Jakie cechy jabłka wpływają na postrzeganie przez ludzi jego jakości? Czy istnieją koleracje między zmienną najbardziej wpływającą na postrzeganie jakości a innymi zmiennymi? Wyćwiczenie modelu wykrywającego jakość jabłka.

Sprawdzam podstawowe parametry, oczyszczam dane.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [2]:
```
df = pd.read_csv(r'C:\Users\48797\Documents\ProjektyPython\Analizadabika\apple_quality.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	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	0.853296	0.867530	-0.722809367	good
2	2.0	-0.292124	-1.351282	-1.738429	-0.342916	2.838636	-0.038433	2.621636473	bad
3	3.0	-0.657196	-2.271627	1.324674	-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

In [4]:
```
df.describe()
```

Out[4]:

	A_id	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000
mean	1999.500000	-0.503015	-0.989547	-0.470479	0.985478	0.512118	0.498277
std	1154.844867	1.928059	1.602507	1.943441	1.402757	1.930266	1.874427
min	0.000000	-7.151703	-7.149848	-6.894485	-6.055058	-5.961897	-5.864599
25%	999.750000	-1.816766	-2.011770	-1.738429	0.062764	-0.801266	-0.779477
50%	1999.500000	-0.513703	-0.984736	-0.504758	0.998249	0.534219	0.503445
75%	2999.250000	0.805526	0.030976	0.801922	1.894254	1.835976	1.796212
max	3999.000000	6.406367	5.790714	6.374916	7.619852	7.364403	7.237837

In [5]:
```
df.info()
```

```
<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
```

In [6]:
```
df =df.drop(labels = 'A_id', axis = 1)
```

In [7]:
```
df.isna()
```

Out[7]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
3996	False	False	False	False	False	False	False	False
3997	False	False	False	False	False	False	False	False
3998	False	False	False	False	False	False	False	False
3999	False	False	False	False	False	False	False	False
4000	True	True	True	True	True	True	False	True

4001 rows × 8 columns

In [8]:
```
df = df.dropna()
```

In [9]:
```
df['Quality'] = (df['Quality'] == 'good').astype(int)
```

In [10]:
```
df['Acidity'] = df['Acidity'].astype(float)
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4000 entries, 0 to 3999
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Size        4000 non-null   float64
 1   Weight      4000 non-null   float64
 2   Sweetness   4000 non-null   float64
 3   Crunchiness 4000 non-null   float64
 4   Juiciness   4000 non-null   float64
 5   Ripeness    4000 non-null   float64
 6   Acidity     4000 non-null   float64
 7   Quality     4000 non-null   int32  
dtypes: float64(7), int32(1)
memory usage: 265.6 KB
```

Sprawdzam korelację wszystkich zmiennych.

In [12]:
```
df.corr()
```

Out[12]:

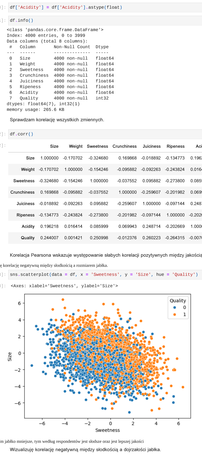
	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
Size	1.000000	-0.170702	-0.324600	0.169868	-0.016892	-0.134773	0.196218	0.244007
Weight	-0.170702	1.000000	-0.154246	-0.095882	-0.092263	-0.243824	0.016414	0.001421
Sweetness	-0.324600	-0.154246	1.000000	-0.037552	0.056682	-0.273600	0.085999	0.250998
Crunchiness	0.169868	-0.095882	-0.037552	1.000000	-0.259607	-0.201982	0.069643	-0.012376
Juiciness	-0.016892	-0.092263	0.056682	-0.259607	1.000000	-0.097144	0.248714	0.265223
Ripeness	-0.134773	-0.243824	-0.273600	-0.201982	-0.097144	1.000000	-0.202669	-0.284516
Acidity	0.196218	0.016414	0.085999	0.069643	0.248714	-0.202669	1.000000	-0.007697
Quality	0.244007	0.001421	0.250998	-0.012376	0.265223	-0.284516	-0.007697	1.000000

Korelacja Pearsona wskazuje występowanie słabych korelacji pozytywnych między jakością jabłka a jego wielkością, słodkością i soczystością oraz słabą korelację negatywną między jakością a dojrzałością jabłka.

Wizualizuję korelację negatywną między słodkością a rozmiarem jabłka.

In [13]:
```
sns.scatterplot(data = df, x = 'Sweetness', y = 'Size', hue = 'Quality')
```

Out[13]:
```
<Axes: xlabel='Sweetness', ylabel='Size'>
```

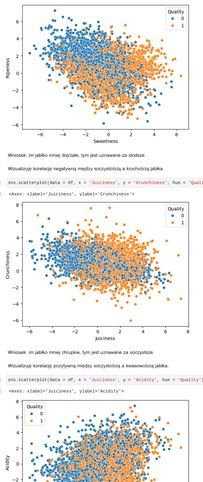


Wniosek: im jabłko mniejsze, tym według respondentów jest słodsze oraz jest bujniej jakości

Wizualizuję korelację negatywną między słodkością a dojrzałością jabłka.

In [14]:
```
sns.scatterplot(data = df, x = 'Sweetness', y = 'Ripeness', hue = 'Quality')
```

Out[14]:
```
<Axes: xlabel='Sweetness', ylabel='Ripeness'>
```



Wniosek: im jabłko mniej dojrzałe, tym jest uznawane za słodsze

Wizualizuję korelację negatywną między soczystością a kruchością jabłka.

In [15]:
```
sns.scatterplot(data = df, x = 'Juiciness', y = 'Crunchiness', hue = 'Quality')
```

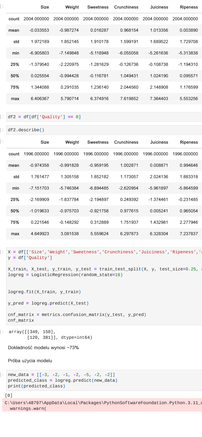
Out[15]:
```
<Axes: xlabel='Juiciness', ylabel='Crunchiness'>
```

Wniosek: im jabłko mniej chrupkie, tym jest uznawane za soczystsze.

Wizualizuję korelację pozytywną między soczystością a kwasowością jabłka.

In [16]:
```
sns.scatterplot(data = df, x = 'Juiciness', y = 'Acidity', hue = 'Quality')
```

Out[16]:
```
<Axes: xlabel='Juiciness', ylabel='Acidity'>
```



Wniosek: im jabłko bardziej kwaskowe, tym jest uznawane za soczystsze.

In [17]:
```
grupa_dobre = df[df['Quality'] == 1]['Sweetness'].values
grupa_zle = df[df['Quality'] == 0]['Sweetness'].values

statystyka, p_value = ttest_ind(grupa_dobre, grupa_zle)

if p_value < 0.05:
    print('Różnice są istotne statystycznie.')
else:
    print('Brak istotnych statystycznie różnic.')
```

```
Różnice są istotne statystycznie.
```

In [18]:
```
grupa_dobre = df[df['Quality'] == 1]['Juiciness'].values
grupa_zle = df[df['Quality'] == 0]['Juiciness'].values

statystyka, p_value = ttest_ind(grupa_dobre, grupa_zle)

if p_value < 0.05:
    print('Różnice są istotne statystycznie.')
else:
    print('Brak istotnych statystycznie różnic.')
```

```
Różnice są istotne statystycznie.
```

In [19]:
```
grupa_dobre = df[df['Quality'] == 1]['Size'].values
grupa_zle = df[df['Quality'] == 0]['Size'].values

statystyka, p_value = ttest_ind(grupa_dobre, grupa_zle)

if p_value < 0.05:
    print('Różnice są istotne statystycznie.')
else:
    print('Brak istotnych statystycznie różnic.')
```

```
Różnice są istotne statystycznie.
```

In [20]:
```
grupa_dobre = df[df['Quality'] == 1]['Ripeness'].values
grupa_zle = df[df['Quality'] == 0]['Ripeness'].values

statystyka, p_value = ttest_ind(grupa_dobre, grupa_zle)

if p_value < 0.05:
    print('Różnice są istotne statystycznie.')
else:
    print('Brak istotnych statystycznie różnic.')
```

```
Różnice są istotne statystycznie.
```

Wszystkie zmienne wpływające na jakość (prosukcją) korelują jabłka są istotne statystycznie.

In [21]:
```
df1 = df[df['Quality'] == 1]
```

In [22]:
```
df1
```

Out[22]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	-3.970049	-2.512336	5.346330	-1.012009	1.844900	0.329840	-0.491590	1
1	-1.195217	-2.839257	3.664059	1.588232	0.853296	0.867530	-0.722809	1
3	-0.657196	-2.271627	1.324674	-0.097875	3.637970	-3.413761	0.790723	1
4	1.364217	-1.296612	-0.384658	-0.553006	3.030874	-1.303849	0.501984	1
6	1.331600	1.633956	0.870974	-1.677796	3.106344	-1.947417	2.414171	1
...	...	...	...	...	...	...	...	...
3992	1.764253	-2.079696	-0.083383	-0.086724	1.703385	3.840101	-0.338261	1
3994	1.482508	-2.581181	-0.306888	1.527877	1.056361	2.560829	-1.229255	1
3996	0.293118	1.949253	-0.204020	-0.640196	0.024523	1.087900	1.854235	1
3998	-4.008004	-1.779337	2.366397	-0.200329	2.161436	0.214488	-2.229720	1
3999	0.278940	-1.715505	0.121217	-1.154070	1.266677	-0.776571	1.599796	1

2004 rows × 8 columns

In [23]:
```
df1.describe()
```

Out[23]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
count	2004.000000	2004.000000	2004.000000	2004.000000	2004.000000	2004.000000	2004.000000	2004.0
mean	-0.033553	-0.987274	0.016287	0.968154	1.013356	0.003690	0.060668	1.0
std	1.972189	1.652145	1.910178	1.598191	1.689522	1.729706	2.006745	0.0
min	-6.905803	-7.149848	-5.110946	-6.055058	-5.261636	-5.313838	-7.010538	1.0
25%	-1.370640	-2.220975	-1.261629	-0.126736	-0.106738	-1.194310	-1.332003	1.0
50%	0.025554	-0.994428	-0.116781	1.044431	1.024190	0.099571	0.060766	1.0
75%	1.344068	0.291205	1.226140	2.044560	2.148900	1.176599	1.454807	1.0
max	6.406367	5.790714	6.374916	7.619852	7.364403	5.553256	7.193374	1.0

In [24]:
```
df2 = df[df['Quality'] == 0]
```

In [27]:
```
df2.describe()
```

Out[27]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
count	1996.000000	1996.000000	1996.000000	1996.000000	1996.000000	1996.000000	1996.000000	1996.0
mean	-0.974358	-0.991828	-0.959195	1.002871	0.008671	0.994646	0.093151	0.0
std	1.761477	1.305158	1.852182	1.172657	2.024136	1.883318	2.209720	0.0
min	-7.151703	-5.746364	-6.894485	-5.620954	-5.961897	-5.864599	-6.959460	0.0
25%	-2.168909	-1.837784	-2.194597	0.249392	-1.374461	-0.231465	-1.427181	0.0
50%	-1.019633	-0.975703	-0.921758	0.978715	0.009241	0.963054	-0.010607	0.0
75%	0.221546	-0.146292	0.312889	1.751937	1.432681	2.277946	1.595519	0.0
max	4.649823	3.081538	5.559624	6.297873	6.328304	7.237837	7.404736	0.0

In [28]:
```
X = df[['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','Acidity']]
y = df['Quality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)
logreg = LogisticRegression(random_state=16)

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[32]:
```
array([[349, 150],
       [126, 381]], dtype=int64)
```

Dokładność modelu wynosi ~73%

Próba użycia modelu

In [29]:
```
new_data = [-3, -2, -1, -2, -5, -2, -2]
predicted_class = logreg.predict(new_data)
print(predicted_class)
```

```
[0]
```

```
C:\Users\48797\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [ ]:

In [ ]: