Python and Mathematics for Machine Learning

Download the reference Google Colab script from the shared files provided in this course or create your own and implement the code for the following exercises. Make sure your functions conform to the function API's and work with the doctest modules included in the Google Colab script provided. The material in this assignment has been covered in the previous lecture units so if you are having trouble understanding the questions, please reference the videos.

Part 1: Displaying and Interpreting Data

1. Normal distribution using scipy.stats
    a. The scipy.stats norm function can be used to generate samples according to the normal (gaussian) distribution. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html
    b. Generate a dataset of x_axis values using numpy arange from -5 to +5 in increments of 0.001. Generate a dataset of y_axis values using the scipy.stats norm function with mean 0 and standard deviation 1.
    c. Plot the distribution using matplotlib pyplot.
2. Normal distribution using numpy.randn
    a. The numpy randn library can be used to generate random samples from the normal distribution: https://numpy.org/doc/stable/reference/random/generated/numpy.random.randn.html
    b. Use numpy.random randn function to generate 10,000 samples centered around 0 and plot the distribution on a histogram plot with 10 bins and the same data with 50 bins (plot both distributions on the same histogram).
3. Numpy mean, median, standard deviation
    a. The numpy mean, median, std functions can used to calculate SOCV statistics
    b. Create 10,000 randn samples with a random offset that changes each time you run the cell. e.g. data = randn(10000) + randint(1,50)
    c. Calculate and print the mean, median and standard deviation of the distribution.
    d. Record the mean, median, standard deviation for 10 runs of the cell. How does each of the statistics change?
    e. Upload your results (mean, median, standard deviation). Comment on which of the statistics changed and why?
4. Interquartile Range. Create functions with perform the following and have the APIs listed below:

    a. The numpy percentile and scipy.stats iqr functions can be used to determine the interquartile range and identify data outliers:
https://numpy.org/doc/stable/reference/generated/numpy.percentile.html
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.iqr.html

    b. Create two datasets comprised of the following points: Note, these datasets will be created for you in the Colab script

```
data4 = [6, 5, 7, 22, 6, 5, 6, 8, 9, 12, 10, 20, 2, 35, 7, 2, 13, 15, 4, 10]
```

data4b = [32, 88, 36, 46, 92, 56, 101, 75, 79, 79, 1, 89, 91, 47, 116, 96, 97, 69, 170, 112, 93, 105]

c. Define a function which calculates and prints the first and third quartile, then find and print the interquartile range (IQR). Use the doctest module to test your function.

```
def IQR_calculator(dataIn):
 #dataIn = list of data
 #return IQR, Q1, Q3
```

d. Define a function which calculates and prints the IQR, upper outlier threshold, and lower outlier threshold.

```
def IQR_outlier_thresholds(IQR, Q1, Q3):
 #IQR = interquartile range
 #Q1, Q3 = Quartile 1, 3
 #return tuple of low and high outlier thresholds
```

e. Define a function which calculates and prints any outliers from the dataset. Use the doctest module to test your function

```
def IQR_find_outliers(dataIn, lowT, highT):
 #dataIn = list of data
 #highT = high outlier threshold
 #lowT = low outlier threshold
 #return list of low outliers, list of high outliers
```

5. Create boxplots for the 2 datasets in #4. Confirm the medan, Q1, Q3, and outlier points on the plots.

```
def plot_results():
 #no input
 #no return
```

Part 2: Interpreting Data with Z-scores

Download the reference Google Colab script from the shared files provided in this course or create your own and implement the code for the following exercises. Make sure your functions conform to the function API's and work with the doctest modules included in the Google Colab script provided. The material in this assignment has been covered in the previous lecture units so if you are having trouble understanding the questions, please reference the videos.

1. Using the numpy, scipy.stats, and matplotlib libraries, write a function that calculates the z-scores for a dataset composed of the following values. Note, these datasets will be created for you in the Colab script: *data = np.array([11, 13, 14, 15, 17, 17, 18, 18, 18, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20]*

2. Create a functions that calculates the the mean, standard deviation, and Z-scores with the following API:

```
def calc_zscores(data1):
 #data1 is the input data list
 #return mean, standard deviation, and z-scores
```

   a. Use this function for the raw dataset
   b. Use this function to compute the mean and standard deviation for the z-scores and ignore the returned z-scores parameter

3. Plot the data and Z-scores on two separate dot plots with the following API:

```
def plot_dot(data1, zScores):
 #Uses matplot lib histogram to plot raw data and z-scores
 #data1 is the raw input data
 #zScores are the computed standardized scores (zScores)
```

Upload your Colab script, screenshots of your console outputs, and reflection comments to this module.