In [462]:

```
# Importing required packages
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.svm import SVC

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plotPointer
import numpy as np
import seaborn as sns
```

In [463]:

```
strokesData = read_csv('F:\ML\For Stroke Risk Dataset\healthcare-dataset-stroke-data.csv')
# strokesData.head()
strokesData
```

Out[463]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

5110 rows × 12 columns

In [464]:

```
strokesData = strokesData.dropna()
```

In [465]:

```
if 'id' in strokesData :
    strokesData = strokesData.drop('id', axis=1)

strokesData
```

Out[465]:

	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
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
5104	Female	13.0	0	0	No	children	Rural	103.08	18.6	Unknown	0
5106	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

4909 rows × 11 columns

In [466]:

```
# Checking Unique Values for some columns

# Marital Status
strokesData.ever_married.unique()
```

Out[466]:

```
array(['Yes', 'No'], dtype=object)
```

In [467]:

```
# Work Type
strokesData.work_type.unique()
```

Out[467]:

```
array(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_worked'],
      dtype=object)
```

In [468]:

```
# Residence Urban/Rural
strokesData.Residence_type.unique()
```

Out[468]:

```
array(['Urban', 'Rural'], dtype=object)
```

In [469]:

```
# Smoking Status
strokesData.smoking_status.unique()
```

Out[469]:

```
array(['formerly smoked', 'never smoked', 'smokes', 'Unknown'],
      dtype=object)
```

In [470]:

```
# Gender
strokesData.gender.unique()
```

Out[470]:

```
array(['Male', 'Female', 'Other'], dtype=object)
```

In [471]:

```
#  Spliting the data in training and testing subsets
D = strokesData.values
# iloc to select specefic rows
x = strokesData.iloc[:,:-1]
y = strokesData.iloc[:,-1]
# x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.50, random_state = 1)
```

In [472]:

```
transformingColoumn = ColumnTransformer(transformers = [('encoder', OneHotEncoder(), [0,4,5,6,9])], remainder='passthrough')
x = np.array(transformingColoumn.fit_transform(x))
x

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.50, random_state = 1)
```

In [473]:

```
sc = StandardScaler()
x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.fit_transform(x_test)
```

In [474]:

```
# SVM
modelSVM = SVC(random_state = 0, kernel = 'linear')
modelSVM.fit(x_train_scaled, y_train)
y_prediction_SVM = modelSVM.predict(x_test_scaled)

accOfSVM = accuracy_score(y_test, y_prediction_SVM)
accOfSVM

print("----------------------------------")
print('The accuracy of the SVM for Testing is: {}'.format(accOfSVM.round(4)))
print("----------------------------------")
# save the accuracy score
score = set()
score.add(('SVM', accOfSVM.round(4)))
```

```
----------------------------------
The accuracy of the SVM for Testing is: 0.9605
----------------------------------
```

In [475]:

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm
modelDT = DecisionTreeClassifier(criterion = 'entropy')
modelDT.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_DT = modelDT.predict(x_test_scaled)

accOfDT = accuracy_score(y_test, y_prediction_DT)
accOfDT


print("----------------------------------")
print('The accuracy of the DT for Testing is: {}'.format(accOfDT.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('DT', accOfDT.round(4)))
```

```
----------------------------------
The accuracy of the DT for Testing is: 0.9218
----------------------------------
```

In [476]:

```
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
modelNB = GaussianNB()
modelNB.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_NB = modelNB.predict(x_test_scaled)

accOfNB = accuracy_score(y_prediction_NB, y_test)
accOfNB


print("----------------------------------")
print('The accuracy of the NB for Testing is: {}'.format(accOfNB.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('NB', accOfNB.round(4)))
```

```
----------------------------------
The accuracy of the NB for Testing is: 0.9605
----------------------------------
```

In [477]:

```
# Linear Regression
from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_LR = LogisticRegression()
model_LR.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_LR = model_LR.predict(x_test_scaled) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
accOfLR = accuracy_score(y_prediction_LR, y_test)
accOfLR


print("----------------------------------")
print('The accuracy of the LR for Testing is: {}'.format(accOfLR.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('LR', accOfLR.round(4)))
```

```
----------------------------------
The accuracy of the LR for Testing is: 0.9605
----------------------------------
```

In [478]:

```
# KNN
from scipy import stats
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
#from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_KNN = KNeighborsClassifier(n_neighbors=3) # 3 neighbours for putting the new data into a class
model_KNN.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_KNN = model_KNN.predict(x_test_scaled)

accOfKNN = accuracy_score(y_prediction_KNN, y_test)
accOfKNN

print("---------------------------------")
print('The accuracy of the KNN for Testing is: {}'.format(accOfKNN.round(4)))
print("---------------------------------")
# save the accuracy score
score.add(('KNN', accOfKNN.round(4)))
```

```
---------------------------------
The accuracy of the KNN for Testing is: 0.9556
---------------------------------

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.
11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is
taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [479]:

```
print("The accuracy scores of different Models:")
print("----------------------------------------")
for s in score:
    print(s)
```

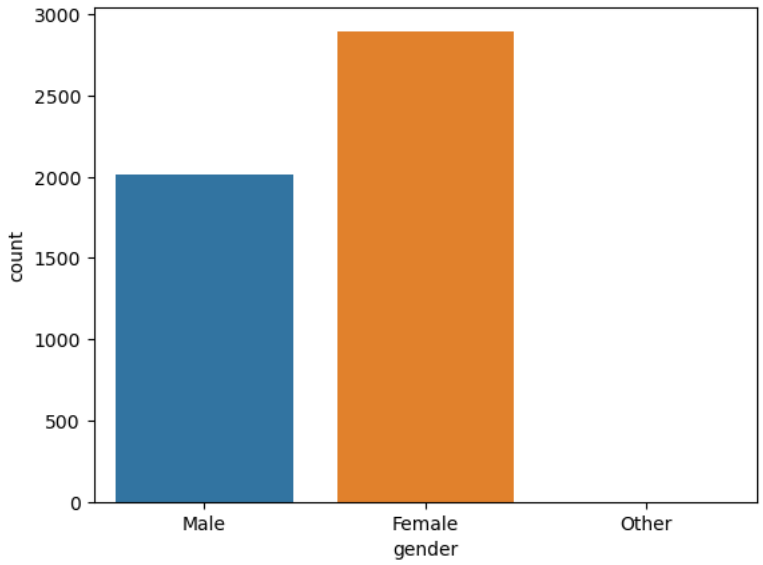
```
The accuracy scores of different Models:
----------------------------------------
('KNN', 0.9556)
('SVM', 0.9605)
('NB', 0.9605)
('LR', 0.9605)
('DT', 0.9218)
```

In [480]:

```
sns.countplot(x = strokesData['gender'])
```

Out[480]:

```
<AxesSubplot:xlabel='gender', ylabel='count'>
```

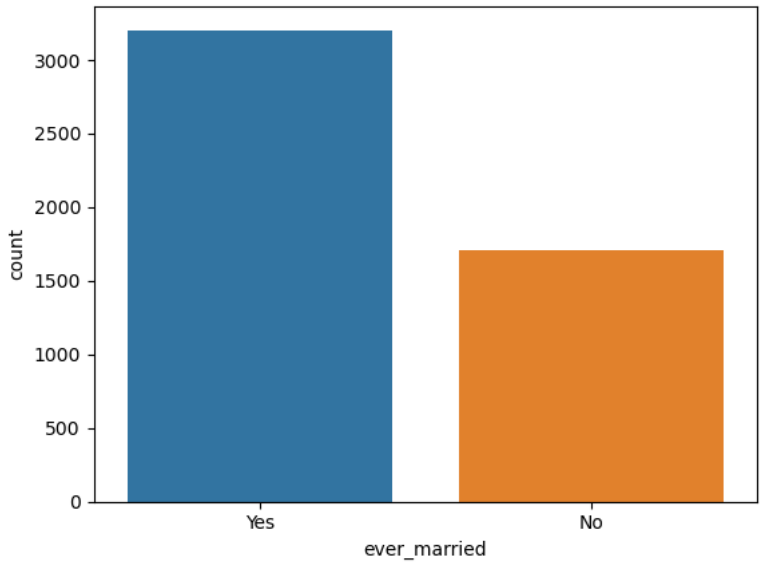


In [481]:

```
sns.countplot(x = strokesData['ever_married'])
```

Out[481]:

```
<AxesSubplot:xlabel='ever_married', ylabel='count'>
```

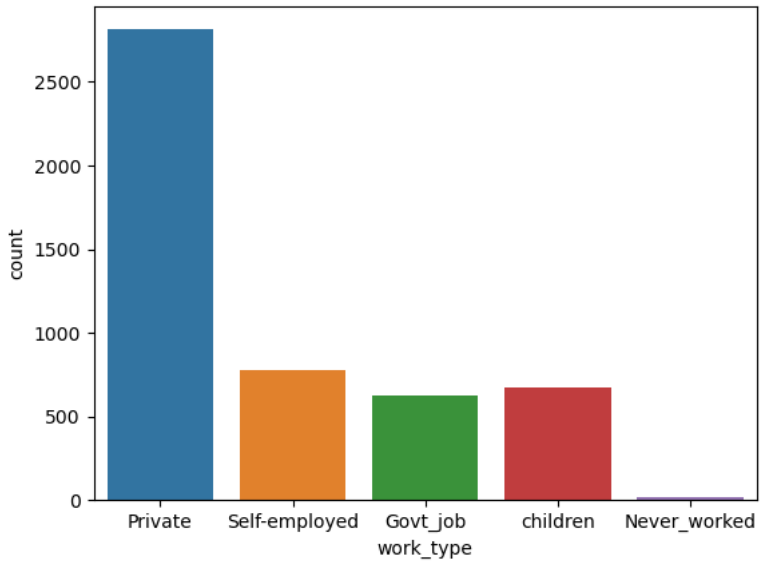


In [482]:

```
sns.countplot(x = strokesData['work_type'])
```

Out[482]:

```
<AxesSubplot:xlabel='work_type', ylabel='count'>
```

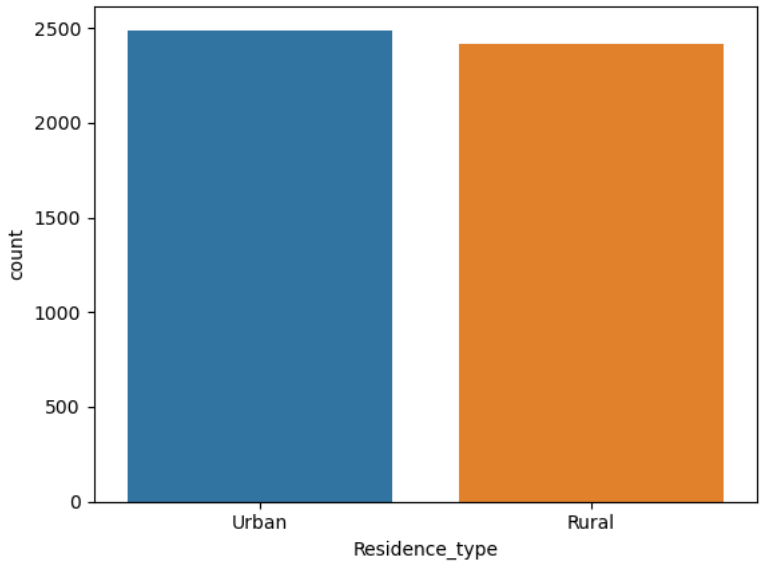


In [483]:

```
sns.countplot(x = strokesData['Residence_type'])
```

Out[483]:

```
<AxesSubplot:xlabel='Residence_type', ylabel='count'>
```

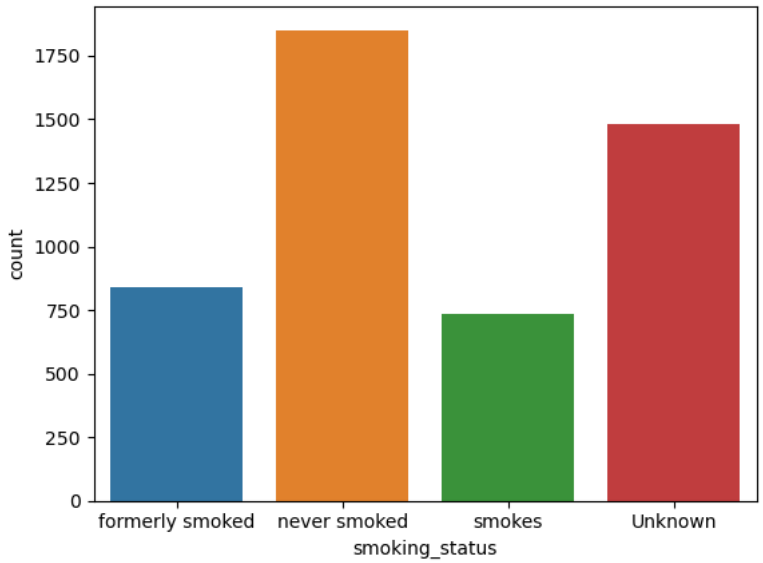


In [484]:

```
sns.countplot(x = strokesData['smoking_status'])
```

Out[484]:

```
<AxesSubplot:xlabel='smoking_status', ylabel='count'>
```

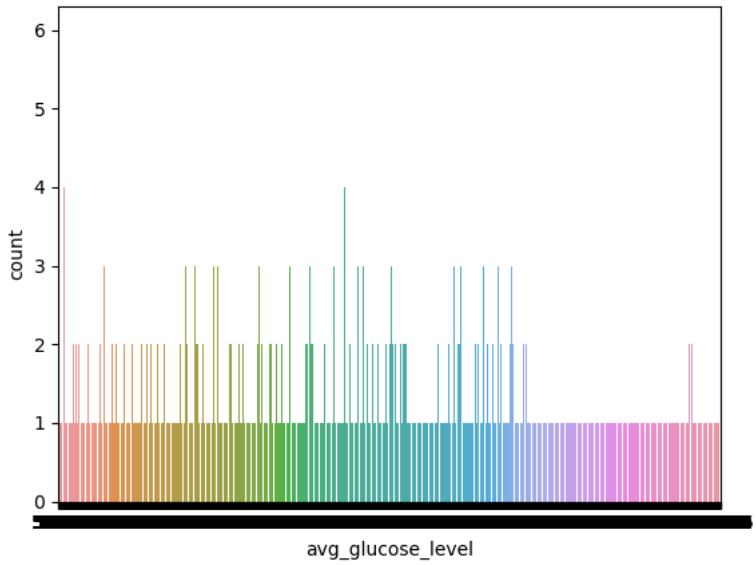


In [485]:

```
sns.countplot(x = strokesData['avg_glucose_level'])
```

Out[485]:

```
<AxesSubplot:xlabel='avg_glucose_level', ylabel='count'>
```

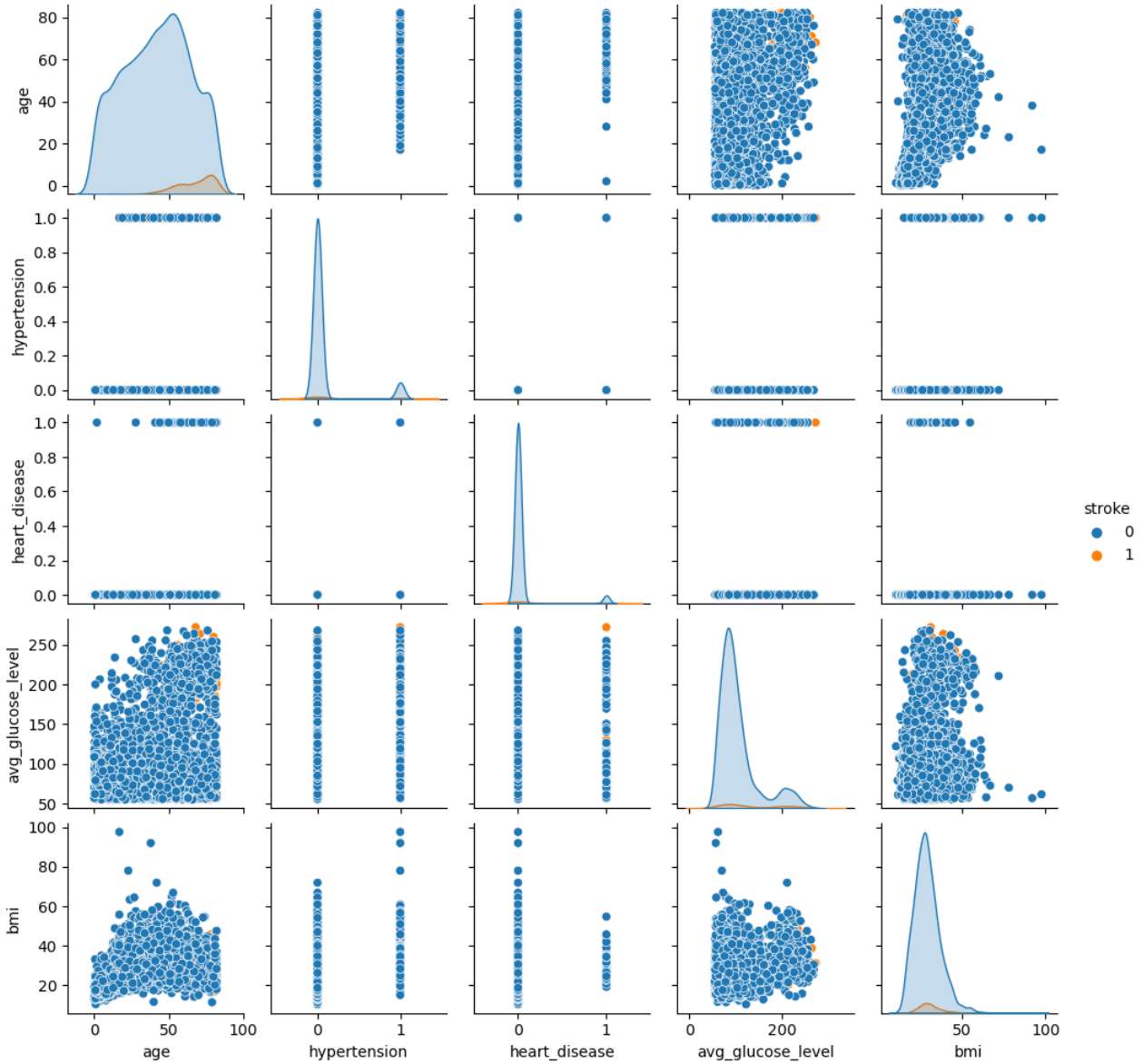


In [486]:

```
sns.pairplot(strokesData, hue="stroke", height=2)
```

Out[486]:

```
<seaborn.axisgrid.PairGrid at 0x1b1e3425580>
```



In [ ]: