Brain Stroke Risk Prediction

Cerebrovascular diseases, like strokes, are one of the leading causes of death in the world. Patients and health care systems are both burdened with significant health and financial issues as a result. Stroke risk can be influenced by health-related behavior, which is becoming an increasingly important prevention concern. This problem can be solved by developing automated stroke prediction algorithms, which may allow early intervention and possibly save lives. Here I am going to develop some machine learning alogorithms for early stroke prediction.

According to the World Health Organization (WHO), stroke is the world's second biggest cause of death, accounting for around 11% of all deaths.

Based on input criteria such as gender, age, various diseases, and smoking status, this dataset is used to predict whether a patient is likely to have a stroke. Each row of data contains pertinent information about the patient. The dataset having a total 5110 rows and 12 columns.

Columns are;

1) id: unique identifier

2) gender: "Male", "Female" or "Other"

3) age: age of the patient

4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension

5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease

6) ever_married: "No" or "Yes"

7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"

8) Residence_type: "Rural" or "Urban"

9) avg_glucose_level: average glucose level in blood

10) bmi: body mass index

11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*

12) stroke: 1 if the patient had a stroke or 0 if not

In [364]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in f:\athira\lib\site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in f:\athira\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in f:\athira\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in f:\athira\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in f:\athira\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in f:\athira\lib\site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in f:\athira\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in f:\athira\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in f:\athira\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in f:\athira\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in f:\athira\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.1
6.0)
```

In [365]:
```
!pip install mglearn
```

```
Requirement already satisfied: mglearn in f:\athira\lib\site-packages (0.2.0)
Requirement already satisfied: numpy in f:\athira\lib\site-packages (from mglearn) (1.24.3)
Requirement already satisfied: matplotlib in f:\athira\lib\site-packages (from mglearn) (3.7.2)
Requirement already satisfied: scikit-learn in f:\athira\lib\site-packages (from mglearn) (1.3.0)
Requirement already satisfied: pandas in f:\athira\lib\site-packages (from mglearn) (2.0.3)
Requirement already satisfied: pillow in f:\athira\lib\site-packages (from mglearn) (9.4.0)
Requirement already satisfied: cycler in f:\athira\lib\site-packages (from mglearn) (0.11.0)
Requirement already satisfied: imageio in f:\athira\lib\site-packages (from mglearn) (2.26.0)
Requirement already satisfied: joblib in f:\athira\lib\site-packages (from mglearn) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in f:\athira\lib\site-packages (from matplotlib->mglearn) (1.0.5)
Requirement already satisfied: fonttools>=4.22.0 in f:\athira\lib\site-packages (from matplotlib->mglearn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in f:\athira\lib\site-packages (from matplotlib->mglearn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in f:\athira\lib\site-packages (from matplotlib->mglearn) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in f:\athira\lib\site-packages (from matplotlib->mglearn) (3.0.
9)
Requirement already satisfied: python-dateutil>=2.7 in f:\athira\lib\site-packages (from matplotlib->mglearn) (2.8.
2)
Requirement already satisfied: pytz>=2020.1 in f:\athira\lib\site-packages (from pandas->mglearn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in f:\athira\lib\site-packages (from pandas->mglearn) (2023.3)
Requirement already satisfied: scipy>=1.5.0 in f:\athira\lib\site-packages (from scikit-learn->mglearn) (1.11.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in f:\athira\lib\site-packages (from scikit-learn->mglearn) (2.
2.0)
Requirement already satisfied: six>=1.5 in f:\athira\lib\site-packages (from python-dateutil>=2.7->matplotlib->mglea
rn) (1.16.0)
```

In [366]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [367]:
```
%matplotlib inline
import seaborn as sns
import plotly.graph_objects as go
import numpy as np
import warnings
import pickle
import os
warnings.filterwarnings('ignore')
```

Read Dataset

In [368]:
```
data = pd.read_csv(r"healthcare-dataset-stroke-data.csv")
data
```

Out[368]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	strok
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	
...	...	...	...	...	...	...	...	...	...	...	...	
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	

5110 rows × 12 columns

Head and Tail of the dataset

In [369]:
```
data.head()
```

Out[369]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

In [370]:
```
data.tail()
```

Out[370]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	strol
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	

Shape Of The Dataset

In [371]:
```
data.shape
```

Out[371]:
```
(5110, 12)
```

In [372]:
```
data.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 [373]:
```
data.describe()
```

Out[373]:

	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

Checking Null Values

In [374]:
```
data.isnull().sum()
```

Out[374]:
```
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
```

Fill Null Values

In [375]:
```
data['bmi'].value_counts()
```

Out[375]:
```
bmi
28.7    41
28.4    38
26.7    37
27.6    37
26.1    37
        ..
48.7     1
49.2     1
51.0     1
49.4     1
14.9     1
Name: count, Length: 418, dtype: int64
```

In [376]:
```
data['bmi'].describe()
```

Out[376]:
```
count    4909.000000
mean       28.893237
std         7.854067
min        10.300000
25%        23.500000
50%        28.100000
75%        33.100000
max        97.600000
Name: bmi, dtype: float64
```

In [377]:
```
data['bmi'].fillna(data['bmi'].mean(),inplace=True)
```

In [378]:
```
data['bmi'].describe()
```

Out[378]:
```
count    5110.000000
mean       28.893237
std         7.698018
min        10.300000
25%        23.800000
50%        28.400000
75%        32.800000
max        97.600000
Name: bmi, dtype: float64
```

In [379]:
```
data.isnull().sum()
```

Out[379]:
```
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
```

Drop Unwnated Column

In [380]:
```
data.drop('id',axis=1,inplace=True)
data
```

Out[380]:

	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	36.600000	formerly smoked	1
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
5105	Female	80.0	1	0	Yes	Private	Urban	83.75	28.893237	never smoked	0
5106	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.000000	never smoked	0
5107	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.600000	never smoked	0
5108	Male	51.0	0	0	Yes	Private	Rural	166.29	25.600000	formerly smoked	0
5109	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.200000	Unknown	0

5110 rows × 11 columns

Outlier Removal

In [381]:
```
data
```

Out[381]:

	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	36.600000	formerly smoked	1
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
5105	Female	80.0	1	0	Yes	Private	Urban	83.75	28.893237	never smoked	0
5106	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.000000	never smoked	0
5107	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.600000	never smoked	0
5108	Male	51.0	0	0	Yes	Private	Rural	166.29	25.600000	formerly smoked	0
5109	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.200000	Unknown	0

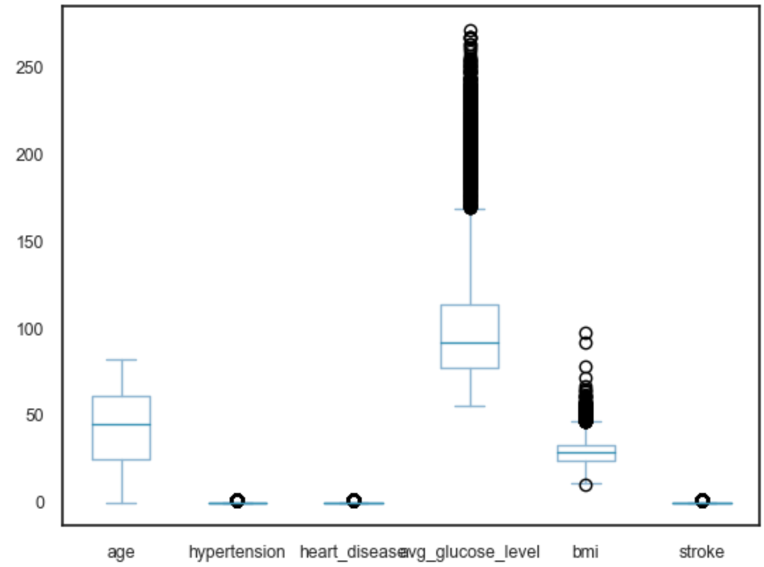
5110 rows × 11 columns

In [382]:
```
from matplotlib.pyplot import figure
figure(num=None, figsize=(20, 10), dpi=3000, facecolor='blue', edgecolor='black')
```

Out[382]:
```
<Figure size 60000x30000 with 0 Axes>
```

```
<Figure size 60000x30000 with 0 Axes>
```

In [383]:
```
data.plot(kind='box')
plt.show()
```



Check the Columns

In [384]:
```
data.columns
```

Out[384]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

In [385]:
```
data_cat=data[['gender', 'hypertension', 'heart_disease', 'ever_married','work_type', 'Residence_type',
                'smoking_status', 'stroke']]
```

Unique Things in the Dataset

In [386]:
```
for i in data_cat.columns:
    print(data_cat[i].unique())
```

```
['Male' 'Female' 'Other']
[0 1]
[1 0]
['Yes' 'No']
['Private' 'Self-employed' 'Govt_job' 'children' 'Never_worked']
['Urban' 'Rural']
['formerly smoked' 'never smoked' 'smokes' 'Unknown']
[1 0]
```

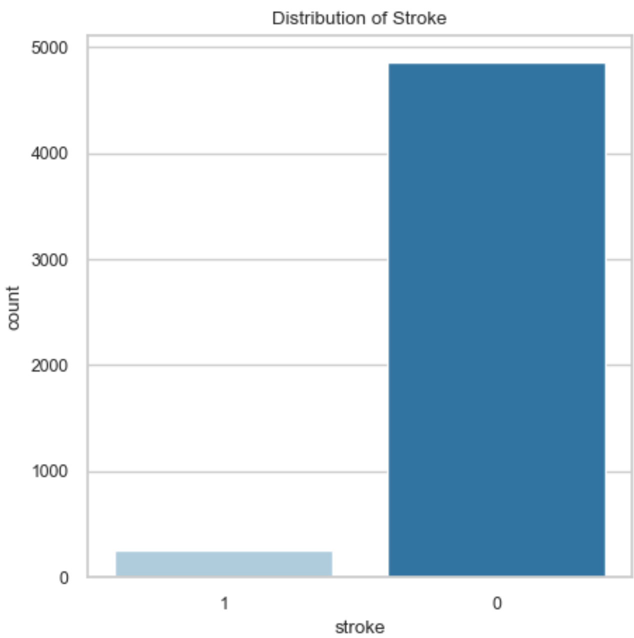
In [387]:
```
for i in data_cat.columns:
    print(data_cat[i].value_counts())
```

```
gender
Female    2994
Male      2115
Other        1
Name: count, dtype: int64
hypertension
0    4612
1     498
Name: count, dtype: int64
heart_disease
0    4834
1     276
Name: count, dtype: int64
ever_married
Yes    3353
No     1757
Name: count, dtype: int64
work_type
Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: count, dtype: int64
Residence_type
Urban    2596
Rural    2514
Name: count, dtype: int64
smoking_status
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: count, dtype: int64
stroke
0    4861
1     249
Name: count, dtype: int64
```

Exploratory Data Analysis(EDA)

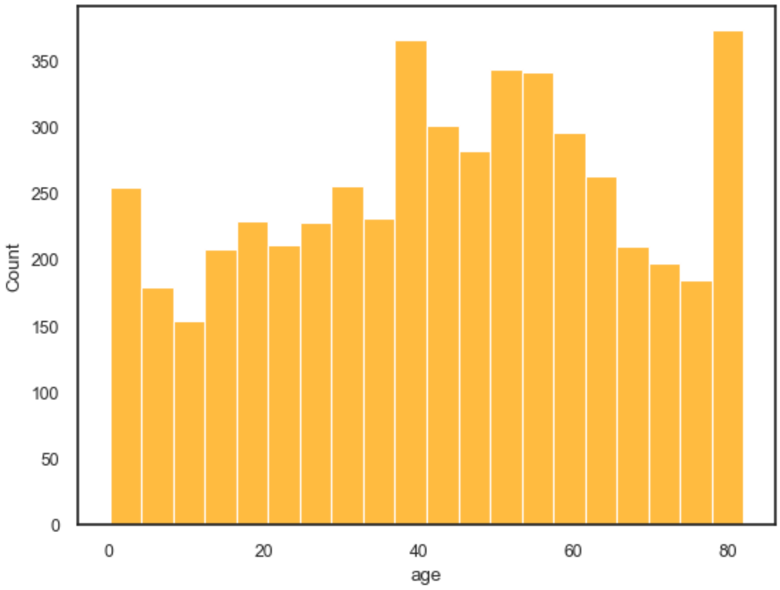
Target Variable

In [508]:
```
sns.set_style("whitegrid")
plt.figure(figsize=(5, 5))
sns.countplot(x=data['stroke'], palette = 'Paired')
plt.title('Distribution of Stroke')
plt.show()
```

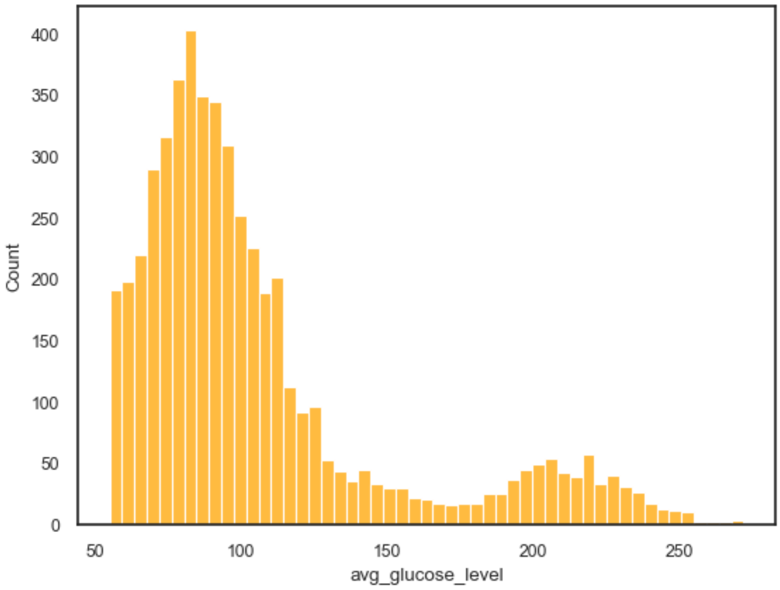


Visualisations for Age, Avg Glucose Level and BMI [Values which we enter]

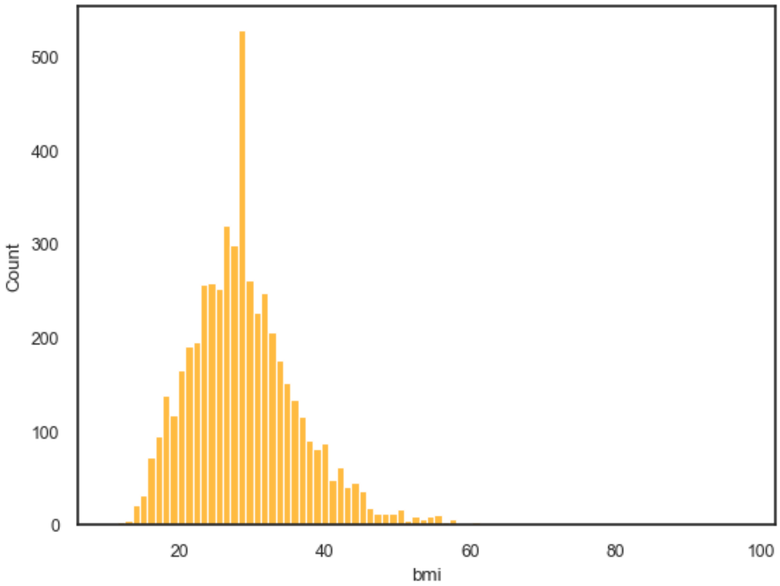
In [467]:
```
sns.histplot(x=data.age, linewidth=.8, color ='Orange')

plt.show()
```



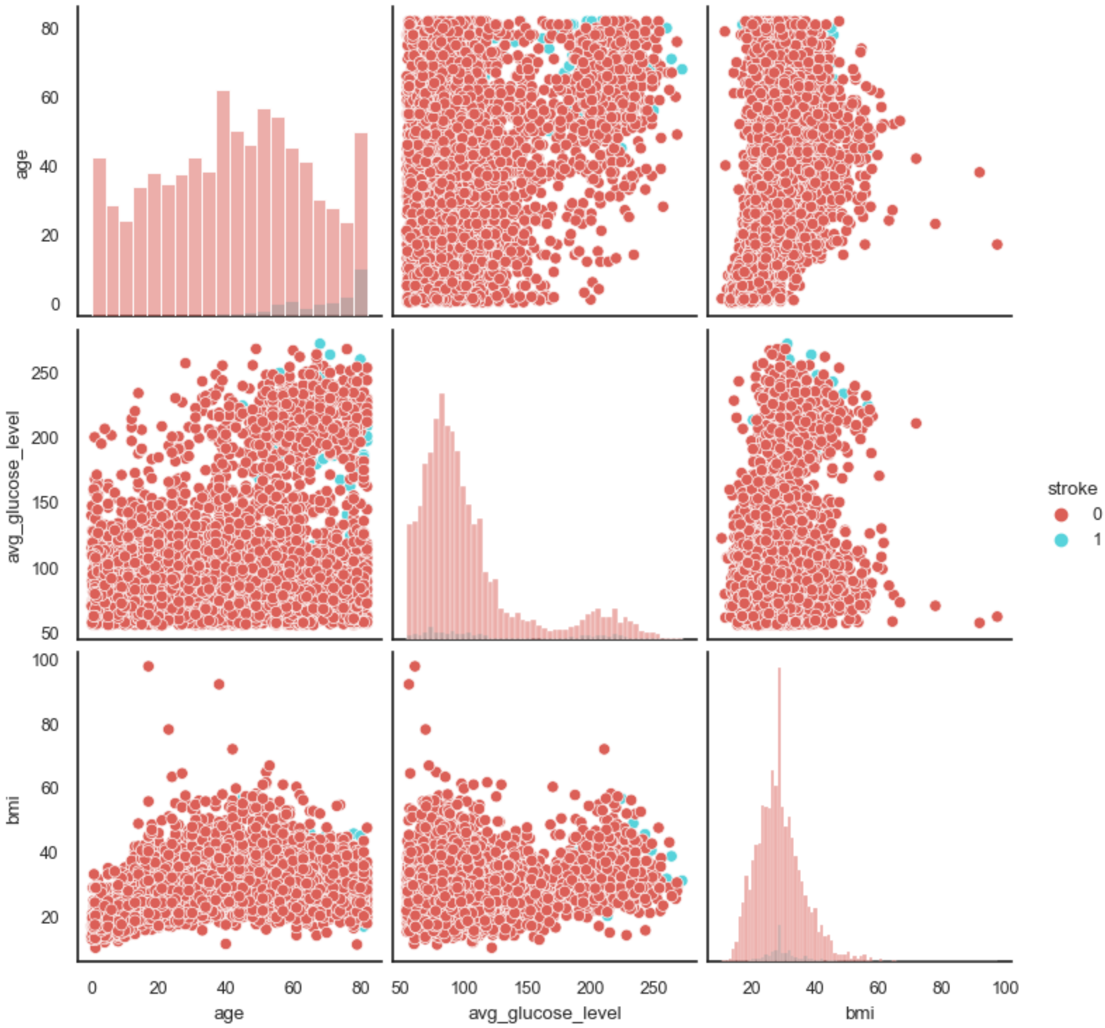
In [463]:
```
sns.histplot(data.avg_glucose_level, linewidth=.8, color = 'Orange')

plt.show()
```



In [464]:
```
sns.histplot(data.bmi,linewidth=.8, color = 'Orange')

plt.show()
```

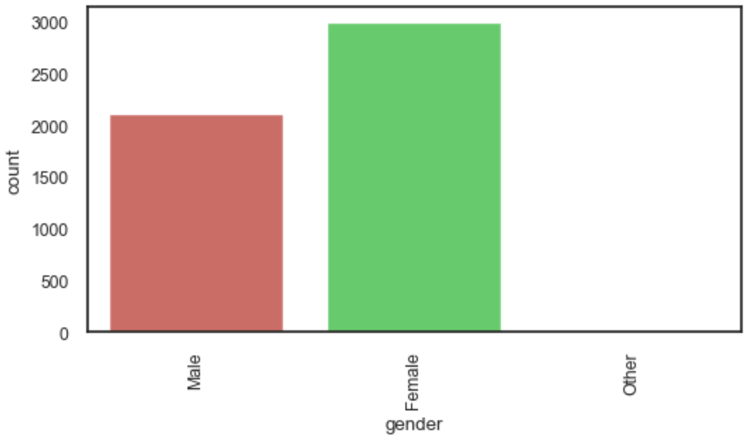


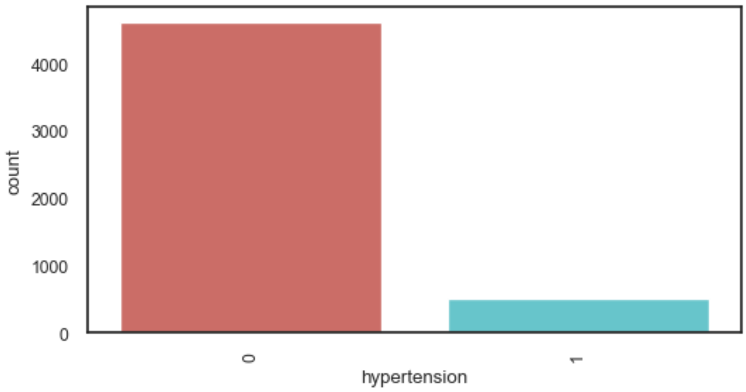
In [391]:
```
sns.pairplot(data.loc[:, ['age', 'avg_glucose_level', 'bmi', 'stroke']], hue="stroke", diag_kind="hist",  palette='hl
plt.show()
```

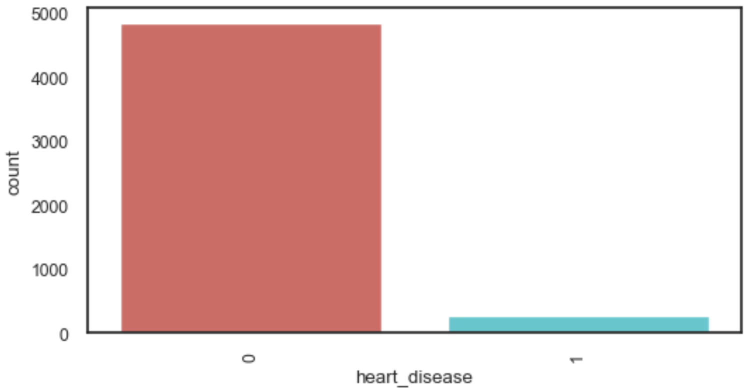


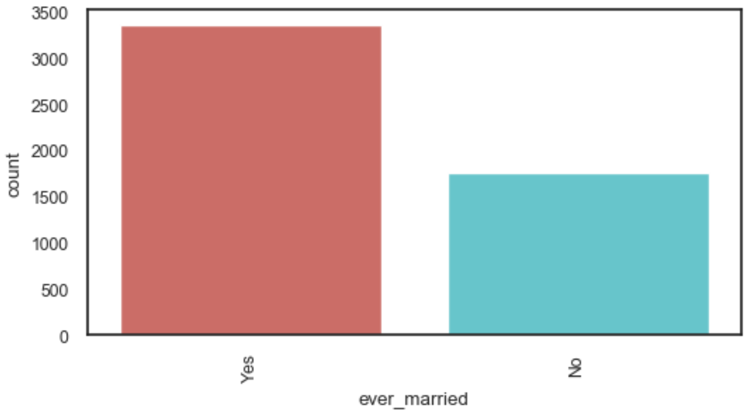
Visualisations for Gender, Hypertension, Heart Disease, Ever Married, Work Type, Residence Type and Smoking Status [Values which we select]

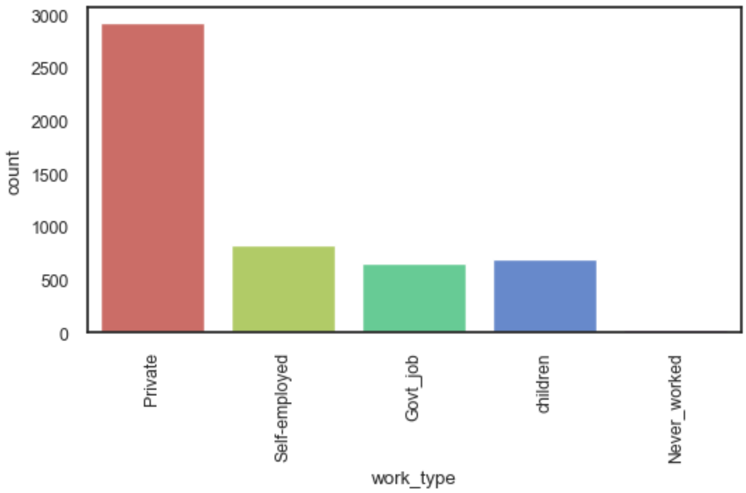
In [488]:
```
for i in data_cat.columns:
    plt.figure(figsize = (6,3))
    sns.countplot(x = data_cat[i], data = data_cat, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

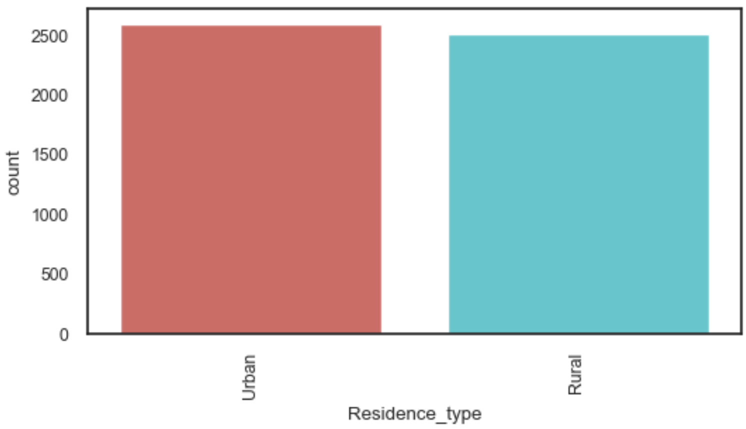


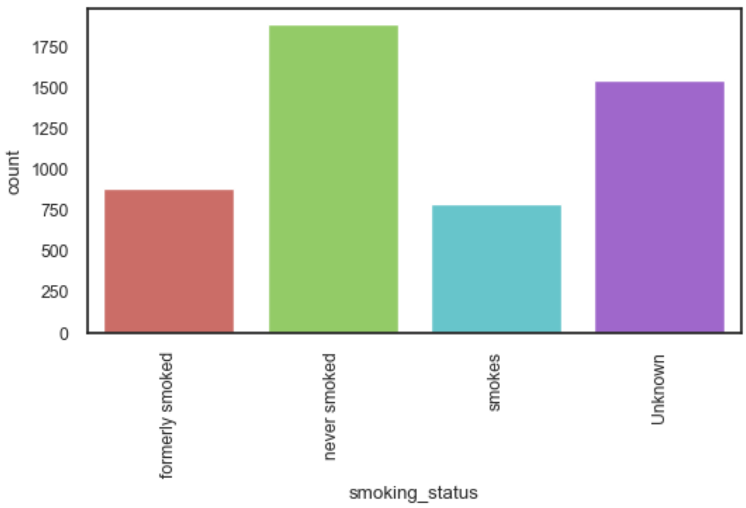


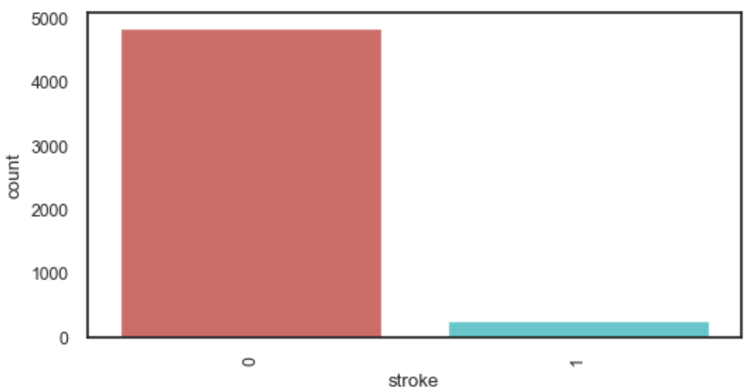





3000
2500
2000
1500
1000
500
0
count
Private
Self-employed
Govt_job
children
Never_worked
work_type


2500
2000
1500
1000
500
0
count
Urban
Rural
Residence_type


1750
1500
1250
1000
750
500
250
0
count
formerly smoked
never smoked
smokes
Unknown
smoking_status

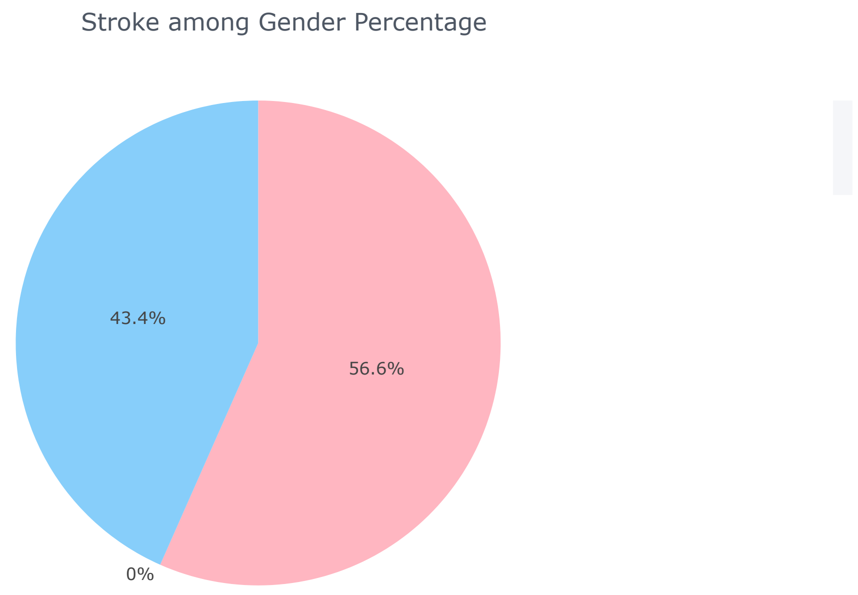

5000
4000
3000
2000
1000
0
count
0
1
stroke

In [393]:
```
!pip install cufflinks
```

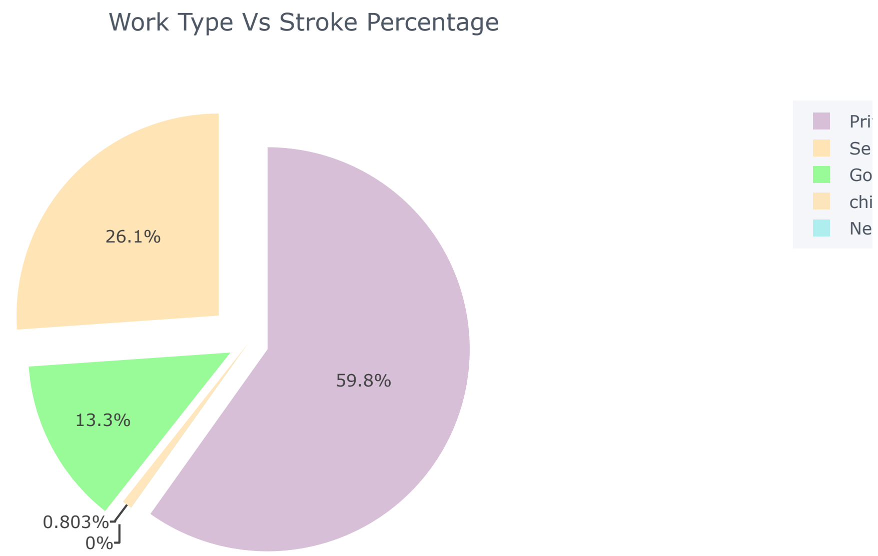
```
Requirement already satisfied: cufflinks in f:\athira\lib\site-packages (0.17.3)
Requirement already satisfied: numpy>=1.9.2 in f:\athira\lib\site-packages (from cufflinks) (1.24.3)
Requirement already satisfied: pandas>=0.19.2 in f:\athira\lib\site-packages (from cufflinks) (2.0.3)
Requirement already satisfied: plotly>=4.1.1 in f:\athira\lib\site-packages (from cufflinks) (5.9.0)
Requirement already satisfied: six>=1.9.0 in f:\athira\lib\site-packages (from cufflinks) (1.16.0)
Requirement already satisfied: colorlover>=0.2.1 in f:\athira\lib\site-packages (from cufflinks) (0.3.0)
Requirement already satisfied: setuptools>=34.4.1 in f:\athira\lib\site-packages (from cufflinks) (68.0.0)
Requirement already satisfied: ipython>=5.3.0 in f:\athira\lib\site-packages (from cufflinks) (8.15.0)
Requirement already satisfied: ipywidgets>=7.0.0 in f:\athira\lib\site-packages (from cufflinks) (8.0.4)
Requirement already satisfied: backcall in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (0.2.0)
Requirement already satisfied: decorator in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (5.1.1)
Requirement already satisfied: jedi>=0.16 in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (0.18.1)
Requirement already satisfied: matplotlib-inline in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (0.
1.6)
Requirement already satisfied: pickleshare in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30 in f:\athira\lib\site-packages (from ipython>=
5.3.0->cufflinks) (3.0.36)
Requirement already satisfied: pygments>=2.4.0 in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (2.1
5.1)
Requirement already satisfied: stack-data in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (0.2.0)
Requirement already satisfied: traitlets>=5 in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (5.7.1)
Requirement already satisfied: colorama in f:\athira\lib\site-packages (from ipython>=5.3.0->cufflinks) (0.4.6)
Requirement already satisfied: ipykernel>=4.5.1 in f:\athira\lib\site-packages (from ipywidgets>=7.0.0->cufflinks)
(6.25.0)
Requirement already satisfied: widgetsnbextension~=4.0 in f:\athira\lib\site-packages (from ipywidgets>=7.0.0->cuffl
inks) (4.0.5)
Requirement already satisfied: jupyterlab-widgets~=3.0 in f:\athira\lib\site-packages (from ipywidgets>=7.0.0->cuffl
inks) (3.0.5)
Requirement already satisfied: python-dateutil>=2.8.2 in f:\athira\lib\site-packages (from pandas>=0.19.2->cufflink
s) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in f:\athira\lib\site-packages (from pandas>=0.19.2->cufflinks) (2023.3.
post1)
Requirement already satisfied: tzdata>=2022.1 in f:\athira\lib\site-packages (from pandas>=0.19.2->cufflinks) (2023.
3)
Requirement already satisfied: tenacity>=6.2.0 in f:\athira\lib\site-packages (from plotly>=4.1.1->cufflinks) (8.2.
2)
Requirement already satisfied: comm>=0.1.1 in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0-
>cufflinks) (0.1.2)
Requirement already satisfied: debugpy>=1.6.5 in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.
0.0->cufflinks) (1.6.7)
Requirement already satisfied: jupyter-client>=6.1.12 in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidg
ets>=7.0.0->cufflinks) (7.4.9)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipy
widgets>=7.0.0->cufflinks) (5.3.0)
Requirement already satisfied: nest-asyncio in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0
->cufflinks) (1.5.6)
Requirement already satisfied: packaging in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->c
ufflinks) (23.1)
Requirement already satisfied: psutil in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->cuff
links) (5.9.0)
Requirement already satisfied: pyzmq>=20 in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->c
ufflinks) (23.2.0)
Requirement already satisfied: tornado>=6.1 in f:\athira\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0
->cufflinks) (6.3.2)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in f:\athira\lib\site-packages (from jedi>=0.16->ipython>=5.3.0->
cufflinks) (0.8.3)
Requirement already satisfied: wcwidth in f:\athira\lib\site-packages (from prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30->
ipython>=5.3.0->cufflinks) (0.2.5)
Requirement already satisfied: executing in f:\athira\lib\site-packages (from stack-data->ipython>=5.3.0->cufflinks)
(0.8.3)
Requirement already satisfied: asttokens in f:\athira\lib\site-packages (from stack-data->ipython>=5.3.0->cufflinks)
(2.0.5)
Requirement already satisfied: pure-eval in f:\athira\lib\site-packages (from stack-data->ipython>=5.3.0->cufflinks)
(0.2.2)
Requirement already satisfied: entrypoints in f:\athira\lib\site-packages (from jupyter-client>=6.1.12->ipykernel>=
4.5.1->ipywidgets>=7.0.0->cufflinks) (0.4)
Requirement already satisfied: platformdirs>=2.5 in f:\athira\lib\site-packages (from jupyter-core!=5.0.*,>=4.12->ip
ykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (3.10.0)
Requirement already satisfied: pywin32>=300 in f:\athira\lib\site-packages (from jupyter-core!=5.0.*,>=4.12->ipykern
el>=4.5.1->ipywidgets>=7.0.0->cufflinks) (305.1)
```

In [394]:
```
import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
```

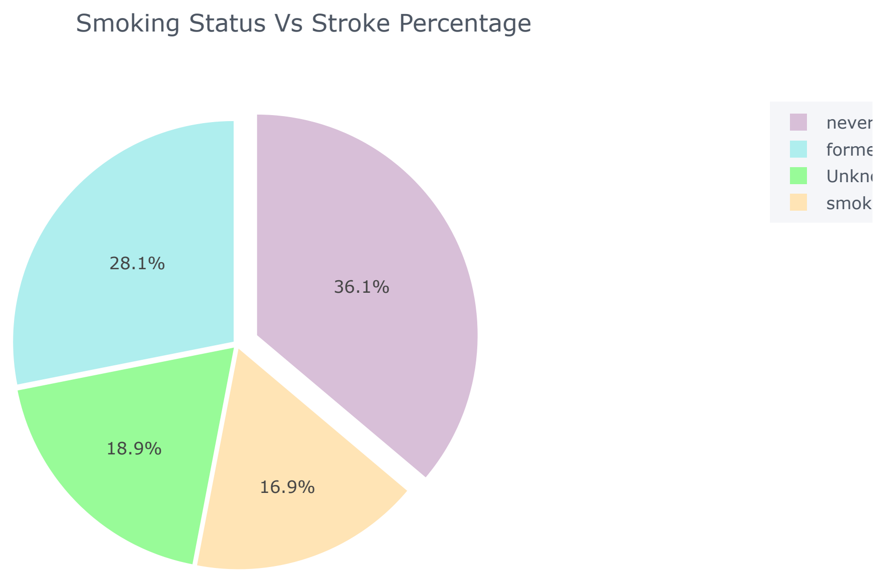
In [395]:
```
gender = data.groupby(data['gender'])['stroke'].sum()
data_gender = pd.DataFrame({'labels': gender.index,
                   'values': gender.values
                  })
colors = ['lightpink', 'lightskyblue']
data_gender.iplot(kind='pie',labels='labels',values='values', title='Stroke among Gender Percentage', colors = colors
```



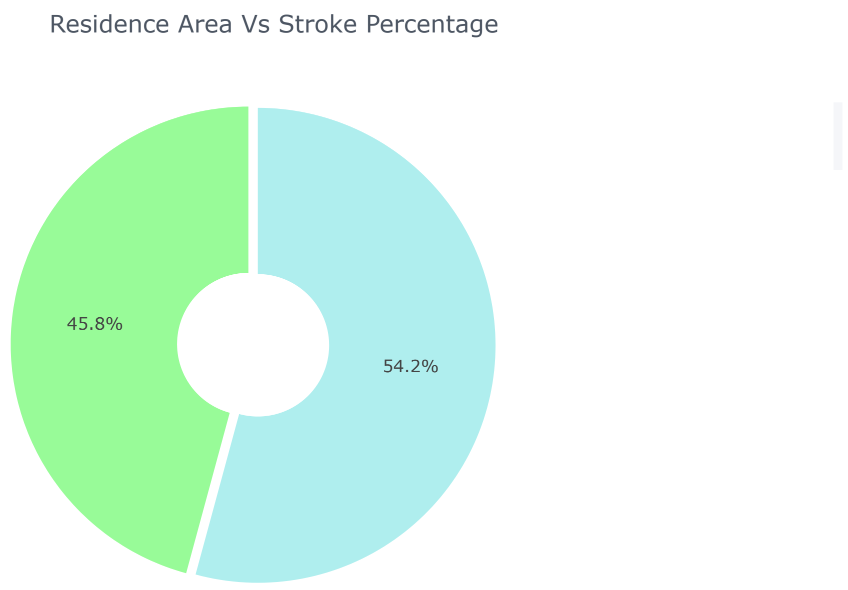
In [396]:
```
job = data.groupby(data['work_type'])['stroke'].sum()
data_job = pd.DataFrame({'labels': job.index,
                   'values': job.values
                  })
colors2= ['palegreen','paleturquoise','thistle','moccasin']
data_job.iplot(kind='pie',labels='labels',values='values', title='Work Type Vs Stroke Percentage', colors = colors2,pu
```



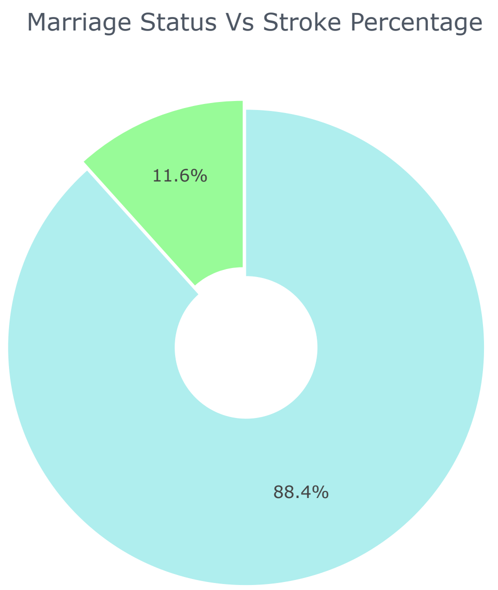
In [397]:
```
smoke = data.groupby(data['smoking_status'])['stroke'].sum()
data_smoke = pd.DataFrame({'labels': smoke.index,'values': smoke.values})
data_smoke.iplot(kind='pie',labels='labels',values='values', title='Smoking Status Vs Stroke Percentage', colors = co
            pull=[0.02, 0.02, 0.1, 0.02])
```



In [398]:
```
Residence = data.groupby(data['Residence_type'])['stroke'].sum()
data_Residence = pd.DataFrame({'labels': Residence.index,
                   'values': Residence.values
                  })
data_Residence.iplot(kind='pie',labels='labels',values='values', title='Residence Area Vs Stroke Percentage', colors
            pull=[0.02, 0.02],hole = 0.3)
```

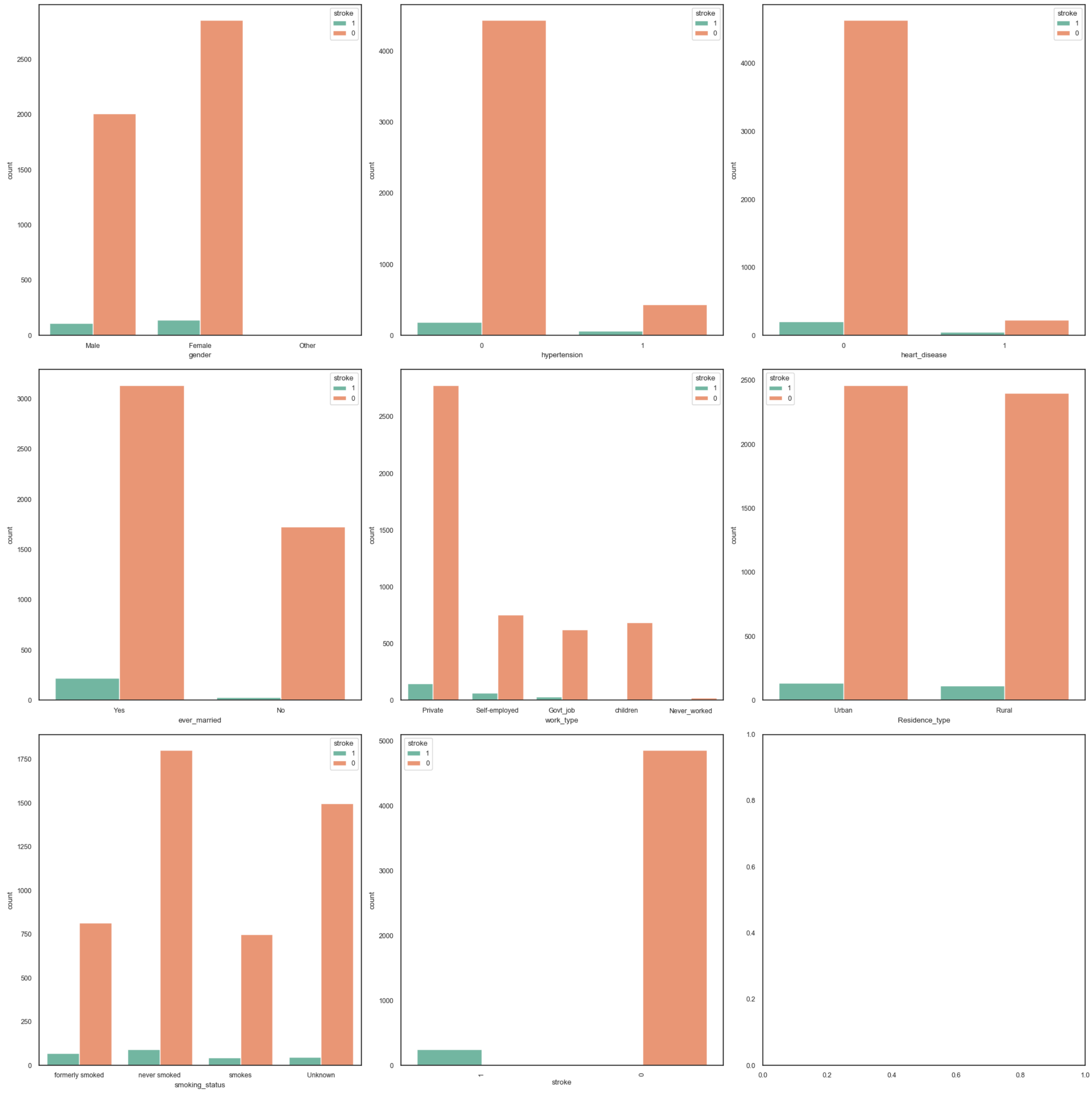


In [399]:
```
Married = data.groupby(data['ever_married'])['stroke'].sum()
data_Married = pd.DataFrame({'labels': Married.index,
                   'values': Married.values
                  })
data_Married.iplot(kind='pie',labels='labels',values='values', title='Marriage Status Vs Stroke Percentage', colors =
            pull=[0.02, 0.02],hole = 0.3)
```



Bivariate Analysis

In [400]:
```
fig, axs = plt.subplots(3, 3, figsize=(20, 20))
data['stroke'] = data['stroke'].astype(str)
for var, subplot in zip(data_cat, axs.flatten()):
 sns.countplot(x=data[var], hue=data['stroke'], ax=subplot, palette = 'Set2')
for label in subplot.get_xticklabels():
 label.set_rotation(90)
plt.tight_layout()
plt.show()
```

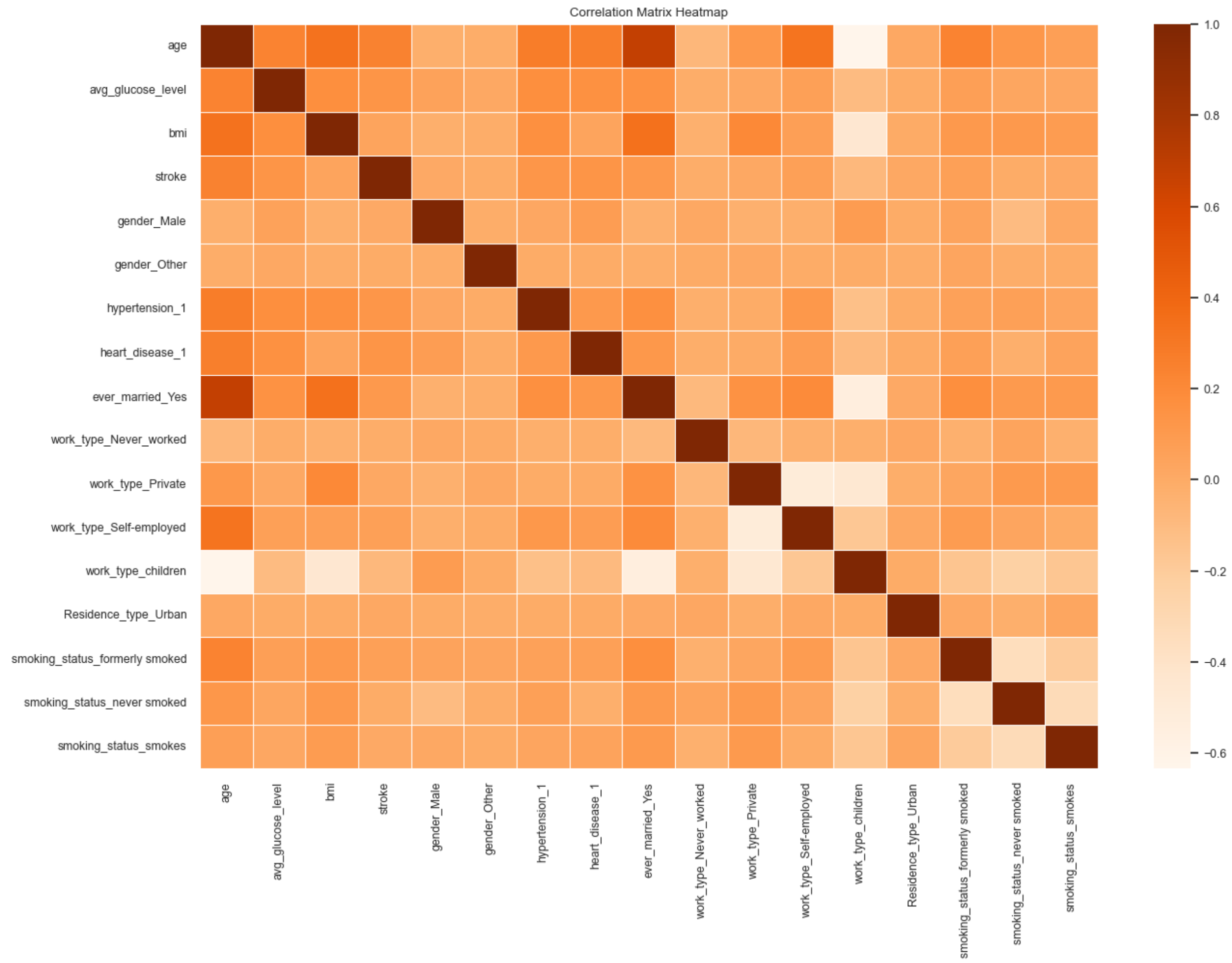


Multivariate Analysis

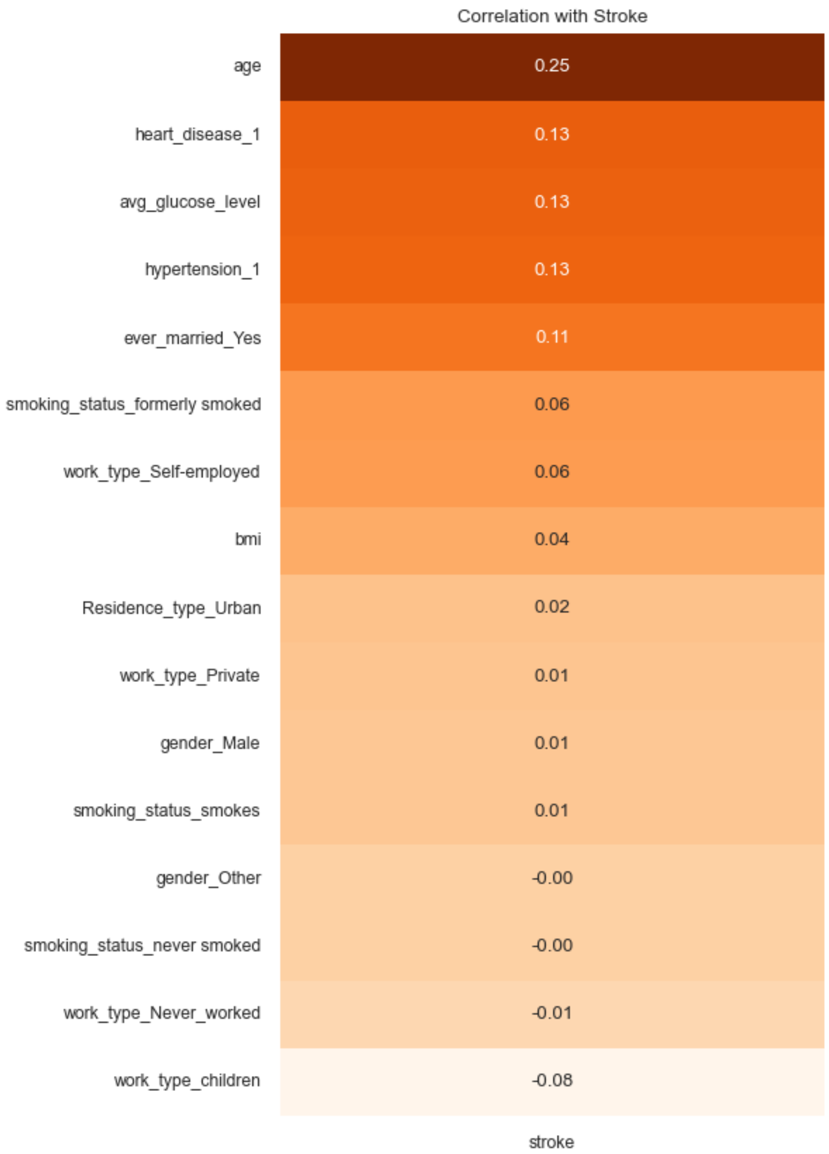
In [401]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [402]:
```
org_data = pd.read_csv(r"healthcare-dataset-stroke-data.csv")
org_data = org_data.drop("id",axis=1)

categorical_data = ['gender', 'hypertension', 'heart_disease', 'ever_married','work_type', 'Residence_type',
                 'smoking_status']

onehotencoder = OneHotEncoder(drop='first', sparse=False)
onehotencoded = onehotencoder.fit_transform(org_data[categorical_data])

onehotencoded_df = pd.DataFrame(onehotencoded, columns=onehotencoder.get_feature_names_out(categorical_data))

org_data.drop(categorical_data, axis=1, inplace=True)

dataonehotencoded = pd.concat([org_data, onehotencoded_df], axis=1)
```

In [403]:
```
correlationmatrix_onehot = dataonehotencoded.corr()

plt.figure(figsize=(15, 10))
sns.heatmap(correlationmatrix_onehot, annot=False, cmap='Oranges', linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()
```



In [404]:
```
target_corr = correlationmatrix_onehot['stroke'].drop('stroke')
target_corr_sorted = target_corr.sort_values(ascending=False)

plt.figure(figsize=(5, 10))
sns.set(font_scale=0.8)
sns.set_style("white")
sns.set_palette("PuBuGn_d")
sns.heatmap(target_corr_sorted.to_frame(), cmap="Oranges", annot=True, fmt='.2f', cbar=False)
plt.title('Correlation with Stroke')
plt.show()
```



Label Encoding

In [405]:
```
data.head()
```

Out[405]:

	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	36.600000	formerly smoked	1
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1

In [406]:
```
from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
```

In [407]:
```
gender=enc.fit_transform(data['gender'])
ever_married=enc.fit_transform(data['ever_married'])
work_type=enc.fit_transform(data['work_type'])
Residence_type=enc.fit_transform(data['Residence_type'])
smoking_status=enc.fit_transform(data['smoking_status'])
```

In [408]:
```
data['gender']=gender
data['ever_married']=ever_married
data['work_type']=work_type
data['Residence_type']=Residence_type
data['smoking_status']=smoking_status
```

In [409]:
```
data
```

Out[409]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.600000	1	1
1	0	61.0	0	0	1	3	0	202.21	28.893237	2	1
2	1	80.0	0	1	1	2	0	105.92	32.500000	2	1
3	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
4	0	79.0	1	0	1	3	0	174.12	24.000000	2	1
...	...	...	...	...	...	...	...	...	...	...	...
5105	0	80.0	1	0	1	2	1	83.75	28.893237	2	0
5106	0	81.0	0	0	1	3	1	125.20	40.000000	2	0
5107	0	35.0	0	0	1	3	0	82.99	30.600000	2	0
5108	1	51.0	0	0	1	2	0	166.29	25.600000	1	0
5109	0	44.0	0	0	1	0	1	85.28	26.200000	0	0

5110 rows × 11 columns

In [410]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             5110 non-null   int32  
 1   age                5110 non-null   float64
 2   hypertension       5110 non-null   int64  
 3   heart_disease      5110 non-null   int64  
 4   ever_married       5110 non-null   int32  
 5   work_type          5110 non-null   int32  
 6   Residence_type     5110 non-null   int32  
 7   avg_glucose_level  5110 non-null   float64
 8   bmi                5110 non-null   float64
 9   smoking_status     5110 non-null   int32  
 10  stroke             5110 non-null   object 
dtypes: float64(3), int32(5), int64(2), object(1)
memory usage: 339.5+ KB
```

Splitting Data as Train and Test

In [411]:
```
x = data.drop('stroke',axis=1)
```

In [412]:
```
x.head()
```

Out[412]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1	67.0	0	1	1	2	1	228.69	36.600000	1
1	0	61.0	0	0	1	3	0	202.21	28.893237	2
2	1	80.0	0	1	1	2	0	105.92	32.500000	2
3	0	49.0	0	0	1	2	1	171.23	34.400000	3
4	0	79.0	1	0	1	3	0	174.12	24.000000	2

In [413]:
```
y = data['stroke']
```

In [414]:
```
y
```

Out[414]:
```
0       1
1       1
2       1
3       1
4       1
       ..
5105    0
5106    0
5107    0
5108    0
5109    0
Name: stroke, Length: 5110, dtype: object
```

In [415]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.2,random_state=10)
```

In [416]:
```
x_train
```

Out[416]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
2285	1	49.0	0	0	1	2	0	79.64	28.893237	3
4733	1	67.0	0	0	1	2	0	83.16	25.500000	1
3905	1	78.0	0	0	1	2	1	208.85	24.400000	1
4700	1	47.0	0	0	1	2	0	110.14	30.500000	3
4939	0	59.0	0	0	1	2	1	71.08	28.100000	2
...	...	...	...	...	...	...	...	...	...	...
1180	0	62.0	0	0	1	2	0	82.57	36.000000	1
3441	0	59.0	0	0	1	3	1	90.06	28.900000	3
1344	1	47.0	0	0	1	2	0	86.37	39.200000	3
4623	1	25.0	0	0	1	0	1	166.38	23.100000	2
1289	0	80.0	0	0	1	3	0	72.61	27.600000	2

4088 rows × 10 columns

In [417]:
```
x_test
```

Out[417]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
2413	0	58.00	0	0	1	2	0	100.42	39.500000	3
1141	1	57.00	0	0	1	2	0	90.06	29.800000	0
146	1	65.00	0	0	1	3	1	68.43	28.893237	1
3883	0	1.64	0	0	0	4	1	69.89	18.100000	0
1044	0	79.00	0	0	1	0	1	93.89	30.400000	2
...	...	...	...	...	...	...	...	...	...	...
2261	1	59.00	0	0	1	2	1	60.35	25.900000	1
4712	1	57.00	0	0	1	2	1	93.04	29.200000	2
4971	0	63.00	0	0	1	2	1	57.06	37.900000	2
2224	1	57.00	0	0	1	2	0	76.28	31.400000	1
4825	0	14.00	0	0	0	4	1	71.80	18.800000	0

1022 rows × 10 columns

In [418]:
```
y_train
```

Out[418]:
```
2285    0
4733    0
3905    0
4700    0
4939    0
       ..
1180    0
3441    0
1344    0
4623    0
1289    0
Name: stroke, Length: 4088, dtype: object
```

In [419]:
```
y_test
```
Out[419]:
```
2413    0
1141    0
146     1
3883    0
1044    0
       ..
2261    0
4712    0
4971    0
2224    0
4825    0
Name: stroke, Length: 1022, dtype: object
```

In [420]:
```
data.describe()
```
Out[420]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smokir
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	511
mean	0.414286	43.226614	0.097456	0.054012	0.656164	2.167710	0.508023	106.147677	28.893237	
std	0.493044	22.612647	0.296607	0.226063	0.475034	1.090293	0.499985	45.283560	7.698018	
min	0.000000	0.080000	0.000000	0.000000	0.000000	0.000000	0.000000	55.120000	10.300000	
25%	0.000000	25.000000	0.000000	0.000000	0.000000	2.000000	0.000000	77.245000	23.800000	
50%	0.000000	45.000000	0.000000	0.000000	1.000000	2.000000	1.000000	91.885000	28.400000	
75%	1.000000	61.000000	0.000000	0.000000	1.000000	3.000000	1.000000	114.090000	32.800000	
max	2.000000	82.000000	1.000000	1.000000	1.000000	4.000000	1.000000	271.740000	97.600000	

In [421]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
```

In [422]:
```
x_train_std = std.fit_transform(x_train)
x_test_std = std.transform(x_test)
```

Save Scalar Values

In [423]:
```
scaler_path=os.path.join('F:\Athira\Stroke_Prediction','models\scaler.pkl')
with open(scaler_path,'wb') as scaler_file:
    pickle.dump(std,scaler_file)
```

In [424]:
```
x_train_std
```
Out[424]:
```
array([[ 1.19359699,  0.2521852 , -0.33069968, ..., -0.58626884,
         0.00238781,  1.51158251],
       [ 1.19359699,  1.04686385, -0.33069968, ..., -0.50843521,
        -0.44065504, -0.35191245],
       [ 1.19359699,  1.5325008 , -0.33069968, ...,  2.27080023,
        -0.58427812, -0.35191245],
       ...,
       [ 1.19359699,  0.16388757, -0.33069968, ..., -0.43745625,
         1.34810513,  1.51158251],
       [ 1.19359699, -0.80738634, -0.33069968, ...,  1.33171097,
        -0.75401449,  0.57983503],
       [-0.83780372,  1.62079843, -0.33069968, ..., -0.74171498,
        -0.16646553,  0.57983503]])
```

In [425]:
```
x_test_std
```
Out[425]:
```
array([[-0.83780372,  0.64952452, -0.33069968, ..., -0.12678509,
         1.38727506,  1.51158251],
       [ 1.19359699,  0.60537571, -0.33069968, ..., -0.35586361,
         0.12078063, -1.28365994],
       [ 1.19359699,  0.95856622, -0.33069968, ..., -0.83414241,
         0.00238781, -0.35191245],
       ...,
       [-0.83780372,  0.87026859, -0.33069968, ..., -1.08555387,
         1.17836876,  0.57983503],
       [ 1.19359699,  0.60537571, -0.33069968, ..., -0.66056457,
         0.32968693, -0.35191245],
       [-0.83780372, -1.29302329, -0.33069968, ..., -0.75962556,
        -1.31545016, -1.28365994]])
```

Model Building

1. Logistic Regression

In [426]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

In [427]:
```
from sklearn.metrics import accuracy_score
```

In [428]:
```
lr.fit(x_train_std,y_train)
```

Out[428]:
```
▾ LogisticRegression
LogisticRegression()
```

In [429]:
```
y_pred_lr = lr.predict(x_test_std)
y_pred_lr
```

Out[429]:
```
array(['0', '0', '0', ..., '0', '0', '0'], dtype=object)
```

In [430]:
```
ac_lr = accuracy_score(y_test,y_pred_lr)
ac_lr
```

Out[430]:
```
0.9383561643835616
```

In [431]:
```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

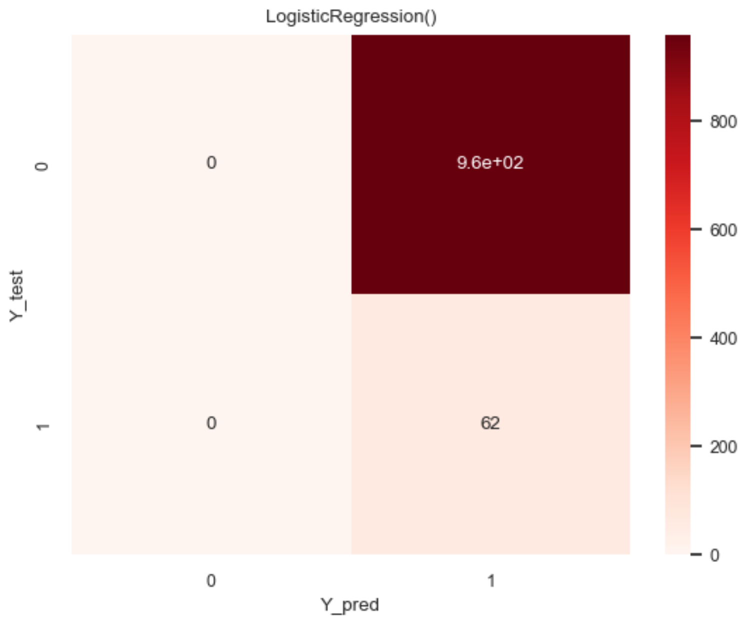
In [432]:
```
cm = confusion_matrix(y_test,y_pred_lr)
```

In [433]:
```
print(cm)
```

```
[[959   1]
 [ 62   0]]
```

In [468]:
```
pred_list = [lr]

for i in pred_list:
    print("Score : ",i.score(x_test,y_test))
    y_pred_lr = i.predict(x_test)
    sns.heatmap(confusion_matrix(y_test,y_pred_lr),annot = True, cmap="Reds")
    plt.xlabel("Y_pred")
    plt.ylabel("Y_test")
    plt.title(i)
    plt.show()
```

```
Score :  0.060665362035225046
```



2. Decision Tree

In [435]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
```

In [436]:
```
dt.fit(x_train_std,y_train)
```

Out[436]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [437]:
```
dt.feature_importances_
```

Out[437]:
```
array([0.02517249, 0.18128016, 0.01323121, 0.02484502, 0.03041762,
       0.04191648, 0.05010702, 0.2847877 , 0.27836895, 0.06987336])
```

In [438]:
```
x_train.columns
```

Out[438]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status'],
      dtype='object')
```

In [439]:
```
y_pred_dt = dt.predict(x_test_std)
y_pred_dt
```

Out[439]:
```
array(['0', '0', '0', ..., '0', '0', '0'], dtype=object)
```

In [440]:
```
ac_dt=accuracy_score(y_test,y_pred_dt)
ac_dt
```

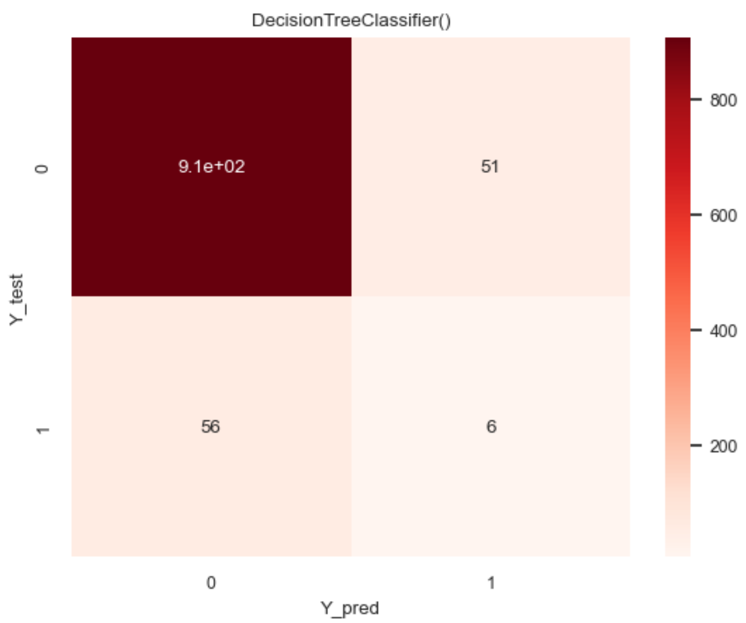
Out[440]:
```
0.8953033268101761
```

In [441]:
```
cm = confusion_matrix(y_test,y_pred_dt)
print(cm)
```

```
[[909  51]
 [ 56   6]]
```

In [469]:
```
pred_list = [dt]

for i in pred_list:
    print("Score : ",i.score(x_test,y_test))
    y_pred = i.predict(x_test)
    sns.heatmap(confusion_matrix(y_test,y_pred_dt),annot = True, cmap="Reds")
    plt.xlabel("Y_pred")
    plt.ylabel("Y_test")
    plt.title(i)
    plt.show()
```

```
Score :  0.9344422700587084
```



3. Random Forest

In [443]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
```

In [444]:
```
rf.fit(x_train_std,y_train)
```

Out[444]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [470]:
```
y_pred_rf=rf.predict(x_test_std)
y_pred_rf
```

Out[470]:
```
array(['0', '0', '0', ..., '0', '0', '0'], dtype=object)
```

In [471]:
```
ac_rf=accuracy_score(y_test,y_pred_rf)
ac_rf
```

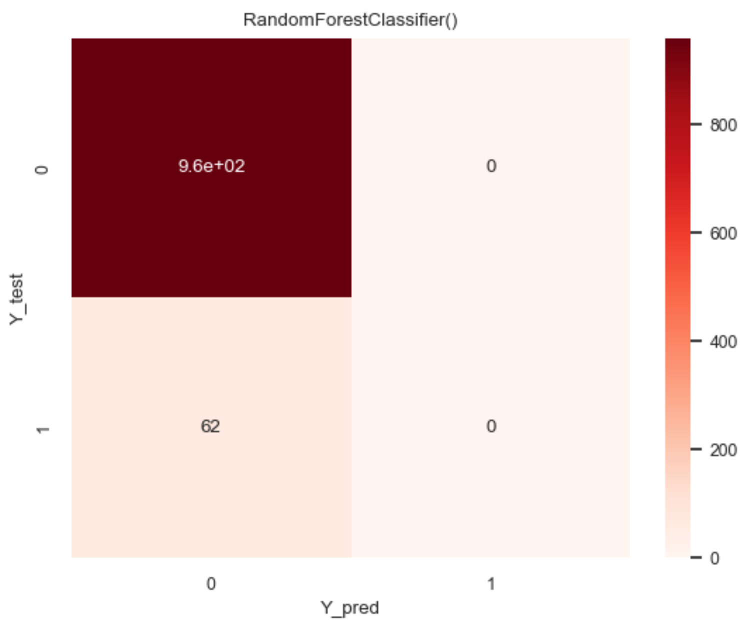
Out[471]:
```
0.9373776908023483
```

In [472]:
```
cm = confusion_matrix(y_test,y_pred_rf)
print(cm)
```

```
[[958   2]
 [ 62   0]]
```

In [473]:
```
pred_list = [rf]

for i in pred_list:
    print("Score : ",i.score(x_test,y_test))
    y_pred_rf = i.predict(x_test)
    sns.heatmap(confusion_matrix(y_test,y_pred_rf),annot = True, cmap="Reds")
    plt.xlabel("Y_pred")
    plt.ylabel("Y_test")
    plt.title(i)
    plt.show()
```

```
Score :  0.9393346379647749
```



4. K-Nearest Neighbors(KNN)

In [447]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

In [448]:
```
knn.fit(x_train_std,y_train)
```

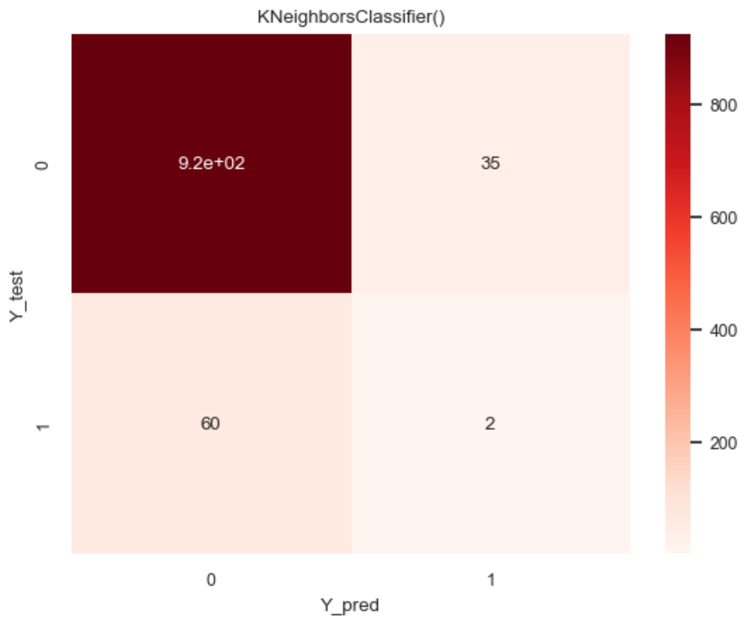
Out[448]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [474]:
```
y_pred_knn=knn.predict(x_test_std)
y_pred_knn
```
Out[474]:
```
array(['0', '0', '0', ..., '0', '0', '0'], dtype=object)
```

In [475]:
```
ac_knn=accuracy_score(y_test,y_pred_knn)
ac_knn
```
Out[475]:
```
0.9344422700587084
```

In [476]:
```
cm = confusion_matrix(y_test,y_pred_knn)
print(cm)
```
```
[[955   5]
 [ 62   0]]
```

In [477]:
```
pred_list = [knn]

for i in pred_list:
    print("Score : ",i.score(x_test,y_test))
    y_pred_knn = i.predict(x_test)
    sns.heatmap(confusion_matrix(y_test,y_pred_knn),annot = True, cmap="Reds")
    plt.xlabel("Y_pred")
    plt.ylabel("Y_test")
    plt.title(i)
    plt.show()
```
```
Score :  0.9070450097847358
```



5. Standard Vector Machine(SVM)

In [451]:
```
from sklearn.svm import SVC
sv=SVC()
```

In [452]:
```
sv.fit(x_train_std,y_train)
```
Out[452]:
```
▾ SVC
SVC()
```

In [478]:
```
y_pred_sv=sv.predict(x_test_std)
y_pred_sv
```
Out[478]:
```
array(['0', '0', '0', ..., '0', '0', '0'], dtype=object)
```

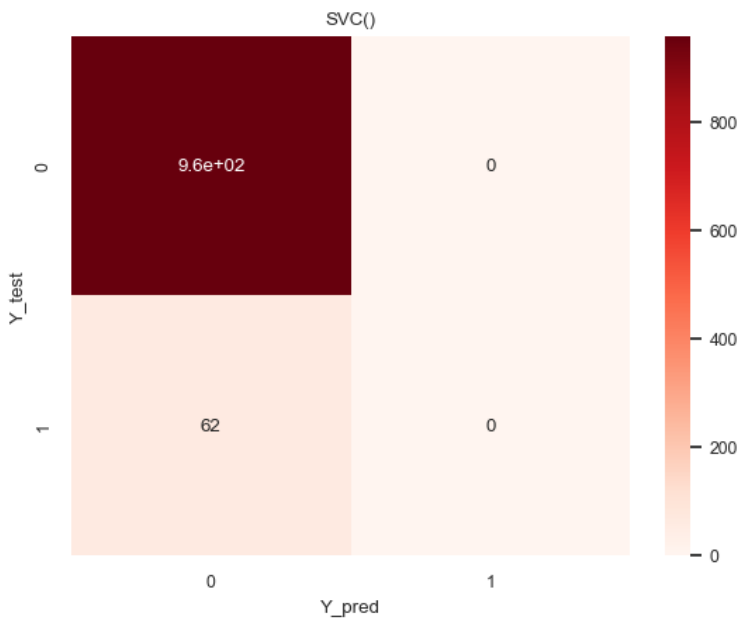
In [454]:
```
ac_sv=accuracy_score(y_test,y_pred_svm)
ac_sv
```
Out[454]:
```
0.9393346379647749
```

In [479]:
```
cm = confusion_matrix(y_test,y_pred_svm)
print(cm)
```

```
[[960   0]
 [ 62   0]]
```

In [480]:
```
pred_list = [sv]

for i in pred_list:
    print("Score : ",i.score(x_test,y_test))
    y_pred_sv = i.predict(x_test)
    sns.heatmap(confusion_matrix(y_test,y_pred_sv),annot = True, cmap="Reds")
    plt.xlabel("Y_pred")
    plt.ylabel("Y_test")
    plt.title(i)
    plt.show()
```

```
Score :  0.9393346379647749
```

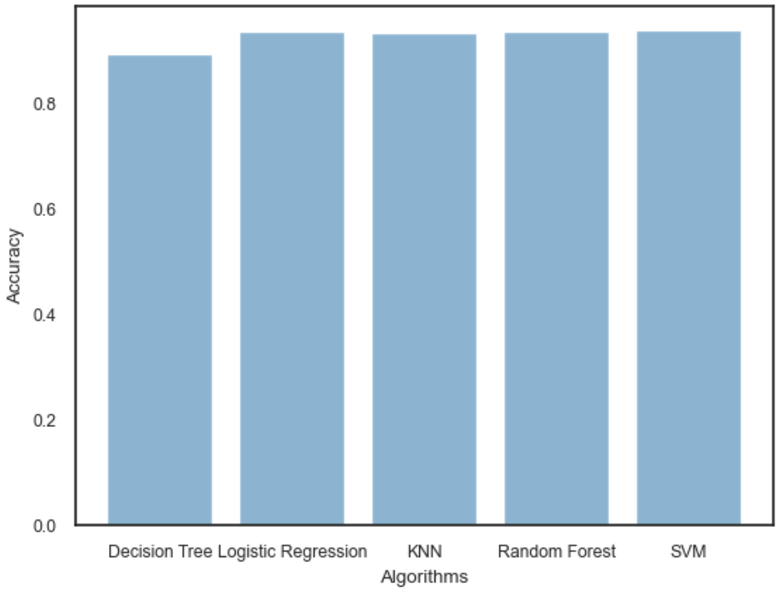


In [455]:
```
import joblib
model_path=os.path.join('F:\Athira\Stroke_Prediction','models\scaler.pkl')
joblib.dump(dt,model_path)
```

Out[455]: ['F:\\Athira\\Stroke_Prediction\\models\\scaler.pkl']

Cross Validation

In [487]:
```
plt.bar(['Decision Tree','Logistic Regression','KNN','Random Forest','SVM'],[ac_dt,ac_lr,ac_knn,ac_rf,ac_sv] )
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")
plt.show()
```



From the above barplot we can see that all the 5 machine learning algorithms providing an accuracy more than 90%. Among them the Logistic Regression, Knn, Random Forest and SVM having highest accuracy. I should prefer SVM as the best.