IBM-EPBL / **IBM-Project-11096-1659262405** Public

Code Issues Pull requests Actions Projects Security Insights

main

IBM-Project-11096-1659262405 / Assignments / Team Lead - Sajen Sarvajith K / **Sajen Sarvajith K - Assignment 4.ipynb**

sajenjeshan1222 Create Sajen Sarvajith K - Assignment 4.ipynb

1 contributor

```
1.81 MB
```

In [128…

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

In [0]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

In [130…

```
# reading the data

data = pd.read_csv('drive/My Drive/Projects/practice/Abalone Age/abalone.csv

# getting the shape
data.shape
```

Out[130…

```
(4177, 9)
```

In [131…

```
# looking at the head of the data

data.head()
```

Out[131…

	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

In [132…

```
# describe the data

data.describe()
```

Out[132…

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	417
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	(
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	(
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	(
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	(
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	(
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	(
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	

```
max    0.815000    0.650000    1.130000    2.825500    1.488000    0.760000
```

In [133…

```
# information of the data

data.info()
```

```
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
Sex                 4177 non-null object
Length              4177 non-null float64
Diameter            4177 non-null float64
Height              4177 non-null float64
Whole weight        4177 non-null float64
Shucked weight      4177 non-null float64
Viscera weight      4177 non-null float64
Shell weight        4177 non-null float64
Rings               4177 non-null int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

In [134…

```
# checking if there is any NULL data

data.isnull().sum()
```

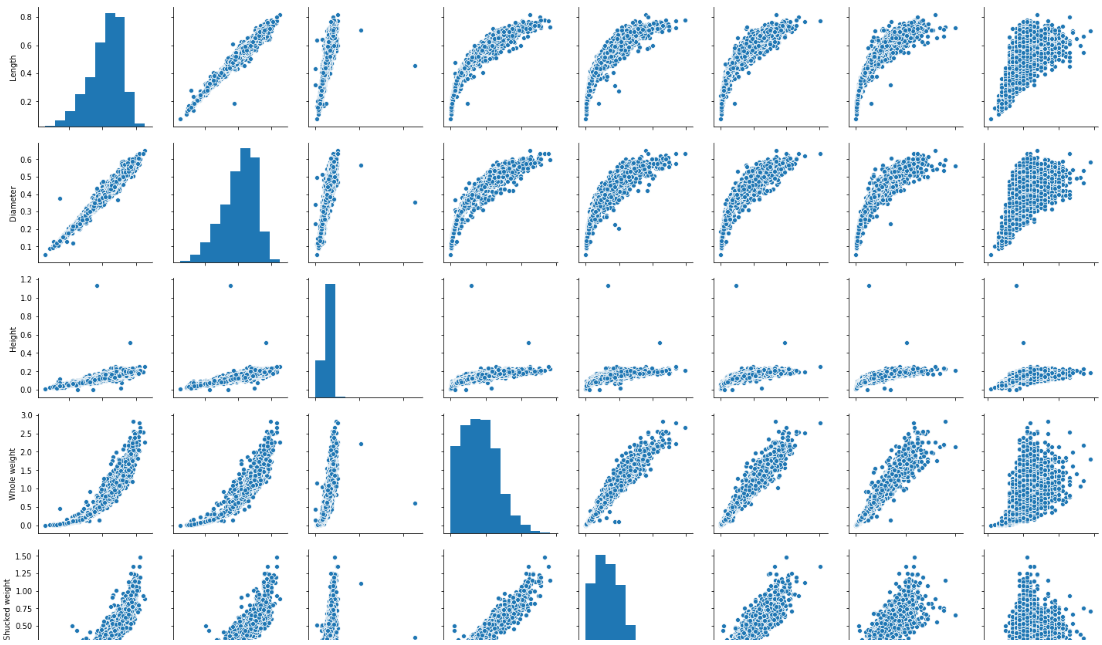
Out[134…

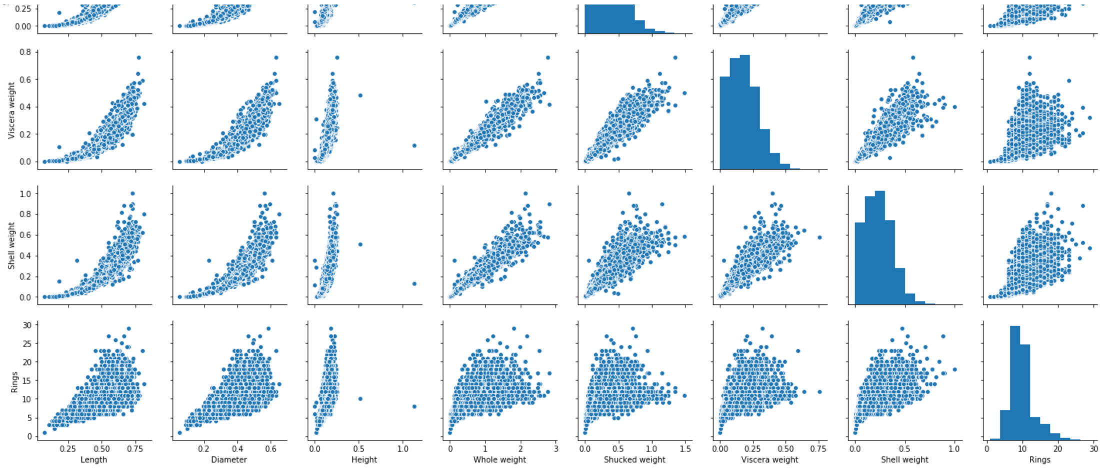
```
Sex                 0
Length              0
Diameter            0
Height              0
Whole weight        0
Shucked weight      0
Viscera weight      0
Shell weight        0
Rings               0
dtype: int64
```

In [135…

```
# pairplot

sns.pairplot(data)
```

Out[135…





In [136…

```
# checking the columns of the data

data.columns
```

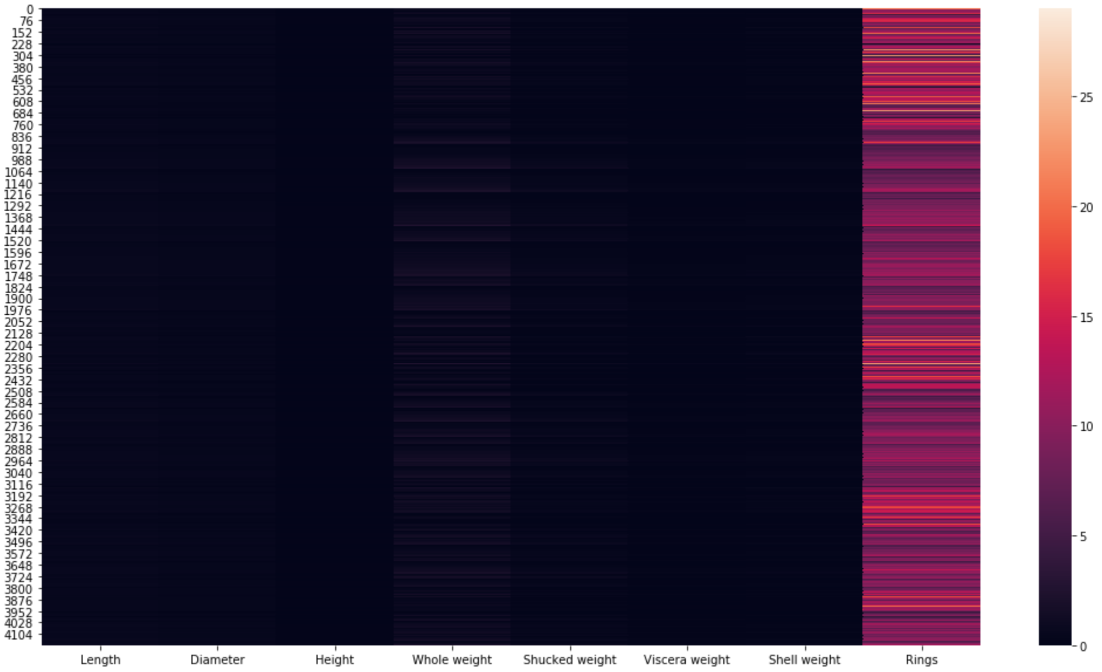
Out[136…

```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weigh
t',
       'Viscera weight', 'Shell weight', 'Rings'],
      dtype='object')
```

In [137…

```
# heatmap

sns.heatmap(data[[ 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked
        'Viscera weight', 'Shell weight', 'Rings']])
```

Out[137…



In [138…

```
# checkig the values of sex

data['Sex'].value_counts()
```

Out[138…

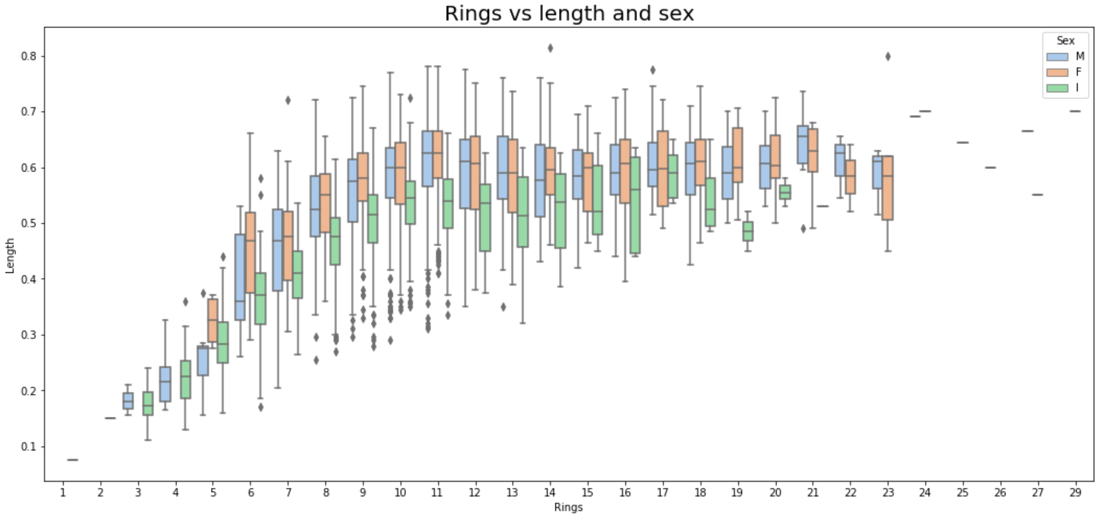
```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [139…

```
# plotting a hue plot

plt.rcParams['figure.figsize'] = (18, 8)
sns.boxplot(data['Rings'], data['Length'], hue = data['Sex'], palette = 'pas
plt.title('Rings vs length and sex', fontsize = 20)
```

Out[139…

```
Text(0.5, 1.0, 'Rings vs length and sex')
```

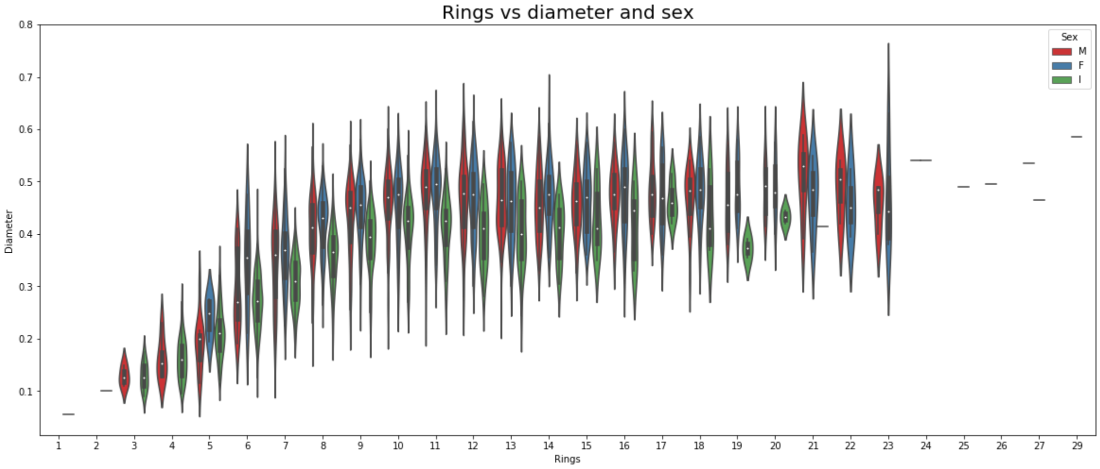


In [140…

```
# rings vs diameter and sex

plt.rcParams['figure.figsize'] = (20, 8)
sns.violinplot(data['Rings'], data['Diameter'], hue = data['Sex'], palette =
plt.title('Rings vs diameter and sex', fontsize = 20)
```

Out[140…

```
Text(0.5, 1.0, 'Rings vs diameter and sex')
```

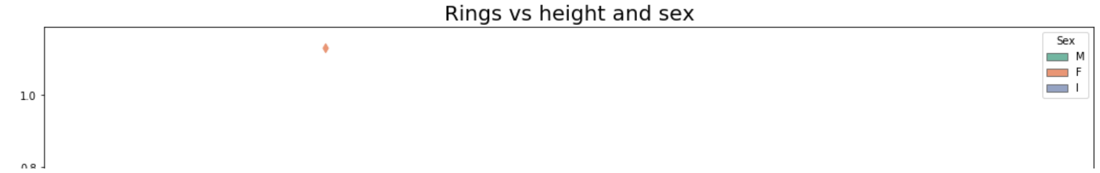


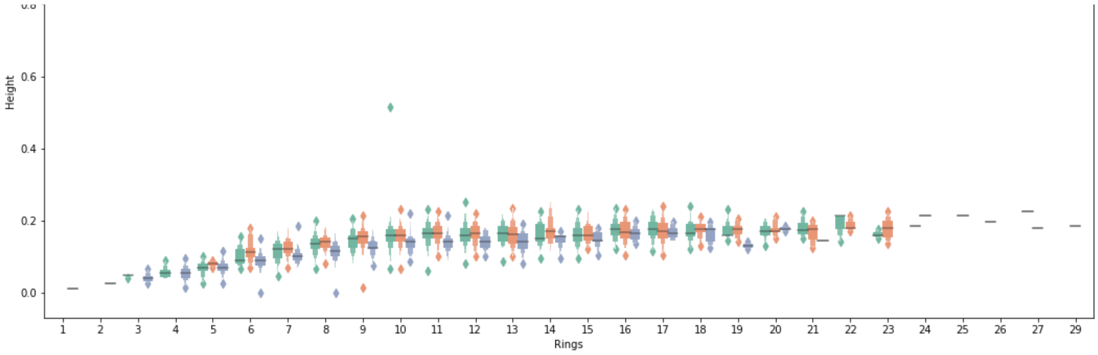
In [141…

```
# rings vs height and sex

plt.rcParams['figure.figsize'] = (18, 8)
sns.boxenplot(data['Rings'], data['Height'], hue = data['Sex'], palette = 'S
plt.title('Rings vs height and sex', fontsize = 20)
```

Out[141…

```
Text(0.5, 1.0, 'Rings vs height and sex')
```



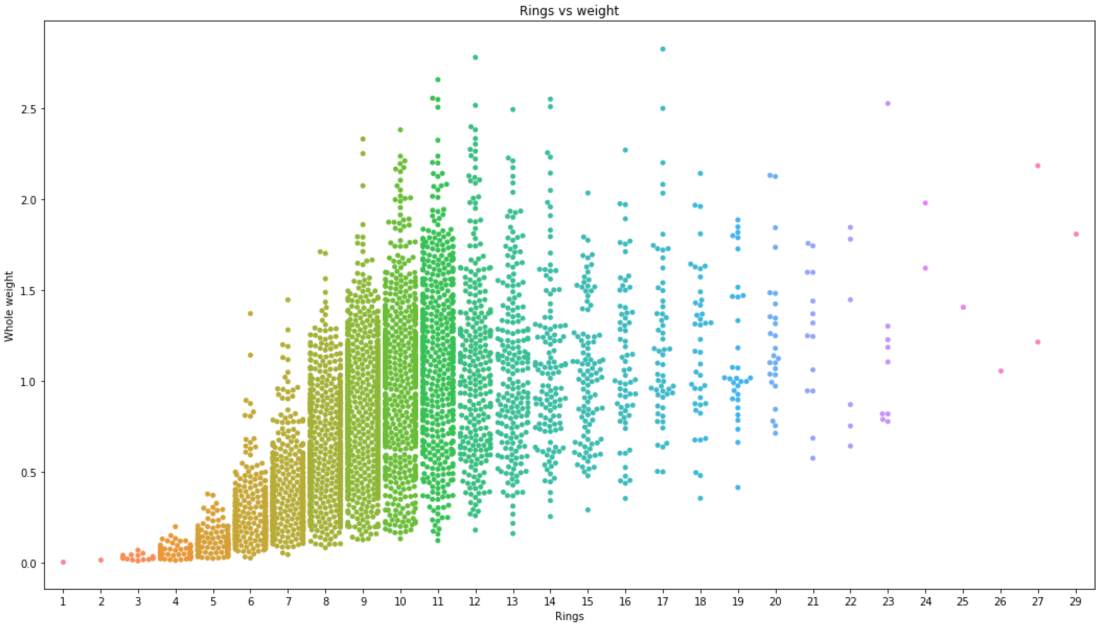


In [142…

```
# ring vs weight

plt.rcParams['figure.figsize'] = (18, 10)
sns.swarmplot(data['Rings'], data['Whole weight'])
plt.title('Rings vs weight')
```

Out[142…

```
Text(0.5, 1.0, 'Rings vs weight')
```

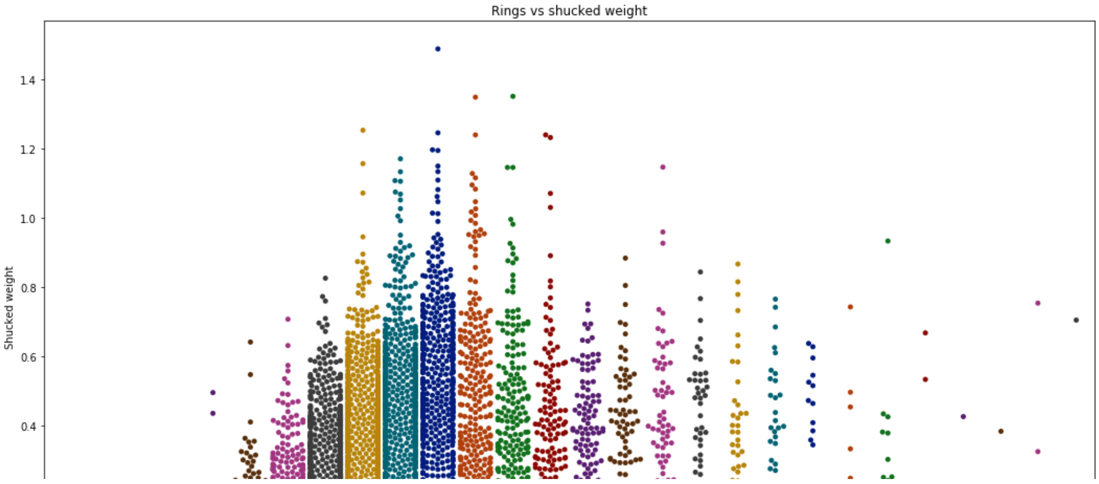


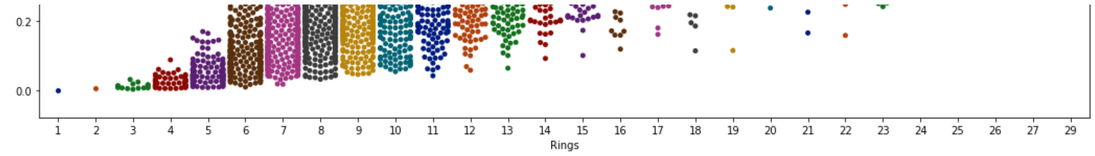
In [143…

```
# ring vs shucked weight

plt.rcParams['figure.figsize'] = (18, 10)
sns.swarmplot(data['Rings'], data['Shucked weight'], palette = 'dark')
plt.title('Rings vs shucked weight')
```

Out[143…

```
Text(0.5, 1.0, 'Rings vs shucked weight')
```



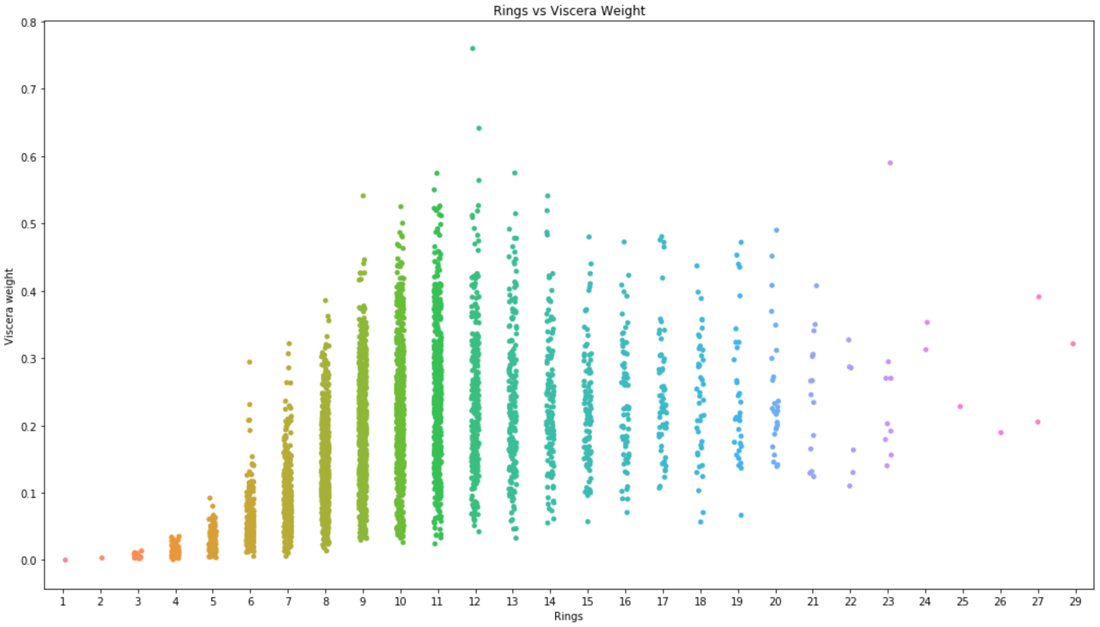


In [144…

```
# ring vs viscera weight

plt.rcParams['figure.figsize'] = (18, 10)
sns.stripplot(data['Rings'], data['Viscera weight'])
plt.title('Rings vs Viscera Weight')
```

Out[144…

```
Text(0.5, 1.0, 'Rings vs Viscera Weight')
```

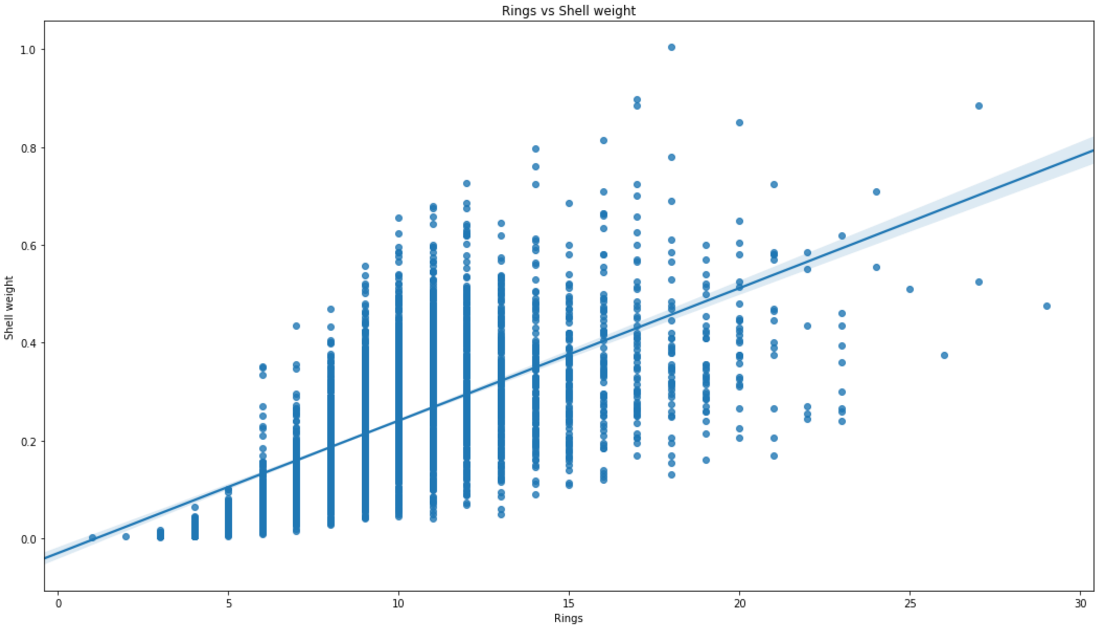


In [145…

```
# ring vs shell weight

plt.rcParams['figure.figsize'] = (18, 10)
sns.regplot(data['Rings'], data['Shell weight'])
plt.title('Rings vs Shell weight')
```

Out[145…

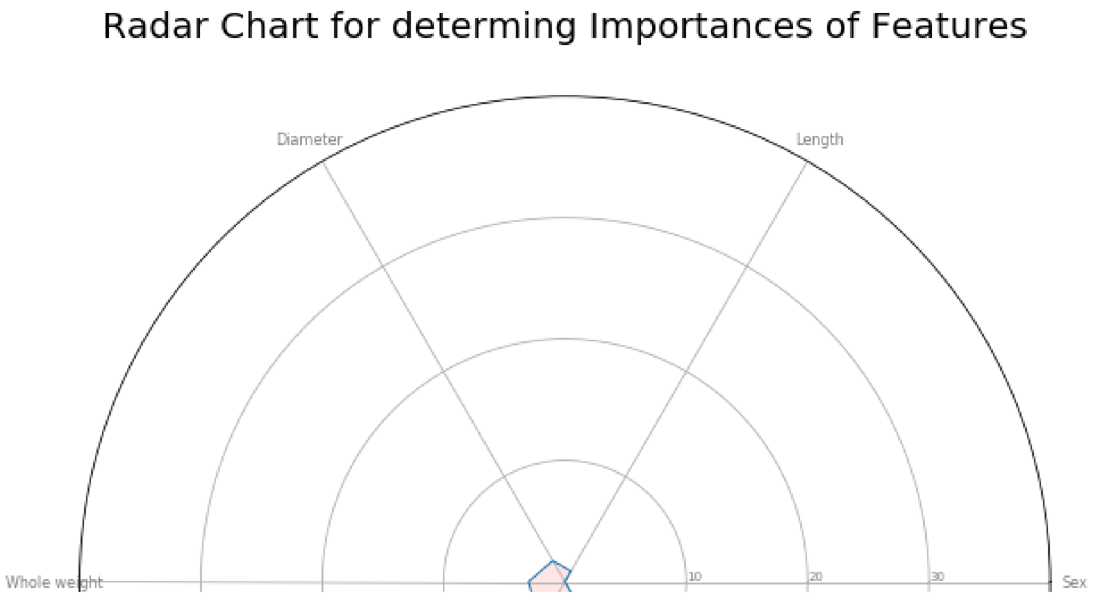
```
Text(0.5, 1.0, 'Rings vs Shell weight')
```

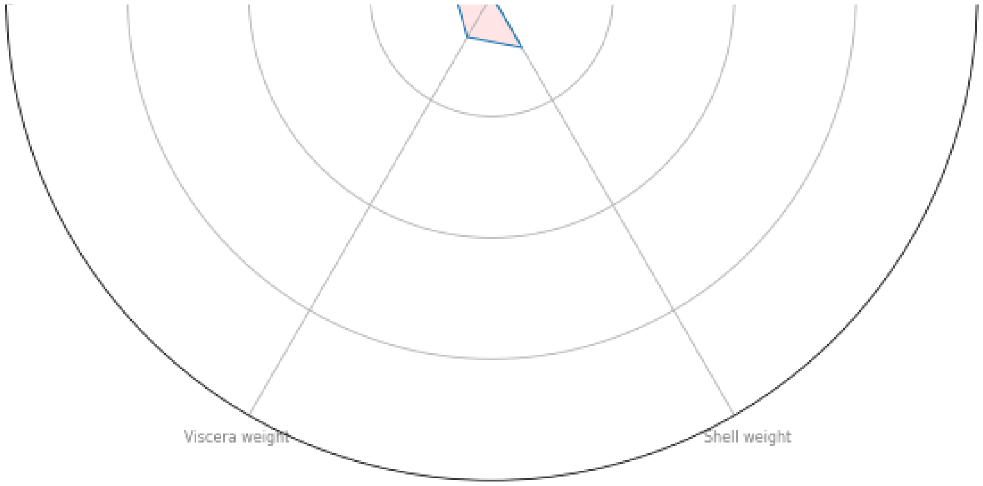


In [146…

```
from math import pi

# Set data
df = pd.DataFrame({
'group': [i for i in range(0, 4177)],
'Sex': data['Sex'],
'Length': data['Length'],
'Diameter': data['Diameter'],
'Whole weight':  data['Whole weight'],
'Viscera weight': data['Viscera weight'],
'Shell weight': data['Shell weight']
})

# number of variable
categories=list(df)[1:]
N = len(categories)

# We are going to plot the first line of the data frame.
# But we need to repeat the first value to close the circular graph:
values = df.loc[0].drop('group').values.flatten().tolist()
values += values[:1]
values

# What will be the angle of each axis in the plot? (we divide the plot / num
angles = [n / float(N) * 2 * pi for n in range(N)]
angles += angles[:1]

# Initialise the spider plot
ax = plt.subplot(111, polar=True)

# Draw one axe per variable + add labels labels yet
plt.xticks(angles[:-1], categories, color='grey', size=8)

# Draw ylabels
ax.set_rlabel_position(0)
plt.yticks([10,20,30], ["10","20","30"], color="grey", size=7)
plt.ylim(0,40)

# Plot data
ax.plot(angles, values, linewidth=1, linestyle='solid')
plt.title('Radar Chart for determing Importances of Features', fontsize = 20
# Fill area
ax.fill(angles, values, 'red', alpha=0.1)
```

Out[146… []





In [0]:

```
'''
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Sex'] = le.fit_transform(data['Sex'])

data['Sex'].value_counts()
'''

data = pd.get_dummies(data)
```

In [148…

```
data.head()
```

Out[148…

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

In [149…

```
# splitting the dependent and independent variables

y = data['Rings']
data = data.drop(['Rings'], axis = 1)
x = data

# getting the shapes
print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
```

```
Shape of x: (4177, 10)
Shape of y: (4177,)
```

In [150…

```
# train test split

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, r
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, r

# getting the shapes
print("Shape of x_train :", x_train.shape)
print("Shape of x_test :", x_test.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of y_test :", y_test.shape)
```

```
Shape of x_train : (3341, 10)
Shape of x_test : (836, 10)
Shape of y_train : (3341,)
Shape of y_test : (836,)
```

In [151…

```
# MODELLING
# RANDOM FOREST REGRESSOR

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

# evaluation
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("RMSE :", rmse)

# r2 score
r2 = r2_score(y_test, y_pred)
print("R2 Score :", r2)
```

```
RMSE : 2.7208656798612623
R2 Score : 0.3183335204159876
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:246: Futur
eWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [152…

```
!pip install eli5
```

```
Requirement already satisfied: eli5 in /usr/local/lib/python3.6/dist-package
s (0.8.2)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.6/dist
-packages (from eli5) (19.1.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from eli5) (1.11.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packa
ges (from eli5) (2.10.1)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist
-packages (from eli5) (1.16.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-pac
kages (from eli5) (0.10.1)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.6/d
ist-packages (from eli5) (0.8.3)
Requirement already satisfied: typing in /usr/local/lib/python3.6/dist-packa
ges (from eli5) (3.6.6)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.
6/dist-packages (from eli5) (0.20.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packag
es (from eli5) (1.2.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/
```

```
dist-packages (from jinja2->eli5) (1.1.1)
```

In [153…

```
# let's check the importance of each attributes


#for purmutation importance
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(model, random_state = 0).fit(x_test, y_test)
eli5.show_weights(perm, feature_names = x_test.columns.tolist())
```

Out[153…

```
          Weight    Feature
 0.0335 ± 0.0291    Shell weight
 0.0079 ± 0.0096    Viscera weight
 0.0012 ± 0.0183    Shucked weight
-0.0005 ± 0.0167    Length
-0.0041 ± 0.0105    Sex_M
-0.0050 ± 0.0079    Sex_I
-0.0067 ± 0.0158    Diameter
-0.0079 ± 0.0175    Whole weight
-0.0100 ± 0.0123    Sex_F
-0.0136 ± 0.0211    Height
```

In [0]: