Practical No: 7

Feature Selection

AIM: Data loading, feature scoring and ranking, feature selection (principal component analysis)

Description:

Principal Component Analysis:

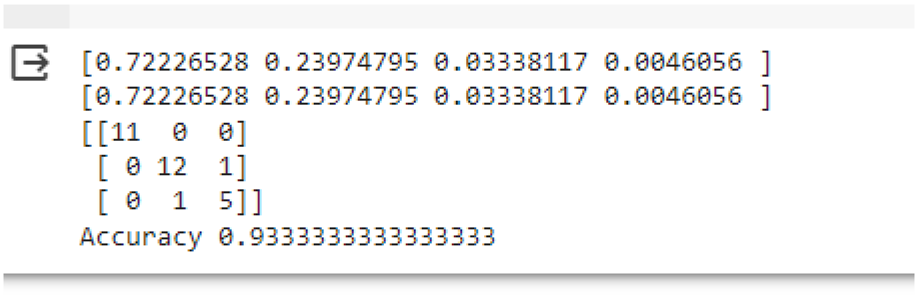
PCA is a way to make data less complex and easier to work with, while still keeping most of the important information. This can be helpful for machine learning tasks, because it can make the data easier for the model to learn from and can lead to better results.

Code with output

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
dataset = pd.read_csv(url, names=names)
dataset.head()

# Store the feature sets into X variable and the series of corresponding variables in y
x = dataset.drop('Class', axis=1)
y = dataset['Class']
x.head()
y.head()

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

sc = StandardScaler()
x_train1 = sc.fit_transform(x_train)
x_test1 = sc.transform(x_test)
```

```
y_train1 = y_train
y_test1 = y_test

pca = PCA()
x_train1 = pca.fit_transform(x_train1)
x_test1 = pca.transform(x_test1)

explained_variance = pca.explained_variance_ratio_
print(explained_variance)

pca = PCA(n_components=1)
x_train1 = pca.fit_transform(x_train1)
x_test1 = pca.transform(x_test1)

classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(x_train1, y_train1)
y_pred = classifier.predict(x_test1)

cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy:', accuracy_score(y_test, y_pred))
```

OUTPUT



```
[0.72226528 0.23974795 0.03338117 0.0046056 ]
[0.72226528 0.23974795 0.03338117 0.0046056 ]
[[11  0  0]
 [ 0 12  1]
 [ 0  1  5]]
Accuracy 0.9333333333333333
```

Learnings

The code first loads the Iris dataset, which is a collection of data about Iris flowers. The data includes four features: sepal length, sepal width, petal length, and petal width. The data also includes a label, which is the species of the Iris flower.

The code then splits the data into two sets: a training set and a test set. The training set is used to train the machine learning model, and the test set is used to evaluate the model's performance.

Before training the model, the code standardizes the features. This means that the code scales the features so that they are all on the same scale. This is important because it ensures that the model does not learn to bias towards any particular feature.

The code then uses PCA to reduce the dimensionality of the features. This means that the code combines the four features into a single feature. This can improve the model's performance because it reduces noise in the data and makes it easier for the model to learn the patterns.

The code then trains a Random Forest Classifier on the training set. A Random Forest Classifier is a type of machine learning model that is well-suited for classification tasks.

Once the model is trained, the code uses it to make predictions on the test set. The code then evaluates the model's performance using a confusion matrix and accuracy score. A confusion matrix shows how many flowers of each species were correctly and incorrectly classified by the model. An accuracy score is a measure of how often the model correctly classified a flower.