Final PPQ1–Regression

December 14, 2023

1 Problem 1 of 3 Final Programming Part

1.1 Regression of Possom head length based on total length

In this problem you will perform a regression of Possom head length based on total length. Will need to load the data, inpect it, select the appropriate columns. You will need to make figures, perform the regression and estimate the error.

1.1.1 Part 0: Data Imports

[ ]:
```
import pandas as pd
import numpy as np
from numpy import linspace
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse

score = {0:0}
```

1.1.2 Part 1: Load and Inspect the Data

Load the data from the URL https://bit.ly/possumcsv and perform inspection tasks.

1.1 Load csv file from the URL https://bit.ly/possumcsv Load a csv from the URL https://bit.ly/possumcsv and store it as a pandas DataFrame in the variable `possum_df`.

[ ]:
```
# 1.1 Load data code here


# Leave the code in this cell below unchanged. Use print if output is desired
score['1.1'] = 1
```

1.2 Inspect the data

1. Print out the first 5 rows of the data
2. Print the columns of the data, and their data types, and the non-null counts

3. Print the summary statistics of the data

[ ]:
```
#### 1.2.1


# Leave the code in this cell below unchanged. Use print if output is desired
score['1.2.1'] = 1
```

[ ]:
```
#### 1.2.2


# Leave the code in this cell below unchanged. Use print if output is desired
score['1.2.2'] = 1
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   case      104 non-null    int64
 1   site      104 non-null    int64
 2   Pop       104 non-null    object
 3   sex       104 non-null    object
 4   age       102 non-null    float64
 5   hdlngth   104 non-null    float64
 6   skullw    104 non-null    float64
 7   totlngth  104 non-null    float64
 8   taill     104 non-null    float64
 9   footlgth  103 non-null    float64
 10  earconch  104 non-null    float64
 11  eye       104 non-null    float64
 12  chest     104 non-null    float64
 13  belly     104 non-null    float64
dtypes: float64(10), int64(2), object(2)
memory usage: 11.5+ KB
```

[ ]:
```
#### 1.2.3


# Leave the code in this cell below unchanged. Use print if output is desired
score['1.2.3'] = 1
```

```
             case        site         age     hdlngth      skullw    totlngth  \
count  104.000000  104.000000  102.000000  104.000000  104.000000  104.000000
mean    52.500000    3.625000    3.833333   92.602885   56.883654   87.088462
std     30.166206    2.349086    1.909244    3.573349    3.113426    4.310549
min      1.000000    1.000000    1.000000   82.500000   50.000000   75.000000
25%     26.750000    1.000000    2.250000   90.675000   54.975000   84.000000
```

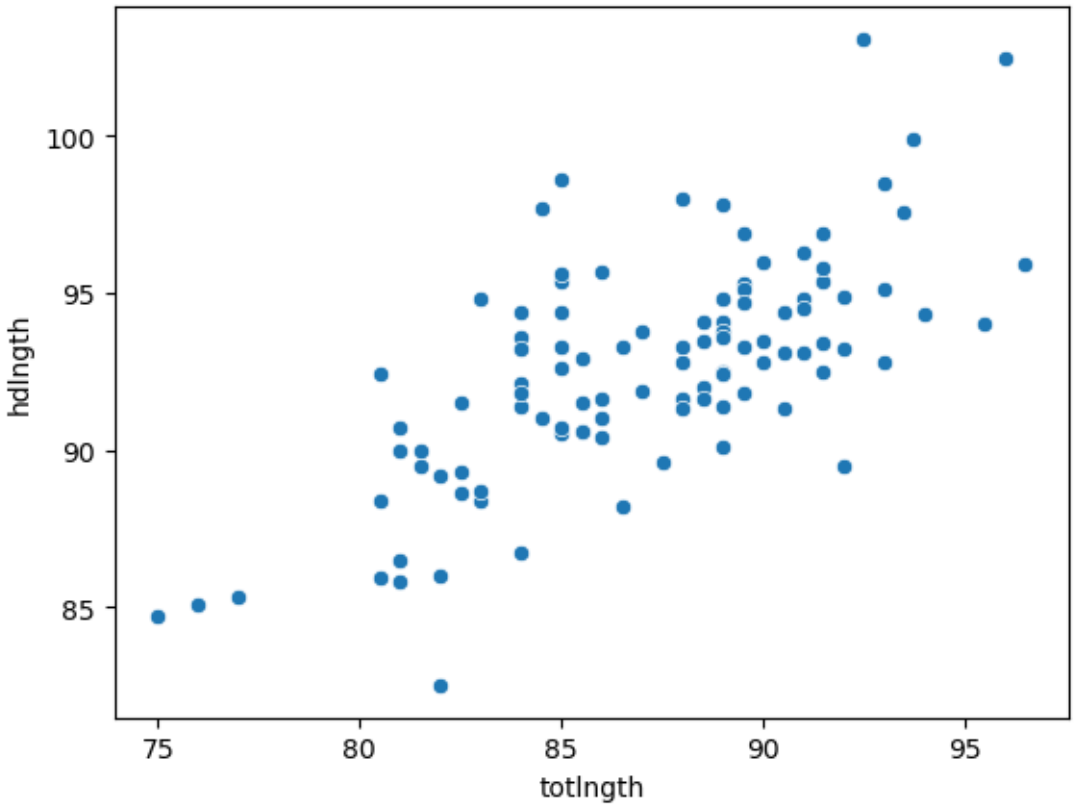
```
50%     52.500000    3.000000    3.000000   92.800000   56.350000   88.000000
75%     78.250000    6.000000    5.000000   94.725000   58.100000   90.000000
max    104.000000    7.000000    9.000000  103.100000   68.600000   96.500000

            taill    footlgth    earconch         eye       chest       belly
count  104.000000  103.000000  104.000000  104.000000  104.000000  104.000000
mean    37.009615   68.459223   48.130769   15.046154   27.000000   32.586538
std      1.959518    4.395306    4.109380    1.050374    2.045597    2.761949
min     32.000000   60.300000   40.300000   12.800000   22.000000   25.000000
25%     35.875000   64.600000   44.800000   14.400000   25.500000   31.000000
50%     37.000000   68.000000   46.800000   14.900000   27.000000   32.500000
75%     38.000000   72.500000   52.000000   15.725000   28.000000   34.125000
max     43.000000   77.900000   56.200000   17.800000   32.000000   40.000000
```

1.1.3 1.3 Set up the data for regression

1. Select the columns `totlnght` (input) and `hdlnght` (output) and store them variables `x` and `y` respectively. Make sure that x and y are numpy arrays so you may need to do a bit more to extract the values from the DataFrame.
2. Make a scatter plot of `x` vs `y` with appropriate labels and title using seaborn
3. Comment in the markdown file on the relationship between `X` and `y`. Does do they look unrelated? Is there a linear relationship? Is there a non-linear relationship?
4. Turn `x` into a column vector `X` using `X=x.reshape(-1,1)`. This is necessary for the regression to work.
5. Split the data into training and testing sets using `train_test_split` with a test size of 0.2 and a random state of 42. Store the results in variables `X_train`, `X_test`, `y_train`, `y_test`.

[ ]:
```
#### 1.3.1 x and y


# Leave the code in this cell below unchanged. Use print if output is desired
score['1.3.1'] = 1
```

[ ]:
```
#### 1.3.2 graph



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.3.2'] = 2
```



1.3.3 Your comments on the relationship between x and y here in this cell There definitely seems to be a relationship betwen totlnght and hdlnght. The relationship seems to be linear.

[ ]:
```
#### 1.3.3 score from markdown above
# Leave the code in this cell below unchanged. Use print if output is desired
score['1.3.3'] = 2
```

[ ]:
```
#### 1.3.4 reshape x


# Leave the code in this cell below unchanged. Use print if output is desired
score['1.3.4'] = 1
```

[ ]:
```
#### 1.3.5 split data



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.3.5'] = 2
```

1.2 1.4 Create, fit linear regression model and make predictions

1. Create a linear regression model using `LinearRegression`
2. Fit that model on the training data.
3. Predict using training data and store the results in `y_pred_train` and for testing data and store the results in `y_pred_test`.

[ ]:
```
#### 1.4.1 create model



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.4.1'] = 1
```

[ ]:
```
#### 1.4.2 fit model



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.4.2'] = 2
```

[ ]:
```
#### 1.4.3 predict



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.4.3'] = 2
```

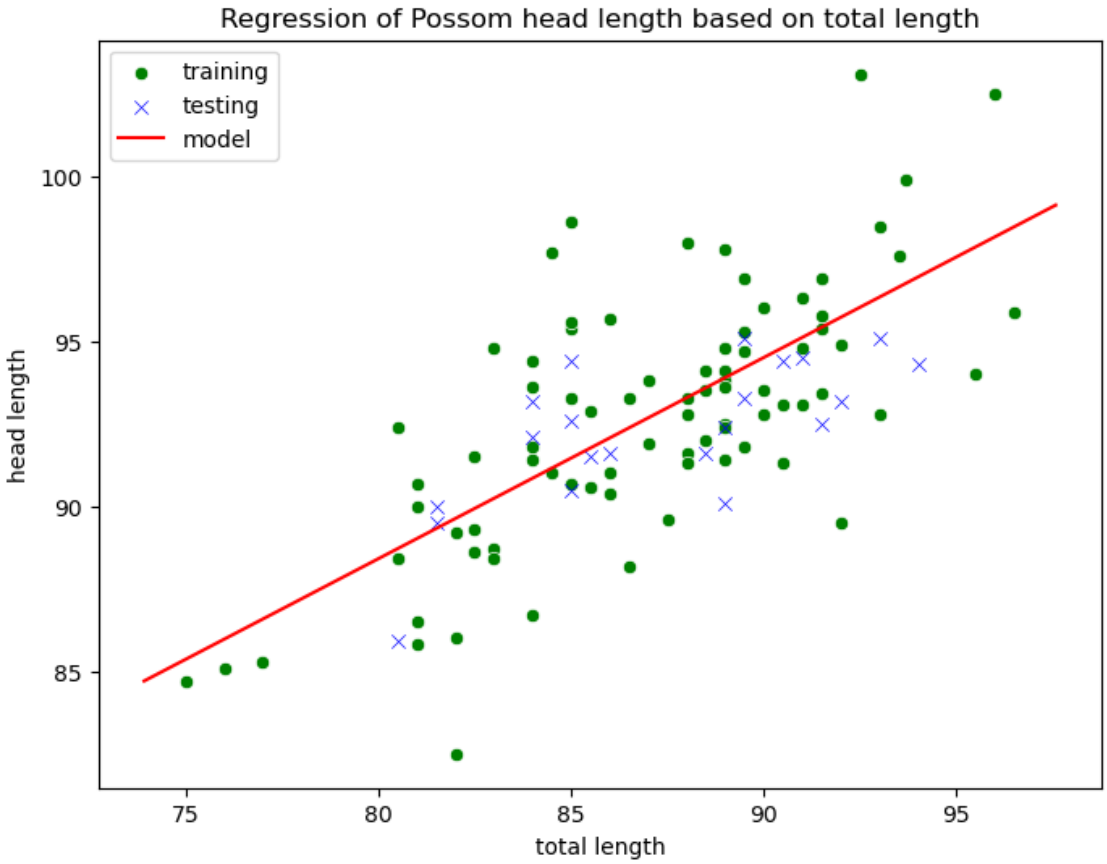
1.3 1.5 Evaluate the model (error)

We need to evaluate the model to see how well it performs. We will use the bias (mean), the mean square error. We will plot the fit and a histogram of the residuals.

1. Compute the min and max for all the data x and store them `x_min` and `x_max` respectively. Use `np.min` and `np.max`. Use `linspace` to create a range of values from 'x_min' to 'x_max' and store them in `x_samps`. Use `num=100` to get 100 samples. Use `reshape(-1,1)` to turn `x_samps` into a column vector and then use the model to predict the values and store them in `y_samps`.
2. Plot the fit of the model. Use seaborn to scatter plot the 'X_train' vs 'y_train' with green dots, 'X_test' vs 'y_test' with blue 'x's, and 'x_samps' vs 'y_samps' with a red line using seaborn's `lineplot`. Make sure to label the axes and add a legend.
3. Calculate the training and testing residuals and store them in `train_residuals` and `test_residuals` respectively.
4. Compute the bias (mean) of the training and testing residuals and store them in `train_bias` and `test_bias` respectively and print them.
5. Compute the mean squared error of the training and testing residuals and store them in `train_mse` and `test_mse` respectively and print them.
6. Plot the histogram of the training and testing residuals using seaborn's `distplot` with 20 bins. Make sure to label the axes and add a legend.

[ ]:
```
#### 1.5.1 predict



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.5.1'] = 2
```

[ ]:
```
#### 1.5.2 plot



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.5.2'] = 3
```



[ ]:
```
#### 1.5.3 compute the residuals



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.5.3'] = 2
```

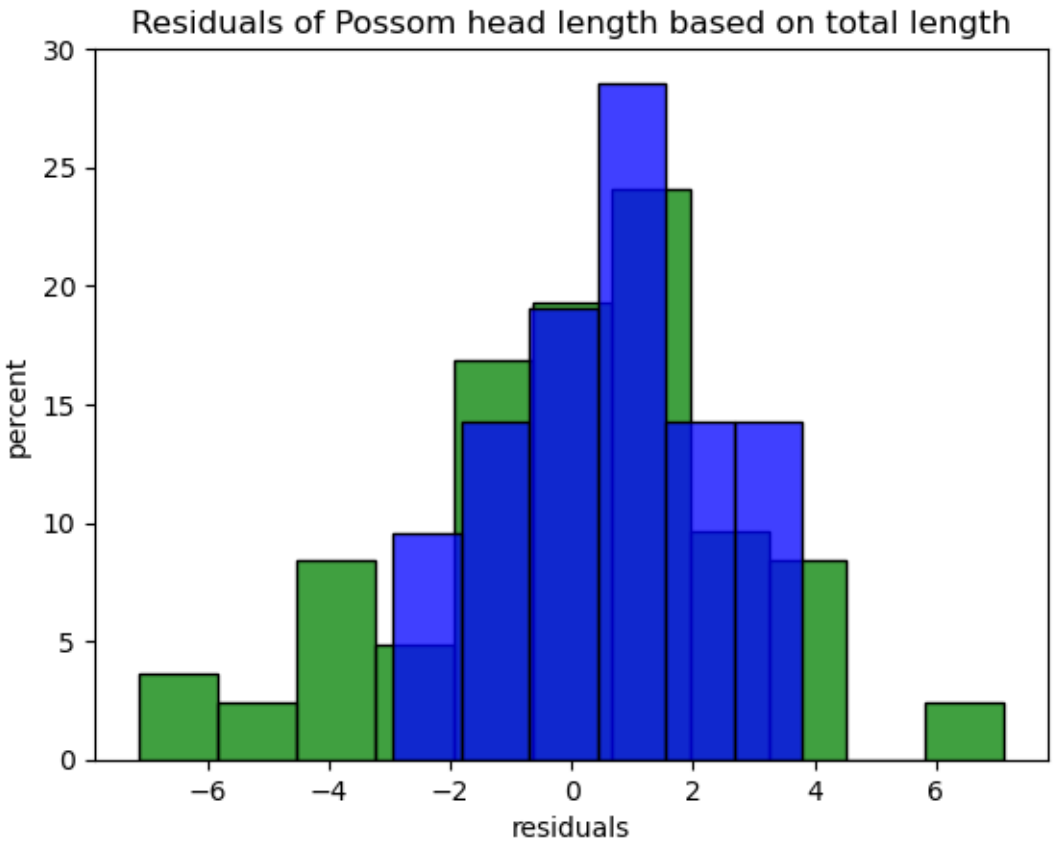
[ ]:
```
#### 1.5.4 compute the bias



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.5.4'] = 1
```

```
train_bias: -9.245616320733834e-15, test_bias_test: 0.6510364061655965
```

[ ]:
```
#### 1.5.5 compute the mean square error




# Leave the code in this cell below unchanged. Use print if output is desired
score['1.5.5'] = 2
```

```
train_mse: 7.451169493458306, test_mse: 3.4761078498395754
```

[ ]:
```
#### 1.5.6 histogram of the residuals



# Leave the code in this cell below unchanged. Use print if output is desired
score['1.5.6'] = 3
```



[ ]:
```
sum(score.values())
```

[ ]: 30

[ ]: