BRAIN STROKE ANALYSIS

Understanding the factors impacting this potentially fatal medical disease is crucial, which is why brain stroke analysis and prediction are important. The dataset, which includes vital characteristics like age, gender, lifestyle, and health indicators, offers insightful information about the intricate interactions between risk factors. Our goal is to use machine learning techniques to find the patterns and connections that lead to stroke incidence so that tailored preventive interventions may be implemented early on. factors. Our goal is to use machine learning techniques to find the patterns and connections that lead to stroke incidence so that tailored preventive interventions may be implemented early on.

Context:

Attribute Information:

1. Age: age of the patient [years]
2. Gender: Sex of the patient [Male, Female]
3. Hypertension: To identify If the patient suffered from hypertension or not [1: yes, 0: no]
4. Heart_disease: To identify If the patient suffered from heart_disease or not [1: yes, 0: no]
5. Ever_married: To identify If the patient is married or not [yes,no]
6. Work_type: To identify which type of job [private, self_employed, govt_job]
7. Residence_type: To identify which type of residence [rural,urban]
8. Avg_glucose_level: To determine glucose level [mmol/L]
9. Bmi: To determine body mass index[kg/m2]
10. Smoking Status: To identify that person smokes or not
11. Stroke: To identify if patient have stroke or not [1: yes, 0:no]

DATA READING FROM CSV FILE & DATA PREPROCESSING

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import  seaborn as sns
%matplotlib inline
```

[3]:
```
df = pd.read_csv('brain_stroke.csv')
df.head()
```

[3]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1

[5]:
```
df.shape
```

[5]:
```
(4981, 11)
```

[6]:
```
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
```

```
 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
```

[7]:
```
df.describe()
```

[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

'

ANALYSIS OF THE DATA

BAR PLOT

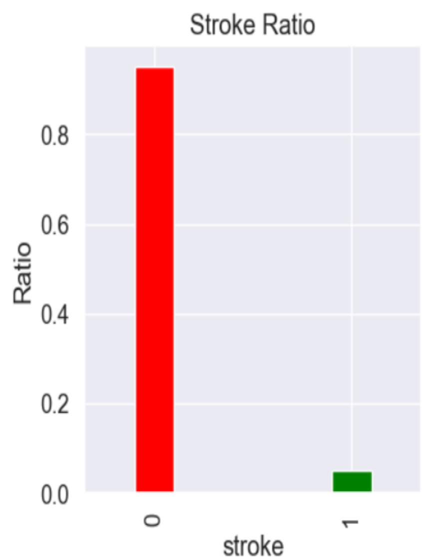
In [7]:
```
#Checking Null values in the dataset
df.isna().sum()
```

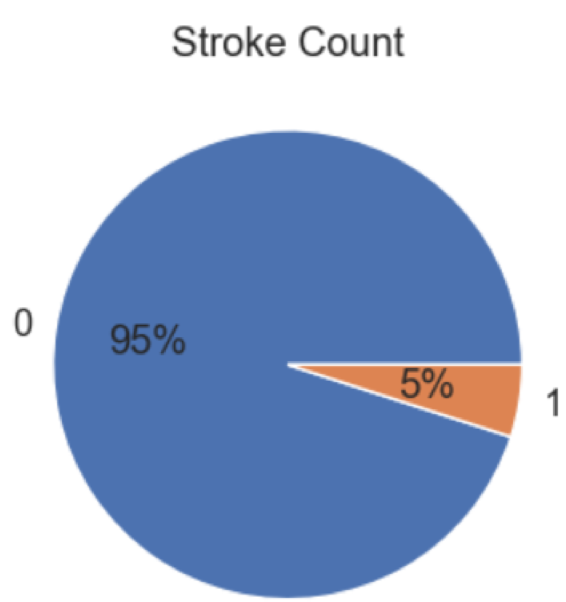
Out[7]:
```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

In [8]:
```
df.columns
```

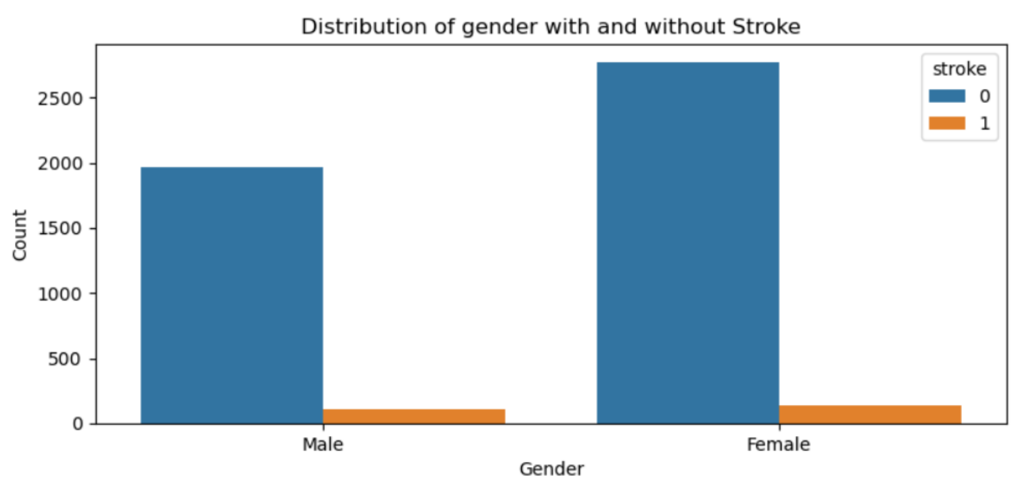
Out[8]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

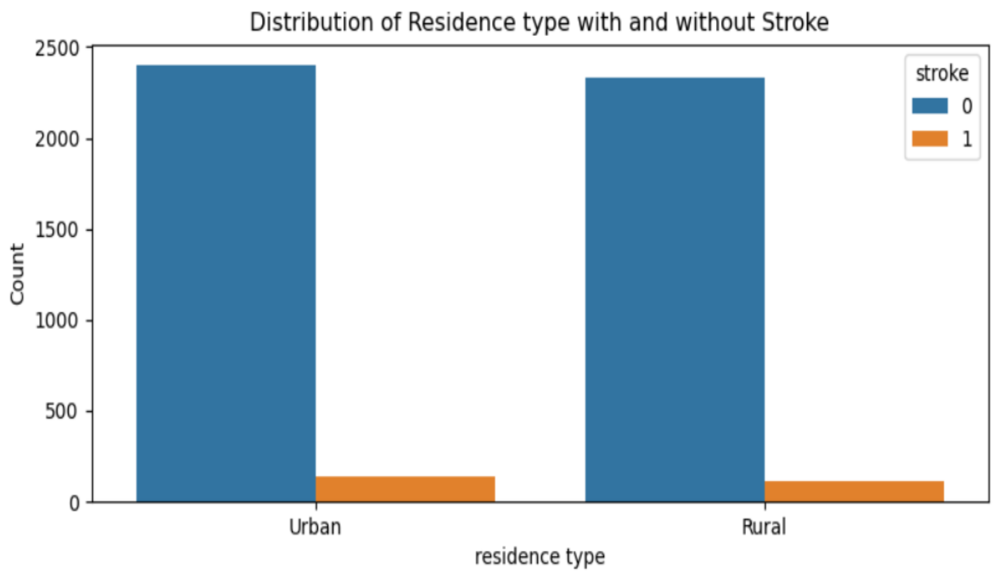
In [64]:
```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(4, 8))

df['stroke'].value_counts(normalize=True).plot.bar(width=0.2, color=('red', 'green'), ax=ax1)
ax1.set_ylabel('Ratio')
ax1.set_title('Stroke Ratio')

stroke_data = df['stroke'].value_counts()
ax2.pie(stroke_data, labels=stroke_data.index, autopct='%.0f%%')
ax2.set_title('Stroke Count')

plt.tight_layout()
plt.show()
```

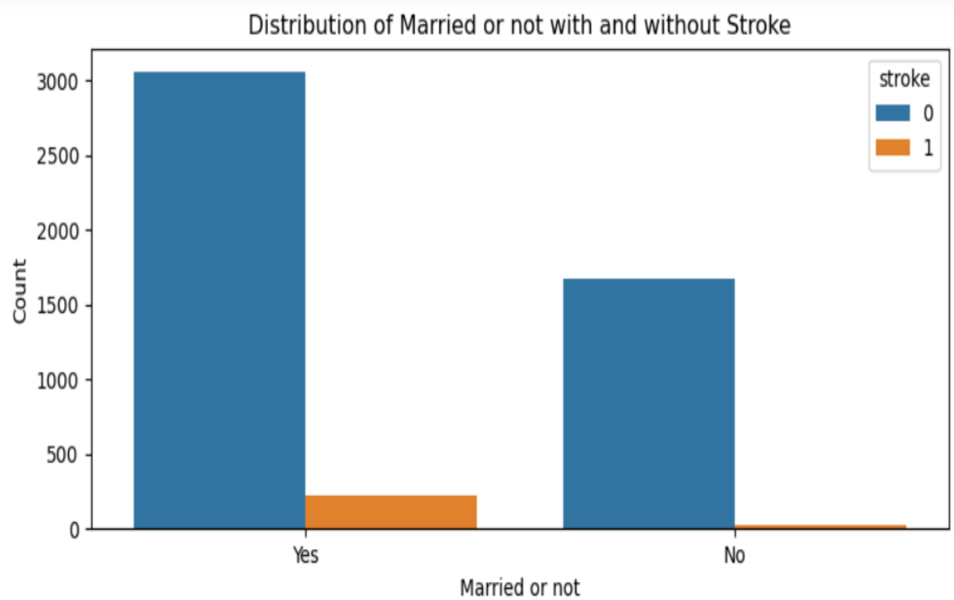


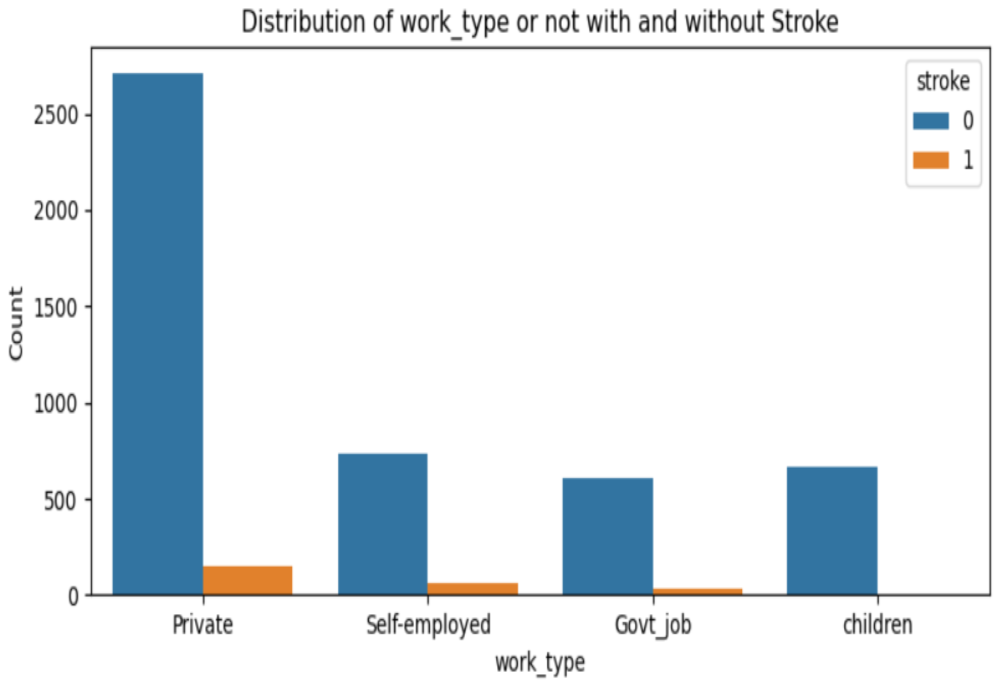


In [26]:
```
plt.figure(figsize=(8, 15))

plt.subplot(5,1,1)
sns.countplot(x='gender', hue='stroke', data=df)
plt.title('Distribution of gender with and without Stroke')
plt.xlabel('Gender')
plt.ylabel('Count')

plt.subplot(5,1,2)
sns.countplot(x='Residence_type', hue='stroke', data=df)
plt.title('Distribution of Residence type with and without Stroke')
plt.xlabel('residence type')
plt.ylabel('Count')

plt.subplot(5,1,3)
sns.countplot(x='ever_married', hue='stroke', data=df)
plt.title('Distribution of Married or not with and without Stroke')
plt.xlabel('Married or not')
plt.ylabel('Count')

plt.subplot(5,1,4)
sns.countplot(x='work_type', hue='stroke', data=df)
plt.title('Distribution of work_type or not with and without Stroke')
plt.xlabel('work_type')
plt.ylabel('Count')

plt.subplot(5,1,5)
sns.countplot(x='smoking_status', hue='stroke', data=df)
plt.title('Distribution of smoking or not with and without Stroke')
plt.xlabel('smoking')
plt.ylabel('Count')

plt.tight_layout()

plt.show()
```

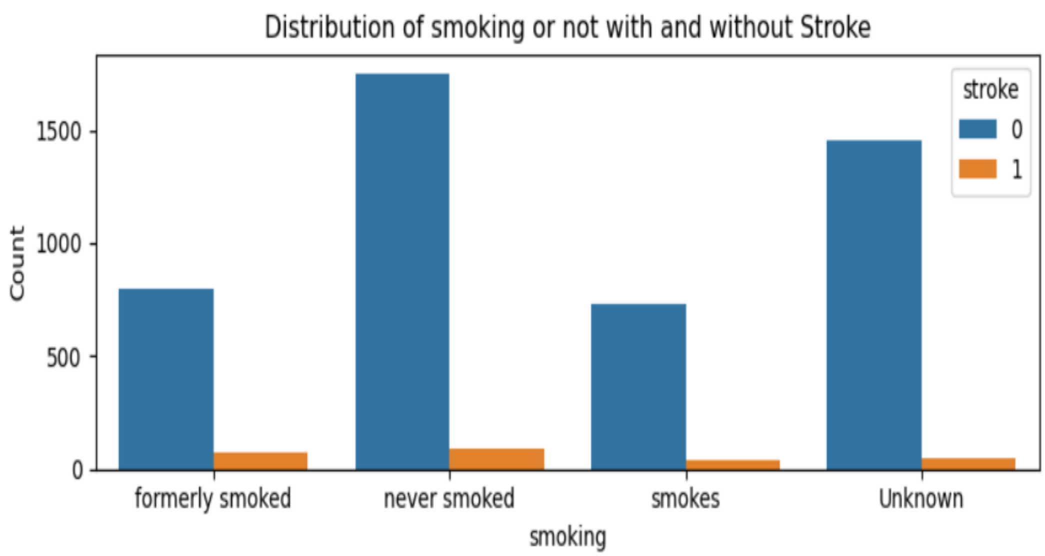



Distribution of Residence type with and without Stroke
stroke
0
1
Count
2500
2000
1500
1000
500
0
Urban
Rural
residence type


Distribution of Married or not with and without Stroke
stroke
0
1
Count
3000
2500
2000
1500
1000
500
0
Yes
No
Married or not

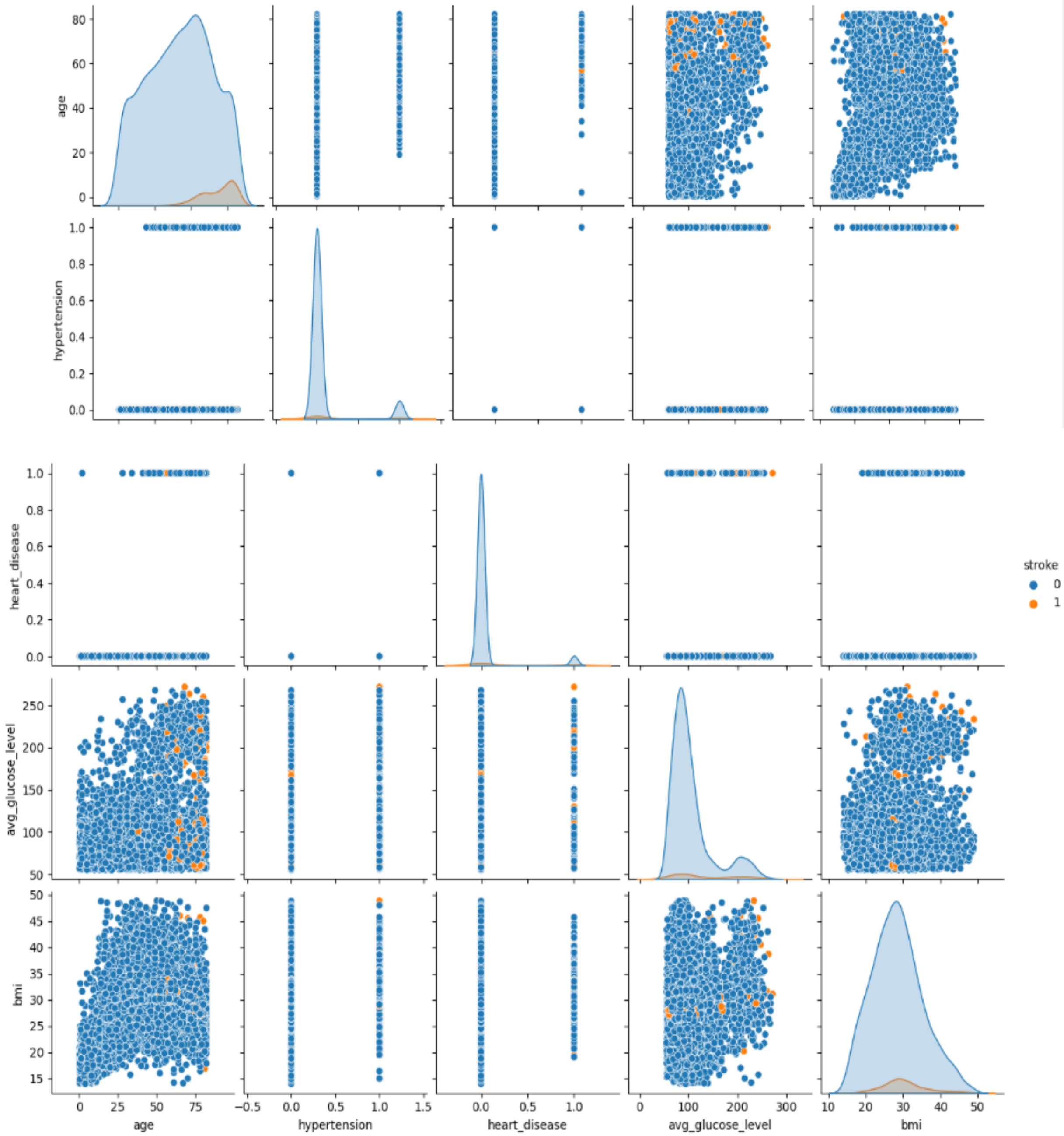

Distribution of work_type or not with and without Stroke
Count
2500
2000
1500
1000
500
0
stroke
0
1
Private
Self-employed
Govt_job
children
work_type

work_type


Distribution of smoking or not with and without Stroke
Count
1500
1000
500
0
stroke
0
1
formerly smoked
never smoked
smokes
Unknown
smoking

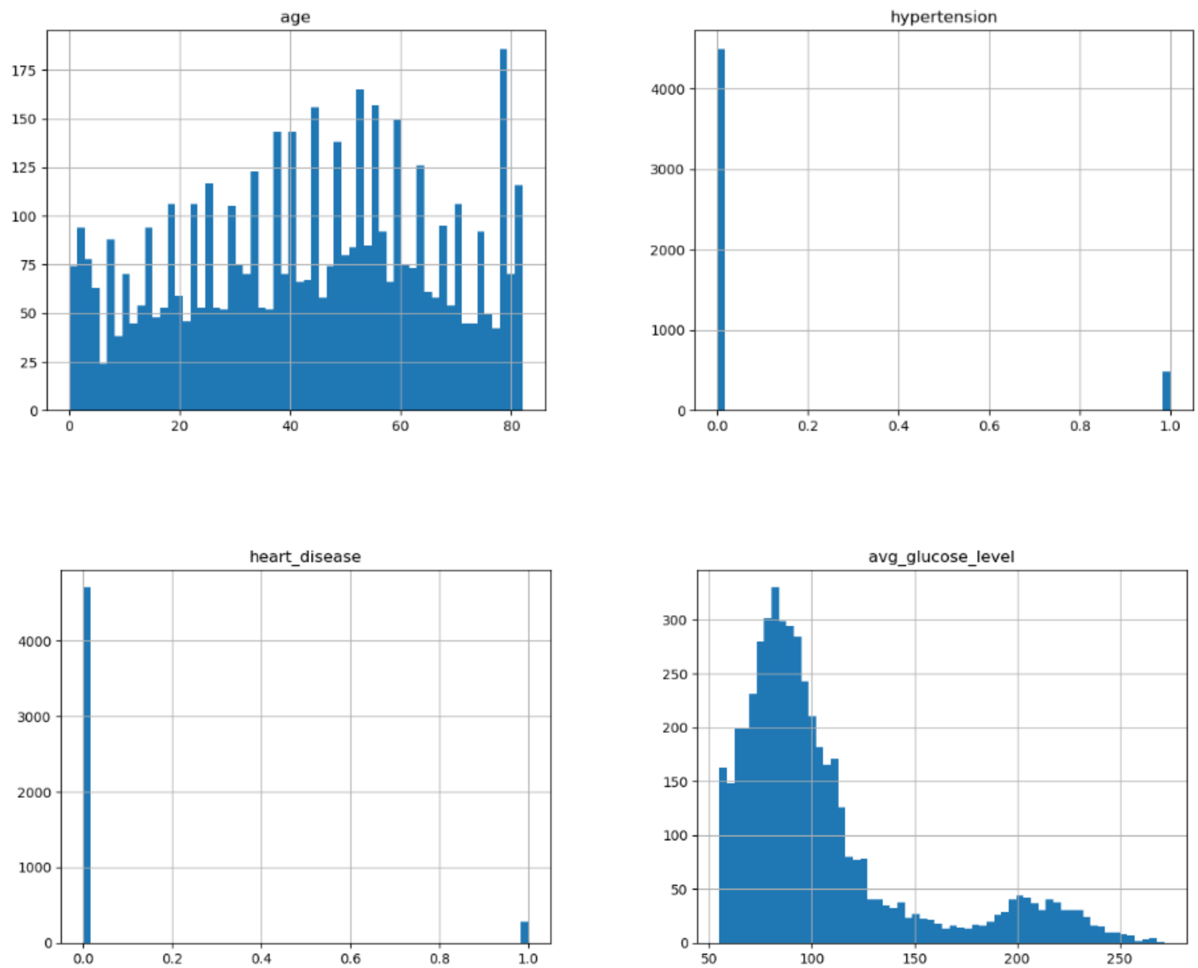
PAIR PLOT

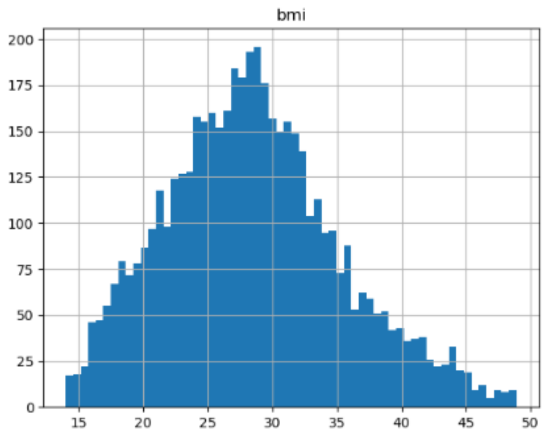
```
[12]: sns.pairplot(df, hue='stroke')

D:\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight

  self._figure.tight_layout(*args, **kwargs)

[12]: <seaborn.axisgrid.PairGrid at 0x14919d4bf50>
```

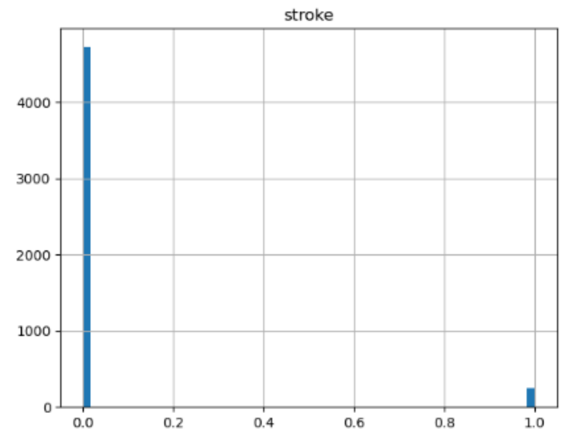


HISTOGRAM

[13]:
```
df.hist(figsize=(15,18),bins=60)

plt.subtitle('Features Distribution', x=0.5,y=1.02,ha='center',fontsize='large')

plt.tight_layout()
```

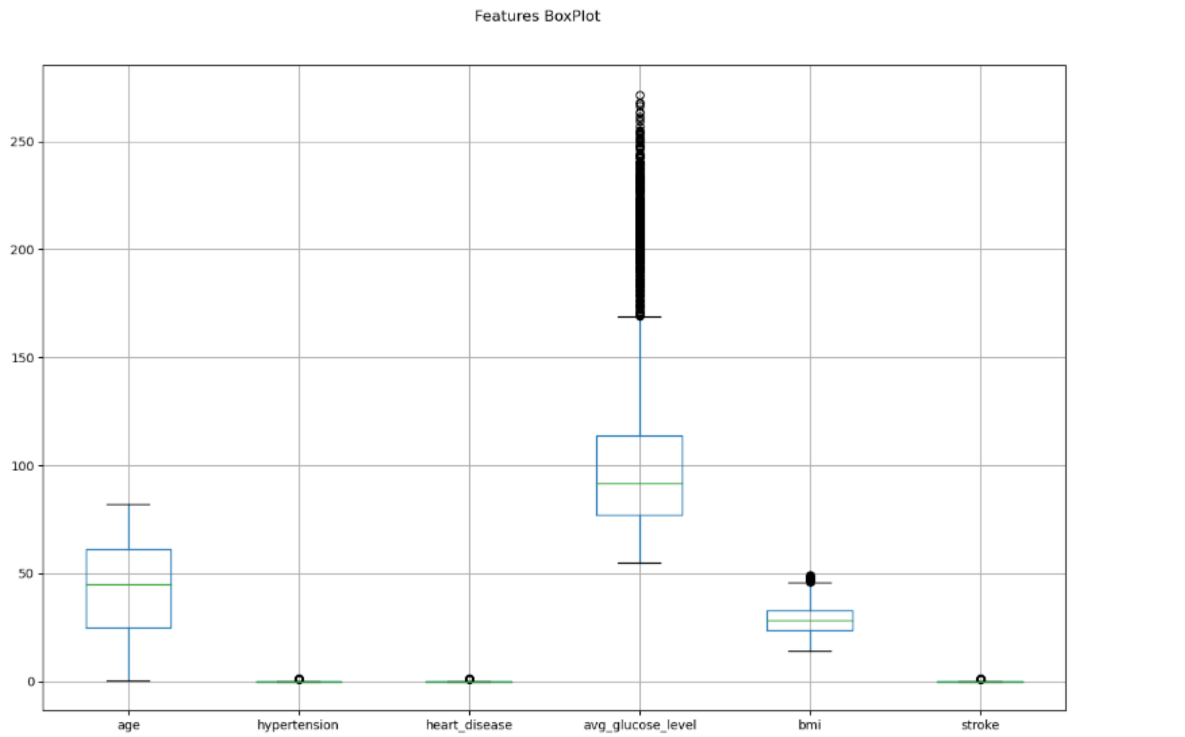



bmi
200
175
150
125
100
75
50
25
0
15
20
25
30
35
40
45
50


stroke
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

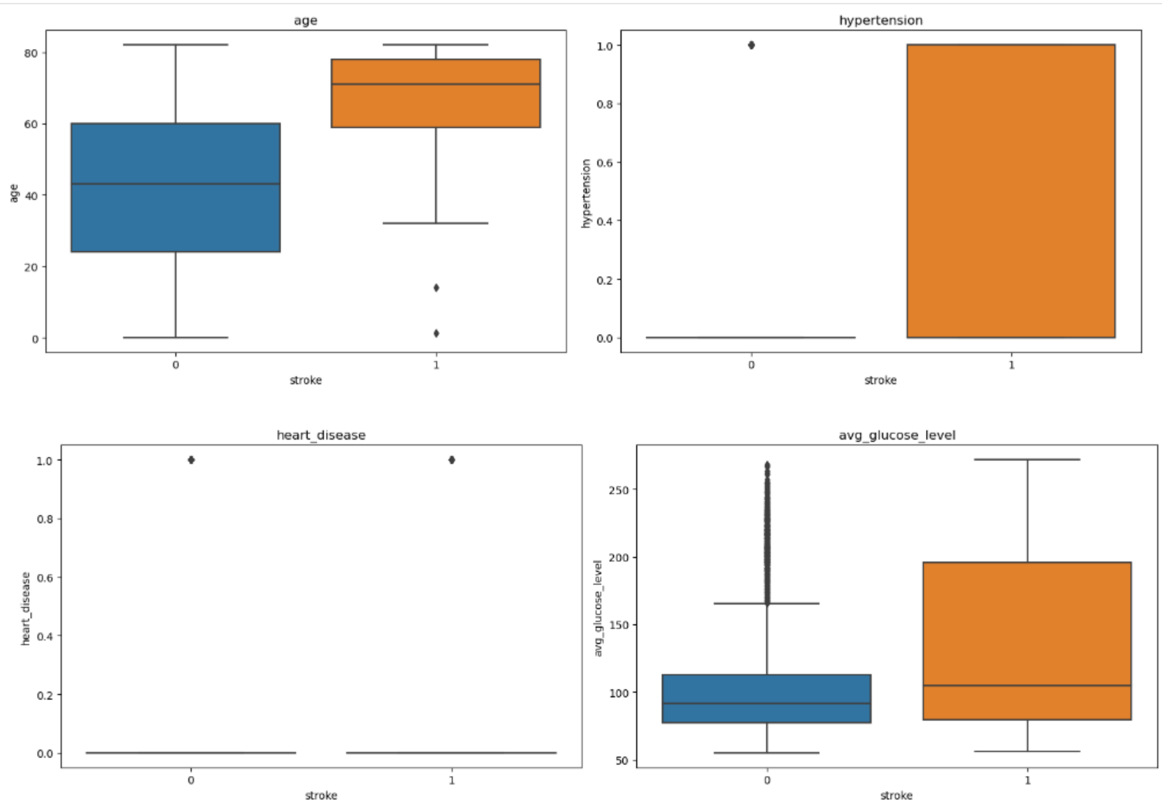
BOXPLOT

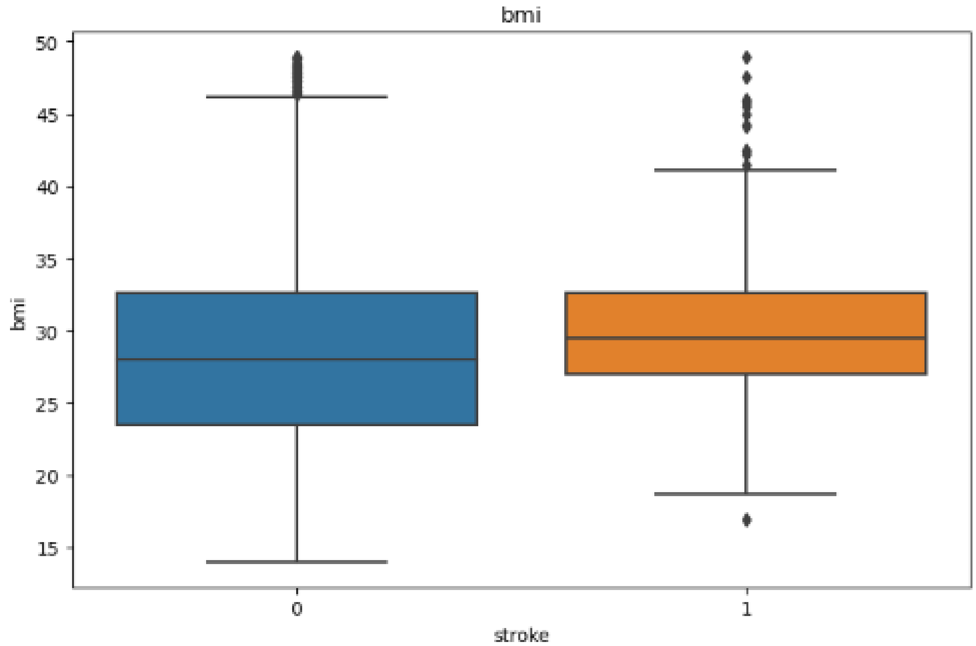
[14]:
```
df.boxplot(figsize=(12,8))
plt.suptitle('Features BoxPlot', x=0.5, y=1.02, ha='center', fontsize='large')
plt.tight_layout()
```



[17]:

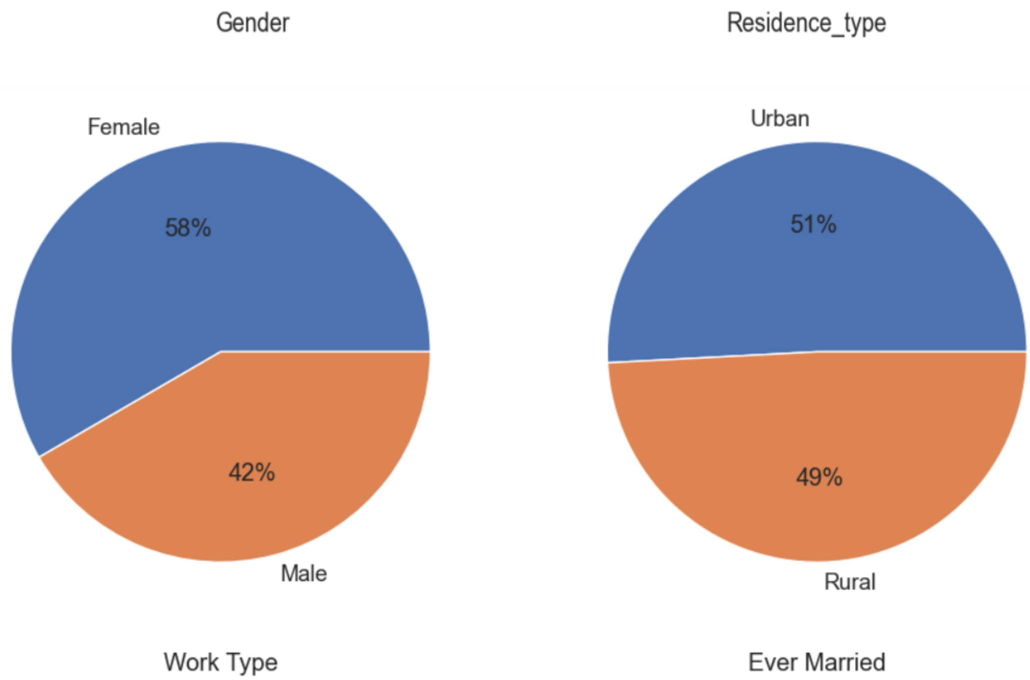
```
plt.figure(figsize=(15,15))

plt.subplot(3,2,1)
sns.boxplot(x=df['stroke'],y=df['age'])
plt.title('age')

plt.subplot(3,2,2)
sns.boxplot(x=df['stroke'],y=df['hypertension'])
plt.title('hypertension')

plt.subplot(3,2,3)
sns.boxplot(x=df['stroke'],y=df['heart_disease'])
plt.title('heart_disease')

plt.subplot(3,2,4)
sns.boxplot(x=df['stroke'],y=df['avg_glucose_level'])
plt.title('avg_glucose_level')

plt.subplot(3,2,5)
sns.boxplot(x=df['stroke'],y=df['bmi'])
plt.title('bmi')

plt.tight_layout()
```

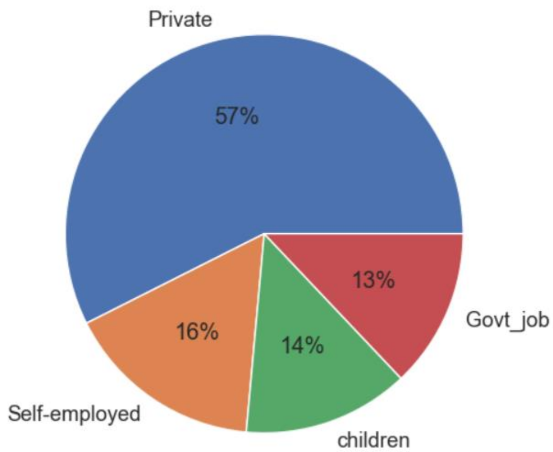


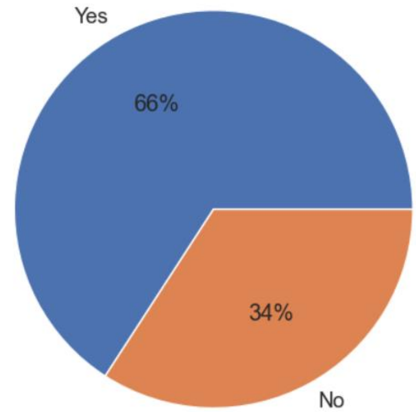


bmi
50
45
40
35
30
25
20
15
bmi
0
1
stroke

In [50]:
```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

gender_data = df['gender'].value_counts()
axs[0, 0].pie(gender_data, labels=gender_data.index, autopct='%.0f%%')
axs[0, 0].set_title('Gender')

residence_data = df['Residence_type'].value_counts()
axs[0, 1].pie(residence_data, labels=['Urban', 'Rural'], autopct='%.0f%%')
axs[0, 1].set_title('Residence_type')


work_type_data = df['work_type'].value_counts()
axs[1, 0].pie(work_type_data, labels=work_type_data.index, autopct='%.0f%%')
axs[1, 0].set_title('Work Type')

ever_married_data = df['ever_married'].value_counts()
axs[1, 1].pie(ever_married_data, labels=ever_married_data.index, autopct='%.0f%%')
axs[1, 1].set_title('Ever Married')

plt.tight_layout()

plt.show()
```




Private
57%
16%
14%
13%
Govt_job
Self-employed
children


Yes
66%
34%
No

HEATMAP

```
[21]: data= df[['age', 'hypertension', 'heart_disease','avg_glucose_level', 'bmi', 'stroke']]
df_corr = data.corr()

plt.figure(figsize=(12,6))

plt.title('Heatmap for data ')
sns.heatmap(df_corr, annot=True, linecolor='white', linewidth=0.4)
```

PREDICTION USING ML

Checking Accuracy on dataset using differnet Machine Learning Algorithms

In [37]:
```
#importing liberaries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import CategoricalNB
```

In [38]:
```
#Featuers
X = df[['age','hypertension', 'heart_disease', 'avg_glucose_level', 'bmi']]

#Target Value
y = df[['stroke']]
```

In [39]:
```
#Spliting data in train_data and test_data

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =0.2, random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[39]:
```
((3984, 5), (997, 5), (3984, 1), (997, 1))
```

In [40]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)
```

LOGISTIC REGRESSION ALGORITHM

In [22]:
```
#using Logistic regression

model = LogisticRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

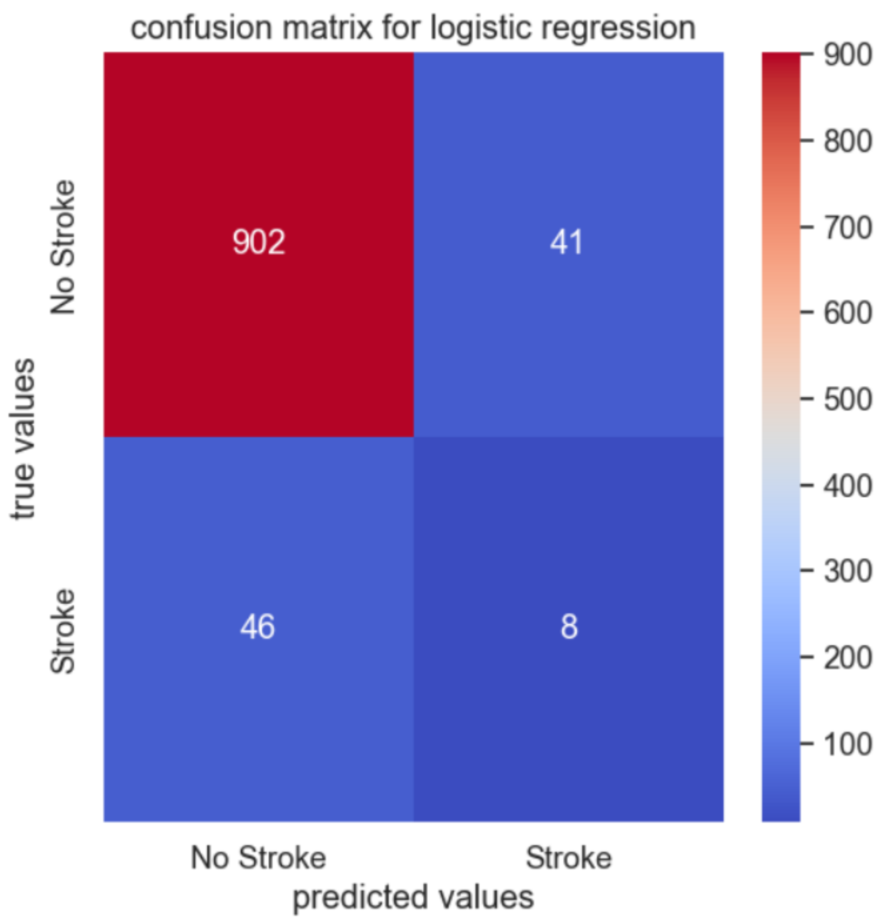
```
D:\Anaconda\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vector y was passed when a 1d a
rray was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
Accuracy: 0.9458375125376128
```

In [66]:
```
from sklearn.metrics import classification_report, confusion_matrix
model.fit(X_train,y_train)

pred = model.predict(X_test)

print(classification_report(y_test, pred))

cm = confusion_matrix(y_test, pred)

plt.figure(figsize=(6,6))
sns.set(font_scale=1.2)

sns.heatmap(cm, annot=True, fmt='d',cmap='coolwarm', xticklabels=['No Stroke', 'Stroke'], yticklabels=['No Stroke', 'Stroke'] )

plt.xlabel('predicted values')
plt.ylabel('true values')
plt.title('confusion matrix for logistic regression')
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       943
           1       0.16      0.15      0.16        54

    accuracy                           0.91       997
   macro avg       0.56      0.55      0.55       997
weighted avg       0.91      0.91      0.91       997
```

Out[66]: Text(0.5, 1.0, 'confusion matrix for logistic regression')



DECISION TREE ALGORITHM

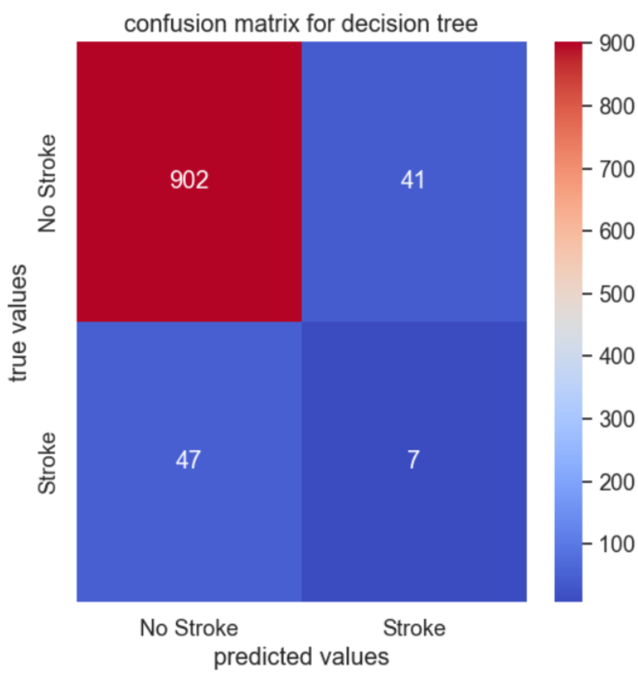
In [25]:
```
#using Decision Tree
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9087261785356068
```

In [26]:
```
from sklearn.metrics import classification_report, confusion_matrix
model.fit(X_train,y_train)
pred = model.predict(X_test)
print(classification_report(y_test, pred))
cm = confusion_matrix(y_test, pred)
plt.figure(figsize=(6,6))
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt='d',cmap='coolwarm', xticklabels=['No Stroke', 'Stroke'], yticklabels=['No Stroke', 'Stroke'] )
plt.xlabel('predicted values')
plt.ylabel('true values')
plt.title('confusion matrix for decision tree')
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       943
           1       0.15      0.13      0.14        54

    accuracy                           0.91       997
   macro avg       0.55      0.54      0.55       997
weighted avg       0.91      0.91      0.91       997
```

Out[26]:
```
Text(0.5, 1.0, 'confusion matrix for decision tree')
```



RANDOM FOREST ALGORITHM

In [27]:
```
#using random Forest
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

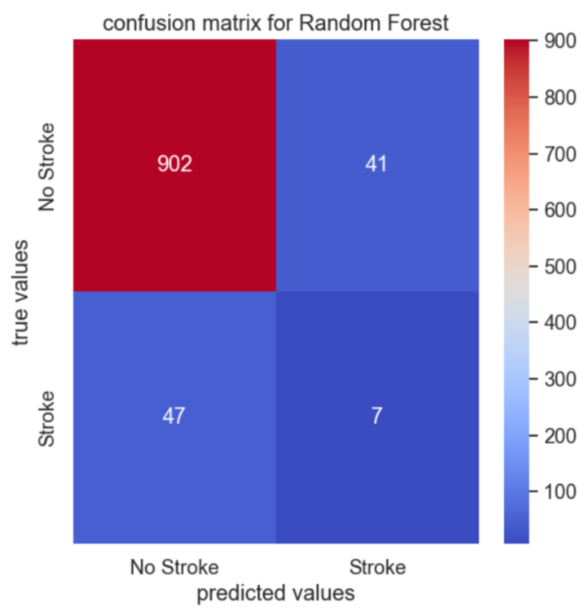
```
D:\Anaconda\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was passed when a 1d array was exp
ected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
Accuracy: 0.9408224674022067
```

In [27]:
```
from sklearn.metrics import classification_report, confusion_matrix
model.fit(X_train,y_train)
pred = model.predict(X_test)
print(classification_report(y_test, pred))
cm = confusion_matrix(y_test, pred)
plt.figure(figsize=(6,6))
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt='d',cmap='coolwarm', xticklabels=['No Stroke', 'Stroke'], yticklabels=['No Stroke', 'Stroke'] )
plt.xlabel('predicted values')
plt.ylabel('true values')
plt.title('confusion matrix for Random Forest')
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       943
           1       0.15      0.13      0.14        54

    accuracy                           0.91       997
   macro avg       0.55      0.54      0.55       997
weighted avg       0.91      0.91      0.91       997
```

Out[27]: `Text(0.5, 1.0, 'confusion matrix for Random Forest')`



K-NEAREST ALGORITHM

In [31]:
```
#using K-nearest Algorithm
model = KNeighborsClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9418254764292878
```

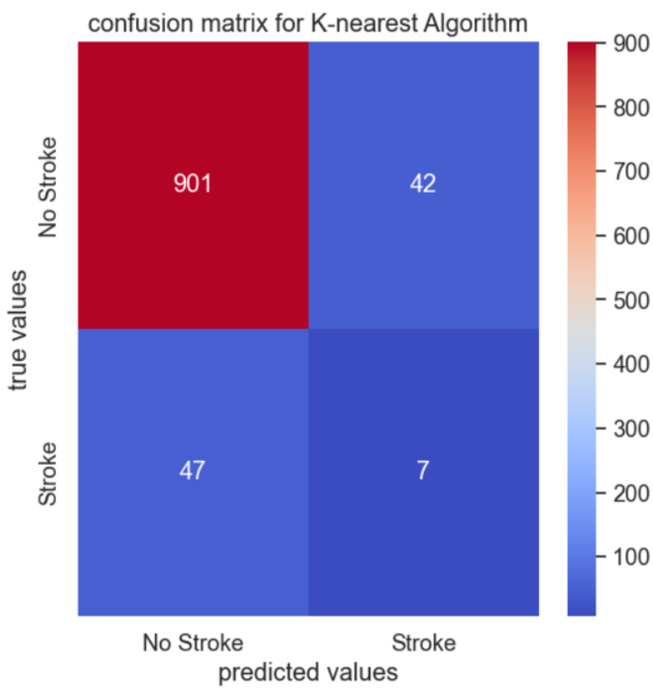
```
D:\Anaconda\Lib\site-packages\sklearn\neighbors\_classification.py:228: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

In [29]:
```
from sklearn.metrics import classification_report, confusion_matrix
model.fit(X_train,y_train)
pred = model.predict(X_test)
print(classification_report(y_test, pred))
cm = confusion_matrix(y_test, pred)
plt.figure(figsize=(6,6))
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt='d',cmap='coolwarm', xticklabels=['No Stroke', 'Stroke'], yticklabels=['No Stroke', 'Stroke'] )
plt.xlabel('predicted values')
plt.ylabel('true values')
plt.title('confusion matrix for K-nearest Algorithm')
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       943
           1       0.14      0.13      0.14        54

    accuracy                           0.91       997
   macro avg       0.55      0.54      0.54       997
weighted avg       0.91      0.91      0.91       997
```

Out[29]: Text(0.5, 1.0, 'confusion matrix for K-nearest Algorithm')

Out[29]: Text(0.5, 1.0, 'confusion matrix for K-nearest Algorithm')



CONCLUSION:

After using some algorithms here are results:

ALGORITHM	ACCUARCY
Logistic regression	0.94
Decision tree	0.90
Random Forest	0.94
K-nearest	0.94