lab9_pca

November 15, 2022

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from matplotlib.colors import ListedColormap
```

[ ]:
```
data=pd.read_csv("C:\Coding\ML_python\machine-learning-lab-main\datasets\iris.
↪csv")
data.head()
```

[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[ ]:
```
x=data.iloc[:,0:4]
y=data.iloc[:,4]
```

[ ]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2,␣
↪random_state = 0)
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)
```

[ ]:
```
pca = PCA(n_components = 2)

xtrain = pca.fit_transform(xtrain)
xtest = pca.transform(xtest)

explained_variance = pca.explained_variance_ratio_
```

[ ]:
```
classifier = LogisticRegression(random_state = 0)
le = preprocessing.LabelEncoder()
ytrain=le.fit_transform(ytrain)
ytest=le.fit_transform(ytest)

classifier.fit(xtrain, ytrain)
```

[ ]:
```
LogisticRegression(random_state=0)
```

[ ]:
```
ypred = classifier.predict(xtest)
```

[ ]:
```
cm = confusion_matrix(ytest, ypred)
print(cm)
```

```
[[11  0  0]
 [ 0 10  3]
 [ 0  1  5]]
```

[ ]:
```
X_set, y_set = xtrain, ytrain
X1,X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,stop = X_set[:, 0].
 ↪max() + 1, step = 0.01),np.arange(start = X_set[:, 1].min() - 1, stop =␣
 ↪X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).
 ↪reshape(X1.shape), alpha = 0.75,cmap = ListedColormap(('yellow', 'white',␣
 ↪'aquamarine')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],c =␣
 ↪ListedColormap(('red', 'green', 'blue'))(i), label = j)

plt.title('Logistic Regression IRIS')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
```

c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.