Final Project KNN

December 3, 2021

[1]:
```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import random
from typing import List, TypeVar, Tuple

heartData = pd.read_csv('heart.csv', encoding = 'latin1', header = 0)

heartTraining, heartTesting = train_test_split(heartData, test_size = 0.30,␣
 ↪random_state = 1, shuffle = True)
knn = KNeighborsClassifier(n_neighbors = 17)
x = heartTraining.iloc[:, :10].values
y = heartTraining.iloc[:, 12].values
knn.fit(x, y)
expected = heartTesting.iloc[:, 12].values
predicted = knn.predict(heartTesting.iloc[:, :10].values)
print(f"Random State = 1")
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print(f"")

heartTraining, heartTesting = train_test_split(heartData, test_size = 0.30,␣
 ↪random_state = 2, shuffle = True)
knn = KNeighborsClassifier(n_neighbors = 17)
x = heartTraining.iloc[:, :10].values
y = heartTraining.iloc[:, 12].values
knn.fit(x, y)
expected = heartTesting.iloc[:, 12].values
predicted = knn.predict(heartTesting.iloc[:, :10].values)
print(f"Random State = 2")
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print(f"")

heartTraining, heartTesting = train_test_split(heartData, test_size = 0.30,␣
 ↪random_state = 3, shuffle = True)
```

```
knn = KNeighborsClassifier(n_neighbors = 17)
x = heartTraining.iloc[:, :10].values
y = heartTraining.iloc[:, 12].values
knn.fit(x, y)
expected = heartTesting.iloc[:, 12].values
predicted = knn.predict(heartTesting.iloc[:, :10].values)
print(f"Random State = 3")
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print(f"")

heartTraining, heartTesting = train_test_split(heartData, test_size = 0.30,␣
 ↪random_state = 4, shuffle = True)
knn = KNeighborsClassifier(n_neighbors = 17)
x = heartTraining.iloc[:, :10].values
y = heartTraining.iloc[:, 12].values
knn.fit(x, y)
expected = heartTesting.iloc[:, 12].values
predicted = knn.predict(heartTesting.iloc[:, :10].values)
print(f"Random State = 4")
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
Random State = 1
              precision    recall  f1-score   support

           0       0.71      0.92      0.80        64
           1       0.29      0.08      0.12        26

    accuracy                           0.68        90
   macro avg       0.50      0.50      0.46        90
weighted avg       0.59      0.68      0.61        90

[[59  5]
 [24  2]]

Random State = 2
              precision    recall  f1-score   support

           0       0.71      0.86      0.78        66
           1       0.10      0.04      0.06        24

    accuracy                           0.64        90
   macro avg       0.41      0.45      0.42        90
weighted avg       0.55      0.64      0.59        90

[[57  9]
 [23  1]]
```

```
Random State = 3
              precision    recall  f1-score   support

           0       0.67      0.95      0.78        61
           1       0.00      0.00      0.00        29

    accuracy                           0.64        90
   macro avg       0.33      0.48      0.39        90
weighted avg       0.45      0.64      0.53        90

[[58  3]
 [29  0]]

Random State = 4
              precision    recall  f1-score   support

           0       0.72      0.95      0.82        66
           1       0.00      0.00      0.00        24

    accuracy                           0.70        90
   macro avg       0.36      0.48      0.41        90
weighted avg       0.53      0.70      0.60        90

[[63  3]
 [24  0]]
```

[ ]: