České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra kybernetiky
Katedra počítačů

Vytěžování dat – cvičení IV

k-NN

Miroslav Čepek: cepekm1@fel.cvut.cz
Pavel Kordík: kordikp@fel.cvut.cz

Program cvičení

- Metoda nejbližších sousedů (KNN)
  - Klasifikace automobilů pomocí KNN
  - Trénovací a testovací data
  - Klasifikace a výpočet chyby
  - Závislost chyby na velikosti trénovací množiny
  - Závislost chyby na počtu sousedů
  - Vizualizace 1NN pomocí Voronoiových diag.
  - Vizualizace kNN pro různá k

Načtení dat

- V dnešním cvičení budeme opět používat databázi aut.
- Načtěte soubor `auto-mpg.data-mod-names.csv` do objektu dataset a definujte jména jednotlivých atributů

Načtení a normalizace dat

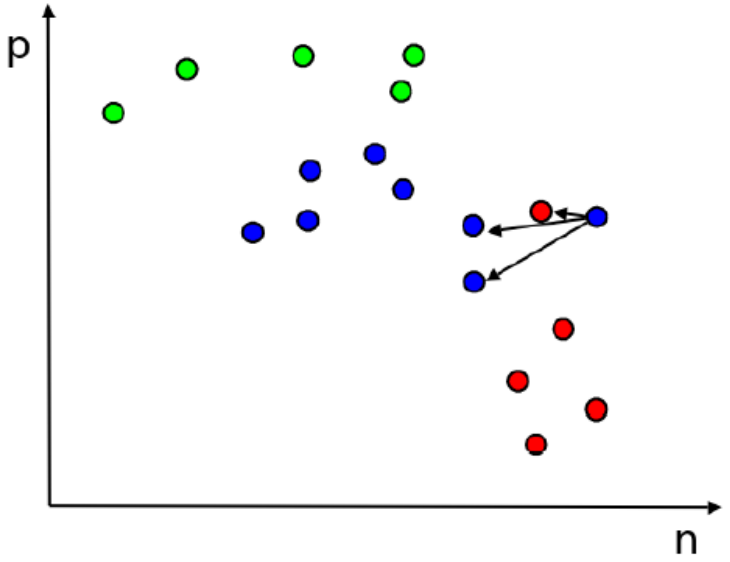
- Skript `loadAndNormalizeData.m`:

```
% Load data
auta = dataset('file','auto-mpg.data-mod-names.csv', ...
    'ReadVarNames', false, 'ReadObsNames', false, ...
    'delimiter', ',', ...
    'VarNames',{'mpg', 'cyl', 'disp', 'hp', ...
                'wgt', 'acc', 'year', 'org', 'name'});

% Normalize them
varInd = 1:7;
auta_norm = datasetfun( @minmax, auta(:,varInd), ...
              'UniformOutput', false );
auta_norm = [auta_norm{:}];
auta = replacedata( auta, auta_norm, varInd );
```

K-NN připomenutí

- Učení modelu: zapamatování trénovací sady
- Použití modelu: při klasifikaci nové instance naleznu v trénovací množině nejbližší instanci (**k** nejbližších instancí) a podle jejich tříd určím výslednou třídu nové instance.



Trénovací a testovací množina

- Nejprve musíme vytvořit trénovací a testovací množiny.
- Rozdělíme náš dataset na dvě části a jednu použijeme na trénování a druhou na testování.
- Jak dataset rozdělit?
  - První polovinu použít na trénování, druhou na testování.
  - Náhodně rozdělovat instance.

Rozdělení dat I

- První polovinu datasetu použijeme pro trénování.
- Druhou polovinu pro testování.
- Jak to udělat?

Rozdělení dat I

- První polovinu datasetu použijeme pro trénování.
- Druhou polovinu pro testování.
- Jak to udělat?

```
auta_tren = auta(1:pocet_aut/2,:);
auta_test = auta(pocet_aut/2+1:pocet_aut,:);
```

- Co může být problém při tomto způsobu dělení? Je trénovací a testovací množina reprezentativní podmnožinou?

Lépe: náhodné rozdělení dat

Vysvětlete:

```
function [itrain, itest] = splitData(N, trainingFraction)
% SPLITDATA Returns indices of training and testing cases

    % Version 1
    rvec = rand(N,1);
    itrain = (rvec <= trainingFraction);
    itest = (rvec > trainingFraction);

%     % Version 2
%     ntrain = ceil(N * trainingFraction);
%     ind = randperm(N);
%     itrain = ind(1:ntrain);
%     itest = ind(ntrain+1:end);
end
```

Učení a použití modelu kNN

- Učení modelu: funkce **`trainClassKNN.m`**

```
function model = trainClassKNN( inputs, outputs, k )
% TRAINCLASSKNN Trains KNN model by storing the training data.
    model.in = inputs;
    model.out = outputs;
    model.k = k;
    model.fun = 'predClassKNN';
end
```

- Použití modelu: funkce **`predClassKNN.m`**

```
function pred = predClassKNN(model, testInputs )
% PREDCLASSKNN k-NN classification
    indNN = knnsearch( testInputs, model.in, model.k );
    nClasses = length( unique( model.out ) );
    pred = classify2( indNN, model.out, nClasses );
end
```

Najdi k nejbližších sousedů

- Funkce pro výpočet nejbližšího souseda:

```
[indexy_nejblizsich, vzdalenosti_k_nejblizsim] =
   knnsearch(testovaci_mn, trenovaci_mn, k)
```

- Pro všechny testovací instance vrátí pole indexů nejbližších sousedů z trénovací množiny a pole vzdáleností k nim
- Najděte v kódu funkce výpočet vzdálenosti

Najdi k nejbližších sousedů

- Pro 1NN

```
if K==1
    % Loop for each query point
    for k=1:N
        d=zeros(L,1);
        for t=1:M
            d=d+(R(:,t)-Q(k,t)).^2;
        end
        [D(k),idx(k)]=min(d);
    end
```

Testovací instance

Trénovací množina

Najdi k nejbližších sousedů

- kNN

```
for k=1:N
        d=zeros(L,1);
        for t=1:M
            d=d+(R(:,t)-Q(k,t)).^2;
        end
        [s,t]=sort(d);
        idx(k,:)=t(1:K);
        D(k,:)=s(1:K);
    end
```

Testovací instance

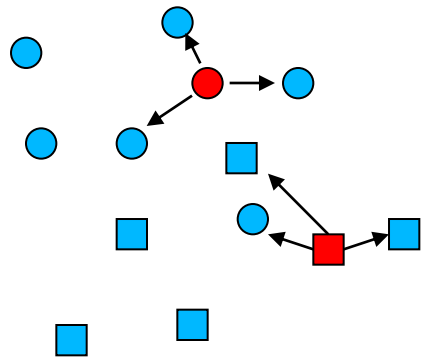
Trénovací množina

Seřaď vzdálenosti
s – vzdálenosti,
t – indexy

Klasifikuj do majoritní třídy

- Funkce pro klasifikaci z indexů nejbližších sousedů

```
[predikovane_tridy] = ...
   classify2(...
   indexy_nejblizsich_sousedu, ...
   tridy_trenovacich_dat, ...
   pocet_trid)
```



3NN

Klasifikuj do majoritní třídy

- Funkce pro klasifikaci z indexů nejbližších sousedů

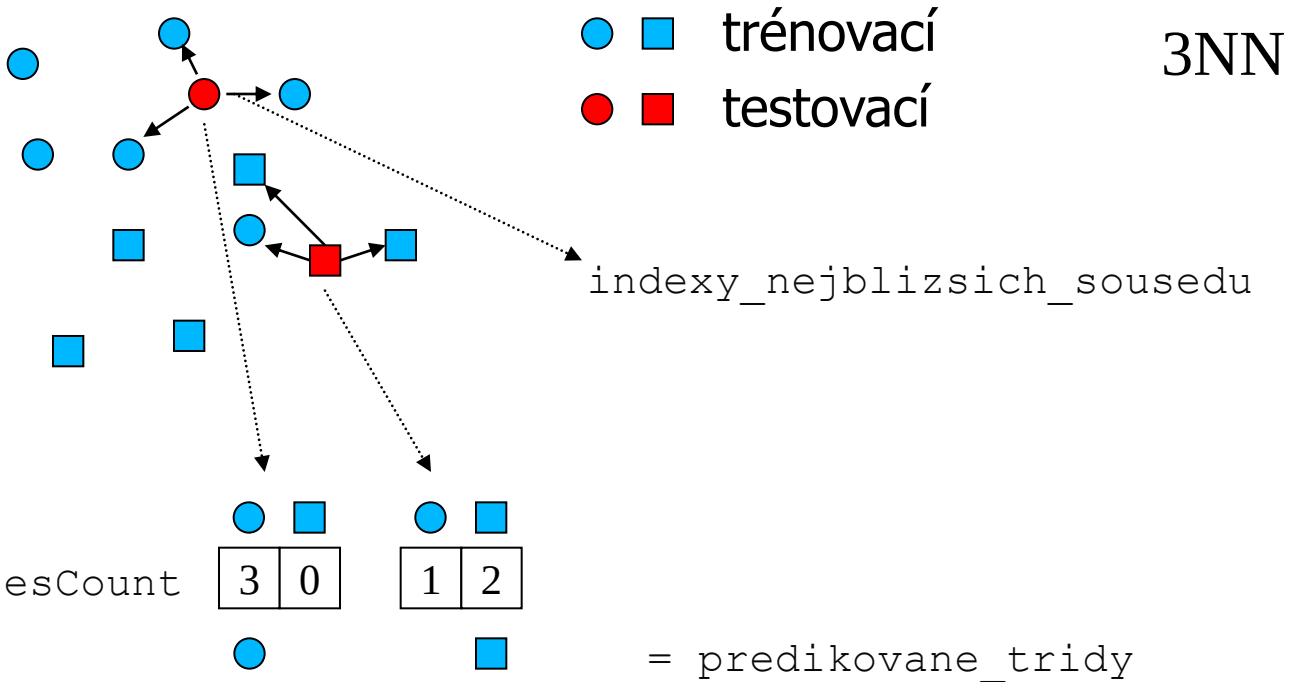
```
[predikovane_tridy] = classify2(indexy_nejblizsich_sousedu,
   tridy_trenovacich_dat, pocet_trid)

function class = classify2(nearestIdxs, trainingClasses, numClasses)
    class = zeros(1,length(nearestIdxs));
    for i = 1:length(nearestIdxs)
        classesCount = zeros(1,numClasses);
        for j = 1:numClasses
            classesCount(j) = sum(...
                trainingClasses(nearestIdxs(i,:),:) == j);
        end
        [cnt,finalClass] = max(classesCount);
        class(i) = finalClass;
    end
end
```

Klasifikuj do majoritní třídy

- Funkce pro klasifikaci z indexu nejbližších sousedů

```
[predikovane_tridy] = classify2(indexy_nejblizsich_sousedu,
   tridy_trenovacich_dat, pocet_trid)
```



Výpočet chyby (přesnosti)

- Jaká je přesnost našeho klasifikátoru?

- Hint:
  - Co vrací funkce `classify2`?
  - Jak zjistíme, zda jsou jednotlivé odpovědi správné?
  - Jak spočítáme počet správných odpovědí a vypočítáme přesnost?

Výpočet chyby (přesnosti)

- Jaká je přesnost našeho klasifikátoru?

```
classes = classify2(idx, auta_tren.org, 3);
isCorrect = (classes == double(auta_test.org));
numCorrectClassif = sum(isCorrect);
```

- Úspěšnost je numCorrectClassif/Počet vektorů testovací množiny * 100%

Tedy:

```
[indtren, indtest] = splitData(auta, 0.5);
auta_tren = auta(indtren,:);
auta_test = auta(indtest,:);
auta_tren_in = [auta_tren.mpg auta_tren.disp];
auta_test_in = [auta_test.mpg auta_test.disp];
model = trainStore(auta_tren_in, auta_tren.org);
classes = predClassKNN(model, auta_test_in, 1);
isCorrect = (classes == double(auta_test.org));
numCorrectClassif = sum(isCorrect);
```

Experiment I: velikost trénovací množiny a testovací chyba

- Náhodně rozdělte data na trénovací 1/4 a testovací 3/4 ze všech dat
- Používejte stále stejnou testovací množinu, z trénovací množiny vyberte 1% do 99% po 1% a vybranou část použijte pro trénování 1NN klasifikátoru
- Sledujte testovací chybu 1NN a následně vykreslete graf

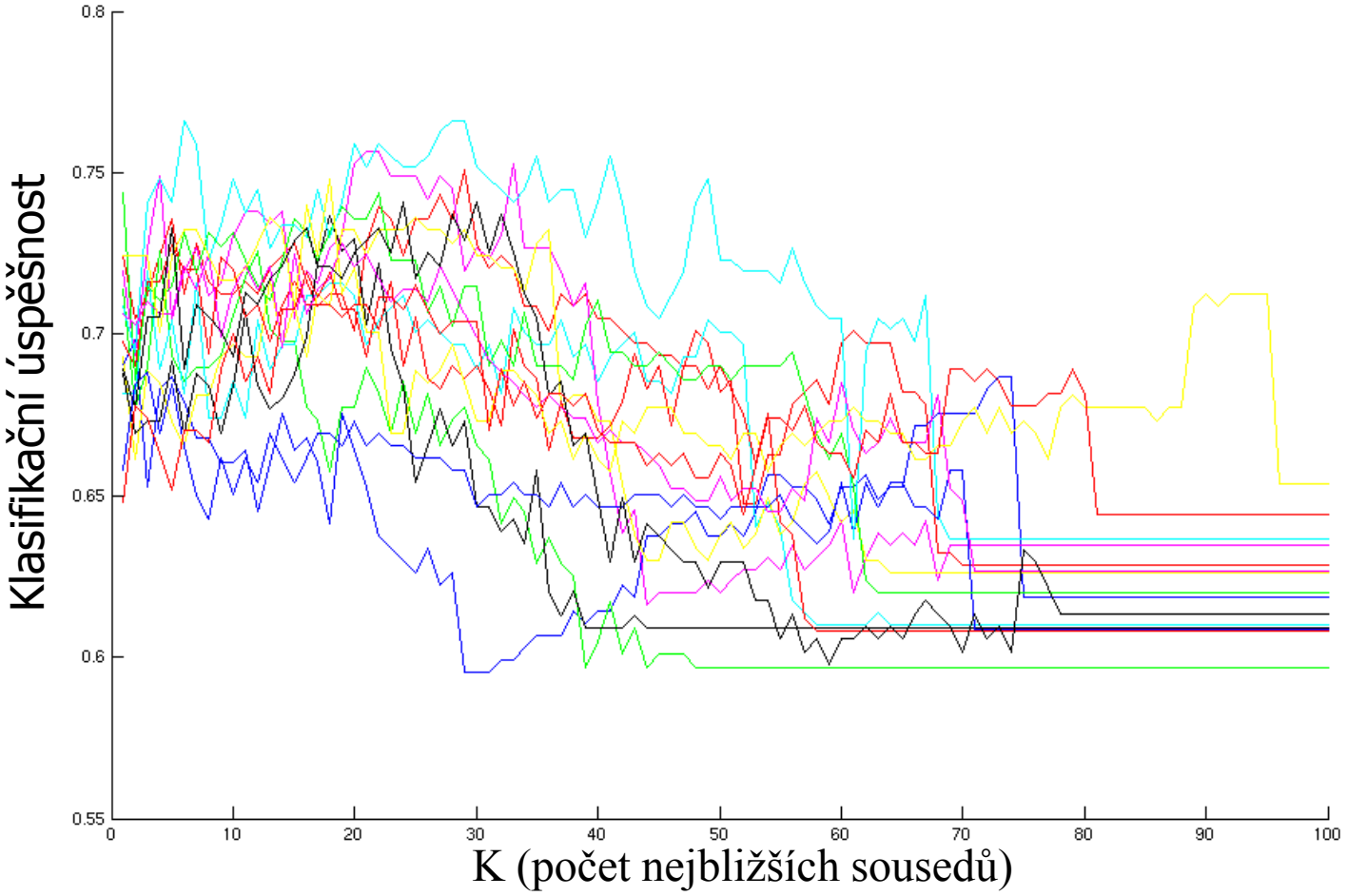
Experiment II: počet sousedů a testovací chyba

- Modifikujte program tak, že
  - budete používat celou trénovací množinu,
  - budete měnit počet sousedů od 1 do 100

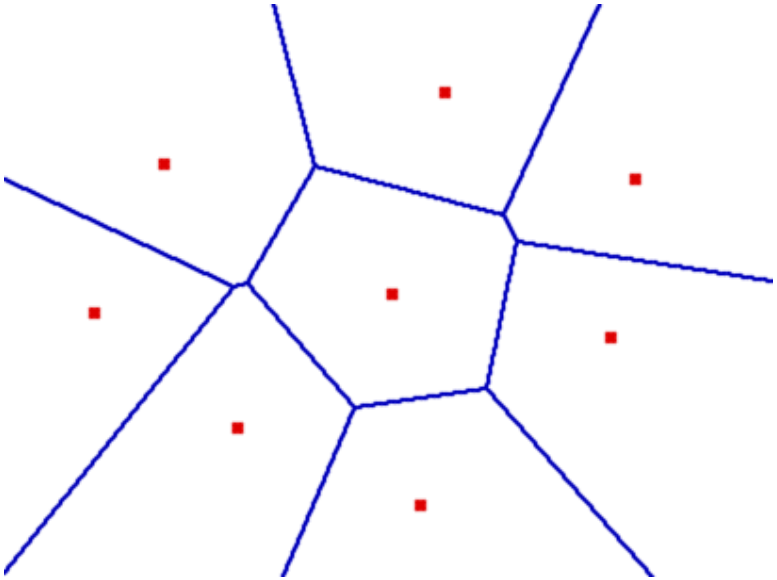
- Sledujte testovací chybu kNN a následně vykreslete graf

Budou grafy pokaždé stejné?

- 10 běhů programu v jednom grafu:

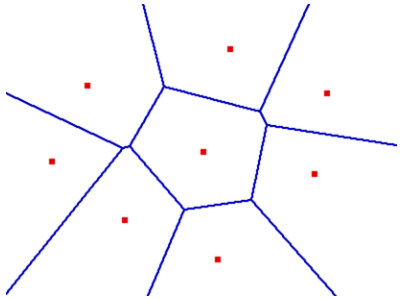


Co se děje uvnitř?

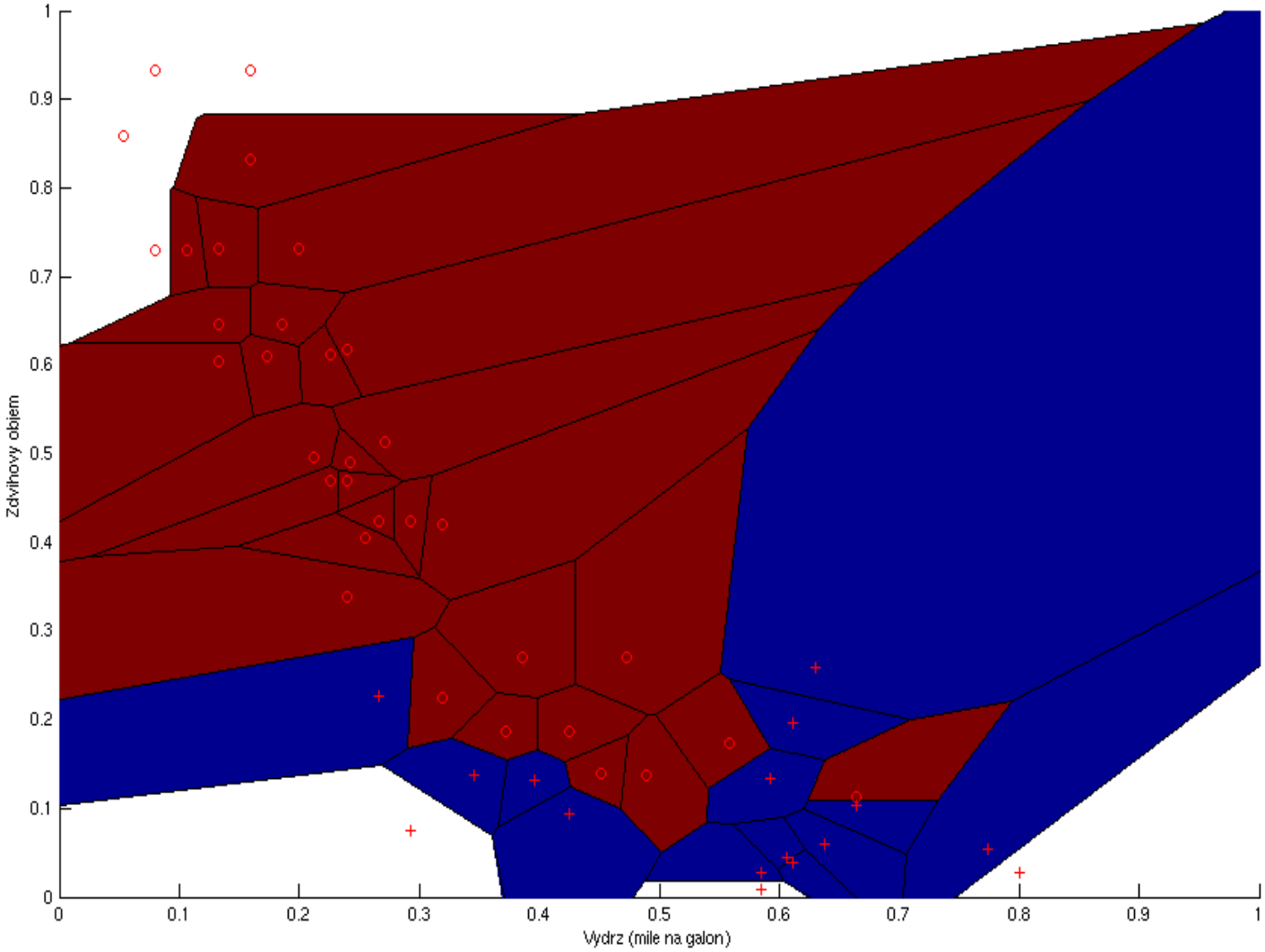


Voronoiův diagram

- je způsob rozdělení metrického prostoru určený vzdálenostmi k dané diskrétní množině (trénovacích) bodů.



- Lze použít pouze pro 1NN



Je vizualizace úplně korektní?

Vizualizace

- Jak vizualizovat výsledky k-NN a jeho rozhodovací hranice?
- Voronoiův diagram použít nelze.
- Vytvoříme umělá auta s různými parametry mpg a disp.
- Zkusíme, kam je zařadí náš k-NN klasifikátor.
- A zobrazíme.

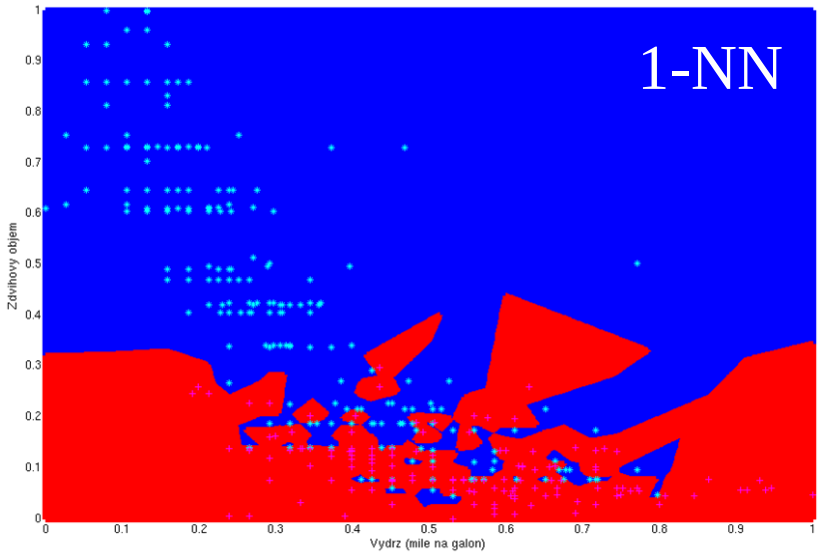
Vizualizace

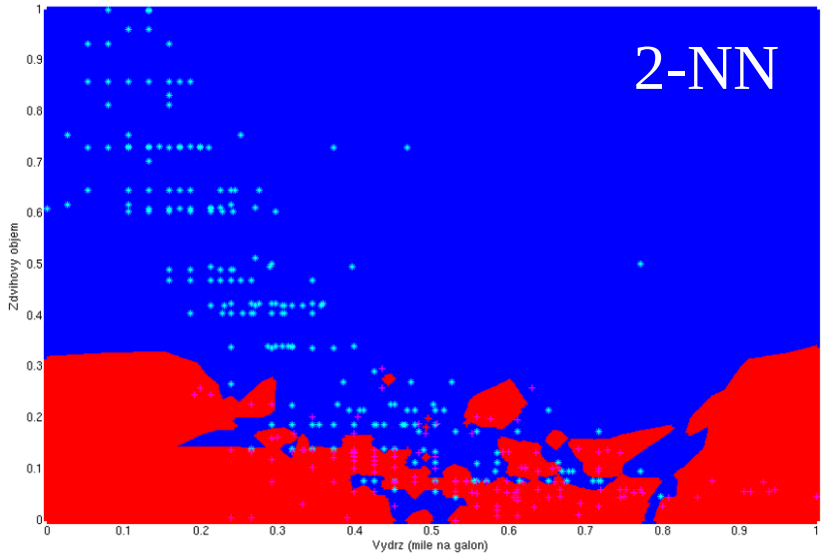
Vytvořte skript, který

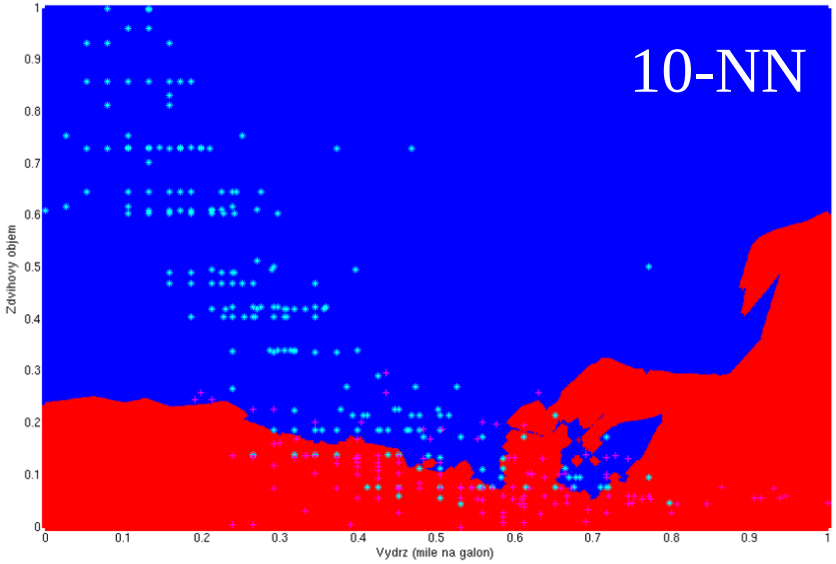
- vygeneruje body v celém rozsahu grafu
- tyto nové body zaklasifikuje metodou KNN
- vykreslí výsledky do grafu
- barva bodu záleží na třídě

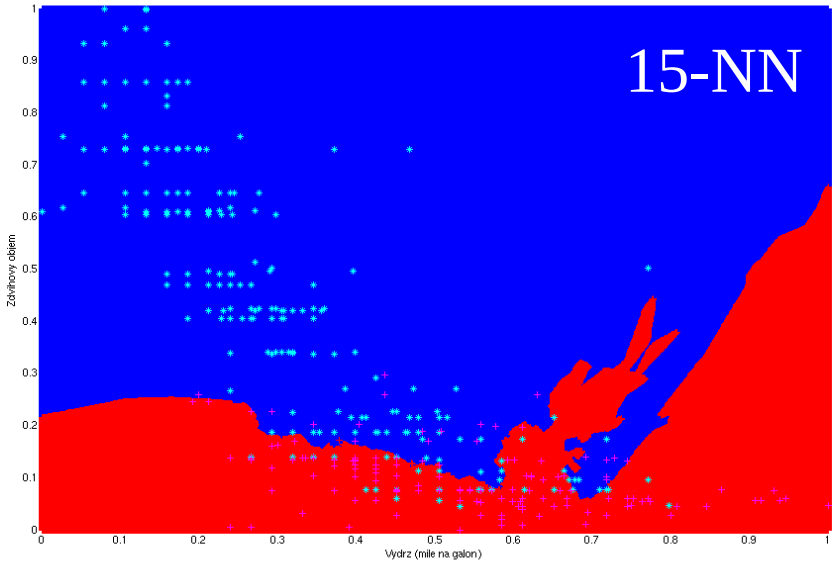
Sledujte, jak se klasifikace mění

- při opakovaném spouštění pro stejné *k*,
- pro různá *k*.


1-NN
Zdvihovy objem
Vydrz (mile na galon)


2-NN
Zdvihovy objem
Vydrz (mile na galon)


10-NN
Zdvihovy objem
Vydrz (mile na galon)


15-NN
Zdvihovy objem
Vydrz (mile na galon)