1.

a) naive bayes-ის დაშვება არის რომ დამოუკიდებელი იქნება კლასების მიხედვით feature-ბი.

მაგრამ ამ შემთხვევაში ეს არასწორი იქნება. რადგანაც X1 არის სიგრძე, ხოლო X3 განმარტებაში შეიცავს შესაძლო k-merების რაოდენობას. შესაძლო k-merების რაოდენობა კიდე პირდაპირაა დამოკიდებული სიგრძეზე. ანუ დამოუკიდებლობა დაირღვა.

b) ჯერ დავითვალოთ P(Y)-ები:

P(Gene) = 3/10;

P(Motif) = 4/10;

P(Repeat) = 3/10;

ახლა დავითვალოთ P(Xi|Y)-ები:

P(GC=Low|Gene) = 2/3

P(GC=Medium|Gene) = 1/3

P(GC=High|Gene) = 0/3

P(GC=Low|Motif) = 0/4

P(GC=Medium|Motif) = 2/4

P(GC=High|Motif) = 2/4

P(GC=Low|Repeat) = 0/3

P(GC=Medium|Repeat) = 0/3

P(GC=High|Repeat) = 3/3

--

P(Length=Short|Gene) = 0/3

P(Length=Long|Gene) = 3/3

P(Length=Long|Motif) = 0/4

P(Length=Short|Motif) = 4/4

P(Length=Short|Repeat) = 0/3

P(Length=Long|Repeat) = 3/3

--

P(Complexity=Low|Gene) = 1/3

P(Complexity=High|Gene) = 2/3

P(Complexity=Low|Motif) = 1/4

P(Complexity=High|Motif) = 3/4

P(Complexity=Low|Repeat) = 2/3

P(Complexity=High|Repeat) = 1/3

C)

P(GC=Medium, Length=Long, Complexity=Low | Gene) = 1/3 \* 1 \* 1/3 = 1/9

P(GC=Medium, Length=Long, Complexity=Low | Motif) = 2/4 \* 0 \* 1/4 = 0

P(GC=Medium, Length=Long, Complexity=Low | Repeat) = 0 \* 1 \* 2/3 = 0

P(X) იკვეცება მაინც კლასების შედარებისას ამიტომ არ გვაინტერესებს.

კლასების შესადარებლად შეგვიძლია შევადაროთ P(X|Y) \* P(Y) მნიშვნელობები.

ამ შემთხვევაში მხოლოდ Gene კლასს აქვს შანსი ყოფნის, რადგანაც დანარჩენი 0-ია

მაგრამ რომ ვადარებდეთ Gene კლასის შესადარებელი მნიშვნელობა 1/9 \* 3/10 = 1/30 იქნებოდა.

2.

a) A C G A C G A C T A

C A G A C G C T G A

T T C C T C T G A T

A G A T G T G A C T

A C A A C G A G T A

A A A A C G A A T A

T C A T C G A G T T

A C A T C T A A C T

ჯერ დავთვალოთ ქულები ყველა სვეტისთვის. ქულები იქნება:

1) 1 0 1 1 1 1 0 0 0 2

4ჯერ მოვიდა 0 ქულა, 5ჯერ 1 ქულა, 1ხელ 2 ქულა

P(S | N) = ((0.1)^4) \* ((0.35)^5) \* (0.25)

P(S | C) = ((0.05)^4) \* ((0.15)^5) \* (0.2)

2) 3 3 6 2 6 3 6 2 3 2

3ჯერ მოვიდა 2 ქულა, 4ჯერ 3 ქულა, 3ჯერ 6 ქულა

P(S | N) = ((0.25)^3) \* ((0.2)^4) \* ((0.1)^3)

P(S | C) = ((0.2)^3) \* ((0.3)^4) \* ((0.3)^3)

b) 1283/10000 შემთხვევაში იყო მეტი

c) 1434/10000 შემთხვევაში იყო მეტი

\*\*b და c ქვეამოცანების კოდი არის simulation.py ფაილში\*\*

(ოღონდ, რადგანაც ჩვენი შესაქმნელი იყო ფაილი პითონის 3.9 ვერსიით დავწერე კოდი, იმედია არ იქნება პრობლემა)

d)

თუ სვეტების მოსვლის ალბათობები ვიცით, ერთ-ერთი მოდელი ავიღოთ და ალბათობაში რაც უფრო დიდი სხვაობაა ამ მოდელის სასარგებლოდ ისეთ სვეტს უფრო დიდი ქულა მივანიჭოთ. და ასე შევადგინოთ ქულები. ამით გავზრდით გარჩევადობას.

ან ალბათობები რომ შევცვალოთ მოდელებში, რაც უფრო დაშორებული იქნება კონკრეტული ქულის ალბათობები მით უფრო მარტივი იქნება გარჩევა და პირიქით, რაც უფრო ახლოს იქნება ალბათობები მით უფრო რთული.

i. მაგალითად ერთ-ერთ მოდელში რომ გავანულოთ რომელიმე ქულის მოსვლის ალბათობა და მეორეში ერთთან ახლოს გავხადოთ ძალიან მარტივი იქნება გარჩევა.

ii. 0.25 ები რომ იყოს ყველაფერი ფაქტობრივად შეუძლებელი იქნება გარჩევა.

ან იგივე ქულებს იგივე განაწილებები რომ ჰქონდეთ მაშინაც შეუძლებელი იქნება გარჩევა.

0 ქულიანი სვეტის გადაგდებით ცალსახად ვერ ვიტყვით შემცირდება თუ არა ერორი, ზოგ შემთხვევაში კი, ზოგ შემთხვევაში არა. შეიძლება 0 ქულიანი სვეტები გვეხმარებოდა გარჩევაში.

e) გრძელ მიმდევრობებზე ისედაც დაბალი უნდა იყოს ცდომილება. არ მაფიქრდება როგორ შეიძლება გაუმჯობესება. ლოგარითმის მოდება იგულისხმება?

3.

a) კოდი არის kmeans.py-ში

b) სურათები დავტოვე plot-ების ფოლდერში.

c) პრობლემა ისაა, რომ მეოთხე კლასტერი ქვედა მარჯვენა კუთხეში კარგად არაა გამოყოფილი

გასაუმჯობესებლად თავიდან ცენტროიდების ინციალიზაციისას მეოთხე ცენტროიდი ჩავამატე [5,60]-ზე.

newCtrs = [[5, 0], [5, 40], [5, 80], [5, 60]]

მაგრამ წინასწარ რომ არ მცოდნოდა რომ 4 კლასტერი იქნებოდა ოპტიმალური კლასტერების რაოდენობას დინამიურად ავარჩევდი და ამით გავაუმჯობესებდი, მაგალითად elbow მეთოდით.

ასევე ცენტროიდების ბევრი რენდომ ინიციალიზაციას და მერე საუკეთესოს არჩევაც გააუმჯობესებდა საქმეს. მაგრამ 4 კლასტერს ვერ მოგვცემდა.