DISCRETIZATION

Instructions:

Please share your answers filled inline in the word document. Submit Python code and R code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

Everything will revolve around the data in Analytics world. Proper data will help you to make useful predictions that improve your business. Sometimes the usage of original data as it is does not help to have accurate solutions. It is needed to convert the data from one form to another form to have better predictions. Explore various techniques to transform the data for better model performance. you can go through this link:

https://360digitmg.com/mindmap-data-science

1) Convert the continuous data into discrete classes on the iris dataset.

Prepare the dataset by performing the preprocessing techniques, to have the data which improves model performance.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

```
import pandas as pd

import numpy as np

# Load the dataset

from sklearn.datasets import load_iris

iris = load_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

df['species'] = iris.target

# Define bin edges and labels

def discretize_feature(df, feature_name, bins, labels):

    df[feature_name] = pd.cut(df[feature_name], bins=bins, labels=labels, include_lowest=True)

# Discretizing Sepal Length

df['sepal length (cm)'] = pd.cut(df['sepal length (cm)'], bins=[4, 5, 6, 7], labels=['Short', 'Medium', 'Long'])

# Discretizing Sepal Width

df['sepal width (cm)'] = pd.cut(df['sepal width (cm)'], bins=[2, 3, 4, 5], labels=['Narrow', 'Medium', 'Wide'])

# Discretizing Petal Length

df['petal length (cm)'] = pd.cut(df['petal length (cm)'], bins=[0, 2, 5, 7], labels=['Short', 'Medium', 'Long'])

# Discretizing Petal Width

df['petal width (cm)'] = pd.cut(df['petal width (cm)'], bins=[0, 1, 2, 3], labels=['Narrow', 'Medium', 'Wide'])

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

# Encode categorical features

df_encoded = pd.get_dummies(df, columns=['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'])

# Separate features and target

X = df_encoded.drop('species', axis=1)

y = df_encoded['species']

# Normalize features

scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

Dummy Variables

Instructions:

Please share your answers filled inline in the word document. Submit code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

Data is one of the most important assets. It is often common that data is stored in distinct systems with different formats and forms. Non-numeric form of data makes it tricky while developing mathematical equations for prediction models. We have the preprocessing techniques to make the data convert to numeric form. Explore the various techniques to have reliable uniform standard data, you can go through this link:

https://360digitmg.com/mindmap-data-science

1) Prepare the dataset by performing the preprocessing techniques, to have all the features in numeric format.

Index	Animals	Gender	Homly	Types
1	Cat	Male	Yes	A
2	Dog	Male	Yes	B
3	Mouse	Male	Yes	C
4	Mouse	Male	Yes	C
5	Dog	Female	Yes	A
6	Cat	Female	Yes	B
7	Lion	Female	Yes	D
8	Goat	Female	Yes	E
9	Cat	Female	Yes	A
10	Dog	Male	Yes	B

Hints:

```
import pandas as pd
df = pd.read_csv(r"Animal_category.csv")
mapping = {
    'Animals': {'Cat': 1, 'Dog': 2, 'Mouse': 3, 'Lion': 4, 'Goat': 5},
    'Gender': {'Male': 0, 'Female': 1},
    'Homly': {'Yes': 1, 'No': 0},
    'Types': {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5}
}
df.replace(mapping, inplace=True)
df.head(30)
```

	Index	Animals	Gender	Homly	Types
0	1	1	0	1	1
1	2	2	0	1	2
2	3	3	0	1	3
3	4	3	0	1	3
4	5	2	1	1	1
5	6	1	1	1	2
6	7	4	1	1	4
7	8	5	1	1	5
8	9	1	1	1	1
9	10	2	0	1	2
10	11	2	0	1	2
11	12	4	0	0	4
12	13	4	0	0	4
13	14	4	0	0	4
14	15	1	1	1	1
15	16	4	1	0	4
16	17	4	1	0	4
17	18	1	1	1	1
18	19	5	0	0	5
19	20	5	1	0	5
20	21	5	0	0	5
21	22	5	1	0	5
22	23	4	0	0	4
23	24	4	1	0	4
24	25	4	0	0	4
25	26	4	1	0	4
26	27	2	0	1	2
27	28	4	1	0	4
28	29	1	0	1	1
29	30	4	1	0	4

Duplication & Typecasting

Instructions:

Please share your answers filled inline in the word document. Submit Python code and R code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Preliminaries for Data Analysis

Problem statement:

Data collected may have duplicate entries, that might be because the data collected were not at regular intervals or for any other reason. Building a proper solution on such data will be a tough ask. The common techniques are either removing duplicates completely or substituting those values with logical data. There are various techniques to treat these types of problems.

Q1. For the given dataset perform the type casting (convert the datatypes, ex. float to int)

```
import pandas as pd

df = pd.read_csv(r"Online Retail.csv", encoding='unicode_escape')

df['UnitPrice'] = df['UnitPrice'].astype(float)

print(df.dtypes)
```

```
In [3]: print(df.dtypes)
InvoiceNo          object
StockCode          object
Description        object
Quantity            int64
InvoiceDate        object
UnitPrice         float64
CustomerID        float64
Country            object
dtype: object
```

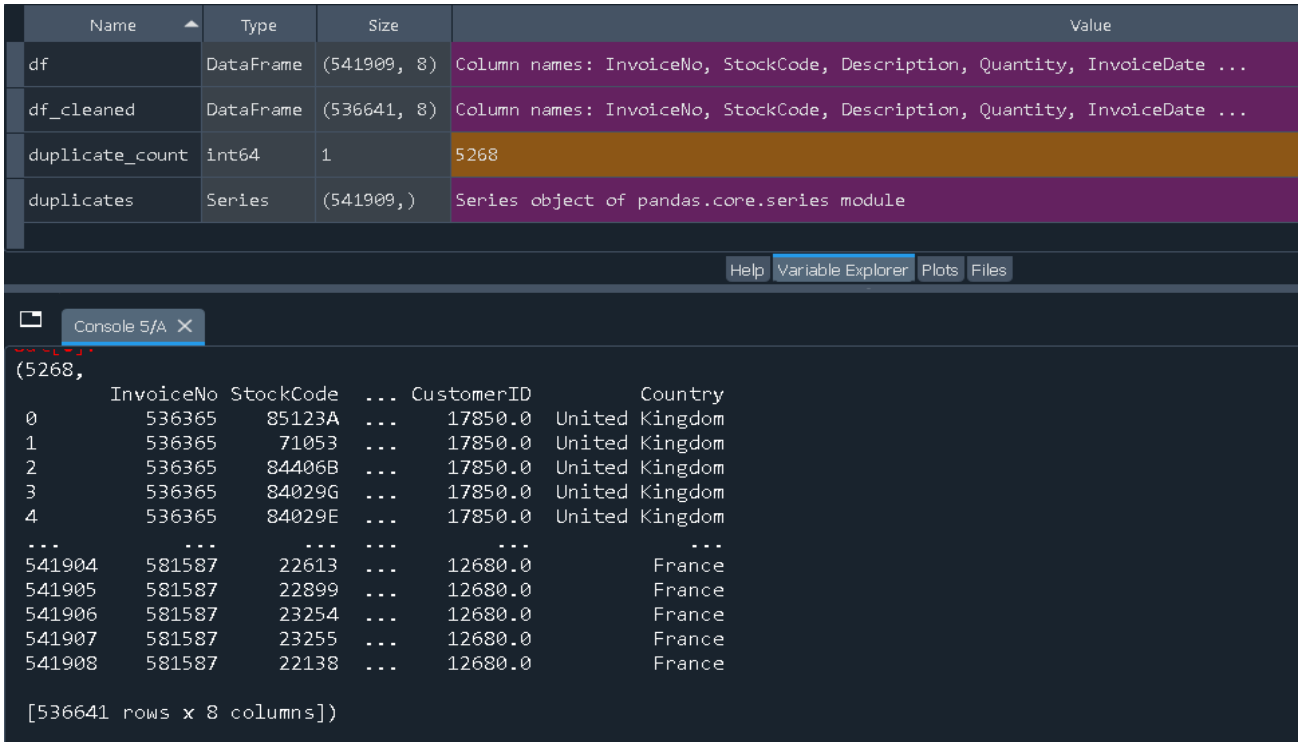
Q2. Check for duplicate values, and handle the duplicate values (ex. drop)

```
import pandas as pd

df = pd.read_csv(r"Online Retail.csv",
                 encoding='unicode_escape')

duplicates = df.duplicated()
```

duplicate_count = duplicates.sum()

df_cleaned = df.drop_duplicates()

duplicate_count, df_cleaned



Name	Type	Size	Value
df	DataFrame	(541909, 8)	Column names: InvoiceNo, StockCode, Description, Quantity, InvoiceDate ...
df_cleaned	DataFrame	(536641, 8)	Column names: InvoiceNo, StockCode, Description, Quantity, InvoiceDate ...
duplicate_count	int64	1	5268
duplicates	Series	(541909,)	Series object of pandas.core.series module

Help | Variable Explorer | Plots | Files

Console 5/A

```
(5268,
        InvoiceNo StockCode  ... CustomerID         Country
 0         536365    85123A  ...    17850.0  United Kingdom
 1         536365     71053  ...    17850.0  United Kingdom
 2         536365    84406B  ...    17850.0  United Kingdom
 3         536365    84029G  ...    17850.0  United Kingdom
 4         536365    84029E  ...    17850.0  United Kingdom
 ...          ...       ...  ...        ...             ...
 541904    581587     22613  ...    12680.0          France
 541905    581587     22899  ...    12680.0          France
 541906    581587     23254  ...    12680.0          France
 541907    581587     23255  ...    12680.0          France
 541908    581587     22138  ...    12680.0          France

 [536641 rows x 8 columns])
```

Q3. Do the data analysis (EDA)?

Such as histogram, boxplot, scatterplot, etc.

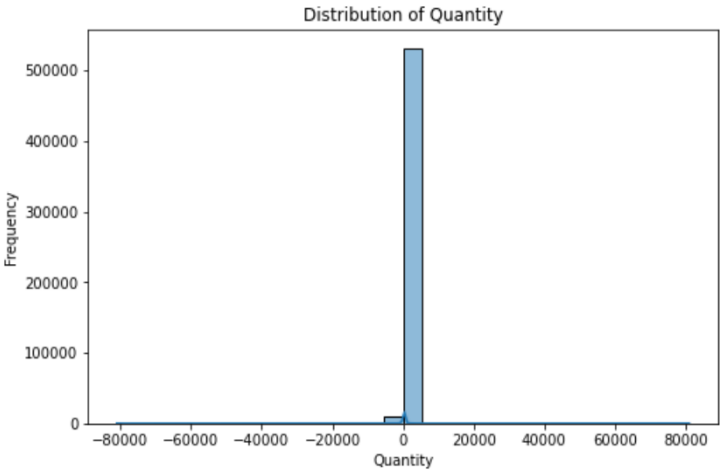
import pandas as pd

import matplotlib.pyplot as plt

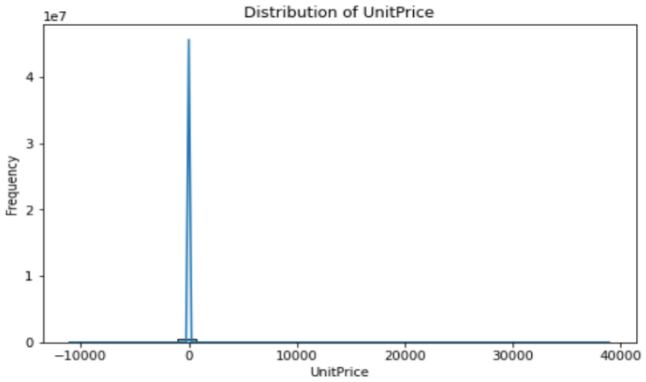
import seaborn as sns

df = pd.read_csv(r"Online Retail.csv",encoding='unicode_escape')

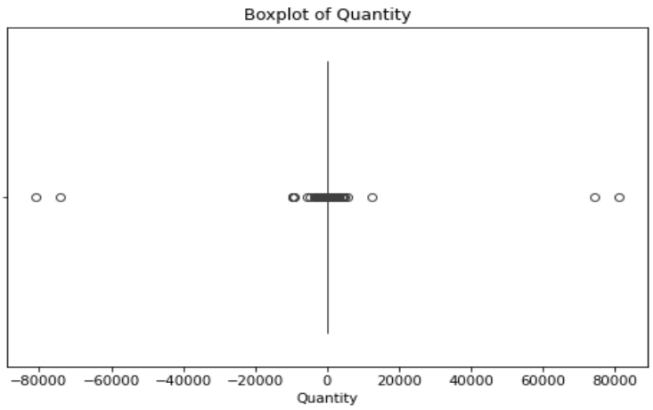
```
# Histogram for Quantity
plt.figure(figsize=(8, 5))
sns.histplot(df['Quantity'], bins=30, kde=True)
plt.title('Distribution of Quantity')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.show()
```



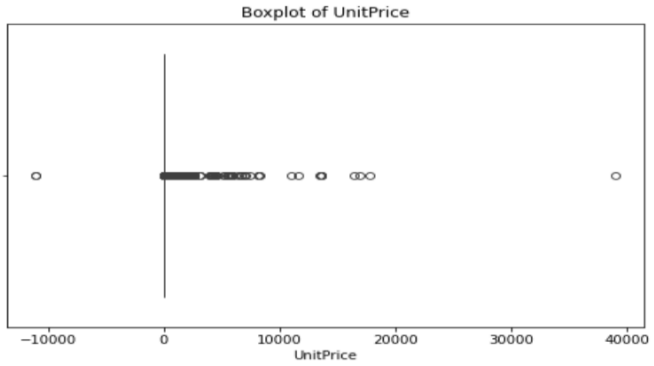

```
# Histogram for UnitPrice
plt.figure(figsize=(8, 5))
sns.histplot(df['UnitPrice'], bins=30, kde=True)
plt.title('Distribution of UnitPrice')
plt.xlabel('UnitPrice')
plt.ylabel('Frequency')
plt.show()
```



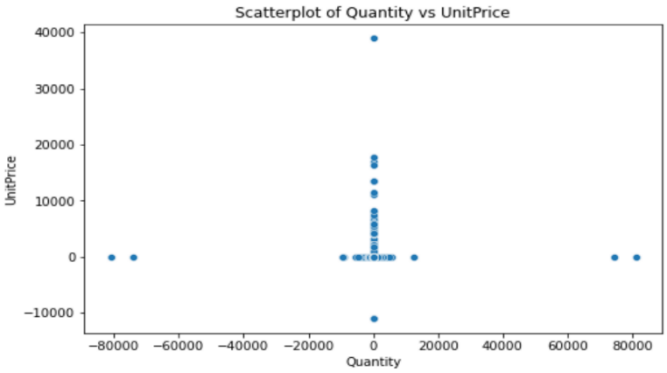

```
# Boxplot for Quantity
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['Quantity'])
plt.title('Boxplot of Quantity')
plt.show()
```



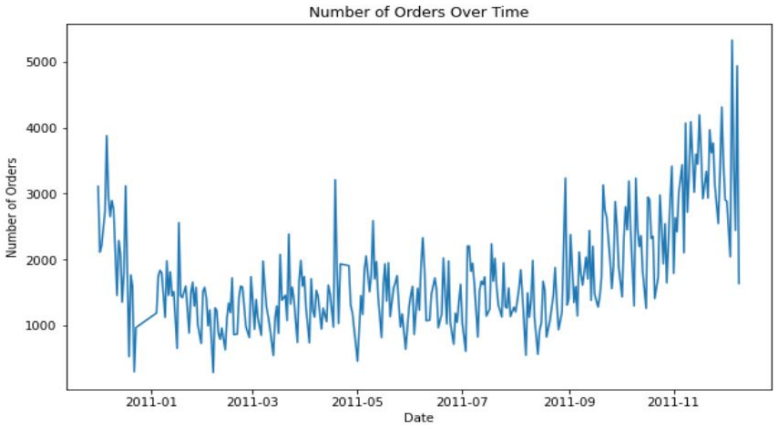
```
# Boxplot for UnitPrice
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['UnitPrice'])
plt.title('Boxplot of UnitPrice')
plt.show()
```




```
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Quantity', y='UnitPrice', data=df)
plt.title('Scatterplot of Quantity vs UnitPrice')
plt.xlabel('Quantity')
plt.ylabel('UnitPrice')
plt.show()
```




```
# Convert InvoiceDate to datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
# Plotting the number of orders over time
plt.figure(figsize=(10, 6))
df['InvoiceDate'].groupby(df['InvoiceDate'].
dt.date).count().plot()
plt.title('Number of Orders Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Orders')
plt.show()
```



Imputation

Instructions:

Please share your answers filled inline in the word document. Submit code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

Majority of the datasets have missing values, that might be because the data collected were not at regular intervals or the breakdown of instruments and so on. It is nearly impossible to build the proper model or in other words, get accurate results. The common techniques are either removing those records completely or substitute those missing values with the logical ones, there are various techniques to treat these types of problems.

1) Prepare the dataset using various techniques to solve the problem, explore all the techniques available and use them to see which gives the best result.

Hint: Go through this link: https://360digitmg.com/mindmap-data-science

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.impute import KNNImputer
df =pd.read_csv(r"claimants.csv")
# 1. Checking for missing values
missing_values = df.isnull().sum()
# 2. Removing rows with missing data
df_dropped = df.dropna()
# 3. Mean Imputation for missing values
mean_imputer = SimpleImputer(strategy='mean')
df_mean_imputed = pd.DataFrame(mean_imputer.fit_transform(df), columns=df.columns)
```

```
# 4. Mode Imputation
mode_imputer = SimpleImputer(strategy='most_frequent')
df_mode_imputed = pd.DataFrame(mode_imputer.fit_transform(df), columns=df.columns)
# 5. Forward Fill/Backward Fill
df_ffill = df.fillna(method='ffill')
df_bfill = df.fillna(method='bfill')
# 6. K-Nearest Neighbors (KNN) Imputation
knn_imputer = KNNImputer(n_neighbors=5)
df_knn_imputed = pd.DataFrame(knn_imputer.fit_transform(df), columns=df.columns)
# 7. Interpolation
df_interpolated = df.interpolate()
# Output the missing values count and imputed datasets
missing_values, df_dropped, df_mean_imputed, df_mode_imputed, df_ffill, df_bfill,
df_knn_imputed, df_interpolated
```

Name	Type	Size	Value
df_ffill	DataFrame	(1340, 7)	Column names: CASENUM, ATTORNEY, CLMSEX, CLMINSUR, SEATBELT, CLMAGE, L ...
df_interpolated	DataFrame	(1340, 7)	Column names: CASENUM, ATTORNEY, CLMSEX, CLMINSUR, SEATBELT, CLMAGE, L ...
df_knn_imputed	DataFrame	(1340, 7)	Column names: CASENUM, ATTORNEY, CLMSEX, CLMINSUR, SEATBELT, CLMAGE, L ...
df_mean_imputed	DataFrame	(1340, 7)	Column names: CASENUM, ATTORNEY, CLMSEX, CLMINSUR, SEATBELT, CLMAGE, L ...
df_mode_imputed	DataFrame	(1340, 7)	Column names: CASENUM, ATTORNEY, CLMSEX, CLMINSUR, SEATBELT, CLMAGE, L ...
knn_imputer	impute._knn.KNNImputer	1	KNNImputer object of sklearn.impute._knn module
mean_imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
missing_values	Series	(7,)	Series object of pandas.core.series module
mode_imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module

Help | Variable Explorer | Plots | Files

Console 7/A

```
[1340 rows x 7 columns],
      CASENUM  ATTORNEY  CLMSEX  CLMINSUR  SEATBELT  CLMAGE    LOSS
0           5         0     0.0       1.0       0.0    50.0  34.940
1           3         1     1.0       0.0       0.0    18.0   0.891
2          66         1     0.0       1.0       0.0     5.0   0.330
3          70         0     0.0       1.0       1.0    31.0   0.037
4          96         1     0.0       1.0       0.0    30.0   0.038
...       ...       ...     ...       ...       ...     ...     ...
1335    34100         1     0.0       1.0       0.0    31.0   0.576
1336    34110         0     1.0       1.0       0.0    46.0   3.705
1337    34113         1     1.0       1.0       0.0    39.0   0.099
1338    34145         0     1.0       0.0       0.0     8.0   3.177
1339    34153         1     1.0       1.0       0.0    30.0   0.688

[1340 rows x 7 columns])
```

Outlier Treatments

Instructions:

Please share your answers filled inline in the word document. Submit code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

Most of the datasets have extreme values or exceptions in their observations. These values affect the predictions (Accuracy) of the model in one way or the other, removing these values is not a very good option. For these types of scenarios, we have various techniques to treat such values.

1. Prepare the dataset by performing the preprocessing techniques, to treat the outliers.

```
import pandas as pd

import numpy as np


data = pd.read_csv(r"Boston.csv")


# Treating outliers with Winsorization

from scipy.stats.mstats import winsorize


# Apply Winsorization to 'crim' as an example

data['crim'] = winsorize(data['crim'], limits=[0.05, 0.05])


# Example of a log transformation on 'tax'

data['tax'] = np.log(data['tax'])


# Capping 'ptratio' at 1st and 99th percentiles

lower_bound = data['ptratio'].quantile(0.01)

upper_bound = data['ptratio'].quantile(0.99)
```

data['ptratio'] = np.clip(data['ptratio'], lower_bound, upper_bound)

Check the transformed dataset

print(data)

```
In [10]: print(data)
         crim    zn  indus  chas    nox  ...       tax  ptratio   black  lstat  medv
0     0.15876   0.0  10.81   0.0  0.413  ...  5.720312     19.2  376.94   9.88  21.7
1     0.10328  25.0   5.13   0.0  0.453  ...  5.648974     19.7  396.90   9.22  19.6
2     0.34940   0.0   9.90   0.0  0.544  ...  5.717028     18.4  396.24   9.97  20.3
3     2.73397   0.0  19.58   0.0  0.871  ...  5.998937     14.7  351.85  21.45  15.4
4     0.04337  21.0   5.64   0.0  0.439  ...  5.493061     16.8  393.97   9.43  20.5
..        ...   ...    ...   ...    ...  ...       ...      ...     ...    ...   ...
399   9.32909   0.0  18.10   0.0  0.713  ...  6.501290     20.2  396.90  18.13  14.1
400  15.57570   0.0  18.10   0.0  0.597  ...  6.501290     20.2    2.60  10.11  15.0
401   0.02875  90.0   1.21   1.0  0.401  ...  5.288267     13.6  395.52   3.16  50.0
402   0.02875  85.0   0.74   0.0  0.410  ...  5.746203     17.3  396.90   5.77  24.7
403   0.08244  30.0   4.93   0.0  0.428  ...  5.703782     16.6  379.41   6.36  23.7

[404 rows x 14 columns]
```

STANDARDIZATION & NORMALIZATION

Instructions:

Please share your answers filled inline in the word document. Submit Python code and R code files wherever applicable.

Please ensure you update all the details:

Name:ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

Data is one of the most important assets. Often the data are stored in distinct systems with different formats and scales. These seemingly small differences in how the data is stored can result in misinterpretations and inconsistencies in your analytics. Inconsistency can make it impossible to deliver reliable information to management for good decision-making. We have the preprocessing techniques to make the data uniform. To explore the various techniques to have reliable uniform standard data, you can go through this link:

https://360digitmg.com/mindmap-data-science

1) Prepare the dataset by performing the preprocessing techniques, to have the standard scale to data.

```
import pandas as pd

from sklearn.preprocessing import MinMaxScaler, StandardScaler

from sklearn.impute import SimpleImputer


# Assuming the dataset is already loaded in df

df = pd.read_csv(r"Seeds_data.csv")  # Replace with your actual dataset file


# 1. Identify and Handle Missing Data (if any)

imputer = SimpleImputer(strategy='mean')

df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)

# 2. Normalization (Min-Max Scaling)

min_max_scaler = MinMaxScaler()
```

```
df_normalized = pd.DataFrame(min_max_scaler.fit_transform(df_imputed),
columns=df.columns)

# 3. Standardization (Z-Score Normalization)

standard_scaler = StandardScaler()

df_standardized = pd.DataFrame(standard_scaler.fit_transform(df_imputed),
columns=df.columns)

# Output the preprocessed datasets

df_imputed, df_normalized, df_standardized
```

Name	Type	Size	Value
df	DataFrame	(210, 8)	Column names: Area, Perimeter , Compactness, length, Width, Assymetry_ ...
df_imputed	DataFrame	(210, 8)	Column names: Area, Perimeter , Compactness, length, Width, Assymetry_ ...
df_normalized	DataFrame	(210, 8)	Column names: Area, Perimeter , Compactness, length, Width, Assymetry_ ...
df_standardized	DataFrame	(210, 8)	Column names: Area, Perimeter , Compactness, length, Width, Assymetry_ ...
imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
min_max_scaler	preprocessing._data.MinMaxScaler	1	MinMaxScaler object of sklearn.preprocessing._data module
standard_scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module

Help | Variable Explorer | Plots | Files

Console 8/A

```
In [6]: df_imputed, df_normalized, df_standardized
Out[6]:
(      Area  Perimeter   Compactness  ...  Assymetry_coeff  len_ker_grove  Type
 0    15.26       14.84       0.8710  ...            2.221          5.220   1.0
 1    14.88       14.57       0.8811  ...            1.018          4.956   1.0
 2    14.29       14.09       0.9050  ...            2.699          4.825   1.0
 3    13.84       13.94       0.8955  ...            2.259          4.805   1.0
 4    16.14       14.99       0.9034  ...            1.355          5.175   1.0
 ..     ...         ...          ...  ...              ...            ...   ...
 205  12.19       13.20       0.8783  ...            3.631          4.870   3.0
 206  11.23       12.88       0.8511  ...            4.325          5.003   3.0
 207  13.20       13.66       0.8883  ...            8.315          5.056   3.0
 208  11.84       13.21       0.8521  ...            3.598          5.044   3.0
 209  12.30       13.34       0.8684  ...            5.637          5.063   3.0

 [210 rows x 8 columns],
         Area  Perimeter   Compactness  ...  Assymetry_coeff  len_ker_grove  Type
 0    0.440982    0.502066     0.570780  ...         0.189302       0.345150   0.0
 1    0.405099    0.446281     0.662432  ...         0.032883       0.215165   0.0
 2    0.349386    0.347107     0.879310  ...         0.251453       0.150665   0.0
```

STRING MANIPULATIONS

Instructions:

Please share your answers filled inline in the word document. Submit code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

It is obvious that as part of data analysis, we encounter a lot of text data which is a collection of strings that in turn is a sequence of characters. Access the text data and manipulate it as per our requirements. you can go through this link for further assistance:

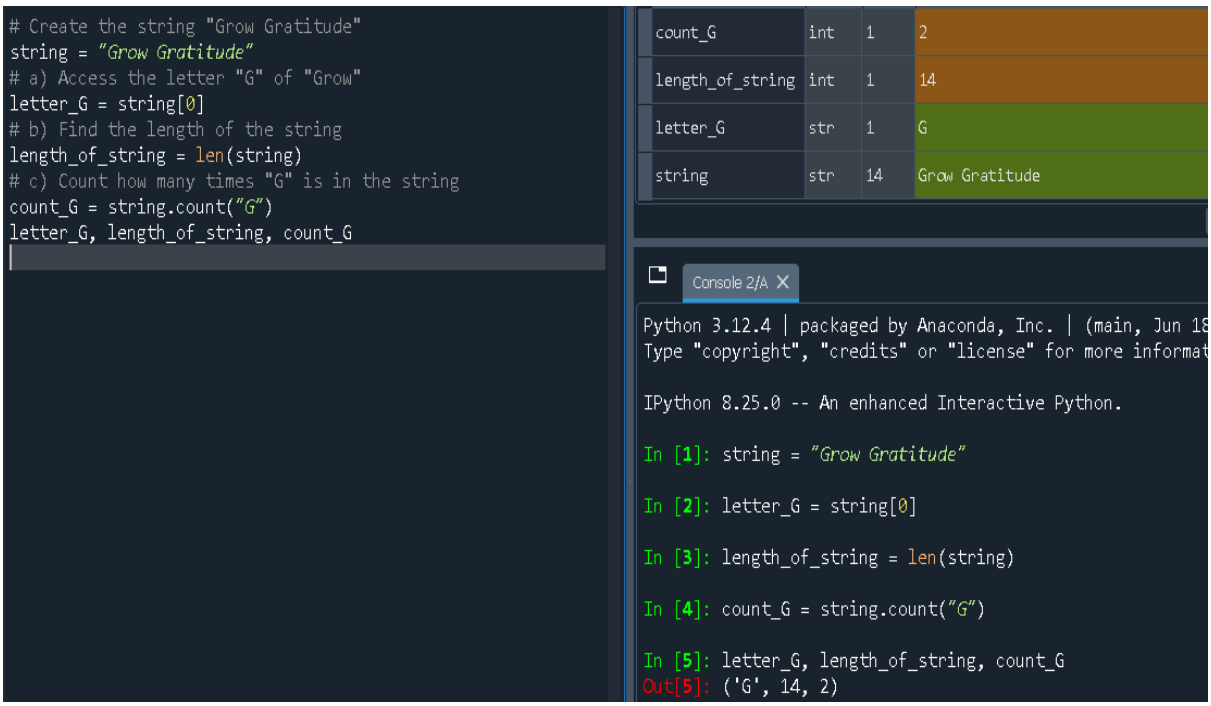
1. Create a string “Grow Gratitude”.

Code for the following tasks:

a) How do you access the letter “G” of “Growth”?

b) How do you find the length of the string?

c) Count how many times “G” is in the string?



2. Create a string “Being aware of a single shortcoming within yourself is far more useful than being aware of a thousand in someone else.”

Code for the following:

a) Count the number of characters in the string.

Create the string

quote = "Being aware of a single shortcoming within yourself is far more useful than being aware of a thousand in someone else."

a) Count the number of characters in the string

number_of_characters = len(quote)

number_of_characters

output:-

118

3. Create a string "Idealistic as it may sound, altruism should be the driving force in business, not just competition and a desire for wealth"

Code for the following tasks:

a) get one char of the word

b) get the first three char

c) get the last three char

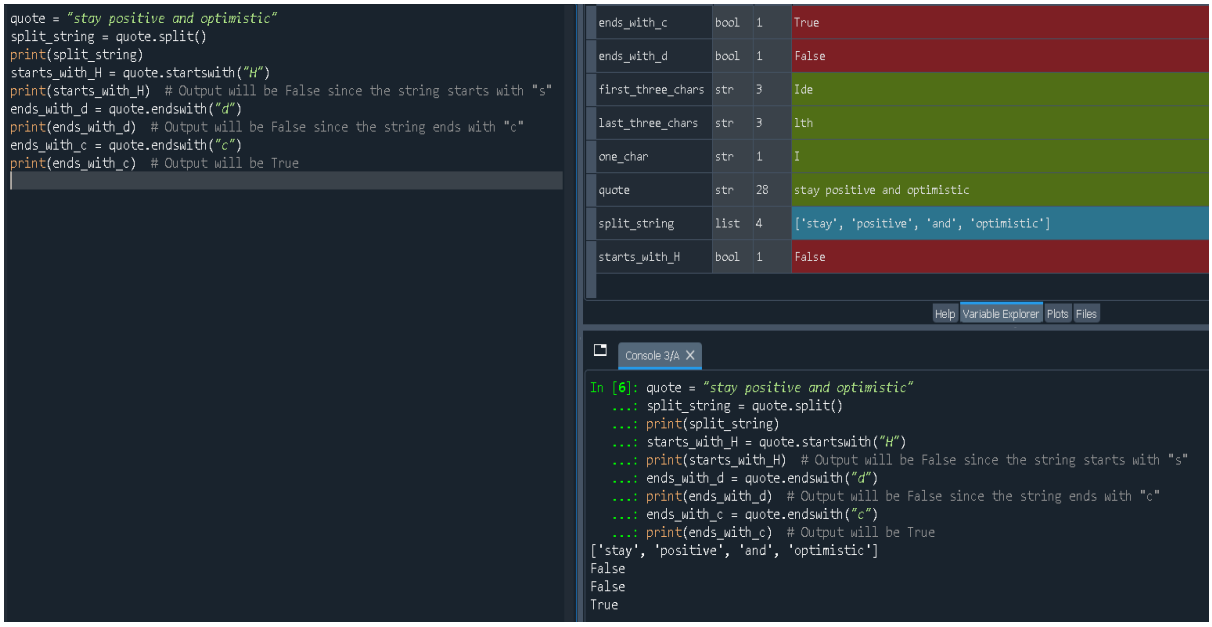
4. create a string "stay positive and optimistic". Now write a code to split on whitespace.

Write a code to find if:

a) The string starts with “H”

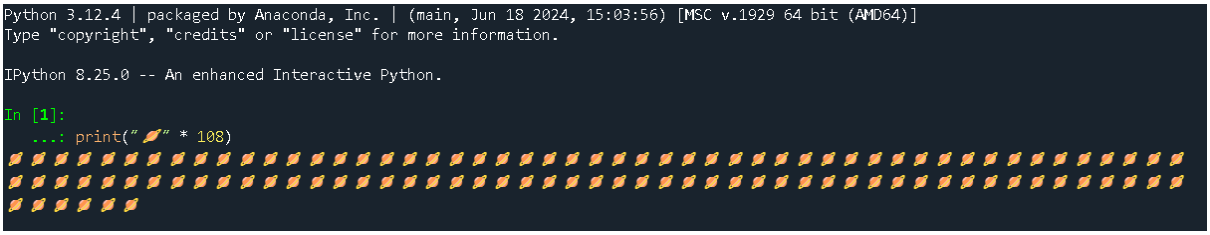
b) The string ends with “d”

c) The string ends with “c”



5. Write a code to print " 🥜 " one hundred and eight times.

print(" 🥜 " * 108)



6. Create a string “Grow Gratitude” and write a code to replace “Grow” with “Growth of”

Create the string

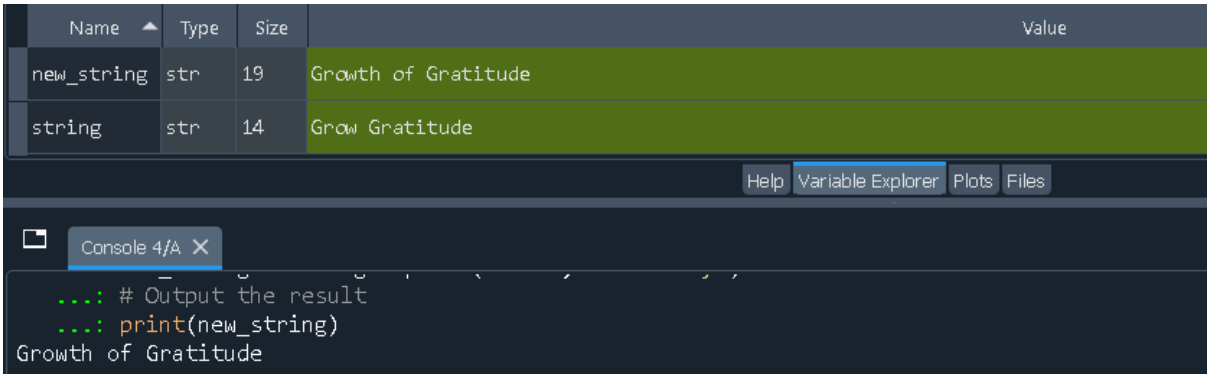
string = "Grow Gratitude"

Replace "Grow" with "Growth of"

new_string = string.replace("Grow", "Growth of")

Output the result

print(new_string)



7. A story was printed in a pdf, which isn't making any sense. i.e.:

".elgnujehtotniffo deps mehtfohtoB .eerfnoilehttesotseporeht no dewangdnanar eh ,ylkciuQ .elbuortninoilehtdecitondnatsapdeklawesuomeht ,nooS .repmihwotdetratsdnatuotegotgnilggurts saw noilehT .eert a tsniagapumihdeityehT .mehthtiwnoilehtkootdnatserofehtotniemacsretnuhwef a ,yad enO .ogmihteldnaecnedifnocs'esuomeht ta dehgualnoilehT ".emevasuoy fi yademosuoyotplehtaergfo eb lliw I ,uoyesimorp I"
.eerfmihtesotnoilehtdetseuqeryletarepsedesuomehtnehwesuomehttaeottuoba saw eH .yrgnaetiuqpuekow eh dna ,peels s'noilehtdebrutsidsihT .nufroftsujydobsihnwoddnapugninnurdetratsesuom a nehwelgnujehtnignipeelsecno saw noil A"

The backward story

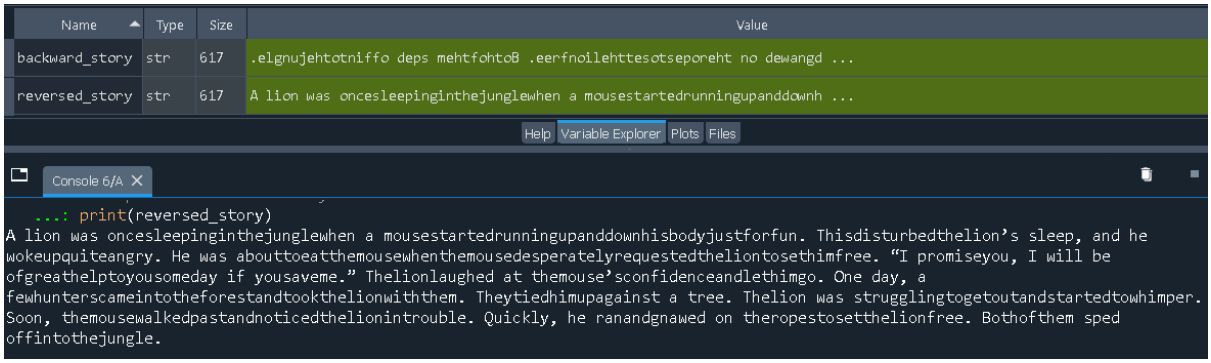
backward_story = """.elgnujehtotniffo deps mehtfohtoB .eerfnoilehttesotseporeht no dewangdnanar eh ,ylkciuQ .elbuortninoilehtdecitondnatsapdeklawesuomeht ,nooS .repmihwotdetratsdnatuotegotgnilggurts saw noilehT .eert a tsniagapumihdeityehT .mehthtiwnoilehtkootdnatserofehtotniemacsretnuhwef a ,yad enO .ogmihteldnaecnedifnocs'esuomeht ta dehgualnoilehT ".emevasuoy fi yademosuoyotplehtaergfo eb lliw I ,uoyesimorp I" .eerfmihtesotnoilehtdetseuqeryletarepsedesuomehtnehwesuomehttaeottuoba saw eH .yrgnaetiuqpuekow eh dna ,peels s'noilehtdebrutsidsihT .nufroftsujydobsihnwoddnapugninnurdetratsesuom a nehwelgnujehtnignipeelsecno saw noil A"""

Reverse the story to make it readable

reversed_story = backward_story[::-1]

Output the readable story

print(reversed_story)



TRANSFORMATIONS

Instructions:

Please share your answers filled inline in the word document. Submit code files wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Problem Statement:

Everything will revolve around the data in Analytics world. Proper data will help you to make useful predictions that improve your business. Sometimes the usage of original data as it is does not help to have accurate solutions. It is needed to convert the data from one form to another form to have better predictions. Explore various techniques to transform the data for better model performance. you can go through this link:

https://360digitmg.com/mindmap-data-science

1) Prepare the dataset by performing the preprocessing techniques, to have the data which improves model performance.

```
import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Creating the DataFrame from the given data

df = pd.read_csv(r"calories_consumed.csv")

# 1. Normalization

scaler = MinMaxScaler()

df_normalized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# 2. Standardization

standard_scaler = StandardScaler()

df_standardized = pd.DataFrame(standard_scaler.fit_transform(df), columns=df.columns)

# 3. Log Transformation (only applicable for positive values)

df_log_transformed = df.apply(np.log)

# 4. Square Root Transformation

df_sqrt_transformed = df.apply(np.sqrt)
```

Display the results

df_normalized, df_standardized, df_log_transformed, df_sqrt_transformed

Name	Type	Size	Value
df	DataFrame	(14, 2)	Column names: Weight gained (grams), Calories Consumed
df_log_transformed	DataFrame	(14, 2)	Column names: Weight gained (grams), Calories Consumed
df_normalized	DataFrame	(14, 2)	Column names: Weight gained (grams), Calories Consumed
df_sqrt_transformed	DataFrame	(14, 2)	Column names: Weight gained (grams), Calories Consumed
df_standardized	DataFrame	(14, 2)	Column names: Weight gained (grams), Calories Consumed
scaler	preprocessing._data.MinMaxScaler	1	MinMaxScaler object of sklearn.preprocessing._data module
standard_scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module

Help | Variable Explorer | Plots | Files

Console 9/A

```
In [7]: df_normalized, df_standardized, df_log_transformed, df_sqrt_transformed
Out[7]:
(    Weight gained (grams)  Calories Consumed
 0                0.044316              0.040
 1                0.132948              0.360
 2                0.807322              0.800
 3                0.132948              0.320
 4                0.229287              0.440
 5                0.046243              0.080
 6                0.063584              0.000
 7                0.000000              0.200
 8                0.518304              0.560
 9                1.000000              1.000
 10               0.036609              0.108
 11               0.084778              0.200
 12               0.277457              0.520
 13               0.614644              0.640,
    Weight gained (grams)  Calories Consumed
 0              -0.776586          -1.160005
 1              -0.490475          -0.056177
 2               1.686452           1.461586
```

```
3               -0.490475          -0.194155
4               -0.179485           0.219780
5               -0.770366          -1.022026
6               -0.714388          -1.297983
7               -0.919641          -0.608091
8                0.753483           0.633715
9                2.308432           2.151478
10              -0.801465          -0.925441
11              -0.645970          -0.608091
12              -0.023991           0.495737
13               1.064473           0.909672,
    Weight gained (grams)  Calories Consumed
0                4.682131           7.313220
1                5.298317           7.740664
2                6.802395           8.131531
3                5.298317           7.696213
4                5.703782           7.824046
5                4.700480           7.377759
6                4.852030           7.244228
7                4.127134           7.549609
8                6.396930           7.937375
9                7.003065           8.268732
10               4.605170           7.420579
11               5.010635           7.549609
12               5.857933           7.901007
13               6.551080           8.006368,
```

	Weight gained (grams)	Calories Consumed
0	10.392305	38.729833
1	14.142136	47.958315
2	30.000000	58.309519
3	14.142136	46.904158
4	17.320508	50.000000
5	10.488088	40.000000
6	11.313708	37.416574
7	7.874008	43.588989
8	24.494897	52.915026
9	33.166248	62.449980
10	10.000000	40.865633
11	12.247449	43.588989
12	18.708287	51.961524
13	26.457513	54.772256)

Zero-Variance Features

Instruction

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR

Batch Id: 04072024HYD10AM

Topic: Data Pre-Processing

Variance measures how far a set of data is spread out. A variance of zero indicates that all the data values are identical. There are various techniques to remove this for transforming the data into the suitable one for prediction.

Problem statement:

Find which columns of the given dataset with zero variance, and explore various techniques used to remove the zero variance from the dataset to perform certain analysis.

```
import pandas as pd

df = pd.read_csv(r"Z_dataset.csv")

print(df.head())

variance = df.var(numeric_only=True)

print(variance)

unique_colours = df['colour'].unique()

print(unique_colours)

from sklearn.feature_selection import VarianceThreshold

# Separating features and target variable if applicable

X = df.drop(columns=['Id'])  # Assuming 'Id' is not a feature

selector = VarianceThreshold(threshold=0.0)

# For variance thresholding, we need to convert categorical variables to numerical

# Let's use get_dummies for 'colour'

X_encoded = pd.get_dummies(X, drop_first=True)

# Fit the selector

selector.fit(X_encoded)

# Get the mask of features that pass the threshold

features_to_keep = X_encoded.columns[selector.get_support()]
```

```
# Create the reduced dataframe

X_reduced = X_encoded[features_to_keep]

print("Features before variance thresholding:", X_encoded.columns.tolist())

print("Features after variance thresholding:", features_to_keep.tolist())

# Identify columns with zero variance

zero_variance_cols = [col for col in df.columns if df[col].nunique() == 1]

print("Columns with zero variance:", zero_variance_cols)

# Drop zero variance columns

df_reduced = df.drop(columns=zero_variance_cols)

print("DataFrame after removing zero variance columns:")

print(df_reduced.head())
```

```
In [2]: df = pd.read_csv(r"Z_dataset.csv")

In [3]: print(df.head())
   Id  square.length  square.breadth  rec.Length  rec.breadth colour
0   1            5.1             3.5         1.4          0.2   Blue
1   2            4.9             3.0         1.4          0.2   Blue
2   3            4.7             3.2         1.3          0.2   Blue
3   4            4.6             3.1         1.5          0.2   Blue
4   5            5.0             3.6         1.4          0.2   Blue

In [4]: variance = df.var(numeric_only=True)
   ...: print(variance)
Id                1887.500000
square.length        0.685694
square.breadth       0.189979
rec.Length           3.116278
rec.breadth          0.581006
dtype: float64

In [5]: unique_colours = df['colour'].unique()
   ...: print(unique_colours)
['Blue' 'Green' 'Orange']
```

```
Features before variance thresholding: ['square.length', 'square.breadth', 'rec.Length', 'rec.breadth', 'colour_Green',
'colour_Orange']
Features after variance thresholding: ['square.length', 'square.breadth', 'rec.Length', 'rec.breadth', 'colour_Green',
'colour_Orange']

In [7]:
   ...: zero_variance_cols = [col for col in df.columns if df[col].nunique() == 1]
   ...: print("Columns with zero variance:", zero_variance_cols)
   ...: # Drop zero variance columns
   ...: df_reduced = df.drop(columns=zero_variance_cols)
   ...: print("DataFrame after removing zero variance columns:")
   ...: print(df_reduced.head())
Columns with zero variance: []
DataFrame after removing zero variance columns:
   Id  square.length  square.breadth  rec.Length  rec.breadth colour
0   1            5.1             3.5         1.4          0.2   Blue
1   2            4.9             3.0         1.4          0.2   Blue
2   3            4.7             3.2         1.3          0.2   Blue
3   4            4.6             3.1         1.5          0.2   Blue
4   5            5.0             3.6         1.4          0.2   Blue
```