```
!pip install -q scikit-learn
import numpy as npp
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier


data={
    'BP':[120,130,140,150,160,170,180,190,200,210],
    'Cholestrol':[200,220,240,260,280,300,320,340,360,380],
    'HeartRisk':[0,0,0,1,1,1,1,1,1,1]
}


df=pd.DataFrame(data)



k=1
knn=KNeighborsClassifier(n_neighbors=k)


x=df[['BP','Cholestrol']]
y=df['HeartRisk']


knn.fit(x,y)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

```
new_data=npp.array([210,250])
prediction=knn.predict([new_data])
print(prediction)
```

```
[1]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier
  warnings.warn(
```

```
if prediction==0:
  print("no Risk")
else:
  print("at Risk")
```

```
at Risk
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import mlsit
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix


digits=load_digits()


x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.2)


knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)


print("classification report")
print(classification_report(y_test,y_prediction))
```

```
classification report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        36
           2       1.00      1.00      1.00        35
           3       0.97      1.00      0.99        34
           4       1.00      1.00      1.00        32
           5       1.00      1.00      1.00        32
           6       1.00      1.00      1.00        41
           7       1.00      0.98      0.99        41
           8       1.00      0.97      0.98        31
           9       0.97      1.00      0.99        38

    accuracy                           0.99       360
   macro avg       0.99      0.99      0.99       360
weighted avg       0.99      0.99      0.99       360
```

```
print("confusion matrix")
print(confusion_matrix(y_test,y_prediction))
```

```
confusion matrix
[[41  0  0  0  0  0  0  0  0  0]
 [ 0 44  0  0  0  1  0  0  0  0]
 [ 0  0 29  0  0  0  0  0  0  0]
 [ 0  0  0 39  0  0  0  0  0  0]
 [ 0  0  0  0 33  0  0  0  0  0]
 [ 0  0  0  0  0 34  0  0  0  1]
 [ 0  0  0  0  0  0 32  0  0  0]
 [ 0  0  0  0  0  0  0 36  0  0]
 [ 0  0  0  0  0  0  0  0 30  0]
 [ 0  1  0  2  1  0  0  0  1 35]]
```

Start coding or generate with AI.