Tutorial- 4
Regression

Part A -

You are given a data file *abalone.csv*. Abalones are marine snails. The dataset has been prepared with the aim of making age predictions easier. Customarily, the age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. But it is a tedious and time-consuming task. Therefore, other measurements, which are easier to obtain, are used to predict age.

CSV Information -
Name / Data Type / Measurement Unit / Description

1. Length / continuous / mm / Longest shell measurement
2. Diameter / continuous / mm / Diameter of the shell calculated as perpendicular to length
3. Height / continuous / mm / Height of the shell with meat in shell
4. Whole weight / continuous / grams / Weight of whole abalone
5. Shucked weight / continuous / grams / Weight of meat
6. Viscera weight / continuous / grams / Gut-weight (after bleeding)
7. Shell weight / continuous / grams / Weight of the shell after being dried
8. Rings / integer / -- / Number of rings in a shell. (Adding 1.5 to the number of rings gives the age of abalone in years)

Tasks -

- Write a python program to split the data from abalone.csv into train data and test data. Train data contain 70% of tuples and test data contain the remaining 30% of tuples. Save the train data as *abalone-train.csv* and save the test data as *abalone-test.csv* in the same working directory.

    1. Use the attribute which has the highest *Pearson correlation coefficien*t with the target attribute Rings as an input variable and build a simple linear (straight-line) regression model to predict rings. (Prerequisite: calculate the Pearson correlation coefficient of every attribute with the target attribute rings). Print the prediction accuracy on the training data and testing data using root mean squared error. Compare the two RMSE.

       You are required to solve this problem by 2 methods:

        - **Manually implementing simple non-linear regression (using all the formulas used in approximating output)**

- **Using the in-built python library (scikit-learn).**

2. Build a multivariate (multiple) linear regression model to predict Rings. All the attributes other than the target attribute should be used as input to the model.
   Print the prediction accuracy on the training data and testing data using root mean squared error. Compare the two RMSE.

3. Use the attribute which has the highest Pearson correlation coefficient with the target attribute Rings as input and build a simple nonlinear regression model using polynomial curve fitting to predict Rings.
   Find the prediction accuracy on the training and testing data for the different values of degree of the polynomial (p = 2, 3, 4, 5) using root mean squared error (RMSE).
   Plot the bar graph of RMSE (y-axis) vs different values of degree of the polynomial (x-axis). What happens when p=1 ? According to you, which value of p should we use for our model ?
   You are required to solve this problem by 2 methods:
   - **Manually implementing simple non-linear regression (using all the formulas used in approximating output)**
   - **Using the in-built python library (scikit-learn).**

4. Build a multivariate nonlinear regression model using polynomial regression to predict Rings. All the attributes other than the target attribute should be used as input to the model.
   Find the prediction accuracy on the training data for the different values of degree of the polynomial (p = 2, 3, 4, 5) using root mean squared error (RMSE). Plot the bar graph of RMSE (y-axis) vs different values of degree of the polynomial (x-axis).According to you, which value of p should we use for our model ?

Hints -

For linear regression use:

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X, y)
#Input arguments: X: Input variable(s) of training data and y: target values
y_pred = predict(X)
#Predict using the linear model.
```

For polynomial curve fitting and regression use:

```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(p) #p is the degree
x_poly = poly_features.fit_transform(X)
regressor = LinearRegression()
 regressor.fit(x_poly, y) #Input arguments: x_poly: Polynomial expansion of  input variable(s) of training data and y: target values
y_pred = regressor.predict(X)
```