### 0.0.1 Question 2.1: Prove that KL Divergence is Non-negative

Prove that

$$D_{KL}(p(x)||q(x)) \geq 0$$

Hint: - Apply Jensen's inequality, which says that $\mathbb{E}[f(z)] \geq f(\mathbb{E}[z])$ for any convex function $f : \mathbb{R} \to \mathbb{R}$ and for any random variable $z$. - In this case, you should use $f(z) = -\log(z)$ - Note that $\log \frac{a}{b} = \log(a) - \log(b) = -(\log(b) - \log(a)) = -\log \frac{b}{a}$ - Think about the case where the divergence would be lowest... when the two distributions are the same!

*Points:* 2

$$D_{KL}(p(x)||q(x)) = \mathbb{E}_p(x)\left[\log \frac{p(x)}{q(x)}\right]$$

Apply Jensen's inequality: $\mathbb{E}[f(z)] \geq f(\mathbb{E}[z])$

$$\mathbb{E}_p(x)\left[-\log \frac{q(x)}{p(x)}\right] \geq -\log\left(\mathbb{E}_p(x)\left[\frac{q(x)}{p(x)}\right]\right)$$

$$\mathbb{E}_p(x)\left[\log \frac{p(x)}{q(x)}\right] \geq -\log\left(\mathbb{E}_p(x)\left[\frac{q(x)}{p(x)}\right]\right)$$

If $p(x) = q(x)$: $\mathbb{E}_p\left[\log \frac{p(x)}{q(x)}\right] = -\log(\mathbb{E}_p[1]) = 0$

Hence, $D_{KL}(p(x)||q(x)) \geq 0$

### 0.0.2 Question 3.4: Compare and discuss

Let's discuss what you saw with the various plots.

Particularly we are looking to see how companies appear to be similar to each other in the reduced dimensions. If two companies are near each other in low-d assumedly their stocks move up and down together. This might happen because they are in the same industry and there are good weeks and bad weeks for oil stocks that are different than good/bad weeks for tech stocks. But industry isn't everything; clearly inside an industry there could be companies doing well while others do poorly. So looking for clusters of industry is simply something I assume might be there, but doesn't have to be. Think of this as the default "pattern" I'm asking you about in the questions below.

**NOTE:** I'm not a broker or an economist, so there are probably other reasons for companies clustering together or separating from each other. If you have particular expertise or are gung-ho to do some research about these companies in this time period I'm curious to see what other things you come up with. So hit me with your best explanations for why "patterns" exist in the low-d data! Think of this as a mini project… this is the kind of subject matter expertise that you will need to bring to your projects :)

So in the Markdown box below please address the following questions in a short essay format. *Your response should probably be 1 (and no more than 2!) paragraphs per bullet point below.* - Characterize what the PCA plot looks like, say what patterns you can see - Characterize differences among t-SNE plots, say what patterns you can see. Talk about how these patterns appear/disappear/modify with the different parameters. Relate these pattern changes to what you know about the nature of high and low perplexity - Characterize differences among UMAP plots, say what patterns you can see. Talk about how these patterns appear/disappear/modify with the different parameters. Relate these pattern changes to what you know about the nature of high and low n_neighbors - Why didn't I ask you to investigate even larger values for perplexity or n_neighbors, like e.g. 100? What happens when you try that with the algorithm? - What differences do you see in the plots between the 3 algorithms? Do you feel like there's a strong reason to prefer one of these algorithms? Reasons might include results, computational complexity, hyper-parameter tuning, and anything else you feel is relevant - Given what you've seen here, what conclusions do you feel comfortable drawing about these stocks and why? What else do you suspect but would feel like you can't prove given this analysis?

*Points:* 3.25

1. In the PCA plot there is a dense cluster of companies near the origin, which includes car manufactures, banks, retail companies, and food industry companies. 2 other cluster are identifiable in the plot: one in the positive direction of the first PC (tech companies like Amazon and Apple) and another one in the positive direction of the second PC (oil/energy companies like Chevron and ConocoPhilips). It might be the case that tech companies, overall, had a rising in stock prices during 2003-2007, corresponding to the years prior the launch of the first iPhone. This rising was probably slow and mild because of the necessary recovery from the dot-com crush of the 2000s. Oil and energy-related stocks, however, had an extreme rise during the same years, with crude oil prices going from 30$ in 2003 to 90$ in 2007. Lastly, the companies clustered in the center of the plot were likely stable during these years because they are well-affirmed and have a consolidated market.

2. In the first t-SNE plot (`perplexity=5`) similar companies are in the same part of the plot, but they are not unequivocally clustered. These including: energy/oil companies, banks, tech companies, food and beverages companies, and car manufacturers. At a first glance, indeed, the plot resembles a

uniform distribution In the second t-SNE plot (`perplexity=20`) there are a few apparent clusters: energy companies (Chevron, Exxon etc.) banks (Goldman Sachs, JP Morgan etc.), defense companies (Northrop Grumman, etc.) and personal care companies clustered with Pepsi and Coca-Cola. The remaining companies are not tightly clustered and appear uniformly distributed. In the last t-SNE plot (`perplexity=40`) there is a dense cluster of companies in the center of the plot, with banks, tech companies, pharmaceutical companies, energy providers and even drink-producers. These plots reflect the different values of perplexity set in the dimensionality reduction algorithm: low perplexity indicates fewer neighbors/smaller standard deviation, hence less dense clusters; Whereas high perplexity tends to generate bigger clusters. There is a tradeoff between having a perplexity high enough to reach the true neighbors, but not too high to cluster together unrelated companies (as in the last plot).

3. In the first UMAP plot (`n_neighbors=5`), similar companies are close in the plot-space, with a few clear clusters: energy companies, banks, personal care companies and defense companies. In the second UMAP plot (`n_neighbors=20`), similar companies are still relatively close to each other in the plot space, but it is harder to identify well-divided clusters. In the last UMAP plot (`n_neighbors=40`), the overall position of the companies in the plot-space is different, and some clusters become more easy to identify compared to the previous plot (including banks and energy/oil companies). These changes reflect the different values of `n_neighbors`:

   - `n_neighbors=5` makes the algorithm focus on local structures, emphasizing similarities in small neighborhoods, and hence resulting in more well-defined clusters.
   - `n_neighbors=20` forces UMAP to consider more broad structures, but these are not broad enough to capture the overall trends in the data. Hence, the result is random-looking low dimensional embedding with hardly identifiable clusters.
   - `n_neighbors=40` allows UMAP to consider the global structure of the data, since each data point is analyzed in the context of its 40 nearest neighbors. This implementation highlights more general trends in the data, with clusters appearing again, although less separated than the first plot.

4. There are only 56 companies in the dataset `daily_change`, therefore a number greater than 55 for `n_neighbors` is meaningless, since there cannot be more than that number of neighbors. Setting `n_neighbors=55`would force UMAP to compare each datapoint to all the other datapoints, and hence attempt to find the most local structure possible - this is in contrast to the purpose of the algorithm itself. If I try to run UMAP with, for example, `n_neighbors=100`, the algorithm itself will point out that the number of neighbors is greater than the number of datapoints, and it will automatically consider only 55 neighbors.

5. There are clear differences between PCA, t-SNE, and UMAP applied to the `daily_change` dataset:

   - PCA is overall under-performing at maintaining a nuanced representation of the dataset. It creates a big cluster with many different types of companies, but it is a good way to identify very different stock trends (tech companies and energy providers). Nonetheless, PCA is very computational effective and deterministic.
   - T-SNE is performing decently on this task (especially with perplexity=20), generating meaningful compact clusters, although likely misplacing some companies (for example, in the second t-SNE plot, McDonald's and Pfizer are far away from all similar companies). T-SNE is also computationally expensive (scales with the square of the dataset size) and highly dependent on the perplexity value.
   - UMAP is performing best at embedding the `daily_change` dataset (especially with `n_neighbors=5`). It largely captures meaningful relationships both locally and globally, generating an overall interpretable plot. UMAP is also faster than t-sne because it scales with the dataset size, although it requires the fine tuning of at least 2 parameters. Overall, in this specific context, UMAP is preferable due to its clearer results and computational effectiveness. The hyper-parameter fine-tuning is not problematic since the dataset has only 56 observation, which bounds the number of neighbors to 55.

6. Overall, there are some common patterns to all three dimensionality reduction algorithms, which suggest that the stocks of these companies are rising or falling together in the years 2003 to 2007. These are: tech companies, oil/gas companies, defense companies, and banks. Tech companies are likely increasing their market value slowly after the dot-com crash and in preparation to the upcoming digital era (release of the iPhone, 2008). Oil and energy companies are increasing their stock value significantly in those years too, more dramatically than tech companies. Defense companies are also experiencing an increase in stock value after the 2001 attacks, and due to the foreign involvement of the United States. The favorable economic climate is leading banks to rapid growth as well, which will come to an abrupt end with the 2008 financial crisis. The fact that these sectors are in different clusters likely represents different rates of stock prices change.

   As for the remaining companies, there is some ambiguity. Some of them are clustered by UMAP but not by t-SNE (for example McDonald's), making it hard to draw conclusions about their stock prices. It could be the case that these companies experience their own trajectory in the market, although it is impossible to determine using dimensionality reduction algorithms.

### 0.0.3 Question 4.3: Model Performance Evaluations

The following code visualize the performance of each individual model in the `GridSearchCV`. Comment on which model performs the best, with which set of hyperparameter.

*Points:* 1

```
In [36]: results = pd.DataFrame( grid.cv_results_['params'] )
         results['score'] = grid.cv_results_['mean_test_score']
         # heres a tibble of your grid search results

         results.head()
```

```
Out[36]:    classifier__n_neighbors  dim_reduce__min_dist  dim_reduce__n_neighbors  \
         0                        5                  0.01                        5
         1                        5                  0.01                       10
         2                        5                  0.01                       15
         3                        5                  0.10                        5
         4                        5                  0.10                       10

               score
         0  0.901333
         1  0.893000
         2  0.872667
         3  0.904667
         4  0.892667
```

```
In [37]: index_of_max = results['score'].idxmax()
         row_with_max = results.loc[index_of_max]

         print(row_with_max)
```

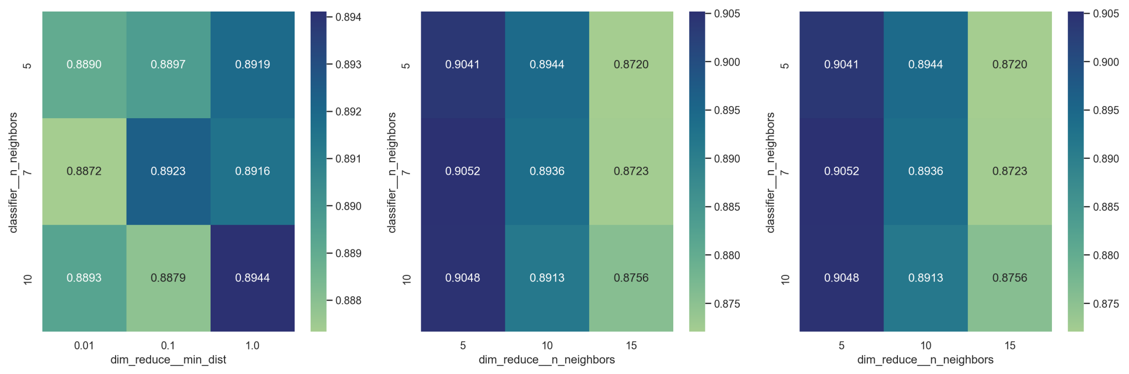
```
classifier__n_neighbors    7.000000
dim_reduce__min_dist       1.000000
dim_reduce__n_neighbors    5.000000
score                      0.906667
Name: 15, dtype: float64
```

```
In [38]: # this makes a 2d heatmap comparing JUST TWO of the parameters you are interested in

         # YOU MUST INTERVENE HERE
         # the first two args to .pivot (index, columns)
         # must be the names of the parameters you want to score
         # right now this assumes you are comparing the first two params
# modify it according to what you need to visualize!!!
# if you're doing 3 params, you may need to make 2 heatmaps, one for each pair of params

grid_params = results.columns
heatvals1 = results.pivot_table(index=grid_params[0],
                                columns=grid_params[1],
                                values='score')

heatvals2 = results.pivot_table(index=grid_params[0],
                                columns=grid_params[2],
                                values='score')

heatvals3 = results.pivot_table(index=grid_params[1],
                                columns=grid_params[2],
                                values='score')

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(20, 6))

sns.heatmap(heatvals1, annot=True, fmt='5.4f', robust=True, cmap="crest", ax=axes[0]);
sns.heatmap(heatvals2, annot=True, fmt='5.4f', robust=True, cmap="crest", ax=axes[1]);
sns.heatmap(heatvals2, annot=True, fmt='5.4f', robust=True, cmap="crest", ax=axes[2]);

display(heatvals1)
display(heatvals2)
display(heatvals3)
```

```
dim_reduce__min_dist         0.01      0.10      1.00
classifier__n_neighbors
5                        0.889000  0.889667  0.891889
7                        0.887222  0.892333  0.891556
10                       0.889333  0.887889  0.894444
```

```
dim_reduce__n_neighbors        5         10        15
classifier__n_neighbors
5                        0.904111  0.894444  0.872000
7                        0.905222  0.893556  0.872333
10                       0.904778  0.891333  0.875556
```

```
dim_reduce__n_neighbors        5         10        15
dim_reduce__min_dist
0.01                     0.903111  0.890556  0.871889
0.10                     0.904889  0.891556  0.873444
1.00                     0.906111  0.897222  0.874556
```



In [39]:
```
# NB some people have a problem where you only see the first row
# of the numeric annotations in the heatmap above
# if that affects you no big deal.  thats why I made it print the results grid too
# if you want, you can try to uncomment the following, run it, and restart your kernel
# %pip install seaborn --upgrade
```

In my grid search, after finding the broad range of hyperparameters to test, I fitted my pipeline 81 times. I tested various combinations of `min_dist`, `n_neighbors` (hyperparameters of UMAP), and `n_neighbors` (hyperparameter of KNN). Ultimately, the best combination of value is: - `classifier__n_neighbors`: 5.000000 - `dim_reduce__min_dist`: 0.100000 - `dim_reduce__n_neighbors`: 5.000000

which results in an accuracy value of 0.901667.