Problem Statement:

The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem. You have to predict the rings of each abalone which will lead us to the age of that abalone.

Importing required libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pickle
import seaborn as sns
from scipy.stats import zscore
import scikitplot as skplt
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_sco
from sklearn.preprocessing import power_transform, StandardScaler
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sco
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

Reading data

In [2]:
```
df = pd.read_csv(r"C:\Users\Kushal Arya\Desktop\Data Analysis With Python\ML File
df.head()
```

Out[2]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Check no of row and column

In [7]:
```
print('No of Rows and Columns ----->', df.shape )
```

```
No of Rows and Columns -----> (4177, 9)
```

Checking for Null values

In [9]:
```
print('----------------------\n')
print(df.isnull().sum())
print('\n----------------------')
```

```
----------------------

Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64

----------------------
```

There is no null value in our dataset

Information about dataset

In [11]:
```
print('-------------------------------------------------\n')
print(df.info())
print('\n-------------------------------------------------')
```

```
-------------------------------------------------

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
None

-------------------------------------------------
```

There is sex feature which is object type we have convert into int.

Checking and converting Sex feature into int

In [24]:
```
print('-------------------------------------------------------')
print('No of Mane, Infant and Female present in our dataset.')
print('-------------------------------------------------------')
print(df['Sex'].value_counts())
print('-------------------------------------------------------')
```

```
-------------------------------------------------------
No of Mane, Infant and Female present in our dataset.
-------------------------------------------------------
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
-------------------------------------------------------
```

In [25]:
```
df['Sex'] = df['Sex'].replace({'M':0,'F':1,'I':2})
df.head()
```

Out[25]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Replace Male : 0, Female : 1, Infant : 2

Statistical data

In [8]:
```
df.describe()
```

Out[8]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.955470	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594
std	0.827815	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614
min	0.000000	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500
25%	0.000000	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500
50%	1.000000	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000
75%	2.000000	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000
max	2.000000	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000

There is some outliers present in our dataset.

Relation feature and label

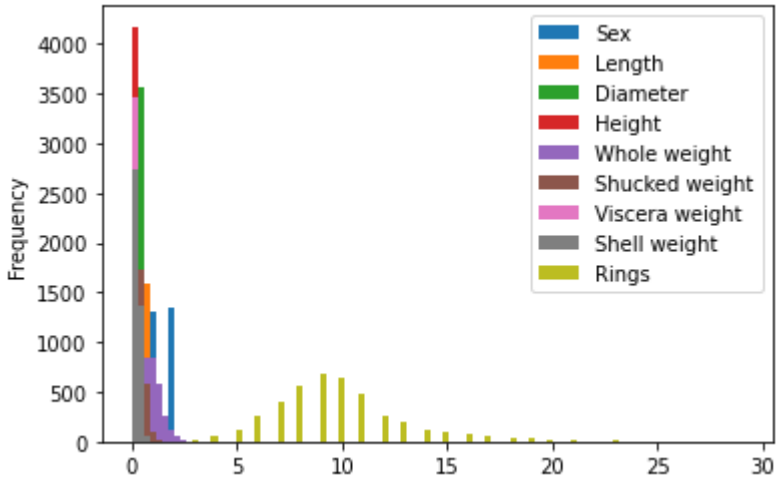
In [28]:
```
print('-------------------------------------')
print(df.groupby('Sex')['Rings'].sum())
print('-------------------------------------')
```

```
-------------------------------------
Sex
0    16358
1    14546
2    10589
Name: Rings, dtype: int64
-------------------------------------
```

Observation : No of rings are vary from sex.

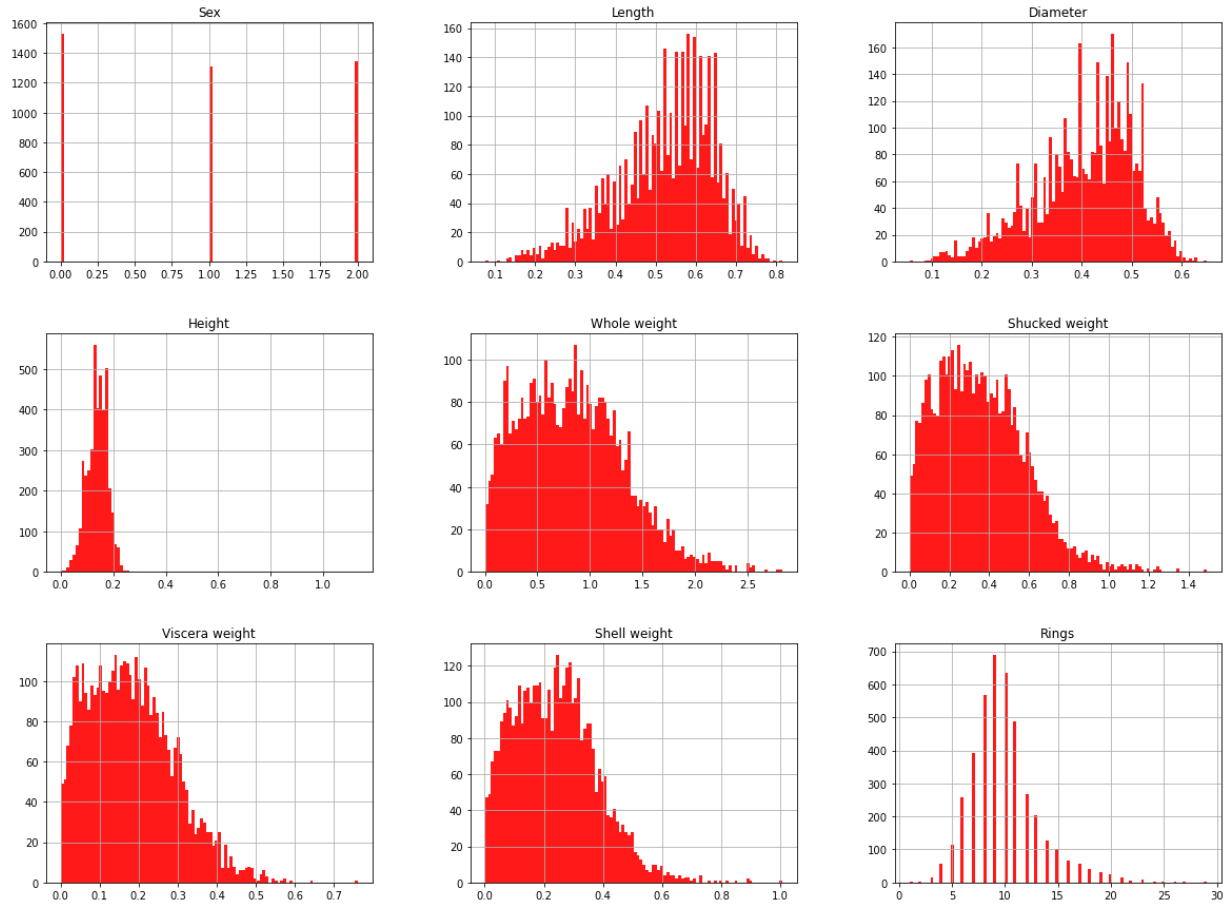
Understanding our dataset through Visualization

In [10]:
```
df.plot(kind = 'hist', bins= 100)
plt.show()
```



Above plot show properties of data distribution scale

In [11]:
```
df.hist(color = 'r', alpha = 0.9, figsize = (20,15), bins = 100)
plt.show()
```

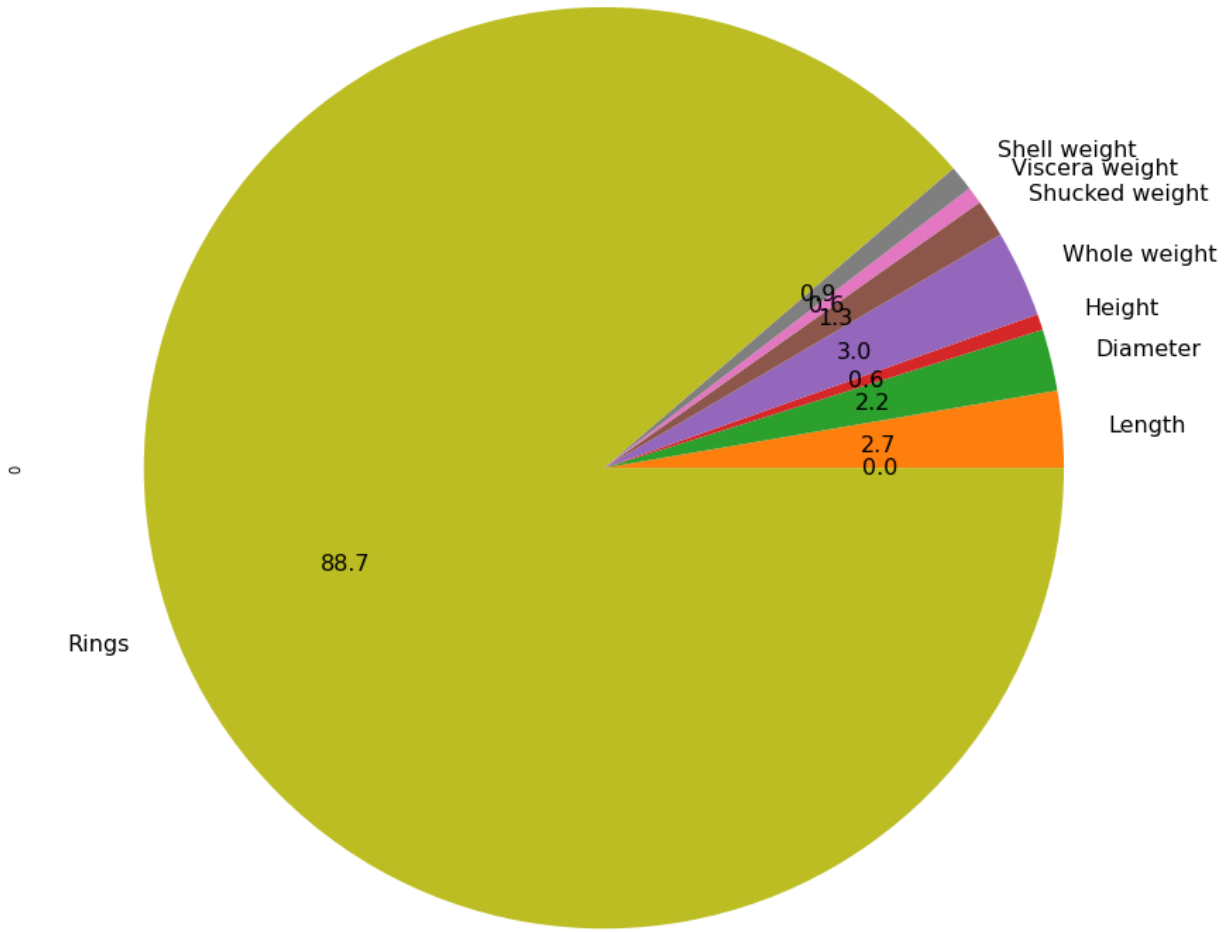


Above plot demonstrate features and how they distributed some are left skewed and some are right skewed

In [12]:
```
s = df.iloc[0]
s
```

Out[12]:
```
Sex                0.0000
Length             0.4550
Diameter           0.3650
Height             0.0950
Whole weight       0.5140
Shucked weight     0.2245
Viscera weight     0.1010
Shell weight       0.1500
Rings             15.0000
Name: 0, dtype: float64
```

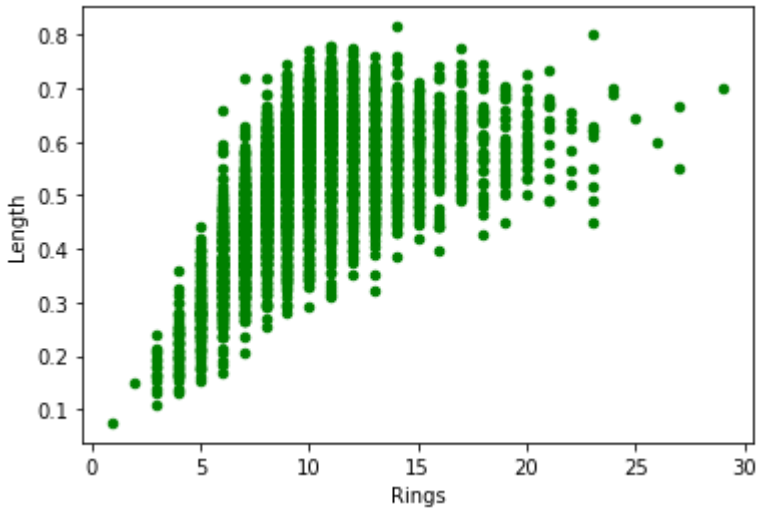
In [13]:
```
s.plot.pie(subplots = True, fontsize = 16, autopct = '%.1f',figsize = (20,15))
plt.show()
```



Abalone made of 88.7% of shell.

df.plot.scatter(x = 'Rings', y = 'Sex' , color = 'r') plt.show()

In [14]:
```
df.plot.scatter(x = 'Rings', y = 'Length' , color = 'g')
plt.show()
```



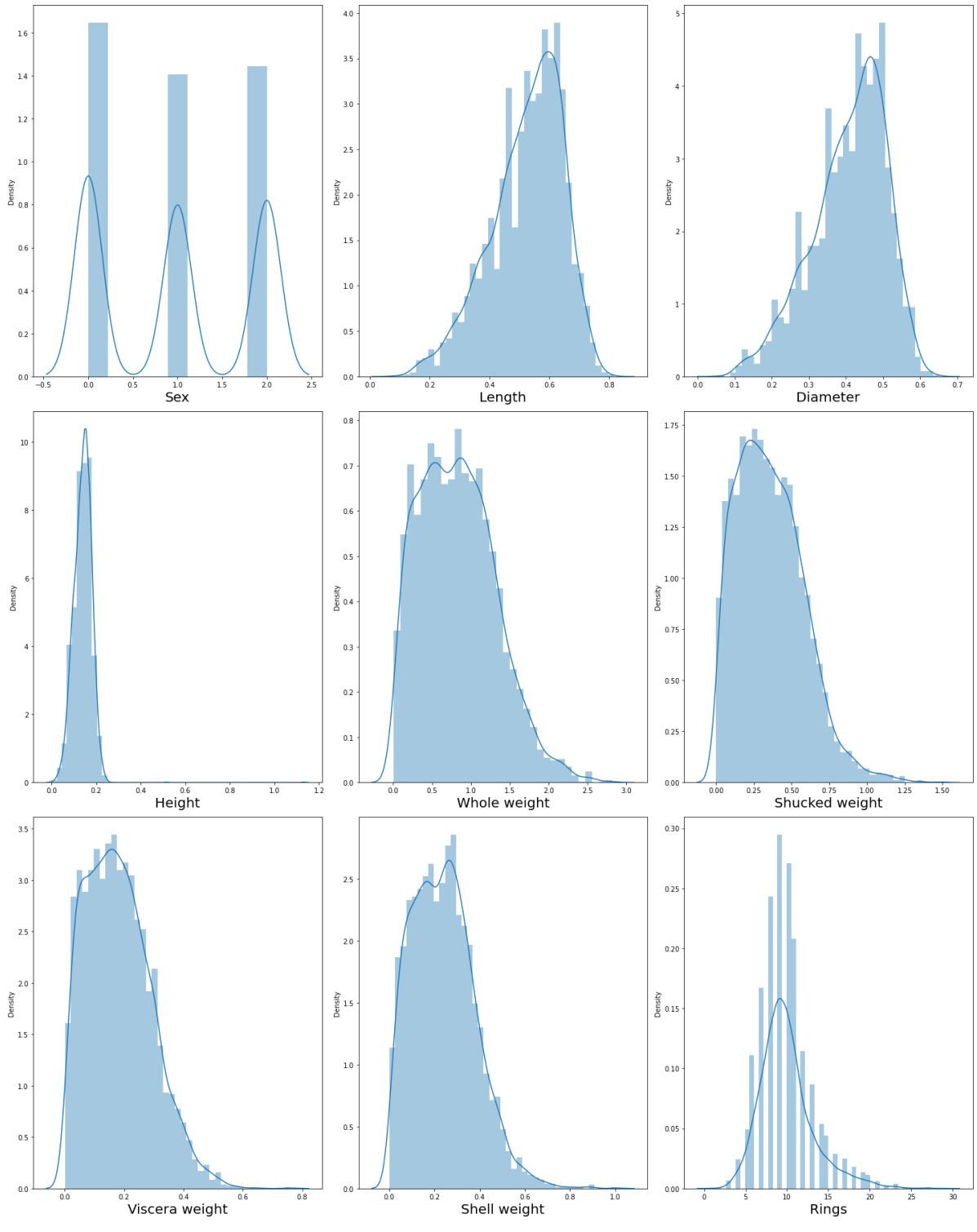
As we see above scattern plot as length increase no of Rings increase

In [35]:
```
print('----------------------')
print('Distribution Plot :- ')
print('----------------------')

plt.figure(figsize = (20,25))
plotnumber = 1

for column in df:
    if plotnumber <=9:
        ax = plt.subplot(3,3, plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column, fontsize = 20)
    plotnumber +=1
plt.tight_layout()
```

```
----------------------
Distribution Plot :-
----------------------
```



Observations : Some features is little skewed.

Checking for Imbalance Label

In [37]:
```
print('-----------------------------------')
print('No of rings present in each class :')
print('-----------------------------------')
print(df['Rings'].value_counts())
print('-----------------------------------')
```

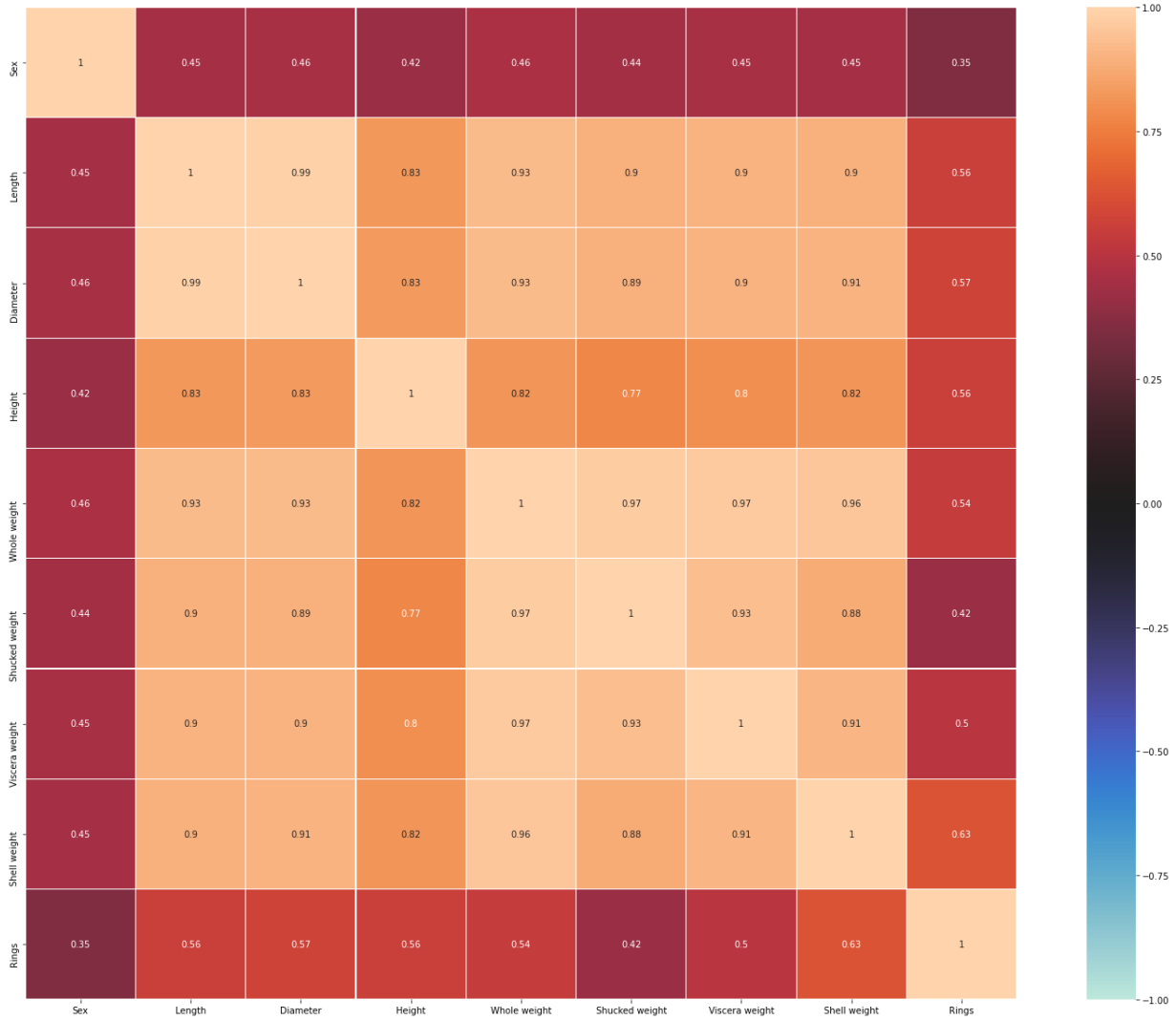
```
-----------------------------------
No of rings present in each class :
-----------------------------------
9     689
10    634
8     568
11    487
7     391
12    267
6     259
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
27      2
24      2
26      1
29      1
25      1
1       1
2       1
Name: Rings, dtype: int64
-----------------------------------
```

Corelation of Feature vs Label using Heat map

In [39]:
```
print('-----------')
print('Heat Map :-')
print('-----------')
df_corr = df.corr().abs()

plt.figure(figsize = (22,16))
sns.heatmap(df_corr, vmin = -1, annot = True, square = True, center = 0, fmt = '
plt.tight_layout()
```

```
-----------
Heat Map :-
-----------
```



All features are related

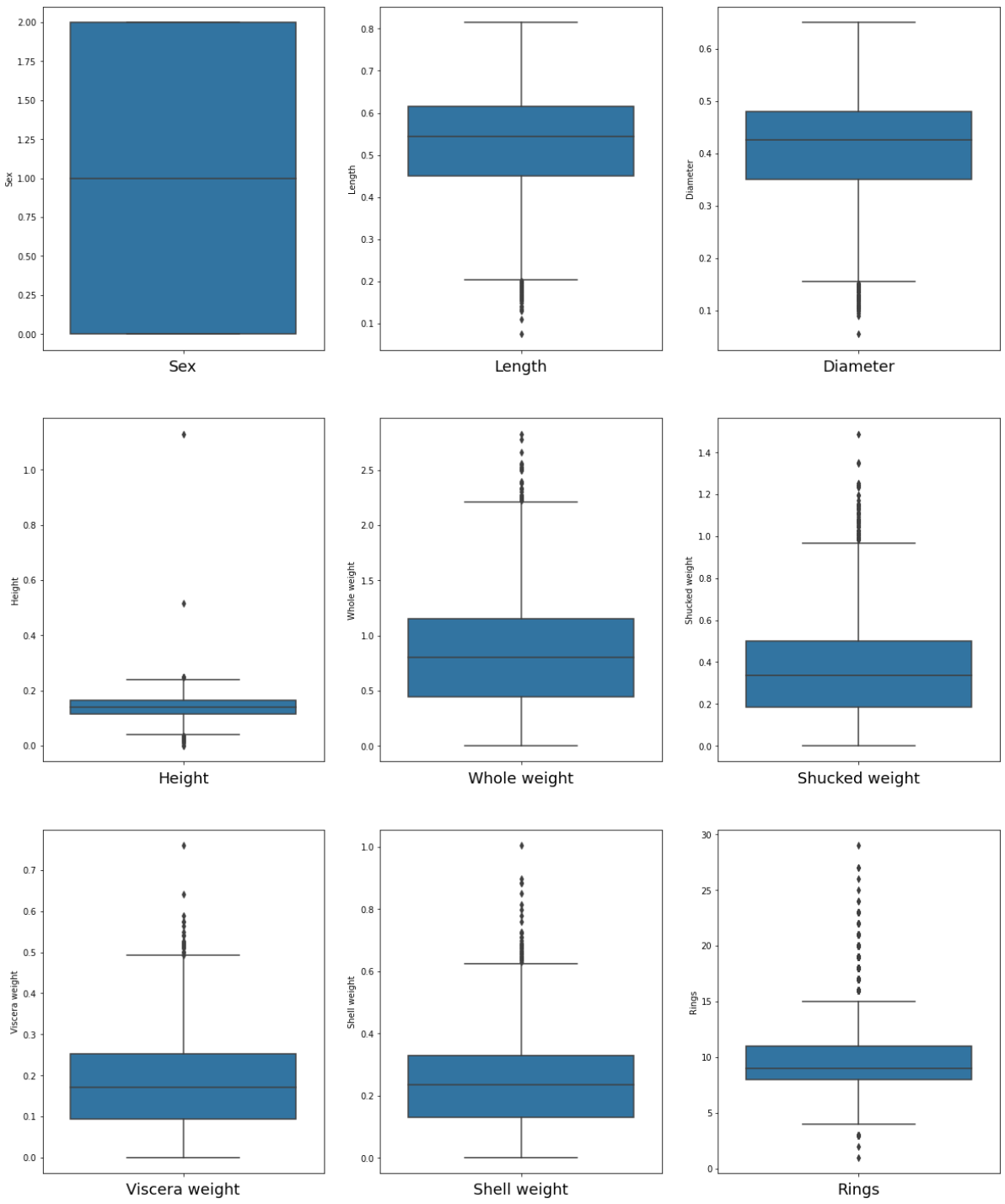
Checking Outliers

In [40]:
```
# Visualize the outliers using boxplot
print('------------------')
print('Box Plot :-')
print('------------------')

plt.figure(figsize = (20,25))
graph = 1

for column in df:
    if graph <=9:
        ax = plt.subplot(3,3, graph)
        sns.boxplot(y=df[column]) # It is the axis for vertical set as y
        plt.xlabel(column, fontsize = 18)
    graph +=1
plt.show()
```

```
------------------
Box Plot :-
------------------
```



There are outliers presents in dataset.

Removing Outliers using Zscore

In [41]:
```
# with std 3 Lets see the stats

z_score = zscore(df[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked wei
abs_z_score = np.abs(z_score)

filtering_entry = (abs_z_score < 3).all(axis = 1)

df = df[filtering_entry]

df.describe()
```

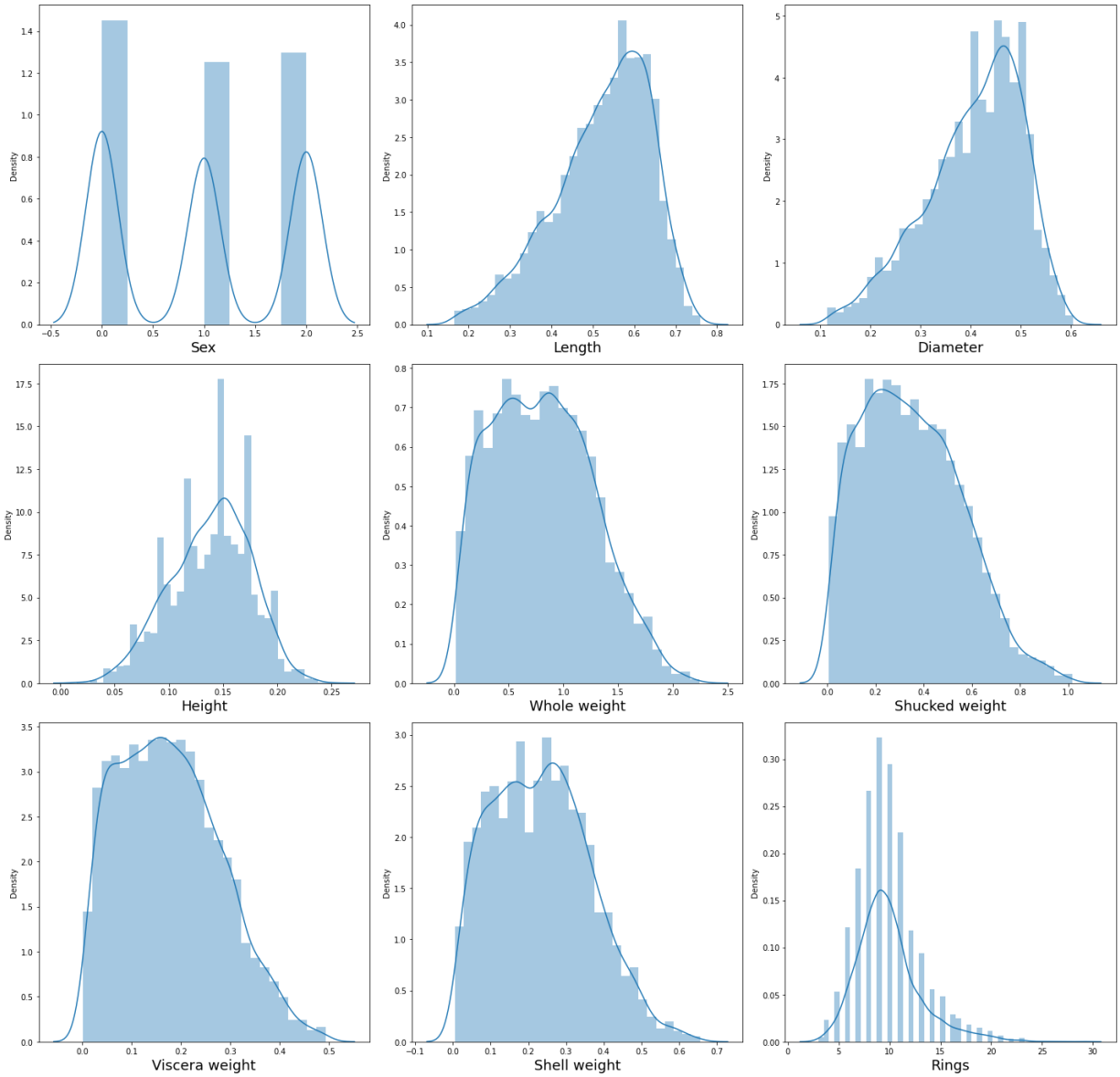
Out[41]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
count	4084.000000	4084.000000	4084.000000	4084.000000	4084.000000	4084.000000	4084.000000
mean	0.961557	0.522090	0.406308	0.138530	0.808404	0.350172	0.176579
std	0.828255	0.115709	0.095848	0.037001	0.457321	0.205827	0.103336
min	0.000000	0.165000	0.115000	0.015000	0.014500	0.005500	0.000500
25%	0.000000	0.450000	0.350000	0.115000	0.441000	0.185375	0.092875
50%	1.000000	0.540000	0.420000	0.140000	0.790750	0.332000	0.168500
75%	2.000000	0.610000	0.480000	0.165000	1.134750	0.494000	0.247000
max	2.000000	0.760000	0.605000	0.250000	2.238500	1.017000	0.492500

In [42]:
```
# Let' see outliers are removed in columns or not.
print('------------------------')
print('Distribution Plot :-')
print('------------------------')

plt.figure(figsize = (20,25), facecolor = 'white')
plotnumber = 1
for column in df:
    if plotnumber <=12:
        ax = plt.subplot(4,3, plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column, fontsize = 18)
    plotnumber += 1
plt.tight_layout()
```

```
------------------------
Distribution Plot :-
------------------------
```



It removes outliers.

In [43]:
```
print('-------------------------------------------------')
print('No of Rows and Columns ------->',df.shape)
print('-------------------------------------------------')
```

```
-------------------------------------------------
No of Rows and Columns -------> (4084, 9)
-------------------------------------------------
```

Above we check shape of remaining data after removal of outliers.

Checking for Skewness

In [46]:
```
print('----------------------------')
print('Skewness in data :-')
print('----------------------------')
print(df.skew())
print('----------------------------')
```

```
----------------------------
Skewness in data :-
----------------------------
Sex               0.071702
Length           -0.633786
Diameter         -0.605450
Height           -0.247192
Whole weight      0.323886
Shucked weight    0.449573
Viscera weight    0.429932
Shell weight      0.358512
Rings             1.136367
dtype: float64
----------------------------
```

There is some skewness but we kept it because it cause data loss.

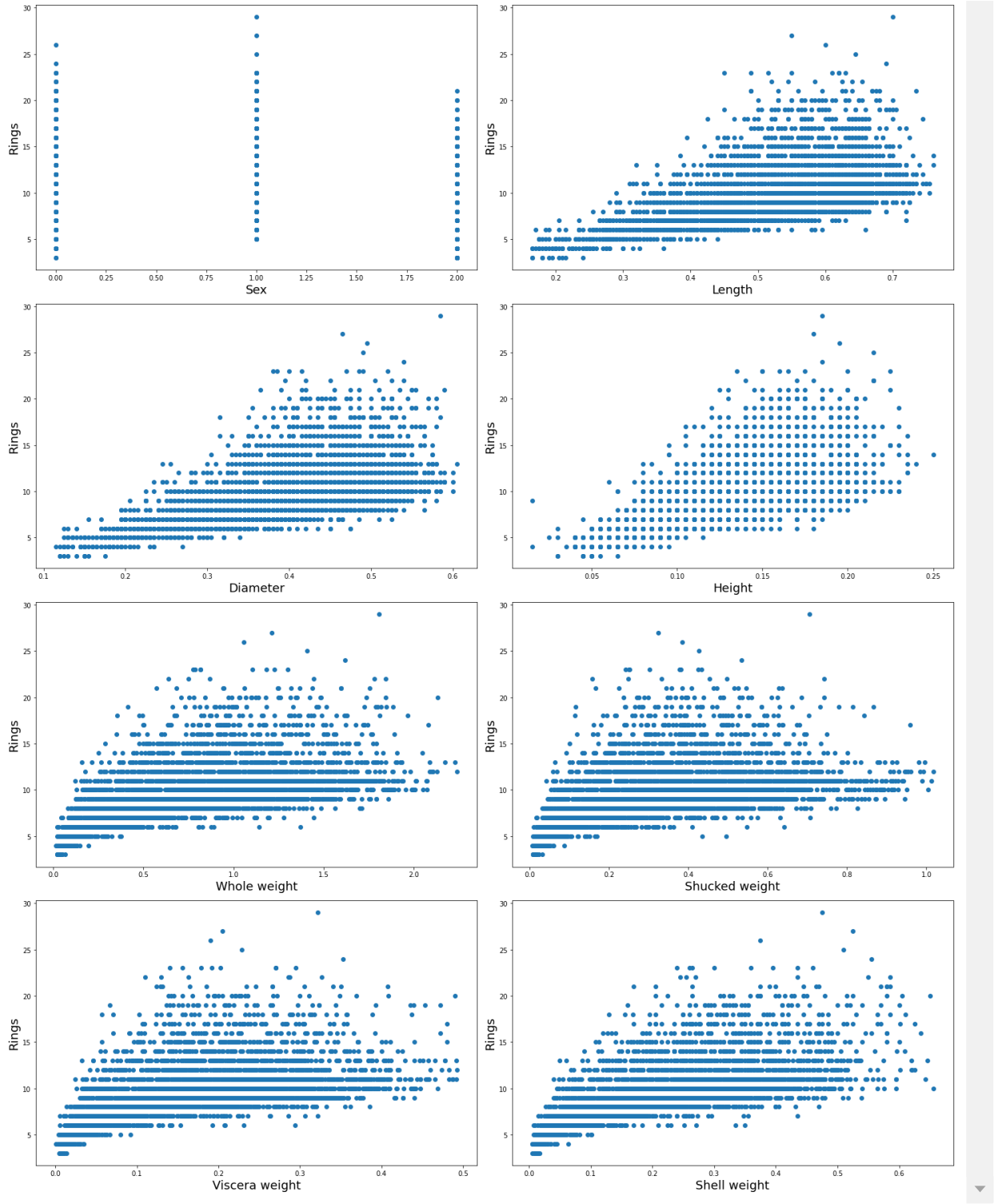
Spliting Dataset into features and labels

In [48]:
```
x = df.drop('Rings', axis = 1)
y = df.Rings
```

In [50]:
```
# Let' see relation between features and labels.
print('------------------------')
print('Distribution Plot :-')
print('------------------------')

plt.figure(figsize = (20,25), facecolor = 'white')
plotnumber = 1
for column in x:
    if plotnumber <=8:
        ax = plt.subplot(4,2, plotnumber)
        plt.scatter(x[column],y)
        plt.xlabel(column, fontsize = 18)
        plt.ylabel('Rings', fontsize = 18)
    plotnumber += 1
plt.tight_layout()
```

```
------------------------
Distribution Plot :-
------------------------
```



Features are related to class

Handling Class Imbalance

In [51]:
```
from imblearn.over_sampling import RandomOverSampler
ros  = RandomOverSampler()
x_over, y_over = ros.fit_resample(x, y)
```

In [52]:
```
print('----------------------')
print('Class are balanced :-')
print('----------------------')
print(y_over.value_counts())
print('----------------------')
```

```
----------------------
Class are balanced :-
----------------------
16    686
17    686
14    686
29    686
13    686
12    686
27    686
11    686
26    686
10    686
25    686
9     686
24    686
8     686
23    686
7     686
22    686
6     686
21    686
5     686
20    686
4     686
19    686
3     686
18    686
15    686
Name: Rings, dtype: int64
----------------------
```

Data Scaling

In [27]:
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
x_scaled
```

Out[27]:
```
array([[-1.16108558, -0.5798883 , -0.43102345, ..., -0.61064456,
        -0.73148499, -0.64218694],
       [-1.16108558, -1.48745122, -1.47447322, ..., -1.21802438,
        -1.23959876, -1.26010684],
       [ 0.04641977,  0.06837092,  0.14287393, ..., -0.45515533,
        -0.33951152, -0.17874702],
       ...,
       [-1.16108558,  0.67341287,  0.7167713 , ...,  0.85192604,
         1.07352867,  0.57820485],
       [ 0.04641977,  0.88949928,  0.82111628, ...,  0.87865076,
         0.81705219,  0.48551686],
       [-1.16108558,  1.62419307,  1.55153111, ...,  2.89272223,
         1.93490248,  2.0225926 ]])
```

Data has been scaled.

Split data into train and test. Model will be bulit on training data and tested on test data

In [53]:
```
x_train, x_test, y_train, y_test = train_test_split(x_over, y_over, test_size = 0
print('Data has been splited.')
```

```
Data has been splited.
```

Model Building

AdaBoost model instantiaing, training and evaluating

In [54]:
```
ada = AdaBoostClassifier()
ada.fit(x_train, y_train)
y_pred = ada.predict(x_test)
```

In [55]:
```
print('-----------------------------------------------------------\n')
print('Confusion Matrix :')
cfm = confusion_matrix(y_test, y_pred)
print(cfm)
print('\n-----------------------------------------------------------')
print('\nClassification Report:')
print(classification_report(y_test, y_pred, digits = 2))
print('-----------------------------------------------------------\n')
```

```
-----------------------------------------------------------

Confusion Matrix :
[[173   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [ 87  79   0   0   0   0   0   0   5   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [ 24 116   0   0   0   0   0   0  43   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  4  46   0   0   0   0   0   0 124   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  1   9   0   0   0   0   0   0 178   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   3   0   0   0   0   0   0 164   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   1   0   0   0   0   0   0 169   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 172   0   0   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   1   0   0   0   0   0   0 137   0   0   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 149   0   0   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 176   0   0   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 173   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 164   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 171   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 198   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 160   0   0   0   0   0   0   0   0   0
    2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 158   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 171   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 175   0   0   0   0   0   0   0   0   0
   16   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 167   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 182   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 162   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
```

```
 [  0   0   0   0   0   0   0   0 178   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 192   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 167   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  158   0   0   0   0   0   0   0]]

------------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           3       0.60      1.00      0.75       173
           4       0.31      0.46      0.37       171
           5       0.00      0.00      0.00       183
           6       0.00      0.00      0.00       174
           7       0.00      0.00      0.00       188
           8       0.00      0.00      0.00       167
           9       0.00      0.00      0.00       170
          10       0.00      0.00      0.00       173
          11       0.04      0.99      0.07       139
          12       0.00      0.00      0.00       150
          13       0.00      0.00      0.00       177
          14       0.00      0.00      0.00       173
          15       0.00      0.00      0.00       164
          16       0.00      0.00      0.00       171
          17       0.00      0.00      0.00       198
          18       0.00      0.00      0.00       162
          19       0.00      0.00      0.00       158
          20       0.00      0.00      0.00       171
          21       0.09      0.08      0.09       191
          22       0.00      0.00      0.00       167
          23       0.00      0.00      0.00       182
          24       0.00      0.00      0.00       162
          25       0.00      0.00      0.00       178
          26       0.00      0.00      0.00       192
          27       0.00      0.00      0.00       167
          29       0.00      0.00      0.00       158

    accuracy                           0.09      4459
   macro avg       0.04      0.10      0.05      4459
weighted avg       0.04      0.09      0.05      4459

------------------------------------------------------------
```

Conclusion : Ada Boost model has 9% score.

Cross Validation score to check if the model is overfitting

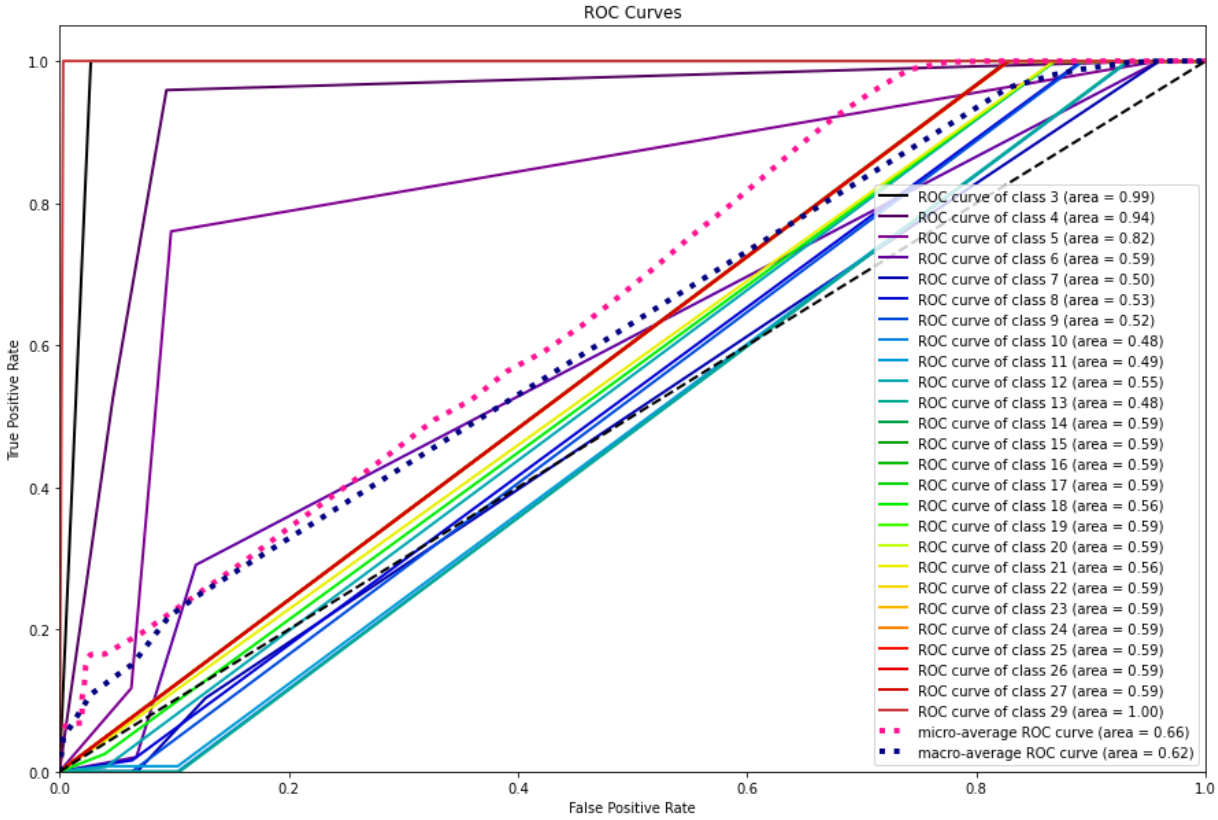
In [56]:
```
cv = cross_val_score(ada, x, y, cv = 5)
print('Cross Validation score of Ada Boost model --->', cv.mean())
```

```
Cross Validation score of Ada Boost model ---> 0.21743886048911606
```

Conclusion : Ada Boost model has 21% Cross Validation score

ROC, AUC Curve

In [51]:
```
prob = ada.predict_proba(x_test) # calculating probability
skplt.metrics.plot_roc(y_pred,prob, figsize = (15,10))
plt.show()
```



Decision Tree model instantiaing, training and evaluating

In [57]:
```
DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)
y_pred = DT.predict(x_test)
```

In [58]:
```
print('-----------------------------------------------------------\n')
print('Confusion Matrix :')
cfm = confusion_matrix(y_test, y_pred)
print(cfm)
print('\n-----------------------------------------------------------')
print('\nClassification Report:')
print(classification_report(y_test, y_pred, digits = 2))
print('-----------------------------------------------------------\n')
```

```
-----------------------------------------------------------

Confusion Matrix :
[[173   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0 171   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0 183   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   1   3 156   4   4   1   2   2   1   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   1   6   8 136  13  10   8   2   1   0   1   0   1   0   1   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   3   4  26  79  25  13   3   6   0   1   4   0   0   0   1   0
    2   0   0   0   0   0   0   0]
 [  0   0   1   3  12  29  42  34  21   8   6   5   2   1   2   1   3   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   6   4  14  31  51  18  15   7   5   8   2   4   1   5   1
    0   0   0   0   0   0   0   1]
 [  0   0   0   1   0   7  10  15  83   4   6   4   3   3   2   0   0   1
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   1   3   3   1 140   0   1   0   1   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   1   2   0   1 167   2   0   2   0   0   2   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 173   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 164   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 171   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 198   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 162   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 158   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 171
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  191   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0 167   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0 182   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0 162   0   0   0   0]
```

```
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 178   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0 192   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0 167   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0 158]]

------------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           3       1.00      1.00      1.00       173
           4       0.99      1.00      0.99       171
           5       0.93      1.00      0.97       183
           6       0.88      0.90      0.89       174
           7       0.75      0.72      0.74       188
           8       0.54      0.47      0.50       167
           9       0.34      0.25      0.29       170
          10       0.40      0.29      0.34       173
          11       0.64      0.60      0.62       139
          12       0.80      0.93      0.86       150
          13       0.90      0.94      0.92       177
          14       0.90      1.00      0.95       173
          15       0.91      1.00      0.95       164
          16       0.94      1.00      0.97       171
          17       0.96      1.00      0.98       198
          18       0.98      1.00      0.99       162
          19       0.93      1.00      0.97       158
          20       0.99      1.00      0.99       171
          21       0.99      1.00      0.99       191
          22       1.00      1.00      1.00       167
          23       1.00      1.00      1.00       182
          24       1.00      1.00      1.00       162
          25       1.00      1.00      1.00       178
          26       1.00      1.00      1.00       192
          27       1.00      1.00      1.00       167
          29       0.99      1.00      1.00       158

    accuracy                           0.89      4459
   macro avg       0.88      0.89      0.88      4459
weighted avg       0.88      0.89      0.88      4459

------------------------------------------------------------
```

Conclusion : Decision Tree model has 89% score

Cross Validation score to check if the model is overfitting

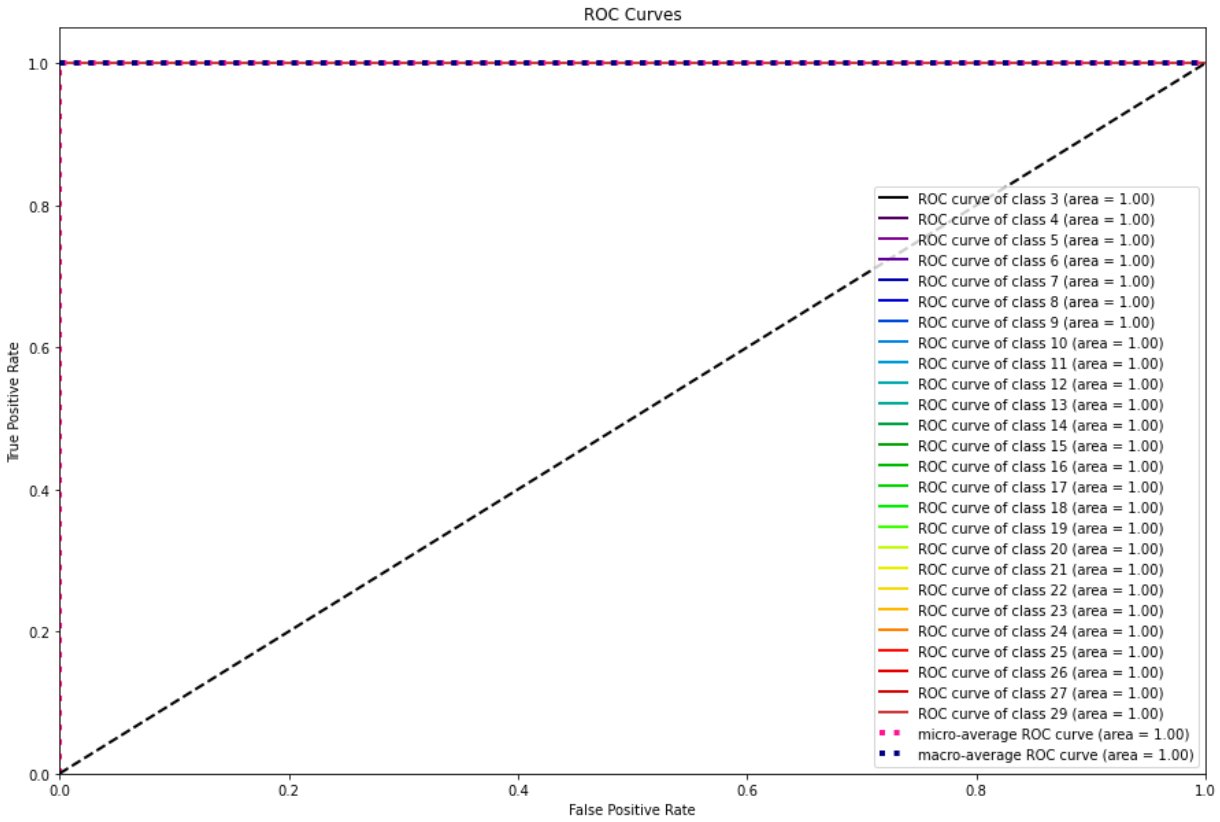
In [59]:
```
cv = cross_val_score(DT, x, y, cv = 5)
print('Cross Validation score of Decision Tree model --->', cv.mean())
```

```
Cross Validation score of Decision Tree model ---> 0.18609361125111
```

Conclusion : Decision Tree model has 18% Cross Validation score

ROC, AUC Curve

In [60]:
```
prob = DT.predict_proba(x_test) # calculating probability
skplt.metrics.plot_roc(y_pred,prob, figsize = (15,10))
plt.show()
```



Knn model instantiaing, training and evaluating

In [61]:
```
Knn = KNeighborsClassifier()
Knn.fit(x_train, y_train)
y_pred = Knn.predict(x_test)
```

In [62]:
```
print('-----------------------------------------------------------\n')
print('Confusion Matrix :')
cfm = confusion_matrix(y_test, y_pred)
print(cfm)
print('\n-----------------------------------------------------------')
print('\nClassification Report:')
print(classification_report(y_test, y_pred, digits = 2))
print('-----------------------------------------------------------')
```

```
-----------------------------------------------------------

Confusion Matrix :
[[173   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0 171   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  1   2 168  12   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  2  14  18 108  28   1   0   1   1   0   0   0   1   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   2  13  52  77  16   8   4   2   5   4   3   1   1   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   1   6  15  30  55  19   7   7   8   5   6   6   2   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   1   5  19  30  39  19  17  15   6  11   5   2   0   1   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   2   6  22  29  20  22  25  11  11  14   1   2   3   3   2   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   1   2   5  15  11  36  25  10   6   8   5   3   3   5   2   1
    1   0   0   0   0   0   0   0]
 [  0   0   0   1   4   4  14   5  19  65  11   8   1   5   4   4   0   1   2
    2   0   0   0   0   0   0   0]
 [  0   0   0   0   1   2   4   7  13   8 111  10   4   2   4   0   3   5   0
    0   3   0   0   0   0   0   0]
 [  0   0   0   1   0   0   1   3   4   1   7 149   3   0   3   0   0   1   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   1   3   1   2 154   2   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 171   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   8   0 190   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 162   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 158   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 171   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 191
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  167   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0 182   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0 162   0   0   0   0   0]
```

```
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 178   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0 192   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0 167   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0 158]]

------------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           3       0.98      1.00      0.99       173
           4       0.90      1.00      0.95       171
           5       0.82      0.92      0.86       183
           6       0.55      0.62      0.58       174
           7       0.46      0.41      0.43       188
           8       0.41      0.33      0.36       167
           9       0.30      0.23      0.26       170
          10       0.26      0.12      0.16       173
          11       0.30      0.26      0.28       139
          12       0.42      0.43      0.43       150
          13       0.67      0.63      0.65       177
          14       0.72      0.86      0.79       173
          15       0.75      0.94      0.83       164
          16       0.90      1.00      0.94       171
          17       0.92      0.96      0.94       198
          18       0.94      1.00      0.97       162
          19       0.93      1.00      0.97       158
          20       0.94      1.00      0.97       171
          21       0.98      1.00      0.99       191
          22       0.98      1.00      0.99       167
          23       0.98      1.00      0.99       182
          24       1.00      1.00      1.00       162
          25       1.00      1.00      1.00       178
          26       1.00      1.00      1.00       192
          27       1.00      1.00      1.00       167
          29       1.00      1.00      1.00       158

    accuracy                           0.80      4459
   macro avg       0.77      0.80      0.78      4459
weighted avg       0.78      0.80      0.79      4459

------------------------------------------------------------
```

Conclusion : Knn model has 80% score

Cross Validation score to check if the model is overfitting

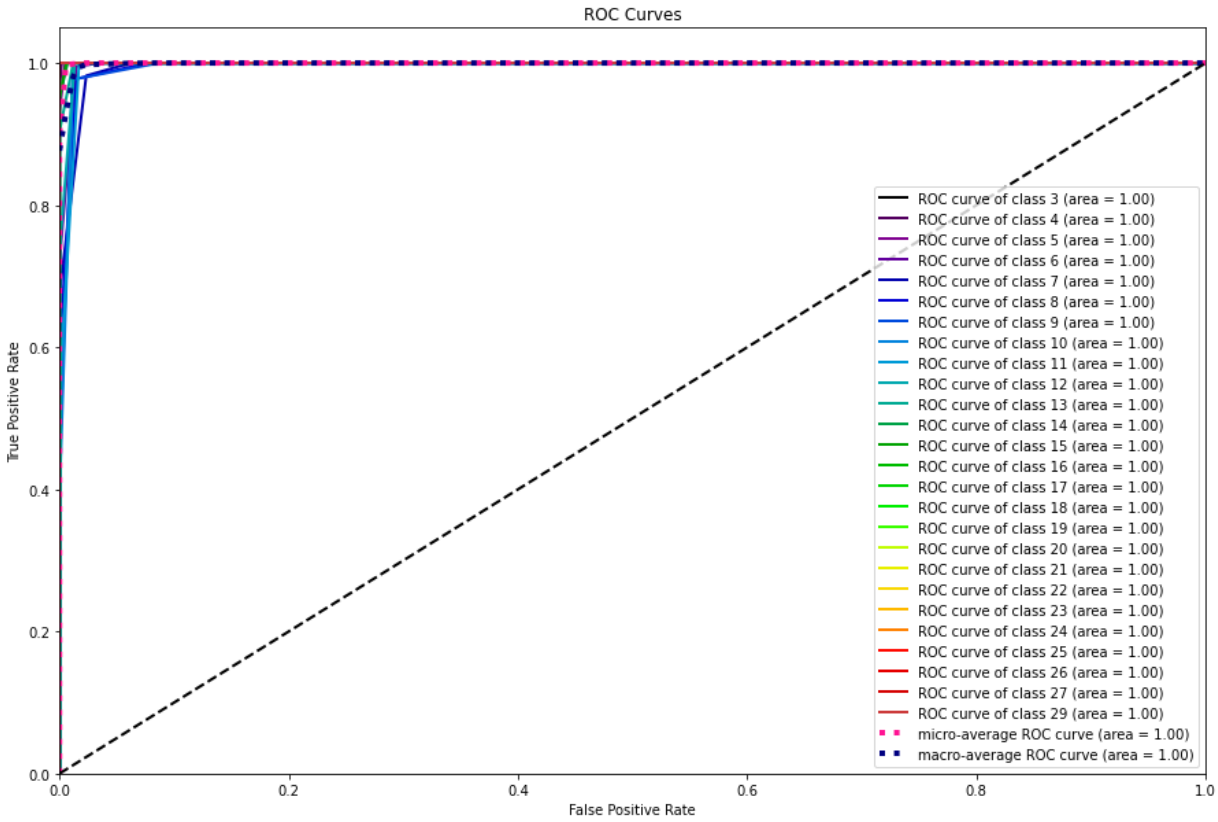
In [63]:
```
cv = cross_val_score(Knn, x, y, cv = 5)
print('Cross Validation score of Knn model --->', cv.mean())
```

```
Cross Validation score of Knn model ---> 0.23482492140062877
```

Conclusion : Knn model has 23% Cross Validation score

ROC, AUC Curve

In [64]:
```
prob = Knn.predict_proba(x_test) # calculating probability
skplt.metrics.plot_roc(y_pred,prob, figsize = (15,10))
plt.show()
```



Random Forest model instantiaing, training and evaluating

In [65]:
```
Rn = RandomForestClassifier()
Rn.fit(x_train, y_train)
y_pred = Rn.predict(x_test)
```

In [66]:
```
print('-----------------------------------------------------------\n')
print('Confusion Matrix :')
cfm = confusion_matrix(y_test, y_pred)
print(cfm)
print('\n-----------------------------------------------------------')
print('\nClassification Report:')
print(classification_report(y_test, y_pred, digits = 2))
print('-----------------------------------------------------------')
```

```
-----------------------------------------------------------

Confusion Matrix :
[[173   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0 171   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0 183   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  2   1   1 160   8   1   1   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   1   2  14 136  19  10   5   0   0   0   0   0   0   0   1   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   5   5  25  88  29   7   4   1   2   0   1   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   1   0   4  15  37  51  29  19   5   5   2   1   1   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   5  15  40  62  27   9   3   3   4   1   0   0   2   1
    0   0   0   0   0   0   0   1]
 [  0   0   0   1   2   7  12  10  85   8   8   2   1   1   1   0   1   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   1   1   1   1   4 139   0   1   1   0   1   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   1   5   2   0 165   2   0   1   0   0   1   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 173   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 164   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 171   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 198   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 162   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 158   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 171
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  191   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0 167   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0 182   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0 162   0   0   0   0]
```

```
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 178   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0 192   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0 167   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0 158]]

------------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           3       0.99      1.00      0.99       173
           4       0.98      1.00      0.99       171
           5       0.96      1.00      0.98       183
           6       0.87      0.92      0.89       174
           7       0.71      0.72      0.72       188
           8       0.52      0.53      0.53       167
           9       0.35      0.30      0.32       170
          10       0.52      0.36      0.42       173
          11       0.60      0.61      0.61       139
          12       0.86      0.93      0.89       150
          13       0.90      0.93      0.92       177
          14       0.95      1.00      0.97       173
          15       0.95      1.00      0.98       164
          16       0.98      1.00      0.99       171
          17       0.99      1.00      0.99       198
          18       0.99      1.00      1.00       162
          19       0.98      1.00      0.99       158
          20       0.99      1.00      1.00       171
          21       1.00      1.00      1.00       191
          22       1.00      1.00      1.00       167
          23       1.00      1.00      1.00       182
          24       1.00      1.00      1.00       162
          25       1.00      1.00      1.00       178
          26       1.00      1.00      1.00       192
          27       1.00      1.00      1.00       167
          29       0.99      1.00      1.00       158

    accuracy                           0.90      4459
   macro avg       0.89      0.90      0.89      4459
weighted avg       0.89      0.90      0.89      4459

------------------------------------------------------------
```

Conclusion : Random Forest model has 90% score

Cross Validation score to check if the model is overfitting

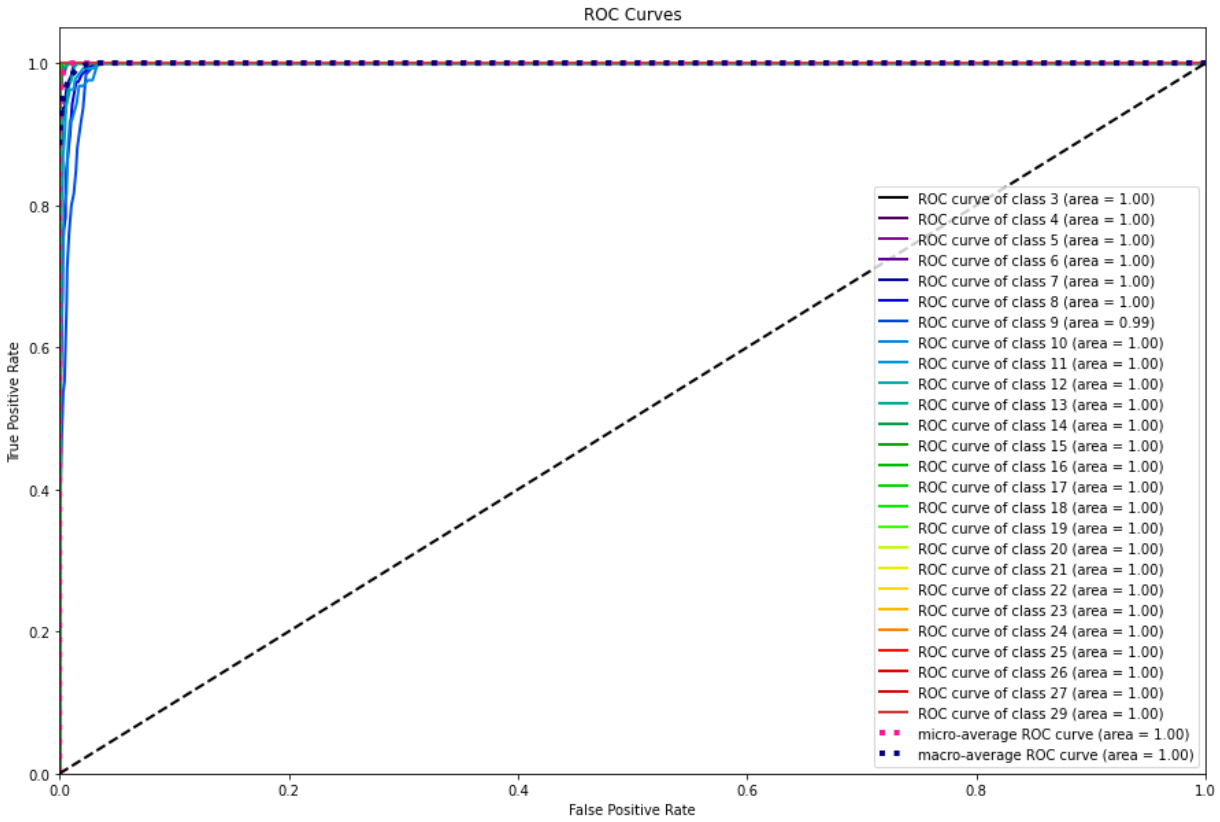
In [67]:
```
cv = cross_val_score(Rn, x, y, cv = 5)
print('Cross Validation score of Random Forest model --->', cv.mean())
```

```
Cross Validation score of Random Forest model ---> 0.253187774497804
```

Conclusion : Random Forest model has 25% Cross Validation score

ROC, AUC Curve

In [54]:
```
prob = Rn.predict_proba(x_test) # calculating probability
skplt.metrics.plot_roc(y_pred,prob, figsize = (15,10))
plt.show()
```



Ada Boost Hyperparameter Tuning .

In [70]:
```
param = {'n_estimators': [50, 100],'learning_rate' : [0.30,0.40,0.45]}
```

In [71]:
```
grid_search = GridSearchCV(estimator = ada, param_grid = param, cv = 5,n_jobs = -
```

In [72]:
```
grid_search.fit(x_train, y_train)
```

Out[72]:
```
GridSearchCV(cv=5, estimator=AdaBoostClassifier(), n_jobs=-1,
             param_grid={'learning_rate': [0.3, 0.4, 0.45],
                         'n_estimators': [50, 100]})
```

In [73]:
```
best_parameters = grid_search.best_params_
print(best_parameters)
```

```
{'learning_rate': 0.3, 'n_estimators': 50}
```

In [74]:
```
hada = AdaBoostClassifier(learning_rate = 0.3, n_estimators= 50)
hada.fit(x_train, y_train)
hada.score(x_test, y_test)
```

Out[74]:
```
0.12648575913882038
```

In [75]:
```
y_pred = hada.predict(x_test)
```

In [76]:
```
print('-----------------------------------------------------------\n')
print('\nClassification Report:')
print(classification_report(y_test, y_pred, digits = 2))
print('-----------------------------------------------------------')
```

```
-----------------------------------------------------------


Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00       173
           4       0.00      0.00      0.00       171
           5       0.00      0.00      0.00       183
           6       0.17      0.80      0.28       174
           7       0.00      0.00      0.00       188
           8       0.19      0.25      0.22       167
           9       0.38      0.11      0.17       170
          10       0.33      0.01      0.01       173
          11       0.00      0.00      0.00       139
          12       0.15      0.04      0.06       150
          13       0.05      0.07      0.06       177
          14       0.00      0.00      0.00       173
          15       0.36      0.02      0.05       164
          16       0.08      0.11      0.09       171
          17       0.01      0.02      0.01       198
          18       0.13      0.22      0.16       162
          19       0.15      0.37      0.21       158
          20       0.15      0.42      0.22       171
          21       0.08      0.20      0.12       191
          22       0.00      0.00      0.00       167
          23       0.14      0.64      0.23       182
          24       0.00      0.00      0.00       162
          25       0.00      0.00      0.00       178
          26       0.00      0.00      0.00       192
          27       0.00      0.00      0.00       167
          29       0.00      0.00      0.00       158

    accuracy                           0.13      4459
   macro avg       0.09      0.13      0.07      4459
weighted avg       0.09      0.13      0.07      4459

-----------------------------------------------------------
```

After Hyperparameter Tuning Ada Boost model accuracy score 13%

.

Saving The Model

In [77]:
```
# saving the model to the Local file system
filename = 'Abalone Case.pickle'
pickle.dump(hada, open(filename, 'wb'))
```

Knn Hyperparameter Tuning model

In [78]:
```
grid_param = {'leaf_size' : [1,3,5], 'n_neighbors': [5],
              'p':[1,2]}
```

In [79]:
```
grid_search = GridSearchCV(estimator = Knn, param_grid = grid_param, cv = 5 , n_j
```

In [80]:
```
grid_search.fit(x_train, y_train)
```

Out[80]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(), n_jobs=-1,
             param_grid={'leaf_size': [1, 3, 5], 'n_neighbors': [5],
                         'p': [1, 2]})
```

In [81]:
```
best_parameters = grid_search.best_params_
print(best_parameters)
```

```
{'leaf_size': 3, 'n_neighbors': 5, 'p': 1}
```

In [82]:
```
hkn = KNeighborsClassifier(leaf_size = 1, n_neighbors = 5, p = 1)
hkn.fit(x_train, y_train)
hkn.score(x_test, y_test)
```

Out[82]:
```
0.8004036779546984
```

In [83]:
```
y_pred = hkn.predict(x_test)
```

In [84]:
```
print('-----------------------------------------------------------')
print('\nClassification Report:')
print(classification_report(y_test, y_pred, digits = 2))
print('-----------------------------------------------------------')
```

```
-----------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           3       0.98      1.00      0.99       173
           4       0.91      1.00      0.96       171
           5       0.80      0.91      0.85       183
           6       0.54      0.60      0.57       174
           7       0.45      0.45      0.45       188
           8       0.44      0.35      0.39       167
           9       0.27      0.19      0.23       170
          10       0.28      0.13      0.18       173
          11       0.24      0.20      0.22       139
          12       0.42      0.43      0.42       150
          13       0.69      0.63      0.66       177
          14       0.70      0.87      0.78       173
          15       0.79      0.92      0.85       164
          16       0.88      1.00      0.93       171
          17       0.93      0.96      0.94       198
          18       0.93      1.00      0.96       162
          19       0.92      1.00      0.96       158
          20       0.95      1.00      0.97       171
          21       0.98      1.00      0.99       191
          22       0.98      1.00      0.99       167
          23       0.98      1.00      0.99       182
          24       1.00      1.00      1.00       162
          25       1.00      1.00      1.00       178
          26       1.00      1.00      1.00       192
          27       1.00      1.00      1.00       167
          29       1.00      1.00      1.00       158

    accuracy                           0.80      4459
   macro avg       0.77      0.79      0.78      4459
weighted avg       0.78      0.80      0.79      4459

-----------------------------------------------------------
```

After Hyperparameter Tuning model accuracy score increase to 80% .

Saving The Model ¶

In [85]:
```
# saving the model to the Local file system
filename = 'Knn Abaline case.pickle'
pickle.dump(hkn, open(filename, 'wb'))
```

Final Conclusion : Knn is our best model.