Stroke Prediction

Data Exploring and Visualization

In [51]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [52]:
```python
df = pd.read_csv("healthcare-dataset-stroke-data.csv")
```

In [53]:
```python
df.head()
```

Out[53]:

	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 [54]:
```python
df.describe()
```

Out[54]:

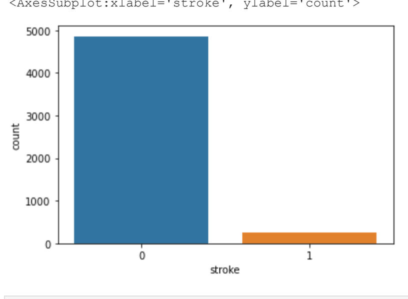
	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
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 [55]:
```python
df.shape
```

Out[55]:
```
(5110, 12)
```

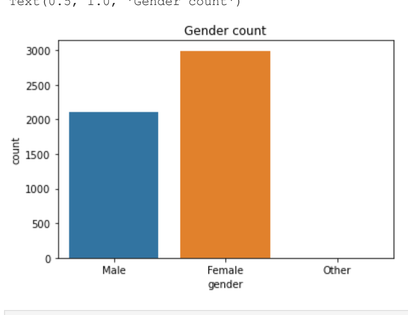
In [56]:
```python
sns.countplot(x = df['stroke'])
```

Out[56]:
```
<AxesSubplot:xlabel='stroke', ylabel='count'>
```



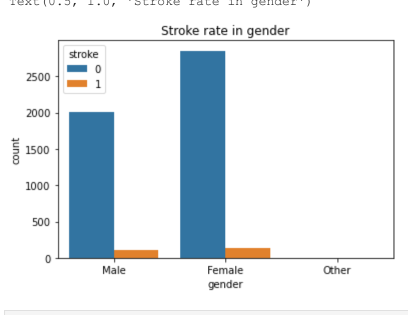
In [57]:
```python
sns.countplot(x = df['gender'])
plt.title('Gender count')
```

Out[57]:
```
Text(0.5, 1.0, 'Gender count')
```



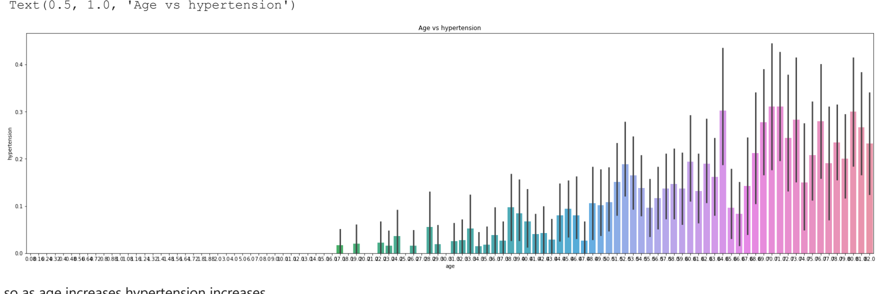
In [58]:
```python
sns.countplot(x=df['gender'],hue=df['stroke'])
plt.title('Stroke rate in gender')
```

Out[58]:
```
Text(0.5, 1.0, 'Stroke rate in gender')
```



In [59]:
```python
plt.figure(figsize=(30,8))
sns.barplot(x=df['age'],y=df['hypertension'])
plt.title('Age vs hypertension')
```

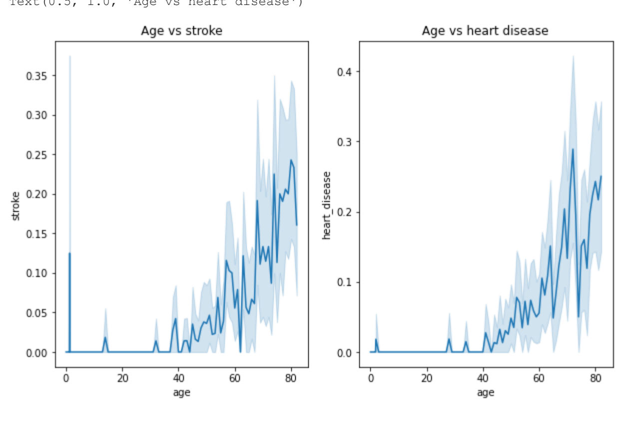
Out[59]:
```
Text(0.5, 1.0, 'Age vs hypertension')
```



so as age increases hypertension increases

In [60]:
```python
plt.figure(figsize=(10,6))

plt.subplot(1,2,1)
sns.lineplot(x=df['age'],y=df['stroke'])
plt.title('Age vs stroke')

plt.subplot(1,2,2)
sns.lineplot(x=df['age'],y=df['heart_disease'])
plt.title('Age vs heart disease')
```

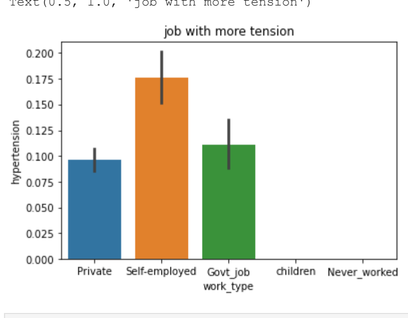
Out[60]:
```
Text(0.5, 1.0, 'Age vs heart disease')
```



> so as age increases chance of having a stroke increases
>
> similarly as age increases chances of having heart diseases also increases

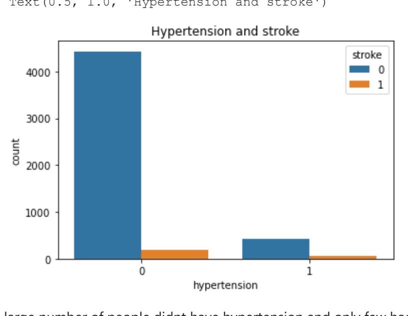
In [61]:
```python
sns.barplot(x=df['work_type'],y=df['hypertension'])
plt.title('job with more tension')
```

Out[61]:
```
Text(0.5, 1.0, 'job with more tension')
```



In [62]:
```python
sns.countplot(x=df['hypertension'],hue=df['stroke'])
plt.title('Hypertension and stroke')
```

Out[62]:
```
Text(0.5, 1.0, 'Hypertension and stroke')
```

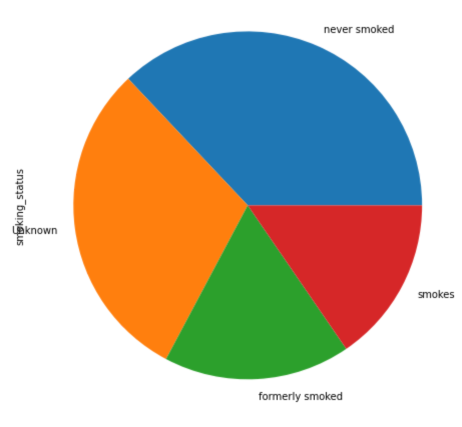


large number of people didnt have hypertension and only few had a stroke where as count of people with hypertension where few but significant number of people had a stroke.

so we can say that chances of having a stroke increases with hypertension

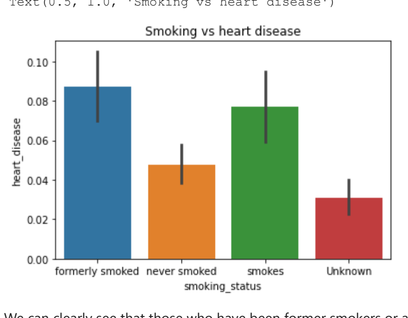
In [63]:
```python
plt.figure(figsize=(8,8))
df['smoking_status'].value_counts().plot(kind='pie')
```

Out[63]:
```
<AxesSubplot:ylabel='smoking_status'>
```



In [64]:
```python
sns.barplot(x=df['smoking_status'],y=df['heart_disease'])
plt.title("Smoking vs heart disease")
```

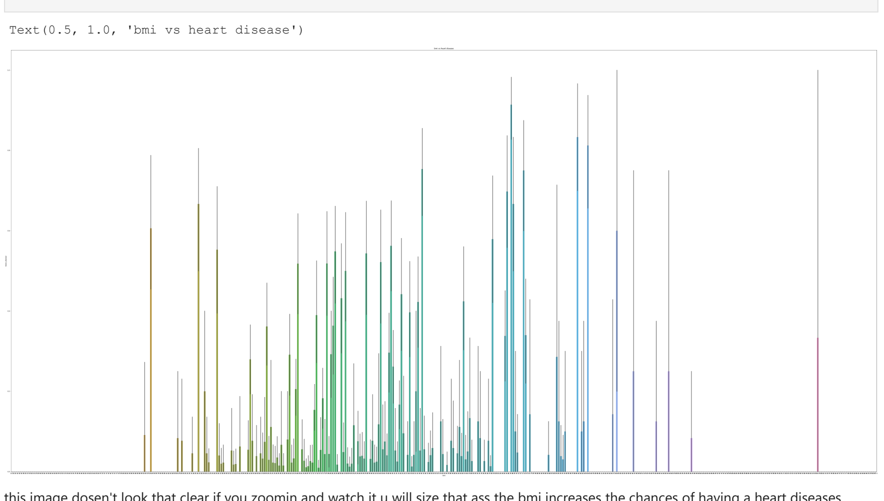
Out[64]:
```
Text(0.5, 1.0, 'Smoking vs heart disease')
```



We can clearly see that those who have been former smokers or are still smoking have a greater chance of heart diseases

In [111]:
```python
plt.figure(figsize=(100,50))
sns.barplot(x=df['bmi'],y=df['heart_disease'])
plt.title('bmi vs heart disease')
```

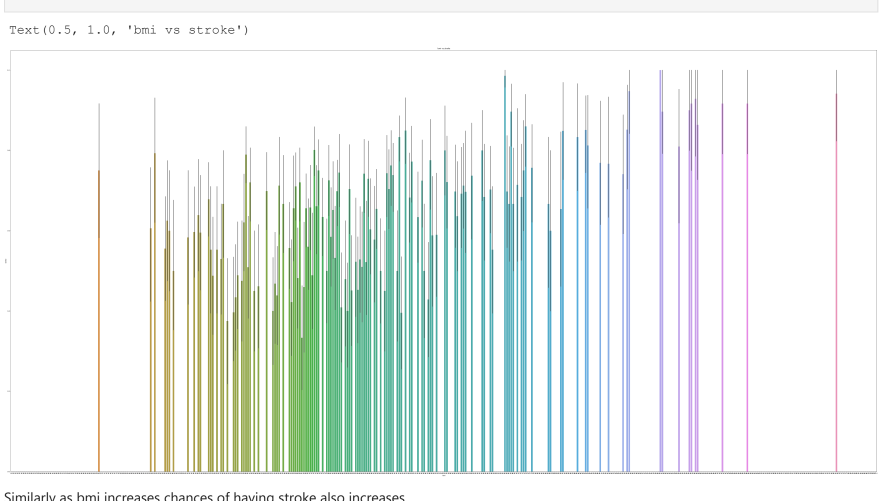
Out[111]:
```
Text(0.5, 1.0, 'bmi vs heart disease')
```



this image dosen't look that clear if you zoomin and watch it u will size that ass the bmi increases the chances of having a heart diseases increases

In [110]:
```python
plt.figure(figsize=(100,50))
sns.barplot(x=df['bmi'],y=df['stroke'])
plt.title('bmi vs stroke')
```

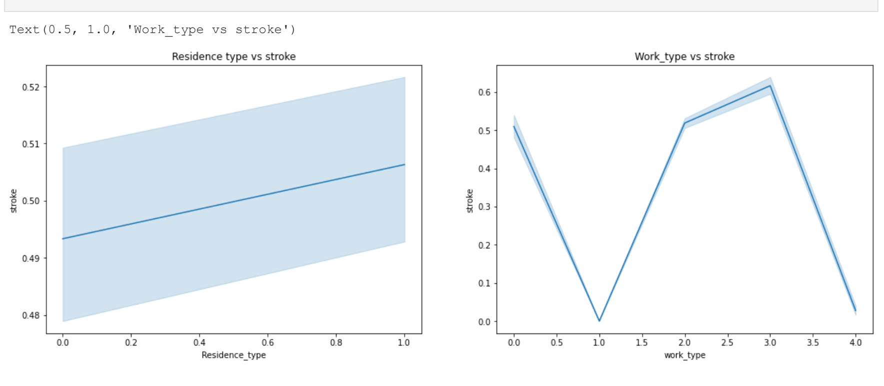
Out[110]:
```
Text(0.5, 1.0, 'bmi vs stroke')
```



Similarly as bmi increases chances of having stroke also increases

In [109]:
```python
plt.figure(figsize=(18,6))

plt.subplot(1,2,1)
sns.lineplot(x=df['Residence_type'],y=df['stroke'])
plt.title("Residence type vs stroke")

plt.subplot(1,2,2)
sns.lineplot(x=df['work_type'],y=df['stroke'])
plt.title("Work_type vs stroke")
```

Out[109]:
```
Text(0.5, 1.0, 'Work_type vs stroke')
```



> People living in Urban area tent to have higher chance of having a strok then rural area people
>
> People who are self_employed have a high chance to have a stroke

In [108]:
```python
sns.barplot(x=df['ever_married'],y=df['stroke'])
plt.title('Married vs stroke')
```

Out[108]:
```
Text(0.5, 1.0, 'Married vs stroke')
```

married people are tent to have a stroke then unmarried people

Data Cleaning

In [69]:
```python
df.isna().sum()
```

Out[69]:
```
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
```

In [70]:
```python
df.dropna(inplace=True)
```

In [71]:
```python
df.isna().sum()
```

Out[71]:
```
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
```

as bmi had a good relation with heart diseases and strokes so we didnt drop the whole column

In [75]:
```python
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(df['smoking_status'])
df['smoking_status'] = le.transform(df['smoking_status'])

le.fit(df['work_type'])
df['work_type'] = le.transform(df['work_type'])

le.fit(df['Residence_type'])
df['Residence_type'] = le.transform(df['Residence_type'])

le.fit(df['ever_married'])
df['ever_married'] = le.transform(df['ever_married'])

le.fit(df['gender'])
df['gender'] = le.transform(df['gender'])
```

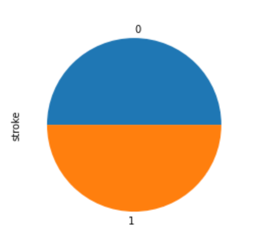
In [73]:
```python
#I did visualization of avg_glucose with other factors and did find much relation
df.drop(['avg_glucose_level','id'],axis=1,inplace=True)
```

resampling the lesser data

In [78]:
```python
from sklearn.utils import resample,shuffle

zero =df[df['stroke']==0]
one = df[df['stroke']==1]

upsampled1 = resample(one, replace=True, n_samples=zero.shape[0])

df = pd.concat([zero,upsampled1])

df = shuffle(df)
```

In [79]:
```python
df['stroke'].value_counts().plot(kind='pie')
```

Out[79]:
```
<AxesSubplot:ylabel='stroke'>
```



Model Building

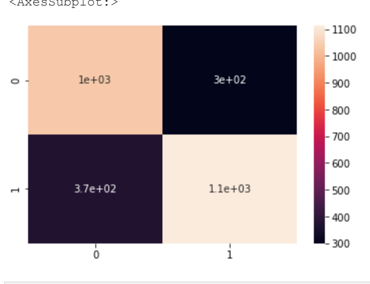
In [80]:
```python
x = df.drop(['stroke'],axis=1)
y = df['stroke']
```

In [84]:
```python
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.7,random_state=101)
```

1) Logistic Regression

In [86]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix

lg = LogisticRegression(max_iter=450)
lg.fit(x_train,y_train)
lg_predict = lg.predict(x_test)
lg_cm = confusion_matrix(lg_predict,y_test)
sns.heatmap(lg_cm,annot=True)
```

Out[86]:
```
<AxesSubplot:>
```



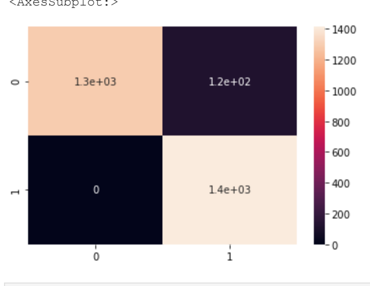
In [113]:
```python
accuracy_score(y_test,lg_predict)*100
```

Out[113]:
```
76.24113475177306
```

2)k-Nearest Neighbours

In [98]:
```python
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier(n_neighbors=4)
kn.fit(x_train,y_train)
kn_predict = kn.predict(x_test)
kn_cm = confusion_matrix(y_test,kn_predict)
sns.heatmap(kn_cm,annot=True)
```

Out[98]:
```
<AxesSubplot:>
```



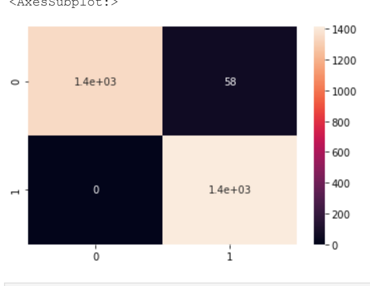
In [114]:
```python
accuracy_score(y_test,kn_predict)*100
```

Out[114]:
```
95.85106382978724
```

3) Decision Tree

In [100]:
```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt_predict = dt.predict(x_test)
dt_cm = confusion_matrix(y_test,dt_predict)
sns.heatmap(dt_cm,annot=True)
```

Out[100]:
```
<AxesSubplot:>
```



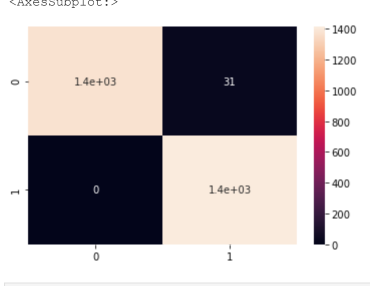
In [115]:
```python
accuracy_score(y_test,dt_predict)*100
```

Out[115]:
```
97.94326241134752
```

4) Random Forest

In [102]:
```python
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
rf_predict = rf.predict(x_test)
rf_cm = confusion_matrix(y_test,rf_predict)
sns.heatmap(rf_cm,annot=True)
```

Out[102]:
```
<AxesSubplot:>
```



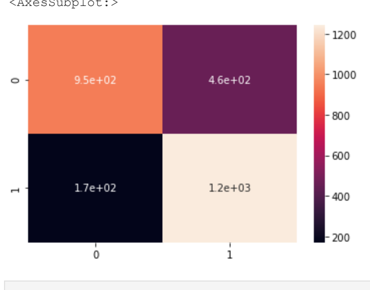
In [116]:
```python
accuracy_score(y_test,rf_predict)*100
```

Out[116]:
```
98.90070921985816
```

5) Suport vector machine

In [104]:
```python
from sklearn.svm import SVC

svm = SVC()
svm.fit(x_train,y_train)
svm_predict = svm.predict(x_test)
svm_cm = confusion_matrix(y_test,svm_predict)
sns.heatmap(svm_cm,annot=True)
```

Out[104]:
```
<AxesSubplot:>
```



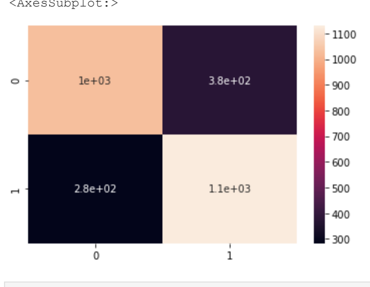
In [117]:
```python
accuracy_score(y_test,svm_predict)*100
```

Out[117]:
```
77.6595744680851
```

6) Naive Bayes

In [106]:
```python
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(x_train,y_train)
nb_predict = nb.predict(x_test)
sns.heatmap(confusion_matrix(y_test,nb_predict),annot=True)
```

Out[106]:
```
<AxesSubplot:>
```



In [118]:
```python
accuracy_score(y_test,nb_predict)*100
```

Out[118]:
```
76.66666666666667
```

Among all the algorithms we used Random Forest outstanded all other algorithms with accuracy of 98.90%

In [ ]: