

**KAUNO TECHNOLOGIJOS UNIVERSITETAS INFORMATIKOS FAKULTETAS**

[**P176B101 Intelektikos pagrindai**](https://moodle.ktu.edu/course/view.php?id=2081)

**technologijos projekto ataskaita**

Data:

2019-05-22

Studentai: Tomas Jasulaitis IFF-6/4

Lukas Biliutavičius IFF-6/4

Tadas Vasiliauskas IFF-6/4

Turinys

[1. Santrauka 2](#_Toc9414984)

[2. **Duomenų surinkimas** (iš atvirų šaltininių panaudojant programinį kodą) 3](#_Toc9414985)

[3. **Duomenų paruošimas ir valymas** (rankiniu būdu ir programiškai) 6](#_Toc9414986)

[4. **Dimensijų sumažinimas** (<pasirinktas sudėtingumas>) 6](#_Toc9414987)

[5. **Įžanginiai eksperimentai, patikrinantys ar dimensijų sumažinimą verta naudoti** (<pasirinktas sudėtingumas>) 6](#_Toc9414988)

[**6.** ***1*-ojo mašininio mokymosi metodo su mokytoju panaudojimas (programinė realizacija iškviečiant atitinkamą biblioteką)** 7](#_Toc9414989)

[**7.** ***2*-tojo mašininio mokymosi metodo su mokytoju panaudojimas (programinė realizacija iškviečiant atitinkamą biblioteką)** 8](#_Toc9414990)

[**8.** ***3*-ojo mašininio mokymosi metodo su mokytoju panaudojimas (programinė realizacija iškviečiant atitinkamą biblioteką)** 10](#_Toc9414991)

[**9.** **Mašininio mokymosi metodų su mokytoju rezultato parinkimas balsavimo principu (pilnai suprogramuota)** 10](#_Toc9414992)

[**10.** **Balsavimo principu gautų rezultatų pritaikymas(pilnai suprogramuota)** 10](#_Toc9414993)

[11. **Mašininio mokymosi be mokytojo metodo panaudojimas** (<pasirinktas sudėtingumas>) 11](#_Toc9414994)

[12. **Mokymosi be mokytojo metodo gautų rezultatų pritaikymas (programoje) / pakomentavimas ataskaitoje** (<pasirinktas sudėtingumas>) 11](#_Toc9414995)

[**13.** **Kryžminė patikra** 11](#_Toc9414996)

[**14.** **Literatūra** 11](#_Toc9414997)

# Santrauka

Duomenų rinkinio pavadinimas – pokerio rankos duomenų rinkinys (Poker Hand Data Set)

<https://archive.ics.uci.edu/ml/datasets/Poker%2BHand?fbclid=IwAR2QtK-sO6WSzK2J4ZpL0g8B9SJIU3uPtoYn6kNs2_oFx4o8uQ6Pn_-eztw>

Problema – nustatyti pokerio kortų kombinaciją iš duotų penkių kortų.

Projekte buvo panaudoti RandomTree, DecisionTree ir K Nearest Neighbours mašininio mokymo metodai, jų tikslumai: 0.956, 0.905 ir 0.66 atitinkamai.

|  |  |  |  |
| --- | --- | --- | --- |
| **Atlikėjas** | **Užsiemimo diena ir laikas** | **Atsakomybė projekte** | **Parengti ataskaitos skyriai** |
| Tomas Jasulaitis | Antradienis 13:30 | 1. Mašininio mokymo metodas su mokytoju (Decision Tree) | Santrauka, duomenų surinkimas, mašininio mokymo su mokytoju metodas. |
| Lukas Biliutavičius | Trečiadienis 11:00 | 1. Mašininio mokymo metodas su mokytoju (Random Forest)  2. Duomenų surinkimas  3. Duomenų paruošimas  4. Balsavimo principu gautų rezultatų pritaikymas.  5. Geriausio metodo su mokytoju rezultato parinkimas. | 1. Duomenų paruošimas ir valymas 2. 2-tojo mašininio mokymosi metodo su mokytoju panaudojimas 3. [Balsavimo principu gautų rezultatų gavimas](#_Toc9367457) 4. Balsavimo principu gautų rezultatų pritaikymas |
| Tadas Vasiliauskas | Trečiadienis 11:00 | 3. Mašininio mokymo metodas su mokytoju | 3-ojo mašininio mokymo su mokytoju metodas |

# **Duomenų surinkimas** (iš atvirų šaltininių panaudojant programinį kodą)

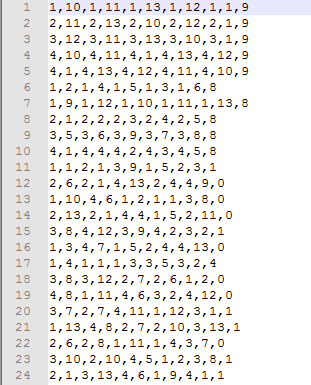
Duomenų rinkinys sudarytas is 11 skaitinių reikšmių. Pirmos 10 reikšmių nusako kortų naudojamų pokerio žaidime rūšį ir skaitinę reikšmę. 11 reikšmė nusako klasę, kuri priskiriama rankos stiprumui (0-9).

Duomenų rinkinyje yra 1025010 įrašų, jo adresas : <https://archive.ics.uci.edu/ml/datasets/Poker%2BHand?fbclid=IwAR2QtK-sO6WSzK2J4ZpL0g8B9SJIU3uPtoYn6kNs2_oFx4o8uQ6Pn_-eztw>

**Atributų informacija:**

1) S1 "Suit of card #1"   
Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}   
  
2) C1 "Rank of card #1"   
Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)   
  
3) S2 "Suit of card #2"   
Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}   
  
4) C2 "Rank of card #2"   
Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)   
  
5) S3 "Suit of card #3"   
Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}   
  
6) C3 "Rank of card #3"   
Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)   
  
7) S4 "Suit of card #4"   
Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}   
  
8) C4 "Rank of card #4"   
Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)   
  
9) S5 "Suit of card #5"   
Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}   
  
10) C5 "Rank of card 5"   
Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)   
  
11) CLASS "Poker Hand"   
Ordinal (0-9)   
  
0: Nothing in hand; not a recognized poker hand   
1: One pair; one pair of equal ranks within five cards   
2: Two pairs; two pairs of equal ranks within five cards   
3: Three of a kind; three equal ranks within five cards   
4: Straight; five cards, sequentially ranked with no gaps   
5: Flush; five cards with the same suit   
6: Full house; pair + different rank three of a kind   
7: Four of a kind; four equal ranks within five cards   
8: Straight flush; straight + flush   
9: Royal flush; {Ace, King, Queen, Jack, Ten} + flush

Pavyzdinis duomenų įrašo pavyzdys.



# **Duomenų paruošimas ir valymas** (rankiniu būdu ir programiškai)

Duomenų formatas pateiktas duomenų failuose:

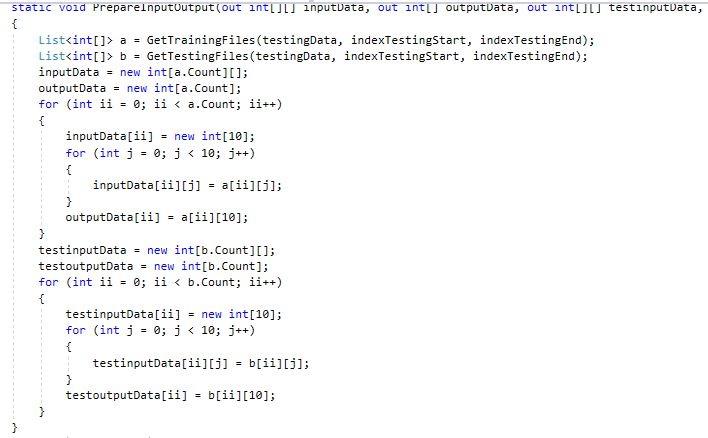
**1,10,1,11,1,13,1,12,1,1,9**

Čia pirmi 10 skaičių – pokerio kortų aprašas (angl. suit, rank). Vienuoliktas skaičius – klasė(stipriausia kortų kombinacija).

Paruošiant duomenis, iš CSV formato reikia perdaryti į dvimačius masyvus.

Pirmasis masyvas [Nx10], kur N – eilučių kiekis duomenų rinkinyję. 10 stulpelių atitinkamai skirta kortos identifikavimui.

Antrasis masyvas [Nx1], paskutinis duomenų eilutės skaitmuo, reiškiantis turimų penkių kortų kombinaciją.



# **Dimensijų sumažinimas** (<pasirinktas sudėtingumas>)

# **Įžanginiai eksperimentai, patikrinantys ar dimensijų sumažinimą verta naudoti** (<pasirinktas sudėtingumas>)

# ***1*-ojo mašininio mokymosi metodo su mokytoju panaudojimas (programinė realizacija iškviečiant atitinkamą biblioteką)**

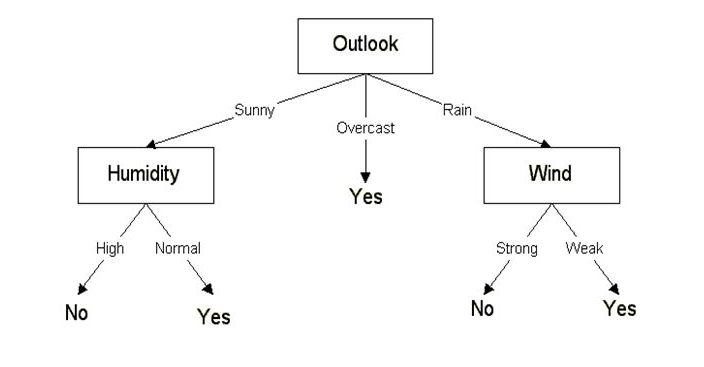
Metodo pavadinimas - Decision Tree

Decision Tree metodas priskiriamas prie mašininio mokymosi su mokytojų algoritmų šeimos. Šiuo algoritmu galima spręsti klasifikacijos ir regresijos problemas. Decision Tree sukuriamas treniravimo modelis, kuris gali būti naudojamas nuspėti klasę ar reikšmę pasirinktų kintamųjų naudojantis mokomojo nusprendimo taisykles (learning decision rules), kurias numanė iš treniravimo duomenų.

Decision Tree metodas yra nesunkiai perprantamas, duomenų paruošimo etapas yra ne toks svarbus nei kitiems modeliavimo metodams.

Šaltinis: <http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/> <https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb>

Iliustracinis pavyzdys:



Nuoroda: <https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html>

|  |  |
| --- | --- |
| Eksperimento numeris | Metodo tikslumo įvertis |
| 1 | 0,90405 |
| 2 | 0,90368 |
| 3 | 0,9058 |
| 4 | 0,90578 |
| 5 | 0,90579 |
| 6 | 0,90614 |
| 7 | 0,90393 |
| 8 | 0,90481 |
| 9 | 0,90639 |
| 10 | 0,90406 |

Kryžminės patikros eksperimentų metodo tikslumo įverčių reikšmės pasiskirsčiusios labai arti viena kitos, tačiau nėra labai aukštos, tik 90.6% tikslumas.

static DecisionTree DecisionTreeClassification(List<int[]> trainingData, List<int[]> testingData, out double precision)

{

int testingCount = testingData.Count / 10;

int trainingCount = testingData.Count - testingCount;

double errorAverage = 0;

int indexTestingStart = testingData.Count - testingCount;

int indexTestingEnd = testingData.Count;

double prec = 0;

Console.WriteLine("Decision Tree Classification");

DecisionTree bestDecision = null;

for (int i = 0; i < iterations; i++){

var watch = System.Diagnostics.Stopwatch.StartNew();

Console.WriteLine("Testing from: {0} to {1}", indexTestingStart, indexTestingEnd);

int[][] inputData, testinputData;

int[] outputData, testoutputData;

PrepareInputOutput(out inputData, out outputData, out testinputData, out testoutputData, trainingData, testingData, indexTestingStart, indexTestingEnd);

ID3Learning teacher = new ID3Learning();

var decision = teacher.Learn(inputData, outputData);

Console.WriteLine("Medis sukurtas - ismokta");

double error = new ZeroOneLoss(testoutputData).Loss(decision.Decide(testinputData));

Console.WriteLine("Apmokymo tikslumas: {0}", 1 - error);

if (1 - error > prec){

prec = 1 - error;

bestDecision = decision;

}

watch.Stop();

var elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Iteracija baigta per: {0}ms", elapsedMs);

indexTestingEnd = indexTestingStart;

indexTestingStart -= testingCount;

errorAverage += error;

bestDecision = decision;

}

precision = 1 - (errorAverage / iterations);

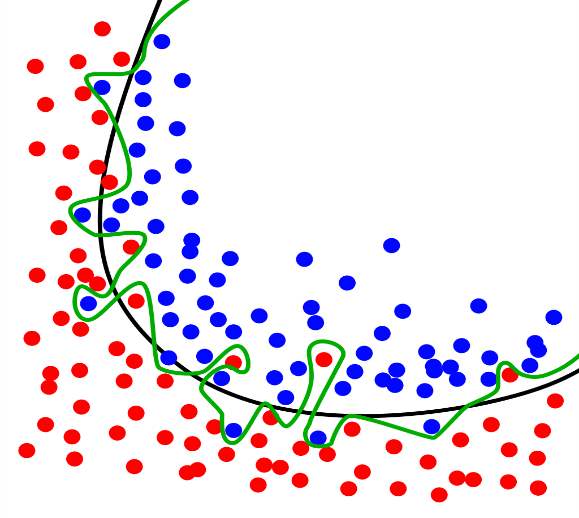
return bestDecision;

}

# ***2*-tojo mašininio mokymosi metodo su mokytoju panaudojimas (programinė realizacija iškviečiant atitinkamą biblioteką)**

Metodo pavadinimas - Random Forest

„Random Forest“ taip pat žinomas kaip „decision forest“ yra populiarus metodas naudojamas kurti modelius klasifikavimui ir regresijai. Šis metodas sukuria daug „decision tree“ kurių pagalba gaunamas geriausias atsakymas. „Random forest“ metodas išsprendžia „decision tree“ metodo persimokymo problemą.



Ši problema atsiranda kai mokymo duomenys yra labai arti vienas kito ir metodas klasifikuodamas gauna didesnę paklaidą. Šioje nuotraukoje juoda kreivė vaizduoja „random forest“ klasifikavima, o žalia – „decision tree“.



Sukūrus mišką, pagal paduodamus duomenis kiekvienas medis balsuoja už atsakymą. Didžiausiai balsų surinkęs atsakymas yra parenkamas kaip klasifikavimo klasė.

Šaltinis: <https://www.datascience.com/resources/notebooks/random-forest-intro>,

<https://en.wikipedia.org/wiki/Random_forest>

|  |  |
| --- | --- |
| Eksperimento numeris | Metodo tikslumo įvertis |
| 1 | 0,95422 |
| 2 | 0,95584 |
| 3 | 0,95683 |
| 4 | 0,95557 |
| 5 | 0,95694 |
| 6 | 0,95517 |
| 7 | 0,95561 |
| 8 | 0,95503 |
| 9 | 0,95565 |
| 10 | 0,95468 |

Kryžminės patikros eksperimentų metodo tikslumo įverčių reikšmės pasiskirsčiusios labai arti viena kitos. Didžiausias klasifikavimo tikslumas – 5 iteracija, **95,7%.**

static RandomForest RandomForestClassification(List<int[]> trainingData, List<int[]> testingData, out double precision)

{

int testingCount = testingData.Count / 10;

int trainingCount = testingData.Count - testingCount;

double errorAverage = 0;

int indexTestingStart = testingData.Count - testingCount;

int indexTestingEnd = testingData.Count;

double prec = 0;

Console.WriteLine("Random Forest Classification");

RandomForest bestforest = null;

for (int i = 0; i < iterations; i++)

{

var watch = System.Diagnostics.Stopwatch.StartNew();

Console.WriteLine("Testing from: {0} to {1}", indexTestingStart, indexTestingEnd);

int[][] inputData, testinputData;

int[] outputData, testoutputData;

PrepareInputOutput(out inputData, out outputData, out testinputData, out testoutputData, trainingData, testingData, indexTestingStart, indexTestingEnd);

var RanForest = new RandomForestLearning()

{

NumberOfTrees = 100,

};

var forest = RanForest.Learn(inputData, outputData);

Console.WriteLine("Medis sukurtas - ismokta");

double er = new ZeroOneLoss(testoutputData).Loss(forest.Decide(testinputData));

Console.WriteLine("Apmokymo tikslumas: {0}", 1 - er);

if(1 - er > prec)

{

prec = 1 - er;

bestforest = forest;

}

watch.Stop();

var elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Iteracija baigta per: {0}ms", elapsedMs);

indexTestingEnd = indexTestingStart;

indexTestingStart -= testingCount;

errorAverage += er;

}

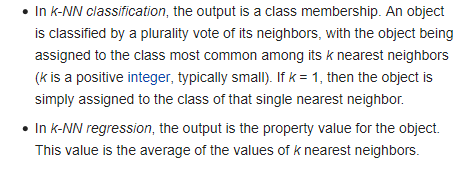
precision = 1 - (errorAverage / iterations);

return bestforest;

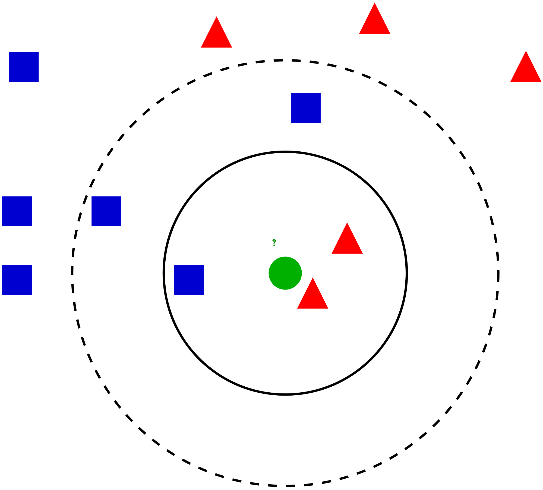
# }

# ***3*-ojo mašininio mokymosi metodo su mokytoju panaudojimas (programinė realizacija iškviečiant atitinkamą biblioteką)**

K-artimiausių kaimynų metodas yra ne parametrinis metodas, kuris naudojamas klasifikavimui ir regresijai. Tiek tam tiek tam metodui, įvedimo duomenys yra k artimiausi apmokymo pavyzdžiai. Išvados priklauso nuo to ar buvo naudojama regresijai ar klasifikavimui.



Šis metodas yra tingus metodas ir yra vienas iš paprasčiausių mašinų mokymo algortimų. Tiek klasifikacijai tiek regresijai galima priskirti svorinius koficientus, tai artimesni kaimynai prisidės prie vidurkio, o ne tolimesni.



Šaltinis: <http://accord-framework.net/docs/html/T_Accord_MachineLearning_KNearestNeighbors_1.htm>

|  |  |
| --- | --- |
| Eksperimento numeris | Metodo tikslumo įvertis |
| 1 | 0,57 |
| 2 | 0.66 |
| 3 | 0.71 |
| 4 | 0.58 |
| 5 | 0.71 |
| 6 | 0.74 |
| 7 | 0.63 |
| 8 | 0.58 |
| 9 | 0.62 |
| 10 | 0.58 |

static KNearestNeighbors kNearestNeighbours(List<int[]> trainingData, List<int[]> testingData, out double precision)

{

KNearestNeighbors temp = null;

int testingCount = testingData.Count / 10;

int trainingCount = testingData.Count - testingCount;

double errorAverage = 0;

double prec = 0;

int indexTestingStart = testingData.Count - testingCount;

int indexTestingEnd = testingData.Count;

Console.WriteLine("k nearest neighbours Classification");

for (int i = 0; i < iterations; i++)

{

var watch = System.Diagnostics.Stopwatch.StartNew();

int[][] inputData, testinputData;

int[] outputData, testoutputData;

PrepareInputOutput(out inputData, out outputData, out testinputData, out testoutputData, trainingData, testingData, indexTestingStart, indexTestingEnd);

double[][] input = new double[inputData.GetLength(0)][];

double a = 0;

for (int j = 0; j < inputData.GetLength(0); j++)

{

input[j] = new double[10];

for(int k = 0; k < 10; k++)

{

a = Convert.ToDouble(inputData[j][k]);

input[j][k] = a;

}

}

double[][] testin = new double[testinputData.Length/1000][];

for (int j = 0; j < testinputData.Length/1000; j++)

{

testin[j] = new double[10];

for (int k = 0; k < 10; k++)

{

testin[j][k] = testinputData[j][k];

}

}

int[] testout = new int[testinputData.Length / 1000];

for (int j = 0; j < testinputData.Length/1000; j++)

{

testout[j] = testoutputData[j];

}

var knn = new KNearestNeighbors(k: 4);

knn.Learn(input, outputData);

var cm = GeneralConfusionMatrix.Estimate(knn, testin, testout);

double error = cm.Error;

double acc = cm.Accuracy;

double kappa = cm.Kappa;

watch.Stop();

var elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Iteracija baigta per: {0}ms", elapsedMs);

Console.WriteLine("Iteracijos tikslumas: {0}", acc);

if (acc > prec)

{

prec = acc;

temp = knn;

}

indexTestingEnd = indexTestingStart;

indexTestingStart -= testingCount;

}

precision = 1 - (errorAverage / iterations);

return temp;

}

# **Mašininio mokymosi metodų su mokytoju rezultato parinkimas balsavimo principu (pilnai suprogramuota)**

|  |  |
| --- | --- |
| Metodas | Tikslumas |
| Random Forest | 0.957 |
| Decision Tree | 0.906 |
| K-nn | 0.66 |

Visiems metodams atlikus kryžminę patikrą ir gražinus vidutinį tikslumą, programinis kodas parenką metodą su aukščiausiu klasifikavimo rodikliu.

# **Balsavimo principu gautų rezultatų pritaikymas(pilnai suprogramuota)**

Programiam kodui parinkus geriausią mašininio mokymosi metodo su mokytoju rezultatą, laukiama įvesties rinkinio.

Rinkinį sudaro 10 skaičių, pvz: 1 1 2 2 3 3 4 4 4 5

Pirmas skaičius: 1 – Širdys, 2 – Vynai, 3 – Būgnai, 4 – Giliai.

Antras skaičius: 1 – Tūzas, 2 – 2, 3 – 3, 4 – 4, 5 – 5, 6 – 6, 7 – 7, 8 – 8, 9 – 9, 10 – 10, 11 – Valetai, 12 – Dama, 13 – Karalius.

Atitinkamai pirmas ir antras skaičiai vaizduoja pirmąją kortą, trečias ir ketvirtas – antrąją kortą ir t.t.

Įvedus šiuos duomenis, geriausias iš trijų klasifikavimo metodų parodys kokią aukščiausia pokerio kombinacija yra galima iš duotų kortų.

static void BestClassificator(RandomForest forest, DecisionTree tree, double[] precision)

{

if(precision[0] > precision[1])

{

Console.WriteLine("Geriausias parinktas klasifikavimo metodas: {0}", "Random Forest");

Console.Write("Iveskite penkiu kortu rinkini, skaicius atskirdamo tarpu: ");

string[] val = Console.ReadLine().Split(' ');

double[] hand = new double[10];

for (int i = 0; i < 10; i++)

{

hand[i] = Convert.ToDouble(val[i]);

}

Console.WriteLine(forest.Decide(hand));

}

else

{

Console.WriteLine("Geriausias parinktas klasifikavimo metodas: %s", "Decision Tree");

Console.Write("Iveskite penkiu kortu rinkini, skaicius atskirdamo tarpu: ");

string[] val = Console.ReadLine().Split(' ');

double[] hand = new double[10];

for (int i = 0; i < 10; i++)

{

hand[i] = Convert.ToDouble(val[i]);

}

Console.WriteLine(tree.Decide(hand));

}

}

# **Mašininio mokymosi be mokytojo metodo panaudojimas** (<pasirinktas sudėtingumas>)

# **Mokymosi be mokytojo metodo gautų rezultatų pritaikymas (programoje) / pakomentavimas ataskaitoje** (<pasirinktas sudėtingumas>)

# **Kryžminė patikra**

Atskirai šis skyrius nepateikiamas, nes yra sudėtinė dalis skyriaus ***i*-tojo mašininio mokymosi metodo su mokytoju panaudojimas**

# **Literatūra**

<http://crsouza.com/2012/01/04/decision-trees-in-c/>

<http://accord-framework.net/docs/html/T_Accord_MachineLearning_DecisionTrees_DecisionTree.htm>

<https://github.com/mdabros/SharpLearning>

<https://csharp.hotexamples.com/examples/Accord.MachineLearning.DecisionTrees.Learning/C45Learning/-/php-c45learning-class-examples.html>

<https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d>

<http://accord-framework.net/docs/html/T_Accord_MachineLearning_KNearestNeighbors_1.htm>

<https://www.datascience.com/resources/notebooks/random-forest-intro>,

<https://en.wikipedia.org/wiki/Random_forest>