**Fruits-Color Text-Classification**

# coding: utf-8

# In[1]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# In[2]:

fruits = pd.read\_table(r'C:\Users\akasriva2\Music\Analysis\fruit\_data.txt')

fruits.head()

# In[3]:

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

# In[4]:

print(fruits.shape)

# In[5]:

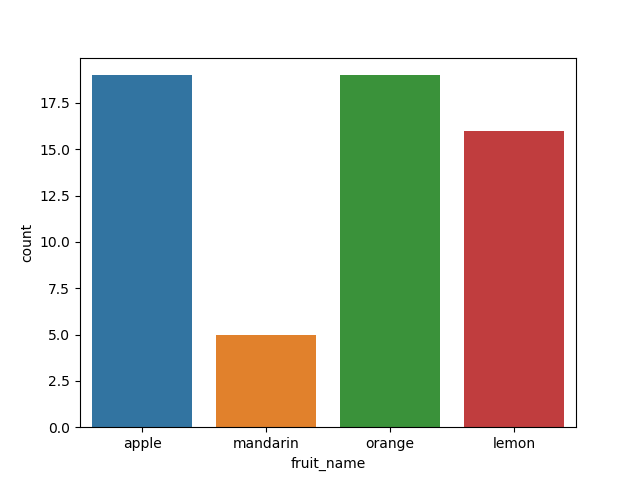
print(fruits.groupby('fruit\_name').size())

# In[6]:

import seaborn as sns

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

plt.show()



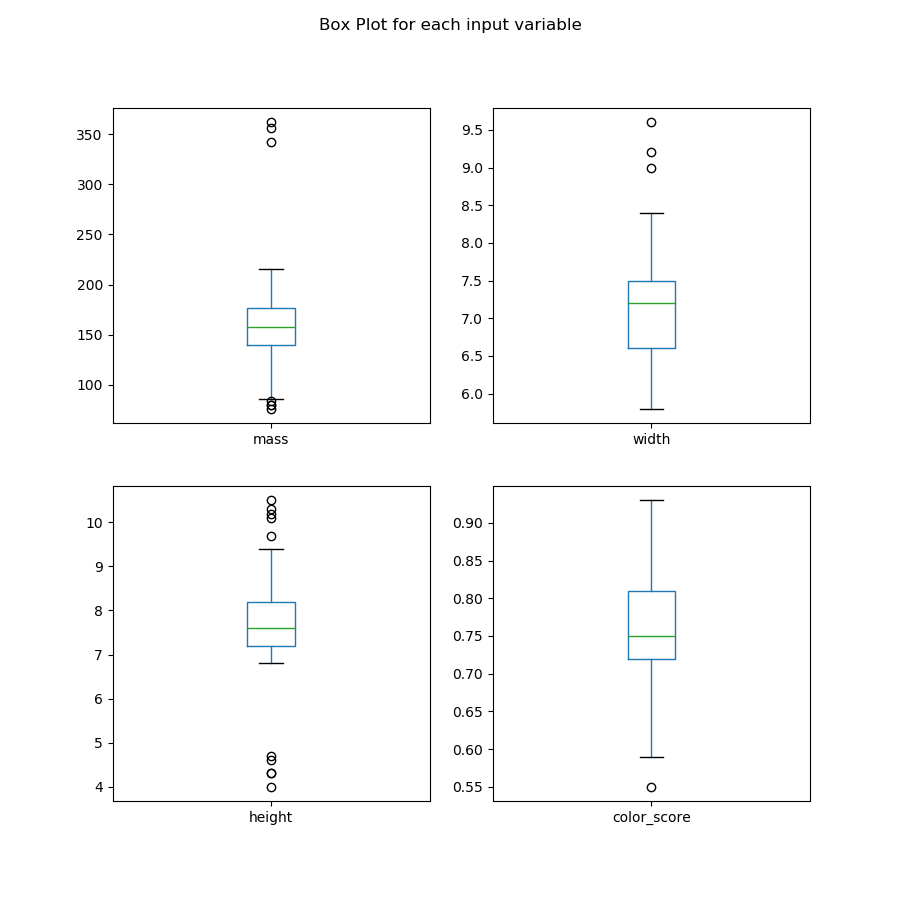
# In[7.1]

fruits.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\_boxplot')

plt.show()



# In[7.2]:

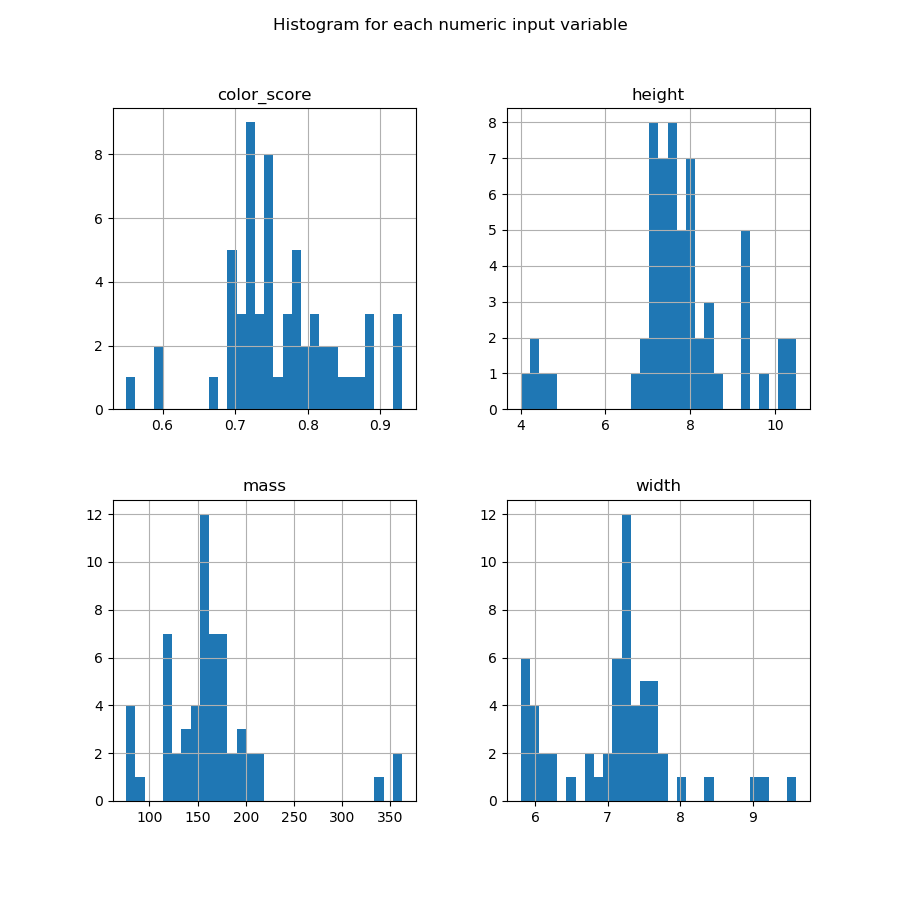
import pylab as pl

fruits.drop('fruit\_label' ,axis=1).hist(bins=30, figsize=(9,9))

pl.suptitle("Histogram for each numeric input variable")

plt.savefig('fruits\_hist')

plt.show()



# In[7.3]:

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

X = fruits[feature\_names]

y = fruits['fruit\_label']

# In[8]:

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

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

# In[9]:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

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

X\_test = scaler.transform(X\_test)

**// Logistic Regression**

# In[10]:

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

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

print('Accuracy of Logistic regression classifier on training set: {:.2f}'

.format(logreg.score(X\_train, y\_train)))

print('Accuracy of Logistic regression classifier on test set: {:.2f}'

.format(logreg.score(X\_test, y\_test)))

Accuracy of Logistic regression classifier on training set: 0.70

Accuracy of Logistic regression classifier on test set: 0.40

**// Decision Tree**

# In[11]:

from sklearn.tree import DecisionTreeClassifier

clf2 = DecisionTreeClassifier(max\_depth=3).fit(X\_train, y\_train)

print('Accuracy of Decision Tree classifier on training set: {:.2f}'

.format(clf2.score(X\_train, y\_train)))

print('Accuracy of Decision Tree classifier on test set: {:.2f}'

.format(clf2.score(X\_test, y\_test)))

Accuracy of Decision Tree classifier on training set: 1.00

Accuracy of Decision Tree classifier on test set: 0.87

**// KNN**

# In[13]:

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

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

print('Accuracy of K-NN classifier on training set: {:.2f}'

.format(knn.score(X\_train, y\_train)))

print('Accuracy of K-NN classifier on test set: {:.2f}'

.format(knn.score(X\_test, y\_test)))

Accuracy of K-NN classifier on training set: 0.95

Accuracy of K-NN classifier on test set: 1.00

**// LDA**

# In[14]:

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()

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

print('Accuracy of LDA classifier on training set: {:.2f}'

.format(lda.score(X\_train, y\_train)))

print('Accuracy of LDA classifier on test set: {:.2f}'

.format(lda.score(X\_test, y\_test)))

Accuracy of LDA classifier on training set: 0.86

Accuracy of LDA classifier on test set: 0.67

**// GaussianNB**

# In[15]:

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

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

print('Accuracy of GNB classifier on training set: {:.2f}'

.format(gnb.score(X\_train, y\_train)))

print('Accuracy of GNB classifier on test set: {:.2f}'

.format(gnb.score(X\_test, y\_test)))

Accuracy of GNB classifier on training set: 0.86

Accuracy of GNB classifier on test set: 0.67

**// SVM**

# In[16]:

from sklearn.svm import SVC

svm = SVC()

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

print('Accuracy of SVM classifier on training set: {:.2f}'

.format(svm.score(X\_train, y\_train)))

print('Accuracy of SVM classifier on test set: {:.2f}'

.format(svm.score(X\_test, y\_test)))

Accuracy of SVM classifier on training set: 0.61

Accuracy of SVM classifier on test set: 0.33

//Confusion Matrix

# In[17]:

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

pred = knn.predict(X\_test)

print(confusion\_matrix(y\_test, pred))

print(classification\_report(y\_test, pred))

[[4 0 0 0]

[0 1 0 0]

[0 0 8 0]

[0 0 0 2]]

precision recall f1-score support

1 1.00 1.00 1.00 4

2 1.00 1.00 1.00 1

3 1.00 1.00 1.00 8

4 1.00 1.00 1.00 2

avg / total 1.00 1.00 1.00 15