Improving Predictive Models

Methods for improving predictive model.

Import and Prepare Data

Load the heart disease data and extract numerical predictors.

```
heartData = readtable("heartDiseaseData.csv");
heartData = convertvars(heartData,12:22,"categorical");
heartDataNum = heartData(:,[1:11 22]);
```

Partition into training and test sets.

```
rng(1234)
pt = cvpartition(heartData.HeartDisease,"Holdout",0.2);
hdTrainNum = heartDataNum(training(pt),:);
hdTestNum = heartDataNum(~training(pt),:);
hdTrain = heartData(training(pt),:);
hdTest = heartData(~training(pt),:);
```

Load the multiclass heart disease data, extract numerical predictors, and partition into training/test sets.

```
heartDataMulti = readtable("heartDiseaseDataMulticlass.csv");
heartDataMulti = convertvars(heartDataMulti,12:22,"categorical");
hdMTrain = heartDataMulti(training(pt),:);
hdMTest = heartDataMulti(~training(pt),:);
hdMTrainNum = heartDataMulti(training(pt),[1:11 22]);
hdMTestNum = heartDataMulti(~training(pt),[1:11 22]);
```

Cross Validation

Create 7-fold cross-validation partition.

```
part = cvpartition(heartDataNum.HeartDisease,"KFold",7);
```

Train kNN and DA models.

```
% kNN
mdlKnn = fitcknn(heartDataNum,"HeartDisease","NumNeighbors",5,"CVPartition",part);
lossKnn = kfoldLoss(mdlKnn);

% Discriminant analysis
mdlDa = fitcdiscr(heartDataNum,"HeartDisease","CVPartition",part);
lossDa = kfoldLoss(mdlDa);
```

Display the results.

```
KFoldLoss = [lossKnn;lossDa];
results = table(KFoldLoss);
results.Properties.RowNames = ["kNN" "Discriminant Analysis"];
disp("Seven-fold cross-validated results")
```

```
Seven-fold cross-validated results
```

```
disp(results)
```

```
                              KFoldLoss
                              _________

    kNN                         0.2904 
    Discriminant Analysis      0.26932 
```

Hyperparameter Optimization

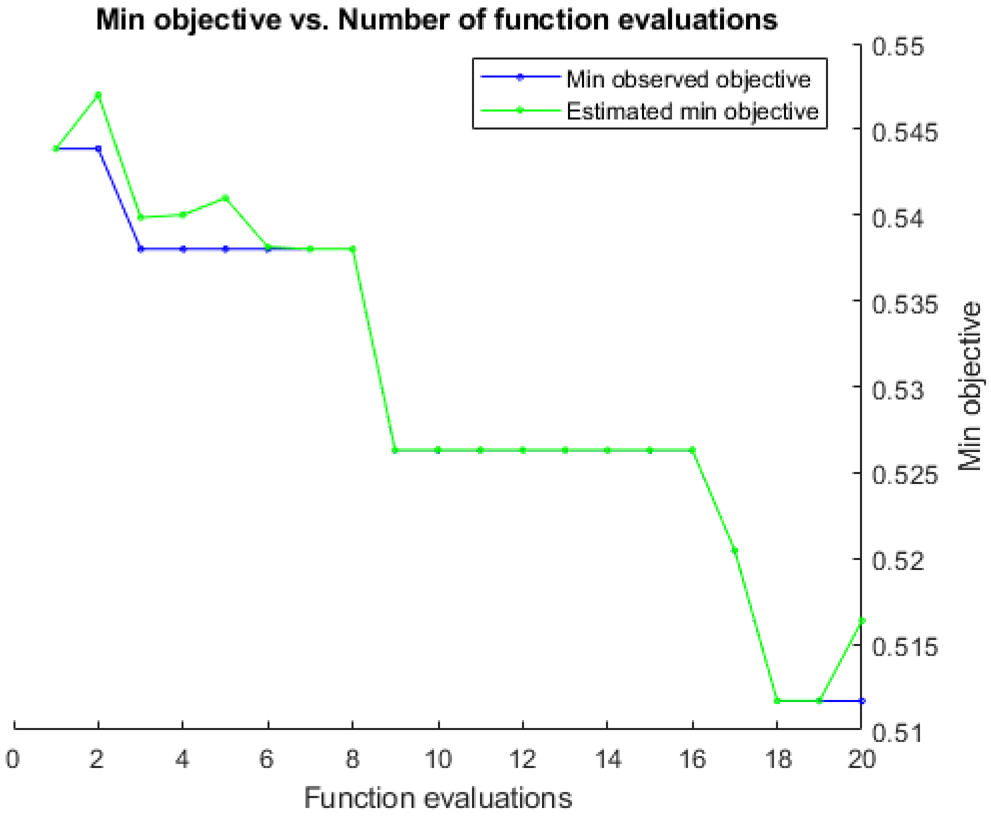
Optimize all hyperparameters kNN classifier for multiclass numeric heart disease data.

Set optimization options.

```
cvpt = cvpartition(hdMTrainNum.HeartDisease,"KFold",10);
opt = struct("CVPartition",cvpt,"MaxObjectiveEvaluations",20);
```

Optimize hyperparameters.

```
mdl = fitcknn(hdMTrainNum,"HeartDisease",...
    "OptimizeHyperparameters","all","HyperparameterOptimizationOptions",opt);
```



```
|=====================================================================================================================================================
| Iter | Eval   | Objective    | Objective    | BestSoFar    | BestSoFar    | NumNeighbors |     Distance | DistanceWeig
|      | result |              | runtime      | (observed)   | (estim.)     |              |              | ht
|=====================================================================================================================================================
|    1 | Best   |      0.54386 |      0.24472 |      0.54386 |      0.54386 |           29 |      hamming | squaredinver
|    2 | Accept |      0.62281 |      0.31742 |      0.54386 |        0.547 |            3 |       cosine |        equal
|    3 | Best   |      0.53801 |      0.30845 |      0.53801 |      0.53985 |          119 |    euclidean | squaredinver
|    4 | Accept |     0.59357 |     0.28116 |     0.53801 |     0.54001 |            3 |    minkowski |      inverse
|    5 | Accept |     0.54386 |     0.15329 |     0.53801 |     0.54099 |           20 |    euclidean | squaredinver
|    6 | Accept |     0.54094 |     0.12529 |     0.53801 |     0.53813 |          170 |    euclidean | squaredinver
|    7 | Accept |     0.64035 |     0.16231 |     0.53801 |     0.53802 |            1 |    euclidean | squaredinver
|    8 | Accept |     0.60526 |     0.12956 |     0.53801 |     0.53802 |            3 |      hamming | squaredinver
|    9 | Best   |     0.52632 |     0.12928 |     0.52632 |     0.52633 |           54 |    euclidean | squaredinver
|   10 | Accept |     0.55263 |     0.12288 |     0.52632 |     0.52633 |          135 |      hamming | squaredinver
|   11 | Accept |     0.52632 |      0.1235 |     0.52632 |     0.52632 |           47 |    euclidean | squaredinver
|   12 | Best   |     0.52632 |     0.12841 |     0.52632 |     0.52632 |           70 |    euclidean | squaredinver
|   13 | Accept |     0.54678 |     0.14566 |     0.52632 |     0.52632 |           71 |      hamming | squaredinver
|   14 | Accept |     0.57018 |     0.13065 |     0.52632 |     0.52632 |            8 |      hamming | squaredinver
|   15 | Accept |     0.54678 |     0.26219 |     0.52632 |     0.52632 |           61 |       cosine | squaredinver
|   16 | Accept |     0.57895 |     0.13753 |     0.52632 |     0.52632 |            7 |       cosine | squaredinver
|   17 | Best   |     0.52047 |     0.12116 |     0.52047 |     0.52047 |          168 |       cosine | squaredinver
|   18 | Best   |      0.5117 |     0.12017 |      0.5117 |      0.5117 |           61 |       cosine | squaredinver
|   19 | Accept |     0.55263 |     0.12815 |      0.5117 |     0.51171 |           19 |       cosine | squaredinver
|   20 | Accept |     0.52047 |     0.11783 |      0.5117 |     0.51637 |           87 |       cosine | squaredinver

__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 20 reached.
Total function evaluations: 20
Total elapsed time: 42.5169 seconds.
Total objective function evaluation time: 3.3896

Best observed feasible point:
    NumNeighbors    Distance    DistanceWeight    Exponent    Standardize
    ____________    ________    ______________    ________    ___________

         61          cosine     squaredinverse      NaN          false   

Observed objective function value = 0.5117
Estimated objective function value = 0.51637
Function evaluation time = 0.12017

Best estimated feasible point (according to models):
    NumNeighbors    Distance    DistanceWeight    Exponent    Standardize
    ____________    ________    ______________    ________    ___________

         87          cosine     squaredinverse      NaN          false   

Estimated objective function value = 0.51637
Estimated function evaluation time = 0.11783
```

Calculate loss.

```
trainLossOpt = resubLoss(mdl)
```

```
trainLossOpt = 0
```

```
testLossOpt = loss(mdl,hdMTestNum)
```

```
testLossOpt = 0.5436
```

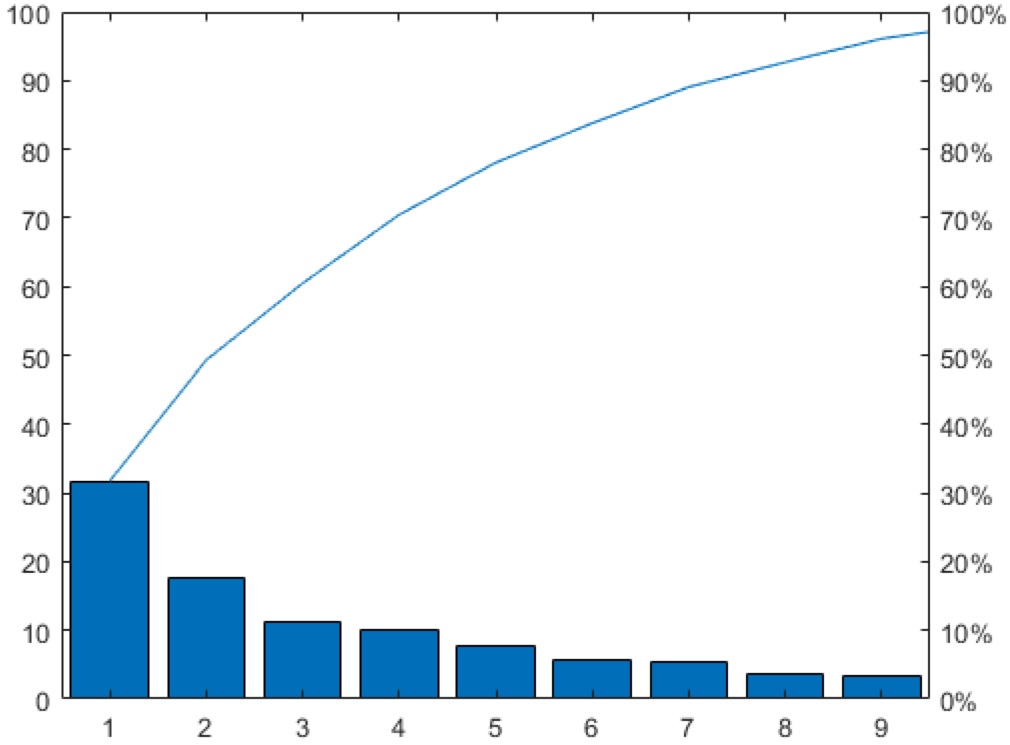
PCA

Extract response and numerical data.

```
HD = heartDataNum.HeartDisease;
numData = heartDataNum{:,1:end-1};
```

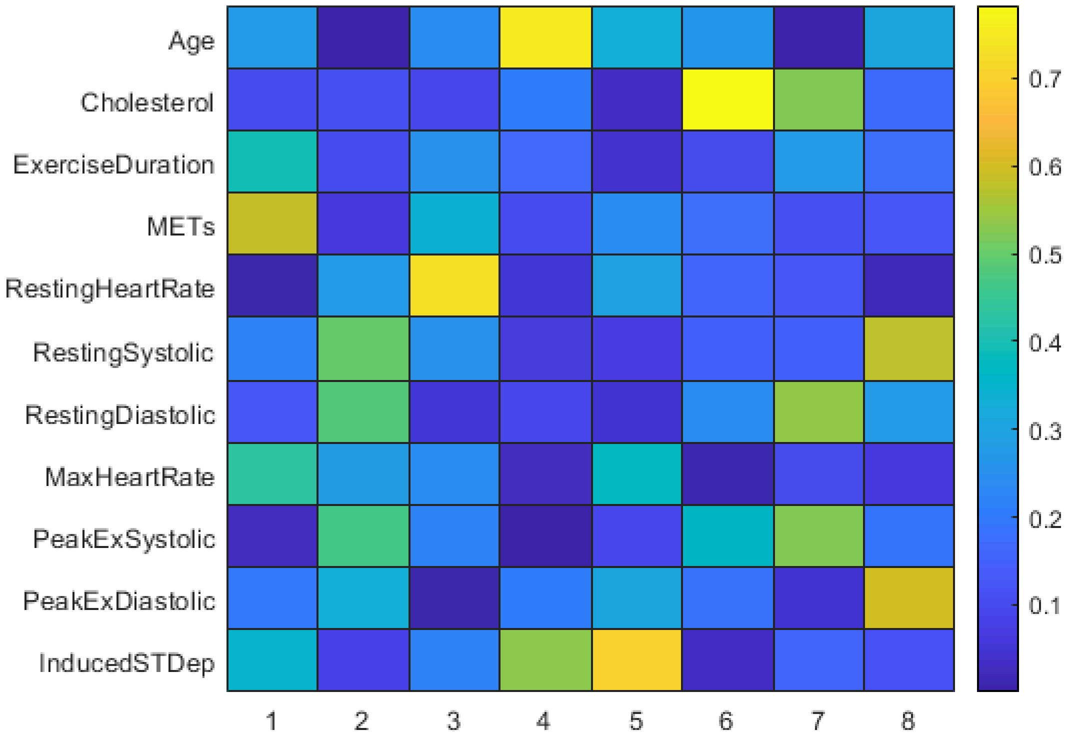
Perform PCA.

```
[pcs,scrs,~,~,pexp] = pca(numData);
pareto(pexp)
```



Visualize principal components.

```
varNames = heartDataNum.Properties.VariableNames(1:end-1);
heatmap(abs(pcs(:,1:8)),"YDisplayLabels",varNames,"Colormap",parula);
```



Fit Naive Bayes model to original data.

```
mdlOrig = fitcnb(numData,HD,"DistributionNames","kernel","KFold",10);
lossOrig = kfoldLoss(mdlOrig)
```

```
lossOrig = 0.2927
```

Fit Naive Bayes model to reduced data.

```
numDataPart = scrs(:,1:8);
```

```
numDataPart = 427×8
    0.1105    0.1089   -0.4042    0.0212    0.2028   -0.0537   -0.0350    0.0884
   -0.0809    0.0750   -0.4009    0.1968    0.0474    0.1480    0.0543    0.0716
   -0.0765   -0.2391   -0.0837    0.0736    0.2836   -0.0581    0.0226    0.1745
    0.5920    0.1121   -0.1163   -0.4166    0.4789    0.0913   -0.2056   -0.0622
    0.2090   -0.0348    0.0836   -0.2165    0.0487    0.0313    0.1505   -0.1157
    0.5692   -0.0778   -0.1129    0.1249    0.2041    0.0019   -0.0195    0.0574
   -0.2675    0.2025   -0.0107   -0.0920    0.3983    0.0359    0.0390    0.1059
    0.1965    0.0043    0.1503    0.1882    0.0553    0.1812   -0.1817    0.1150
   -0.0323   -0.0352    0.0026    0.1261    0.1574    0.1868    0.3468    0.1598
   -0.2570    0.3158    0.0590   -0.2898    0.1685   -0.1400    0.1123    0.2013
       ⋮
```

```
mdlPart = fitcnb(numDataPart,HD,"DistributionNames","kernel","KFold",10);
lossPart = kfoldLoss(mdlPart)
```

```
lossPart = 0.2740
```

Sequential Feature Selection

Create 10-fold partition of mixed heart disease data set.

```
HD = heartData.HeartDisease;
rng(1234)
cvpt = cvpartition(HD,"KFold",10);
```

Make dummy variables for categorical predictors.

```
T = splitvars(convertvars(heartData(:,1:end-1),vartype("categorical"),@dummyvar));
X = T{:,:};
Xnames = T.Properties.VariableNames;
```

Fit Naive Bayes model to the full data.

```
dists = [repmat("kernel",1,11),repmat("mvmn",1,10)];
mdlFull = fitcnb(heartData,"HeartDisease","DistributionNames",dists,"CVPartition",cvpt);
```

Perform sequential feature selection.

```
rng(1234)
fmodel = @(X,y) fitcnb(X,y,"DistributionNames","kernel");
ferror = @(Xtrain,ytrain,Xtest,ytest) nnz(predict(fmodel(Xtrain,ytrain),Xtest) ~= ytest);
toKeep = sequentialfs(ferror,X,HD,"cv",cvpt,"options",statset("Display","iter"));
```

```
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 14, criterion value 0.234192
Step 2, added column 8, criterion value 0.222482
Step 3, added column 11, criterion value 0.213115
Step 4, added column 17, criterion value 0.208431
Step 5, added column 25, criterion value 0.203747
Step 6, added column 2, criterion value 0.199063
Step 7, added column 12, criterion value 0.194379
Step 8, added column 31, criterion value 0.185012
Step 9, added column 1, criterion value 0.18267
Final columns included:  1 2 8 11 12 14 17 25 31
```

Which variables are in the final model?

```
Xnames(toKeep)
```

```
ans = 1×9 cell
'Age'        'Cholesterol'    'MaxHeartRate'    'InducedSTDep'    'Sex_1'      '...
```

Fit NB model to the selected variables.

```
mdlPart = fitcnb(X(:,toKeep),HD,"DistributionNames","kernel","CVPartition",cvpt);
```

Display loss values.

```
lossFull = kfoldLoss(mdlFull)
```

```
lossFull = 0.2131
```

```
lossPart = kfoldLoss(mdlPart)
```

```
lossPart = 0.1827
```

Ensemble Learning

```
rng(1234)
cvpt = cvpartition(heartData.HeartDisease,"KFold",10);
```

Train a single decision tree.

```
mdl = fitctree(heartData,"HeartDisease","CVPartition",cvpt);
singleTLoss = kfoldLoss(mdl)
```

```
singleTLoss = 0.2787
```

Build an ensemble of bagged trees.

```
rng(1234)
mdl = fitcensemble(heartData,"HeartDisease","Method","Bag",...
    "NumLearningCycles",50,"CVPartition",cvpt);
bagTLoss = kfoldLoss(mdl)
```

```
bagTLoss = 0.2319
```

Try a different ensemble method.

```
rng(1234)
mdl = fitcensemble(heartData,"HeartDisease","Method","RUSBoost",...
    "Learners","tree","NumLearningCycles",50,"CVPartition",cvpt);
RUSBoostTLoss = kfoldLoss(mdl)
```

```
RUSBoostTLoss = 0.2225
```

Compare a single kNN model to an ensemble of kNN learners.

```
mdl = fitcknn(heartDataNum,"HeartDisease",...
    "NumNeighbors",5,"DistanceWeight","squaredinverse","CVPartition",cvpt);
singleKLoss = kfoldLoss(mdl)
```

```
singleKLoss = 0.2974
```

```

knntemp = templateKNN("NumNeighbors",5,"DistanceWeight","squaredinverse");
mdl = fitcensemble(heartDataNum,"HeartDisease","Method","Subspace",...
    "Learners",knntemp,"NumLearningCycles",50,"CVPartition",cvpt);
subspaceKLoss = kfoldLoss(mdl)
```

```
subspaceKLoss = 0.3724
```