VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMOS

Skaitmeninis intelektas ir sprendimų priėmimas
Vieno neurono mokymas sprendžiant klasifikavimo uždavinį

Darbą atliko:
Pijus Zlatkus

Vilnius
2024

Turinys

1. Užduoties tikslas

Šios užduoties tikslas yra apmokyti vieną neuroną spręsti dviejų klasių uždavinį ir atlikti tyrimą su dvejomis duomenų aibėmis. Užduoties variantas pasirinktas pagal studento numerio paskutinio skaitmens dalybos iš trijų liekaną.

Studento numeris: 2110648

Užduoties variantas: Neurono mokymui naudoti stochastinį gradientinį nusileidimą ir ADALINE mokymo taisyklę.

2. Duomenų aibės ir klasės

Šiai užduočiai atlikti buvo naudojamos dvi duomenų aibes: Irisų ir Krūties vėžio. Irisų duomenyse yra trys klasės Setosa, Versicolor ir Virginica. Tačiau analizei buvo paimtos tik dvi: Versicolor ir Virginica. Šis duomenų rinkinys, kurį sudaro 150 įrašai, turi 4 požymių laukus ir vieną klasės lauką. Kadangi mes naudosime tik Versicolor ir Virginica klases, tai turėsime ištrinti nereikalingus mums laukus. Dvi likusios klasės turi lygiai po 50 įrašų. Paskutinis žingsnis būtų klasių laukus paversti į 0 ir 1, kad būtų paprasčiau prognozuoti klases su dirbtiniu neuronu.

```
5.0,2.3,3.3,1.0,Iris-versicolor
5.6,2.7,4.2,1.3,Iris-versicolor
5.7,3.0,4.2,1.2,Iris-versicolor
5.7,2.9,4.2,1.3,Iris-versicolor
6.2,2.9,4.3,1.3,Iris-versicolor
5.1,2.5,3.0,1.1,Iris-versicolor
5.7,2.8,4.1,1.3,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica
```

1 pav. *Nesutvarkyti irisų duomenys*

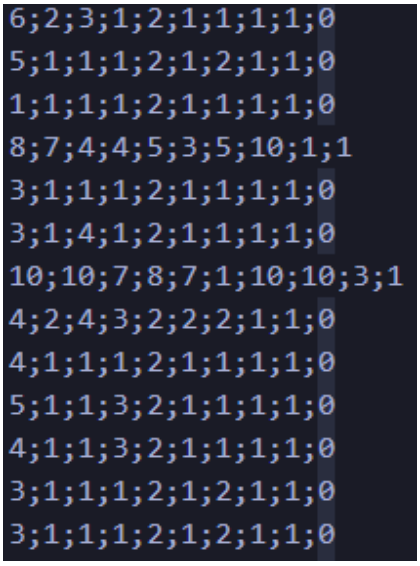
```
5.7;3.0;4.2;1.2;0
5.7;2.9;4.2;1.3;0
6.2;2.9;4.3;1.3;0
5.1;2.5;3.0;1.1;0
5.7;2.8;4.1;1.3;0
6.3;3.3;6.0;2.5;1
5.8;2.7;5.1;1.9;1
7.1;3.0;5.9;2.1;1
6.3;2.9;5.6;1.8;1
6.5;3.0;5.8;2.2;1
```

2 pav. *Sutvarkyti irisų duomenys*

Kita duomenų bazė – krūties vėžio, yra didesnė ne tik požymių, bet ir įrašų skaičiumi. Šią duomenų bazę sudaro 9 požymių laukai, viena klasės laukas ir ID laukas. ID laukai klasifikavimui nėra reikalingi, todėl buvo pašalinti. Taip pat vieno iš požymių, kai kurie laukai yra tušti, todėl buvo pašalintos tos eilutės, kurios buvo su praleistais įrašais. Paskutinis žingsnis buvo pakeisti klasių 2 – nepiktybinio naviko ir 4 – piktybinio naviko laukus atitinkamai į 0 ir 1. Po visų duomenų pertvarkymų iš 699 įrašų liko 684.

```
1070935,3,1,1,1,1,1,2,1,1,2
1071760,2,1,1,1,2,1,3,1,1,2
1072179,10,7,7,3,8,5,7,4,3,4
1074610,2,1,1,2,2,1,3,1,1,2
1075123,3,1,2,1,2,1,2,1,1,2
1079304,2,1,1,1,2,1,2,1,1,2
1080185,10,10,10,8,6,1,8,9,1,4
1081791,6,2,1,1,1,1,7,1,1,2
1084584,5,4,4,9,2,10,5,6,1,4
1091262,2,5,3,3,6,7,7,5,1,4
1096800,6,6,6,9,6,?,7,8,1,2
1099510,10,4,3,1,3,3,6,5,2,4
1100524,6,10,10,2,8,10,7,3,3,4
1102573,5,6,5,6,10,1,3,1,1,4
1103608,10,10,10,4,8,1,8,10,1,4
1103722,1,1,1,1,2,1,2,1,2,2
```

3 pav. *Nesutvarkyti krūties vėžio duomenys*



4 pav. *Sutvarkyti krūties vėžio duomenys*

3. Neurono mokymas

3.1. Duomenų paskirstymas ir pradinės svorių reikšmės

Apmokinant neuroną ar neuroninį tinklą reikalingos dažniausiai trys duomenų aibės: mokymo, testavimo ir validacijos. Nors šį kart pastaroji šį kart nebuvo naudojama, tačiau vis tiek reikia mokymo ir testavimo aibių, kad būtų galima apmokyti modelį ir jį įvertinti nepriklausomai nuo išmoktų duomenų. Santykis tarp aibių buvo pasirinktas 70:30, tai yra 70% duomenų atitenka mokymo aibei ir 30% testavimo aibei.

Abiem klasifikavimo uždaviniam spręsti yra reikalingi svoriai $\boldsymbol{w_0, w_1, ..., w_n}$, kuriuos apmokomas neuronas keis, kad būtų galima išspręsti klasifikavimo uždavinius. Svoriai pradžioje buvo pasirinkti atsitiktinai naudojant pseudo-atsitiktinio normaliojo pasiskirstymo funkcija.

3.2. Stochastinis gradientinis nusileidimas

Stochastinis gradientinis nusileidimas (SGD) yra optimizavimo algoritmas, naudojamas rasti funkcijos minimumą keičiant jos parametrus (svorius). SGD pagrindinis principas yra atnaujinti modelio svorius (parametrus) priešingai nuo gradiento krypties kiekvieno mokymo pavyzdžio. Šis metodas yra itin populiarus mokant neuroninius tinklus ir kitus mašininio mokymosi modelius dėl jo efektyvumo dirbant su dideliais duomenų kiekiais.

3.3. Epocha

Epocha yra vienas iteracijos ciklas per visą mokymo duomenų rinkinį. Mano aprašytoje ***AdalineSGD*** klasėje, epocha apima tiek iteracijų, kiek yra duomenų. Kiekvienoje epochoje mokymo duomenys yra sumaišomi, o paskui modelis atnaujina svorius naudodamasis kiekvienu mokymo pavyzdžiu atskirai. Tai tęsiama iki tol, kol pasiekiama nustatyta epochų skaičiaus riba arba kol mokymo klaidos dydis sumažėja žemiau nustatyto slenksčio (ankstyvo sustabdymo kriterijus).

3.4. ADALINE taisyklė

Šiai užduočiai buvo paskirta ADALINE taisyklė, kuri nusako kaip yra mokomas neurono modelis. Pagrindinis dalykas, kuris yra susijęs su ja, tai yra tiesinės aktyvacijos funkcijos naudojimas apskaičiuojant prognozes $f(a_i) = f(\sum_{k=0}^{n} w_k x_{ik})$. Kartu su aktyvacijos funkcija yra naudojamo mokymo taisyklė, kuri nusako kaip turi būti atnaujinami svoriai (parametrai) $w_k := w_k + \eta(t_i - y_i)(x_{ik})$.

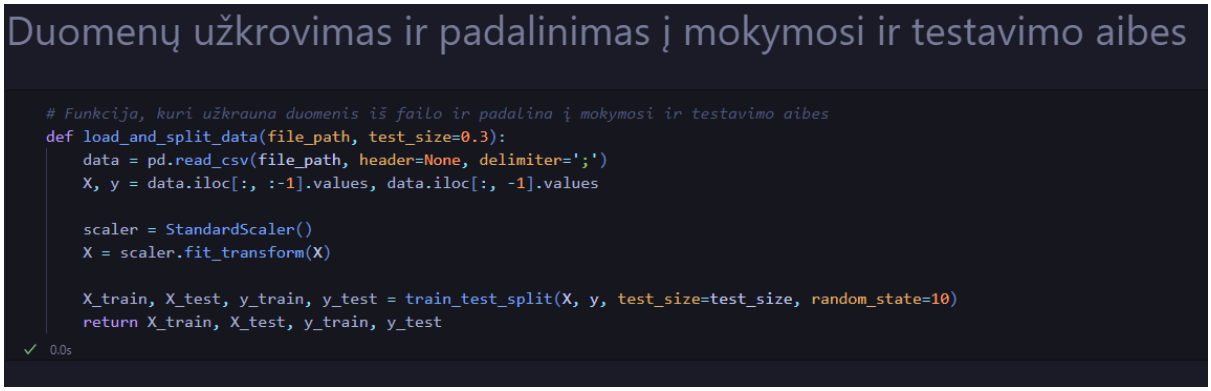
3.5. Neurono mokymo įgyvendinimas programoje

Neuronas buvo sukurtas ir apmokytas naudojant Jupyter Notebook įrankį ir Python programavimo kalbą. Python kalba buvo pasirinkta dėl plataus bibliotekų pasirinkimo, kurios turi pagalbines funkcijas kaip diagramų atvaizdavimą ar duomenų aibių paskirstymą.

```python
class AdalineSGD:
    # Konstruktorius su parametrais: mokymo greitis, epochų skaičius ir atsitiktinių skaičių generatoriaus pradinė padėtis.
    def __init__(self, learning_rate=0.01, epochs=50, random_state=10):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self.random_state = random_state # Atsitiktinių skaičių generatoriaus pradinė padėtis
        self.weights = None
        self.bias = None
        self.loss = []
        self.accuracy = []

    # Mokymo funkcija, kuri atnaujina svorius naudojant stochastinį gradientinį nusileidimą.
    def fit(self, X, y, E_min):
        rgen = np.random.RandomState(self.random_state) # Sukuriame atsitiktinių skaičių generatorių
        self.weights = rgen.normal(loc=0.0, scale=0.01, size=X.shape[1]) # Inicializuojame svorius mažais atsitiktiniais skaičiais
        self.bias = 1.
        self.loss = []
        self.accuracy = []

        # Pradedame mokymo procesą, kuris vyks iki nurodyto epochų skaičiaus arba kol pasieksime minimalią paklaidą
        for epoch in range(self.epochs):
            X, y = shuffle(X, y, random_state=self.random_state)
            loss = 0

            # Iteruojame per kiekvieną mokymo duomenų pavyzdį
            for x, target in zip(X, y):
                output = self.neuron_input(x) # Apskaičiuojame neurono išvestį
                error = (target - output) # Apskaičiuojame paklaidą
                self.weights += self.learning_rate * x.dot(error) # Atnaujiname svorius
                self.bias += self.learning_rate * error
                loss += 0.5 * error**2 # Skaičiuojame paklaidos kvadratą ir jį pridedame prie bendros paklaidos


            avg_loss = loss / len(y)
            self.loss.append(avg_loss)
            self.accuracy.append(self.calculate_accuracy(X, y))
            if avg_loss <= E_min:  # Jeigu vidutinė paklaida mažesnė ar lygi nustatytam slenksiui, stabdome mokymo procesą
                print(f"Mokymasis sustabdytas {epoch+1} epochoje, nes pasiektas minimalus paklaidos slenkstis.")
                break
```

5 pav. *Neurono inicializavimas ir mokymo funkcija*

Kode esančiame **5 pav.** pateikta dirbtinio neurono realizacija naudojant klases, kurioje laikome reikalingą informaciją ir funkcijas, reikalingas dirbti su dirbtiniu neuronu. Viena iš pagrindinių funkcijų yra pavadinta ***fit***, kurioje yra įgyvendintas neurono mokymasis naudojant stochastinį gradientinį nusileidimą ir ADELINE taisyklę. Pateiktoje klasėje sukuriant neuroną, galima nurodyti tam tikrus hiperparametrus kaip mokymosi greitis, epochų skaičius, svoriai ir poslinkis.



6 pav. *Duomenų užkrovimo ir padalinimo funkcija.*

6 pav. esančiame kode yra atvaizduota funkcija, kuri leidžia nuskaityti duomenis iš csv failų, normalizuoti nuskaitytus duomenis ir išskirstyti į mokymo ir testavimo aibes.

```
def test_data_evaluation(X_test, y_test, model):
    predictions = model.predict(X_test)
    print("Testavimo duomenų įrašų klasifikacija:")
    for i, (pred, actual) in enumerate(zip(predictions, y_test)):
        print(f"Įrašas {i+1}: Nustatyta klasė = {pred}, Turėjo būti = {actual}")

# Funkcija atliekanti eksperimentą ir apskaičiuojanti visas metrikas
def run_experiment(data_file_path, learning_rate, epochs, E_min):
    X_train, X_test, y_train, y_test = load_and_split_data(data_file_path)
    adaline = AdalineSGD(learning_rate=learning_rate, epochs=epochs)
    adaline.fit(X_train, y_train, E_min)

    epochs_range = range(1, len(adaline.loss) + 1)
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, adaline.loss, color='red', label='Paklaida')
    plt.xlabel('Epocha')
    plt.ylabel('Paklaida')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, adaline.accuracy, color='blue', label='Tikslumas')
    plt.xlabel('Epocha')
    plt.ylabel('Tikslumas')
    plt.legend()

    plt.tight_layout()
    plt.show()

    print(f"Epochų skaičius: {adaline.epochs}")
    print(f"Svoriai: {adaline.weights}")
    print(f"Poslinkis: {adaline.bias}")
    print(f"Mokymo duomenų tikslumas: {adaline.accuracy[-1]}")
    print(f"Mokymo duomenų paklaida: {adaline.loss[-1]}")
    print(f"Testavimo duomenų tikslumas: {adaline.calculate_accuracy(X_test, y_test)}")
    print(f"Testavimo duomenų paklaida: {adaline.calculate_MSE(X_test, y_test)}")
    print("\n\n")

    test_data_evaluation(X_test, y_test, adaline)
```

7 pav. *Pagalbinės ir metrikų funkcijos.*

7 pav. esančios funkcijos atlieka svarbų vaidmenį apmokant neuroną ir atliekant tyrimus, kurie parodyti kitame skyriuje. Pagrindinis jš funkcionalumas yra paskaičiuoti metrikas po modelio apmokymo ir testavimo, ir atvaizduoti duomenis grafe ir kaip lentelę.

4. Tyrimo rezultatai

4.1. Apmokyto neurono geriausi rezultatai

Ieškant geriausio apmokyto neurono modelio varianto, buvo keičiami hiperparametrai pagal, kuriuos buvo galima gana nesunkiai atrasti optimalų sprendimą. Rezultatai yra pateikti dviem skirtingoms duomenų bazėms: irisų ir krūtų vėžio.

Pirmasis irisų duomenų aibės modelis buvo geriausias su **1 lentelė.** Irisų duomenų geriausi parametrai pateiktais hiperparametrais. Antrasis krūtų vėžio modelis buvo geriausias su **2 lentelė** pateiktais hiperparametrais. Pradiniai geriausi hiperparametrai abiejuose atvejuose sutapo, todėl galima manyti kad su nedideliu duomenų kiekiu 0,001 mokymosi greitis yra arti optimalaus.

1 lentelė. *Irisų duomenų geriausi parametrai.*

Mokymosi greitis	Epochų skaičius	E_{min}
0,001	100	0,01

2 lentelė. *Krūtų vėžio duomenų geriausi parametrai.*

Mokymosi greitis	Epochų skaičius	E_{min}
0,001	100	0,01

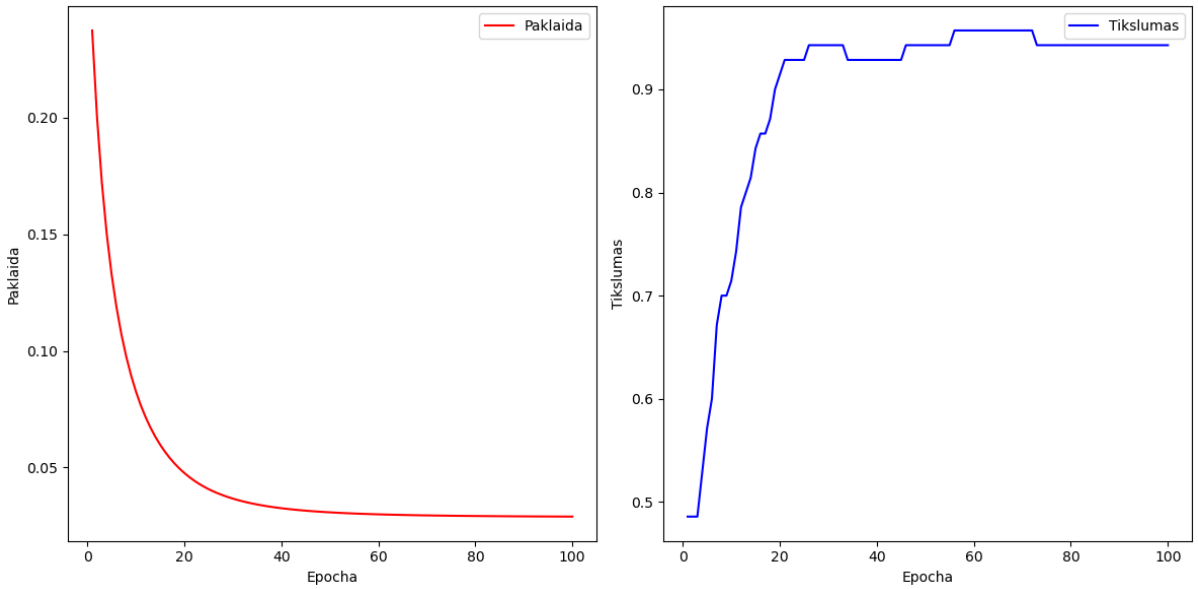
```
Epochų skaičius: 100
Svoriai: [-0.06111703 -0.12332067  0.25120512  0.31989833]
Poslinkis: 0.5051230149850406
Mokymo duomenų tikslumas: 0.9428571428571428
Mokymo duomenų paklaida: 0.028908274408891082
Testavimo duomenų tikslumas: 1.0
Testavimo duomenų paklaida: 0.0



Testavimo duomenų įrašų klasifikacija:
Įrašas 1: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 2: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 3: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 4: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 5: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 6: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 7: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 8: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 9: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 10: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 11: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 12: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 13: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 14: Nustatyta klasė = 1, Turėjo būti = 1
...
Įrašas 27: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 28: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 29: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 30: Nustatyta klasė = 1, Turėjo būti = 1
```

8 pav. *Irisų duomenų modelio rezultatai*

```
Epochų skaičius: 100
Svoriai: [0.09867365 0.03579249 0.05001816 0.02471733 0.03366868 0.16372938
 0.06031235 0.06170386 0.01170531]
Poslinkis: 0.3527550999547938
Mokymo duomenų tikslumas: 0.9644351464435147
Mokymo duomenų paklaida: 0.017494195663526585
Testavimo duomenų tikslumas: 0.9658536585365853
Testavimo duomenų paklaida: 0.03414634146341464



Testavimo duomenų įrašų klasifikacija:
Įrašas 1: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 2: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 3: Nustatyta klasė = 0, Turėjo būti = 1
Įrašas 4: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 5: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 6: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 7: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 8: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 9: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 10: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 11: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 12: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 13: Nustatyta klasė = 0, Turėjo būti = 0
...
Įrašas 202: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 203: Nustatyta klasė = 0, Turėjo būti = 0
Įrašas 204: Nustatyta klasė = 1, Turėjo būti = 1
Įrašas 205: Nustatyta klasė = 0, Turėjo būti = 0
```

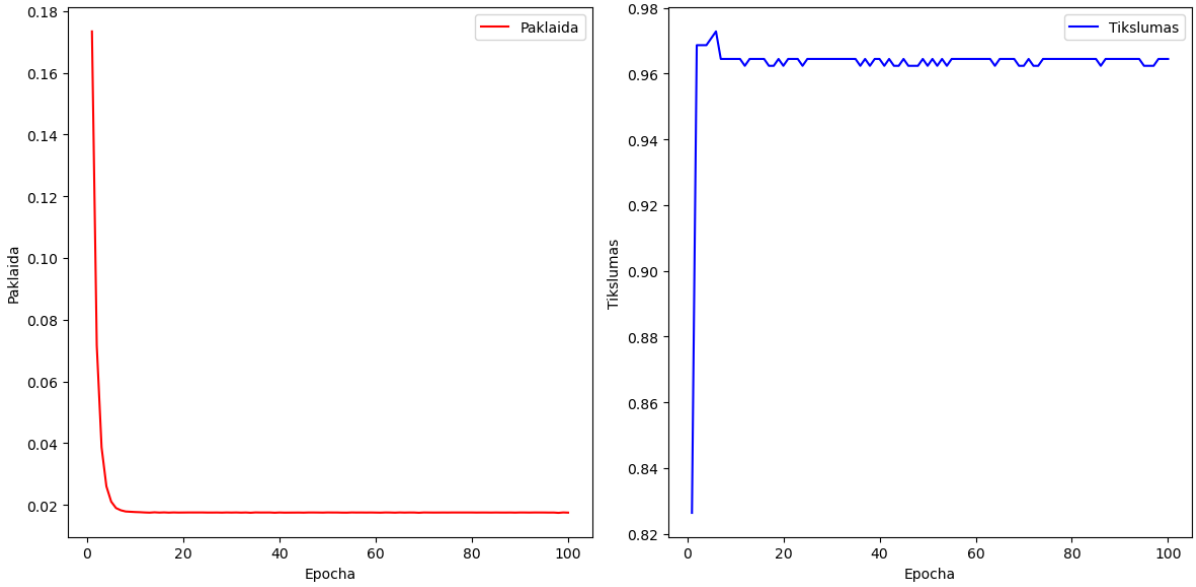
9 pav. *Krūties vėžio modelio rezultatai*

Pagal gautus rezultatus **8 pav.** galima pastebėti, kad gautas tikslumas ir paklaida testavimo aibei yra ideali, tačiau tai nebūtinai reiškia, kad modelis yra geras. Kadangi duomenų nėra pakankamai daug, tai gali reikšti tiesiog persimokymą (overfitting). Ant mokymosi duomenų taip pat buvo gautas aukštas tikslumas ir maža paklaida. Tą patį, nors ir neidealų atvejį galima pastebėti antrame variante su krūtų vėžio duomenimis, kurių rezultatai yra **9 pav.** Krūties vėžio modelio rezultatai



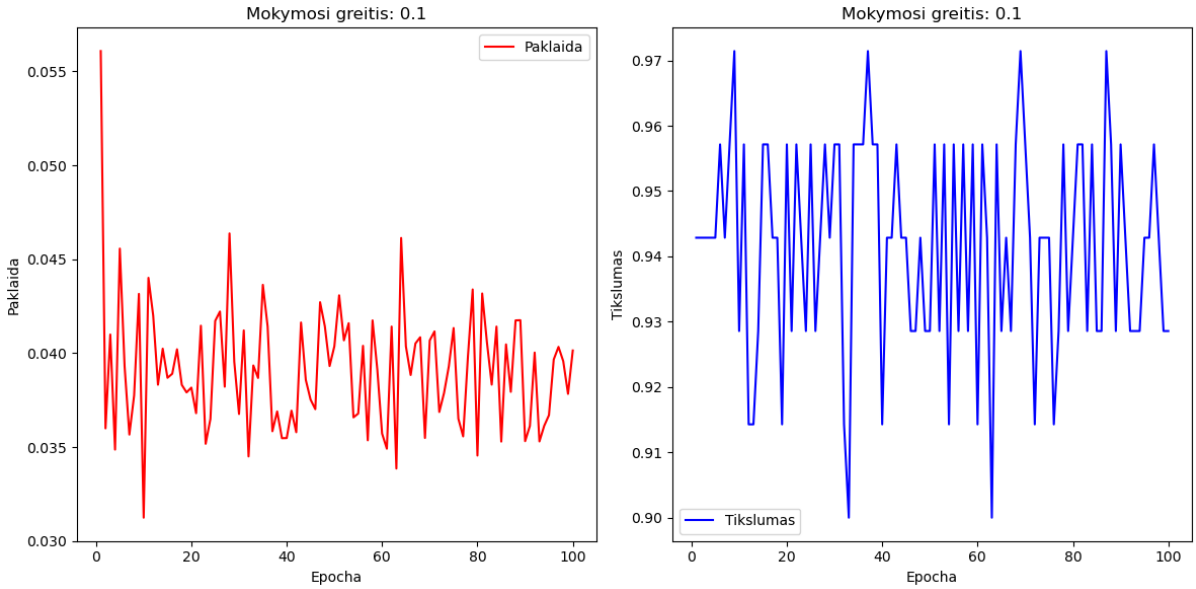
10 pav. *Paklaidos ir tikslumo priklausymas nuo epochos irisų duomenims*

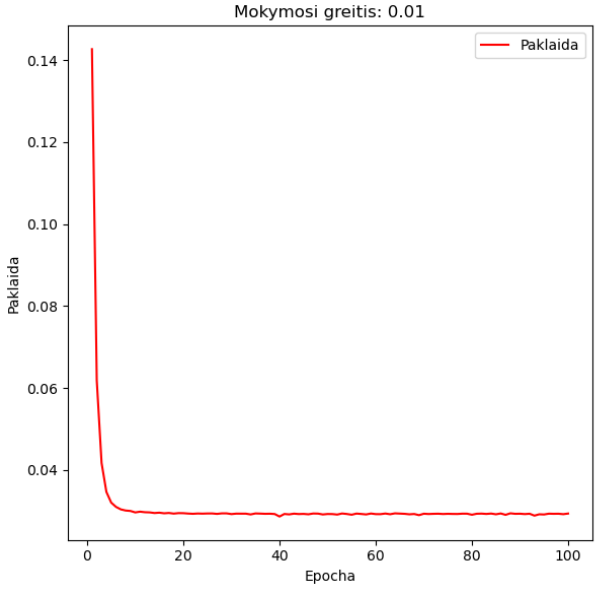
Iš **10 pav.** pateiktų diagramų galima pastebėti, kad ADALINE modelio mokymosi proceso metu paklaidos vertė greitai mažėja pradžioje, o vėliau stabilizuojasi, kas rodo efektyvų modelio pradinį mokymąsi ir konvergavimą į tam tikrą optimalų svorių rinkinį. Klasifikavimo tikslumo grafikas taip pat rodo greitą tikslumo didėjimą pradinėse mokymo epochose, bet toliau jis pasižymi nežymiais svyravimais, išliekant arti 90% tikslumo. Tai gali reikšti apie modelio gerą veikimą su mokymo duomenimis. Tačiau tokie tikslumo svyravimai kelia klausimų dėl modelio gebėjimo generalizuoti, todėl būtų naudinga papildomai įvertinti modelį su neapmokytų duomenų rinkiniu, kad galima būtų nustatyti jo tikslumą realiomis sąlygomis.

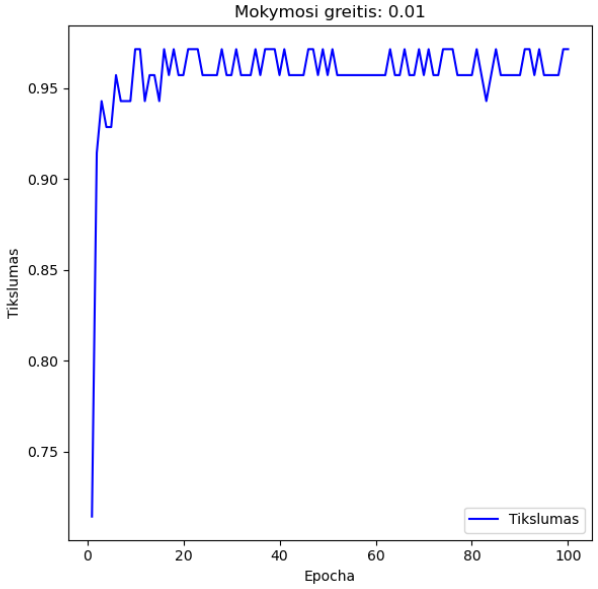


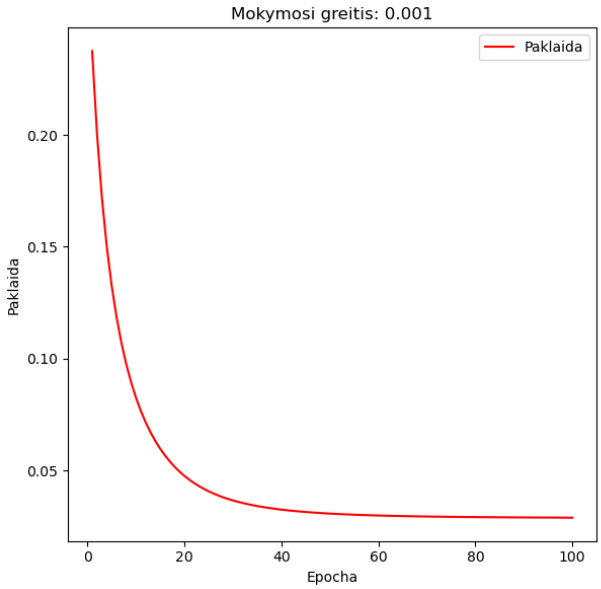
11 pav. *Paklaidos ir tikslumo priklausymas nuo epochos krūtų vėžio duomenims*

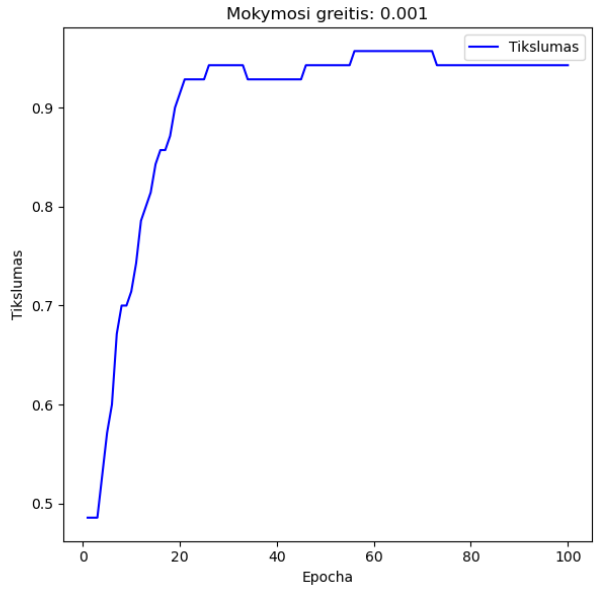
Iš **11 pav.** galima pastebėti, kad grafai labai panašūs į pirmiau pateiktas diagramas. Šį kart galima pastebėti, kad paklaidos ir tikslumo reikšmės labai greitai didėja ar mažėja. Visa tai gali dar labiau patvirtinti, kad modelis su nedaug duomenų gali labai greitai juos išmokti. Tačiau tai tikriausiai neužtikrina gero veikimo su visiškai kitais testavimo duomenimis.

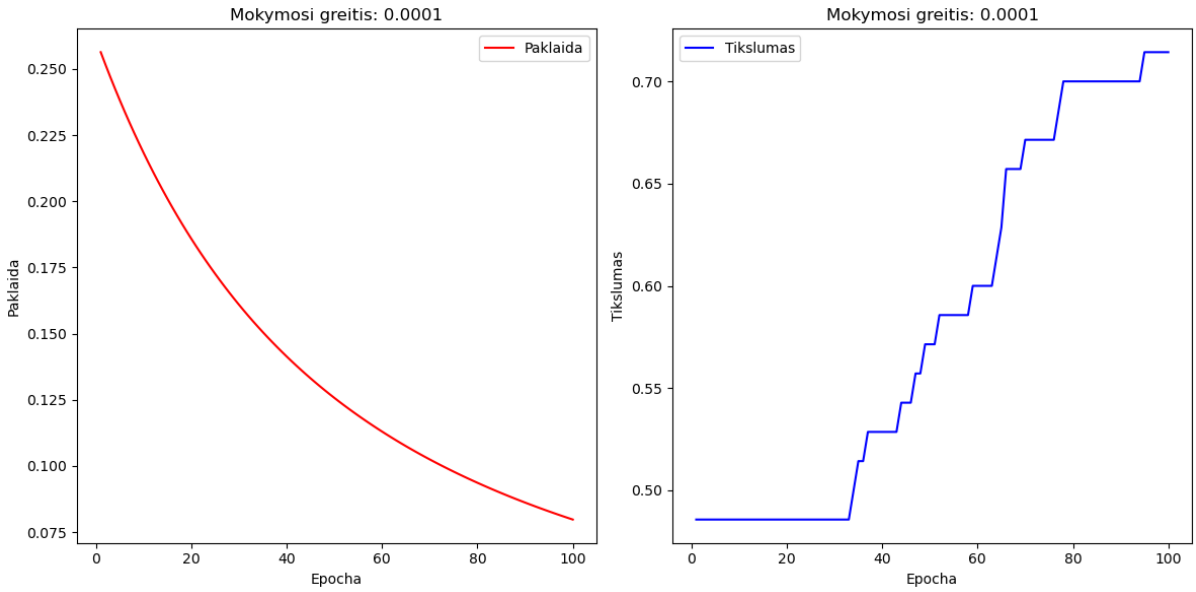



Mokymosi greitis: 0.01
Paklaida
Epocha
0.14
0.12
0.10
0.08
0.06
0.04
0
20
40
60
80
100


Mokymosi greitis: 0.01
Tikslumas
Epocha
0.95
0.90
0.85
0.80
0.75
0
20
40
60
80
100


Mokymosi greitis: 0.001
Paklaida
Epocha
0.20
0.15
0.10
0.05
0
20
40
60
80
100


Mokymosi greitis: 0.001
Tikslumas
Epocha
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100



12 pav. *Paklaidos ir tikslumo priklausymas nuo mokymosi greičio*

Pagal **12 pav.** pavaizduotas diagramas galima pastebėti, kad per greitas mokymasis gali trukdyti pasiekti optimalų tikslumą ir paklaidą, nes skaičiuojant gradientą reikšmė tikriausiai peršoka minimumą. Labai mažinant mokymosi greitį galima pastebėti kad paklaida daug lėčiau mažėja ir kreivė vis labiau panašėja į tiesę. Tikslumas taip pat dėl nukenčia nes nebepasiekia 90% tikslumo. Iš šių matavimų galima pastebėti, kad labai didinant ir mažinant mokymosi greitį nebūtinai bus įmanoma pasiekti gerą tikslumą.

5. Išvados

Tyrimas, atliktas mokant vieną neuroną sprendžiant klasifikavimo uždavinį su Irisų ir Krūties vėžio duomenimis, parodė, kad sėkmingas modelio mokymas priklauso nuo tinkamo duomenų paruošimo, hiperparametrų derinimo ir modelio gebėjimų išvengti persimokymo. ADALINE taisyklė ir stochastinis gradientinis nusileidimas pasirodė esantys efektyvios strategijos siekiant aukšto tikslumo. Tačiau svarbu atkreipti dėmesį į modelio generalizavimo gebėjimus, kad būtų užtikrintas jo pritaikomumas realioms problemoms spręsti.