**EDINBURG FRUIT SURVEY**

The input dataset created by the Edinburg professor. The dataset download link finds below

<https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/fruit_data_with_colors.txt>

**Looking into Data: -**

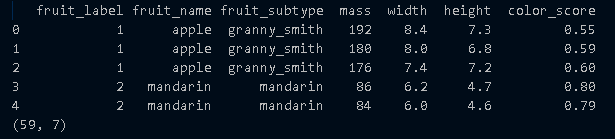
import pandas as pd

import matplotlib as plt

fruit = pd.read\_table(*r*"C:\Users\n0278588\GITHUB-Local\myML\Proj2-FruitSurvey-SimpleClassificationModels\InputDataSet.txt")

print(fruit.head())

print(fruit.shape)



The dataset has 59 rows and 7 feature vectors or columns.

Unique Fruits (instances) in dataset are

print(fruit['fruit\_name'].unique())

['apple' 'mandarin' 'orange' 'lemon']

Grouping the fruits by fruit name

print(fruit.groupby('fruit\_name').size())

fruit\_name

apple 19

lemon 16

mandarin 5

orange 19

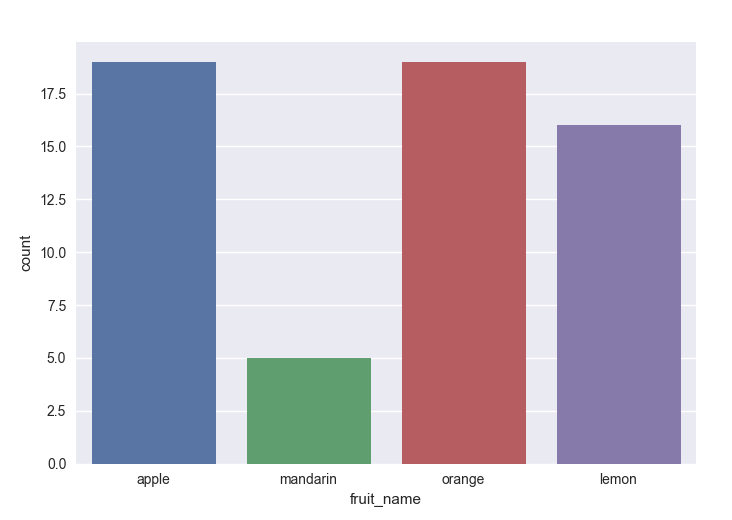
dtype: int64

Plotting the fruit name feature and see in graph for better data visualization. We will use Seaborn one of the most popular Python statistical data visualization tool

import seaborn as sns

sns.countplot(fruit['fruit\_name'],*label* = "Count")

plt.show()



**Visualizing the input variable distribution: -**

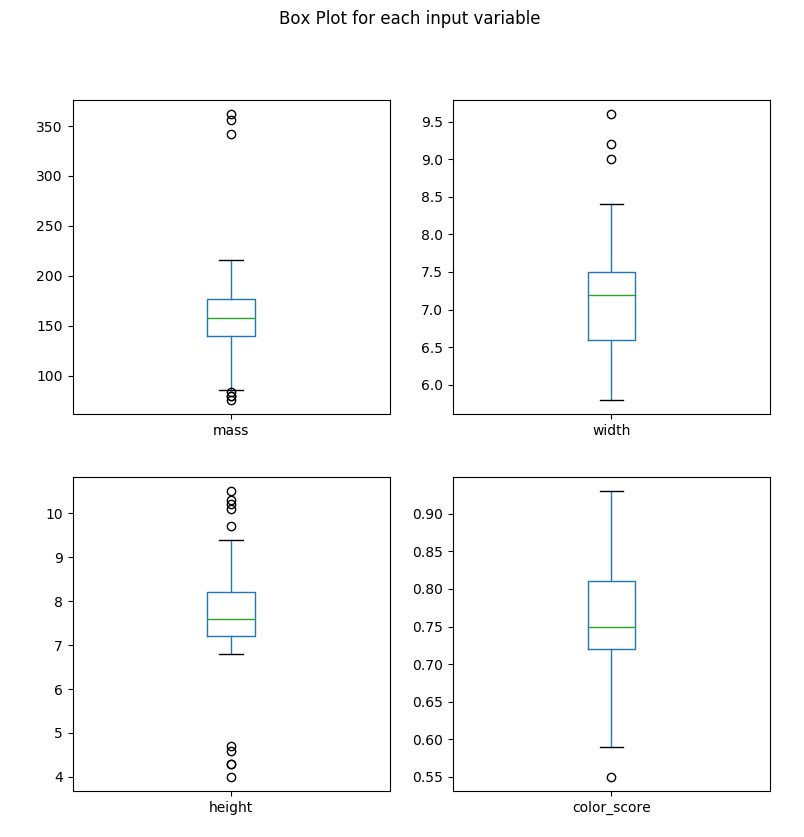
fruit.drop('fruit\_label', *axis*=1).plot(*kind*='box', *subplots*=True, *layout*=(2,2), *sharex*=False, *sharey*=False, *figsize*=(9,9),

*title*='Box Plot for each input variable')

#plt.savefig('fruits\_box')

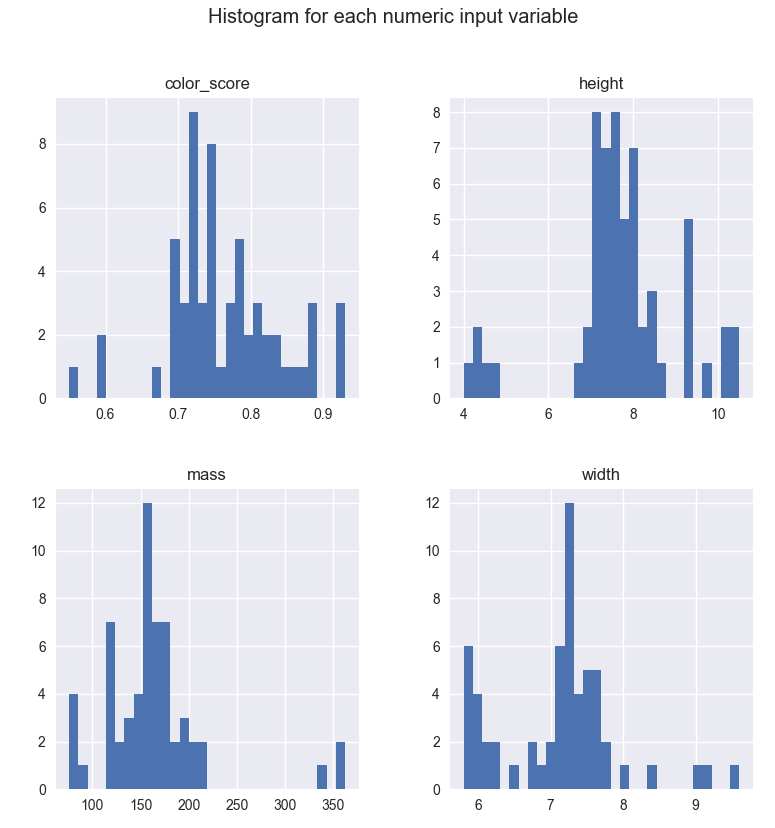
plt.show()

Only plotting happening for Features with numerical data.



**Visualizing in histogram: -**

The histogram shows there is some correlation between mass and width. Both the plots looks quite similar.



**SCATTER MATRIX FOR EACH INPUT VARIABLE: -**

from pandas.tools.plotting import scatter\_matrix

from matplotlib import cm

feature\_names = ['mass','width','height','color\_score']

X = fruit[feature\_names]

y = fruit['fruit\_label']

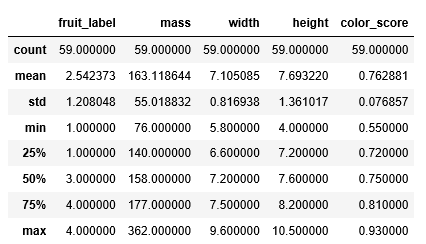
cmap = cm.get\_cmap('gnuplot')

scatter = pd.scatter\_matrix(X, *c*=y, *marker* = 'o', *s*=40, *hist\_kwds*=('bins':15),*figsize*=(9,9),*cmap*=cmap)

plt.suptitle('Scatter matrix for each input varible')

plt.savefig('fruits\_scatter\_matrix')

The numerical values are not in proper scale, we have to scale it for Train and test data



Creating the scaling for Train and Test data

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

feature\_names = ['mass','width','height','color\_score']

X = fruit[feature\_names]

y = fruit['fruit\_label']

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,*random\_state* = 0)

scaler = MinMaxScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.fit\_transform(X\_test)

**BUILDING MODELS (Supervised Learnings)**

**Regression Model: -**

Note: - Regression model no more present under model selection. Use linear model instead

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(X\_train,y\_train)

print(*f*'Accuracy of Logistit regression on Train data {logreg.score(X\_train,y\_train)*:.2f*}')

print(*f*'Accuracy of Logistit regression on Test data {logreg.score(X\_test,y\_test)*:.2f*}')

Accuracy of Logistit regression on Train data 0.70

Accuracy of Logistit regression on Test data 0.40

**Decision Tree Model: -**

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf.fit(X\_train,y\_train)

print(*f*'\nAccuracy of Decision Tree Classifier on Train data {clf.score(X\_train,y\_train)*:.2f*}')

print(*f*'Accuracy of Decision Tree Classifier on Test data {clf.score(X\_test,y\_test)*:.2f*}')

Accuracy of Decision Tree Classifier on Train data 1.00

Accuracy of Decision Tree Classifier on Test data 0.67

**K-Nearest Neighbor: -**

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(X\_train,y\_train)

print(*f*'\nAccuracy of K-NN on Train data {knn.score(X\_train,y\_train)*:.2f*}')

print(*f*'Accuracy of K-NN on Test data {knn.score(X\_test,y\_test)*:.2f*}')

Accuracy of K-NN on Train data 0.95

Accuracy of K-NN on Test data 0.80

**Linear Discriminant Analysis: -**

This is the most commonly used dimension reduction technique in data preprocessing for pattern classification in ML.

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()

lda.fit(X\_train,y\_train)

print(*f*'\nAccuracy of Linear Discriminant Analysis on Train data {knn.score(X\_train,y\_train)*:.2f*}')

print(*f*'Accuracy of Linear Discriminant Analysis on Test data {knn.score(X\_test,y\_test)*:.2f*}')

Accuracy of Linear Discriminant Analysis on Train data 0.95

Accuracy of Linear Discriminant Analysis on Test data 0.80

**Gaussian naïve Bayes: -**

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X\_train,y\_train)

print(*f*'\nAccuracy of GNB classifier on Train data {gnb.score(X\_train,y\_train)*:.2f*}')

print(*f*'Accuracy of GNB classifier on Test data {gnb.score(X\_test,y\_test)*:.2f*}')

Accuracy of GNB classifier on Train data 0.86

Accuracy of GNB classifier on Test data 0.47

**Support Vector Machine: -**

from sklearn.svm import SVC

svm = SVC()

svm.fit(X\_train,y\_train)

print(*f*'\nAccuracy of SVM on Train data {svm.score(X\_train,y\_train)*:.2f*}')

print(*f*'Accuracy of SVM classifier on Test data {svm.score(X\_test,y\_test)*:.2f*}')

Accuracy of SVM on Train data 0.61

Accuracy of SVM classifier on Test data 0.40

Consolidated Table: -

|  |  |  |
| --- | --- | --- |
| Models | Train Accuracy | Test Accuracy |
| Regression Model | 0.7 | 0.4 |
| Decision Tree Model | 1.0 | 0.67 |
| K-NN | 0.95 | 0.80 |
| Linear Discriminant Analysis | 0.86 | 0.73 |
| Gaussian Naïve Byes | 0.86 | 0.47 |
| Support Vector machine | 0.61 | 0.40 |

Out of above bucket of models K-NN and LDA both are having more close results in between Train and Test data but after multiple runs you can see that K-NN is consistently providing more close results than LDA.