In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
#Loading the data
data = pd.read_excel("healthpdata.xlsx")
```

In [3]:
```python
data.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [4]:
```python
# treating null values
data.isnull().sum()
```

Out[4]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [5]:
```python
# No null values found
```

In [6]:
```python
data.shape
```

Out[6]:
```
(303, 14)
```

In [7]:
```python
#Dropping the duplicate data
data.drop_duplicates()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

302 rows × 14 columns

In [8]:
```python
# 1 row deleted
```

In [9]:
```python
# Stastical summary of the data
data.describe
```

Out[9]:
```
<bound method NDFrame.describe of      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]>
```

In [10]:
```python
data.nunique()
```

Out[10]:
```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

In [11]:
```python
data_cat = data.select_dtypes(include=[np.object]) # Categorical data
data_num=data.select_dtypes(include=[np.number]) #Numerical data
```

```
C:\Users\rishabh\AppData\Local\Temp\ipykernel_1228\87017024.py:1: DeprecationWarning: `np.object` is a deprecat
ed alias for the builtin `object`. To silence this warning, use `object` by itself. Doing this will not modify
any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#de
precations
  data_cat = data.select_dtypes(include=[np.object]) # Categorical data
```

In [12]:
```python
data_cat.nunique()
```

Out[12]:
```
Series([], dtype: float64)
```

In [13]:
```python
data_cat.columns
```

Out[13]:
```
Index([], dtype='object')
```

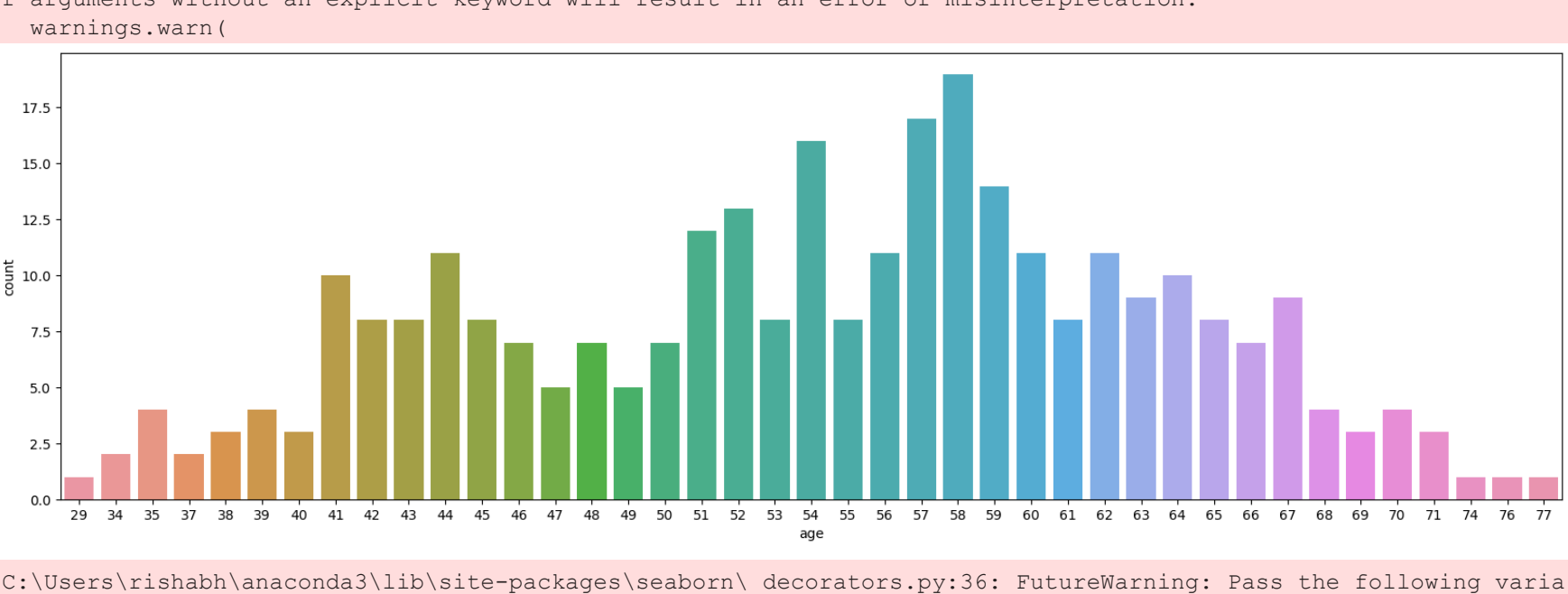
In [14]:
```python
# No categorical values
```

In [15]:
```python
data.columns
```

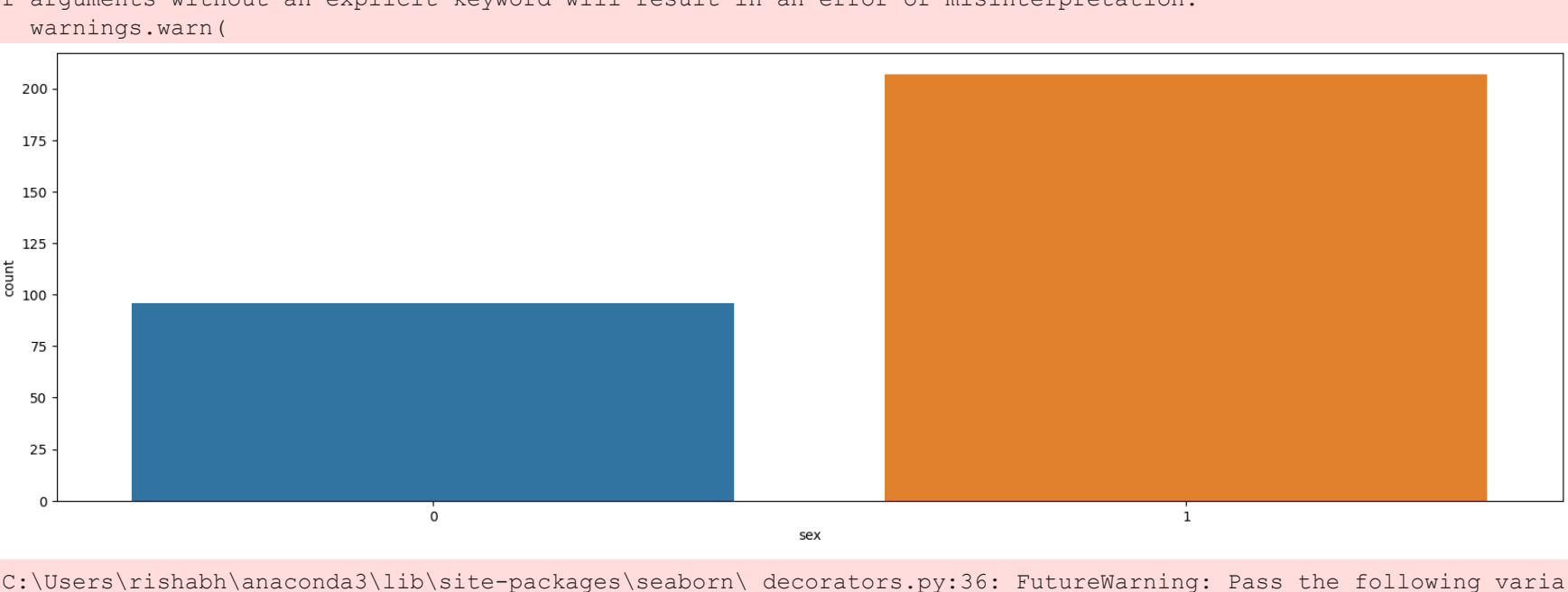
Out[15]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [16]:
```python
# count plot

for i in data.columns:
    plt.figure(figsize=(20,6))
    sns.countplot(data[i])
    plt.show()
```

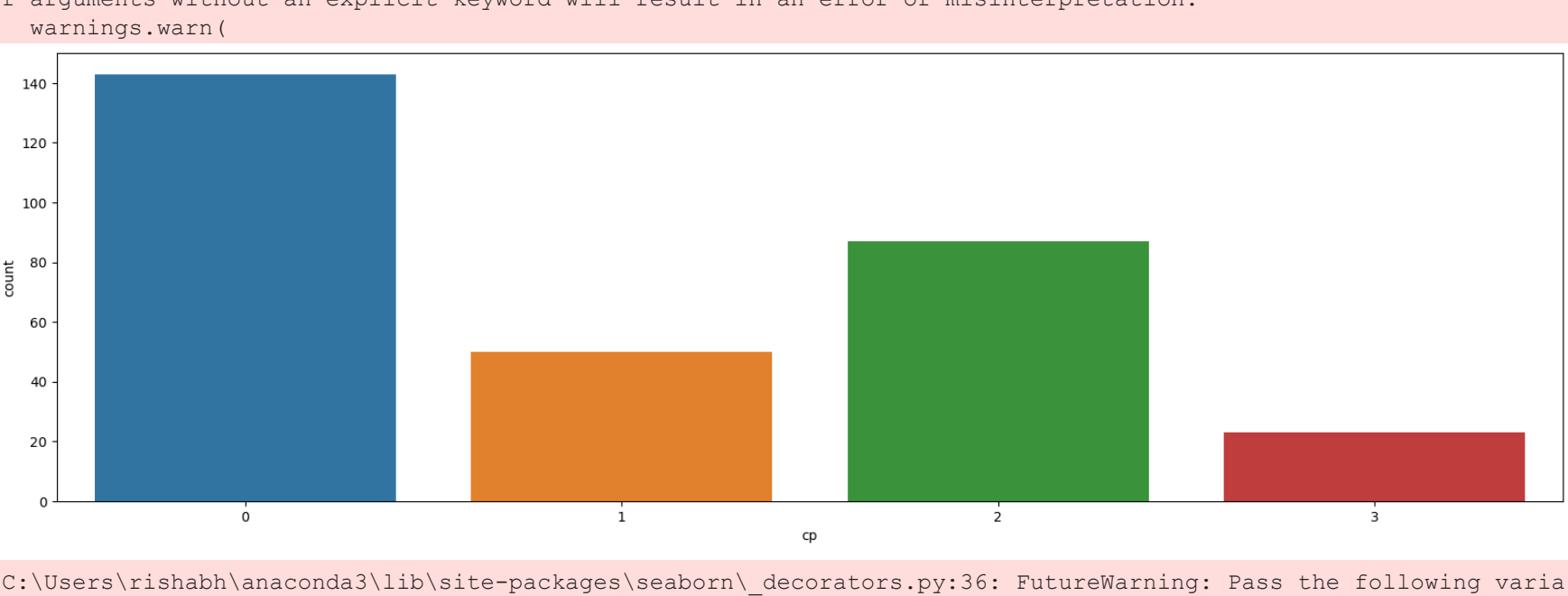
```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



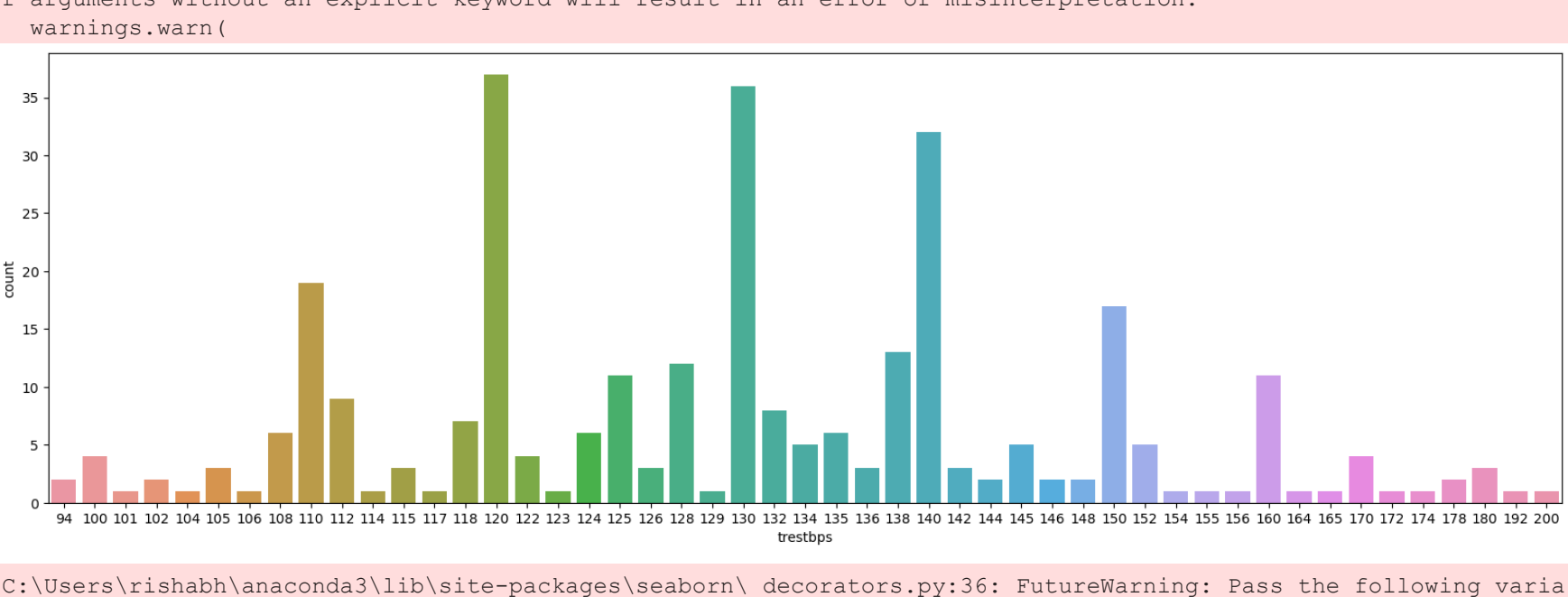

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```




```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

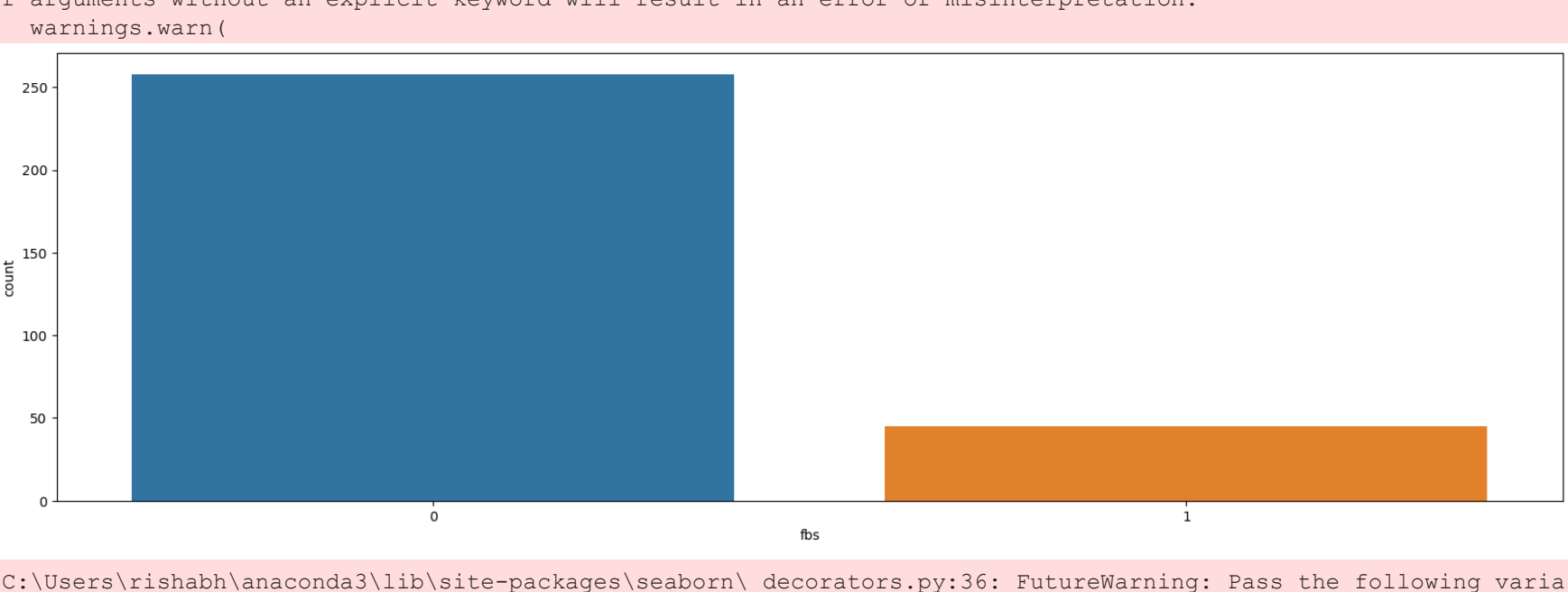



```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

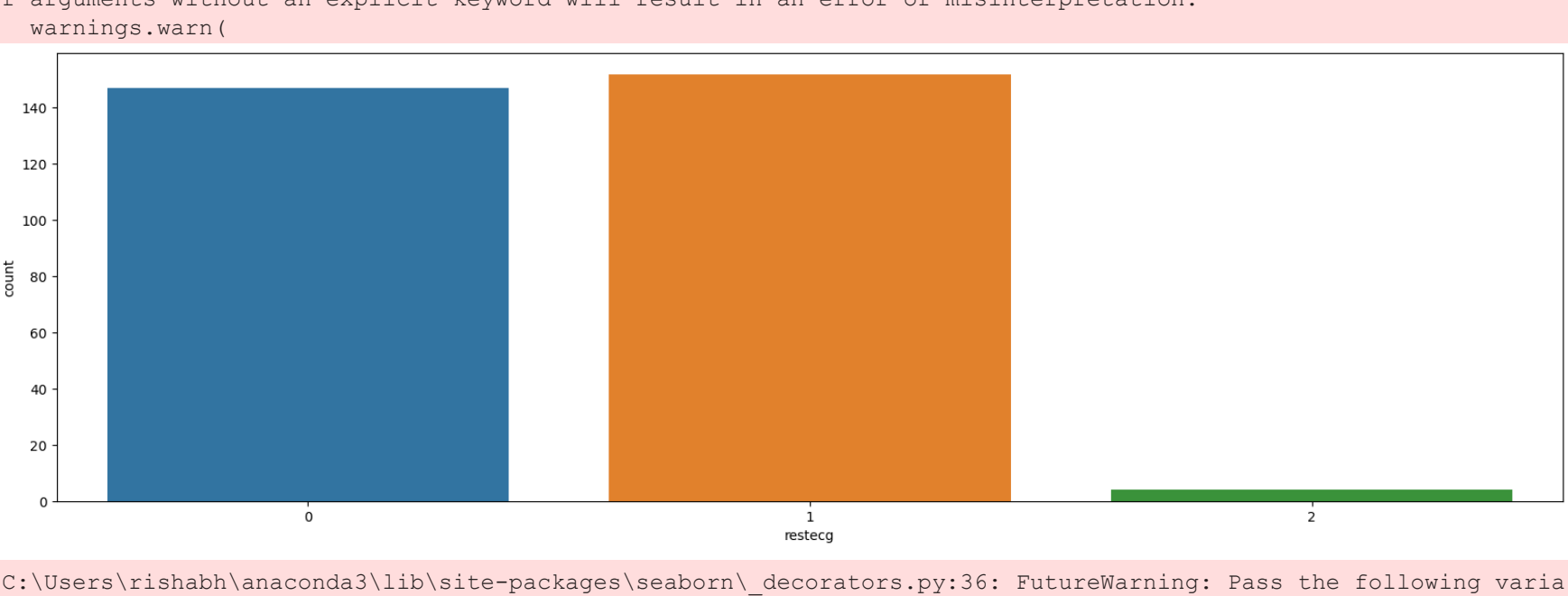



```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

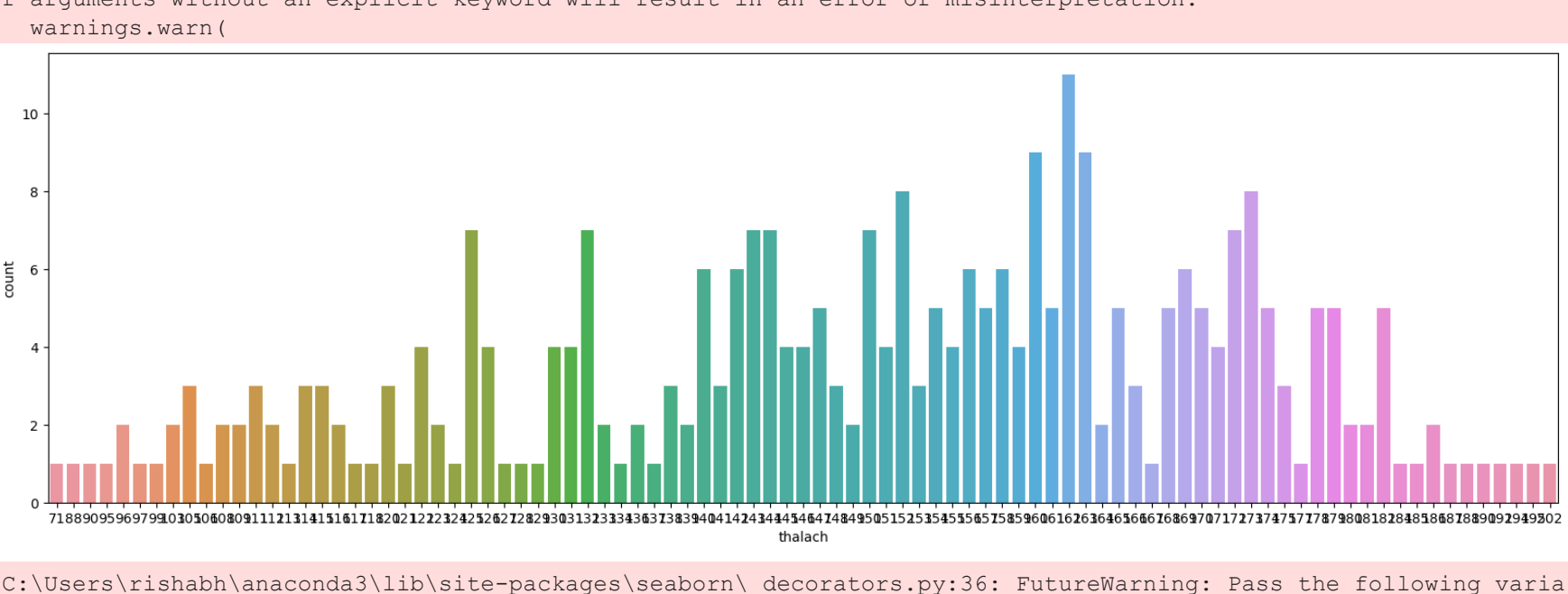

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



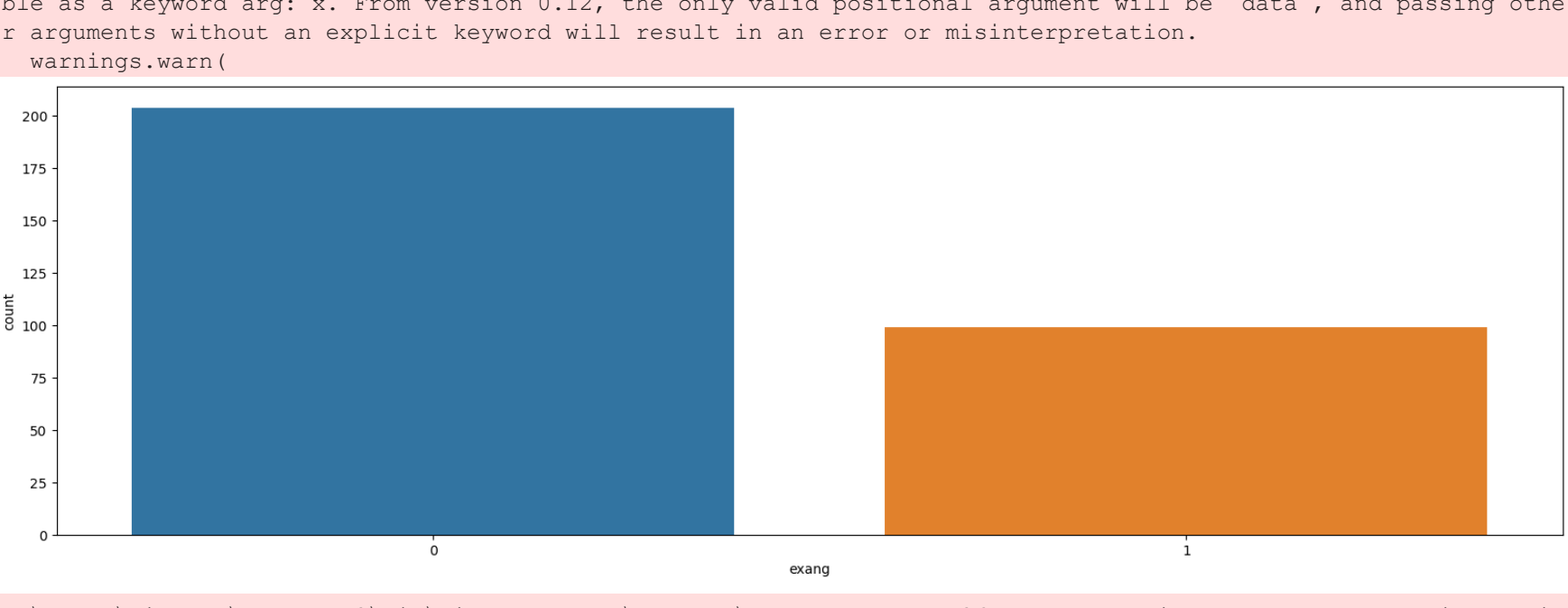

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



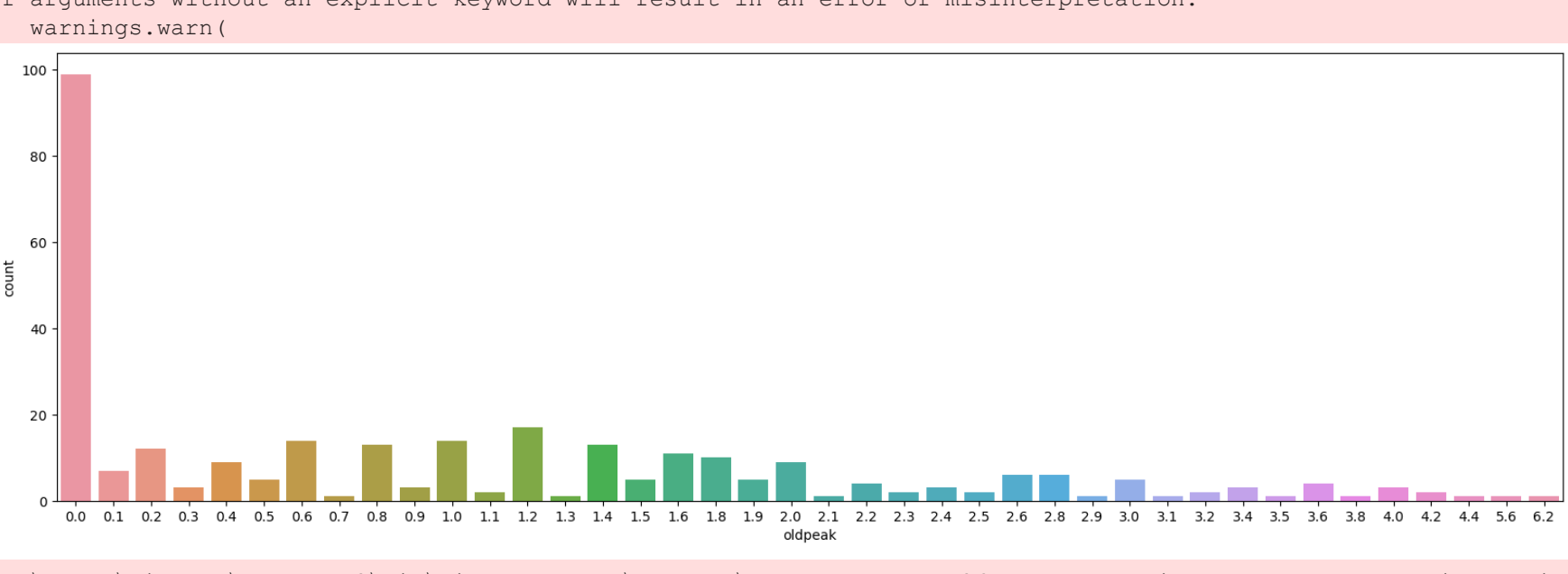

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



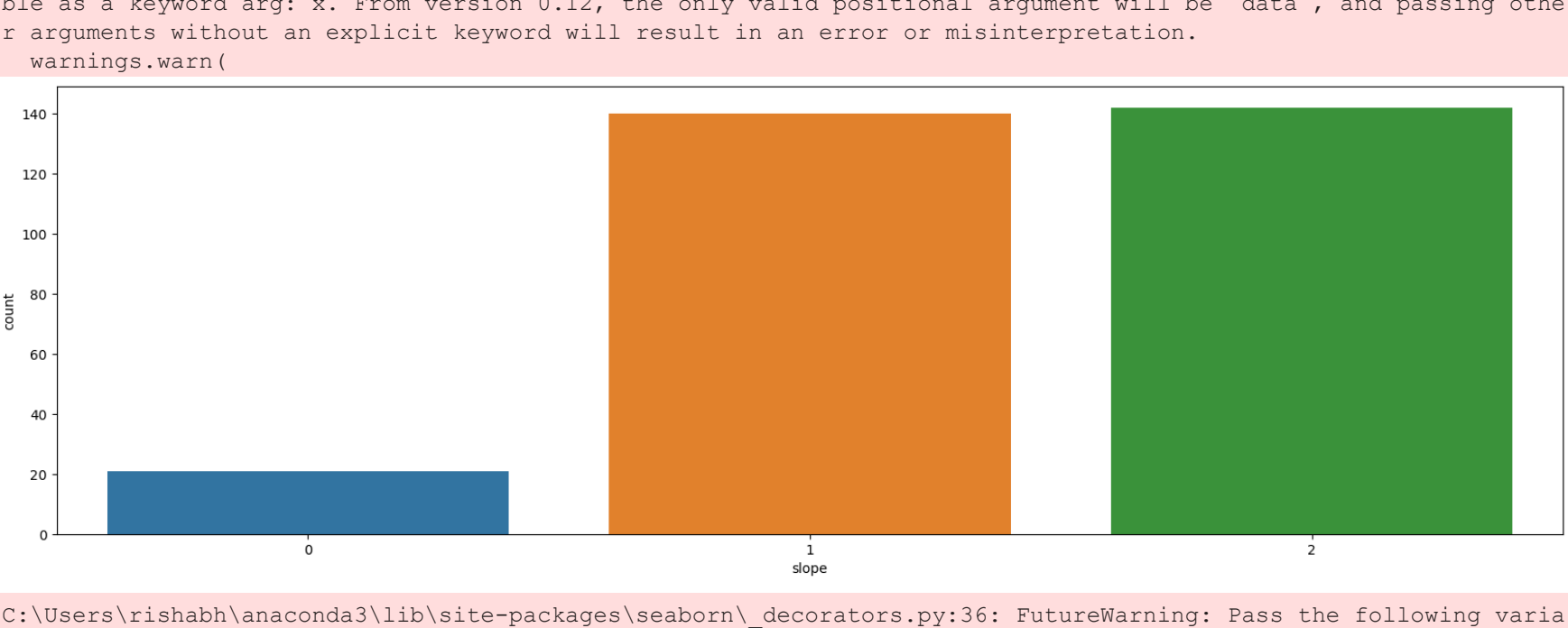

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



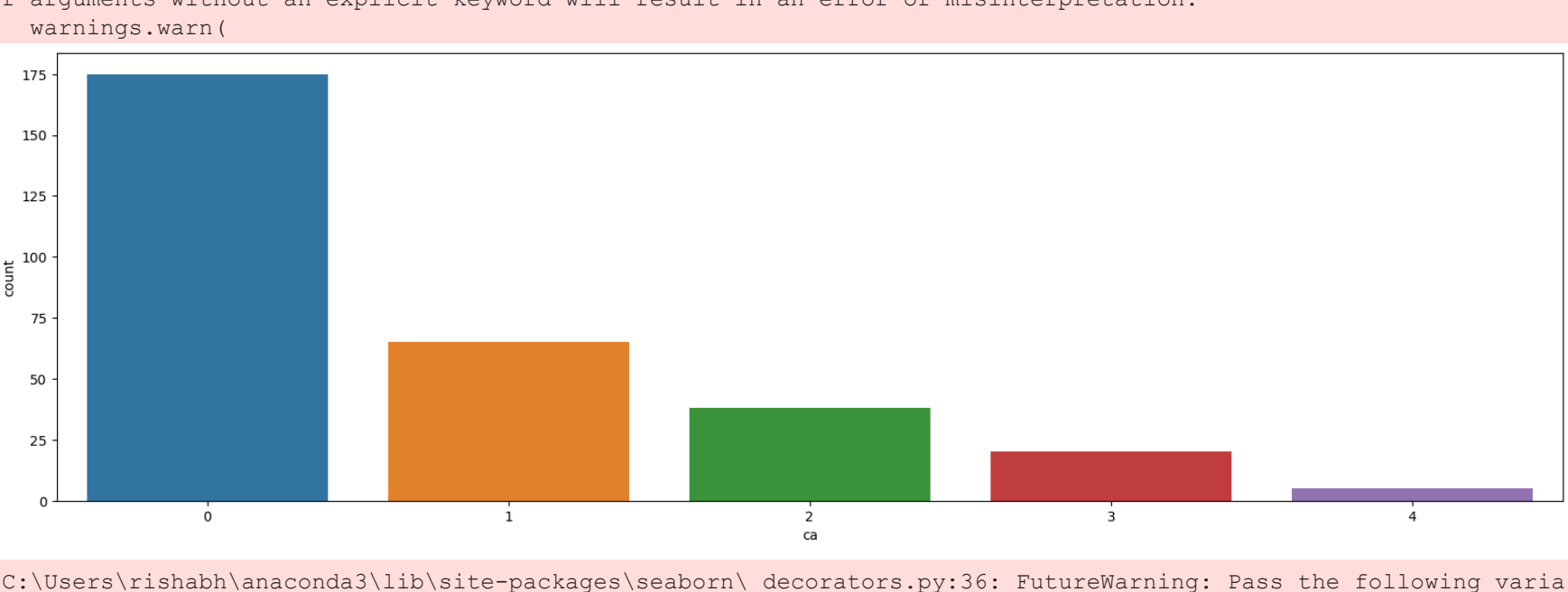

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```




```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

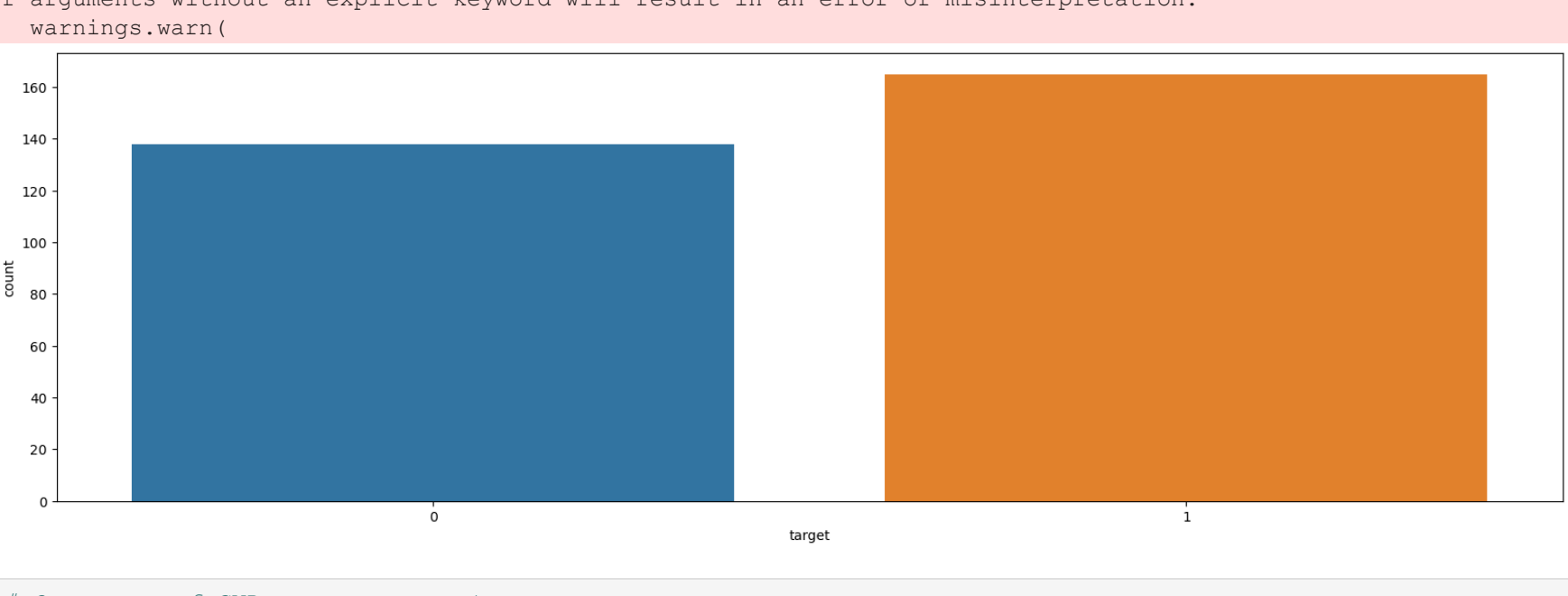



```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

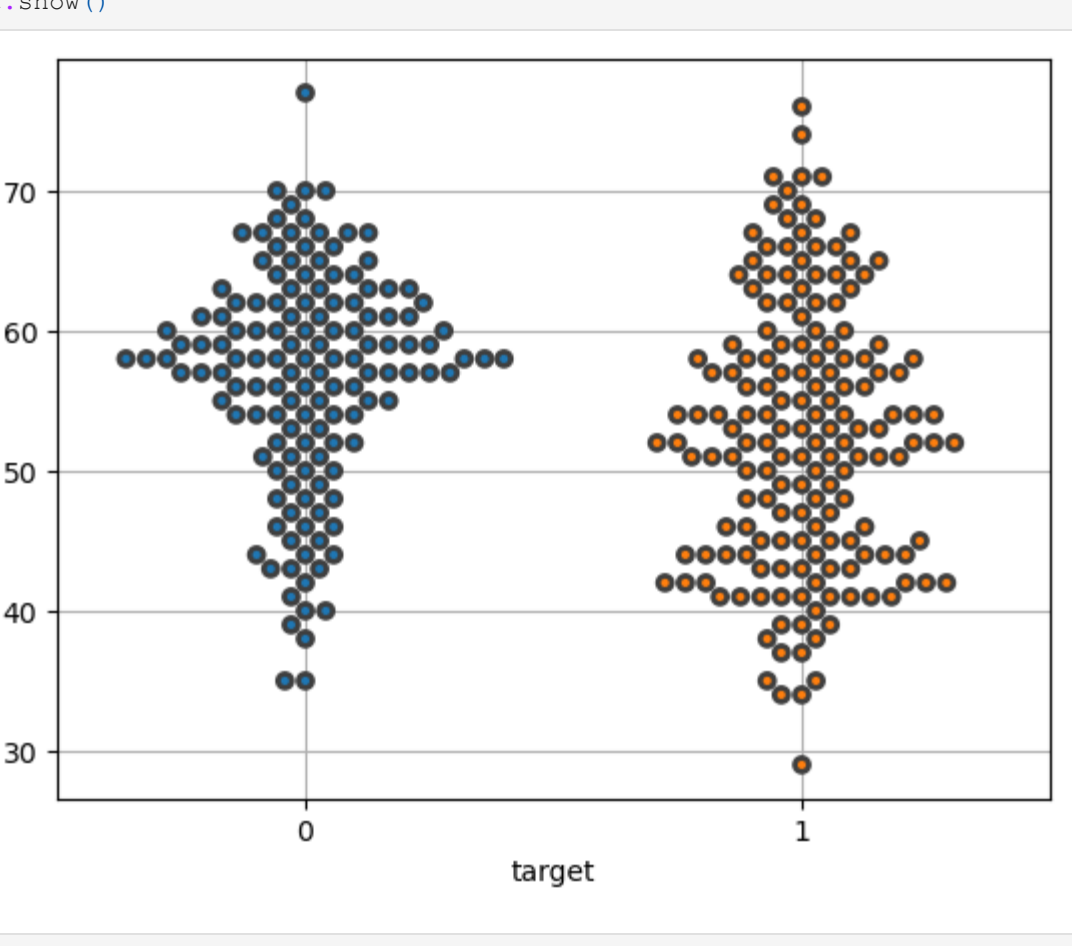



```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```


```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



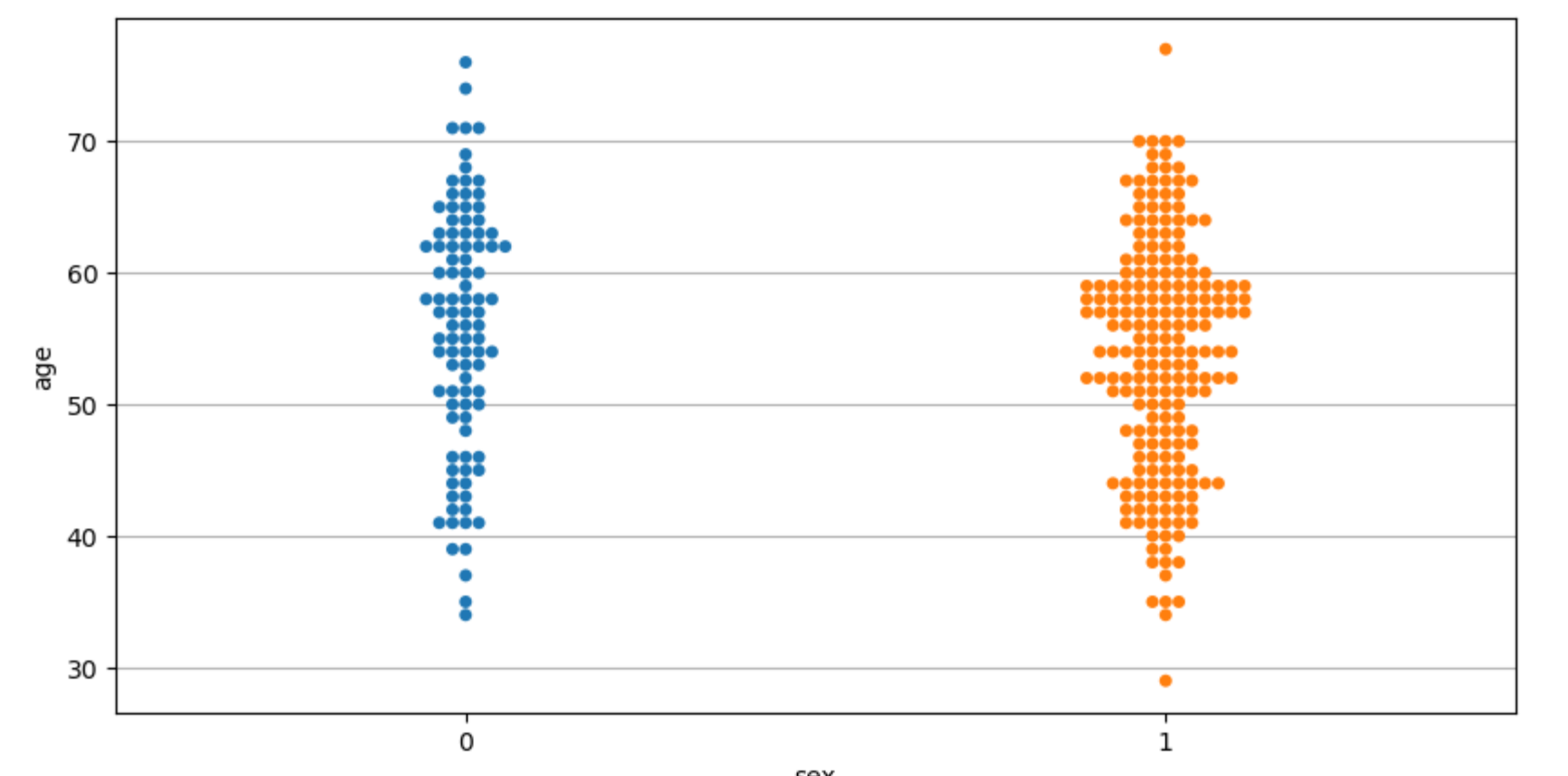
In [17]:
```python
# Occurence of CVD accross age category

sns.swarmplot(x=data['target'], y=data['age'],linewidth=2)
plt.grid()
plt.show()
```



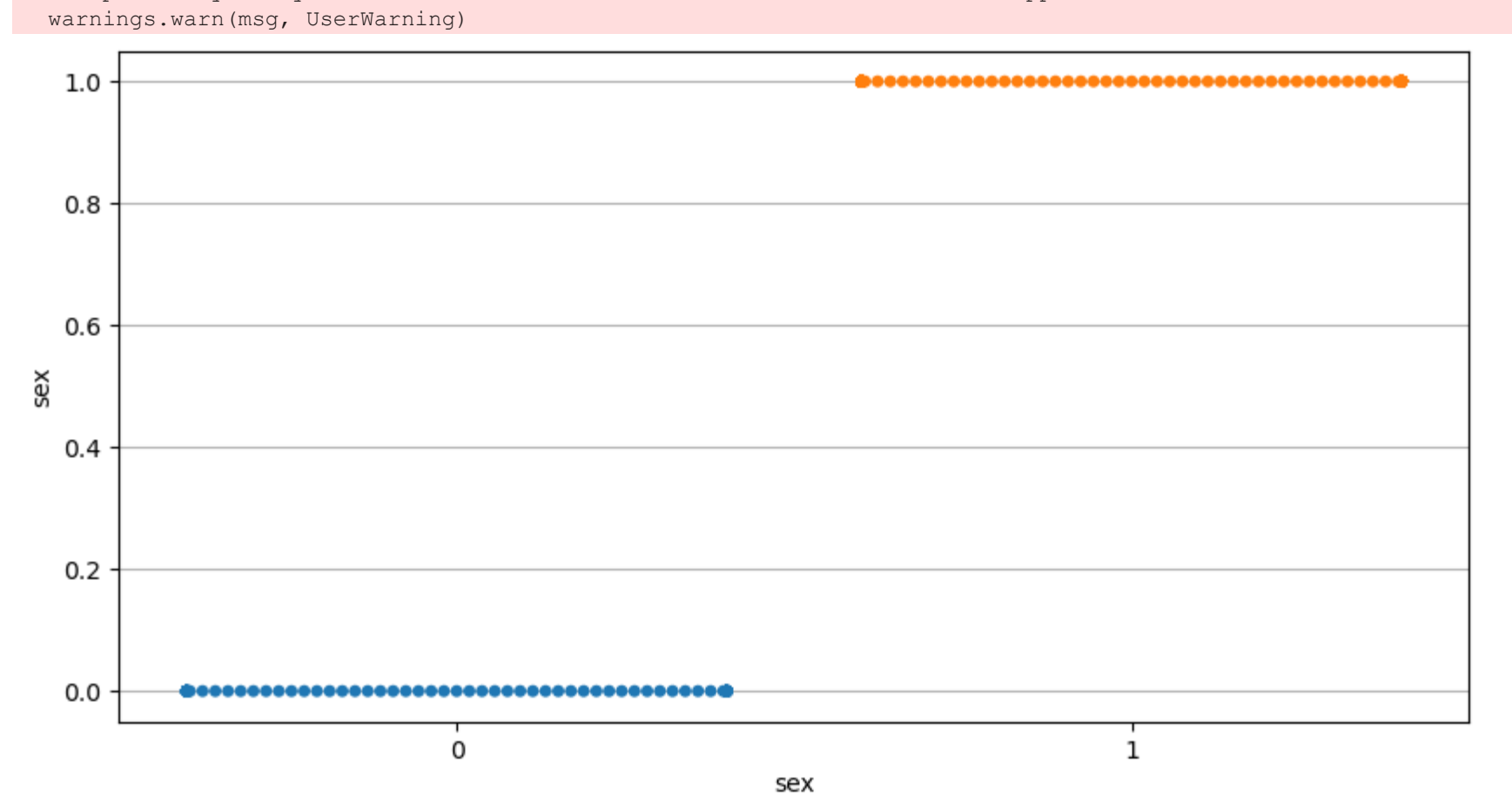
In [18]:
```python
# From the above graph it can be interpreted that that the CVD i most likely to occur in between age 40-60
```

In [19]:
```python
# d.    Study the composition of all patients with respect to the Sex category
```

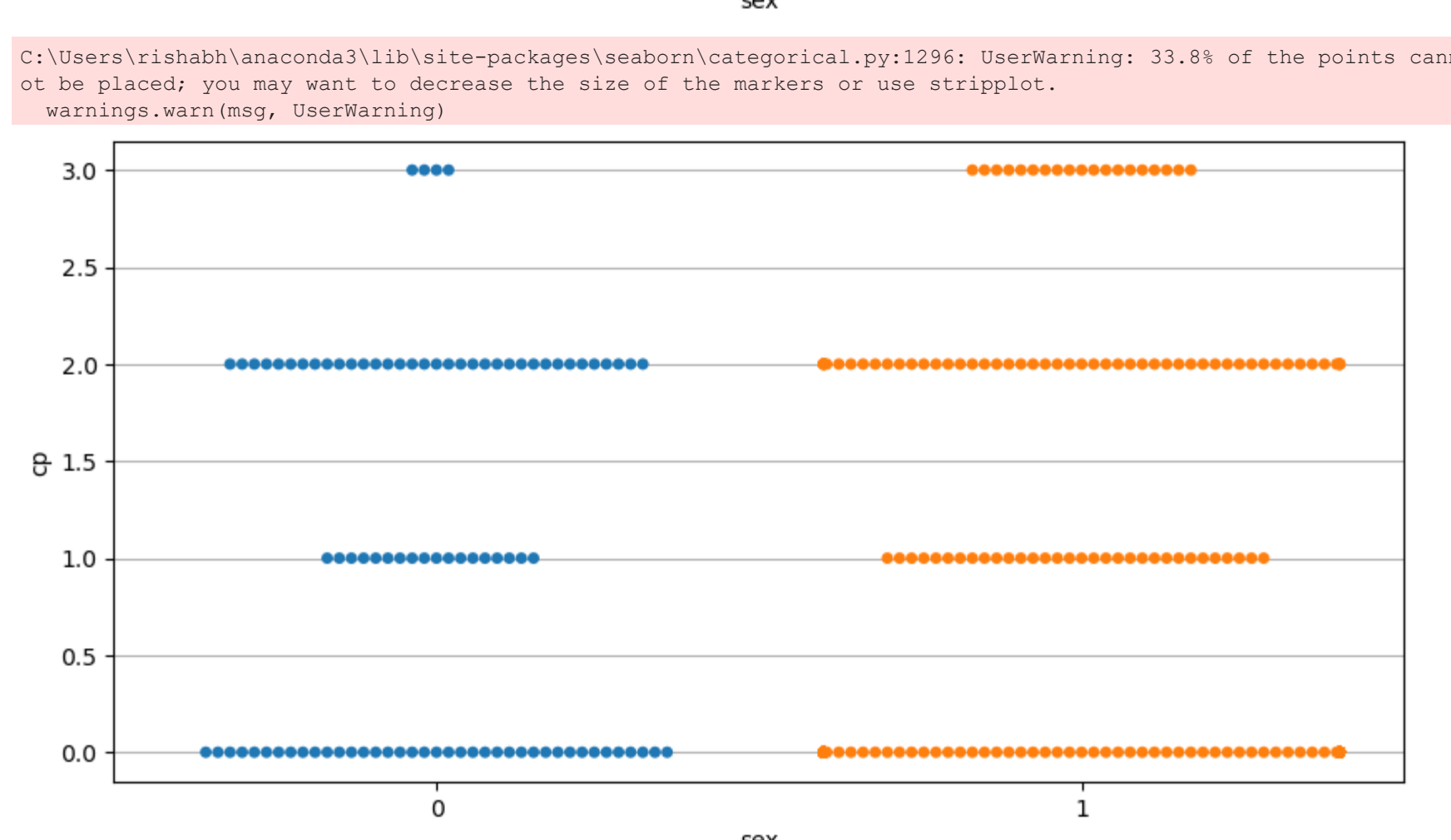
In [20]:
```python
for z in data.columns:

    plt.figure(figsize=(10,5))
    plt.grid()
    sns.swarmplot(x=data["sex"],y=data[z])
    plt.show()
```

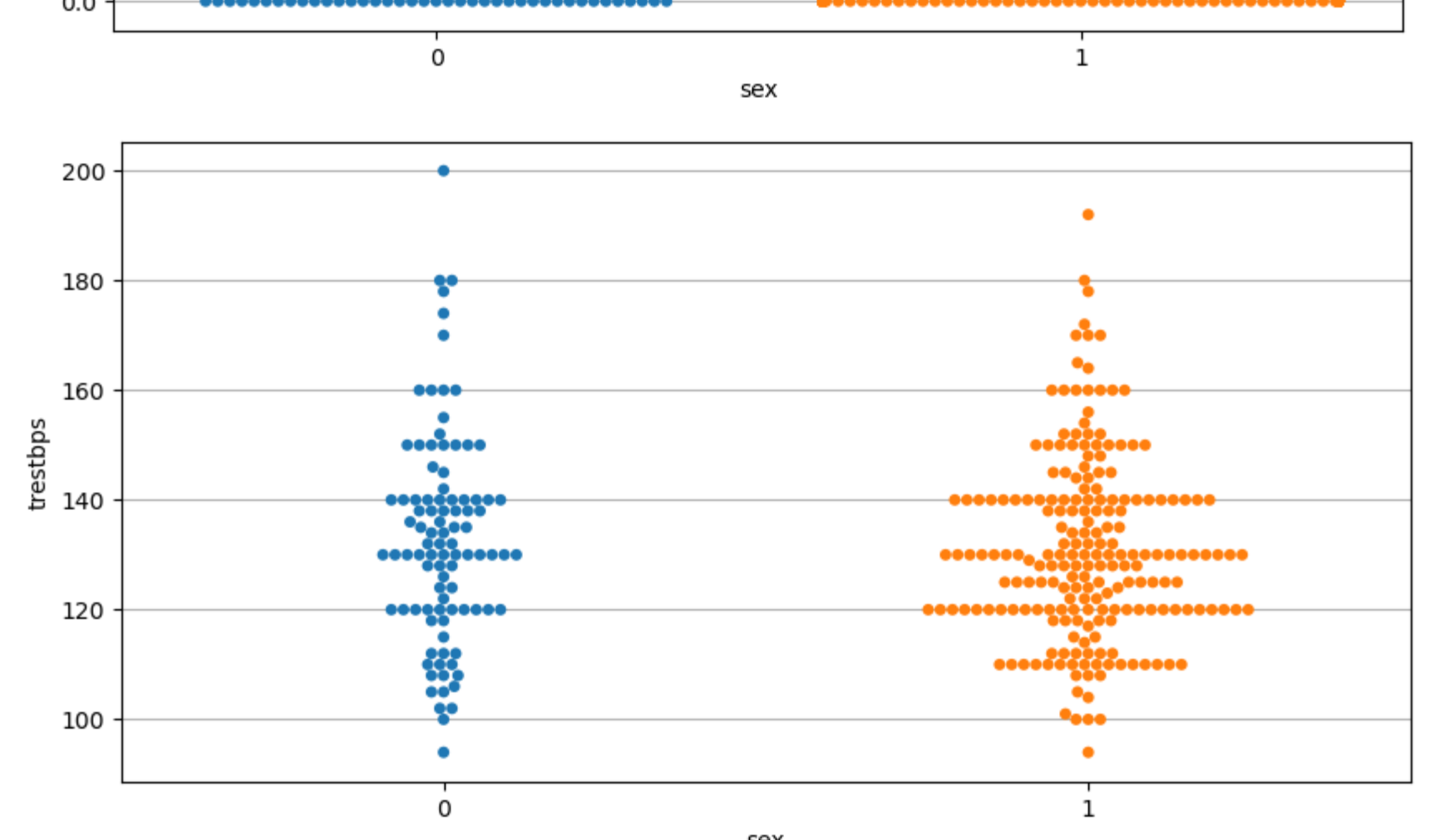


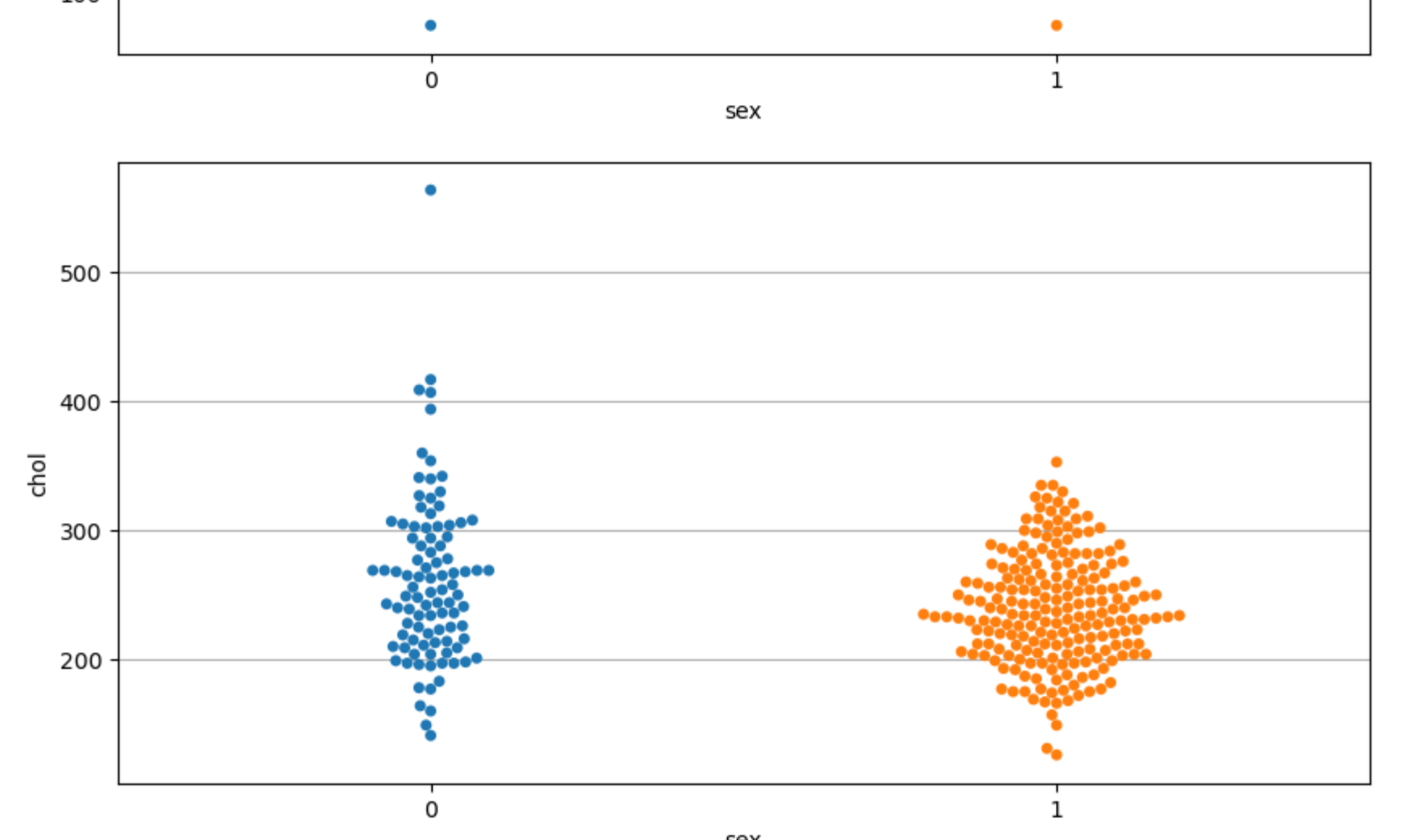

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 55.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 79.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



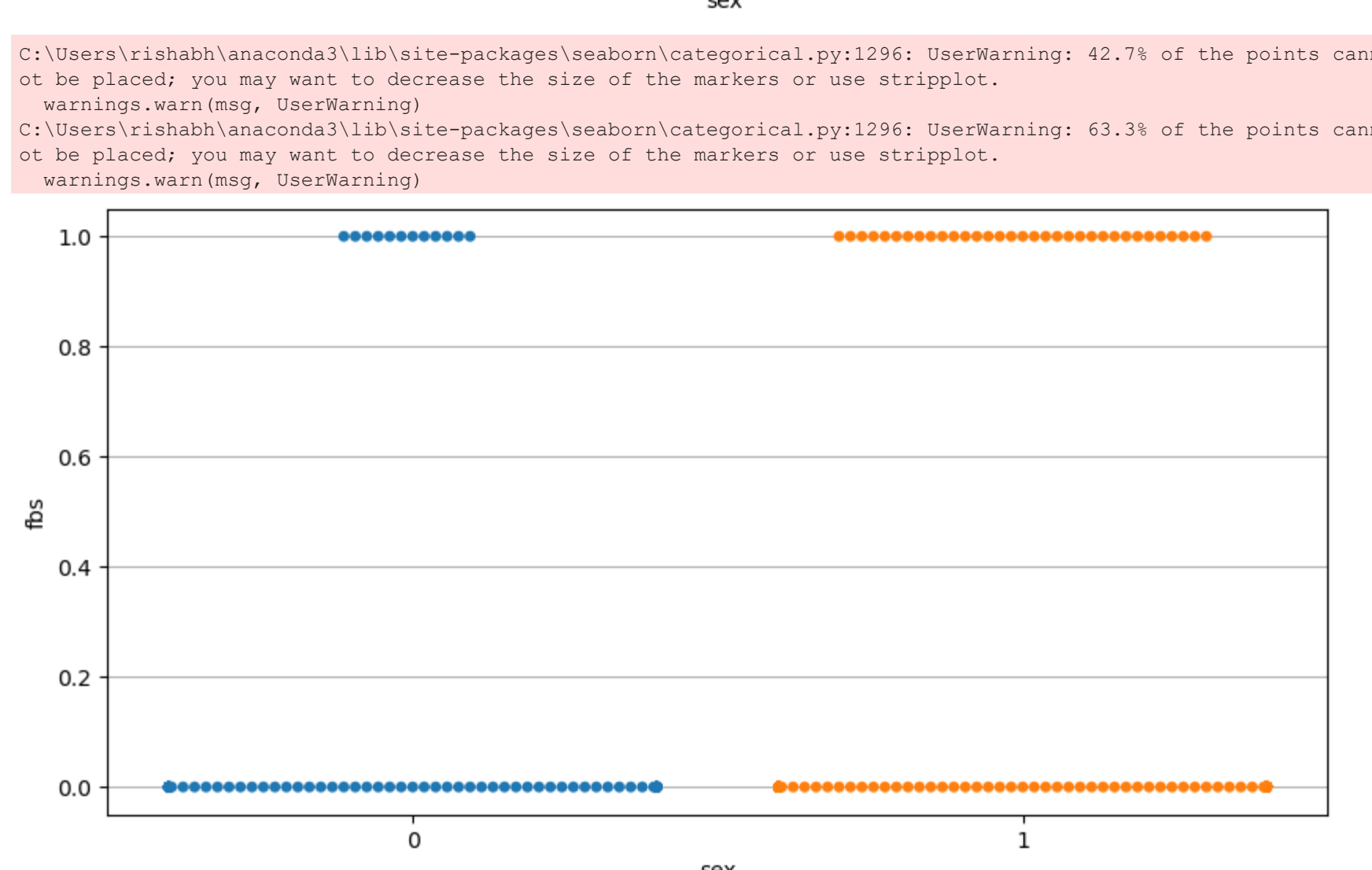

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 33.8% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



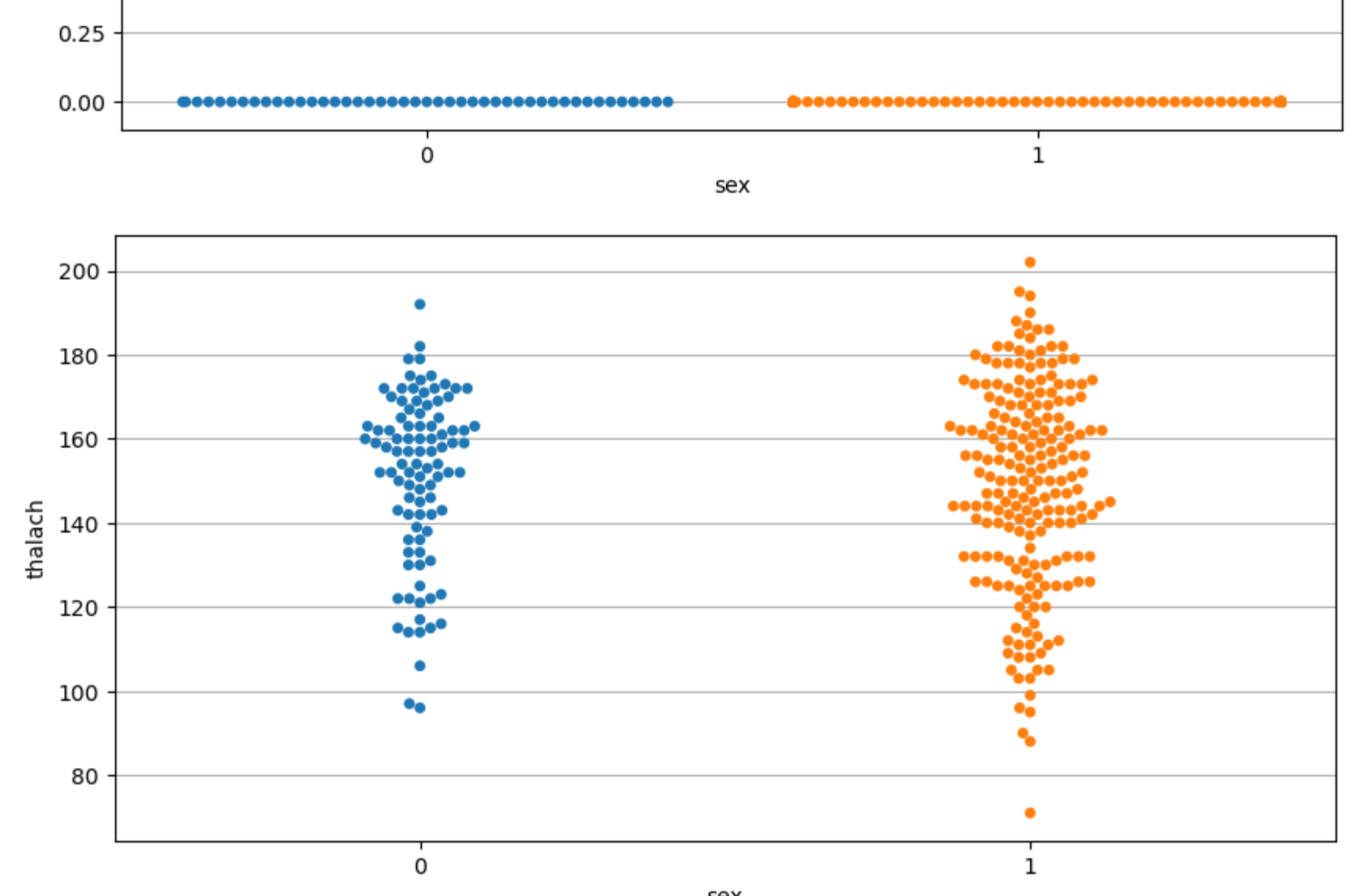




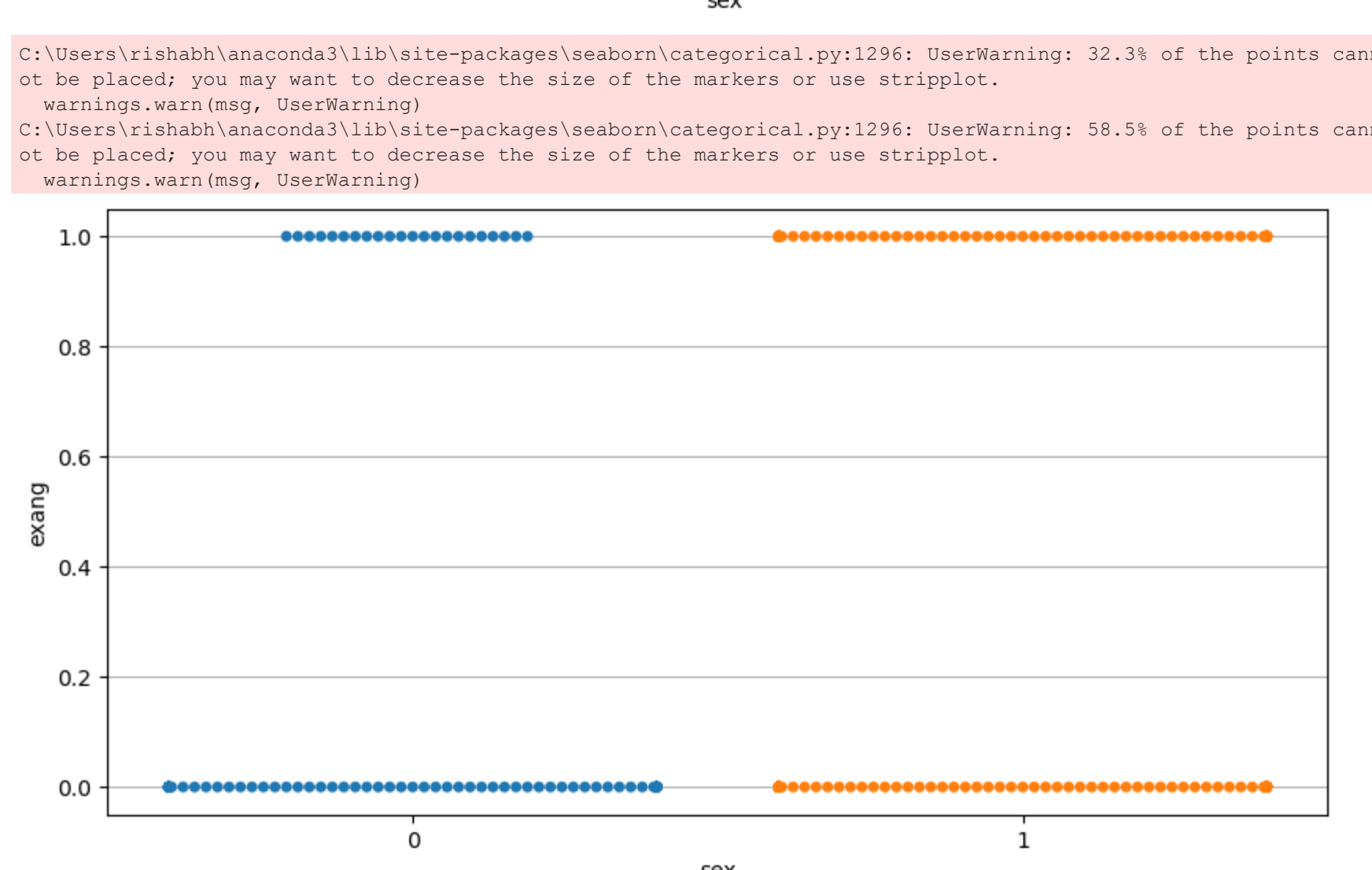

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 42.7% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 63.3% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



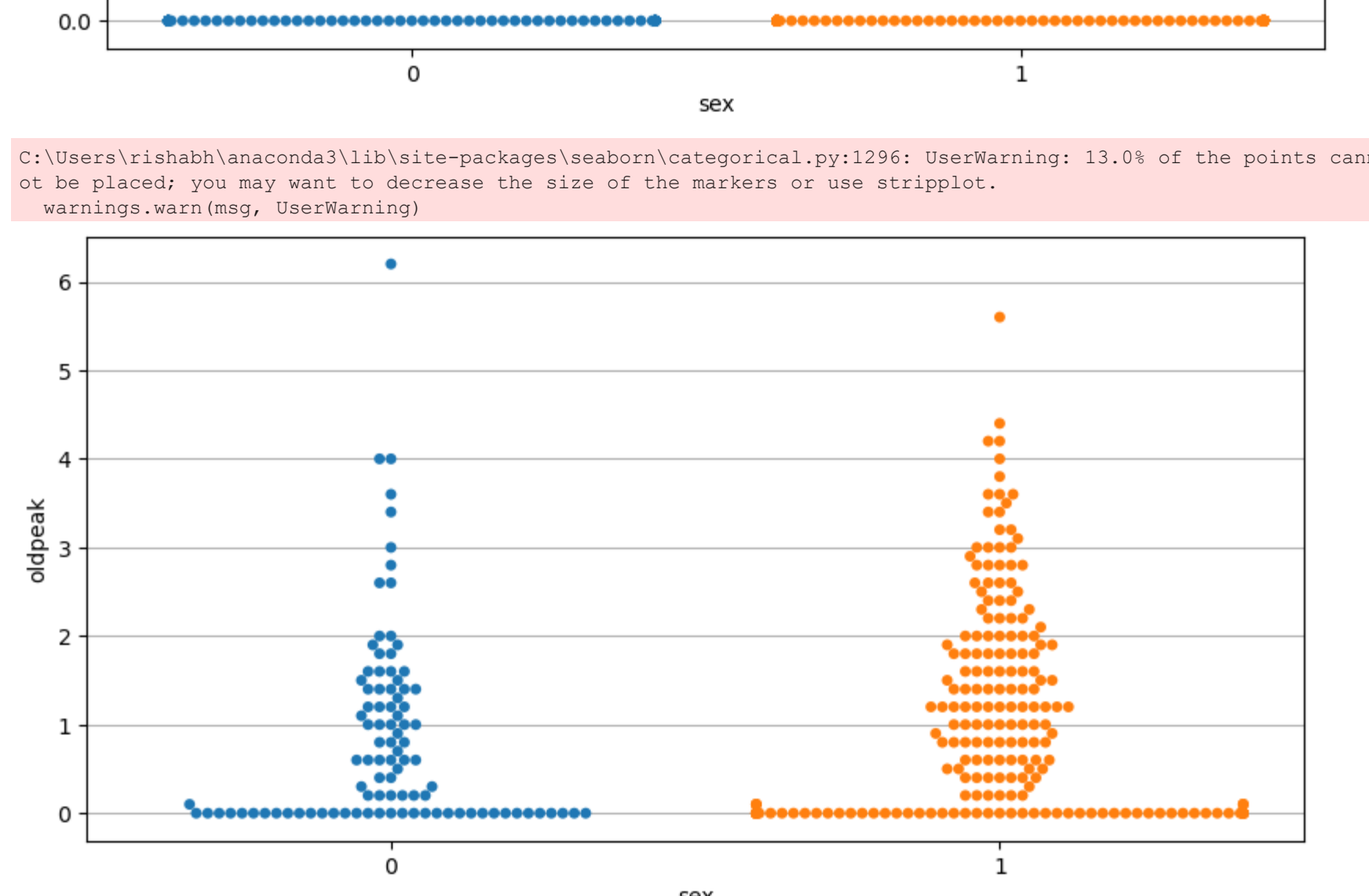

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 7.3% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 58.0% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



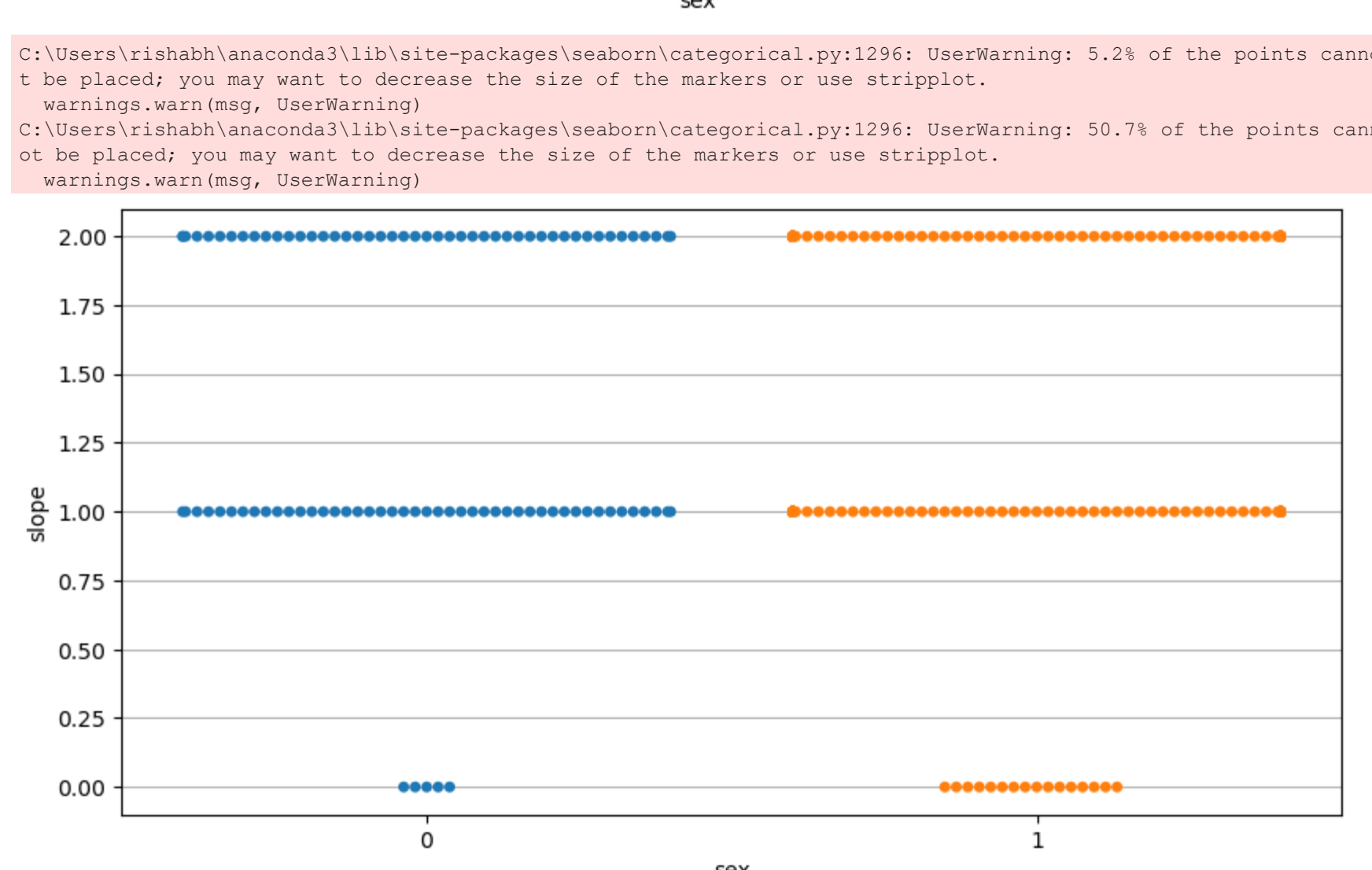

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 32.3% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 58.5% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



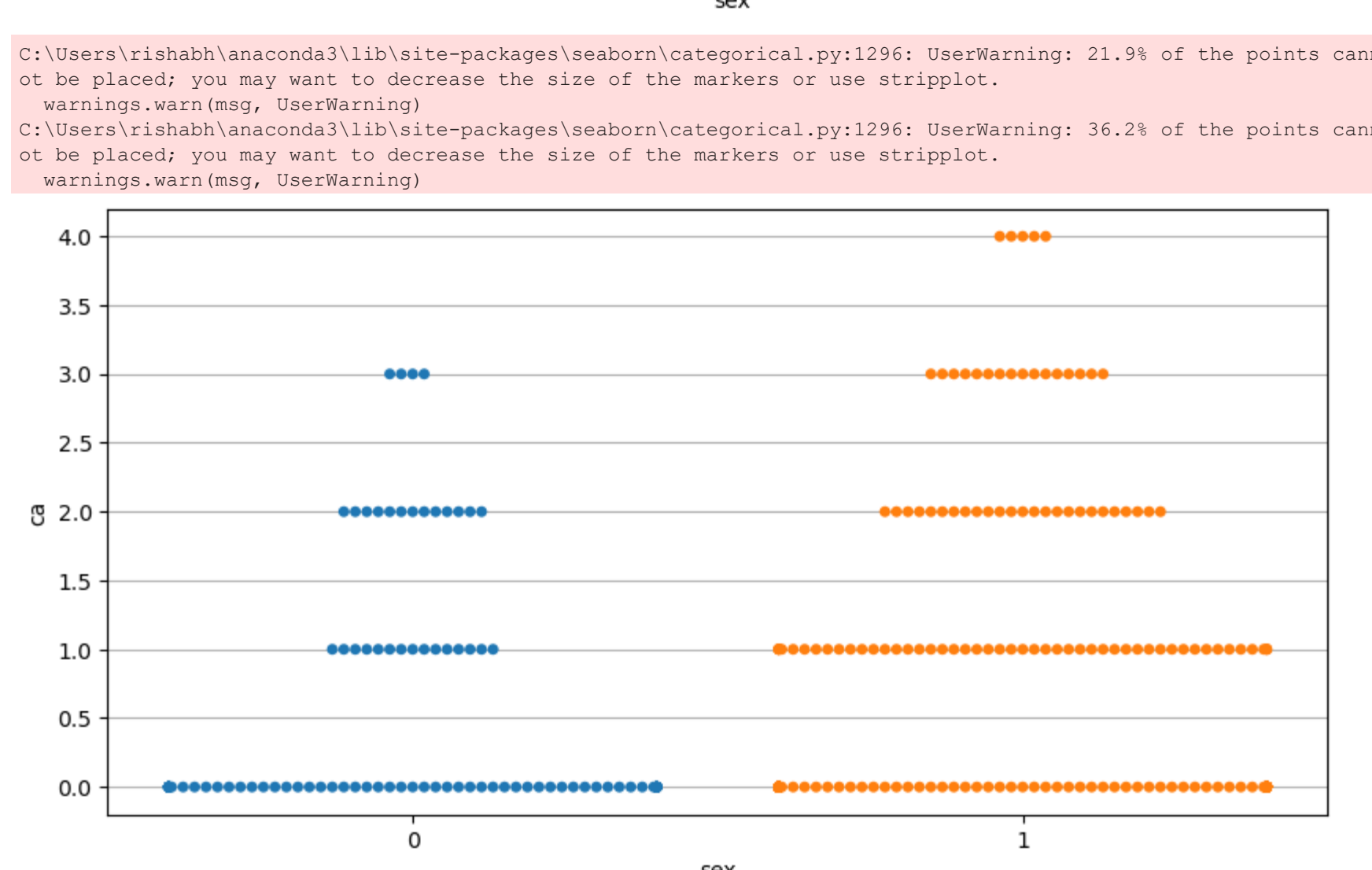

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 13.0% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



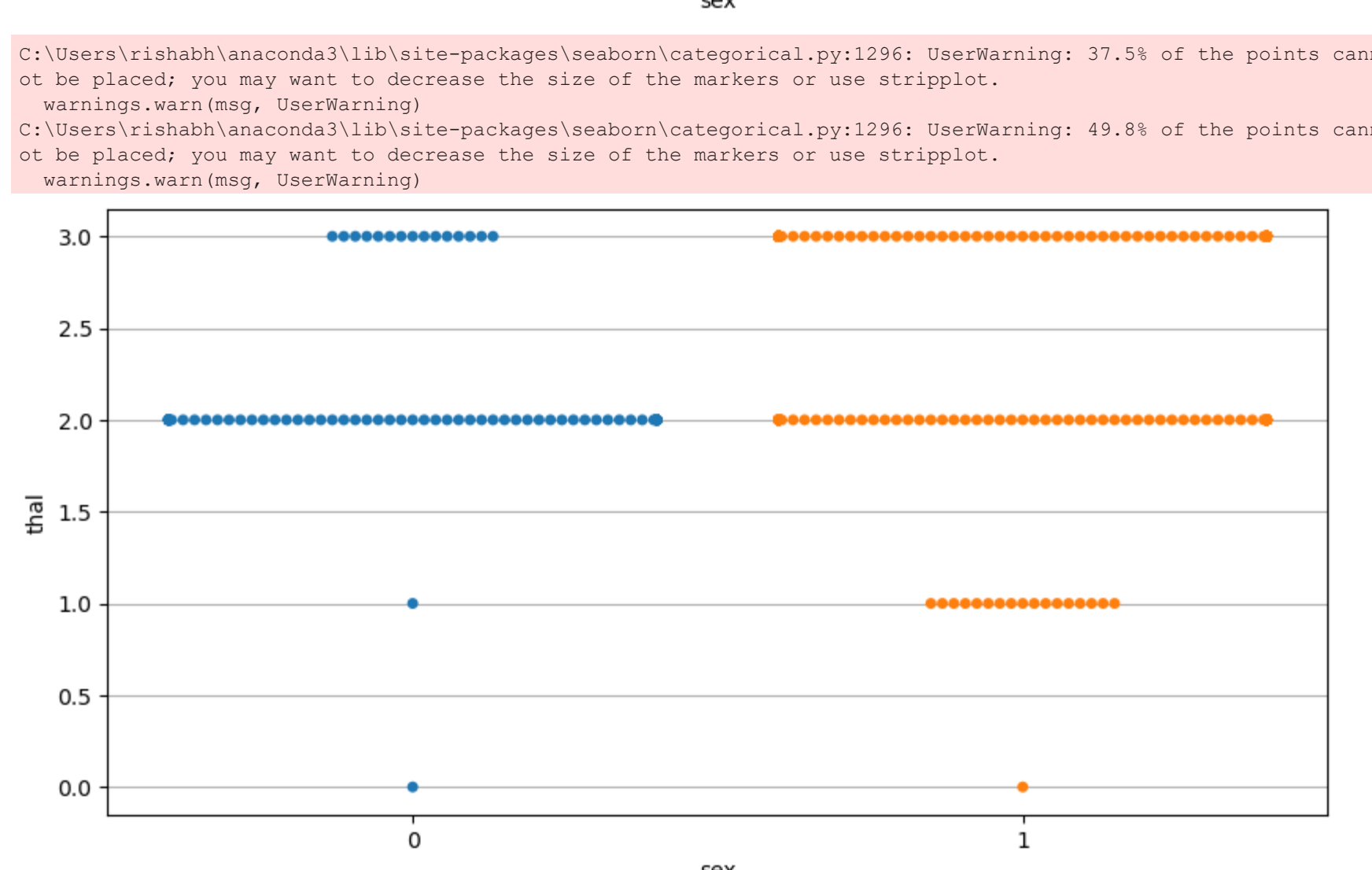

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 5.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 50.7% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



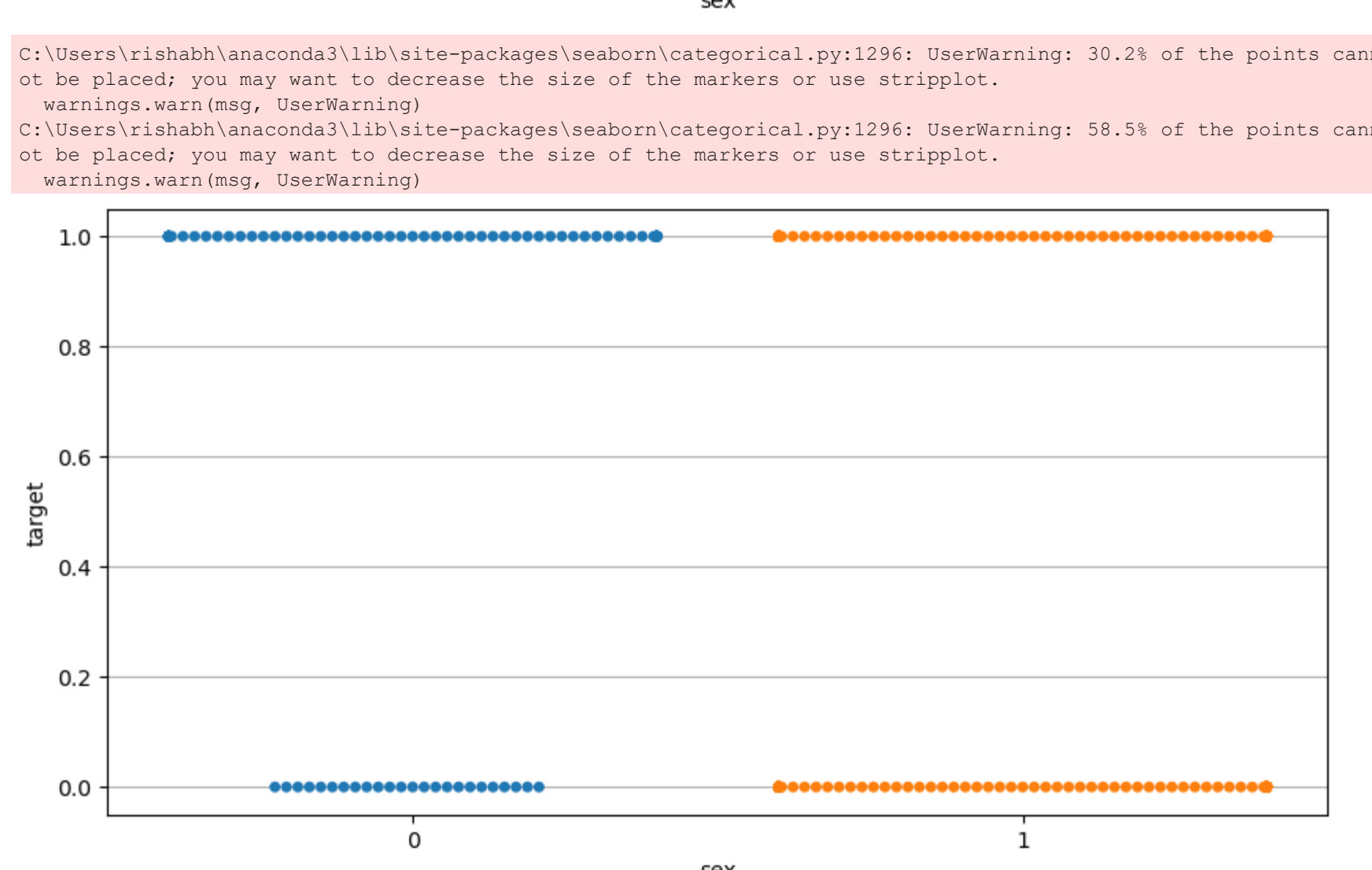

```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 21.9% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 36.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```




```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 37.5% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 49.8% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

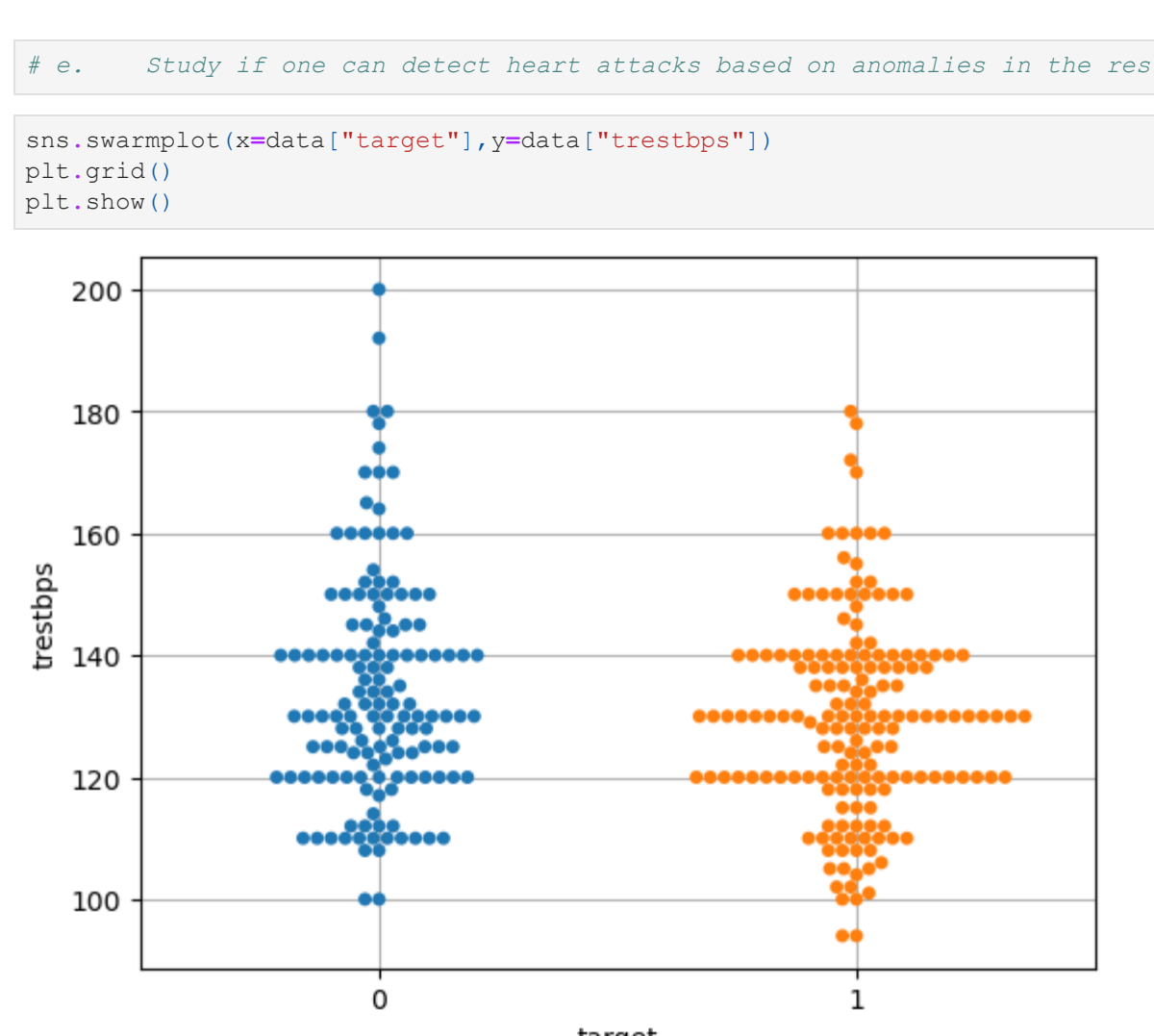



```
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 30.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\rishabh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 58.5% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



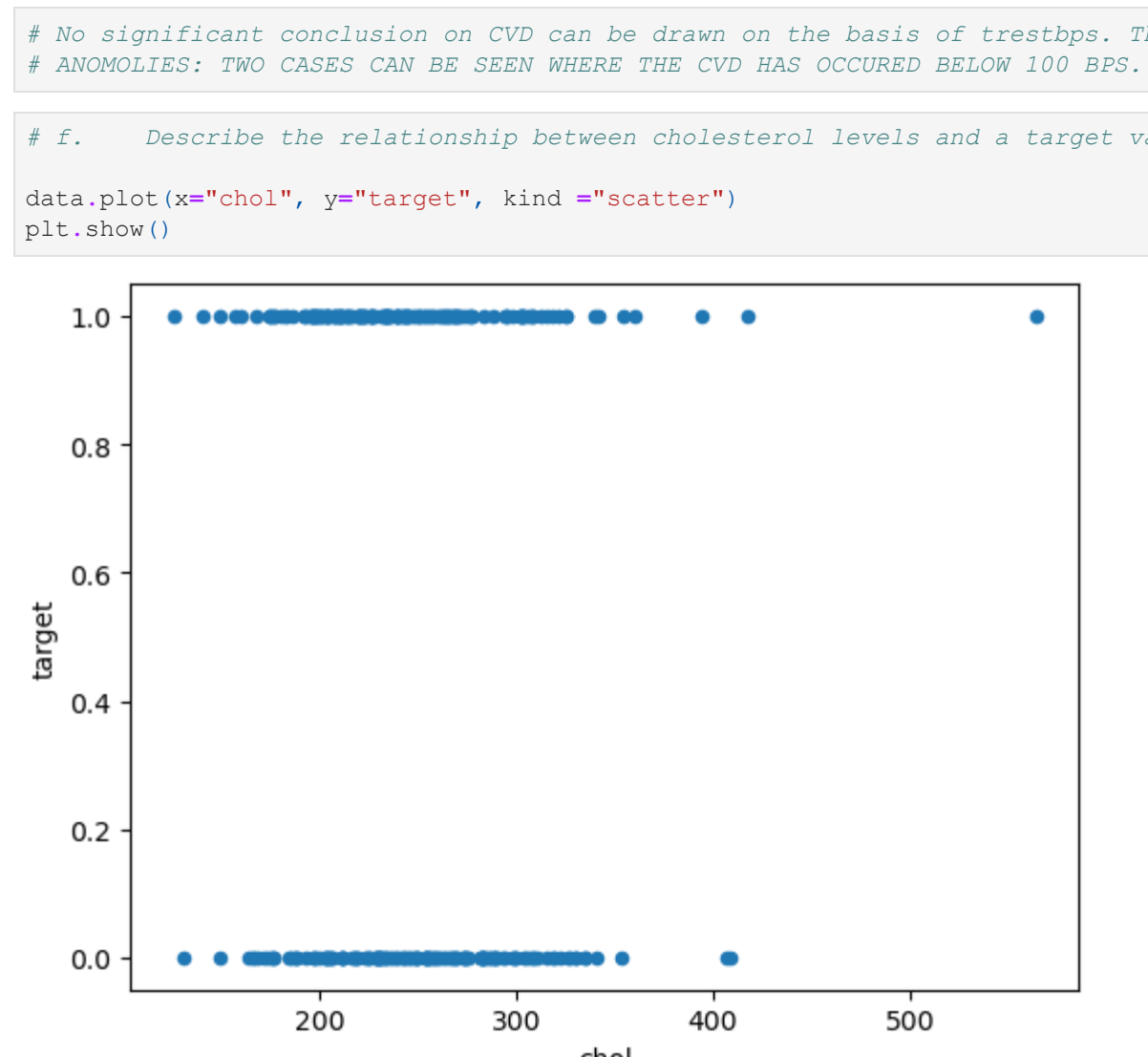
In [21]:
```
# e.    Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a
```

In [22]:
```
sns.swarmplot(x=data["target"],y=data["trestbps"])
plt.grid()
plt.show()
```



In [23]:
```
# No significant conclusion on CVD can be drawn on the basis of trestbps. There is no distinctive readings whic
# ANOMOLIES: TWO CASES CAN BE SEEN WHERE THE CVD HAS OCCURED BELOW 100 BPS.
```

In [24]:
```
# f.    Describe the relationship between cholesterol levels and a target variable

data.plot(x="chol", y="target", kind ="scatter")
plt.show()
```



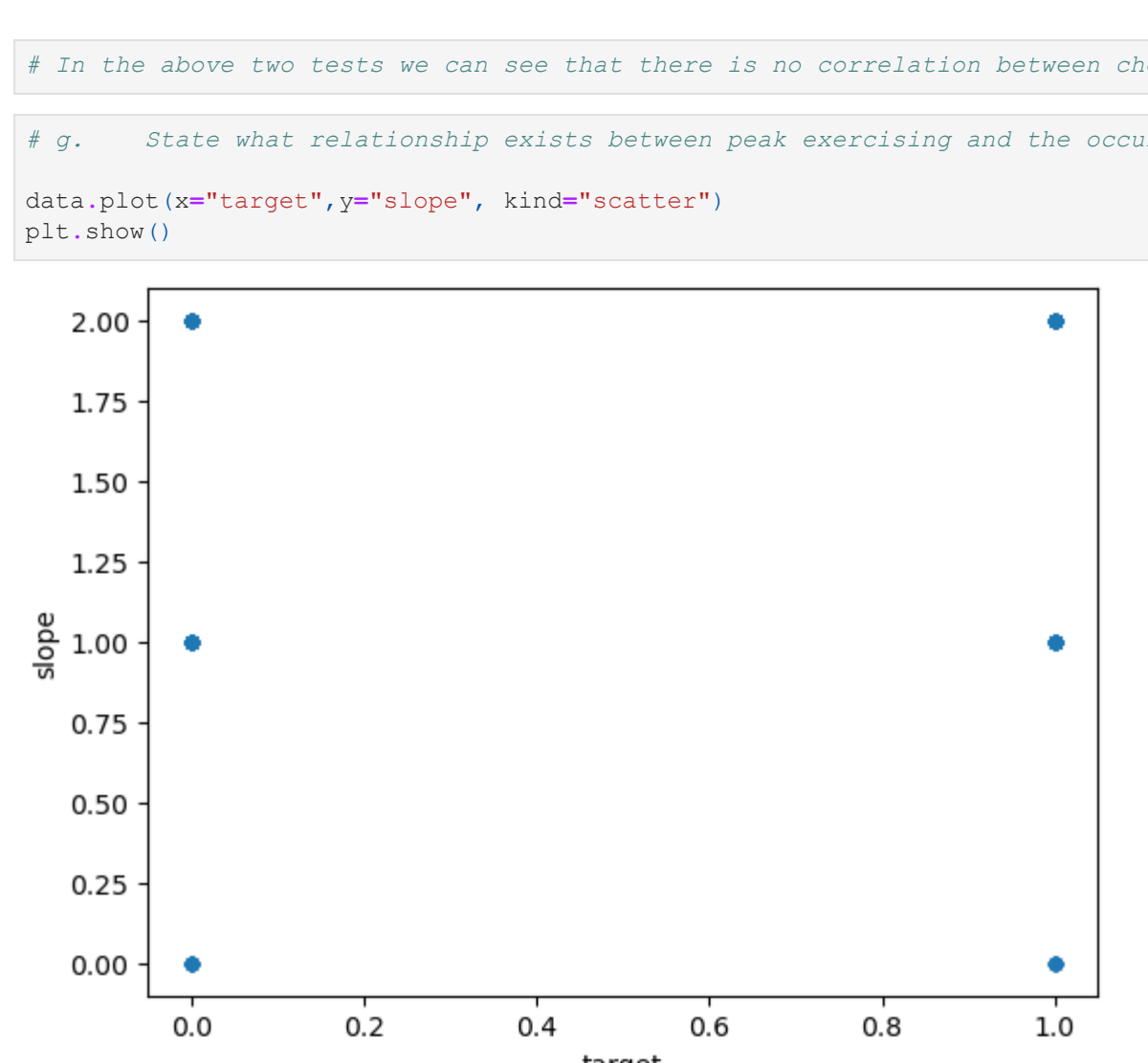
In [25]:
```
data.corr()
```

Out[25]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724

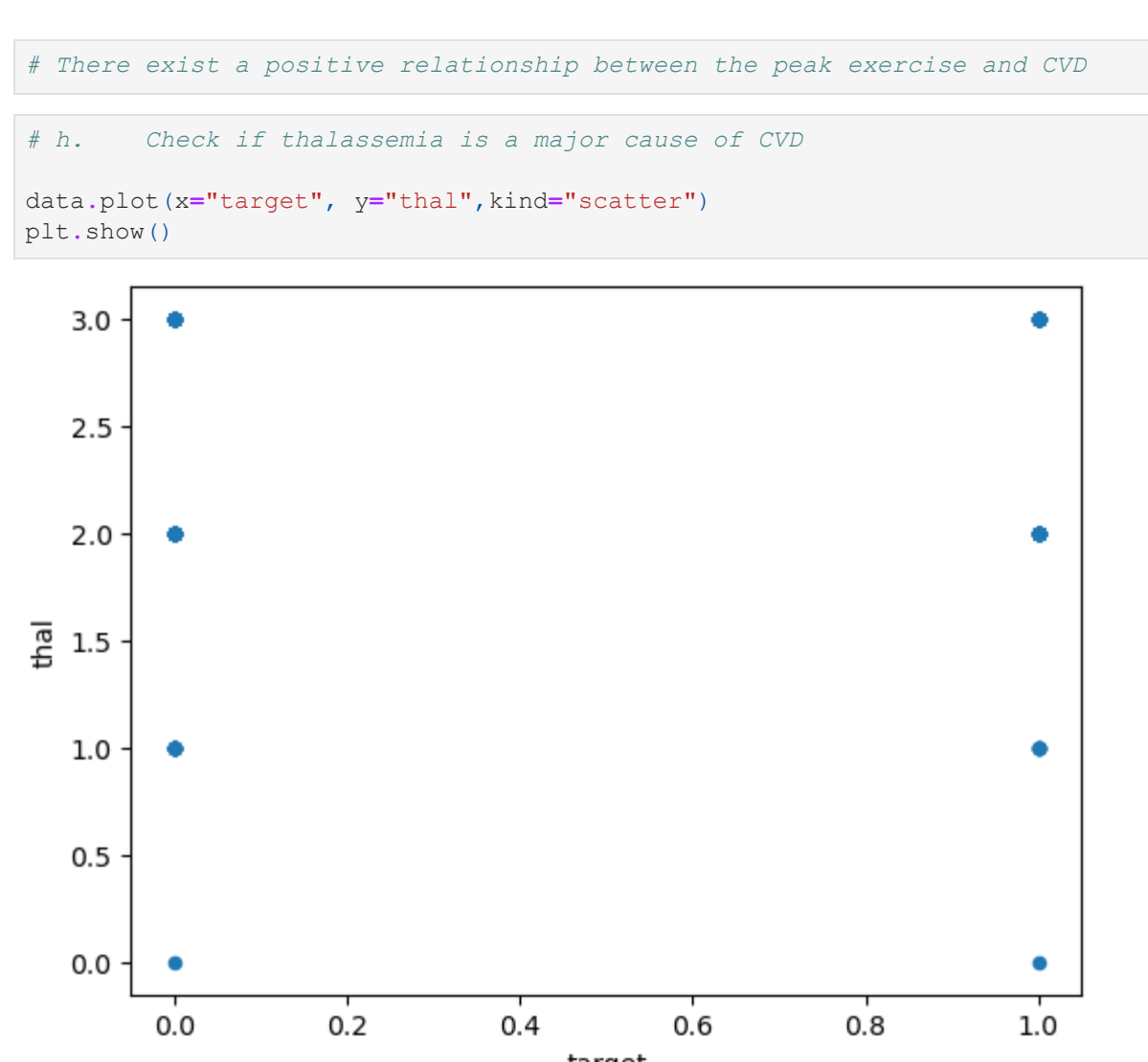
In [26]:
```
# In the above two tests we can see that there is no correlation between cholestrol levels and CVD.
```

In [27]:
```
# g.    State what relationship exists between peak exercising and the occurrence of a heart attack

data.plot(x="target",y="slope", kind="scatter")
plt.show()
```



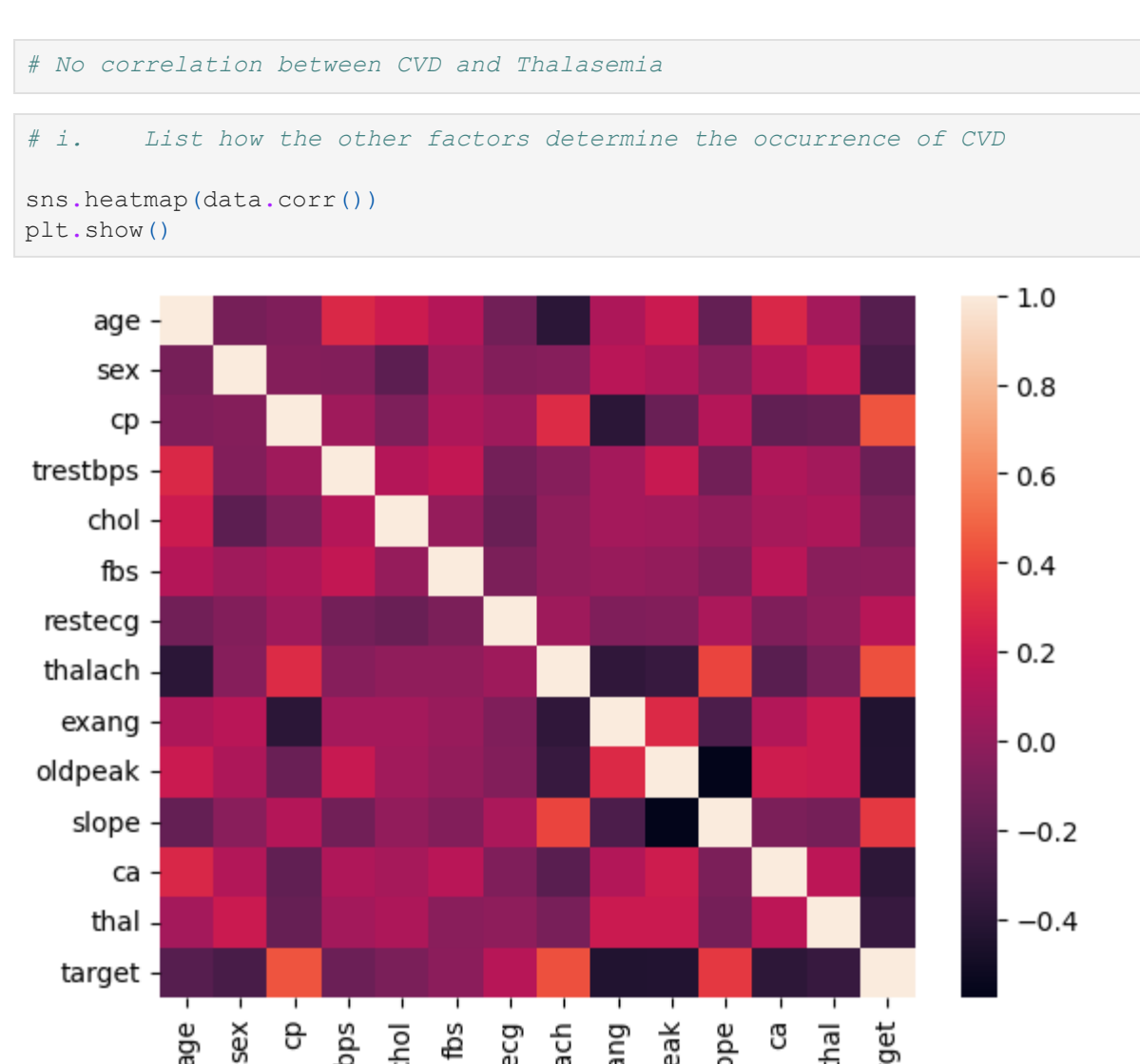
In [28]:
```
# There exist a positive relationship between the peak exercise and CVD
```

In [29]:
```
# h.    Check if thalassemia is a major cause of CVD

data.plot(x="target", y="thal",kind="scatter")
plt.show()
```



In [30]:
```
# No correlation between CVD and Thalasemia
```

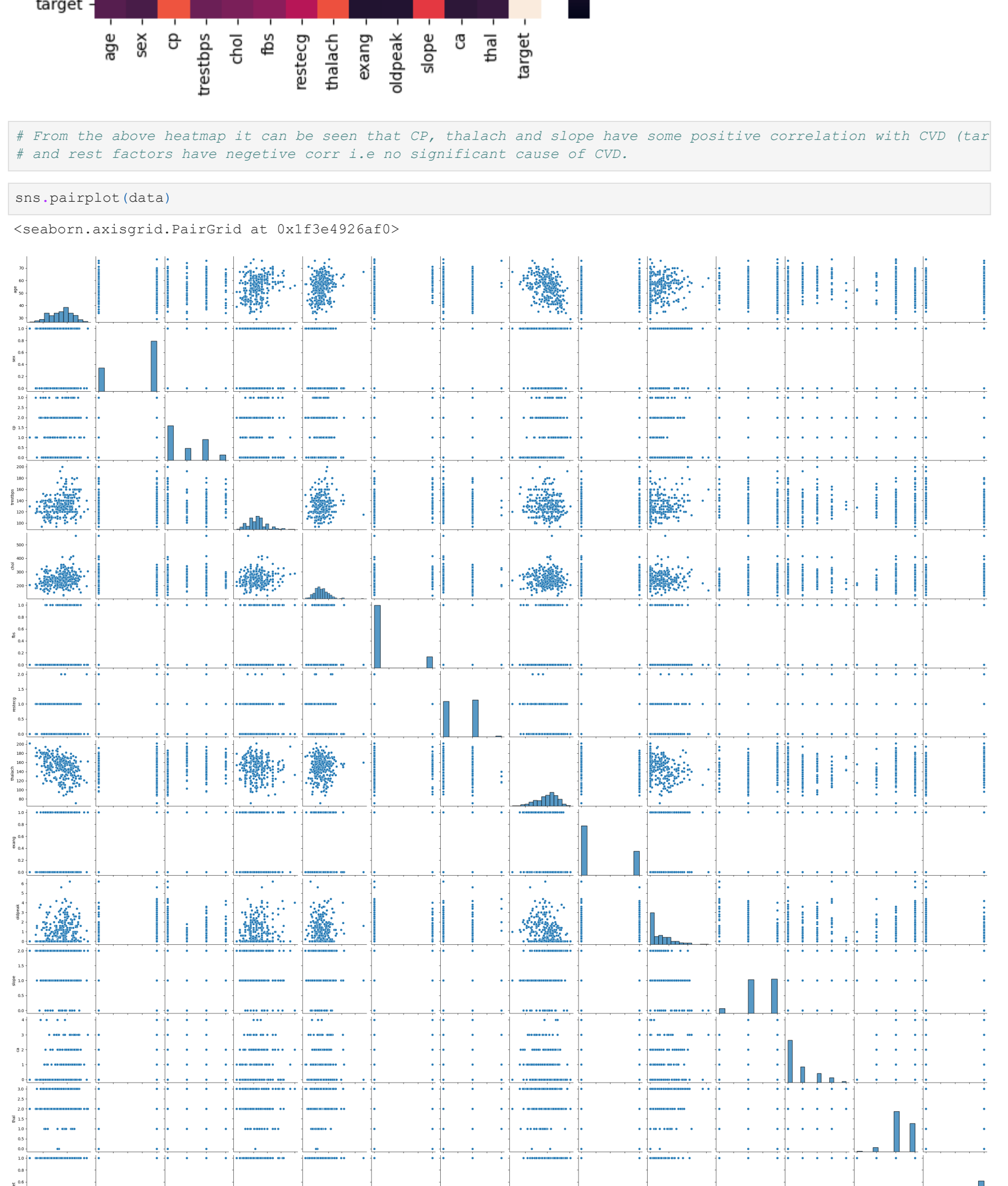
In [31]:
```
# i.    List how the other factors determine the occurrence of CVD

sns.heatmap(data.corr())
plt.show()
```



In [32]:
```
# From the above heatmap it can be seen that CP, thalach and slope have some positive correlation with CVD (tar
# and rest factors have negetive corr i.e no significant cause of CVD.
```

In [33]:
```
sns.pairplot(data)
```

Out[33]:
```
<seaborn.axisgrid.PairGrid at 0x1f3e4926af0>
```



Building the Logistic Regression Model

In [34]:
```
# 1. Splitting the data into X and y
X = data.drop("target",axis=1)
y= data["target"]
```

In [35]:
```
X.head()
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

In [36]:
```
# Separating the data into train and test dataset
```

In [37]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test =train_test_split(X,y, test_size=0.3,random_state=21)
```

In [38]:
```
print("Shape of X_train is : ",X_train.shape)
print("Shape of X_test is : ",X_test.shape)
print("Shape of y_train is : ",y_train.shape)
print("Shape of y_test is : ",y_test.shape)
```

```
Shape of X_train is :  (212, 13)
Shape of X_test is :  (91, 13)
Shape of y_train is :  (212,)
Shape of y_test is :  (91,)
```

In [39]:
```
# Builing the logistic Regression Model

from sklearn.linear_model import LogisticRegression
logreg_model = LogisticRegression(solver="newton-cg")
```

In [40]:
```
logreg_model.fit(X_train,y_train)
```

Out[40]:
```
LogisticRegression(solver='newton-cg')
```

In [41]:
```
y_pred_test = logreg_model.predict(X_test)
```

In [42]:
```
# Evaluate the model
```

In [43]:
```
from sklearn.metrics import accuracy_score

print("The accuracy score of the log reg model on the test dataset is : ")
print(accuracy_score(y_test,y_pred_test)) #actual values, predicted values
```

```
The accuracy score of the log reg model on the test dataset is :
0.8241758241758241
```

In [44]:
```
y_pred_train = logreg_model.predict(X_train)
```

In [45]:
```
print("The accuracy score of the log reg model on the train dataset is : ")
print(accuracy_score(y_train,y_pred_train)) #actual values, predicted values
```

```
The accuracy score of the log reg model on the train dataset is :
0.8584905660377359
```

In [46]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_test)) # classification report on test dataset
```

```
              precision    recall  f1-score   support

           0       0.80      0.84      0.82        43
           1       0.85      0.81      0.83        48

    accuracy                           0.82        91
   macro avg       0.82      0.82      0.82        91
weighted avg       0.83      0.82      0.82        91
```

In [47]:
```
print(classification_report(y_train, y_pred_train)) # classification report on train dataset
```

```
              precision    recall  f1-score   support

           0       0.90      0.77      0.83        95
           1       0.83      0.93      0.88       117

    accuracy                           0.86       212
   macro avg       0.87      0.85      0.85       212
weighted avg       0.86      0.86      0.86       212
```

Training Random Forest Model

In [48]:
```
from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier(n_estimators=600)
rfc.fit(X_train,y_train)
```

Out[48]:
```
RandomForestClassifier(n_estimators=600)
```

In [49]:
```
y_pred_test1 = rfc.predict(X_test)
```

In [50]:
```
print("The accuracy score of the random model on the test dataset is : ")
print(accuracy_score(y_test,y_pred_test1)) #actual values, predicted values
```

```
The accuracy score of the random model on the test dataset is :
0.8351648351648352
```

In [51]:
```
y_pred_train1 = rfc.predict(X_train)
```

In [52]:
```
print("The accuracy score of the random forest model on the train dataset is : ")
print(accuracy_score(y_train,y_pred_train1)) #actual values, predicted values
```

```
The accuracy score of the random forest model on the train dataset is :
1.0
```

In [53]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_test1)) # classification report on test dataset
```

```
              precision    recall  f1-score   support

           0       0.79      0.88      0.84        43
           1       0.88      0.79      0.84        48

    accuracy                           0.84        91
   macro avg       0.84      0.84      0.84        91
weighted avg       0.84      0.84      0.84        91
```

In [54]:
```
print(classification_report(y_train, y_pred_train1)) # classification report on train dataset
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        95
           1       1.00      1.00      1.00       117

    accuracy                           1.00       212
   macro avg       1.00      1.00      1.00       212
weighted avg       1.00      1.00      1.00       212
```

In [55]:
```
# It can be concluded that Random forest is best suited model for the above case
```