1.Locate open Iris dataset from the URL csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
2.Perform the following operation on dataset
a)Display total no of rows and column
b)Display type of each column
c)Sort the data in descending order , by considering column sepal.length
d)Slice the data: 11 to 20 rows, and only two columns, sepal.length and Species
rename the column Species to Type
Ans.

```
import pandas as pd

# 1. Load the Iris dataset
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

# Define column names (as the dataset has no header)
column_names = ['sepal.length', 'sepal.width', 'petal.length', 'petal.width', 'Species']
df = pd.read_csv(csv_url, header=None, names=column_names)

# 2a. Display total number of rows and columns
print("Shape of the dataset (rows, columns):", df.shape)

# 2b. Display type of each column
print("\nData types of each column:")
print(df.dtypes)

# 2c. Sort the data in descending order by 'sepal.length'
sorted_df = df.sort_values(by='sepal.length', ascending=False)
print("\nTop 5 rows after sorting by 'sepal.length' in descending order:")
print(sorted_df.head())

# 2d. Slice the data: rows 11 to 20 (index 10 to 19), columns sepal.length and Species
sliced_df = df.loc[10:19, ['sepal.length', 'Species']].copy()
sliced_df.rename(columns={'Species': 'Type'}, inplace=True)
print("\nSliced Data (rows 11 to 20 with selected columns):")
print(sliced_df)
```

Consider the given dataset Students Performance Test1
1. Check that is there any missing values in dataframe as a whole
2. is there any missing values across each column
3. count of missing values across each column
4. count row wise missing value

5. count of missing values of a gender column.
6. groupby count of missing values of a column , consider column gender and score
7. replace the missing value of score column with average value of the column
Ans.
import pandas as pd

# Assuming you already have the dataset loaded:
csv_url = 'https://docs.google.com/spreadsheets/d/1qfMIoIbjY66gGdrKIOUCaRyPZ4q8_5K4wwjjceDkBJQ/export?format=csv'
df = pd.read_csv(csv_url)

```
# 1. Check if there are any missing values in the DataFrame
print("Any missing values in dataframe:", df.isnull().values.any())

# 2. Check for missing values across each column
print("\nMissing values in each column:")
print(df.isnull().any())

# 3. Count of missing values across each column
print("\nCount of missing values in each column:")
print(df.isnull().sum())

# 4. Count row-wise missing values
print("\nRow-wise count of missing values:")
print(df.isnull().sum(axis=1))

# 5. Count of missing values in 'gender' column
print("\nMissing values in 'gender' column:",
df['gender'].isnull().sum())

# 6. Groupby count of missing values using 'gender' and 'score'
print("\nGroupby count (including missing values):")
print(df.groupby(['gender', 'score'], dropna=False).size())


# 7. Replace missing values in 'score' column with column's average
average_score = df['score'].mean(skipna=True)
df['score'].fillna(average_score, inplace=True)

print("\nUpdated DataFrame after filling missing scores with average:")
print(df)
```

1. Load the Academic Performance dataset in data frame object.
2. Check null values in the dataset.
3. Check missing values in dataset and replace the null values with standard null value NaN
4. Replace the missing value of Math Score with Mean Value
5. Replace the missing value of Reading Score with standard deviation
6. Replace the missing value of place with common value "Nashik
Ans.

```
import pandas as pd
import numpy as np
# Corrected direct download link from Google Drive
file_id = '1ZZRz4k0KFDt6uZD-EcR62u9UIwDVdvV4'
csv_url = f'https://drive.google.com/uc?export=download&id={file_id}'

# 1. Load the Academic Performance dataset
df = pd.read_csv(csv_url)
Df

# 2. Check for null values in the dataset
print("Null values in dataset:\n", df.isnull())
# 3. Check and replace missing (empty or blank) values with np.nan
df.replace(['', ' ', 'NA', 'na'], np.nan, inplace=True)
# 4. Replace missing values in Math Score with Mean
mean_math = df['Math Score'].astype(float).mean()
df['Math Score'].fillna(mean_math, inplace=True)
Df

# 5. Replace missing values in Reading Score with Standard Deviation
std_reading = df['Reading Score'].astype(float).std()
df['Reading Score'].fillna(std_reading, inplace=True)
Df

# 6. Replace missing values in 'Place' column with common value
"Nashik"
df['Place'].fillna('Nashik', inplace=True)
Df

# Display cleaned DataFrame
print("\nCleaned Dataset:\n", df)
```

1. **Load the MallCustomer dataset in dataframe object df**

2. Display summary statistics (mean, median, minimum, maximum, standard deviation) for
a dataset for each column
3. Display Measures of Dispersion (Mean Absolute Deviation, Variance, Standard Deviation,
Range, Quartiles, Skewness)
4. if your categorical variable is age groupsa nd quantitative variable is income, then
provide summary statistics (minimum and maximum) of income grouped by the age
groups.
Ans.

```
import pandas as pd
import numpy as np
from scipy.stats import skew
csv_url =
'https://drive.google.com/uc?id=1UlVL-OQcGAsYnJq9wuXSvShUPXiat2wM'
df = pd.read_csv(csv_url)
Df

# 2. Display summary statistics (mean, median, min, max, std deviation)
for each column
print("Summary statistics for each column:")
print(df.describe())

# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])
# Mean Absolute Deviation (MAD)
mad = numeric_df.apply(lambda x: np.mean(np.abs(x - x.mean())), axis=0)

print("Mean Absolute Deviation (MAD):\n", mad)
# Variance, Standard Deviation, Range, Quartiles, Skewness
variance = numeric_df.var()
std_dev = numeric_df.std()
data_range = numeric_df.max() - numeric_df.min()
quartiles = numeric_df.quantile([0.25, 0.5, 0.75])
skewness = numeric_df.apply(lambda x: skew(x.dropna()), axis=0)
# Displaying all the calculated measures of dispersion
print("\nMeasures of Dispersion:")
print("Mean Absolute Deviation (MAD):\n", mad)
print("Variance:\n", variance)
print("Standard Deviation:\n", std_dev)
print("Range:\n", data_range)
```

```
print("Quartiles (25%, 50%, 75%):\n", quartiles)
print("Skewness:\n", skewness)

# 4. Provide summary statistics of Income grouped by Age Group
print("\nSummary statistics of Income grouped by Age Group:")
income_grouped = df.groupby('Age Group')['Income'].agg(['min', 'max'])
print(income_grouped)
```

1. Load the demo dataset in dataframe object df
2. Detect the outlier using BoxPlot.
3. Handle the outlier using Quantile based flooring and capping (Hint: the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value)
Ans.

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

csv_url='https://drive.google.com/uc?id=1BaILYt6zWFGGWkKGtinOI4swa-VI_U54'

# 1. Load the demo dataset
df = pd.read_csv(csv_url)
print("Original Dataset:")
print(df.head())

# 2. Detect outliers using boxplot
sns.boxplot(data=df)
plt.title("Boxplot for Outlier Detection")
plt.show()

# 3. Handle outliers using Quantile based flooring and capping
# Apply to all numeric columns
for col in df.select_dtypes(include=[np.number]).columns:
    Q1 = df[col].quantile(0.10)
    Q3 = df[col].quantile(0.90)
    df[col] = np.where(df[col] < Q1, Q1, df[col])
    df[col] = np.where(df[col] > Q3, Q3, df[col])

sns.boxplot(data=df)
plt.title("Boxplot for Outlier Detection")
plt.show()
```

Create a Linear Regression Model using Python to predict home prices using Boston Housing Dataset. The objective is to predict the value of prices of the house using the given features.

Ans.

```
# 1. Import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# 2. Load the Boston Housing Dataset
csv_url =
'https://raw.githubusercontent.com/selva86/datasets/master/BostonHousin
g.csv'
df = pd.read_csv(csv_url)
Df

# 3. Separate features and target variable
X = df.drop('medv', axis=1)  # Features
y = df['medv']    # Target (house price)

# 4. Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 5. Create and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# 6. Make predictions
y_pred = model.predict(X_test)

# 7. Evaluate the model
print("\nModel Evaluation:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R-squared Score:", r2_score(y_test, y_pred))

# 8. Plot Actual vs Predicted values
plt.scatter(y_test, y_pred, edgecolors=(0, 0, 0))
plt.xlabel("Actual")
```

```
plt.ylabel("Predicted")
plt.title("Actual vs Predicted House Prices")
plt.show()
```

1. Load the Academic Performance dataset in data frame object.
2. Check null values in the dataset.
3. Count the number of null values in complete data set (Hint: eplace the null values with standard null value NaN).
4. Dropping rows with at least 1 null value.
5. Dropping rows if all values in that row are missing
6. Dropping columns with at least 1 null value.
7. Dropping Rows with at least 1 null value in CSV file.
Ans.

```
import pandas as pd
import numpy as np

# 1. Load the Academic Performance dataset
url =
'https://drive.google.com/uc?id=1Z_k34IEgYjSAs7Wsr88nJkcpqGX7RNsK'
df = pd.read_csv(url)
Df

# 2. Check for null values in the dataset
print("Null Values in Each Column:\n", df.isnull().sum())

# 3. Count total null values in the dataset
total_nulls = df.isnull().sum().sum()
print("\nTotal Null Values in the Dataset:", total_nulls)

# 4. Dropping rows with at least 1 null value
df_dropped_any = df.dropna()
print("\nData After Dropping Rows with Any Null Value:\n",
df_dropped_any)

# 5. Dropping rows where all values are missing
df_dropped_all = df.dropna(how='all')
print("\nData After Dropping Rows where All Values are Missing:\n",
df_dropped_all)

# 6. Dropping columns with at least 1 null value
df_dropped_cols = df.dropna(axis=1)
print("\nData After Dropping Columns with Any Null Value:\n",
df_dropped_cols)
```

```
# 7. Dropping rows with at least 1 null value and saving to new CSV
df_cleaned = df.dropna()
df_cleaned.to_csv("AcademicPerformance_Cleaned.csv", index=False)
print("\nCleaned CSV saved as 'AcademicPerformance_Cleaned.csv'")
```

1. Load the demo dataset in dataframe object df
2. Detect the outlier using Z-score
3. replace the outliers with the median value.
Ans.

```
import pandas as pd
import numpy as np
from scipy import stats

# Sample demo dataset creation
np.random.seed(0)
data = {
    'Age': np.random.randint(20, 40, 50),
    'Income': np.random.normal(50000, 10000, 50),
    'Score': np.random.normal(75, 10, 50)
}
# Introduce some outliers
df.loc[2, 'Income'] = 150000
df.loc[10, 'Score'] = 130
df
# 2. Detect outliers using Z-score
z_scores = np.abs(stats.zscore(df))
outliers = (z_scores > 3)

# 3. Replace outliers with the median value
for col in df.columns:
    median_val = df[col].median()
    df.loc[outliers[:, df.columns.get_loc(col)], col] = median_val
print("\nData after replacing outliers with median:\n", df)

# Save to CSV
df.to_csv("demo_outliers.csv", index=False)
```

OR
1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv
dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on

the given dataset.

Ans.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score

# Load iris dataset
df =
pd.read_csv('https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pa
ges/data/iris.csv')

# Prepare data
X = df.drop('species', axis=1)
y = df['species']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)

# Naïve Bayes classifier
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=model.classes_)
print("Confusion Matrix:\n", cm)

# Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
error_rate = 1 - accuracy

print(f"\nAccuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

1. Load the demo dataset in dataframe object df

2. Detect the outlier using Inter Quantile Range(IQR)

3. remove the outliers from the dataset.

Ans.

```
import pandas as pd
```

```
import numpy as np

# Step 1: Create a demo dataset
np.random.seed(42)
df = pd.DataFrame({
    'Age': np.random.randint(20, 50, 100),
    'Income': np.random.normal(50000, 12000, 100),
    'Score': np.random.normal(70, 10, 100)
})

# Add outliers
df.loc[5, 'Income'] = 150000
df.loc[15, 'Score'] = 130

# Step 2: Detect outliers using IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

# Step 3: Remove outliers
df_clean = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 *
IQR))).any(axis=1)]

print("Original Data Size:", df.shape[0])
print("Cleaned Data Size:", df_clean.shape[0])
```

OR

Write a Scala program to find a number is zero, positive or negative and write a Scala program to print your name.

Ans.

1. Load the demo dataset in dataframe object df
2. Detect the outlier using ScatterPlot
3. Handle the outlier using Quantile based flooring and capping (Hint: the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value)

Ans.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# 1. Load the demo dataset (create a simple one for illustration)
data = {
```

```
    'Student_ID': range(1, 21),
    'Math_Score': [65, 67, 70, 69, 66, 72, 68, 74, 71, 69, 67, 100, 64,
62, 61, 60, 78, 79, 80, 150]  # Outlier at 150
}
df = pd.DataFrame(data)

print("Original Dataset:")
print(df)

# 2. Detect outlier using scatter plot
sns.scatterplot(x='Student_ID', y='Math_Score', data=df)
plt.title("Scatter Plot to Detect Outliers")
plt.xlabel("Student ID")
plt.ylabel("Math Score")
plt.grid(True)
plt.show()

# 3. Handle outlier using Quantile based flooring and capping
q_low = df['Math_Score'].quantile(0.10)
q_high = df['Math_Score'].quantile(0.90)

df['Math_Score_Capped'] = np.where(df['Math_Score'] < q_low, q_low,
                            np.where(df['Math_Score'] > q_high, q_high,
df['Math_Score']))

print("\nDataset after flooring and capping outliers:")
print(df)

# Plot after capping
sns.scatterplot(x='Student_ID', y='Math_Score_Capped', data=df)
plt.title("Scatter Plot After Outlier Handling")
plt.xlabel("Student ID")
plt.ylabel("Capped Math Score")
plt.grid(True)
plt.show()
```

1. Implement logistic regression using Python to perform classification on Social_Network_Ads.csv dataset

1. Implement logistic regression using Python to perform classification on Social_Network_Ads.csv dataset. 2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Ans.

```
# Step 1: Import libraries
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Step 2: Load the dataset
url =
'https://drive.google.com/uc?export=download&id=1krI1VYrEaStiLRRbuXBnwH
Rn8b5sAeiD'
df = pd.read_csv(url)
Df

# Step 3: Select features and target
X = df[['Age', 'EstimatedSalary']]
y = df['Purchased']

# Step 4: Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)


# Step 5: Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 6: Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 7: Predictions
y_pred = model.predict(X_test)

# Step 8: Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
```

```
# Step 9: Compute TP, FP, TN, FN, Accuracy, Error Rate, Precision,
Recall
cm = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = cm.ravel()

print(f"True Positives (TP): {TP}")
print(f"False Positives (FP): {FP}")
print(f"True Negatives (TN): {TN}")
print(f"False Negatives (FN): {FN}")

accuracy = (TP + TN) / (TP + TN + FP + FN)
error_rate = 1 - accuracy
precision = TP / (TP + FP) if (TP + FP) != 0 else 0
recall = TP / (TP + FN) if (TP + FN) != 0 else 0

print(f"\nAccuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency
Ans.

Use the inbuilt dataset 'titanic'. Use the Seaborn library to see if we can find any patterns in
the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.
Ans.

```
# Step 1: Import required libraries
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Step 2: Load the Titanic dataset
df = sns.load_dataset('titanic')

# Step 3: Display the first few rows
print("First 5 records:\n", df.head())
```

```
# Step 4: Summary of the dataset
print("\nDataset Info:")
df.info()

# Step 5: Check for null values
print("\nMissing values:\n", df.isnull().sum())

# Step 6: Plot histogram for 'fare' column
sns.histplot(df['fare'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Fare')
plt.xlabel('Fare')
plt.ylabel('Number of Passengers')
plt.grid(True)
plt.show()
```

Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age') 2. Write observations on the inference from the above statistics.

Ans.

```
# Step 1: Import required libraries
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Step 2: Load the Titanic dataset
df = sns.load_dataset('titanic')

# Step 3: Drop rows where 'age' is missing, since we can't plot without
it
df = df.dropna(subset=['age'])

# Step 4: Convert 'survived' to categorical for better labels
df['survived'] = df['survived'].map({0: 'Not Survived', 1: 'Survived'})

# Step 5: Create a box plot
sns.boxplot(data=df, x='sex', y='age', hue='survived', palette='Set2')
plt.title('Age Distribution by Gender and Survival Status')
plt.xlabel('Sex')
plt.ylabel('Age')
plt.grid(True)
plt.show()
```

Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:
1. List down the features and their types (e.g., numeric, nominal) available in the dataset.
2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
3. Create a box plot for each feature in the dataset.
4. Compare distributions and identify outliers.
Ans.

```
# Step 1: Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load the Iris dataset from Seaborn
df = sns.load_dataset('iris')



# Step 3: Display basic info
print("Dataset Info:\n")
print(df.info())
print("\nFirst 5 rows of the dataset:\n")
print(df.head())

# Step 4: Histogram for each numeric feature
df.hist(figsize=(10, 8), edgecolor='black')
plt.suptitle("Histograms of Iris Features", fontsize=16)
plt.tight_layout()
plt.show()

# Step 5: Boxplot for each feature grouped by species
numeric_features = df.select_dtypes(include='float')
plt.figure(figsize=(12, 8))

for i, feature in enumerate(numeric_features.columns, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='species', y=feature, data=df, palette="Set3")
    plt.title(f'Box Plot of {feature} by Species')

plt.tight_layout()
plt.show()
```


All
THE
Best