ივანე ჯავახიშვილის სახელობის თბილისის
სახელმწიფო უნივერსიტეტი

სტუდენტი ცოტნე ჩიქოვანი

მანქანური სწავლება რთული ობიექტების კლასიფიკაციაში

სამაგისტრო პროგრამა: ინფორმაციული ტექნოლოგიები

ნაშრომი შესრულებულია ინფორმაციული ტექნოლოგიების
მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: დავით ხაჩიძე

ფიზ მათ. მეცნიერებათა კანდიდატი

თბილისი

2017

ანოტაცია

კომპიუტერული ტექნოლოგიების განთვითარებასთან ერთად, მანქანური სწავლება უფრო და უფრო პოპულარული ხდება. იხილეთ მანქანური სწავლების გამოყენების რამდენიმე მაგალითი რეალურ ამოცანებში:

- თვით-მართვადი, ჭკვიანი მანქანების წარმოება
- თახლითობის დადგენა (ერთ-ერთი ყველაზე მნიშვნელოვანი გამოყენება)
- ტექსტის კლასიფიკაცია, ობიექტების ამოცნობა

ამ ნაშრომში განვიხილავით მანქანურ სწავლებას და ღრმა ნეირონულ ქსელს კლასიფიკაციის ამოცანის გადაწყვეტის მხრივ.

ეს ნაშრომი წყვეტს კონკრეტულ ამოცანას. დავალება არის კალორიმეტრით ჩატარებული სისხლის ანალიზით მიღებული და დაგროვებული მონაცემებით ვასწავლოთ კომპიუტერს. შედეგად, კომპიუტერს უნდა შეეძლოს ახალი ანალიზის საფუძველზე გაარჩიოს ადამიანს აქვს თუ არა სიმსივნით დაავადებული უჯრედები.

Abstract

Because of new computing technologies, machine learning is becoming more popular and widely used. Here are a few widely publicized examples of machine learning applications you may be familiar with:

- The heavily hyped, self-driving Google car? The essence of machine learning.
- Fraud detection? One of the more obvious, important uses in our world today.
- Text classification, object identifier and data classification problems

In this work we discuss using of machine learning and deep neural networks for solving classification problems.

This work solves the specific task. The task is to teach computer with big data received from blood examination with calorimeter and at the end computer should be able to determine either human has a cancer or not.

სარჩევი

შესავალი

მანქანური სწავლება დღესდეობით ძალიან პოპულარული სიტყვაა ტექნოლოგიურ სამყაროში. ის არის მთავარი წინგადადგმული ნაბიჯი იმის დასადგენად, თუ როგორ უნდა ისწავლოს კომპიუტერმა. მარტივად რომ ვთქვათ, კომპიუტერი სწავლობს დიდ მონაცემებზე და შემდეგ ამ ცოდნით ცდილობს სწორად გასცეს პასუხი დასმულ კითხვებს, მაგ: ჩვენ შეგვიძლია კომპიუტერს ვასწავლოთ დიდი რაოდენობის ფოტოებით კატის გამოსახულება და შემდეგ ის ავტომატურად მოახდენს კატის იდენტიფიცირებას სხვა ფოტოებზე. მანქანური სწავლება გამოიყენება ისეთ მიზნებისთვის როგორებიცაა :

- მონაცემთა უსაფრთხოება
- პერსონალური უსაფრთხოება
- ფინანსური მიზნები
- ჯამრთელობა
- მარკეტინგი
- თაღლითობის გამოაშკარავება
- რეკომენდაციების შეთავაზება
- ონლაინ ძებნები
- ენის დამუშავება, ტექსტის შენაარშის გაგება და მათი დახარისხება
- ჭკვიანი, თვით-მართვადი მანქანების წარმოება

მანქანური სწავლების სისტემები იყოფა ორ მთავარ ქვეკატეგორიად:

- რეგრესიული მანქანური სწავლების სისტემები (ესეთი სისტემები პასუხობენ ისეთ კითვებს როგორიცაა, რამდენად? რამდენი?)
- კლასიფიკაციის მანქანური სწავლების სისტემები (ესეთი სისტემები პასუხობენ კითხვას კი თუ არა?, ანუ პასუხი ყოველთვის არის ბულის ტიპის, ან ჭეშმარიტი ან უარყოფიტი)

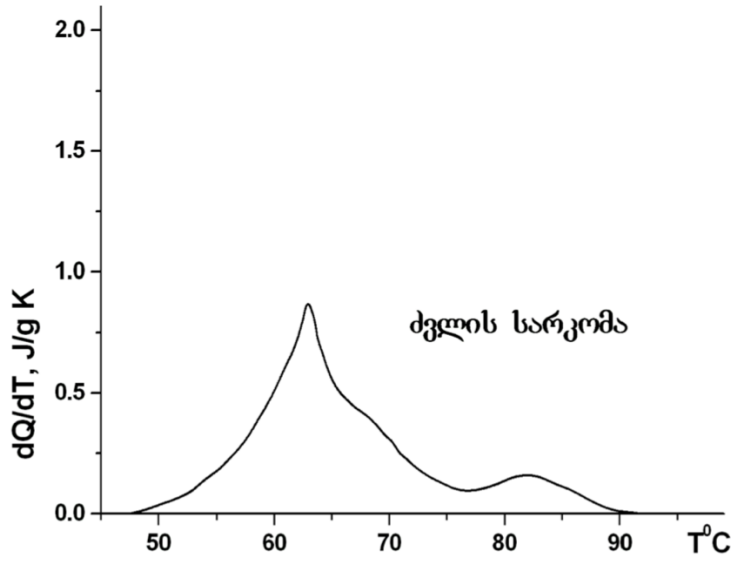
ჩვენ განვიხილავთ კლასიფიკაციის ტიპის სისტემას, რომელიც იქნება მორგებული კონკრეტულ ამოცანაზე და გასცემს პასუხს კითხვაზე - აქვს თუ არა სიმსივნე პაციენტს?

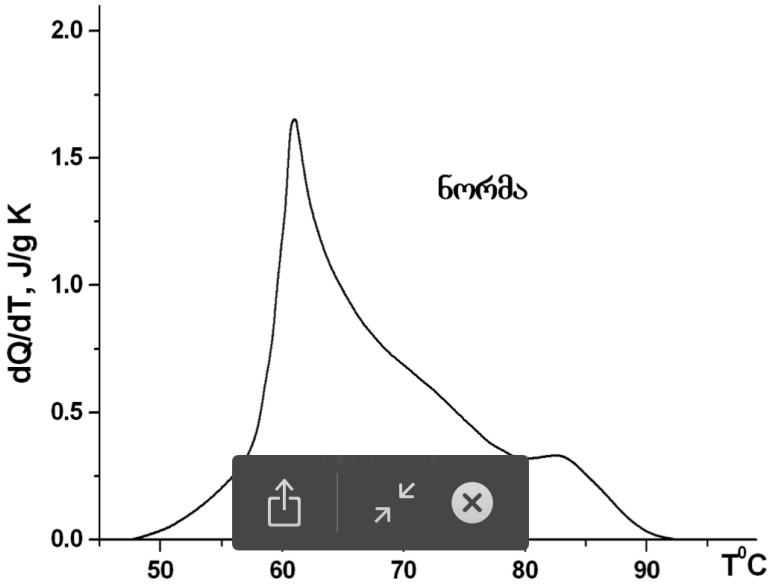
ჩვენი ამოცანა მდგომარეობს შემდეგში: გვაქვს წლების განმავლობაში დაგროვებული პაციენტთა სისხლის ანალიზები, რომლებიც მიიღება სპეციალური ხელსაწყოს

გამოყენებით, რომელიც იყენებს კალორიმეტრს, და ასევე ვიცით ამ პაციენტების გამოკვლევების შედეგად საბოლოო დიაგნოზი სიმსივნეზე. ჩვენი მიზანია სისხლის ანალიზის და საბოლოო დიაგნოზის წყვილები ვასწავლოთ კომპიუტერს და ამის საფუძველზე მომავალში მოვახდინოთ სისხლის ანალიზის საფუძველზე პაციენტის ჯამრთელობის დადგენა.

პაციენტისთვის ჩატარებული ანალიზი იძლევა პასუხს, რომელიც წარმოდგენილია მათემატიკური გრაფიკის სახით.

სურათზე მოცემულია ჯამრთელი და არაჯამრთელი პაციენტის ანალიზები





ამ ნაშრომში ასევე არის განხორციელებული ამოცანის მათემატიკური გზით გადაწყვეტა და წარმოდგენილია მიღებული შედეგების შედარება მანქანური სწავლებით მიღებულ შედეგებთან.

წლების განმავლობაში დაკვირვების შედეგად ჩამოყალიბდა კრიტერიუმები, რომლებიც გვეხმარება პაციენტისთვის დიაგნოზის დასმაში. ჩვენ გამოვიყენებთ ინტერპოლაციის ერთ-ერთ სახეობას - კუბურ სპლაინ ინტერპოლაციას მათემატიკური მოდელის შესაქმნელად. შემდეგ მოვახდენთ კრიტერიუმებით შეფასებას და საბოლოო დიაგნოზის დასმას.

ამის შემდეგ მოვახდენთ მანქანური სწავლებით მიღებული შედეგების შედარებას მათემათიკური გზით მიღებულ შედეგებთან და გამოვიტანთ დასკვნას.

1. მანქანური სწავლება

1.1. ზოგადი მიმოხილვა

მოკლედ, რა არის მანქანური სწავლება? ტომ მიტჩელის განმარტებით "კომპიუტერული პროგრამას შეუძლია სწავლა E გამოცდილებიდან რაღაც T დავალების შესასრულებლად რაღაც შრომისუნარიანობის საზომით P, თუ P შრომისუნარიანობა T დავალებისთვის ახდენს E გამოცდილებას გაუმჯობესებას" მაგალითად : თუ გინდათ ივარაუდოთ ტრანსპორტის გზებზე შესაძლო გადატვირთვის წერტილები, შეგიძლიათ პროგრამას ასწავლოთ წარსულში დაგროვებული მონაცემებით და თუ მან წარმატებით ისწავლა, მაშინ ის იწინასწარმეტყველებს მომავალ გადატვირთვის წერტილებს.

მანქანური სწავლება თავისი ბუნებით იყოფა ორ ნაწილად :

- კონტროლირებადი მანქანური სწავლება (Supervised ML) (პროგრამა სწავლობს უკვე მოცემული დიდი რაოდენობის სასწავლის მასალით და შემდეგ იძლება კორექტულ პასუხს ახალი მონაცემებისთვის)
- არაკონტროლირებადი მანქანური სწავლება (Unsupervised ML) (პროგრამას ეძლევა დიდი რაოდენობით მონაცემი და მან უნდა იპოვოს პატერნი თუ როგორ უნდა ისწავლოს და შესაბამისობა მათ შორის)

ჩვენ განვიხილავთ მხოლოდ კონტროლირებად მანქანურ სწავლებას

1.2. კონტროლირებადი მანქანური სწავლება (Supervised ML)

კონტროლირებადი მანქანური სწავლების მთავარი მიზანი არის შეიქმნას საბოლოო ფუნქცია h(x), რომელიც კორექტულად იწინასწარმეტყველებს h(x) ჩვენთვის სასურველი x მონაცემისთვის. პრაქტიკაში x ყოველთვის აღწერ კომპლექსურ მონაცემებს. მაგ: თუ გვინდა გავაკეთოთ სახლის ფასის პროგნოზი, ამ დროს ჩვენი შემავალი მონაცემები იქნება : ფართობი, ოთახების რაოდენობა, საძინებლების რაოდენობა, აბაზანების რაოდენობა, სართულების რაოდენობა და ა.შ.

ვთქვათ ჩვენი საბოლოო ფუნქცია არის :

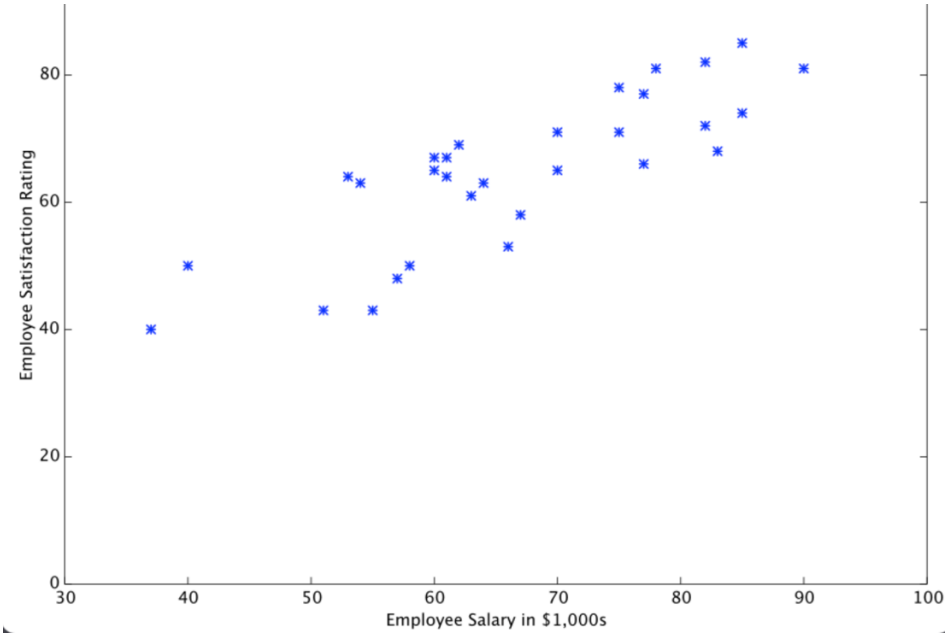
$$h(x) = \theta_0 + \theta_1 x$$

სადაც θ_0 და θ_1 არიან კონსტანტები. ჩვენი მიზანია ვიპოვოთ საუკეთესო წყვილი, რომ მოვახდინოთ წინასწარმეტყველება რაც შეიძლება უკეთ.

პროგნოზისტი $h(x)$ ფუნქციის ოპტიმიზაცია ხდება სასწავლი მასალის გამოყენებით. ყოველი სასწავლი მაგალითისვის, ჩვენ გვაქვს შემავალი მონაცემი x და ასევე ცნობილი შედეგი y. ყოველი მაგალითისთვის ვეძებთ განსხვავებას y-სა და $h(x)$ – ს შორის. საკმარისი სასწავლი მასალის გამოყენებით ეს გასხვავებები გვაძლევს $h(x)$ - ის მცდარობის გაზომვის საშუალებას. ამის შემდეგ ვცლით θ_0 – ს და θ_1 - ს განუწყვეტლივ, სანამ არ მივიღებთ საუკეთესო შედეგს. ამ გზით პროგნოზისტი ხდება ნასწავლი და მას უკვე შეუძლია რეალური მონაცემებისთვის წინასწარმეტყველება

1.2.1. მარტივი მაგალითი

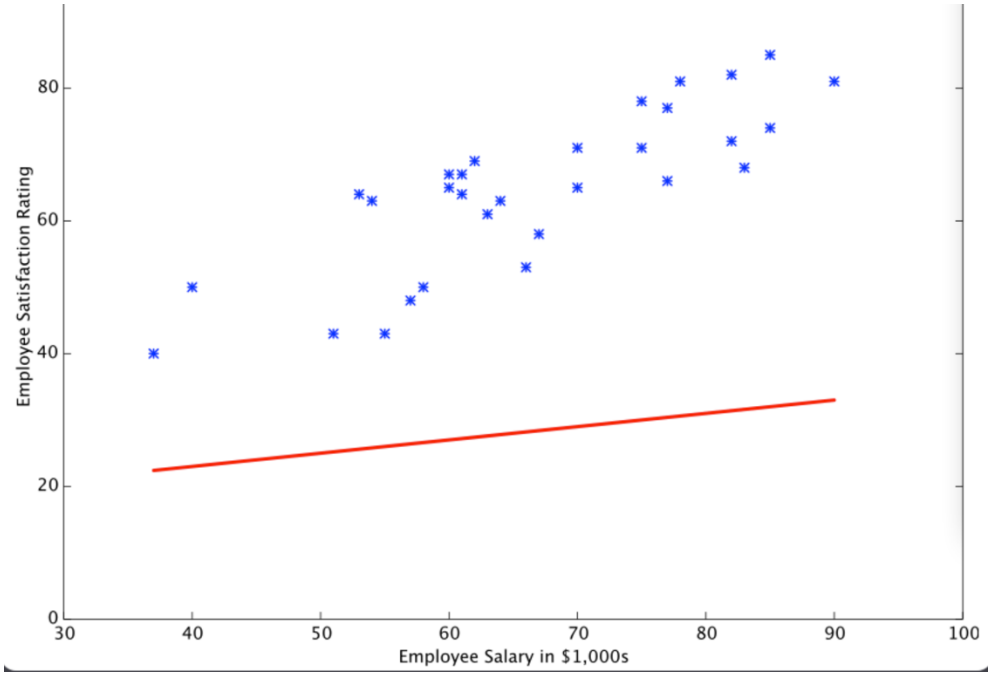
განვიხილოთ მარტივი კონტოლირებადი მანქანური სწავლები მაგალითი. ვთქვათ, მოცემული გვაქვს დასაქმებულის კმაყოფილების შეფასება და ვიცით მათი ხელფასები, რომელიც გრაფიკულად გამოიყურება შემდეგად



ჩვენი ამოცანაა მივიღოთ h(x) ფუნქციას რომელიც საუკეთესო პროგნოზს მოგვცემს ახალი მონაცემებისთვის. თავდაპირველად θ_0 – ს და θ_1 - ს უნდა შევურჩიოთ რაიმე მნიშვნელობები. ვთქვათ მივიღეთ შემდეგი ფუნქცია :

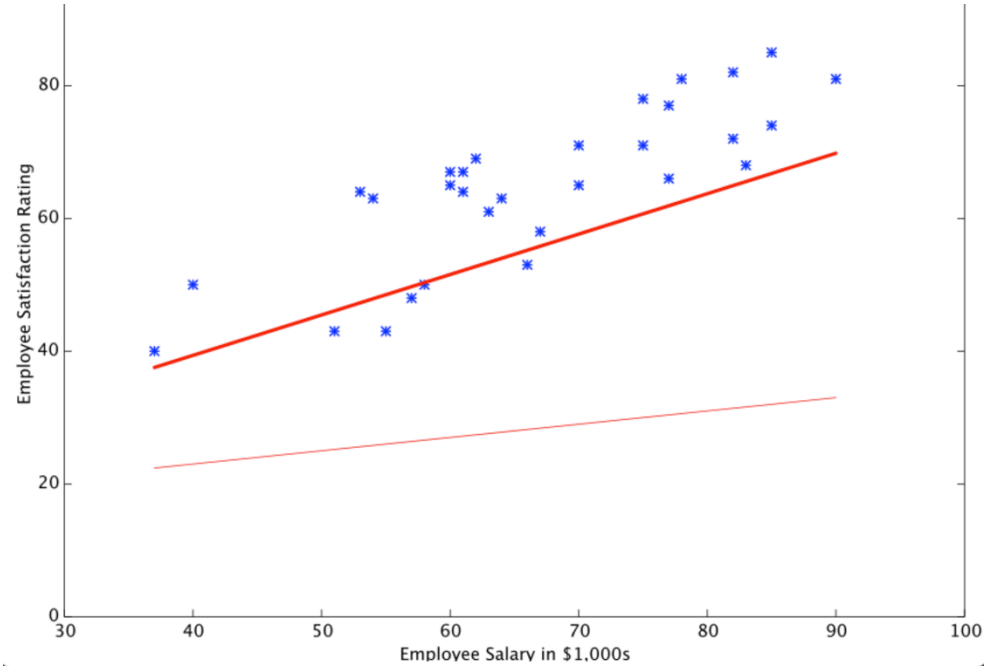
$$h(x) = 12 + 0.2x$$

მივიღებთ შემდეგ სურათს :



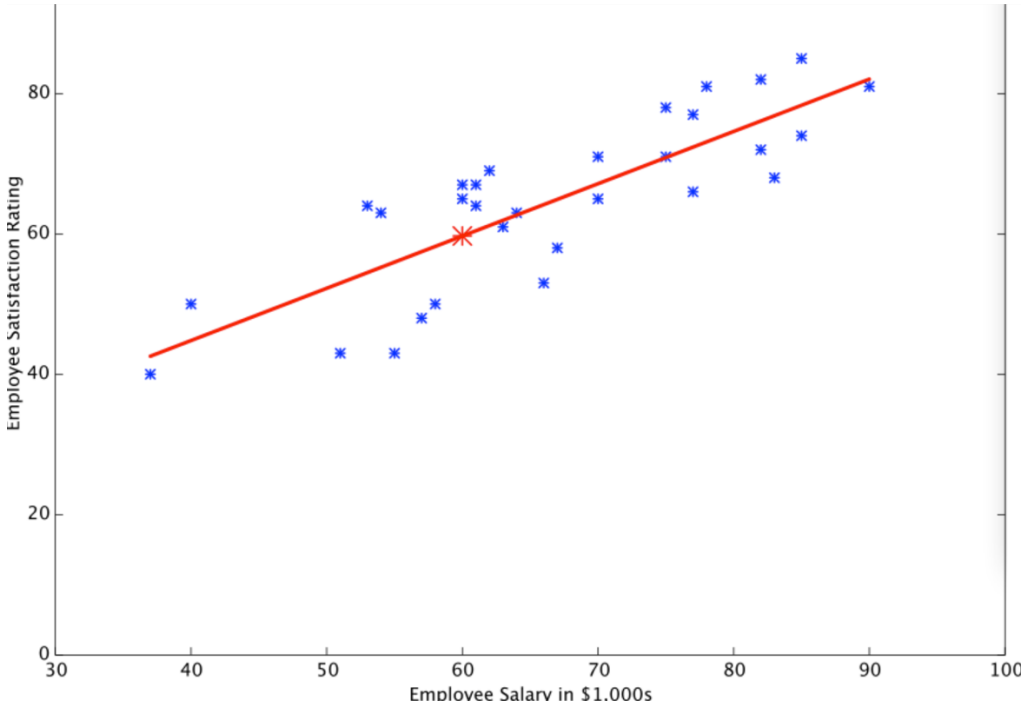
აშკარაა რომ ეს ფუნქცია მოგვცემს საშინელ პროგნოზს. ახლა ფუნქციას ვასწავლოთ განსხვავებები რეალურ და ჩვენს პროგნოზირებად მნიშვნელობებს შორის განსხავებები და ვნახოთ რას მივიღებთ. ეს ის ეტაპია, სადაც ერთვება მათემატიკა და კალკულუსი რასაც შემდეგში შევეხებით. თუ მოვახდენთ ამ გამოთვლებს, ჩვენ მივიღებთ შემდეგ შედეგს :

$$h(x) = 13.12 + 0.61x$$



როგორც შეამჩნიეთ, უკვე ბევრად უკეთესი პროგნოზის გაკეთება შეგვიძლია. თუ ამ გამოთვლებს ჩავატებთ ისევ და ისევ, თუ გავიმეორებთ 1500-ჯერ მივიღებთ შემდეგ სურათს.

$$h(x) = 15.54 + 0.75x$$



რომელიც, უკვე გვაძლევს ბევრად უკეთეს შედეგს, და რომელიც საკმარისია გავაკეთოთ უკვე კორექტული პროგნოზირება

1.2.2. მუშაობის პრინციპი

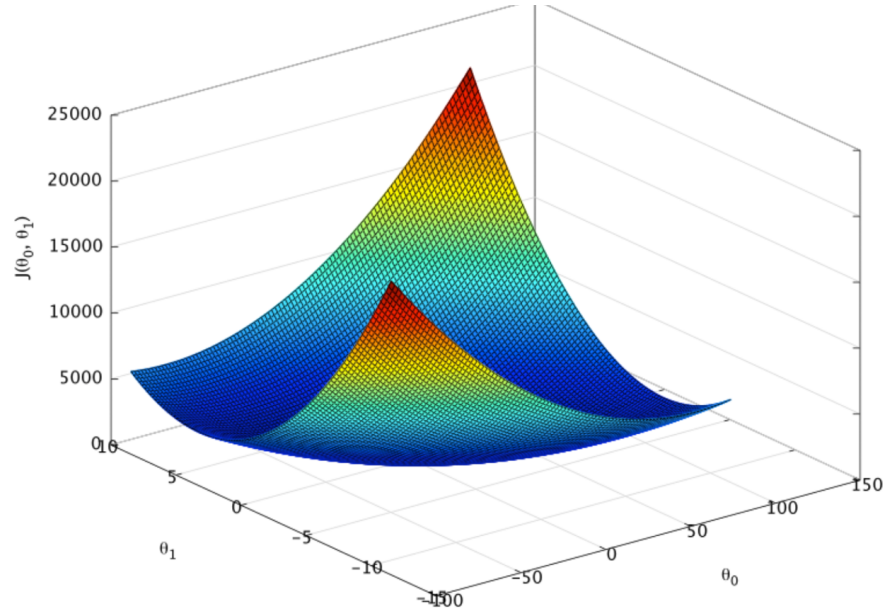
განვიხილოთ იტერაციული პროცესი, რომლითაც ხდება რეალურად მანქანური სწავლების პროცესი. ისმის კითხვა, ზემოთ მოცემულ მაგალითში როგორ ვრწმუნდებოდით რომ θ_0 და θ_1 - ს ვღებულობდით უფრო და უფრო უკეთესს ყოველი იტერაციის შემდეგ? პასუხი არის ჩვენი ცდომილების საზომი.

ცდომილების საზომი არის ცნობილი როგორც ღირებულების ფუნქცია $J(\theta)$, სადაც θ არის ამ შემთხვევაში θ_0 და θ_1 - ის კომბინაცია. J ფუნქცია გვიჩვენებს თუ რამდენად ცდება ჩვენი მიღებული ფუნქცია. ღირებულების ფუნქციის არჩევა არის ასევე მნიშვნელოვანი ნაწილი მანქანური სწავლების პროცესში.

ჩვენი ამოცანისთვის საუკეთესო ღირებულების ფუნქცია იქნება შემდეგი:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_{t,i}) - y)^2$$

რომელიც გამოისახება სხვაობების კვადრატების ჯამით. ჩვენი ღირებულების ფუნქციის გრაფიკი იქნება შემდეგი:



შესაბამისად, თუ კარგად დავაკვირდებით ჩვენი მიზანია ვიპოვოთ ისეთი θ_0 და θ_1, რომ ფუნქციის მნიშვნელობა იყოს მინიმუმი ანუ იყოს რაც შეიძლება ფსკერზე. ეს პროცესი კი მიიღწევა θ_0 – ის და θ_1 - ის მუდმივი ცვლილებით, ერთი გაზრდით ან მეორეს შემცირებით და ა.შ. პერიოდულად. გრადიენტის დათვლის პროცესი და შედეგების მიხედვით θ - ს ცვლილება ცნობილია როგორც გრადიენტული დაშვების მეთოდი.

1.2.3. გრადიენტული დაშვება

გრადიენტული დაშვების არის პირველი რიგის იტერაციული ოპტიმიზაციის ალგორითმი, რომელიც გამოიყენება ფუნქციის მინიმუმის საპოვნელად. გრადიენტული დაშვების ალგორითმი არის პოპულარული მანქანურ სწავლებაში, რადგან მანქანურ სწავლებაში მთავარი მიზანი არის სისწორის მაქსიმიზაცია, ანუ იგივე შეცდომის მინიმიზაცია. სწორედ გრადიენტული დაშვება გამოიყენება შეცდომის მინიმიზაციისთვის, რადგან თუ მოვახდენთ ღირებულების ფუნქციის მინიმიზაციას გრადიენტული დასვებით, მაშინ მივიღებთ სასურველ შედეგს.

გრადიენტული დაშვების ალგორითმი ეყრდნობა იმ ფაქტს, რომ თუ F ფუნქცია განსაზღვრული არის a - ს მიდამოში. მაშინ $F(x)$ შემცირდება უსწრაფესად თუ მიმართულება იქნება a - დან უარყოფითი გრადიენტის მიმართულებით F ფუნქციის a წერტილში. შესაბამისად მივიღებთ შემდეგს:

$$\mathbf{a^{n+1}} = \mathbf{a^n} - \gamma \nabla F(\mathbf{a^n})$$

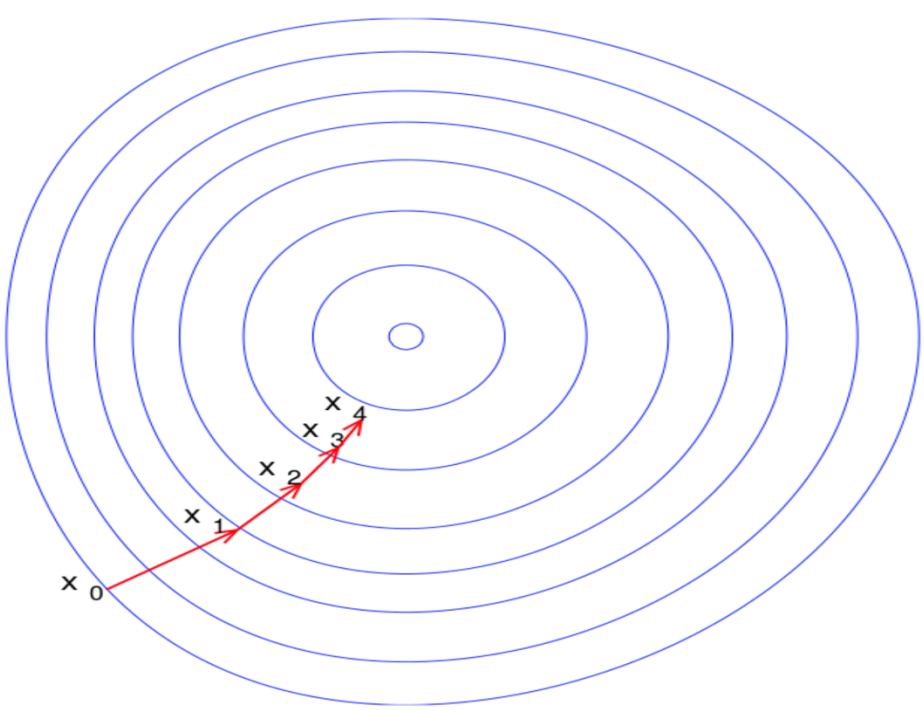
შესაბამისად ამას თუ მოვარგებთ ჩვენ შემთხვევას მივიღებთ შემდეგს:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

და აქიდან გამომდინარე :

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$

ეს სწორედ ის არის რაც გვინდოდა, ჩვენ მივაღწიეთ ჩვენი ფუნქციის მინიმიზაციას. გრადიენტული დაშვების ალგორითმის ვიზუალიზაცია შეგვიძლია მოვახდინოთ შემდეგნაირად :



1.3. ნეირონული ქსელი

მანქანური სწავლების განხილვა შეუძლებელია ნეირონული ქსელის ხსენების გარეშე. ნეირონული ქსელები გთავაზობენ არამარტო ექსტრემალურად ძლევამოსილ ხელსაწყოებს პრობლემების გადასაჭრელად, არამედ ისინი გვაძლევენ მითითებებს იმაზე, თუ როგორ მუშაობს ჩვენი ტვინი. ასევე გვიქმნიან რეალურად წარმოდგენას რომ ერთ დღესაც შეიქმნება ნამდვილი ინტელექტუალური მანქანა.

ნეირონული ქსელები კარგად არის თავსებადი მანქანურ სწავლების პრობლემებთან როცა მონაცემები არის ძალიან დიდი რაოდენობით.

მარტივად რომ ვთქვათ ნეირონული ქსელი არის შრეებისგან შემდგარი ქსელი. ყოველ შრეს გააჩნია შემავალი და გამომავალი და ყოველ შრეს გააჩნია წონები და გადაბრები. წარმოიდგინეთ რომ ყოველი შრის გავლის შემდეგ შემავალი

მნიშვნელობა იცვლება რაღაც მათემატიკური მოქმედებების შედეგად და ბოლო შრის გავლის შედეგად მივიღებთ შედეგს.

1.4. ღრმა ნეირონული ქსელი

ღრმა ნეირონული ქსელი არის მეტი არაფერი, თუ არა უბრალო ნეირონული ქსელი უამრავი შრით, ხშირად გამოიყნება ნეირონული ქსელები 3 შრით, 6 შრით. თუ ის დაახლოებით 9, 12 ან უფრო მეტი შრისგან შედგება მას უკვე ღრმა ნეირონულ ქსელს ეძახიან.

1.5. კლასიფიკაციის ამოცანა

როგორც ვახსენეთ მანქანური სწავლების სისტემები იყოფა ორ მთავარ ქვეკატეგორიად:

- რეგრესიული მანქანური სწავლების სისტემები
- კლასიფიკაციის მანქანური სწავლების სისტემები

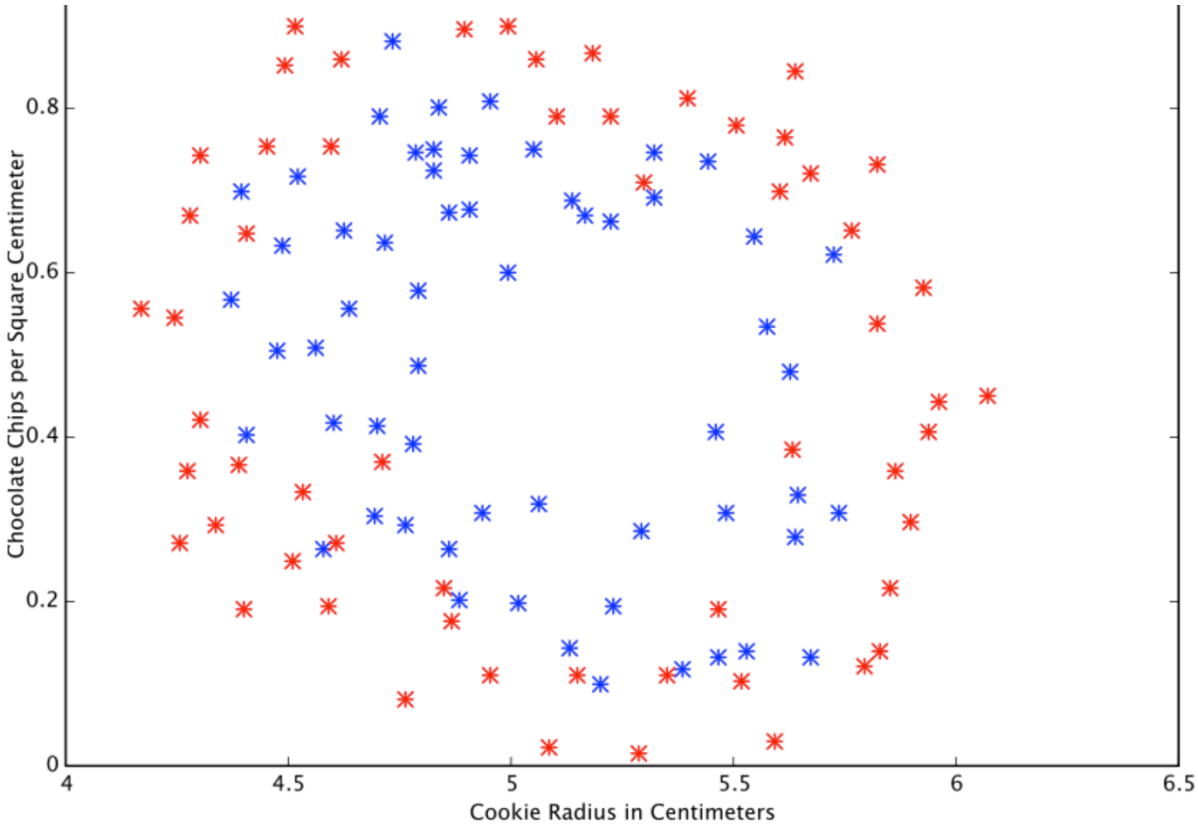
კლასიფიკაციის მანქანური სწავლების სისტემები პასუხობენ კითხვაზე კი თუ არა, ანუ მათი შედეგი არის ან 0 ან 1.

კლასიფიცირების ამოცანები ხშირად გვხვდება ყოველდღიურობაში. მანქანური სწავლების კლასიფიკაციის გამოყენება შეგვიძლია ბევრი დანიშნულებით. მათ შორისაა:

- ტექსტის კლასიფიცირება დადებით და უარყოფით შეფასებებად
- ფოტოებზე სხვადასხვა ობიექტების ამოცნობა
- თაღლითობის დადგენა
- მარკეტინგული დანიშნულებით, შეგვიძლია დავადგინოთ აინტერესებს თუ არა რაიმე პროდუქტი კონკრეტული ასაკის ჯგუფს
- ჯამრთელობის დარგში, შეგვიძლია დავადგინოთ აქვს თუ არა პაციენტს რაიმე დაავადება

და ა.შ.

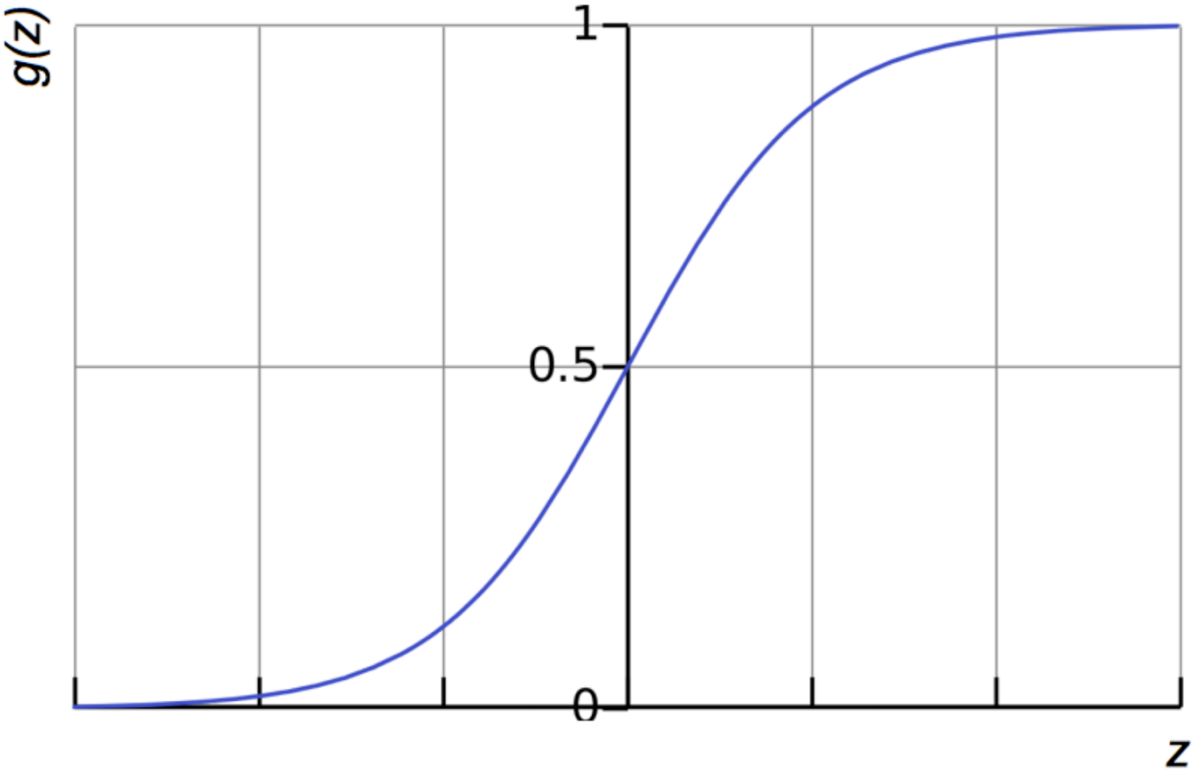
განვიხილოთ შედარებით მარტივი მაგალითი, რომლის ვიზუალიზაციაც მარტივია და შემგომ რთული ობიექტების კლასიფიკაციაში ძალიან დაგვეხმარება. გვაქვს ფუნთუშები ორი პარამეტრით, რადიუსი და შოკოლადის შემცველობა და ასევე გვაქვს ინფორმაცია არიან თუ არა ისინი ვარგისები ბაზარზე. ჩვენი ამოცანაა ვასწავლოთ ნეირონულ ქსელს ეს მონაცემები და შევქმნათ პროგრამა, რომელიც მოგვცემს პროგნოზს არის თუ არა კონკრეტული ფუნთუშა ვარგისი ბაზრისთვის. ჩვენი მონაცემები გამოიყურება შემდეგად



სადაც x ღერძი აღნიშნავს ფუნთუშის რადიუსს, y ღერძი კი შოკოლადის შემცველობას. ლურჯი ფერის ფიფქი აღნიშნავს ფუნთუშის ვარგისიანობას, წითელი კი უვარგისობას.

როგორც ვიცით, ფუნქციის შედეგი უნდა იყოს ან 1 ან 0. ამიტომ ჩვენ შეგვიძლია 0.6 აღვიქვათ როგორც 1 და 0.4 როგორც 0. ამისთვის ზედგამოჭრილი არის სიგმოიდური ფუნქცია.

$$h(x) = g(z)$$



ჩვენი ფუნქცია აქამდე გამოიყურებოდა როგორც $z = \theta_0 + \theta_1 x$

ახლა კი თუ g ფუნქციას მოვადგამთ გარედან მივიღებთ $h(x) = g(\theta_0 + \theta_1 x)$

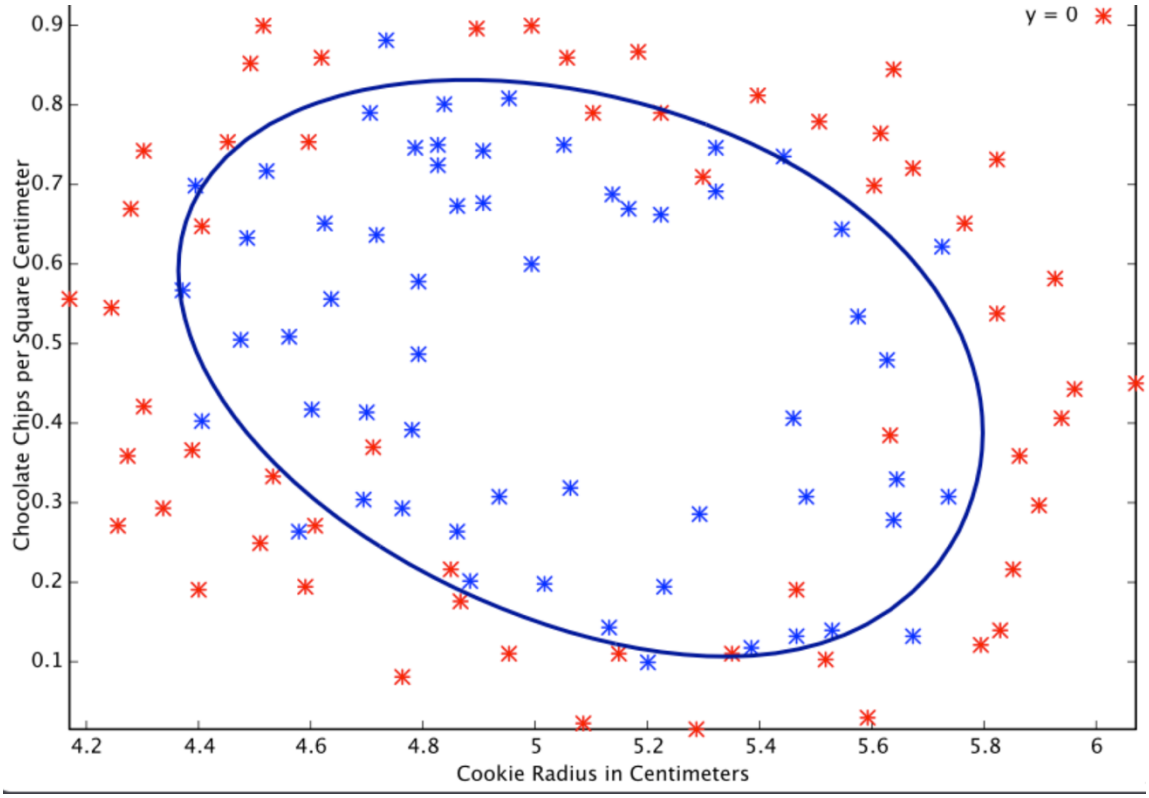
როგორც ხედავთ, სიგმოიდური ფუნქცია ჩვენს შედეგს გარდაქმნის 0-სა და 1-ის შუალედში.

ღირებულების ფუნქციაც კლასიკიციის ამოცანებში განსხვავებული. ისმევა შეკითხვა, რას ნიშნავს რომ ვარაუდი არის მცდარი, თუ სწორი პასუხი არის 0 და ჩვენ ვივარაუდეთ 1, ეს ნიშნავს რომ ჩვენ ვართ სრულიად მცდარი და პირიქით. რადგან ჩვენ არ შეგვიძლია ვიყოთ უფრო მცდარები ვიდრე სრულიად მცდარები, ამიტომ ჩვენი სასჯელი არის უზარმაზარი. ასევე თუ სწორი პასუხი არის 0 და ჩვენ ვივარაუდეთ 0, ჩვენ არ უნდა დავამატოთ არაფერი ჩვენს ღირებულების ფუნქციას არცერთ ასეთ შემთხვევებში. შესაბამისად ღირებულების ფუნქცია გამოიყურება შემდეგნაირად

$$cost = \begin{cases} -log(h(x)) & if y = 1 \\ -log(1 - h(x)) & if y = 0 \end{cases}$$

გავმეორდეთ, ღირებულების ფუნქცია ჩვენ გვაძლევს საშუალოს ცდომილებას სასწავლი მონაცემების ირგვლივ.
მოკლედ ჩვენ ვაჩვენეთ, თუ როგორ განსხვავდება პროგნოზისტი h(x) ფუნქცია და ღირებულების J(x) ფუნქცია რეგრესიასა და კლასიფიკაციას შორის, თუმცა ისიც უნდა აღვნიშნოთ რომ გრადიენტული დაშვება აქაც მშვენივრად მუშაობს.

კლასიფიკაციის პროგნოზისტი ფუნქცია შეგვიძლია გამოვსახოთ როგორც ზღვრული ხაზი, რომელიც დაახლოებით გამოიყურება ასე:



2. პროექტის აღწერა

პროგრამა არის დაწერილი პროგრამირების ენაზე python და ის შედგება ორი ნაწილისგან

- ამოცანის ამოხსნა ნეირონული ქსელის გამოყენებით
- ამოცანის ამოხსნა მათემატიკური გზით

2.1. ამოცანის ამოხსნა ნეირონული ქსელის გამოყენებით

2.1.1. პროგრამის აღწერა

პროგრამის გამოყენებისთვის ოპერაციულ სისტემაში უდნა იყოს დაინსტალირებული შემდეგი ხელსაწყოები:

- Python
- SciPy და NumPy
- Keras და TensorFlow

პროგრამის ეს ნაწილი შედგება 6 ნაწილისაგან

- მონაცემების წაკითხვა
- მოდელის განსაზღვრა
- მოდელის კომპილირება
- მოდელის შეფასება
- წინასწარმეტყველება

2.1.1.1. მონაცემების წაკითხვა

როგორც ვიცით ჩვენი ნეირონული ქსელის შემავალ პარამეტრს წარმოადგენს მათემატიკური ფუნქციის გრაფიკი, რომელიც მიიღება სისხლის სპეციალური ანალიზის შედეგად

მონაცემების წაკითხვა ხდება ტექსტური ფაილიდან, ფაილში მონაცემები უნდა იყოს ჩაწერილი შემდეგი ფორმატით

x1;y1;
x2;y2;

სადაც (x1, y1) და (x2, y2) ფუნქციის გრაფიკის წერტილებია, ასევე შესაძლებელი რამდენი გრაფიკის ერთდროულად წაკითხვა ფაილიდან

2.1.1.2. მოდელის განსაზღვრა

ნეირონული ქსელი მოდელი შედგება შრეებისგან, შემდეგი კოდის ფრაგმენტი ახდენს მოდელის განსაზღვარს.

```
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

ჩვენი ნეირონული ქსელი შედგება სამი შრისგან. პირველი და მეორე შრე არის relu ტიპის. პირველი შეიცავს 12 ნეირონს და შემავალი აქვს 8 პარამეტრი. როცა მეორე შრე შეიცავს 8 ნეირონს და საბოლოო შრე კი შეიცავს 1 ნეირონს, რათა ივარაუდოს შედეგი 0 ან 1.

relu ფუნქცია წარმოადგენს დეტექტორ ფუნქციას და განისაზღვრება შემდეგად

$f(x) = \max(0,x)$

გამოცდილება გვკარნახობს რომ კლასიფიკაციის ამოცანებში ეს ფუნქციას საკმაოდ კარგ შედეგებს იძლევა

ბოლო შრეზე ჩვენ ვიყენებთ sigmoid ფუნქციას, რათა დავრწმუნდეთ რომ ჩვენი გამომავალი იქნება 0-სა და 1-ს შუა

2.1.1.3. მოდელის კომპილირება

მოდელის კომპილირებისას ჩვენ უნდა მივუთითოთ დამატებითი პარამეტრები, რომლებიც საჭიროა ნეირონული ქსელის დასწავლისთვის. გავიხსენოთ, ნეირონული ქსელის დასწავლა ნიშნავს ვიპოვოთ საუკეთესო წონები ჩვენი შემთხვევისთვის.

ჩვენ უნდა მივუთითოთ დანაკარგის იგივე ღირებულების ფუნქცია, რომელიც ჩვენს შემთხვევაში არის ლოგარითმული დანაკარგის ფუნქცია.

მოდელის კომპილირებისთვის გამოვიყენებთ ასევე ოპტიმიზირებული გრადიენტული დაშვების ალგორითმს adam-ს. შედეგად მივიღებთ შემდეგ კოდის ფრაგმენტს

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

რის გაშვების შემდეგაც ჩაითვლება რომ მოდელი კომპირირებულია

2.1.1.4. მოდელის მორგება

ჩვენ გავამზადეთ მოდელი და შესაბამისად პროგრამა მზად არის გამოთვლების ჩასატარებლად. პროგრამის ამ ნაწილში ხდება ნეირონული ქსელის სწავლება, რომელიც მიიღწევა კოდის შემდეგი ფრაგმენტი:

```
model.fit(X, Y, epochs=150, batch_size=10)
```

ჩვენ შეგვიძლია მივუთითოთ იტერაციების რაოდენობა epochs პარამეტრის გამოყენებით. ეს არის სწორედ ის ნაწილი სადაც რეალური გამოთვლები ხდება პროცესორზე.

2.1.1.5. მოდელის შეფასება

პროგრამის ეს ნაწილი გვაძლევს წარმოდგენას თუ რამდენად კარგად მოვახდინეთ ჩვენი მოდელის დასწავლა. ამ ეტაპზე ჩვენ არ ვიცით თუ რა შედეგს მოგვცემს მოდელი რეალური მონაცემებისთვის, მაგრამ შეგვიძლია ვნახოთ თუ რა სიზუსტეს იძლევა დასწავლილ მონაცემებზე. ამას უზრუნველყოფს პროგრამის შემდეგი ნაწილი :

```
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

მოდელმა ჩვენს შემთხვევაში აჩვენა 98 პროცენტიანი სიზუსტე

2.1.1.6. წინასწარმეტყველება

ამ ნაწილში ხდება ახალი მონაცემებისთვის (რომლებიც არ მონაწილეობდნენ დასწავლაში) შედეგის წინასწარმეტყველება. ამას უზრუნველყოფს კოდის შემდეგი ფრაგმენტი :

model.predict(X)

ამ ნაწილში პროგრამამ აჩვენა 97% - იანი სიზუსტე

2.1.2. 300 ჩანაწერზე გატესტვა

დაგროვებული მონაცემების რაოდენობა იყო 300 და პროგრამის საშუალებით ნეირონულ ქსელმა ისწავლა 300 ჩანაწერზე და მან მოვცა 97 % - იანი სიზუზტე, რაც ნიშნავს რომ ის შეცდა 9 მონაცემის შემთხვევაში

2.1.3. მიღებული სიზუსტე

ტესტირება მოხდა რამდენიმეჯერ სხვადასხვა რაოდენობის მონაცემებზე და მათ აჩვენეს სხვადასხვა სიზუსტე, მონაცემების ზრდასთან ერთად სიზუსტე იზრდებოდა შედეგები შემდეგია :

მონაცემების რაოდენობა	სიზუსტე
50	69%
100	77%
150	84%
200	90%
250	94%
300	97%

ჩატარებული გათვლებით, თუ მომავალში დაემატება ასევე 300 ჩანაწერი, სიზუსტე ავა 99.5% - მდე.

2.2. ამოცანის ამოხსნა მათემატიკური გზით

2.2.1. პროგრამის აღწერა

პროგრამის გამოყენებისთვის ოპერაციულ სისტემაში უდნა იყოს დაინსტალირებული შემდეგი ხელსაწყოები:

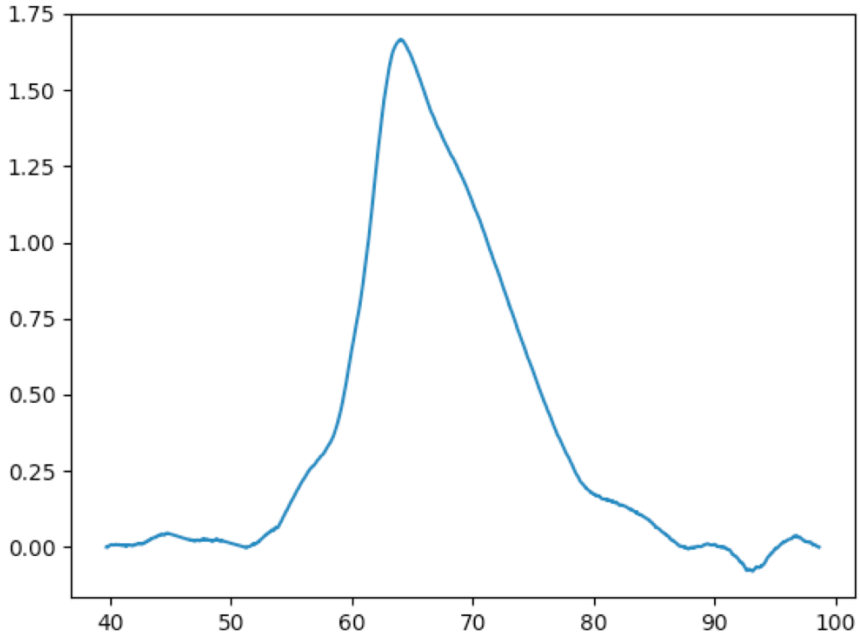
- Python
- SciPy და NumPy

პროგრამის ამ ნაწილში ხდება ამოცანის ამოხსნა პირდაპირი - მათემატიკური გზით, გრაფიკის მიხედვით ვაკეთებთ მათემატიკურ მოდელს, ვახდეთ სპლაინ ინტერპოლაციას ამ გრაფიკისთვის და შემდეგ ვიკვლევთ ამ ინტერპოლს კრიტერიუმების მიხედვით

გამოცდილებიდან გამომდინარე დადგინდა რომ პაციენტის ჯამრთელობის დადგენა დიდი სიზუსტით შესაძლებელია კრიტერიუმის შეფასებით. კრიტერიუმები არის შემდეგი:

1. მხარის არ არსებობა პიკამდე
2. მთავარი პიკის სიმაღლე დაახლოებით 1.4 და მეტი
3. ძირითადი პიკის ნახევარსიმაღლის სიგანე უნდა იყოს 7-დან 13-მდე
4. მთავარი პიკიდან მარჯვნივ მხრის ან ქვეპიკის არსებობა 81-დან 84-მდე შუალედში
5. მრუდს ქვემოთ ფართობი (ინტეგრალი) უნდა იყოს 300 და მეტი

განვიხილოთ ერთ-ერთი მაგალითი, ეს არის პაციენტის ანალიზის შესაბამისი გრაფიკი:

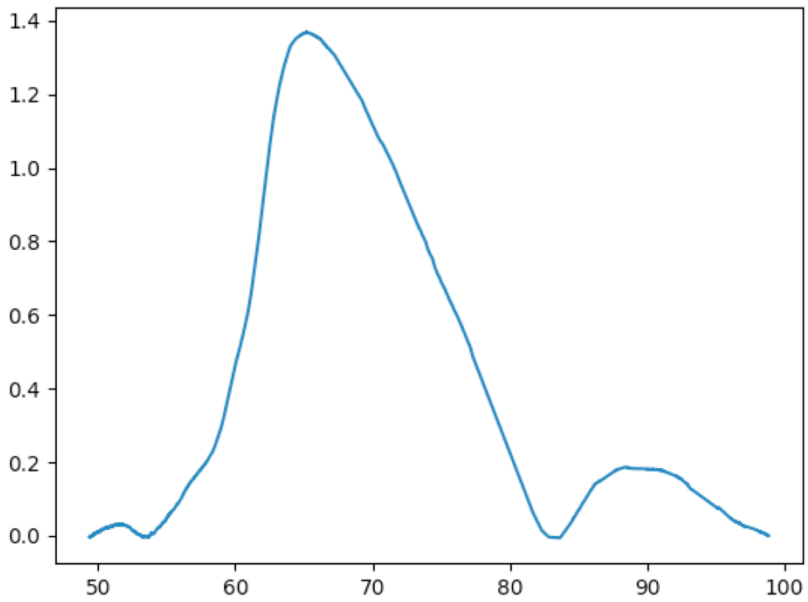


გრაფიკს აქვს 4331 წერტილი, ის აკმაყოფილებს ყველა ზემოთ აღნიშნულ კრიტერიუმს

- გრაფიკს მხარი არ აქვს მთავარ პიკამდე
- მთავარი პიკის სიმაღლე 1.4 ზე მეტია და ის 1.665 ის ტოლია
- ძირითადი პიკის ნახევარსიმაღლის სიგანე უნდა იყოს 7-დან 13-მდე და ის არის 11.9
- მთავარი პიკიდან მარჯვნივ მხარი არის 81-დან 84-მდე შუალედში
- მრუდს ქვემოთ ფართობი (ინტეგრალი) არის 300-ზე მეტი და ის არის 613.8 ის ტოლი

შესაბამისად, რადგან გრაფიკი აკმაყოფილებს ყველა პირობას, ვთვლით რომ პაციენტს არ გააჩნია სიმსივნის უჯრედები

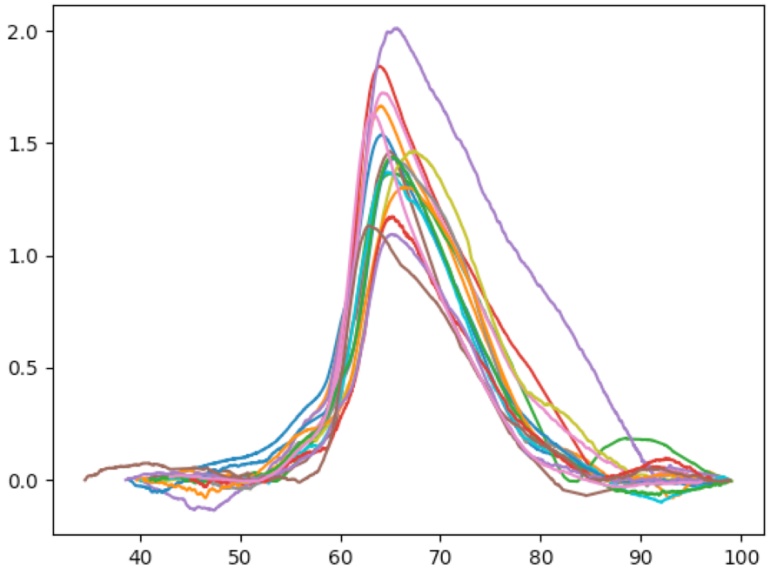
განვიხილოთ კიდევ ერთი გრაფიკი:



როგორც გრაფიკზე ჩანს მთავარი პიკის y კოორდინატი არის 1.4 ზე ნაკლები, რაც იმას ნიშნავს, რომ ირღვევა ჩვენი მეორე კრიტერიუმი. შესაბამისად გამოგვაქვს დასკვნა რომ პაციენტს აქვს სიმსივნის უჯრედები

ისიც უნდა აღვნიშნოთ, რომ პროგრამას ასევე შეუძლია რამოდენიმე ანალიზის ერთად დამუშავება. ეს მიიღწერა ამ ანალიზების ერთ ფაილში გაწერით.

შედეგად მიიღება საერთო გრაფიკი და დიაგნოზები, 16 ანალიზის ერთდროულად დამუშავების შედეგი:



2.2.2.სპლაინით ინტერპოლაცია

როგორც უკვე მიხვდით, მატემატიკური მოდელის შექმნისთვის ვიყენებთ ინტერპოლაციას, კერძოდ სპლაინით კუბურ ინტერპოლაციას.
ამისთვის ვიყენებთ SciPy და NumPy - ის მზა ფუნქციებს

ისმის კითხვა, რა არის სპლაინით ინტერპოლაცია?

სპლაინით ინტერპოლაცია არის ინტერპოლაციის ერთ-ერთი სახეობა, სადაც ინტერპოლანტი არის სპეციალური ტიპის ჰიბრიდული პოლინომი, რომელსაც ეძახიან სპლაინს. ჰიბრიდული პოლინომი ან ფუნქციას არის ისეთი ფუნქცია რომლის გასაზღვრული ქვეფუნქციებით. ასეთია მაგ:

$$|x| = \begin{cases} -x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$$

სპლაინით ინტერპოლაცია ხშირად ჯობნის პოლინომიალურ ინტერპოლაციას, რადგან ინტერპოლაციის ცდომილება შეიძლება იყოს ბევრად მცირე დაბალი ხარისხის პოლინომისთვისაც კი სპლაინისთვის. სპლაინით ინტერპოლაცია არიდებს თავს პრობლემას - რუნჯის მოვლენას, რომელიც განისაზღვრება იმით, რომ როცა ვზრდით პოლინომის ხარისხს, ყოველთვის ვერ ვღებულობთ უკეთეს შედეგს.

2.2.3. მიღებული სიზუსტე

მათემატიკურმა მოდელმა აჩვენა 100% - იანი სიზუსტე არსებულ მონაცემებზე, თუმცა არ ვიცით რამდენად კარგ შედეგს მოგცემს მომავალში, რადგან ეს კრიტერიუმები არის ადამიანური დაკვირვებით მიღებული და არ ვართ დარწმუნებულები მის სრულ სისწორეში.

2.3. მიღებული შედეგების შედარება

ჩვენ მოვახდინეთ ამოცანის ამოხსნა ორი გზით ნეირონული ქსელის გამოყენებით და მათემატიკური გზით, სიზუსტის შემოწმებამ გვაჩვენა რომ მათემატიკურმა მოდელმა არსებულ მონაცემებზე მოგვცა 100% - იანი სიზუსტე, როცა ნეირონულმა ქსელმა მოგვცა 97 % - იანი სიზუსტე

დასკვნა

მიუხედავად იმისა რომ ნეირონული ქსელისთვის დასასწავლი მონაცემების რაოდენობა არ იყო ბევრი, ქსელმა მაინც კარგი შედეგი მოგვცა და 97 % - იანი სიზუსტე აჩვენა, ასევე მათემატიკურმა მოდელმა აჩვენა 100% - იანი სიზუსტე არსებულ მონაცემებზე. (თუმცა არ ვიცით რამდენად კარგ შედეგს გვაჩვენებს მომავალში)

პროგრამა ისეა დაწერილი რომ მას შეუძლია შეისწავლოს მომავალში დაგროვებული მონაცემებიც და უფრო გაზარდოს სიზუსტე.

ასევე შესაძლებელია ამოცანის განვრცობა და უფრო საინტერესო მიზნების განხორციელება. შესაძლებელია არა მარტო სიმსივნის უჯრედების აღმოჩენისთვის გამოვიყენოთ ნეირონული ქსელი, არამედ სხვადასხვა დაავადებების გარჩევადობისთვის.

გამოყენებული ლიტერატურა

1. https://www.forbes.com/sites/bernardmarr/2016/09/30/what-are-the-top-10-use-cases-for-machine-learning-and-ai/ - 79aff9da94c9
2. https://en.wikipedia.org/wiki/Spline_interpolation
3. https://medium.com/towards-data-science/intro-to-deep-learning-d5caceedcf85
4. https://en.wikipedia.org/wiki/Spline_interpolation
5. https://en.wikipedia.org/wiki/Piecewise
6. https://en.wikipedia.org/wiki/Runge%27s_phenomenon
7. https://en.wikipedia.org/wiki/Gradient_descent