**ASSIGNMENT – 3**

| Assignment Date | 05 October 2022 |
| --- | --- |
| Student Name | Divya Rani.R |
| Student Roll Number | 111619104024 |
| Maximum Marks | 2 Marks |

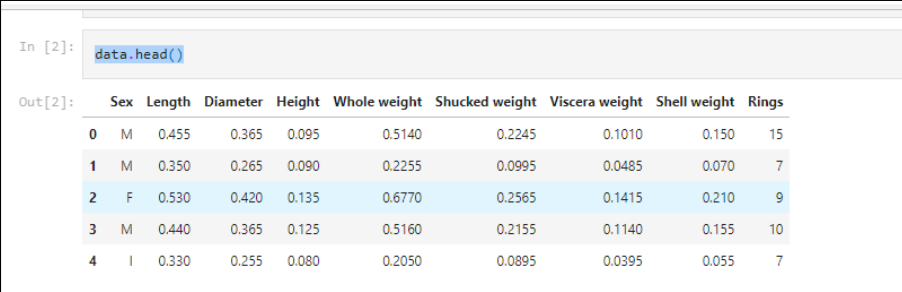
**Building a Regression Model**

1. Download the dataset: Dataset

**data=pd.read\_csv("abalone.csv")**

2. Load the dataset into the tool.

**data.head()**

****

3. Perform Below Visualizations.

∙ Univariate Analysis

***#univariate analysis***

cols **=** 3

rows **=** 3

num\_cols **=** data**.**select\_dtypes(exclude**=**'object')**.**columns

fig **=** plt**.**figure( figsize**=**(cols**\***5, rows**\***5))

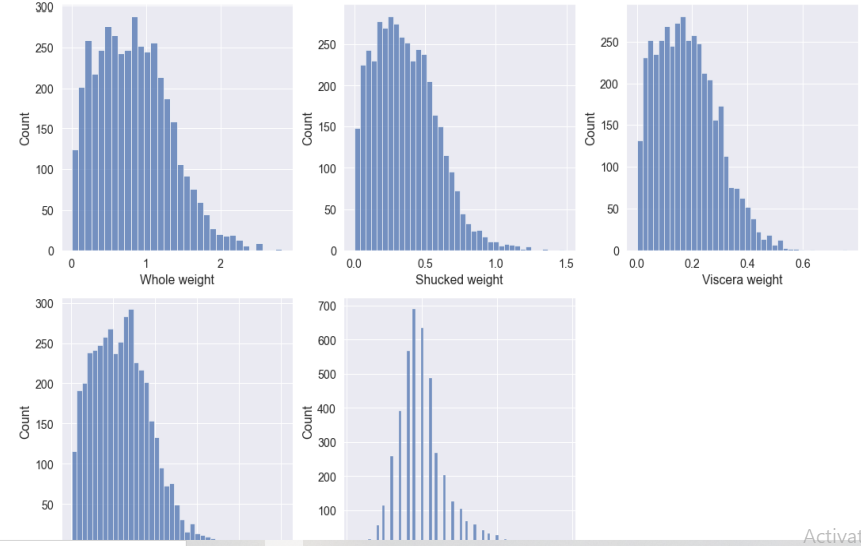
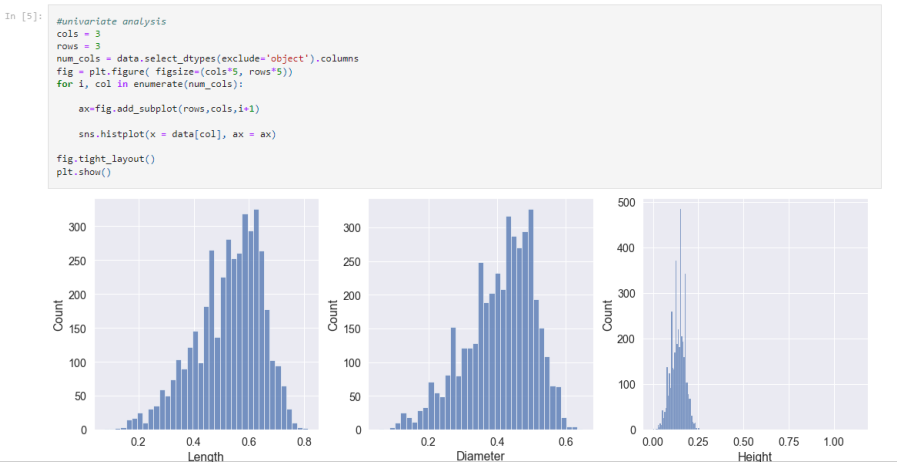
**for** i, col **in** enumerate(num\_cols):

ax**=**fig**.**add\_subplot(rows,cols,i**+**1)

sns**.**histplot(x **=** data[col], ax **=** ax)

fig**.**tight\_layout()

plt**.**show()

∙

Bi-Variate Analysis

***#Bivariate analysis***

**import** matplotlib.pyplot **as** plt

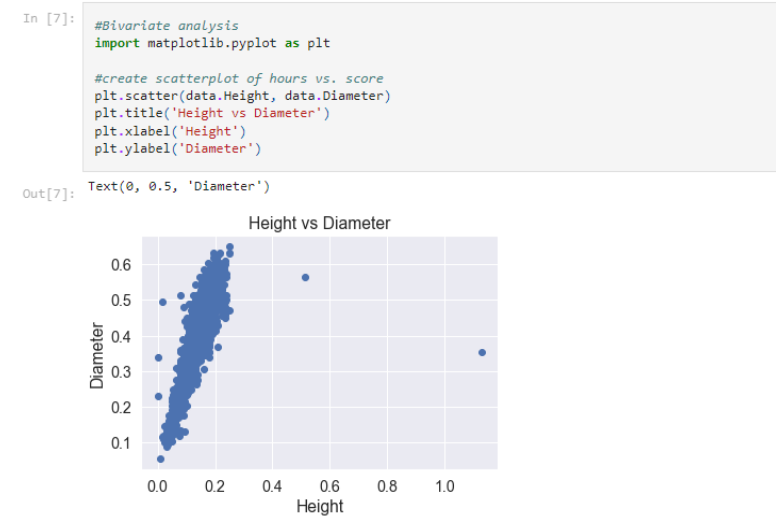
*#create scatterplot of hours vs. score*

plt**.**scatter(data**.**Height, data**.**Diameter)

plt**.**title('Height vs Diameter')

plt**.**xlabel('Height')

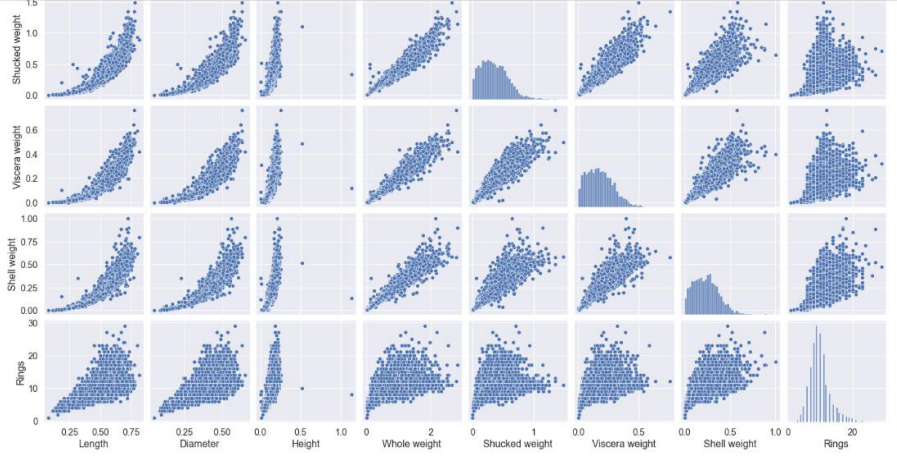
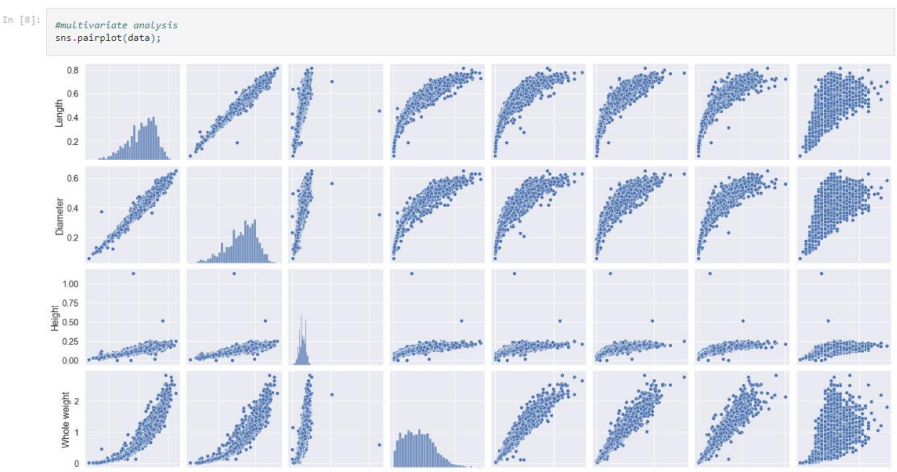
Plt**.**ylabel



Multi-Variate Analysis

***#multivariate analysis***

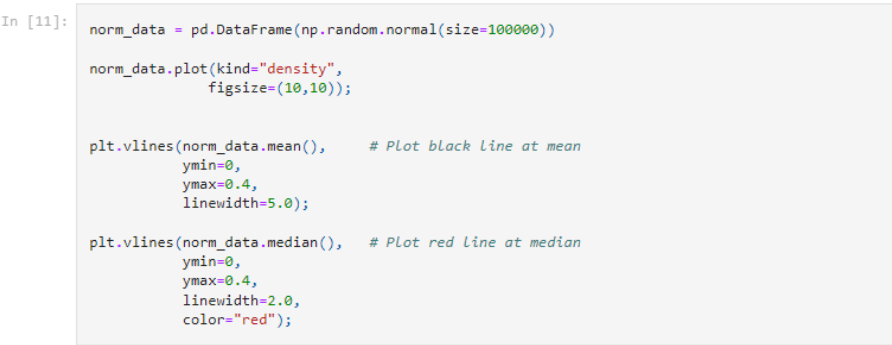
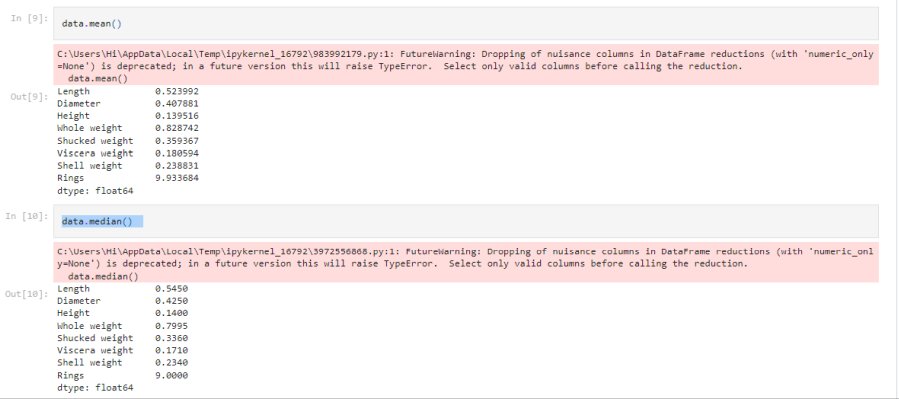
sns**.**pairplot(d)

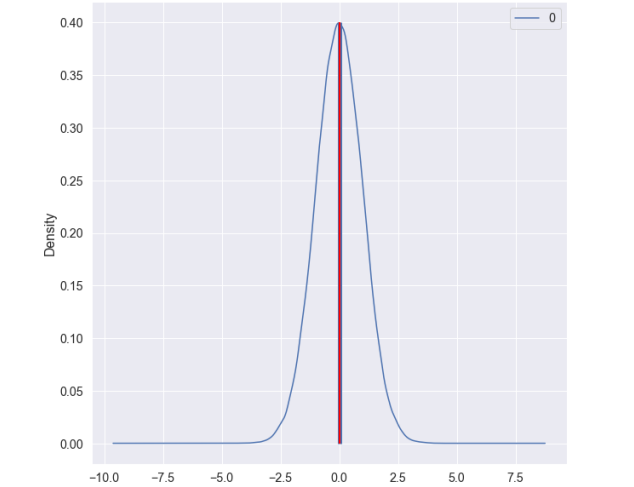


4. Perform descriptive statistics on the dataset

data**.**mean()

data**.**median()



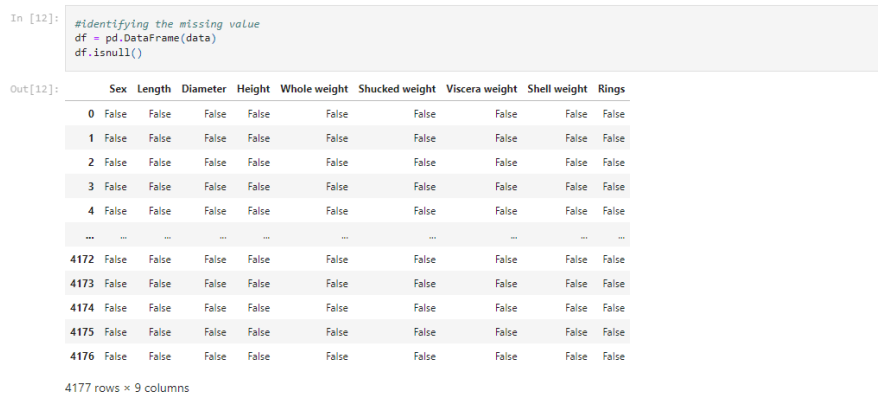


5. Check for Missing values and deal with them.

***#identifying the missing value***

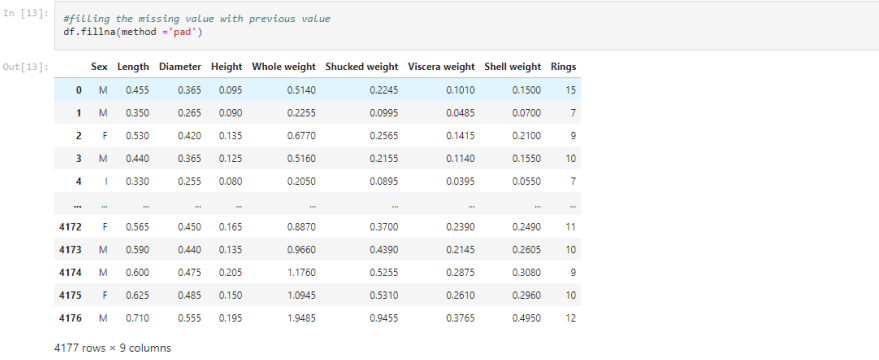
df **=** pd**.**DataFrame(data)

df**.**isnull()



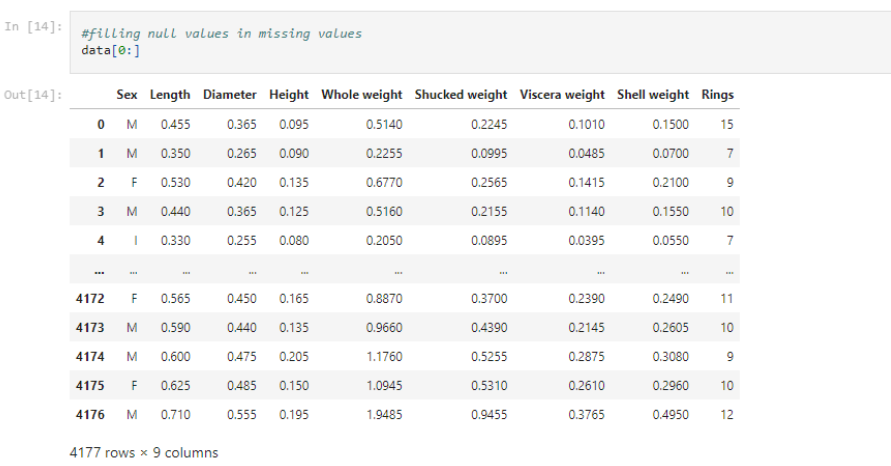
***#filling the missing value with previous value***

df**.**fillna(method **=**'pad')



***#filling null values in missing values***

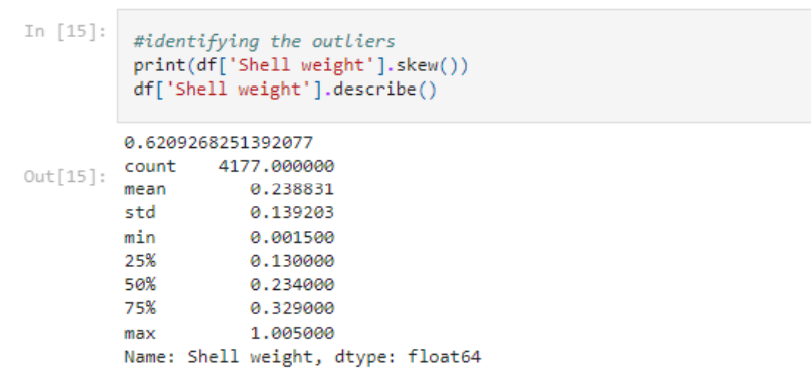
data[0:]

6. Find the outliers and replace them outliers

***#identifying the outliers***

print(df['Shell weight']**.**skew())

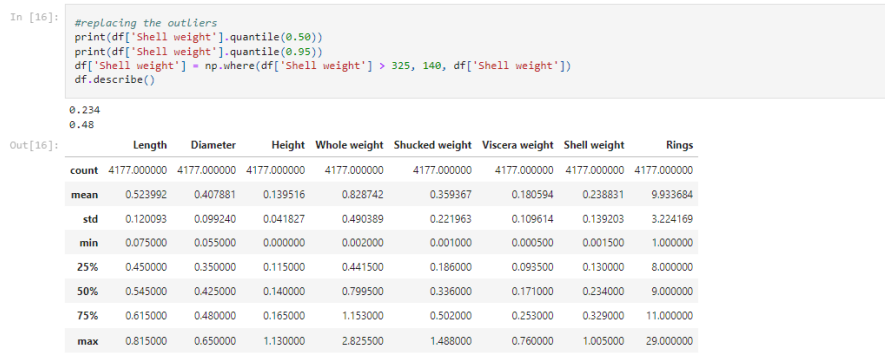
df['Shell weight']**.**describe()

***#replacing the outliers***

print(df['Shell weight']**.**quantile(0.50))

print(df['Shell weight']**.**quantile(0.95))

df['Shell weight'] **=** np**.**where(df['Shell weight'] **>** 325, 140, df['Shell weight']) df**.**describe()

7. Check for Categorical columns and perform encoding.

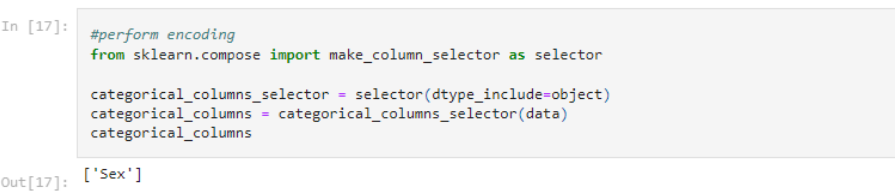
***#perform encoding***

**from** sklearn.compose **import** make\_column\_selector **as** selector

categorical\_columns\_selector **=** selector(dtype\_include**=**object)

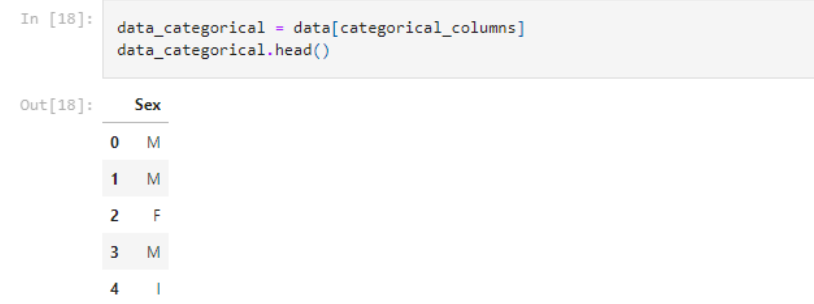
categorical\_columns **=** categorical\_columns\_selector(data)

categorical\_columns



data\_categorical **=** data[categorical\_columns]

data\_categorical.head()



8. Split the data into dependent and independent variables.

**from** sklearn **import** preprocessing

***# label\_encoder object knows how to understand word labels.***

label\_encoder **=** preprocessing**.**LabelEncoder()

***# Encode labels in column 'species'.***

df['Sex']**=** label\_encoder**.**fit\_transform(df['Sex'])

df['Sex']**.**unique()

X= data.iloc[ : , :-1].values

y= data.iloc[ : , 4].values

print(X,y)

*# import packages*

**import** numpy **as** np

**import** pandas **as** pd

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

*# importing data*

print(df**.**shape)

*# head of the data*

print('Head of the dataframe : ')

print(df**.**head())

print(df**.**columns)

X**=** df['Whole weight']

y**=**df['Shucked weight']

*# using the train test split function*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

X,y , random\_state**=**104,test\_size**=**0.25, shuffle**=True**)

*# printing out train and test sets*

print('X\_train : ')

print(X\_train**.**head())

print(X\_train**.**shape)

print('')

print('X\_test : ')

print(X\_test**.**head())

print(X\_test**.**shape)

print('')

print('y\_train : ')

print(y\_train**.**head())

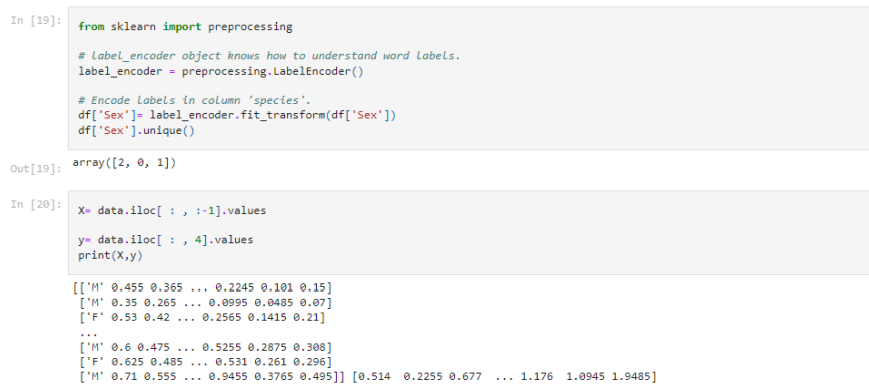
print(y\_train**.**shape)

print('')

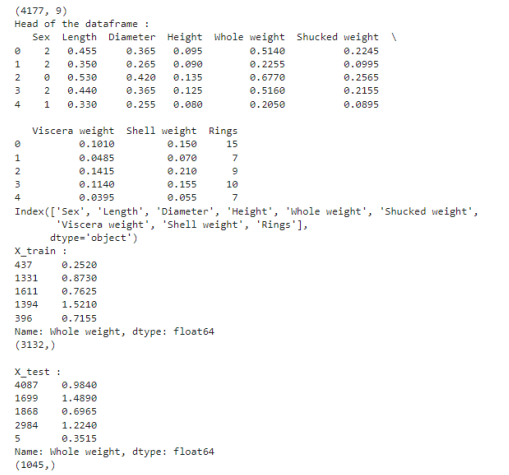
print('y\_test : ')

print(y\_test**.**head())

print(y\_test**.**shape)







9. Scale the independent variables

*#scaling*

df\_scaled = df.copy()

col\_names = ['Shucked weight', 'Whole weight']

features = df\_scaled[col\_names]

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

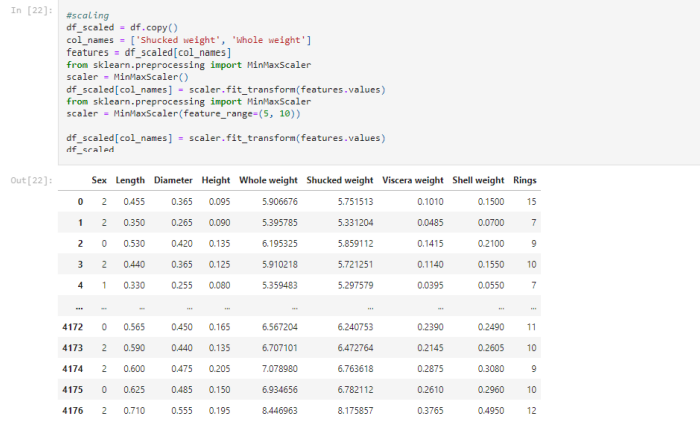
df\_scaled[col\_names] = scaler.fit\_transform(features.values)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(5, 10))

df\_scaled[col\_names] = scaler.fit\_transform(features.values)

df\_scaled



10. Split the data into training and testing

***#testing and training***

X **=** df**.**iloc[:, :**-**1]

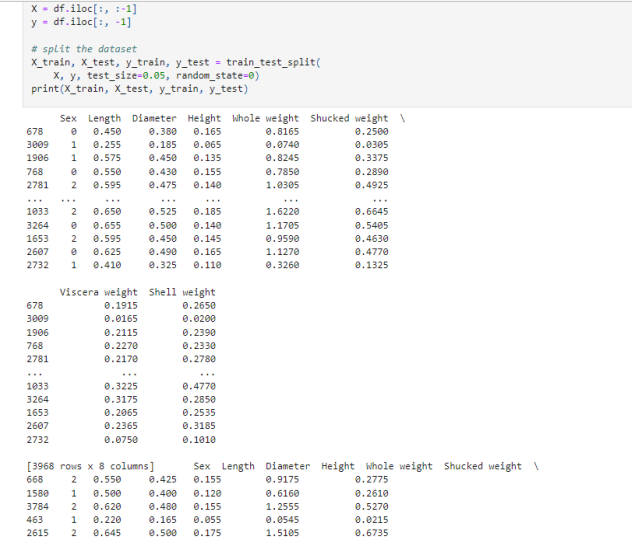
y **=** df**.**iloc[:, **-**1]

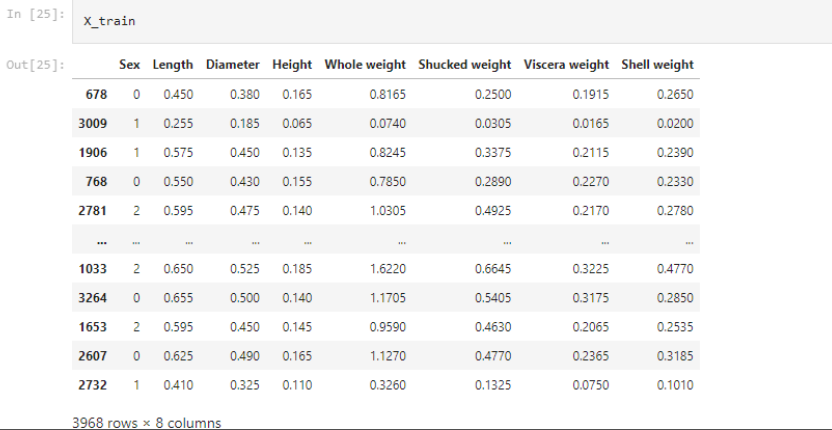
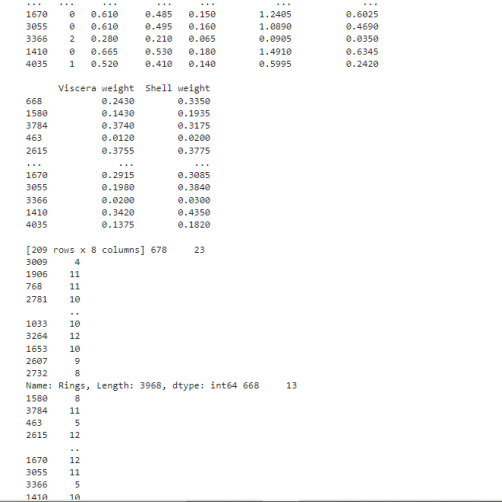
***# split the dataset***

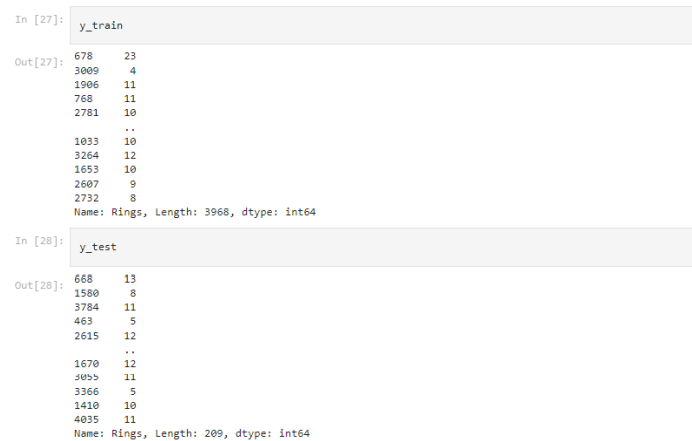
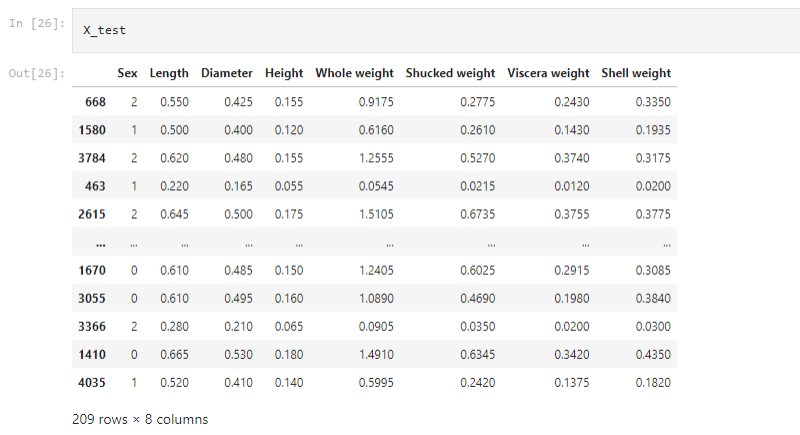
X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

X, y, test\_size**=**0.05, random\_state**=**0)

print(X\_train, X\_test, y\_train, y\_test)





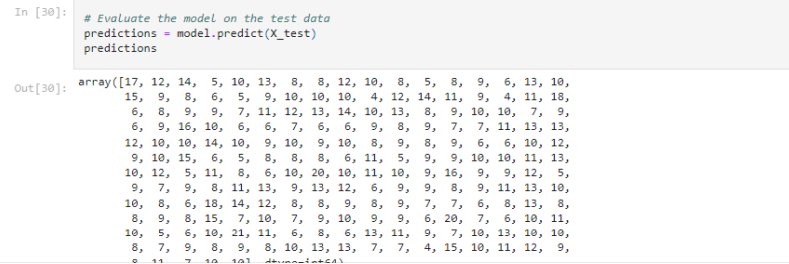


11. Build the Model

***# Evaluate the model on the test data***

predictions **=** model**.**predict(X\_test)

predictions

12. Train the Model

***# Select algorithm***

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.metrics **import** accuracy\_score

model **=** DecisionTreeClassifier()

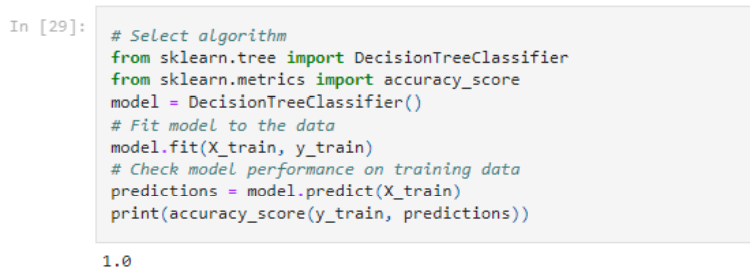
***# Fit model to the data***

model**.**fit(X\_train, y\_train)

***# Check model performance on training data***

predictions **=** model**.**predict(X\_train)

print(accuracy\_score(y\_train, predictions))

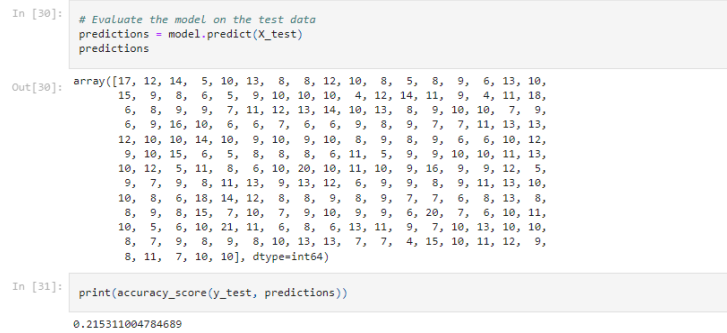


13. Test the Model

***# Evaluate the model on the test data***

predictions **=** model**.**predict(X\_test)

predictions



14. Measure the performance using Metrics.

**import** os

os**.**environ["PATH"] **+=** os**.**pathsep **+** 'C:/Program Files (x86)/Graphviz2.38/bin' from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import log\_loss

X\_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]

Y\_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

results = confusion\_matrix(X\_actual, Y\_predic)

print ('Confusion Matrix :')

print(results)

print ('Accuracy Score is',accuracy\_score(X\_actual, Y\_predic))

print ('Classification Report : ')

print (classification\_report(X\_actual, Y\_predic))

print('AUC-ROC:',roc\_auc\_score(X\_actual, Y\_predic))

print('LOGLOSS Value is',log\_loss(X\_actual, Y\_predic))

