In [4]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
```

In [5]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
# Using double backslashes
file_path = 'C:\\Users\\Dell\\Downloads\\jupyter nootbook\\ML\\DB\\diabetes_pre
df = pd.read_csv(file_path)

# Now, 'df' contains the data from the CSV file
```

In [7]:
```
df.head()
```

Out[7]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucos
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	

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

Out[8]:
```
(100000, 9)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

In [10]:
```
df.isnull().sum()#no NULL values
```

Out[10]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

In [11]:
```
dup=df.duplicated().sum()
dup
#we have 3854 duplicate values
```

Out[11]:
```
3854
```

EDA

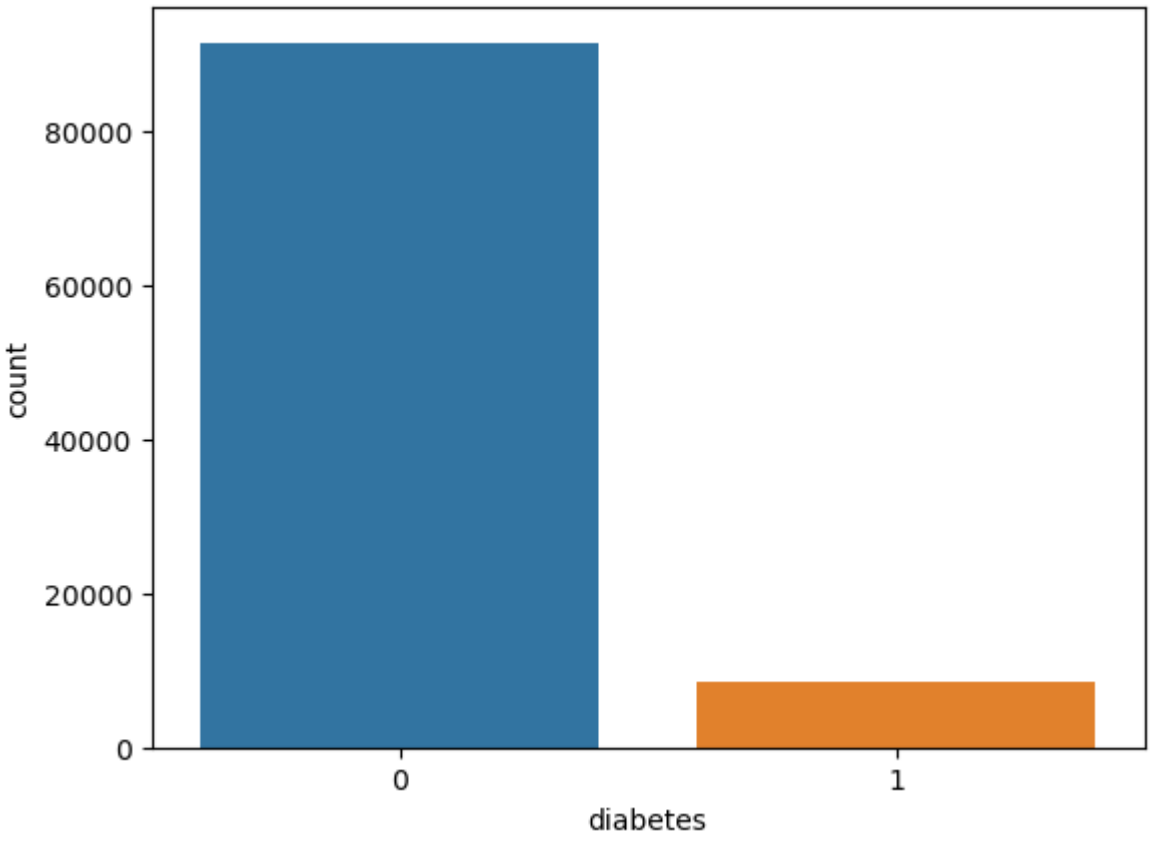
In [12]:
```
for i in df.columns:
    print(df[i].value_counts())
    print('\n')
```

```
Female    58552
Male      41430
Other        18
Name: gender, dtype: int64


80.00    5621
51.00    1619
47.00    1574
48.00    1568
53.00    1542
         ...
0.48       83
1.00       83
0.40       66
0.16       59
0.08       36
Name: age, Length: 102, dtype: int64


0    92515
1     7485
Name: hypertension, dtype: int64


0    96058
1     3942
Name: heart_disease, dtype: int64


No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: smoking_history, dtype: int64


27.32    25495
23.00      103
27.12      101
27.80      100
24.96      100
         ...
58.23        1
48.18        1
55.57        1
57.07        1
60.52        1
Name: bmi, Length: 4247, dtype: int64


6.6    8540
5.7    8413
6.5    8362
5.8    8321
```

```
6.0    8295
6.2    8269
6.1    8048
3.5    7662
4.8    7597
4.5    7585
4.0    7542
5.0    7471
8.8     661
8.2     661
9.0     654
7.5     643
6.8     642
7.0     634
Name: HbA1c_level, dtype: int64


130    7794
159    7759
140    7732
160    7712
126    7702
145    7679
200    7600
155    7575
90     7112
80     7106
158    7026
100    7025
85     6901
280     729
300     674
240     636
260     635
220     603
Name: blood_glucose_level, dtype: int64


0    91500
1     8500
Name: diabetes, dtype: int64
```

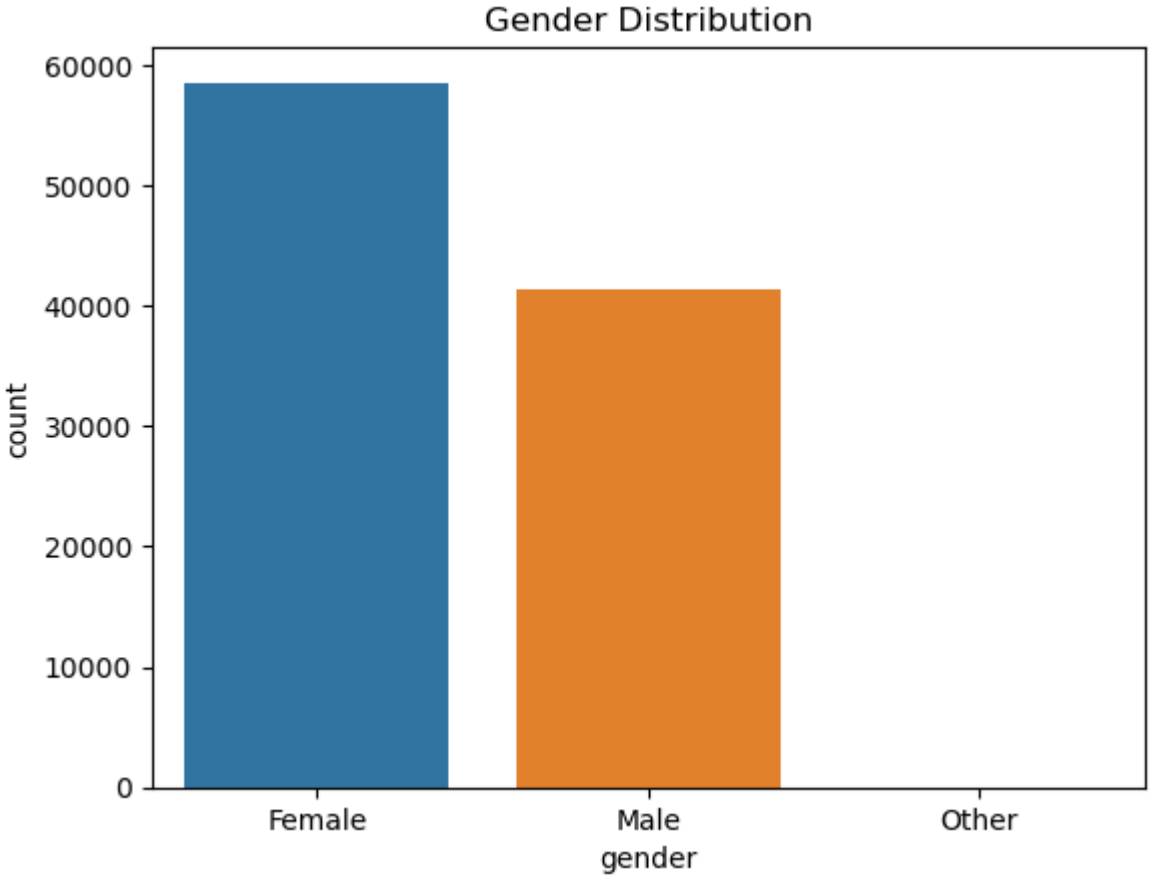
In [13]:
```
sns.countplot(x='diabetes',data=df)#imbalace dataset
```

Out[13]:
```
<Axes: xlabel='diabetes', ylabel='count'>
```



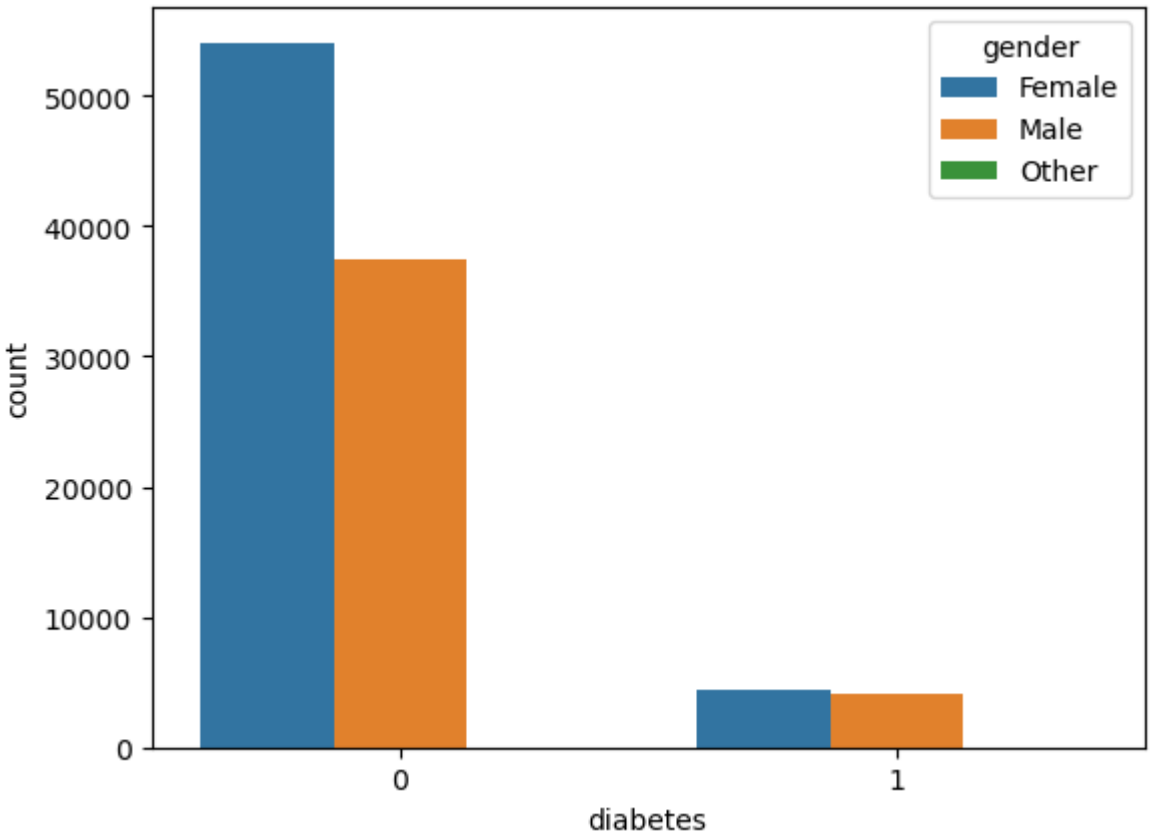
In [14]:
```
plt.title('Gender Distribution')
sns.countplot(x='gender',data=df)
#we got more Females than Male
```

Out[14]:
```
<Axes: title={'center': 'Gender Distribution'}, xlabel='gender', ylabel='coun
t'>
```



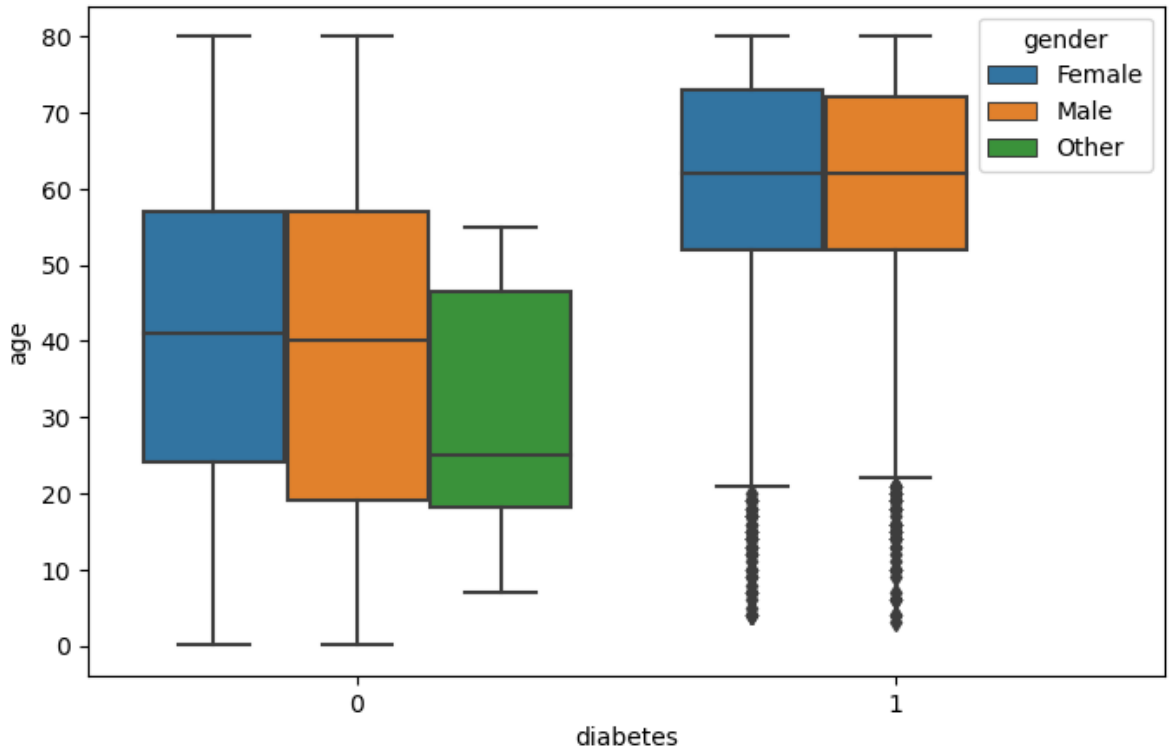
In [15]:
```
sns.countplot(x='diabetes',data=df,hue='gender')
#gender=other have no diabetes patient
```

Out[15]:
```
<Axes: xlabel='diabetes', ylabel='count'>
```



In [16]:
```
plt.figure(figsize=(8,5))
sns.boxplot(y='age',x='diabetes',hue='gender',data=df)
```

Out[16]: <Axes: xlabel='diabetes', ylabel='age'>

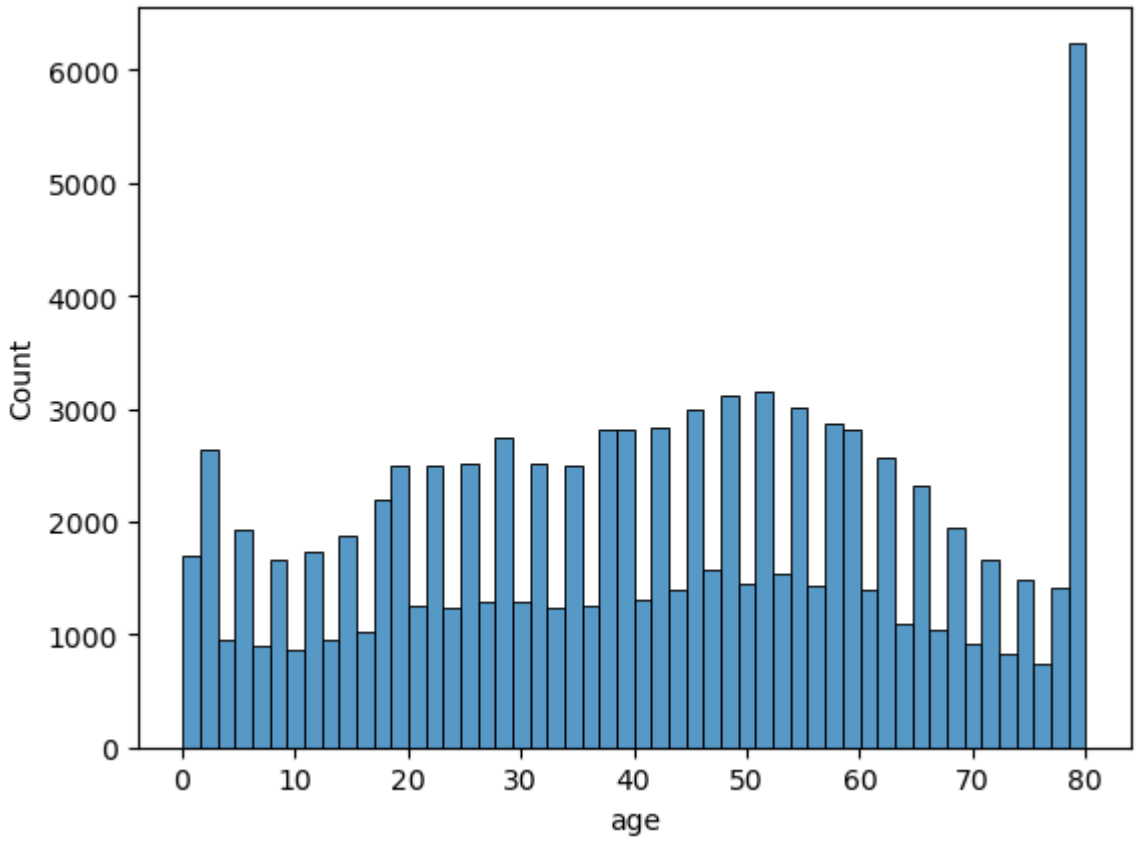


In [17]:
```
for i in df['gender'].value_counts().keys():
    total=df[df['gender']==i].shape[0]
    diabetes=df[(df['gender']==i)&(df['diabetes']==1)].shape[0]
    pre=round((diabetes/total)*100,5)
    print(f"Total no. of {i} is {total} out of them {pre}% have diabetes that
```

```
Total no. of Female is 58552 out of them 7.61887% have diabetes that is 4461
no. of Female

Total no. of Male is 41430 out of them 9.74897% have diabetes that is 4039 n
o. of Male

Total no. of Other is 18 out of them 0.0% have diabetes that is 0 no. of Othe
r
```

In [18]:
```
sns.histplot(x='age',data=df)
#Around 6200 people are 80 years old
```

Out[18]: <Axes: xlabel='age', ylabel='Count'>

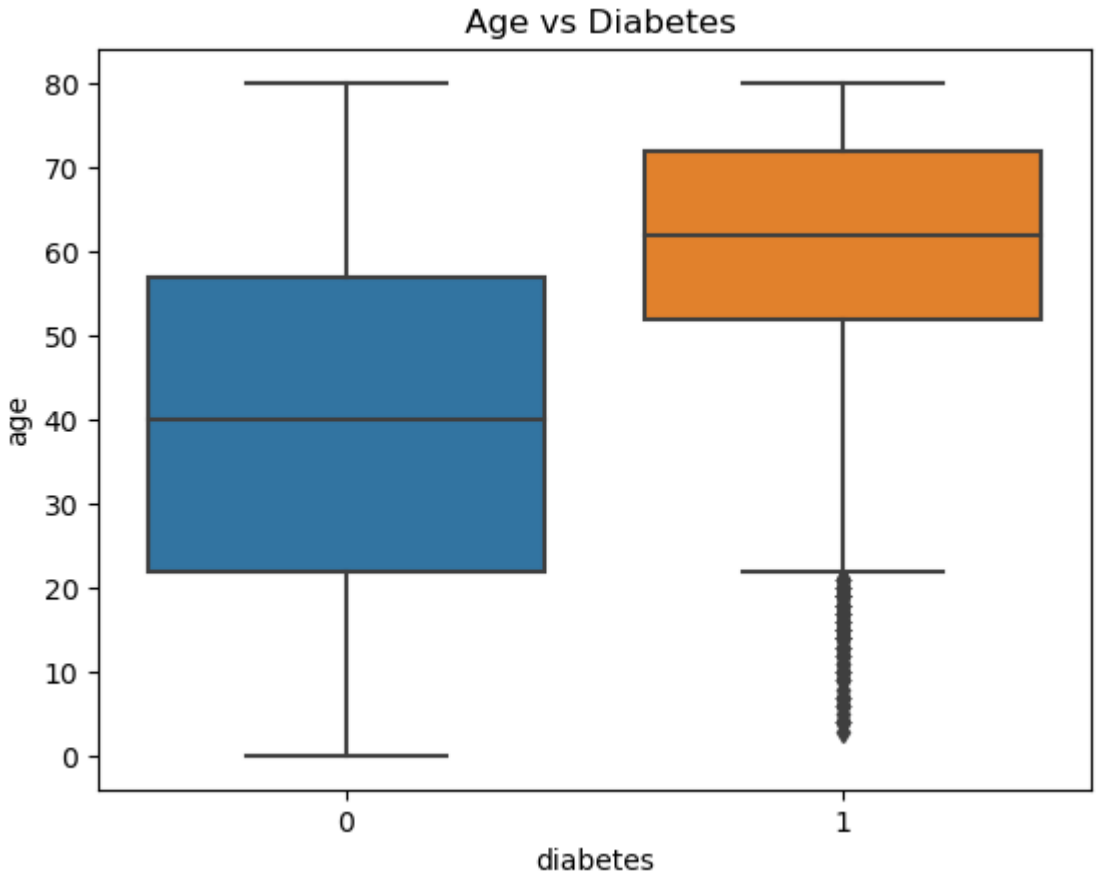


In [19]:
```
diabetes_yes=df[df['diabetes']==1].shape[0]
aged=df[(df['age']>60) & (df['diabetes']==1)].shape[0]
per=round((aged/diabetes_yes)*100,1)
print(f"total diabetes patients is {diabetes_yes}, {per}% are over the age of (

# Diabetes is a common health issue among old people
```

```
total diabetes patients is 8500, 55.5% are over the age of 60
```

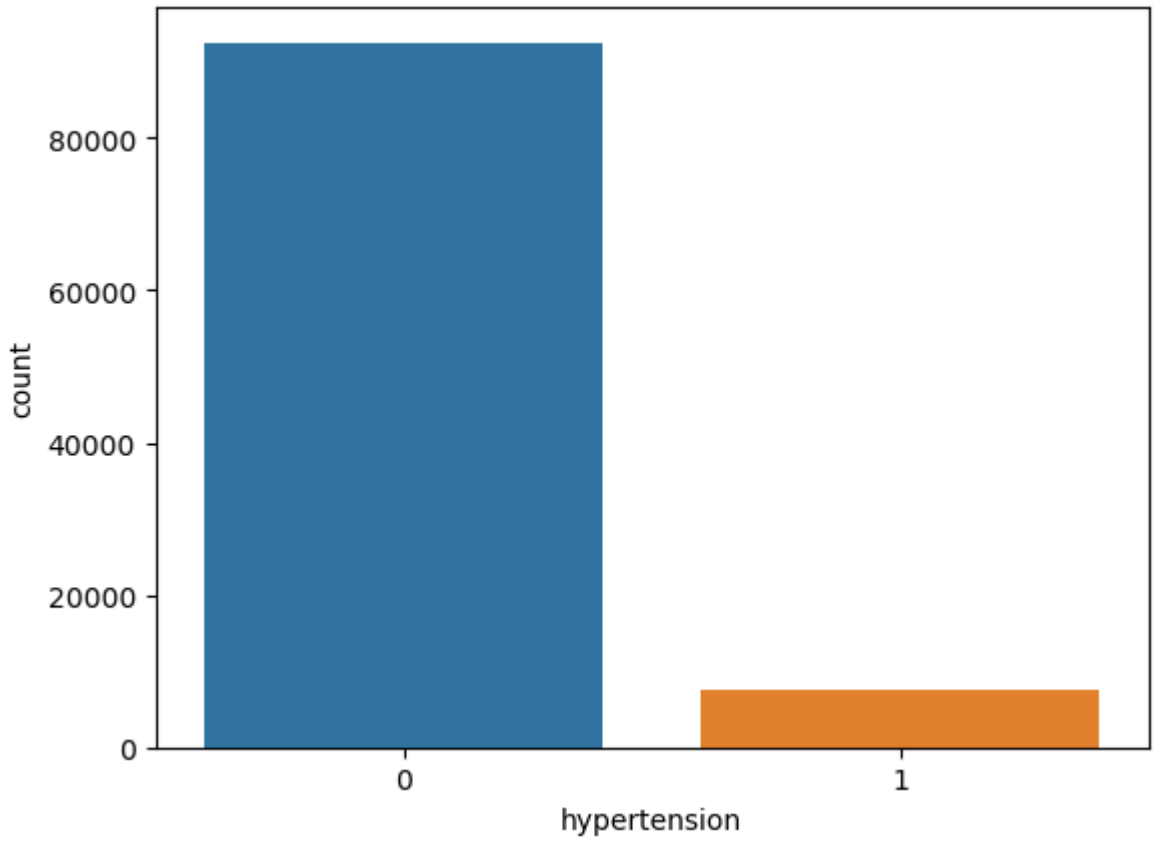
In [20]:
```
sns.boxplot(x='diabetes',y='age',data=df)
plt.title(' Age vs Diabetes')
#Old people have diabetes
```

Out[20]: Text(0.5, 1.0, ' Age vs Diabetes')



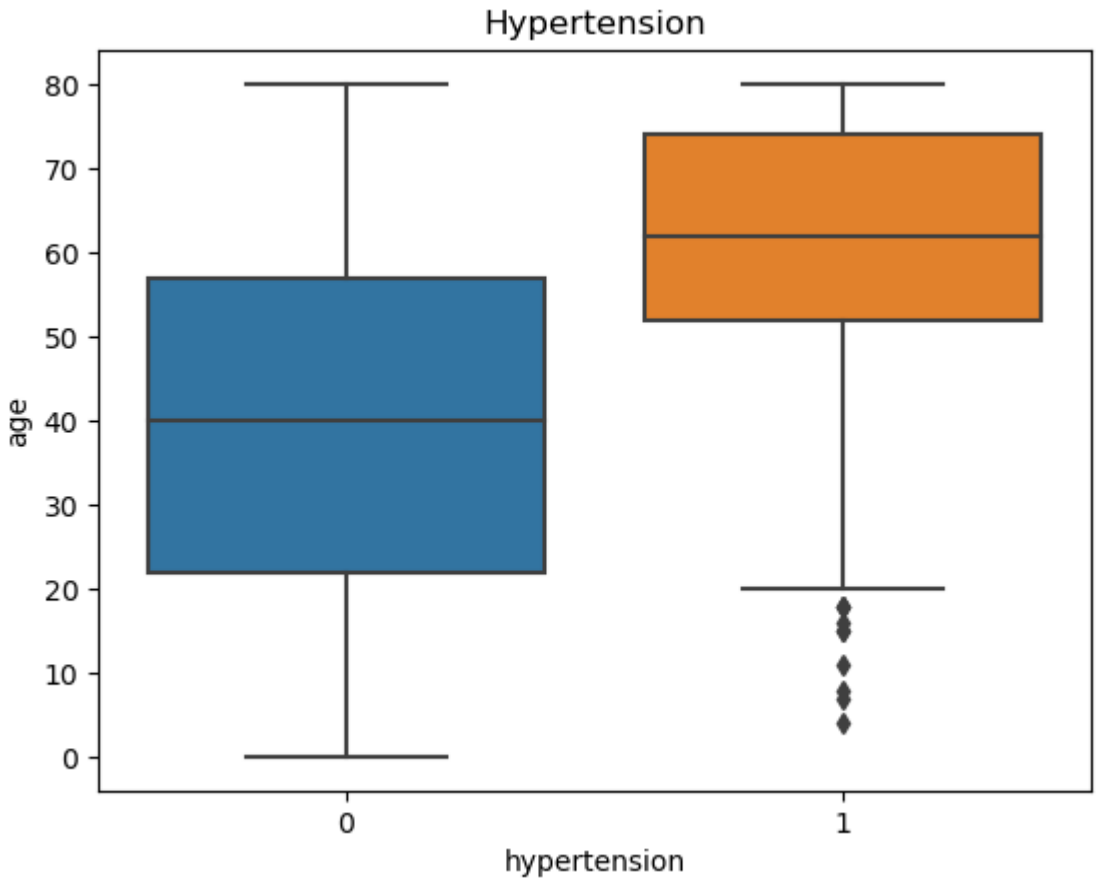
In [21]:
```
sns.countplot(x='hypertension',data=df)
```

Out[21]:
```
<Axes: xlabel='hypertension', ylabel='count'>
```



In [22]:
```
sns.boxplot(x='hypertension',y='age',data=df)
plt.title('Hypertension')

#Old people tend to have hypertension
```

Out[22]:
```
Text(0.5, 1.0, 'Hypertension')
```



In [23]:
```
mid_age_hyper=df[(df['age']>30) & (df['age']<50) &(df['hypertension']==1)].sha
mid=df[(df['age']>30) & (df['age']<50)].shape[0]
print(mid_age_hyper/mid*100)
#only 5% of mid age people have hypertension
```

```
5.017268207522107
```

In [24]:
```
old_age_hyper=df[(df['age']>60) &(df['hypertension']==1)].shape[0]
old=df[(df['age']>60)].shape[0]
print(old_age_hyper/old*100)
#17% of old age people have hypertension
```

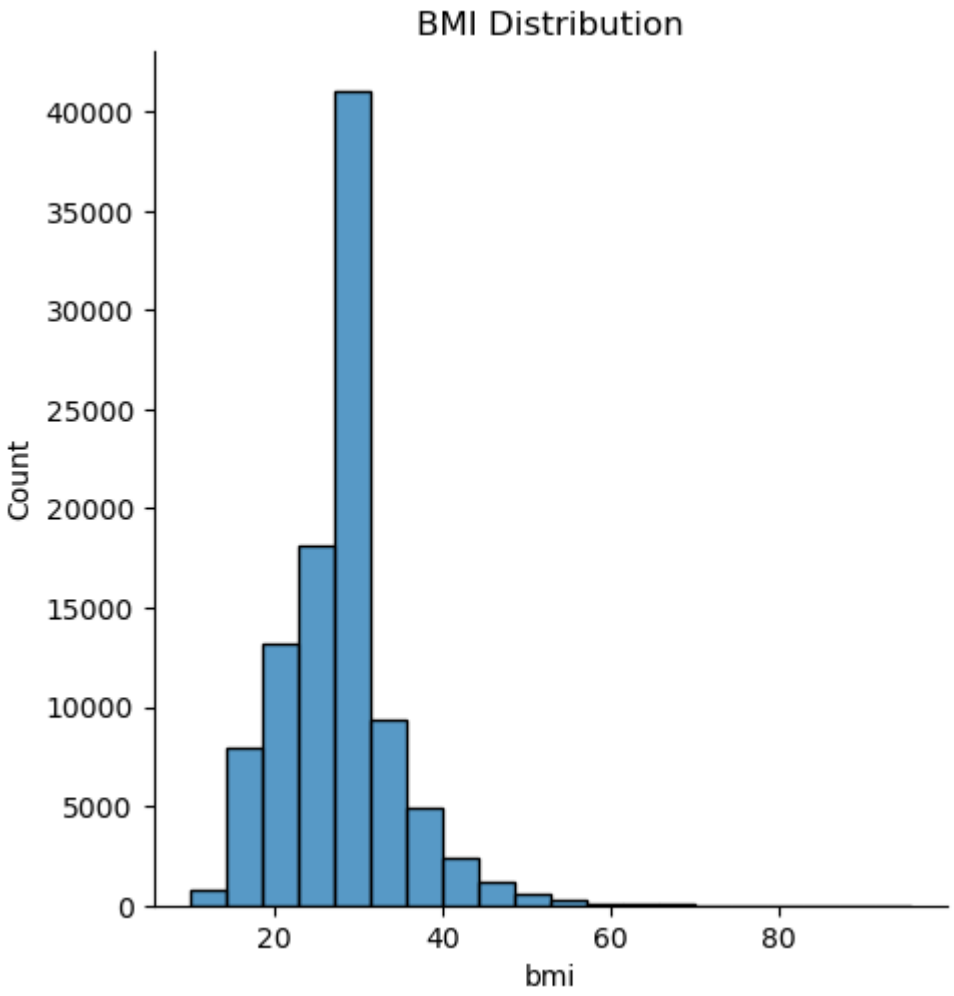
```
17.52708192281652
```

In [25]:
```
for i in ['Female','Male']:
    val=(df[(df['gender']==i) & (df['hypertension']==1)].shape[0])/df.shape[0]
    print(f"{round(val,2)}% of {i} have hypertension")

#There's the same level of hypertension in both genders
```

```
4.2% of Female have hypertension
3.29% of Male have hypertension
```

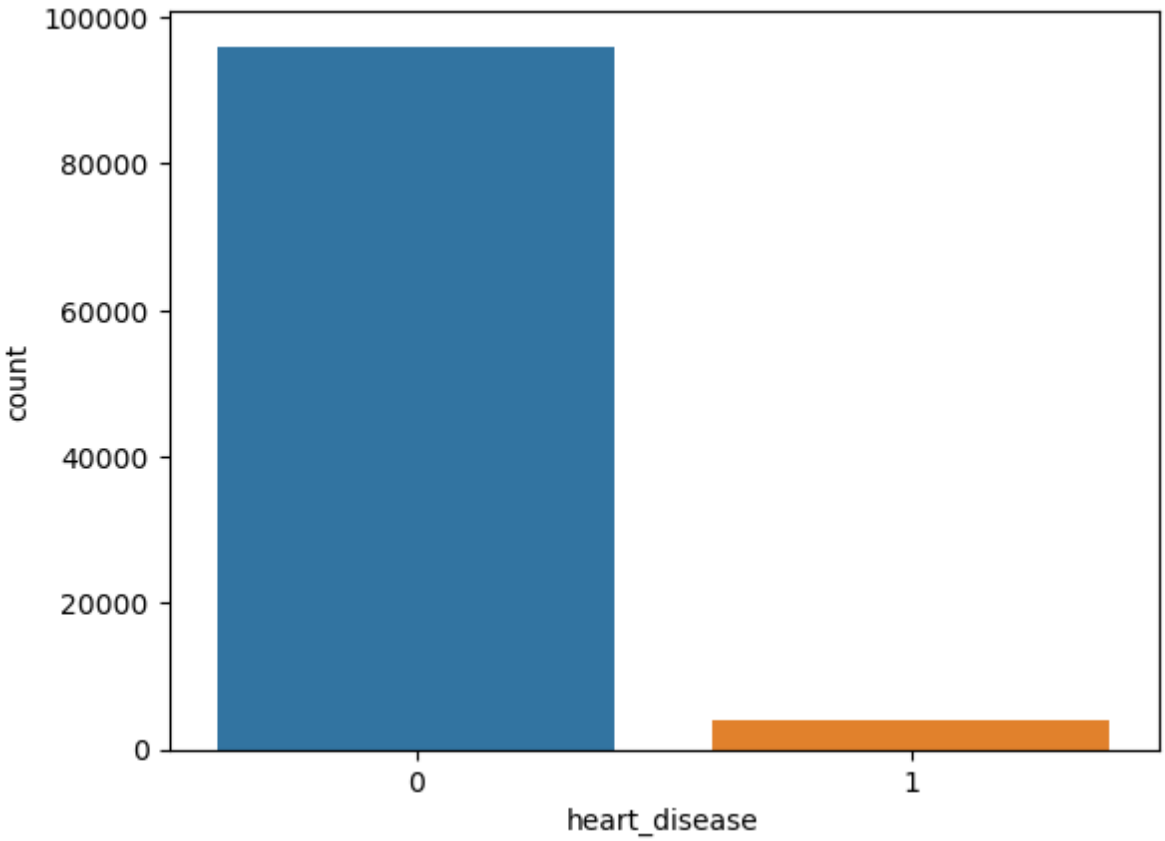
In [26]:
```
sns.displot(df['bmi'],bins=20)
plt.title('BMI Distribution')
```

Out[26]:
```
Text(0.5, 1.0, 'BMI Distribution')
```



In [27]:
```
sns.countplot(x='heart_disease',data=df)
```

Out[27]:
```
<Axes: xlabel='heart_disease', ylabel='count'>
```



In [28]:
```
mid_age_heart=df[(df['age']>30) & (df['age']<50) &(df['heart_disease']==1)].sh
mid=df[(df['age']>30) & (df['age']<50)].shape[0]
print(mid_age_heart/mid*100)
#only 0.95% of mid age people have heart_disease
```

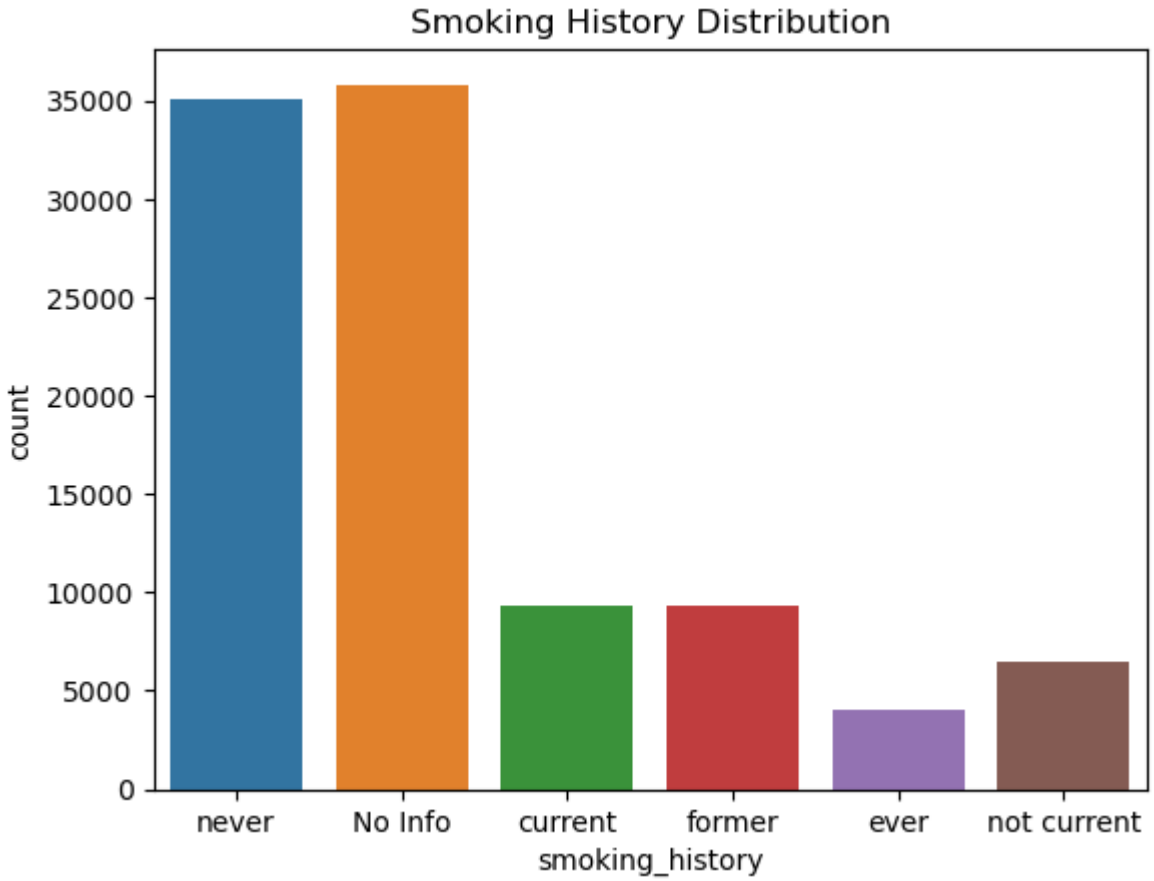
```
0.9525978215492049
```

In [29]:
```
old_age_heart=df[(df['age']>60) &(df['hypertension']==1)].shape[0]
old=df[(df['age']>60)].shape[0]
print(old_age_heart/old*100)
#17% of old age people have hypertension
```

```
17.52708192281652
```

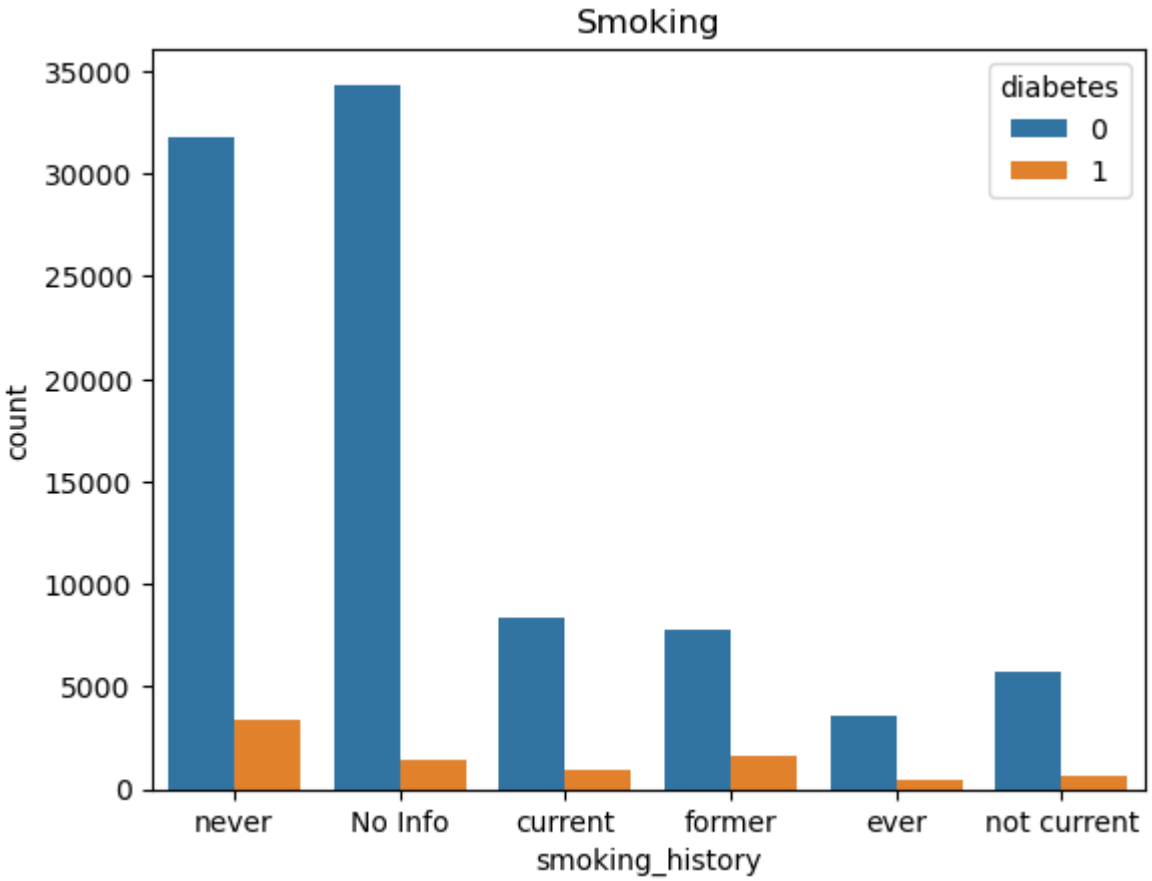
In [30]:
```
sns.countplot(x='smoking_history', data=df)
plt.title('Smoking History Distribution')
```

Out[30]:
```
Text(0.5, 1.0, 'Smoking History Distribution')
```



In [31]:
```
sns.countplot(x='smoking_history',hue='diabetes',data=df)
plt.title('Smoking')
# I don't think smoking really affects diabetes based on this distribution
```

Out[31]:
```
Text(0.5, 1.0, 'Smoking')
```

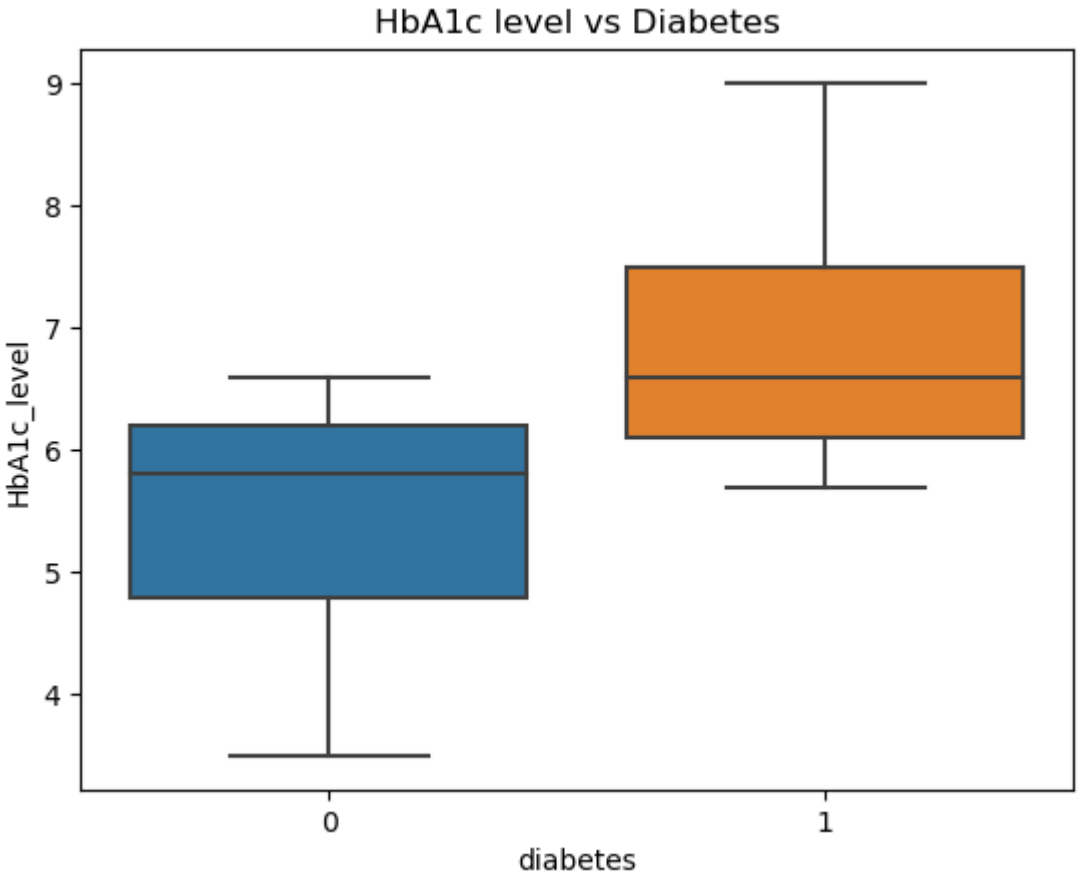


In [32]:
```
for i in ['No Info', 'never', 'former', 'current', 'not current', 'ever']:
  total=df['smoking_history'].value_counts()[i]
  num=df[(df['smoking_history']==i) & (df['diabetes']==1)].shape[0]
  print(f"No of patents with smoking history: {i} is {total} and having diabet
  print('\n')
```

```
No of patents with smoking history: No Info is 35816 and having diabetes is:
1454, percentage 4.059638150547242


No of patents with smoking history: never is 35095 and having diabetes is: 33
46, percentage 9.534121669753526


No of patents with smoking history: former is 9352 and having diabetes is: 15
90, percentage 17.001710863986315


No of patents with smoking history: current is 9286 and having diabetes is: 9
48, percentage 10.208916648718501


No of patents with smoking history: not current is 6447 and having diabetes i
s: 690, percentage 10.702652396463472


No of patents with smoking history: ever is 4004 and having diabetes is: 472,
percentage 11.78821178821179
```

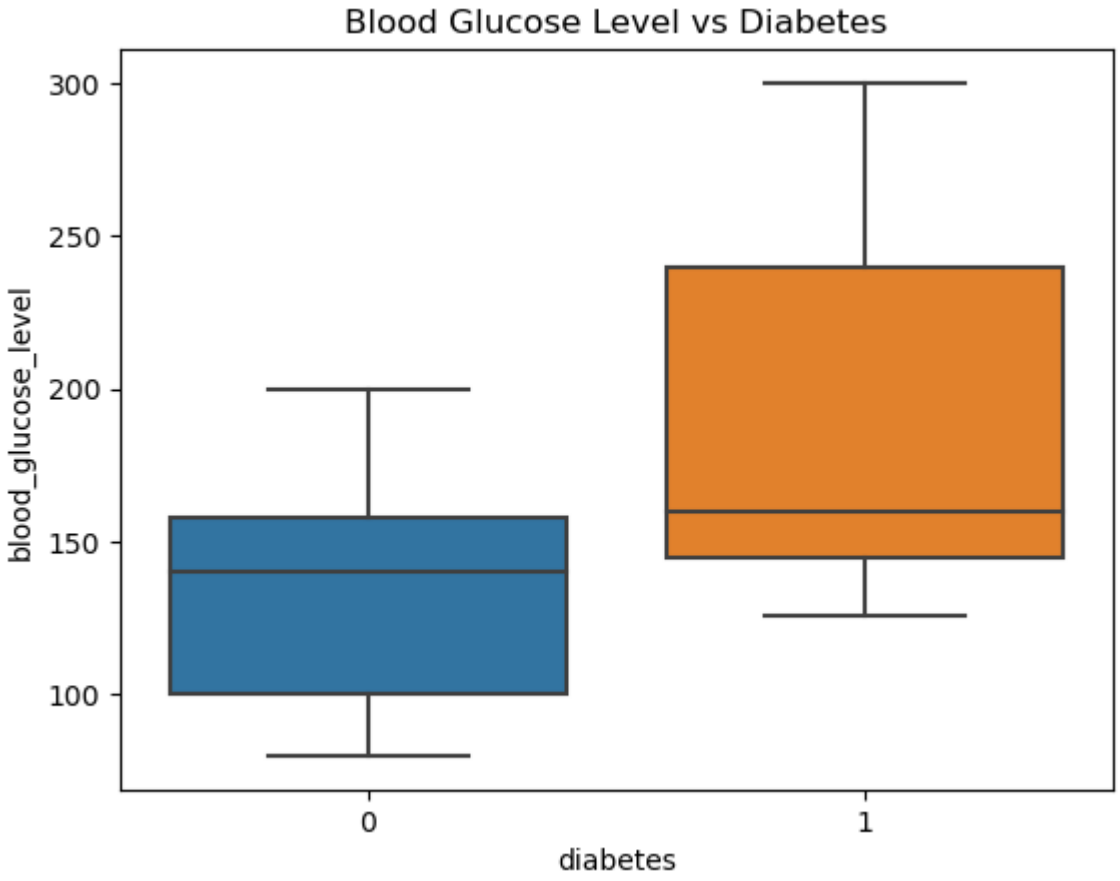
In [33]:
```
sns.boxplot(x='diabetes', y='HbA1c_level', data=df)
plt.title('HbA1c level vs Diabetes')
#Those who have diabetes have a higher hemoglobin label than normal people.
```

Out[33]: Text(0.5, 1.0, 'HbA1c level vs Diabetes')



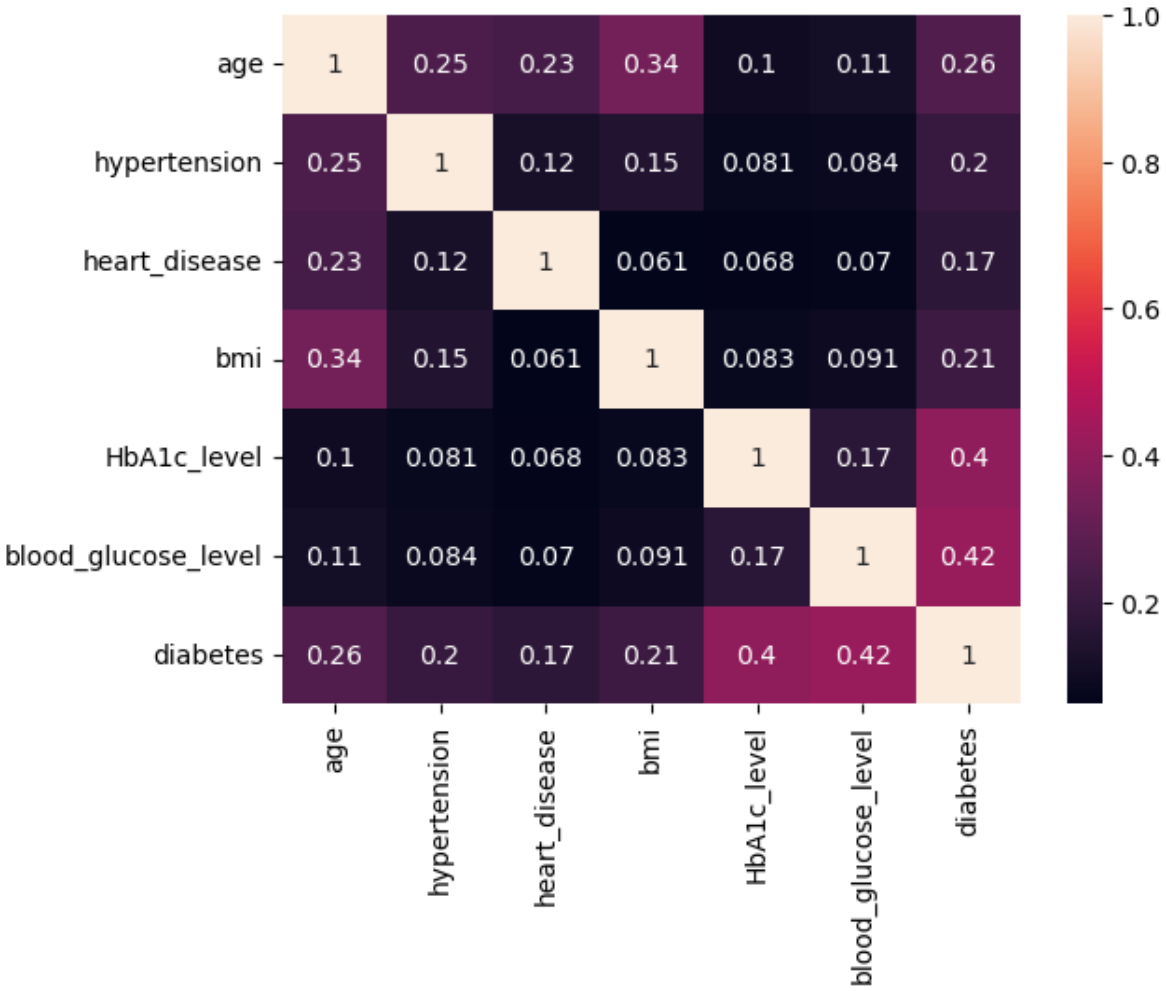
In [34]:
```
sns.boxplot(x='diabetes', y='blood_glucose_level', data=df)
plt.title('Blood Glucose Level vs Diabetes')
```

Out[34]:
```
Text(0.5, 1.0, 'Blood Glucose Level vs Diabetes')
```



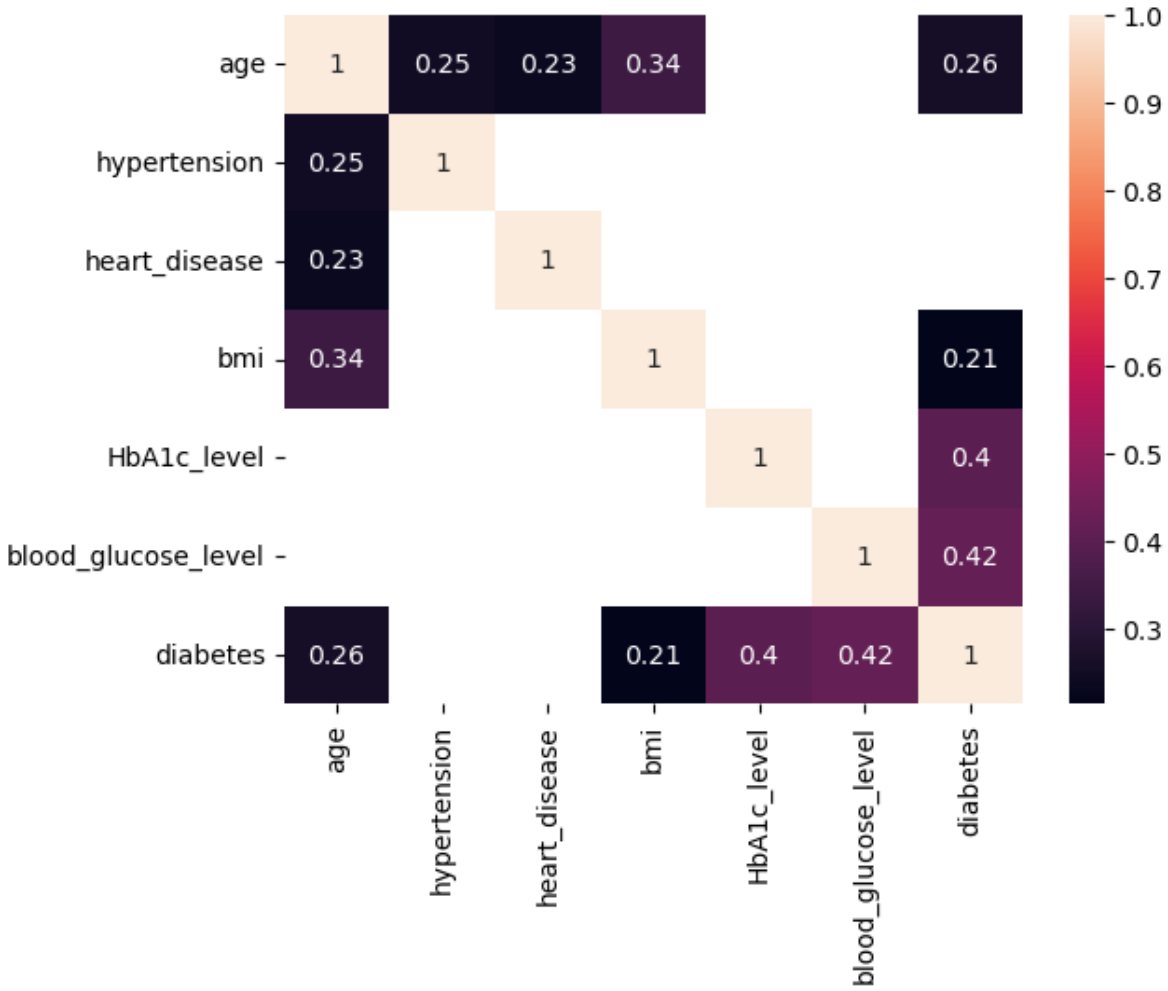
In [35]:
```
c=df.corr()
sns.heatmap(c,annot=True)
```

Out[35]: <Axes: >



In [36]:
```
sns.heatmap(c[(c>=0.2) & (c>-0.2)],annot=True)
```

Out[36]: <Axes: >



In [37]:
```
df['gender']=df['gender'].map({'Female':0,'Male':1,'Other':2})
df.drop('smoking_history',axis=1,inplace=True)#droping because too much data i
```

Scaling

In [38]:
```
from sklearn.preprocessing import StandardScaler
```

In [39]:
```
sc=StandardScaler()
d=sc.fit_transform(df[['age','HbA1c_level','blood_glucose_level']])
df.drop(['age','HbA1c_level','blood_glucose_level'],axis=1,inplace=True)
d2=pd.DataFrame(d,columns=['age','HbA1c_level','blood_glucose_level'])
final=pd.concat([d2,df],axis=1)
```

In [40]:
```
final.head()
```

Out[40]:

	age	HbA1c_level	blood_glucose_level	gender	hypertension	heart_disease	bmi	diab
0	1.692704	1.001706	0.047704	0	0	1	25.19	
1	0.538006	1.001706	-1.426210	0	0	0	27.32	
2	-0.616691	0.161108	0.489878	1	0	0	27.32	
3	-0.261399	-0.492690	0.416183	0	0	0	23.45	
4	1.515058	-0.679490	0.416183	1	1	1	20.14	

Model Building

In [41]:
```
pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in c:\users\dell\appdata\roaming\pytho
n\python311\site-packages (1.7.6)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pac
kages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-pac
kages (from xgboost) (1.10.1)
Note: you may need to restart the kernel to use updated packages.
```

In [42]:
```
conda install -c conda-forge xgboost
```

```
Collecting package metadata (current_repodata.json): ...working... done
Note: you may need to restart the kernel to use updated packages.

Solving environment: ...working... unsuccessful initial attempt using frozen
solve. Retrying with flexible solve.
Solving environment: ...working... done
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\ProgramData\anaconda3

  added / updated specs:
    - xgboost


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    notebook-6.5.2             |     pyha770c72_0         270 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         270 KB

The following NEW packages will be INSTALLED:

  _py-xgboost-mutex  conda-forge/win-64::_py-xgboost-mutex-2.0-cpu_0
  libxgboost         pkgs/main/win-64::libxgboost-1.7.3-hd77b12b_0
  py-xgboost         pkgs/main/win-64::py-xgboost-1.7.3-py311haa95532_0
  python_abi         conda-forge/win-64::python_abi-3.11-2_cp311
  xgboost            pkgs/main/win-64::xgboost-1.7.3-py311haa95532_0

The following packages will be UPDATED:

  ca-certificates    pkgs/main::ca-certificates-2023.05.30~ --> conda-forge::
ca-certificates-2023.7.22-h56e8100_0
  certifi            pkgs/main/win-64::certifi-2023.5.7-py~ --> conda-forge/n
oarch::certifi-2023.7.22-pyhd8ed1ab_0
  conda              pkgs/main::conda-23.5.2-py311haa95532~ --> conda-forge::
conda-23.7.3-py311h1ea47a8_0
  openssl                                1.1.1u-h2bbff1b_0 --> 1.1.1v-h2bbff
1b_0

The following packages will be SUPERSEDED by a higher-priority channel:

  notebook           pkgs/main/win-64::notebook-6.5.4-py31~ --> conda-forge/n
oarch::notebook-6.5.2-pyha770c72_0



Downloading and Extracting Packages

notebook-6.5.2       | 270 KB    |            |   0%
notebook-6.5.2       | 270 KB    | 5          |   6%
notebook-6.5.2       | 270 KB    | ####7      |  47%
notebook-6.5.2       | 270 KB    | ########## | 100%
```

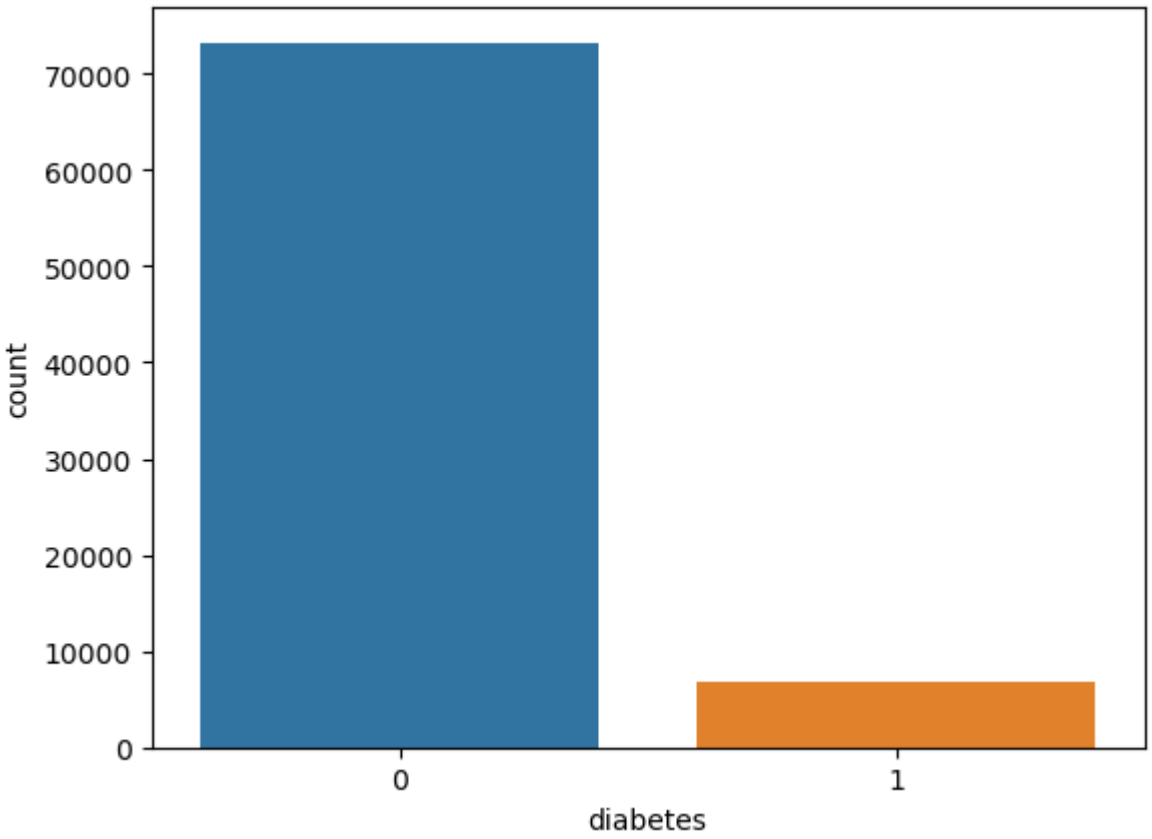
```
notebook-6.5.2       | 270 KB    | ########## | 100%


Preparing transaction: ...working... done
Verifying transaction: ...working... failed


==> WARNING: A newer version of conda exists. <==
  current version: 23.5.2
  latest version: 23.7.3

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.7.3



==> WARNING: A newer version of conda exists. <==
  current version: 23.5.2
  latest version: 23.7.3

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.7.3



EnvironmentNotWritableError: The current user does not have write permissions
to the target environment.
  environment location: C:\ProgramData\anaconda3
```

In [43]:
```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_m
```

In [44]:
```
x=final.drop('diabetes',axis=1)
y=final['diabetes']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [45]:
```
sns.countplot(x=y_train)
```

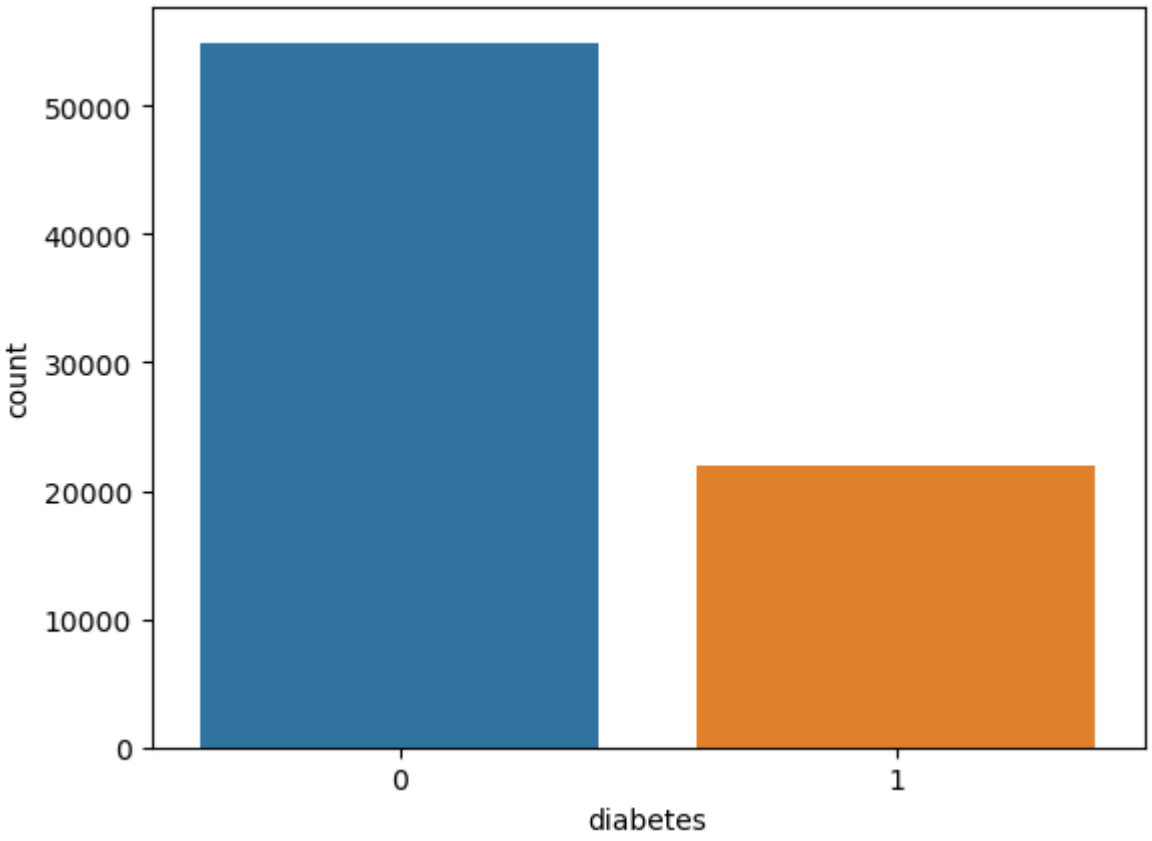
Out[45]:
```
<Axes: xlabel='diabetes', ylabel='count'>
```



In [46]:
```
sm=SMOTE(sampling_strategy=0.3)
us=RandomUnderSampler(sampling_strategy=0.4)
x_train,y_train=sm.fit_resample(x_train,y_train)
x_train,y_train=us.fit_resample(x_train,y_train)
```

In [47]:
```
sns.countplot(x=y_train)
```

Out[47]:
```
<Axes: xlabel='diabetes', ylabel='count'>
```



In [48]:
```
models = {
    "Logistic regression":LogisticRegression(),
    "Tree ":DecisionTreeClassifier(),
    "RandomForest":RandomForestClassifier(),
    "xg":xgb.XGBClassifier()
}
for name,model in models.items():
    model.fit(x_train,y_train)
    p = model.predict(x_test)
    print("Model: " , name)
    print("------------------------------")
    print(classification_report(y_test,p))
    print("..............................................................|")
```

```
Model:  Logistic regression
------------------------------
              precision    recall  f1-score   support

           0       0.98      0.95      0.96     18276
           1       0.60      0.79      0.68      1724

    accuracy                           0.94     20000
   macro avg       0.79      0.87      0.82     20000
weighted avg       0.95      0.94      0.94     20000

.....................................................|
Model:  Tree
------------------------------
              precision    recall  f1-score   support

           0       0.98      0.96      0.97     18276
           1       0.66      0.75      0.70      1724

    accuracy                           0.95     20000
   macro avg       0.82      0.86      0.84     20000
weighted avg       0.95      0.95      0.95     20000

.....................................................|
Model:  RandomForest
------------------------------
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18276
           1       0.83      0.73      0.78      1724

    accuracy                           0.96     20000
   macro avg       0.90      0.86      0.88     20000
weighted avg       0.96      0.96      0.96     20000

.....................................................|
Model:  xg
------------------------------
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18276
           1       0.92      0.71      0.80      1724

    accuracy                           0.97     20000
   macro avg       0.95      0.85      0.89     20000
weighted avg       0.97      0.97      0.97     20000

.....................................................|
```

In [*]:
```
m=xgb.XGBClassifier()
params = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7]
        }
grid= GridSearchCV(m, params, cv=5)
grid.fit(x_train,y_train)
print(grid.best_params_)
print(grid.best_score_)
```

In [*]:
```
p = grid.predict(x_test)
print(classification_report(y_test,p))
print(sns.heatmap(confusion_matrix(y_test,p),annot=True,fmt="0.2f"))
```

In [*]:
```
rf = RandomForestClassifier()
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,  cv = 5 )
grid_search.fit(x_train,y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)
```

In [*]:
```
p = grid_search.predict(x_test)
print(classification_report(y_test,p))
print(sns.heatmap(confusion_matrix(y_test,p),annot=True,fmt="0.2f"))
```

In [ ]:

In [ ]:

In [ ]: