TU: K-Nearest Neighbors

Industrial AI & Automation by Y.K.Kim

Mod: 2024-2

Introduction

Classification with kNN

Gyeonheal An

21900416

Example: Classification Using Nearest Neighbors

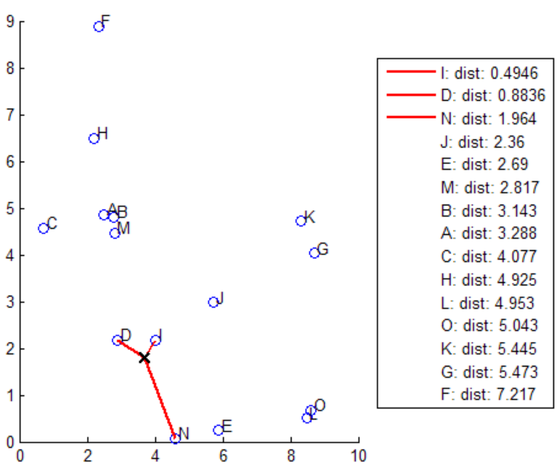
Distance Metrics

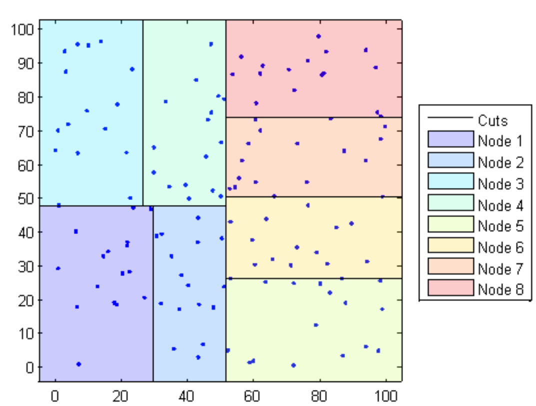
Use `pdist2` to find the distance between a set of data and query points.

- Euclidean distance
- Standardized Euclidean distance
- Mahalanobis distance
- Cosine distance

k-Nearest Neighbor Search and Radius Search

- **Exhaustive Search(default)**
- **Kd-Tree (feature <10)**





Randomly generate normally distributed data into two matrices. The number of rows can vary, but the number of columns must be equal. This example uses 2-D data for plotting.

```
rng(1) % For reproducibility

% Input Data
X = randn(50,2);

% Query
Y = randn(4,2);

h = zeros(3,1);
figure
h(1) = plot(X(:,1),X(:,2),'bx');
hold on
h(2) = plot(Y(:,1),Y(:,2),'rs','MarkerSize',10);
title('Heterogeneous Data')
```

Mahalanobis distance

Find the indices of the three nearest observations in `X` to each observation in `Y`.

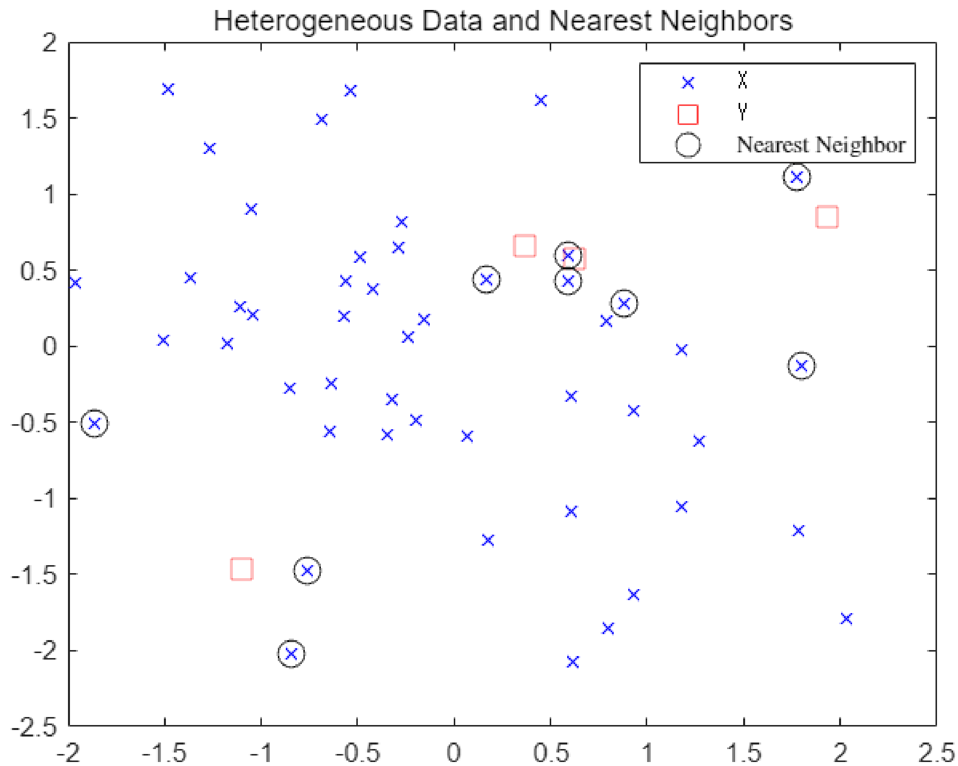
```
k = 3;
[Idx,D] = knnsearch(X,Y,'Distance','mahalanobis','k',k);
```

`idx` and `D` are 4-by-3 matrices.

- `idx(j,1)` is the row index of the closest observation in `X` to observation *j* of `Y`, and `D(j,1)` is their distance.
- `idx(j,2)` is the row index of the next closest observation in `X` to observation *j* of `Y`, and `D(j,2)` is their distance.
- And so on.

Identify the nearest observations in the plot.

```
for j = 1:k
    h(3) = plot(X(Idx(:,j),1),X(Idx(:,j),2),'ko','MarkerSize',10);
end
legend(h,{'\texttt{X}','\texttt{Y}','Nearest Neighbor'},'Interpreter','latex')
title('Heterogeneous Data and Nearest Neighbors')
hold off
```



Exercise

Exercise : K-NN Classification with CWRU

Dataset : CWRU dataset features

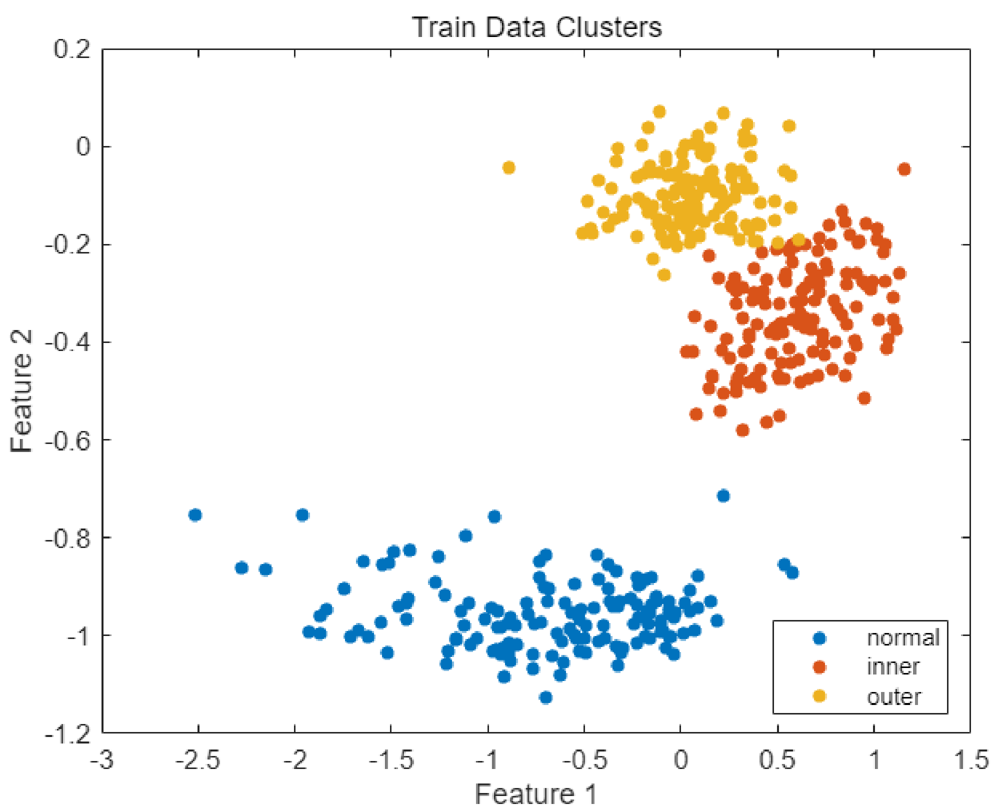
- Given dataset contains many features extracted from CWRU dataset
- We will select 2~3 features for exercise
- Normal, Outer and Inner Race Fault

```
clear

%% Train
load("../../Dataset/CWRU_selected_dataset/Feature_data/sample_train.mat");

feature1 = "sv";                    % skewness value of time data
feature2 = "ipf";                   % impulse factor
X(:, 1) = table2array(glob_all_train(:, feature1));
X(:, 2) = table2array(glob_all_train(:, feature2));
Y = class_cwru_train;       % fault class
N = size(X,1);
tbl=table(X(:, 1),X(:, 2),Y);
```

```
%% Test
load("../../Dataset/CWRU_selected_dataset/Feature_data/sample_test.mat");

Xtest(:, 1) = table2array(glob_all_test(:, feature1));
Xtest(:, 2) = table2array(glob_all_test(:, feature2));
Ytest = class_cwru_test;
Ntest=size(Xtest,1);
tblTest=table(Xtest(:, 1),Xtest(:, 2),Ytest);
```

Plot Test Data

```
figure
gscatter(X(:,1),X(:,2),Y)
title('Train Data Clusters')
xlabel('Feature 1')
ylabel('Feature 2')
```



KNN Train

Construct the classifier using `fitcknn`.

```
rng(10); % For reproducibility

Mdl = fitcknn(X,Y)
```

```
Mdl = 
  ClassificationKNN
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: {'inner'  'normal'  'outer'}
           ScoreTransform: 'none'
          NumObservations: 432
                 Distance: 'euclidean'
             NumNeighbors: 1


  Properties, Methods
```

Training loss (all train set)

Examine the resubstitution loss, which, by default, is the fraction of misclassifications from the predictions of `Mdl`. (For nondefault cost, weights, or priors, see loss.).

```
%%% YOUR CODE GOES HERE
mlResubErr = resubLoss(Mdl)
```

```
mlResubErr = 0
```

The classifier predicts incorrectly for 4% of the training data.

Cross-validation (k-fold)

Construct a cross-validated classifier from the model.

```
%%% YOUR CODE GOES HERE
cp = cvpartition(Y,'KFold',10)  %k-fold
```

```
cp = 
K-겹 교차 검증 분할
   NumObservations: 432
       NumTestSets: 10
         TrainSize: 389   388   388   389   389   389   389   389   389   389
          TestSize: 43   44   44   43   43   43   43   43   43   43
          IsCustom: 0
```

Examine the cross-validation loss, which is the average loss of each cross-validation model when predicting on data that is not used for training.

```
%%% YOUR CODE GOES HERE
cvml = crossval(Mdl,'CVPartition',cp);
mlCVErr = kfoldLoss(cvml)
```

```
mlCVErr = 0.0139
```

The cross-validated classification accuracy resembles the resubstitution accuracy.

Therefore, you can expect `Mdl` to misclassify approximately 4% of new data, assuming that the new data has about the same distribution as the training data.

Predict test data

Predict the classification of test data

```
%%% YOUR CODE GOES HERE
pred = predict(Mdl,Xtest)
```

```
pred = 108×1 cell
'normal'
'normal'
'normal'
'normal'
'normal'
'normal'
'normal'
'normal'
'normal'
'normal'
    ⋮
```

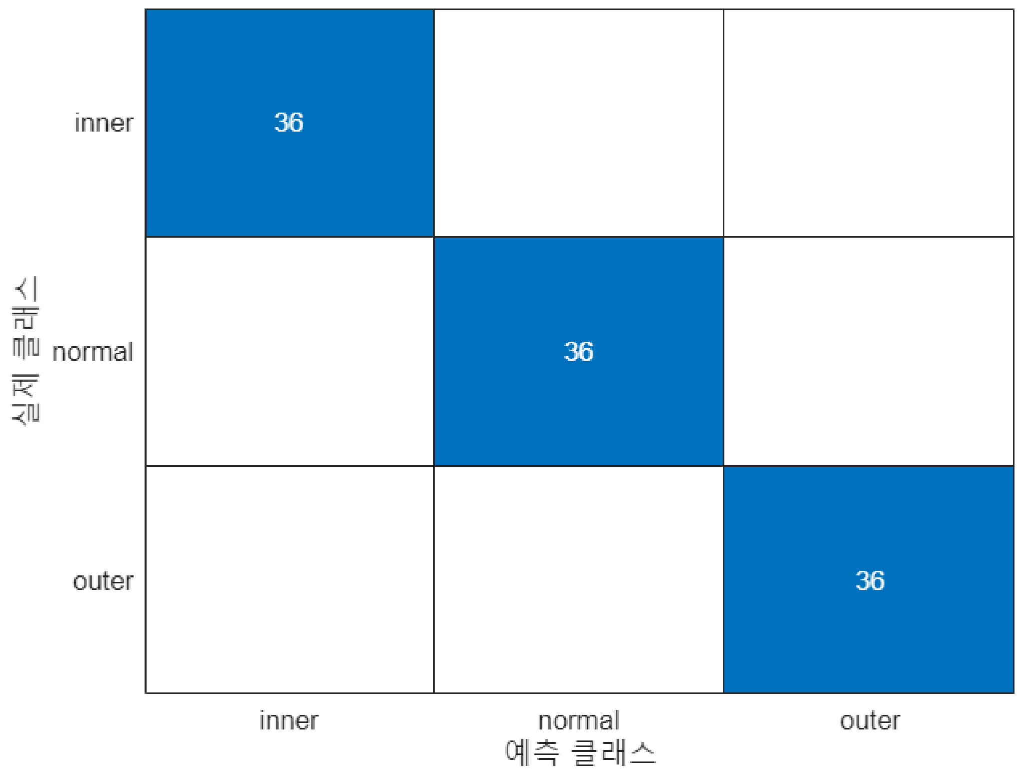
Calculate the loss of Test

```
%%% YOUR CODE GOES HERE
loss = loss(Mdl,Xtest,Ytest)
```

```
loss = 0
```

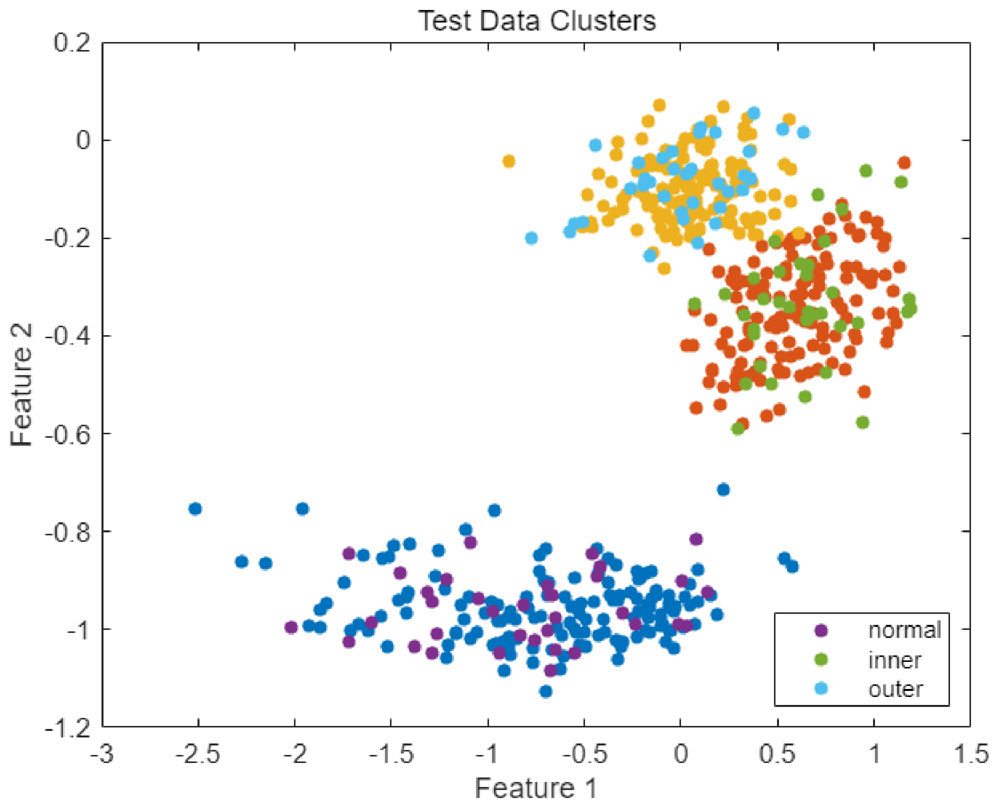
Plot Confusion matrix of Test data

```
%%% YOUR CODE GOES HERE
figure
KNNResubCM = confusionchart(Ytest,pred);
```



Plot Test Results and Misclassified Test data

```
%%% YOUR CODE GOES HERE
bad = ~strcmp(pred,Ytest);

color = lines(6);                  % Generate color values
figure
gscatter(X(:,1),X(:,2),Y, color(1:3,:))
hold on
gscatter(Xtest(:,1),Xtest(:,2),Ytest, color(4:6,:))
hold on;
plot(Xtest(bad,1), Xtest(bad,2), 'kx');
hold off;
title('Test Data Clusters')
xlabel('Feature 1')
ylabel('Feature 2')
```

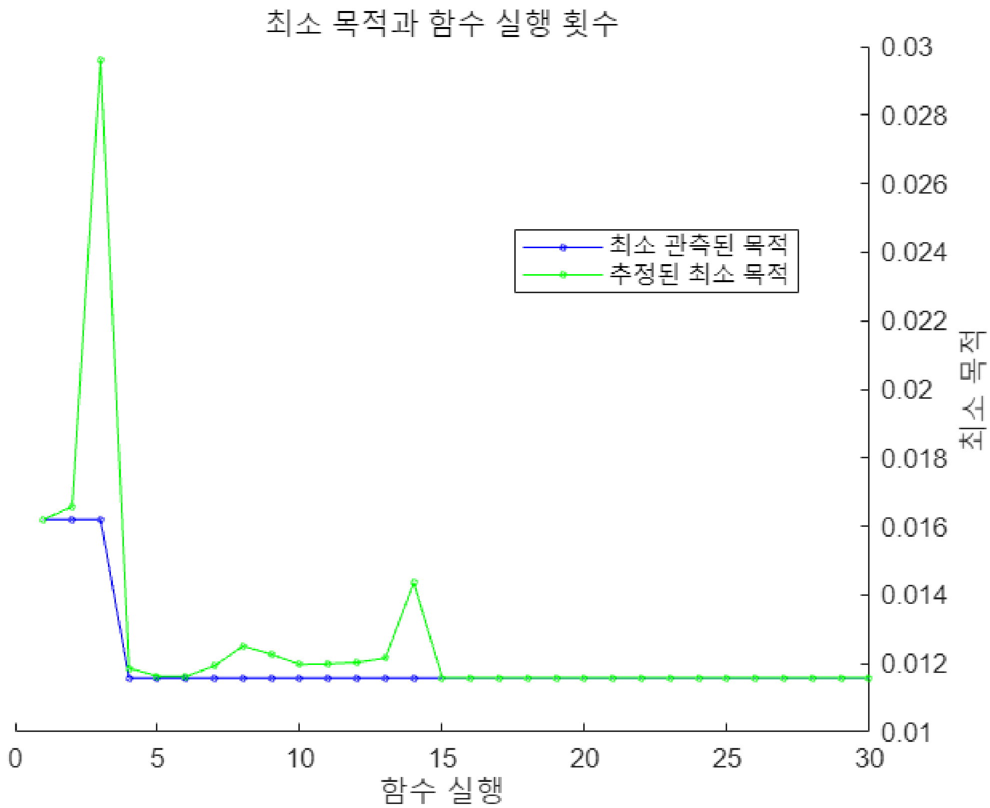


Optimization of Fitted KNN

```
Mdl = fitcknn(X,Y,'OptimizeHyperparameters','auto',...
    'HyperparameterOptimizationOptions',...
    struct('AcquisitionFunctionName','expected-improvement-plus'))
```

```
|====================================================================================================================
| Iter | Eval   | Objective    | Objective    | BestSoFar    | BestSoFar    | NumNeighbors |     Distance | Standardize
|      | result |              | runtime      | (observed)   | (estim.)     |              |              |
|====================================================================================================================
|    1 | Best   |     0.016204 |     0.046001 |     0.016204 |     0.016204 |            1 |    seuclidean |       false
|    2 | Accept |     0.025463 |     0.039783 |     0.016204 |     0.016572 |           23 |    chebychev |       false
|    3 | Accept |      0.66667 |     0.042796 |     0.016204 |     0.029609 |          179 |      hamming |        true
|    4 | Best   |     0.011574 |     0.045459 |     0.011574 |     0.011864 |            2 |    chebychev |        true
|    5 | Accept |     0.011574 |     0.038971 |     0.011574 |      0.01163 |            2 |    cityblock |        true
|    6 | Accept |     0.011574 |     0.039378 |     0.011574 |     0.011619 |            6 |    cityblock |       false
|    7 | Accept |      0.66667 |     0.036636 |     0.011574 |     0.011932 |            2 |  correlation |        true
|    8 | Accept |      0.66667 |      0.07005 |     0.011574 |     0.012507 |            1 |     spearman |       false
|    9 | Accept |     0.016204 |      0.03637 |     0.011574 |     0.012267 |            1 |    minkowski |       false
|   10 | Accept |     0.016204 |     0.039461 |     0.011574 |     0.011974 |            1 |    minkowski |        true
|   11 | Accept |     0.016204 |     0.039691 |     0.011574 |     0.011993 |            1 |  mahalanobis |       false
|   12 | Accept |      0.66667 |     0.038728 |     0.011574 |     0.012033 |            1 |      jaccard |       false
|   13 | Accept |      0.34954 |     0.042705 |     0.011574 |     0.012153 |          215 |       cosine |       false
|   14 | Accept |     0.016204 |     0.039328 |     0.011574 |     0.014361 |            1 |    euclidean |       false
|   15 | Accept |     0.011574 |     0.042534 |     0.011574 |     0.011585 |           29 |    cityblock |        true
|   16 | Accept |     0.016204 |     0.040828 |     0.011574 |     0.011584 |            1 |    euclidean |        true
|   17 | Accept |     0.037037 |     0.040307 |     0.011574 |     0.011584 |          148 |       cosine |        true
|   18 | Accept |       0.6088 |     0.040362 |     0.011574 |     0.011585 |          209 |  correlation |       false
```

```
|   19 | Accept |     0.36343 |    0.073921 |    0.011574 |    0.011585 |          190 |     spearman |         true
|   20 | Accept |     0.66667 |    0.036408 |    0.011574 |    0.011587 |            1 |      hamming |        false
|===================================================================================================================
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | NumNeighbors |     Distance |  Standardize
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|===================================================================================================================
|   21 | Accept |     0.66667 |    0.042365 |    0.011574 |    0.011588 |          208 |      jaccard |         true
|   22 | Accept |    0.085648 |    0.040062 |    0.011574 |    0.011588 |          215 |    cityblock |        false
|   23 | Accept |     0.11343 |    0.041401 |    0.011574 |    0.011587 |          216 |    euclidean |        false
|   24 | Accept |     0.05787 |    0.042783 |    0.011574 |    0.011587 |          216 |    euclidean |         true
|   25 | Accept |     0.11343 |    0.044176 |    0.011574 |    0.011587 |          216 |    minkowski |        false
|   26 | Accept |     0.05787 |    0.042419 |    0.011574 |    0.011587 |          216 |    minkowski |         true
|   27 | Accept |     0.17593 |      0.0414 |    0.011574 |    0.011587 |          215 |  mahalanobis |        false
|   28 | Accept |      0.0625 |    0.040399 |    0.011574 |    0.011587 |          213 |   seuclidean |        false
|   29 | Accept |     0.12731 |    0.041974 |    0.011574 |    0.011586 |          211 |    chebychev |         true
|   30 | Accept |    0.055556 |    0.036466 |    0.011574 |    0.011586 |            1 |       cosine |         true

__________________________________________________________
최적화가 완료되었습니다.
MaxObjectiveEvaluations 30회에 도달했습니다.
총 함수 실행 횟수: 30
총 경과 시간: 8.0028초
총 목적 함수 실행 시간: 1.2832

최선의 관측된 실현가능점:
    NumNeighbors    Distance     Standardize
    ____________    _________    ___________

         2          chebychev       true

관측된 목적 함수 값 = 0.011574
추정된 목적 함수 값 = 0.011716
함수 실행 시간 = 0.045459

최선의 추정된 실현가능점(모델에 따라 다름):
    NumNeighbors    Distance     Standardize
    ____________    _________    ___________

         2          cityblock       true

추정된 목적 함수 값 = 0.011586
추정된 함수 실행 시간 = 0.040004
```




```
Mdl = 
  ClassificationKNN
                         ResponseName: 'Y'
                CategoricalPredictors: []
                           ClassNames: {'inner'  'normal'  'outer'}
                       ScoreTransform: 'none'
                      NumObservations: 432
    HyperparameterOptimizationResults: [1×1 BayesianOptimization]
                             Distance: 'cityblock'
                         NumNeighbors: 2


  Properties, Methods
```

Compare performance of KNN with other classification methods