VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Papildomi vizualizavimo skyriai

Praktinis darbas Nr. 4

Darbą atliko:

Roland Gulbinovič ir Matas Amšiejus

Duomenų mokslas III kursas, 2 grupė

Vilnius 2022

Turinys

Įvadas ... 3
Duomenys ... 3
Tyrimo tikslas ... 3
Tyrimo uždaviniai ... 3
AVK be dimensijos mažinimo ... 4
Optimalių parametrų radimas ... 4
Klasifikatoriaus modeliavimas ... 5
AVK su t-SNE ... 7
Optimalių parametrų radimas ... 7
Klasifikatoriaus modeliavimas ... 7
Atsitiktiniai miškai be t-SNE ... 9
Optimalių parametrų radimas ... 9
Klasifikatoriaus modeliavimas ir rezultatai ... 9
Atsitiktiniai miškai su t-SNE ... 11
Optimalių parametrų radimas ... 11
Klasifikatoriaus modeliavimas ... 11
Klasifikavimo kokybės vertinimas ... 13
Kryžminio validavimo strategija ... 13
ROC kreivės ir AUC matas ... 14
Išvados ... 15
Šaltiniai ... 16
Priedai ... 16
Lentelės ... 16
Kodas ... 19

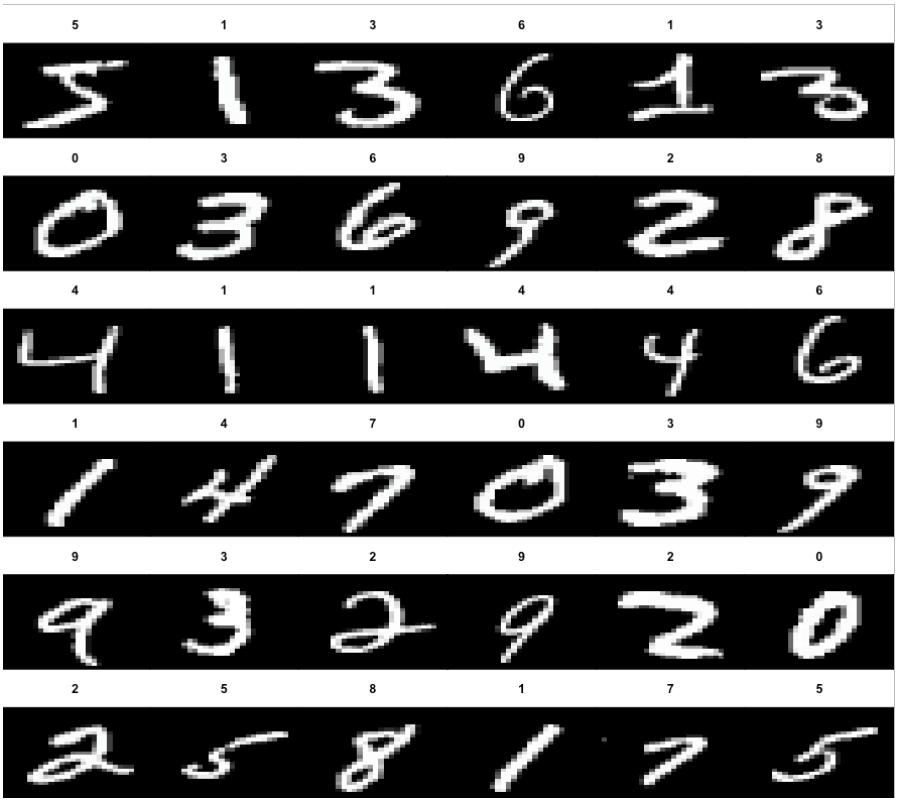
Įvadas

Duomenys

Darbui naudojama *MNIST* duomenų aibė. Tai duomenų rinkinys, sudarytas iš 60 000 mažų kvadratinių 28 × 28 pikselių pilkų atspalvių vaizdų su ranka rašytais skaitmenimis nuo 0 iki 9. Duomenų atributai:

- *784* stulpeliai atitinkantis kiekvienam pikseliui, kiekvieno stulpelio reikšmė parodo kaip stipriai nuspalvintas pikselis. [0; 255];
- *Label* – skaitmuo nuo 0 iki 9.

Kadangi reikia duomenų aibė su dviem klasėmis, pasirinkome skaitmenis 5 ir 8, nes iš praeitų tyrimų žinome, kad šie skaitmenys yra sunkiai atskiriami. Iš viso išsirinkome po 1000 įrašų kiekvieno skaitmens. Taigi, darbui naudosime tik 2000 skaitmenų imtį.



1 pav. MNIST duomenų aibė.

Tyrimo tikslas

Šio darbo tikslas yra ištirti skirtingus klasifikatorius ir rasti geriausią.

Tyrimo uždaviniai

- Pasirinkti duomenų aibę klasifikavimui;

- Padalinti duomenų aibę į mokymo ir validavimo aibes;
- Apmokyti pasirinktą klasifikatorių ir suklasifikuoti duomenis;
- Apskaičiuoti sumaišymo matricas, accurary, precision, recall, F1- matus;
- Fiksuoti optimalius pasirinkto klasifikatoriaus parametrus;
- Vizualizuoti klasifikavimo rezultatus;
- Įvertinti klasifikavimo kokybės rezultatus naudojant:
    - Išlaikymo validavimo strategiją;
    - Kryžminio validavimo strategiją.
- Vizualizuoti klasifikavimo rezultatus ROC kreivių grafikais, apskaičiuoti AUC matą;
- Pateikti apibendrintas išvadas.

AVK be dimensijos mažinimo

AVK (angl. Support vector classificator) siekia surasti tiesę (hiperplokštumą, kai dimensija didesnė už 2) tokią, kad atstumas nuo jos iki dviejų taškų (paraštė), priklausančių skirtingoms klasėms, būtų didžiausias [1], [2].

Optimalių parametrų radimas

Norint gauti geriausius rezultatus, mums reikia pasirinkti optimalius algoritmo parametrus. Susikursime 4 modelius su skirtingais branduoliais ir geriausiais pasirinkto branduolio parametrais.

Pirmam branduoliui **„*Linear*“** galima pasirinkti 2 parametrus: kaina *price* ir ϵ. Pritaikę mūsų modelį su skirtingomis parametrų kombinacijomis, gauname, kad klasifikavimo tikslumas nuo šių parametrų nekito, todėl parinkome parametrus, kurie buvo numatyti funkcijos: *kaina* = 1 ir $\epsilon = 0.1$. Šio modelio tikslumas yra lygus 0.9225 ir F-1 = 0.92249.

Pastaba: toliau nenaudosime *price* ir ϵ parametrų, nes atlikus kelis papildomus eksperimentus, jų pokyčiai algoritmų tikslumų nekeitė, o skaičiavimo laiko pridėdavo eksponentiškai.

Branduoliui **„*Polynomial*“** galima pasirinkti 3 parametrus: *laipsnis*, γ ir nulinis koeficientas. Pritaikę mūsų modelį su skirtingomis parametrų kombinacijomis geriausią klasifikavimo tikslumą gauname su tokiais parametrais: *laipsnis* = 3, $\gamma = 0.001$ ir koeficientas = -0.5 . Šio modelio tikslumas yra lygus 0.9725 ir F-1 = 0.972 (9 lentelė).

Branduoliui **„*Radial Basis*“** galima pasirinkti vieną parametrą: γ. Pritaikę mūsų modelį su skirtingomis parametrų kombinacijomis, gauname, kad visada klasifikavimo tikslumas yra lygus 0.5. Šio branduolio tipo su šiais duomenimis naudoti negalima (10 lentelė).

Branduoliui **„*Sigmoid*“** galima pasirinkti 2 parametrus: nulinis koeficientas ir γ. Pritaikę mūsų modelį su skirtingomis parametrų kombinacijomis vėl gauname, kad tikslumas visada yra

lygus 0.5, pusė duomenų yra vertinami visiškai neteisingai (ta pati problema kaip ir su „Radial basis“ modeliu) (11 lentelė).

Matome, kad geriausius rezultatus gauname naudojant trečio laipsnio polinominį branduolį, nes pastarojo testinių duomenų aibės rezultatai buvo geriausi.

Klasifikatoriaus modeliavimas

Sumodeliuojame trečio laipsnio polinominį AVK modelį. Pritaikome jį ant mūsų testavimo aibės ir gauname tokius rezultatus.

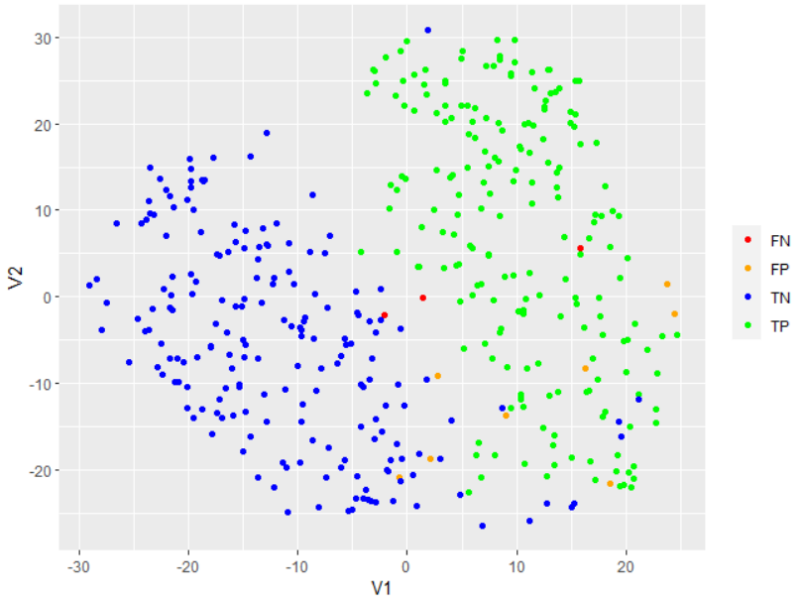
1 lentelė. Sumaišymo matrica

	5	8
5	197	8
8	3	192

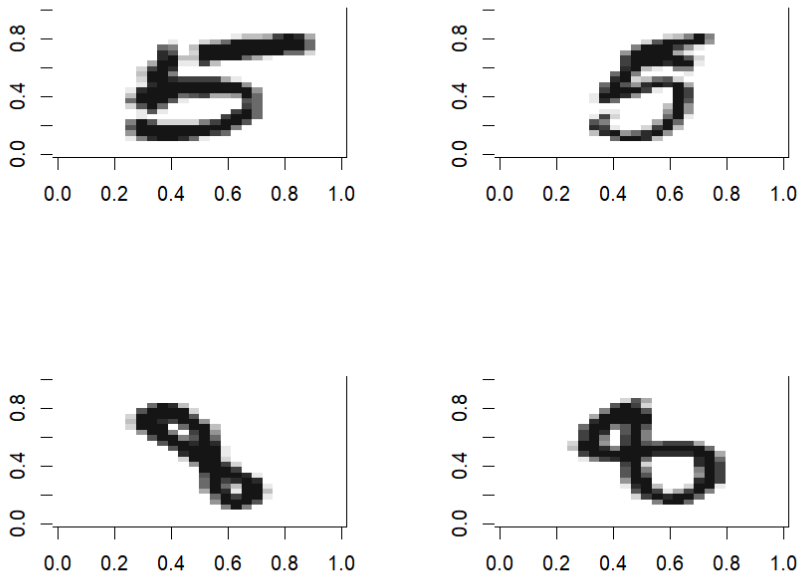
2 lentelė. Testavimo aibes klasifikavimo tikslumo matai

```
[1] "Matai atskiroms klasems"
  precision recall        f1
5 0.9609756  0.985 0.9728395
8 0.9846154  0.960 0.9721519
[1] "Matai apskaiciuoti testavimo aibei"
  macroPrecision macroRecall   macroF1
1      0.9727955      0.9725 0.9724957
```

Čia ir toliau klasifikavimo matricos išdėstymas yra toks (einant skaitymo būdu per lentelę) : TP, FP, FN, TN. Matome, kad penketus modeliui sekėsi geriau atskirti nei aštuonetus (*recall* didesnis). Kita vertus, taip yra ir todėl, kad modelis turėjo polinkį rinktis penketus vietoje aštuonetų, dėl to penketų daugiau, tačiau *precision* yra mažesnis.



2 pav. AVK be t-SNE klasifikacijų taškinė diagrama



3 pav. Blogai suklasifikuoti skaitmenys SVM metodu

Iš taškinės diagramos galime matyti, kad modelis vietomis „pagavo“ aštuonetus (mėlyni taškai), kurie t-SNE metodo buvo susimaišyti tarp penketų. Kita vertus, yra daug oranžinių taškų, kurie vietoje aštuoneto buvo priskirti penketams. T-SNE metodas irgi nesugebėjo jų atskirti. Tai

gali indikuoti, kad sumažinus dimensiją iki 2 modelis nebūtinai bus geresnis. Iš trečio paveikslėlio matome, kad visi skaitmenys turi išsiskirinačių bruožų – aštuonetai yra pakreipti, o penketai – nesujungti arba su užrašymo klaida.

AVK su t-SNE

Iš praeitų tyrimų žinome, kad geriausias MNIST duomenų rinkinio dimensijos mažinimo algoritmas yra t-SNE. Tad mes sumažinome duomenų dimensija iki *dim* = 2 ir pakartojome praeitus žingsnius.

Optimalių parametrų radimas

Panaudoje tuos pačius branduolius ir parametrų kombinacijas gavome tokius rezultatus (lentelės iš 17 - 18 puslapių):

- Linear
  - **Tikslumas = 0.95**
  - **F-1 = 0.949**
- Polynomial
  - $laipsnis = 1$
  - $\gamma = 0.001$
  - $koeficientas = -0.5$
  - **Tikslumas = 0.95**
  - **F-1 = 0.949**
- Radial Basis
  - $\gamma = 0.1$
  - **Tiksluams = 0.9650**
  - **F-1 = 0.9649**
- Sigmoid
  - $\gamma = 0.001$
  - $koeficientas = -0.5$
  - **Tikslumas = 0.95**
  - **F1 = 0.949**

Matome kad geriausius rezultatus gauname su „*Radial Basis*“ branduoliu. Jo tikslumas yra 0.9650 ir F-1 = 0.9649.

Klasifikatoriaus modeliavimas

Sumodeliuojame atraminių vektoriu Radial Basis klasifikatorių ir gauname tokius rezultatus ant testavimo aibės.

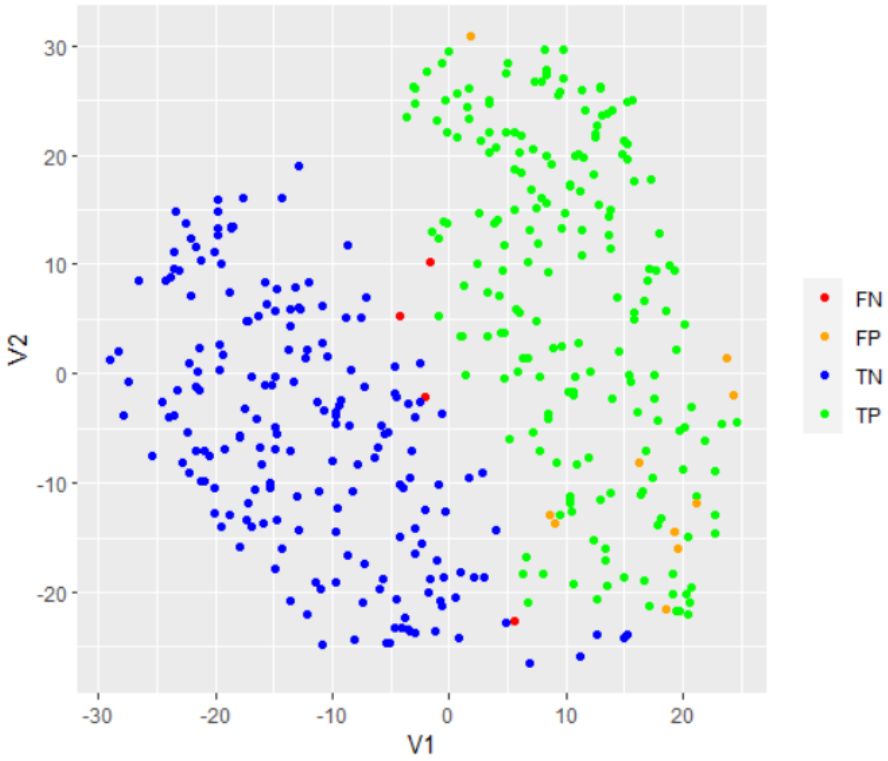
3 lentelė. Sumaišymo matrica

	5	8
5	196	10
8	4	190

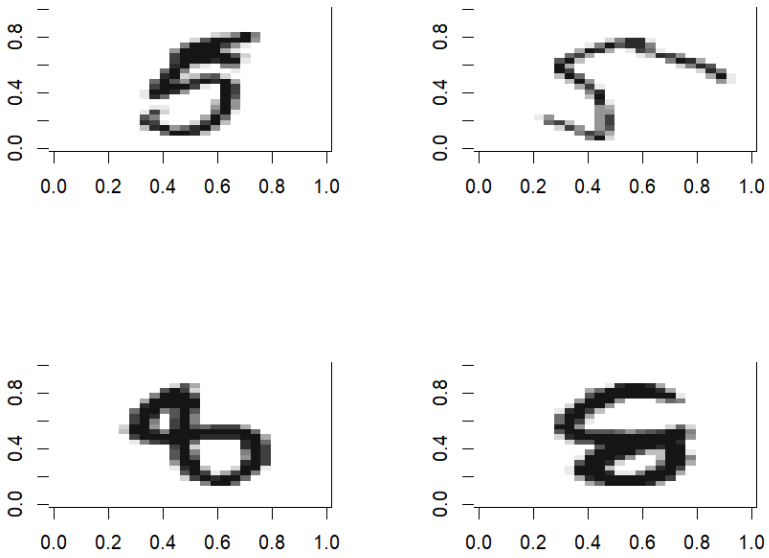
4 lentelė. Testavimo aibes klasifikavimo tikslumo matai

```
.] "Matai atskiroms klasems"
precision recall        f1
0.9514563   0.98 0.9655172
0.9793814   0.95 0.9644670
.] "Matai apskaiciuoti testavimo aibei"
macroPrecision macroRecall   macroF1
     0.9654189       0.965 0.9649921
```

Kaip ir buvo nuspėta anksčiau, t-SNE duomenų dimensijos sumažinimas šį kartą nedavė didesnio tikslumo. Tačiau taip gali būti ir todėl, kad AVK veikia geriau, kai yra daug kovariančių. Modelis turėjo tą pačią tendenciją skirti rinktis daugiau penketų nei aštuonetų, kas indikuoja didesnį 5 recall, bet mažesnį precision.



pav. 4 AVK su t-SNE klasifikacijų taškinė diagrama.



5 pav. Blogai suklasifikuoti skaitmenys SVM t-SNE metodu

Kaip matome, šį kartą klasifikatorius nebeatrinko aštuonetų iš taip „giliai“ kaip ankstesnis modelis. Visi keturi neteisingai priskirti aštuonetai (FN - raudoni) yra mėlyno klasterio pakraštyje prie žaliojo, kas indikuoja, kad šie taškai buvo sunkiai atskiriami t-SNE. Iš 5 pav. matome, kad kai kurios klaidos sutampa su SVM (originalių duomenų), yra ir naujų, išskirtinių skaitmenų užrašymų.

Atsitiktiniai miškai be t-SNE

Atsitiktinio miško klasifikatorius (angl. *Random Forest classifier*) yra prižiūrimas mokymosi algoritmas, naudojamas klasifikavimui ir regresijai. Atsitiktinio miško algoritmas pagal duomenų pavyzdžius sukuria sprendimų medžius, tada iš kiekvieno jų gauna prognozę ir galiausiai išrenka geriausią sprendimą. Kuo didesnis medžių skaičius miške, tuo aukštesni tikslumo rezultatai [3], [4].

Optimalių parametrų radimas

Šitam algoritmui bandėme keisti tik medžių skaičių (*ntree*). Gavome, kad geriausius rezultatus gauname kai *ntree* = 500. Tikslumas yra lygus 0.9675, o F-1 lygus 0.9674 (15 lentelė).

Klasifikatoriaus modeliavimas ir rezultatai

Sumodeliuojame atsitiktinio miško klasifikatorių su 500 medžių ir gauname tokius rezultatus naudojant testavimo aibę.

5 lentelė. Sumaišymo matrica

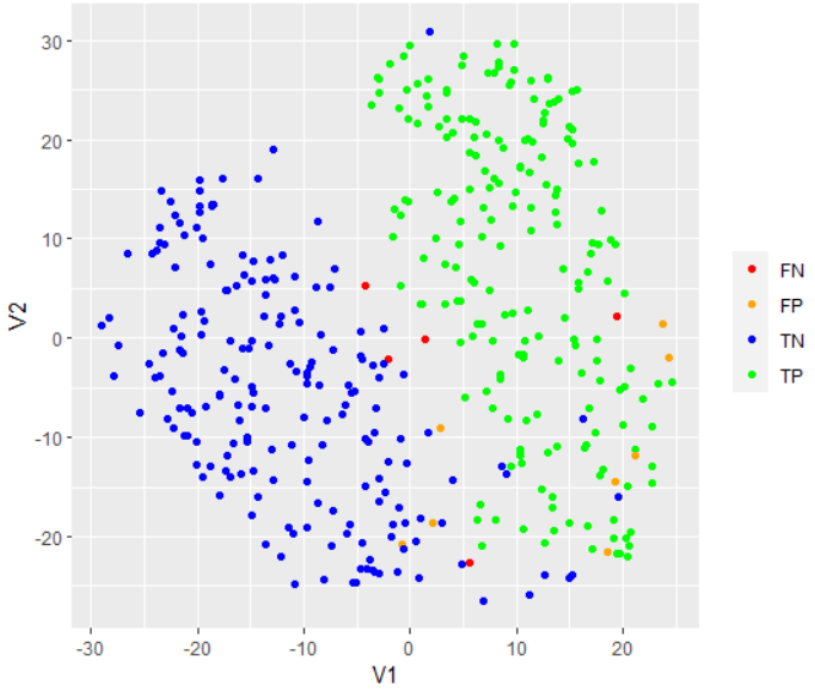
	5	8
5	195	8
8	5	192

6 lentelė. Testavimo aibės klasifikavimo tikslumo matai

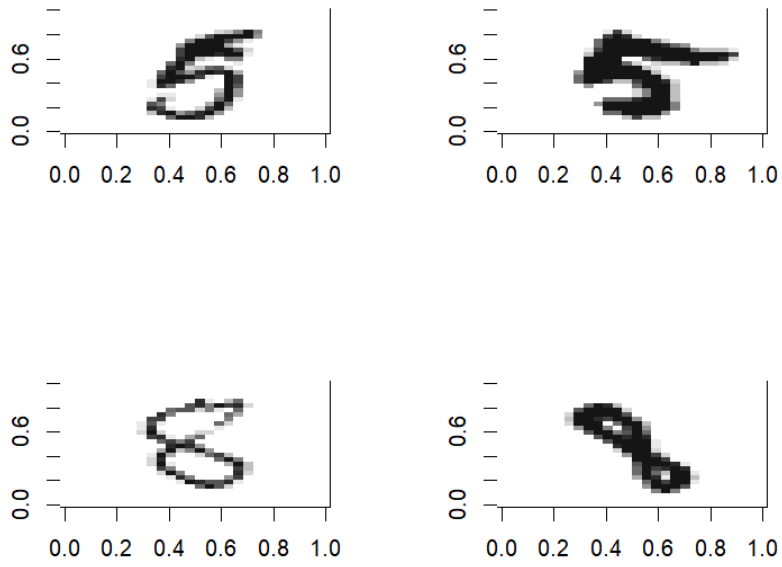
```
] "Matai atskiroms klasems"
precision recall        f1
0.9605911  0.975 0.9677419
0.9746193  0.960 0.9672544
] "Matai apskaiciuoti testavimo aibei"
macroPrecision macroRecall   macroF1
     0.9676052      0.9675 0.9674982
```

Kaip ir AVK, RF buvo linkęs rinktis penketus. Šio klasifikavimo metodo tikslumas buvo mažesnis nei AVK, tačiau taip gali būti dėl atsitiktinumo, ypač kai skirtumai tokie minimalūs.

Iš žemiau esančio 6 pav. matome, kad dalis FP taškų yra susimaišę tarp penketų, ko ir galime tikėtis. Tačiau įdomu tai, kad trys FP oranžiniai taškai yra ir aštuonetų aibėje, t.y. t-SNE pajuto, kad jie yra labiau aštuonetai nei penketai, o RF modelis – ne. Galima tikėtis, kad dimensijos mažinimas šį kartą padės pagerinti klasifikavimo rezultatus.



6 pav. RF be t-SNE klasifikacijų taškinė diagrama.



7 pav. Blogai suklasifikuoti skaitmenys RF metodu

Matome, kad ta pati klaida kartojasi jau trečią kartą. Taip pat atsirado naujas ryškus penketas bei siaurias aštuonetas.

Atsitiktiniai miškai su t-SNE

Optimalių parametrų radimas

Sumažinus dimensiją gavome, kad tikslumo matai visiškai nepriklauso nuo parametro *ntree*, tai mes vėl naudosime *ntree* = 500 (16 lentelė).

Klasifikatoriaus modeliavimas

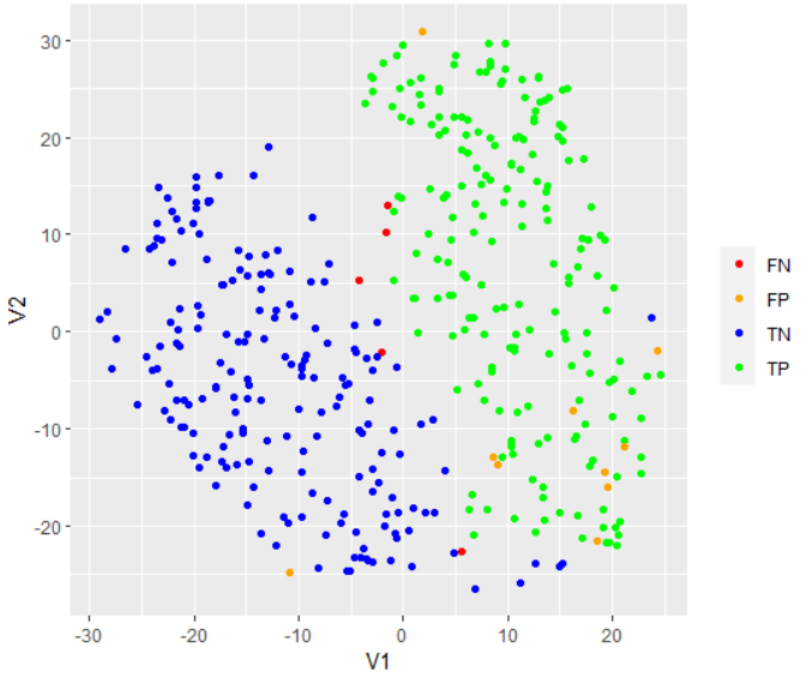
7 lentelė. Sumaišymo matrica

	5	8
5	195	10
8	5	190

Nors buvo tikėtasi geresnių rezultatų, taip vis dėlto neįvyko. Žinoma, skirtumai tarp visų tikslumų yra minimalūs ir gali priklausyti nuo labai nedidelių faktorių ir „parankesnės“ tastavimo aibės. Dėl to verta atlikti cross – validation patikrą taip užtikrinant, ar šis tikslumas yra atsitiktinumas ar ne.

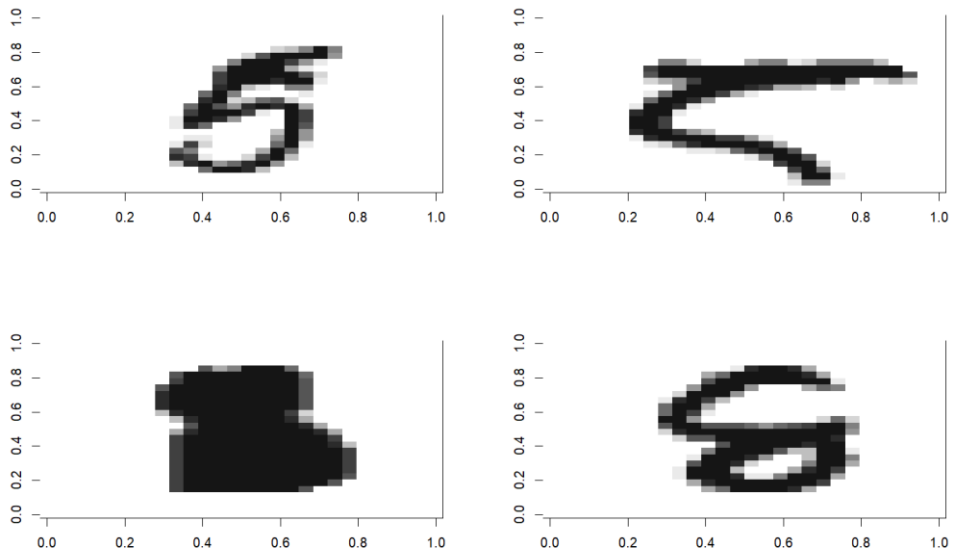
8 lentelė. Testavimo aibės klasifikavimo matai

```
] "Matai atskiroms klasems"
precision recall        f1
0.9512195  0.975 0.9629630
0.9743590  0.950 0.9620253
] "Matai apskaiciuoti testavimo aibei'
macroPrecision macroRecall   macroF1
     0.9627892      0.9625 0.9624941
```



8 pav. RF su t-SNE klasifikacijų taškinė diagrama

Anksčiau minėti taškai šį kartą buvo suklasifikuoti teisingai, tačiau potencialiai dėl pačio t-SNE netikslumo mūsų klasifikavimo tikslumas sumažėjo.



9 pav. Blogai suklasifikuoti skaitmenys RF t-SNE metodu

Iš 9 pav. matome, kad vienas iš penketų kartojasi per visus klasifikatorius. Naujas penketas yra neperskaitomas, taip pat atsiradę aštuonetai irgi nėra patys aiškiausi.

Klasifikavimo kokybės vertinimas

Kryžminio validavimo strategija

Kadangi jau įvertinome klasifikatoriaus rezultatus ankstesniuose skyriuose naudodami išlaikymo (angl. *holdout*) metodą, belieka įvertinti naudojant kryžminį validavimą.

Kryžminio validavimo procesų duomenų rinkinys yra dalinamas į *k* grupių, kur kiekvienoje grupėje mokymuisi skiriama k-1 duomenų dalis, o testavimui – likusi viena. Taip iteruojama, kol testine aibe „pabuvo“ visos k duomenų dalys.

Pritaikę mūsų sumažintų dimensijų AVK ir Atsitiktinio miško modeliams kryžminio validavimo strategiją su parametru k = 10 gauname tokius rezultatus:

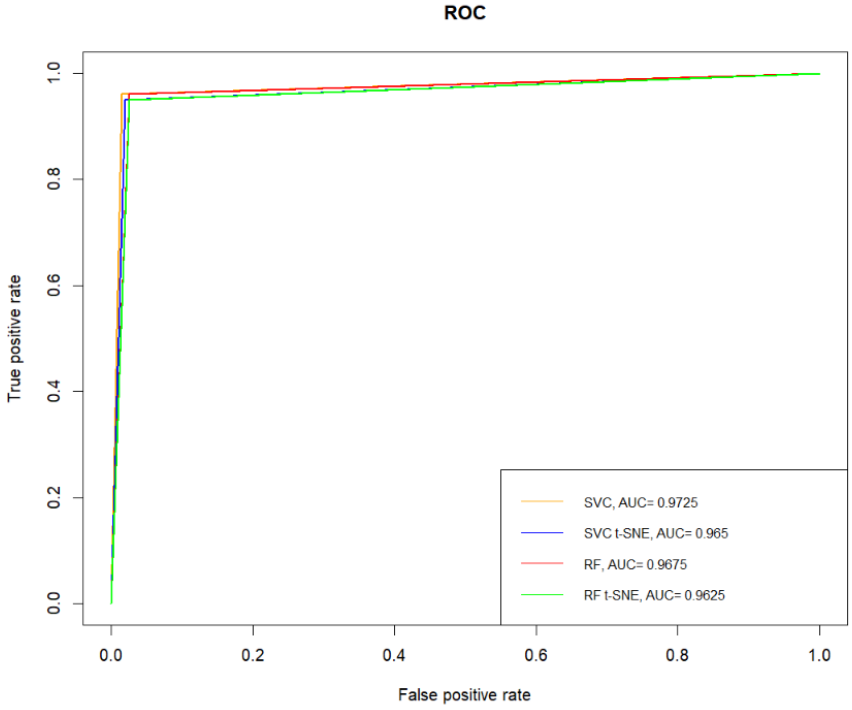
- AVK su t-SNE
  - Klasifikavimo tikslumas = 0.981
  - Klasifikavimo tikslumo standartinis nuokrypis = 0.013
- Atsitiktinis miškas su t-SNE
  - Klasifikavimo tikslumas = 0.981
  - Klasifikavimo tikslumo standartinis nuokrypis = 0.011

Matome, kad klasifikavimo tikslumai visiškai nesiskiria, tačiau standartinis nuokrypis atsitiktinio miško modelyje yra mažesnis.

ROC kreivės ir AUC matas

ROC (angl. Receiver Operating Curve) kreivė rodo kompromisą tarp jautrumo (arba TPR) ir specifiškumo (1 – FPR). Klasifikatoriai, pateikiantys kreives arčiau viršutinio kairiojo kampo, rodo geresnį tikslumą. Kuo kreivė priartėja prie ROC erdvės 45 laipsnių įstrižainės, tuo tikslumas bus mažesnis.

Norint palyginti skirtingus klasifikatorius, gali būti naudinga apibendrinti kiekvieno klasifikatoriaus kokybę į vieną matą. Vienas įprastas būdas yra apskaičiuoti plotą po ROC kreive, kuris sutrumpintai reiškia AUC (angl. Area Under Curve).



10 pav. Visų klasifikatorių ROC kreivės ir AUC matai

Iš 10 pav. matome, kad visos kreivės yra labai panašios ir yra viršutiniame kairiajame kampe, kas yra geras rezultatas, nes tai reiškia, kad visi klasifikatoriai sugebėjo atskirti penketus ir aštuonetus labai tiksliai. Aišku, kad AUC matai irgi yra labai panašūs, bet polinominis AVK be t-SNE dimensijos mažinimo turi geriausią AUC matą. To buvo galima tikėtis, nes AVK geriausiai veikia kai yra didesnis dimensijų skaičius.

Išvados

Visi parinkti klasifikavimo algoritmai klasifikavo labai geru tikslumu (atitinkamas AUC pagal tyrimo eigą: SVM = 0,9725, SVM t-SNE = 0,965, RF = 0,9675, RF t-SNE = 0,9625), o skirtumai tarp jų buvo minimalūs. Tiksliausias buvo SVM (be t-SNE) klasifikatorius su trečio laipsnio polinominiu branduoliu. Šį kartą t-SNE dimensijos mažinimas nepadėjo (klasifikavimo tikslumas, AUC nepadidėja), tačiau taip gali būti ir dėl atrinktos imties ir kitokių atsitiktinumų. AVK buvo parinktas dėl labai didelio kovariančių skaičiaus, kaip matėme šis sprendimas pasiteisino. Kita vertus, išrinkti optimalius parametrus šiam metodui yra žymiai sunkiau, nes galimų kombinacijų yra labai daug. RF metodas buvo žymiai paprastesnis, visada klasifikavo tiksliai. Nors naudojant *holdout* metodą jis nusileido AVK, cv patikros būdas parodė, kad tai yra tik atsitiktinumas ir abu klasifikatoriai yra labai panašūs (SVM = 0,981, RF = 0,981).

Šaltiniai

SVM

[1] SVM mokslinis – Shmilovici, A. (2009). Support vector machines. In *Data mining and knowledge discovery handbook* (pp. 231-247). Springer, Boston, MA. Internetinė nuoroda :https://link.springer.com/chapter/10.1007/978-0-387-09823-4_12

[2] SVM - Gandhi, R. (2018). Support Vector Machine - Introduction to Machine Learning Algorithms. *Medium*, Towards Data Science. Internetinė nuoroda: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

RF

[3] RF mokslinis – Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32. Internetinė nuoroda: https://link.springer.com/article/10.1023/a:1010933404324

[4] RF - Yiu, T. (2021). Understanding Random Forest. *Medium*, Towards Data Science. Internetinė nuoroda: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Priedai

Lentelės

9 lentelė. Polinominio AVK modelio parametrai ir jų tikslumas

degree	gamma	coef0	Accuracy	F1
3	0,001	-0,5	0,9725	0,972495702
3	0,001	0	0,9725	0,972495702
3	0,001	0,1	0,9725	0,972495702
3	0,00127551	-0,5	0,9725	0,972495702
3	0,00127551	0	0,9725	0,972495702
3	0,00127551	0,1	0,9725	0,972495702
3	0,1	-0,5	0,9725	0,972495702
3	0,1	0	0,9725	0,972495702
3	0,1	0,1	0,9725	0,972495702
5	0,001	-0,5	0,96	0,959996
5	0,001	0	0,96	0,959996
5	0,001	0,1	0,96	0,959996

5	0,00127551	-0,5	0,96	0,959996
5	0,00127551	0	0,96	0,959996
5	0,00127551	0,1	0,96	0,959996
5	0,1	-0,5	0,96	0,959996
5	0,1	0	0,96	0,959996
5	0,1	0,1	0,96	0,959996
1	0,001	-0,5	0,9225	0,922499516
1	0,001	0	0,9225	0,922499516
1	0,001	0,1	0,9225	0,922499516
1	0,00127551	-0,5	0,9225	0,922499516
1	0,00127551	0	0,9225	0,922499516
1	0,00127551	0,1	0,9225	0,922499516
1	0,1	-0,5	0,9225	0,922499516
1	0,1	0	0,9225	0,922499516
1	0,1	0,1	0,9225	0,922499516

10 lentelė. Radail basis AVK modelio parametrai ir jų tikslumas

gamma	Accuracy	F1
0,001	0,5	NA
0,00127551	0,5	NA
0,1	0,5	NA

11 lentelė. Sigmoidinio AVK modelio parametrai ir jų tikslumas

gamma	coef0	Accuracy	F1
0,001	-0,5	0,5	NA
0,001	0	0,5	NA
0,001	0,1	0,5	NA
0,00127551	-0,5	0,5	NA
0,00127551	0	0,5	NA
0,00127551	0,1	0,5	NA
0,1	-0,5	0,5	NA
0,1	0	0,5	NA
0,1	0,1	0,5	NA

12 lentelė. Polinominio AVK modelio parametrai ir jų tikslumas (po t-SNE transformacijos)

degree	gamma	coef0	Accuracy	F1
1	0,001	-0,5	0,95	0,949979992
1	0,001	0	0,95	0,949979992
1	0,001	0,1	0,95	0,949979992
1	0,00127551	-0,5	0,95	0,949979992

1	0,00127551	0	0,95	0,949979992
1	0,00127551	0,1	0,95	0,949979992
1	0,1	-0,5	0,9475	0,947483917
1	0,1	0	0,9475	0,947483917
1	0,1	0,1	0,9475	0,947483917
3	0,001	-0,5	0,7825	0,782487765
3	0,001	0	0,925	0,924992499
3	0,001	0,1	0,95	0,949988747
3	0,00127551	-0,5	0,78	0,78
3	0,00127551	0	0,925	0,924992499
3	0,00127551	0,1	0,95	0,949988747
3	0,1	-0,5	0,8625	0,862492265
3	0,1	0	0,9175	0,917487107
3	0,1	0,1	0,95	0,949979992
5	0,001	-0,5	0,74	0,7399935
5	0,001	0	0,8225	0,819795811
5	0,001	0,1	0,88	0,879566439
5	0,00127551	-0,5	0,725	0,724993125
5	0,00127551	0	0,845	0,843272075
5	0,00127551	0,1	0,915	0,914946842
5	0,1	-0,5	0,83	0,82999575
5	0,1	0	0,655	0,634833691
5	0,1	0,1	0,675	0,67064427

13 lentelė. Radial basis AVK modelio parametrai ir jų tikslumas (po t-SNE transformacijos)

gamma	Accuracy	F1
0,001	0,9525	0,952475941
0,00127551	0,9525	0,952475941
0,1	0,965	0,964992123

14 lentelė. Sigmoidinio AVK modelio parametrai ir jų tikslumas (po t-SNE transformacijos)

gamma	coef0	Accuracy	F1
0,001	-0,5	0,95	0,949979992
0,001	0	0,95	0,949979992
0,00127551	-0,5	0,95	0,949979992
0,00127551	0	0,9475	0,947473408
0,001	0,1	0,945	0,944987622
0,00127551	0,1	0,94	0,9399985
0,1	-0,5	0,6475	0,646117647
0,1	0	0,635	0,634085213
0,1	0,1	0,635	0,634085213

15 lentelė. RF modelio parametrai ir jų tikslumas

ntree	mtree	Accuracy	F1
500	20	0,9675	0,967498172
500	40	0,9675	0,967498172
500	60	0,9675	0,967498172
100	20	0,965	0,964999125
100	40	0,965	0,964999125
100	60	0,965	0,964999125
300	20	0,965	0,964999125
300	40	0,965	0,964999125
300	60	0,965	0,964999125

16 lentelė. RF modelio parametrai ir jų tikslumas po (t-SNE transformacijos)

ntree	mtree	Accuracy	F1
100	20	0,9625	0,96249414
100	40	0,9625	0,96249414
100	60	0,9625	0,96249414
300	20	0,9625	0,96249414
300	40	0,9625	0,96249414
300	60	0,9625	0,96249414
500	20	0,9625	0,96249414
500	40	0,9625	0,96249414
500	60	0,9625	0,96249414

Kodas

```
# 4 uzduotis
# Roland G. ir Matas A.

library(mnist)
library(dplyr)
library(ggplot2)
library(caTools)# test-train split
library(caret)# CV
library(Rtsne)
library(e1071)# SVM
library(randomForest)# RF
library(ROCR)
library(pROC)


###################################
# Duomenu tvarkymas
###################################
```

```r
mnist <- download_mnist()

# Atrenkame tik 5 ir 8 skaitmenis
set.seed(67)
data <- mnist %>% filter(Label == '5' | Label == '8') %>%
  group_by(Label) %>% sample_n(size = 1000)
data$Label <- as.character(data$Label)

# Duomenu normuoti nereikia - jie visi toje pacioje skaleje


# Duomenu test train indeksu sudarymas
set.seed(67)
split = sample.split(data$Label, SplitRatio = 0.8)
sum(split) # 1600 mokymo
sum(!split) # 400 testavimo


# Paprastu duomenu mokymo testavimo aibiu sudarymas
train = subset(data, split == TRUE)
test = subset(data, split == FALSE)


# t-SNE duomenu sudarymas
set.seed(67)
tsne <- Rtsne(data, dims = 2, perplexity=30, verbose=TRUE, max_iter = 1000)
data_tsne <- as.data.frame(tsne$Y)
data_tsne$Label <- data$Label

# t-SNE duomenu dalinimas
train_tsne = subset(data_tsne, split == TRUE)
test_tsne = subset(data_tsne, split == FALSE)


###################################
# SVM
###################################

# Klasifikavimo tikslumo matai (funkcija)
tikslumo_matai <- function(cm, n){
  diag = diag(cm) # number of correctly classified instances per class
  rowsums = apply(cm, 1, sum) # number of instances per class
  colsums = apply(cm, 2, sum) # number of predictions per class

  accuracy = sum(diag) / n # general accuracy of classification
  # matai skaiciuojami atskiroms klasems
  precision = diag / colsums
  recall = diag / rowsums
  f1 = 2 * precision * recall / (precision + recall)
  print("Matai atskiroms klasems")
  print(data.frame(precision, recall, f1))

  # matai apskaiciuoti testavimo aibei
  macroPrecision = mean(precision)
  macroRecall = mean(recall)
  macroF1 = mean(f1)
  print("Matai apskaiciuoti testavimo aibei")
  print(data.frame(macroPrecision, macroRecall, macroF1))
}

# Klasifikavimo tikslumo matai (funkcija) (parametru parinkimui)
tikslumo_matai_param <- function(cm, n){
  diag = diag(cm) # number of correctly classified instances per class
```

```
  rowsums = apply(cm, 1, sum) # number of instances per class
  colsums = apply(cm, 2, sum) # number of predictions per class

  accuracy = sum(diag) / n # general accuracy of classification
  # matai skaiciuojami atskiroms klasems
  precision = diag / colsums
  recall = diag / rowsums
  f1 = 2 * precision * recall / (precision + recall)

  # matai apskaiciuoti testavimo aibei
  macroF1 = mean(f1)

  data.frame(accuracy, macroF1)
}

cost <- 1
epsilon <- 0.1
degree <- c(1,3,5) # default = 3
gamma <- c(1/784,0.001,0.1) # default = 1/dim(duomenys)<=>1/784
coef0 <- c(-0.5, 0, 0.1) # default = 0

# LINEAR
temp <- tidyr::crossing(cost, epsilon)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train,
                        type = 'C-classification',
                        kernel = 'linear',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2])
  y_pred_temp <- predict(classifier_temp, newdata = test[,-785])
  cm_temp <- table(y_pred_temp,t(test[, 785]))
  temp_iv <- tikslumo_matai_param(cm_temp, nrow(test[,785]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
param_svm_linear <- temp

# POLYNOMIAL
temp <- tidyr::crossing(cost, epsilon, degree, gamma, coef0)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train,
                        type = 'C-classification',
                        kernel = 'polynomial',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2],
                        degree = temp[index,3],
                        gamma = temp[index,4],
                        coef0 = temp[index,5])
  y_pred_temp <- predict(classifier_temp, newdata = test[,-785])
  cm_temp <- table(y_pred_temp,t(test[, 785]))
  temp_iv <- tikslumo_matai_param(cm_temp, nrow(test[,785]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
param_svm_poly <- temp[,c(3:7)]

# RADIAL BASIS
```

```
temp <- tidyr::crossing(cost, epsilon, gamma)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train,
                        type = 'C-classification',
                        kernel = 'radial',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2],
                        gamma = temp[index,3])
  y_pred_temp <- predict(classifier_temp, newdata = test[,-785])
  cm_temp <- table(y_pred_temp,t(test[, 785]))
  temp_iv <- tikslumo_matai_param(cm_temp, nrow(test[,785]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
param_svm_rb <- temp[,c(3:5)]

# SIGMOID
temp <- tidyr::crossing(cost, epsilon, gamma, coef0)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train,
                        type = 'C-classification',
                        kernel = 'sigmoid',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2],
                        gamma = temp[index,3],
                        coef0 = temp[index,4])
  y_pred_temp <- predict(classifier_temp, newdata = test[,-785])
  cm_temp <- table(y_pred_temp,t(test[, 785]))
  temp_iv <- tikslumo_matai_param(cm_temp, nrow(test[,785]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
param_svm_sigmoid <- temp[,c(3:6)]

# Geriausia su kernel polynomial su hyperparametais degree=3 (default),
# kiti parametrai itakos neturejo.

# Geriausio varianto modelis
classifier = svm(formula = Label ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = 'polynomial',
                 scale = FALSE)
  #kadangi kitos reiksmes nesvarbios, paliksim default


y_pred1 = predict(classifier, newdata = test[,-785])
# Klasifikavimo matrica
a1<- as.data.frame.matrix((cm1 = table(y_pred1,t(test[, 785]))))
tikslumo_matai(cm1, nrow(test[,785]))

###################################
# SVM t-SNE
###################################

# LINEAR
temp <- tidyr::crossing(cost, epsilon)
```

```
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train_tsne,
                        type = 'C-classification',
                        kernel = 'linear',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2])
  y_pred_temp <- predict(classifier_temp, newdata = test_tsne[,-3])
  cm_temp <- table(y_pred_temp,t(test_tsne[,3]))
  temp_iv <- tikslumo_matai_param(cm_temp, length(test_tsne[,3]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
par_tsne_svm_linear <- temp

# POLYNOMIAL
temp <- tidyr::crossing(cost, epsilon, degree, gamma, coef0)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train_tsne,
                        type = 'C-classification',
                        kernel = 'polynomial',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2],
                        degree = temp[index,3],
                        gamma = temp[index,4],
                        coef0 = temp[index,5])
  y_pred_temp <- predict(classifier_temp, newdata = test_tsne[,-3])
  cm_temp <- table(y_pred_temp,t(test_tsne[, 3]))
  temp_iv <- tikslumo_matai_param(cm_temp, length(test_tsne[,3]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
par_tsne_svm_poly <- temp[,c(3:7)]

# RADIAL BASIS
temp <- tidyr::crossing(cost, epsilon, gamma)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
                        data = train_tsne,
                        type = 'C-classification',
                        kernel = 'radial',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2],
                        gamma = temp[index,3])
  y_pred_temp <- predict(classifier_temp, newdata = test_tsne[,-3])
  cm_temp <- table(y_pred_temp,t(test_tsne[, 3]))
  temp_iv <- tikslumo_matai_param(cm_temp, length(test_tsne[,3]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
par_tsne_svm_rb <- temp[,c(3:5)]

# SIGMOID
temp <- tidyr::crossing(cost, epsilon, gamma, coef0)
for(index in 1:nrow(temp)){
  classifier_temp = svm(formula = Label ~ .,
```

```
                        data = train_tsne,
                        type = 'C-classification',
                        kernel = 'sigmoid',
                        scale = FALSE,
                        cost = temp[index,1],
                        epsilon = temp[index,2],
                        gamma = temp[index,3],
                        coef0 = temp[index,4])
  y_pred_temp <- predict(classifier_temp, newdata = test_tsne[,-3])
  cm_temp <- table(y_pred_temp,t(test_tsne[, 3]))
  temp_iv <- tikslumo_matai_param(cm_temp, length(test_tsne[,3]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}
par_tsne_svm_sigmoid <- temp[,c(3:6)]

# Geriausias tikslumas buvo su branduoliu radial ir gamma = 0.1

classifier = svm(formula = Label ~ .,
                 data = train_tsne,
                 type = 'C-classification',
                 kernel = 'radial',
                 scale = FALSE,
                 gamma = 0.1)

plot(classifier, test_tsne)

y_pred2 = predict(classifier, newdata = test_tsne[,-3])

# Klasifikavimo matrica
a2<- as.data.frame.matrix((cm2 = table(y_pred2, t(test_tsne[, 3]))))
tikslumo_matai(cm2, length(test_tsne[, 3]))

classifier

###################################
# RF
###################################

train$Label <- as.factor(train$Label)
test$Label <- as.factor(test$Label)
ntree <- c(100,300,500)
mtree <- c(20,sqrt(nrow(train)),60) #sqrt(nrow(train)) - default <=> 40

temp <- tidyr::crossing(ntree, mtree)
for(index in 1:nrow(temp)){
  set.seed(67)
  classifier_temp = randomForest(
                            x = train[,-785],
                            y = train$Label,
                            ntree = as.numeric(temp[index, 1]),
                            mtree = as.numeric(temp[index, 2]))
  y_pred_temp <- predict(classifier_temp, newdata = test[,-785])
  cm_temp <- table(y_pred_temp,t(test[, 785]))
  temp_iv <- tikslumo_matai_param(cm_temp, nrow(test[,785]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}

par_RF <- temp
?randomForest
```

```r
# Geriusias buvo RF su ntree = 500, mtree nesvarbu. Jo modelis:
set.seed(67)
classifier = randomForest(x = train[,-785],
                          y = train$Label,
                          ntree = 500)

# testavimo aibes klasifikavimas
y_pred3 = predict(classifier, newdata = test[,-785])

# Sumaisymo matrica
a3<- as.data.frame.matrix((cm3 = table(as.character(y_pred3) ,test$Label)))

tikslumo_matai(cm3, nrow(test[, 785]))

#show_digit(test, 306)

#################################
# RF t-SNE
#################################
train_tsne$Label <- as.factor(train_tsne$Label)
test_tsne$Label <- as.factor(test_tsne$Label)

temp <- tidyr::crossing(ntree, mtree)
for(index in 1:nrow(temp)){
  set.seed(67)
  classifier_temp = randomForest(
    Label ~ .,
    data = train_tsne,
    ntree = as.numeric(temp[index, 1]),
    mtree = as.numeric(temp[index, 2]))
  y_pred_temp <- predict(classifier_temp, newdata = test_tsne[,-3])
  cm_temp <- table(y_pred_temp,t(test_tsne[,3]))
  temp_iv <- tikslumo_matai_param(cm_temp, length(test_tsne[,3]))
  temp$Accuracy[index] <- temp_iv$accuracy
  temp$F1[index] <- temp_iv$macroF1
  temp
}

par_RF_tsne <- temp

# Visi klasifikavo vienodai
set.seed(67)
classifier = randomForest(Label ~ ., data = train_tsne, ntree = 500)

# testavimo aibes klasifikavimas
y_pred4 = predict(classifier, newdata = test_tsne[,-3])

# Sumai?ymo matrica
a4<-as.data.frame.matrix((cm4 = table(as.character(y_pred4), test_tsne$Label)))

tikslumo_matai(cm4, nrow(test_tsne[, 3]))

#show_digit(data, 310)

#################################
# VIZUALIZAVIMAS CM
#################################

plot(classifier, test_tsne)

viz_scatter <- function(y_pred) {
  data_viz <- test_tsne
```

```
  data_viz$LabelPred <- y_pred

  data_viz$color <- ifelse(data_viz$Label == 5 & data_viz$LabelPred == 5, "TP",
                           ifelse(data_viz$Label == 5 & data_viz$LabelPred == 8, "FN",
                           ifelse(data_viz$Label == 8 & data_viz$LabelPred == 8, "TN",
                                  "FP")))


  ggplot(data_viz, aes(x = V1, y = V2, col = color)) + geom_point() +
    scale_color_manual(values = c("Red", "orange", "Blue", "Green")) +
    theme(legend.title=element_blank())

}

viz_scatter(y_pred1)
viz_scatter(y_pred2)
viz_scatter(y_pred3)
viz_scatter(y_pred4)


# 4.2: Holdout jau padarytas? nes tipo holdout tai ir yra tiesiog train test split
# apmokyti duomenys su train ir ivertinti su test?

###################################
# CV IR ROC
###################################
set.seed(67)
folds = createFolds(train_tsne$Label, k = 10)


# SVM be t-SNE
cv = lapply(folds, function(x) {
  training_fold = train_tsne[-x, ]
  test_fold = train_tsne[x, ]

  classifier = svm(formula = Label ~ .,
                   data = train_tsne,
                   type = 'C-classification',
                   kernel = 'polynomial',
                   gamma = 0.001,
                   coef0 = -0.5,
                   scale = FALSE,
                   degree = 3)

  y_pred = predict(classifier, newdata = test_fold[-3])
  cm = table(test_fold[, 3], y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  #cia analogiskai skaiciuokite taip pat ir precision, recall ir F1
  return(accuracy)
})
cv
(accuracy = mean(as.numeric(cv)))
(accuracy_sd =sd(as.numeric(cv)))



# SVM t-SNE
cv = lapply(folds, function(x) {
  training_fold = train_tsne[-x, ]
  test_fold = train_tsne[x, ]

  classifier = svm(formula = Label ~ .,
                   data = train_tsne,
```

```
                    type = 'C-classification',
                    kernel = 'radial',
                    scale = FALSE,
                    gamma = 0.1)

  y_pred = predict(classifier, newdata = test_fold[-3])
  cm = table(test_fold[, 3], y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  #cia analogiskai skaiciuokite taip pat ir precision, recall ir F1
  return(accuracy)
})
cv
(accuracy = mean(as.numeric(cv)))
(accuracy_sd =sd(as.numeric(cv)))


# RF t-SNE
cv = lapply(folds, function(x) {
  training_fold = train_tsne[-x, ]
  test_fold = train_tsne[x, ]

  set.seed(67)
  classifier = randomForest(x = training_fold[-3],
                            y = training_fold$Label,
                            ntree = 500)

  y_pred = predict(classifier, newdata = test_fold[-3])
  cm = table(test_fold[, 3], y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  #cia analogiskai skaiciuokite taip pat ir precision, recall ir F1
  return(accuracy)
})
cv
(accuracy = mean(as.numeric(cv)))
(accuracy_sd =sd(as.numeric(cv)))

# Matome, kad su cross-validation rezultatai nesiskiria

#### ROC

roc_curve <- function(predictions, df) {
  pred <- prediction(as.numeric(predictions), as.numeric(df$Label))
  perf <- performance(pred,"tpr","fpr")
  plot(perf,colorize=TRUE, main = "ROC")
}

# ROC kazkaip galima su CV daryti bet nezinau kaip
roc_curve(y_pred1, test)
roc_curve(y_pred2, test_tsne)
roc_curve(y_pred3, test)
roc_curve(y_pred4, test_tsne)

# kodas suskaiciuoti AUC
auc(as.numeric(test_tsne$Label), as.numeric(y_pred3))

# bendras grafikas su visaiss ROC ir AUC
pred1 <- prediction(as.numeric(y_pred1), as.numeric(test$Label))
pred2 <- prediction(as.numeric(y_pred2), as.numeric(test_tsne$Label))
pred3 <- prediction(as.numeric(y_pred3), as.numeric(test$Label))
pred4 <- prediction(as.numeric(y_pred4), as.numeric(test_tsne$Label))
perf1 <- performance(pred1,"tpr","fpr")
perf2 <- performance(pred2,"tpr","fpr")
perf3 <- performance(pred3,"tpr","fpr")
```

```
perf4 <- performance(pred4,"tpr","fpr")

plot(perf1, col = "orange",lwd = 2, main = "ROC")
plot(perf2, add = T, col="blue",lwd = 2,  main = "ROC")
plot(perf3, add = T, col="red", lwd = 2, main = "ROC")
plot(perf4, add = T, col="green",lwd = 2,  main = "ROC")
legend("bottomright", legend=c(paste('SVC, AUC= ', auc(as.numeric(test_tsne$Label),
                                                       as.numeric(y_pred1)), sep=""),
                               paste('SVC t-SNE, AUC= ',
auc(as.numeric(test_tsne$Label),
                                                                 as.numeric(y_pred2)),
sep=""),
                               paste('RF, AUC= ', auc(as.numeric(test_tsne$Label),
                                                      as.numeric(y_pred3)),
sep=""),
                               paste('RF t-SNE, AUC= ',
auc(as.numeric(test_tsne$Label),
                                                      as.numeric(y_pred4)),
sep="")),
       col=c("orange","blue", "red", "green"), lty=1, cex=0.8)
```