Machine Learning Week – 1 Programs

1. Write a program to perform basic array operations using NumPy?

Python program that performs basic array operations using the NumPy library. The program covers the following operations:

1. **Creating arrays**
2. **Array indexing and slicing**
3. **Reshaping arrays**
4. **Array statistics (mean, median, mode, sum, etc.)**

Program

```
import numpy as np
# 1. Creating arrays
# Create a 1D array
array1 = np.array([1, 2, 3, 4, 4])
print("1D Array:")
print(array1)
# Create a 2D array
array2 = np.array([[1, 2, 3], [4, 5, 6]])
print("\n2D Array:")
print(array2)
# Create an array of zeros
zeros_array = np.zeros((2, 3))  # 2x3 matrix of zeros
print("\nArray of Zeros:")
print(zeros_array)
# Create an array of ones
ones_array = np.ones((3, 2))  # 3x2 matrix of ones
print("\nArray of Ones:")
print(ones_array)
# Create an array with a specific range
range_array = np.arange(0, 10, 2)  # array with values from 0 to 10 with a step of 2
print("\nArray with Specific Range (0 to 10, step 2):")
print(range_array)
# 2. Array Indexing and Slicing
# Accessing elements
print("\nElement at index 2 of array1:", array1[2])
# Slicing an array
print("\nSliced array1 (from index 1 to 4):", array1[1:4])
# 3. Reshaping Arrays
# Reshape a 1D array to a 2D array (5 elements to 1 row and 5 columns)
reshaped_array = array1.reshape(1, 5)
print("\nReshaped Array (1x5):")
print(reshaped_array)
```

```
# Reshape a 1D array to a 2D array (5 elements to 5 rows and 1 column)
reshaped_array_2d = array1.reshape(5, 1)
print("\nReshaped Array (5x1):")
print(reshaped_array_2d)
# 4. Array Statistics
# Mean of array1
mean_value = np.mean(array1)
print("\nMean of array1:", mean_value)
# Median of array1
median_value = np.median(array1)
print("\nMedian of array1:", median_value)
# Sum of array1
sum_value = np.sum(array1)
print("Sum of array3:", sum_value)
# Maximum and Minimum of array1
max_value = np.max(array1)
min_value = np.min(array1)
print("Maximum of array1:", max_value)
print("Minimum of array1:", min_value)
# Standard deviation of array1
std_dev = np.std(array1)
print("Standard Deviation of array1:", std_dev)
```

2. Write a program to perform linear algebra operations (like matrix multiplication) using NumPy?

Program

```
#  Array Arithmetic Operations
# Array addition, subtraction, multiplication, and division
array1 = np.array([[10, 20],[30, 40]])
array2 = np.array([[1, 2],[3, 4]])
# Addition
addition = array1 + array2
print("\nArray Addition:", addition)
# Subtraction
subtraction = array1 – array2
print("Array Subtraction:", subtraction)
# Multiplication
multiplication = np.dot(array1, array2)
print("Array Multiplication:", multiplication)
# Division
division = array1 / array2
print("Array Division:", division)
```

3. Program to clean and preprocess data using Pandas (handling missing values, removing duplicates, etc.)

Program :

```
import pandas as pd
import numpy as np
# Sample DataFrame with missing values, duplicates, and inconsistent data
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Alice', None],
    'Age': [25, 30, np.nan, 35, 25, 40],
    'City': ['New York', 'Los Angeles', 'New York', None, 'New York', 'Chicago'],
    'Salary': [70000, np.nan, 80000, 120000, 70000, 95000]
}
# Create DataFrame
df = pd.DataFrame(data)
# Display the original DataFrame
print("Original DataFrame:")
print(df)
# 1. Handling Missing Values:
# a. Handling missing values in numerical columns: Use mean or median
df['Age'] = df['Age'].fillna(df['Age'].mean())  # Filling missing Age with the mean
df['Salary'] = df['Salary'].fillna(df['Salary'].median())  # Filling missing Salary with the median
# b. Handling missing values in categorical columns: Use the mode or drop rows/columns
df['City'] = df['City'].fillna(df['City'].mode()[0])  # Filling missing City with the mode
# Display DataFrame after handling missing values
print("\nDataFrame after handling missing values:")
print(df)
# 2. Removing Duplicates:
df_cleaned = df.drop_duplicates()  # Drop duplicate rows
# Display DataFrame after removing duplicates
print("\nDataFrame after removing duplicates:")
print(df_cleaned)
# 3. Handling Inconsistent Data:
# For example, let's assume 'Age' has negative values, which is incorrect.
# We can clean by removing rows where 'Age' is negative.
df_cleaned = df_cleaned[df_cleaned['Age'] >= 0]
# Display DataFrame after handling inconsistent data
print("\nDataFrame after handling inconsistent data (Age >= 0):")
print(df_cleaned)
# 4. Converting Data Types:
# Let's say we want to convert the 'Salary' column to integer
df_cleaned['Salary'] = df_cleaned['Salary'].astype(int)
# Display DataFrame after converting data types
print("\nDataFrame after converting 'Salary' to integer:")
```

```
print(df_cleaned)
# 5. Resetting Index (optional):
# After cleaning, we might want to reset the DataFrame's index.
df_cleaned.reset_index(drop=True, inplace=True)

# Display the final cleaned DataFrame
print("\nFinal Cleaned DataFrame:")
print(df_cleaned)
```

4. Python program to load the csv file data into dataframe and print dataframe

Program :

```
import pandas as pd
# Step 1: Create a DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David'],
   'Age': [25, 30, 35, 40],
   'City': ['New York', 'Los Angeles', 'Chicago', 'San Francisco']
}
# Create a DataFrame
df = pd.DataFrame(data)
# Step 2: Save the DataFrame to a CSV file
file_path = 'example_data.csv'  # Specify the file path to save the CSV
df.to_csv(file_path, index=False)  # Save DataFrame to CSV (index=False avoids saving the index)
# Step 3: Load the CSV file into a DataFrame
df_loaded = pd.read_csv(file_path)
# Step 4: Print the DataFrame
print("DataFrame loaded from CSV:")
print(df_loaded)
```

5. Program to analyze and manipulate time series data using Pandas?

Program :

```
import pandas as pd
# Step 1: Create a time series DataFrame
# Generate dates from 2024-01-01 to 2024-01-10, with daily frequency
dates = pd.date_range('2024-01-01', periods=10, freq='D')
# Create a simple data dictionary with dates and corresponding sales data
data = {
   'Date': dates,
   'Sales': [100, 120, 150, 180, 200, 210, 230, 250, 280, 300]
```

```
}
# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)
# Step 2: Set 'Date' as the index to make it a time series DataFrame
df.set_index('Date', inplace=True)
# Step 3: Print the DataFrame
print("Time Series Data:")
print(df)
.
```

6.Program to aggregate data using the Pandas groupby function?

Program :

```
import pandas as pd
# Step 1: Create a sample DataFrame
data = {
    'Department': ['HR', 'Sales', 'HR', 'Sales', 'IT', 'IT', 'HR', 'Sales'],
    'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Helen'],
    'Salary': [50000, 60000, 55000, 62000, 70000, 75000, 52000, 61000],
    'Age': [25, 30, 35, 40, 45, 50, 28, 33]
}
df = pd.DataFrame(data)
# Step 2: Group data by 'Department' and calculate aggregate statistics
grouped = df.groupby('Department')
# Step 3: Calculate the sum of salaries for each department
salary_sum = grouped['Salary'].sum()
# Step 4: Calculate the average salary for each department
salary_avg = grouped['Salary'].mean()
# Step 5: Calculate the average age for each department
age_avg = grouped['Age'].mean()
# Step 6: Print the results
print("Total Salary by Department:")
print(salary_sum)
print("\nAverage Salary by Department:")
print(salary_avg)
print("\nAverage Age by Department:")
print(age_avg)
```