```
import pandas as pd
import numpy as np
from numpy import nan
from sklearn.decomposition import PCA
import matplotlib as mpl
from matplotlib import pyplot as plt, Axes3D
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report, accuracy_score, ConfusionMatrixDisplay
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

Load data

```
data = pd.read_csv('adult-dataset.csv', header=None)
```

Cleaning Data

```
# Provide a descriptive header
data.columns = ['age', 'work-class', 'education', 'marital-status', 'occupation', 'race', 'sex', 'income']
```

```
# Define column names
data.columns = ['age', 'work-class', 'education', 'marital-status', 'occupation', 'race', 'sex', 'income']
```

```
data.head()
```

	age	work-class	education	marital-status	occupation	race	sex	income
0	39	State-gov	Bachelors	Never-married	Adm-clerical	White	Male	<50K
1	50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	White	Male	<50K
2	38	Private	11th-grad	Divorced	Handlers-cleaners	White	Male	<50K
3	53	Private	11th	Married-civ-spouse	Handlers-cleaners	Black	Male	<50K
4	28	Private	Bachelors	Married-civ-spouse	Prof-specialty	Black	Female	<50K

```
# Replace missing values with NaN
data = data.replace(' ?', np.nan)
```

```
# Remove rows with missing values
data = data.dropna()
```

```
data.head()
```

	age	work-class	education	marital-status	occupation	race	sex	income
0	39	State-gov	Bachelors	Never-married	Adm-clerical	White	Male	<50K
1	50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	White	Male	<50K
2	38	Private	11th-grad	Divorced	Handlers-cleaners	White	Male	<50K
3	53	Private	11th	Married-civ-spouse	Handlers-cleaners	Black	Male	<50K
4	28	Private	Bachelors	Married-civ-spouse	Prof-specialty	Black	Female	<50K

```
# Convert categorical variables to numerical
data['work-class'] = pd.Categorical(data['work-class'].factorize)[1].codes
data['education'] = pd.Categorical(data['education'].factorize)[1].codes
data['marital-status'] = pd.Categorical(data['marital-status'].factorize)[1].codes
data['occupation'] = pd.Categorical(data['occupation'].factorize)[1].codes
data['race'] = pd.Categorical(data['race'].factorize)[1].codes
data['sex'] = pd.Categorical(data['sex'].factorize)[1].codes
data['income'] = pd.Categorical(data['income'].factorize)[1].codes
```

```
data.head()
```

	age	work-class	education	marital-status	occupation	race	sex	income
0	39	0	0	0	0	0	0	0
1	50	1	0	1	1	0	0	0
2	38	2	1	2	2	0	0	0
3	53	2	2	1	2	1	0	0
4	28	2	0	1	3	1	1	0

```
# Convert categorical variables to dummy variables
data = pd.get_dummies(data, columns=['work-class', 'education', 'marital-status', 'occupation', 'race', 'sex'])
```

```
data.head()
```

	age	income	work-class_0	work-class_1	work-class_2	work-class_3	work-class_4	work-class_5	education_0	education_1	education_2	education_3	education_4	education_5	education_6	education_7	education_8	education_9	education_10	education_11	education_12	education_13	education_14	education_15	marital-status_0	marital-status_1	marital-status_2	marital-status_3	marital-status_4	marital-status_5	marital-status_6	occupation_0	occupation_1	occupation_2	occupation_3	occupation_4	occupation_5	occupation_6	occupation_7	occupation_8	occupation_9	occupation_10	occupation_11	occupation_12	occupation_13	race_0	race_1	race_2	race_3	race_4	sex_0	sex_1
0	39	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
1	50	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
2	38	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
3	53	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
4	28	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1

5 rows × 55 columns

```
# Split features and target variable
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

SVD

```
# Perform SVD
U, s, Vt = np.linalg.svd(X, full_matrices=False)
```

```
cov_eigenvectors = Vt.T
cov_eigenvalues = s ** 2 / np.shape(X)[0]
```

```
# Print eigenvectors and eigenvalues
print('Eigenvectors\n', cov_eigenvectors)
print('\n\nEigenvalues\n', cov_eigenvalues)
```

```
Eigenvectors
 [[ 8.94388254e-01  4.16054546e-01  1.17922154e-02 ...  1.95161124e-04
   1.95161124e-04  1.95161124e-04]
 [ 1.12128344e-02  1.00000000e-01  1.00000000e-01 ...  2.72007675e-01
   2.72007675e-01  2.72007675e-01]
 [ 2.14862424e-02  2.00000000e-01  2.00000000e-01 ...  1.36047544e-02
   1.36047544e-02  1.36047544e-02]
 [ 1.00000000e-01  1.00000000e-01  1.42727884e-01 ...  1.51128004e-01
   1.51128004e-01  1.51128004e-01]
 [ 1.00000000e-01  1.00000000e-01  1.00000000e-01 ...  2.30767354e-01
   2.30767354e-01  2.30767354e-01]
 [ 1.42727884e-01  1.51128004e-01  1.51128004e-01 ...  4.45208427e-02
   4.45208427e-02  4.45208427e-02]]

Eigenvalues
 [ 3.07127514e+01  1.38407317e+04  1.38411294e+04  1.38412204e+04
   1.38412204e+04  1.38412204e+04  1.38412204e+04  1.38412204e+04
   1.38412204e+04  1.38412204e+04  1.77387556e+04  1.85261218e+04
   1.85261218e+04  1.85261218e+04  1.85261218
```