ML HW1

Name: Sai Mounica Chenuri Venkata V Lakshmi Phalguna

WSU ID:Z963A577

Question 1

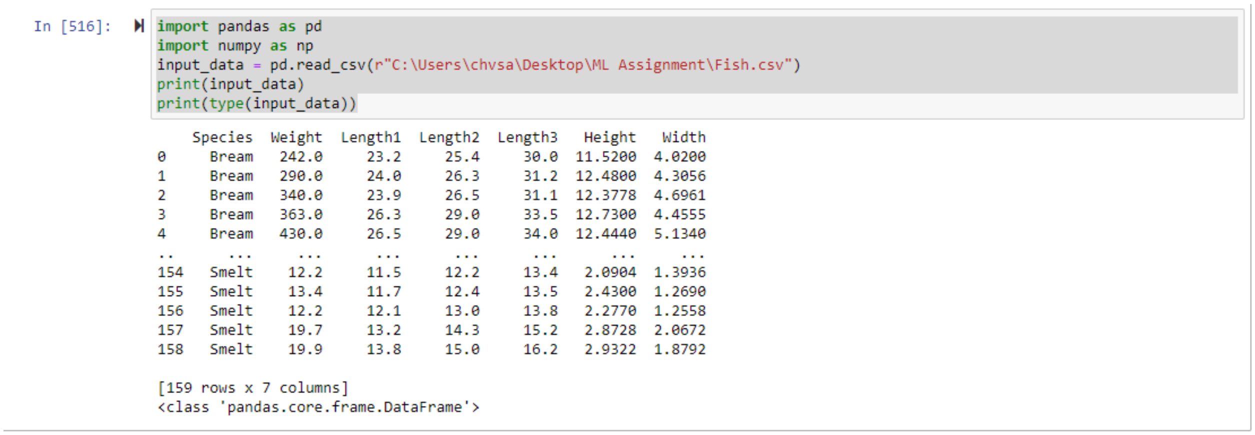
import pandas as pd

import numpy as np

input_data = pd.read_csv(r"C:\Users\chvsa\Desktop\ML Assignment\Fish.csv")

print(input_data)

print(type(input_data))



```
In [516]:  import pandas as pd
           import numpy as np
           input_data = pd.read_csv(r"C:\Users\chvsa\Desktop\ML Assignment\Fish.csv")
           print(input_data)
           print(type(input_data))

               Species  Weight  Length1  Length2  Length3   Height   Width
           0     Bream   242.0     23.2     25.4     30.0  11.5200  4.0200
           1     Bream   290.0     24.0     26.3     31.2  12.4800  4.3056
           2     Bream   340.0     23.9     26.5     31.1  12.3778  4.6961
           3     Bream   363.0     26.3     29.0     33.5  12.7300  4.4555
           4     Bream   430.0     26.5     29.0     34.0  12.4440  5.1340
           ..      ...     ...      ...      ...      ...      ...     ...
           154   Smelt    12.2     11.5     12.2     13.4   2.0904  1.3936
           155   Smelt    13.4     11.7     12.4     13.5   2.4300  1.2690
           156   Smelt    12.2     12.1     13.0     13.8   2.2770  1.2558
           157   Smelt    19.7     13.2     14.3     15.2   2.8728  2.0672
           158   Smelt    19.9     13.8     15.0     16.2   2.9322  1.8792

           [159 rows x 7 columns]
           <class 'pandas.core.frame.DataFrame'>
```

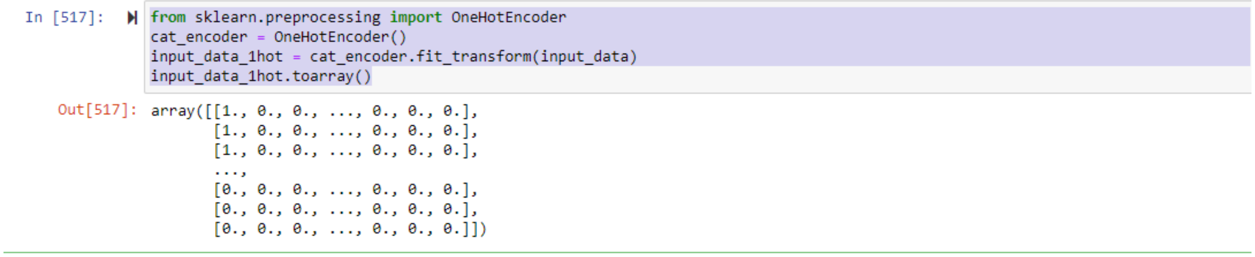
One-hot encoding

from sklearn.preprocessing import OneHotEncoder

cat_encoder = OneHotEncoder()

input_data_1hot = cat_encoder.fit_transform(input_data)

input_data_1hot.toarray()



```
In [517]:  from sklearn.preprocessing import OneHotEncoder
           cat_encoder = OneHotEncoder()
           input_data_1hot = cat_encoder.fit_transform(input_data)
           input_data_1hot.toarray()

Out[517]:  array([[1., 0., 0., ..., 0., 0., 0.],
                  [1., 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., ..., 0., 0., 0.]])
```

Reference from “Machine Learning Text book-One Hot Encoding”

NORMALIZATION

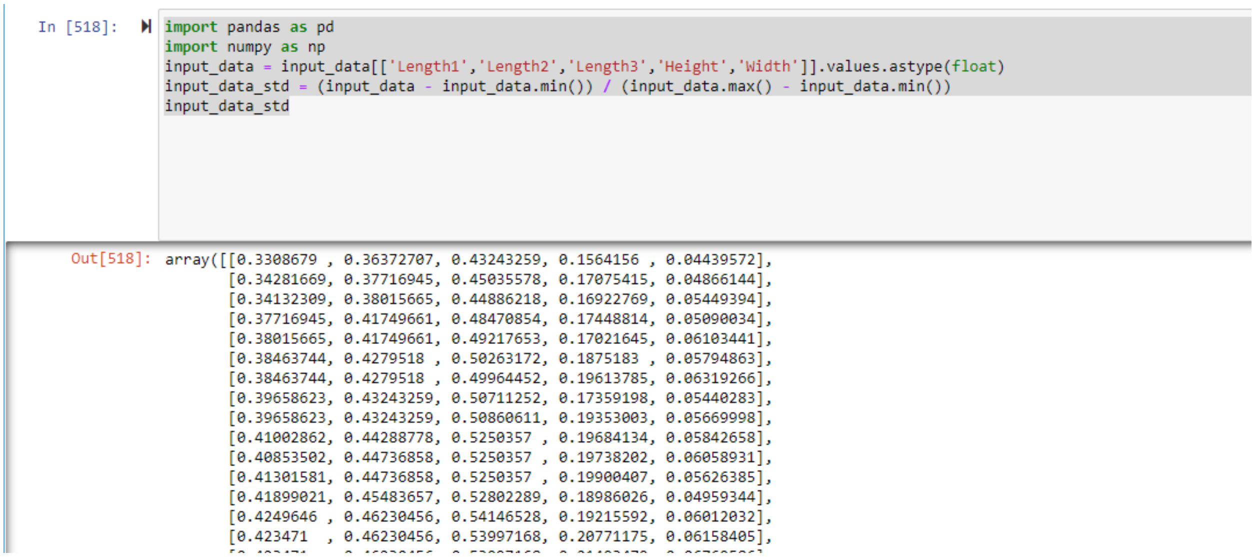
import pandas as pd

import numpy as np

input_data = input_data[['Length1','Length2','Length3','Height','Width']].values.astype(float)

input_data_std = (input_data - input_data.min()) / (input_data.max() - input_data.min())

input_data_std



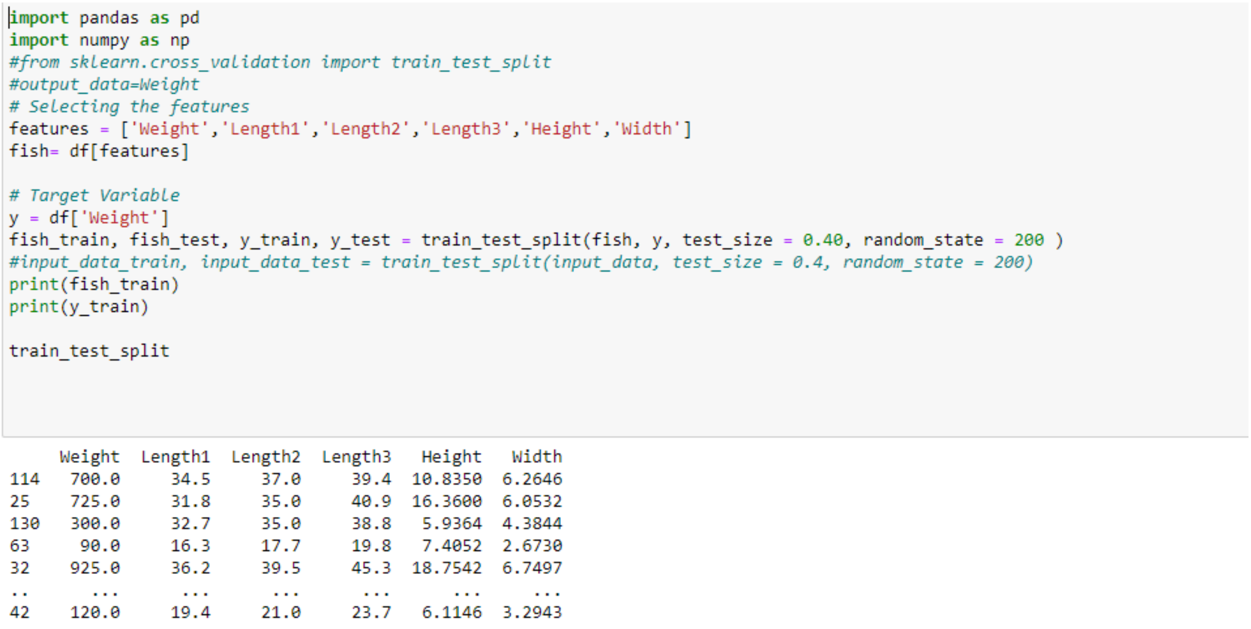
```
In [518]:  import pandas as pd
            import numpy as np
            input_data = input_data[['Length1','Length2','Length3','Height','Width']].values.astype(float)
            input_data_std = (input_data - input_data.min()) / (input_data.max() - input_data.min())
            input_data_std
```

```
Out[518]: array([[0.3308679 , 0.36372707, 0.43243259, 0.1564156 , 0.04439572],
                 [0.34281669, 0.37716945, 0.45035578, 0.17075415, 0.04866144],
                 [0.34132309, 0.38015665, 0.44886218, 0.16922769, 0.05449394],
                 [0.37716945, 0.41749661, 0.48470854, 0.17448814, 0.05090034],
                 [0.38015665, 0.41749661, 0.49217653, 0.17021645, 0.06103441],
                 [0.38463744, 0.4279518 , 0.50263172, 0.1875183 , 0.05794863],
                 [0.38463744, 0.4279518 , 0.49964452, 0.19613785, 0.06319266],
                 [0.39658623, 0.43243259, 0.50711252, 0.17359198, 0.05440283],
                 [0.39658623, 0.43243259, 0.50860611, 0.19353003, 0.05669998],
                 [0.41002862, 0.44288778, 0.5250357 , 0.19684134, 0.05842658],
                 [0.40853502, 0.44736858, 0.5250357 , 0.19738202, 0.06058931],
                 [0.41301581, 0.44736858, 0.5250357 , 0.19900407, 0.05626385],
                 [0.41899021, 0.45483657, 0.52802289, 0.18986026, 0.04959344],
                 [0.4249646 , 0.46230456, 0.54146528, 0.19215592, 0.06012032],
                 [0.423471  , 0.46230456, 0.53997168, 0.20771175, 0.06158405],
```

Reference from “the formula given in HW”

Question 2

```
import pandas as pd

import numpy as np

#from sklearn.cross_validation import train_test_split

#output_data=Weight

# Selecting the features

features = ['Weight','Length1','Length2','Length3','Height','Width']

fish= df[features]

# Target Variable

y = df['Weight']

fish_train, fish_test, y_train, y_test = train_test_split(fish, y, test_size = 0.40, random_state = 200 )

#input_data_train, input_data_test = train_test_split(input_data, test_size = 0.4, random_state = 200)

print(fish_train)

print(y_train)

train_test_split
```



```
import pandas as pd
import numpy as np
#from sklearn.cross_validation import train_test_split
#output_data=Weight
# Selecting the features
features = ['Weight','Length1','Length2','Length3','Height','Width']
fish= df[features]

# Target Variable
y = df['Weight']
fish_train, fish_test, y_train, y_test = train_test_split(fish, y, test_size = 0.40, random_state = 200 )
#input_data_train, input_data_test = train_test_split(input_data, test_size = 0.4, random_state = 200)
print(fish_train)
print(y_train)

train_test_split
```

```
     Weight  Length1  Length2  Length3   Height   Width
114   700.0     34.5     37.0     39.4  10.8350  6.2646
25    725.0     31.8     35.0     40.9  16.3600  6.0532
130   300.0     32.7     35.0     38.8   5.9364  4.3844
63     90.0     16.3     17.7     19.8   7.4052  2.6730
32    925.0     36.2     39.5     45.3  18.7542  6.7497
..      ...      ...      ...      ...      ...     ...
42    120.0     19.4     21.0     23.7   6.1146  3.2943
```

Linear Regression:

import pandas as pd

import numpy as np

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

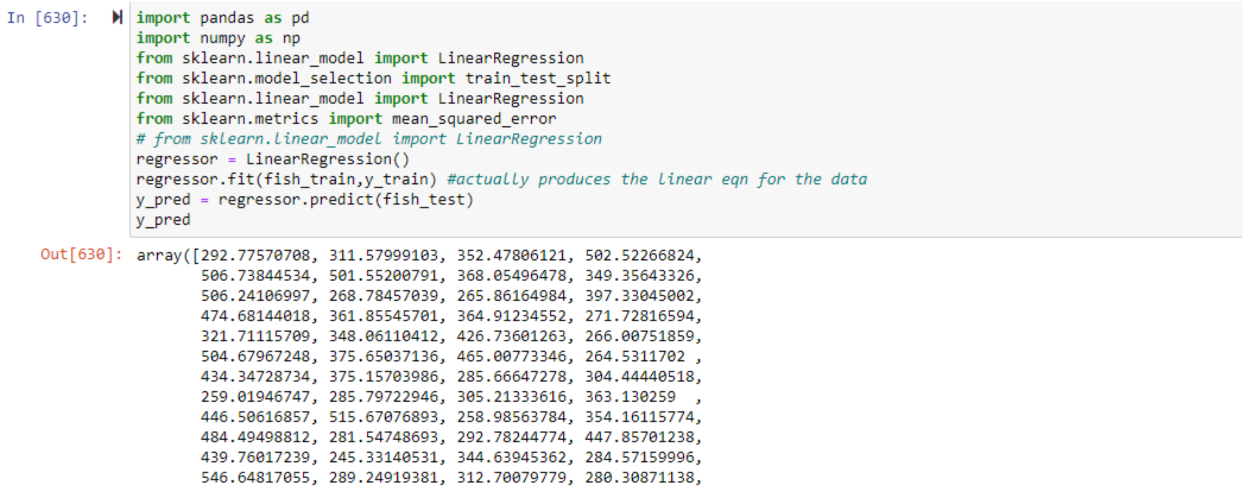
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

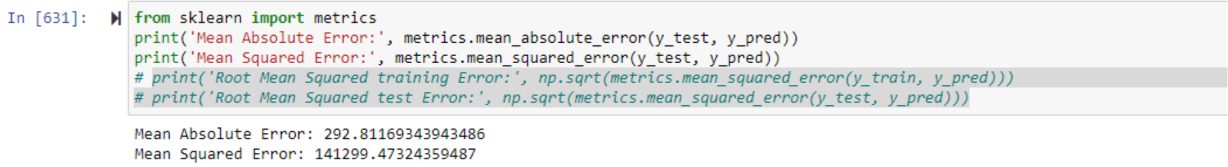
regressor.fit(x_train,y_train) #actually produces the linear eqn for the data

y_pred = regressor.predict(x_test)

y_pred



```
In [630]:  import pandas as pd
            import numpy as np
            from sklearn.linear_model import LinearRegression
            from sklearn.model_selection import train_test_split
            from sklearn.linear_model import LinearRegression
            from sklearn.metrics import mean_squared_error
            # from sklearn.linear_model import LinearRegression
            regressor = LinearRegression()
            regressor.fit(fish_train,y_train) #actually produces the linear eqn for the data
            y_pred = regressor.predict(fish_test)
            y_pred

Out[630]:  array([292.77570708, 311.57999103, 352.47806121, 502.52266824,
                  506.73844534, 501.55200791, 368.05496478, 349.35643326,
                  506.24106997, 268.78457039, 265.86164984, 397.33045002,
                  474.68144018, 361.85545701, 364.91234552, 271.72816594,
                  321.71115709, 348.06110412, 426.73601263, 266.00751859,
                  504.67967248, 375.65037136, 465.00773346, 264.5311702 ,
                  434.34728734, 375.15703986, 285.66647278, 304.44440518,
                  259.01946747, 285.79722946, 305.21333616, 363.130259  ,
                  446.50616857, 515.67076893, 258.98563784, 354.16115774,
                  484.49498812, 281.54748693, 292.78244774, 447.85701238,
                  439.76017239, 245.33140531, 344.63945362, 284.57159996,
                  546.64817055, 289.24919381, 312.70079779, 280.30871138,
```



```
In [631]:  from sklearn import metrics
           print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
           print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
           # print('Root Mean Squared training Error:', np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
           # print('Root Mean Squared test Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

           Mean Absolute Error: 292.81169343943486
           Mean Squared Error: 141299.47324359487
```

Reference: “text book diving the given data into test set n training set and also Linear Regression in Python – Real Python ”

POLYNIMIAL REGRESSION:

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import LinearRegression

poly=PolynomialFeatures(degree=30)

fish_train_poly=poly.fit_transform(fish_train)

fish_test_poly = poly.fit_transform(fish_test)

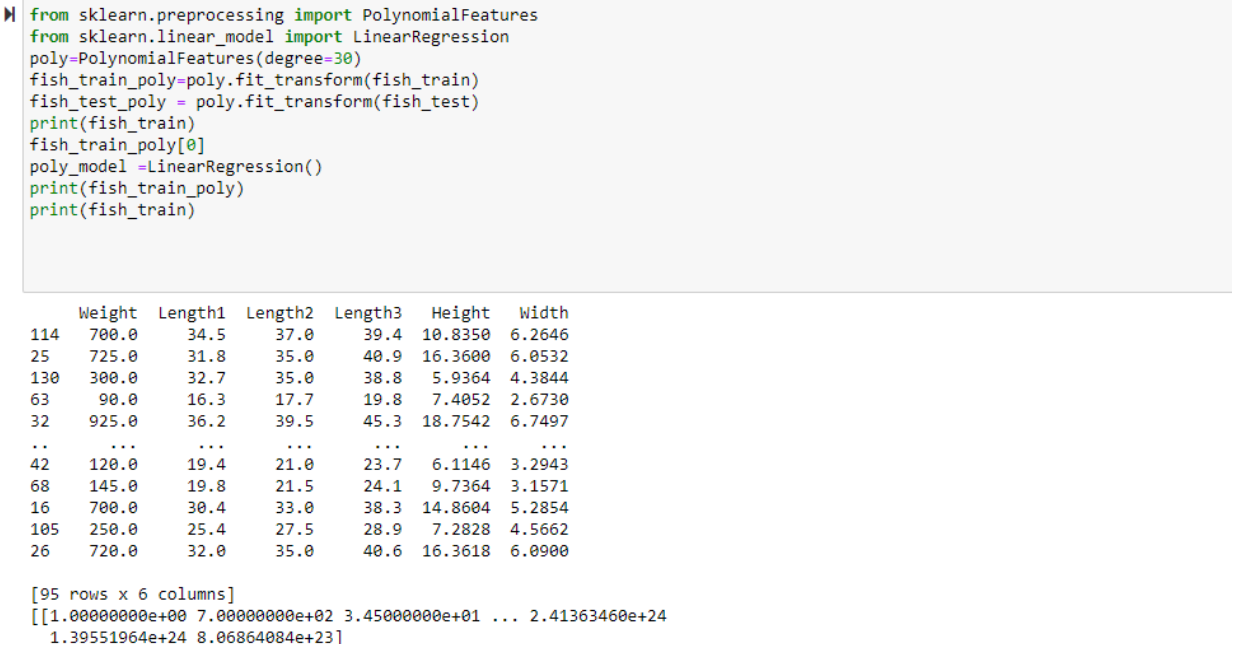
print(fish_train)

fish_train_poly[0]

poly_model =LinearRegression()

print(fish_train_poly)

print(fish_train)



```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly=PolynomialFeatures(degree=30)
fish_train_poly=poly.fit_transform(fish_train)
fish_test_poly = poly.fit_transform(fish_test)
print(fish_train)
fish_train_poly[0]
poly_model =LinearRegression()
print(fish_train_poly)
print(fish_train)
```

```
     Weight  Length1  Length2  Length3   Height   Width
114   700.0     34.5     37.0     39.4  10.8350  6.2646
25    725.0     31.8     35.0     40.9  16.3600  6.0532
130   300.0     32.7     35.0     38.8   5.9364  4.3844
63     90.0     16.3     17.7     19.8   7.4052  2.6730
32    925.0     36.2     39.5     45.3  18.7542  6.7497
..      ...      ...      ...      ...      ...     ...
42    120.0     19.4     21.0     23.7   6.1146  3.2943
68    145.0     19.8     21.5     24.1   9.7364  3.1571
16    700.0     30.4     33.0     38.3  14.8604  5.2854
105   250.0     25.4     27.5     28.9   7.2828  4.5662
26    720.0     32.0     35.0     40.6  16.3618  6.0900

[95 rows x 6 columns]
[[1.00000000e+00 7.00000000e+02 3.45000000e+01 ... 2.41363460e+24
  1.39551964e+24 8.06864084e+23]
```

poly_model.fit(fish_train_poly, y_train)

poly_model.coef_

poly_model.intercept_

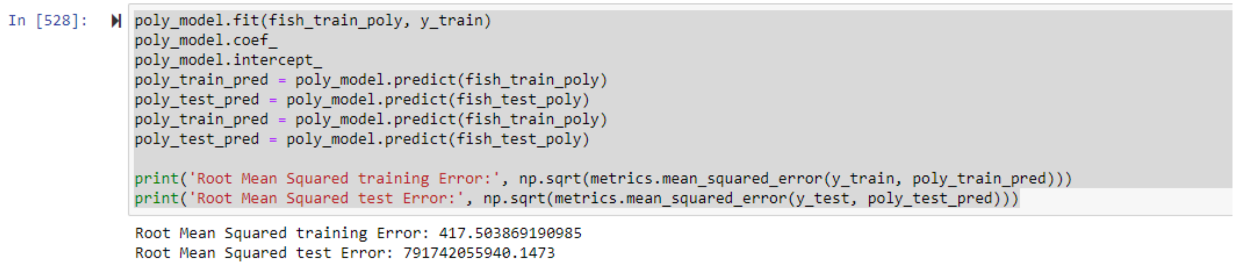
poly_train_pred = poly_model.predict(fish_train_poly)

poly_test_pred = poly_model.predict(fish_test_poly)

poly_train_pred = poly_model.predict(fish_train_poly)

poly_test_pred = poly_model.predict(fish_test_poly)

```
print('Root Mean Squared training Error:', np.sqrt(metrics.mean_squared_error(y_train, poly_train_pred)))

print('Root Mean Squared test Error:', np.sqrt(metrics.mean_squared_error(y_test, poly_test_pred)))
```

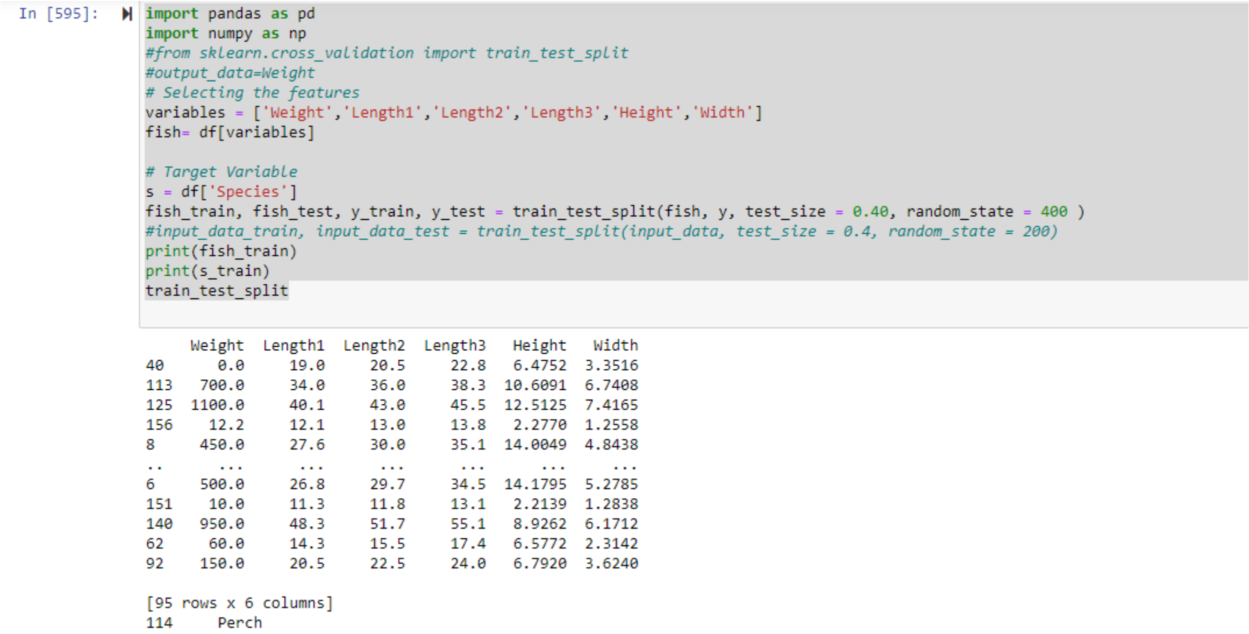


```
In [528]:  poly_model.fit(fish_train_poly, y_train)
           poly_model.coef_
           poly_model.intercept_
           poly_train_pred = poly_model.predict(fish_train_poly)
           poly_test_pred = poly_model.predict(fish_test_poly)
           poly_train_pred = poly_model.predict(fish_train_poly)
           poly_test_pred = poly_model.predict(fish_test_poly)

           print('Root Mean Squared training Error:', np.sqrt(metrics.mean_squared_error(y_train, poly_train_pred)))
           print('Root Mean Squared test Error:', np.sqrt(metrics.mean_squared_error(y_test, poly_test_pred)))

           Root Mean Squared training Error: 417.503869190985
           Root Mean Squared test Error: 791742055940.1473
```

Reference: “Text book Polynomial regression and also Python | Implementation of Polynomial Regression - GeeksforGeeks”

Question 3:

```
import pandas as pd

import numpy as np

#from sklearn.cross_validation import train_test_split

#output_data=Weight

# Selecting the features

variables = ['Weight','Length1','Length2','Length3','Height','Width']

fish= df[variables]

# Target Variable

s = df['Species']

fish_train, fish_test, y_train, y_test = train_test_split(fish, y, test_size = 0.40, random_state = 400 )

#input_data_train, input_data_test = train_test_split(input_data, test_size = 0.4, random_state = 200)

print(fish_train)

print(s_train)

train_test_split
```

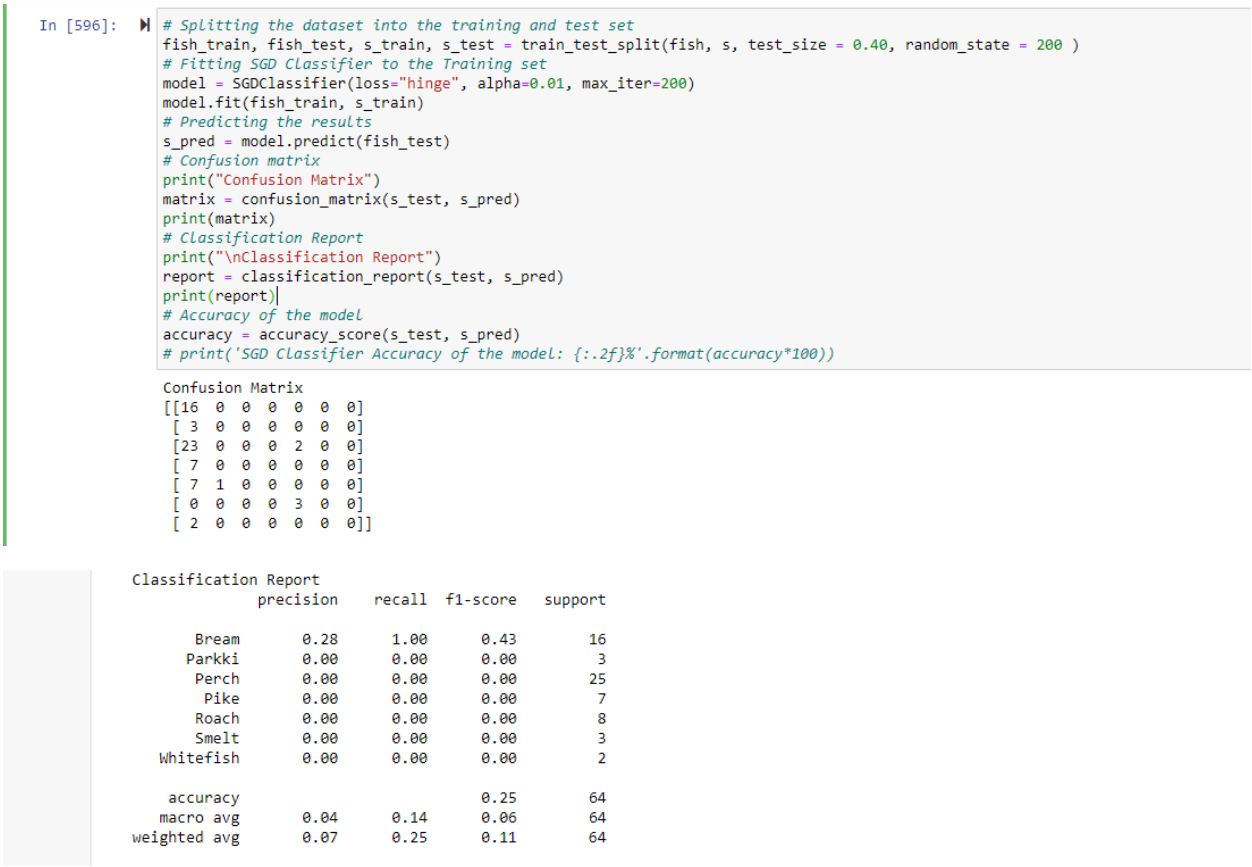


```
import pandas as pd
import numpy as np
#from sklearn.cross_validation import train_test_split
#output_data=Weight
# Selecting the features
variables = ['Weight','Length1','Length2','Length3','Height','Width']
fish= df[variables]

# Target Variable
s = df['Species']
fish_train, fish_test, y_train, y_test = train_test_split(fish, y, test_size = 0.40, random_state = 400 )
#input_data_train, input_data_test = train_test_split(input_data, test_size = 0.4, random_state = 200)
print(fish_train)
print(s_train)
train_test_split
```

```
     Weight  Length1  Length2  Length3   Height   Width
40      0.0     19.0     20.5     22.8   6.4752  3.3516
113   700.0     34.0     36.0     38.3  10.6091  6.7408
125  1100.0     40.1     43.0     45.5  12.5125  7.4165
156    12.2     12.1     13.0     13.8   2.2770  1.2558
8     450.0     27.6     30.0     35.1  14.0049  4.8438
..      ...      ...      ...      ...      ...     ...
6     500.0     26.8     29.7     34.5  14.1795  5.2785
151    10.0     11.3     11.8     13.1   2.2139  1.2838
140   950.0     48.3     51.7     55.1   8.9262  6.1712
62     60.0     14.3     15.5     17.4   6.5772  2.3142
92    150.0     20.5     22.5     24.0   6.7920  3.6240

[95 rows x 6 columns]
114      Perch
```

SGD Classifier:

```
# Splitting the dataset into the training and test set

fish_train, fish_test, s_train, s_test = train_test_split(fish, s, test_size = 0.40, random_state = 200 )

# Fitting SGD Classifier to the Training set

model = SGDClassifier(loss="hinge", alpha=0.01, max_iter=200)

model.fit(fish_train, s_train)

# Predicting the results

s_pred = model.predict(fish_test)

# Confusion matrix

print("Confusion Matrix")

matrix = confusion_matrix(s_test, s_pred)

print(matrix)

# Classification Report

print("\nClassification Report")

report = classification_report(s_test, s_pred)

print(report)

# Accuracy of the model

accuracy = accuracy_score(s_test, s_pred)
```



```
In [596]:  # Splitting the dataset into the training and test set
            fish_train, fish_test, s_train, s_test = train_test_split(fish, s, test_size = 0.40, random_state = 200 )
            # Fitting SGD Classifier to the Training set
            model = SGDClassifier(loss="hinge", alpha=0.01, max_iter=200)
            model.fit(fish_train, s_train)
            # Predicting the results
            s_pred = model.predict(fish_test)
            # Confusion matrix
            print("Confusion Matrix")
            matrix = confusion_matrix(s_test, s_pred)
            print(matrix)
            # Classification Report
            print("\nClassification Report")
            report = classification_report(s_test, s_pred)
            print(report)
            # Accuracy of the model
            accuracy = accuracy_score(s_test, s_pred)
            # print('SGD Classifier Accuracy of the model: {:.2f}%'.format(accuracy*100))
```

```
Confusion Matrix
[[16  0  0  0  0  0  0]
 [ 3  0  0  0  0  0  0]
 [23  0  0  0  2  0  0]
 [ 7  0  0  0  0  0  0]
 [ 7  1  0  0  0  0  0]
 [ 0  0  0  0  3  0  0]
 [ 2  0  0  0  0  0  0]]

Classification Report
              precision    recall  f1-score   support

       Bream       0.28      1.00      0.43        16
      Parkki       0.00      0.00      0.00         3
       Perch       0.00      0.00      0.00        25
        Pike       0.00      0.00      0.00         7
       Roach       0.00      0.00      0.00         8
       Smelt       0.00      0.00      0.00         3
   Whitefish       0.00      0.00      0.00         2

    accuracy                           0.25        64
   macro avg       0.04      0.14      0.06        64
weighted avg       0.07      0.25      0.11        64
```

Reference from: "text book SGD Clarifier and Stochastic Gradient Descent (SGD) Classifier - The Click Reader"

KNeighbour Classifier:

from sklearn.neighbors import KNeighborsClassifier

from sklearn.datasets import load_iris

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

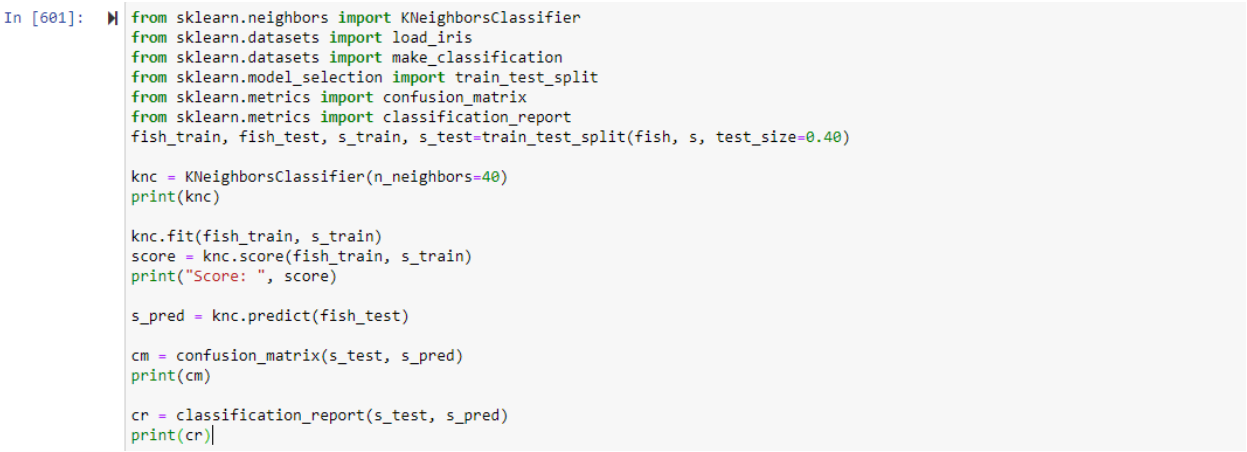
from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

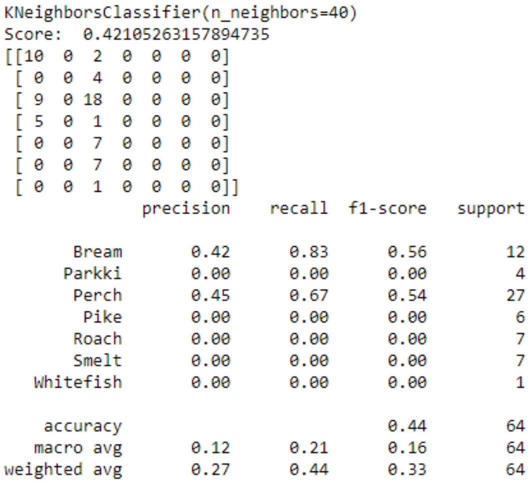
fish_train, fish_test, s_train, s_test=train_test_split(fish, s, test_size=0.40)

knc = KNeighborsClassifier(n_neighbors=40)

```
print(knc)

knc.fit(fish_train, s_train)

score = knc.score(fish_train, s_train)

print("Score: ", score)

s_pred = knc.predict(fish_test)

cm = confusion_matrix(s_test, s_pred)

print(cm)

cr = classification_report(s_test, s_pred)

print(cr)
```



```
In [601]:  from sklearn.neighbors import KNeighborsClassifier
           from sklearn.datasets import load_iris
           from sklearn.datasets import make_classification
           from sklearn.model_selection import train_test_split
           from sklearn.metrics import confusion_matrix
           from sklearn.metrics import classification_report
           fish_train, fish_test, s_train, s_test=train_test_split(fish, s, test_size=0.40)

           knc = KNeighborsClassifier(n_neighbors=40)
           print(knc)

           knc.fit(fish_train, s_train)
           score = knc.score(fish_train, s_train)
           print("Score: ", score)

           s_pred = knc.predict(fish_test)

           cm = confusion_matrix(s_test, s_pred)
           print(cm)

           cr = classification_report(s_test, s_pred)
           print(cr)
```



```
KNeighborsClassifier(n_neighbors=40)
Score:  0.42105263157894735
[[10  0  2  0  0  0  0]
 [ 0  0  4  0  0  0  0]
 [ 9  0 18  0  0  0  0]
 [ 5  0  1  0  0  0  0]
 [ 0  0  7  0  0  0  0]
 [ 0  0  7  0  0  0  0]
 [ 0  0  1  0  0  0  0]]
              precision    recall  f1-score   support

       Bream       0.42      0.83      0.56        12
      Parkki       0.00      0.00      0.00         4
       Perch       0.45      0.67      0.54        27
        Pike       0.00      0.00      0.00         6
       Roach       0.00      0.00      0.00         7
       Smelt       0.00      0.00      0.00         7
   Whitefish       0.00      0.00      0.00         1

    accuracy                           0.44        64
   macro avg       0.12      0.21      0.16        64
weighted avg       0.27      0.44      0.33        64
```

Reference: “Text book KNeighbour Classifier and DataTechNotes: Classification Example with KNeighborsClassifier in Python”

4th Question:

a)

$\boldsymbol{FshWti}$=($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$

$E= (t-y)^2$

dE/dw_0= 2*(t-($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$)*Lenght1

dE/dw_1=2*(t-($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$)*Length2

dE/dw_2=2*(t- ($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$)*Length3

dE/dw_3= 2*(t-($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$)*Height

dE/dw_4= 2*(t-($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$)***Width**

dE/dw_b= 2*(t-($\boldsymbol{w0*Lengthi1}$)+ ($\boldsymbol{w1*Lengthi2}$)+ ($\boldsymbol{w2*Lengthi3}$)+ ($\boldsymbol{w3*Height}$)+ ($\boldsymbol{w4*Width}$)+ $\boldsymbol{wb}$)

Updated $\mathbf{w_i=w_i-\alpha dw_i}$