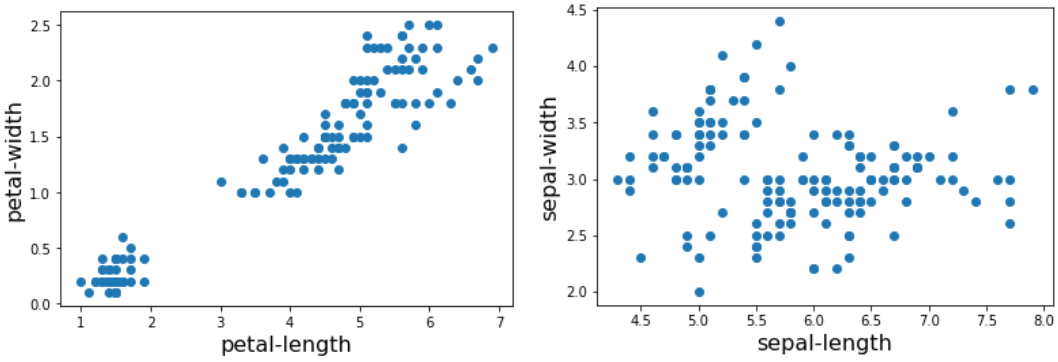
a. Based on accuracy, which dimensionality reduction method, PCA, simulated annealing, or the genetic algorithm, worked the best?

Considering accuracy alone, the genetic algorithm (0.960) for feature selection was most effective at generating a high-scoring truncated list of features. Second to this was simulated annealing (0.953).

b. For each of the two other methods, explain why you think it did not perform as well as the best one.

PCA

The idea of PCA is to isolate a subset of transformed features such that the PoV > 0.90, according to both lecture notes and assignment instructions. Inherent in the transformation, however, is the covariance matrix, which assumes that the features are linearly correlated. I suspect this is problematic in that some features *are* linearly correlated (petal width v. petal length) while others are simply not (sepal width vs. sepal length).



Simulated Annealing

This feature selection method is not as ambitious as GA in terms of sampling phase space in order to find the optimal solution. As such, despite finding a feature subset that improves the model score beyond using the original features, the algorithm seemed to have ultimately settled in a sub-optimum valley and struggled to eject itself. Quite relatable.

c. Did the best dimensionality reduction method produce a better accuracy than using none (i.e. the results of Part 1)? Explain possible reasons why it did or did not.

GA produced a 0.96 compared with Part One's 0.927. If we again refer only to the accuracy scores, then, GA generates a more suitable set of features. I do think there is more to the narrative, however:

- GA's best individual after 50 generations consisted of 6 features, two more than that for Part One. One aim of feature selection is to reduce dimensionality while simultaneously not surrendering, and perhaps even raising, accuracy. An accuracy rise of 0.033 may not justify the addition of two features. *Note, however, that I (and/or others more prolific) could have tweaked the code such that GA seeks the lowest subset of features with accuracy ~ original, as I had done with SimulatedAnnealing.*
- The Decision Tree classification model also is a bit flimsy in that each iteration may output a model inconsistent with the last -- that is, rather than selecting for the tree with the best accuracy, it simply generates *a* tree. Having run this Part One model code multiple times on the original features, the accuracy can reach comparable values to GA's output. Thus, not in every instance is the gap between accuracies even as high as 0.033 (which is somewhat low to begin with). What is neat about GA, though, is that it settles on a 0.96 accuracy solution for nearly every execution!
- Also noteworthy is that DecisionTrees is run both on the set of 4 features *and* the 8 features. For the three iterations of CompareFeatureSelectionMethods.py, GA's resultant accuracy score was highest.

d. Did Part 2 produce the same set of best features as Part 3? Explain possible reasons why it did or did not.

PCA was only measuring the proportion of variance for the four transformed features (z1, z2, z3, z4), while Simulated Annealing analyzed all eight features (x1, x2, x3, x4, z1, z2, z3, z4). The former found an accuracy of 0.913 for one DecisionTree iteration with the z4 feature alone, while Simulated Annealing found a combination of the original and transformed features to yield a higher accuracy of 0.953. Simply stated, Simulated Annealing had more options to utilize.

e. Did Part 2 produce the same set of best features as Part 4? Explain possible reasons why it did or did not.

I think a similar reasoning as above applies here -- Simulated Annealing simply had more options to puzzle over to find a high accuracy score, any one of those options happened to yield a higher accuracy than that of PCA.

f. Did Part 3 produce the same set of best features as Part 4? Explain possible reasons why it did or did not.

I wrote briefly in a previous response that I tweaked the Simulated Annealing algorithm such that if the code found a subset with an accuracy score that matched the current best accuracy score, but that subset's length was *lower* than that of the best subset, then the new best subset would become that lower-length subset. I considered doing likewise for GA, but I settled adding a comment in the script itself as an exercise for the reader because:

1. The code functioned as desired, and I feared fumbling further by adding additional loop statements would just prolong the assignment longer than necessary
2. GA found an optimal accuracy solution of 0.96 with a somewhat large feature subset that was, for 50 generations, a higher score than the accuracies calculated with smaller subsets (though the second highest accuracy, notably for subsets of 2-3 features, was 0.953). I used this as a pseudo-justification for leaving GA as is, despire (1) being the more pressing consideration.

****A second exercise for the reader: determine if the z transformed features selected for Part 2, Part 3, Part 4 correspond with the features that exhibit linear correlations.*