Practical No. 4

Title: Implementation of Feature Selection and Dimensionality Reduction Techniques

Aim: To understand and implement various techniques of feature extraction, feature selection, normalization, transformation, and principal component analysis (PCA) with visualizations for high-dimensional datasets.

Introduction:

In machine learning and data science, handling high-dimensional data is a critical step. Raw data often contains redundant, irrelevant, or noisy features that can degrade model performance. To build efficient and accurate models, it is important to reduce dimensionality by extracting relevant features, selecting important ones, and transforming them into usable formats. This manual covers essential steps including feature extraction, selection, normalization, transformation, and PCA with visual insights.

1. Feature Extraction

Definition:
Feature extraction involves transforming raw data into numerical features that can be processed while preserving the information in the original data.

Examples:

- Text: TF-IDF, word embeddings
- Image: Edge detection, color histograms
- Time Series: Statistical metrics like mean, std, etc.

2. Feature Selection

Definition:
Feature selection involves selecting the most relevant features from the dataset for model training to improve performance and reduce overfitting.

Techniques:

- Filter Methods (e.g., correlation, chi-square)
- Wrapper Methods (e.g., Recursive Feature Elimination)
- Embedded Methods (e.g., Lasso Regression)

Python Example:

```
from sklearn.feature_selection import SelectKBest, chi2

from sklearn.datasets import load_iris

iris = load_iris()

X, y = iris.data, iris.target

X_new = SelectKBest(score_func=chi2, k=2).fit_transform(X, y)

print(X_new[:5])
```

3. Normalization

Definition:
Normalization scales numerical data to a standard range (usually 0 to 1) to ensure uniformity and prevent bias towards features with large scales.

Python Example:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_scaled = scaler.fit_transform(X)
```

```
print(X_scaled[:5])
```

4. Transformation

Definition:
Transformation changes the distribution of the data, often to approximate normality or stabilize variance.

Common Transformations:

- Log Transformation
- Box-Cox Transformation
- Power Transformation

Python Example:

```
import numpy as np

from sklearn.preprocessing import PowerTransformer


X_transformed = PowerTransformer().fit_transform(X)

print(X_transformed[:5])
```

5. Principal Component Analysis (PCA)

Definition:
PCA is a dimensionality reduction technique that transforms the data to a new coordinate system, selecting the directions (principal components) that maximize variance.

Python Example with Visualization:

```
from sklearn.decomposition import PCA
```

```
import matplotlib.pyplot as plt


pca = PCA(n_components=2)

X_pca = pca.fit_transform(X)


plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA of Iris Dataset')

plt.colorbar()

plt.show()
```

Exercise -

1. Implementation of Features Extraction and Selection, Normalization, Transformation, Principal Components Analysis.

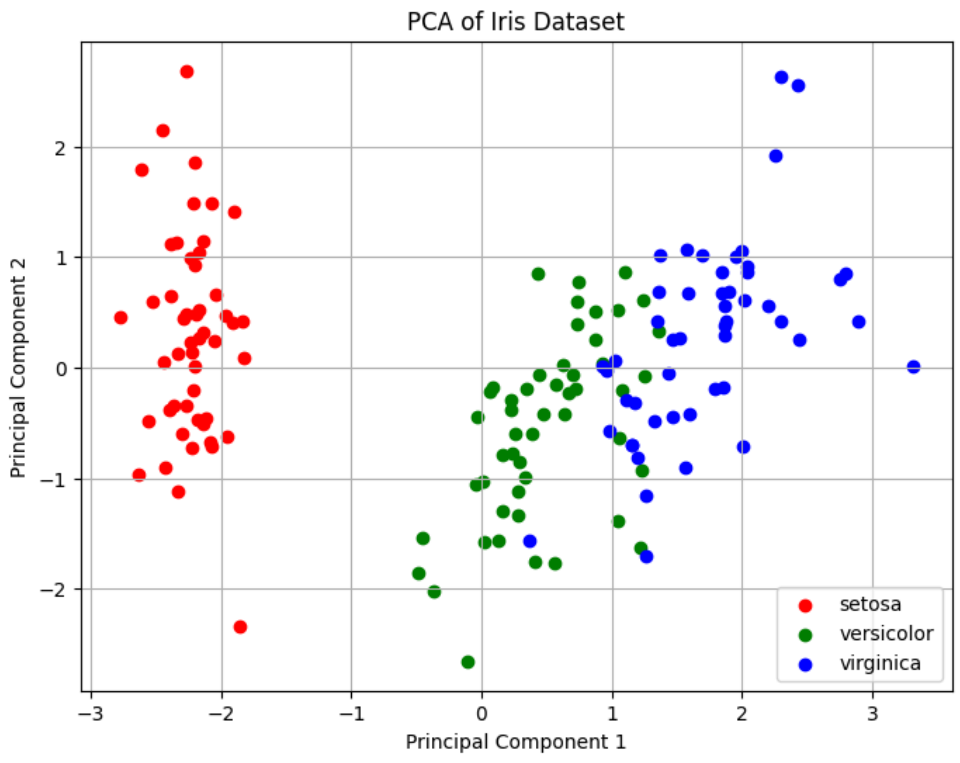
Implementation:

Program:

```
#Feature Selection: Features Extraction, Feature Selection, Normalization, Transformation,

#Principal Components Analysis-visualizations of complex datasets.


import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Step 1: Load the dataset
iris = load_iris()
X = iris.data
y = iris.target

# Step 2: Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Perform PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Step 4: Visualize the results
plt.figure(figsize=(8, 6))
targets = np.unique(y)
colors = ['r', 'g', 'b']
```

```
for target, color in zip(targets, colors):
  plt.scatter(X_pca[y == target, 0],X_pca[y == target, 1],color=color, label=target)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset')
plt.legend(labels=iris.target_names)
plt.grid()
plt.show()
```

Output:


PCA of Iris Dataset
Principal Component 1
Principal Component 2
setosa
versicolor
virginica