VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

2 užduotis (Vieno neurono mokymas sprendžiant klasifikavimo uždavinį)

Praktinio darbo ataskaita

Atliko: Monika Mirbakaitė
VU el. p.: monika.mirbakaite@mif.stud.vu.lt

Vertino: dr. Viktor Medvedev

Vilnius
2025

TURINYS

1. UŽDUOTIES TIKSLAS

Apmokyti vieną neuroną spręsti dviejų klasių uždavinį ir atlikti tyrimą.

2. DARBO ATLIKIMAS

2.1. Naudoti duomenys

Buvo naudojama krūties vėžio duomenų aibė iš tinklalapio https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original) (breast cancer-wisconsin.data). Šiuose duomenyse yra dvi klasės: 2 – nepiktybinis navikas , 4 – piktybinis navikas.

Dirbtinio neurono mokymas vyko naudojant 9 požymius: gumbelio storis, ląstelių dydžio vienodumas, ląstelių formos vienodumas, kraštų prisijungimas, vienos epitelinės ląstelės dydis, plikieji branduoliai, švelnusis chromatinas, normalieji branduoliai, mitozės.

Jog neurono mokymas vyktų sėkmingai duomenys, pirmiausia, buvo atliktas duomenų pirminis apdorojimas: duomenų požymių atrinkimas (panaikintas ID stulpelis, eilutės, kurios turėjo nežinomų požymių) bei duomenų normalizavimas (nepiktybiniai navikai žymimi 0, o piktybiniai – 1). Taip pat, buvo išmaišytos duomenų eilutės tikslesniam neurono mokymui. Atlikus pradinį duomenų apdorojimą mokymui buvo naudojamas failas, kuriame yra 10 stulpelių (9 požymiai ir klasė), 683 eilutės (buvo panaikinta 16 eilučių, kuriose buvo nežinomų duomenų). Šio proceso įgyvendinimas pateikiamas 5.1 poskyryje.

2.2. Duomenų aibės

Buvo įgyvendintas neurono mokymas, validavimas ir testavimas, todėl turimi duomenys buvo padalinti į tris aibes santykiu 80:10:10. Paprastesnio darbo dėlei, apdorotų duomenų failas buvo išskirstytas į 3 kitus failus (mokymo, validavimo, testavimo). Šio proceso realizacija pateikiama 5.1 poskyryje.

2.3. Pradinės svorio, poslinkio reikšmės

Realizuojant paketinio bei stochastinio gradientinio nusileidimo mokymo modelius pradinės svorių, poslinkio reikšmės buvo generuojamos atsitiktinai, naudojant funkciją `np.random.randn()`.

2.4. Programos kodas su komentarais

Programos kodas su komentarais bei paaiškinimais pateikiamas 5 skyriuje.

2.5. Naudoti mokymo metodai

Neuronas buvo apmokytas dviem metodais: paketiniu gradientiniu nusileidimu bei stochastiniu gradientiniu nusileidimu. Abu algoritmai yra naudoti neurono mokymo modeliui įgyvendinti, siekiant sumažinti prognozės paklaidą ir padidinti klasifikavimo tikslumą ir tokiu būdu išgauti optimalius modelio sprendinius.

Taikant paketinį gradientinį nusileidimą, vienos iteracijos metu panaudojami visi mokymo duomenys. Siekiama, kad paklaida būtų minimali visiems mokymo duomenims. Pirmiausia į neuroną perduodami visus mokymo duomenis ir apskaičiuojami kiekvieno duomenų įrašo paklaidos funkcijos gradientas. Tada imamas gradientų vidurkis ir atnaujinami svoriai (ir poslinkis) naudojant apskaičiuotą vidurkį.

Stochastinio gradientinio nusileidimo algoritme tikras funkcijos gradientas aproksimuojamas gradientu, gautu pagal vieną mokymo duomenų įrašą. T. y. siekiama, kad paklaida būtų minimali i-tajam mokymo duomenų įrašui. Taikant stochastinį gradientinį nusileidimą, vienos iteracijos metu panaudojamas tik vienas mokymo duomenų įrašas: kiekvienam įrašui skaičiuojamas gradientas ir atnaujinami svoriai (ir poslinkis).

Paketinio gradientinio nusileidimo atveju viena epocha atitinka vieną iteraciją. Stochastinio gradientinio nusileidimo atveju viena epocha atitinka m iteracijų, čia m yra mokymo duomenų kiekis.

2.6. Gauti duomenys apmokius dirbtinį neuroną

Žemiau šiame poskyryje pateiktose lentelėse (1 lentelė, 2 lentelė) nurodyti gauti duomenys apmokius neuroną ir nustačius atvejį, kada gaunamas didžiausias klasifikavimo tikslumas ir mažiausia paklaida validavimo duomenims.

2.6.1. Paketinio gradientinio nusileidimo algoritmo atveju

1 lentelė. Pateikti duomenys, kai gaunamas didžiausias klasifikavimo tikslumas ir mažiausia paklaida validavimo duomenims (1).

Svoriai	-0,49304838; 0,60515874; 0,36653348; 0,28176715; -0,7708675; 1,65900871
Poslinkis	-0,09429413611167349
Epochų skaičius	1000 (didžiausias klasifikavimo tikslumas ir mažiausia paklaida validavimo duomenims epochoje 681).
Paklaidos paskutinėje epochoje mokymo duomenims.	0,1911760
Paklaidos paskutinėje epochoje validavimo duomenims	0,135010
Klasfikavimo tikslumas paskutinėje epochoje mokymo duomenims	0,8088
Klasfikavimo tikslumas paskutinėje epochoje validavimo duomenims	0,8571
Paklaida testavimo duomenims	0,898551
Klasfikavimo tikslumas testavimo duomenims	0,1014

Kiekvieno testavimo duomenų įrašo prognozuojamos ir tikrosios klasės nurodytos 5.4.1 poskyryje.

2.6.2. Stochastinio gradientinio nusileidimo algoritmo atveju

2 lentelė. Pateikti duomenys, kai gaunamas didžiausias klasifikavimo tikslumas ir mažiausia paklaida validavimo duomenims (2).

Svoriai	-1,14681097; 3,82696885; 0,96677118; -1,09145476; -5,22677025; 4,38780068
Poslinkis	-0,09429413611167349

Epochų skaičius	1000 (didžiausias klasifikavimo tikslumas ir mažiausia paklaida validavimo duomenims epochoje 426).
Paklaidos paskutinėje epochoje mokymo duomenims.	0,092099
Paklaidos paskutinėje epochoje validavimo duomenims	0,088235
Klasfikavimo tikslumas paskutinėje epochoje mokymo duomenims	0,9011
Klasfikavimo tikslumas paskutinėje epochoje validavimo duomenims	0,9118
Paklaida testavimo duomenims	0,086957
Klasfikavimo tikslumas testavimo duomenims	0,9130

Kiekvieno testavimo duomenų įrašo prognozuojamos ir tikrosios klasės nurodytos 5.4.2 poskyryje.

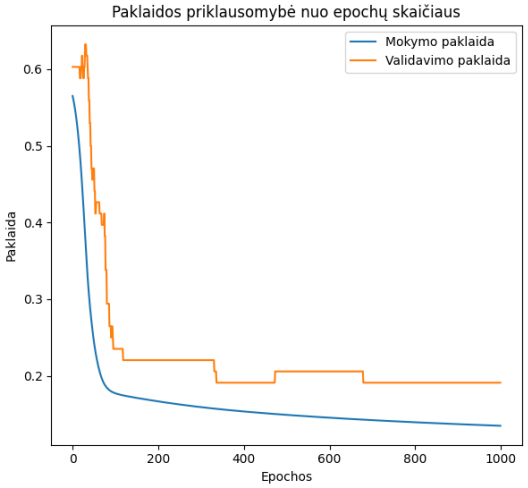
3. TYRIMAS

3.1. Paklaidos reikšmės priklausomybė nuo epochų skaičiaus mokymo ir validavimo duomenims

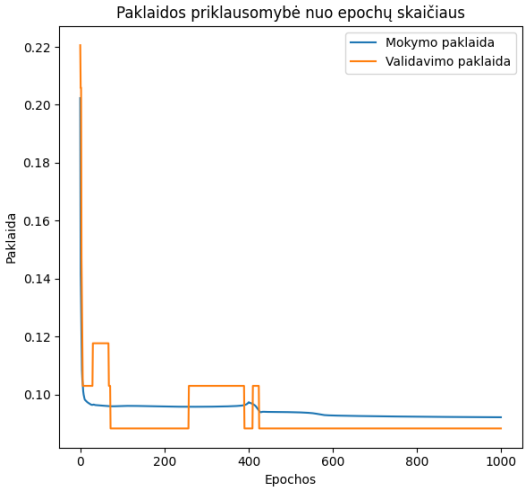
Pateiktose nuotraukoje vaizduojami du grafikai, kurie parodo paklaidos reikšmės priklausomybę nuo epochų skaičiaus, naudojant du skirtingus gradientinio nusileidimo metodus: paketinio (1 pav.) ir stochastinio (2 pav.). Mokymo paklaida pažymėta mėlyna spalva, o validavimo paklaida – oranžine. x ašyje yra atidėdamos epochos, o y ašyje paklaidų reikšmės.

1 pav. paklaida greitai mažėja per pirmas kelias epochas, o vėliau, stabilizuojasi. Įdomu tai, kad validavimo paklaida taip pat mažėja, tačiau ji stabilizuojasi anksčiau. Tai parodo, jog neuronas buvo apmokytas sėkmingai.

2 pav. validavimo ir mokymo paklaidos mažėja panašiai kaip ir ankstesniame algoritme, tačiau galima pastebėti, kad mokymo paklaidos linija stabilizuojasi greičiau, o validavimo – turi didesnių svyravimų tarp paklaidų reikšmių.



1 pav. Paklaidos reikšmės priklausomybė paketiniu gradientiniu nusileidimo mokymo metodu.



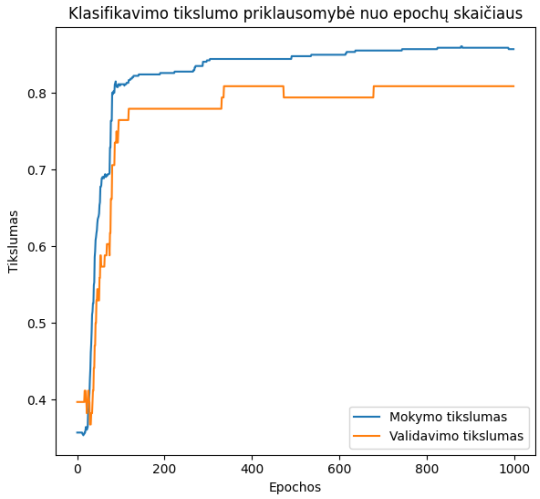
2 pav. Paklaidos reikšmės priklausomybė stochastiniu gradientiniu nusileidimo mokymo metodu.

3.2. Klasifikavimo tikslumo priklausomybė nuo epochų skaičiaus mokymo ir validavimo duomenims

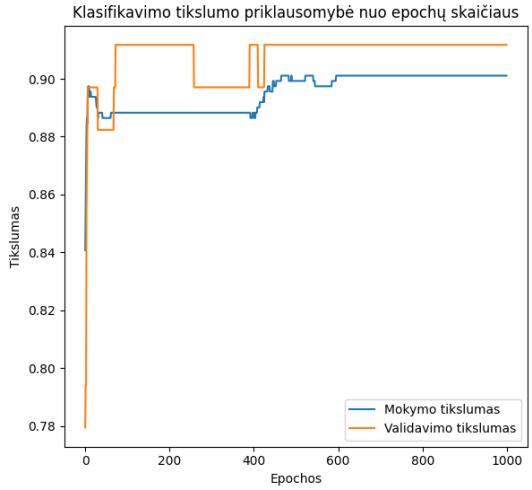
Pateiktose nuotraukoje vaizduojami du grafikai, kurie parodo klasifikavimo tikslumo reikšmės priklausomybę nuo epochų skaičiaus, naudojant du skirtingus gradientinio nusileidimo metodus: paketinio (3 pav.) ir stochastinio (4 pav.). Mokymo paklaida pažymėta mėlyna spalva, o validavimo paklaida – oranžine. x ašyje yra atidėdamos epochos, o y ašyje klasifikavimo tikslumo reikšmės.

3 pav. galima pastebėti, kad klasifikavimo tikslumo grafikai yra priešingi paklaidos grafikams. Modelis yra tiksliausias, kai paklaida yra kuo mažesnė, o tikslumas – kuo didesnis. Šiuo atveju taip ir yra: mokymo grafikas sparčiai didėja, o, po kiek laiko, didėja žymiai lėčiau. Validavimo duomenų grafikas turi panašią tendenciją: po kiek laiko tikslumas stabilizuojasi, tačiau yra didesnių „šuolių“.

4 pav. tikslumas įgyjamas greičiau, tačiau, po kiek laiko, rezultatai yra itin nestabilūs. Tai rodo, jog dideli „šuoliai“ gali lemti optimalaus atsakymo „prašokimo“.



3 pav. Klasifikavimo tikslumo priklausomybė gradientiniu nusileidimo mokymo metodu.



4 pav. Klasifikavimo tikslumo priklausomybė stochastiniu nusileidimo mokymo metodu.

3.3. Rezultatai su skirtingomis mokymosi greičio reikšmėmis

3.3.1. Paketinio gradientinio nusileidimo algoritmo atveju

Kintant mokymosi greičiui nuo 0,05 iki 0,5, mokymo paklaida sumažėjo (nuo 0,135010 iki 0,092381). Klasifikavimo tikslumas šiek tiek padidėjo (0,8571 su 0,05 greičiu ir 0,9029 su 0,5 greičiu). Reikalingų epohų skaičius, mokymo trukmė taip pat sumažėjo.

Validavimo paklaida mažėja didėjant mokymosi greičiui (0,191176 su 0,05 greičiu ir 0,088235 su 0,5 greičiu). Taip pat pagerėja validavimo tikslumas: nuo 0,8088 (0,05 greitis) iki 0,9118 (0,5 greitis).

Testavimo etape paklaida šiek tiek sumažėja (0,1014 su 0,05 greičiu ir 0,086957 su 0,5 greičiu), o testavimo klasifikavimo tikslumas yra aukštesnis (0,898551 su 0,05 greičiu ir 0,9130 su 0,5 greičiu).

Visa tai rodo, kad greitesnis mokymosi greitis leidžia greičiau užbaigti mokymą, padeda greičiau pasiekti optimalesnį modelio variantą.

Svoriai keičiasi priklausomai nuo mokymosi greičio. Aukštesnis mokymosi greitis (0,5) lemia didesnius svorių pokyčius. Tai gali prisidėti prie modelio mažesnio tikslumo, didesnio klaidų kiekio atsiradimo.

3 lentelė. Paketinio gradientinio nusileidimo metodo rezultatai kintant mokymosi greičiui.

	Mokymosi greitis		
	0,05	**0,2**	**0,5**
MOKYMO ETAPAS			
Paklaida	0,135010	0,094607	0,092381
Klasifikavimo tikslumas	0,8571	0,9011	0,9029
Epocha	989	988	961
Mokymo trukmė	4,4378	4,3615	4,3780
Poslinkis	-0,09429413611167349	-0.09429413611167349	-0.09429413611167349
VALIDAVIMO ETAPAS			
Paklaida	0,191176	0,088235	0,088235
Klasfikavimo tikslumas	0,8088	0,9118	0,9118
Epocha	680	782	392
TESTAVIMO ETAPAS			
Paklaida	0,1014	0,072464	0,086957
Klasfikavimo tikslumas	0,898551	0,9275	0,9130

4 lentelė. Paketinio gradientinio nusileidimo metodo svoriai kintant mokymosi greičiui.

	Mokymosi greitis		
	0,05	**0,2**	**0,5**
Svoris Nr. 1	-0,49304838	-0,40812863	-0.44215687
Svoris Nr. 2	0,60515874	1,05920912	1.37311061

Svoris Nr. 3	0,36653348	0,47491003	0.40654923
Svoris Nr. 4	0,28176715	-0,01891844	-0.14660778
Svoris Nr. 5	-0,7708675	-1,44108927	-1.69365293
Svoris Nr. 6	1,65900871	1,25034948	1.37285122
Svoris Nr. 7	-0,46116334	-0,56735608	-0.59871048
Svoris Nr. 8	-0,68157454	0,46948328	0.70530637
Svoris Nr. 9	-0,54460605	-0,8701378	-1.04038802

3.3.2. Stochastinio gradientinio nusileidimo algoritmo atveju

5 lentelėje ir 6 lentelėje pateikti stochastinio gradientinio nusileidimo metodo rezultatai, kintant mokymosi greičiui. Matoma kaip mokymosi greitis daro įtaką mokymo, validavimo ir testavimo rezultatams, svorių reikšmėms.

Mokymosi greičio padidėjimas nuo 0,05 iki 0,5 lemia nedidelį paklaidų padidėjimą bei nedidelį tikslumo sumažėjimą Tačiau neurono modelis didinant mokymosi greitį greičiau apsimoko (4,6836 s, kai greitis 0,05 ir 4,5229 s, kai greitis 0,5).

Su didesniu mokymosi greičiu testavimo duomenų paklaida mažėja (0,088235 su 0,05 greičiu ir 0,073529 su 0,5 greičiu). Klasifikavimo tikslumas didėja nuo 0,9118 (0,05 greitis) iki 0,9265 (0,5 greitis). Tai rodo, kad didesnis mokymosi greitis padidina modelio tikslumą dėl optimaliau parinktų svorių.

Testavimo etape paklaidos šiek tiek svyruoja, tad su didesniu greičiu modelio paklaida kinta nežymiai. Testavimo klasifikavimo tikslumas šiek tiek svyruoja, tačiau vis tiek išlieka labai aukštas – 0,9130 (0,05; 0,5 greitis).

Kuo didesnis mokymosi greitis, tuo didesnis skirtumas tarp svorių reikšmių, o tai reiškia, kad svoriai keičiasi greičiau, kas gali lemti mažesnį modelio tikslumą, didina klaidų riziką.

5 lentelė. Stochastinio gradientinio nusileidimo metodo rezultatai kintant mokymosi greičiui.

	Mokymosi greitis		
	0,05	**0,2**	**0,5**
MOKYMO ETAPAS			
Paklaida	0,092099	0,092496	0,094015
Klasifikavimo tikslumas	0,9011	0,9084	0,9066
Epocha	596	986	916
Mokymo trukmė (sek.)	4,6836	4,4511	4,5229
Poslinkis	-0,09429413611167349	-0,09429413611167349	-0,09429413611167349
VALIDAVIMO ETAPAS			
Paklaida	0,088235	0,088235	0,073529
Klasfikavimo tikslumas	0,9118	0,9118	0,9265
Epocha	426	42	32
TESTAVIMO ETAPAS			
Paklaida	0,086957	0,101449	0,086957

Klasfikavimo tikslumas	0,9130	0,8986	0,9130

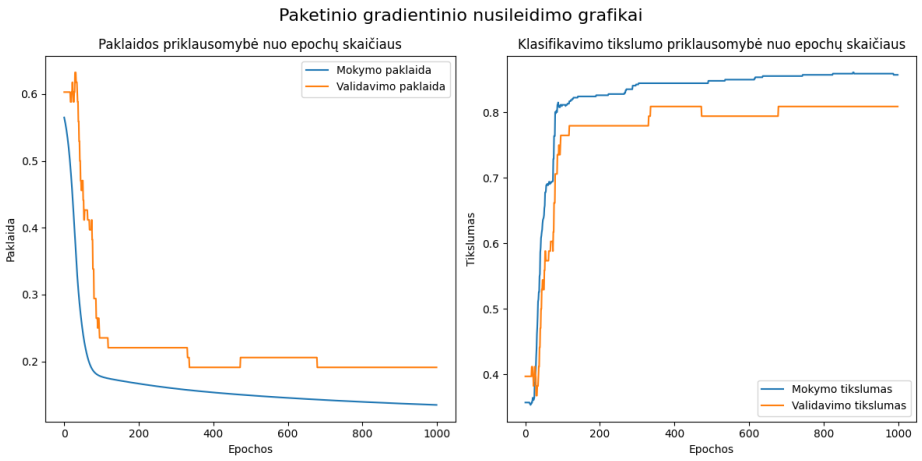
6 lentelė. Stochastinio gradientinio nusileidimo metodo svoriai kintant mokymosi greičiui.

	Mokymosi greitis		
	0,05	**0,2**	**0,5**
Svoris Nr. 1	-1,14681097	-3,27736064	-5,7340281
Svoris Nr. 2	3,82696885	11,02508565	20,14449701
Svoris Nr. 3	0,96677118	1,70017274	1,37530403
Svoris Nr. 4	-1,09145476	-2,27733787	-3,13326592
Svoris Nr. 5	-5,22677025	-15,67825324	-28,72702867
Svoris Nr. 6	4,38780068	12,53062173	23,0265145
Svoris Nr. 7	-1,20669059	-2,41338829	-3,88286243
Svoris Nr. 8	2,45117457	6,0183312	11,39658813
Svoris Nr. 9	-3,253517	-8,43563732	-16,63421389

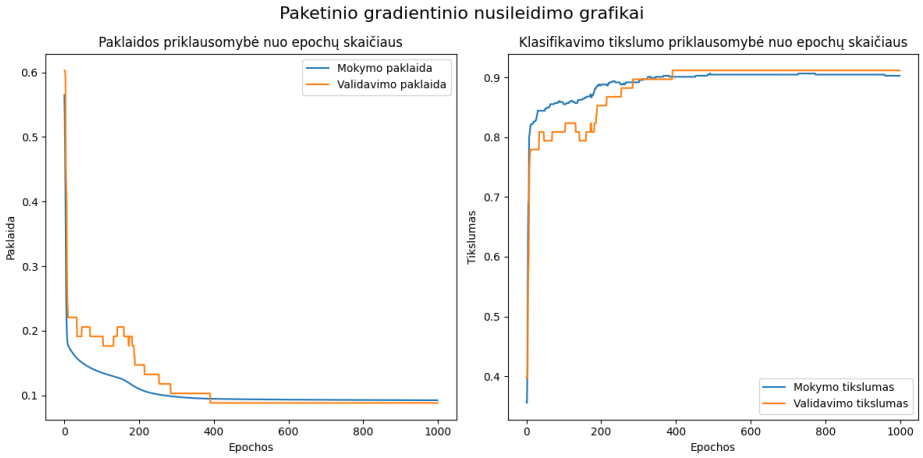
3.4. Gradientinio nusileidimo pasirinkto metodo įtaka rezultatams

Lyginant algoritmus, kai mokymosi greitis yra lygus 0,05 (5 pav. ir 7 pav.) galima matyti, kad abu algoritmai yra gan stabilūs, jei rinkamės didesnį epochų kiekį geriau rinktis stochastinį gradientinį metodą. Pasirinkus mažai epochų svyravimai, kurie nutinka naudojantis šiuo metodu gali mažinti rezultatų teisingumą. Kitu atveju, geriau rinktis paketinį gradientinį nusileidimo metodą.

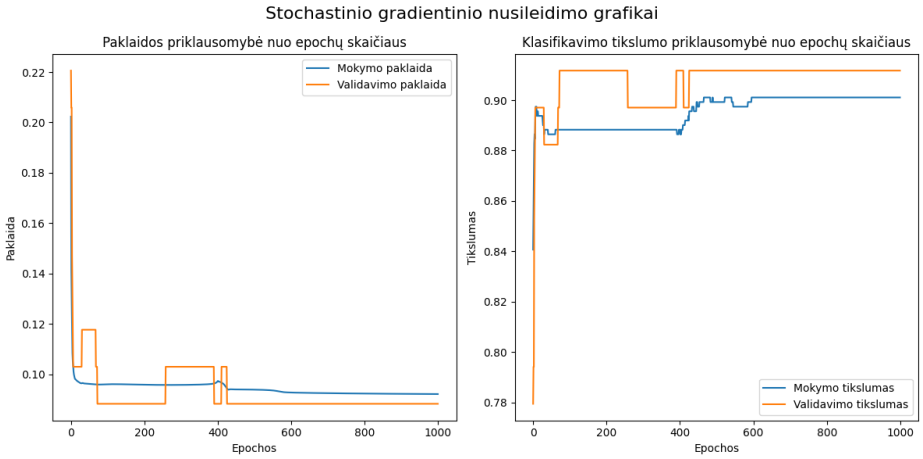
Jei lyginami algoritmai, kai mokymosi greitis yra lygus 0,5 (6 pav. ir 8 pav.) aiškiai matoma, kad geriau rinktis paketinį gradientinį metodą, nes jo rezultatai stabilesni, t. y. tikslesni.



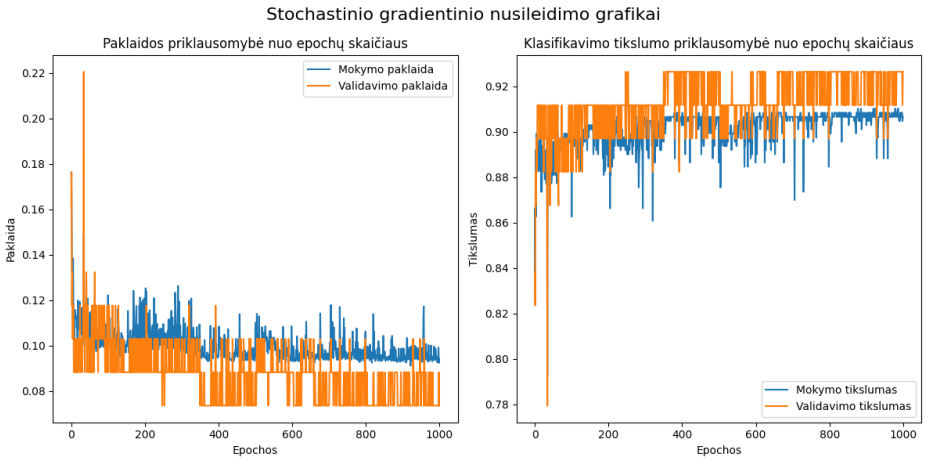
5 pav. Paketinio gradientinio nusileidimo grafikai, kai mokymosi greitis = 0,05.



6 pav. Paketinio gradientinio nusileidimo grafikai, kai mokymosi greitis = 0,5.



7 pav. Stochastinio gradientinio nusileidimo grafikai, kai mokymosi greitis = 0,05.



8 pav. Stochastinio gradientinio nusileidimo grafikai, kai mokymosi greitis = 0,5.

3.5. Taikomo gradientinio nusileidimo pasirinkto metodo įtaka mokymo laikui (esant vienodui epochų skaičiui)

7 lentelėje, 8 lentelėje, 9 lentelėje vaizduojama taikomo gradientinio nusileidimo įtaka mokymo laikui esant vienodam epochų skaičiui. Matoma, jog paketinis gradientinis nusileidimas yra greitesnis algoritmas, kai epochų skaičius yra mažesnis. Kai naudojame didelį epochų kiekį stochastinis metodas tampa spartesnis.

7 lentelė. Algoritmų trukmė, kai epochų sk. = 100.

Mokymas paketiniu gradientiniu nusileidimu Nr. 1	0,6538
Mokymas stochastiniu gradientiniu nusileidimu Nr. 1	0,8480

8 lentelė. Algoritmų trukmė, kai epochų sk. = 1000.

Mokymas paketiniu gradientiniu nusileidimu Nr. 2	6,5121
Mokymas stochastiniu gradientiniu nusileidimu Nr. 2	8,1735

9 lentelė. Algoritmų trukmė, kai epochų sk. = 10000.

Mokymas paketiniu gradientiniu nusileidimu Nr. 3	82,2357
Mokymas stochastiniu gradientiniu nusileidimu Nr. 3	78,6537

4. REZULTATAI

4.1. Dirbtinio intelekto įrankių indėlis

Dirbtinio intelekto įrankis ChatGPT padėjo duomenų pirminiam apdorojimui bei algoritmų vizualizavimui.

4.2. Išvados

Stochastinis gradientinis nusileidimas yra greitesnis nei paketinis metodas, tačiau jo sprendiniai mažiau stabilūs (optimalus epochų skaičius: 426 stochastiniam gradientiniam nusileidimui, 681 – paketiniam). Paketinis gradientinis nusileidimas užtikrina stabilesnius rezultatus, tačiau mokymo trukmė ilgesnė (paklaida validavimo duomenims: 0,088235 stochastiniam gradientiniam nusileidimui, 0,135010 – paketiniam). Mokymosi greičio didinimas (nuo 0,05 iki 0,5) mažina paklaidą ir didina tikslumą (validavimo tikslumas: iš 0,8088 į 0,9118), tačiau gali lemti nestabilumą. Didėjant epochų skaičiui, stochastinis metodas išlieka greitesnis (10 000 epochų: 78,6537 sek. stochastiniam gradientiniam nusileidimui, 82,2357 sek. – paketiniam). Paketinis metodas tinka tiksliems ir stabiliems modeliams, o stochastinis – greitam mokymui, kai sprendinių stabilumas nėra prioritetas.

5. PRIEDAI

Šiame skyriuje pateikiamas programos kodas su komentarais bei paaiškinimais.

5.1. Duomenų paruošimas

Šiame poskyriuje pateikiamas duomenų paruošimo kodas su komentarais bei paaiškinimais.

```
import random

def pasalinti_id_stulpeli(duomenys):
    """Funkcija, kuri pašalina ID stulpelį (pirmą stulpelį)."""
    return [linija[1:] for linija in duomenys]

def atnaujinti_klasiu_zymes(duomenys):
    """Funkcija, kuri keičia klasių žymes: 2 -> 0, 4 -> 1."""
    for linija in duomenys:
        if linija[-1] == '2':  # Nepiktybinis (benign)
            linija[-1] = '0'
        elif linija[-1] == '4':  # Piktybinis (malignant)
            linija[-1] = '1'
    return duomenys

def pasalinti_trūkstamus_duomenis(linijos):
    """Funkcija, kuri pašalina eilutes, kuriose yra '?'."""
    isvalytos_linijos = []
    for linija in linijos:
        stulpeliai = linija.strip().split(',')
        if '?' not in stulpeliai:
            isvalytos_linijos.append(stulpeliai)
    return isvalytos_linijos

def ismaisyti_duomenis(duomenys):
    """Funkcija, kuri išmaišo eilutes atsitiktine tvarka."""
    random.shuffle(duomenys)
    return duomenys

def issaugoti_i_faila(duomenys, isvesties_failas):
    """Funkcija, kuri išsaugo duomenis į naują failą."""
    with open(isvesties_failas, 'w') as failas:
        for linija in duomenys:
            failas.write(','.join(linija) + '\n')

def padalinti_duomenis(duomenys, mokymo_procentai=80, validavimo_procentai=10):
    """Funkcija, kuri padalina duomenis į mokymo, validavimo ir testavimo aibes."""
    eiluciu_skaicius = len(duomenys)
    mokymo_indeksas = int(eiluciu_skaicius * mokymo_procentai / 100)
    validavimo_indeksas = mokymo_indeksas + int(eiluciu_skaicius *
validavimo_procentai / 100)
    mokymo_duomenys = duomenys[:mokymo_indeksas]                              # 80:
    validavimo_duomenys = duomenys[mokymo_indeksas:validavimo_indeksas]       # :10:
    testavimo_duomenys = duomenys[validavimo_indeksas:]                       # :10
    return mokymo_duomenys, validavimo_duomenys, testavimo_duomenys

def main(ivesties_failas, isvesties_failas):
    """Pagrindinė programa."""

    # 1. Nuskaityti duomenis iš failo
    with open(ivesties_failas, 'r') as failas:
        linijos = failas.readlines()
    # 2. Pašalinti eilutes su trūkstamais duomenimis ('?')
    isvalyti_duomenys = pasalinti_trūkstamus_duomenis(linijos)
    # 3. Pašalinti ID stulpelį
```

```
    duomenys_be_id = pasalinti_id_stulpeli(isvalyti_duomenys)
    # 4. Atnaujinti klasių žymes
    atnaujinti_duomenys = atnaujinti_klasiu_zymes(duomenys_be_id)
    # 5. Išmaišyti eilutes atsitiktine tvarka
    ismaisyti_duomenys = ismaisyti_duomenis(atnaujinti_duomenys)
    # 6. Išsaugoti visus duomenis į vieną failą (esamas funkcionalumas)
    issaugoti_i_faila(ismaisyti_duomenys, isvesties_failas)

    # 7. Papildomas funkcionalumas: padalinti duomenis ir išsaugoti į tris failus
    mokymo_duomenys, validavimo_duomenys, testavimo_duomenys =
padalinti_duomenis(ismaisyti_duomenys)
    issaugoti_i_faila(mokymo_duomenys, '2uzd_MokymoDuomenys_DIP.data')
    issaugoti_i_faila(validavimo_duomenys, '2uzd_ValidavimoDuomenys_DIP.data')
    issaugoti_i_faila(testavimo_duomenys, '2uzd_TestavimoDuomenys_DIP.data')

    print(f"Duomenys sėkmingai apdoroti ir išsaugoti į {isvesties_failas}")
    print("Papildomai duomenys išskirstyti į tris failus:")
    print("- 2uzd_MokymoDuomenys_DIP.data (mokymo duomenys)")
    print("- 2uzd_ValidavimoDuomenys_DIP.data (validavimo duomenys)")
    print("- 2uzd_TestavimoDuomenys_DIP.data (testavimo duomenys)")

if __name__ == "__main__":
    """Pagrindinės programos paleidimas."""
    ivesties_failas = 'breast-cancer-wisconsin.data'
    isvesties_failas = '2uzd_ParuostiDuomenys_DIP.data'
    main(ivesties_failas, isvesties_failas)
```

5.2. Algoritmų įgyvendinimas

5.2.1. Paketinis gradientinis nusileidimas

Šiame poskyriuje pateikiamas kodas, kuriame įgyvenidinamas dirbtinio neurono mokymas, validavimas bei testavimas paketinio gradientinio nusileidimo metodu.

```
import numpy as np
import pandas as pd
import time
from tyrimas_2uzd_DIP import paketinio_gradientinio_nusileidimo_grafikai

def sigmoidine_funkcija(z):
    """Sigmoidinė funkcija."""
    return 1 / (1 + np.exp(-z))

def nuskaityti_duomenis(failas):
    """Duomenų nuskaitymas iš failo."""
    df = pd.read_csv(failas, header=None)
    pozymiai = df.iloc[:, :-1].values
    klase = df.iloc[:, -1].values
    return pozymiai, klase

def a_apskaiciavimas(pozymiai, svoriai, poslinkis):
    """Aktyvacijos funkcijos įėjimo reikšmės apskaičiavimas."""
    return np.dot(pozymiai, svoriai) + poslinkis

def paketinis_gradientinis_nusileidimas(mokymo_pozymiai, mokymo_klase,
mokymosi_greitis=0.05, epochos=10000, mokymo_riba=1e-6, validavimo_pozymiai=None,
validavimo_klase=None):
    """Sigmoidinio neurono mokymas gradientinį taikant paketinį gradientinį
nusileidimą."""
    np.random.seed(150)
    m, n = mokymo_pozymiai.shape
    svoriai = np.random.randn(n)
    poslinkis = np.random.randn()
    totalError = float('inf')
    epocha = 0
```

```
    mokymo_totalErrors_grafikui = []
    validavimo_totalErrors_grafikui = []
    mokymo_tikslumas_grafikui = []
    validavimo_tikslumas_grafikui = []

    pradzios_laikas = time.time()

    while totalError > mokymo_riba and epocha < epochos:
        totalError = 0
        gradientSum = np.zeros(n)

        for i in range(m):
            z = a_apskaiciavimas(mokymo_pozymiai[i], svoriai, poslinkis)
            yi = sigmoidine_funkcija(z)
            ti = mokymo_klase[i]

            for k in range(n):
                gradientSum[k] += (yi - ti) * yi * (1 - yi) * mokymo_pozymiai[i, k]

            error = (ti - yi) ** 2
            totalError += error
            totalError_avg = totalError / (i + 1)

        for k in range(n):
            svoriai[k] -= mokymosi_greitis * (gradientSum[k] / m)

        epocha += 1

        mokymo_prognoze = np.round(sigmoidine_funkcija(np.dot(mokymo_pozymiai,
svoriai) + poslinkis))
        mokymo_tikslumas = np.mean(mokymo_prognoze == mokymo_klase)
        mokymo_totalErrors_grafikui.append(totalError_avg)
        mokymo_tikslumas_grafikui.append(mokymo_tikslumas)

        validavimo_pradzios_laikas = time.time()
        # Klasifikavimo tikslumo ir paklaidos skaičiavimas su validavimo duomenimis
        if validavimo_pozymiai is not None and validavimo_klase is not None:
            validavimo_prognoze =
np.round(sigmoidine_funkcija(np.dot(validavimo_pozymiai, svoriai) + poslinkis))
            validavimo_tikslumas = np.mean(validavimo_prognoze == validavimo_klase)
            validavimo_paklaida = np.mean((validavimo_klase - validavimo_prognoze) **
2)
            print(f"Epocha {epocha}: Paklaida mokymo metu = {totalError_avg:.6f},
Tikslumas mokymo metu = {mokymo_tikslumas:.4f}, Paklaida validavimo metu =
{validavimo_paklaida:.6f}, Tikslumas validavimo metu = {validavimo_tikslumas:.4f}")
        else:
            print(f"Epocha {epocha}: Paklaida mokymo metu = {totalError_avg:.6f},
Tikslumas mokymo metu = {mokymo_tikslumas:.4f}")
        validavimo_pabaigos_laikas = time.time()
        bendras_validavimo_laikas = validavimo_pabaigos_laikas -
validavimo_pradzios_laikas
        validavimo_totalErrors_grafikui.append(validavimo_paklaida)
        validavimo_tikslumas_grafikui.append(validavimo_tikslumas)

    pabaigos_laikas = time.time()
    bendras_laikas = pabaigos_laikas - pradzios_laikas
    bendras_mokymo_laikas = bendras_laikas - bendras_validavimo_laikas
    print(f"\nNeurono mokymo laikas: {bendras_mokymo_laikas:.4f} sekundės")

    return svoriai, poslinkis, mokymo_totalErrors_grafikui,
validavimo_totalErrors_grafikui, mokymo_tikslumas_grafikui,
validavimo_tikslumas_grafikui

def testavimas(testavimo_pozymiai, testavimo_klase, svoriai, poslinkis):
    """Tikslumas ir paklaida su testavimo duomenimis."""
```

```
    testavimo_prognoze = np.round(sigmoidine_funkcija(np.dot(testavimo_pozymiai,
svoriai) + poslinkis))
    testavimo_tikslumas = np.mean(testavimo_prognoze == testavimo_klase)
    testavimo_paklaida = np.mean((testavimo_klase - testavimo_prognoze) ** 2)
    return testavimo_tikslumas, testavimo_paklaida

def main():
    """Pagrindinė funkcija main."""
    mokymo_duomenys = "2uzd_MokymoDuomenys_DIP.data"
    validavimo_duomenys = "2uzd_ValidavimoDuomenys_DIP.data"
    testavimo_duomenys = "2uzd_TestavimoDuomenys_DIP.data"

    # Epochų skaičiaus ir mokymo greičio įvedimas
    epochos = int(input("Įveskite epochų skaičių: "))
    mokymosi_greitis = float(input("Įveskite mokymo greitį (nuo 0 iki 1): "))

    # Duomenų nuskaitymas
    mokymo_pozymiai, mokymo_klase = nuskaityti_duomenis(mokymo_duomenys)
    validavimo_pozymiai, validavimo_klase = nuskaityti_duomenis(validavimo_duomenys)
    testavimo_pozymiai, testavimo_klase = nuskaityti_duomenis(testavimo_duomenys)

    # Modelio mokymas su mokymo duomenimis ir patikrinimas su validavimo duomenimis
    svoriai, poslinkis, mokymo_totalErrors_grafikui, validavimo_totalErrors_grafikui,
mokymo_tikslumas_grafikui, validavimo_tikslumas_grafikui =
paketinis_gradientinis_nusileidimas(
        mokymo_pozymiai, mokymo_klase, mokymosi_greitis=mokymosi_greitis,
epochos=epochos, validavimo_pozymiai=validavimo_pozymiai,
validavimo_klase=validavimo_klase
    )

    # Kreipiamasis į funkciją grafikų generavimui
    paketinio_gradientinio_nusileidimo_grafikai(range(epochos),
mokymo_totalErrors_grafikui, validavimo_totalErrors_grafikui,
mokymo_tikslumas_grafikui, validavimo_tikslumas_grafikui)

    # Rezultatų išvedimas su testavimo duomenimis
    testavimo_tikslumas, testavimo_paklaida = testavimas(testavimo_pozymiai,
testavimo_klase, svoriai, poslinkis)
    print(f"Paklaida testavimo metu = {testavimo_tikslumas:.6f}, Tikslumas testavimo
metu = {testavimo_paklaida:.4f}\n")

if __name__ == "__main__":
    """Pagrindinės funkcijos paleidimas."""
    main()
```

5.2.2. Stochastinis gradientinis nusileidimas

Šiame poskyriuje pateikiamas kodas, kuriame įgyvenidinamas dirbtinio neurono mokymas, validavimas bei testavimas stochastinio gradientinio nusileidimo metodu.

```
import numpy as np
import pandas as pd
import time
from tyrimas_2uzd_DIP import stochastinio_gradientinio_nusileidimo_grafikai

def sigmoidine_funkcija(z):
    """Sigmoidinė funkcija."""
    return 1 / (1 + np.exp(-z))

def nuskaityti_duomenis(failas):
    """Duomenų nuskaitymas iš failo."""
    df = pd.read_csv(failas, header=None)
    pozymiai = df.iloc[:, :-1].values
    klase = df.iloc[:, -1].values
    return pozymiai, klase
```

```
def a_apskaiciavimas(pozymiai, svoriai, poslinkis):
    """Aktyvacijos funkcijos įėjimo reikšmės apskaičiavimas."""
    return np.dot(pozymiai, svoriai) + poslinkis

def stochastinis_gradientinis_nusileidimas(mokymo_pozymiai, mokymo_klase,
mokymosi_greitis=0.05, epochos=10000, mokymo_riba=1e-6, validavimo_pozymiai=None,
validavimo_klase=None):
    """Sigmoidinio neurono mokymas taikant stochastinį gradientinį nusileidimą."""
    np.random.seed(150)
    m, n = mokymo_pozymiai.shape
    svoriai = np.random.randn(n)
    poslinkis = np.random.randn()
    totalError = float('inf')
    epocha = 0

    mokymo_totalErrors_grafikui = []
    validavimo_totalErrors_grafikui = []
    mokymo_tikslumas_grafikui = []
    validavimo_tikslumas_grafikui = []

    pradzios_laikas = time.time()

    while totalError > mokymo_riba and epocha < epochos:
        totalError = 0

        for i in range(m):
            z = a_apskaiciavimas(mokymo_pozymiai[i], svoriai, poslinkis)
            yi = sigmoidine_funkcija(z)
            ti = mokymo_klase[i]

            for k in range(n):
                svoriai[k] -= mokymosi_greitis * (yi - ti) * yi * (1 - yi) *
mokymo_pozymiai[i, k]

            error = (ti - yi) ** 2
            totalError += error

        totalError_avg = totalError / m

        epocha += 1

        mokymo_prognoze = np.round(sigmoidine_funkcija(np.dot(mokymo_pozymiai,
svoriai) + poslinkis))
        mokymo_tikslumas = np.mean(mokymo_prognoze == mokymo_klase)
        mokymo_totalErrors_grafikui.append(totalError_avg)
        mokymo_tikslumas_grafikui.append(mokymo_tikslumas)

        validavimo_pradzios_laikas = time.time()
        # Klasifikavimo tikslumo ir paklaidos skaičiavimas su validavimo duomenimis
        if validavimo_pozymiai is not None and validavimo_klase is not None:
            validavimo_prognoze =
np.round(sigmoidine_funkcija(np.dot(validavimo_pozymiai, svoriai) + poslinkis))
            validavimo_tikslumas = np.mean(validavimo_prognoze == validavimo_klase)
            validavimo_paklaida = np.mean((validavimo_klase - validavimo_prognoze) **
2)
            print(f"Epocha {epocha}: Paklaida mokymo metu = {totalError_avg:.6f},
Tikslumas mokymo metu = {mokymo_tikslumas:.4f}, Paklaida validavimo metu =
{validavimo_paklaida:.6f}, Tikslumas validavimo metu = {validavimo_tikslumas:.4f}")
        else:
            print(f"Epocha {epocha}: Paklaida mokymo metu = {totalError_avg:.6f},
Tikslumas mokymo metu = {mokymo_tikslumas:.4f}")
        validavimo_pabaigos_laikas = time.time()
        bendras_validavimo_laikas = validavimo_pabaigos_laikas -
validavimo_pradzios_laikas
        validavimo_totalErrors_grafikui.append(validavimo_paklaida)
        validavimo_tikslumas_grafikui.append(validavimo_tikslumas)
```

```
    pabaigos_laikas = time.time()
    bendras_laikas = pabaigos_laikas - pradzios_laikas
    bendras_mokymo_laikas = bendras_laikas - bendras_validavimo_laikas
    print(f"\nNeurono mokymo laikas: {bendras_mokymo_laikas:.4f} sekundės")

    return svoriai, poslinkis, mokymo_totalErrors_grafikui,
validavimo_totalErrors_grafikui, mokymo_tikslumas_grafikui,
validavimo_tikslumas_grafikui

def testavimas(testavimo_pozymiai, testavimo_klase, svoriai, poslinkis):
    """Tikslumas ir paklaida su testavimo duomenimis."""
    testavimo_prognoze = np.round(sigmoidine_funkcija(np.dot(testavimo_pozymiai,
svoriai) + poslinkis))
    testavimo_tikslumas = np.mean(testavimo_prognoze == testavimo_klase)
    testavimo_paklaida = np.mean((testavimo_klase - testavimo_prognoze) ** 2)
    return testavimo_tikslumas, testavimo_paklaida

def main():
    """Pagrindinė funkcija main."""
    mokymo_duomenys = "2uzd_MokymoDuomenys_DIP.data"
    validavimo_duomenys = "2uzd_ValidavimoDuomenys_DIP.data"
    testavimo_duomenys = "2uzd_TestavimoDuomenys_DIP.data"

    # Epochų skaičiaus ir mokymo greičio įvedimas
    epochos = int(input("Įveskite epochų skaičių: "))
    mokymosi_greitis = float(input("Įveskite mokymo greitį (nuo 0 iki 1): "))

    # Duomenų nuskaitymas
    mokymo_pozymiai, mokymo_klase = nuskaityti_duomenis(mokymo_duomenys)
    validavimo_pozymiai, validavimo_klase = nuskaityti_duomenis(validavimo_duomenys)
    testavimo_pozymiai, testavimo_klase = nuskaityti_duomenis(testavimo_duomenys)

    # Modelio mokymas su mokymo duomenimis ir patikrinimas su validavimo duomenimis
    svoriai, poslinkis, mokymo_totalErrors_grafikui, validavimo_totalErrors_grafikui,
mokymo_tikslumas_grafikui, validavimo_tikslumas_grafikui =
stochastinis_gradientinis_nusileidimas(
        mokymo_pozymiai, mokymo_klase, mokymosi_greitis=mokymosi_greitis,
epochos=epochos, validavimo_pozymiai=validavimo_pozymiai,
validavimo_klase=validavimo_klase
    )

    # Kreipiamasis į funkciją grafikų generavimui
    stochastinio_gradientinio_nusileidimo_grafikai(range(epochos),
mokymo_totalErrors_grafikui, validavimo_totalErrors_grafikui,
mokymo_tikslumas_grafikui, validavimo_tikslumas_grafikui)

    # Rezultatų išvedimas su testavimo duomenimis
    testavimo_tikslumas, testavimo_paklaida = testavimas(testavimo_pozymiai,
testavimo_klase, svoriai, poslinkis)
    print(f"Paklaida testavimo metu = {testavimo_paklaida:.6f}, Tikslumas testavimo
metu = {testavimo_tikslumas:.4f}\n")

if __name__ == "__main__":
    """Pagrindinės funkcijos paleidimas."""
    main()
```

5.3. Tyrimo atlikimas (grafikų braižymas)

Šiame poskyriuje pateikiamas grafikų braižymo kodas su komentarais bei paaiškinimais.

```
import matplotlib.pyplot as plt

def paketinio_gradientinio_nusileidimo_grafikai(epochos, mokymo_paklaidos,
validavimo_paklaidos, mokymo_tikslumas, validavimo_tikslumas):
    """Paketinio gradientinio nusileidimo modelio grafikų generavimas"""
```

```
    plt.figure(figsize=(12, 6))

    plt.suptitle('Paketinio gradientinio nusileidimo grafikai', fontsize=16)

    # Paklaidos grafikas
    plt.subplot(1, 2, 1)
    plt.plot(epochos, mokymo_paklaidos, label='Mokymo paklaida')
    plt.plot(epochos, validavimo_paklaidos, label='Validavimo paklaida')
    plt.xlabel('Epochos')
    plt.ylabel('Paklaida')
    plt.title('Paklaidos priklausomybė nuo epochų skaičiaus')
    plt.legend()

    # Klasifikavimo tikslumo grafikas
    plt.subplot(1, 2, 2)
    plt.plot(epochos, mokymo_tikslumas, label='Mokymo tikslumas')
    plt.plot(epochos, validavimo_tikslumas, label='Validavimo tikslumas')
    plt.xlabel('Epochos')
    plt.ylabel('Tikslumas')
    plt.title('Klasifikavimo tikslumo priklausomybė nuo epochų skaičiaus')
    plt.legend()

    plt.tight_layout()
    plt.savefig('paketinis_gradientinis_nusileidimas.png')
    plt.show()

def stochastinio_gradientinio_nusileidimo_grafikai(epochos, mokymo_paklaidos,
validavimo_paklaidos, mokymo_tikslumas, validavimo_tikslumas):
    """Paketinio gradientinio nusileidimo modelio grafikų generavimas"""
    plt.figure(figsize=(12, 6))

    plt.suptitle('Paketinio gradientinio nusileidimo grafikai', fontsize=16)

    # Paklaidos grafikas
    plt.subplot(1, 2, 1)
    plt.plot(epochos, mokymo_paklaidos, label='Mokymo paklaida')
    plt.plot(epochos, validavimo_paklaidos, label='Validavimo paklaida')
    plt.xlabel('Epochos')
    plt.ylabel('Paklaida')
    plt.title('Paklaidos priklausomybė nuo epochų skaičiaus')
    plt.legend()

    # Klasifikavimo tikslumo grafikas
    plt.subplot(1, 2, 2)
    plt.plot(epochos, mokymo_tikslumas, label='Mokymo tikslumas')
    plt.plot(epochos, validavimo_tikslumas, label='Validavimo tikslumas')
    plt.xlabel('Epochos')
    plt.ylabel('Tikslumas')
    plt.title('Klasifikavimo tikslumo priklausomybė nuo epochų skaičiaus')
    plt.legend()

    plt.tight_layout()
    plt.savefig('stohastinis_gradientinis_nusileidimas.png')
    plt.show()
```

5.4. Testavimo duomenų įrašų klasės

5.4.1. Paketinio gradientinio nusileidimo algoritmo atveju

```
Testavimo įrašas 1: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 2: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 3: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 4: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 5: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 6: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 7: Nustatyta klasė = 0.0, Tikroji klasė = 0
```

```
Testavimo įrašas 8: Nustatyta klasė = 0.0, Tikroji klasė = 1
Testavimo įrašas 9: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 10: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 11: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 12: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 13: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 14: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 15: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 16: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 17: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 18: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 19: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 20: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 21: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 22: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 23: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 24: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 25: Nustatyta klasė = 0.0, Tikroji klasė = 1
Testavimo įrašas 26: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 27: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 28: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 29: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 30: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 31: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 32: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 33: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 34: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 35: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 36: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 37: Nustatyta klasė = 0.0, Tikroji klasė = 1
Testavimo įrašas 38: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 39: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 40: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 41: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 42: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 43: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 44: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 45: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 46: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 47: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 48: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 49: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 50: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 51: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 52: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 53: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 54: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 55: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 56: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 57: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 58: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 59: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 60: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 61: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 62: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 63: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 64: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 65: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 66: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 67: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 68: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 69: Nustatyta klasė = 1.0, Tikroji klasė = 1
```

5.4.2. Stochastinio gradientinio nusileidimo algoritmo atveju

```
Testavimo įrašas 1: Nustatyta klasė = 1.0, Tikroji klasė = 1
```

```
Testavimo įrašas 2: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 3: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 4: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 5: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 6: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 7: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 8: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 9: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 10: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 11: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 12: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 13: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 14: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 15: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 16: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 17: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 18: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 19: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 20: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 21: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 22: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 23: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 24: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 25: Nustatyta klasė = 0.0, Tikroji klasė = 1
Testavimo įrašas 26: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 27: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 28: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 29: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 30: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 31: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 32: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 33: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 34: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 35: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 36: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 37: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 38: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 39: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 40: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 41: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 42: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 43: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 44: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 45: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 46: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 47: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 48: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 49: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 50: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 51: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 52: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 53: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 54: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 55: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 56: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 57: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 58: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 59: Nustatyta klasė = 1.0, Tikroji klasė = 0
Testavimo įrašas 60: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 61: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 62: Nustatyta klasė = 1.0, Tikroji klasė = 1
Testavimo įrašas 63: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 64: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 65: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 66: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 67: Nustatyta klasė = 0.0, Tikroji klasė = 0
```

```
Testavimo įrašas 68: Nustatyta klasė = 0.0, Tikroji klasė = 0
Testavimo įrašas 69: Nustatyta klasė = 1.0, Tikroji klasė = 1
```