KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INTELEKTIKOS PAGRINDAI (P176B101)

4 laboratorinis darbas

Atliko: IFF-6/11 gr. studentas Nerijus Dulkė
Priėmė: doc. Germanas Budnikas

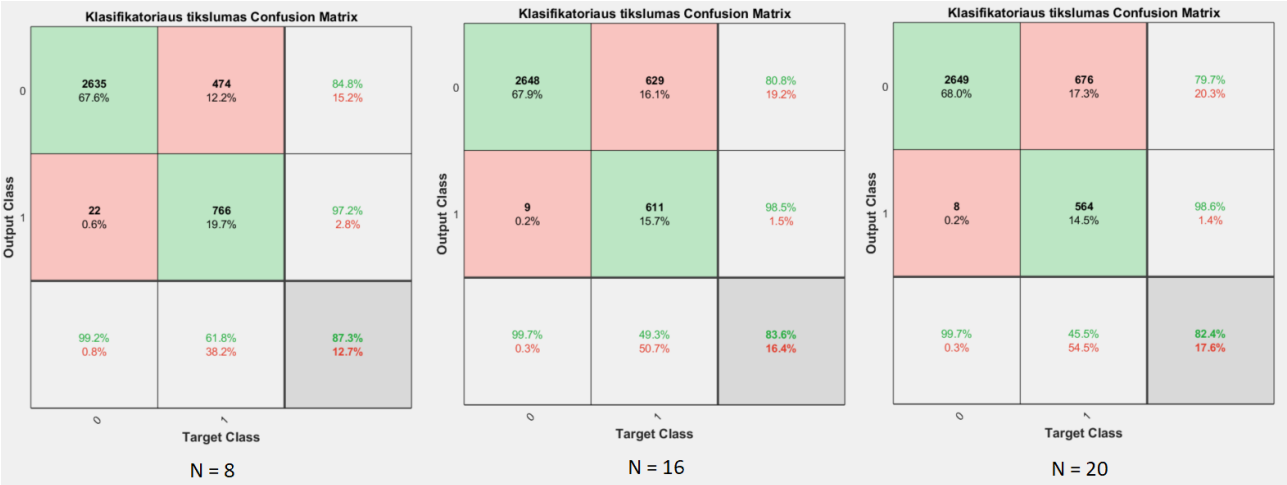
KAUNAS
2019

1. Užduotis

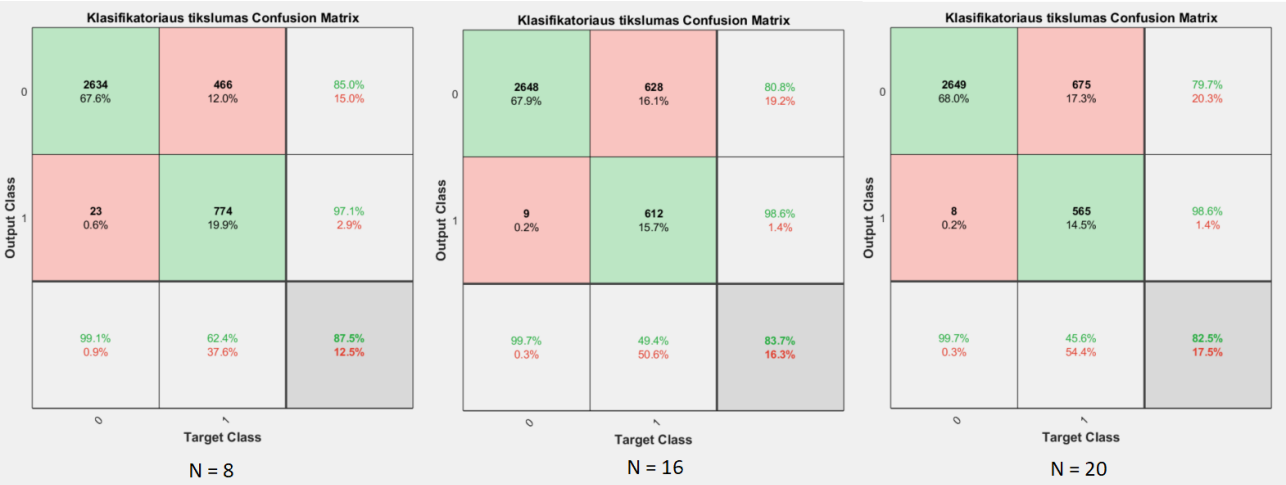
Sukurti programą SPAMui klasifikuoti panaudojant Bajeso teoremą. Ištirti priklausomybę tarp programoje naudojamų nustatymų ir klasifikatoriaus darbo efektyvumo (*žr.reikalavimus ataskaitai*). Programavimo kalba pasirenkama laisvai.

2. Rezultatai

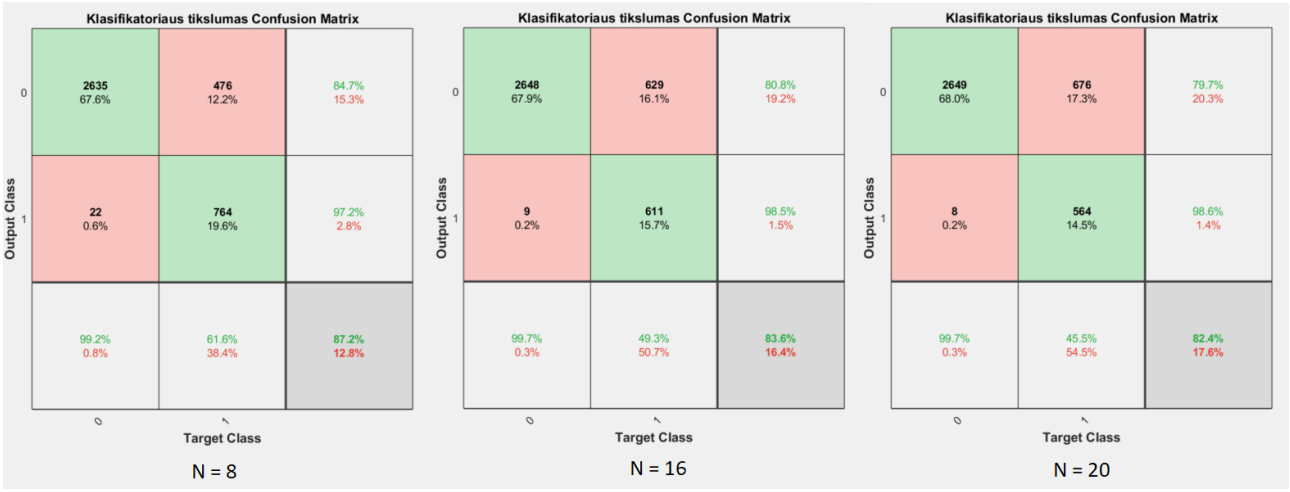
Toliau pateikiami klasifikatoriaus veikimo ir tikslumo rezultatai imant skirtingus parametrus. Tikrinimui naudojama kryžminė patikra (segmentų skaičius – 10).



pav. 1 Klasifikatoriaus tikslumas kai N = 8, N = 16, N = 20, o pirma kartą sutinkamos leksemos spamiškumo tikimybė - 0,4



pav. 2 Klasifikatoriaus tikslumas kai N = 8, N = 16, N = 20, o pirma kartą sutinkamos leksemos spamiškumo tikimybė - 0,8



pav. 3 Klasifikatoriaus tikslumas kai N = 8, N = 16, N = 20, o pirma kartą sutinkamos leksemos spamiškumo tikimybė - 0,1

Iš rezultatų matome, kad didėjant N, mažėja bendras klasifikatoriaus tikslumas. Kuo didesnis N, tuo geriau klasifikatorius atpažįsta ne spamo failus, o kuo N mažesnis, tuo geriau atžįsta spamo failus.

Pirmą kartą sutinkamos leksemos spamiškumo tikimybės kitimas didelės įtakos nesudaro.

3. Programos kodas

```
clear all;

delimiters = {'.','?','!',',',';',':','/','<','>','-','*','+','-',
'[',']','&','_','(',')','=',' ','#','%','@','^','\f','\n','\r',
'\t','\v','\\','\0','','{','}','\b','\a'};

pr = 0.5;
N = 8;

files = dir('Spamas');
files = files(3:size(files,1),1);
SpamPath = string(zeros(size(files)));
n = size(files,1);
for i = 1:n
    SpamPath(i,1) = strcat('Spamas\', files(i,1).name);
end

files  =  dir('Ne_spamas');
files = files(3:size(files,1),1);
NoSpamPath = string(zeros(size(files)));
m = size(files,1);
for i = 1:m
    NoSpamPath(i,1) = strcat('Ne_spamas\', files(i,1).name);
end

parts = 10;
partSizeN = round(n/parts);
partSizeM = round(m/parts);


output = zeros((partSizeN*9)+(partSizeM*9),1);
target = output;
```

```
index = 1;
for k = 0:(parts-2)
    startN = k * partSizeN + 1;
    endN = (k + 1) * partSizeN;
    learnSpam = SpamPath(startN:endN);

    startM = k * partSizeM + 1;
    endM = (k + 1) * partSizeM;
    learnNotSpam = NoSpamPath(startM:endM);

    startTestN = startN + partSizeN;
    if k ~= parts - 2
        endTestN = endN + partSizeN;
    else
        endTestN = size(SpamPath, 1);
    end
    testSpam = SpamPath(startTestN:endTestN);

    startTestM = startM + partSizeM;
    if k ~= parts - 2
        endTestM = endM + partSizeM;
    else
        endTestM = size(NoSpamPath, 1);
    end
    testNotSpam = NoSpamPath(startTestM:endTestM);

    map = Probabilities(learnSpam, learnNotSpam, delimiters);

    for i = 1:size(testSpam,1)
        target(index) = 1;
        p = Frequency(testSpam(i), map, N, delimiters);
        if p > pr
            tmp = " Spamas";
            output(index) = 1;
        else
            tmp = " Nespamas";
            output(index) = 0;
        end
        tmp = strcat("Failo ", testSpam(i), " tikimybe kad yra spam: ", string(p), tmp);
        disp(tmp);
        index = index + 1;
    end

    for i = 1:size(testNotSpam, 1)
        target(index) = 0;
        p = Frequency(testNotSpam(i), map, N, delimiters);
        if p > pr
            tmp = " Spamas";
            output(index) = 1;
        else
            tmp = " Nespamas";
            output(index) = 0;
        end
        tmp = strcat("Failo ", testNotSpam(i), " tikimybe kad yra spam: ", string(p),
tmp);
        disp(tmp);
        index = index + 1;
    end
end


plotconfusion(target',output', 'Klasifikatoriaus tikslumas');


function p = Frequency(filename, map, n, delimiters)
    defaultVal = 0.1;
    f = fileread(char(filename));
```

```
    C = lower(strsplit(f,delimiters))';
    uniq = unique(C);
    prob = zeros(length(uniq),1);
    bool = isKey(map,uniq);

    for i = 1:length(uniq)
        if bool(i) == 1, prob(i) = map(uniq{i});
        else, prob(i) = defaultVal; end
    end

    prob = sort(prob);

    if length(prob) < 2 * n
        n = floor(length(prob) / 2);
    end
        top = prod(prob(1:n)) * prod(prob(length(prob)-n+1:length(prob)));
        bottom = prod(prob(1:n)) + prod(1 - prob(length(prob)-n+1:length(prob)));
        p = top / bottom;
    end

function map = Probabilities(SpamFiles, NoSpamFiles, delimiters)
    mapSpam = containers.Map('KeyType', 'char', 'ValueType', 'double');
    mapNotSpam = containers.Map('KeyType', 'char', 'ValueType', 'double');
    map = containers.Map('KeyType', 'char', 'ValueType', 'double');

    for i = 1:size(SpamFiles,1)
        f = fileread(char(SpamFiles(i)));
        C = lower(strsplit(f,delimiters))';
        [uniq, ~, j] = unique(C);
        freq = accumarray(j, 1);
        bool = isKey(mapSpam, uniq);

        for j = 1:length(uniq)
            if (bool(j) == 1), mapSpam(uniq{j}) = mapSpam(uniq{j}) + freq(j);
            else, mapSpam(uniq{j}) = freq(j); end
        end
    end

    for i = 1:size(NoSpamFiles,1)
        f = fileread(char(NoSpamFiles(i)));
        C = lower(strsplit(f,delimiters))';
        [uniq, ~, j] = unique(C);
        freq = accumarray(j, 1);
        bool = isKey(mapNotSpam, uniq);

        for j = 1:length(uniq)
            if (bool(j) == 1), mapNotSpam(uniq{j}) = mapNotSpam(uniq{j}) + freq(j);
            else, mapNotSpam(uniq{j}) = freq(j); end
        end
    end

    words = unique(horzcat(keys(mapSpam), keys(mapNotSpam)))';
    bool = isKey(mapSpam,words);
    bool2 = isKey(mapNotSpam,words);
    PSW = ones(length(words),1);
    totalS = sum(double(string(values(mapSpam))));
    totalN = sum(double(string(values(mapNotSpam))));

    for i = 1:length(words)
        if bool(i) ~= 1, PSW(i) = 0.01; end
        if bool2(i) ~= 1, PSW(i) = 0.99; end
        if bool(i) && bool2(i)
            PSW(i) = 1/(1+(mapNotSpam(words{i})*totalS)/(mapSpam(words{i})*totalN));
        end
        map(words{i}) = PSW(i);
    end
    end
```