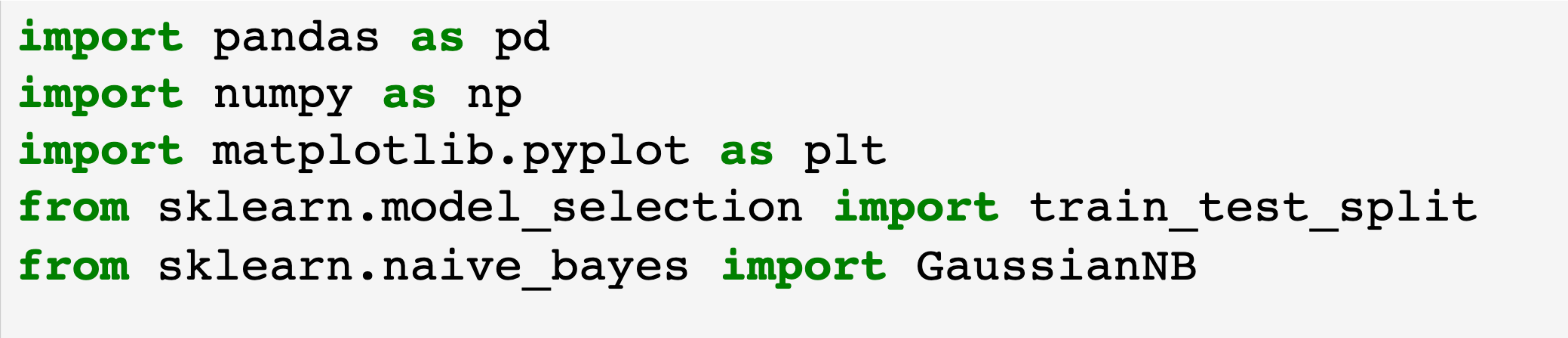
Shahid Beheshti University

Implementation of

The possibility of heart disease with gaussian naive bayes algorithm

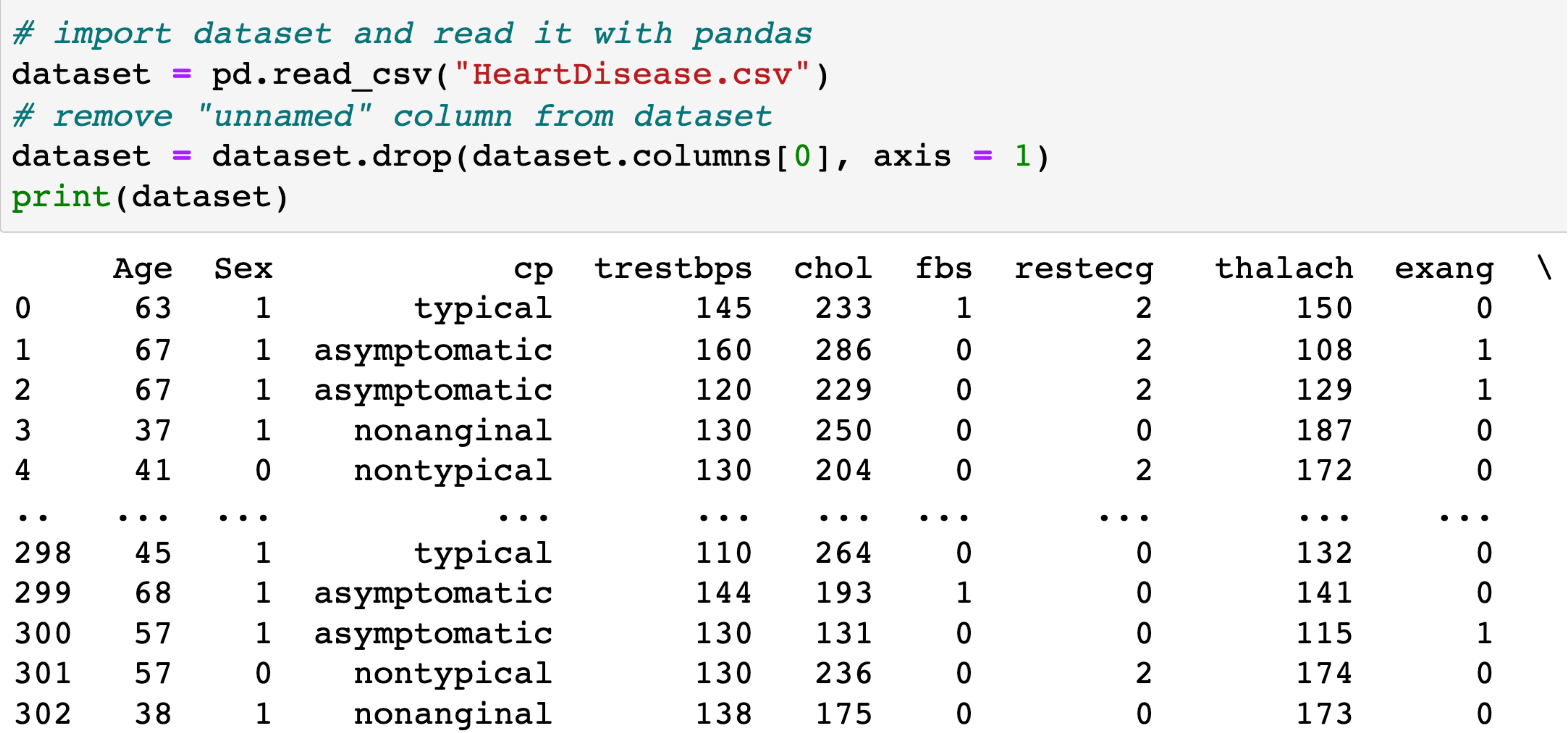
Kian Anvari Hamedani
401443016



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

Import pandas for reading the cdv file and pre-process the dataset
matplotlib.pyplot for draw graphs
Sklearn.model_selection for split test and train datas
Sklearn.naive_bayes for learning and predicting

Pre-processing dataset

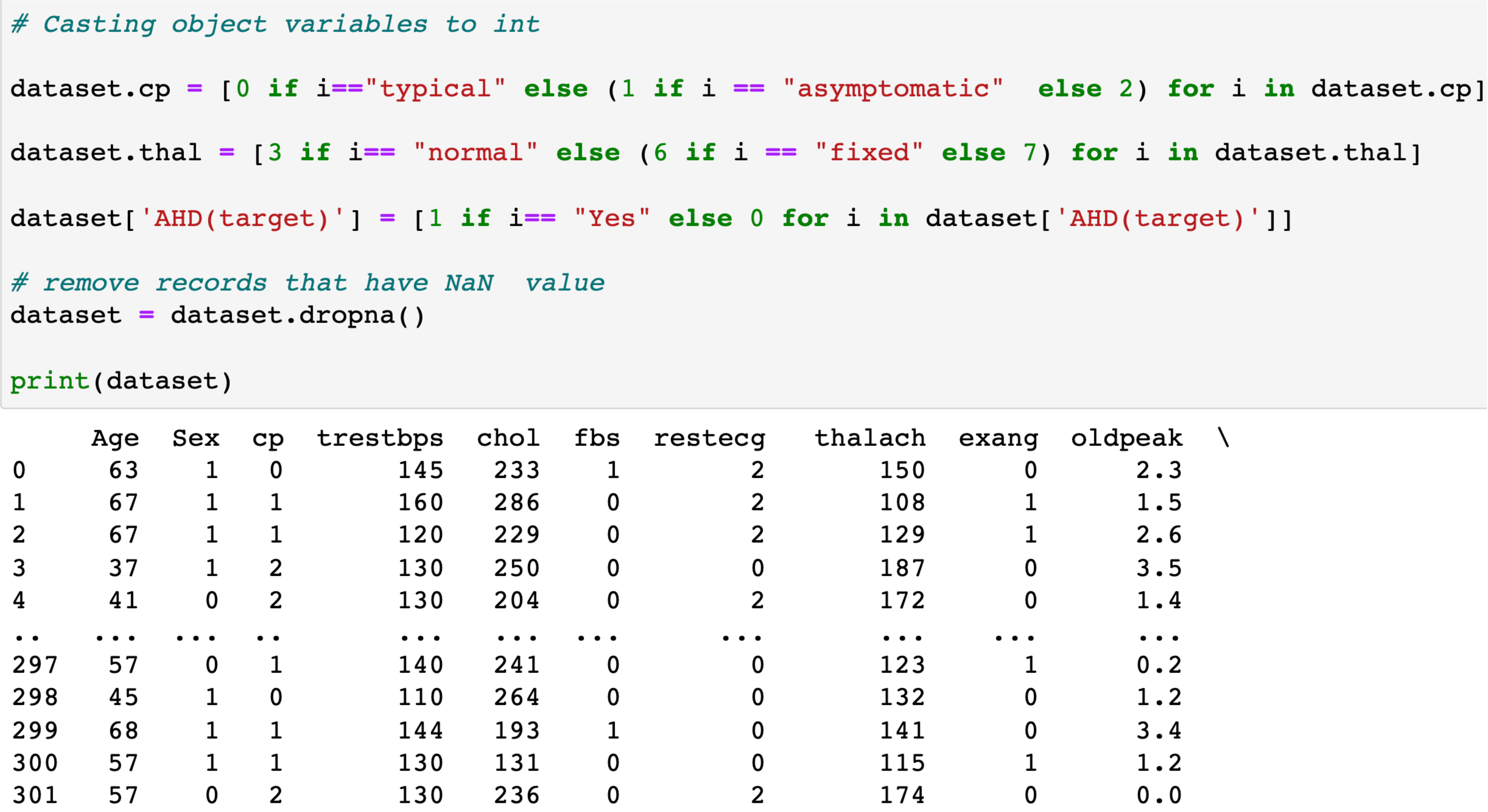


```python
# import dataset and read it with pandas
dataset = pd.read_csv("HeartDisease.csv")
# remove "unnamed" column from dataset
dataset = dataset.drop(dataset.columns[0], axis = 1)
print(dataset)
```

```
     Age  Sex           cp  trestbps  chol  fbs  restecg   thalach  exang  \
0     63    1      typical       145   233    1        2       150      0   
1     67    1  asymptomatic       160   286    0        2       108      1   
2     67    1  asymptomatic       120   229    0        2       129      1   
3     37    1    nonanginal       130   250    0        0       187      0   
4     41    0    nontypical       130   204    0        2       172      0   
..   ...  ...          ...       ...   ...  ...      ...       ...    ...   
298   45    1      typical       110   264    0        0       132      0   
299   68    1  asymptomatic       144   193    1        0       141      0   
300   57    1  asymptomatic       130   131    0        0       115      1   
301   57    0    nontypical       130   236    0        2       174      0   
302   38    1    nonanginal       138   175    0        0       173      0   
```

Read the dataset and remove unnamed column

Pre-processing dataset

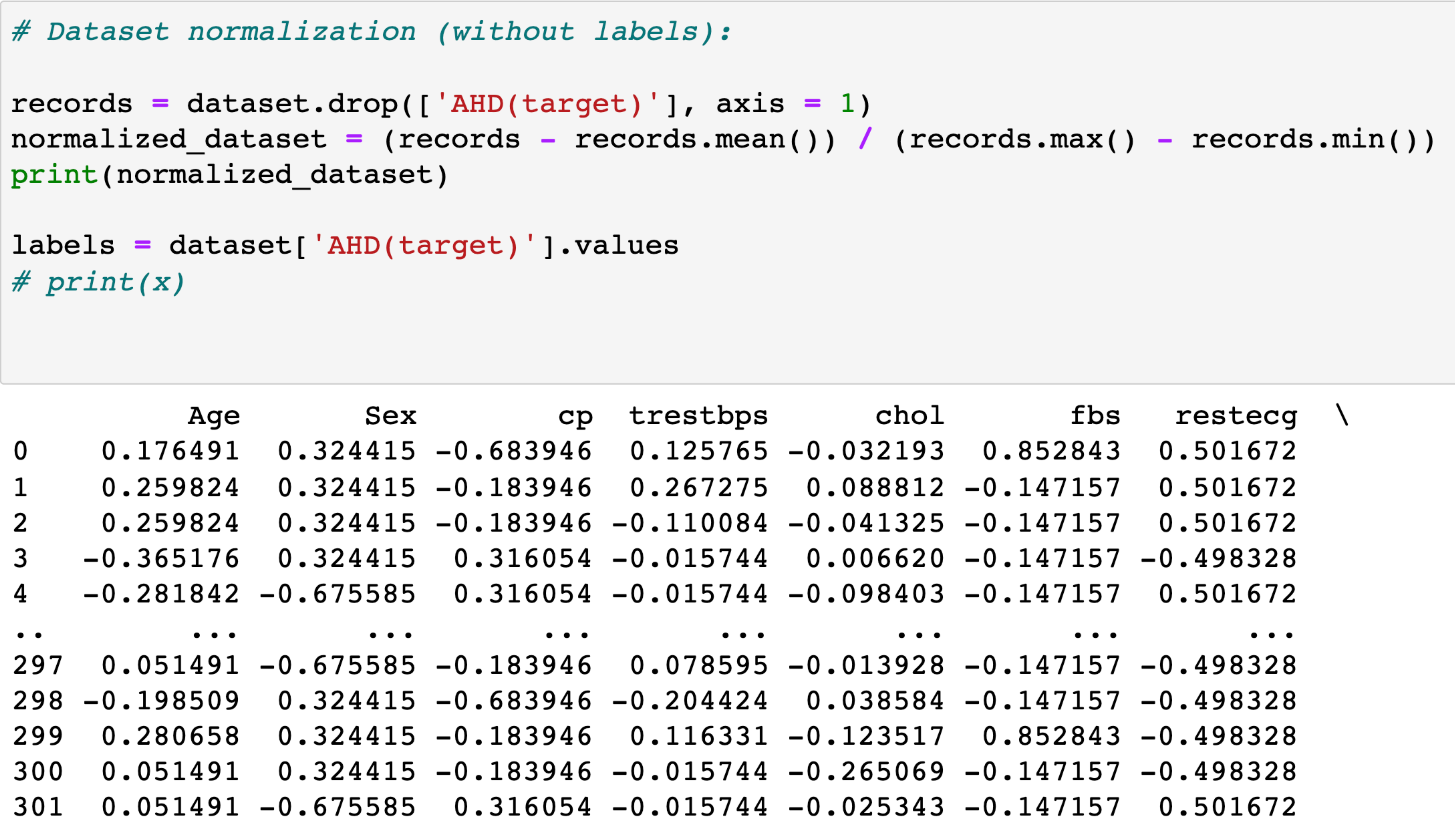


```
# Casting object variables to int

dataset.cp = [0 if i=="typical" else (1 if i == "asymptomatic"  else 2) for i in dataset.cp]

dataset.thal = [3 if i== "normal" else (6 if i == "fixed" else 7) for i in dataset.thal]

dataset['AHD(target)'] = [1 if i== "Yes" else 0 for i in dataset['AHD(target)']]

# remove records that have NaN  value
dataset = dataset.dropna()

print(dataset)
```

```
     Age  Sex  cp  trestbps  chol  fbs  restecg   thalach  exang  oldpeak  \
0     63    1   0       145   233    1        2       150      0      2.3
1     67    1   1       160   286    0        2       108      1      1.5
2     67    1   1       120   229    0        2       129      1      2.6
3     37    1   2       130   250    0        0       187      0      3.5
4     41    0   2       130   204    0        2       172      0      1.4
..   ...  ...  ..       ...   ...  ...      ...       ...    ...      ...
297   57    0   1       140   241    0        0       123      1      0.2
298   45    1   0       110   264    0        0       132      0      1.2
299   68    1   1       144   193    1        0       141      0      3.4
300   57    1   1       130   131    0        0       115      1      1.2
301   57    0   2       130   236    0        2       174      0      0.0
```

Casting objects vars to int vars with for loop

Pre-processing dataset

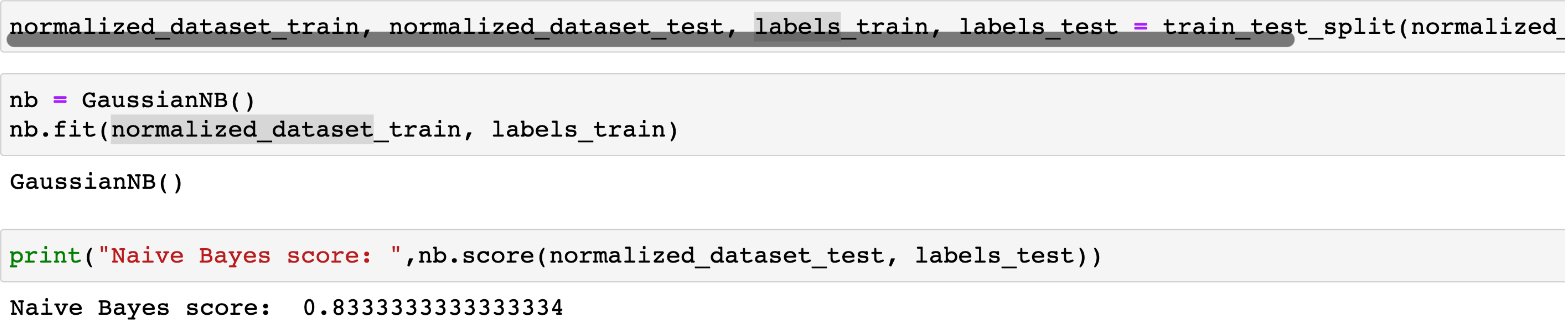


```
# Dataset normalization (without labels):

records = dataset.drop(['AHD(target)'], axis = 1)
normalized_dataset = (records - records.mean()) / (records.max() - records.min())
print(normalized_dataset)

labels = dataset['AHD(target)'].values
# print(x)
```

```
          Age       Sex        cp  trestbps      chol       fbs   restecg  \
0    0.176491  0.324415 -0.683946  0.125765 -0.032193  0.852843  0.501672
1    0.259824  0.324415 -0.183946  0.267275  0.088812 -0.147157  0.501672
2    0.259824  0.324415 -0.183946 -0.110084 -0.041325 -0.147157  0.501672
3   -0.365176  0.324415  0.316054 -0.015744  0.006620 -0.147157 -0.498328
4   -0.281842 -0.675585  0.316054 -0.015744 -0.098403 -0.147157  0.501672
..        ...       ...       ...       ...       ...       ...       ...
297  0.051491 -0.675585 -0.183946  0.078595 -0.013928 -0.147157 -0.498328
298 -0.198509  0.324415 -0.683946 -0.204424  0.038584 -0.147157 -0.498328
299  0.280658  0.324415 -0.183946  0.116331 -0.123517  0.852843 -0.498328
300  0.051491  0.324415 -0.183946 -0.015744 -0.265069 -0.147157 -0.498328
301  0.051491 -0.675585  0.316054 -0.015744 -0.025343 -0.147157  0.501672
```

Normalize the dataset for learning

Learn and Test



```
normalized_dataset_train, normalized_dataset_test, labels_train, labels_test = train_test_split(normalized_
```

```
nb = GaussianNB()
nb.fit(normalized_dataset_train, labels_train)
```

```
GaussianNB()
```

```
print("Naive Bayes score: ",nb.score(normalized_dataset_test, labels_test))
```

```
Naive Bayes score:  0.8333333333333334
```

Use GaussianNB() for learn data with gaussian noise

Use nb.score to get the accuracy of the trained function based on the test data