VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Mindaugas Gaidys

KARO NUSIKALTIMŲ NUSTATYMAS, TAIKANT MAŠININIO MOKYMOSI ALGORITMUS
IDENTIFYING WAR CRIMES USING MACHINE LEARNING ALGORITHMS

Baigiamasis bakalauro darbas

Programų inžinerijos studijų programa, valstybinis kodas 6121BX023
Programų sistemų studijų kryptis

Vilnius, 2023

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Mindaugas Gaidys

KARO NUSIKALTIMŲ NUSTATYMAS, TAIKANT MAŠININIO MOKYMOSI ALGORITMUS
IDENTIFYING WAR CRIMES USING MACHINE LEARNING ALGORITHMS

Baigiamasis bakalauro darbas

Programų inžinerijos studijų programa, valstybinis kodas 6121BX023
Programų sistemų studijų kryptis

Vadovas Rokas Štrimaitis
(Pedag. vardas, vardas, pavardė)

Konsultantas _____
(Pedag. vardas, vardas, pavardė)

Konsultantas _____
(Pedag. vardas, vardas, pavardė)

Vilnius, 2023

TURINYS

Paveikslėlių sąrašas

Lentelių sąrašas

Terminų žodynas

RNN (angl. *Recurrent neural network*) – rekurentinis neuroninis tinklas.

LSTM (angl. *Long short-term memory*) – rekurentinio neuroninio tinklo architektūra.

MLP (ang. *Multilayer perceptron*) daugiasluoksnis perceptronas.

ReLU (angl. *Rectified linier unit*) – aktyvacijos funkcija pakeičianti neigiamas reikšmes į 0.

NN (angl. *Neural network*) – neuroninis tinklas.

ML (angl. *Machine learning*) – mašininis mokymasis.

Įvadas

Karas yra vienas labiausiai visuomenę galinčių paveikti geopolitinių veiksnių. Siekiant išvengti arba minimalizuoti karo sukeltus padarinius, valstybės deda milžiniškas pastangas. Vienas svarbiausių įrankių šioje kovoje yra informacija. Pasitelkus tinkamą informaciją galima iš anksto nuspėti agresoriaus veiksmus ir kovoti su jo skleidžiama propaganda. Nors informacija yra itin naudinga, tačiau šiame amžiuje jos kiekiai yra be galo dideli, ko padarinyje jos apdorojimo kaštai reikalauja beprotiškai daug žmogiškųjų resursų. Dėl šios priežasties naujausios karinės analitinės sistemos dažniausiai naudoja itin aukšto lygio dirbtinius neuroninius tinklus, kurie gali būti apmokyti atpažinti specifinius objektus palydovinėse nuotraukose, filtruoti komunikacijas arba socialinių tinklų įrašus pagal pasirinktus raktinius žodžius ir ieškoti pasikartojančių bruožų, kurie yra būdingi stebimam objektui[20]. To padarinyje žmogiškieji resursai gali būti nukreipiami į strateginių veiksmų planavimą, pagal apdorotą informaciją.

Nors karinės institucijos, pasitelkdamos DI gali efektyviai išgauti informaciją, dažniausiai tik maža dalis šios informacijos yra atskleidžiama visuomenei. Kadangi informaciją galima laikyti galios forma, netinkamas jos atskleidimas gali suteikti priešiškoms valstybėms pranašumą. Dėl šios priežasties, geopolitinių konfliktų metu, dažniausiai sukuriamas „karo rūkas“. Tai reiškia, kad karinės įstaigos nustoja dalintis savo turima informacija ir vienintelis būdas visuomenei išlikti informuotai lieka žiniasklaida. Nors žiniasklaidos teikiama informacija yra itin svarbi kiekvienam piliečiui, tačiau tokio pobūdžio sklaida turi ir neigiamų veiksnių, tokių kaip potencialios propagandos sklaida bei netikslingas ir lėtas informacijos perteikimas. Siekiant išvengti šių nepageidautinų reiškinių kyla poreikis, pačiai visuomenei, filtruoti gaunamą informaciją [20]. Tai ypač aktualu tokiai informacijai kaip potencialūs karo nusikaltimai. Valstybinės įstaigos šiuos duomenis atskleidžia visuomenei santykinai lėtai, nes nori užtikrinti maksimalų jų tikslumą, o žiniasklaida, kaip ir minėta ankščiau gali būti nepatikima, o greitas karo nusikaltimų atskleidimas gali suburti žmones siekiančius juos sustabdyti arba suteikti paramą aukoms. Dėl šių priežasčių juntamas poreikis modeliui, gebančiam efektyviai nustatyti ar specifiniai įvykiai gali būti laikomi karo nusikaltimais. Siekiant užtikrinti šios sistemos nešališkumą ir tikslumą yra poreikis pritaikyti DI.

Darbo objektas – mašininio mokymosi modeliai skirti duomenų klasifikavimui.

Darbo tikslas – nustatyti karo nusikaltimus tekstiniuose dokumentuose, taikant mašininio mokymosi klasifikavimo modelius.

Užduotys

1. Atlikti lyginamąją mašininio mokymosi tekstinių duomenų klasifikavimo modelių analizę.
2. Atlikti, jau egzistuojančių, technologijų , skirtų duomenų surinkimui ir modelių įgyvendinimui, analizę.
3. Surinkti duomenis ir jais apmokyti klasifikavimo modelius.
4. Atlikti gautų rezultatų analizę.

Naujumas

Šiame darbe kuriami modeliai bus treniruojami naudojantis Ukrainoje vykstančios geopolitinės krizės duomenimis. Taip pat, šio darbo įgyvendinimo metu, įgyvendinti mašininio mokymosi algoritmai sugeneravo naudingas svorines matricas, kurios gali būti naudojamos kituose karinio konteksto darbuose kaip pradinės vertės.

Praktinė vertė

Nustatyti tam tikru tikslumu ar karinis įvykis, geopolitinės krizės metu, gali būti laikomas karo nusikaltimu. Gauta informacija gali būti taikoma visuomenės švietimui, palaikymo sutelkimui ar propagandos aptikimui.

Darbo struktūra

Bakalauro baigiamąjį darbą sudaro įvadas, 1 skyrius ir išvados. Darbo apimtis – 51 puslapis, tekste yra 19 paveikslėlių ir 6 lentelės. Rengiant darbą buvo panaudota 21 literatūros šaltinis.

Įvade pristatomas darbo aktualumas, problematika, formuluojamas darbo tikslas bei uždaviniai.

Pirmame skyriuje apžvelgiamas ML, nagrinėjami klasifikavimo algoritmai ir jų veikimo principas, palyginamos technologijos tinkančios sentimentų analizei ir klasifikavimui skirtų algoritmų įgyvendinimui bei su Europos žemyne vykstančia geopolitine krize susijusių duomenų išgavimui.

Antrame skyriuje yra pateikiama tyrumo eiga, duomenų surinkimas ir apdorojimas. Pateikiami karo nusikaltimų apibrėžimai ir jų klasifikacijos taisyklės. Taip pat aprašomas modelių paruošimas ir realizacija. Galiausiai pateikiamas visų modelių rezultatų palyginimas ir kitos tyrimo įžvalgos.

1. Analitinė dalis

1.1.Mašininis mokymasis

Mašininis mokymasis (angl. *machine learning*) – mokslas tiriantis gebėjimą mokytis iš duomenų juose aptinkant braižus ar savybes. Bendrinį mašininio mokymosi algoritmų veikimą galima apibūdinti trijų žingsnių seka:

- Pasirinktam algoritmui pateikiami duomenys.
- Įvertinamas gautų rezultatų teisingumas.
- Pritaikomas grįžtamojo ryšio mechanizmas, kuris pagal gauto rezultato nuokrypį, nustato veiksmus rezultatų tikslinimui.

Taikant ML vienas svarbiausių žingsnių yra nuspręsti kaip tinklas bus apmokomas. Pagal tai kokio žmogaus įsikišimo lygio mokymo procesas reikalauja, šis yra skirstomas į tris grupes : neprižiūrimas (angl. *unsupervised*), dalinai prižiūrimas (angl. *semi-supervised*) bei prižiūrimas (angl. *supervised*) [4]. Neprižiūrimam modeliui pateikiami duomenys neturi jokių klasifikuotų grupių, dėl to neuroninis tinklas pats turi atlikti klasifikaciją pagal pateikiamus parametrus. Dalinai prižiūrimo mokymosi atveju dalis duomenų yra klasifikuoti ir turi žymes, tačiau likusiai daliai duomenų žymės suteikiamos paties modelio. Galiausiai prižiūrimo mokymosi metu visi pateikiami duomenys turi žymes. Neprižiūrimo mokymosi modelio projektavimas reikalauja kur kas daugiau darbo nei prižiūrimo arba dalinai prižiūrimo modelio, tačiau išvengiama rankinio duomenų žymėjimo. Tuo tarpu prižiūrimas arba dalinai prižiūrimas modelis reikalauja didelių laiko kaštų tikslingam duomenų žymių priskyrimui, tačiau yra našesnis klasifikavimo uždaviniams spręsti. Šio darbo metu taikomas prižiūrimas mokymasis, todėl tolesniuose skyriuose bus analizuojamos tik su prižiūrimu mokymusi susijusios savybės.

Kadangi šiame darbe sprendžiami klasifikavimo uždaviniai, bus naudojamas mokymosi su mokytoju būdas ir kiti būdai nebus apžvelgiami.

Populiariausi klasifikavimo algoritmai:

- Bajeso klasifikatorius (angl. *Naive Bayes*),
- sprendimų medžiai (angl. *decision trees*),
- atraminių vektorių mašinos (angl. *support-vector machines*),
- neuroniniai tinklai.

1.1.1 Naiviojo Bajeso klasifikatorius

Naiviojo Bajeso klasifikatorių veikimas yra paremtas Bajeso teorema. Šie klasifikatoriai dažnai taikomi tekstinių duomenų sentimentų analizėms atlikti. Jie veikia priimdami dvi prielaidas. Pirmoji prielaida yra kad žodis, nepriklausomai nuo jo padėties tekste, turi tokį patį sentimentą. Antroji prielaida, dar vadinama naiviąja Bajeso prielaida, nurodo, kad savybės tikimybė, būti priskirtai konkrečiai klasei, yra nepriklausoma nuo kitų[1].

Dėl minėtų prielaidų, šis modelis yra mažiau tinkamas duomenų rinkiniams, kuriuose įrašai turi daugiau nei vieną sentimentą. Tokio pobūdžio įrašai, dažniausiai, būna didesnės apimties, toki kaip pastraipos arba ištisi tekstai.

Bendrinė Naiviojo Bajeso formule yra :

$$P(y|X) = \frac{P(X|y)P(y)}{\mathrm{P(X)}} \quad (1)$$

y: klasifikavimo klasė, *X*: įrašo savybių rinkinys, *P(y|X)*: klasės y tikimybė atsižvelgiant į duomenis, *P(X|y)*: savybių tikimybėje tam tikroje klasėje, *P(y)*: klasės tikimybė, *P(X)*: įrašo savybių rinkinio tikimybė.

Tuomet priėmus minėtas prielaidas norėdami nustatyti klasę pagal turimą duomenų savybių rinkinį, turime apskaičiuoti maksimalią tikimybę, kurią skaičiuojame pagal šią formulę:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \quad (2)$$

Šių klasifikatorių teigiamos savybės:

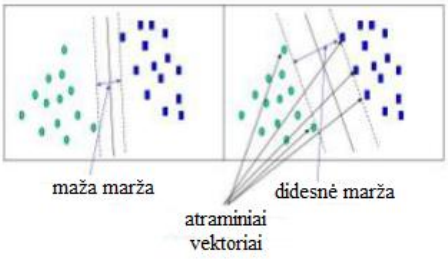
- Esant nepriklausomiems duomenims pasiekiamas aukštas tikslumas,
- puikiai tinkamas dirbti su teksto klasifikacija,
- greitai realizuojamas modelis.

Neigiamos savybe:

- Priklausomi nuo duomenų nepriklausomybės,
- neišlaiko konteksto.

1.1.2 Atraminių vektorių mašina

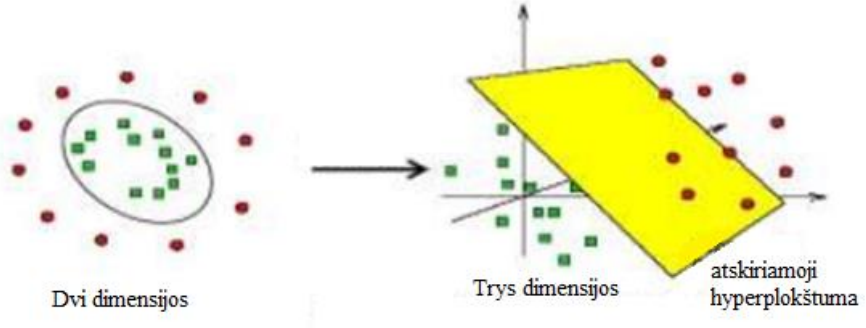
Šio algoritmo pagrindinė paskirtis yra atrasti ribas, kuriomis yra atskiriamos duomenų klasės. Tai atliekama perkeliant duomenis į plokštumas ir didinant šių plokštumų dimensijų kiekį, taip ieškant duomenų klasterių. Kadangi hiperplokštumų, kuriomis yra atskiriami duomenys įvairovė yra begalinė, pasirenkamos tokios plokštumos, kuriomis gaunamos didžiausios maržos tarp duomenų. Tuo tarpu duomenys, kurie yra arčiausiai minėtų plokštumų. vadinami atraminiais vektoriais[3]. 1 pav. Pavaizduotas pasirinkimas tarp atskirties plokštumų.



pav. 1 Atraminis vektorius ir jo marža[3]

Minėtu atveju buvo pateikti paprasčiausi tiesiniai duomenys, tačiau realiam pasaulyje, daugumoje atvejų, duomenys yra kur kas labiau kompleksiški ir negali būti atskirti tiesiomis linijomis. Tokiu atveju yra naudojamos *kernel* funkcijos. 2pav. pateikiama transformacija į aukštesnės dimensijos hiperplokštumas, taip atskiriant duomenis[3].

Commented [RŠ1]: Su kuo čia daryti abu pav, baisiai atrodo.



pav. 2 Atraminis vektorius ir marža[3]

Nors naudojant *kernel* funkcijas įgaunama galimybė sukurti duomenų atskirtį, tam atlikti funkcija turi būti pasirinkta tikslingai, o tai padaryti gali būti sudėtinga, jeigu klasifikuojami duomenys yra labai kompleksiški.

Atraminės vektorių mašinos privalumai:

- Aiški klasių atskirtis,
- lengva vizualizacija.

Trūkumai:

- Prastai veikia kada klasės negali būti aiškiai atskirtos,
- sudėtinga parinkti tinkamą *kernel* funkciją.

1.1.3 Sprendimų medžiai

Vienas pirmų bandymų imituoti žmogaus sprendimus buvo pasitelkiant sprendimų medžius. Bendru atveju juos galima apibūdinti kaip klausymų ar taisyklių rinkinį, kuris skaido duomenis pagal jų savybes. Tuomet pasirenkant konkrečią šaką ir surinkus joje nurodytas specifines savybes, priskirti jai klasę.

Mokslinėje literatūroje dažniausiai sutinkami yra trys pagrindiniai sprendimų medžių algoritmai: *ID3, CART* ir *CHAID* [4]. Esminis skirtumas tarp šių algoritmų yra tai kad *ID3* ir *CHAID* algoritmai yra ne dvejetainiai, tai reiškia gali sudaryti daugiau nei 2 šakas iš kamieno. Tuo tarpu *CART* algoritmas yra dvejetainis. Esminis skirtumas tarp *ID3* ir *CHAID* algoritmų yra funkcijos naudojamos šakų dalinimui.

Sprendimo metrika yra funkcija, kuria yra nustatomas medžio duomenų dalinimas. Populiariausios metrikos yra:

Klasifikavimo klaida. Tai viena intuityviausių ir dažniausiai naudojamų metrikų. Ši metrika iš esmės apskaičiuoja kokia tikimybė jog atlikta netinkama klasifikacija ir apskaičiuojama 1-klasės ar poklasio tikimybė,

Gini indeksas. Šis indeksas nusako tikimybę, kad atsitiktinai pasirinkti duomenys bus netinkamai klasifikuojami. Ji apskaičiuojama iš vieneto atimant kiekvienos klasės tinkamų baigčių tikimybes. Indeksas yra matuojamas nuo 0 iki 0,5, kur 0 reiškia labai tikslinga medžio šakų skilimą, o 0,5 nusako kad duomenys yra klasifikuoti visiškai atsitiktinai,

Kryžminė entropija. Šiuo metodu yra skaičiuojama entropija ir informacijos gavimas. Savybės turinčios mažiausia entropiją ir didžiausią informacijos gavimą yra pasirenkamos kaip šaknys. Veiksmas kartojamas kol pasiekimas laukiamas rezultatas[4].

Kadangi šio darbo metu kuriama sistema atlieka dvejetainį klasifikavimą, toliau bus analizuojamas *CART* algoritmas ir jo sudarymas. *CART* sprendimų medžiai sudaromi penkiais esminiais žingsniais :

- Išskiriamas treniravimui skirtas duomenų rinkinys.
- Pasirenkamos sprendimų metrikos (*CART* algoritmo atveju populiariausia metrika – gini indeksas).
- Pasirenkama optimali pradinė šaka.

- Naudojantis sprendimo metrikomis ir sukurtomis taisyklėmis vykdomas skėlimas į vaikinęs atšakas.
- Trečias ir ketvirtas žingsniai kartojami tol, kol pasiekiama iš anksto nurodyta sustojimo taisyklė.

Sprendimo medžių pliusai:

- lengva interpretuoti duomenis,
- galima lengvai filtruotis ieškomas duomenų savybes.

Sprendimo medžių minusai:

- ne visada gaunami patikimi rezultatai,

1.2. Dirbtiniai neuroniniai tinklai

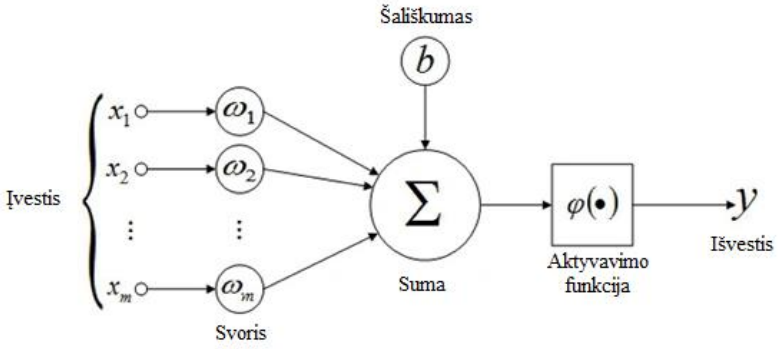
Šiame skyriuje yra apžvelgiami ir analizuojami dirbtiniai neuroniniai tinklai. Toliau bus pateikiama bazinių neuroninių tinklų sandara, panaudojimo atvejai ir praktinis taikymas darbe. Kadangi šiame darbe atliekami klasifikavimo uždaviniai, tai didesnis dėmesys sutelkiamas į klasifikavimo uždavinius skirtus atlikti NN.

1.2.1. Dirbtinių neuroninių tinklų sandara

Dirbtiniai neuroniniai tinklai, ką galima pastebėti iš pavadinimo, yra bandymas imituoti žmogaus neuroninės sistemos veikimą kompiuterinėse sistemose. Kaip ir žmogaus smegenys yra sudaryti iš didelio biologinių neuronų tinklo, dirbtiniai neuroniniai tinklai irgi sudaryti iš dirbtinių neuronų, tačiau žmogaus smegenyse vykstančios cheminės reakcijos yra pakeičiamos į matematines funkcijas, kurios reguliuoja informacijos apdorojimą.

1.2.1.1 Dirbtinis neuronas

Dirbtinis neuronas – modelis, leidžiantis didelį kiekį įvesties duomenų apdoroti ir gauti vieną atsakymą. 3pav. pavaizduotą veikimo principą galima apibūdinti taip: įvesties duomenims yra priskiriama skaitinė vertė ir ji yra padauginama iš jai skirto svorio. Tuomet visos įvestys yra sudedamos ir pridedamas šališkumo parametras. Gautas rezultatas yra dauginamas iš aktyvavimo, dar vadinamas perdavimo, funkcijos. Galiausiai yra gaunamas vienas skaitinis rezultatas.



pav. 3 Dirbtinis neuronas[17]

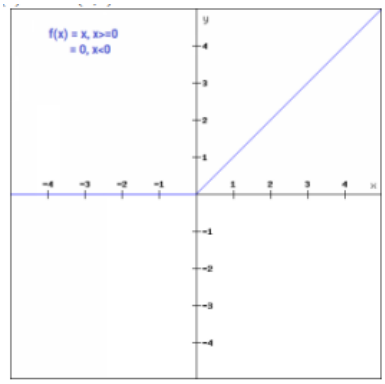
1.2.1.2 Aktyvacijos funkcija

Dirbtiniuose neuronuose dažniausiai sutinkamos penkios aktyvacijos funkcijos: dvejetainė žingsnio funkcija, sigmoidė, hiperbolinio tangento, *ReLU* ir tiesinė [5]. Kadangi atliekant tyrimą buvo pastebėta, kad *ReLU* funkcija yra plačiausiai paplitusi ir gauna geriausius rezultatus, ji bus toliau nagrinėjama.

ReLU matematinė išraiška:

. $$f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases} \quad (3)$$

Ši funkcija yra pati populiariausia aktyvavimo funkcija, kuriant neuroninius tinklus. Viena pagrindinių to priežasčių yra tai, kad neuronai aktyvūs beveik visada, išskyrus tuos atvejus kuomet rezultatas yra mažesnis už nulį [5].



pav. 4 *ReLU* aktyvavimo funkcija[5]

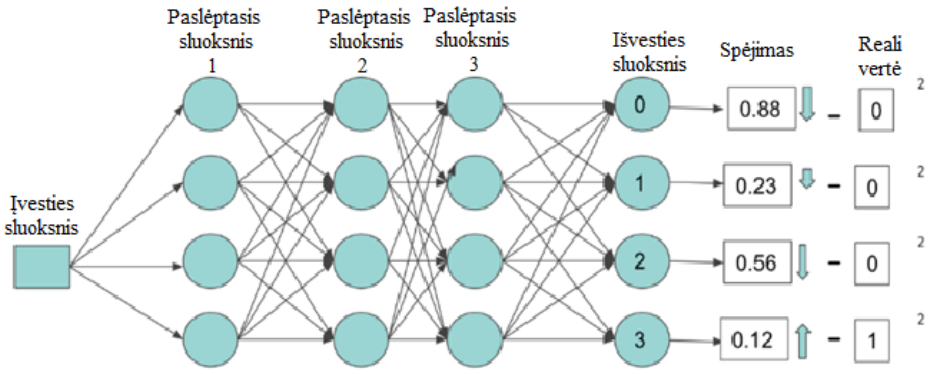
1.2.2. Dirbtinių neuroninių tinklų rūšys

Literatūroje dažniausiai sutinkami trys pagrindinės neuroninių tinklų rūšys: tiesioginio sklidimo, rekurentiniai ir kovuliuciniai. Kadangi pagrindinė konvuliucinių neuroninių tinklų taikymo sfera yra vaizdo medžiagos apdorojimas, o šiame darbe yra atliekami teksto klasifikavimo uždaviniai, šis neuroninių tinklų modelis nebus analizuojamas.

1.2.2.1 Tiesioginio sklidimo neuroninis tinklas

Paprasčiausias iš visų neuroninių tinklų modelių yra tiesioginio sklidimo neuroninis tinklas. Šis tinklas gali būti apibūdintas, kaip dviejų ar daugiau, tarpusavyje sąveikaujančių, dirbtinių neuronų darinys, todėl kartais yra vadinamas „daugiasluoksniu perceptronu“ (angl. *multilayer perceptron*). Nuo pat šių tinklų atsiradimo, vienas pagrindinių jų panaudos atvejų buvo atrasti dideliame kiekyje duomenų, žmogui sunkiai pastebimas, struktūras [15]. Tiesioginio sklidimo neuroninis tinklas gali būti laikomas universaliu priartintoju (ang. *Approximator*). (Hornik, 1991).

Pav. 5 Pateikta labai paprasta tiesioginio sklidimo neuroninio tinklo sandara. Ji yra sudaryta iš tarpusavyje sąryšį turinčių dirbtinių neuronų, kurie išvedą rezultato reikšmę. Šiame tinkle galime pastebėti kainos funkciją, kuri paveikslėlyje atspindi veiksmą „spėjimas – reali vertė“ [17].



pav. 5 Tiesioginio sklidimo neuroninis tinklas su kainos funkcija

Šios funkcijos prasmė yra apskaičiuoti, kiek gautas rezultatas yra nutolęs nuo to kurio siekiama. Taip pat šią funkciją gali pakeisti praradimo funkcija. Iš esmės abi minėtos funkcijos atlieka tą patį darbą, tačiau praradimo funkcija yra taikoma vienam duomenų įrašui, tuo tarpu kainos funkcija yra taikoma visam duomenų rinkiniui. Bendrinė kainos funkcijos formulė yra:

$$C = \sum_1^m (y - y')^2 \qquad (4)$$

C: skaitinė vertė nurodanti modelio gautų rezultatų nuokrypį, kuo didesnis skaičius tuo didesnis yra nukrypimas nuo tikėtųsi rezultatų, *m*: įrašų kiekis, *y*: tikroji duomenų vertė, *y'*: tinklo gautų rezultatų vertė.

Kainos funkcijos dėmenys atspindintys realia vertę ir spėjamą, gali būti tarpusavyje keičiami, ko padarinyje gradientas tiesiog pakeis kryptį, tačiau į šią kryptį būtina atsižvelgti pritaikant kainos funkciją pasirinktam neuroninio tinklo modeliui.

Kadangi kainos funkcija nurodo modelio tikslumą, dažniausiai naudojami mokymo algoritmai, atsižvelgdami į kainos funkciją, koreguoja tinklo svorių parametrus. Vienas populiariausių tokio pobūdžio apmokymo algoritmų yra vadinamas „klaidos skleidimo atgal“(angl. *Back propagation*). Šio algoritmo esmė yra koreguoti neuroninio tinklo svorius, siekiant sumažinti praradimo funkcijos reikšmes. Tai yra atliekama keturiais žingsniais:

- Tinklas atlieka skaičiavimą,
- Apskaičiuojama kainos funkcija,
- Grįžtama į paslėptąjį sluoksnį,

- Pakeičiami svorių parametrai

Šis procesas yra kartojamas tol, kol pasiekiamas norimas rezultatas[15].

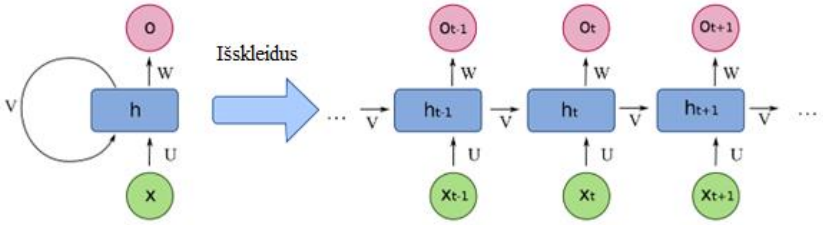
Kainos funkcija ir “klaidos skleidimo atgal” algoritmas taikomas ir kitokių architektūrų NN.

1.2.2.2 Rekurentinis neuroninis tinklas

RNN yra labai panašios struktūros kaip ir tiesinis neuroninis tinklas. Esminis skirtumas yra tai jog RNN yra pritaikoma papildoma būsena, kurioje pateikiami duomenys iš ankstesnių paslėptųjų sluoksniu. Ši papildoma būsena suteikia tinklui galimybę išsaugoti kontekstą, kas yra itin svarbu dirbant su teksto generavimu ir didesnio kiekio duomenų sentimentų atlikimu. Nors minėtos būsenos buvimas suteikia galimybę išsaugoti duomenų kontekstą, tačiau tai sukelia ir papildomų problemų. Viena iš pagrindinių kylančių problemų yra nykstantis gradientas [19]. Algoritmai, kaip „klaidos skleidimo atgal“ gali stipriai iškraipyti duomenis jeigu ankstyvose mokymosi stadijose patiriamas stiprus gradiento kritimas. To padarinyje susiduriama su situacija, kuomet tolimesni duomenys negali būti tikslingai koreguojami.

6pav pateikiama bendrinė RNN veikimo schema. Šiame paveikslėlyje h_t: paslėptoji būsena t žingsnyje, o_t: išvesties būsena t žingsnyje, x_t: įvesties būsena t žingsnyje, U: iš įvesties vektorių matrica, W: išvesties matrica, V: ankstesnio žingsnio paslėptasis sluoksnis.

Analizuojant paveikslėlį, akivaizdu, jog esant situacijai, kuomet RNN turi šimtus, tūkstančius ar daugiau žingsniu, o pirmuose žingsniuose bus patirtas labai stiprus gradiento pokytis, tolimesniuose žingsniuose ankstyvas tinklo svorių pakeitimas iškraipys duomenų apdorojimą ir nepavyks pasiekti norėtų rezultatų. Šiai problemai spręsti sukurtas *LSTM* algoritmas, kuris apžvelgiamas detaliau kitame skyriuje.

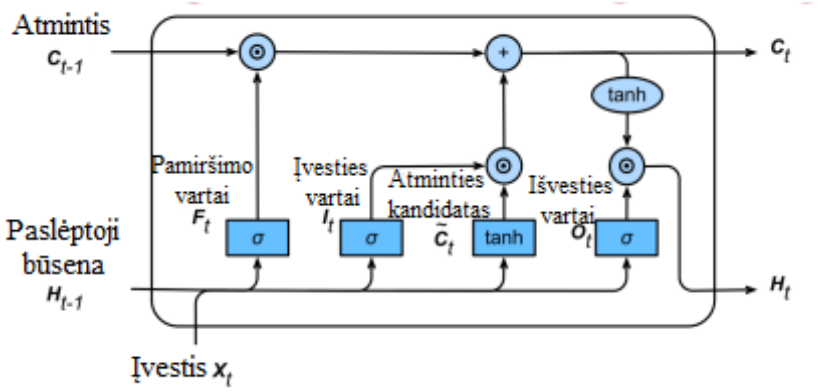


pav. 6 Rekurentinio neuroninio tinklo veikimo schema

1.2.2.3 Ilgos trumpos atminties modelis

Kaip minėta ankstesniame skyriuje, RNN susiduria su problema, kuomet mokymosi metu vyksta gradiento nykimas. Šiai problemai išspręsti buvo sukurtas *LSTM* modelis. Pagrindinis skirtumas tarp šio modelio ir įprastinių RNN yra tai, kad *LSTM* modelyje yra įtraukiami „vartai“ (angl. *Gates*).[19] Pasitelkiant vartų modulius, išvengiama nepageidautinų duomenų priklausomybių, kurios gali nulemti netikslų laukiamą rezultatą. Minėti vartai yra skirstomi į tris grupes : užmaršties vartai (angl. *Forget gate*) , įvesties vartai (angl. *Input gate*) ir išvesties vartai (angl. *Output gate*).

6pav. pateikiama *LSTM* rekurentinio tinklo modelio veikimo schema. Priklausomai nuo atliktos konfigūracijos kiekvieni iš vartų turi savo svorio matricą, kuri dažniausiai išgaunama mokymosi eigoje arba specifiškai parenkama. Taip pat yra naudojama atminties būsena, kuri perneša ankstesniuose cikluose buvusius duomenis. Pirmiausia gavus naują įvestį ir buvusio ciklo paslėptų sluoksnių duomenis, pasitelkiant sigmoidės suspaudimą, užmaršties vartai nustato kurie duomenys turi būti perduodami tolyn, o kurie ignoruojami. Lygiagrečiai pasitelkiant sigmoidės suspaudimą yra atliekami veiksmai įvesties ir išvesties vartuose. Tuomet yra pasitelkiant hiperbolinį tangentą randama potenciali nauja atminties būsena. Ši būsena yra dauginama iš įvesties vartų rezultatų, taip nustatant kurie duomenys esamame cikle yra aktualūs. Tuomet atrinkti duomenys papildo ankstesniame žingsnyje užmaršties vartų modifikuotą atminties būseną. Gauta atminties būsena yra perduodama tolimesniam ciklui. Galiausiai yra apskaičiuojamas ankstesniuose veiksmuose gautos atminties būsenos hiperbolinis tangentas ir padauginamas iš rezultato gauto išvesties vartuose, ko padarinyje gaunama nauja paslėptojo sluoksnio būsena ir ši yra perduodama į tolimesnius ciklus.



pav. 7 LSTM rekurentinio tinklo veikimo schema

Matematiškai *LSTM* modelis yra aprašomas taip:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (5)$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (6)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (7)$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (8)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (9)$$

$$h_t = o_t \circ tanh\,(c_t) \quad (10)$$

i_t : įvesties vartų aktyvacijos vektorius, f_t : užmaršties vartų aktyvacijos vektorius, o_t : išvesties vartų aktyvacijos vektorius, $\tilde{c}_t$: potencialios atminties būsenos vektorius, c_t : naujos atminties būsenos vektorius, h_t : naujo paslėptojo sluoksnio būsenos vektorius, W: rekurentinė svorio matrica gauta treniravimo metu, U: įvesties į paslėptąjį sluoksnį matrica gauta treniravimosi metu, b: šališkumo matrica, σ: sigmoidės aktyvacijos funkcija, $tanh$: hiperbolinio tangento aktyvacijos funkcija, $\circ$: elementų lygio daugyba.

1.3.Technologijų analizė

Šiame skyriuje pateikta technologijų, taikomų mašininio mokymosi ir duomenų apdorojimo realizavimui, analizė. Pateikiami technologijų palyginimai, pabrėžiamas kiekvienos technologijos naudojamumas ir stipriosios pusės. Galiausiai pateikiamos išsirinktos technologijos ir jų pasirinkimą lemiantys argumentai.

1.3.1. Panašios sistemos

Commented [RŠ2]: Teksto prieš skyrelį.

Atlikus paiešką internete, nebuvo aptikta tokio paties pobūdžio modelių. Potenciali to priežastis yra kad užduočiai atlikti reikalingi modeliai yra pritaikyti labai specifiniam kontekstui ir duomenims.

1.3.2. Technologijų, naudojamų modelių įgyvendinimui, analizė

Programavimo technologijos bus vertinamos pagal keturis esminius faktorius: įrankio lankstumą manipuliuojant modeliais, kainą, mokomosios medžiagos kiekį bei išmokimo naudotis įrankiu laiko kaštus. Pagrindiniai iš šių faktorių yra kaina ir bendruomenėje skleidžiamos informacijos kiekis. Šie kriterijai pasirinkti todėl, kad siekiama aukšto modelio lankstumo lygio ir sistemos įgyvendinimo kaštų minimizavimo.

1.3.2.1. Java programavimo kalba

Java yra objektinio programavimo kalba. Tai yra viena populiariausių programavimo kalbų, dėl to nenuostabu, kad ji yra naudojama ir dirbtinio intelekto sferoje. Toliau šiame poskyryje bus apžvelgiamos bibliotekos ir karkasai skirti dirbtinio intelekto kūrimui.

Deeplearning4j yra viena iš populiariausių Java kalbos karkasų naudojamų darbui su dirbtiniu intelektu. Tai yra vienintelis karkasas leidžiantis inkorporuoti python programavimo kalbos modelius.

Apache OpenNLP yra Java biblioteka skirta darbui su natūralios kalbos apdorojimu. Ši biblioteka leidžia atlikti pagrindinius veiksmus su tekstiniais duomenis, tokius kaip : prieigos raktas, sakinio segmentavimas, kalbos dalies žymėjimas, pavadinto objekto išskyrimas, grupavimas, analizavimas ir pagrindinių nuorodų skyrą. Ši kalba taip pat palaiko maksimalia entropija ir perceptronais paremtus mašininio mokymosi modelius.

Nors ši kalba yra nemokama ir internete yra pakankamai mokomosios medžiagos, tačiau jos išmokstamumas yra sunkus ir trūksta lankstumo, nes darbui su DI reikia inkorporuoti python modelius.

1.3.2.2. Python programavimo kalba

Python yra aukšto lygmens, bendrojo naudojimo programavimo kalba. Dėl plataus darbui su dirbtiniu intelektu skirtų bibliotekų skaičiaus ši kalba yra plačiausiai naudojama šioje sferoje. Šiame poskyryje bus apžvelgiamos bibliotekos ir karkasai skirti mašininiam mokymuisi ir neuroninių tinklų kūrimui.

Pytorch yra *python* programavimo kalba ir *Torch* biblioteka paremtas karkasas skirtas darbui su natūralios kalbos apdorojimu, neuroninių tinklų kūrimu ir duomenų vizualizacijai. Kadangi *Torch* biblioteka yra parašyta naudojant *C++* kalbą, ji yra labai greita ir efektyvi.

Keras yra giliojo mokymosi aplikacijos valdymo sąsaja parašyta python programavimo kalba ir veikianti ant *TensorFlow* platformos. Ši sąsaja yra skirta greitiems bandymams dirbant su neuroniniais tinklais. Jos išskirtiniai bruožai yra paprastumas, našumas, lankstumas ir galia.

Python programavimo kalba yra populiariausia kalba dirbant su DI. Internete nesunku rasti mokomosios medžiagos, naudojimas nemokamas, yra galimybė manipuliuoti DI algoritmais ir kodo rašymas yra labai intuityvus.

1.3.2.3. Kiti įrankiai

Google Cloud AI yra aplikacijos valdymo sąsaja, suteikianti galimybę įgyvendinti įvairius su ML susijusius modelius. Šis API yra vienas greičiausių rinkoje. Naudojant šį įrankį galima santykinai lengvai ir greitai apmokyti neuroninius tinklus naudojant siūlomus modelius. Taip pat šiam įrankiui yra pateikta labai išsami dokumentacija ir pateikiama daug aktualios informacijos šaltinių. API yra greitas, lengvai išmokstamas ir suteikia galimybė lengvai keisti algoritmus, tačiau naudojimo kaštai yra neoptimalūs.

Microsoft Azure Machine Learning yra platforma skirta darbui dirbtiniu intelektu. Šioje platformoje yra didelė gausa modelių skirtų duomenų apdorojimui ir analizei, taip modelių skirtų darbui su neuroniniais tinklais. Platforma yra itin draugiška naujiems vartotojams, nes daugumą modelių galimą įgyvendinti grafinės sąsajos pagalba. Ši platforma yra lengvai išmokstama, tačiau naudojimo kaštai yra neoptimalūs.

Knime yra analitinė platforma skirta darbui su dideliu duomenų kiekiu ir duomenų spėjimais. Tai yra atvirojo kodo įrankis, kuris suteikia galimybę vartotojui modeliuoti grafinės vartotojo sąsajos pagalba. Taip pat ši platforma suteikia galimybę naudotis populiariausiomis dirbtinio intelekto bibliotekomis. Įrankis yra lengvai išmokstamas, turi santykinai pakankamą kiekį mokomosios medžiagos, tačiau nėra itin lankstus ir riboto nemokamo naudojimo.

1.3.2.4. Technologijų palyginimas

Žemiau pateiktoje lentelėje yra pateikiamas apžvelgtų technologijų palyginimas. Kaip galime pastebėti *Azure Machine Learning* ir *Google Cloud AI* geras sprendimas, jeigu projektas yra labai mažos arba turi didelį finansavimą. *Knime* turi labai nedidelę bendruomenę, todėl kilus nesklandumams, problemos sprendimo paieška gali pareikalauti didelių laiko kaštų. *Python* ir *Java* kalbos yra nemokamos ir turi didžiausias bendruomenes iš visų programavimo kalbų. Šiuo atveju vienintelis *Python* pranašumas yra tai, kad šios kalbos mokymosi procesas yra lengvesnis už *Java*.

lentelė 1 Technologijų, skirtų modelių įgyvendinimui, palyginimas

Kriterijai	**Java**	**Python**	**Knime**	**Azure Machine Learning**	**Google Cloud AI**
Kaina	Nemokama	Nemokama	Nemokama	Mokama, kaina priklauso nuo sunaudotų resursų kiekio	Mokama, kaina priklauso nuo sunaudotų resursų kiekio
Išmokstamumas	Kalba yra santykinai kompleksiška ir jautri sintaksei.	Kalba yra lengvai išmokstama.	Dėl grafinės vartotojo sąsajos lengvai perprantama	Dėl grafinės vartotojo sąsajos lengvai perprantama	Dėl grafinės vartotojo sąsajos lengvai perprantama
Lankstumas	Dėl poreikio integruoti kitų kalbų modelius, panaudos lankstumas yra nedidelis.	Labai lanksti, modeliai gali būti lengvai modifikuojami.	Esamų modelių modifikacija yra sudėtinga	Esamų modelių modifikacija yra ribota.	Esamų modelių modifikacija yra ribota.
Mokomosios medžiagos kiekis	Vidutinis	Didelis	Mažas	Vidutinis	Vidutinis

1.3.3. Technologijų, naudojamų duomenų surinkimui, analizė.

Oxylabs Web Scraper API yra aplikacijos valdymo sąsaja, suteikianti galimybę gauti tinklapio HTML turinį. Naudojantis šiuo *API* yra galimybė filtruoti norimus duomenis pagal nurodytus parametrus. Yra suteikiama ribotą bandomojo laikotarpio prieiga, sąsaja lengvai suprantama, tačiau galima rinktis tik iš riboto kiekio iš anksto numatytų metodų.

Bright data Web Sraping IDE yra įrankis, suteikiantis galimybę gauti pasirinkto tinklapio *HTML* turinį. Išskirtinė šio įrankio savybė yra labai našus paslėptų ir interaktyvių elementų išgavimas. Taip pat įrankis yra lengvai integruojamas į norimą projektą. Nors siūlomi metodai gali būti modifikuojami, tačiau suteikiama bandomoji versija yra itin apribota.

„My beautiful soup“ arba dar vadinama *„bs4“* yra *Python* biblioteka leidžianti vartotojui išgauti tinklapių *HTML* ir *XLM* failus naudojant *python* programavimo kalbą. Ši biblioteka yra lanksti ir patogu, nes vartotojas pats rašo kodą, tad galimos išgavimo modifikacijos pagal poreikį. Vienas pagrindinių šios bibliotekos trūkumų yra tai, jog paslėptų arba interaktyvių tinklapio elementų išgavimas reikalauja didesnio turimų žinių kiekio.

Žemiau pateiktoje lentelėje yra atliekamas minėtų įrankių palyginimas. Kadangi darbe naudojami įvairūs duomenų šaltiniai, reikalingi labai lankstūs duomenų surinkimo modeliai. Taip pat, kadangi surenkamų duomenų kiekis bus santykinai didelis, optimalūs sąnaudų kaštai yra svarbiausias kriterijus.

lentelė 2 Technologijų, skirtų duomenų surinkimui, palyginimas

Kriterijus	My beautiful soup	Web Scraping IDE	Web Scraper Api
Kaina	Nemokama	Nemokama bandomoji versija, vėliau mokama priklausomai nuo reikiamų resursų kiekio.	Nemokama bandomoji versija, vėliau mokama priklausomai nuo reikiamų resursų kiekio.
Lankstumas	Vartotojas gali modifikuoti pagal asmeninius poreikius	Prieinami tik iš anksto nustatyti metodai	Prieinami tik iš anksto nustatyti metodai
Sudėtingumas	Reikalauja python programavimo kalbos žinių.	Lengvai suprantama grafinė vartotojo sąsaja.	Lengvai suprantama grafinė vartotojo sąsaja.

2. Tyrimas

Šio tyrimo pagrindinė užduotis yra nustatyti ar mašininio mokymosi modeliai gali būti naudingai taikomi karo nusikaltimų paieškoje. Tyrimui pasirinkti trys klasifikavimo modeliai, paremti skirtingomis architektūromis, gebantys skirstyti įrašus į tris kategorijas : „ne karo nusikaltimas“, „potencialus karo nusikaltimas“, „karo nusikaltimas“. Vėliau gauti rezultatai tarpusavyje palyginti ir pateikiamos įžvalgos.

2.1. Tyrimo eiga

Darbo pradžioje buvo atliekamas duomenų surinkimas, kuriam įgyvendinti parašyta trumpa programa, automatiškai renkanti duomenis iš Australijos naujienų portalo „Skynews“. Tuomet buvo atliktas pirminis duomenų valymas, toks kaip tuščių įrašų trynimas ir pan. Po pirminio duomenų išvalymo, remiantis Amerikos saugumo departamento išleistu karo įstatymų leidiniu, buvo sukurtos duomenų kategorijos. Pagal pasirinktas kategorijas kiekvienas duomenų rinkinio įrašas buvo perskaitomas ir priskiriamas jam tinkamai kategorijai.

Kadangi kiekvieno iš tiriamų modelių duomenų įvestis turi šiek tiek skirtumų, duomenų rinkiniui buvo sukurtos trys kopijos. Tuomet atsižvelgiant į modelio reikalavimus duomenys buvo apdorojami ir padalinami į treniravimo ir testavimo aibes. Testavimo aibes sudarė 20% visų turimų duomenų išlaikant jų proporcijas. Likusieji 80% duomenų, kurie naudojami treniravimo procesui, padalinti santykiu 8:2, 8 dalis skiriant treniravimuisi ir 2 dalis kryžminei patikrai. Taigi galutinis visų duomenų paskirstymas buvo 20% testavimui, 64% mokymuisi ir 16% kryžminei patikrai. Testavimo ir mokymo duomenų aibės yra sukuriamos duomenų apdorojimo proceso metu ir jos sudaromos atsitiktinai skirstant elementus, tačiau išlaikant kategorijų proporcijas sukurtuose rinkiniuose. Mokymuisi skirti duomenys buvo padalinti taikant tuos pačius principus modelio kūrimo metu. Kiekvienam modeliui buvo atlikta hyperparametrų paieška, kurios metu buvo siekiama atrasti optimalius parametrus, modelio efektyvumui užtikrinti. Gavus geriausius hyperparametrus, pagal juos buvo apmokomas galutinis modelis ir atliekamas jo testavimas su niekada nematytais duomenimis. Baigus mokymosi procesą išsaugomi modelių rezultatai pagal kuriuos atrenkami geriausi modelių variantai. Taip pat atliekama analizė, kurios metu siekiama nustatyti modelio veiksmingumą įtakojančius veiksnius. Galiausiai palyginami visi tyrimo metu realizuoti mašininio mokymosi algoritmai ir atliekama rezultatų analizė.

Commented [RŠ3]: Duomenų išvalymas pav gal nelabai tinka, imti bendresnį – duomenų apdorojimas.

Gali nurodyti prie padalijimo ir procentus, kaip dalinai. Ar cross validation ir kokių dalių. Kitaip sakant pateikti daugiau info, kad realiai vien ziurint čia būtų aišku, ką darei. Tas pats ir su kitomis daimis (pvz žymėjimą patikslinti)

„modelių paluygnimas “ - > „rezultatų analizė“

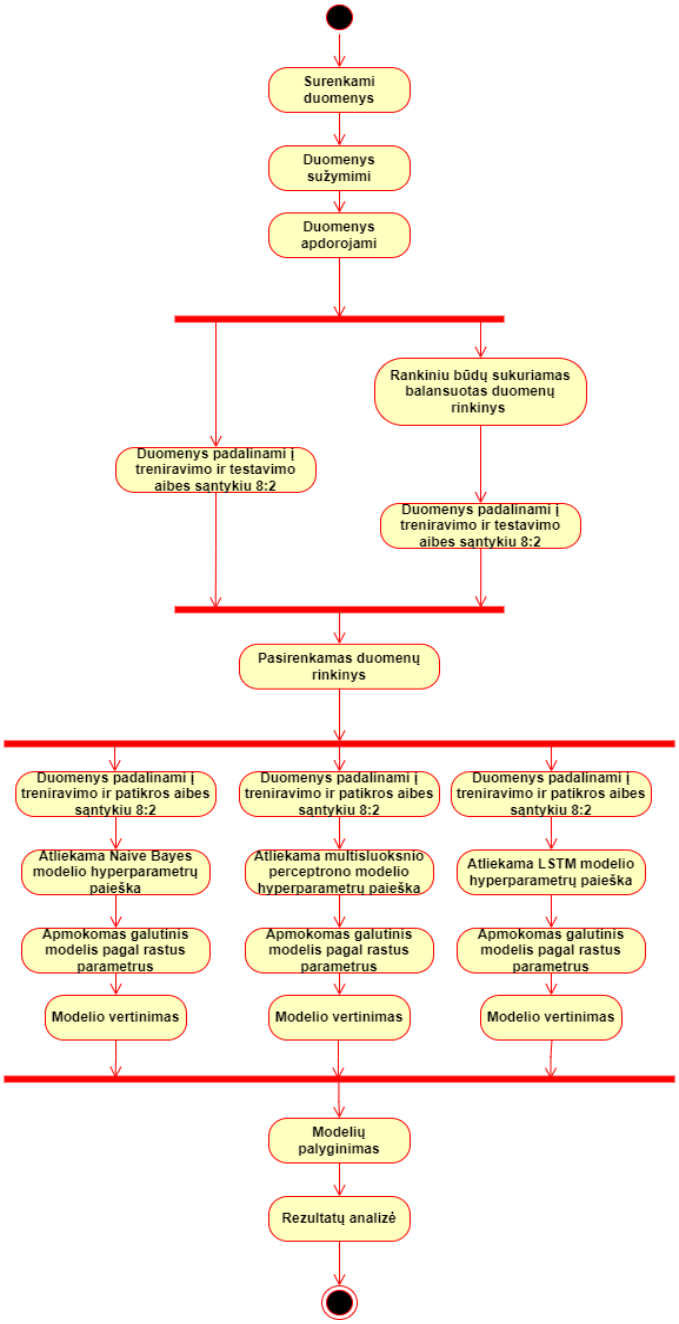
„priimamos darbo išvados“ - nereikia

Commented [RŠ4]: Pav galėtų būti didesnis. Atkriepk dėmesį į tekstą jame, tai turėtų būti pan dydžio kaip tekste. Pastraipą prieš pav skaidyti į dvi dalis.

Commented [RŠ5]: Iš vieno ar kelių šaltinių,

Commented [RŠ6]: Pav. ir lentelių pavadinimai rašomi ma rodos 11 pts

Commented [RŠ7]: Taip pat grafike, gal galima padaryti, kad matytųsi, jog atlikai testus ir su subalansuotais ir nesubalansuotais duomenimus.

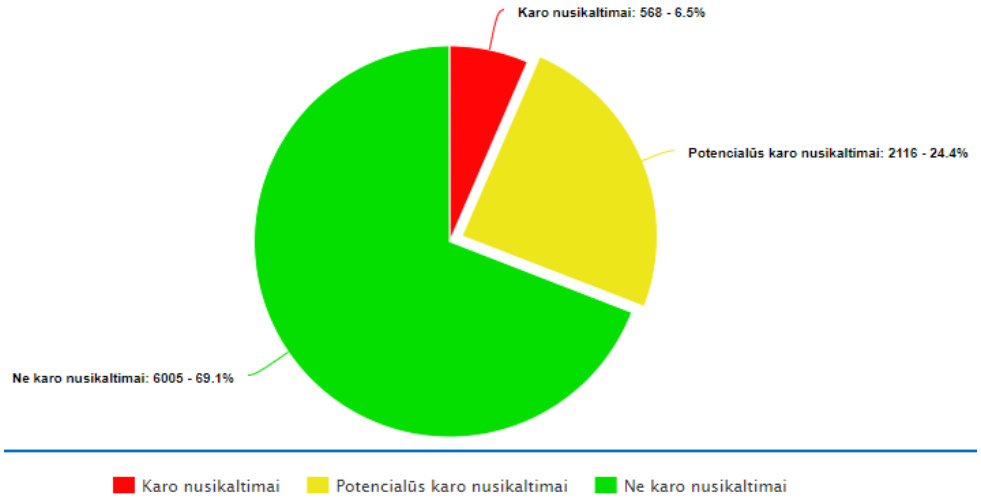


pav. 8 Tyrimo eiga

2.2. Duomenys ir jų paruošimas

Tyrimui atlikti buvo pasirinktas tinklapis *EyeOnRussia.org* ir žinių portalas *SkyNews*. Duomenų rinkimo metu EyeOnRussia.org svetainė buvo atnaujinta, ko pasėkoje dalis duomenų turėjo būti renkama rankiniu būdu, nes naujajai svetainės versijai parašyti automatinio duomenų surinkimo nepavyko. Pradinėje svetainės versijoje duomenys buvo renkami iš svetainės *har* failo, kuriame visa informacija buvo pateikta *json* formatu. Iš viso iš šios svetainės buvo surinkta 4897 teksto įrašų, kurių laikotarpis buvo nuo 2022-04-01 iki 2022-12-31.

Duomenų surinkimui iš naujienų portalo *SkyNews* buvo parašyta trumpa programa (ang. *script*), kuri spausdama mygtuką *„Load more“* plėtė tinklapyje pateikiamų įrašų kiekį. Išplėtus iki reikiamo įrašų kiekio tinklapio HTML failas buvo nuskaitomas ir išsaugoma informacija pagal pasirinktus parametrus. Dėl naudojamos įrangos limitų, kurie pasireiškė greitosios atminties trūkumu, informacija buvo renkama nuo 2023-01-01 iki 2023-04-17. Šiuo laikotarpiu buvo surinkti 3804 įrašai. Visi failai buvo išsaugomi *Excel* lentelėse.



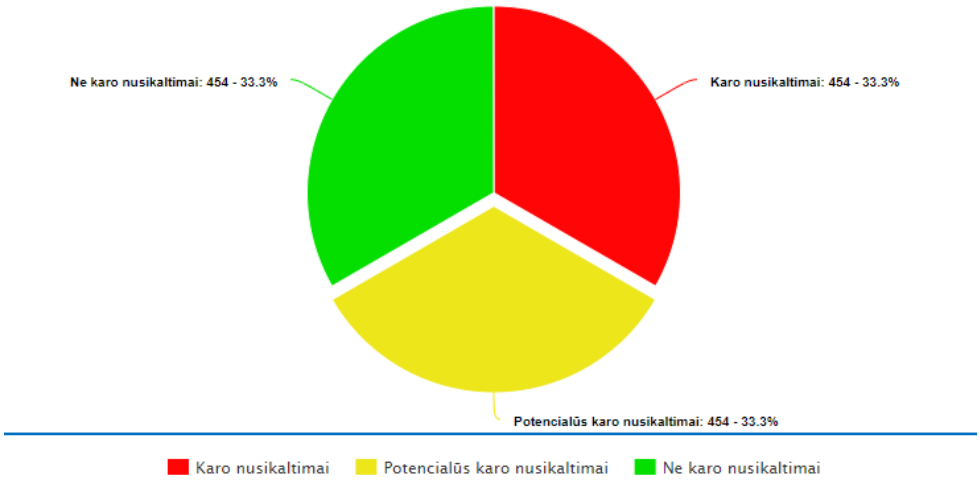
pav. 9 Visų surinktų duomenų pasiskirstymas pagal kategoriją

Commented [RŠ8]: Vėlei nereiktų visko sukišti š vieną pastrtaipą.
Viena pastraipa apie duomenų rinkimą
Kita pastraipa apie duomenų savybes,
Trečia pastraipa galbut apie duomenų apdorijimą.
Išryškinti duomenų balansavimą: galima ir kaip nors vizualiai.

Ai čia vėliau aprašai. Bet tada keistokai skaitos.
Ar reikia čia tokios didelės ir išsamios pastraipos. Gal pateikti kas šiame skyrelyje bus pateikta, o jau tą info ir aprašyti detaliau poskyriuose.
- Duomenų išgavimas.- kaip rinkai iš skiritngų portalų, niuansai ir pan.
- Duomenys ir jų žymėjimas. – Čia kiek duomenų surinkai, ir kaip žymėjai ir pagal ką.
- Duomenų apdorojimas.

Kaip ir ok viskas tik nzn ar tokia tvarka gerai. Kaip nori.

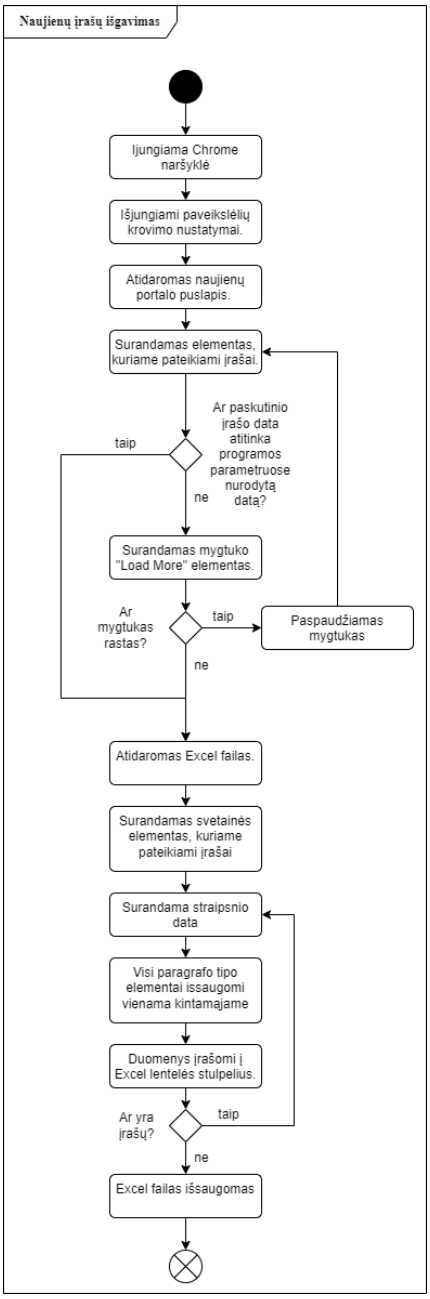
Iš turimų įrašų sudaryti du duomenų rinkiniai, balansuotas ir nebalansuotas. Nesubalansuotas rinkinys buvo sudarytas iš visų tyrimo pradžioje surinktų duomenų, tuo tarpu balansuotas rinkinys yra sudaromas naudojant visus antros kategorijos įrašus ir įtraukiant tokį patį kiekį nulinės ir pirmosios kategorijos įrašų. Tai yra, balansuotas rinkinys sudarytas iš mažo kiekio, bet lygių proporcijų kategorijų atžvilgiu, duomenų.



pav. 10 Subalansuoto duomenų rinkinio sandara

2.2.1. SkyNews naujienų portalo duomenų išgavimas

Naujienų portalo *SkyNews* duomenų išgavimui buvo sukurta šį procesą efektyviai automatizuojanti programa. Šiai programai parašyti buvo naudojamos *Selenium* ir *beautifulSoup* bibliotekos. Programa veikė paleisdama *Chrome* naršyklę su išjungtu paveikslėlių krovimu. Tai buvo daroma todėl, kad svetainės turinys buvo plečiamas kiekvienu praplėtimo mygtuko paspaudimu, ko pasėkoje po daugybės paspaudimų *RAM* atminties poreikiai pasiekė labai didelius reikalavimus. Atvėrus naršyklę, užkraunama svetainė kurioje yra pateikiamos visos naujienos apie karą Ukrainoje. Tuomet surandamas mygtukas *„Load More“* ir jis paspaudžiamas. Palaukiama 2 sekundes, leidžiant svetainei užsikrauti naują turinį ir vėl ieškoma mygtuko. Šis veiksmas vyksta tol kol yra randama parametruose nurodyta data arba neberandama mygtuko. Tuomet yra nuskaitomas svetainės *HTML* kodas ir surandami datos ir turinio laukai. Turinio tekstas, priklausantis tam pačiam įrašui, sudarytas iš atskirų paragrafų, yra suliejamas į vieną įrašą. Duomenys įrašomi į *Excel* lentelės stulpelius nurodančius datą ir tekstą.



pav. 11 Naujienų įrašų išgavimo programos veikimas

2.2.2. Duomenų skirstymas į kategorijas

Iš abiejų šaltinių nukopijuoti aprašų ir naujienų įrašų duomenys buvo įkelti į vieną failą. Tuomet dėl tyrime nagrinėjamos tematikos jautrumo įvykdytas rankinis įrašų žymėjimas. Kiekvienas perskaitytas įrašas buvo priskirtas vienai iš trijų kategorijų: įprastas karinis įvykis (atitinka kategoriją 0), potencialus karo nusikaltimas (kategorija 1) , karo nusikaltimas (kategorija 2). Kadangi tikras pagrindimas ar buvo įvykdytas karo nusikaltimas gali būti pateiktas tik nusikaltimui narplioti paskirtų prokurorų, atliekant šį žymėjimą turėjo būti priimtos tam tikros prielaidos. Bendruoju atveju yra apibrėžiami šie esminiai faktoriai konstatuojantys karo nusikaltimą: tyčinis civilių žudymas, kankinimas, įkaitų naudojimas, civilinės infrastruktūros naikinimas, uždraustų ginklų naudojimas, civilių naudojimas kaip žmoniškųjų skydų, neteisėti žudymai (įvairių organizacijų, kaip „Raudonasis kryžius“ narių, politikų ir pan.) ir neteisėta civilių deportacija [21]. Kai kurie veiksmai yra neabejotinai priskiriami šioms kategorijoms, tačiau kiti gali būti abejotini. Pavyzdžiui civilinės infrastruktūros naikinimas yra priskiriamas karo nusikaltimams tik tuo atveju, jeigu nebuvo įvykdyta pilna populiacijos evakuacija. Arba civilių žudymas turi būti tyčinis, kas yra be galo sunku įrodyti. Šio tyrimo metu yra priimamos 3 prielaidos:

- Visas rusų karių vykdytas civilių žudymas laikomas tyčiniu. Visi įvykiai susiję su civilių ar kitų, karo veiksmuose nedalyvaujančių, asmenų mirtimis priskiriami 2 kategorijai. Šiai kategorijai taip pat priskiriamos deportacijos, kankinimas ir uždrausti ginklai.
- Miestų evakuacijos statusas ne visada randamas, todėl miestų ir juose esančių kultūrinių objektų sunaikinimas, kuris neturi nurodytų aukų bus laikomas potencialiu karo nusikaltimu, tai yra bus priskiriamas 1 kategorijai.
- Įrašai, kuriuose karo nusikaltimas nėra esminė mintis, tačiau pats nusikaltimas yra santykinai detaliai apibūdinamas yra priskiriami atitinkamai kategorijai 1 arba 2.

Visi kiti įrašai, kurie neatitiko aukščiau išvardintų kriterijų yra priskiriami kategorijai 0. Kadangi buvo priimtos papildomos prielaidos, visas kategorijų priskyrimo procesas buvo vykdomas vieno asmens, taip siekiant išvengti papildomų asmeninių interpretacijų, kurios galėtų iškraipyti duomenis.

2.2.3. Duomenų apdorojimas ir paruošimas modeliams

Commented [RŠ9]: Apdorojimas. Valymas atrodo, kad ištrynei tik kažką.

Tyrimo metu buvo realizuoti 3 skirtingi metodai, todėl net ir esant panašiems teksto paruošimo poreikiams, buvo sukurtos trys programos kiekviena paruošianti tekstą atitinkamam modeliui. Tačiau iš esmės visos trys programos atlieką labai panašius veiksmus. Pirmiausia tekstas yra praplečiamas. Tai reiškia, visi tekste esantys anglų kalbos trumpiniai yra pakeičiami ilgosiomis formomis. Tuomet visas tekstas yra paverčiamas į mažąsias raides ir iš jo yra pašalinami mažos reikšmės žodžiai pritaikant

"Stopwords". Galiausiai yra panaikinami skyrybos ženklai. Tyrimo metu buvo atliekami bandymai taikant žodžių šaknų atskyrimą ir jo netaikant. Šis veiksmas iš esmės sumažina teksto kompleksiškumą ir leidžia modeliui lengviau atrasti potencialius duomenų sąryšius. Kadangi turimas duomenų rinkinys gali būti laikomas kaip mažas, jame esančių įrašu kompleksiškumo nėra poreikio mažinti. Tyrimo metu pastebėta kad šaknies išskyrimas, treniruojant modelius su surinktais duomenimis, neturėjo apčiuopiamos įtakos. Taip pat duomenų paruošimo metu duomenys turėtų būti vektorizuojami, tačiau dėl techninių sunkumų, buvo pasirinkta šį žingsnį perkelti į modelio failą.

Tyrimo metu taip pat buvo naudojama *GloVe* įterptis (ang. *Embedding*). Kadangi karo nusikaltimų kontekstas yra santykinai retai tyrinėjamas, vektorinių verčių įterptims pasirinktas didžiausias *GloVe* duomenų rinkinys, kuris sudarytas 840 milijardų interneto įrašų. Šios įterpties vektorinės vertės buvo naudojamos kaip modelių įvesties vektoriaus pradinės vertės.

Naïve Bayes klasifikatoriaus atveju buvo panaudota *TF-IDF* vektorizatorius ir svorių paskyrimas. Ši funkcija patikrina teksto žodžius, dažnai įvairiuose dokumentuose besikartojantiems žodžiams priskiriami mažesni svoriai, o retesniems ir svarbesnį kontekstą potencialiai turintiems žodžiams didesnis svoris.

2.3.Modelių įgyvendinimas ir hyperparametrų paieška.

Commented [RŠ10]:

Šiame skyrelyje yra apžvelgiami *Naïve Bayes*, *LSTM* ir *MLP* klasifikatoriai. Analizuojami atsitiktinės tinklelio paieškos metu gauti rezultatai ir pateikiamos mokymosi proceso įžvalgos.

2.3.1. Naïve Bayes klasifikatorius

Naïve Bayes modelis yra paprasčiausias iš visų šiame tyrime naudojamų modelių. Jis yra paremtas statistiniu tikimybės skaičiavimu ir vienintelis kintamasis, kuris gali būti keičiamas atliekant tinklelio paiešką yra *alpha.* Šis parametras yra naudojamas glotninimui (ang. *Smoothing*), kurio tikslas yra išvengti 0 procentų tikimybės kintamajam. Šis glotninimas yra reikalingas todėl, kad pasitaiko situacijos kuomet testuojamame duomenų rinkinyje aptinkami žodžiai kurie nebuvo aptikti treniravimosi metu ir jiems priskiriamos 0 procentų tikimybės. Iš esmės kuo didesnis *alpha* parametras tuo didesnė „dirbtinė" tikimybė pridedama prie nematytų testo duomenų rinkinio žodžių.

Tyrimo metu sukurtame *Naive Bayes* modelyje, buvo naudojamas pagal poreikius sukurtas įvertinimo modulis, kuris vertina modelio kokybę pagal antrosios kategorijos (karo nusikaltimai) atsiminimą. Taip pat buvo panaudotas Chi kvadratu (ang. *Chi-squered*) testas, kuris pateikia pasirinkto kiekio įrašų priklausomybių lentelę. Remiantis šia lentele (pav. 11) galima nustatyti kokį sąryšį kiekvienas įrašas turi su atitinkama kategorija (pirmas stulpelis atspindi kategoriją 0, antras – kategoriją 2, trečias- kategoriją 3).

Commented [RŠ11]: Pav nr kur ziureti

BakalauroDarbas > NB > FinalResults > NB_class_prob.txt

1	2.922639306047671304e-01	3.414645792509836131e-01	3.662714901442492010e-01
2	8.738588963435320878e-01	1.254112144724266013e-02	1.135999822092249451e-01
3	3.123108739166826492e-02	7.926472213218569429e-01	1.761216912864755069e-01
4	8.368094317191333564e-01	2.234798488167374572e-02	1.408425833991910348e-01
5	5.320348115164079683e-01	1.102999469780191755e-01	3.576652415055722178e-01
6	7.594851082660283859e-02	7.273729779136282048e-01	1.966785112597689289e-01
7	9.439113820912561348e-01	1.323559776316088343e-02	4.285302014558278055e-02
8	8.317702797638055756e-01	5.306744474466597306e-02	1.151622754915283958e-01
9	1.506584732599324794e-01	7.898896560587294058e-01	5.945187068133835073e-02
10	5.863791672143884304e-01	1.235872319638636474e-01	2.900336008217466177e-01
11	1.455371923435054282e-01	7.120234176142219429e-01	1.424393900422724624e-01
12	5.674594165896160725e-01	1.738398780168885216e-01	2.587007053934964329e-01
13	6.864885169246286445e-01	1.096006060189475412e-01	2.039108770564233286e-01
14	8.695004824583192349e-01	3.948869992338324177e-03	1.265506475493415328e-01
15	7.252585406050940375e-02	6.118915430587537063e-01	3.155826028807370842e-01

pav. 12 Įrašų priklausomybės pagal kategoriją pavizdys

Siekiant atrasti geriausius hyperparametrus turimiems duomenims, buvo taikoma tinklelio paieška. Kadangi *Naive Bayes* klasifikatorius yra itin greito veikimo, buvo naudojama nuosekli paieška, o ne atsitiktinė. *Alpha* reikšmių diapazonu pasirinktos reikšmės 0.01-15, jas didinant po maždaug 0.01. Kadangi *alpha* yra skaičiuojama logaritminėje skalėje šio diapazono išraiška pateikta žemiau esančiame paveikslėlyje (pav. 12). Galime pastebėti, kad *alpha* kis nuo 10^{-2} iki $10^{1.176}$ (apytikslė reikšmė lygi 15) ir šiame diapazone išbandys 1180 *alpha* reikšmių.

Commented [RŠ12]: Nurodyti nr

Taip pat tinklelio paieškos metu buvo taikoma 2 sluoksnių kryžminė patikra, tai reiškia kad

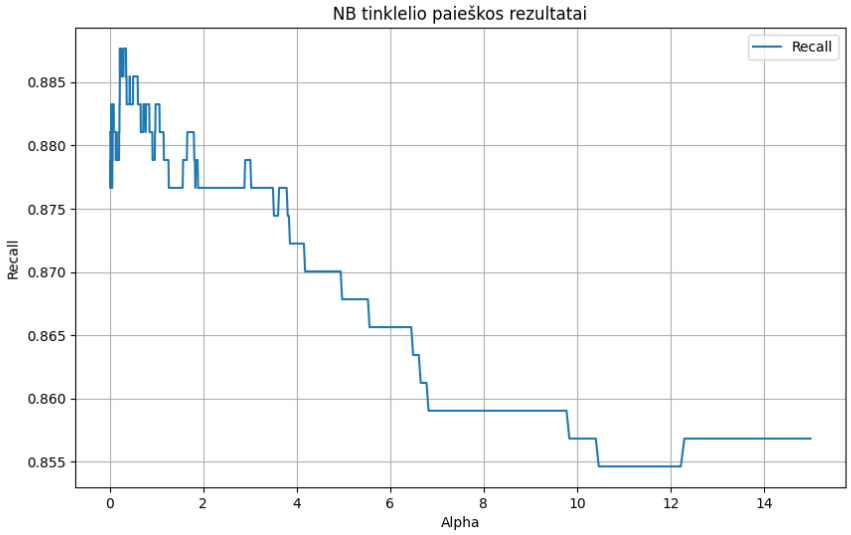
```
param_grid = {
    'alpha': np.logspace(start=-2, stop=1.176, num=1180, base=10)
}
```

pav. 13 *Naive Bayes* tinklelio paieškos *alpha* parametro diapazonas

mokymosi duomenų aibė skaidoma į dvi dalis ir jos apmokomos nepriklausomai, tuomet jų rezultatai yra palyginami ir pateikiamas jų vidurkis.

Atlikęs tinklelio paiešką modelis išsaugo geriausią rezultatą pasiekusio algoritmo vektorių vertes, metrikas ir įrašų priklausomybės lentelę. Tuomet yra panaudojama kita *Naïve Bayes* modelio versija, kurioje nebenaudojama tinklelio paieška, tačiau yra užkraunamas testavimo duomenų rinkinys, tinklelio

paieškoje gautos vektorių reikšmės bei *alpha* reikšmės.Galiausiai su turimais parametrais buvo atliekamas testavimo duomenų rinkinio klasifikavimas ir gaunama klaidų matrica, metrikų verčių ataskaita, ir testavimo duomenų rinkinio duomenų priklausomybės lentelė. Žemiau esančioje lentelėje pateikiamas kas šimtasis *Neive Bayes* klasifikatoriaus rezultatas, siekiant pavaizduoti modelio atsiminimo rezultato pokyčio priklausomybę nuo *alpha* reikšmės.



pav. 14 *Naive Bayes* tinklelio paieškos rezultatų priklausomybė nuo *alpha* parametro

```
Best Alpha: 0.21416145373379775
Best Recall for Category 2: 0.8686708860759493
Classification Report:
              precision    recall  f1-score   support

         0.0       0.96      0.71      0.81      1206
         1.0       0.61      0.79      0.69       418
         2.0       0.33      0.90      0.49       113

    accuracy                           0.74      1737
   macro avg       0.63      0.80      0.66      1737
weighted avg       0.83      0.74      0.76      1737

Confusion Matrix:
[[854 203 149]
 [ 32 331  55]
 [  6   5 102]]
Accuracy Score: 0.7409326424870466
```

pav. 15 *Naive Bayes* modelio geriausi rezultatai

Naïve Bayes klasifikatoriaus gauti rezultatai iš tiesų yra labai geri. Kaip galima pastebėti paveikslėlyje pateiktoje klaidų matricoje, iš 113 įrašų tik 6 karo nusikaltimų įrašai buvo priskirti ne karo nusikaltimų kategorijai. Kadangi pirmoji kategorija (potencialūs karo nusikaltimai) yra labai glaudžiai susijusi su karo nusikaltimų kategorija ir idealiu atveju vykdant teisinį tyrimą taip pat būtų atidžiai narpliojami, 5 neteisingai kategorizuoti antros kategorijos kaip pirmosios įrašai gali būti ignoruojami. Nors šio modelio bendras tikslingumas yra apie 74 procentus, tačiau teisingas antrosios kategorijos nustatymas sudaro apie 87 procentus.

2.3.2. LSTM klasifikatorius

LSTM modelis yra kur kas labiau kompleksiškas nei kiti šiame tyrime naudoti mašininio mokymosi modeliai. Ši rekurentinio neuroninio tinklo variacija gali keisti įvairius hiperparametrus, siekiant gauti reikiamą modelį. Kadangi šis modelis yra kur kas sunkesnis, hyperparametrų paieškai naudojama atsitiktinė tinklelio paieška. Tai reiškia, jog parametrai yra parenkami iš vartotojo pateikto diapazono atsitiktinai ir sukuriamas modelis. Šio tyrimo metu buvo keičiami šie hyperparametrai:

- *LSTM* sluoksnių skaičius (ang. *Lstm layer*) – sluoksniai, kuriuose yra naudojami užmaršties ir kiti *LSTM* modeliui priklausantys vartai.
- *LSTM* neuronų skaičius (ang. *Lstm units*) – nurodoma kiek neuronų turi kiekvienas *LSTM* sluoksnis.
- Atkritimo tempas (ang. *Dropout rate*) - nurodo kokia dalis neuronų yra atsitiktinai išjungiami, siekiant sumažinti modelio perdėtinį (ang. *Overfitting*) mokymasi.
- Mokymosi tempas (ang. *Learning rate*) – nurodo svorių pokyčio kitimo tempas , mokymosi metu.
- Tankusis sluoksnis (ang. *Dense layer*) – nurodo sluoksnių, kuriuose esantys neuronai, kurie priima visų ankstesnio sluoksnio neuronų išvestį.
- Tankiųjų neuronų skaičius (ang. *Dense units*) – nurodomas neuronų kiekis tankiajame sluoksnyje.
- Mokymosi paketo dydis (ang. *Batch size*) – nurodo kokį įvesčių kiekį priėmus duomenų svoriai atnaujinami.
- Kantrybė (ang. *Patiance*) – nurodo epochų skaičių, per kurį modeliui negavus rezultatų pagerėjimo, nutraukiamas mokymosi procesas.

Pradinėje tyrimo stadijoje visi šie hyperparametrai buvo keičiami plačiuose intervaluose ir buvo stebimi modelių mokymosi proceso vyksmai. Pastebėjus indikacijas nurodančias tinkamą mokymosi procesą, modeliai buvo stabdomi ir tinklelio paieška buvo atnaujinama sumažintame hyperparametrų intervale. Pirminėje atsitiktinėje tinklelio paieškoje buvo naudojami šie parametrai:

- lstm sluoksnių skaičius : [2, 3, 4, 6, 8],
- lstm neuronų skaičius: [16, 32, 64],
- atkritimo tempas: [0.1, 0.2, 0.4, 0.5],
- mokymosi tempas: [0.001, 0.0001],
- tankiųjų sluoksnių skaičius: [1, 2, 3, 4],
- tankiųjų neuronų skaičius: [16, 32, 64],
- mokymosi paketo dydis: [32, 64]

Atliekant minėtą paiešką taip pat buvo atliekamos ir kitos modelio architektūros modifikacijos, tokios kaip kryptingumas ir reguliarizavimas.

Viena svarbiausių, tyrimo metu įvykdytų, architektūrinių modifikacijų buvo kryptingumo pakeitimas. Kryptingumo atžvilgiu *LSTM* modelis turi dvi variacijas: dvikryptis (ang. *Bidirectional*) ir vienkryptis (ang. *Unidirectional*). Vienkrypčiame *LSTM*, įvesties seka yra apdorojama pagal jos originalią tvarką, nuo pirmojo elemento iki paskutiniojo. Paslėpti sluoksniai kiekviename žingsnyje yra atnaujinami remiantis esama įvestimi ir ankstesnio paslėpto sluoksnio būsena. Galutinė paslėpta būsena apdoroja informaciją iš visos sekos, tačiau šis metodas turi ribojimą: *LSTM* išmoksta sekos struktūras daugiausiai remdamasis ankstesniais elementais, kas gali būti neoptimalu, jei svarbi informacija yra vėliau sekoje. Tuo tarpu dvikryptis *LSTM* išsprendžia vienakrypčio *LSTM* ribotumą apdorodamas įvesties seką abiem kryptimis - originalia tvarka ir atvirkštine tvarka. Jame yra du atskiri *LSTM*, vienas apdoroja seką pirmyn, o kitas - atgal. Abiejų *LSTM* paslėptos būsenos yra sujungiamos kiekviename laiko intervale, leidžiant modeliui išmokti struktūras, remiantis tiek praeities, tiek ateities kontekstu.

Tyrimo metu taip pat atlikti eksperimentai taikant reguliarizaciją, kuria buvo siekiama sumažinti modelio priklausomybę nuo mokymosi duomenų. *LSTM* modeliui buvo panaudota L1L2 reguliarizacija ir neuronų išmetimas, kuris yra keičiamas kaip hyperparametras. L1L2 reguliarizacijos forma susideda iš dviejų metodų: L1 ir L2. Abu šie metodai yra skirti mažinti modelio priklausomybei nuo duomenų, siekiant užtikrinti tinkamą mokymosi procesą. L1 reguliacija sumažina maža įtaką turinčių duomenų svorius iki 0, tai siekiant išryškinti tik esmines duomenų savybes, tuo tarpu L2 reguliarizacija didžiausius svorius turinčias savybes „baudžia“ mažinant jų svorį. Tuo siekiama suvienodinti įvesties duomenų svarbą taip sumažinant tikimybę, jog modelis sukurs nepageidautinus sąryšius tarp duomenų savybių. Šio tyrimo metu buvo atlikti bandymai taikant L2 reguliarizaciją *LSTM* modeliui, tačiau vienintelė šios reguliacijos naudos indikacija buvo padidėjęs mokymosi proceso kreivės glotnumas.

Kadangi tyrimui naudojami duomenys yra nesubalansuoti, *LSTM* modelio mokymo metu buvo vykdomi eksperimentai, kuriais buvo tikrinama balansavimo technikų įtaka galutiniams rezultatams. Buvo bandomos dvi technikos: SMOTE (ang. *Minority Over-sampling Technique*) ir klasių svorių paskyrimas. SMOTE duomenų balansavimo technika yra paremta dirbtinių duomenų kūrimu, kuomet mažumos klasės duomenys yra sukuriami vykdant turimų įrašų jungimus ir kitas manipuliacijas. Tuo tarpu klasių svorių technika veikia labai panašiai kaip reguliarizacija ir modifikuoja mažumos klasių įrašų svorius taip, kad šie turėtų didesnę įtaką, skirtumas tik tai jog reguliarizacijos atveju svoriai keičiami mokymosi proceso metu, o klasių balansavimas taikomas pradiniams duomenims. Atlikus bandymus, pritaikant šias technikas, buvo pastebėta jog taikant SMOTE pasiekti rezultatai buvo prastesni lyginant su atvejais kuomet sintetiniai duomenys nebuvo naudojami. Viena iš priimtų prielaidų, buvo tai, jog tiriama problematika yra labai jautraus pobūdžio ir sukurti duomenys neturi reikalingo konteksto. Kita vertus kuomet duomenims buvo pritaikytas klasių svoris, buvo pastebėtas nežymus modelio rezultatų pagerėjimas, mokymosi proceso metu.

Praradimo vertės kito šiek tiek stabiliau ir lygiau, ko pasėkoje galima daryti prielaidą, kad taikant klasių svorius nesubalansuotiems duomenims, modelis mokosi šiek tiek tikslingiau.

Galiausiai buvo vykdomi bandymai su aktyvacijos funkcijomis. Ankstyvose tyrimo stadijose paslėptuose sluoksniuose buvo naudojama *ReLu* aktyvacijos funkcija, tačiau pastebėta kad taikant *LeakyReLu* modifikaciją, kuri skiriasi nuo originalios formos tuo, kad net įprastu atveju svoriui pasiekus 0 neuronas nesimoko ir tampa „miręs“, o atlikus modifikaciją, neuronas net tais atvejais keičia svorio reikšmę labai mažais intervalais, ko pasėkoje bendru atveju pasiekiamas sklandesnis mokymosi procesas. Tuo tarpu išvesties sluoksniuose buvo naudojama *softmax* aktyvacijos funkcija, kuri yra dažniausiai naudojama aktyvacijos funkcija, skirta darbui su daugiau nei dviejų kategorijų klasifikavimo užduotimis. Atliekant visas minėtas modelio modifikacijas buvo taikomas Adam optimizatorius.

Žemiau esančioje lentelėje (lentelė 3**)** yra pateikiami keletas tyrimo metu apmokytų modelių, kuriems buvo taikomos visos ar dalis aukščiau minėtų modifikacijų. Lentelėje yra pateikiama keletas tyrimo metu gautų rezultatų. Pirmas aiškiai pastebimas faktas yra, kad rezultatai naudojant *SMOTE* yra prastesni nei jo netaikant. Taip pat galime pastebėti, kad modeliuose, kurių kompleksiškumas yra mažesnis, tai yra mažiau sluoksnių ir neuronų, rezultatai yra gaunami geresni. Tai indikuoja, kad turimame duomenų rinkinyje kompleksijos lygis yra santykinai žemas dėl to, kad didinant modelio sudėtingumą nėra pastebima akivaizdžių galutinio rezultato teigiamų pokyčių. Dar vienas veiksnys, kurį galime pastebėti, yra tai kad modeliai kurie buvo treniruojami didesniu mokymosi tempu, gavo prastesnius rezultatus, dėl ko galima daryti prielaidą, kad esant didesniam mokymosi tempui, bet išlaikant mažą kompleksiškumą modelis nebesugeba atrasti tinkamų sąryšių tarp duomenų. Ši problema potencialiai gali būti sprendžiama didinant modelio sandaros sudėtingumą, tačiau tai lygiagrečiai didintų ir naudojimo kaštus, ko pasėkoje modelis tampa nenaudingas iš ekonominės pusės. Galiausiai pastebima, kad kuomet modelio sluoksnių ir neuronų skaičius yra labai mažas ir yra taikoma išmetimo technika, tinklas tampa nebepajėgus atrasti sąsajas tarp duomenų ir gaunamas rezultatas yra itin prastas.

lentelė 3 LSTM modelio rezultatai

Nr.	Lstm sluoksnių sk.	Lstm neuronų sk.	Atkritimo tempas	Mokymosi tempas	Tankių sluoksnių sk.	Tankių neuronų sk.	Mokymosi paketo dydis	SMOTE	L2	Klasių svoris	Duomenų rinkinio balansavimas	Karo nusikaltimų teisingos klasifikacijos tikimybė
1.	2	32	0.3	0.0001	2	64	64	-	+	+	Rankinis balansuotas	0.754385965
2.	2	32	0.1	0.0001	1	64	64	-	-	-	Rankinis balansuotas	0.833333333
3.	2	16	0.3	0.0001	2	64	64	-	-	-	Pilnas nebalansuotas	0.210526316
4.	2	16	0.3	0.0001	1	32	64	-	-	-	Rankinis balansuotas	0.868421052
5.	4	32	0.3	0.0001	3	32	64	-	-	+	Rankinis balansuotas	0.754385964
6.	4	32	0.5	0.0001	2	32	32	-	-	-	Rankinis balansuotas	0.684210526
7.	3	32	0.1	0.0001	2	16	64	-	-	+	Rankinis balansuotas	0.780701754
8.	4	16	0.2	0.0001	1	16	64	-	-	-	Rankinis balansuotas	0.657894736
9.	6	16	0.2	0.0001	1	16	64	-	-	-	Rankinis balansuotas	0.728070175
10.	2	128	0.2	0.0001	1	64	64	+	-	-	Pilnas nebalansuotas	0.298178344
11.	1	64	0.4	0.001	1	32	32	+	-	-	Pilnas nebalansuotas	0.572463750
12.	8	64	0.1	0.001	4	32	64	+	-	-	Pilnas nebalansuotas	0.697340071
13.	2	16	0.5	0.0001	1	16	64	-	+	+	Rankinis balansuotas	0.842922925
14.	1	16	0.5	0.0001	2	32	64	-	-	-	Pilnas nebalansuotas	0.008761329
15.	2	16	0.4	0.0001	1	16	32	-	+	+	Pilnas nebalansuotas	0.692982456

2.3.3. Daugiasluoksnio perceptrono modelis

Tyrimo metu išbandytas daugiasluoksnio perceptrono modelis yra labai panašus į prieš tai skyriuje aprašytą *LSTM* modelį. *MLP* modeliui taip pat buvo pritaikyta tinklelio paieška su dviguba kryžmine patikra, išbandyta SMOTE technika, panaudotas *Adam* optimizatorius ir *softmax* aktyvavimo funkciją.Tyrimo metu buvo naudojami šie tinklelio paieškos parametrai:

- Paslėptų sluoksnių kiekis: [2,4,6,8]
- paslėptų neuronų skaičius: [16, 32],
- atkritimo tempas: [0.1, 0.3 , 0.5],
- mokymosi tempas: [0.001, 0.0001],
- mokymosi paketo dydis: [32, 64].

Iš hyperparametrų galime pastebėti, kad vienas esminių skirtumų tarp *MLP* modelio ir *LSTM* yra *LSTM* sluoksnių ir neuronų trūkumas. Taip pat verta paminėti, kad visi daugiasluoksnio perceptrono paslėptieji sluoksniai yra laikomi tankiais sluoksniais, tai yra visi šio tinklo neuronai yra tarpusavyje susiję. Kaip ir bandymų su *LSTM* modeliu metu buvo pastebėta, kad *SMOTE* technika nesuteikia jokio pranašumo lyginant su modeliais kurie šios technikos nenaudojo.

Galiausiai modelio mokymo metu buvo pastebėta, kad net naudojant tokius pačius mokymosi tempus, tiesioginio sklidimo daugiasluoksnio perceptrono modelis kur kas labiau linkęs į perdėtiną mokymąsi specifiniams duomenims, nei *LSTM*.

Žemiau pateiktoje lentelėje (lentelė 4) pateikta keletas tyrimo metu gautų rezultatų. Galima pastebėti, kad gauti daugiasluoksnio perceptrono rezultatai nėra patenkinami, nors buvo pritaikytos tos pačios technikos, kurios buvo naudojamos ir kituose modeliuose.

lentelė 4 MLP modelio rezultatai

Nr.	Paslėptų sluoksnių sk.	Paslėptų neuronų sk.	Atkritimo tempas	Mokymosi tempas	Mokymosi paketo dydis	SMOTE	L2	Klasių svoris	Balansavimas	Karo nusikaltimų teisingos klasifikacijos tikimybė
1.	3	64	0.2	0.0001	64	-	-	+	Rankinis balansuotas	0.756616771
2.	3	64	0.2	0.0001	64	-	-	-	Rankinis balansuotas	0.739930927
3.	2	128	0.5	0.0001	64	-	+	-	Rankinis balansuotas	0.737629473
4.	6	64	0.2	0.0001	64	-	-	-	Rankinis balansuotas	0.725546598
5.	2	128	0.2	0.0001	64	-	-	+	Rankinis balansuotas	0.7485615611
6.	6	128	0.2	0.0001	32	-	-	-	Rankinis balansuotas	0.766973555
7.	8	32	0.1	0.0001	32	+	-	+	Pilnas nebalansuotas	0.569877815
8.	8	32	0.1	0.0001	64	-	-	-	Pilnas nebalansuotas	0.387894736
9.	1	64	0.1	0.0001	64	-	-	-	Rankinis balansuotas	0.732451081
10.	1	16	0.1	0.0001	32	-	-	-	Rankinis balansuotas	0.714039146
11.	4	32	0.1	0.001	128	-	-	-	Rankinis balansuotas	0.729574203
12.	7	128	0.1	0.001	32	-	-	-	Rankinis balansuotas	0.765247404
13.	5	64	0.1	0.0001	64	-	+	-	Pilnas nebalansuotas	0.715189874
14.	3	64	0.5	0.0001	128	-	+	-	Pilnas nebalansuotas	0.572669732
15.	3	128	0.5	0.0001	128	-	-	-	Rankinis balansuotas	0.753164529

2.4. Modelių rezultatų vertinimas ir palyginimas

Commented [RŠ13]: Gal ne pačių modelių o rezulatų

2.4.1. Vertinimo metrikos

Šio tyrimo pagrindinis tikslas yra nustatyti ar mašininio mokymosi algoritmai gali nustatyti karo nusikaltimus iš viešai prieinamų tekstinių duomenų. Siekiant nustatyti modelio tinkamumą ir tai kaip šis įvykdo jam skirtą užduotį, buvo stebimos pagrindinės modelių vertinimo metrikos. Jas sudaro: tikslumas (ang. *Precision*), tikslingumas (ang. *Accuracy*), atsiminimas (ang. *Recall*), F-1 įvertis ir klaidų matrica. Taip pat buvo stebimi modelių mokymosi proceso grafikai, kuriais buvo nustatomas modelių mokymosi proceso korektiškumas.

Klaidų matrica yra lentelė, kuri nurodo visų atliktų spėjimų skaičių. Žemiau pateiktoje pavyzdinėje letenėlėje eilutėse yra nurodytos tikrosios kategorijos, o stulpeliai atspindi kiek tikrosios kategorijos narių buvo priskirta kiekvienai kategorijai. Pvz. “Category 0” eilutė būtų skaitoma taip: kategorija 0 buvo priskirta 0-inei kategorijai 1 kartą, pirmai kategorijai 2 kartus, antrajai kategorijai 3 kartus. Iš šios lentelės galima gauti 3 pagrindines reikšmes: tikroji reikšmė (ang. *true-positive*), klaidinga teigiama (ang. *false-positive*) ir klaidingai neigiama (ang. *false-negative*). Šioje diagramoje kategorijai 2, tikroji reikšmė pažymėta žalia spalva, klaidingai teigiama reikšmė lygi geltona spalva pažymėtų langelių verčių sumai, o klaidinga neigiama reikšmė raudonai pažymėtų langelių suma.

lentelė 5 Klaidų matricos pavyzdys

	Category 0	Category 1	Category 2
Category 0	1	2	3
Category 1	4	5	6
Category 2	7	8	9

Tikslingumas yra metrika kuri yra apskaičiuojama taikant šią formulę:

$$Tikslingumas = \frac{visų\ tikrųjų\ reikšmių\ suma}{visų\ reikšmių\ suma} \quad (11)$$

Commented [RŠ14]: Formulės numeruojamos, Palik tarpus tarp formules ir teksto kažkokius, daba nesimato ☺

Ši reikšmė nusako kokia dalis priskirtų kategorijų buvo tinkamai priskirtos atitinkamoms klasėms.

Atsiminimas yra metrika, kuri nusako santykinę vertę tarp tikrųjų verčių ir tikrųjų verčių bei klaidingai neigiamų sumos. Šios metrikos formulė:

$$Atsiminimas = \frac{tikroji\ reikšmė}{tikroji\ reikšmė + klaidingai\ neigiama\ reikšmė} \quad (12)$$

Ši metrika yra labai naudinga kuomet siekiamas rezultatas yra tikslus specifinės klasės kategorizavimas.

Tikslumas yra metrika kuri nurodo santykį tarp tikrųjų reikšmių ir tikrųjų bei klaidingai teigiamų reikšmių sumos. Šios metrikos formulė:

$$Tikslumas = \frac{tikroji\ reikšmė}{tikroji\ \ reikšmė + klaidingai\ \ teigiama\ \ reikšmė} \quad (13)$$

Ši metrika yra naudinga, kuomet norima sumažinti klaidingų teigiamųjų klasifikavimų kiekį.

F-1 taškai, nusako tikslumo ir atsiminimo santykį. Šios metrikos formulė:

$$F - 1\ taškai = 2 * \frac{Tikslumas * Atsiminimas}{(Tikslumas - atsiminimas)} \quad (14)$$

Šio tyrimo pradinėje dalyje tikslingumo metrika buvo naudojama modelių tinklelio paieškos metu, tačiau dėl esamo didelio duomenų disbalanso, buvo pastebėta, kad modeliai treniruojasi spėti pirmą kategoriją, nes taip pasiekiamas didžiausias tikslingumas, tačiau tokiu atveju buvo suprasta kad iš esmės tikslingai atskiriami įprasti įvykiai, o karo nusikaltimų tikslingumas labai žemas.

Tikslumas taip pat buvo atmestas kaip esminė metrika, nes šio tyrimo metu nebuvo rūpinamasi, jeigu ne karo nusikaltimai priskiriami karo nusikaltimams. Tai paremiama tuo, kad jeigu nekalti įvykiai priskiriami karo nusikaltimams, tuomet jie bus išteisinti arba atmesti, ko pasėkoje bus aptikta daug karo nusikaltimų, tačiau jeigu karo nusikaltimas priskiriamas paprastiems įvykiams, tai yra potencialiai didelė žala, nes kaltieji bus laikomi nekaltais. Dėl to šiame darbe galutinis modelio vertinimas bus skaičiuojamas taip:

Commented [RŠ15]: Good, bet labiau akcentuoti, kad nepradingtų vidury pastraipos

$$Rezultatas = \frac{Tikroji\ \ reikšmė\ + klaidingai\ \ neigiama\ \ pirmos\ \ kategorijos\ \ reikšmė}{visi\ \ antrosios\ \ kategorijos\ \ įrašai} \quad (15)$$

2.4.2. Mokymosi proceso vertinimas

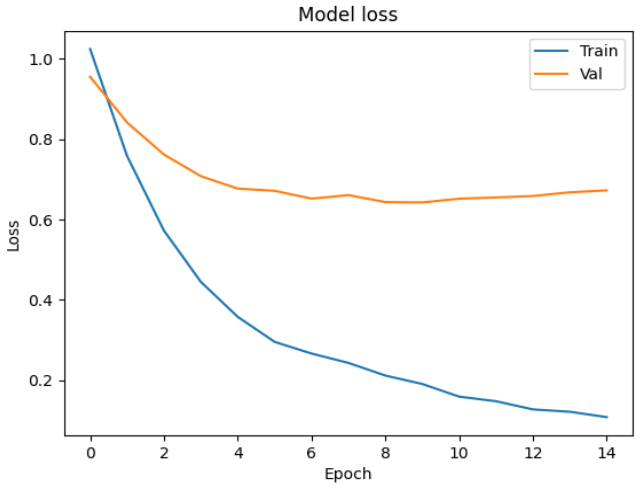
Tiesiaeigio daugiasluoksnio perceptrono ir rekurentinio *LSTM* tinklo treniravimo metu buvo pastebėtas veiksnys, kuomet modelio mokymasis treniravimosi metu gerėja, tačiau atliekant validaciją rezultatai nebegerėja. Tai vadinama perdėtinu mokymosi (ang. *Overfitting*). Šis veiksnys pasireiškia kuomet modelis mokymosi proceso metu pradeda išmokti ne bendrinius pateiktų duomenų sąryšius, o „pripranta“ prie pateiktų duomenų ir atranda sąryšius būdingus būtent duomenims iš kurių yra mokomasi, o ne bendrai situacijai, kurią tie duomenys reprezentuoja. Vienos pagrindinių perdėtino mokymosi priežasčių yra:

- Per didelis modelio kompleksiškumas. Kuomet modelis turi pernelyg didelį kiekį sluoksnių ir neuronų, jis gali pradėti „atsiminti“ duomenis, o ne iš jų mokytis. To pasėkoje dažniausiai modelis gauna labai gerus rezultatus mokymuisi naudotam duomenų rinkiniui, tačiau pateikus naujus duomenis, gaunami rezultatai būna kur kas prastesni.
- Per ilgas mokymosi procesas. Jeigu mokymuisi skiriama per daug epochų, modelis pernelyg iškraipo svorinius rezultatus, kas gali lemti netinkamas duomenų sąryšio interpretacijas.
- Per mažas mokymo duomenų kiekis: Jei mokymo duomenų rinkinys yra per mažas, modelis gali neturėti pakankamai informacijos, kad tinkamai išmoktų bendrąsias duomenų

tendencijas. Vietoj to, jis gali pradėti per daug prisitaikyti prie triukšmo ar kitų nereikšmingų detalių mokymo duomenyse.

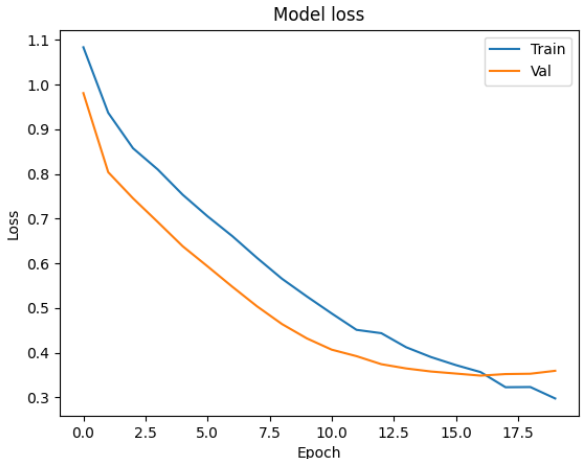
Tyrimo metu, kuomet buvo treniruojami modeliai, jų specifikacijos buvo keičiamos atsižvelgiant į modelio gautus rezultatus ir mokymosi grafikus. Nors buvo išbandytos įvairios pardėtiną mokymąsi mažinančios technikos, nei viena iš jų nesuteikė akivaizdžiai geresnių rezultatų, dėl šios priežasties dedukciniu būdu, buvo priimta išvada, kad tyrimo metu naudotų duomenų rinkinys buvo pernelyg mažos apimties.

Pateiktame paveikslėlyje (pav. 15) galime pastebėti itin greitą perdėtino mokymosi pradžią. Kadangi šis veiksnys pradedamas pastebėti antrojoje epochoje, galime teigti, kad jis nėra sukeltas pernelyg ilgo mokymosi proceso. Taip pat pastebime, kad patikros duomenų klasifikavimo kreivė beveik iš karto nustoja kristi. Tai indikuoja, kad modelis potencialiai yra pernelyg kompleksiškas ir pradeda perdėtiną mokymąsi ankstyvosiose stadijose.



pav. 16 Didelis perdėtinis mokymasis

Tuo tarpu pav. 16 pateiktas grafikas, vaizduojantis mokymosi procesą, kinta kur kas stabiliau tiek patikros, tiek treniravimosi atvejais. Tai indikuoja, kad modelio kompleksiškumas yra santykinai tinkamas turimiems duomenims. Taip pat pastebima, kad patikros ir treniravimo procesai pasiekia optimalią reikšmę esant 16 epochai. Tai nusako, kad pasirinktas mokymosi proceso ilgis yra tinkamas ir procesas buvo nutrauktas laiku, modeliui tik pradėjus atrasti nepageidautinus sąryšius. Kadangi vis tiek pastebima, kad mokymąsi indikuojantis praradimas (ang. *Loss*) nepasiekia labai žemų reikšmių, prieš pradėdamas perdėtina mokymąsi. Tai gali būti laikoma pernelyg mažo duomenų rinkinio požymiu.



pav. 17 Mažas perdėtinis mokymasis

2.4.3. Modelių palyginimas

Įvykdžius tyrimą ir gavus visų apmokytų modelių rezultatus buvo atliktas modelių palyginimas. Kiekvienam mašininio mokymosi modeliui buvo pasirinktas geriausią rezultatą pateikusi variacija. Pirminis palyginimas vykdytas pagal testavimo duomenų gautas atsiminimo reikšmes, tačiau bendruoju atveju geriausias modelis bus vertinamas pagal šiuos kriterijus:

- Mažiausias kiekis antros kategorijos duomenų priskirtą nulinei kategorijai,
- Mokymosi laikas,
- Modelio kompleksiškumas.

2.4.3.1. Naïve Bayes geriausias rezultatas

Tyrimo metu geriausias *Naïve Bayes* klasifikatoriaus rezultatas buvo 90 procentų tikslumas antros kategorijos duomenų atsiminimo. Iš viso šiame modelyje 6 duomenų įrašai priklausantys karo nusikaltimų kategorijai buvo priskirti ne karo nusikaltimams. Jeigu atliksime modifikuotą rezultato skaičiavimą pagal (15) formulę gausime:

$$Rezultatas = \frac{102 + 5}{102 + 5 + 6} = \sim 0.947$$

Šio modelio mokymosi laikas yra ypač greitas ir turimų duomenų apmokymas truko keletą sekundžių. Taip pat šis modelis yra itin paprastas, nes neturi didelės parametrų įvairovės. Šie rezultatai gauti ant natūraliai balansuoto duomenų rinkinio.

```
Best Alpha: 0.21416145373379775
Best Recall for Category 2: 0.8686708860759493
Classification Report:
              precision    recall  f1-score   support

         0.0       0.96      0.71      0.81      1206
         1.0       0.61      0.79      0.69       418
         2.0       0.33      0.90      0.49       113

    accuracy                           0.74      1737
   macro avg       0.63      0.80      0.66      1737
weighted avg       0.83      0.74      0.76      1737

Confusion Matrix:
[[854 203 149]
 [ 32 331  55]
 [  6   5 102]]
Accuracy Score: 0.7409326424870466
```

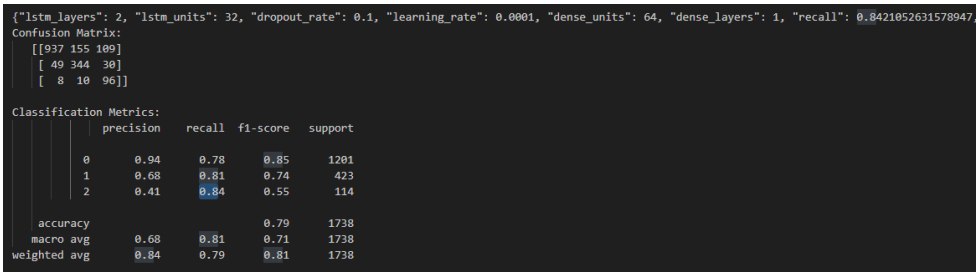
pav. 18 Naive Bayes geriausias rezultatas

2.4.3.2. LSTM geriausias rezultatas

Geriausias *LSTM* modelio gautas rezultatas buvo 84 procentai. Taikant šį rekurentinį neuroninį tinklą 8 atvejais karo nusikaltimai buvo priskirti ne karo nusikaltimams. Pritaikius modifikuotą rezultatų skaičiavimo (15) formulę gauname:

$$Rezultatas = \frac{96 + 10}{96 + 10 + 8} = \sim 0{,}9298$$

Nors gautas rezultatas yra panašus į tą kuris buvo gaunamas taikant *Naive Bayes* klasifikatorių, *LSTM* neuroninis tinklas reikalavo kur kas didesnių resursų. Paties galutinio modelio apmokymo laikas buvo apie 1 valanda 30 minučių. Taip pat šis modelis yra santykinai sudėtingas ir turintis kiekį keičiamų parametrų. To pasėkoje tinklelio paieškos procesas, kuriuo buvo atrastas geriausias rezultatas truko apie 70 valandų.



pav. 19 LSTM modelio geriausias rezultatas

Geriausių rezultatų gavimui *LSTM* modelis turėjo šį hiperparametrų rinkinį:

- lstm sluoksnių skaičius – 2,
- lstm neuronų skaičius – 32,
- atkritimo tempas – 0.1,
- mokymosi tempas – 0.0001,
- tankiųjų sluoksnių skaičius – 1,
- tankiųjų neuronų skaičius – 64,
- mokymosi paketo dydis – 64

Taip pat buvo panaudota L2 reguliacija ir klasių svoriai, bei duomenys buvo mokomi ant natūraliai balansuoto, bet mažo, duomenų rinkinio. Modeliui taip pat buvo pritaikyta bekryptė architektūros modifikacija.

2.4.3.3. Daugiasluoksnio perceptrono geriausias rezultatas

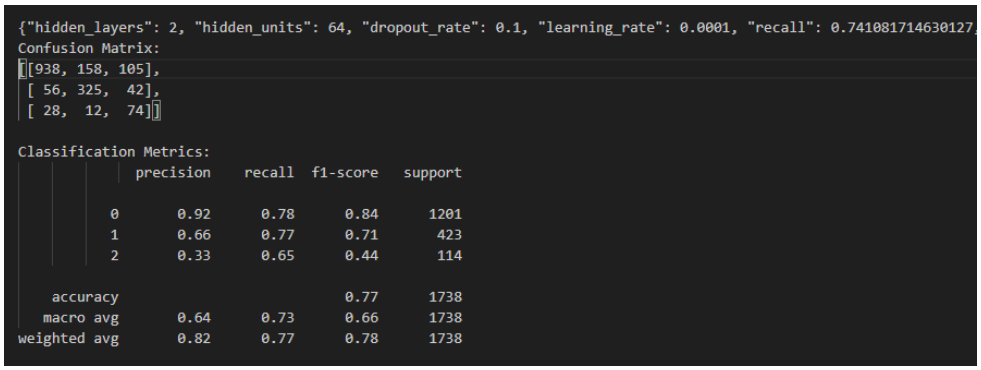
Geriausias MLP modelio gautas rezultatas buvo 65 procentai. Naudojant šį tiesioginį neuroninį tinkle 28 karo nusikaltimai buvo priskirti ne karo nusikaltimams. Pritaikius modifikuotą rezultatų skaičiavimo (15) formulę gauname:

$$Rezultatas = \frac{74 + 12}{74 + 12 + 28} = \sim 0.7544$$

Kaip galima pastebėti rezultatai nėra geri. Nors šis modelis yra mažiau sudėtingas nei *LSTM*, šio modelio apmokymas taikant tinklelio atsitiktinę paiešką truko apie 45 valandas. Geriausio rezultatą gavęs modelis turėjo šiuos hyperparametrus:

- paslėptų sluoksnių skaičius – 2,
- paslėptų neuronų skaičius – 64,
- atkritimo tempas – 0.1,
- mokymosi tempas – 0.0001,
- mokymosi paketo dydis – 64

Taip pat buvo panaudota L2 reguliacija ir klasių svoriai, bei duomenys buvo mokomi ant natūraliai balansuoto, bet mažo, duomenų rinkinio.



pav. 20 MLP modelio geriausias rezultatas

2.4.3.4. Rezultatų palyginimas

Žemiau esančioje lentelėje (lentelė 6) yra pateikiamas geriausių, kiekvienos tirtos rūšies, mašininio mokymosi algoritmų statistika ir rezultatai. Atsižvelgiant į šiuos duomenis galima daryti išvadą, kad esant santykinai mažiems duomenų rinkiniams, naudotiems šio tyrimo metu, efektyviausias iš visų modelių yra *Naïve Bayes* klasifikatorius. Taip pat galime pastebėti, kad *LSTM* modelis rezultatų atžvilgiu irgi gavo panašias reikšmes, tačiau jo sudėtingumas ir mokymosi laikas padaro jį nenaudingu sprendžiamam uždaviniui. Nors šio darbo eigoje *LSTM* modelis nepasižymėjo geru reikiamų resursų ir gautų rezultatų santykiu, tačiau galutinis rezultatas vis tiek buvo tenkinamas. Galiausiai galime pastebėti, kad daugiasluoksnio perceptrono gauti rezultatai yra prasčiausi iš visų. Šis modelis netik gavo žemiausią rezultatų vertinimą, bet ir reikalavo santykinai didelių resursų.

lentelė 6 Modelių rezultatai

Modelis	**Bendras mokymo laikas**	**Karo nusikaltimų teisingos klasifikacijos tikimybė**
Naive Bayes	~1/60h	~0.947
LSTM	~70h	~0.9298
MLP	~45h	~0.7544

2.5.Tyrimo rezultatų taikymas ir nauda

Atlikus tyrimą su pasirinktais mašininio mokymosi algoritmais, buvo pastebėta kad karo nusikaltimų klasifikavimui labiausiai tinkamas yra *Naïve Bayes* klasifikatorius. Šio algoritmo naudingumą lemia keletas veiksnių:

- Užduoties jautrumas. Kadangi karo nusikaltimai yra labai aiškiai apibrėžiami ir jiems priskiriamų įvykių, tekstinių ar žodinių variacijų kiekis yra labai nedidelis galima daryti prielaidą, jog statistiniai duomenų sąryšiai yra tikslingesni, nei kontekstiniai. Tai pastebima iš to, kad didinant *MLP* ir *LSTM* modelių kompleksiškumą gaunami prastėjantys rezultatai.
- Mažas duomenų kiekis. Kadangi karo nusikaltimai yra santykinai reti veiksniai civilizuotame pasaulyje, juos nusakančių įrašų kiekis taip pat yra nedidelis. To pasėkoje *LSTM* ir *MLP* modeliai kenčia nuo perdėtino mokymosi.
- Resursų poreikis. Esant situacijai, kuomet tyrimas vykdomas naudojantis asmenine įranga, pastebimas labai didelis resursų poreikio skirtumas lyginant tikimybinį *Naïve Bayes* modelį su neuroniniais tinklais.

Tyrimo metu buvo pastebėta, kad karo nusikaltimų kontekste, modelio rezultatų vertinimas pagal tikslingumą yra netinkamas, ypač su nebalansuotais duomenimis. Kuomet modeliai dirbdami su nesubalansuotais duomenimis, greitai pradeda treniruotis tiksliai nustatyti daugumos grupę, ko pasėkoje esant poreikiui tirti mažumos grupes, gaunami labai prasti rezultatai.

Bendruoju atveju, atliekant tyrimą buvo pastebėta, kad visi mašininio mokymosi algoritmai gauna geresnius rezultatus, kuomet yra apmokomi ant mažesnių, bet natūraliai balansuotų, duomenų rinkinių, nei ant didelės apimties nebalansuotų duomenų. Taip pat pastebėta, kad šio pobūdžio problemoms spręsti tiesioginio sklidimo neuroniniai tinklai nėra efektyvūs. Tai potencialiai vyksta dėl to, kad šio pobūdžio algoritmai neturi galimybės išlaikyti duomenų kontekstą. Galiausiai, neuroninių tinklų apmokymo metu, pastebėta, kad didinant modelių kompleksiškumą, bet nekeičiam mokymosi greičio, rezultatai ima prastėti, tačiau kuomet didinamas kompleksiškumas ir kartu didinamas mokymosi tempas, gaunami rezultatai gerėja. Tuo remiantis galima teigti, kad priklausomai nuo keliamos problemos, optimalus neuroninis tinklas išlaiko tam tikrą santykį tarp savo kompleksiškumo ir mokymosi tempo.

Šio tyrimo metu gauti vektoriniai svoriai gali būti naudojami, kaip ateities tyrimų ar sistemų pradiniai svoriai. Taip pat ištirti modeliai gali būti taikomi, kaip pirminis filtras, kurį taikant galima efektyvus pirminis karo nusikaltimo įrašų išrinkimas iš didelių duomenų rinkinių. Galiausiai, kadangi pradiniai duomenys buvo žymimi pagal kontekstą, o ne pagal specifinių žodžių buvimą, apmokyti modeliai gali padėti atskirti propagandinius įrašus internete, ypač tokiose situacijose kai yra skelbiamas karo nusikaltimas, tačiau įvykio turinys neatitinka šios apibrėžties.

Išvados

Commented [RŠ16]: Atskirame lape

Išvadų nespėjau ir analizės rez

1.1. Atlikus mašininio mokymosi algoritmų, skirtų tekstinių duomenų klasifikavimui, analizę buvo priimtas sprendimas naudoti Naiviojo Bajeso, *LSTM* ir daugiasluoksnio perceptrono modelius. Naiviojo Bajeso modelis pasirinktas dėl literatūroje minimo didelio efektyvumo dirbant su mažos apimties tekstiniais failais. LSTM ir daugiasluoksnis perceptronas pasirinkti siekiant palyginti tiesiaeigių ir rekurentinių tinklų efektyvumą klasifikuojant tekstinius failus, karo nusikaltimų kontekste.

1.2. Atlikus technologijų analizę, skirtą modelių įgyvendinimui, pasirinkta *Python* programavimo kalba. Pasirinkimas yra paremtas tuo, kad *Python* kalba yra plačiau naudojama dirbtinio intelekto sferoje ir turi daugiau šiam darbui pritaikytų įrankių. Taip pat, atsižvelgiant į asmeninę patirtį, su *Python* kalba bus pasiektas aukštesnis našumas dėl jau turimų šios programavimo kalbos žinių. Atlikus technologinę analizę, skirtą duomenų išgavimui, pasirinkta *Python* programavimo kalboje naudojama „*My beautiful soup*“ biblioteka. Turimos žinios, bei galimybė lengvai modifikuoti duomenų surinkimo parametrus, leis efektyviau rinkti duomenis. Taip pat, kadangi kiti įrankiai yra mokami arba riboti, pasirinkus minėtą biblioteką sumažėjo baigiamojo darbo įgyvendinimo kaštai.

1.3. Darbo metu visi pasirinkti klasifikavimo modeliai buvo sėkmingai įgyvendinti ir ištestuoti. *Naive Bayes* modeliui atliktos keturios tinklelio paieškos sudariusios 4720 skirtingų bandymų. Su *LSTM* naudojant atsitiktinę tinklelio paiešką atlikti 88 bandymai, o su daugiasluoksniu perceptronu 714 bandymai. Neuroniniais tinklais paremtiems modeliais buvo taikomos dirbtinio balansavimo ir relugiarizacijos technikos ir klasių svoriai. Visi modeliai buvo bandomi su pilnu nebalansuotu ir daliniu subalansuotu duomenų rinkiniu.

1.4. Atlikus tyrimą ir įvertinus gautus rezultatus buvo nustatyta, kad turimam duomenų rinkiniui klasifikuoti yra tinkami visi išbandyti algoritmai, tačiau efektyviausias yra *Naive Bayes*, kuris pasiekė 94,7% tikslumą. Visi modeliai gavo geresnius rezultatus naudojant dalinį subalansuotą duomenų rinkinį. Geriausias LSTM modelio rezultatas buvo 92,98%, tuo tarpu daugiasluoksnis perceptronas pasiekė 75,44% tikslumą. Kadangi geriausias rezultatas pasiektas tikimybinio mašininio mokymosi modelio, priimta išvada, kad karo nusikaltimus aprašančiuose tekstuose dažnai pasikartoja specifiniai žodžiai ar frazės, kurių įvairovė yra santykinai maža.

Šaltiniai

1. Daniel Jurafsky, James H. Martin (2016). *Speech and Language Processing*. Prieiga per internetą: https://web.stanford.edu/~jurafsky/slp3/4.pdf
2. Tony Mullen, Nigel Collier (2004) *Sentiment analysis using support vector machines with diverse information sources.* Prieiga per internetą: https://aclanthology.org/W04-3253.pdf
3. Himani Bhavsar, Mahesh H. Panchal (2012) *A Review on Support Vector Machine for Data Classification.* Prieiga per internetą: *http://text2fa.ir/wp-content/uploads/Text2fa.ir-A-Review-on-Support-Vector-Machine.pdf*
4. Ameet V Joshi (2019).*Machine Learning and Artificial Intelligence*
5. Siddharth Sharma, Simone Sharma, Anidhya Athaiya (2020) *Activation functions in neural networks.* Prieiga per internetą: *https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf*
6. J. Naskath, G. Sivakamasundari, A. Alif Siddiqua Begum (2022) *A Study on Different Deep Learning Algorithms Used in Deep Neural Nets: MLP SOM and DBN*. Prieiga per internetą: *https://link.springer.com/article/10.1007/s11277-022-10079-4*
7. Kiran Baktha, B K Tripathy (2017) *Investigation of Recurrent Neural Networks in the field of Sentiment Analysis*. Prieiga per internetą: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8286763&tag=1*
8. Michael Nielsen (2013) *Neural Networks and Deep Learning*. Prieiga per internetą: *https://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf*
9. Batta Mahesh (2020) *Machine Learning Algorithms - A Review*. Prieiga per internetą: *https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762*
10. Osisanwo F.Y., Akinsola J.E.T, Awodele O., Hinmikaiye J. O., Olakanmi O.,Akinjobi J. (2017) *Supervised Machine Learning Algorithms: Classification and Comparison*. Prieiga per internetą: *https://www.researchgate.net/profile/J-E-T-Akinsola/publication/318338750*
11. Hannes Muellerab, Andre Groegerb, Jonathan Hershd, Andrea Matrangade, Joan Serratf (2020) *Monitoring war destruction from space usingmachine learning*. Prieiga per internetą: *https://www.pnas.org/doi/epdf/10.1073/pnas.2025400118*

12. Witold Pedrycz, Shyi-Ming Chen (2020). *Deep Learning: Concepts and Architectures*. Prieiga per internetą:

https://www.researchgate.net/publication/336904955_Deep_Learning_Architectures

13. Walaa Medhat, Ahmed Hassan, Hoda Korashy (2014). *Sentiment analysis algorithms and applications: A survey*. Prieiga per internetą:

https://www.sciencedirect.com/science/article/pii/S2090447914000550

14. Serafeim Loukas (2020) *Text Classification Using Naive Bayes: Theory & A Working Example*. Prieiga per internetą:

https://towardsdatascience.com/text-classification-using-naive-bayes-theory-a-working-example-2ef4b7eb7d5a

15. Varun Ojha, Ajith Abraham, Vaclav Snasel (2017). *Metaheuristic Design of Feedforward Neural Networks: A Review of Two Decades of Research.* Prieiga per internetą:

https://www.researchgate.net/publication/313556406_Metaheuristic_Design_of_Feedforward_Neural_Networks_A_Review_of_Two_Decades_of_Research

16. Simon Haykin (1999) *Neural Networks and Learning Machines.* Prieiga per internetą:

https://cours.etsmtl.ca/sys843/REFS/Books/ebook_Haykin09.pdf

17. Mohammed Zeeshan Mulla (2020) *Cost, Activation, Loss Function|| Neural Network|| Deep Learning. What are these?* . Prieiga per internetą:

https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de

18. Filippo Maria Bianchi1a, Enrico Maiorinob, Michael C. Kampffmeyera, Antonello Rizzib, Robert Jenssena (2018). *An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting.* Prieiga per internetą:

https://arxiv.org/pdf/1705.04378.pdf

19. Robin M. Schmidt (2019). *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview.* Prieiga per internetą:

https://arxiv.org/pdf/1912.05911.pdf%20%5b20.pdf

20. Marta Bistron, Zbigniew Piotrowski (2017). *Artificial Intelligence Applications in Military Systems and Their Influence on Sense of Security of Citizens*

21. Ofice of General Defence Councel Department Of Defense (2016) *Department of Defense law of war manual*. Prieiga per internetą: https://dod.defense.gov/Portals/1/Documents/pubs/DoD%20Law%20of%20War%20Manual%20-%20June%202015%20Updated%20Dec%202016.pdf?ver=2016-12-13-172036-190

Priedai

```python
def recall_category_2(y_true, y_pred):
    return recall_score(y_true, y_pred, labels=[2], average='macro')

train_data = pd.read_excel("Data/Datasets/NB.xlsx", sheet_name="Sheet2")

train_data = train_data.dropna(subset=['text_cleaned', 'category'])

X_train = train_data['text_cleaned']
y_train = train_data['category']

vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)

k = 1500
selector = SelectKBest(chi2, k=k)
selector.fit(X_train_vec, y_train)
X_train_vec_selected = selector.transform(X_train_vec)

param_grid = {
    'alpha': np.logspace(start=-2, stop=1.176, num=1180, base=10)
}

nb_classifier = MultinomialNB()
custom_scorer = make_scorer(recall_category_2)
grid_search = GridSearchCV(nb_classifier, param_grid=param_grid, cv=2, n_jobs=-1, scoring=custom_scorer)

grid_search.fit(X_train_vec_selected, y_train)

best_model = grid_search.best_estimator_
best_score = grid_search.best_score_

joblib.dump((vectorizer, selector, best_model), 'NB/parametersData/best_NB.joblib')

with open("NB/parametersData/NB_grid_results_recall.txt", "w") as f:
    for i, params in enumerate(grid_search.cv_results_['params']):
        alpha = params['alpha']
        mean_test_score = grid_search.cv_results_['mean_test_score'][i]
        std_test_score = grid_search.cv_results_['std_test_score'][i]
        f.write(f"Alpha: {alpha}, Mean Test Score: {mean_test_score}, Std Test Score: {std_test_score}\n")

with open("NB/parametersData/Nb_results_best_recall.txt", "w") as f:
    f.write(f"Best Alpha: {best_model.alpha}\n")
    f.write(f"Best Recall for Category 2: {best_score}\n")
```

A priedas. Naïve Bayes tinklelio paieška

```python
def create_model(hidden_layers=1, hidden_units=16, dropout_rate=0.5, learning_rate=0.001, patience=5, batch_size=64, class_weights=None):
    model = Sequential()
    model.add(Embedding(vocab_size, embedding_dim, weights=[embedding_matrix], input_length=max_length, trainable=True))
    model.add(Flatten())
    model.add(Dense(hidden_units, input_shape=(max_length,)))
    model.add(LeakyReLU())
    model.add(Dropout(dropout_rate))

    for _ in range(hidden_layers - 1):
        model.add(Dense(hidden_units))
        model.add(LeakyReLU())
        model.add(Dropout(dropout_rate))

    model.add(Dense(3, activation='softmax'))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='categorical_crossentropy', metrics=[tf.keras.metrics.Recall(name='recall')])

    model_info = {
        'hidden_layers': hidden_layers,
        'hidden_units': hidden_units,
        'dropout_rate': dropout_rate,
        'learning_rate': learning_rate
    }

    X_train, X_test, y_train, y_test = train_test_split(padded_sequences, categories, test_size=0.2, random_state=42, shuffle=True)
    early_stop = EarlyStopping(monitor='val_loss', patience=patience, verbose=1, mode='min')

    if class_weights is not None:
        history = model.fit(X_train, y_train, epochs=50, batch_size=batch_size, validation_split=0.2, callbacks=[early_stop], class_weight=class_weights)
    else:
        history = model.fit(X_train, y_train, epochs=50, batch_size=batch_size, validation_split=0.2, callbacks=[early_stop])


    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")

    plt.figure()
    plt.plot(history.history['recall'])
    plt.plot(history.history['val_recall'])
    plt.title('Model recall')
    plt.ylabel('Recall')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Val'], loc='upper left')
    plt.savefig(f'Data/Graphs/mlp_allData3catCV2/MLP_RECALL_{timestamp}.png')
    plt.close()

    plt.figure()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Val'], loc='upper right')
    plt.savefig(f'Data/Graphs/mlp_allData3catCV2/MLP_LOSS_{timestamp}.png')
    plt.close()

    _, test_recall = model.evaluate(test_padded_sequences, test_categories, verbose=0)

    test_y_pred = model.predict(test_padded_sequences)
    test_y_pred_classes = np.argmax(test_y_pred, axis=-1)
    test_y_true_classes = np.argmax(test_categories, axis=-1)

    test_conf_matrix = confusion_matrix(test_y_true_classes, test_y_pred_classes)
    test_classification_metrics = classification_report(test_y_true_classes, test_y_pred_classes)
    model_info['recall'] = test_recall
    model_info['timestamp'] = timestamp
    model_info['epochs'] = len(history.history['recall'])
    model_info['patience'] = early_stop.patience
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    model_info_file = f'MLP/MLPresults/model_info_{timestamp}.txt'
    with open(model_info_file, 'a') as outfile:
        json.dump(model_info, outfile)
        outfile.write('\n')
        outfile.write('Confusion Matrix:\n\t')
        outfile.write(np.array2string(test_conf_matrix, separator=', '))  # convert numpy array to string
        outfile.write("\n\nClassification Metrics:\n")
        outfile.write(test_classification_metrics)
        outfile.write('\n')
        outfile.write('\n')

    return model

X_train, X_test, y_train, y_test = train_test_split(padded_sequences, categories, test_size=0.2, random_state=42)

y_integers = np.argmax(y_train, axis=1)
class_weights = class_weight.compute_sample_weight('balanced', y_integers)
class_weights_dict = dict(enumerate(class_weights))
params = {
    'hidden_layers': [3],
    'hidden_units': [64,128],
    'dropout_rate': [0.5],
    'learning_rate': [0.0001],
    'batch_size': [128],
    'patience': [5]
}
model = KerasClassifier(build_fn=create_model, hidden_layers=None, hidden_units=None, dropout_rate=None, learning_rate=None, patience=5, class_weights=class_weights_dict, verbose=1)
random_search = RandomizedSearchCV(estimator=model, param_distributions=params, n_iter=10, scoring=recall_scorer, cv=2, n_jobs=-1, random_state=42)
random_search_result = random_search.fit(X_train, y_train)
```

B priedas. MLP atsitiktinė tinklelio paieška

```python
def create_model(lstm_layers=8, lstm_units=16, dropout_rate=0.3, dense_layers=3, dense_units=32, learning_rate=0.001, patience=20, batch_size=64):
    model = Sequential()
    model.add(Embedding(vocab_size, embedding_dim, weights=[embeddings_matrix], input_length=max_length, trainable=True))

    for i in range(lstm_layers):
        model.add(Bidirectional(LSTM(lstm_units, return_sequences=(i < lstm_layers - 1))))

    model.add(Dropout(dropout_rate))

    for _ in range(dense_layers):
        model.add(Dense(dense_units))
        model.add(LeakyReLU(alpha=0.001))
        model.add(Dropout(dropout_rate))

    model.add(Dense(3, activation='softmax'))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='categorical_crossentropy', metrics=[Recall()])

    model_info = {
        'lstm_layers': lstm_layers,
        'lstm_units': lstm_units,
        'dropout_rate': dropout_rate,
        'learning_rate': learning_rate,
        'dense_units': dense_units,
        'dense_layers': dense_layers
    }
    print(f"Model parameters: {model_info}")
    if tuple(model_info.items()) not in unique_params:
        unique_params.add(tuple(model_info.items()))
        X_train, X_val, y_train, y_val = train_test_split(padded_sequences_train, categories_train, test_size=0.5, random_state=42, shuffle=True)
        early_stop = EarlyStopping(monitor='val_loss', patience=patience, verbose=1, mode='min')
        history = model.fit(X_train, y_train, epochs=55, batch_size=batch_size, validation_data=(X_val, y_val), callbacks=[early_stop], class_weight=class_weight_dict)
        timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")

        plt.figure()
        plt.plot(history.history['recall'])
        plt.plot(history.history['val_recall'])
        plt.title('Model recall')
        plt.ylabel('Recall')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Val'], loc='upper left')
        plt.savefig(f'Data/Graphs/lstm_allData3catGloVeBidirectionalLeakyReLu_recall/LSTM_Recall_{timestamp}.png')
        plt.close()

        plt.figure()
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('Model loss')
        plt.ylabel('Loss')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Val'], loc='upper right')
        plt.savefig(f'Data/Graphs/lstm_allData3catGloVeBidirectionalLeakyReLu_recall/LSTM_LOSS_{timestamp}.png')
        plt.close()
        y_pred = model.predict(X_test)
        test_recall = custom_recall_scorer(y_test, y_pred)
        model_info['recall'] = test_recall
        model_info['timestamp'] = timestamp
        model_info['epochs'] = len(history.history['recall'])
        model_info['patience'] = early_stop.patience
    y_pred_classes = np.argmax(y_pred, axis=-1)
    y_test_classes = np.argmax(y_test, axis=-1)
    conf_matrix = confusion_matrix(y_test_classes, y_pred_classes)
    conf_matrix_str = str(conf_matrix).replace('\n', '\n\t')
    classification_metrics = classification_report(y_test_classes, y_pred_classes)

    with open('LSTM/model_info_recall.txt', 'a') as outfile:
        json.dump(model_info, outfile)
        outfile.write('Bidirectional, no SMOTE CW, small but balanced dataset, leakyRelu\n')
        outfile.write('Confusion Matrix:\n\t')
        outfile.write(conf_matrix_str)
        outfile.write("\n\nClassification Metrics:\n")
        outfile.write(classification_metrics)
        outfile.write('\n')
    return model
X_train, y_train = padded_sequences_train, categories_train
X_test, y_test = padded_sequences_test, categories_test

params = {
    'lstm_layers': [2,3],
    'lstm_units': [16,32],
    'dropout_rate': [0.1,0.3,0.5],
    'learning_rate': [0.0001,0.001],
    'dense_layers': [2,3],
    'dense_units': [16,32],
    'batch_size': [32,64],
    'patience': [15]
}
model = KerasClassifier(build_fn=create_model, lstm_layers=None, lstm_units=None, dropout_rate=None, dense_layers=None, dense_units=None, learning_rate=None, patience=20, verbose=1)
random_search = RandomizedSearchCV(estimator=model, param_distributions=params, n_iter=4, scoring=custom_scorer, cv=2, n_jobs=-1, random_state=42)
random_search_result = random_search.fit(X_train, y_train)
```

C priedas. LSTM atsitiktinė tinklelio paieška

Debris from a destroyed drone has hit Kyiv's northeastern residential district of Desnianskiy, the city's mayor says. Vitali Klitschko said the debris hit a road and damaged a building. One person - a 19-year-old man - has been taken to hospital, he added. It is the second consecutive night of Russian attacks on Ukraine's capital - on New Year's Eve, missiles were fired at the city.	1
An 18-year-old man has died after a house was hit by Russian forces in central Kherson, according to regional governor Yaroslav Yanushevych. The man died "on the spot", Mr Yanushevych said. The city of Kherson and parts of the surrounding area were liberated from Russian control in November, but Russian forces who fled to the other side of the nearby Dnipro river have continued to pound the area.	2
Oleksiy Danilov, the secretary of the national security and defence council of Ukraine has added his response to the calls for a ceasefire, calling Russia "devils". On Twitter, he wrote (translated from Ukrainian): "What does a flock of small Kremlin... devils have to do with the Christian holiday of Christmas? "Who will believe the abomination that kills children, shells maternity hospitals, tortures prisoners? Ceasefire? Lies and hypocrisy. "We will gnaw at you in the singing silence of the Ukrainian night."	0

D priedas. Kategorizuotų duomenų pavyzdys.