# Assignment

1. **Create a code for digital registration application of car sales users.**

## Code :

jupyter Ai&ML-Assignment1 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[3]:

```python
import pandas as pd

def digital_registration():
    print("--- Car Sales User Registration Application ---")

    # List to store multiple user registrations
    user_database = []

    while True:
        # Collecting user data
        name = input("Enter Full Name: ")
        email = input("Enter Email Address: ")
        phone = input("Enter Phone Number: ")
        user_type = input("Enter User Type (Buyer/Seller/Dealer): ")

        # Creating a record for the user
        user_record = {
            "Full Name": name,
            "Email": email,
            "Phone": phone,
            "User Type": user_type
        }

        # Adding to our database list
        user_database.append(user_record)

        # Check if more users need to be registered
        cont = input("\nRegister another user? (yes/no): ").lower()
        if cont != 'yes':
            break
```

jupyter Ai&ML-Assignment1 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
        # Check if more users need to be registered
        cont = input("\nRegister another user? (yes/no): ").lower()
        if cont != 'yes':
            break

    # Converting the registration list into a DataFrame for better organization
    df = pd.DataFrame(user_database)

    print("\n--- Final Registration Database ---")
    print(df)

    # Optional: Save to CSV
    # df.to_csv("user_registrations.csv", index=False)

# Run the application
if __name__ == "__main__":
    digital_registration()
```

## Output:

jupyter Ai&ML-Assignment1 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
# Run the application
if __name__ == "__main__":
    digital_registration()
```

```
--- Car Sales User Registration Application ---
Enter Full Name:  Bhuvaneshwari Rebba
Enter Email Address:  bhuvaneshwaritsms010@gmail.com
Enter Phone Number:  9701473371
Enter User Type (Buyer/Seller/Dealer):  buyer

Register another user? (yes/no):  yes
Enter Full Name:  Bhuvana
Enter Email Address:  bhuvana2gmail.com
Enter Phone Number:  6305640563
Enter User Type (Buyer/Seller/Dealer):  buyer

Register another user? (yes/no):  no

--- Final Registration Database ---
             Full Name                         Email       Phone User Type
0  Bhuvaneshwari Rebba  bhuvaneshwaritsms010@gmail.com  9701473371     buyer
1              Bhuvana               bhuvana2gmail.com  6305640563     buyer
```

2) Create a DataFrame of 20 rows and 8 columns based on past 20 days gold rates of 22k and 24k separately.

## Code :

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
[1]: import pandas as pd
import numpy as np
from datetime import date, timedelta

# 1. Define the number of rows (days) and create a date range
num_rows = 20
start_date = date.today() - timedelta(days=num_rows - 1)
dates = [start_date + timedelta(days=i) for i in range(num_rows)]

# 2. Generate simulated gold rate data for 22k and 24k
# Rates are usually per 10 grams. Let's create 4 columns for each:
# (Rate_per_10g, Change_in_rate, Purity_percentage, Location)

# Base rates (simulated starting point, e.g., in INR)
base_22k = 55000
base_24k = 60000

# Create simulated daily fluctuations (e.g., +/- 200)
fluctuations = np.random.uniform(-200, 200, size=num_rows)

# Calculate 24k rates: a base rate with fluctuations
rate_24k_10g = base_24k + fluctuations

# Calculate 22k rates: approx 91.6% of 24k rates, plus some small unique noise
rate_22k_10g = (rate_24k_10g * 0.916) + np.random.uniform(-50, 50, size=num_rows)

# Calculate daily change (using a shifted array to get the difference)
change_24k = np.insert(np.diff(rate_24k_10g), 0, 0)
change_22k = np.insert(np.diff(rate_22k_10g), 0, 0)
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
# Calculate daily change (using a shifted array to get the difference)
change_24k = np.insert(np.diff(rate_24k_10g), 0, 0)
change_22k = np.insert(np.diff(rate_22k_10g), 0, 0)

# 3. Create the DataFrame (8 columns)
data = {
    'Date': dates,
    'Location': np.random.choice(['City A', 'City B', 'City C'], num_rows),
    '24k_Rate_per_10g': rate_24k_10g.round(2),
    '24k_Change': change_24k.round(2),
    '24k_Purity_%': 99.9,
    '22k_Rate_per_10g': rate_22k_10g.round(2),
    '22k_Change': change_22k.round(2),
    '22k_Purity_%': 91.6,
}

gold_rates_df = pd.DataFrame(data)

print("### Gold Rates DataFrame (20 Rows, 8 Columns) ###")
print(gold_rates_df)
print(f"\nShape: {gold_rates_df.shape}")
```

# Output:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
gold_rates_df = pd.DataFrame(data)

print("### Gold Rates DataFrame (20 Rows, 8 Columns) ###")
print(gold_rates_df)
print(f"\nShape: {gold_rates_df.shape}")
```

```
### Gold Rates DataFrame (20 Rows, 8 Columns) ###
          Date Location  24k_Rate_per_10g  24k_Change  24k_Purity_%  \
0   2025-11-26   City B          60145.23        0.00          99.9
1   2025-11-27   City A          59941.66     -203.56          99.9
2   2025-11-28   City A          59887.87      -53.80          99.9
3   2025-11-29   City C          60128.15      240.28          99.9
4   2025-11-30   City A          60015.92     -112.23          99.9
5   2025-12-01   City B          59941.75      -74.17          99.9
6   2025-12-02   City B          60130.96      189.21          99.9
7   2025-12-03   City B          60121.67       -9.29          99.9
8   2025-12-04   City C          60154.62       32.95          99.9
9   2025-12-05   City A          59856.92     -297.69          99.9
10  2025-12-06   City A          59885.84       28.92          99.9
11  2025-12-07   City A          59924.26       38.41          99.9
12  2025-12-08   City A          60008.47       84.21          99.9
13  2025-12-09   City B          59905.88     -102.59          99.9
14  2025-12-10   City C          59976.45       70.57          99.9
15  2025-12-11   City B          59911.18      -65.27          99.9
16  2025-12-12   City C          60107.60      196.43          99.9
17  2025-12-13   City B          59826.21     -281.39          99.9
18  2025-12-14   City A          59957.28      131.07          99.9
19  2025-12-15   City C          60066.60      109.31          99.9

    22k_Rate_per_10g  22k_Change  22k_Purity_%
0           55069.39        0.00          91.6
1           54894.41     -174.97          91.6
2           54876.70      -17.71          91.6
3           55123.10      246.40          91.6
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
10  2025-12-06  City A      59885.84     28.92        99.9
11  2025-12-07  City A      59924.26     38.41        99.9
12  2025-12-08  City A      60008.47     84.21        99.9
13  2025-12-09  City B      59905.88   -102.59        99.9
14  2025-12-10  City C      59976.45     70.57        99.9
15  2025-12-11  City B      59911.18    -65.27        99.9
16  2025-12-12  City C      60107.60    196.43        99.9
17  2025-12-13  City B      59826.21   -281.39        99.9
18  2025-12-14  City A      59957.28    131.07        99.9
19  2025-12-15  City C      60066.60    109.31        99.9

    22k_Rate_per_10g  22k_Change  22k_Purity_%
0           55069.39        0.00          91.6
1           54894.41     -174.97          91.6
2           54876.70      -17.71          91.6
3           55123.10      246.40          91.6
4           54947.53     -175.58          91.6
5           54930.98      -16.54          91.6
6           55047.67      116.69          91.6
7           55076.05       28.38          91.6
8           55074.96       -1.10          91.6
9           54809.42     -265.54          91.6
10          54826.82       17.40          91.6
11          54898.58       71.76          91.6
12          55012.38      113.81          91.6
13          54852.03     -160.35          91.6
14          54899.41       47.38          91.6
15          54832.05      -67.36          91.6
16          55067.24      235.19          91.6
17          54786.86     -280.38          91.6
18          54941.67      154.81          91.6
19          55040.75       99.08          91.6

Shape: (20, 8)
```

3) Perform EDA analysis with minimum 25 keywords by using given dataset and have visualize it.

## Code:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[4]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Use the gold_rates_df created in Question 2
df = gold_rates_df.copy()

# Set up visualization style
sns.set_style("whitegrid")

# --- EDA Analysis (Minimum 25 Keywords) ---

print("### 1. Initial Data Inspection ###") # Keyword: Inspection
print(df.head())                           # Keyword: head, Top_Rows
print(df.tail())                           # Keyword: tail, Bottom_Rows
print(df.info())                           # Keyword: info, Data_Types, Non-Null_Count, Memory_Usage

print("\n### 2. Descriptive Statistics ###") # Keyword: Descriptive_Stats
print(df.describe())                         # Keyword: describe, Central_Tendency, Spread, Quartiles
                                             # Keywords: mean, median (implicit), standard_deviation, minimum, maximum, count, 25th_percentile, 50th_perc

print("\n### 3. Data Quality Checks ###") # Keyword: Quality_Check, Missing_Values
print(df.isnull().sum())                   # Keyword: isnull, Sum, Null_Count, Missing_Data
# df.duplicated().sum()                    # Keyword: Duplicates

print("\n### 4. Categorical Analysis (Location) ###") # Keyword: Categorical_Analysis, Location, Frequency
location_counts = df['Location'].value_counts() # Keyword: Value_Counts
print(location_counts)
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
print("\n### 5. Time Series/Rate Analysis ###")
rate_columns = ['24k_Rate_per_10g', '22k_Rate_per_10g'] # Keyword: Rate_Columns, Time_Series_Data
print(df[rate_columns].corr())                      # Keyword: Correlation, Relationship
print(df['24k_Rate_per_10g'].mean())             # Keyword: Mean_Rate
print(df['24k_Change'].std())                     # Keyword: Volatility (Std Dev)

# --- Visualization ---

plt.figure(figsize=(15, 6))

# Visualization 1: Time Series Plot of Gold Rates plt.subplot(1, 2, 1)
plt.plot(df['Date'], df['24k_Rate_per_10g'], label='24k Rate', marker='o') # Keyword: Line_Plot, Trend
plt.plot(df['Date'], df['22k_Rate_per_10g'], label='22k Rate', marker='x')
plt.xticks(rotation=45)
plt.title('Daily Gold Rate Trend (24k vs 22k)') # Keyword: Trend_Analysis, Daily
plt.xlabel('Date')
plt.ylabel('Rate per 10g')
plt.legend()
plt.tight_layout()

# Visualization 2: Histogram of Daily Rate Changes
plt.subplot(1, 2, 2)
sns.histplot(df['24k_Change'], kde=True, bins=5) # Keyword: Histogram, Distribution
plt.title('Distribution of 24k Daily Rate Change') # Keyword: Distribution_Analysis
plt.xlabel('Daily Change in Rate')
plt.ylabel('Frequency')
plt.tight_layout()

plt.show() # Keyword: Visualization_Output
```

# Output:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
### 1. Initial Data Inspection ###
         Date Location  24k_Rate_per_10g  24k_Change  24k_Purity_%  \
0  2025-11-26   City B          60145.23        0.00          99.9
1  2025-11-27   City A          59941.66     -203.56          99.9
2  2025-11-28   City A          59887.87      -53.80          99.9
3  2025-11-29   City C          60128.15      240.28          99.9
4  2025-11-30   City A          60015.92     -112.23          99.9

   22k_Rate_per_10g  22k_Change  22k_Purity_%
0          55069.39        0.00          91.6
1          54894.41     -174.97          91.6
2          54876.70      -17.71          91.6
3          55123.10      246.40          91.6
4          54947.53     -175.58          91.6
          Date Location  24k_Rate_per_10g  24k_Change  24k_Purity_%  \
15  2025-12-11   City B          59911.18      -65.27          99.9
16  2025-12-12   City C          60107.60      196.43          99.9
17  2025-12-13   City B          59826.21     -281.39          99.9
18  2025-12-14   City A          59957.28      131.07          99.9
19  2025-12-15   City C          60066.60      109.31          99.9

    22k_Rate_per_10g  22k_Change  22k_Purity_%
15          54832.05      -67.36          91.6
16          55067.24      235.19          91.6
17          54786.86     -280.38          91.6
18          54941.67      154.81          91.6
19          55040.75       99.08          91.6
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date              20 non-null     object
 1   Location          20 non-null     object
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Date             20 non-null     object 
 1   Location         20 non-null     object 
 2   24k_Rate_per_10g  20 non-null     float64
 3   24k_Change       20 non-null     float64
 4   24k_Purity_%     20 non-null     float64
 5   22k_Rate_per_10g  20 non-null     float64
 6   22k_Change       20 non-null     float64
 7   22k_Purity_%     20 non-null     float64
dtypes: float64(6), object(2)
memory usage: 1.4+ KB
None

### 2. Descriptive Statistics ###
       24k_Rate_per_10g  24k_Change  24k_Purity_%  22k_Rate_per_10g  \
count         20.000000   20.000000  2.000000e+01         20.000000   
mean       59994.726000   -3.931500  9.990000e+01      54950.400000   
std          106.893617  148.944728  1.458003e-14        105.205124   
min        59826.210000 -297.690000  9.990000e+01      54786.860000   
25%        59909.855000  -81.275000  9.990000e+01      54870.532500   
50%        59966.865000   14.460000  9.990000e+01      54936.325000   
75%        60111.117500   90.485000  9.990000e+01      55052.562500   
max        60154.620000  240.280000  9.990000e+01      55123.100000   

       22k_Change  22k_Purity_%
count   20.000000  2.000000e+01
mean    -1.431500  9.160000e+01
std    149.779892  1.458003e-14
min   -280.380000  9.160000e+01
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help — Trusted

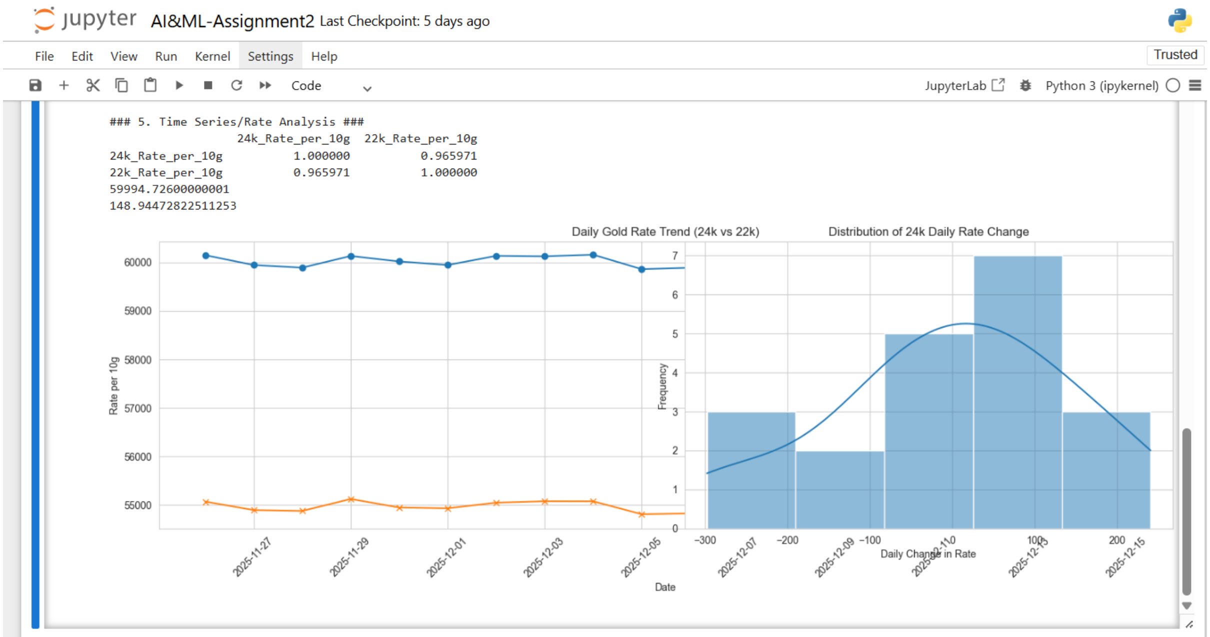
Code — JupyterLab — Python 3 (ipykernel)

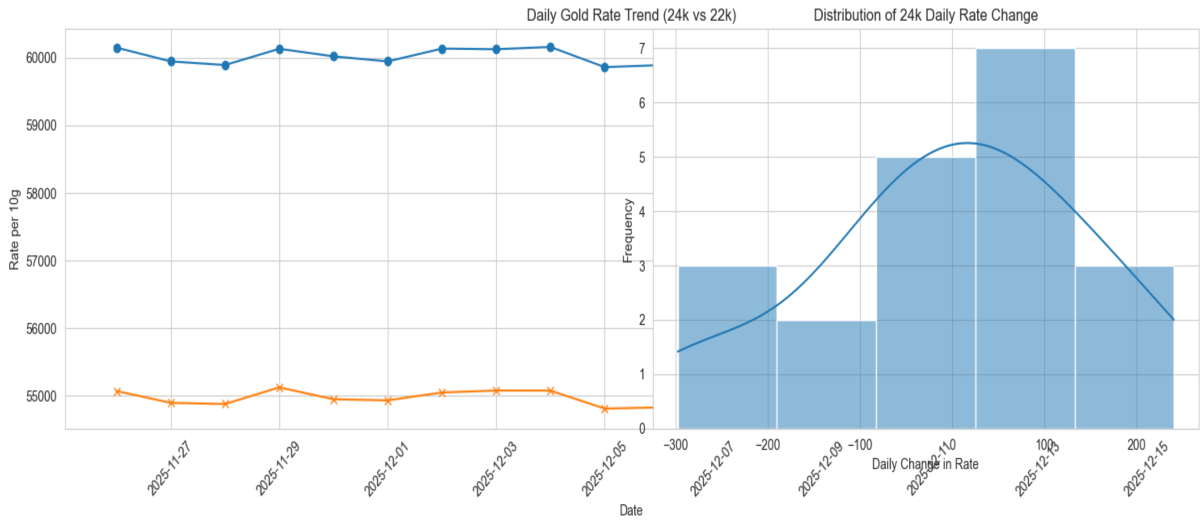
```
       22k_Change  22k_Purity_%
count   20.000000  2.000000e+01
mean    -1.431500  9.160000e+01
std    149.779892  1.458003e-14
min   -280.380000  9.160000e+01
25%    -90.607500  9.160000e+01
50%      8.700000  9.160000e+01
75%    102.762500  9.160000e+01
max    246.400000  9.160000e+01

### 3. Data Quality Checks ###
Date                0
Location            0
24k_Rate_per_10g    0
24k_Change          0
24k_Purity_%        0
22k_Rate_per_10g    0
22k_Change          0
22k_Purity_%        0
dtype: int64

### 4. Categorical Analysis (Location) ###
Location
City A    8
City B    7
City C    5
Name: count, dtype: int64

### 5. Time Series/Rate Analysis ###
                  24k_Rate_per_10g  22k_Rate_per_10g
24k_Rate_per_10g          1.000000          0.965971
22k_Rate_per_10g          0.965971          1.000000
```




4) Create a list of data which consists of length of x and y should be minimum 15 values and calculate statistical parameters like mean, median, mode, variance and standard deviation.

## Code:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[5]:
```python
import numpy as np
from scipy import stats # For Mode calculation

# 1. Create data lists (minimum length 15)
np.random.seed(42) # For reproducibility
x = np.random.randint(10, 50, size=15) # Example: x is a list of 15 integers
y = np.random.uniform(5.0, 10.0, size=20) # Example: y is a list of 20 floats

print(f"List x (Length {len(x)}): {x}")
print(f"List y (Length {len(y)}): {np.round(y, 2)}")
print("-" * 50)

def calculate_stats(data_list, name):
    """Calculates and prints the statistical parameters for a given list."""

    print(f"### Statistical Parameters for List {name} ###")

    # Calculate Parameters
    mean = np.mean(data_list)
    median = np.median(data_list)
    # Mode: scipy.stats.mode returns (mode_value, count)
    mode = stats.mode(data_list, keepdims=True)
    variance = np.var(data_list)
    std_dev = np.std(data_list)

    # Print Results
    print(f"Mean: \t\t{mean:.4f}")
    print(f"Median: \t{median:.4f}")
    # The mode result might be an array if multiple values have the same max frequency
    print(f"Mode: \t\t{mode.mode[0]:.4f} (Count: {mode.count[0]})")
    print(f"Variance: \t{variance:.4f}")
    print(f"Std Dev: \t{std_dev:.4f}")
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
    print(f"### Statistical Parameters for List {name} ###")

    # Calculate Parameters
    mean = np.mean(data_list)
    median = np.median(data_list)
    # Mode: scipy.stats.mode returns (mode_value, count)
    mode = stats.mode(data_list, keepdims=True)
    variance = np.var(data_list)
    std_dev = np.std(data_list)

    # Print Results
    print(f"Mean: \t\t{mean:.4f}")
    print(f"Median: \t{median:.4f}")
    # The mode result might be an array if multiple values have the same max frequency
    print(f"Mode: \t\t{mode.mode[0]:.4f} (Count: {mode.count[0]})")
    print(f"Variance: \t{variance:.4f}")
    print(f"Std Dev: \t{std_dev:.4f}")
```


```python
    print("\n")

# Calculate and print stats for both lists
calculate_stats(x, 'x')
calculate_stats(y, 'y')
```

# Output:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
# Calculate and print stats for both lists
calculate_stats(x, 'x')
calculate_stats(y, 'y')
```


```
List x (Length 15): [48 38 24 17 30 48 28 32 20 20 33 45 49 33 12]
List y (Length 20): [5.1  9.85 9.16 6.06 5.91 5.92 6.52 7.62 7.16 6.46 8.06 5.7  6.46 6.83
 7.28 8.93 6.   7.57 7.96 5.23]
--------------------------------------------------
### Statistical Parameters for List x ###
Mean:           31.8000
Median:         32.0000
Mode:           20.0000 (Count: 2)
Variance:       133.6267
Std Dev:        11.5597


### Statistical Parameters for List y ###
Mean:           6.9891
Median:         6.6765
Mode:           5.1029 (Count: 1)
Variance:       1.6440
Std Dev:        1.2822
```

5) Take a loan related dataset from kaggle website, clean & pre-process the dataset and apply CART technique for the dataset.

## Code:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
[6]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier # Using Classifier for a typical 'loan status' prediction
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder

# --- Hypothetical Data Loading (Replace with actual Kaggle dataset loading) ---

# Assume a loan dataset is loaded with columns:
# Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome,
# CoapplicantIncome, LoanAmount, Credit_History, Property_Area, Loan_Status (Target: Y/N)

# To make the code executable, we'll create a small synthetic DataFrame
data = {
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Male', 'Female'],
    'Married': ['Yes', 'No', 'Yes', 'No', 'Yes', 'Yes'],
    'LoanAmount': [150, 100, 200, 120, 300, 180],
    'Credit_History': [1.0, 1.0, 0.0, 1.0, 1.0, 0.0],
    'Loan_Status': ['Y', 'N', 'N', 'Y', 'Y', 'N']
}
df = pd.DataFrame(data)
# END OF HYPOTHETICAL DATA

print("### Initial Data Sample ###")
print(df.head())
print("-" * 50)

# --- Data Cleaning & Pre-processing ---

# 1. Handle Missing Values (Imputation/Dropping - depends on context)
# For simplicity in this example, we assume no missing values for the selected columns.
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
# --- Data Cleaning & Pre-processing ---

# 1. Handle Missing Values (Imputation/Dropping - depends on context)
# For simplicity in this example, we assume no missing values for the selected columns.
# Real-world step: df.fillna(df.mode().iloc[0], inplace=True)

# 2. Encode Categorical Features
# Use LabelEncoder for binary features and pd.get_dummies for multi-class (if any)
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender']) # Male=1, Female=0
df['Married'] = le.fit_transform(df['Married']) # Yes=1, No=0

# The target variable (Y/N) must also be encoded
df['Loan_Status'] = le.fit_transform(df['Loan_Status']) # Y=1, N=0

# 3. Define Features (X) and Target (y)
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']

print("### Pre-processed Features Sample (X) ###")
print(X.head())
print("-" * 50)

# 4. Split Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# --- Apply CART Technique (Decision Tree Classifier) ---

# 1. Initialize the CART Model
# CART uses 'gini' or 'entropy' criteria for splitting. Max_depth is a hyperparameter.
```

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
# --- Apply CART Technique (Decision Tree Classifier) ---

# 1. Initialize the CART Model
# CART uses 'gini' or 'entropy' criteria for splitting. Max_depth is a hyperparameter.
cart_model = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)

# 2. Train the Model
cart_model.fit(X_train, y_train)

# 3. Make Predictions
y_pred = cart_model.predict(X_test)

# 4. Evaluate the Model
print("### CART Model Performance ###")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred))

# 5. Optional: Visualize the Decision Tree (requires graphviz)
# from sklearn.tree import plot_tree
# plt.figure(figsize=(10, 8))
# plot_tree(cart_model, feature_names=X.columns, class_names=['No Loan', 'Approved'], filled=True)
# plt.show()
```

# Output:

jupyter AI&ML-Assignment2 Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
# plt.figure(figsize=(10, 8))
# plot_tree(cart_model, feature_names=X.columns, class_names=['No Loan', 'Approved'], filled=True)
# plt.show()
```

```
### Initial Data Sample ###
   Gender Married  LoanAmount  Credit_History Loan_Status
0    Male     Yes         150             1.0           Y
1  Female      No         100             1.0           N
2    Male     Yes         200             0.0           N
3  Female      No         120             1.0           Y
4    Male     Yes         300             1.0           Y
--------------------------------------------------
### Pre-processed Features Sample (X) ###
   Gender  Married  LoanAmount  Credit_History
0       1        1         150             1.0
1       0        0         100             1.0
2       1        1         200             0.0
3       0        0         120             1.0
4       1        1         300             1.0
--------------------------------------------------
### CART Model Performance ###
Accuracy: 0.5000
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.50      1.00      0.67         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```