1. Write a Python program to implement Simple Linear Regression

[16]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

[11]:
```
#download salary_datas.csv file from github
df = pd.read_csv('salary_datas.csv')
```

[12]:
```
df.dropna(inplace=True)
```

[29]:
```
# Relationship between Salary and Experience
plt.scatter(df['YearsExperience'], df['Salary'], color = 'lightcoral')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()
```

[29]:
```
# Relationship between Salary and Experience
plt.scatter(df['YearsExperience'], df['Salary'], color = 'lightcoral')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()
```

[20]:
```
# Splitting variables
X = df.iloc[:, :-1].values  # independent
y = df.iloc[:, -1].values  # dependent
```

[21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

[22]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

[23]:
```
# Regressor model
regressor = LinearRegression()

regressor.fit(X_train, y_train)
```

[23]:
```
▾ LinearRegression

▶ Parameters
```

[24]:
```
# Prediction result
y_pred_test = regressor.predict(X_test)     # predicted value of y_test
y_pred_train = regressor.predict(X_train)   # predicted value of y_train
```

[30]:
```
plt.scatter(X_train, y_train, color = 'lightcoral',label='Actual Data')
plt.plot(X_train, y_pred_train, color = 'firebrick',label='Regression Line')
plt.title('Salary vs Experience (Training Set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend( title = 'Sal/Exp', loc='best', facecolor='white')
plt.box(False)
plt.show()
```


```
from sklearn.metrics import mean_absolute_error, mean_squared_error, root_mean_squared_error, r2_score

print(f"Mean Absolute Error (MAE): {mean_absolute_error(y_test, y_pred_test)}")
print(f"Mean Squared Error (MSE): {mean_squared_error(y_test, y_pred_test)}")
print(f"Root Mean Squared Error (RMSE): {root_mean_squared_error(y_test, y_pred_test)}")
print(f"R² Score: {r2_score(y_test, y_pred_test)}")
```

```
Mean Absolute Error (MAE): 0.08875600562570485
Mean Squared Error (MSE): 0.009752997593575868
Root Mean Squared Error (RMSE): 0.0987572660292693
R² Score: 0.9441906374625757
```

2. Write a program to demonstrate the use of various methods for data analysis and manipulation on a given dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# download student dataset from pandas example datasets in github
df = pd.read_csv("student_dataset.csv")
df.head()
```

	stud_id	stud_name	college_name	course	cgpa	placed or not	company_name
0	1	John Smith	XYZ University	Computer Science	8.4	Yes	TechCorp
1	2	Alice Brown	ABC Institute	Mechanical Engg	7.1	No	NaN
2	3	Bob White	PQR College	Electrical Engg	8.8	Yes	ElectroTech
3	4	Sarah Lee	XYZ University	Civil Engg	6.7	No	NaN
4	5	Tom Black	ABC Institute	IT	9.2	Yes	CodeWorks

```
df.shape
```

```
(60, 7)
```

```
df.columns
```

```
Index(['stud_id', 'stud_name', 'college_name', 'course', 'cgpa',
       'placed or not', 'company_name'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   stud_id        60 non-null     int64  
 1   stud_name      60 non-null     object 
 2   college_name   60 non-null     object 
 3   course         60 non-null     object 
 4   cgpa           59 non-null     float64
 5   placed or not  60 non-null     object 
 6   company_name   54 non-null     object 
dtypes: float64(1), int64(1), object(5)
memory usage: 3.4+ KB
```

```
df.describe()
```

	stud_id	cgpa
count	60.000000	59.000000
mean	30.500000	7.938983
std	17.464249	0.967759
min	1.000000	6.300000
25%	15.750000	7.000000
50%	30.500000	7.900000
75%	45.250000	8.800000
max	60.000000	9.400000

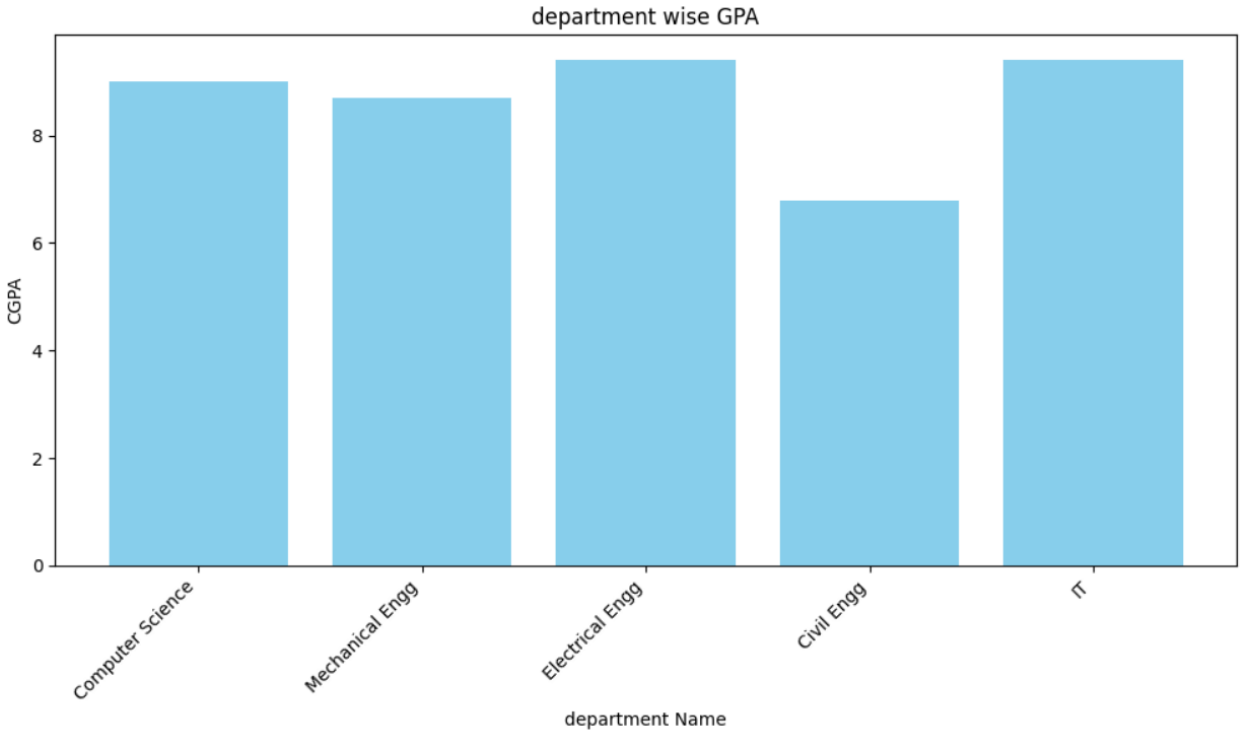
```
df.isnull().sum()
```

```
stud_id          0
stud_name        0
college_name     0
course           0
cgpa             1
placed or not    0
company_name     6
dtype: int64
```

```
names = df[(df['cgpa'] >= 7.0) & (df['cgpa'] <= 8.0) & (df['placed or not'] == 'Yes')]['stud_name']
print(names)
```

```
5      Emily Green
13      Megan Hall
20     Ethan Clark
25     Emma Walker
31    Elijah Adams
35    James Taylor
46    Harper Moore
49    Henry Taylor
54    Olivia Brown
58     Emma Walker
Name: stud_name, dtype: object
```

```python
import matplotlib.pyplot as plt

# Filtered DataFrame

# Plot
plt.figure(figsize=(10, 6))
plt.bar(df['course'], df['cgpa'], color='skyblue')
plt.xticks(rotation=45, ha='right')
plt.xlabel("department Name")
plt.ylabel("CGPA")
plt.title("department wise GPA")
plt.tight_layout()
plt.show()
```



3. Develop a Binary classification model using Logistic Regression and apply it to classify a new instance.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
df=pd.read_csv("student_data_synthetic_extreme_noisy.csv")
```

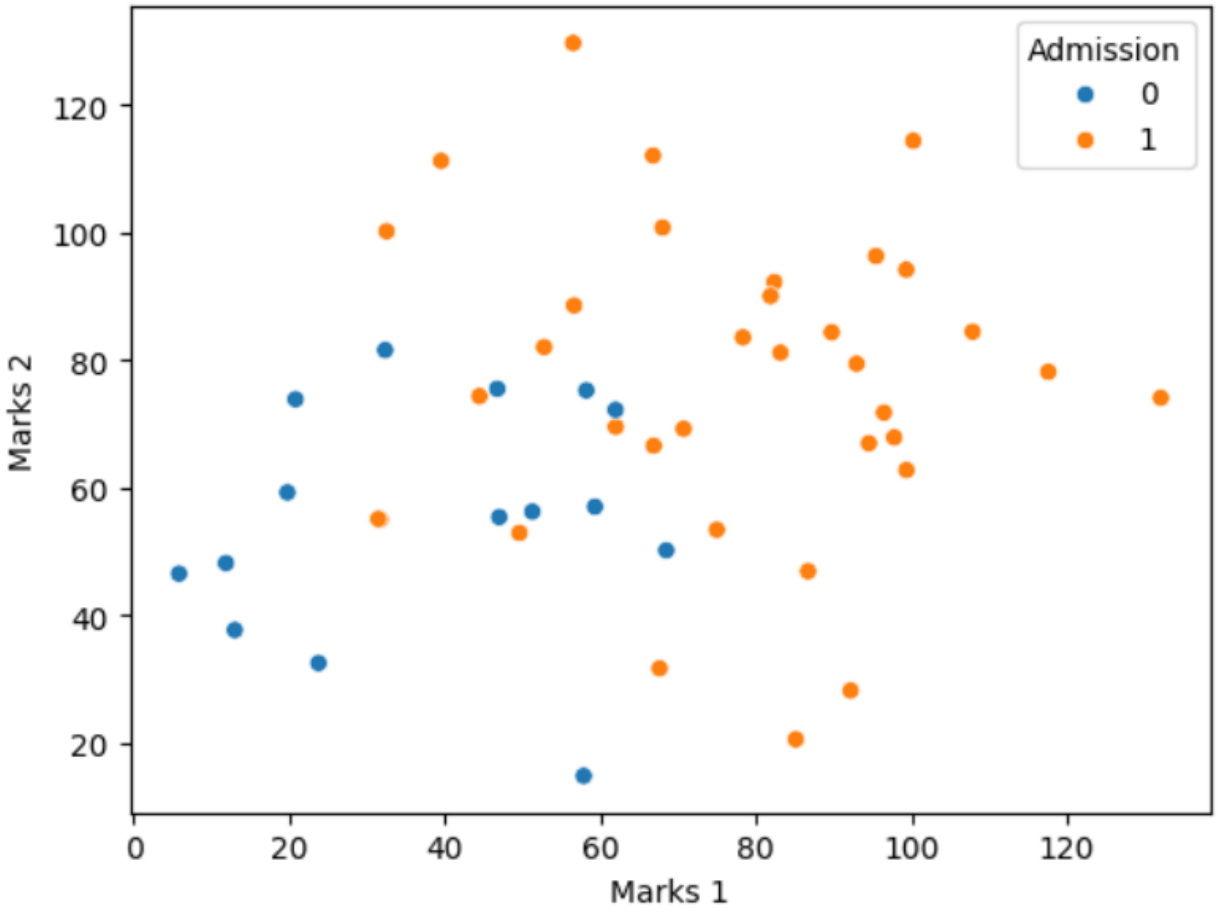
```
feature_cols = ['Marks 1', 'Marks 2']
X = df[feature_cols] # Features
y = df.Admission # Target variable
```

```
#finding whether dataset is imbalanced?
class_counts = df["Admission"].value_counts()
# Print class counts
print(class_counts)
```

```
Admission
1    35
0    15
Name: count, dtype: int64
```

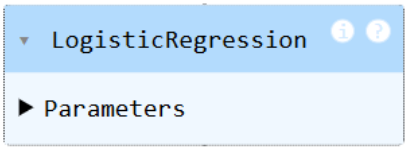
```
#visualize imbalanced dataset
#x-axis represents values of Marks1
#y-axis represents values of Marks 2
#hue represents the points are colored based on admission   column yes or no
import seaborn as sns
sns.scatterplot(x=df["Marks 1"], y=df["Marks 2"], hue=df["Admission"])
```

```
<Axes: xlabel='Marks 1', ylabel='Marks 2'>
```



```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0,stratify=y)
```

```
m1 = LogisticRegression(class_weight='balanced', penalty='l2', solver='lbfgs', max_iter=1000)

# fit the model with data
m1.fit(X_train,y_train)
```




```
m1.score(X_test,y_test)
```

```
0.9230769230769231
```

```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(m1, X, y, cv=5)
print("Cross-Validation Accuracy:", np.mean(cv_scores))
```

```
Cross-Validation Accuracy: 0.78
```

```
y_pred=m1.predict(X_test)
```

```
new_pred=m1.predict([[60,70]])
```

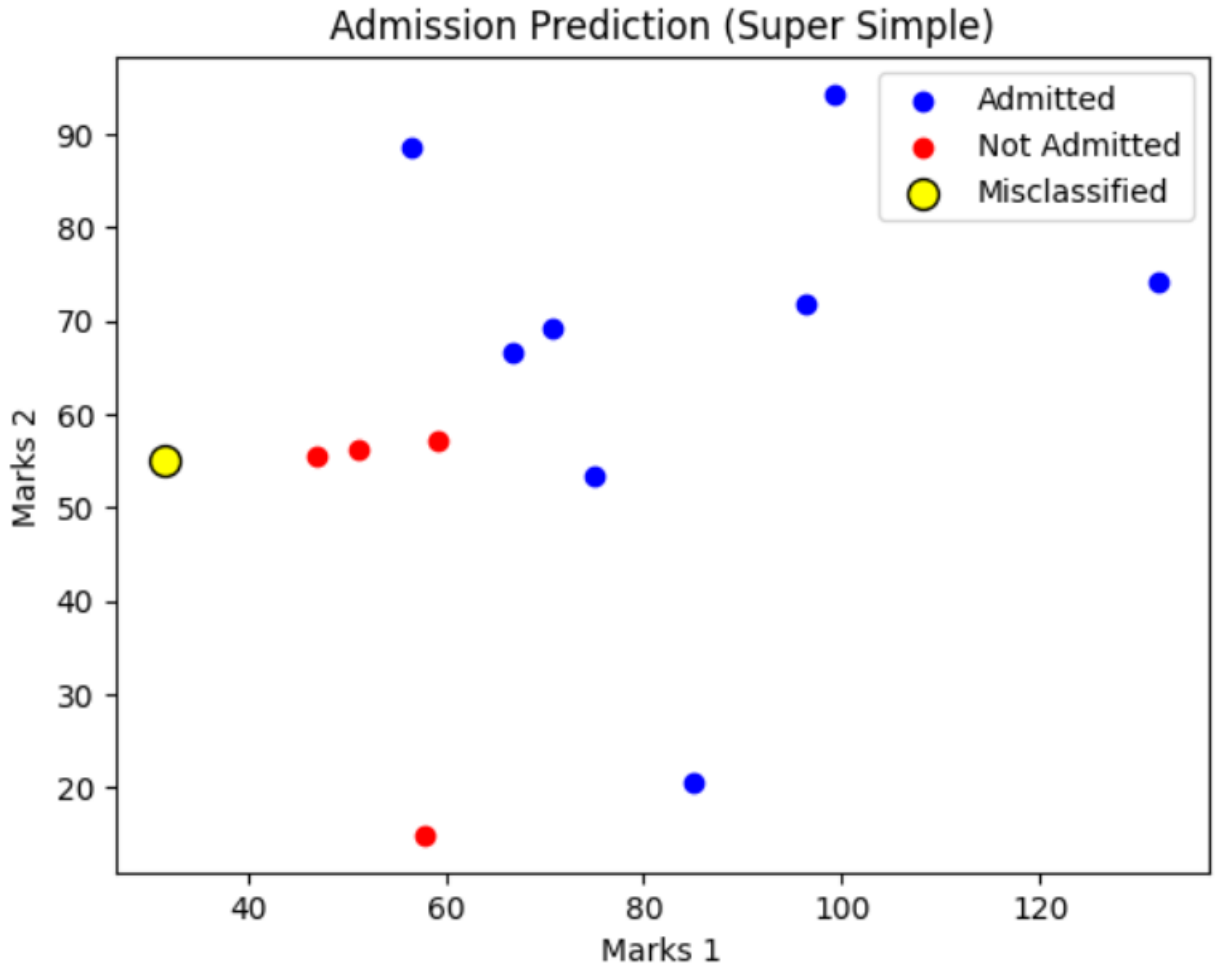
```
c1 = metrics.confusion_matrix(y_pred, y_test)
print(c1)

print("actual correct matrix")
tp=c1[1][1]
tn=c1[0][0]
fp=c1[0][1]
fn=c1[1][0]
print(np.array([tp,fn,fp,tn]))
```

```
[[4 1]
 [0 8]]
actual correct matrix
[8 0 1 4]
```

```
#evaluation metrices
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.9230769230769231
Precision: 1.0
Recall: 0.8888888888888888
```

```python
# Plot Admitted students (predicted correctly)
plt.scatter(X_test[y_pred==1]['Marks 1'], X_test[y_pred==1]['Marks 2'], color='blue', label='Admitted')

# Plot Not Admitted students (predicted correctly)
plt.scatter(X_test[y_pred==0]['Marks 1'], X_test[y_pred==0]['Marks 2'], color='red', label='Not Admitted')

# Plot Misclassified points (highlight in yellow)
misclassified = (y_test != y_pred)
plt.scatter(X_test[misclassified]['Marks 1'], X_test[misclassified]['Marks 2'],
            color='yellow', edgecolor='k', s=100, label='Misclassified')

# Add labels and title
plt.xlabel('Marks 1')
plt.ylabel('Marks 2')
plt.title('Admission Prediction (Super Simple)')

# Add a simple legend
plt.legend()

# Show the plot
plt.show()
```



4. Write a python program using Numpy to perform the following tasks

- Create a 1-D array and display the data type of the array.
- Create two 2D-arrays of the same shape and perform arithmetic operation on their elements.
- Concatenate the above 2D-arrays along rows or columns and display the result.
- Convert the concatenated array into a 1D-Array and display it.
- Create a 3x3 identity matrix and print its shape, number of dimensions and datatype.

```
import numpy as np
```

```
# Create a 1-D array and display the data type of the array.
array1=np.array([1,2,3,4 ,5])
array1
```

```
array([1, 2, 3, 4, 5])
```

```
# Create two 2D-arrays of the same shape and perform arithmetic operation on their elements.
ar1=np.array([[1,2,3],[4,5,6]])
ar2=np.array([[5,6,7],[8,9,10]])
#addition
addition=ar1+ar2
print("addition:\n",addition)
```

```
addition:
 [[ 6  8 10]
 [12 14 16]]
```

```
subtraction=ar1-ar2
print("subtraction:\n",subtraction)
```

```
subtraction:
 [[-4 -4 -4]
 [-4 -4 -4]]
```

```
multiplication=ar1*ar2
print("multiplication:\n",multiplication)
```

```
multiplication:
 [[ 5 12 21]
 [32 45 60]]
```

```
division=ar1/ar2
print("division:\n", division)
```

```
division:
 [[0.2        0.33333333 0.42857143]
 [0.5        0.55555556 0.6       ]]
```

```
power=np.power(ar1,ar2)
print("power:\n",power)
```

```
power:
 [[       1       64     2187]
 [   65536  1953125 60466176]]
```

```
# Concatenate the above 2D-arrays along rows or columns and display the result.
con = np.concatenate([ar1,ar2])
print(con)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 5  6  7]
 [ 8  9 10]]
```

```
#Convert the concatenated array into a 1D-Array and display it.
flat = con.flatten()
print(flat)
```

```
[ 1  2  3  4  5  6  5  6  7  8  9 10]
```

```
# Create a 3x3 identity matrix and print its shape, number of dimensions and datatype
array3d = np.identity(3)
array3d.shape
```

```
(3, 3)
```

```
array3d.ndim
```

```
2
```

```
array3d.dtype
```

```
dtype('float64')
```

5. Write a program to demonstrate Decision tree classifier. Use an appropriate dataset for building the model. 6. For the given dataset 'cars.csv' , perform the following

- Read the dataset
- Display last 5 rows
- Display first 5 rows
- Check for missing values
- Handle missing values
- Display the shape of dataset and statistical summary.
- Visualize the distribution of 'Doors' column using histogram.

```python
import pandas as pd
import numpy as np
```

```python
# Read the dataset
df = pd.read_csv("cars.csv")
```

```python
# First 5 rows
print("First 5 rows:\n", df.head())

# Last 5 rows
print("\nLast 5 rows:\n", df.tail())
```

```python
# check for missing values
print("\nMissing values per column:\n", df.isnull().sum())
```

```python
# Handle missing values
# Drop all rows with any missing values
df_cleaned = df.dropna()
```

```python
# Display Shape and Statistical Summary
print("\nShape of dataset:", df.shape)
print("\nStatistical Summary:\n", df.describe(include='all'))
```

```python
import matplotlib.pyplot as plt

df["Door"].hist(bins=range(1, 7))
plt.title(f"Distribution of '{door_col[0]}'")
plt.xlabel("Number of Doors")
plt.ylabel("Count")
plt.show()
```

7. Develop a model for multiclass classification using KNN classifier with K=5.Use an appropriate dataset for building the model.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.metrics import accuracy_score
```

```
df=pd.read_csv("./datasets/weight_height_dataset.csv")
df
```

	Height(cm)	Weight(kg)	Class
0	171.408421	69.037935	Normal
1	153.935688	47.797508	Underweight
2	176.573961	78.871438	Overweight

```
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

```
print("X_train:",X_train.shape)
print("X_test:",X_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
X_train: (112, 2)
X_test: (38, 2)
y_train (112,)
y_test (38,)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_trainn=sc.fit_transform(X_train)
X_testt=sc.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
#KNeighborsClassifier?
#metric- often we use euclidean distance, but there are many such as mahattan d
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_trainn, y_train)
```

8. Write a python program using Numpy to perform the following tasks

- Create a 2x4 array of zeros and 4x2 array of ones and display it.
- For each of the above arrays , print their shape, number of dimensions and datatype.
- Convert 4x2 array of ones into 1D- Array and display the result.
- Reshape it into a different valid shape.
- Create a 4D-Array and convert its element to float

```python
import numpy as np

# Create a 2x4 array of zeros
zeros_array = np.zeros((2, 4))
print("2x4 Array of Zeros:\n", zeros_array)
```

```
2x4 Array of Zeros:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```python
# Create a 4x2 array of ones
ones_array = np.ones((4, 2))
print("\n4x2 Array of Ones:\n", ones_array)
```

```
4x2 Array of Ones:
 [[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]
```

```python
# Print shape, dimensions, and data type of both arrays
print("Zeros Array - Shape:", zeros_array.shape, ", Dimensions:", zeros_array.ndim, ", DataType:", zeros_array.dtype)
print("Ones Array - Shape:", ones_array.shape, ", Dimensions:", ones_array.ndim, ", DataType:", ones_array.dtype)
```

```
Zeros Array - Shape: (2, 4) , Dimensions: 2 , DataType: float64
Ones Array - Shape: (4, 2) , Dimensions: 2 , DataType: float64
```

```python
# Convert 4x2 array of ones into 1D array
ones_flat = ones_array.flatten()
print("\n1D version of 4x2 Ones Array:\n", ones_flat)
```

```
1D version of 4x2 Ones Array:
 [1. 1. 1. 1. 1. 1. 1. 1.]
```

```python
# Reshape into a different valid shape (e.g., 2x4)
reshaped = ones_flat.reshape((2, 4))
print("\nReshaped to 2x4:\n", reshaped)
```

```
Reshaped to 2x4:
 [[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

```python
# Create a 4D array (e.g., shape 2x1x2x2)
array_4d = np.array([[[[1, 2], [3, 4]]], [[[5, 6], [7, 8]]]])
print("\n4D Array:\n", array_4d)
```

```
4D Array:
 [[[[1 2]
   [3 4]]]


 [[[5 6]
   [7 8]]]]
```

```python
# Convert its elements to float
array_4d_float = array_4d.astype(float)
print("\n4D Array with Float Type:\n", array_4d_float)
```

```
4D Array with Float Type:
 [[[[1. 2.]
   [3. 4.]]]


 [[[5. 6.]
   [7. 8.]]]]
```