CTML Take-Home Test

Hello! Thanks again for your interest in a role in the **Corporate Technology Machine Learning** (CTML) team at JPMorgan Chase & Co. This test consists of two challenges that should be completed using Python in a Jupyter Notebook; each part is intended to take no more than one hour.

Once complete, please save your work as a single pdf or html file and email it to your recruiter. Also, if you happen to have a public github repository, feel free to include a link to it in your email so we can inspect your previous work.

Challenge #1: Breast Cancer Dataset EDA

The "wdbc" data-set you will be working with can be downloaded here.

Please produce a *well-presented* Jupyter notebook – including any visualisations you feel are useful – which addresses the following:

a. What are the mean, median and standard deviation of the "perimeter" feature?
b. Is the first feature in this data set (the "radius") normally distributed? Quantify your answer. If not, what might be a more appropriate statistical distribution for modelling purposes?
c. Train a classifier to predict the diagnosis of malignant or benign. Compare the results of two classifiers (e.g. SVM, logistic regression and/or decision tree) and make some suggestions on how the classifier's performance could be further improved.

Challenge #2: Spearman's Footrule Distance

Suppose we have several different methods for scoring a set of items; perhaps we're asking different people or using different scoring algorithms. We'd like to figure out how to aggregate these to produce a single combined ranking.

A useful tool here is *Spearman's Footrule Distance* which computes the distance between two rankings (don't worry, we don't expect you to have heard of this before, we do expect you to do some Googling!)

Your task here is to implement a function with the following signature:

```
def sumSpearmanDistances(scores, proposedRanking):
    “”” Calculate the sum of Spearman’s Footrule Distances for a given proposedRanking.

        scores : A dict of {itemId: list of scores}, e.g. {‘A’: [100, 0.1], ‘B’: [90, 0.3], ‘C’: [20, 0.2]}
        means that item ‘A’ was given a score of 100 by metric 1 and a score of 0.1 by metric 2 etc...

        proposedRanking : An ordered list of itemIds where the first entry is the proposed-best and last the
        proposed worst, e.g. [‘A’, ‘B’, ‘C’] “””
```

Please think about splitting your function into appropriate sub-functions and add tests to demonstrate that everything works as expected. You may assume in your implementation that a higher score is better.