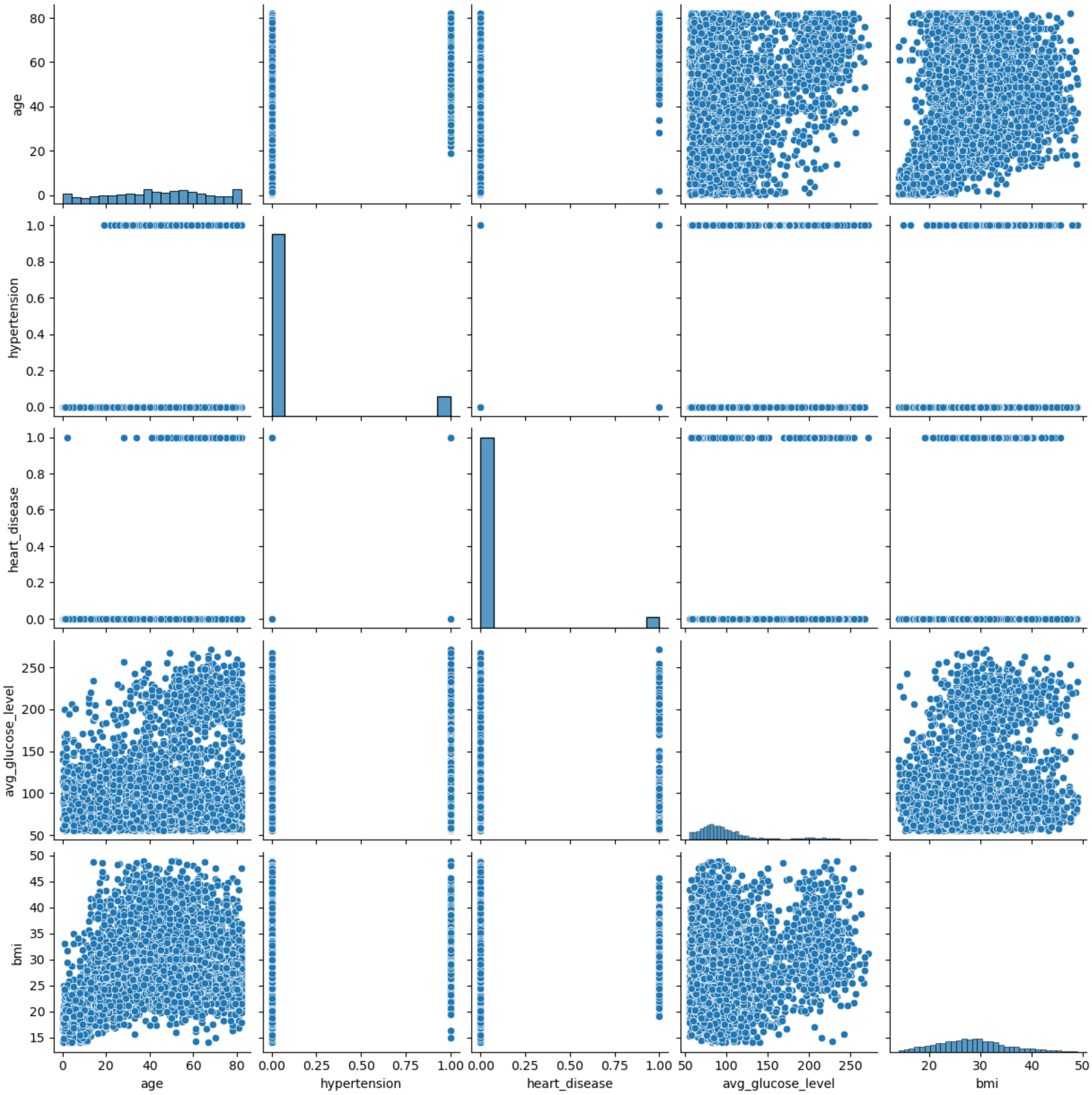
In [18]:
```
print(df["heart_disease"].value_counts())
```

```
0    4706
1     275
Name: heart_disease, dtype: int64
```

In [19]:
```
# Exluding the object datatype in the dataset
df = df.select_dtypes(exclude=['object'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   age                4981 non-null   float64
 1   hypertension       4981 non-null   int64  
 2   heart_disease      4981 non-null   int64  
 3   avg_glucose_level  4981 non-null   float64
 4   bmi                4981 non-null   float64
 5   stroke             4981 non-null   int64  
dtypes: float64(3), int64(3)
memory usage: 233.6 KB
```

In [20]:
```
y = df["stroke"].values
x_data = df.drop(["stroke"], axis = 1)
sns.pairplot(x_data)
plt.show()
```



Data Normalization

In [21]:

```
x = (x_data - np.min(x_data))/(np.max(x_data) - np.min(x_data)).values
x.head()
```

Out[21]:

	age	hypertension	heart_disease	avg_glucose_level	bmi
0	0.816895	0.0	1.0	0.801265	0.647564
1	0.975586	0.0	1.0	0.234512	0.530086
2	0.597168	0.0	0.0	0.536008	0.584527
3	0.963379	1.0	0.0	0.549349	0.286533
4	0.987793	0.0	0.0	0.605161	0.429799

Correlation Matrix

In [22]:

```
df.corr() #calculates the correlation matrix between the numerical columns
```

Out[22]:

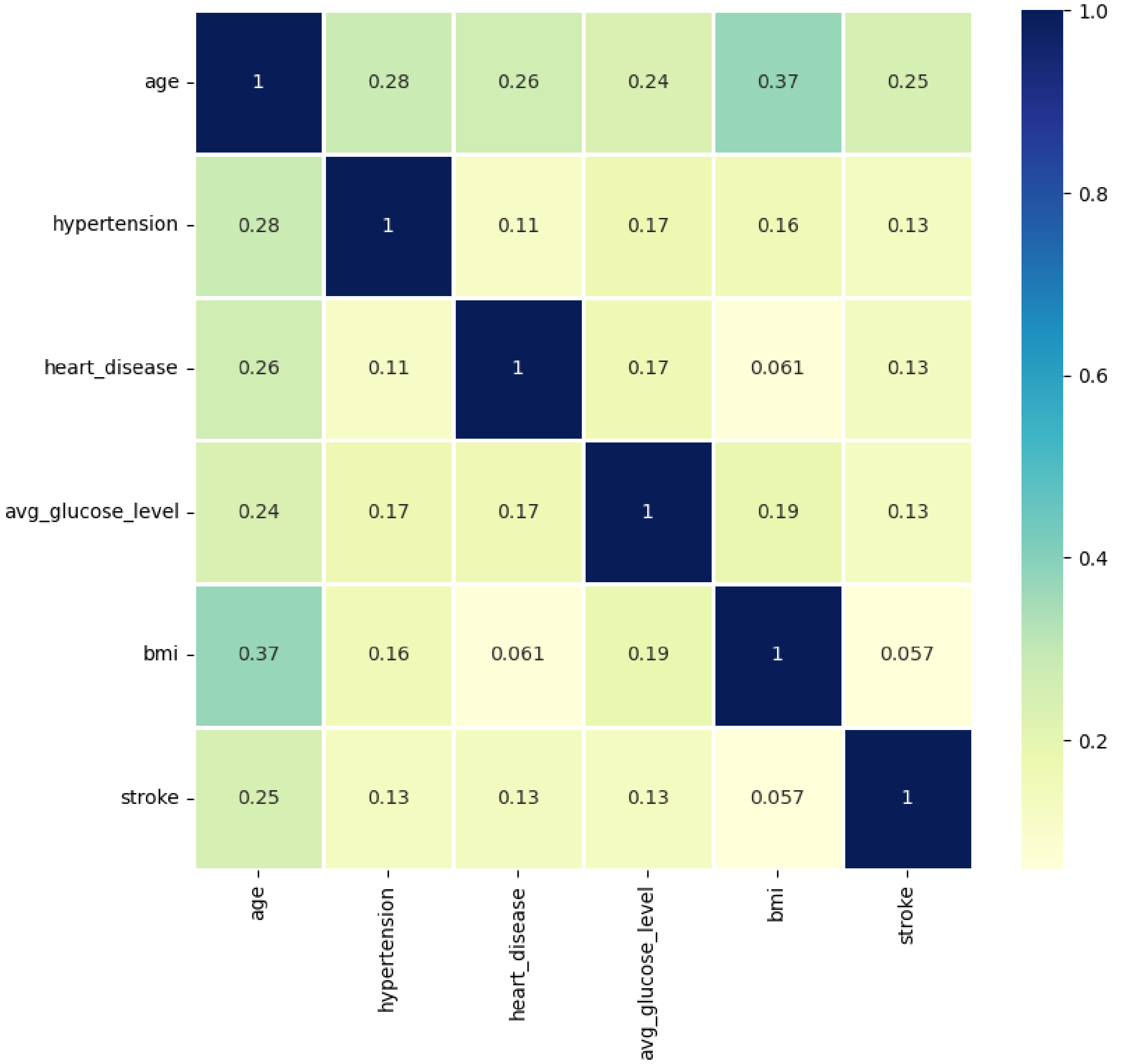
	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
age	1.000000	0.278120	0.264852	0.236763	0.373703	0.246478
hypertension	0.278120	1.000000	0.111974	0.170028	0.158762	0.131965
heart_disease	0.264852	0.111974	1.000000	0.166847	0.060926	0.134610
avg_glucose_level	0.236763	0.170028	0.166847	1.000000	0.186348	0.133227
bmi	0.373703	0.158762	0.060926	0.186348	1.000000	0.056926
stroke	0.246478	0.131965	0.134610	0.133227	0.056926	1.000000

In [23]:

```
# Correlation matrix between the attributes in the dataset to find if any attributes are correlated
corrmat=df.corr()
f,ax=plt.subplots(figsize=(9,8))
sns.heatmap(corrmat,ax=ax,cmap="YlGnBu",linewidth=0.8,annot=True)
```

Out[23]:

```
<AxesSubplot:>
```



Spliting the dataset

In [24]:

```
target = "stroke"
X = df.drop(columns=target)
y = df[target]
```

20% test set & 80% train set

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (3984, 5)
y_train shape: (3984,)
X_test shape: (997, 5)
y_test shape: (997,)
```

In [26]:
```
acc_baseline = y_train.value_counts(normalize=True).max()
print("Baseline Accuracy:", round(acc_baseline, 2))
```

```
Baseline Accuracy: 0.95
```

Building Model

In [27]:
```
# Build model using Logistic Regression
model = LogisticRegression(max_iter=1000)

# Fit model to training data
model.fit(X_train, y_train)
```

Out[27]:
```
LogisticRegression(max_iter=1000)
```

In [28]:
```
# Predicting the result
y_pred=model.predict(X_test)
```

In [30]:
```
model.score(X_test, y_test)
```

Out[30]:
```
0.9458375125376128
```

In [31]:
```
# Confusion Matrix
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,plot_confusion_matr
ac = accuracy_score(y_test,y_pred)*100
cr= classification_report(y_test,y_pred)
print("accuracy score : ",ac)
print("classification report :",cr)
```

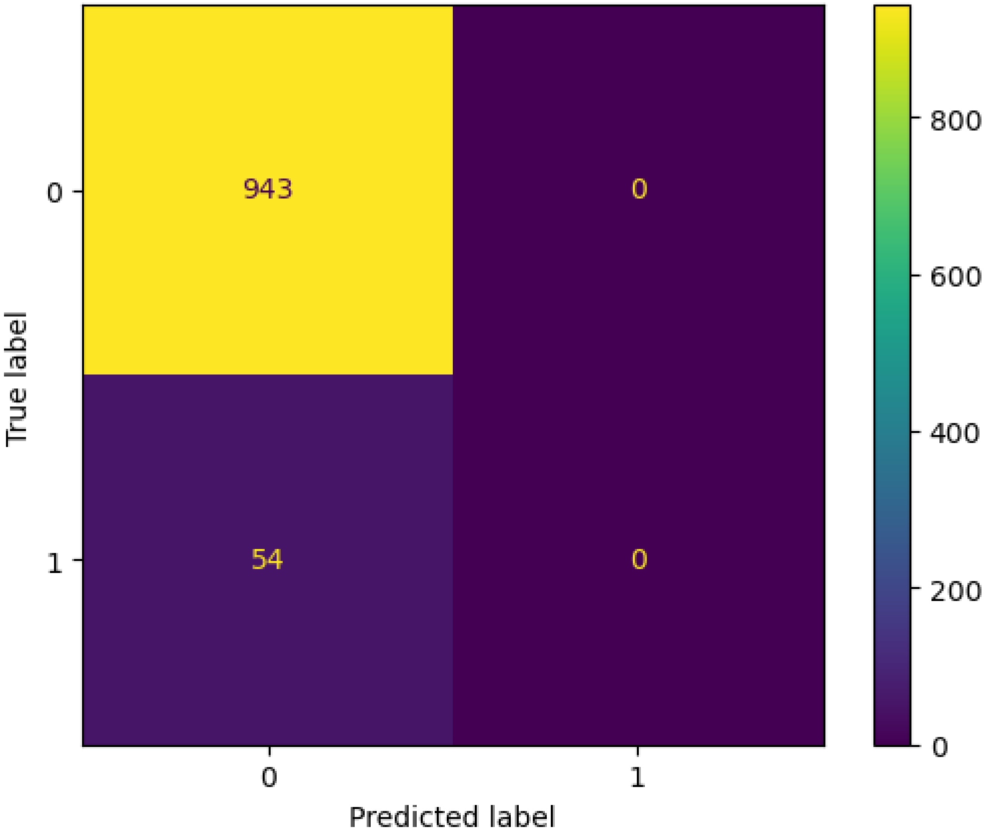
```
accuracy score :  94.58375125376128
classification report :               precision    recall  f1-score   support

           0       0.95      1.00      0.97       943
           1       0.00      0.00      0.00        54

    accuracy                           0.95       997
   macro avg       0.47      0.50      0.49       997
weighted avg       0.89      0.95      0.92       997
```

In [32]:
```
plot_confusion_matrix(model,X_test,y_test)
```

Out[32]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1e212924b20>
```



```
In [ ]:
```