Assignment #1 CS-697AB ML

Name: Harsha Siddapura Gnaneshwara
WsuID: W786P696
Email: hxsiddapuragnaneshwara@shockers.wichita.edu

1. Fit a predictive linear regression model to estimate weight of the fish from its length, height and width? (the data source fish.csv can be found here: https://www.kaggle.com/aungpyaeap/fish-market) (50 points)

   -Report the coefficients values by using the standard Least Square Estimates

```
regressor.coef_
print("coefficients values by using the standard Least Square Estimates are -\n", regressor.coef_)
```

```
coefficients values by using the standard Least Square Estimates are -
 [102.36098666 -33.89771707 -33.96666471  28.4816904   -0.87430779]
```

   -What is the standard error of the estimated coefficients, R-squared term ?

```
regressor.coef_
print("Coefficients are", regressor.coef_)
print("R-squared term value is " + str(r2_score(y_pred, y_test)))
print("The Mean Squared Error is " + str(mean_squared_error(y_test, y_pred)))
```

```
Coefficients are [102.36098666 -33.89771707 -33.96666471  28.4816904   -0.87430779]
R-squared term value is 0.8035095525723281
The Mean Squared Error is 21814.86179533342
```

- 95% confidence interval

```
#Least square as an optimizer
import statsmodels.api as sm
results = sm.OLS(y_train, X_train).fit()
print("The 95% confidence interval:", results.conf_int(0.05))
results.summary()
```

```
The 95% confidence interval:                      0           1
Length1   106.185590  456.191354
Length2  -326.643326   17.311038
Length3  -156.852512  -18.644847
Height     23.786320   93.703741
Width    -152.876759   11.970048
```

1]:

OLS Regression Results

Dep. Variable:	y	**R-squared (uncentered):**	0.850
Model:	OLS	**Adj. R-squared (uncentered):**	0.843
Method:	Least Squares	**F-statistic:**	120.4
Date:	Sat, 19 Feb 2022	**Prob (F-statistic):**	4.58e-42
Time:	13:15:41	**Log-Likelihood:**	-749.56
No. Observations:	111	**AIC:**	1509.
Df Residuals:	106	**BIC:**	1523.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Length1	281.1885	88.270	3.186	0.002	106.186	456.191
Length2	-154.6661	86.743	-1.783	0.077	-326.643	17.311
Length3	-87.7487	34.855	-2.518	0.013	-156.853	-18.645
Height	58.7450	17.633	3.332	0.001	23.786	93.704
Width	-70.4534	41.573	-1.695	0.093	-152.877	11.970

Omnibus:	30.535	**Durbin-Watson:**	1.529
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	46.728
Skew:	1.303	**Prob(JB):**	7.13e-11
Kurtosis:	4.820	**Cond. No.**	316.

-Is there any dependence between the length and weight of the fish?

- Yes, there is dependence between the length and weight of fish.

2. Using the data source in Q1 fit the Ridge and Lasso Regression Models. (25 points)
   - Report the coefficients for both the models

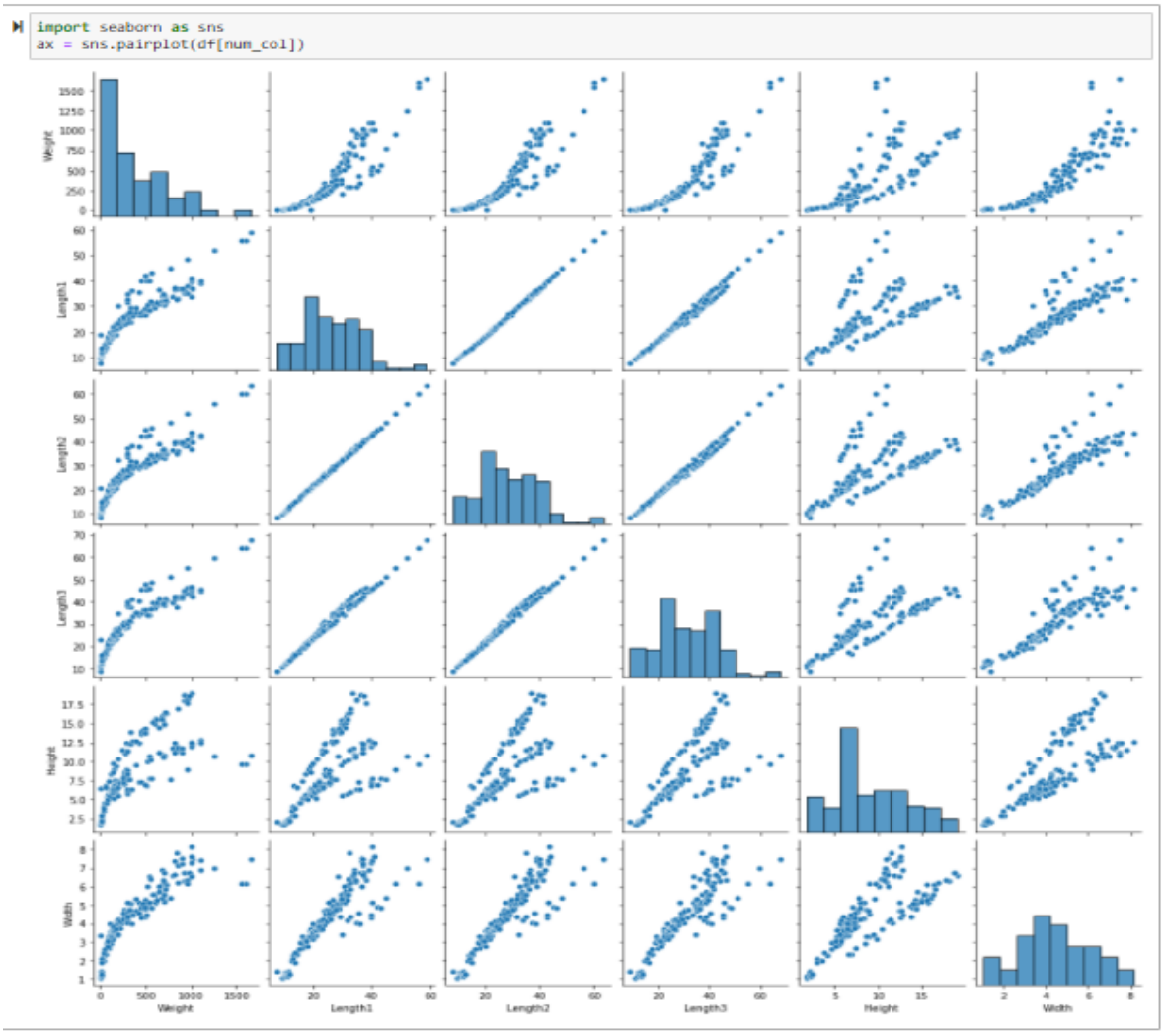
```
lasso = Lasso(alpha=0.2)

# Fit the regressor to the data
lasso.fit(X,y)

# Compute and print the coefficients
lasso_coef = lasso.coef_
print(lasso_coef)
```

```
[ 57.26889086  -4.57378381 -26.33466176  26.89081153  24.25900587]
```

```
ridge = Ridge(alpha=1,max_iter=40) #ridge model
ridge.fit(X_train,y_train)
ridge_coef = ridge.coef_
print(ridge_coef)
```

```
[ 83.4591279  -18.04075474 -32.09257368  26.77762824   0.33036577]
```

- Report the attribute(s) least impacting the weight of the fish.
    - The height of the fish is the least impacting attribute. We can see that the weight and height attributes are not closely scattered compared to others.



3. Modify the example code for Logistic Regression to include all the four attributes in iris dataset for two class and multi-class classification. Report any difference in the performance if noted.

   - Results in multi class classification is more accurate compared to two class classification. Because, we are using **multi_class="multinomial"** in multi class classification.

   - Two-Class Classification

```
In [30]: classifier.predict([[5.1,3.5,1.4,0.2]])
Out[30]: array(['Iris-versicolor'], dtype=object)

In [31]: classifier.predict_proba([[5.0,3.6,1.4,0.2]])
Out[31]: array([[7.31265952e-04, 7.93000502e-01, 2.06268232e-01]])

In [32]: classifier.score(X_train, y_train)
Out[32]: 0.9642857142857143

In [33]: expected = y_train
         predicted = classifier.predict(X_train)

In [34]: from sklearn import metrics
         print(metrics.classification_report(expected, predicted))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        35
Iris-versicolor       0.97      0.92      0.95        39
 Iris-virginica       0.93      0.97      0.95        38

       accuracy                           0.96       112
      macro avg       0.97      0.97      0.97       112
   weighted avg       0.97      0.96      0.96       112
```

- Multi-class classification

```
In [66]: classifier.predict([[5.1,3.5,1.4,0.2]])
Out[66]: array(['Iris-versicolor'], dtype=object)
```

```
In [76]: classifier.predict_proba([[5.0,3.6,1.4,0.2]])
Out[76]: array([[4.74403867e-04, 6.74769740e-01, 3.24755856e-01]])
```

```
In [68]: classifier.score(X_train, y_train)
Out[68]: 0.9732142857142857
```

```
In [73]: expected = y_train
         predicted = classifier.predict(X_train)
```

```
In [74]: from sklearn import metrics
         print(metrics.classification_report(expected, predicted))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        37
Iris-versicolor       0.97      0.94      0.96        34
 Iris-virginica       0.95      0.98      0.96        41

       accuracy                           0.97       112
      macro avg       0.97      0.97      0.97       112
   weighted avg       0.97      0.97      0.97       112
```