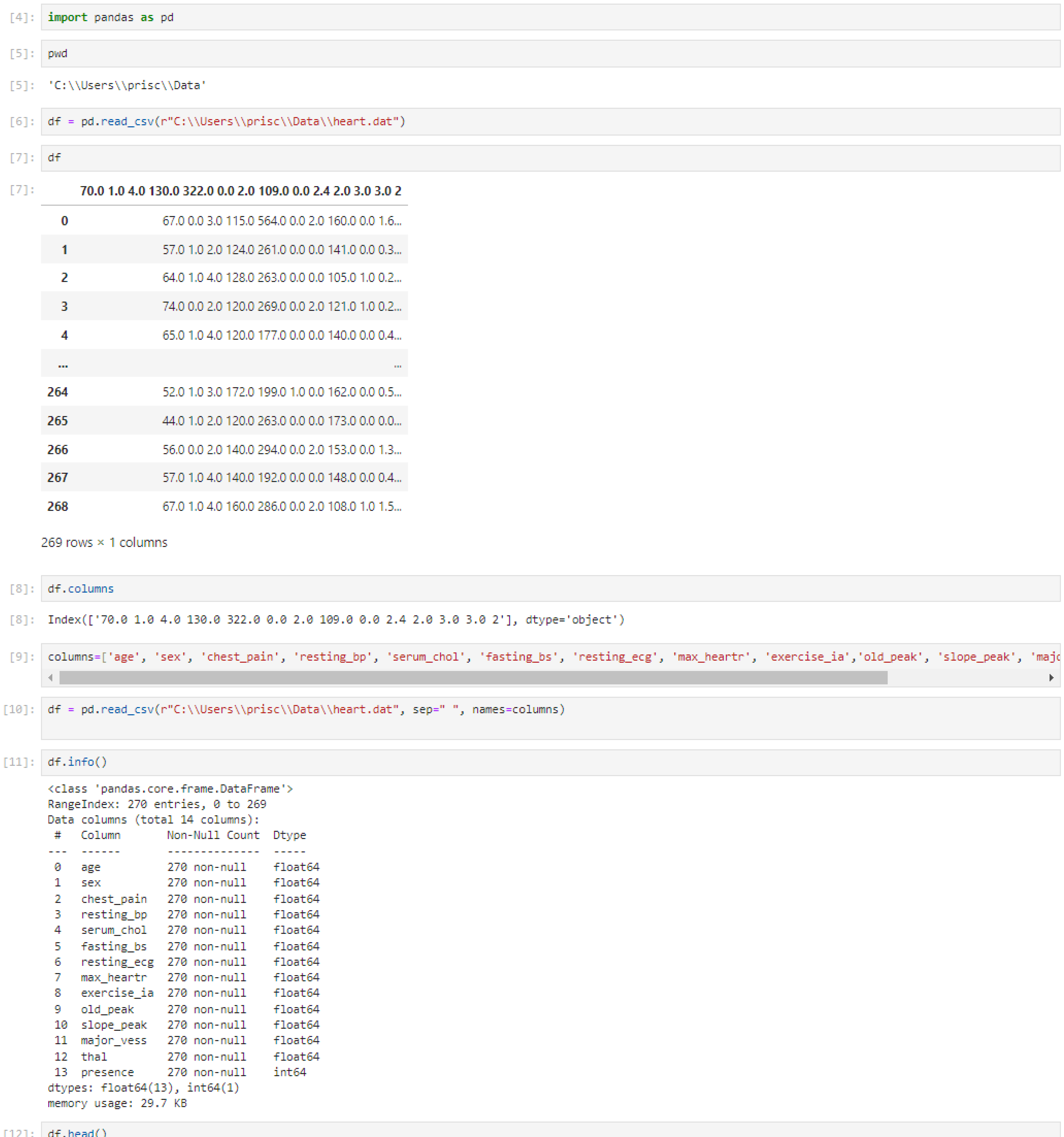


[4]:
```
import pandas as pd
```

[5]:
```
pwd
```

[5]:
```
'C:\\Users\\prisc\\Data'
```

[6]:
```
df = pd.read_csv(r"C:\\Users\\prisc\\Data\\heart.dat")
```

[7]:
```
df
```

[7]:

	70.0 1.0 4.0 130.0 322.0 0.0 2.0 109.0 0.0 2.4 2.0 3.0 3.0 2
0	67.0 0.0 3.0 115.0 564.0 0.0 2.0 160.0 0.0 1.6...
1	57.0 1.0 2.0 124.0 261.0 0.0 0.0 141.0 0.0 0.3...
2	64.0 1.0 4.0 128.0 263.0 0.0 0.0 105.0 1.0 0.2...
3	74.0 0.0 2.0 120.0 269.0 0.0 2.0 121.0 1.0 0.2...
4	65.0 1.0 4.0 120.0 177.0 0.0 0.0 140.0 0.0 0.4...
...	...
264	52.0 1.0 3.0 172.0 199.0 1.0 0.0 162.0 0.0 0.5...
265	44.0 1.0 2.0 120.0 263.0 0.0 0.0 173.0 0.0 0.0...
266	56.0 0.0 2.0 140.0 294.0 0.0 2.0 153.0 0.0 1.3...
267	57.0 1.0 4.0 140.0 192.0 0.0 0.0 148.0 0.0 0.4...
268	67.0 1.0 4.0 160.0 286.0 0.0 2.0 108.0 1.0 1.5...

269 rows × 1 columns

[8]:
```
df.columns
```

[8]:
```
Index(['70.0 1.0 4.0 130.0 322.0 0.0 2.0 109.0 0.0 2.4 2.0 3.0 3.0 2'], dtype='object')
```

[9]:
```
columns=['age', 'sex', 'chest_pain', 'resting_bp', 'serum_chol', 'fasting_bs', 'resting_ecg', 'max_heartr', 'exercise_ia','old_peak', 'slope_peak', 'majc
```

[10]:
```
df = pd.read_csv(r"C:\\Users\\prisc\\Data\\heart.dat", sep=" ", names=columns)
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          270 non-null    float64
 1   sex          270 non-null    float64
 2   chest_pain   270 non-null    float64
 3   resting_bp   270 non-null    float64
 4   serum_chol   270 non-null    float64
 5   fasting_bs   270 non-null    float64
 6   resting_ecg  270 non-null    float64
 7   max_heartr   270 non-null    float64
 8   exercise_ia  270 non-null    float64
 9   old_peak     270 non-null    float64
 10  slope_peak   270 non-null    float64
 11  major_vess   270 non-null    float64
 12  thal         270 non-null    float64
 13  presence     270 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 29.7 KB
```

[12]:
```
df.head()
```

[12]:
```
df.head()
```

[12]:

	age	sex	chest_pain	resting_bp	serum_chol	fasting_bs	resting_ecg	max_heartr	exercise_ia	old_peak	slope_peak	major_vess	thal	presence
0	70.0	1.0	4.0	130.0	322.0	0.0	2.0	109.0	0.0	2.4	2.0	3.0	3.0	2
1	67.0	0.0	3.0	115.0	564.0	0.0	2.0	160.0	0.0	1.6	2.0	0.0	7.0	1
2	57.0	1.0	2.0	124.0	261.0	0.0	0.0	141.0	0.0	0.3	1.0	0.0	7.0	2
3	64.0	1.0	4.0	128.0	263.0	0.0	0.0	105.0	1.0	0.2	2.0	1.0	7.0	1
4	74.0	0.0	2.0	120.0	269.0	0.0	2.0	121.0	1.0	0.2	1.0	1.0	3.0	1

[13]:
```
df.tail()
```

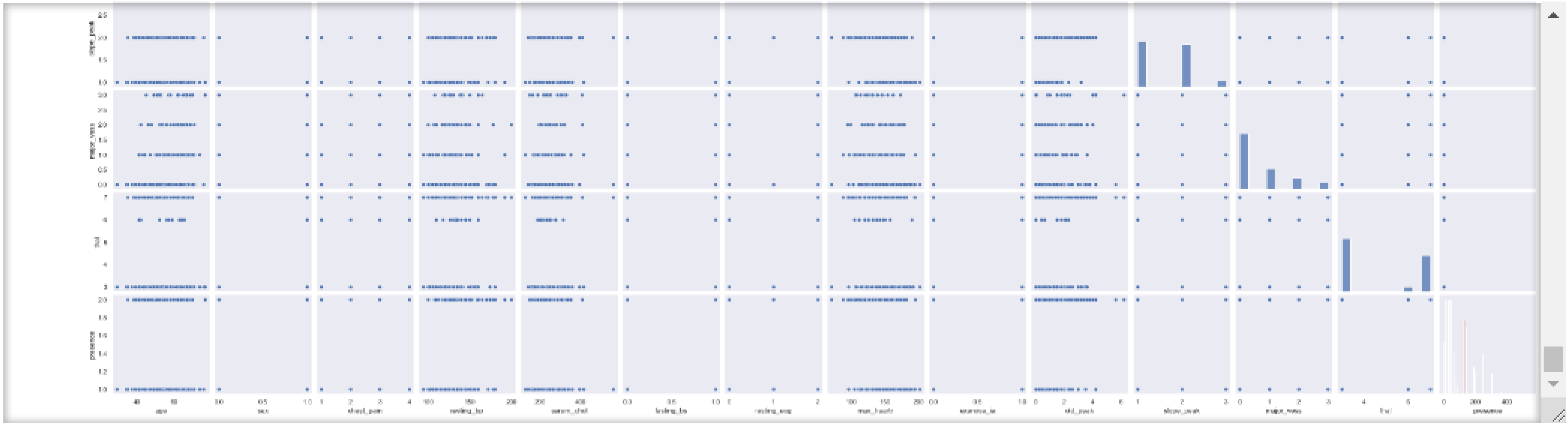
[13]:

	age	sex	chest_pain	resting_bp	serum_chol	fasting_bs	resting_ecg	max_heartr	exercise_ia	old_peak	slope_peak	major_vess	thal	presence
265	52.0	1.0	3.0	172.0	199.0	1.0	0.0	162.0	0.0	0.5	1.0	0.0	7.0	1
266	44.0	1.0	2.0	120.0	263.0	0.0	0.0	173.0	0.0	0.0	1.0	0.0	7.0	1
267	56.0	0.0	2.0	140.0	294.0	0.0	2.0	153.0	0.0	1.3	2.0	0.0	3.0	1
268	57.0	1.0	4.0	140.0	192.0	0.0	0.0	148.0	0.0	0.4	2.0	0.0	6.0	1
269	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2

[14]:
```
print(df.shape)
```

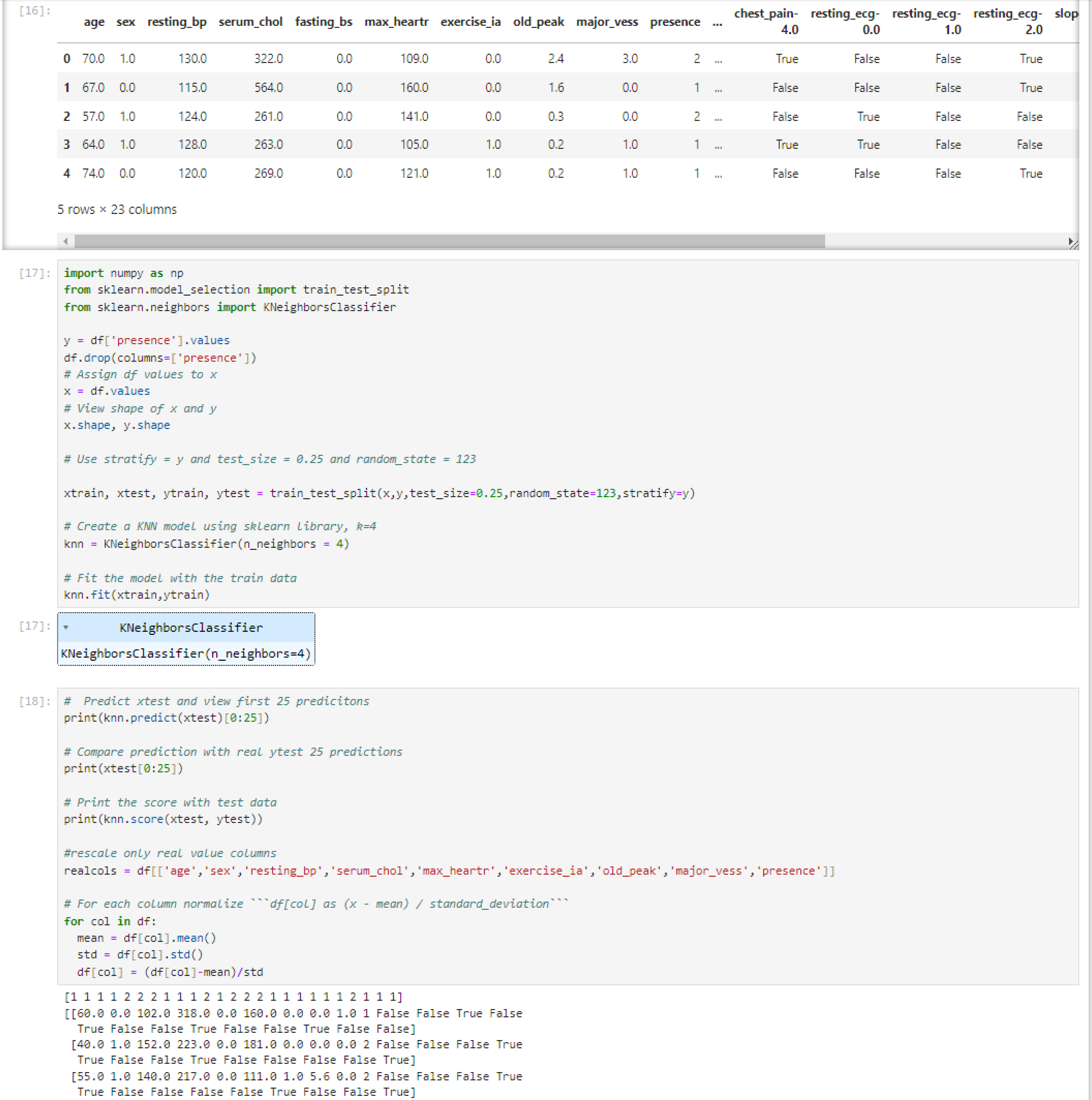
```
(270, 14)
```

[15]:
```
import seaborn as sns

sns.set(style="dark", color_codes=True)
g = sns.pairplot(df)

import matplotlib.pyplot as plt
plt.hist(df)
```



With age, it seems that the heart beat gets slower and is not as efficient as in the middle ages.

[16]:
```
import pandas as pd
dummy_list = ['chest_pain','resting_ecg','slope_peak','thal']
df = pd.get_dummies(df,columns=dummy_list, prefix= dummy_list, prefix_sep='-')
df.head()
```



[16]:

	age	sex	resting_bp	serum_chol	fasting_bs	max_heartr	exercise_ia	old_peak	major_vess	presence	...	chest_pain-4.0	resting_ecg-0.0	resting_ecg-1.0	resting_ecg-2.0	slop
0	70.0	1.0	130.0	322.0	0.0	109.0	0.0	2.4	3.0	2	...	True	False	False	True	
1	67.0	0.0	115.0	564.0	0.0	160.0	0.0	1.6	0.0	1	...	False	False	False	True	
2	57.0	1.0	124.0	261.0	0.0	141.0	0.0	0.3	0.0	2	...	False	True	False	False	
3	64.0	1.0	128.0	263.0	0.0	105.0	1.0	0.2	1.0	1	...	True	True	False	False	
4	74.0	0.0	120.0	269.0	0.0	121.0	1.0	0.2	1.0	1	...	False	False	False	True	

5 rows × 23 columns

[17]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

y = df['presence'].values
df.drop(columns=['presence'])
# Assign df values to x
x = df.values
# View shape of x and y
x.shape, y.shape

# Use stratify = y and test_size = 0.25 and random_state = 123

xtrain, xtest, ytrain, ytest = train_test_split(x,y,test_size=0.25,random_state=123,stratify=y)

# Create a KNN model using sklearn library, k=4
knn = KNeighborsClassifier(n_neighbors = 4)

# Fit the model with the train data
knn.fit(xtrain,ytrain)
```

[17]:

```
▾      KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

[18]:

```
#  Predict xtest and view first 25 predicitons
print(knn.predict(xtest)[0:25])

# Compare prediction with real ytest 25 predictions
print(xtest[0:25])

# Print the score with test data
print(knn.score(xtest, ytest))

#rescale only real value columns
realcols = df[['age','sex','resting_bp','serum_chol','max_heartr','exercise_ia','old_peak','major_vess','presence']]

# For each column normalize ```df[col] as (x - mean) / standard_deviation```
for col in df:
  mean = df[col].mean()
  std = df[col].std()
  df[col] = (df[col]-mean)/std
```

```
[1 1 1 1 2 2 2 1 1 1 2 1 2 2 2 1 1 1 1 1 1 2 1 1 1]
[[60.0 0.0 102.0 318.0 0.0 160.0 0.0 0.0 1.0 1 False False True False
  True False False True False False True False False]
 [40.0 1.0 152.0 223.0 0.0 181.0 0.0 0.0 0.0 2 False False False True
  True False False True False False False False True]
 [55.0 1.0 140.0 217.0 0.0 111.0 1.0 5.6 0.0 2 False False False True
  True False False False False True False False True]
```

```
 [40.0 1.0 152.0 223.0 0.0 181.0 0.0 0.0 0.0 2 False False False True
  True False False True False False False False True]
 [55.0 1.0 140.0 217.0 0.0 111.0 1.0 5.6 0.0 2 False False False True
  True False False False False True False False True]
 [59.0 1.0 170.0 288.0 0.0 159.0 0.0 0.2 0.0 2 True False False False
  False False True False True False False False True]
 [56.0 1.0 130.0 256.0 1.0 142.0 1.0 0.6 1.0 2 False False True False
  False False True False True False False True False]
 [65.0 0.0 160.0 360.0 0.0 151.0 0.0 0.8 0.0 1 False False True False
  False False True True False False True False False]
 [41.0 1.0 135.0 203.0 0.0 132.0 0.0 0.0 0.0 1 False True False False
  True False False False True False False True False]
 [57.0 0.0 128.0 303.0 0.0 159.0 0.0 0.0 1.0 1 False False False True
  False False True True False False True False False]
 [34.0 1.0 118.0 182.0 0.0 174.0 0.0 0.0 0.0 1 True False False False
  False False True True False False True False False]
 [35.0 1.0 126.0 282.0 0.0 156.0 1.0 0.0 0.0 2 False False False True
  False False True True False False False False True]
 [59.0 1.0 110.0 239.0 0.0 142.0 1.0 1.2 1.0 2 False False False True
  False False True False True False False False True]
 [54.0 0.0 110.0 214.0 0.0 158.0 0.0 1.6 0.0 1 False False True False
  True False False False True False True False False]
 [69.0 1.0 140.0 254.0 0.0 146.0 0.0 2.0 3.0 2 False False True False
  False False True False True False False False True]
 [68.0 1.0 118.0 277.0 0.0 151.0 0.0 1.0 1.0 1 False False True False
  True False False True False False False False True]
 [57.0 0.0 120.0 354.0 0.0 163.0 1.0 0.6 0.0 1 False False False True
  True False False True False False True False False]
 [58.0 0.0 120.0 340.0 0.0 172.0 0.0 0.0 0.0 1 False False True False
  True False False True False False True False False]
 [42.0 1.0 120.0 240.0 1.0 194.0 0.0 0.8 0.0 1 False False True False
  True False False False False True False False True]
 [56.0 0.0 140.0 294.0 0.0 153.0 0.0 1.3 0.0 1 False True False False
  False False True False True False True False False]
 [41.0 0.0 126.0 306.0 0.0 163.0 0.0 0.0 0.0 1 False True False False
  True False False True False False True False False]
 [42.0 1.0 148.0 244.0 0.0 178.0 0.0 0.8 2.0 1 True False False False
  False False True True False False True False False]
 [54.0 1.0 192.0 283.0 0.0 195.0 0.0 0.0 1.0 2 False True False False
  False False True True False False False False True]
 [57.0 1.0 165.0 289.0 1.0 124.0 0.0 1.0 3.0 2 False False False True
  False False True False True False False False True]
 [60.0 0.0 158.0 305.0 0.0 161.0 0.0 0.0 0.0 2 False False False True
  False False True True False False True False False]
 [48.0 1.0 124.0 255.0 1.0 175.0 0.0 0.0 2.0 1 False False True False
  True False False True False False True False False]
 [44.0 1.0 120.0 263.0 0.0 173.0 0.0 0.0 0.0 1 False True False False
  True False False True False False False False True]]
0.6617647058823529
```

[19]:

```

x = df.values

# Train test Split
xtrain, xtest, ytrain, ytest = train_test_split(x,y,test_size=0.25,random_state=123,stratify=y)

# Model Initialization
knn = KNeighborsClassifier(n_neighbors = 4)

# Model fitting with training data
knn.fit(xtrain,ytrain)

# Now print score on test data
knn.score(xtest, ytest)
```

[19]:

```
0.8823529411764706
```

[20]:

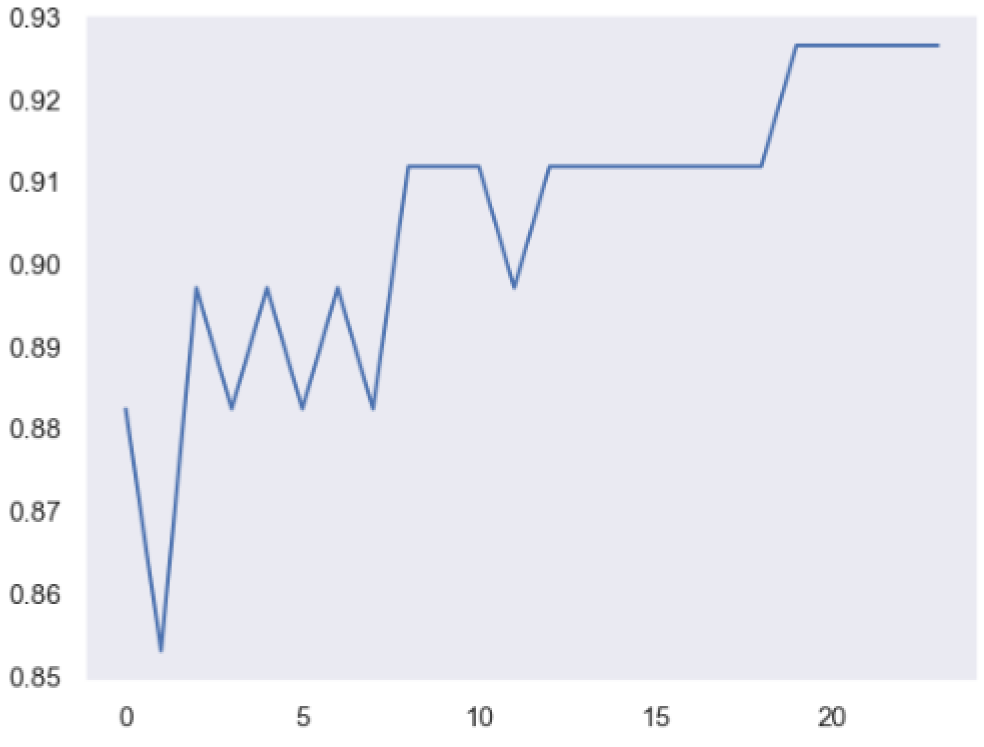
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
```

```
# Train test Split
xtrain, xtest, ytrain, ytest = train_test_split(x,y,test_size=0.25,random_state=123,stratify=y)

# Model Initialization
knn = KNeighborsClassifier(n_neighbors = 4)

# Model fitting with training data
knn.fit(xtrain,ytrain)

# Now print score on test data
knn.score(xtest, ytest)
```

```
[19]: 0.8823529411764706
```

```
[20]: import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

def returnScore(k, xtrain, xtest, ytrain, ytest):
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(xtrain, ytrain)
  return knn.score(xtest, ytest)


result = [*map(lambda i:returnScore(i,xtrain, xtest, ytrain, ytest), range(1,25))]
print(result)
plt.plot(result)


print('BESt VALUE OF K',np.argmax(result) + 1 )
```

```
[0.8823529411764706, 0.8529411764705882, 0.8970588235294118, 0.8823529411764706, 0.8970588235294118, 0.8823529411764706, 0.8970588235294118, 0.882352941
1764706, 0.9117647058823529, 0.9117647058823529, 0.9117647058823529, 0.8970588235294118, 0.9117647058823529, 0.9117647058823529, 0.9117647058823529, 0.9
117647058823529, 0.9117647058823529, 0.9117647058823529, 0.9117647058823529, 0.9264705882352942, 0.9264705882352942, 0.9264705882352942, 0.9264705882352
942, 0.9264705882352942]
BESt VALUE OF K 20
```



```
[22]: bestknn = KNeighborsClassifier(n_neighbors=np.argmax(result) + 1)
bestknn.fit(xtrain,ytrain)
bestknn.score(xtest,ytest)
```

[22]:

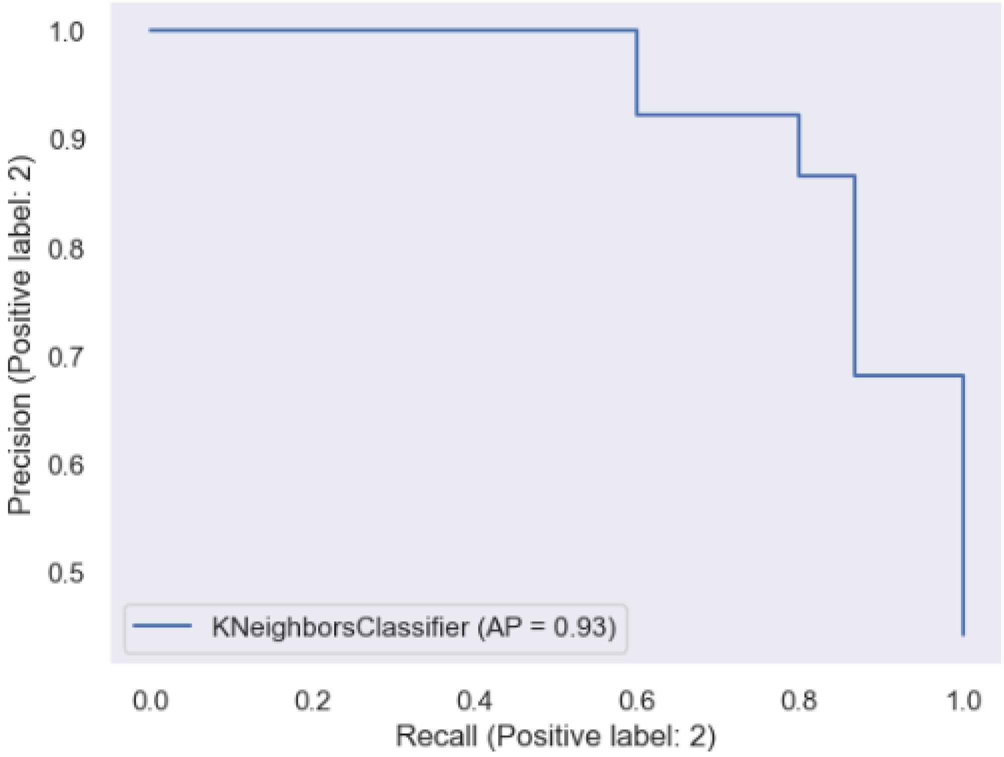
```
bestknn = KNeighborsClassifier(n_neighbors=np.argmax(result) + 1)
bestknn.fit(xtrain,ytrain)
bestknn.score(xtest,ytest)

ypred = bestknn.predict(xtest)
matrix = confusion_matrix(ytest,ypred)
print(matrix)
```

```
[[37  1]
 [ 4 26]]
```

[26]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import PrecisionRecallDisplay
import matplotlib.pyplot as plt

mse = mean_squared_error(ytest,ypred)          # Calculate the test MSE
print("Test mean squared error (MSE): {:.2f}".format(mse))

print(bestknn.score(xtest,ytest))


PrecisionRecallDisplay.from_estimator(knn,xtest,ytest)
plt.show()
```

```
Test mean squared error (MSE): 0.07
0.9264705882352942
```



[ ]: