

Program 11

Implement Dimensionality reduction using Principal Component Analysis (PCA) method

Screenshot:

→ Dimensionality Reduction using principle component analysis
Reduce dim from 2 to 1 using PCA

Feature	Ex-1	Ex-2	Ex-3	Ex-4
X ₁	4	8	13	7
X ₂	11	4	5	14

eigen values $\lambda_1 = 30.3849$ $\lambda_2 = 6.6151$
eigen vectors $e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$ $e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$

Mean of X₁ = 8
Mean of X₂ = 8.5

X_{centered} = $\begin{bmatrix} 4-8 & 8-8 & 13-8 & 7-8 \\ 11-8.5 & 4-8.5 & 5-8.5 & 14-8.5 \end{bmatrix} = \begin{bmatrix} -4 & 0 & 5 & -1 \\ 2.5 & -4.5 & -3.5 & 5.5 \end{bmatrix}$

Output eigen value = λ_1
Corresponding vector = $e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$

DATE PAGE

$$Z = e^T \cdot X_{\text{reduced}}$$

$$Z = [0.5574 \quad -0.8303] \begin{bmatrix} -4 & 0 & 5 & -1 \\ 2.5 & -4.5 & 3.5 & 5.5 \end{bmatrix}$$

$$Z_1 = (0.5574)(-4) + (-0.8303)(2.5)$$

$$Z_1 = 1.5385$$

Scanned with

Code:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

from sklearn.decomposition import PCA

# Load the dataset

df = pd.read_csv('/content/heart (2).csv')

# Detect and encode categorical columns

text_cols = df.select_dtypes(include=['object']).columns.tolist()
```

```

for col in text_cols:

    le = LabelEncoder()

    df[col] = le.fit_transform(df[col].astype(str))


# Use correct target column

target_col = 'HeartDisease'

X = df.drop(columns=[target_col])

y = df[target_col]


# Scale the features

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)


# Models

models = {

    'SVM': SVC(),

    'Logistic Regression': LogisticRegression(),

    'Random Forest': RandomForestClassifier()

}


# Evaluate without PCA

```

```

results = {}

for name, model in models.items():

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    results[name] = accuracy_score(y_test, y_pred)

    print(f"{name} Accuracy: {results[name]:.4f}")


# Apply PCA

pca = PCA(n_components=0.95)

X_train_pca = pca.fit_transform(X_train)

X_test_pca = pca.transform(X_test)


# Evaluate after PCA

pca_results = {}

for name, model in models.items():

    model.fit(X_train_pca, y_train)

    y_pred = model.predict(X_test_pca)

    pca_results[name] = accuracy_score(y_test, y_pred)

    print(f"{name} Accuracy after PCA: {pca_results[name]:.4f}")


# Final comparison

print("\nComparison (Original vs PCA):")

for name in models:

    print(f"{name}: Original = {results[name]:.4f}, PCA = {pca_results[name]:.4f}")

```

