one

February 13, 2024

[ ]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Read the CSV data
df = pd.read_csv(r"C:\Users\Network lab-16\Desktop\apple_quality.csv")

# Use pandas describe() function
description = df.describe()
```

```
              A_id         Size       Weight    Sweetness  Crunchiness  \
count  4000.000000  4000.000000  4000.000000  4000.000000  4000.000000
mean   1999.500000    -0.503015    -0.989547    -0.470479     0.985478
std    1154.844867     1.928059     1.602507     1.943441     1.402757
min       0.000000    -7.151703    -7.149848    -6.894485    -6.055058
25%     999.750000    -1.816765    -2.011770    -1.738425     0.062764
50%    1999.500000    -0.513703    -0.984736    -0.504758     0.998249
75%    2999.250000     0.805526     0.030976     0.801922     1.894234
max    3999.000000     6.406367     5.790714     6.374916     7.619852

          Juiciness     Ripeness
count  4000.000000  4000.000000
mean      0.512118     0.498277
std       1.930286     1.874427
min      -5.961897    -5.864599
25%      -0.801286    -0.771677
50%       0.534219     0.503445
75%       1.835976     1.766212
max       7.364403     7.237837
```

[ ]:
```
# Display the first 5 rows of the DataFrame
print("Displaying the first 5 rows of the DataFrame:")
print(df.head())
```

```python
# Display the last 5 rows of the DataFrame
print("\nDisplaying the last 5 rows of the DataFrame:")
print(df.tail())

# Display the shape of the DataFrame
print("\nThe shape of the DataFrame is:")
print(df.shape)

# Display the column names of the DataFrame
print("\nThe column names of the DataFrame are:")
print(df.columns)

# Display the data types of each column
print("\nThe data types of each column are:")
print(df.dtypes)

# Display summary statistics of the DataFrame
print("\nSummary statistics of the DataFrame:")
print(df.describe())

# Display the number of missing values in each column
print("\nThe number of missing values in each column are:")
print(df.isnull().sum())

# Fill missing values with the mean (for numerical columns)
df.fillna(df.mean(numeric_only=True), inplace=True)

# Drop duplicates
df.drop_duplicates(inplace=True)
print("\nDataFrame after dropping duplicates:")
print(df.head())
```

```
Displaying the first 5 rows of the DataFrame:
   A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0   0.0 -3.970049 -2.512336   5.346330    -1.012009   1.844900  0.329840
1   1.0 -1.195217 -2.839257   3.664059     1.588232   0.853286  0.867530
2   2.0 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -0.038033
3   3.0 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -3.413761
4   4.0  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -1.303849

        Acidity Quality
0  -0.491590483    good
1  -0.722809367    good
2   2.621636473     bad
3   0.790723217    good
4   0.501984036    good
```

```
Displaying the last 5 rows of the DataFrame:
        A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
3996  3996.0 -0.293118  1.949253  -0.204020    -0.640196   0.024523 -1.087900
3997  3997.0 -2.634515 -2.138247  -2.440461     0.657223   2.199709  4.763859
3998  3998.0 -4.008004 -1.779337   2.366397    -0.200329   2.161435  0.214488
3999  3999.0  0.278540 -1.715505   0.121217    -1.154075   1.266677 -0.776571
4000  1999.5 -0.503015 -0.989547  -0.470479     0.985478   0.512118  0.498277

                                 Acidity Quality
3996                          1.854235285    good
3997                         -1.334611391     bad
3998                         -2.229719806    good
3999                          1.599796456    good
4000  Created_by_Nidula_Elgiriyewithana     NaN

The shape of the DataFrame is:
(4001, 9)

The column names of the DataFrame are:
Index(['A_id', 'Size', 'Weight', 'Sweetness', 'Crunchiness', 'Juiciness',
       'Ripeness', 'Acidity', 'Quality'],
      dtype='object')

The data types of each column are:
A_id           float64
Size           float64
Weight         float64
Sweetness      float64
Crunchiness    float64
Juiciness      float64
Ripeness       float64
Acidity         object
Quality         object
dtype: object

Summary statistics of the DataFrame:
              A_id         Size       Weight    Sweetness  Crunchiness  \
count  4001.000000  4001.000000  4001.000000  4001.000000  4001.000000
mean   1999.500000    -0.503015    -0.989547    -0.470479     0.985478
std    1154.700502     1.927818     1.602307     1.943198     1.402582
min       0.000000    -7.151703    -7.149848    -6.894485    -6.055058
25%    1000.000000    -1.816202    -2.011767    -1.738424     0.062865
50%    1999.500000    -0.513569    -0.984984    -0.504307     0.997933
75%    2999.000000     0.805329     0.029773     0.801810     1.893586
max    3999.000000     6.406367     5.790714     6.374916     7.619852

         Juiciness     Ripeness
```

```
count  4001.000000  4001.000000
mean      0.512118     0.498277
std       1.930044     1.874192
min      -5.961897    -5.864599
25%      -0.801238    -0.770970
50%       0.533504     0.502737
75%       1.835903     1.765281
max       7.364403     7.237837

The number of missing values in each column are:
A_id           0
Size           0
Weight         0
Sweetness      0
Crunchiness    0
Juiciness      0
Ripeness       0
Acidity        0
Quality        1
dtype: int64

DataFrame after dropping duplicates:
   A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0   0.0 -3.970049 -2.512336   5.346330    -1.012009   1.844900  0.329840
1   1.0 -1.195217 -2.839257   3.664059     1.588232   0.853286  0.867530
2   2.0 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -0.038033
3   3.0 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -3.413761
4   4.0  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -1.303849

        Acidity Quality
0  -0.491590483    good
1  -0.722809367    good
2   2.621636473     bad
3   0.790723217    good
4   0.501984036    good
```

[ ]:
```python
# Return the number of dimensions of an array.
print("Return the number of dimensions of an array:")
print(df.ndim)

# Return the size of the fram along the axis
print("Return the size of the fram along the axis")
print(df.size)
```

```
Return the number of dimensions of an array:
2
Return the size of the fram along the axis
36009
```

[ ]:
```
# Assuming df is your DataFrame and 'A_id' is the column
df['A_id'] = df['A_id'].astype('int64')

# Print the datatypes and display the changes
df.dtypes
```

[ ]:
```
A_id             int64
Size           float64
Weight         float64
Sweetness      float64
Crunchiness    float64
Juiciness      float64
Ripeness       float64
Acidity         object
Quality         object
dtype: object
```

[ ]:
```
# Assuming df is your DataFrame and 'Quality' is the column
quality_dummies = pd.get_dummies(df['Quality'], prefix='Quality')
print(quality_dummies)
```

```
      Quality_bad  Quality_good
0               0             1
1               0             1
2               1             0
3               0             1
4               0             1
…             …             …
3996            0             1
3997            1             0
3998            0             1
3999            0             1
4000            0             0

[4001 rows x 2 columns]
```

[ ]:
```
# Assuming df is your original DataFrame and quality_dummies is the DataFrame␣
  ↪with dummy variables
df_with_dummies = pd.concat([df['Weight'], quality_dummies], axis=1)
print(df_with_dummies)
```

```
        Weight  Quality_bad  Quality_good
0    -2.512336            0             1
1    -2.839257            0             1
2    -1.351282            1             0
3    -2.271627            0             1
4    -1.296612            0             1
…          …          …             …
```

```
3996  1.949253           0            1
3997 -2.138247           1            0
3998 -1.779337           0            1
3999 -1.715505           0            1
4000 -0.989547           0            0

[4001 rows x 3 columns]
```

[ ]:
```
# Merging data with itself
merged_dataframe = pd.merge(df, df, on='Quality')
print(merged_dataframe.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8000033 entries, 0 to 8000032
Data columns (total 17 columns):
 #   Column         Dtype
---  ------         -----
 0   A_id_x         int64
 1   Size_x         float64
 2   Weight_x       float64
 3   Sweetness_x    float64
 4   Crunchiness_x  float64
 5   Juiciness_x    float64
 6   Ripeness_x     float64
 7   Acidity_x      object
 8   Quality        object
 9   A_id_y         int64
 10  Size_y         float64
 11  Weight_y       float64
 12  Sweetness_y    float64
 13  Crunchiness_y  float64
 14  Juiciness_y    float64
 15  Ripeness_y     float64
 16  Acidity_y      object
dtypes: float64(12), int64(2), object(3)
memory usage: 1.1+ GB
None
```

[ ]:
```

```