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

Type your answer here, replacing this text.

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

The PCA plot looks like it has a tight cluster at the bottom with some more well known companies that are spread out throughout the space. I think the pattern to be observed is that the more stable stocks that are less likely to dip or shoot up drastically aren't clustered with other companies but are stand alone. While the more unstable stocks are clustered together.

I noticed in the TSNE plots that the higher the perplexity values are, the more clustering there is but less clusters. This makes sense because perplexity is the number of neighbors you want to retain for a point while moving from high-D to low-D.

In the UMAP plots, I've noticed that the higher the n_neighbors parameters is the more equally spread out. The lower it is, the more distinct clusters appear, and the higher it is, the less likely clusters are to be made because there's less points being placed close to each other.

There's no point in investigating higher values for perplexity and n_neighbors because that would lead to overfitting.

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 [30]: 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[30]:

	knn__n_neighbors	umap__n_neighbors	score
0	3	3	0.902333
1	3	5	0.911667
2	3	7	0.909667
3	5	3	0.901667
4	5	5	0.910333

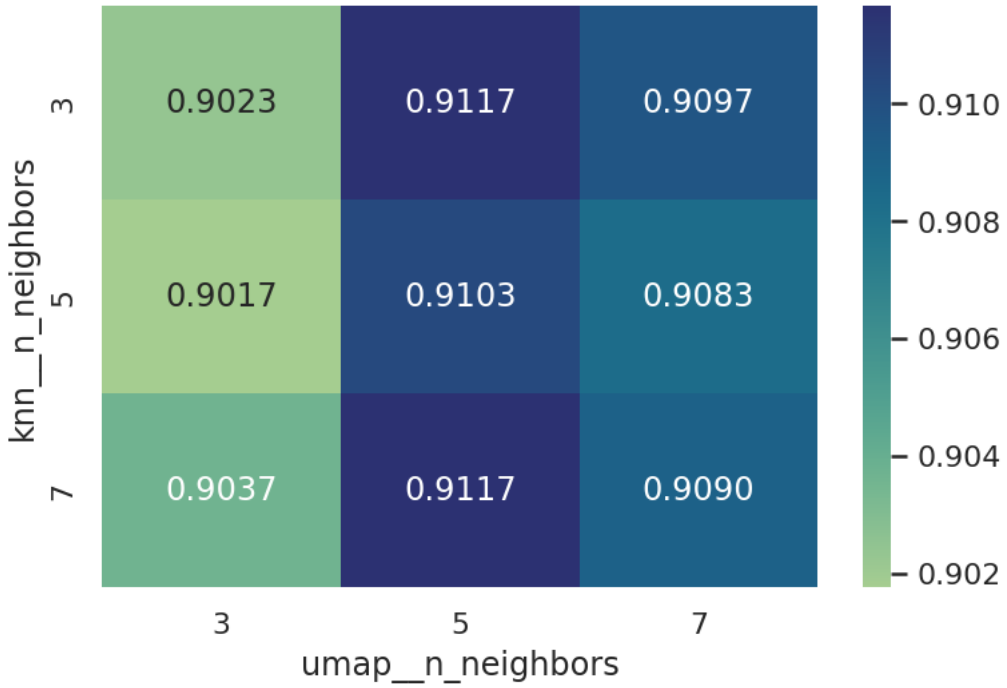
```
In [31]: # this makes a 2d heatmap comparing JUST TWO of the parameters you are interested in

         # 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
         heatvals = results.pivot(index=grid_params[0],
                                  columns=grid_params[1],
                                  values='score')
         sns.heatmap( heatvals, annot=True, fmt='5.4f', robust=True, cmap="crest");
         heatvals
```

Out[31]:

umap__n_neighbors	3	5	7
knn__n_neighbors			
3	0.902333	0.911667	0.909667
5	0.901667	0.910333	0.908333
7	0.903667	0.911667	0.909000



In [32]:
```
# 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
```

The model with 5 nearest neighbors for both umap and knn is the best model.