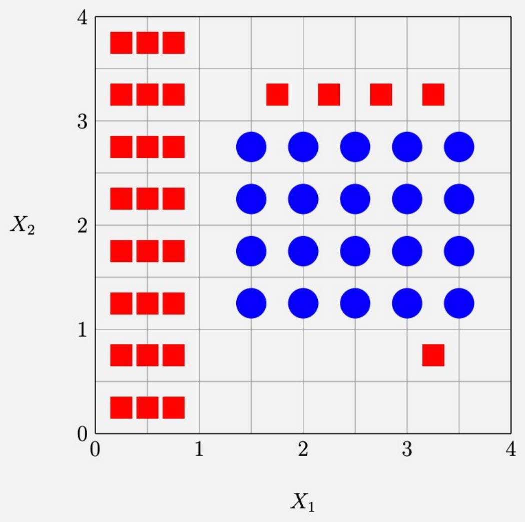
ComS 474 Final

Gavin Monroe - Spring 2020

1) Consider the following scatter plot of training samples. There are two classes, Y = red and Y = blue. We will use a single decision tree to achieve perfect classification.

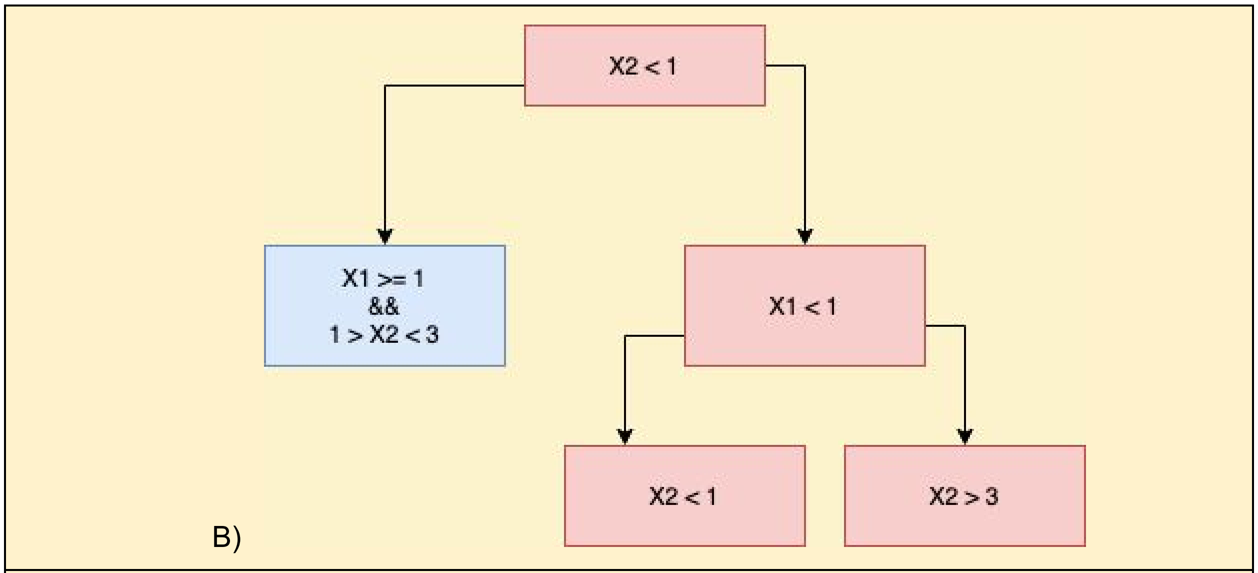


A. In 2-5 sentences, briefly explain how a decision tree is constructed.

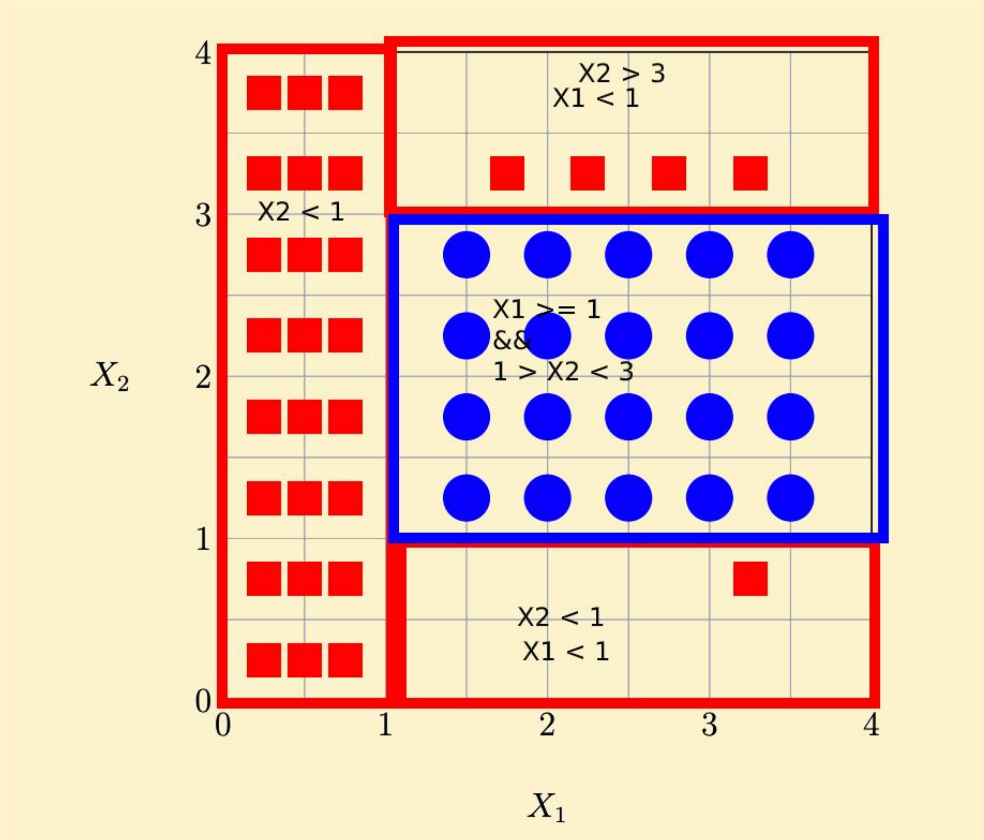
B. Draw the decision tree diagram, labeling edges and the predictions. You do not need to explicitly calculate Gini values for this problem. You should be able to use your knowledge of what Gini values measure to determine the splits.

C. Sketch the corresponding partition of the X1, X2 plane. Label each region with the corresponding prediction. You do not need to redraw the training samples themselves. Just the decision boundaries corresponding to a decision tree.

A) A decision tree is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits. It uses a “tree-like” model of decisions as the name gives. They can be used either to drive informal discussion or to overall map out an algorithm that predicts the best choice mathematically.



B)



C)

2)Suppose we have a training data set of 100 samples with a feature Y that we want to predict on new data. We have five features X1, X2, . . . , X5 that we can use to predict Y . We will use (multiple) linear regression.
The mean value of Y across training samples is 1 100 X 100 i=1 Y (i) = 3.8.
We are concerned that this model may under-fit.

Using all of the features, we find that arg min a0,a1,...,a5 1 100 X 100 i=1 Y (i) − (a0 + a1X1(i) + · · · + a5X5(i))2 is Yb = 2.1 − 0.4X1 + 1.3X2 + 3.8X3 − 0.9X4 + 0.5X5.
We are concerned that this model may over-fit.
Suppose we decide to include a penalty on complexity of the form arg min a0,a1,...,a5 1 100 X 100 i=1 Y (i) − (a0 + a1X1(i) + · · · + a5X5(i))2 + λ X 5 j=1 |aj | p .

A) 1: λ = 0
Penalty is eliminated which causes the data to over fit. This allows the graph to hit most if not all targeted data points, which could be used for the sample. We then use all the models that over fit, which is caused when lambda is to over fit. Ridge
2: λ → ∞
Penalties wouldn't be limited, making the model underfit due to the two equations not cancelling. Lasso

B) When λ = 0, the punishment term has no effect, and the assessments created by edge relapse will be equivalent to least squares. In any case, as λ→∞, the effect of the shrinkage punishment develops, and the edge relapse coefficient evaluations will move toward zero. As can be seen, choosing a decent estimation of λ is basic. Cross approval proves to be useful for this reason. The coefficient gauges created by this strategy are otherwise called the L2 standard.

a) ^ up above.
b) Overfit with the penalty removed on the right side we can see the iteration for the equation with Lasso would be optimized with 100 calls.
c) The two core heuristics are Metaheuristics and Hyper-heuristics which use sensor data trees to overall help with enforced supervised learning. Meta: search method for finding a reasonable solution within a reasonable time. For Hyper-heuristics perform search over the space formed by a set of low level heuristics, rather than solutions directly. Hyper-heuristics are not allowed to access problem domain specific information. Hyper-heuristics has an advantage of being more general than the existing search methods. Metaheuristics (i.e., their implementations) have to be tailored for a specific problem domain and often, they are successful in obtaining high quality solutions for that domain. However, metaheuristics being a subset of heuristics come with no guarantee for the optimality of the obtained solutions

C) P =1 then we would have 100 iterations of iterations with 5 iterations with each one of those iterations because of Lasso, with multiple with ace of j. P =1 the value we multiply by with penalty just being bigger. With p = 2 just being a bigger penalty but still reaching 500 iterations.

D) The difference between lasso and ridge is the explantional grow that leads to lambda reaching further into infinite, causing the penalty to grow along with it. This is because ridge and lasso don't operate the same as one square and the other does not.

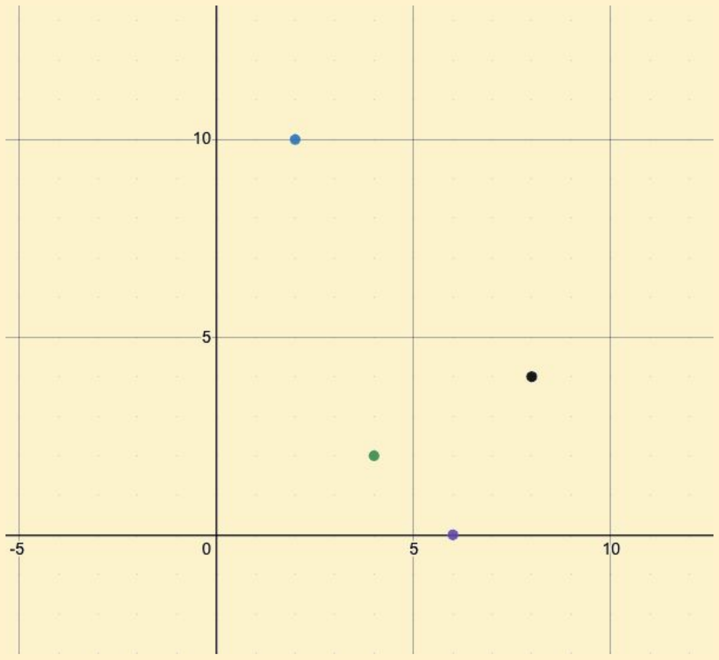
3) Problem 3: In this problem, we consider k nearest neighbor regression. Suppose we

have the following (training) data set.

X	Y
2	10
4	2
6	0
8	4

For each k ∈ {1, 2, 3, 4},

• Make a scatter plot with X values on horizontal axis, Y values on the vertical axis.
• Plot the (unweighted) k-nearest neighbor prediction function over the range X ∈ [−2, 12].
• Show the calculations for determining the prediction function over that range and explain the process.

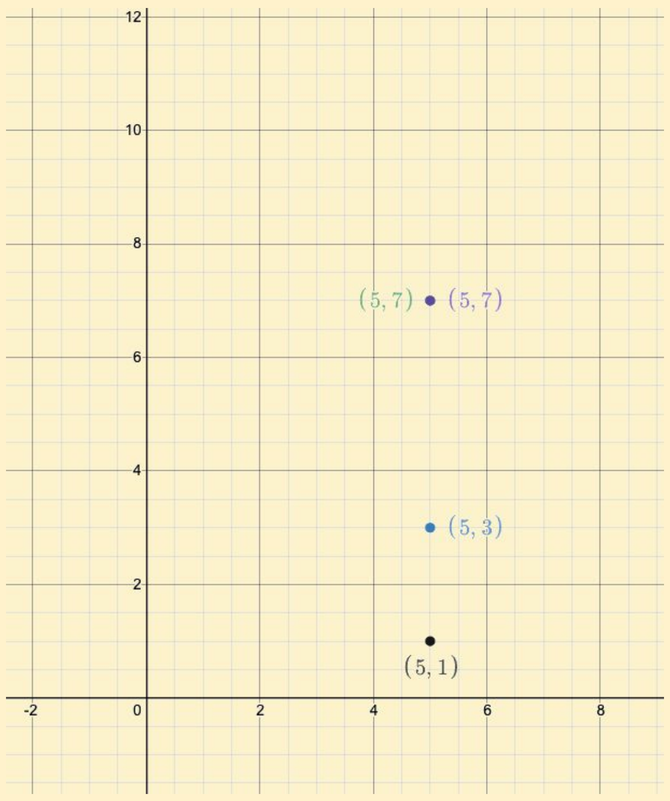


Running kNN classifier mathematics in python gets me this out put we can then predict x and y from the following information:

For: [2, 10, None]
Dist: 2.0 for: 2 , 4
Dist: 8.0 for: 10 , 2
Dist: 4.0 for: 2 , 6

```
Dist:  10.0  for: 10 , 0
Distances:  [([4, 2, 2], 8.246211251235321), ([6, 0, 2], 10.770329614269007)]
Neighbors:  [[4, 2, 2], [6, 0, 2]]
kNN = 2
```
For: [8, 4, None]
```
Dist:  4.0  for: 8 , 4
Dist:  2.0  for: 4 , 2
Dist:  2.0  for: 8 , 6
Dist:  4.0  for: 4 , 0
Distances:  [([4, 2, 2], 4.47213595499958), ([6, 0, 2], 4.47213595499958)]
Neighbors:  [[4, 2, 2], [6, 0, 2]]
kNN = 2
```
For: [4, 2, None]
```
Dist:  2.0  for: 4 , 2
Dist:  8.0  for: 2 , 10
Dist:  4.0  for: 4 , 8
Dist:  2.0  for: 2 , 4
Distances:  [([8, 4, 2], 4.47213595499958), ([2, 10, 2], 8.246211251235321)]
Neighbors:  [[8, 4, 2], [2, 10, 2]]
kNN = 2
```
For: [6, 0, None]
```
Dist:  4.0  for: 6 , 2
Dist:  10.0  for: 0 , 10
Dist:  2.0  for: 6 , 8
Dist:  4.0  for: 0 , 4
Distances:  [([8, 4, 2], 4.47213595499958), ([2, 10, 2], 10.770329614269007)]
Neighbors:  [[8, 4, 2], [2, 10, 2]]
kNN = 2
Scores: [100.0, 100.0]
Mean Accuracy: 100.000%
```
(8 + 2) / 2 = 5

X	Y	Distance	Pred X	Pred Y
2	10	13.625	(8 + 2) / 2 = 5	(6 + 0)/2 =3
4	2	8.59	(8 + 2) / 2 = 5	(4 + 10)/2 =7
6	0	9.855	(8 + 2) / 2 = 5	(4 + 10)/2 =7
8	4	6.705	(6 + 4) / 2 = 5	(4 + 10)/2 =1



Stage 1: Calculate the separation

So as to make any expectations, you need to compute the separation between the new point and the current focus, as you will require k nearest focus. For this situation for computing the separation, we will utilize the Euclidean separation. This is characterized as the square base of the entirety of the squared contrasts between the two varieties of numbers.Specifically, we need just initial 4 attributes(features) for separation estimation as the last property is a class name. So one of the methodology is to constrain the Euclidean separation to a fixed length, along these lines overlooking the last measurement.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Step 2: Number of neighbors set to 2, due to the predictor function found in the main.py file.

$$\hat{y} = \underset{y=1,\dots,K}{\arg\min} \sum_{j=1}^{K} \hat{P}(j|x)C(y|j),$$

where

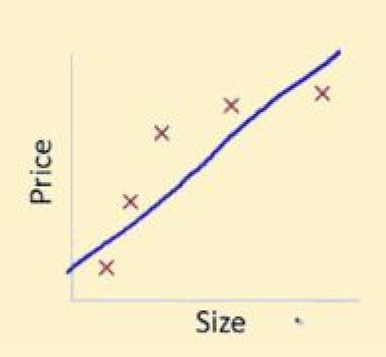
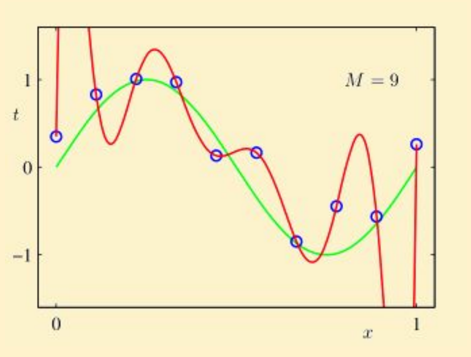
- $\hat{y}$ is the predicted classification.
- K is the number of classes.
- $\hat{P}(j|x)$ is the posterior probability of class j for observation x.
- $C(y|j)$ is the cost of classifying an observation as y when its true class is j.

Step 3: Calculate new prediction point using the formula kNN simply (Xn + Xn + 1 …. Xn) / number of neighbors for each x and y's point.

4) In 4-6 sentences, describe
• what the terms "under-fitting" and "over-fitting" refer to,
• why both phenomena are undesirable, and
• how we can identify when they happen. You may use drawings to help illustrate your point

The first under-fitting is that the model doesn't get the information incorporated well and doesn't fit the information well.

Generally talking, over-fitting infers that the model takes in the information too carefully, so the highlights of the commotion information are also realized, which prompts the inability to perceive the information well in the later test, that is, it can't be arranged precisely. The speculation capacity of the model is exorbitantly poor.

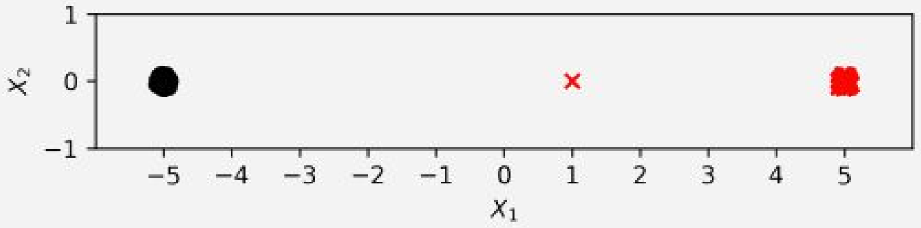
 This would be an example of under fitting. You can see a linear or even a pattern of no change with the graph above coming nowhere near the targeted data for the x's.	 This is what over fitting looks like. You can see each without any variable change that the graph line intersects with each point without any changes or variability in data.

5) In 3-5 sentences, describe the goal of PCA, including how it differs from linear regression.

PCA is an unaided technique (just takes in information, no reliant factors) and Linear regression (when all is said in done) is a regulated learning strategy(supervised). On the off chance that you have a dependent variable, a supervised method would be fit to your objectives. The goal of PCA is to discover basic factors, the principal components, in the form of linear combinations of the variables under investigation, and to rank them as indicated by their significance.

6)

Below is a scatter plot of a training data set.



- Class $Y = -1$: there are 1000 samples, plotted as black circles. They follow a multivariate normal distribution

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -5 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}\right)$$

  For the purposes of this problem you can treat all of the 1000 samples in class -1 as having $X_1 = -5$ and $X_2 = 0$.

- Class $Y = +1$: there are 1000 samples, plotted as red 'x's. They follow a multivariate normal distribution

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}\right)$$

  For the purposes of this problem you can treat 999 samples in class $+1$ as having $X_1 = 5$ and $X_2 = 0$ and there is one (outlier) sample at $X_1 = 1$ and $X_2 = 0$.

Solve the following questions by inspection (e.g. no need to do calculations or solve algebraic formulas; you can round values to the nearest integer).

A. **If we fit an LDA classifier on this data, where would the boundary be? Briefly explain why.**

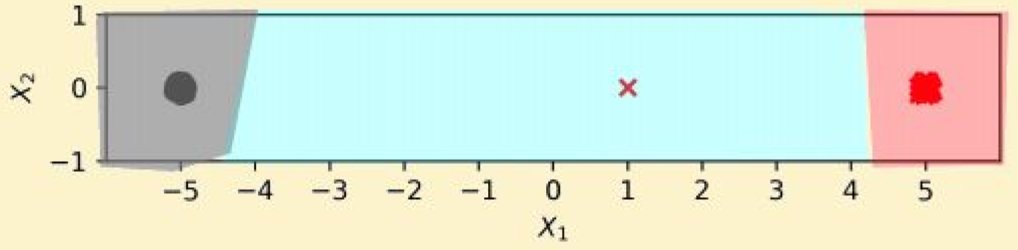
B. **If we fit a QDA classifier on this data, where would the boundary be? Briefly explain why.**

C. **If we fit a max margin classifier (linear SVM with no mistakes allowed), where**

would the boundary be? Where would the margins be? Briefly explain why.

D. Consider fitting a linear SVM classifier where we have a budget C for mistakes, Y (i) [b + w1X1(i) + w2X2(i)] ≥ 1 − ξi for all i ∈ {1, 2, . . . , 2000} ξi ≥ 0 for all i ∈ {1, 2, . . . , 2000} C = X 2000 i=1 ξi Describe what happens to the boundary and the margins as we increase C beginning at 0.

A) LDA's assumption that the predictor variable shares a common variance throughout every Y response classification is badly off, then LDA can go through from high bias. Roughly speaking, LDA tends to be a higher wager than QDA. This means we can see similar theory from other homeworks in this, if I had to draw the boundaries there would be strict vertical boundaries around the -5 and -4 for X1 going up from decimal changes around the X2. Making the boundaries strict and tight. Same goes for the opposite side for the red points.

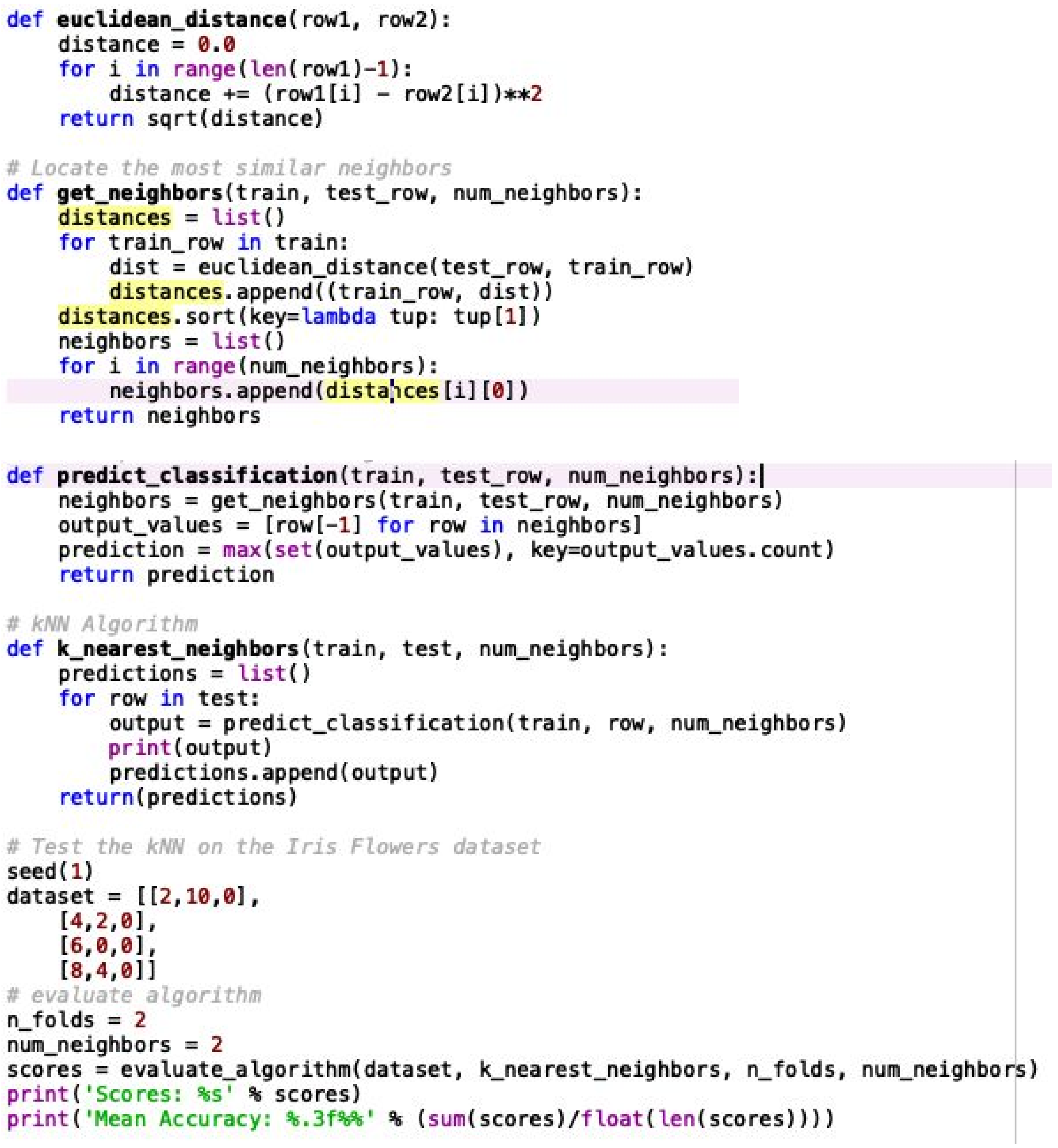


B) If there are tremendously few coaching observations and so reducing variance is crucial. In contrast, QDA is advocated if the training set is very large, so that the variance of the classifier is now not a predominant concern, or if the assumption of a frequent covariance matrix is in reality untenable. As we look toward the boundaries we can see the similar to the ones found in homework three, where they resemble more loose and round edges pulling in tighter to the more populated. Instead of tight linear borders we find that we have rounded borders depending on where the data ends up in this case similar to LDA.

C) Looking at the previous data generated along with the boundaries and graphics generated. I think it's clear to see in some instances that SVM performed a little better in some areas than QDA. It suggests that your information set is no longer without problems separable the use of the decision planes that you have let SVM use; i.e. the simple SVM uses linear hyperplanes to separate classes, and if you provide a exceptional kernel, then that will alternate the form of the choice manifold that can be used. two QDA can generate a quite convoluted decision boundary as it is driven by using the uncooked education statistics itself. For example, suppose how a Voronoi format can separate a couple of areas with a non-convex boundary made up of piecewise linear hyperplanes; LDA in the end behaves in a comparable way. SVM makes use of an enormously restricted parametric approximation of the decision boundary, which is an outstanding trade-off for classification performance towards statistics storage space/ processing speed.

D) For huge estimations of C, the punishment for misclassifying focuses is high, so the choice limit will consummately isolate the information if conceivable. See underneath for the limit got the hang of utilizing SVM and C = 2000. The classifier can expand the edge between a large portion of the focuses, while misclassifying a couple of focuses, in light of the fact that the punishment is so low. See underneath for the limit learned by SVM with C = 0.001. Since C is huge, including a point that would be mistakenly ordered by the first limit will constrain the limit to move.

Code used for Problem 3:



```
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)-1):
        distance += (row1[i] - row2[i])**2
    return sqrt(distance)

# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors

def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction

# kNN Algorithm
def k_nearest_neighbors(train, test, num_neighbors):
    predictions = list()
    for row in test:
        output = predict_classification(train, row, num_neighbors)
        print(output)
        predictions.append(output)
    return(predictions)

# Test the kNN on the Iris Flowers dataset
seed(1)
dataset = [[2,10,0],
    [4,2,0],
    [6,0,0],
    [8,4,0]]
# evaluate algorithm
n_folds = 2
num_neighbors = 2
scores = evaluate_algorithm(dataset, k_nearest_neighbors, n_folds, num_neighbors)
print('Scores: %s' % scores)
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```