Predicting Apple Quality Documentation

Aim: The aim of the project is to predict the quality of the apple whether it is good or bad using the inputs – size, weight, sweetness, crunchiness, juiciness, ripeness, acidity, quality.

Importing all the required libraries:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import emoji
import nltk
from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,LancasterStemmer,SnowballStemmer,wordnet
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
import pickle
import scipy.stats as ss
from sklearn.naive_bayes import GaussianNB,MultinomialNB
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,precision_score,recall_score,
        f1_score,classification_report,roc_auc_score,roc_curve
```

- Import all the libraries as shown in above figure.
- Using the libraries like sklearn, nltk, pickle etc.,

Importing the dataset:

```
data = pd.read_csv(r"C:\Users\srava\Downloads\archive (7)\apple_quality.csv")
```

- Importing the dataset and assigning a variable to it as ‘data’.

```
data
```

	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
...	...	...	...	...	...	...	...	...	...
3996	3996.0	-0.293118	1.949253	-0.204020	-0.640196	0.024523	-1.087900	1.854235285	good
3997	3997.0	-2.634515	-2.138247	-2.440461	0.657223	2.199709	4.763859	-1.334611391	bad
3998	3998.0	-4.008004	-1.779337	2.366397	-0.200329	2.161435	0.214488	-2.229719806	good
3999	3999.0	0.278540	-1.715505	0.121217	-1.154075	1.266677	-0.776571	1.599796456	good
4000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Created_by_Nidula_Elgiriyewithana	NaN

4001 rows × 9 columns

- Data has 4001 rows and 9 columns.

```
data.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
```

- Data has two columns in object data and seven columns as float64.
- Data has one null value in the ‘acidity’ column.

Finding duplicate values and removing them:

```
# dropping null row
data.drop(4000,axis=0,inplace=True)
```

```
data['Acidity']= data['Acidity'].astype(float)
```

- Converted the Acidity column into float after removing the null value.

Dividing data into Feature variables and class variable

```
fv = data.iloc[:,1:-1]
cv = data.iloc[:,-1]
```

- Splatted the data into feature and class variables.

Splitting the data into training and testing

```
x_train,x_test,y_train,y_test = train_test_split(fv,cv,test_size=0.2,random_state=3,stratify=cv)
```

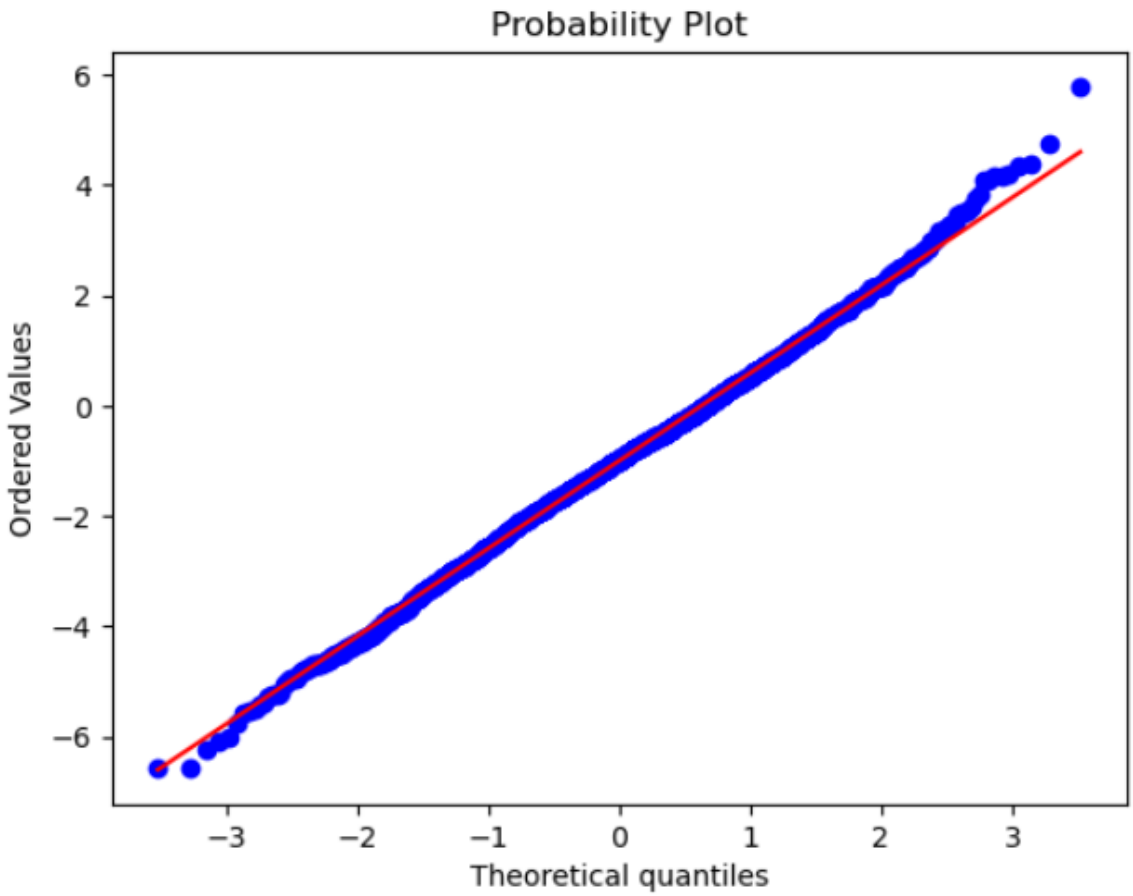
- Divided the data into train and test.

Visualizing the data

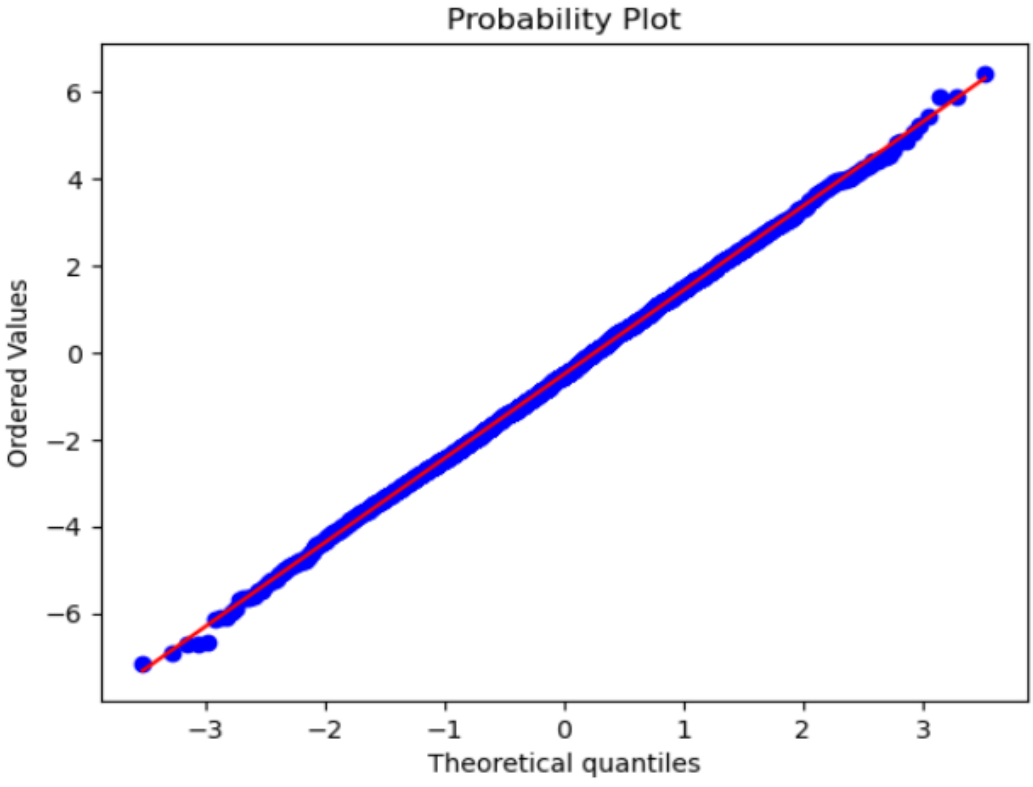
```
for y in x_train.columns:
    plt.subplot(111)
    ss.probplot(x_train[y],dist='norm',fit=True,plot=plt)
    print(y)
    plt.show()
```

- Checking the data either it is liner or nonlinear using probplot.

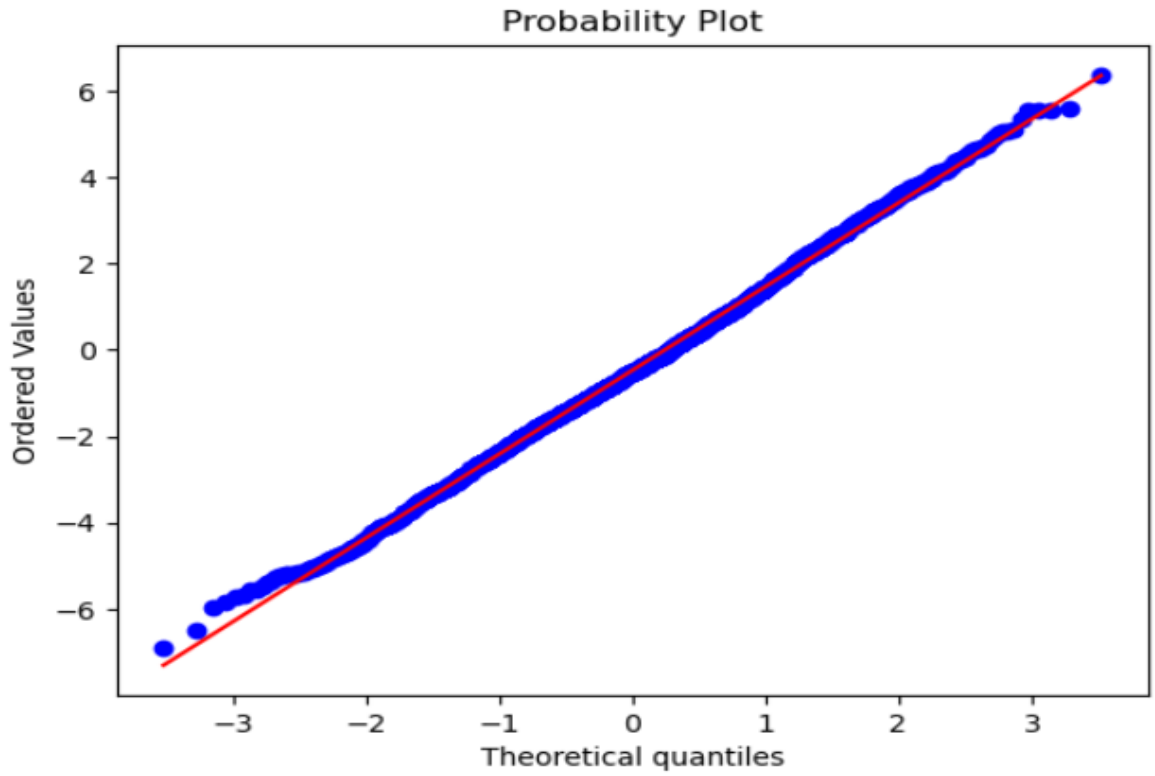
Weight


Probability Plot
Ordered Values
Theoretical quantiles
6
4
2
0
−2
−4
−6
−3
−2
−1
0
1
2
3

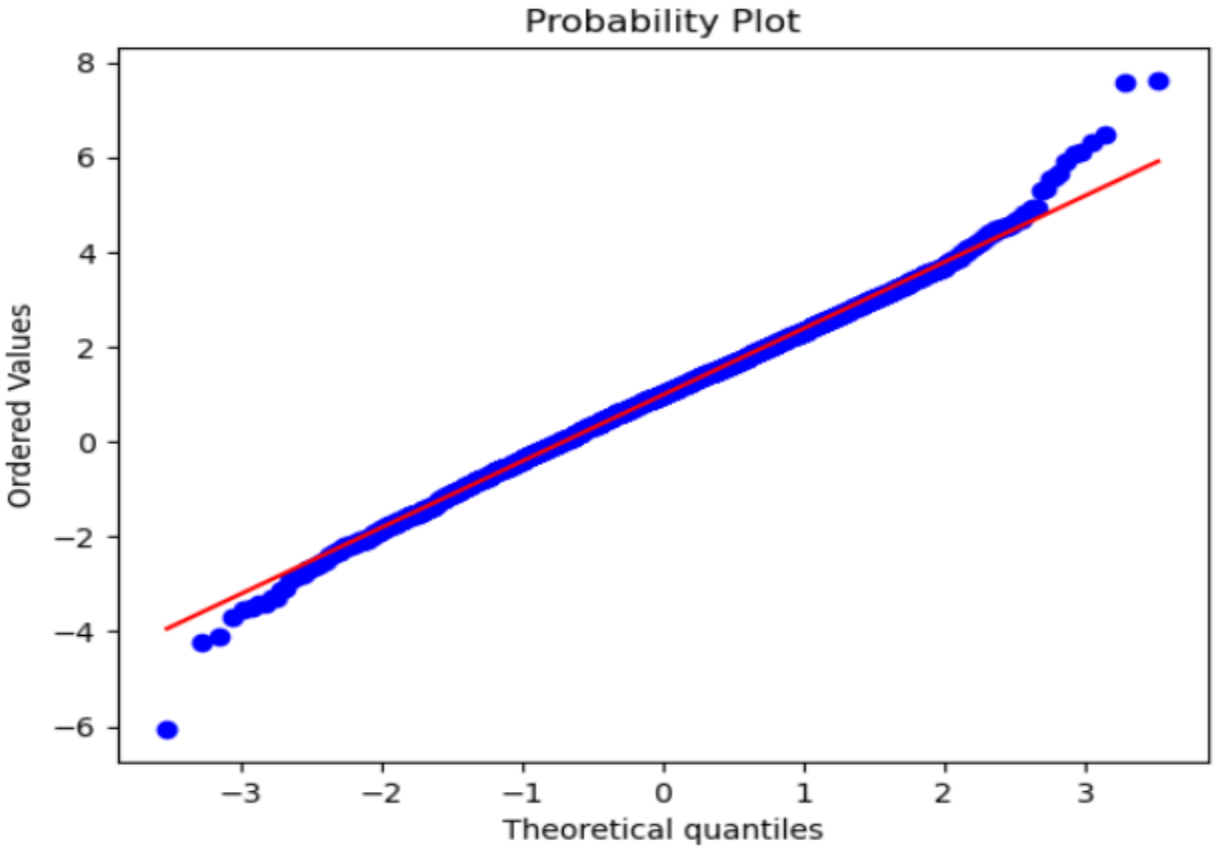
Size


Probability Plot
Ordered Values
Theoretical quantiles
6
4
2
0
−2
−4
−6
−3
−2
−1
0
1
2
3

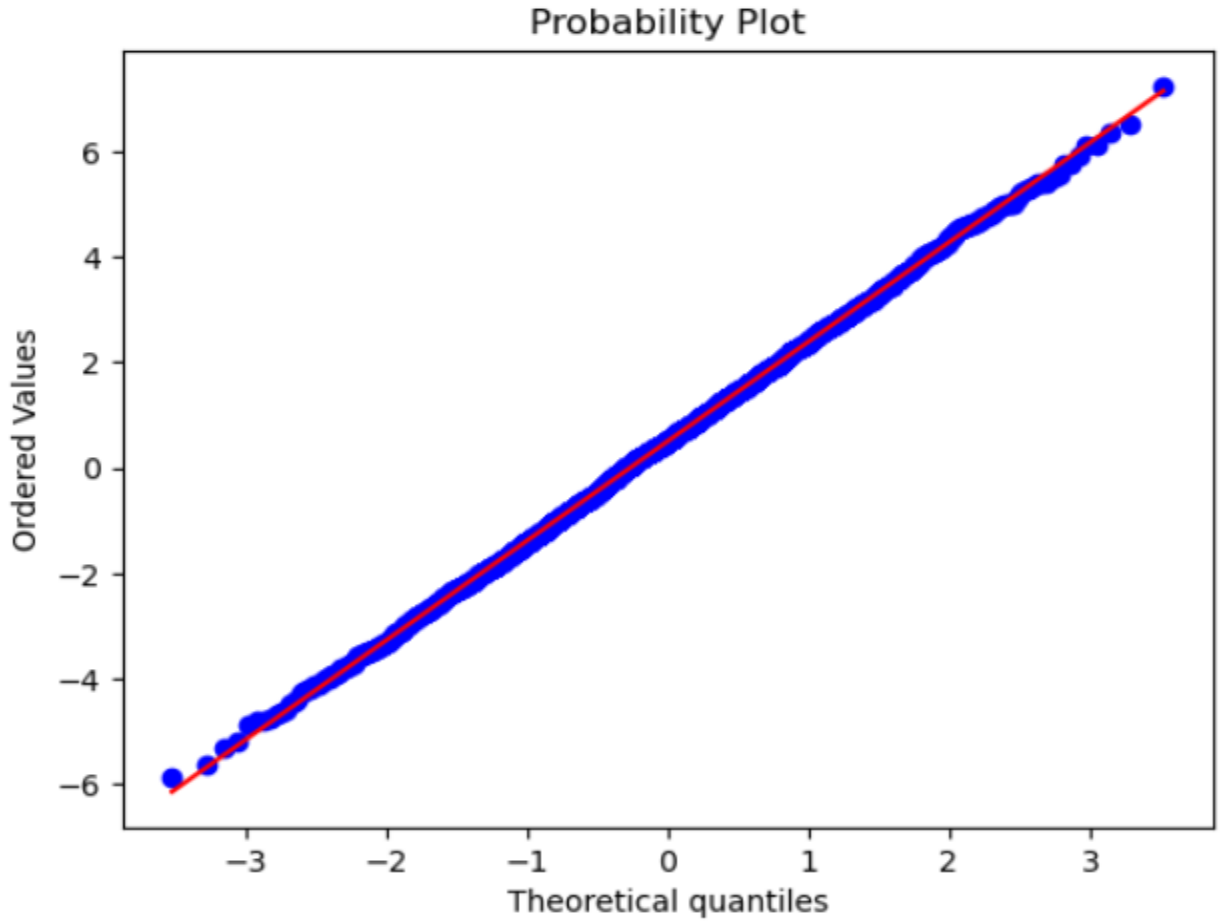
Sweetness



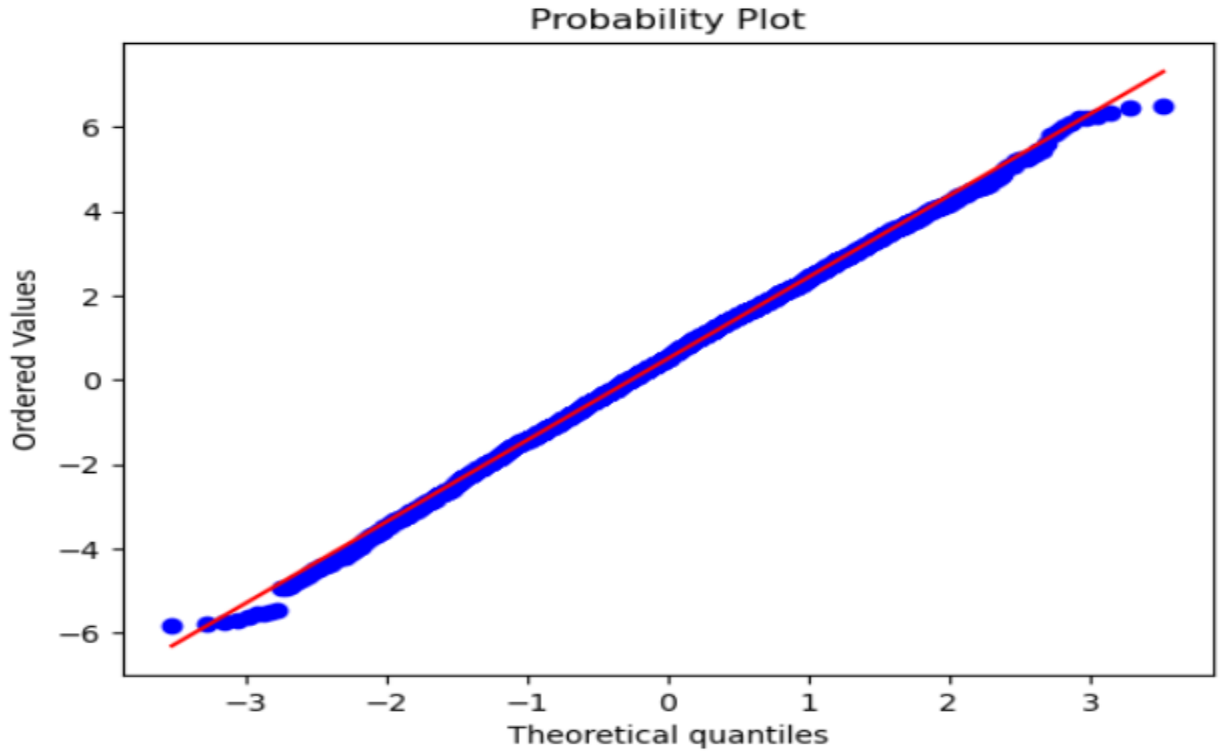
Crunchiness



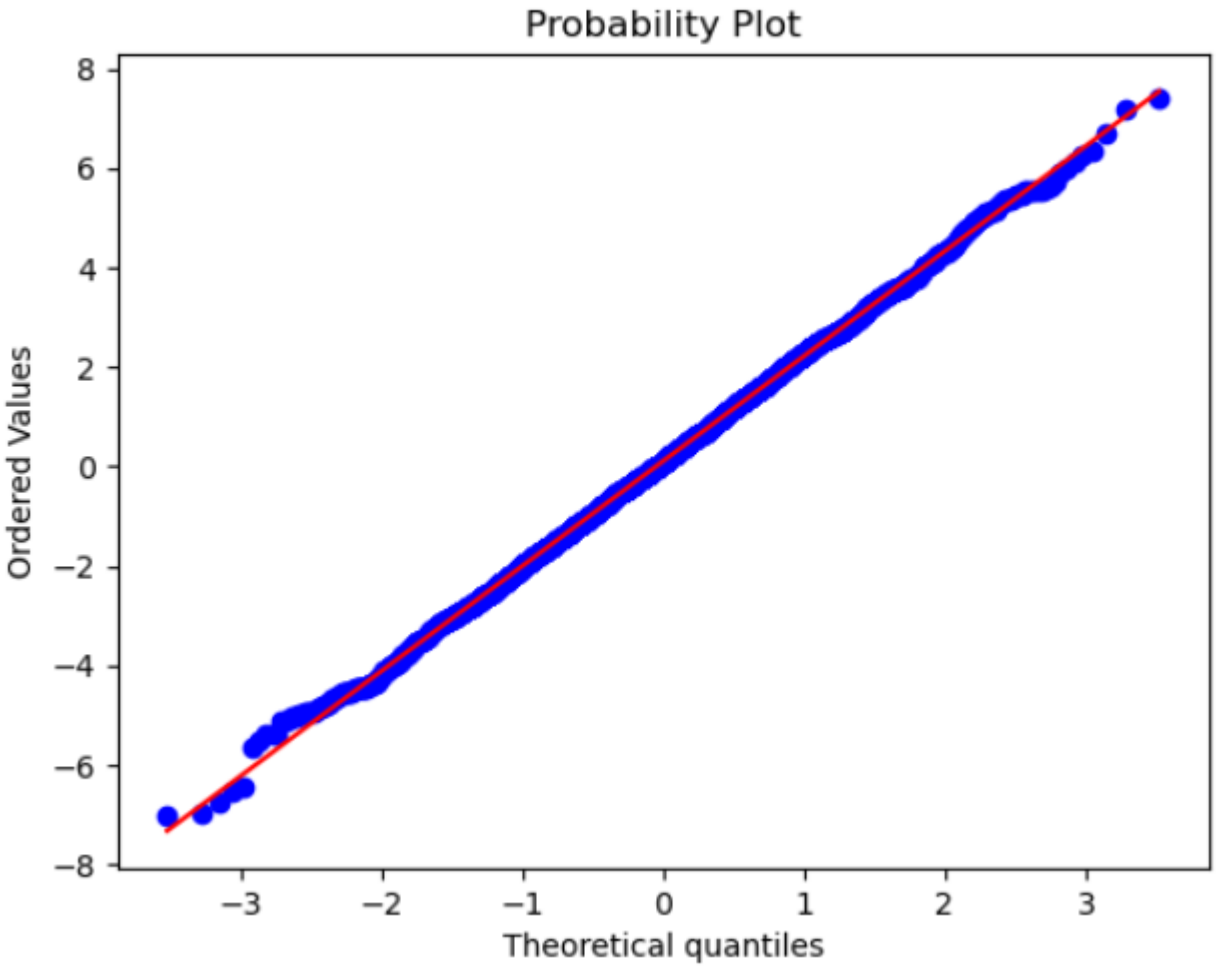
Ripeness



Juiciness



```
Acidity
```



- Almost all the columns are having linear data.

```
gb = GaussianNB()
```

```
model = gb.fit(x_train,y_train)
```

```
pred_y = model.predict(x_cv)
```

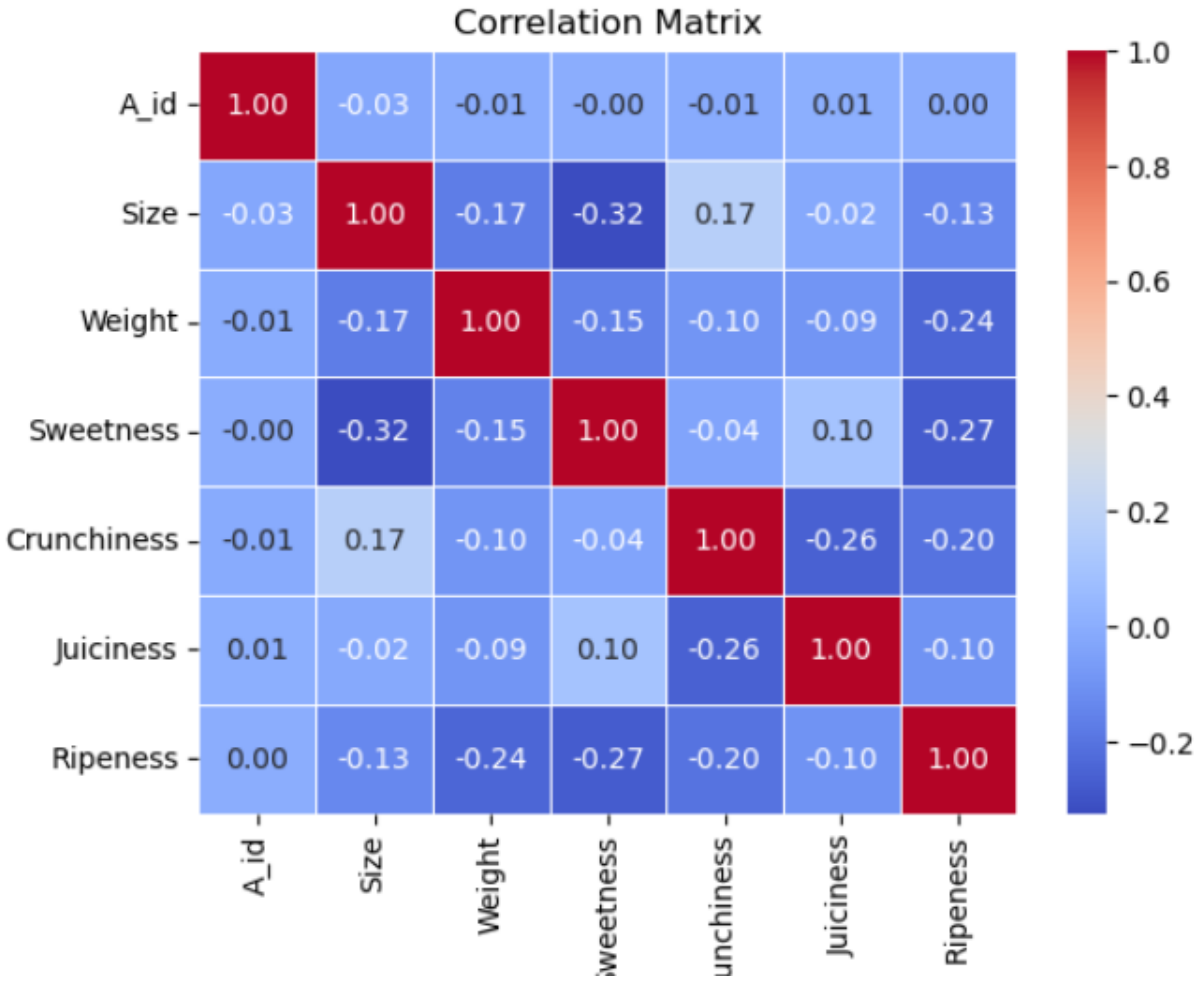
```
accuracy_score(y_cv,pred_y)
```

```
0.7640625
```

- Using gaussian naïve bayes as model for predicting the test .
- Got 76 percent accuracy. It was not a good model.
- Tried another methods.

```
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\srava\AppData\Local\Temp\ipykernel_16628\3412677725.py:1: FutureWarning: The defa
e.corr is deprecated. In a future version, it will default to False. Select only valid col
only to silence this warning.
  correlation_matrix = data.corr()
```



- Correlation of the data by visualization.
- The data is correlated.

Creating a model:

```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
knn_ = knn.fit(x_train,y_train)
y_pred_knn = knn_.predict(x_test)
```

```
print(accuracy_score(y_test,y_pred_knn))
```

```
0.90125
```

- Using ‘KNeighborsClassifier’ gave 90 percent accuracy.
- This is the best fit model for predicting.

Metrics

Classification report

```
print(classification_report(y_test,y_pred_knn))
```

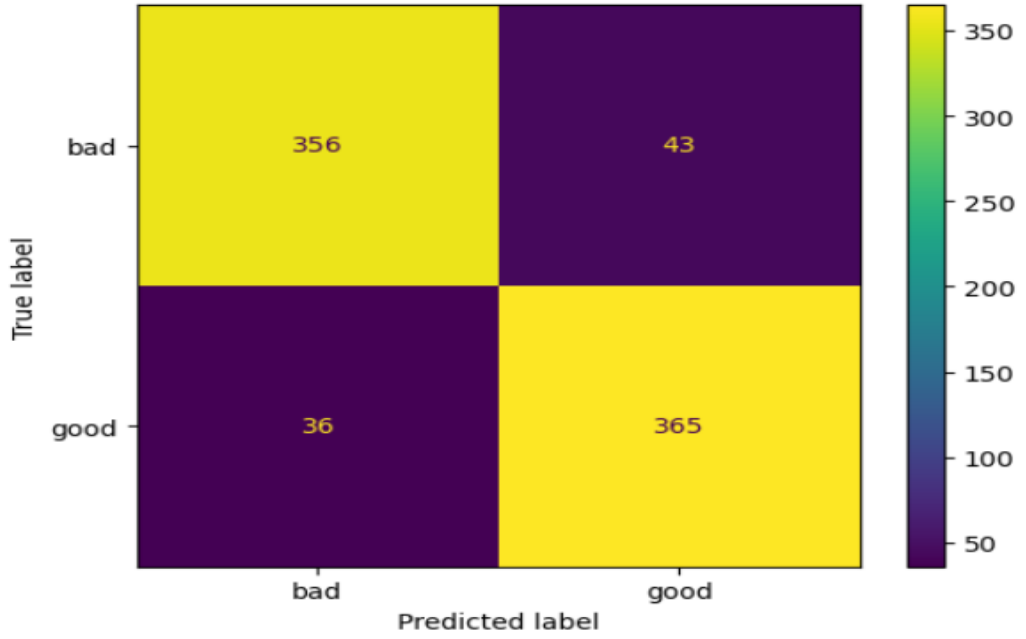
```
              precision    recall  f1-score   support

         bad       0.91      0.89      0.90       399
        good       0.89      0.91      0.90       401

    accuracy                           0.90       800
   macro avg       0.90      0.90      0.90       800
weighted avg       0.90      0.90      0.90       800
```

- Classification report of the model.
- Model has average of 90 percent score in precision, recall and f1 score in both good and bad classes.

```
cm = confusion_matrix(y_test,y_pred_knn)
```

```
i=ConfusionMatrixDisplay(cm,display_labels=knn_.classes_)
i.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2695e52
```



- The above figure is confusion matrix of the predicted values of the model, where the values in yellow color are True positives and True negatives and remaining are False positives and false negatives.

Deployment of model

```
pickle.dump(y_pred_knn,open(r'C:\Users\srava\Downloads\Apple_quality.pkl','wb'))
```

```
final_model = pickle.load(open(r"C:\Users\srava\Downloads\Apple_quality.pkl",'rb'))
```

```
def test():
    test_point = int(input())
    result = knn_.predict(x_test.iloc[[test_point]])
    return result[0]
```

```
test()
```

```
45

'bad'
```

- Model is deployed and loaded successfully and is ready to use.

Web application using the model