Program 1: 2D Array Manipulation

Procedure:

1. Create a 4x4 matrix.
2. Perform slicing operations.

Code:

python

Copy code

```
import numpy as np

matrix = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])
print(matrix[1:])              # Exclude first row
print(matrix[:, :-1])          # Exclude last column
print(matrix[1:3, 0:2])        # 1st & 2nd columns in 2nd & 3rd rows
print(matrix[:, 1:3])          # 2nd & 3rd columns
print(matrix[0, 1:3])          # 2nd & 3rd elements of 1st row
```

Program 2: Matrix Operations

Procedure:

1. Create two matrices.
2. Perform arithmetic and matrix operations.

Code:

python

Copy code

```
ar1 = np.array([[1, 2], [3, 4]])
ar2 = np.array([[5, 6], [7, 8]])

print(np.add(ar1, ar2))          # Addition
print(np.subtract(ar1, ar2))     # Subtraction
print(np.multiply(ar1, ar2))     # Element-wise Multiplication
print(np.dot(ar1, ar2))          # Matrix Multiplication
print(ar1.transpose())           # Transpose
print(np.trace(ar1))             # Sum of Diagonal
```

Program 3: Matrix Properties

Procedure:

1. Create a square matrix with random values.
2. Calculate determinant, inverse, rank, eigenvalues, eigenvectors, and convert to a 1D array.

Code:

python

Copy code

```
matrix = np.random.randint(10, size=(2, 2))

print(np.linalg.det(matrix))               # Determinant
print(np.linalg.inv(matrix))               # Inverse
print(np.linalg.matrix_rank(matrix))       # Rank
eigen_vals, eigen_vecs = np.linalg.eig(matrix)
print(eigen_vals, eigen_vecs)              # Eigenvalues & Eigenvectors
print(matrix.flatten())                    # 1D array
```

Program 4: Singular Value Decomposition (SVD)

Procedure:

1. Perform SVD and reconstruct the matrix.

Code:

python

Copy code

```
A = np.array([[1, 2], [3, 4], [5, 6]])
U, s, VT = np.linalg.svd(A)
```

```
Sigma = np.zeros((A.shape[0], A.shape[1]))
np.fill_diagonal(Sigma, s)
print(U @ Sigma @ VT)                    # Reconstructed matrix
```

Program 5: Scatter Plot for Sales Data

Procedure:

1. Create scatter plots for different segments.

Code:

python

Copy code

```
import matplotlib.pyplot as plt

months = ['Jan', 'Feb', 'Mar', 'Apr']
affordable = [150, 200, 250, 200]
luxury = [80, 90, 100, 110]
plt.scatter(months, affordable, color='green', label='Affordable')
plt.scatter(months, luxury, color='yellow', label='Luxury')
plt.xlabel('Months'); plt.ylabel('Sales'); plt.legend()
plt.show()
```

Program 6: Bar Graph and Histogram

Procedure:

1. Create a bar graph and a histogram.

Code:

python

Copy code

```
modes = ['Walking', 'Cycling', 'Car']
students = [30, 12, 18]
plt.bar(modes, students, color='cyan')
plt.show()


ages = [5, 18, 22, 27, 30, 40]
plt.hist(ages, bins=5, color='green')
plt.show()
```

Program 7: Iris Dataset Analysis

Procedure:

1. Load the dataset, display properties, and visualize.

Code:

python

Copy code

```
import seaborn as sns
import pandas as pd


iris = pd.read_csv("iris.csv")
print(iris.shape, iris.head(), iris.describe())
sns.pairplot(iris, hue="variety")
sns.displot(iris["sepal_length"], bins=10, color="green")
plt.show()
```

Program 8: K-Nearest Neighbors (KNN) on Iris Dataset

Procedure:

1. Implement KNN on Iris dataset and evaluate.

Code:

python

Copy code

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X = iris.iloc[:, :-1].values
y = iris["variety"].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Program 9: Simple Linear Regression on Student Scores

Procedure:

1. Implement linear regression and evaluate with metrics.

Code:

python

Copy code

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

X = student_data.iloc[:, :-1].values
y = student_data.iloc[:, 1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)
print("MAE:", mean_absolute_error(y_test, y_pred))
```

Program 10: Multiple Linear Regression on Company Data

Procedure:

1. Implement multiple linear regression and evaluate.

Code:

python

Copy code

```
from sklearn.metrics import mean_squared_error, r2_score

X = company_data[['TV', 'Radio', 'Newspaper']]
y = company_data['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred), "R2:", r2_score(y_test, y_pred))
```

Program 11: Naive Bayes Classification on Iris Dataset

Procedure:

1. Implement Gaussian Naive Bayes and evaluate.

Code:

python

Copy code

```
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Program 12: K-Means Clustering on Customer Data

Procedure:

1. Perform K-Means clustering and visualize.

Code:

python

Copy code

```
from sklearn.cluster import KMeans


customers = pd.read_csv("customer_data.csv")
points = customers.iloc[:, 3:5].values
kmeans = KMeans(n_clusters=6).fit(points)
plt.scatter(points[:, 0], points[:, 1], c=kmeans.labels_, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red')
plt.show()
```