# Take-Home Coding Assessment: L3 Associate Data Scientist

1. **Machine Learning:**
   1. Fit a decision tree classifier. How does it perform?

Answer: I was able to achieve an average classification accuracy of **96.6 %** across 10-fold cross validation.

* 1. What about a K-Nearest Neighbors model? Please try with 1, 10, 20, 50 and 80 neighbors. Why do you think 50 and 80 neighbors works less well (answer in one or two sentences)?

Answer: My model had **98 %** accuracy with 1 neighbor. But as I increased the size of neighbors the error rates seemed to drastically increase. I believe that this is because of the fact that in KNN model, we try to find the k nearest neighbors of a data point and do a majority class voting to therefore predict the test data point class. Now when the size of the dataset is relatively small and if the K value is large (in our case if we assume k=50 then it means that we’ll have to consider 50% of our training set that has 105 data points to label an unknown data), the algorithm faces a lot of confusion while making its decision based of the majority voting and hence is prone to make wrong classifications leading to increase in error rate. The same applies for k=80. Now this might work well if our training dataset is quite large with thousands of data points.

* 1. Fit a Random Forest model. How does it perform? Is it better or worse than your decision tree? Briefly comment on why (answer in one or two sentences)?

Answer: Random Forest almost always performs better than decision trees and my accuracy speaks for that. My model had **97.3 %** accuracy on an average on 10-fold cross validation. This is because the trees are more independent of each other compared to regular decision trees. It works by generating multiple classifiers/models which learn and make predictions independently. Those predictions are then combined into a single (mega) prediction that should be as good or better than the prediction made by any one classifer. which often results in better predictive performance. That being said, it’s also faster than bagging, because each tree learns only from a subset of features.

**NOTE: I’ve attached an ipython notebook file also to provide an option for you to better understand my analysis. (Iris Classification.ipynb)**

1. **Implementing an Edit-Distance algorithm:**
   1. Describe a scenario (3-4 sentences) where implementing the standard Hamming distance algorithm would be applicable.

Answer: Hamming distance is crucial to many big data search applications. One particular areas of application that strikes my mind is the usage of HD in approximate near neighbor search for high dimensional data such as images and document collections. Here, using similarity hash functions, high dimensional data is mapped into one dimensional binary codes that are then linearly scanned to perform hamming distance comparisons to better understand feature vectors.

1. **Data Cleaning:**
   1. How many of the descriptions mention an embodiment or that they embody something? That is, in how many of the description does the stem “embod-” appear with any ending? (i.e., embody, embodiment, embodying, etc. should all count).

Answer:

Total no. of patents that had the substring ‘embod’ with any ending – 571

Total no. of distinct image descriptions that had ‘embod’ with any ending – 1317

* 1. What if we are specifically interested in drawing descriptions that embody an **invention**? So, here, we would like to identify descriptions that contain phrases like the "embodiment of the present invention", "embodies my invention", etc. How many descriptions have the word “embod-” (with any ending) followed by "invention", even if the two terms are separated by words?

Answer:

Total no. of patents that had the substring ‘embod’ with any ending and also followed by the keyword ‘invention’ – 155

Total no. of distinct image descriptions that had ‘embod’ word with any ending and followed by keyword ‘invention’ - 226