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

pk=pd.read_csv('apple_quality prediction 1.csv')

pk.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   A_id         4000 non-null   int64  
 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      4000 non-null   float64
 8   Quality      4000 non-null   object 
dtypes: float64(7), int64(1), object(1)
memory usage: 281.4+ KB
```

```python
pk.head(20)
```

```
    A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  
Ripeness  \
0      0 -3.970049 -2.512336   5.346330    -1.012009   1.844900   
0.329840   
1      1 -1.195217 -2.839257   3.664059     1.588232   0.853286   
0.867530   
2      2 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -
0.038033   
3      3 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -
3.413761   
4      4  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -
1.303849   
5      5 -3.425400 -1.409082  -1.913511    -0.555775  -3.853071   
1.914616   
6      6  1.331606  1.635956   0.875974    -1.677798   3.106344 -
1.847417   
7      7 -1.995462 -0.428958   1.530644    -0.742972   0.158834   
0.974438   
8      8 -3.867632 -3.734514   0.986429    -1.207655   2.292873   
4.080921   
9      9 -0.727983 -0.442820  -4.092223     0.597513   0.393714   
1.620857   
```

```
10    10 -2.699336 -1.329507  -1.418507    -0.625546   2.371074
3.403165
11    11  2.450960 -0.564177  -1.635041     0.942400  -2.087317
1.214322
12    12 -0.170812 -1.867271  -1.771845     2.413155  -3.094555 -
0.624884
13    13 -1.345531 -1.623701   2.044144     1.754813   0.997567
0.434180
14    14  2.839581 -0.344798  -1.019797     0.894581  -1.300061
0.582379
15    15 -2.659887 -2.795684   4.230404     0.697550   2.180911 -
0.088775
16    16 -1.468952 -1.950360  -2.214373     0.909759   2.864449
3.965956
17    17 -0.074370 -4.714750   0.249768     2.935319   1.409755 -
2.643810
18    18 -0.302364  1.724396  -2.442337     3.465108   0.449792 -
0.074362
19    19 -2.108050  0.356467  -1.156193     4.326723   1.561543 -
4.630174

     Acidity Quality
0  -0.491590    good
1  -0.722809    good
2   2.621636     bad
3   0.790723    good
4   0.501984    good
5  -2.981523     bad
6   2.414171    good
7  -1.470125    good
8  -4.871905     bad
9   2.185608     bad
10 -2.810808     bad
11  1.294324    good
12 -2.076114     bad
13  1.724026    good
14  1.709708    good
15 -1.083621    good
16 -0.558209     bad
17  1.250970    good
18  2.493782     bad
19 -1.376657    good
```

```
pk.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	Acidity
count	4000.000000	4000.000000	4000.000000
mean	0.512118	0.498277	0.076877
std	1.930286	1.874427	2.110270
min	-5.961897	-5.864599	-7.010538
25%	-0.801286	-0.771677	-1.377424
50%	0.534219	0.503445	0.022609
75%	1.835976	1.766212	1.510493
max	7.364403	7.237837	7.404736

```
pk.dtypes
```

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

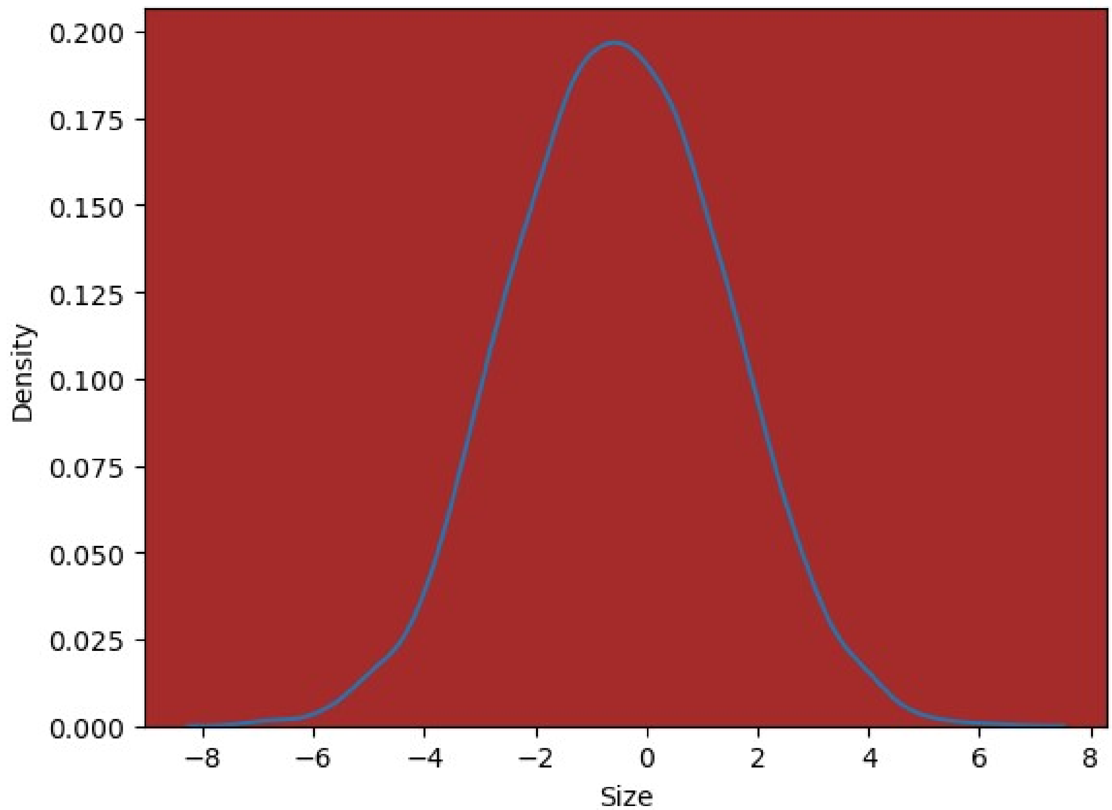
```
std=pk.select_dtypes(include=['float']).std()

standard_error=std/np.sqrt(len(pk))

std
```

```
Size           1.928059
Weight         1.602507
Sweetness      1.943441
Crunchiness    1.402757
Juiciness      1.930286
Ripeness       1.874427
Acidity        2.110270
dtype: float64
```

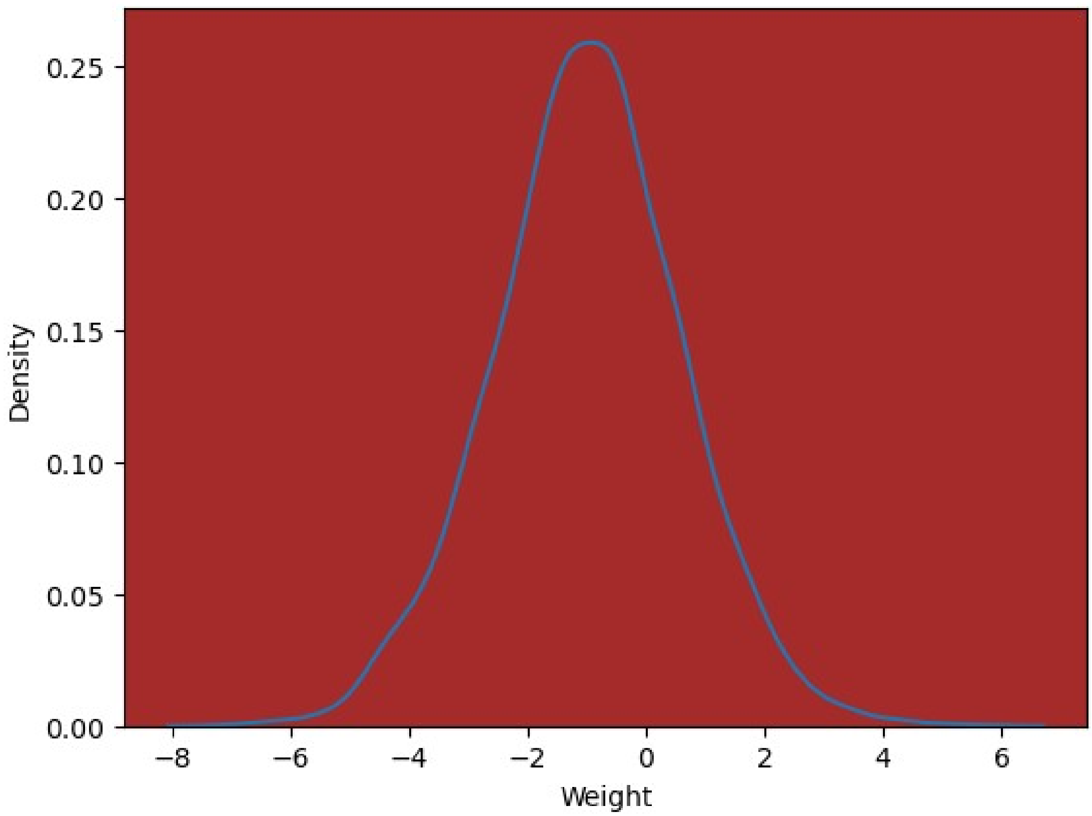
```
standard_error

Size           0.030485
Weight         0.025338
Sweetness      0.030728
Crunchiness    0.022180
Juiciness      0.030520
Ripeness       0.029637
Acidity        0.033366
dtype: float64
```

Data have less variability

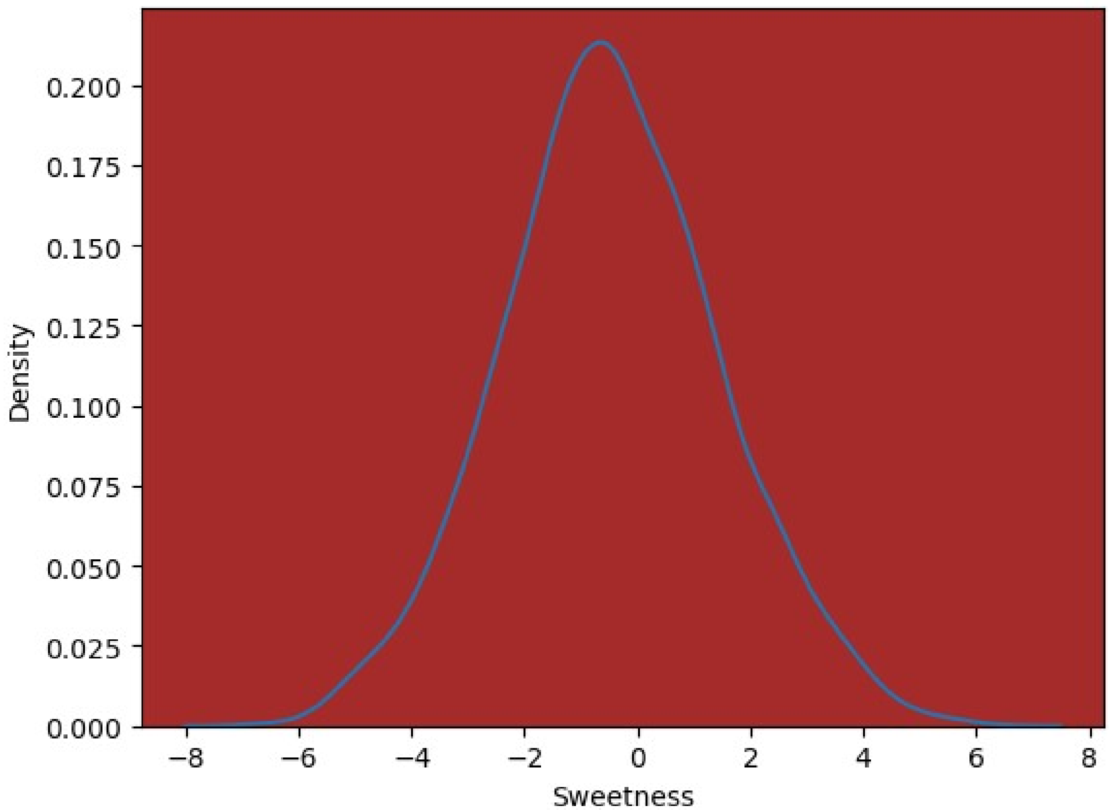
```
pk.select_dtypes(include=['float']).skew()

Size          -0.002437
Weight         0.003102
Sweetness      0.083850
Crunchiness    0.000230
Juiciness     -0.113421
Ripeness      -0.008764
Acidity        0.055783
dtype: float64

pk.select_dtypes(include=['float']).kurt()

Size          -0.083341
Weight         0.359050
Sweetness      0.014472
Crunchiness    0.722020
Juiciness      0.028735
Ripeness      -0.071850
Acidity       -0.093451
dtype: float64

for i in
['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','Aci
dity']:
    sns.kdeplot(data=pk,x=i)
    plt.gca().set_facecolor('brown')
    plt.show()

C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



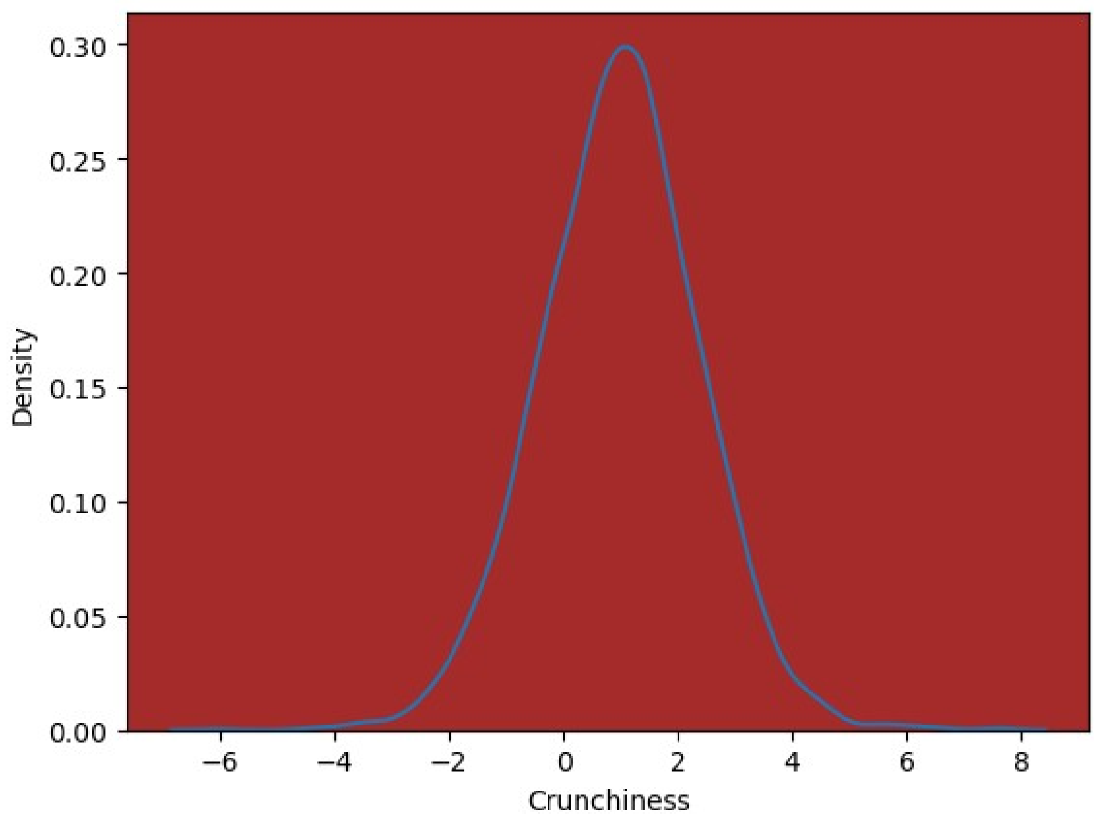

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



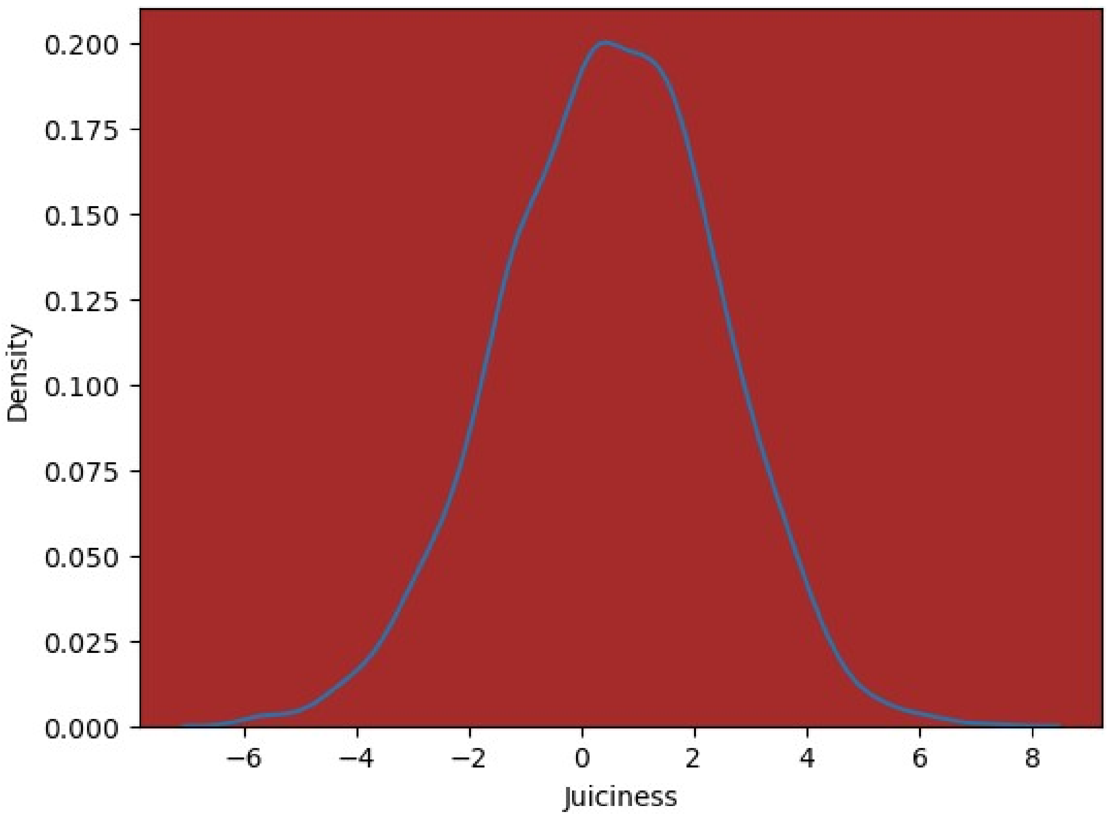

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



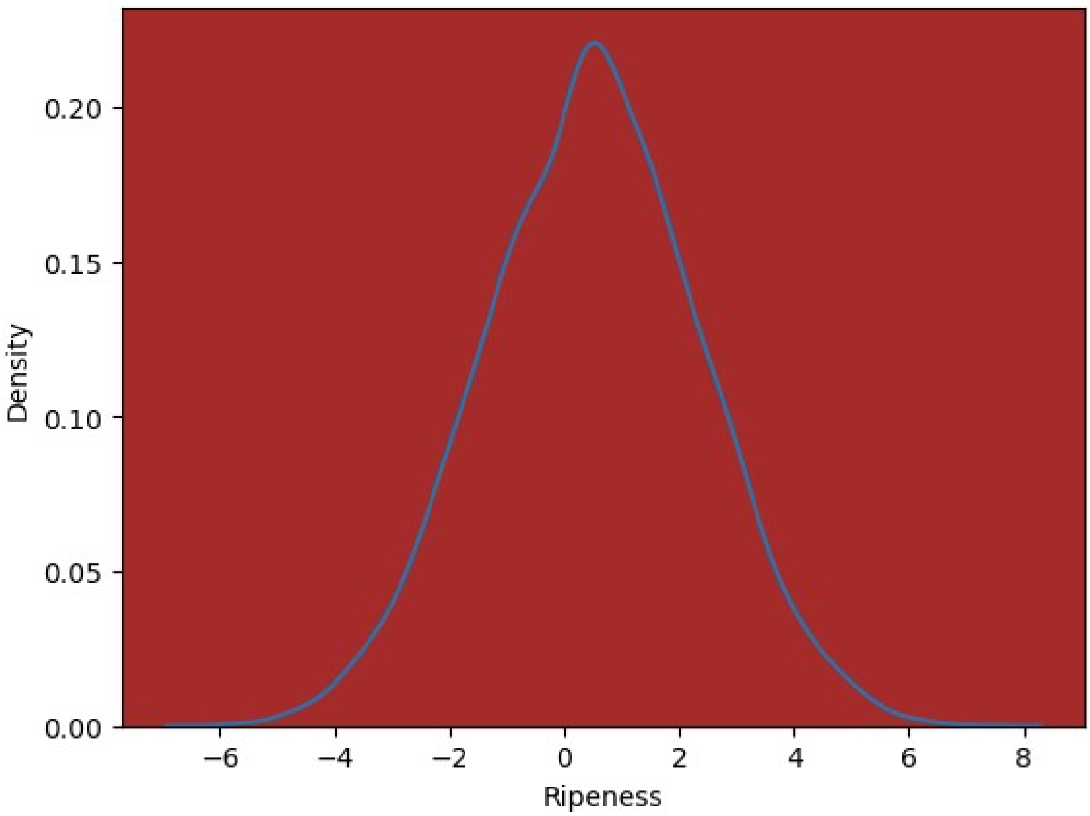

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



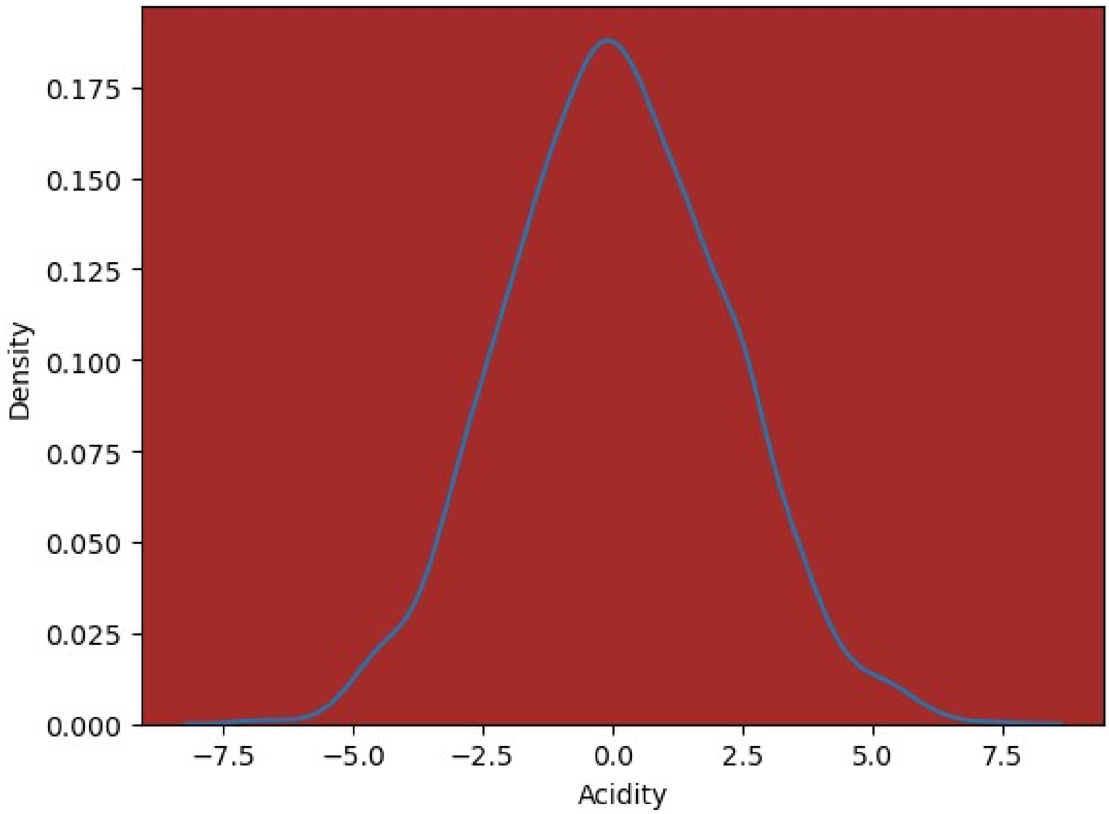

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

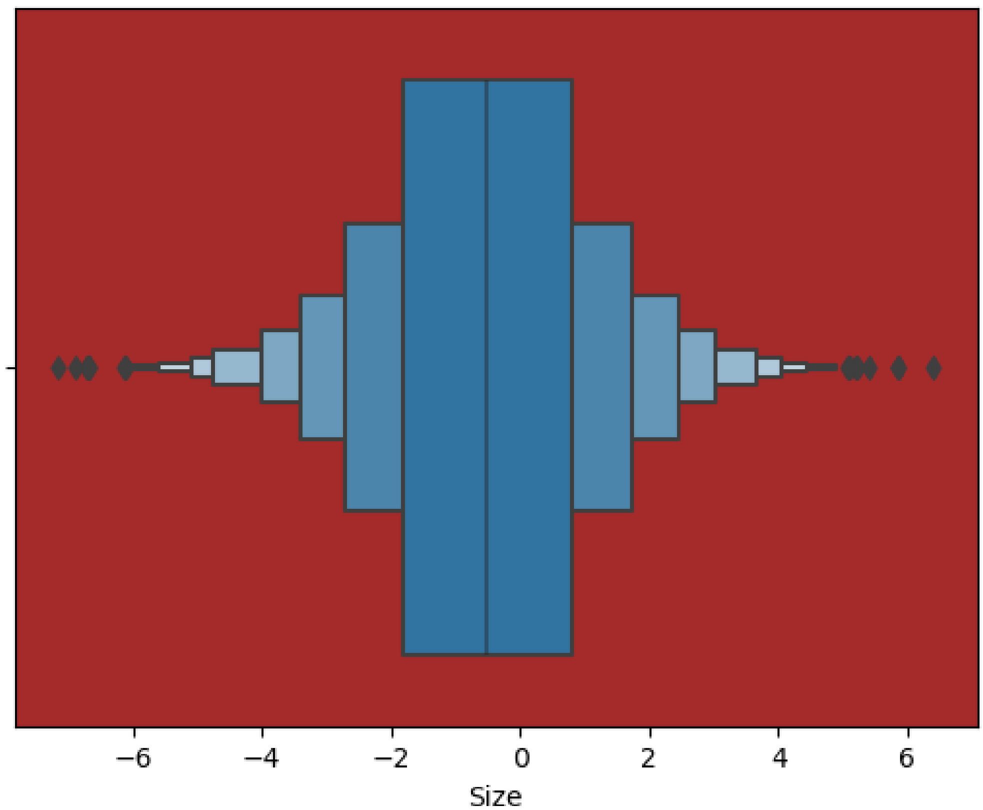



```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

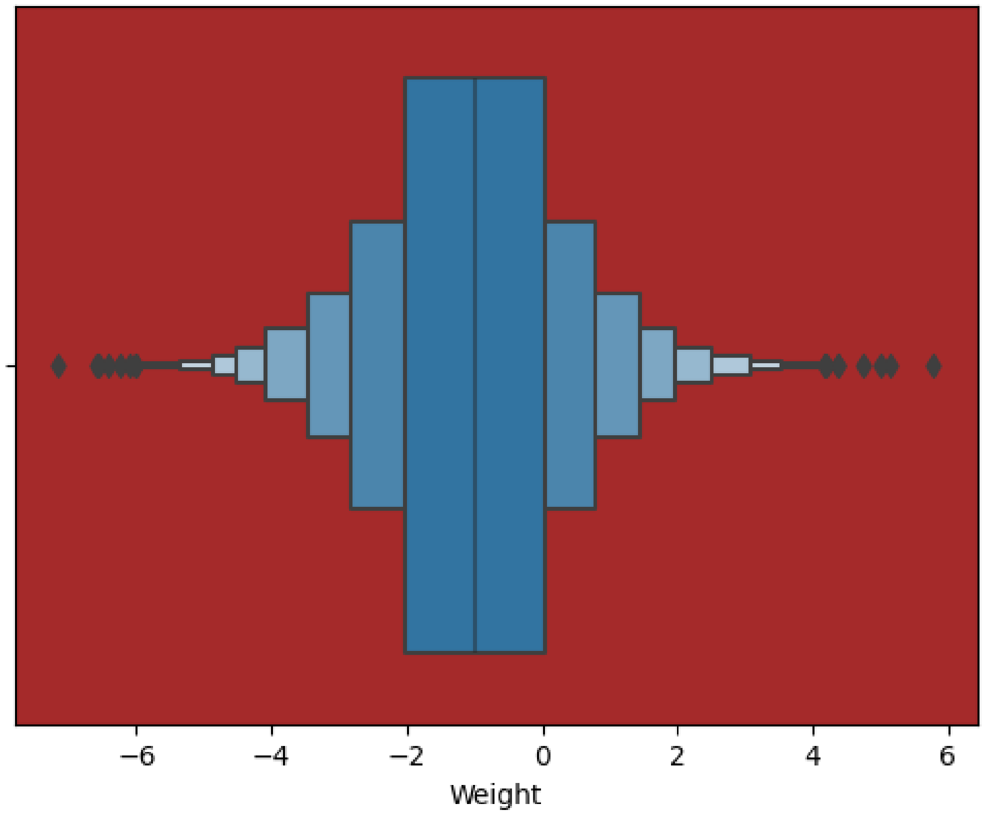



```
for i in
['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','Acidity']:
    sns.boxenplot(data=pk,x=i)
    plt.gca().set_facecolor('brown')
    plt.show()
```

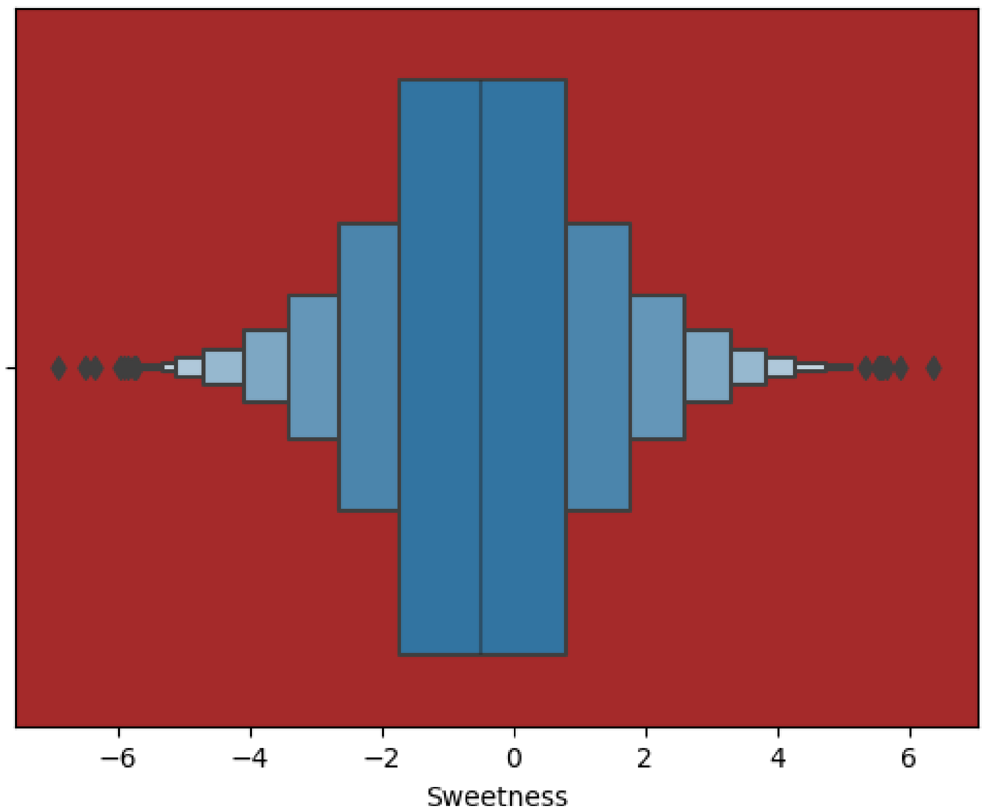
```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



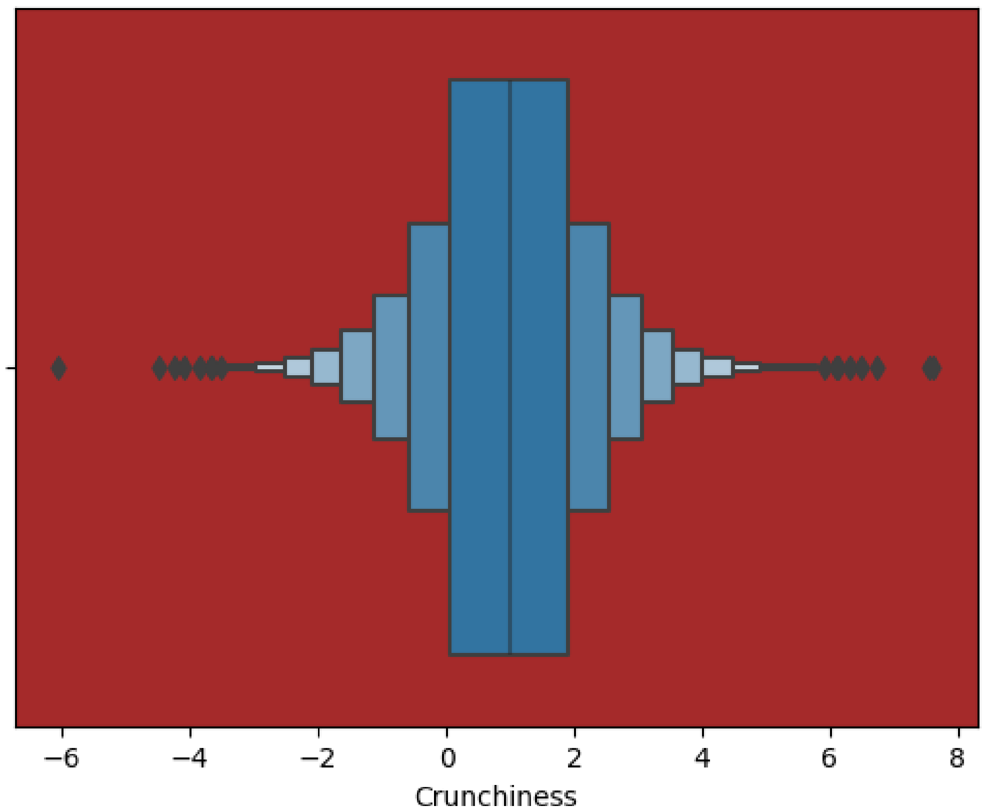

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



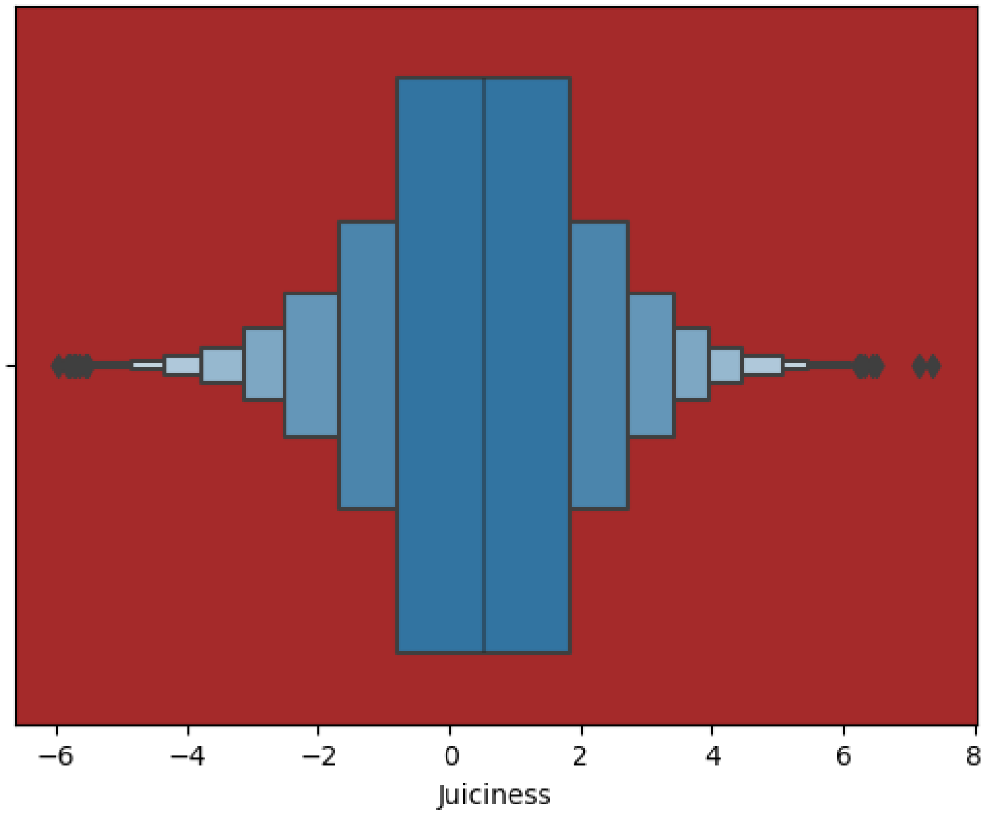

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



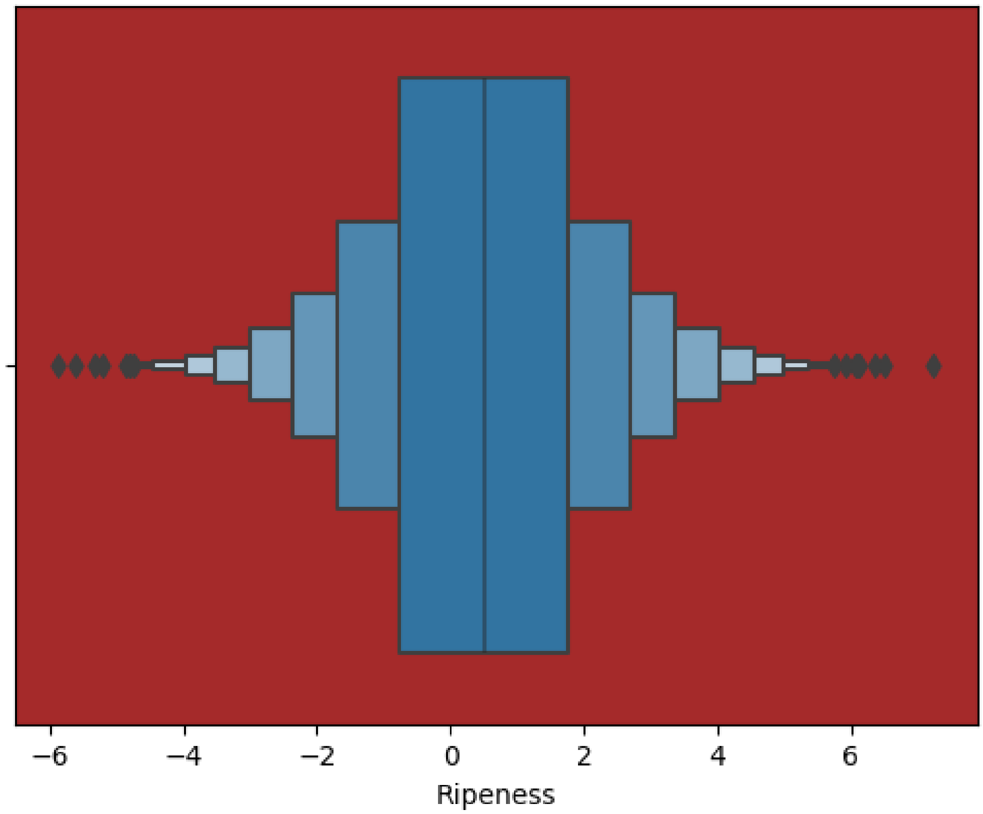

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



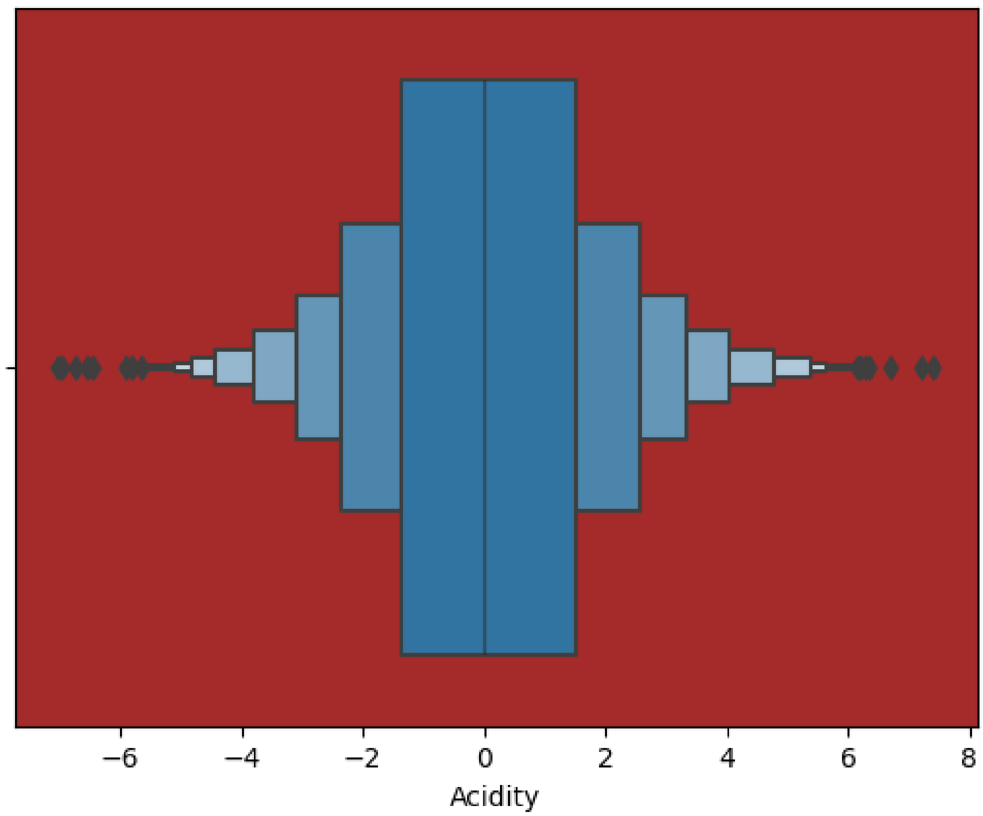

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

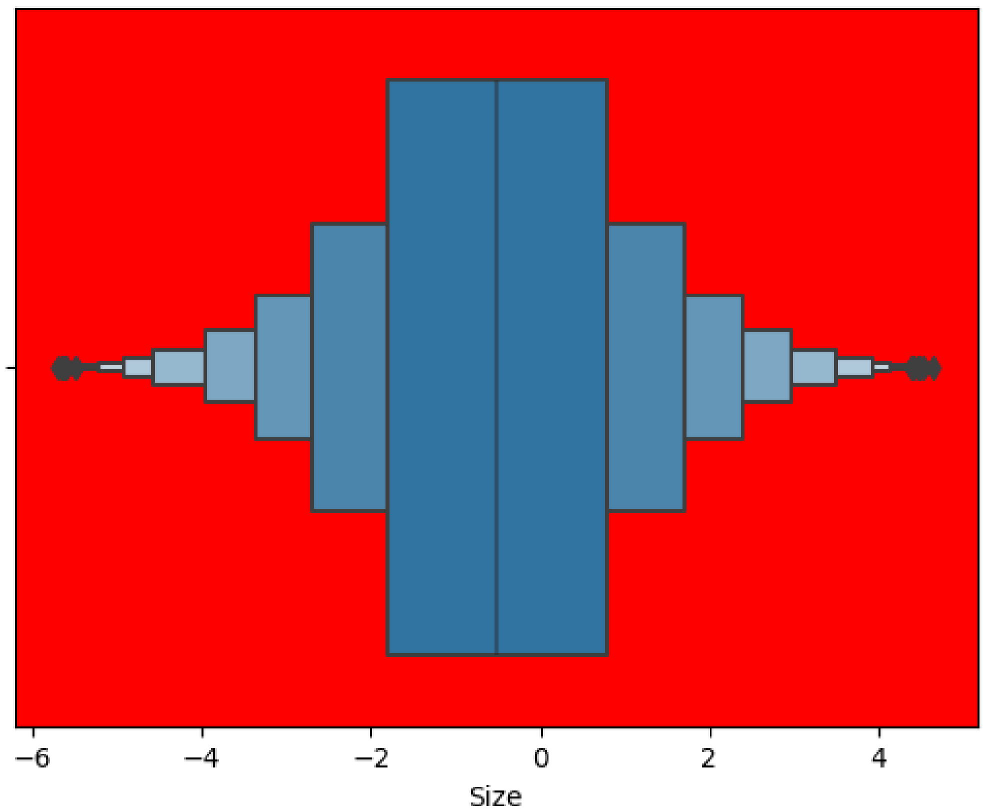



```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

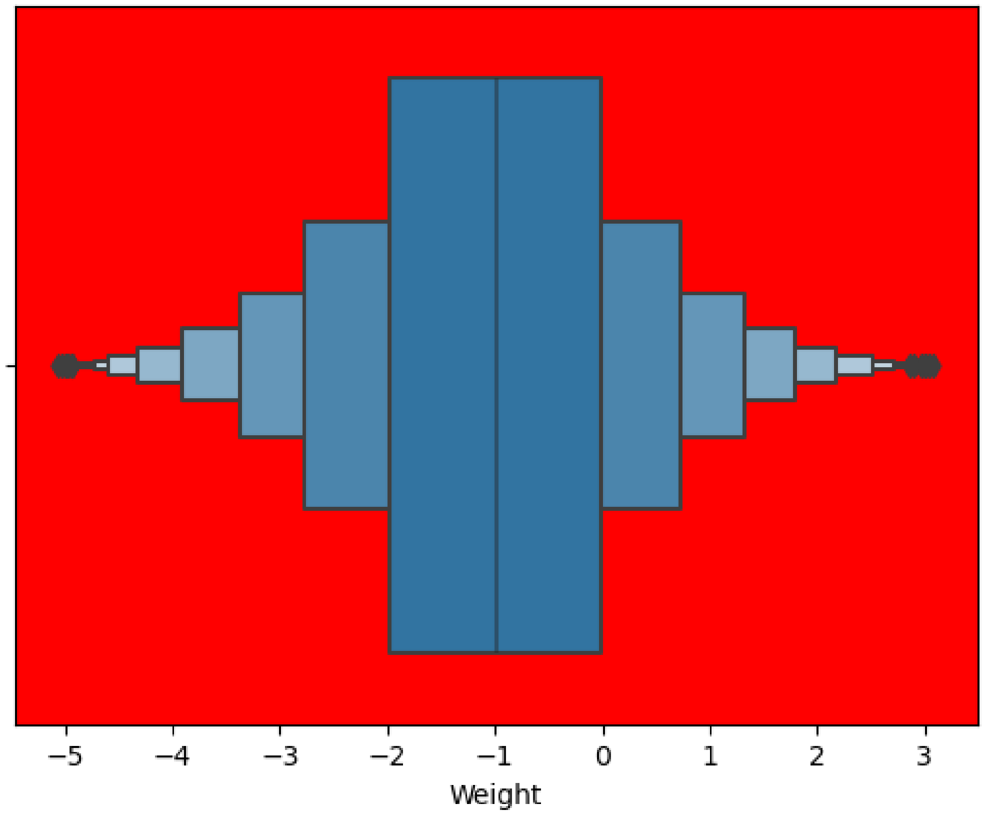



```
def treat_outlier(col):
    Q1=pk[col].quantile(0.25)
    Q3=pk[col].quantile(0.75)
    IQR=Q3-Q1
    UL=Q3+1.5*IQR
    LL=Q1-1.5*IQR
    upperotlier=pk[col]>UL
    loweroutlier=pk[col]<LL
    median=pk[col].median()
    pk.loc[upperotlier,col]=median
    pk.loc[loweroutlier,col]=median

for i
in['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','A
cidity']:
    treat_outlier(i)

for i
in['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','A
cidity']:
    sns.boxenplot(data=pk,x=i)
    plt.gca().set_facecolor('red')
    plt.show()
```

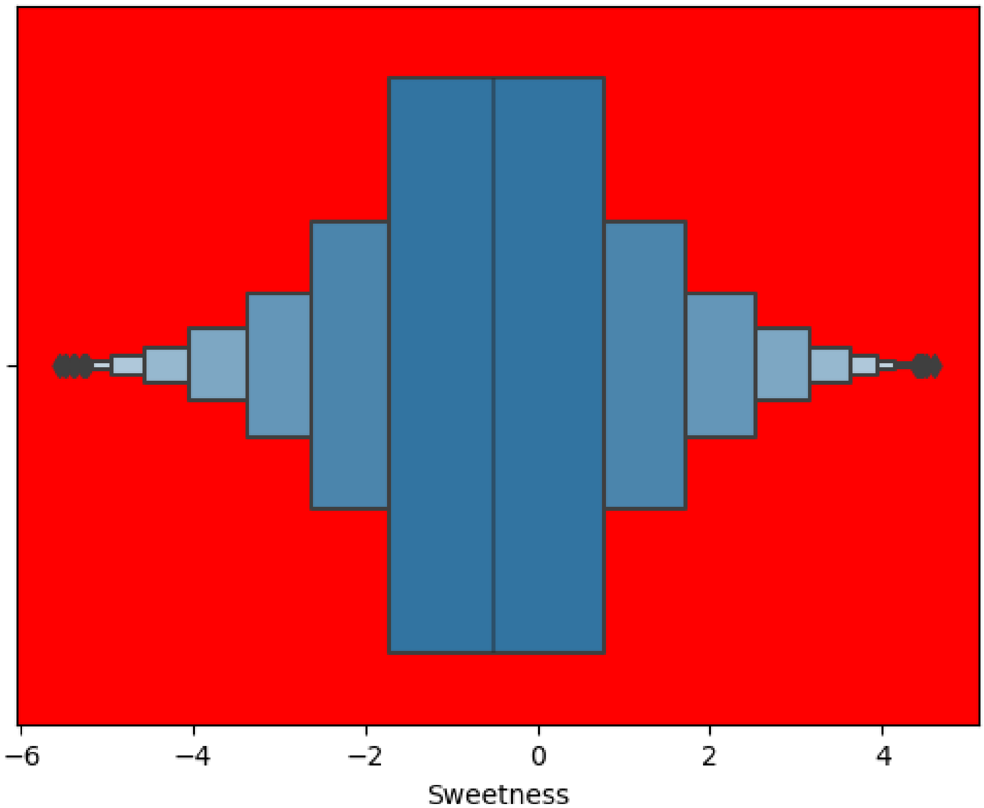
```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



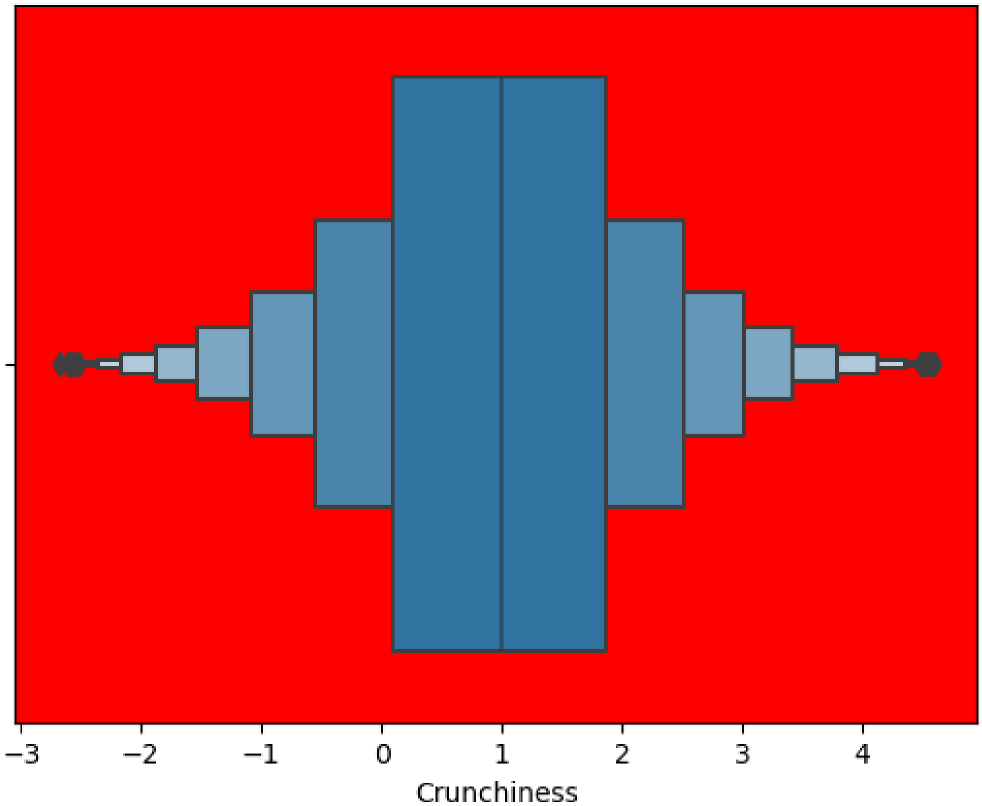

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



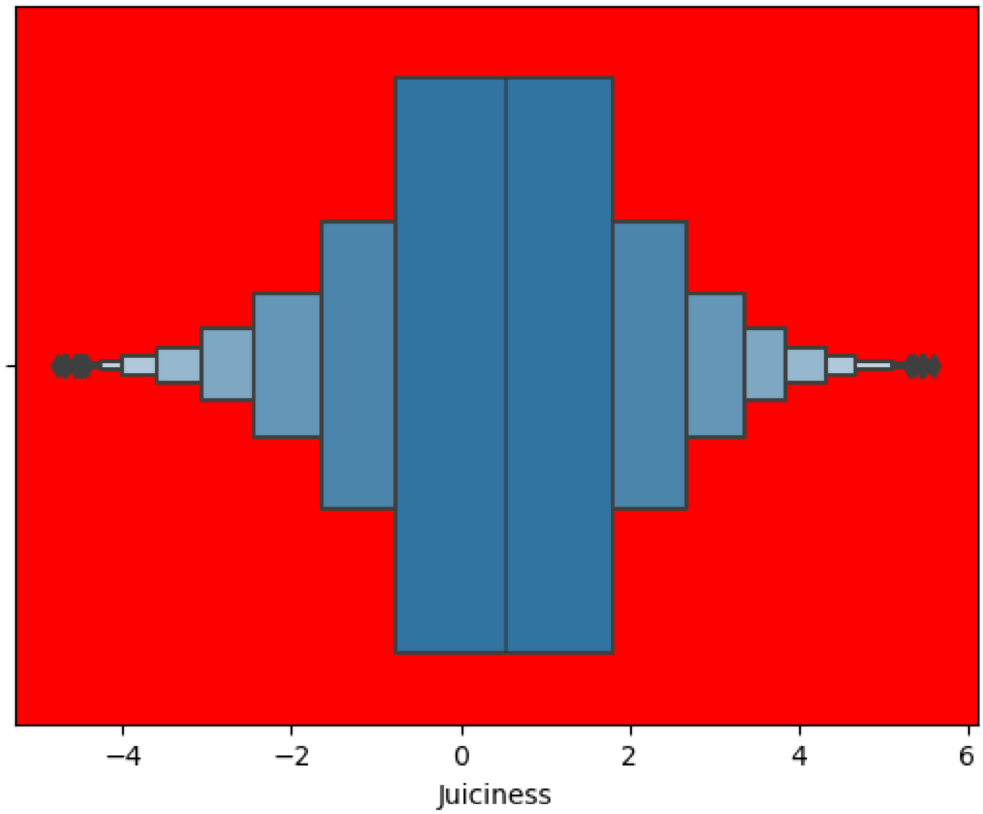

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



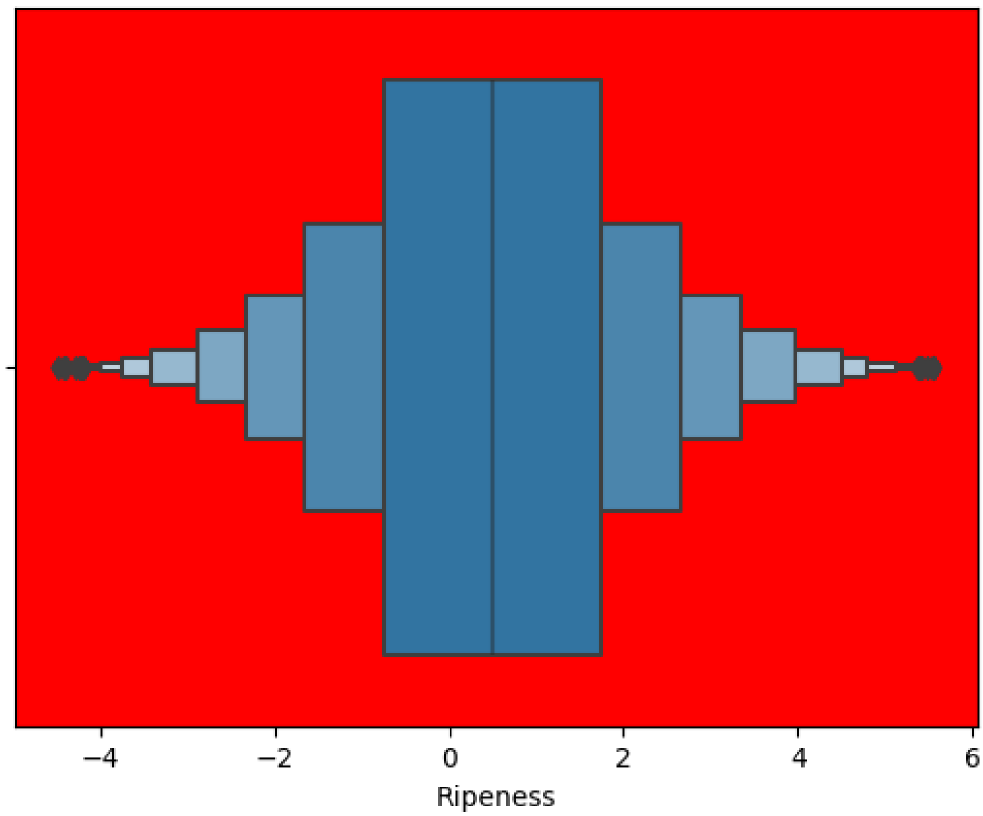

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



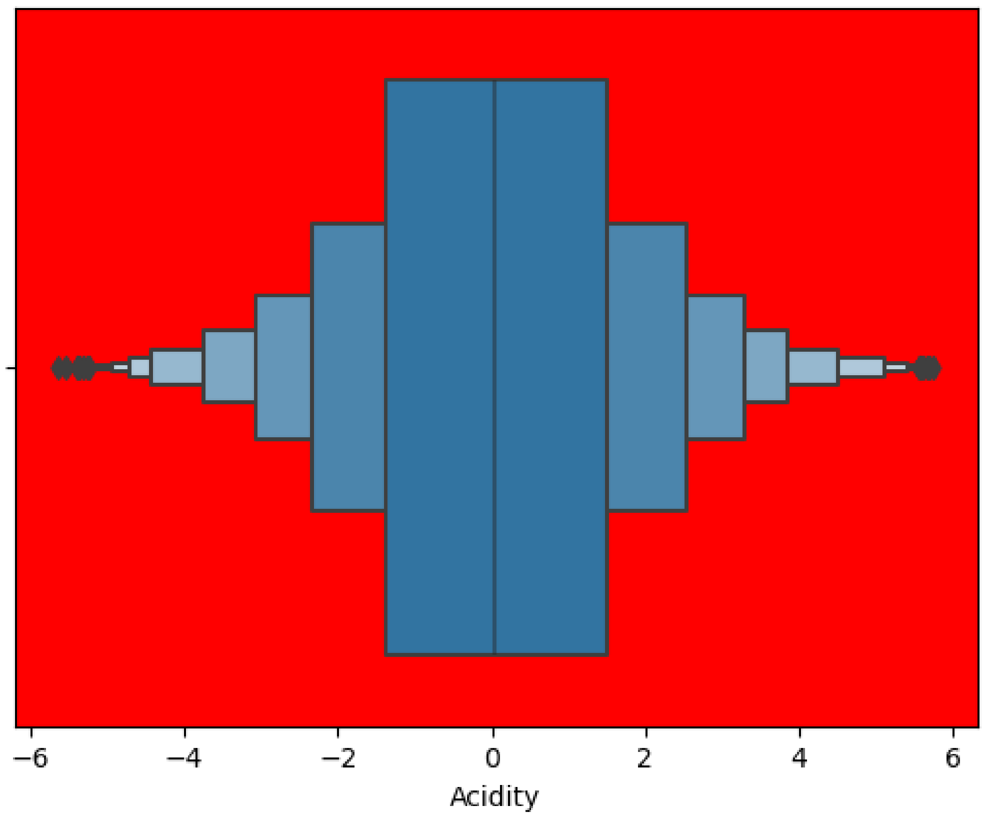

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\
categorical.py:1794: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```python
from scipy.stats import zscore

zscore(pk.select_dtypes(include=['float']))>+3
```

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
3995	False	False	False	False	False	False	False
3996	False	False	False	False	False	False	False
3997	False	False	False	False	False	False	False

```
3998  False   False      False        False      False     False
False
3999  False   False      False        False      False     False
False

[4000 rows x 7 columns]
```

```
zscore(pk.select_dtypes(include=['float']))<-3
```

```
       Size  Weight  Sweetness  Crunchiness  Juiciness  Ripeness
Acidity
0     False   False      False        False      False     False
False
1     False   False      False        False      False     False
False
2     False   False      False        False      False     False
False
3     False   False      False        False      False     False
False
4     False   False      False        False      False     False
False
...     ...     ...        ...          ...        ...       ...
...
3995  False   False      False        False      False     False
False
3996  False   False      False        False      False     False
False
3997  False   False      False        False      False     False
False
3998  False   False      False        False      False     False
False
3999  False   False      False        False      False     False
False

[4000 rows x 7 columns]
```

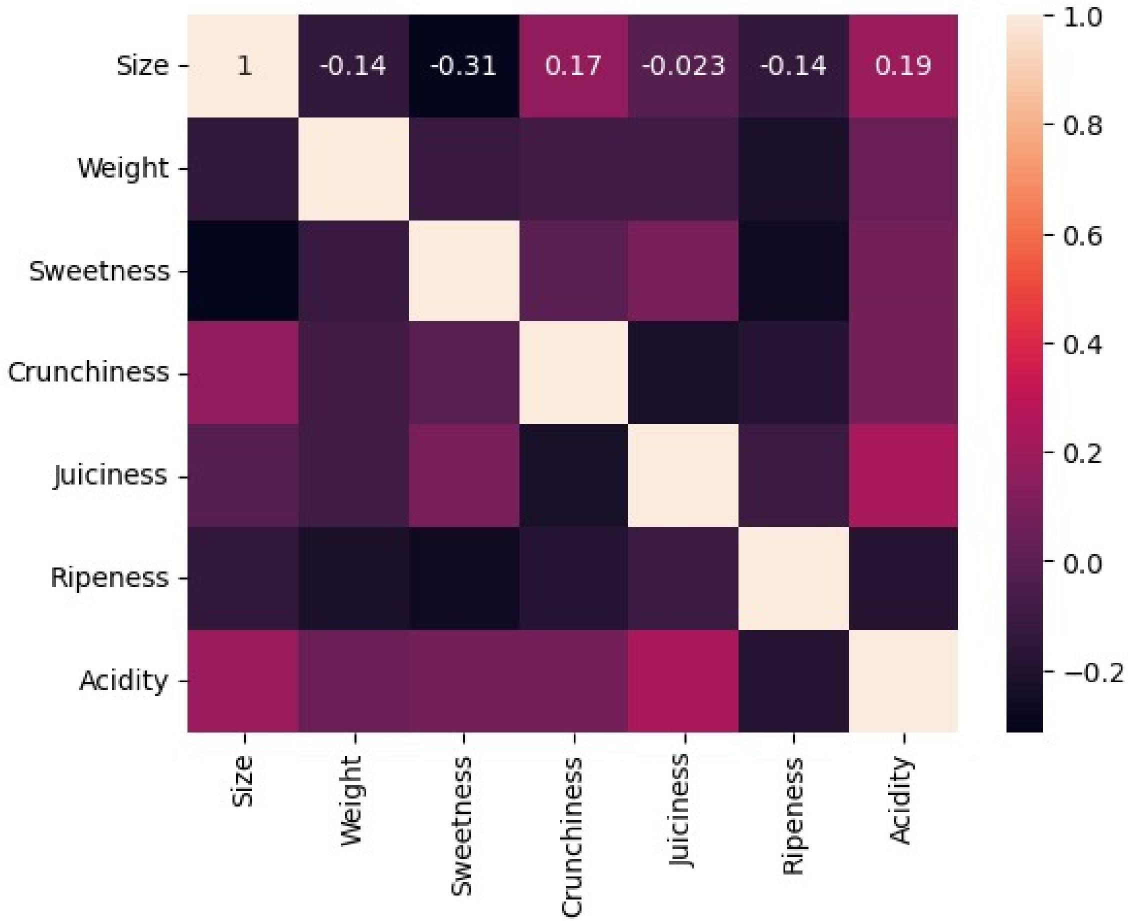
```
pk[['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','
Acidity']].corr()
```

```
                 Size    Weight  Sweetness  Crunchiness  Juiciness
Ripeness  \
Size         1.000000 -0.140180  -0.312955     0.165760  -0.022888 -
0.139821
Weight      -0.140180  1.000000  -0.120500    -0.086807  -0.090456 -
```

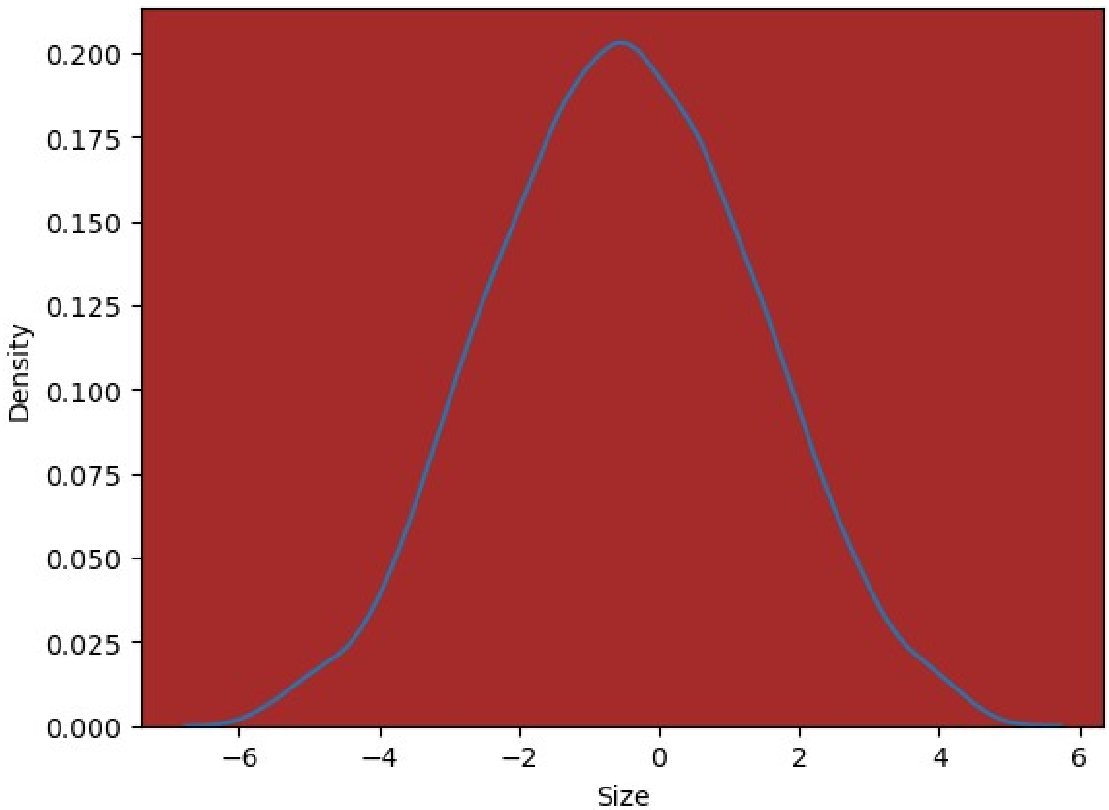
```
0.221947
Sweetness   -0.312955 -0.120500   1.000000    -0.014191   0.089395 -
0.258363
Crunchiness  0.165760 -0.086807  -0.014191     1.000000  -0.227767 -
0.181666
Juiciness   -0.022888 -0.090456   0.089395    -0.227767   1.000000 -
0.108158
Ripeness    -0.139821 -0.221947  -0.258363    -0.181666  -0.108158
1.000000
Acidity      0.192140  0.039086   0.071963     0.077810   0.234223 -
0.188371

              Acidity
Size         0.192140
Weight       0.039086
Sweetness    0.071963
Crunchiness  0.077810
Juiciness    0.234223
Ripeness    -0.188371
Acidity      1.000000
```

```python
sns.heatmap(pk[['Size','Weight','Sweetness','Crunchiness','Juiciness',
'Ripeness','Acidity']].corr(),annot=True)
```

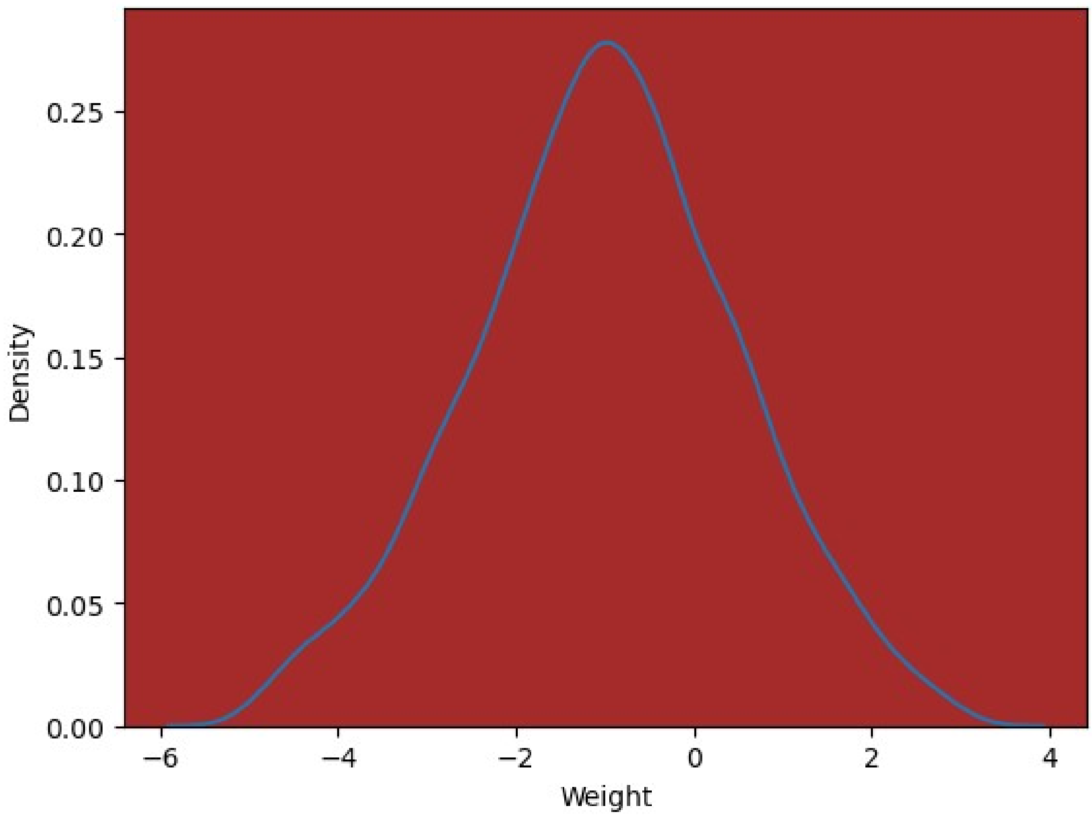
```
<Axes: >
```



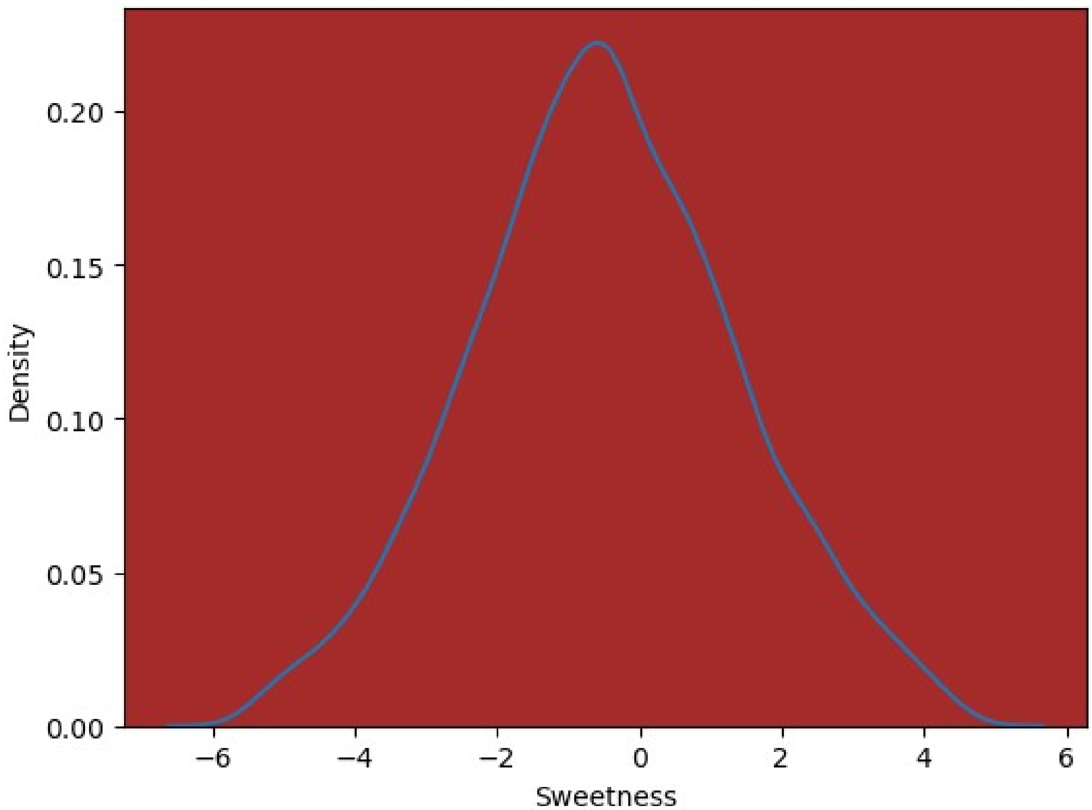

```
for i in
['Size','Weight','Sweetness','Crunchiness','Juiciness','Ripeness','Aci
dity']:
    sns.kdeplot(data=pk,x=i)
    plt.gca().set_facecolor('brown')
    plt.show()

C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



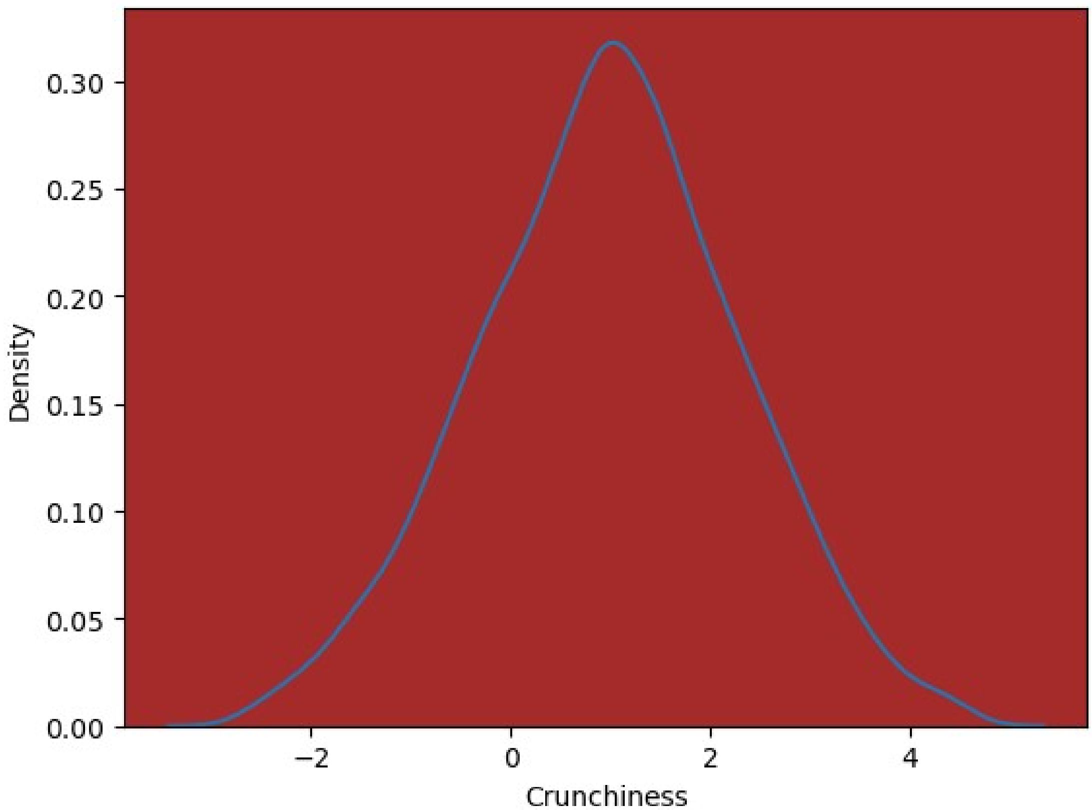

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



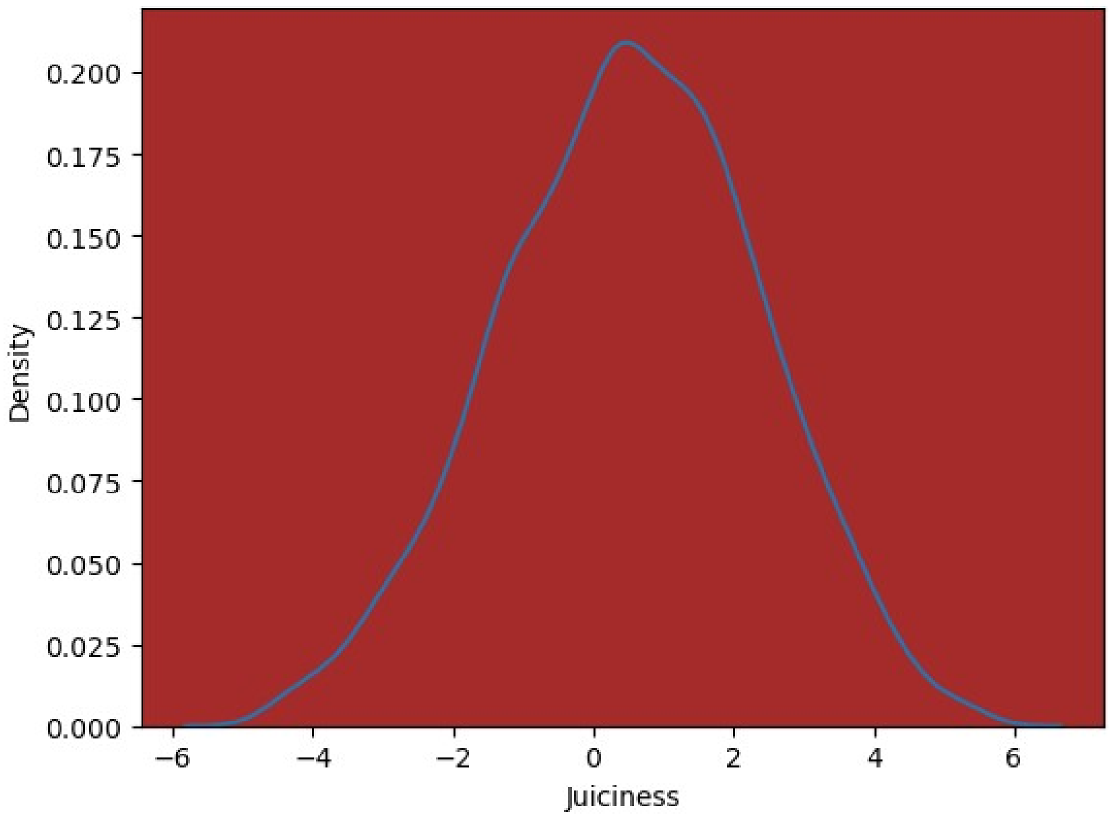

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



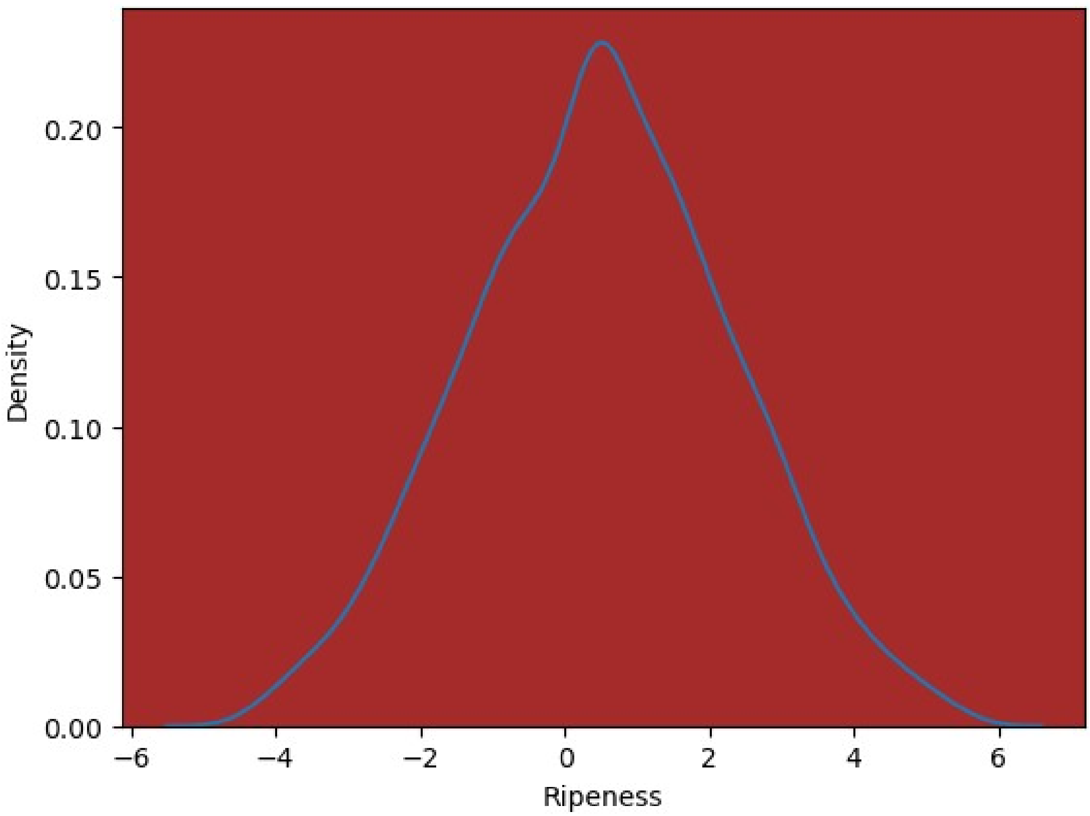

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



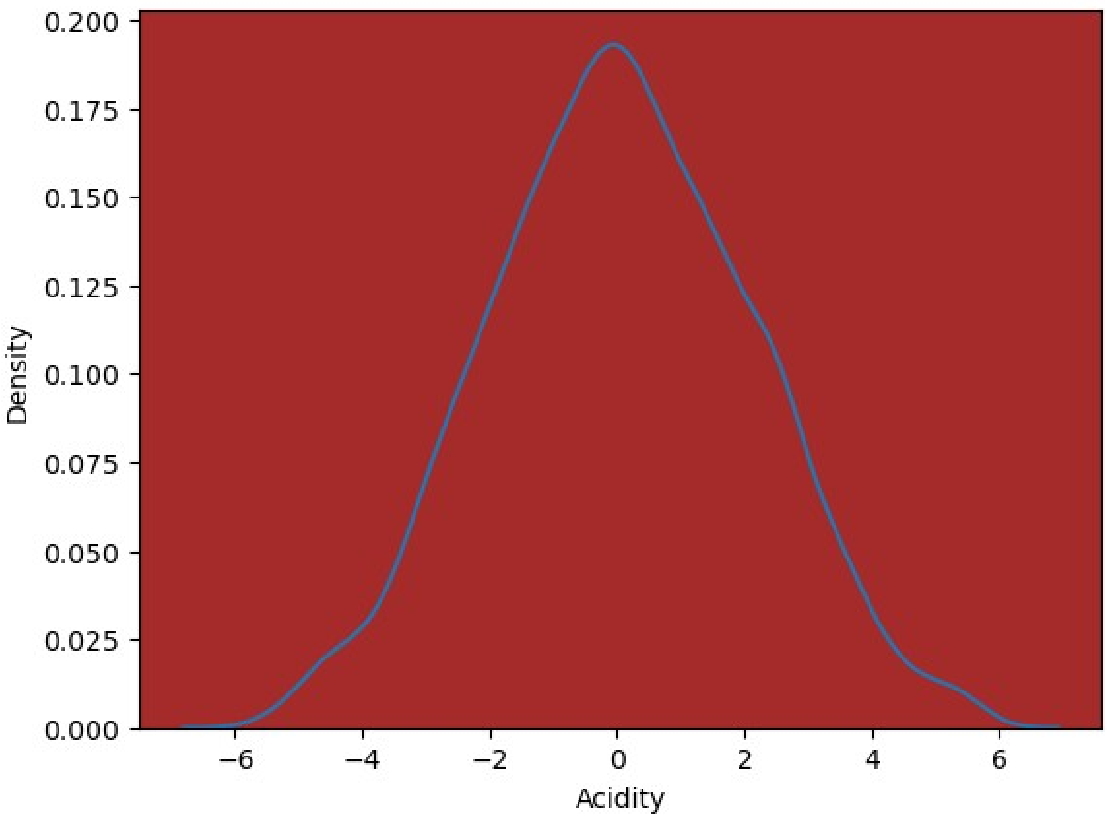

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
pk

      A_id      Size    Weight  Sweetness  Crunchiness  Juiciness
Ripeness  \
0        0 -3.970049 -2.512336  -0.504758    -1.012009   1.844900
0.329840
1        1 -1.195217 -2.839257   3.664059     1.588232   0.853286
0.867530
2        2 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -
0.038033
3        3 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -
3.413761
4        4  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -
1.303849
...    ...       ...       ...        ...          ...        ...
...
3995  3995  0.059386 -1.067408  -3.714549     0.473052   1.697986
2.244055
3996  3996 -0.293118  1.949253  -0.204020    -0.640196   0.024523 -
1.087900
3997  3997 -2.634515 -2.138247  -2.440461     0.657223   2.199709
4.763859
3998  3998 -4.008004 -1.779337   2.366397    -0.200329   2.161435
0.214488
```

```
3999  3999  0.278540 -1.715505   0.121217    -1.154075   1.266677 -
0.776571

       Acidity Quality
0    -0.491590    good
1    -0.722809    good
2     2.621636     bad
3     0.790723    good
4     0.501984    good
...        ...     ...
3995  0.137784     bad
3996  1.854235    good
3997 -1.334611     bad
3998 -2.229720    good
3999  1.599796    good

[4000 rows x 9 columns]
```

```
pk.isnull().sum()
```

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

```
from sklearn.preprocessing import LabelEncoder,PowerTransformer

LE=LabelEncoder()

pk['Quality']=LE.fit_transform(pk['Quality'])

pk['Quality']
```

```
0       1
1       1
2       0
3       1
4       1
       ..
3995    0
3996    1
3997    0
3998    1
3999    1
Name: Quality, Length: 4000, dtype: int32
```

```
pk=pk.drop('A_id',axis=True)

PT=PowerTransformer()

x=pk.drop('Quality',axis=1)

y=pk['Quality']

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,rand
om_state=42)

from sklearn.linear_model import LogisticRegression

LR=LogisticRegression()

LR.fit(x_train,y_train)

LogisticRegression()

model_pred=LR.predict(x_test)

y_test

555     1
3491    0
527     0
3925    1
2989    0
       ..
1922    0
865     1
3943    1
1642    1
2483    1
Name: Quality, Length: 800, dtype: int32

LR.score(x_train,y_train)

0.7334375

LR.score(x_test,y_test)

0.74375

from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,f1_score

accuracy_score(model_pred,y_test)

0.74375

confusion_matrix(model_pred,y_test)
```

```
array([[297, 101],
       [104, 298]], dtype=int64)
```

```
print(classification_report(model_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.74      0.75      0.74       398
           1       0.75      0.74      0.74       402

    accuracy                           0.74       800
   macro avg       0.74      0.74      0.74       800
weighted avg       0.74      0.74      0.74       800
```

```
pd.DataFrame(y_test).value_counts()
```

```
Quality
0          401
1          399
Name: count, dtype: int64
```

```
pd.DataFrame(model_pred).value_counts()
```

```
1    402
0    398
Name: count, dtype: int64
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
KNN=KNeighborsClassifier(n_neighbors=7)
```

```
KNN.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```
KNN_pred=KNN.predict(x_test)
```

```
y_test
```

```
555     1
3491    0
527     0
3925    1
2989    0
       ..
1922    0
865     1
3943    1
1642    1
2483    1
Name: Quality, Length: 800, dtype: int32
```

```
accuracy_score(y_test,KNN_pred)
```

```
0.89
```

```
confusion_matrix(y_test,KNN_pred)
```

```
array([[359,  42],
       [ 46, 353]], dtype=int64)
```

```
print(classification_report(y_test,KNN_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.90      0.89       401
           1       0.89      0.88      0.89       399

    accuracy                           0.89       800
   macro avg       0.89      0.89      0.89       800
weighted avg       0.89      0.89      0.89       800
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
DTC=DecisionTreeClassifier(max_depth=31)
```

```
DTC.fit(x_train,y_train)
```

```
DecisionTreeClassifier(max_depth=31)
```

```
DTC_pred=DTC.predict(x_test)
```

```
y_test
```

```
555     1
3491    0
527     0
3925    1
2989    0
       ..
1922    0
865     1
3943    1
1642    1
2483    1
Name: Quality, Length: 800, dtype: int32
```

```
accuracy_score(DTC_pred,y_test)
```

```
0.81125
```

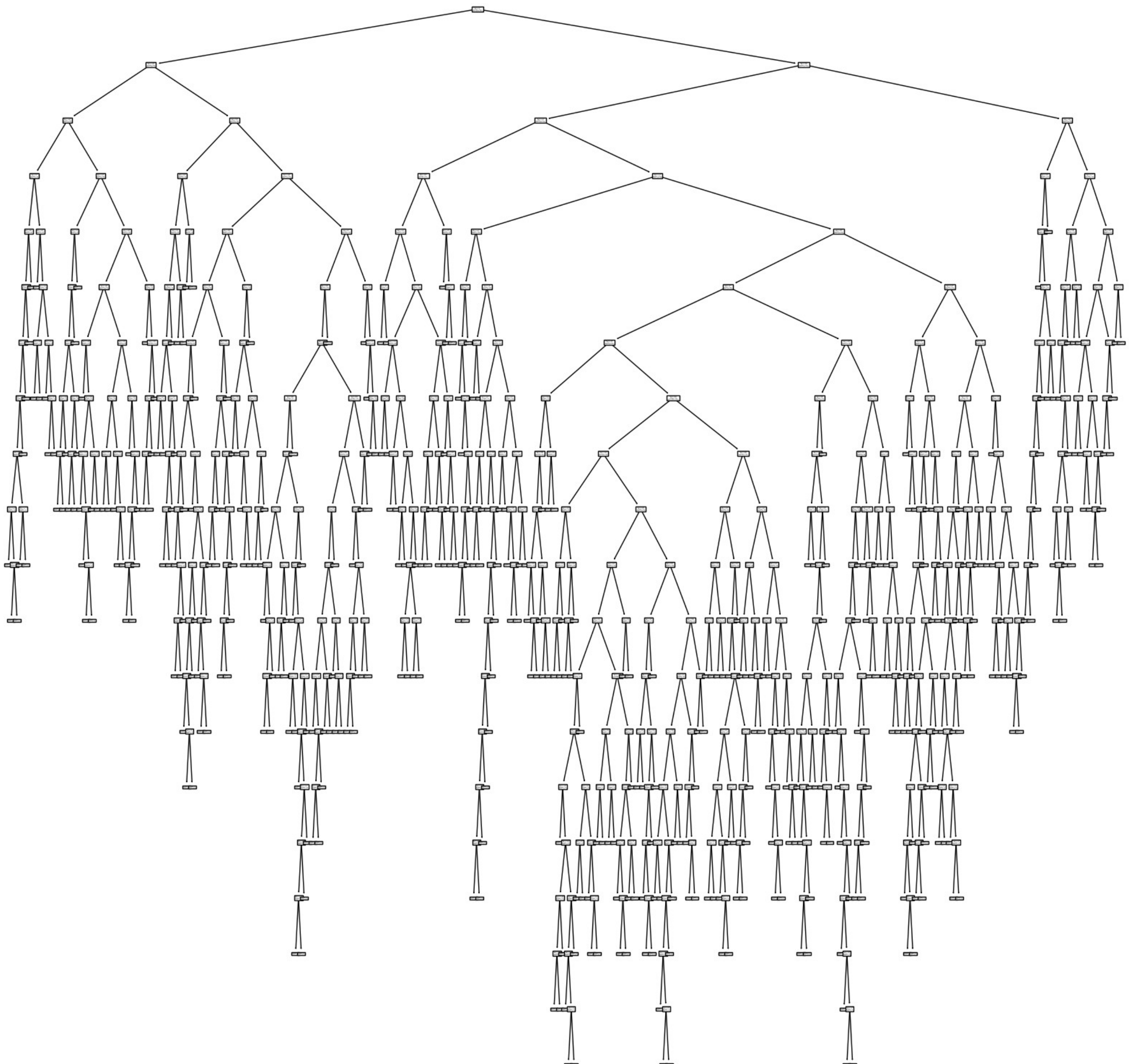
```
print(classification_report(y_test,DTC_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.82      0.81       401
           1       0.82      0.80      0.81       399

    accuracy                           0.81       800
   macro avg       0.81      0.81      0.81       800
weighted avg       0.81      0.81      0.81       800
```

```
confusion_matrix(y_test,DTC_pred)
```

```
array([[330,  71],
       [ 80, 319]], dtype=int64)
```

```
from sklearn import tree
plt.figure(figsize=(20,20))
tree.plot_tree(DTC)
plt.show()
```



```
from sklearn.ensemble import RandomForestClassifier

RFC=RandomForestClassifier(n_estimators=89)

RFC.fit(x_train,y_train)

RandomForestClassifier(n_estimators=89)

RFC_pred=RFC.predict(x_test)

y_test

555     1
3491    0
```

```
527     0
3925    1
2989    0
       ..
1922    0
865     1
3943    1
1642    1
2483    1
Name: Quality, Length: 800, dtype: int32
```

```
accuracy_score(y_test,RFC_pred)
```

```
0.89375
```

```
confusion_matrix(y_test,RFC_pred)
```

```
array([[356,  45],
       [ 40, 359]], dtype=int64)
```

```
print(classification_report(y_test,RFC_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.89      0.89       401
           1       0.89      0.90      0.89       399

    accuracy                           0.89       800
   macro avg       0.89      0.89      0.89       800
weighted avg       0.89      0.89      0.89       800
```

```
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV
```

```
parameters={'n_estimators':
[1,7,9,11,19,21,25,29,31,51,71,89,99,101,119,201,251],'criterion':
['gini','entropy'],
            'max_depth':range(1,10)}
```

```
RSCV=RandomizedSearchCV(estimator=RFC,scoring='accuracy',param_distrib
utions=parameters,cv=10)
```

```
RSCV.fit (x_train,y_train)
```

```
RandomizedSearchCV(cv=10,
estimator=RandomForestClassifier(n_estimators=89),
                   param_distributions={'criterion': ['gini',
'entropy'],
                                        'max_depth': range(1, 10),
                                        'n_estimators': [1, 7, 9, 11,
19, 21,
```

```
                                                         25, 29, 31,
51, 71, 89,
                                                         99, 101, 119,
201,
                                                         251]},
                   scoring='accuracy')
```

```
RSCV.best_params_
```

```
{'n_estimators': 119, 'max_depth': 9, 'criterion': 'entropy'}
```

```
RSCV.best_score_
```

```
0.8474999999999999
```

```
GSCV=GridSearchCV(estimator=RFC,scoring='accuracy',param_grid=paramete
rs,cv=10)
```

```
GSCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=10, estimator=RandomForestClassifier(n_estimators=89),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(1, 10),
                         'n_estimators': [1, 7, 9, 11, 19, 21, 25, 29,
31, 51,
                                          71, 89, 99, 101, 119, 201,
251]},
             scoring='accuracy')
```

```
GSCV.best_params_
```

```
{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 89}
```

```
GSCV.best_score_
```

```
0.8534375000000001
```