**1.** Naive Bayes Classification (4 ქულა) In this problem, we will use a Naive Bayes classifier to label fragments of the genome based on sequence properties.

(a) Suppose we want to classify sequence fragments into categories (represented by random variable Y): genes, regulatory motifs, or repetitive elements. We want to use the following features: length X1, GC content (proportion of bases which are G or C) X2, and complexity X3 (intuitively, what fraction of possible k-mers are observed). Does the naive Bayes assumption hold in this setting? Explain why or why not

Naive Bayes assumption გულისხმობს, რომ feature-ები უნდა იყონ conditionally independent. ჩვენს შემთხვევაში ეს პირობა არ სრულდება, რადგან მაგალითად X3 დამოკიდებულია X1-ზე. უფრო გრძელ მიმდევრობებში შესაძლოა GC კონტენტიც განსხვავებული პროპორციით შეგვხვდეს და ა.შ. შერჩეული 3 მახასიათებელი ერთმანეთზე გავლენას ახდენს.

(b) Regardless of whether the naive Bayes assumption holds, we can still build a classifier. (Surprisingly, naive Bayes

classifiers perform well in many applications where this assumption does not hold.) To simplify, we will discretize

each of the features.

Given the training set below, write down the maximum likelihood estimates (recall these are relative frequencies)

of each of the conditional probability distributions P(Xi | Y) and the prior probability distributionP(Y).

მოცემული ცხრილის მიხედვით Prior Probability Distribution P(Y) არის:

P(Y = Gene) = 3/10. P(Y = Repeat) = 3/10. P(Y = Motif) = 4/10.

ხოლო პირობითი ალბათობები:

P(X1=Low | Y = Gene) = 2/3 P(X2=Short | Y = Gene) = 0

P(X1=Medium | Y = Gene) = 1/3 P(X2=Long | Y = Gene) = 1

P(X1=High | Y = Gene) = 0 P(X2=Short | Y = Repeat) = 0

P(X1=Low | Y = Repeat) = 0 P(X2=Long| Y = Repeat) = 1

P(X1=Medium | Y = Repeat) = 0 P(X2=Short | Y = Motif) = 1

P(X1=High | Y = Repeat) = 1 P(X2=Long | Y = Motif) = 0

P(X1=Low | Y = Motif) = 0 P(X3=Low| Y = Gene) = 1/3

P(X1=Medium | Y = Motif) = ½ P(X3=High| Y = Gene) = 2/3

P(X1=High | Y = Motif) = ½ P(X3=Low| Y = Repeat) = 2/3

P(X3=High| Y = Repeat) = 1/3

P(X3=Low| Y = Motif) = 1/4

P(X3=High| Y = Motif) = 3/4

(c) Given the model, compute the maximum a posteriori estimate of the class of the new observation below. (Hint: Is it necessary to compute the denominator in Bayes Theorem?)

P(Y | X) = (P(X | Y) \* P(Y)) / P(X)

ამ ფორმულაში უნდა ვნახოთ, Y-ის რომელი მნიშვნელობა გვაძლევს ყველაზე მაღალ მნიშვნელობას. რადგან X სამივე კლასისთვის (Gene, Motif, Repeat) საერთოა, იგი არანაირ განსხვავებას არ მოგვცემს (სამივე შემთხვევაში მნიშვნელი იგივე იქნება), ამიტომ შეგვიძლია არც გამოვთვალოთ. თავად X კი პირობის თანახმად არის: X1=Medium, X2=Long, X3=Low.

P(Y = Gene | X1 = Medium, X2 = Long, X3 = Low) = P(X1=Medium, X2 = Long, X3 = Low | Y = Gene) \* P( Y = Gene) = P(X1 = Medium | Y = Gene) \* P(X2 = Long | Y = Gene) \* P(X3 = Low | Y = Gene) \* P(Y = Gene) =

1/3 \* 1 \* 1/3 \* 3/10 = 1/30

P(Y = Repeat | X1 = Medium, X2 = Long, X3 = Low) = P(X1=Medium, X2 = Long, X3 = Low | Y = Repeat) \* P( Y = Repeat) = P(X1 = Medium | Y = Repeat) \* P(X2 = Long | Y = Repeat) \* P(X3 = Low | Y = Repeat) \* P(Y = Repeat) = 0

P(Y = Motif| X1 = Medium, X2 = Long, X3 = Low) = P(X1=Medium, X2 = Long, X3 = Low | Y = Motif) \* P( Y = Motif) = P(X1 = Medium | Y = Motif) \* P(X2 = Long | Y = Motif) \* P(X3 = Low | Y = Motif) \* P(Y = Motif) = ½ \* 0 = 0

ყველაზე მეტი ალბათობა აქვს Y = Gene შემთხვევას.

**3.** (c) Now run your code on tissue2 (your algorithm should converge in six steps this time). What went wrong? What

strategy would you employ to find the settings of the algorithm so that it identifies the most obvious clusters,

assuming you couldn’t see the clusters ahead of time?

Experiment with the code in main() to try different approaches. Use insights from your strategy to make

corresponding changes to the main() function in kmeans.py (you should only have to tinker with one line). Run the

algorithm again, and describe how your solution addressed the problem using some of the output plots for

reference.

ამ შემთხვევაში tissue2\_plots ფოტოებიდან გამომდინარე პრობლემად შეგვიძლია ჩავთვალოთ მესამე კლასტერი, რომელიც გრაფიკზე მწვანე წერტილების სახითაა მოცემული. რეალურად ამ წერტილების გადანაწილება ორ კლასტერში უფრო ლოგიკური იქნებოდა, ვიდრე ერთში. მაგრამ არსებული კოდი სამ კლასტერში ცდილობს წერტილების გადანაწილებას, რადგან ინიციალიზაციის დროს 3 წერტილს ირჩევს ცენტრად, შემდეგ კი ცენტრების რაოდენობა არ იცვლება.

კონკრეტულად ამ მონაცემებისთვის პრობლემის ყველაზე მარტივი და პრიმიტიული გადაჭრა იქნება თუ ცენტრების ინიციალიზაციისას main ფუნქციაში მეოთხე წერტილსაც ავირჩევთ, რაც საბოლოოდ ოთხ კლასტერად გადანაწილებას გამოიწვევს.

თუმცა ამ შემთხვევაში წინასწარ ავირჩიეთ კლასტერების რაოდენობა, თუ ამის გაკეთება არ შეგვეძლებოდა, მაშინ შეგვეძლო რამდენიმე განსხვავებული ვარიანტი მოგვესინჯა და მათგან საუკეთესო, მაგალითად ყველაზე ნაკლები დისტანციის (stopDist) მქონე ვარიანტი დაგვეტოვებინა.