Brain-stroke-prediction

December 15, 2022

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
df = pd.read_csv('/content/brain_stroke.csv')
df.head()
```

[2]:

	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

[3]:
```
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
```

[4]:
```
df.shape
```

[4]:
```
(4981, 11)
```

[5]:
```
df.describe()
```

[5]:
```
                age  hypertension  heart_disease  avg_glucose_level  \
count  4981.000000   4981.000000    4981.000000        4981.000000
mean     43.419859      0.096165       0.055210         105.943562
std      22.662755      0.294848       0.228412          45.075373
min       0.080000      0.000000       0.000000          55.120000
25%      25.000000      0.000000       0.000000          77.230000
50%      45.000000      0.000000       0.000000          91.850000
75%      61.000000      0.000000       0.000000         113.860000
max      82.000000      1.000000       1.000000         271.740000

               bmi       stroke
count  4981.000000  4981.000000
mean     28.498173     0.049789
std       6.790464     0.217531
min      14.000000     0.000000
25%      23.700000     0.000000
50%      28.100000     0.000000
75%      32.600000     0.000000
max      48.900000     1.000000
```

[6]:
```
df.tail()
```

[6]:
```
      gender   age  hypertension  heart_disease ever_married work_type  \
4976    Male  41.0             0              0           No   Private
4977    Male  40.0             0              0          Yes   Private
4978  Female  45.0             1              0          Yes  Govt_job
4979    Male  40.0             0              0          Yes   Private
4980  Female  80.0             1              0          Yes   Private

     Residence_type  avg_glucose_level   bmi   smoking_status  stroke
4976          Rural              70.15  29.8  formerly smoked       0
4977          Urban             191.15  31.1           smokes       0
4978          Rural              95.02  31.8           smokes       0
4979          Rural              83.94  30.0           smokes       0
```

```
4980          Urban              83.75  29.1     never smoked       0
```

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

[7]:
```
gender                 2
age                  104
hypertension           2
heart_disease          2
ever_married           2
work_type              4
Residence_type         2
avg_glucose_level    3895
bmi                   342
smoking_status         4
stroke                 2
dtype: int64
```

[8]:
```
df_ = df.drop(['age','avg_glucose_level','bmi'], axis = 1)
for i in df_.columns:
    print(df_[i].nunique())
```

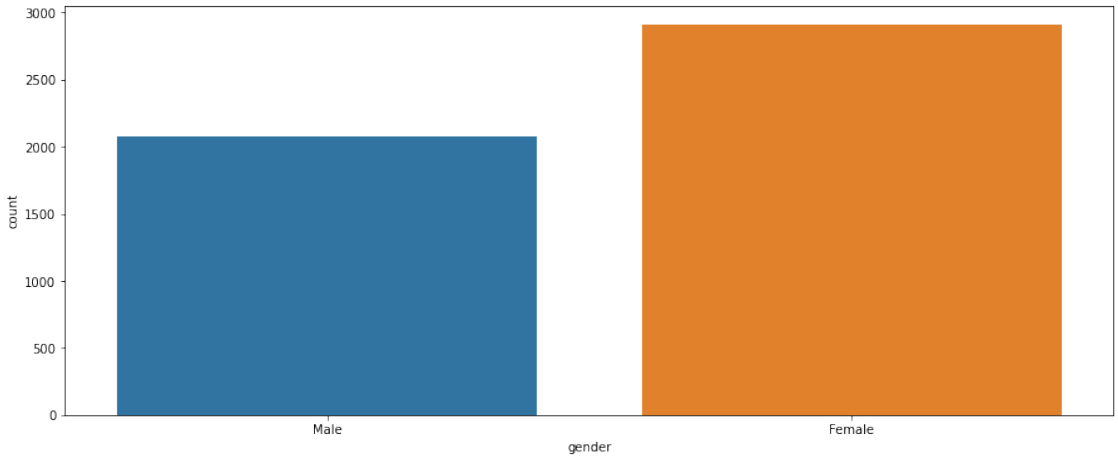
```
2
2
2
2
4
2
4
2
```

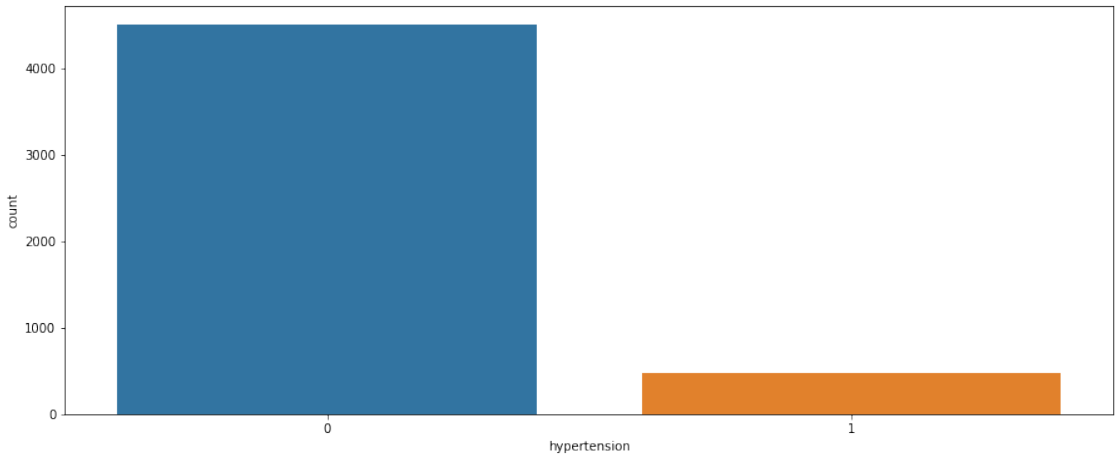
[9]:
```
for i in df_.columns:
    print(df_[i].value_counts())
```

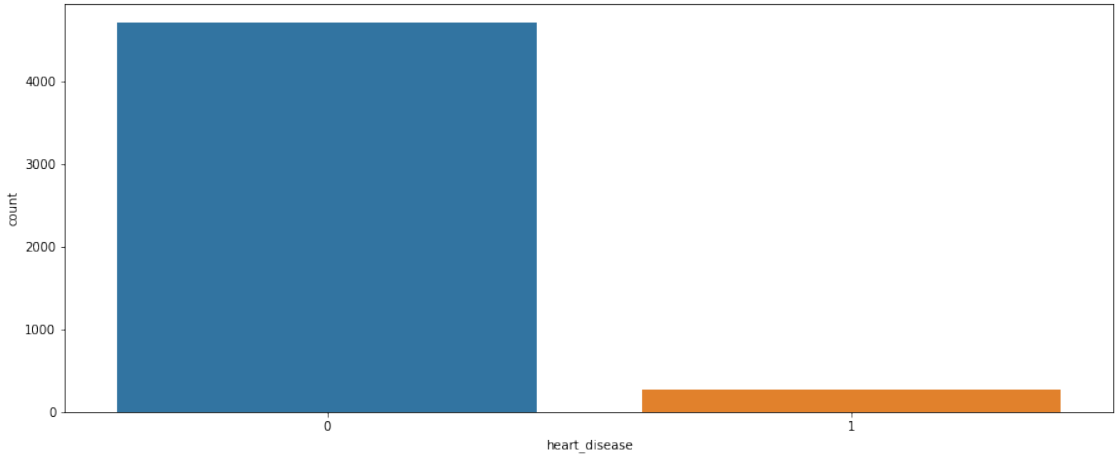
```
Female    2907
Male      2074
Name: gender, dtype: int64
0    4502
1     479
Name: hypertension, dtype: int64
0    4706
1     275
Name: heart_disease, dtype: int64
Yes    3280
No     1701
Name: ever_married, dtype: int64
Private          2860
Self-employed     804
children          673
```

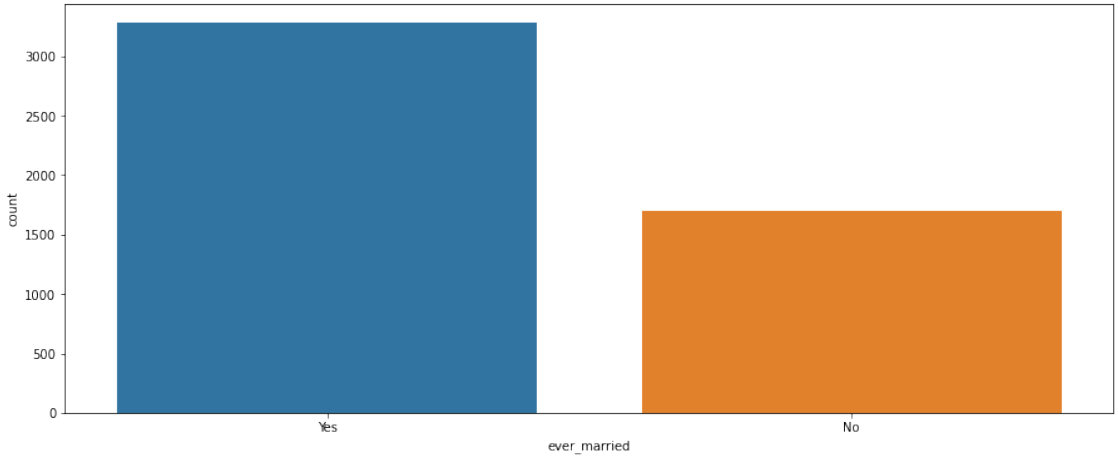
```
Govt_job          644
Name: work_type, dtype: int64
Urban    2532
Rural    2449
Name: Residence_type, dtype: int64
never smoked       1838
Unknown            1500
formerly smoked     867
smokes              776
Name: smoking_status, dtype: int64
0    4733
1     248
Name: stroke, dtype: int64
```

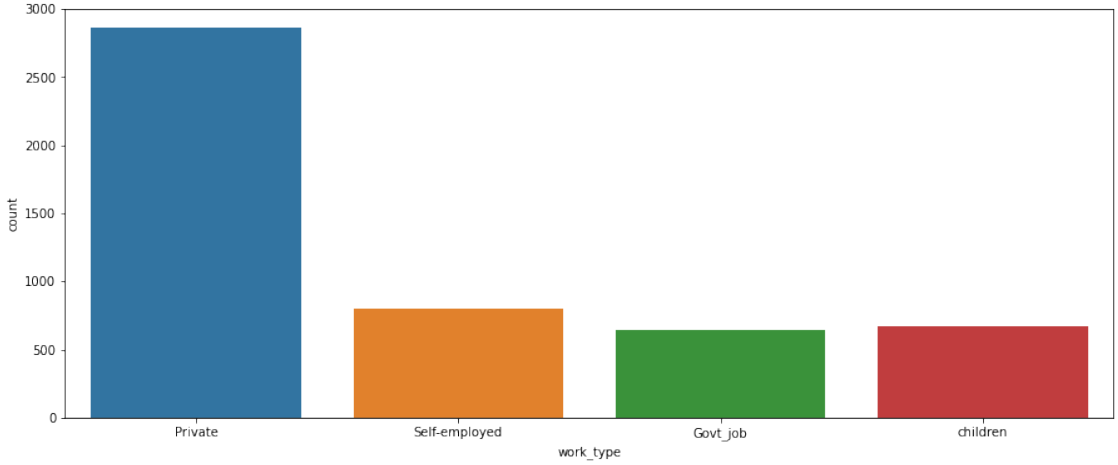
[10]:
```
import seaborn as sns
for i in df_.columns:
    plt.figure(figsize=(15,6))
    sns.countplot(df_[i], data=df_)
```

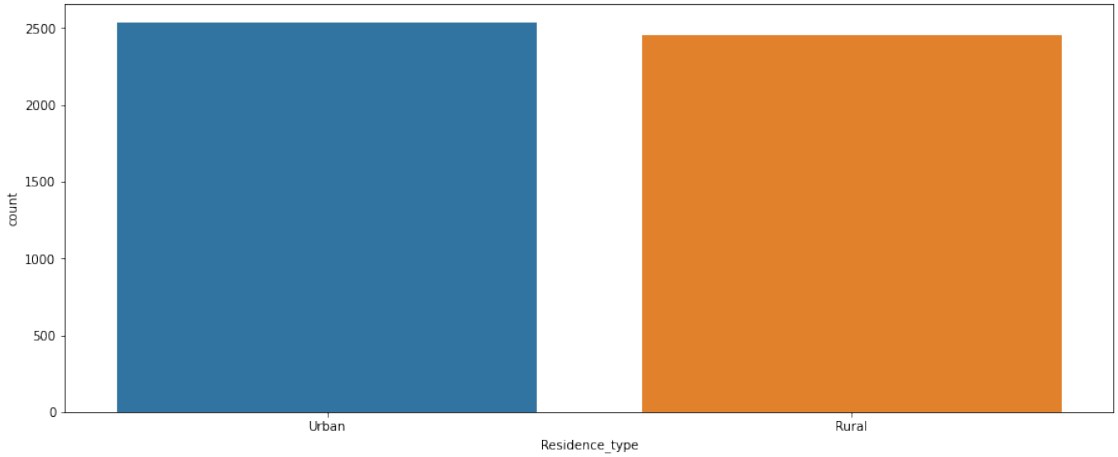


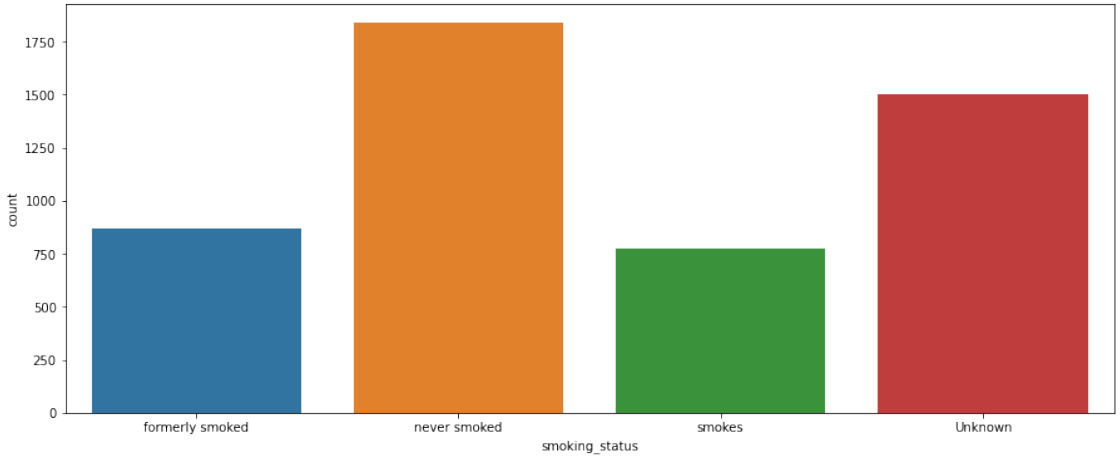


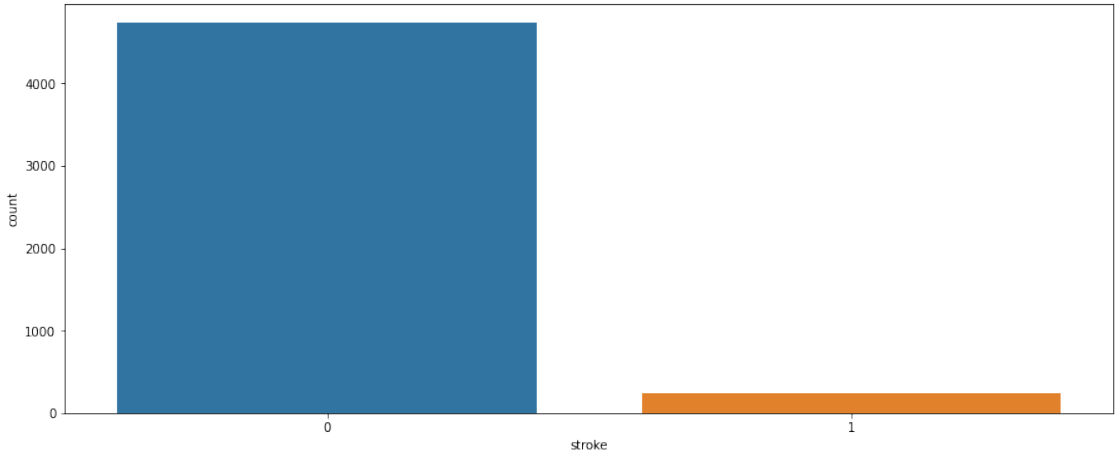

4000
3000
2000
1000
0
count
0
1
heart_disease


3000
2500
2000
1500
1000
500
0
count
Yes
No
ever_married

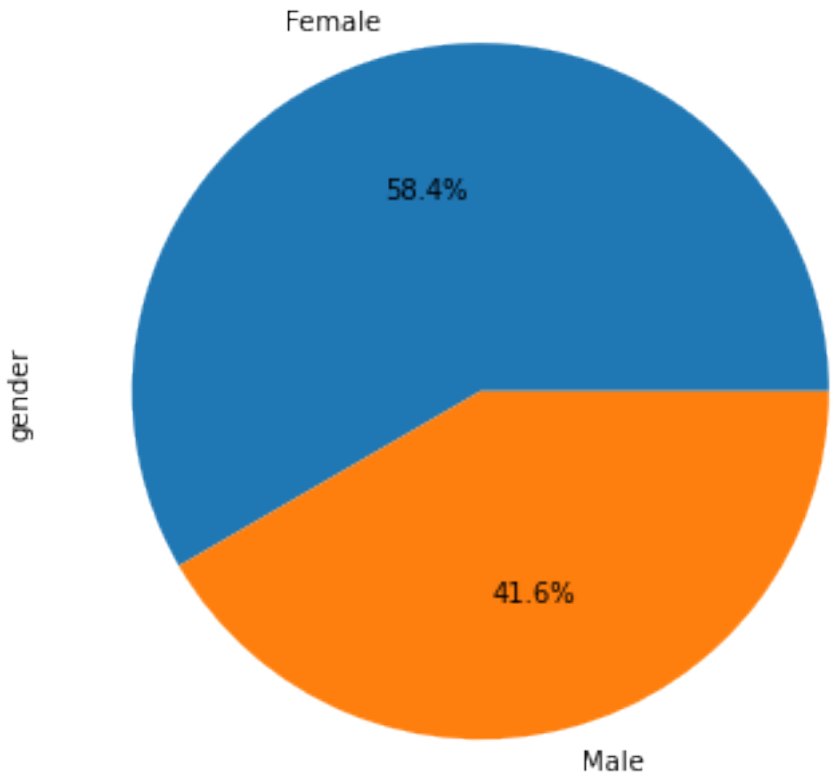

3000
2500
2000
1500
1000
500
0
count
Private
Self-employed
Govt_job
children
work_type

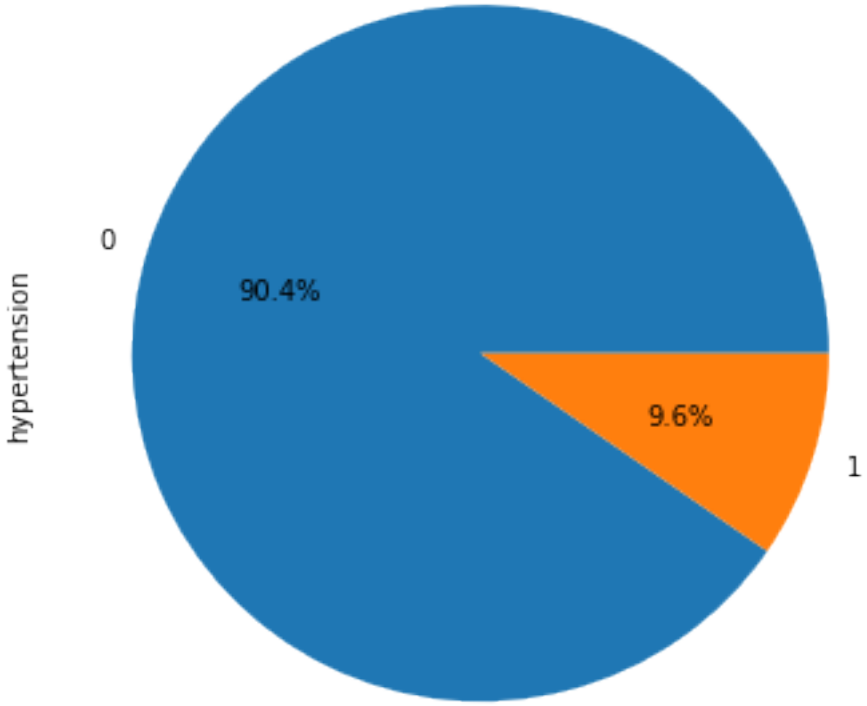

2500
2000
1500
1000
500
0
count
Urban
Rural
Residence_type

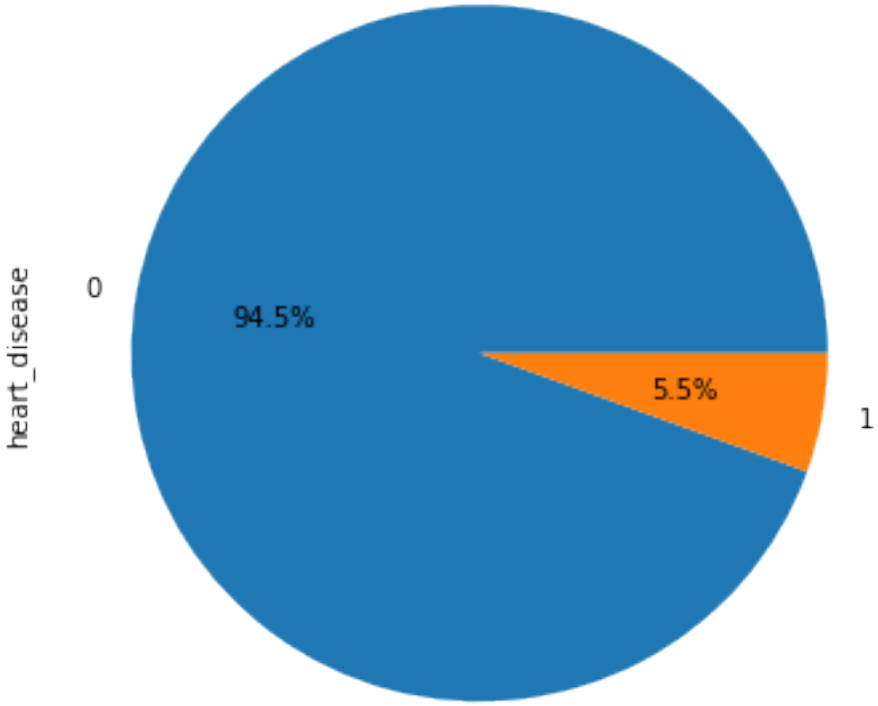

1750
1500
1250
1000
750
500
250
0
count
formerly smoked
never smoked
smokes
Unknown
smoking_status

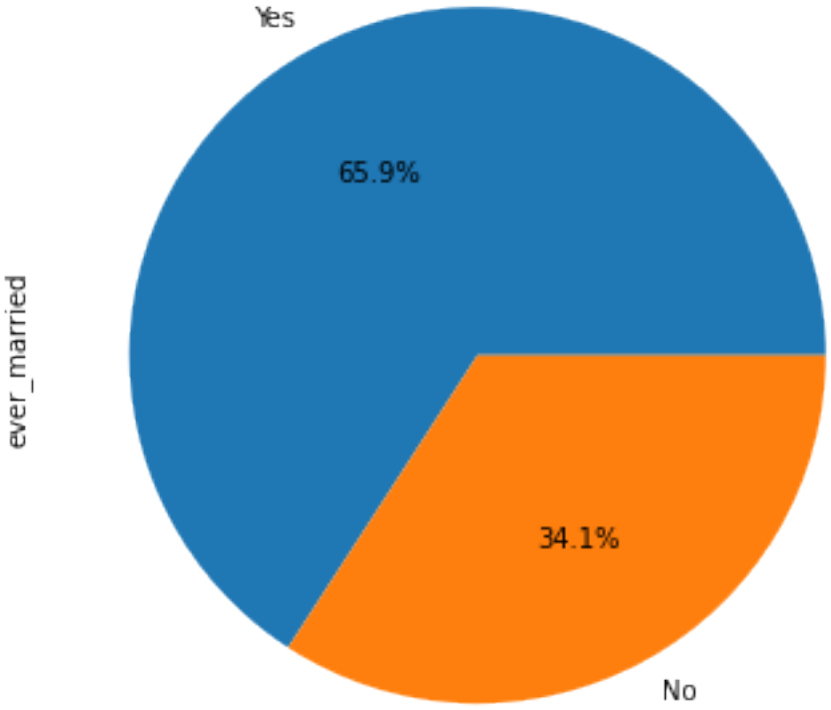

4000
3000
2000
1000
0
count
0
1
stroke

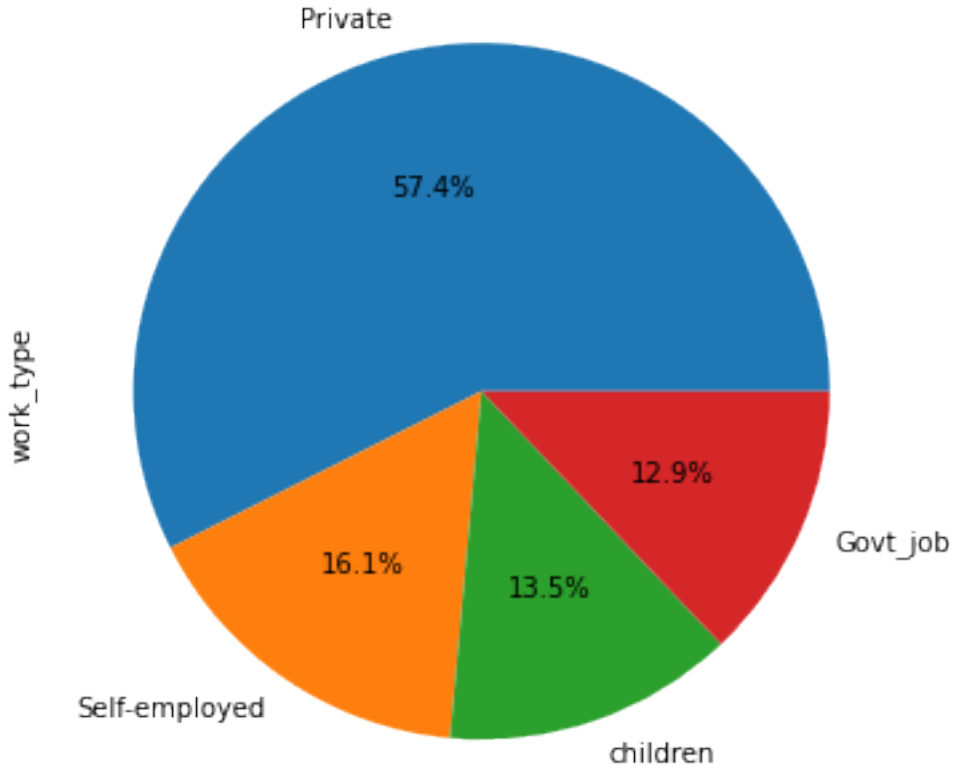
[11]:
```
for i in df_.columns:
    plt.figure(figsize=(15,6))
    df_[i].value_counts().plot(kind='pie',autopct='%1.1f%%')
```

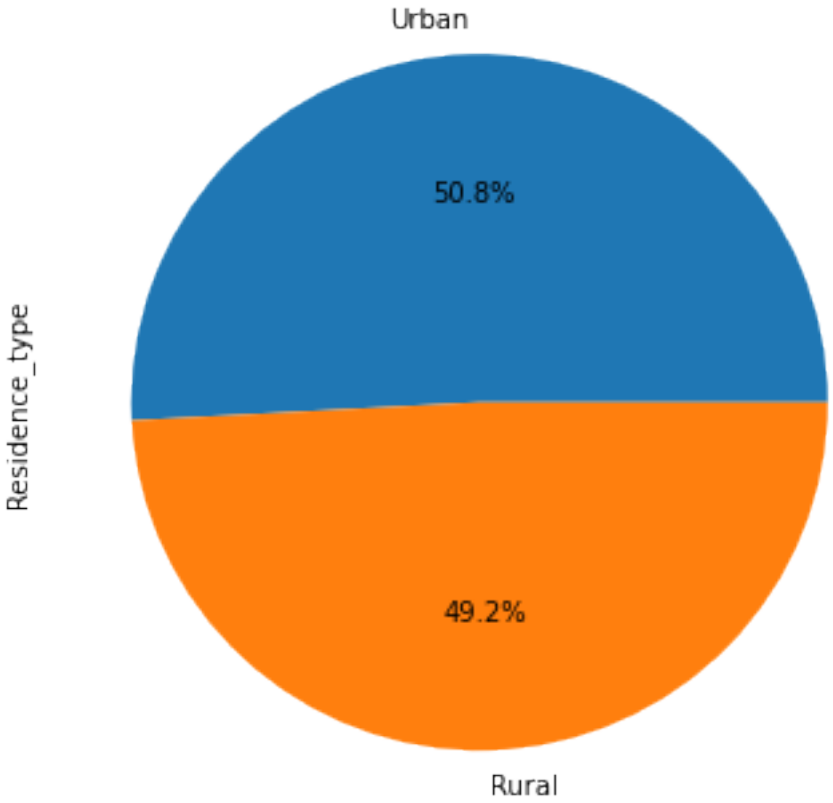


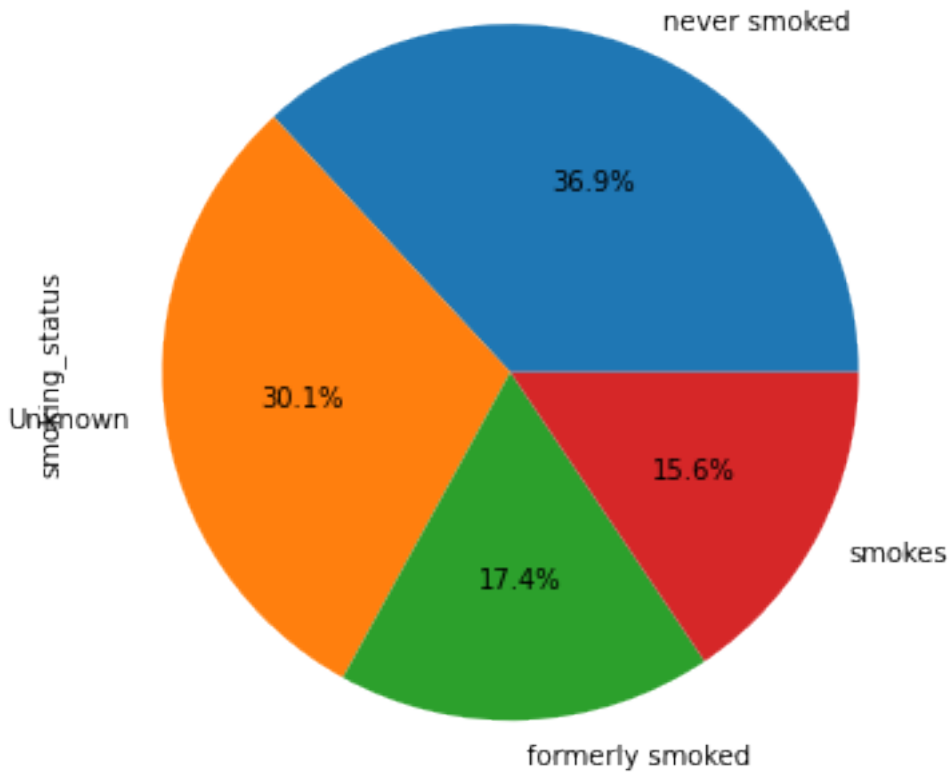

hypertension
0
90.4%
9.6%
1

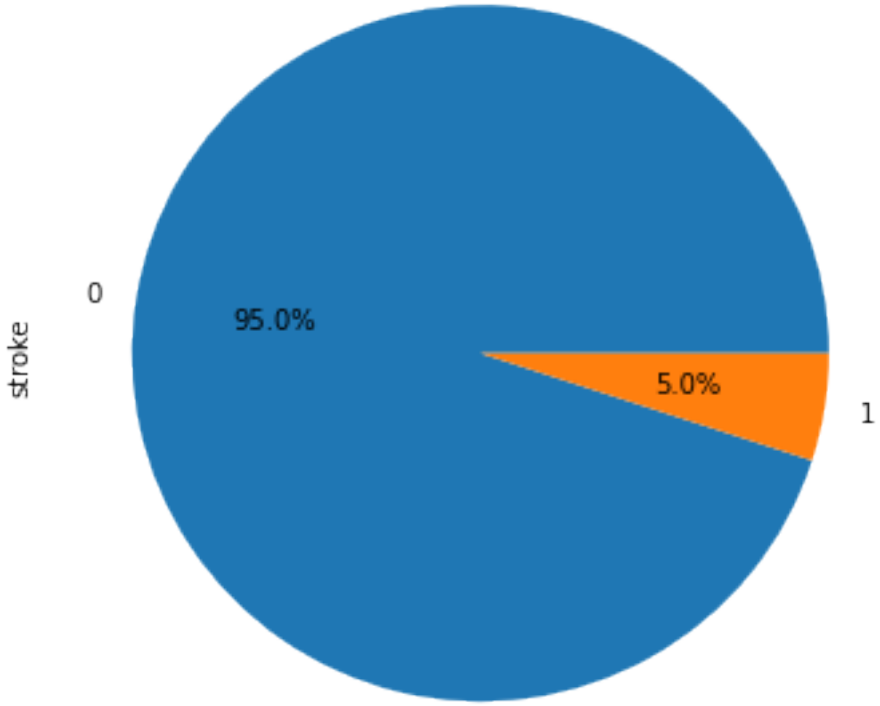

heart_disease
0
94.5%
5.5%
1


Yes
65.9%
ever_married
34.1%
No

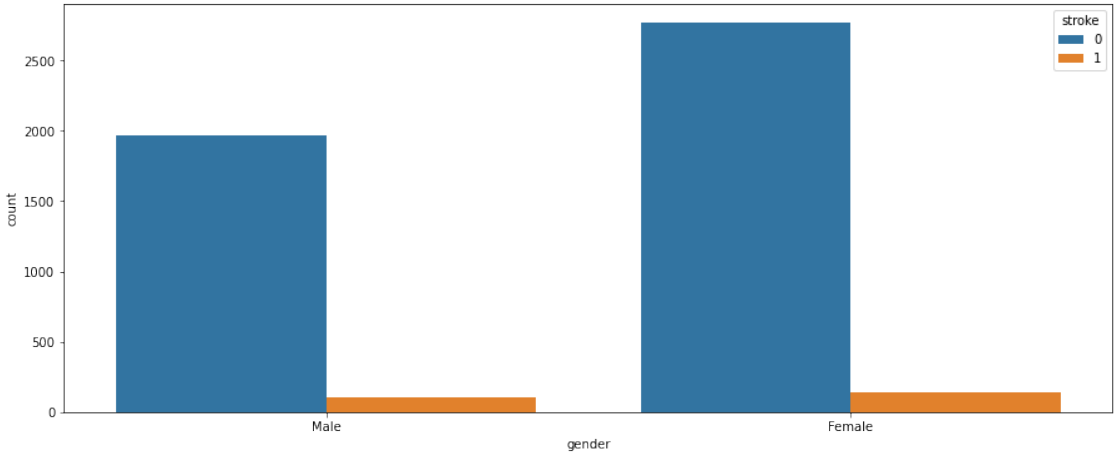

Private
57.4%
work_type
12.9%
Govt_job
16.1%
13.5%
Self-employed
children

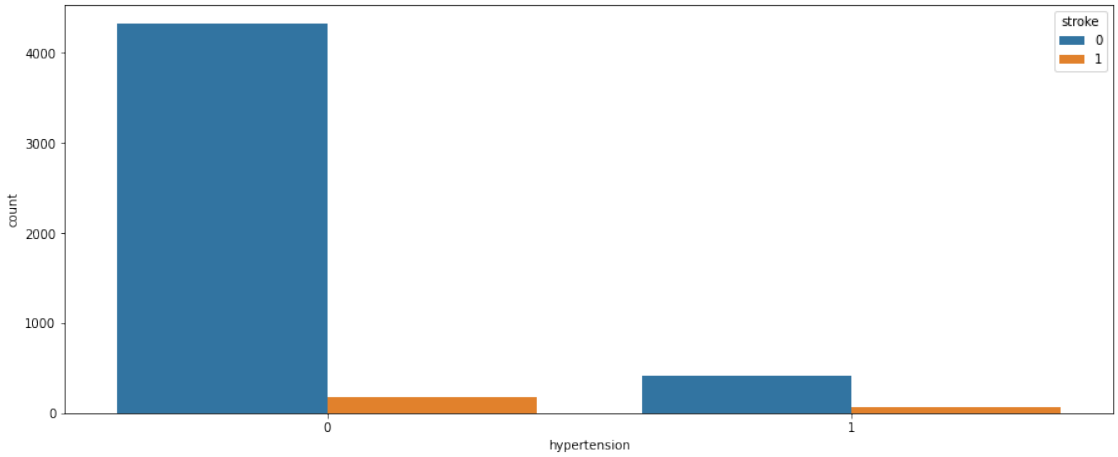

Urban
50.8%
Residence_type
49.2%
Rural

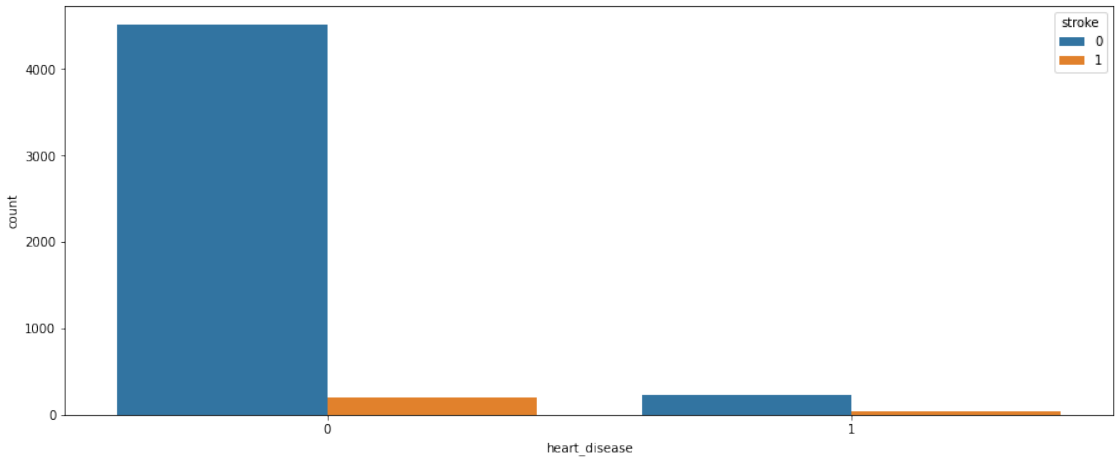

never smoked
36.9%
smoking_status
Unknown
30.1%
15.6%
smokes
17.4%
formerly smoked

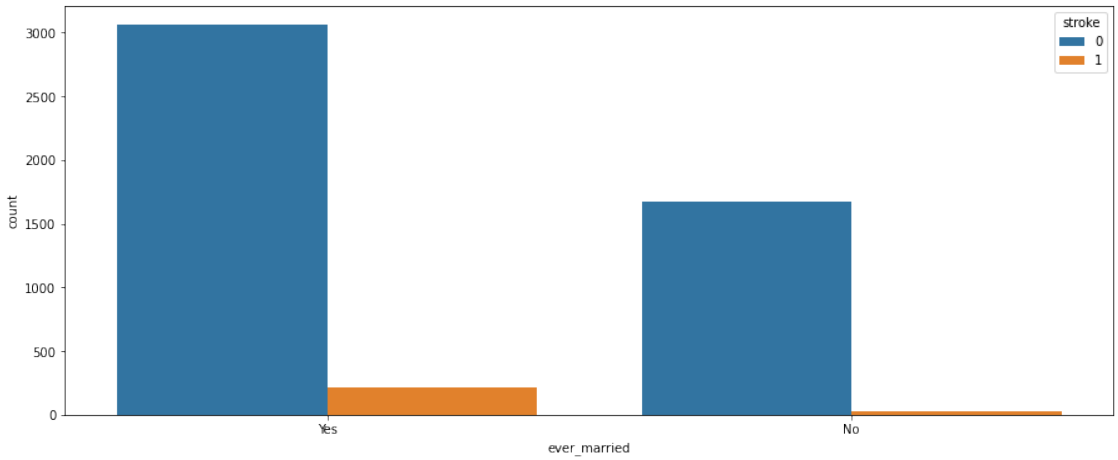


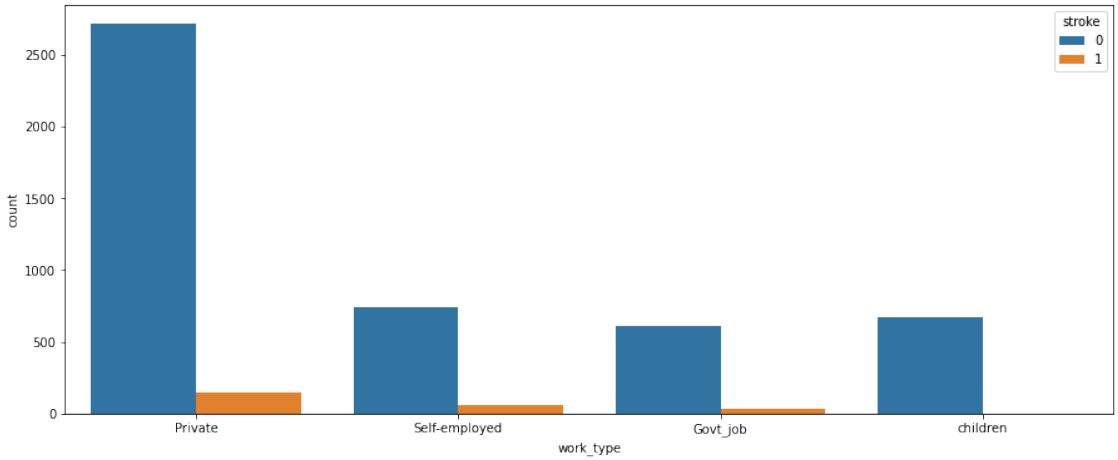
[12]:
```
for i in df_.columns:
    plt.figure(figsize=(15,6))
    sns.countplot(df_[i], data=df_,hue='stroke')
```

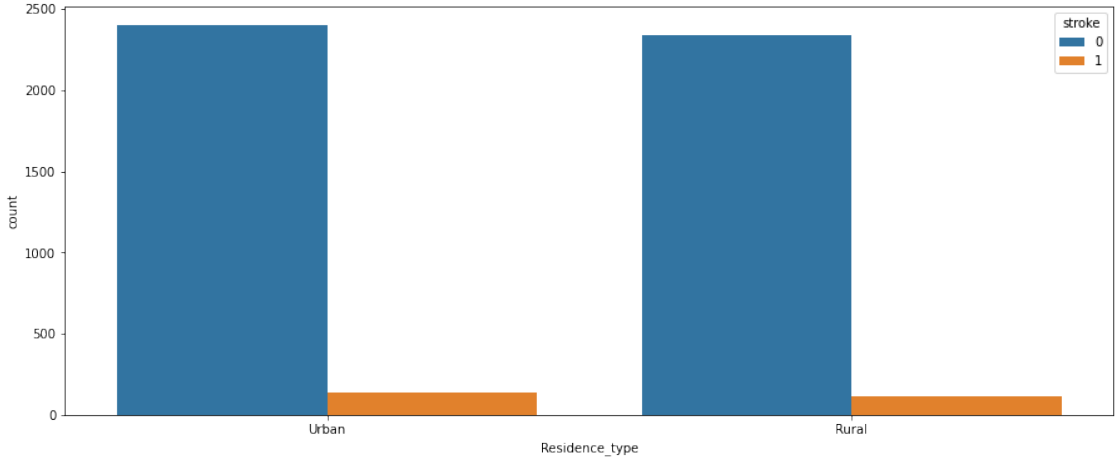


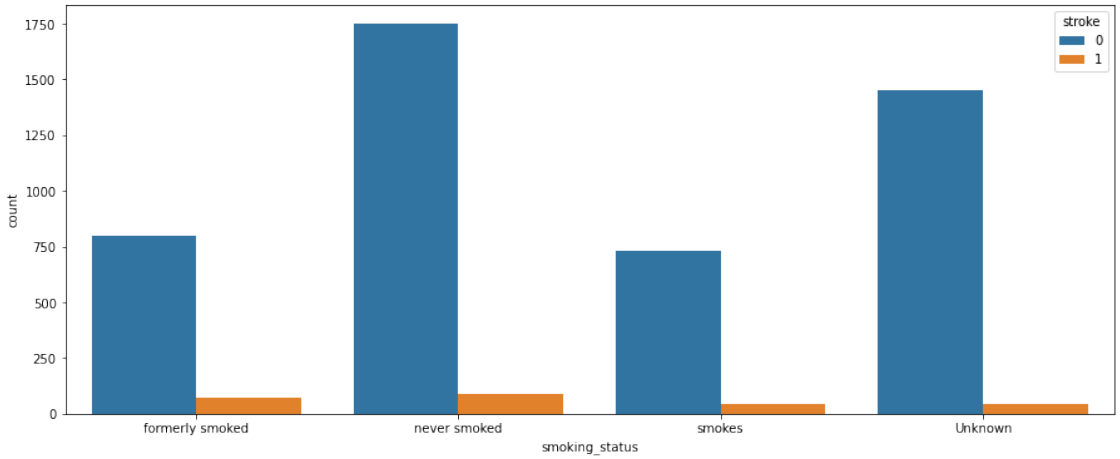

stroke
0
1
count
4000
3000
2000
1000
0
0
1
hypertension

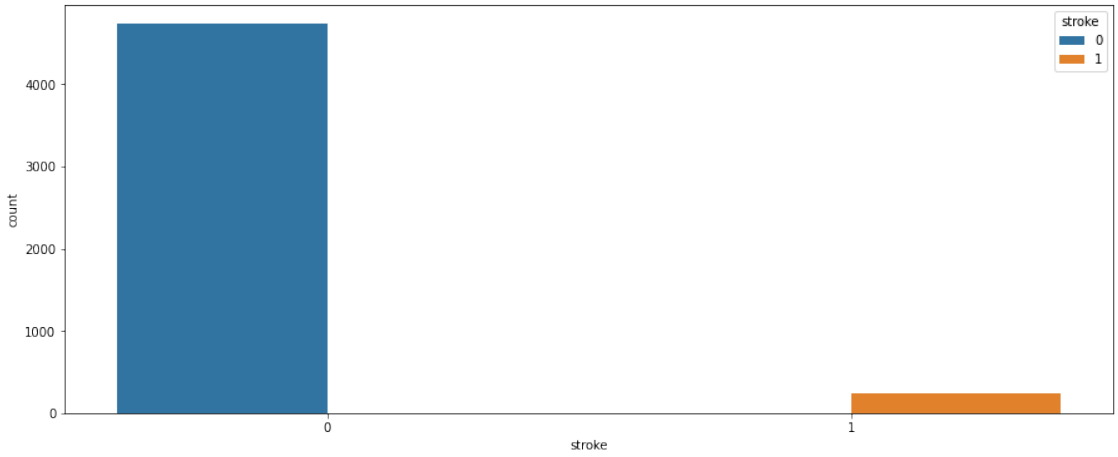

stroke
0
1
count
4000
3000
2000
1000
0
0
1
heart_disease


stroke
0
1
count
3000
2500
2000
1500
1000
500
0
Yes
No
ever_married


stroke
0
1
count
2500
2000
1500
1000
500
0
Private
Self-employed
Govt_job
children
work_type


stroke
0
1
count
2500
2000
1500
1000
500
0
Urban
Rural
Residence_type

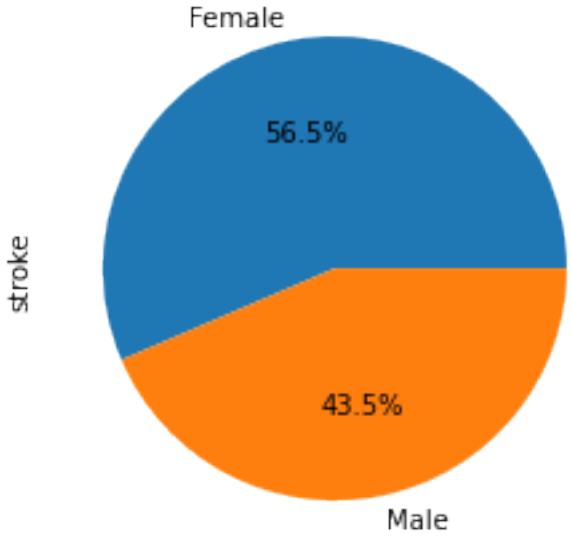

stroke
0
1
count
1750
1500
1250
1000
750
500
250
0
formerly smoked
never smoked
smokes
Unknown
smoking_status



0.0.1 Ratio of people diagnosed with stroke based on Gender

[13]:
```
Gender = df.groupby(['gender'])['stroke'].sum()
Gender.plot(kind='pie',autopct='%1.1f%%')
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17feb70a30>

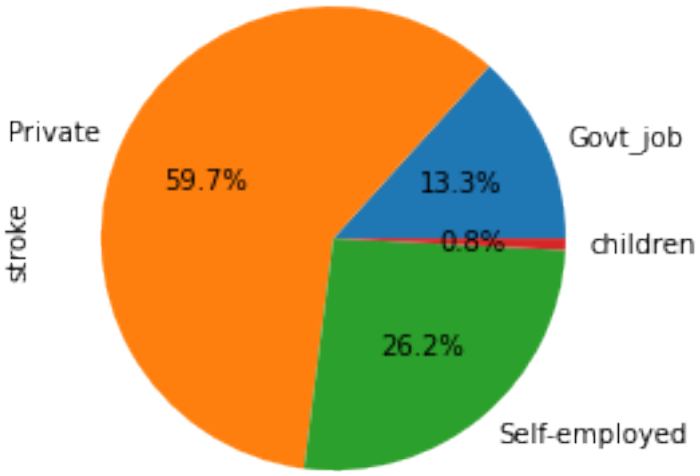


This shows that female are more diagnosed with stroke with 56.5% than male of 43.5%

0.0.2 People who has stroke based on type of work they do. Private, Government, Self-employed et Children

[14]:
```
workType = df.groupby(['work_type'])['stroke'].sum()
workType.plot(kind='pie',autopct='%1.1f%%')
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17feaebe80>
```

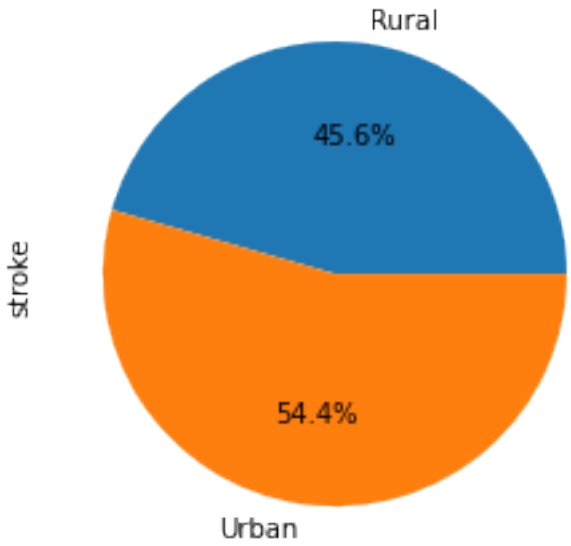


People who work in private seconds have increase proportion of being diagnosed with stroke (59.7%)

0.0.3 Proportion based on Residence

[15]:
```
Residence = df.groupby(['Residence_type'])['stroke'].sum()
Residence.plot(kind='pie',autopct='%1.1f%%')
```

[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17feab7be0>
```

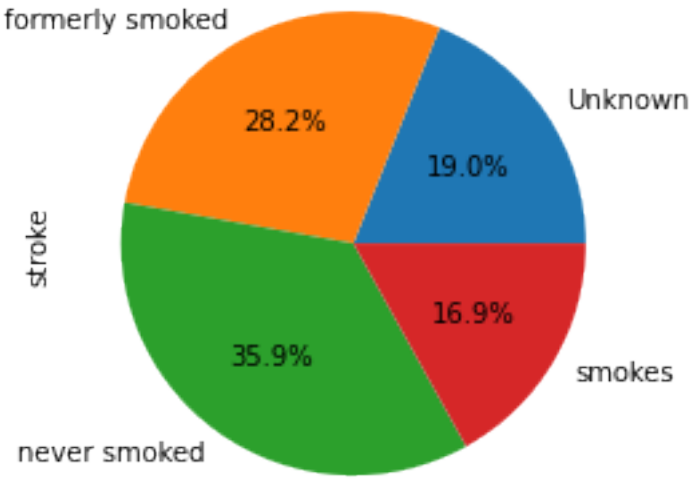


This shows that people residing in Urban area have high proportion of being diagnosed with stroke.

0.0.4 Smoking status proportion

[16]:
```
Smoke = df.groupby(['smoking_status'])['stroke'].sum()
Smoke.plot(kind='pie',autopct='%1.1f%%')
```

[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17fea070d0>



0.0.5 Dummies Variable

df = df.get_dummies(df,columns=['work_type','Resident_type','smoking_status'])
Instead of using dummies, I will use Label encoder

[17]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['gender'] = encoder.fit_transform(df['gender'])
df['Residence_type'] = encoder.fit_transform(df['Residence_type'])
df['work_type'] = encoder.fit_transform(df['work_type'])
df['smoking_status'] = encoder.fit_transform(df['smoking_status'])
df['ever_married'] = encoder.fit_transform(df['ever_married'])
```

[18]:
```
pd.DataFrame(df.head())
```

[18]:

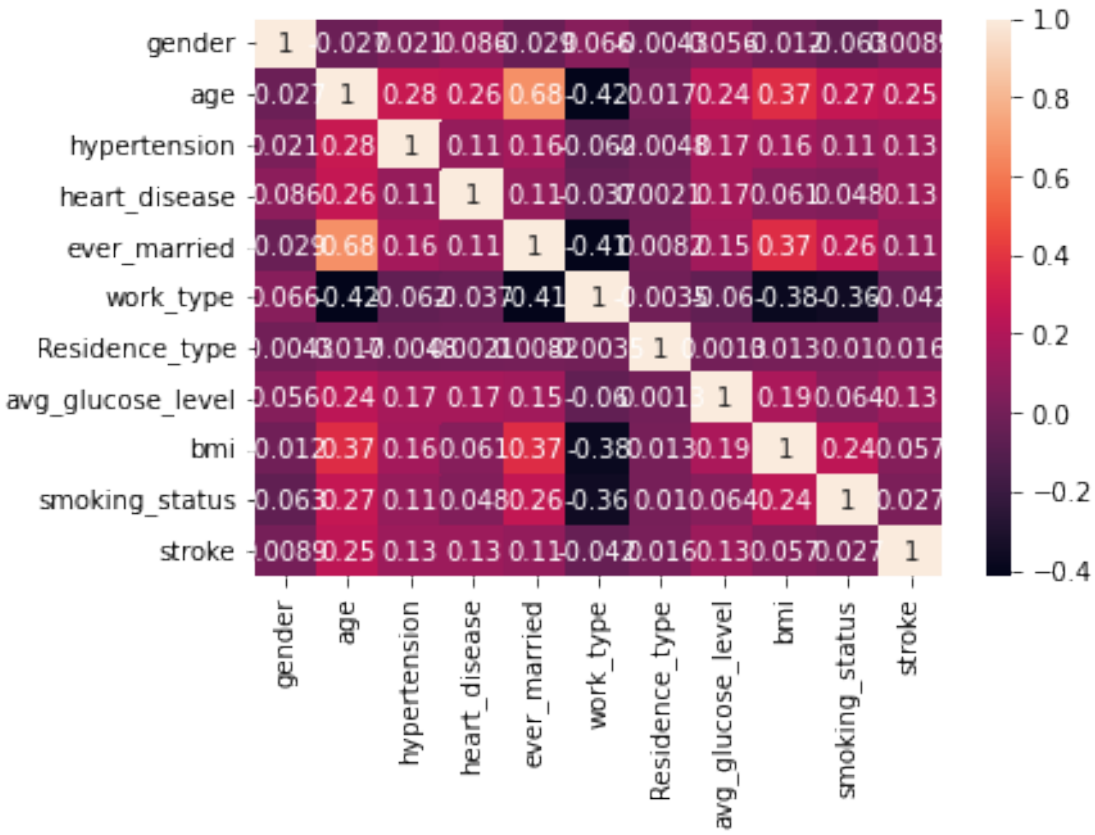
	gender	age	hypertension	heart_disease	ever_married	work_type	\
0	1	67.0	0	1	1	1	
1	1	80.0	0	1	1	1	
2	0	49.0	0	0	1	1	
3	0	79.0	1	0	1	2	
4	1	81.0	0	0	1	1	

	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	228.69	36.6	1	1
1	0	105.92	32.5	2	1
2	1	171.23	34.4	3	1
3	0	174.12	24.0	2	1
4	1	186.21	29.0	1	1

0.0.6 Correlation between the Variables

[19]:
```
corr = df.corr()
sns.heatmap(corr, annot=True)
```

[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f18110fae20>
```



0.1 Training the Model

[20]:
```
#Split the data into dependent and independent variable
X = df.drop(['stroke'], axis=1)
y = df['stroke']
```

Training and Testing data

[26]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0,test_size=0.
↪2)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(3984, 10)
(3984,)
(997, 10)
(997,)
```

[27]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

[27]:
```
LogisticRegression()
```

[29]:
```
y_pred = lr.predict(X_test)
pd.DataFrame(y_pred)
```

[29]:
```
     0
0    0
1    0
2    0
3    0
4    0
..  ..
992  0
993  0
994  0
995  0
996  0

[997 rows x 1 columns]
```

[30]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

[30]:
```
array([[946,   1],
       [ 49,   1]])
```

[38]:
```
#Accuracy
print('Training accuracy: ',lr.score(X_train,y_train))
print('Training accuracy: ',lr.score(X_test,y_test))
```

```
Training accuracy:  0.9503012048192772
Training accuracy:  0.9498495486459378
```

[31]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train,y_train)
```

[31]:
```
DecisionTreeClassifier()
```

[32]:
```
dtc_pred = dtc.predict(X_test)
pd.DataFrame(dtc_pred)
```

[32]:
```
      0
0     0
1     1
2     0
3     0
4     0
..   ..
992   0
993   0
994   1
995   0
996   0

[997 rows x 1 columns]
```

[35]:
```
cm = confusion_matrix(y_test,dtc_pred)
cm
```

[35]:
```
array([[893,  54],
       [ 43,   7]])
```

[39]:
```
print('Training accuracy: ',dtc.score(X_train,y_train))
print('Training accuracy: ',dtc.score(X_test,y_test))
```

```
Training accuracy:  1.0
Training accuracy:  0.9027081243731193
```

[ ]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```