Banana Quality

In [76]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

#Encoding
import sklearn.preprocessing
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
#Traing and testing
import sklearn.linear_model
from sklearn.model_selection import train_test_split
#Development
from sklearn.linear_model import LinearRegression
linear_regression_model =LinearRegression()
#Evaluation
import sklearn.metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,r2_score,f1_score,recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [38]:
```
data =pd.read_csv("C:/Users/Oooba/Desktop/Analysis with pyhton/banana_quality.csv")
data
```

Out[38]:

	Size	Weight	Sweetness	Softness	HarvestTime	Ripeness	Acidity	Quality
0	-1.924968	0.468078	3.077832	-1.472177	0.294799	2.435570	0.271290	Good
1	-2.409751	0.486870	0.346921	-2.495099	-0.892213	2.067549	0.307325	Good
2	-0.357607	1.483176	1.568452	-2.645145	-0.647267	3.090643	1.427322	Good
3	-0.868524	1.566201	1.889605	-1.273761	-1.006278	1.873001	0.477862	Good
4	0.651825	1.319199	-0.022459	-1.209709	-1.430692	1.078345	2.812442	Good
...	...	...	...	...	...	...	...	...
7995	-6.414403	0.723565	1.134953	2.952763	0.297928	-0.156946	2.398091	Bad
7996	0.851143	-2.217875	-2.812175	0.489249	-1.323410	-2.316883	2.113136	Bad
7997	1.422722	-1.907665	-2.532364	0.964976	-0.562375	-1.834765	0.697361	Bad
7998	-2.131904	-2.742600	-1.008029	2.126946	-0.802632	-3.580266	0.423569	Bad
7999	-2.660879	-2.044666	0.159026	1.499706	-1.581856	-1.605859	1.435644	Bad

8000 rows × 8 columns

In [39]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8000 entries, 0 to 7999
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Size         8000 non-null   float64
 1   Weight       8000 non-null   float64
 2   Sweetness    8000 non-null   float64
 3   Softness     8000 non-null   float64
 4   HarvestTime  8000 non-null   float64
 5   Ripeness     8000 non-null   float64
 6   Acidity      8000 non-null   float64
 7   Quality      8000 non-null   object 
dtypes: float64(7), object(1)
memory usage: 500.1+ KB
```

In [40]:
```
data.isnull().sum()
```

Out[40]:
```
Size           0
Weight         0
Sweetness      0
Softness       0
HarvestTime    0
Ripeness       0
Acidity        0
Quality        0
dtype: int64
```

In [41]:
```
data.describe()
```

Out[41]:

	Size	Weight	Sweetness	Softness	HarvestTime	Ripeness	Acidity
count	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000
mean	-0.747802	-0.761019	-0.770224	-0.014441	-0.751288	0.781098	0.008725
std	2.136023	2.015934	1.948455	2.065216	1.996661	2.114289	2.293467
min	-7.998074	-8.283002	-6.434022	-6.959320	-7.570008	-7.423155	-8.226977
25%	-2.277651	-2.223574	-2.107329	-1.590458	-2.120659	-0.574226	-1.629450
50%	-0.897514	-0.868659	-1.020673	0.202644	-0.934192	0.964952	0.098735
75%	0.654216	0.775491	0.311048	1.547120	0.507326	2.261650	1.682063
max	7.970800	5.679692	7.539374	8.241555	6.293280	7.249034	7.411633

In [42]:

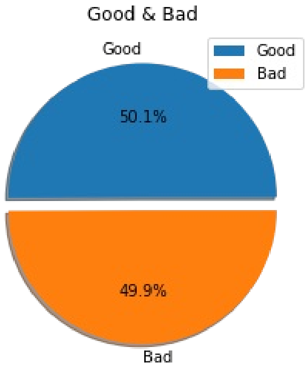
```
data_Quality=data["Quality"].value_counts()
data_Quality
```

Out[42]:

```
Good     4006
Bad      3994
Name: Quality, dtype: int64
```

In [67]:

```
plt.pie(data_Quality ,labels=['Good', 'Bad'], autopct='%1.1f%%', explode=[0,0.1],shadow=True)
plt.title('Good & Bad')
plt.legend()
plt.show()
```



Dependent and Independent

In [44]:

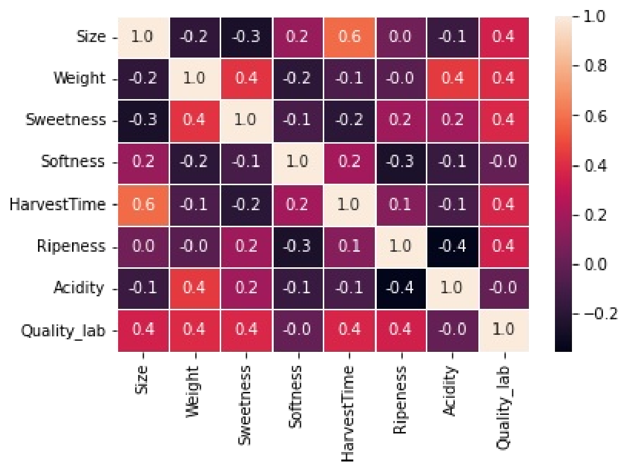
```
data["Quality_lab"]=le.fit_transform(data["Quality"])
```

In [45]:

```
data_corr= data.corr()
sns.heatmap(data_corr,annot=True,fmt="0.1f",linewidth=0.5)
```

Out[45]:

```
<AxesSubplot:>
```

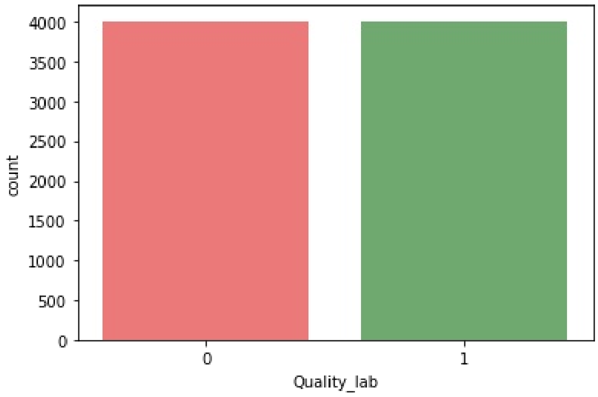


In [65]:

```
sns.countplot(x="Quality_lab",data=data,palette=["r","g"],alpha=0.6)
```

Out[65]:

```
<AxesSubplot:xlabel='Quality_lab', ylabel='count'>
```



Splitting and Traning Testing

In [82]:
```
x= data[["Size","Weight","Sweetness","Softness","HarvestTime","Ripeness","Acidity"]]
y= data["Quality_lab"]
```

In [47]:
```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=42)
x_train.shape #80%
```

Out[47]:
```
(6400, 7)
```

Model Development and predection and Error

In [70]:
```
model=linear_regression_model.fit(x_train,y_train)
model
```

Out[70]:
```
LinearRegression()
```

In [78]:
```
y_prede=model.predict(x_test)
y_error= y_test-y_prede
predection=pd.DataFrame({"Actual":y_test,"predicted":y_prede,"Error":y_error})
predection["abs_error"]=abs(predection["Error"])
mean_absolut_error=predection["abs_error"].mean()
predection.head(10)
```

Out[78]:

	Actual	predicted	Error	abs_error
2215	0	0.484022	-0.484022	0.484022
2582	0	0.571736	-0.571736	0.571736
1662	1	0.962904	0.037096	0.037096
3027	0	0.530962	-0.530962	0.530962
4343	1	0.382106	0.617894	0.617894
2680	0	0.551756	-0.551756	0.551756
1765	1	0.596448	0.403552	0.403552
1123	1	0.813579	0.186421	0.186421
4054	1	0.909351	0.090649	0.090649
3761	0	-0.083207	0.083207	0.083207

Model Accuracy and Evaluation

In [50]:
```
r2_score(y_test,y_prede)
print(f"Accuracy of the model={round(r2_score(y_test,y_prede)*100)}%")
```

```
Accuracy of the model=57%
```

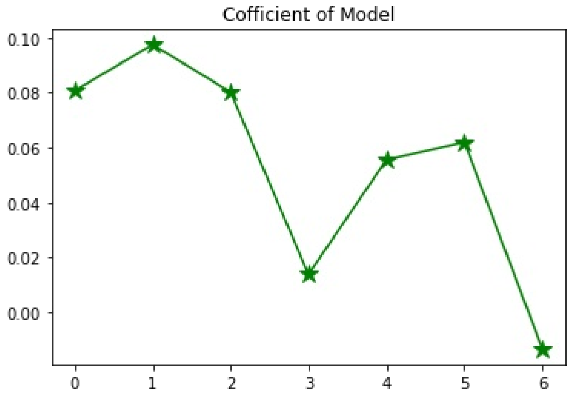
In [58]:
```
print("Root Mean Squared Error (RMSE)=",mean_absolut_error**(0.5))
```

```
Root Mean Squared Error (RMSE)= 0.515631529137344
```

cofficients

In [95]:
```
model_cof=model.coef_
plt.plot(model_cof,color="g",marker="*",markersize=12)
plt.title("Cofficient of Model")
```

Out[95]:
```
Text(0.5, 1.0, 'Cofficient of Model')
```



In [96]:
```
I=model.intercept_
print(f"intercept of the model={round(I*100)}%")
```

```
intercept of the model=69%
```

In [97]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js