CS412: Introduction to Data Mining, Fall 2017, Homework 1

Name: Nestor Alejandro Bermudez Sarmiento (nab6)

Worked individually

For all the questions below, whenever a sum or a max operation appears it corresponds to a *sum* or *max* function available in Python's standard library. Both of which accept a list of values as an argument.
Wherever the size of the vector (n) is needed I use the *len* function.
All the results are rounded to 3 decimal positions.

Question 1

The *Question1.656944870.py* file implements the various statistical descriptions of the online scores data set.
Since we are asked to calculate the quartiles of the data one of the first things the program does is to sort the array of scores.

(a) MAX: since the data is sorted we only need to pick the last element; its value is **100.0**.

(b) MIN: since the data is sorted we only need to pick the first element; its value is **37.0**

(c) Quartiles: I implemented a function called *median* which, based on the length of the vector, it calculates the median. To calculate the quartiles I executed the median function over the entire vector and over the vector $\mathbf{H_1}$ and $\mathbf{H_2}$ where they were obtained by splitting the original vector in two.
As a result, the quartiles are:

$$Q_1 = \mathbf{68.0},\ Q_2 \text{ (median)} = \mathbf{77.0} \text{ and } Q_3 = \mathbf{87.0}$$

(d) Mean: as we know, the mean is calculated by summing all the values in the vector and then dividing by the length of the vector. The *mean* function does exactly that. After executing it over our online score data set the calculated mean is **76.715**.

(e) Mode: the *mode* function takes a vector and finds the element(s) with the highest frequency. To do so, it first creates a *frequencies* dictionary where the keys are the scores from our data set and the values are the number of times such values appears in the data set. Then it finds the maximum frequency and, finally, it looks for all the elements for which the frequency matches the maximum frequency. As a result, two modes are found: **77.0** and **83.0**.

(f) Empirical Variance: the *variance* function implements the following formula

$$\sigma^2 = \frac{\displaystyle\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}$$

Executing the *variance* function over the online scores data set we obtain **173.279**.

Question 2

Over the same data set for online scores we will perform z-score normalization and find the correlation and covariance between the mid-term scores and final scores. The z-score is given by

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of the values and σ is the standard deviation. This is implemented in the *Question2.656944870.py* file by implementing the *mean*, *variance* and *standard deviation* functions. Some of which reuse the code written for Question 1.

(a) Empirical variance before and after normalization. Before: **173.279**, After: **1.0**

(b) Normalize the original value 90: just apply the formula above to $x = 90$ and the result is **1.009**

(c) Pearson's correlation coefficient: this is given by the

$$r = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{A})^2}\sqrt{\sum_{i=1}^{n}(b_i - \bar{B})^2}}$$

where:
n is the number of samples
a_i, b_i are the single samples indexed with i
$\bar{A} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the mean of the samples); and analogously for $\bar{B}$.

The Python program implements it by first calculating the mean for both samples and uses list comprehension to calculate the numerator of the formula and two more list comprehensions to calculate the denominator.

After applying the function to the midterm and final scores we get a $r =$ **0.544**

(d) Covariance between the midterm and final scores.

$$\frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}$$

where the variables are defined as in the previous item.

The Python program implements a function that calculates the mean (same as for the previous item) and uses list comprehension to calculate the covariance. Finally, instead of using n as the denominator, the problem uses $n - 1$ as requested in the assignment. For the given midterm and final scores data sets the covariance is **78.254**

Question 3

Given the data sets for the book inventory for both CML and CBL libraries.

(a) Jaccard similarity coefficient

Given the following table, calculate the Jaccard similarity coefficient.

	In CML	Not in CML
In CBL	58	2
Not in CBL	120	20

As described in the slides I'll drop the value for when a book is not present in either library (20) so

$$sim_{jaccard} = \frac{58}{58 + 2 + 120} \approx \mathbf{0.322}$$

(b) Minkowski distance for $h = 1, 2$ and ∞
I have implemented each of them using the list comprehension feature of Python under the *Question3.656944870.py* file. I use the *math* package for calculating the absolute value and square roots where needed.

i. For $h = 1$ (Manhattan distance),

$$L_0 = \sum_{i=1}^{n} |a_i - b_i|$$

where a_i is the book inventory for CML and b_i is the book inventory for CBL. As a result we obtain: **6152.0**

ii. For $h = 2$ (Euclidean distance),

$$L_1 = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

where a_i is the book inventory for CML and b_i is the book inventory for CBL. As a result we obtain: **715.328**

iii. For $h = \infty$ (Supremum distance),

$$L_\infty = max|a_i - b_i|$$

where a_i is the book inventory for CML and b_i is the book inventory for CBL. As a result we obtain: **170.0**

(c) Cosine similarity,

$$cos(A, B) = \frac{A \bullet B}{\|A\| \times \|B\|}$$

where A is the book inventory vector for CML and B is the one for CBL.
This function has been implemented in *Question3.656944870.py* Python program under the name *cosine_similarity*.
As a result we obtain: **0.841**.

(d) KL Divergence,
For this one we are assuming that the probabilistic distribution for CML's inventory is

$$P(i) = \frac{a_i}{X}$$

where X is the total count of books in CML and a_i is the count of the ith book.

We will assume the same for CBL inventory and denote it by Q.

As we know,

$$D_{KL}(P\|Q) = \sum_{i=1}^{X} P(i) log \frac{P(i)}{Q(i)}$$

The *kl_divergence* function implements this by first calculating the total count of the books for both CML and CBL and then iterating over all the elements calculating $P(i)$ and $Q(i)$ for each of them. And finally sum all the calculated values
As a result we obtain: **0.207**

Question 4

Using

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r} \frac{(o_{\text{ij}} - e_{\text{ij}})^2}{e_{\text{ij}}}$$

Calculate χ^2 correlation value given the table below.

	Buy diaper	Do not buy diaper
Buy beer	150	40
Do not buy beer	15	3300

The Python program takes an arbitrary matrix (not necessarily 2x2) and it will calculate the sum of each column, sum of each row and n (sum of all values). For this particular problem, the totals are

	Buy diaper	Do not buy diaper	Total
Buy beer	150	40	190
Do not buy beer	15	3300	3315
Total	165	3340	3505

To calculate the sums of the columns and rows I use list comprehension.
Once the values are calculated we just need to iterate over each item in the matrix (c columns and r rows) and use the observed and expected values. Where the expected values are calculated based on the sums of the columns, rows and n.

The expected values for this data (rounded to 3 decimals in this report) are:

	Buy diaper	Do not buy diaper
Buy beer	8.944	181.056
Do not buy beer	156.056	3158.944

Finally, the χ^2 correlation value for the data above is **2468.183**.

Additionally, the degrees of freedom this distribution is

$$(c-1) \times (r-1) = (2-1)(2-1) = 1$$

.
Which means that the p-value is very small and the null hypothesis of independence can be rejected. Therefore we can conclude that there is very strong evidence that there is a correlation between the purchase of beers and diapers.