مهدی محمدی - شماره دانشجویی: ۴۰۱۱۲۰۹۴

مجموعه آموزشی = ۷۵ نمونه تصادفی، مجموعه تست = ۷۵ نمونه باقیمانده

Model	Training MSE	Test MSE	# of Iterations until Convergence
SVM	0.0533	0.0533	...
LDA	0.0266	0.0133	...
Naive Bayes	0.0533	0.0533	...
Neural Network	0.0133	0.0400	...
Decision Trees	0.0	0.0933	...

طبقه بند: SVM – داده‌ها پس از نرمال‌سازی مورد استفاده قرارگرفته‌اند.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 27 21:55:57 2023

@author: Mehdi Mohammadi
"""

import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

IRIS = datasets.load_iris()

X = IRIS.data
Y_Data = IRIS.target

pca = PCA(n_components = 3)
X_pca = pca.fit_transform(X)

kpca = KernelPCA(n_components = 3)
X_kpca = kpca.fit_transform(X)

sc = StandardScaler()
X = sc.fit_transform(X)
X_pca = sc.fit_transform(X_pca)
X_kpca = sc.fit_transform(X_kpca)

X_Data = np.column_stack((X, X_pca, X_kpca))
```

```
Xtr, Xte, Ytr, Yte = train_test_split(X_Data, Y_Data, train_size = 0.5, random_state
= 1)

SVM = SVC(gamma='scale')
SVM.fit(Xtr, Ytr);
Ypr = SVM.predict(Xte)

train_error_rate = 1 - SVM.score(Xtr, Ytr)
test_error_rate = 1 - SVM.score(Xte, Yte)

print('train Error Rate: ', train_error_rate)
print('test Error Rate: ', test_error_rate)
```

طبقه بند: LDA – داده‌ها پس از نرمال‌سازی مورد استفاده قرارگرفته‌اند.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 27 21:55:57 2023

@author: Mehdi Mohammadi
"""

import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

IRIS = datasets.load_iris()

X = IRIS.data
Y_Data = IRIS.target

pca = PCA(n_components = 3)
X_pca = pca.fit_transform(X)

kpca = KernelPCA(n_components = 3)
X_kpca = kpca.fit_transform(X)

sc = StandardScaler()
X = sc.fit_transform(X)
X_pca = sc.fit_transform(X_pca)
X_kpca = sc.fit_transform(X_kpca)

X_Data = np.column_stack((X, X_pca, X_kpca))

Xtr, Xte, Ytr, Yte = train_test_split(X_Data, Y_Data, train_size = 0.5, random_state
= 1)

LDA = LinearDiscriminantAnalysis(solver='lsqr')
LDA.fit(Xtr, Ytr);
Ypr = LDA.predict(Xte)

train_error_rate = 1 - LDA.score(Xtr, Ytr)
```

```
test_error_rate = 1 - LDA.score(Xte, Yte)

print('train Error Rate: ', train_error_rate)
print('test Error Rate: ', test_error_rate)
```

طبقه بند: **Naive Bayes** – داده‌ها پس از نرمال‌سازی مورد استفاده قرارگرفته‌اند.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 27 21:55:57 2023

@author: Mehdi Mohammadi
"""

import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

IRIS = datasets.load_iris()

X = IRIS.data
Y_Data = IRIS.target

pca = PCA(n_components = 3)
X_pca = pca.fit_transform(X)

kpca = KernelPCA(n_components = 3)
X_kpca = kpca.fit_transform(X)

sc = StandardScaler()
X = sc.fit_transform(X)
X_pca = sc.fit_transform(X_pca)
X_kpca = sc.fit_transform(X_kpca)

X_Data = np.column_stack((X, X_pca, X_kpca))

Xtr, Xte, Ytr, Yte = train_test_split(X_Data, Y_Data, train_size = 0.5, random_state
= 1)

GNB = GaussianNB()
GNB.fit(Xtr, Ytr);
Ypr = GNB.predict(Xte)

train_error_rate = 1 - GNB.score(Xtr, Ytr)
test_error_rate = 1 - GNB.score(Xte, Yte)

print('train Error Rate: ', train_error_rate)
print('test Error Rate: ', test_error_rate)
```

طبقه بند: **Neural Network** – داده‌ها پس از نرمال‌سازی مورد استفاده قرارگرفته‌اند.

```
# -*- coding: utf-8 -*-
"""
```

```
Created on Thu Jan 27 21:55:57 2023

@author: Mehdi Mohammadi
"""

import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

IRIS = datasets.load_iris()

X = IRIS.data
Y_Data = IRIS.target

pca = PCA(n_components = 3)
X_pca = pca.fit_transform(X)

kpca = KernelPCA(n_components = 3)
X_kpca = kpca.fit_transform(X)

sc = StandardScaler()
X = sc.fit_transform(X)
X_pca = sc.fit_transform(X_pca)
X_kpca = sc.fit_transform(X_kpca)

X_Data = np.column_stack((X, X_pca, X_kpca))

Xtr, Xte, Ytr, Yte = train_test_split(X_Data, Y_Data, train_size = 0.5, random_state
= 1)

NN = MLPClassifier(solver = 'lbfgs', alpha = 1e-3, hidden_layer_sizes = (7, 3))
NN.fit(Xtr, Ytr);
Ypr = NN.predict(Xte)

train_error_rate = 1 - NN.score(Xtr, Ytr)
test_error_rate = 1 - NN.score(Xte, Yte)

print('train Error Rate: ', train_error_rate)
print('test Error Rate: ', test_error_rate)
```

طبقه بند: **Decision Tree** – داده‌ها پس از نرمال‌سازی مورد استفاده قرارگرفته‌اند.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 27 21:55:57 2023

@author: Mehdi Mohammadi
"""

import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

```
from sklearn.decomposition import KernelPCA
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

IRIS = datasets.load_iris()

X = IRIS.data
Y_Data = IRIS.target

pca = PCA(n_components = 3)
X_pca = pca.fit_transform(X)

kpca = KernelPCA(n_components = 3)
X_kpca = kpca.fit_transform(X)

sc = StandardScaler()
X = sc.fit_transform(X)
X_pca = sc.fit_transform(X_pca)
X_kpca = sc.fit_transform(X_kpca)

X_Data = np.column_stack((X, X_pca, X_kpca))

Xtr, Xte, Ytr, Yte = train_test_split(X_Data, Y_Data, train_size = 0.5, random_state
= 1)

DT = DecisionTreeClassifier()
DT.fit(Xtr, Ytr);
Ypr = DT.predict(Xte)

train_error_rate = 1 - DT.score(Xtr, Ytr)
test_error_rate = 1 - DT.score(Xte, Yte)

print('train Error Rate: ', train_error_rate)
print('test Error Rate: ', test_error_rate)
```