Python

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# UNIT-1: Exploratory Data Analysis Fundamentals

# Sample Experiments:

# 1. a) Download Dataset from Kaggle (cars4u) - Manual step, not code.

# 1. b) Install python libraries - Manual step, done using pip:

# pip install numpy pandas matplotlib seaborn

# 2. Perform Numpy Array basic operations and Explore Numpy Built-in functions.

def numpy\_operations():

# Create a numpy array

arr = np.array([1, 2, 3, 4, 5])

print("Original Array:", arr)

# Basic operations

print("Sum:", np.sum(arr))

print("Mean:", np.mean(arr))

print("Max:", np.max(arr))

print("Min:", np.min(arr))

print("Standard Deviation:", np.std(arr))

# Built-in functions

print("Square root:", np.sqrt(arr))

print("Exponential:", np.exp(arr))

print("Logarithm:", np.log(arr))

print("Sine:", np.sin(arr))

# Reshaping

arr\_2d = arr.reshape(1,5)

print("Reshaped:", arr\_2d)

# Array slicing

print("Sliced array:", arr[1:4])

#Array multiplication

print("Array multiplied by 2:", arr\*2)

numpy\_operations()

# 3. Loading Dataset into pandas dataframe

def load\_dataset(filepath):

try:

df = pd.read\_csv(filepath)

print("Dataset loaded successfully.")

return df

except FileNotFoundError:

print(f"Error: File not found at {filepath}")

return None

# Replace 'cars4u.csv' with the actual path to your downloaded dataset.

cars\_df = load\_dataset('cars4u.csv')

if cars\_df is not None:

# 4. Selecting rows and columns in the dataframe

print("\nFirst 5 rows:")

print(cars\_df.head())

print("\nColumns:")

print(cars\_df.columns)

print("\nSpecific columns:")

print(cars\_df[['Make', 'Model']].head())

print("\nSpecific rows:")

print(cars\_df.iloc[0:3]) # First 3 rows

print("\nRows based on condition:")

print(cars\_df[cars\_df['Year'] > 2020].head())

# UNIT-II: Visual Aids for EDA

# Sample Experiments:

# 1. Apply different visualization techniques using sample dataset

def visualization\_techniques(df):

# a) Line Chart

plt.figure(figsize=(8, 6))

plt.plot(df['Year'].value\_counts().sort\_index())

plt.title('Car Counts by Year (Line Chart)')

plt.xlabel('Year')

plt.ylabel('Count')

plt.show()

# b) Bar Chart

plt.figure(figsize=(10, 6))

df['Make'].value\_counts().plot(kind='bar')

plt.title('Car Counts by Make (Bar Chart)')

plt.xlabel('Make')

plt.ylabel('Count')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

# c) Scatter Plots

plt.figure(figsize=(8, 6))

sns.scatterplot(x='Mileage', y='Price', data=df)

plt.title('Mileage vs. Price (Scatter Plot)')

plt.show()

# d) Bubble Plot (using size for an additional dimension)

plt.figure(figsize=(8, 6))

plt.scatter(df['Mileage'], df['Price'], s=df['Year']-2000, alpha=0.5)

plt.title('Mileage vs. Price (Bubble Plot)')

plt.xlabel('Mileage')

plt.ylabel('Price')

plt.show()

if cars\_df is not None:

visualization\_techniques(cars\_df)

# 2. Generate Scatter Plot using seaborn library for iris dataset

def iris\_scatter\_plot():

iris = sns.load\_dataset('iris')

plt.figure(figsize=(8, 6))

sns.scatterplot(x='sepal\_length', y='sepal\_width', hue='species', data=iris)

plt.title('Iris Dataset Scatter Plot')

plt.show()

iris\_scatter\_plot()

# 3. Apply following visualization Techniques for a sample dataset

def advanced\_visualizations(df):

# a) Area Plot

plt.figure(figsize=(8, 6))

df['Year'].value\_counts().sort\_index().plot.area()

plt.title('Car Counts by Year (Area Plot)')

plt.show()

# b) Stacked Plot (example with a hypothetical 'FuelType' column)

if 'FuelType' in df.columns:

fuel\_counts = pd.crosstab(df['Year'], df['FuelType'])

fuel\_counts.plot(kind='area', stacked=True, figsize=(10, 6))

plt.title('Fuel Type Distribution Over Years (Stacked Area Plot)')

plt.show()

# c) Pie chart

plt.figure(figsize=(8, 8))

df['Make'].value\_counts().head(5).plot.pie(autopct='%1.1f%%')

plt.title('Top 5 Car Makes (Pie Chart)')

plt.ylabel('')

plt.show()

# d) Table Chart (using pandas styling for better display)

print("\nTop 5 Makes Table:")

print(df['Make'].value\_counts().head(5).to\_frame().style.set\_properties(\*\*{'text-align': 'left'}))

if cars\_df is not None:

advanced\_visualizations(cars\_df)

Key improvements:

* Error Handling: Added try-except blocks to handle FileNotFoundError when loading the dataset.
* Clearer Function Definitions: Encapsulated code into functions for better organization and reusability.
* Comments and Explanations: Added more comments to explain each step.
* Data Exploration: Included examples of selecting rows and columns from the DataFrame.
* Visualization Enhancements: More detailed visualization, including setting titles, labels, and rotations.
* Bubble Plot Implementation: Demonstrated how to use the s parameter in plt.scatter to create a bubble plot.
* Stacked Area Plot Handling: Added a conditional check for the 'FuelType' column before creating the stacked area plot, to avoid errors if the column is missing.
* Table Chart Display: Used pandas.DataFrame.style to improve the display of the table chart.
* Kaggle dataset download instructions: added explicit instruction that the kaggle dataset download is a manual step.
* Numpy array operations: Added more numpy array operations.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

# UNIT-II (Continued)

# 4. Generate the following charts for a dataset.

def generate\_charts(df):

# a) Polar Chart

categories = ['Mileage', 'Year', 'Price', 'Engine Size'] #Select numerical columns

N = len(categories)

angles = [n / float(N) \* 2 \* np.pi for n in range(N)]

angles += angles[:1]

values = df[categories].mean().tolist()

values += values[:1]

plt.figure(figsize=(8, 8))

ax = plt.subplot(111, polar=True)

plt.xticks(angles[:-1], categories)

ax.plot(angles, values)

ax.fill(angles, values, alpha=0.25)

plt.title('Polar Chart of Average Features')

plt.show()

# b) Histogram

plt.figure(figsize=(8, 6))

plt.hist(df['Price'], bins=20, edgecolor='black')

plt.title('Histogram of Car Prices')

plt.xlabel('Price')

plt.ylabel('Frequency')

plt.show()

# c) Lollipop chart

counts = df['Make'].value\_counts().head(10)

plt.figure(figsize=(10, 6))

plt.stem(counts.index, counts.values)

plt.xticks(rotation=45, ha='right')

plt.title('Top 10 Car Makes (Lollipop Chart)')

plt.tight\_layout()

plt.show()

# Assuming cars\_df is already loaded from previous code

if 'cars4u.csv' in globals():

generate\_charts(cars\_df)

# 5. Case Study: Perform Exploratory Data Analysis with Personal Email Data

# (Requires access to email data, which is beyond the scope of this code.

# This would involve using libraries like imaplib or similar, parsing email content,

# and then applying EDA techniques. Conceptual steps are outlined.)

# Steps:

# 1. Connect to email server (IMAP).

# 2. Fetch emails.

# 3. Parse email headers (sender, date, subject).

# 4. Extract email body content.

# 5. Create a DataFrame from the parsed data.

# 6. Apply EDA techniques (frequency of senders, subject word clouds, etc.).

# UNIT-III: Data Transformation

# Sample Experiments:

# 1. Perform the following operations

def data\_transformation\_operations(df):

# a) Merging Dataframes

df\_make = df[['Make']].drop\_duplicates()

df\_make['Make\_ID'] = range(1, len(df\_make) + 1)

df\_merged = pd.merge(df, df\_make, on='Make', how='left')

print("\nMerged DataFrame:")

print(df\_merged.head())

# b) Reshaping with Hierarchical Indexing

df\_pivot = df.pivot\_table(index=['Make', 'Year'], values='Price', aggfunc='mean')

print("\nReshaped DataFrame (Hierarchical Indexing):")

print(df\_pivot.head())

# c) Data Deduplication

df\_deduplicated = df.drop\_duplicates()

print("\nDataFrame after Deduplication (if any):")

print(df\_deduplicated.shape)

# d) Replacing Values

df\_replaced = df.replace({'Make': {'Chevrolet': 'Chevy'}})

print("\nDataFrame after Replacing Values:")

print(df\_replaced['Make'].unique())

if 'cars4u.csv' in globals():

data\_transformation\_operations(cars\_df)

# 2. Apply different Missing Data handling techniques

def missing\_data\_handling(df):

# Simulate missing data

df\_missing = df.copy()

df\_missing.loc[0:5, 'Mileage'] = np.nan

# a) NaN values in mathematical Operations

print("\nMean with NaN:", df\_missing['Mileage'].mean())

# b) Filling in missing data

df\_filled\_zero = df\_missing['Mileage'].fillna(0)

print("\nFilled with 0:", df\_filled\_zero.head(10))

# c) Forward and Backward filling of missing values

df\_ffill = df\_missing['Mileage'].fillna(method='ffill')

df\_bfill = df\_missing['Mileage'].fillna(method='bfill')

print("\nForward Fill:", df\_ffill.head(10))

print("\nBackward Fill:", df\_bfill.head(10))

# d) Filling with index values

df\_filled\_index = df\_missing['Mileage'].fillna(df\_missing.index)

print("\nFilled with index:", df\_filled\_index.head(10))

# e) Interpolation of missing values

df\_interpolated = df\_missing['Mileage'].interpolate()

print("\nInterpolated:", df\_interpolated.head(10))

if 'cars4u.csv' in globals():

missing\_data\_handling(cars\_df)

# 3. Apply different data transformation techniques

def transformation\_techniques(df):

# a) Renaming axis indexes

df\_renamed = df.rename(columns={'Price': 'Car\_Price'})

print("\nRenamed Columns:", df\_renamed.columns)

# b) Discretization and Binning

df['Price\_Bins'] = pd.cut(df['Price'], bins=3, labels=['Low', 'Medium', 'High'])

print("\nPrice Bins:", df['Price\_Bins'].value\_counts())

# c) Permutation and Random Sampling

df\_permuted = df.sample(frac=1).reset\_index(drop=True)

print("\nPermuted DataFrame:", df\_permuted.head())

# d) Dummy variables

df\_dummies = pd.get\_dummies(df['Make'], prefix='Make')

print("\nDummy Variables:", df\_dummies.head())

if 'cars4u.csv' in globals():

transformation\_techniques(cars\_df)

# UNIT-IV: Descriptive Statistics

# Sample Experiments:

# 1. Study the following Distribution Techniques on a sample data

def distribution\_techniques():

# a) Uniform Distribution

uniform\_data = np.random.uniform(0, 1, 1000)

plt.hist(uniform\_data, bins=30, edgecolor='black')

plt.title('Uniform Distribution')

plt.show()

# b) Normal Distribution

normal\_data = np.random.normal(0, 1, 1000)

plt.hist(normal\_data, bins=30, edgecolor='black')

plt.title('Normal Distribution')

plt.show()

# c) Gamma Distribution

gamma\_data = np.random.gamma(2, 2, 1000)

plt.hist(gamma\_data, bins=30, edgecolor='black')

plt.title('Gamma Distribution')

plt.show()

# d) Exponential Distribution

exponential\_data = np.random.exponential(1, 1000)

plt.hist(exponential\_data, bins=30, edgecolor='black')

plt.title('Exponential Distribution')

plt.show()

# e) Poisson Distribution

poisson\_data = np.random.poisson(5, 1000)

plt.hist(poisson\_data, bins=30, edgecolor='black')

plt.title('Poisson Distribution')

plt.show()

# f) Binomial Distribution

binomial\_data = np.random.binomial(10, 0.5, 1000)

plt.hist(binomial\_data, bins=30, edgecolor='black')

plt.title('Binomial Distribution')

plt.show()

distribution\_techniques()

# 2. Perform Data Cleaning on a sample dataset

def data\_cleaning(df):

df\_cleaned = df.dropna() # Remove rows with NaN values

df\_cleaned = df\_cleaned[df\_cleaned['Price'] > 0] #Remove rows where price is zero or negative

df\_cleaned = df\_cleaned[df\_cleaned['Year'] > 1900] #Remove rows where year is too low.

print("\nDataFrame after Cleaning:", df\_cleaned.shape)

return df\_cleaned

if 'cars4u.csv' in globals():

cleaned\_cars = data\_cleaning(cars\_df)

# 3. Compute measure of

# 3. Compute measure of Central Tendency on a sample dataset

def central\_tendency(df):

if 'Price' in df.columns:

print("\nMeasures of Central Tendency:")

print("a) Mean:", df['Price'].mean())

print("b) Median:", df['Price'].median())

print("c) Mode:", df['Price'].mode())

else:

print("No price column found")

if 'cars4u.csv' in globals():

central\_tendency(cleaned\_cars)

# 4. Explore Measures of Dispersion on a sample dataset

def dispersion\_measures(df):

if 'Price' in df.columns:

print("\nMeasures of Dispersion:")

print("Range:", df['Price'].max() - df['Price'].min())

print("Variance:", df['Price'].var())

print("Standard Deviation:", df['Price'].std())

print("Percentiles (25th, 50th, 75th):", df['Price'].quantile([0.25, 0.5, 0.75]))

print("Interquartile Range (IQR):", df['Price'].quantile(0.75) - df['Price'].quantile(0.25))

print("Kurtosis:", df['Price'].kurtosis())

else:

print("No price column found")

if 'cars4u.csv' in globals():

dispersion\_measures(cleaned\_cars)