Data Collection and Preprocessing Phase

Date	10 June 2024
Team ID	-
Project Title	Golden Harvesting: A Predictive Model For Apple Quality Assurance
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

This report examines a dataset of 4001 rows and 9 columns, addressing data quality concerns such as missing values and duplicates, and implements strategies for accurate analysis. It employs univariate, bivariate, and multivariate analyses to assess apple quality attributes and utilizes preprocessing techniques to optimize predictive modeling for quality assurance.

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>Dimensions of data:
<pre>data.shape</pre>
<pre>(4001, 9)</pre>
4001 rows × 9columns

Information about DataFrame:
<pre>data.info()</pre>
<pre><class 'pandas.core.frame.DataFrame'>
RangeIndex: 4001 entries, 0 to 4000
Data columns (total 9 columns):
 #   Column   Non-Null Count Dtype
--- ------   -------------- -----
 0   A_id   4000 non-null   float64
 1   Size   4000 non-null   float64
 2   Weight   4000 non-null   float64
 3   Sweetness   4000 non-null   float64
 4   Crunchiness 4000 non-null   float64
 5   Juiciness   4000 non-null   float64
 6   Ripeness   4000 non-null   float64
 7   Acidity   4001 non-null   object
 8   Quality   4000 non-null   object
dtypes: float64(7), object(2)
memory usage: 281.4+ KB</pre>
Descriptive statistics:</td></tr>
</table>

```
# Descriptive Statistics
data.describe()
```

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000
mean	-0.502695	-0.991229	-0.472248	0.984194	0.513127	0.498102	0.076639
std	1.917446	1.574517	1.931684	1.369437	1.917024	1.866614	2.101441
min	-5.750201	-5.075890	-5.548946	-2.684440	-4.757179	-4.578510	-5.709299
25%	-1.816765	-2.011770	-1.738425	0.062764	-0.801286	-0.771677	-1.377424
50%	-0.513703	-0.984736	-0.504758	0.998249	0.534219	0.503445	0.022609
75%	0.805526	0.030976	0.801922	1.894234	1.835976	1.766212	1.510493
max	4.738963	3.095097	4.612442	4.641439	5.791870	5.573044	5.842368

Univariate Analysis

```
# Univariate Analysis
plt.figure(figsize=(15,7))
data.hist(bins=10, figsize=(15, 10), color='c')
plt.suptitle("Histograms of Numerical Features", y=1.02)
plt.show()
```

Bivariate Analysis

```
# Bivariate Analysis
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Sweetness', y='Crunchiness', hue='Quality', palette='coolwarm', data=data, s=100)
sns.regplot(x='Sweetness', y='Crunchiness', data=data, scatter=False, color='blue')
plt.title('Scatter Plot of Sweetness vs. Crunchiness by Quality')
plt.xlabel('Sweetness')
plt.ylabel('Crunchiness')
plt.show()
```

Multivariate Analysis

```
# Multivariate analysis
pair_plot = sns.pairplot(data, hue="Quality", diag_kind="kde", palette='coolwarm')
pair_plot.fig.suptitle("Pairplot of Features by Quality", y=1.02)
plt.show()
```

Outliers and Anomalies

Handling outliers for the column Size

```
quant = data['Size'].quantile(q=[0.75,0.25])
print(quant)
Q3 = quant.loc[0.75]
print("Q3(75th percentile)->",Q3)
Q1 = quant.loc[0.25]
print("Q1(25th percentile)->",Q1)
IQR = Q3 - Q1
print("IQR(InterQuartile Range)->",IQR)
maxwhisker = Q3 + 1.5 * IQR
print("Max outliers->",maxwhisker)
minwhisker = Q1 - 1.5 * IQR
print("Min outliers->",minwhisker)
data['Size'] = np.where(data['Size'] > 4.73896291425, 4.73896291425, data['Size'])
data['Size'] = np.where(data['Size'] < -5.75020099175, -5.75020099175, data['Size'])
sns.boxplot(data['Size'])
```

	Handling outliers for the column Weight
	`quant = data['Weight'].quantile(q=[0.75,0.25])`
	`print(quant)`
	`Q3 = quant.loc[0.75]`
	`print("Q3(75th percentile)->",Q3)`
	`Q1 = quant.loc[0.25]`
	`print("Q1(25th percentile)->",Q1)`
	`IQR = Q3 - Q1`
	`print("IQR(InterQuartile Range)->",IQR)`
	`maxwhisker = Q3 + 1.5 * IQR`
	`print("Max outliers->",maxwhisker)`
	`minwhisker = Q1 - 1.5 * IQR`
	`print("Min outliers->",minwhisker)`
	`data['Weight'] = np.where(data['Weight'] > 3.0950965391249996, 3.0950965391249996, data['Weight'])`
	`data['Weight'] = np.where(data['Weight'] < -5.075890391874999, -5.075890391874999, data['Weight'])`
	`sns.boxplot(data['Weight'])`
	
	Handling outliers for the column Sweetness
	`quant = data['Sweetness'].quantile(q=[0.75,0.25])`
	`print(quant)`
	`Q3 = quant.loc[0.75]`
	`print("Q3(75th percentile)->",Q3)`
	`Q1 = quant.loc[0.25]`
	`print("Q1(25th percentile)->",Q1)`
	`IQR = Q3 - Q1`
	`print("IQR(InterQuartile Range)->",IQR)`
	`maxwhisker = Q3 + 1.5 * IQR`
	`print("Max outliers->",maxwhisker)`
	`minwhisker = Q1 - 1.5 * IQR`
	`print("Min outliers->",minwhisker)`
	`data['Sweetness'] = np.where(data['Sweetness'] > 4.61244239625, 4.61244239625, data['Sweetness'])`
	`data['Sweetness'] = np.where(data['Sweetness'] < -5.54894553775, -5.54894553775, data['Sweetness'])`
	`sns.boxplot(data['Sweetness'])`
	
	Handling outliers for the column Crunchiness
	`quant = data['Crunchiness'].quantile(q=[0.75,0.25])`
	`print(quant)`
	`Q3 = quant.loc[0.75]`
	`print("Q3(75th percentile)->",Q3)`
	`Q1 = quant.loc[0.25]`
	`print("Q1(25th percentile)->",Q1)`
	`IQR = Q3 - Q1`
	`print("IQR(InterQuartile Range)->",IQR)`
	`maxwhisker = Q3 + 1.5 * IQR`
	`print("Max outliers->",maxwhisker)`
	`minwhisker = Q1 - 1.5 * IQR`
	`print("Min outliers->",minwhisker)`
	`data['Crunchiness'] = np.where(data['Crunchiness'] > 4.641438949625, 4.641438949625, data['Crunchiness'])`
	`data['Crunchiness'] = np.where(data['Crunchiness'] < -2.6844403373750003, -2.6844403373750003,`
	`data['Crunchiness'])`
	`sns.boxplot(data['Crunchiness'])`

Handling outliers for the column Juiciness

```
quant = data['Juiciness'].quantile(q=[0.75,0.25])
print(quant)
Q3 = quant.loc[0.75]
print("Q3(75th percentile)->",Q3)
Q1 = quant.loc[0.25]
print("Q1(25th percentile)->",Q1)
IQR = Q3 - Q1
print("IQR(InterQuartile Range)->",IQR)
maxwhisker = Q3 + 1.5 * IQR
print("Max outliers->",maxwhisker)
minwhisker = Q1 - 1.5 * IQR
print("Min outliers->",minwhisker)
data['Juiciness'] = np.where(data['Juiciness'] > 5.791869691624999, 5.791869691624999, data['Juiciness'])
data['Juiciness'] = np.where(data['Juiciness'] < -4.7571791193749995, -4.7571791193749995,
                             data['Juiciness'])
sns.boxplot(data['Juiciness'])
```

Handling outliers for the column Ripeness

```
quant = data['Ripeness'].quantile(q=[0.75,0.25])
print(quant)
Q3 = quant.loc[0.75]
print("Q3(75th percentile)->",Q3)
Q1 = quant.loc[0.25]
print("Q1(25th percentile)->",Q1)
IQR = Q3 - Q1
print("IQR(InterQuartile Range)->",IQR)
maxwhisker = Q3 + 1.5 * IQR
print("Max outliers->",maxwhisker)
minwhisker = Q1 - 1.5 * IQR
print("Min outliers->",minwhisker)
data['Ripeness'] = np.where(data['Ripeness'] > 5.573044401624999, 5.573044401624999, data['Ripeness'])
data['Ripeness'] = np.where(data['Ripeness'] < -4.578509627375, -4.578509627375, data['Ripeness'])
sns.boxplot(data['Ripeness'])
```

Handling outliers for the column Acidity

```
data['Acidity'] = pd.to_numeric(data['Acidity'], errors='coerce')
data = data.dropna(subset=['Acidity'])
quant = data['Acidity'].quantile(q=[0.75, 0.25])
print(quant)
Q3 = quant.loc[0.75]
print("Q3 (75th percentile) ->", Q3)
Q1 = quant.loc[0.25]
print("Q1 (25th percentile) ->", Q1)
IQR = Q3 - Q1
print("IQR (InterQuartile Range) ->", IQR)
maxwhisker = Q3 + 1.5 * IQR
print("Max outliers ->", maxwhisker)
minwhisker = Q1 - 1.5 * IQR
print("Min outliers ->", minwhisker)
data['Acidity'] = np.where(data['Acidity'] > maxwhisker, maxwhisker, data['Acidity'])
data['Acidity'] = np.where(data['Acidity'] < minwhisker, minwhisker, data['Acidity'])
sns.boxplot(data['Acidity'])
```

Data Preprocessing Code Screenshots

Loading Data

```
# Read The Dataset
data = pd.read_csv('apple_quality.csv')
data.head()
```

	A_id	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	0.0	-3.970049	-2.512336	5.346330	-1.012009	1.844900	0.329840	-0.491590483	good
1	1.0	-1.195217	-2.839257	3.664059	1.588232	0.853286	0.867530	-0.722809367	good
2	2.0	-0.292024	-1.351282	-1.738429	-0.342616	2.838636	-0.038033	2.621636473	bad
3	3.0	-0.657196	-2.271627	1.324874	-0.097875	3.637970	-3.413761	0.790723217	good
4	4.0	1.364217	-1.296612	-0.384658	-0.553006	3.030874	-1.303849	0.501984036	good

Handling Missing Data

```
data.isnull().sum()
```

```
A_id           1
Size           1
Weight         1
Sweetness      1
Crunchiness    1
Juiciness      1
Ripeness       1
Acidity        0
Quality        1
dtype: int64
```

Data Transformation

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x = scaler.fit_transform(x)
x
```

```
array([[-1.79842417, -0.95037339,  2.99342063, ...,  0.69054495,
        -0.08987211, -0.26941526],
       [-0.35906018, -1.15440431,  2.12769769, ...,  0.17676683,
         0.1970196 , -0.37899737],
       [ 0.1094454 , -0.22575916, -0.65250727, ...,  1.20542179,
        -0.28615565,  1.20604367],
       ...,
       [-1.1056547 , -0.71690397, -1.01378401, ...,  0.87437918,
         2.27595716, -0.66895013],
       [-1.81811235, -0.49290842,  1.45990059, ...,  0.85454883,
        -0.15141937, -1.09317096],
       [ 0.40540882, -0.45307081,  0.30449592, ...,  0.39095445,
        -0.68021237,  0.72176064]])
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
y
```

```
C:\Users\fs22a\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:114: DataConver
sionWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
array([1, 1, 0, ..., 0, 1, 1])
```

Feature Engineering	```data.dropna(inplace=True)``` ```data.shape``` ```(4000, 9)```
Save Processed Data	Saved the cleaned and processed data in ‘X’ and ‘y’ (variables) for future use.