import the data

```
importdata = SemanticImport["C:\\Users\\joshf\\Google Drive\\Rich
     Internship 2020\\XGBoost\\xgboost_covid\\data\\Data.csv"];
data1 = Values[Normal[importdata]][[All, 2 ;; 78]];
labels = Import["C:\\Users\\joshf\\Google Drive\\Rich Internship
      2020\\XGBoost\\xgboost_covid\\data\\Data.csv"][[1]] // TableForm
```

Out[15]//TableForm=

PATIENT_ID
age
age group
gender
Red blood cell count
White blood cell count
Hematocrit
globulin
Mean hemoglobin concentration
Monocytes (#)
Lactate dehydrogenase
Urea
Lymphocyte (#)
Gamma-glutamyl transpeptidase
Eosinophil (#)
Creatinine
Neutrophils (#)
Direct bilirubin
Bicarbonate
Uric acid
Aspartate aminotransferase
Average RBC volume
Total cholesterol
Lymphocytes (%)
Platelet count
Eosinophils (%)
albumin
Basophil (%)
eGFR (based on CKD-EPI equation)
Total bilirubin
Alkaline phosphatase
Alanine aminotransferase
Hemoglobin
Mean hemoglobin content
Basophil (#)
Neutrophils (%)
Total protein
Monocytes (%)
Indirect bilirubin
chlorine
calcium
Potassium
sodium
Hypersensitive C-reactive protein
Corrected calcium
International standardized ratio
Prothrombin time
Prothrombin activity

```
glucose
RBC distribution width SD
RBC distribution width CV
Average PLT volume
Platelet compression
Platelet ratio
PLT distribution width
D-D dimer quantification
Procalcitonin
Thrombin time
Activated partial thromboplastin time
Fibrinogen
Erythrocyte sedimentation rate
High-sensitivity cardiac troponin I
Quantification of hepatitis C antibody
Treponema pallidum antibody quantification
Quantification of hepatitis B surface antigen
HIV antibody quantification
Amino-terminal brain natriuretic peptide precursor (NT-proBNP)
PH
Interleukin 6
Tumor necrosis factor alpha
Interleukin 8
Interleukin 1-Beta
Interleukin 2 receptor
Interleukin 10
Ferritin
Antithrombin
Fibrin (pro) degradation products
Outcome
```

Classification

In[•]:=
```
Prepend[Range[74] + 2, 2]
```

Out[•]=
```
{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,
 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76}
```

Survival probability by age and gender

randomize data for training/test split

In[•]:=
```
SeedRandom[123];
rs = RandomSample[data1[[All, {1, 2, 76}]], 176];
```

Split data into 70% training and 30% test

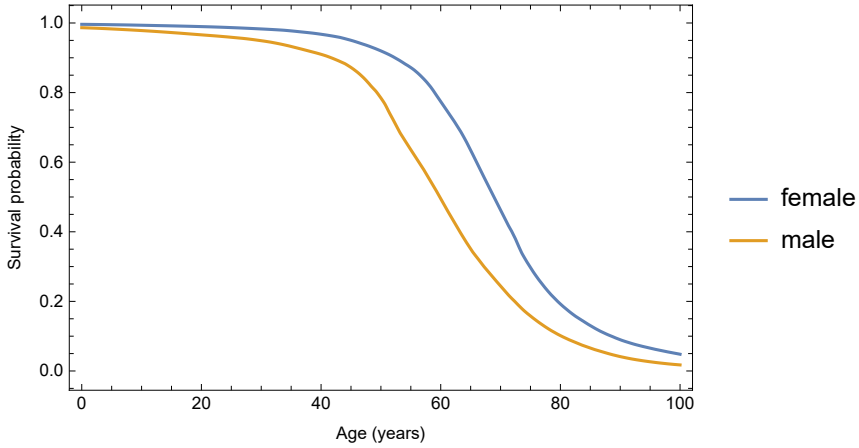
In[•]:=
```
trainingset = rs[[1 ;; 123]];
testset = rs[[124 ;; 176]];
```

```
In[ ]:= c = Classify[data1[[All, {1, 3, 77}]] → 3,
          Method → "NeuralNetwork", PerformanceGoal → "Quality"]
```

```
Out[ ]= ClassifierFunction[ Input type: {Numerical, Nominal}
                            Classes: died, survived ]
```

```
In[ ]:= p[age_, gender_] := c[{age, gender}, {"Probability", "survived"}];
        Plot[{p[x, "female"], p[x, "male"]}, {x, 0, 100}, PlotLegends → {"female", "male"},
         Frame → True, FrameLabel → {"Age (years)", "Survival probability"}, Exclusions → None]
```

Out[ ]=



Accuracy and Confusion Matrix

```
In[ ]:= cm = ClassifierMeasurements[c, testset → 3]
```

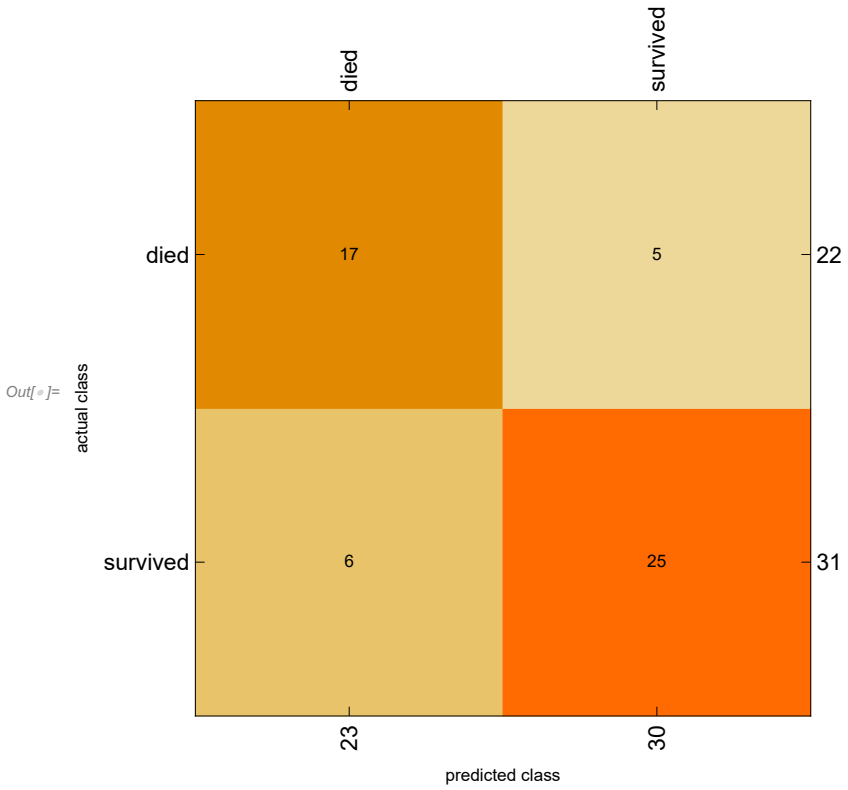
```
Out[ ]= ClassifierMeasurementsObject[ Classifier: NeuralNetwork
                                      Number of test examples: 53
                                      Number of classes: 2
                                      Accuracy: 0.79 ± 0.06 ]
```

```
In[ ]:= cm["Accuracy"]
```

```
Out[ ]= 0.792453
```

In[ ]:=
```
cm["ConfusionMatrixPlot"]
```

Out[ ]=



Neural Network approach

training/test split of data

randomize data for training/test split

In[ ]:=
```
SeedRandom[123];
genrs = Table[RandomSample[data1[[All, {i, 3, 77}]], 176], {i, 4, 76, 1}];
```

Split data into 75% training and 25% test

In[ ]:=
```
trainingset1 = Table[genrs[[i]][[1 ;; 132]], {i, 1, 73}];
testset1 = Table[genrs[[i]][[133 ;; 176]], {i, 1, 73}];
```

classification by gender

In[ ]:=
```
ProgressIndicator[Dynamic[i], {4, 76}]
```

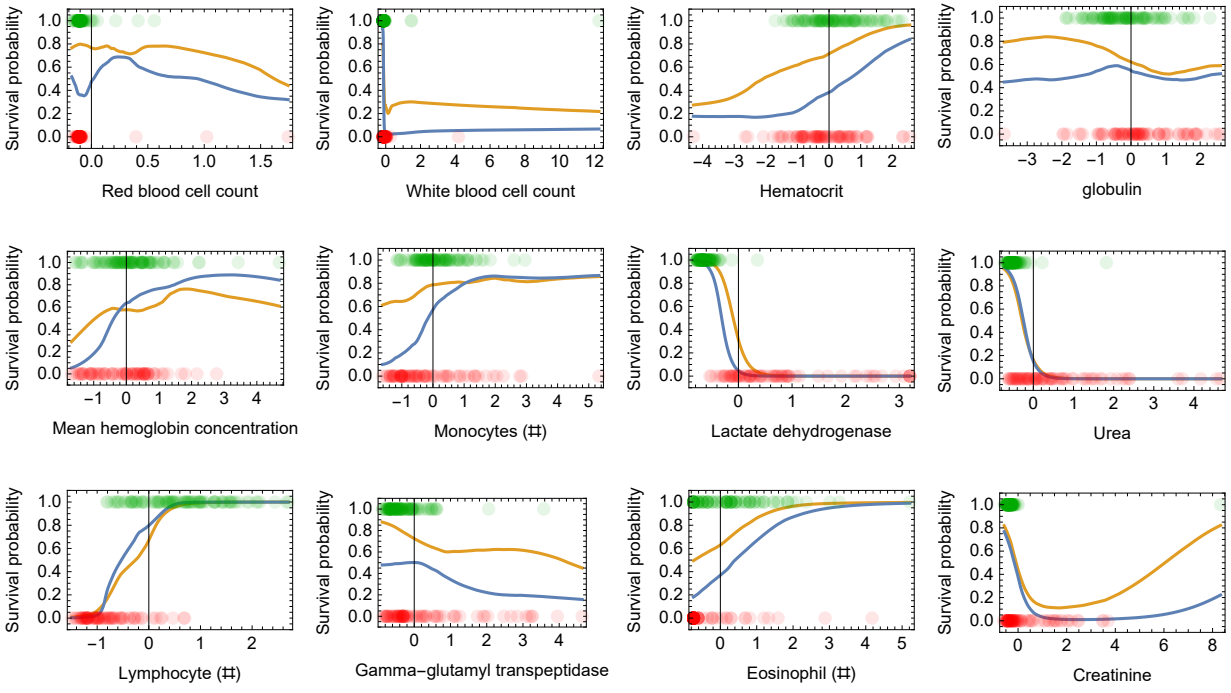
Out[ ]=

In[ ]:=
```
c1 = Table[Classify[trainingset1[[i]] → 3,
    Method → "NeuralNetwork", PerformanceGoal → "Quality"], {i, 1, 73}];
```

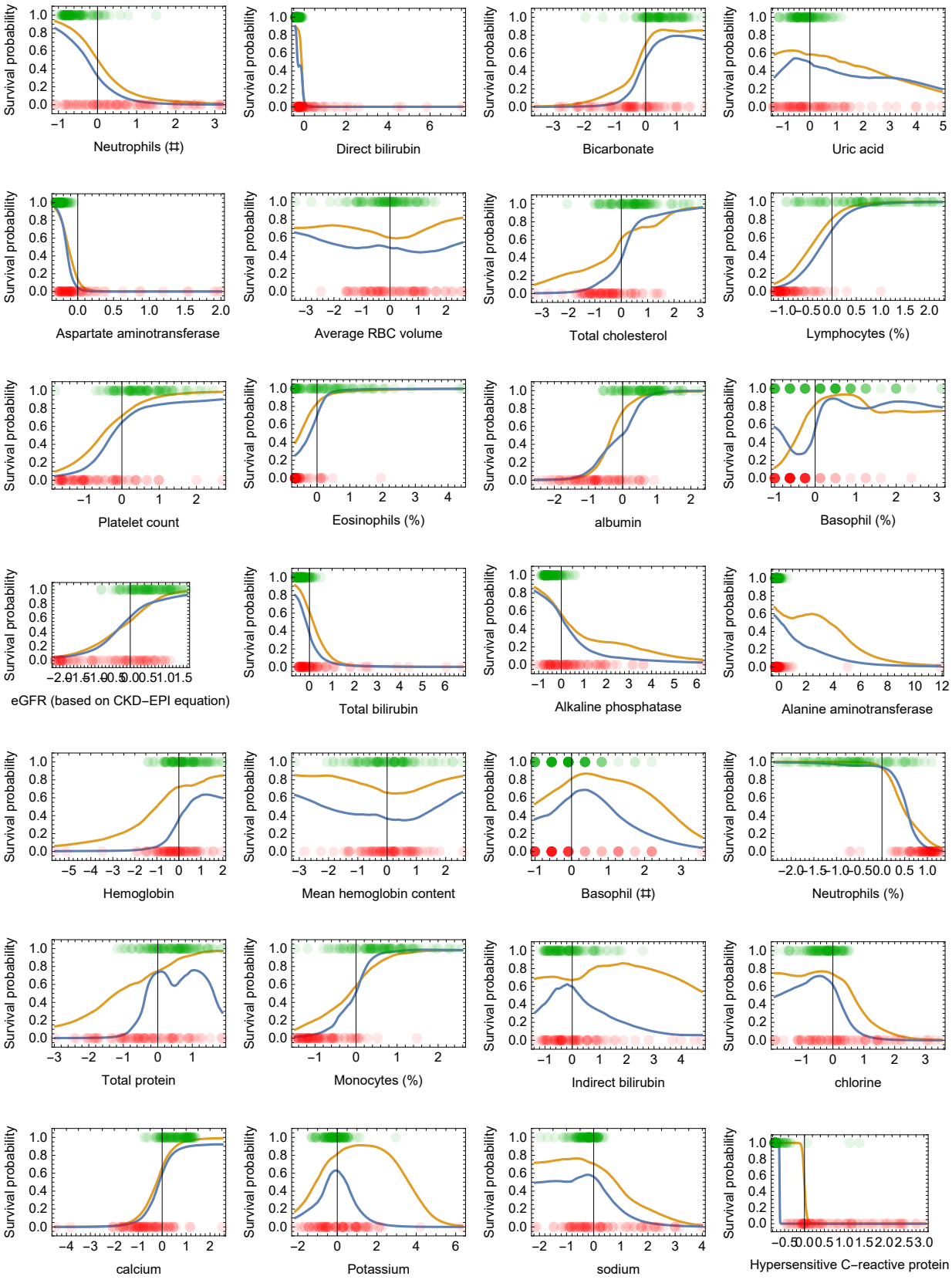
In[ ]:=
```
ProgressIndicator[Dynamic[k], {1, 73}]
```

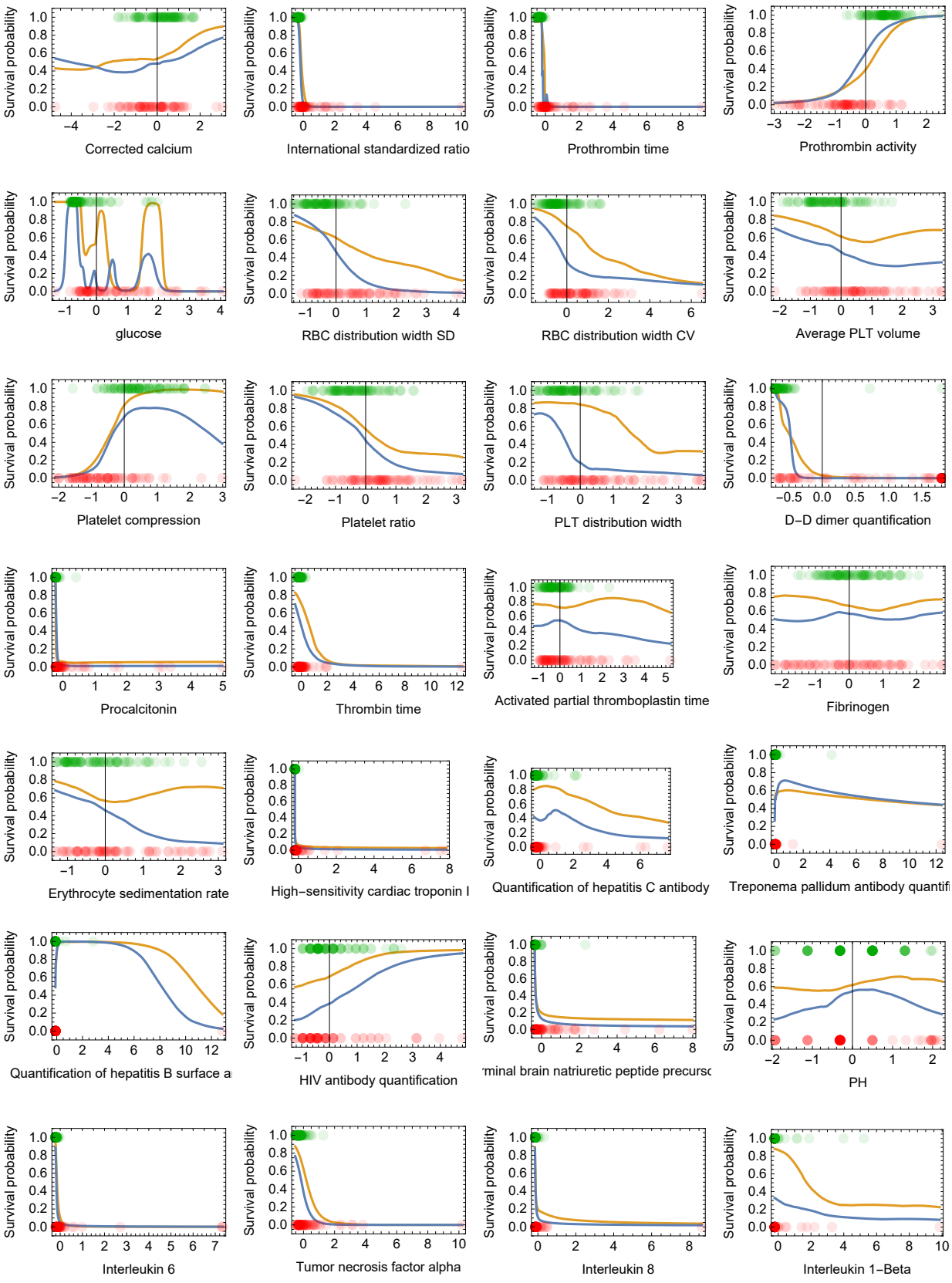
Out[ ]=

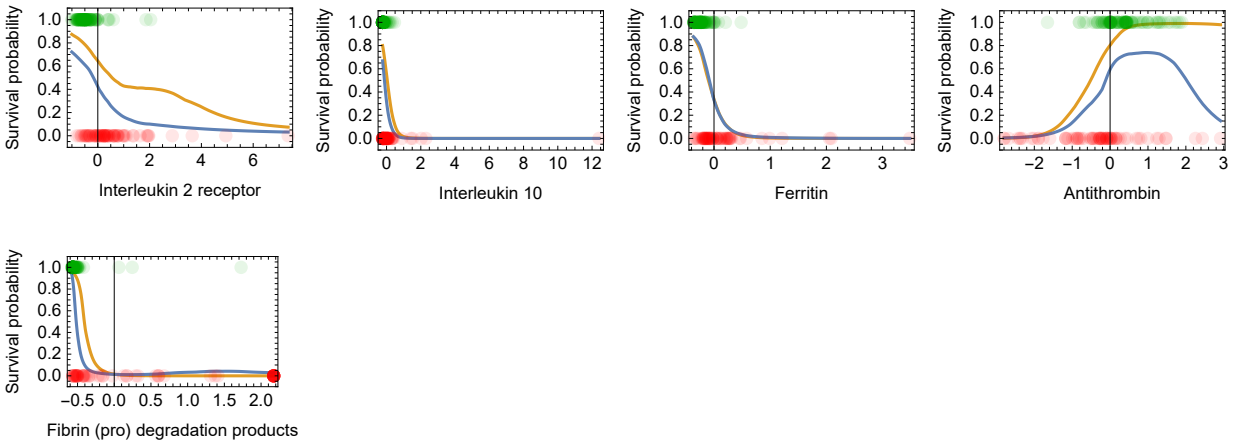
In[ ]:=
```
Multicolumn[Table[Show[Plot[{c1[[k]][{x, "female"}], {"Probability", "survived"}],
     c1[[k]][{x, "male"}], {"Probability", "survived"}]},
    {x, Min[trainingset1[[k]][[All, 1]]], Max[trainingset1[[k]][[All, 1]]]}, Frame → True,
    FrameLabel → {labels[[4 + k]], "Survival probability"}, Exclusions → None,
    PlotRange → {-0.1, 1.1}, PlotStyle → {, }, Ticks → {Automatic, Range[0, 1, 0.2]}],
   ListPlot[{ArrayReshape[Riffle[data1[[1 ;; 99, 3 + k]], ConstantArray[1,
       Length[data1[[1 ;; 99, 3 + k]]]]], {Length[data1[[1 ;; 99, 3 + k]], 2}], ArrayReshape[
     Riffle[data1[[100 ;; 176, 3 + k]], ConstantArray[0, Length[data1[[100 ;; 176, 3 + k]]]]],
     {Length[data1[[100 ;; 176, 3 + k]], 2}]},
    PlotStyle → {Directive[Darker[Green], PointSize[Large], Opacity[0.1]],
      Directive[Red, PointSize[Large], Opacity[0.1]]}]],
  {k, 1, 73, 1}], 4, Appearance → "Horizontal"]
SwatchLegend[{, , , }, {"female", "male", "survived", "died"},
 LegendFunction → "Frame"]
```



Out[•]=


Survival probability
Neutrophils (#)
Direct bilirubin
Bicarbonate
Uric acid
Aspartate aminotransferase
Average RBC volume
Total cholesterol
Lymphocytes (%)
Platelet count
Eosinophils (%)
albumin
Basophil (%)
eGFR (based on CKD−EPI equation)
Total bilirubin
Alkaline phosphatase
Alanine aminotransferase
Hemoglobin
Mean hemoglobin content
Basophil (#)
Neutrophils (%)
Total protein
Monocytes (%)
Indirect bilirubin
chlorine
calcium
Potassium
sodium
Hypersensitive C−reactive protein


Survival probability
Corrected calcium
International standardized ratio
Prothrombin time
Prothrombin activity
glucose
RBC distribution width SD
RBC distribution width CV
Average PLT volume
Platelet compression
Platelet ratio
PLT distribution width
D–D dimer quantification
Procalcitonin
Thrombin time
Activated partial thromboplastin time
Fibrinogen
Erythrocyte sedimentation rate
High–sensitivity cardiac troponin I
Quantification of hepatitis C antibody
Treponema pallidum antibody quantifi
Quantification of hepatitis B surface a
HIV antibody quantification
minal brain natriuretic peptide precurso
PH
Interleukin 6
Tumor necrosis factor alpha
Interleukin 8
Interleukin 1–Beta

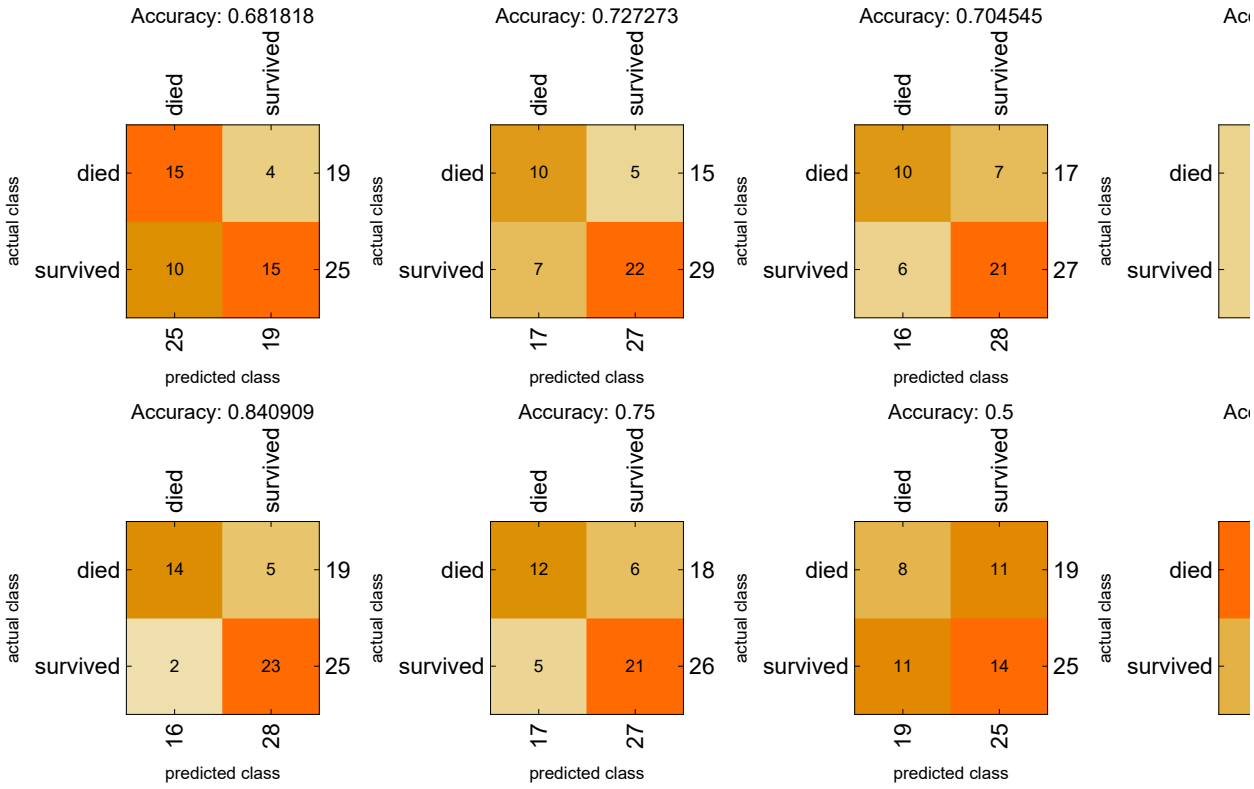


Out[ ]=

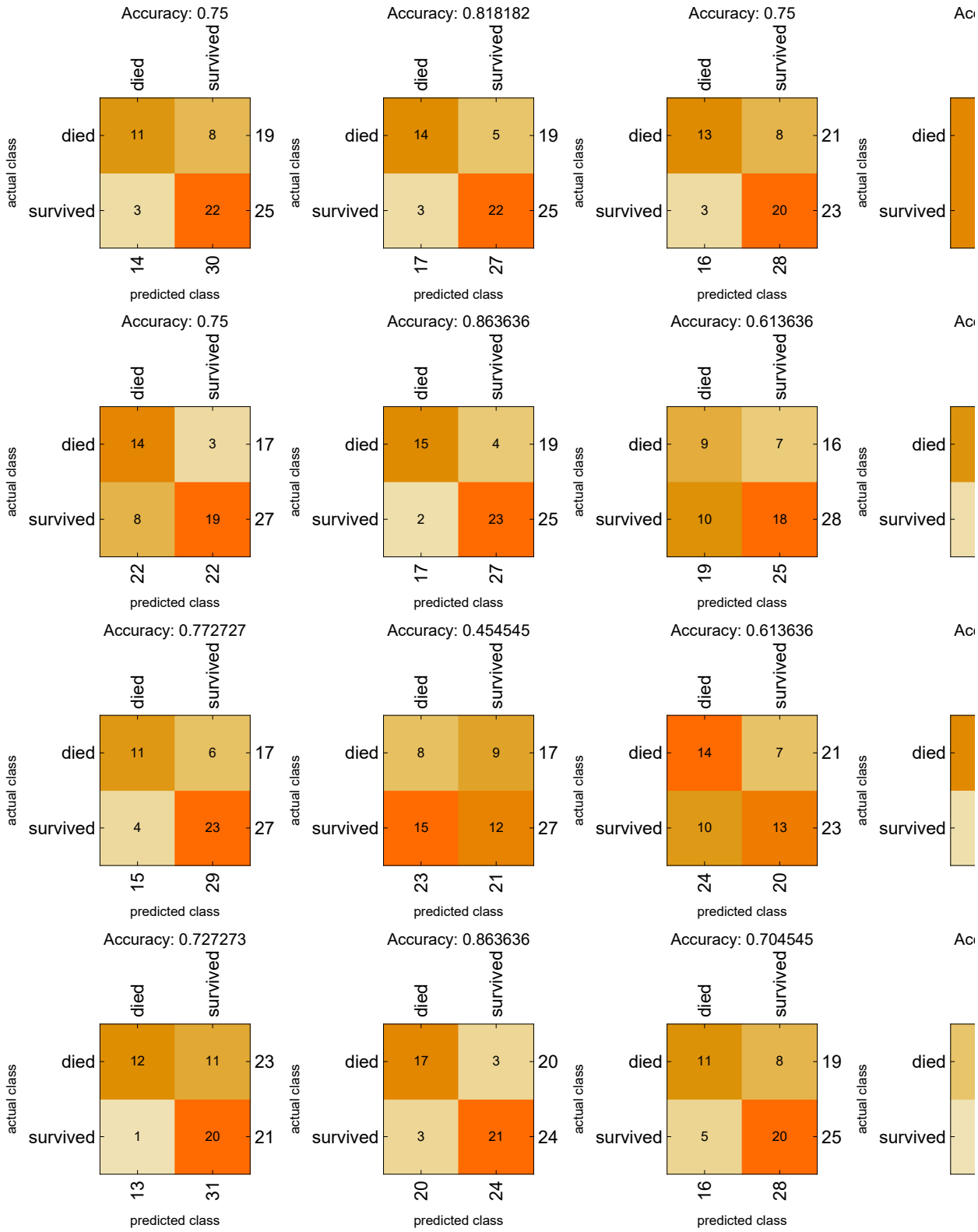
Confusion matrix and accuracy

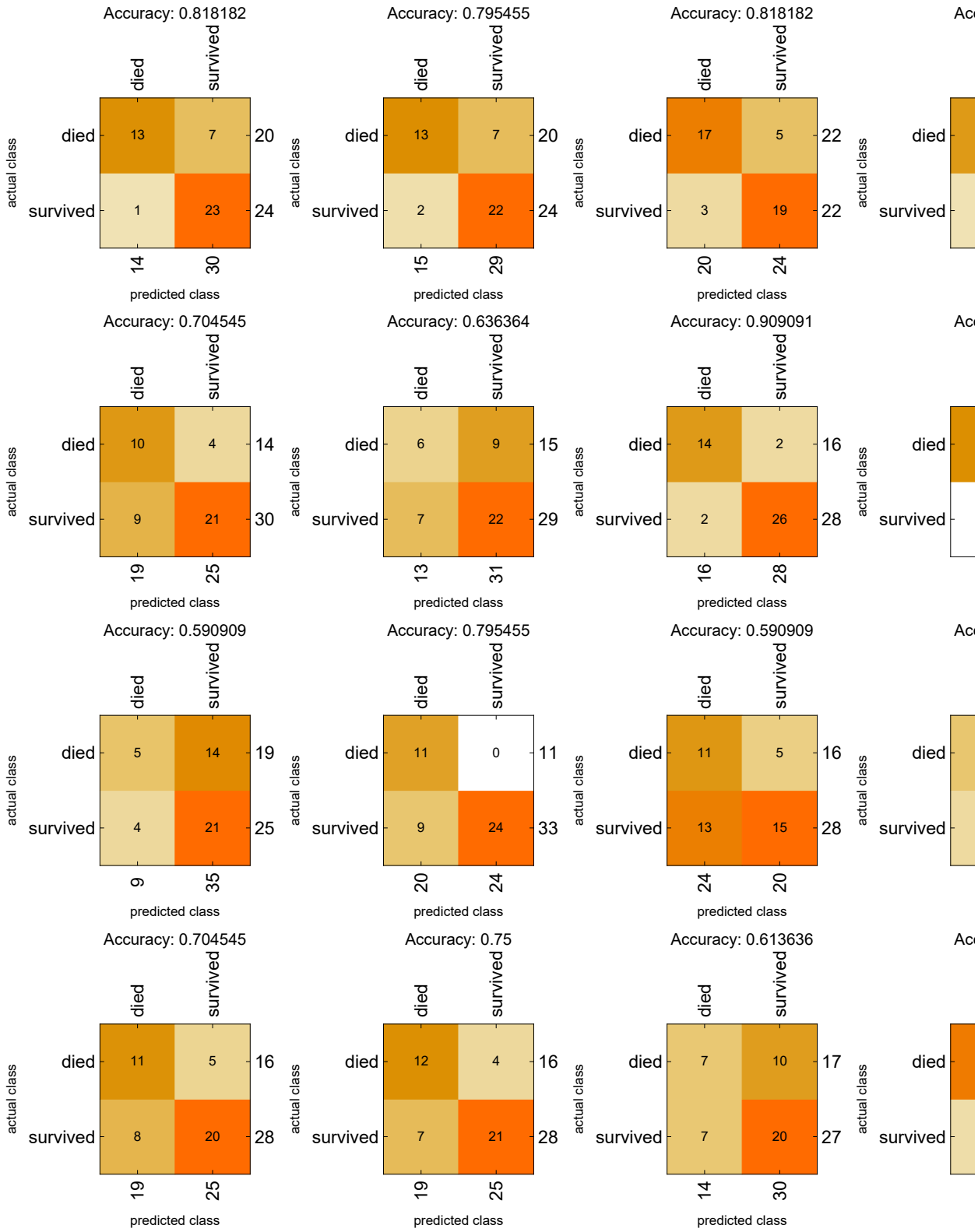
```
cm1 = Table[ClassifierMeasurements[c1[[i]], testset1[[i]] → 3], {i, 1, 73}];
```

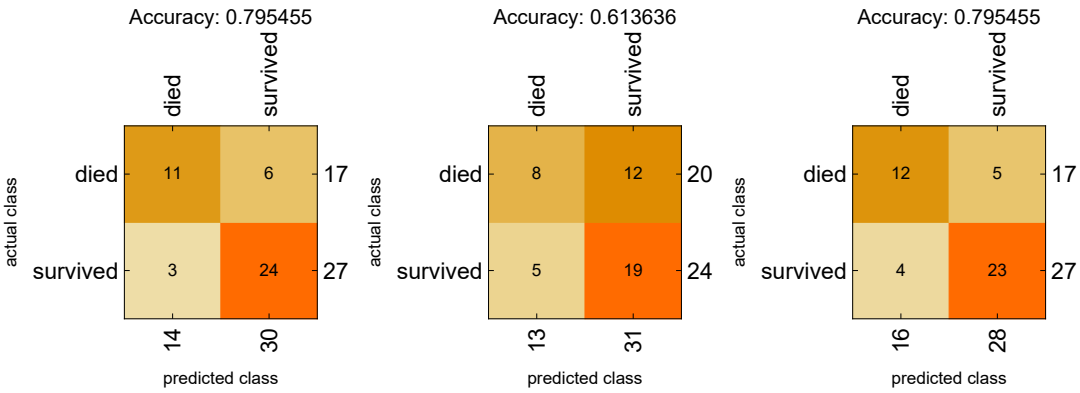
In[ ]:=
```
Multicolumn[Table[Show[cm1[[i]]["ConfusionMatrixPlot"], ImageSize → Small,
    PlotLabel → StringJoin["Accuracy: ", ToString[cm1[[i]]["Accuracy"]]]],
  {i, 1, 73}], 7, Appearance → "Horizontal"]
```



Out[ ]=


Accuracy: 0.75
died survived
died 11 8 19
survived 3 22 25
14 30
actual class
predicted class
Accuracy: 0.818182
died survived
died 14 5 19
survived 3 22 25
17 27
actual class
predicted class
Accuracy: 0.75
died survived
died 13 8 21
survived 3 20 23
16 28
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.75
died survived
died 14 3 17
survived 8 19 27
22 22
actual class
predicted class
Accuracy: 0.863636
died survived
died 15 4 19
survived 2 23 25
17 27
actual class
predicted class
Accuracy: 0.613636
died survived
died 9 7 16
survived 10 18 28
19 25
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.772727
died survived
died 11 6 17
survived 4 23 27
15 29
actual class
predicted class
Accuracy: 0.454545
died survived
died 8 9 17
survived 15 12 27
23 21
actual class
predicted class
Accuracy: 0.613636
died survived
died 14 7 21
survived 10 13 23
24 20
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.727273
died survived
died 12 11 23
survived 1 20 21
13 31
actual class
predicted class
Accuracy: 0.863636
died survived
died 17 3 20
survived 3 21 24
20 24
actual class
predicted class
Accuracy: 0.704545
died survived
died 11 8 19
survived 5 20 25
16 28
actual class
predicted class
Ac
died
survived
actual class


Accuracy: 0.818182
died survived
died 13 7 20
survived 1 23 24
14 30
actual class
predicted class
Accuracy: 0.795455
died survived
died 13 7 20
survived 2 22 24
15 29
actual class
predicted class
Accuracy: 0.818182
died survived
died 17 5 22
survived 3 19 22
20 24
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.704545
died survived
died 10 4 14
survived 9 21 30
19 25
actual class
predicted class
Accuracy: 0.636364
died survived
died 6 9 15
survived 7 22 29
13 31
actual class
predicted class
Accuracy: 0.909091
died survived
died 14 2 16
survived 2 26 28
16 28
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.590909
died survived
died 5 14 19
survived 4 21 25
9 35
actual class
predicted class
Accuracy: 0.795455
died survived
died 11 0 11
survived 9 24 33
20 24
actual class
predicted class
Accuracy: 0.590909
died survived
died 11 5 16
survived 13 15 28
24 20
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.704545
died survived
died 11 5 16
survived 8 20 28
19 25
actual class
predicted class
Accuracy: 0.75
died survived
died 12 4 16
survived 7 21 28
19 25
actual class
predicted class
Accuracy: 0.613636
died survived
died 7 10 17
survived 7 20 27
14 30
actual class
predicted class
Ac
died
survived
actual class



classification by age group

training/test split of data

randomize data for training/test split

In[ ]:=
```
SeedRandom[123];
agers = Table[RandomSample[data1[[All, {i, 2, 77}]], 176], {i, 4, 76, 1}];
```

Split data into 75% training and 25% test

In[ ]:=
```
trainingset2 = Table[agers[[i]][[1 ;; 132]], {i, 1, 73}];
testset2 = Table[agers[[i]][[133 ;; 176]], {i, 1, 73}];
```

In[ ]:=
```
ProgressIndicator[Dynamic[i], {4, 76}]
```

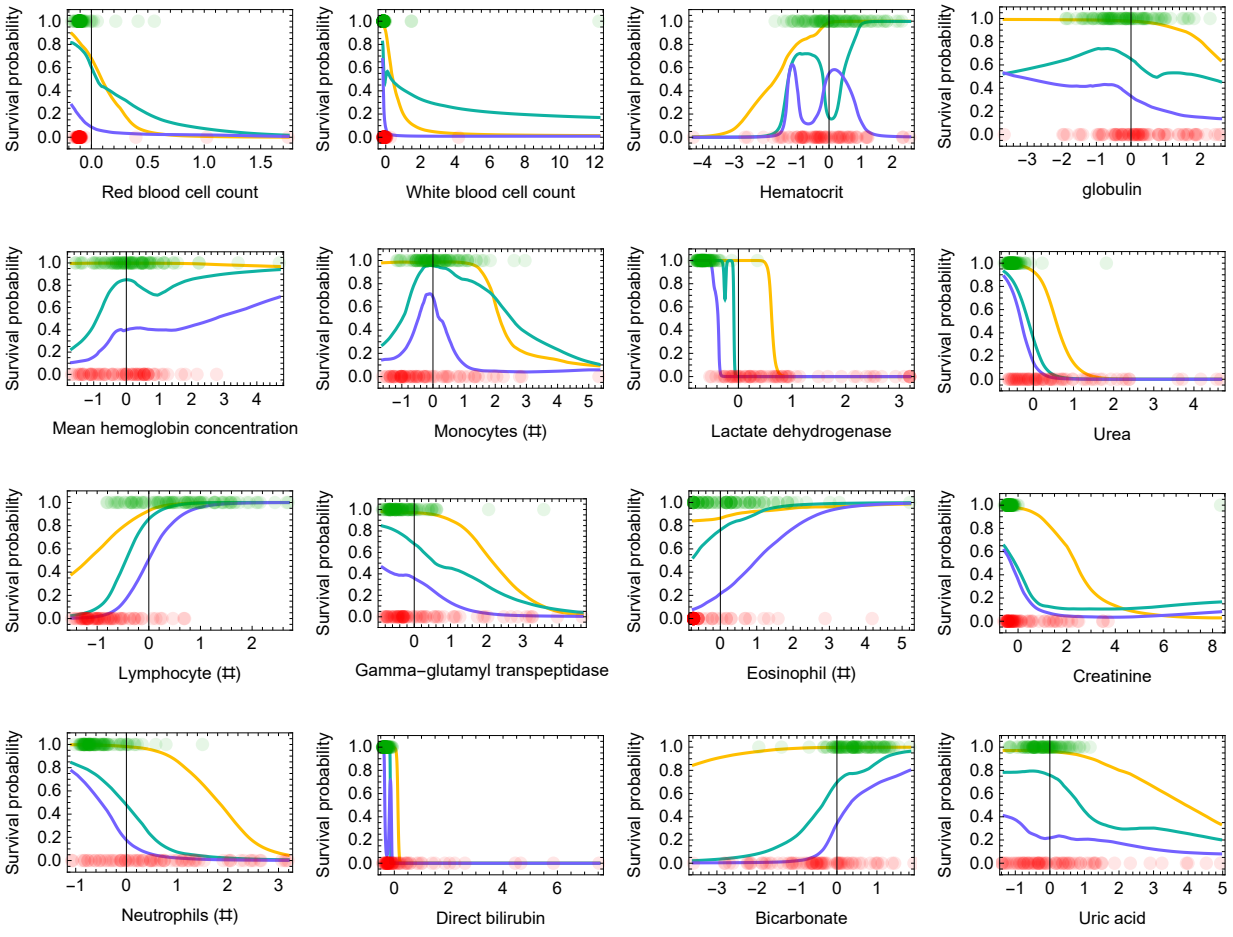
Out[ ]=

In[ ]:=
```
ProgressIndicator[Dynamic[k], {1, 73}]
```

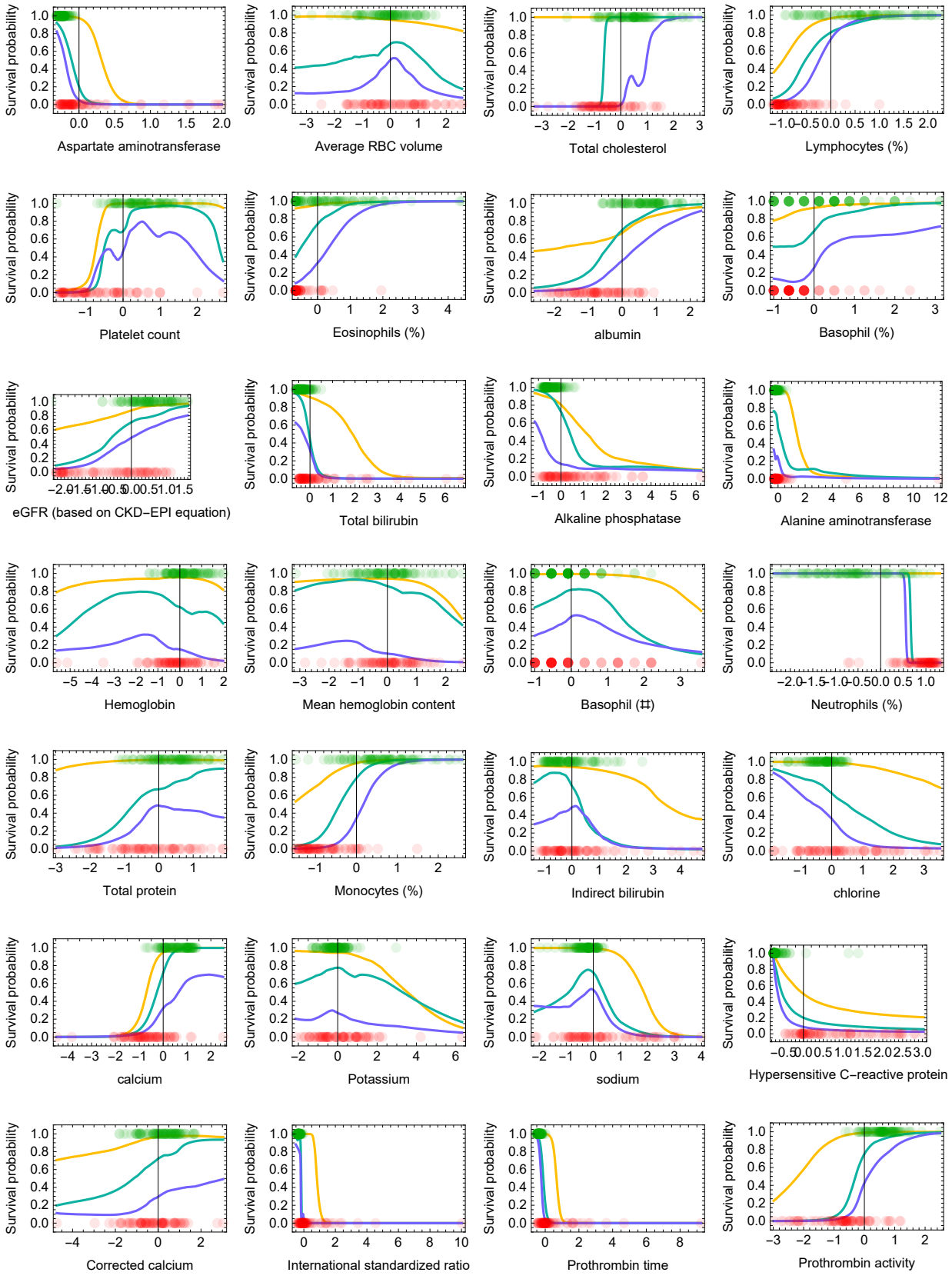
Out[ ]=

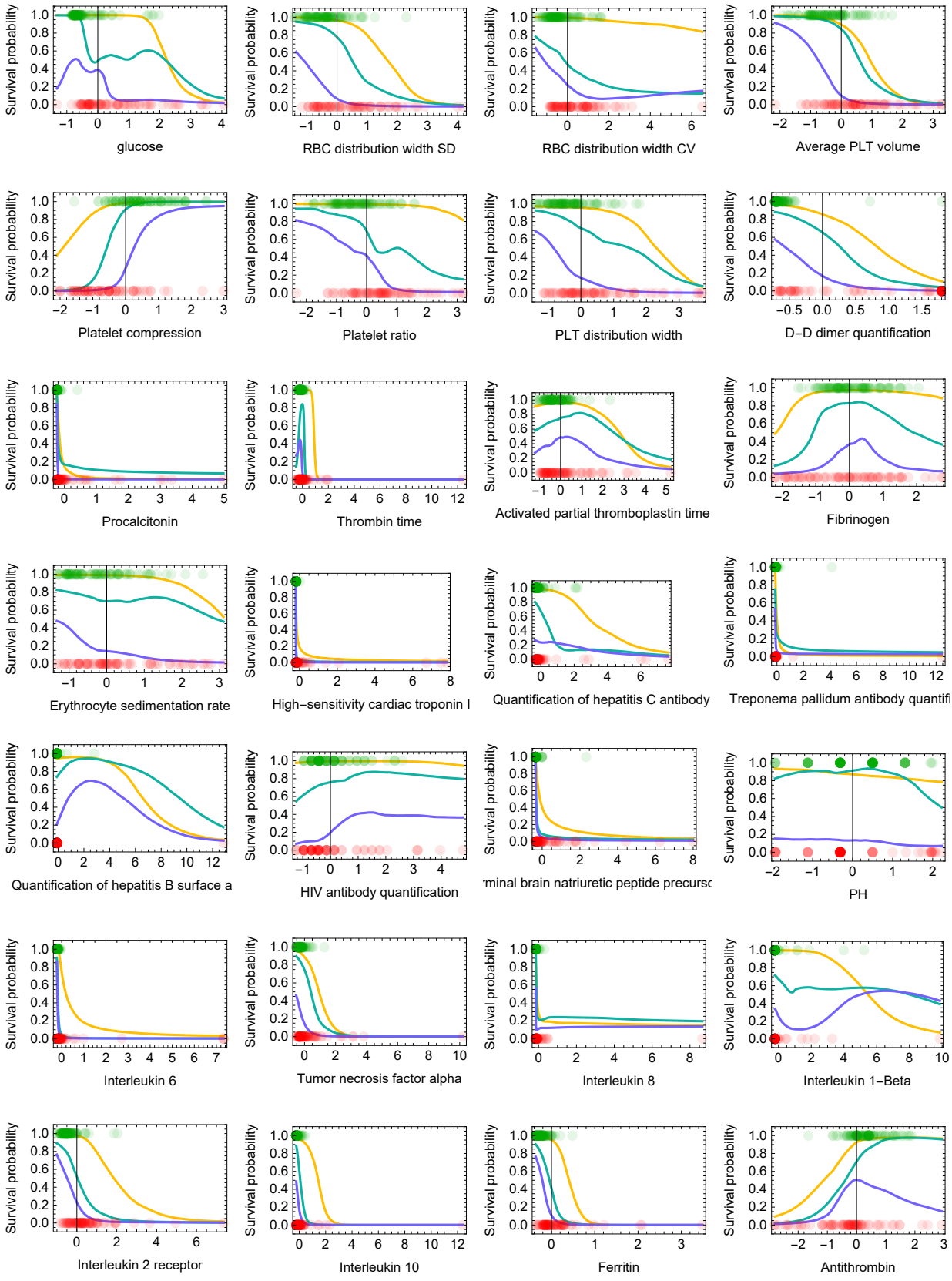
```
c2 = Table[Classify[trainingset2[[i]] → 3,
    Method → "NeuralNetwork", PerformanceGoal → "Quality"], {i, 1, 73}];
```

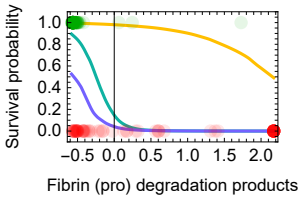
In[ ]:=
```
Multicolumn[Table[
  Show[{Plot[{c2[[k]][{x, "young"}, {"Probability", "survived"}], c2[[k]][{x, "middle age"},
       {"Probability", "survived"}], c2[[k]][{x, "old"}, {"Probability", "survived"}]},
     {x, Min[trainingset2[[k]][All, 1]], Max[trainingset2[[k]][All, 1]]},
     Frame → True, FrameLabel → {labels[[4 + k]], "Survival probability"},
     Exclusions → None, PlotRange → {-0.1, 1.1}, PlotStyle → {■, ■, ■}],
    ListPlot[{ArrayReshape[Riffle[data1[[1 ;; 99, 3 + k]], ConstantArray[1,
        Length[data1[[1 ;; 99, 3 + k]]]]], {Length[data1[[1 ;; 99, 3 + k]], 2}],
      ArrayReshape[Riffle[data1[[100 ;; 176, 3 + k]], ConstantArray[0,
        Length[data1[[100 ;; 176, 3 + k]]]]], {Length[data1[[100 ;; 176, 3 + k]], 2}]},
     PlotStyle → {Directive[Darker[Green], PointSize[Large], Opacity[0.1]],
       Directive[Red, PointSize[Large], Opacity[0.1]]}]}],
  {k, 1, 73, 1}], 4, Appearance → "Horizontal"]
SwatchLegend[{■, ■, ■, ■, ■}, {"young", "middle age", "old", "survived", "died"},
 LegendFunction → "Frame"]
```



Out[ ]=


Survival probability
Aspartate aminotransferase
Average RBC volume
Total cholesterol
Lymphocytes (%)
Platelet count
Eosinophils (%)
albumin
Basophil (%)
eGFR (based on CKD–EPI equation)
Total bilirubin
Alkaline phosphatase
Alanine aminotransferase
Hemoglobin
Mean hemoglobin content
Basophil (#)
Neutrophils (%)
Total protein
Monocytes (%)
Indirect bilirubin
chlorine
calcium
Potassium
sodium
Hypersensitive C–reactive protein
Corrected calcium
International standardized ratio
Prothrombin time
Prothrombin activity


Survival probability
glucose
RBC distribution width SD
RBC distribution width CV
Average PLT volume
Platelet compression
Platelet ratio
PLT distribution width
D–D dimer quantification
Procalcitonin
Thrombin time
Activated partial thromboplastin time
Fibrinogen
Erythrocyte sedimentation rate
High–sensitivity cardiac troponin I
Quantification of hepatitis C antibody
Treponema pallidum antibody quantifi
Quantification of hepatitis B surface a
HIV antibody quantification
minal brain natriuretic peptide precurso
PH
Interleukin 6
Tumor necrosis factor alpha
Interleukin 8
Interleukin 1–Beta
Interleukin 2 receptor
Interleukin 10
Ferritin
Antithrombin

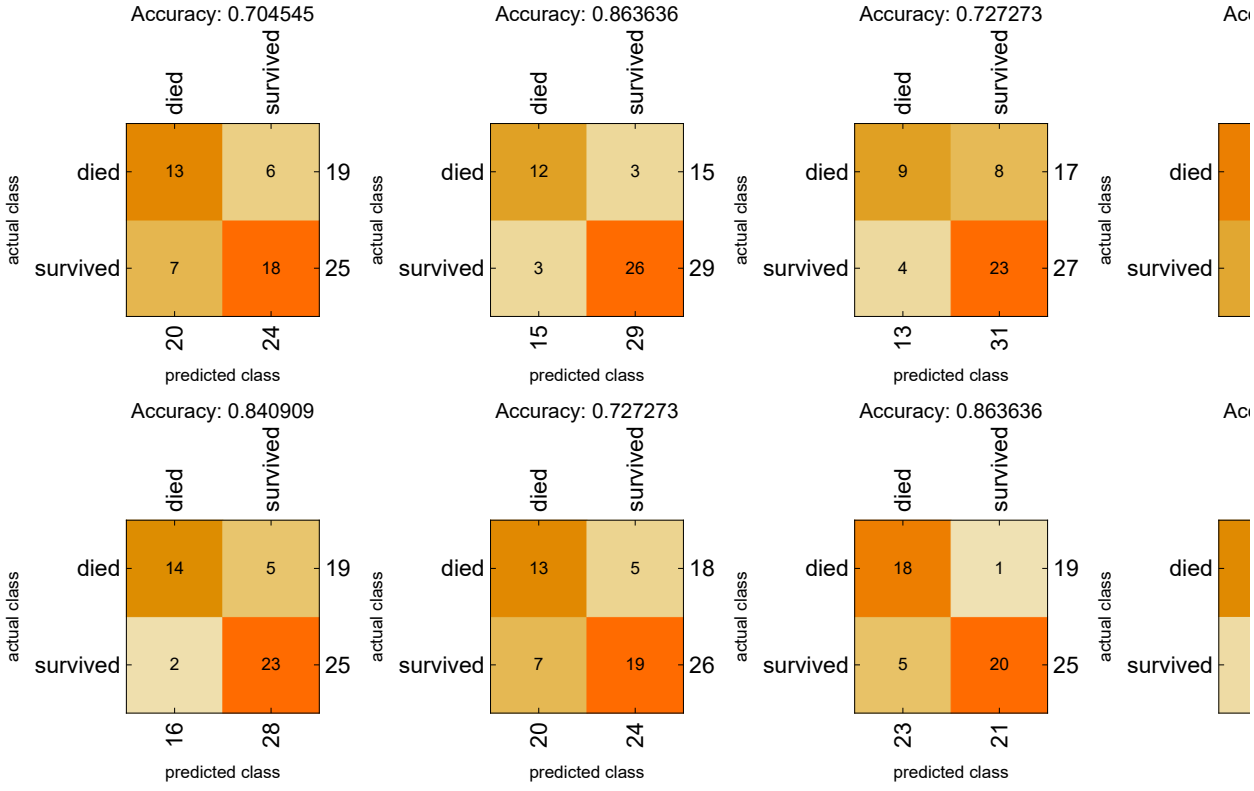


Out[ ]=
young
middle age
old
survived
died

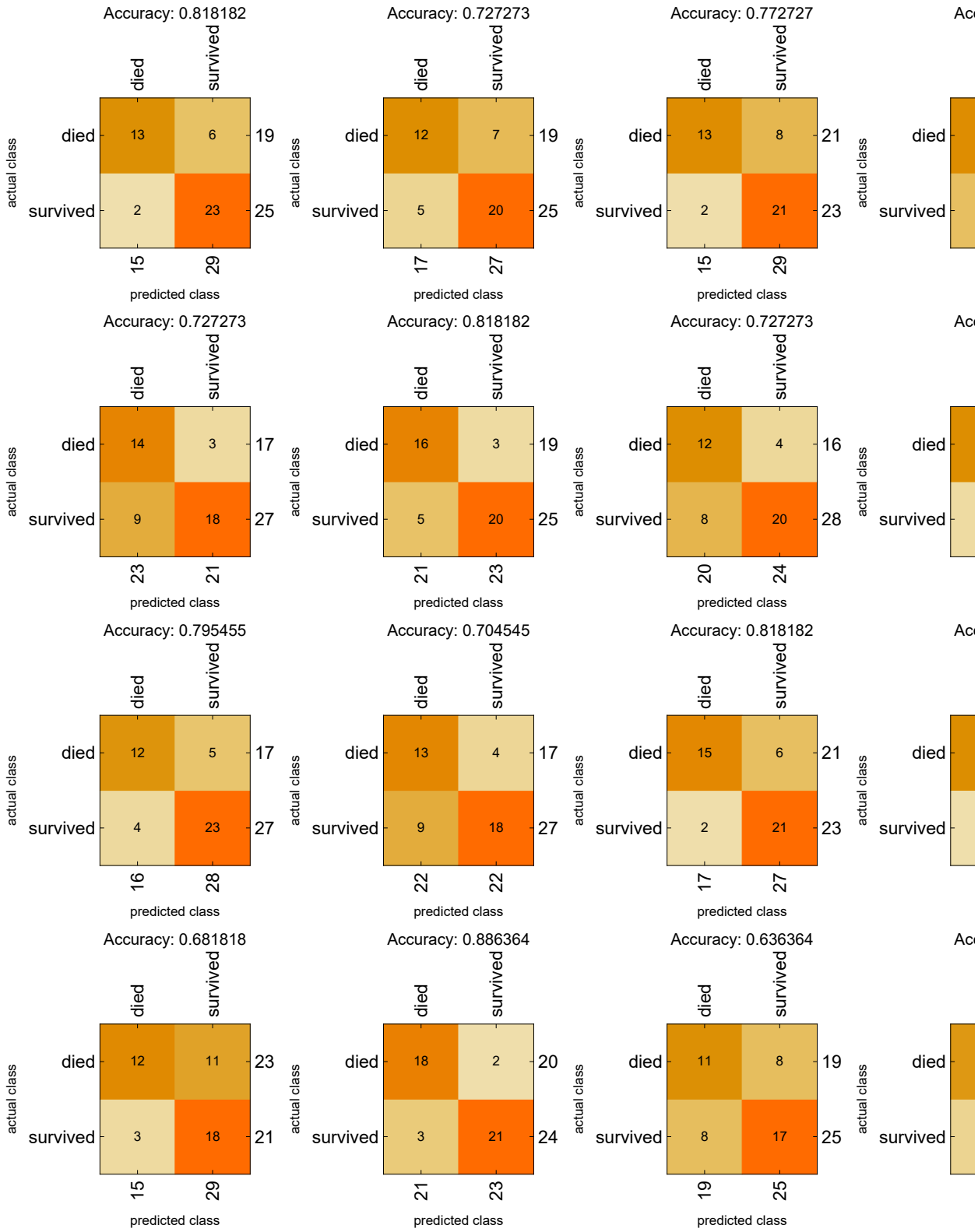
Confusion matrix and accuracy

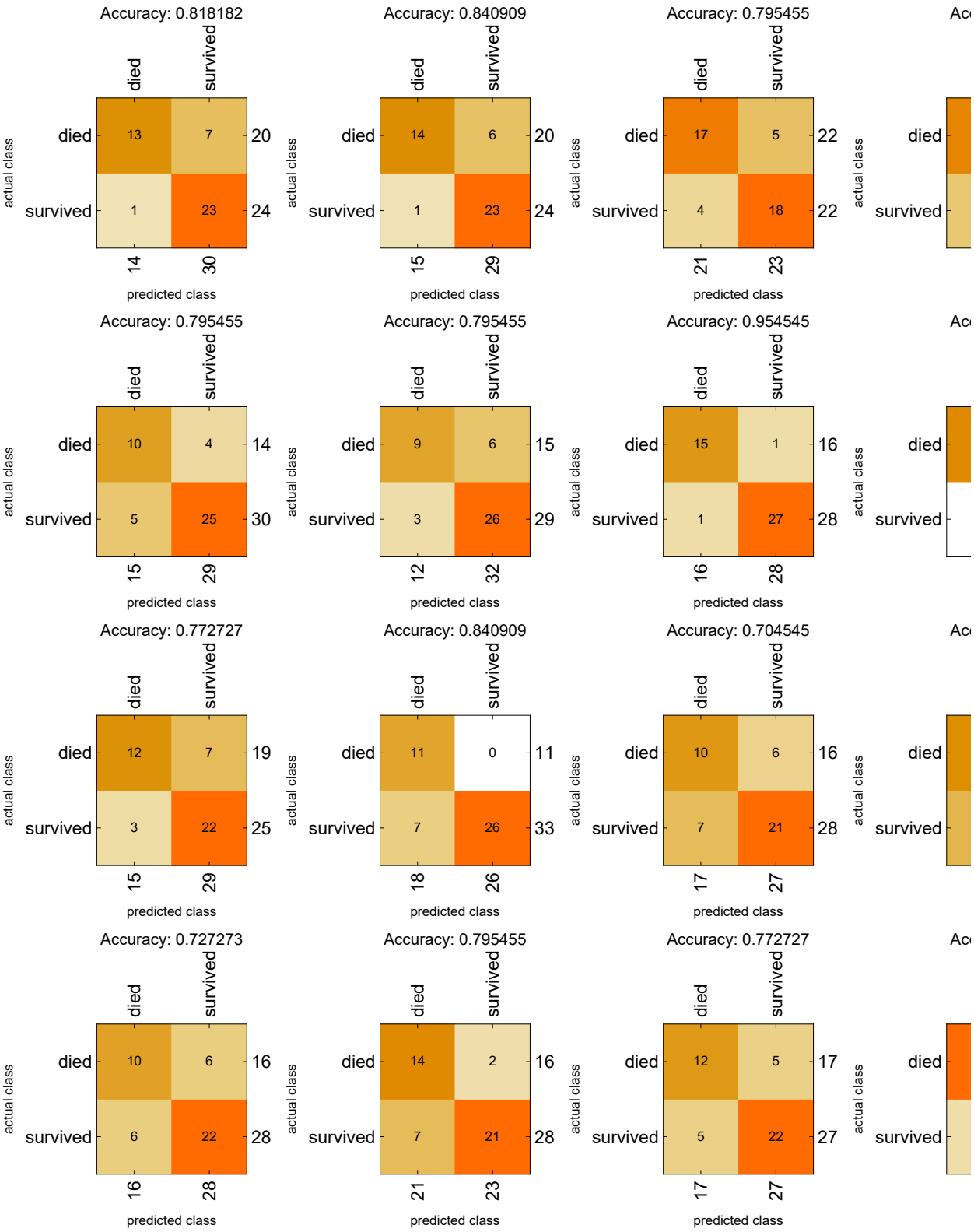
In[ ]:=
```
cm2 = Table[ClassifierMeasurements[c2[[i]], testset2[[i]] → 3], {i, 1, 73}];
```

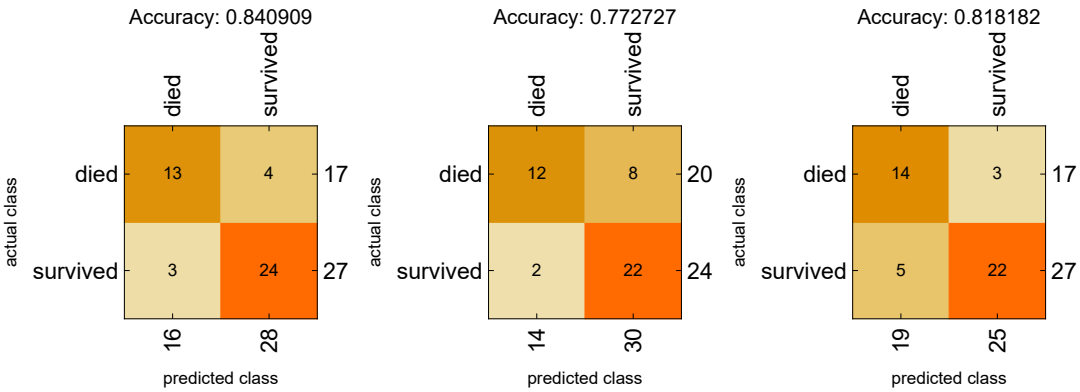
In[ ]:=
```
Multicolumn[Table[Show[cm2[[i]]["ConfusionMatrixPlot"], ImageSize → Small,
   PlotLabel → StringJoin["Accuracy: " , ToString[cm2[[i]]["Accuracy"]]]],
  {i, 1, 73}], 7, Appearance → "Horizontal"]
```



Out[ ]=


Accuracy: 0.818182
died survived
died 13 6 19
survived 2 23 25
15 29
actual class
predicted class
Accuracy: 0.727273
died survived
died 12 7 19
survived 5 20 25
17 27
actual class
predicted class
Accuracy: 0.772727
died survived
died 13 8 21
survived 2 21 23
15 29
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.727273
died survived
died 14 3 17
survived 9 18 27
23 21
actual class
predicted class
Accuracy: 0.818182
died survived
died 16 3 19
survived 5 20 25
21 23
actual class
predicted class
Accuracy: 0.727273
died survived
died 12 4 16
survived 8 20 28
20 24
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.795455
died survived
died 12 5 17
survived 4 23 27
16 28
actual class
predicted class
Accuracy: 0.704545
died survived
died 13 4 17
survived 9 18 27
22 22
actual class
predicted class
Accuracy: 0.818182
died survived
died 15 6 21
survived 2 21 23
17 27
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.681818
died survived
died 12 11 23
survived 3 18 21
15 29
actual class
predicted class
Accuracy: 0.886364
died survived
died 18 2 20
survived 3 21 24
21 23
actual class
predicted class
Accuracy: 0.636364
died survived
died 11 8 19
survived 8 17 25
19 25
actual class
predicted class
Ac
died
survived
actual class


Accuracy: 0.818182
died survived
died 13 7 20
survived 1 23 24
14 30
actual class
predicted class
Accuracy: 0.840909
died survived
died 14 6 20
survived 1 23 24
15 29
actual class
predicted class
Accuracy: 0.795455
died survived
died 17 5 22
survived 4 18 22
21 23
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.795455
died survived
died 10 4 14
survived 5 25 30
15 29
actual class
predicted class
Accuracy: 0.795455
died survived
died 9 6 15
survived 3 26 29
12 32
actual class
predicted class
Accuracy: 0.954545
died survived
died 15 1 16
survived 1 27 28
16 28
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.772727
died survived
died 12 7 19
survived 3 22 25
15 29
actual class
predicted class
Accuracy: 0.840909
died survived
died 11 0 11
survived 7 26 33
18 26
actual class
predicted class
Accuracy: 0.704545
died survived
died 10 6 16
survived 7 21 28
17 27
actual class
predicted class
Ac
died
survived
actual class
Accuracy: 0.727273
died survived
died 10 6 16
survived 6 22 28
16 28
actual class
predicted class
Accuracy: 0.795455
died survived
died 14 2 16
survived 7 21 28
21 23
actual class
predicted class
Accuracy: 0.772727
died survived
died 12 5 17
survived 5 22 27
17 27
actual class
predicted class
Ac
died
survived
actual class

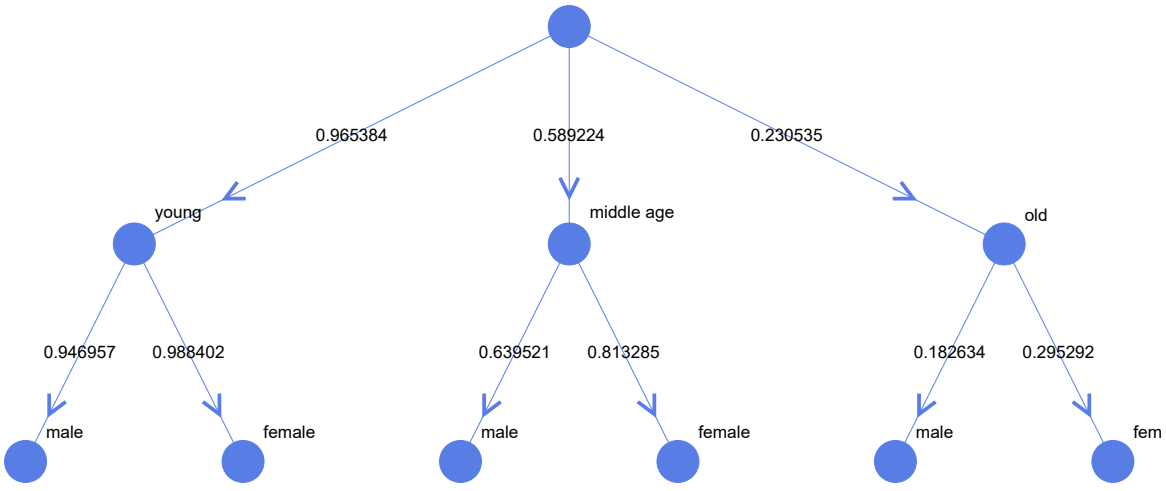


Decision Tree Classifier

In[ ]:=
```
c3 = Classify[data1[[All, {2, 3, 77}]] → 3,
   Method → "NeuralNetwork", PerformanceGoal → "Quality"];
```

In[ ]:=
```
Show[TreePlot[{Labeled[1 → 2, c3[{"young", Missing[]}, {"Probability", "survived"}]],
   Labeled[1 → 3, c3[{"middle age", Missing[]}, {"Probability", "survived"}]],
   Labeled[1 → 4, c3[{"old", Missing[]}, {"Probability", "survived"}]],
   Labeled[2 → 5, c3[{"young", "male"}, {"Probability", "survived"}]],
   Labeled[2 → 6, c3[{"young", "female"}, {"Probability", "survived"}]],
   Labeled[3 → 7, c3[{"middle age", "male"}, {"Probability", "survived"}]],
   Labeled[3 → 8, c3[{"middle age", "female"}, {"Probability", "survived"}]],
   Labeled[4 → 9, c3[{"old", "male"}, {"Probability", "survived"}]],
   Labeled[4 → 10, c3[{"old", "female"}, {"Probability", "survived"}]]},
  PlotTheme → "Minimal", VertexLabels → {2 → "young", 3 → "middle age", 4 → "old",
    5 → "male", 6 → "female", 7 → "male", 8 → "female", 9 → "male", 10 → "female"},
  DirectedEdges → True], ImageSize → Large]
```

Out[ ]=

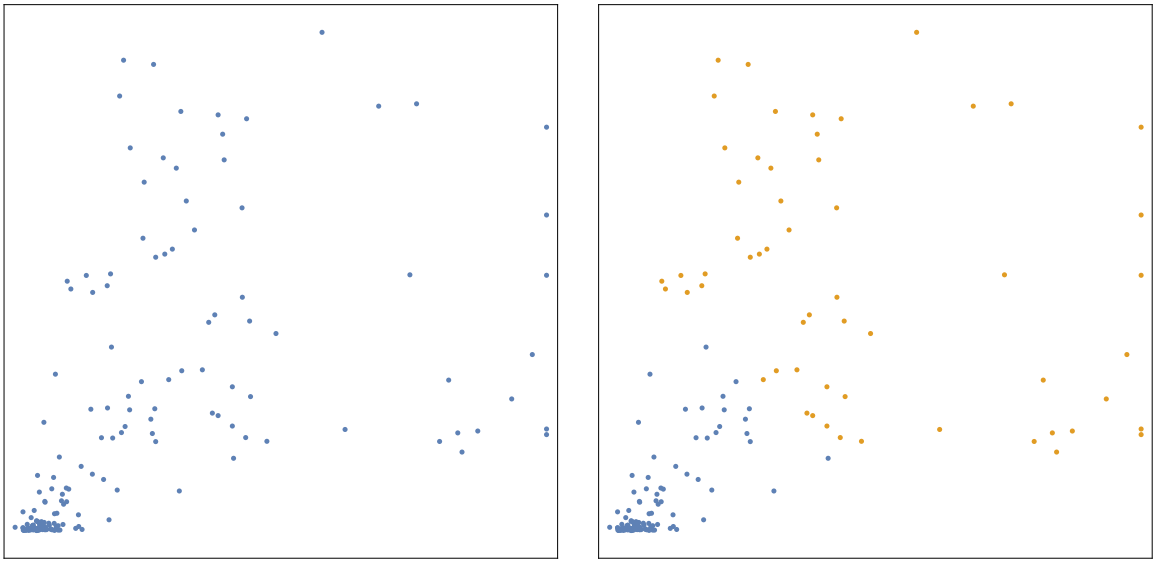


Clustering analysis

Clustering of lactate dehydrogenase and Hypersensitive c reactive protein.

In[ ]:=
```
data2 = data1[[All, {9, 42}]];
plots = Table[ListPlot[FindClusters[data2, n, Method → "KMeans"],
    AspectRatio → 1, Frame → True, Axes → False, FrameTicks → None], {n, 2}];
GraphicsGrid[Partition[plots, 2]]
```

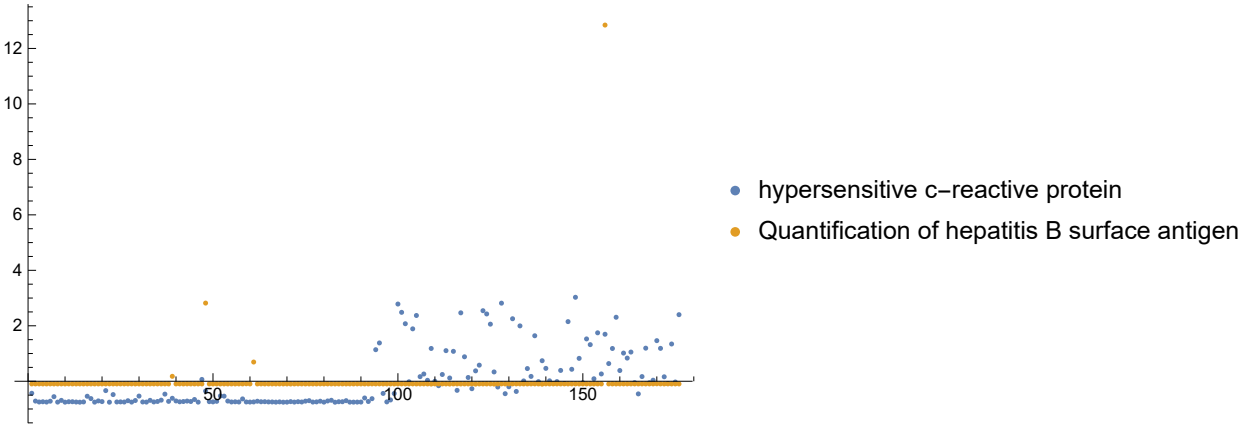
Out[ ]=



Need to drop Quantification of hepatitis B surface antigen because the ratio just shows Hypersensitive C reactive protein which is the most important feature

In[ ]:=
```
ListPlot[{data1[[All, 42]], data1[[All, 63]]},
 PlotLegends → {"hypersensitive c-reactive protein",
   "Quantification of hepatitis B surface antigen"}, PlotRange → All]
```

Out[ ]=



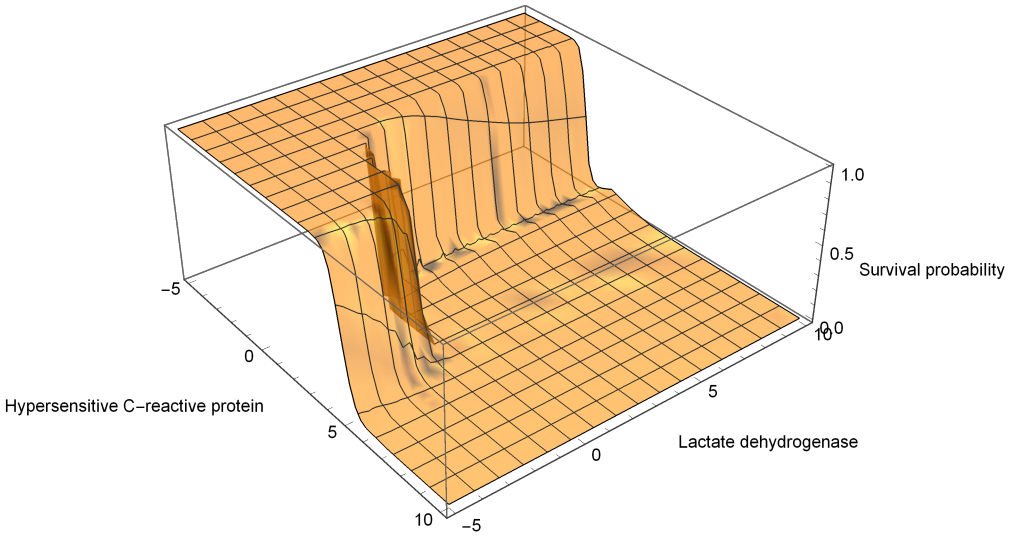
In[ ]:=
```
c = Classify[data1[[All, {9, 42, 76}]] → 3,
  Method → "NeuralNetwork", PerformanceGoal → "Quality"]
```

Out[ ]=
```
ClassifierFunction[ Input type: NumericalVector (length: 2)
                    Classes: died, survived ]
```

Survival probability of lactate dehydrogenase and Hypersensitive c reactive protein.

In[ ]:=
```
p[hsCRP_, LD_] := c[{hsCRP, LD}, {"Probability", "survived"}];
Plot3D[{p[x, y]}, {x, -5, 10}, {y, -5, 10}, Boxed → True,
 AxesLabel → {"Hypersensitive C-reactive protein", "Lactate dehydrogenase",
   "Survival probability"}, Exclusions → None, PlotStyle → Opacity[0.6]]
```

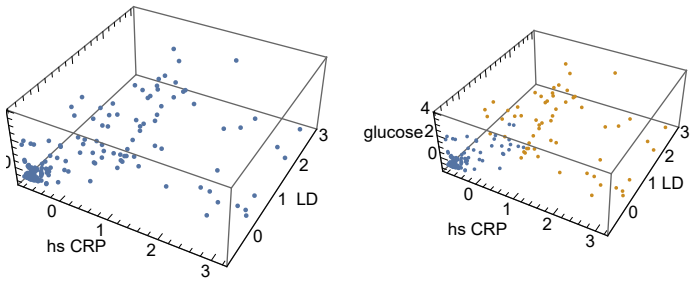
Out[ ]=



Analysis of top 3 features (hs CRP, LD, glucose)

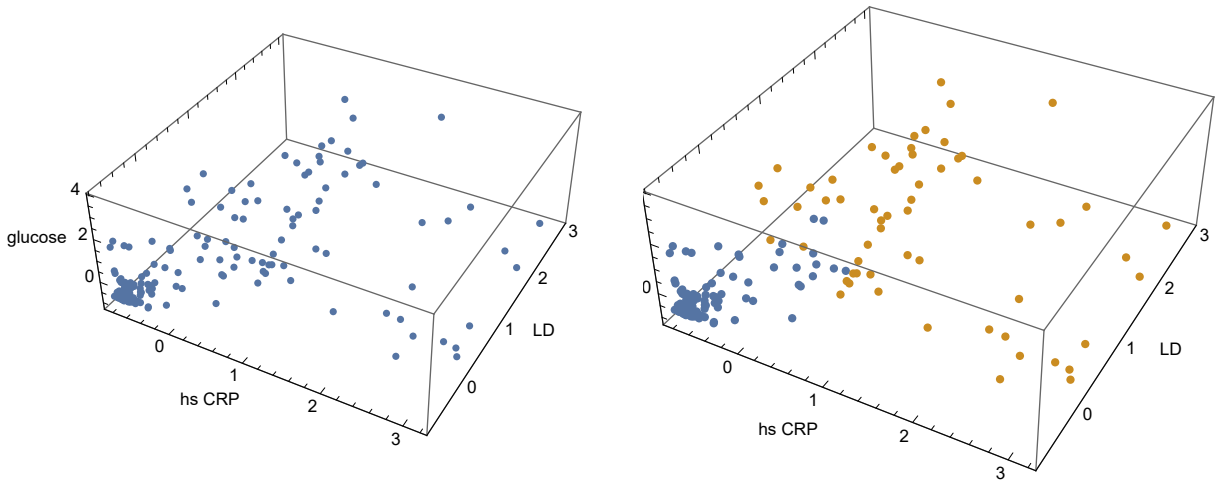
In[ ]:=
```
data2 = data1[[All, {9, 42, 47}]];
plots = Table[ListPointPlot3D[FindClusters[data2, n, Method → "KMeans"],
    AspectRatio → 1, Axes → True, AxesLabel → {"hs CRP", "LD", "glucose"}], {n, 2}];
GraphicsGrid[Partition[plots, 2]]
```

Out[ ]=



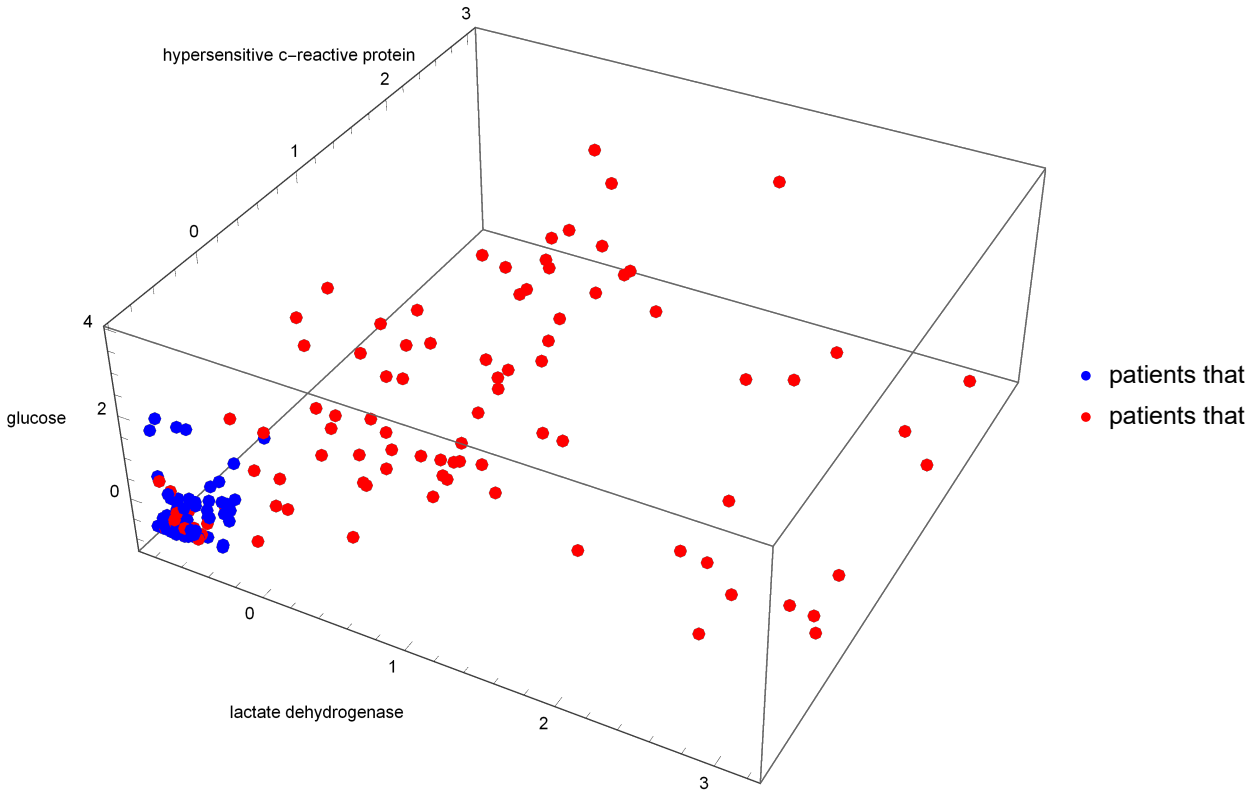
In[ ]:=
```
Show[%460, ImageSize → Full]
```

Out[ ]=



In[ ]:=
```
ListPointPlot3D[{data1[[Range[99], {9, 42, 47}]], data1[[Range[99] + 77, {9, 42, 47}]]},
 PlotStyle → {Blue, Red}, PlotLegends → {"patients that survived", "patients that died"},
 AxesLabel → {"lactate dehydrogenase", "hypersensitive c-reactive protein", "glucose"}]
```

Out[ ]=



Survival probability prediction

randomize data for training/test split

In[ ]:=
```
SeedRandom[123];
rs = RandomSample[data1[[All, Prepend[Range[75] + 2, 1]]], 176];
```

Split data into 75% training and 25% test

In[ ]:=
```
trainingset = rs[[1 ;; 132]];
testset = rs[[133 ;; 176]];
```

In[ ]:=
```
c = Classify[trainingset → 76, Method → "NeuralNetwork", PerformanceGoal → "Quality"]
```

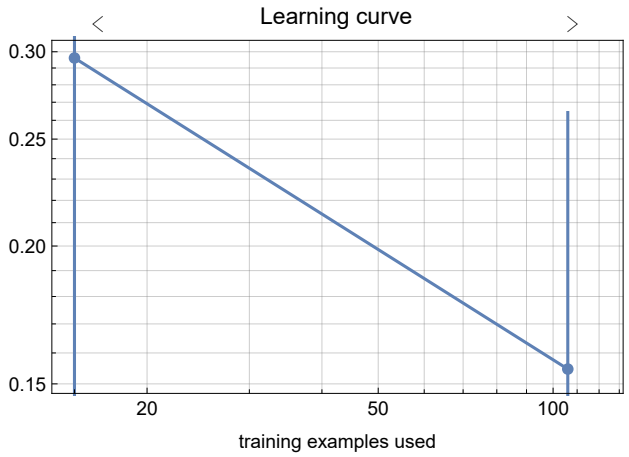
Out[ ]=
```
ClassifierFunction[ Input type: Mixed (number: 75)
                    Classes: died, survived ]
```

In[ ]:=
```
Information[c]
```

Out[ ]=

Classifier information

Data type	Mixed (number: 75)
Classes	died, survived
Accuracy	(94. ± 8.)%
Method	NeuralNetwork
Single evaluation time	3.22 ms/example
Batch evaluation speed	29.6 examples/ms
Loss	0.155 ± 0.11
Model memory	648. kB
Training examples used	132 examples
Training time	9.1 s



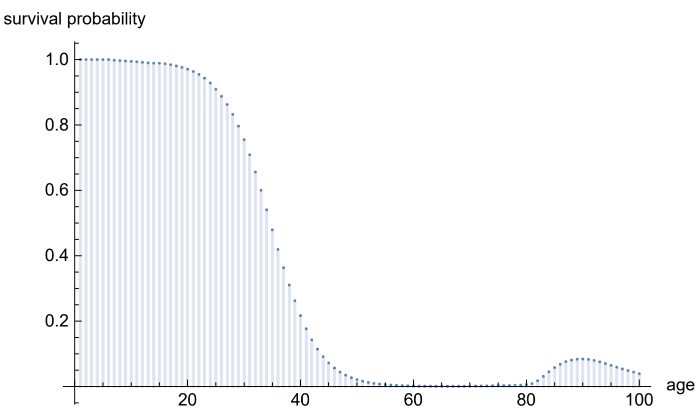
classifier probability function

In[∘]:=
```
Manipulate[{c[testset[[patientID, Range[75]]], "Probabilities"],
  testset[[patientID, 76]]}, {patientID, 1, 52, 1}]
```

With missing values

In[∘]:=
```
DiscretePlot[
 Values[c[{age, "female", Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
     Missing[], Missing[], Missing[], Missing[], Missing[]}, "Probabilities"]][[2]],
 {age, 1, 100, 1}, AxesLabel → {"age", "survival probability"}]
```

Out[∘]=



Accuracy and Confusion Matrix

In[∘]:=
```
cm = ClassifierMeasurements[c, testset → 76]
```

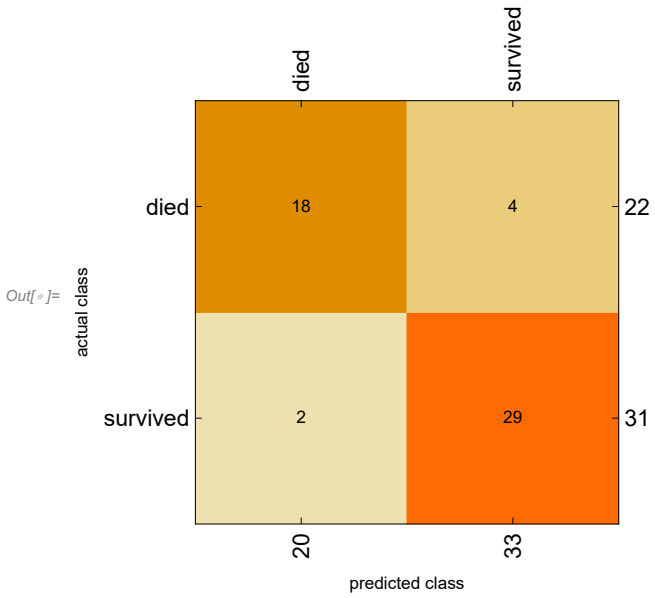
Out[∘]=
```
ClassifierMeasurementsObject[ Classifier: NeuralNetwork
                              Number of test examples: 44 ]
```

In[∘]:=
```
cm["Accuracy"]
```

Out[∘]=
```
0.931818
```

In[ ]:=
```
cm["ConfusionMatrixPlot"]
```

Out[ ]=



Repeating above model with 50 random shuffling

In[ ]:=
```
ProgressIndicator[Dynamic[i], {1, 50}]
```

Out[ ]=

randomize data for training/test split

In[ ]:=
```
randomshuffles = Table[RandomSample[data1[[All, Prepend[Range[75] + 2, 1]]], 176], 50];
```

Split data into 75% training and 25% test

In[ ]:=
```
trainingtensor = Transpose[Transpose[randomshuffles][[1 ;; 132]]];
testtensor = Transpose[Transpose[randomshuffles][[133 ;; 176]]];
```

In[ ]:=
```
ctensor = Table[Classify[trainingtensor[[i]] → 76,
    Method → "NeuralNetwork", PerformanceGoal → "Quality"], {i, 1, 50, 1}];
```

In[ ]:=
```
accuracylist =
  Table[ClassifierMeasurements[ctensor[[i]], testtensor[[i]] → 76]["Accuracy"], {i, 1, 50, 1}];
```

In[ ]:=
```
accuracylist
```

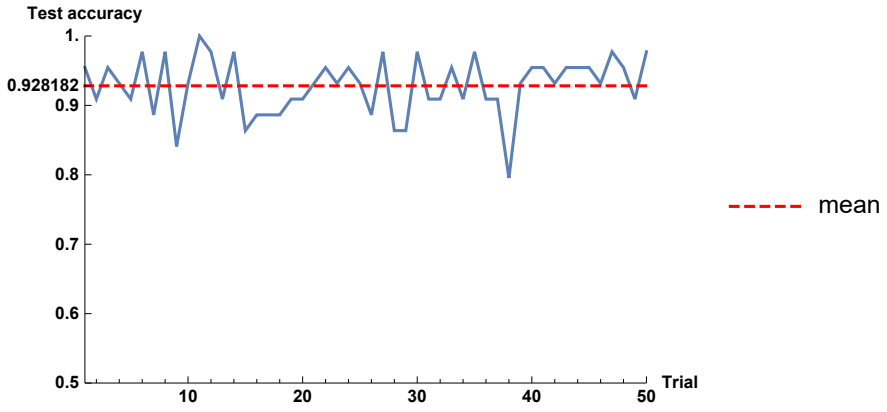
Out[ ]=
```
{0.954545, 0.909091, 0.954545, 0.931818, 0.909091, 0.977273, 0.886364, 0.977273, 0.840909,
 0.931818, 1., 0.977273, 0.909091, 0.977273, 0.863636, 0.886364, 0.886364, 0.886364,
 0.909091, 0.909091, 0.931818, 0.954545, 0.931818, 0.954545, 0.931818, 0.886364,
 0.977273, 0.863636, 0.863636, 0.977273, 0.909091, 0.909091, 0.954545, 0.909091,
 0.977273, 0.909091, 0.909091, 0.795455, 0.931818, 0.954545, 0.954545, 0.931818,
 0.954545, 0.954545, 0.954545, 0.931818, 0.977273, 0.954545, 0.909091, 0.977273}
```

In[ ]:=
```
Mean[accuracylist]
```

Out[ ]=
```
0.928182
```

```
In[•]:= Show[ListLinePlot[accuracylist, PlotRange → {{1, 50}, {0.5, 1}},
          Ticks → {Automatic, {0.5`, 0.6`, 0.7`, 0.8`, 0.9`, Mean[accuracylist], 1.`}},
          AxesLabel → {"Trial", "Test accuracy"}, LabelStyle → Bold],
        Plot[Legended[Mean[accuracylist], "mean"], {x, 1, 50}, PlotStyle → {Red, Dashed}]]
```

Out[•]=



Repeating above model by selecting 75% of training data from each age group

import data

```
In[•]:= importdata = SemanticImport["D:\\Google Drive\\Rich
            Internship 2020\\XGBoost\\xgboost_covid\\data\\Data (young).csv"];
        dataYoung = Values[Normal[importdata]][[All, 2 ;; 78]];
        importdata = SemanticImport["D:\\Google Drive\\Rich Internship
            2020\\XGBoost\\xgboost_covid\\data\\Data (middle age).csv"];
        dataMiddle = Values[Normal[importdata]][[All, 2 ;; 78]];
        importdata = SemanticImport["D:\\Google Drive\\Rich
            Internship 2020\\XGBoost\\xgboost_covid\\data\\Data (old).csv"];
        dataOld = Values[Normal[importdata]][[All, 2 ;; 78]];
```

training/test split

randomize data for training/test split

```
randomshuffles1 =
  Table[RandomSample[dataYoung[[All, Prepend[Range[75] + 2, 1]]], Length[dataYoung]], 50];
randomshuffles2 = Table[RandomSample[
    dataMiddle[[All, Prepend[Range[75] + 2, 1]]], Length[dataMiddle]], 50];
randomshuffles3 = Table[RandomSample[dataOld[[All, Prepend[Range[75] + 2, 1]]],
    Length[dataOld]], 50];
```

Split data into 75% training and 25% test

```
In[•]:= trainingtensor1 = Transpose[Transpose[randomshuffles1][[1 ;; 20]]];
        testtensor1 = Transpose[Transpose[randomshuffles1][[21 ;; 26]]];

        trainingtensor2 = Transpose[Transpose[randomshuffles2][[1 ;; 56]]];
        testtensor2 = Transpose[Transpose[randomshuffles2][[57 ;; 75]]];

        trainingtensor3 = Transpose[Transpose[randomshuffles3][[1 ;; 56]]];
        testtensor3 = Transpose[Transpose[randomshuffles3][[57 ;; 75]]];

        combinedTrainingTensor =
          Table[Join[trainingtensor1[[i]], trainingtensor2[[i]], trainingtensor3[[i]]], {i, 1, 50, 1}];
        combinedTestTensor = Table[Join[testtensor1[[i]], testtensor2[[i]], testtensor3[[i]]],
           {i, 1, 50, 1}];
```

train the model 50 times

```
In[•]:= ProgressIndicator[Dynamic[i], {1, 50}]
```

Out[•]= 

```
In[•]:= ProgressIndicator[Dynamic[k], {1, 50}]
```

Out[•]= 

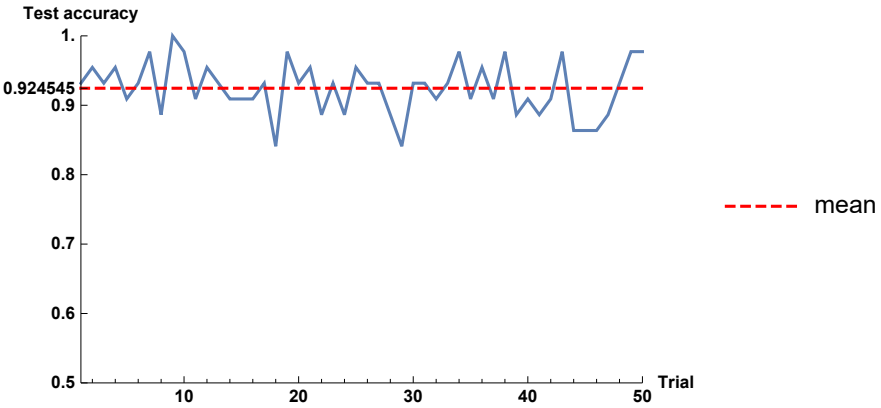
```
In[•]:= ctensor2 = Table[Classify[combinedTrainingTensor[[i]] → 76,
            Method → "NeuralNetwork", PerformanceGoal → "Quality"], {i, 1, 50, 1}];
        accuracylist2 = Table[ClassifierMeasurements[ctensor2[[k]], combinedTestTensor[[k]] → 76][
            "Accuracy"], {k, 1, 50, 1}];

In[•]:= Mean[accuracylist2]

Out[•]= 0.924545
```

accuracy plot

```
In[•]:= Show[ListLinePlot[accuracylist2, PlotRange → {{1, 50}, {0.5, 1}},
          Ticks → {Automatic, {0.5`, 0.6`, 0.7`, 0.8`, 0.9`, Mean[accuracylist2], 1.`}},
          AxesLabel → {"Trial", "Test accuracy"}, LabelStyle → Bold],
         Plot[Legended[Mean[accuracylist2], "mean"], {x, 1, 50}, PlotStyle → {Red, Dashed}]]
```

Out[•]= 

Final model

randomize data for training/test split

In[ ]:=
```
SeedRandom[321];
rs = RandomSample[data1[[All, Prepend[Range[75] + 2, 1]]], 176];
```

Split data into 75% training and 25% test

In[ ]:=
```
trainingset = rs[[1 ;; 176]];
```

In[ ]:=
```
c = Classify[trainingset → 76, Method → "NeuralNetwork", PerformanceGoal → "Quality"]
```

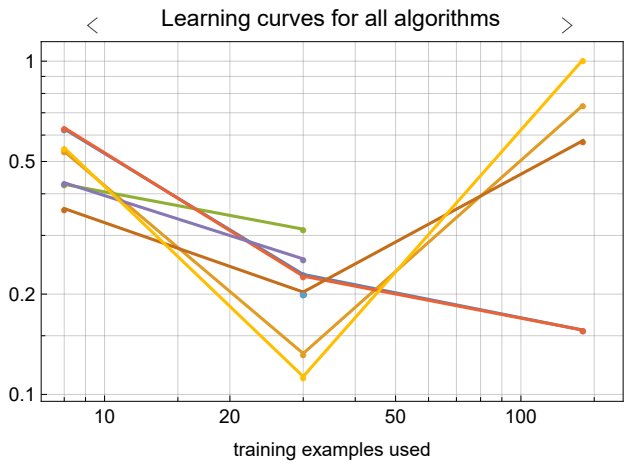
Out[ ]=
```
ClassifierFunction[ Input type: Mixed (number: 75)
                    Classes: died, survived ]
```

In[ ]:=
```
Information[c]
```

Out[ ]=

Classifier information

Data type	Mixed (number: 75)
Classes	died, survived
Accuracy	(93.9 ± 3.0)%
Method	NeuralNetwork
Single evaluation time	3.59 ms/example
Batch evaluation speed	26.5 examples/ms
Loss	0.113 ± 0.056
Model memory	670. kB
Training examples used	176 examples
Training time	19.3 s



In[ ]:=
```
c[[1]]
```

Out[•]= ⟨| ExampleNumber → 176, ClassNumber → 2, Input → ⟨| Preprocessor → ToMLDataset,

Processor → ToVector, ToVector → ImputeMissing, ImputeMissing → LogRescaleNumericalVector |⟩,

→ Standardize → EmbedNominalVector → MergeVectors

Output → ⟨| Preprocessor → ToMLDataset,

Processor → ToVector → IntegerEncodeNominalVector → FromVector → FirstValues ,

ProbabilityPostprocessor → Identity, Name → class,

Marginal → ⟨| died → 0.438202, survived → 0.561798 |⟩ |⟩,

Prior → Automatic, Utility → SparseArray [Specified elements: 2 Dimensions: {2, 3}],

Threshold → 0, TieBreaker → RandomChoice,

PerformanceGoal → Quality, BatchProcessing → Automatic,

Model → ⟨| Method → NeuralNetwork,

Network → NetChain [

	Input	vector (size: 76)
1	LinearLayer	vector (size: 50)
2	ScaledExponentialLinearUnit[*x*]	vector (size: 50)
3	DropoutLayer	vector (size: 50)
4	LinearLayer	vector (size: 50)
5	ScaledExponentialLinearUnit[*x*]	vector (size: 50)
6	LinearLayer	vector (size: 2)
	Output	vector (size: 2)

1: LinearLaver

Parameters

Output dimensions: 50

Arrays

Weights: matrix (size: 50 × 76)

Biases: vector (size: 50)

Ports

Input: vector (size: 76)

Output: vector (size: 50)

],

Training → ⟨| Optimizer → {ADAM, L2Regularization → None}, TrainingProgressFunction →

{Null &, Interval → 1}, TotalTrainingTime → 2.35592, MeanInputsPerSecond → 81 497. |⟩,

InputType → NumericalVector, Processor → Standardize → FirstValues ,

FeatureNumber → 76, PostProcessor → Identity,

Options → ⟨| NetworkType → ⟨| Value → FullyConnected, Options → ‹| |› |⟩, NetworkDepth →

⟨| Value → 2, Options → ‹| |› |⟩, NumberOfParameters → ⟨| Value → 6400, Options → ‹| |› |⟩,

ActivationFunction → ⟨| Value → SELU, Options → ‹| |› |⟩, L2Regularization →

```
    ⟨|Value → None, Options → ⟨||⟩|⟩, Dropout → ⟨|Value → 0.01, Options → ⟨||⟩|⟩,
   NetInitializationMethod → ⟨|Value → Automatic, Options → ⟨||⟩|⟩,
   OptimizationMethod → ⟨|Value → {ADAM, L2Regularization → None}, Options → ⟨||⟩|⟩,
   MaxTrainingRounds → ⟨|Value → 1000, Options → ⟨||⟩|⟩,
   ValidationSet → ⟨|Value → Automatic, Options → ⟨||⟩|⟩,
   EarlyStopping → ⟨|Value → False, Options → ⟨||⟩|⟩,
   TrainingProgressReporting → ⟨|Value → None, Options → ⟨||⟩|⟩, NetTrainOptions →
    ⟨|Value → {LearningRateMultipliers → {}, TargetDevice → CPU}, Options → ⟨||⟩|⟩,
   LossFunction → ⟨|Value → Automatic, Options → ⟨||⟩|⟩,
   ValidationSetRatio → ⟨|Value → 0.153409, Options → ⟨||⟩|⟩|⟩,
TrainingInformation → ⟨|PanelCell → CellObject[ 10881 ], TrainingFunction → Classify,
  EMIterations → Missing[KeyAbsent, EMIterations], ProcessorEntropyShift → 0,
  PreprocessingTime → 2.7364114, LossName → MeanCrossEntropy,
BestModelInformation →
```

MeanCrossEntropy	0.11 ± 0.06
Accuracy	0.939 ± 0.030
EvaluationTime	0.0000793486
TestSize	146
ModelMemory	86928
ModelUtility	3.47616
TrainingSize	30
TrainingTime	0.794328
TrainingMemory	271616
ExperimentCount	1
MeanCrossEntropyHistory	{0.11 ± 0.04}
AccuracyHistory	{0.939 ± 0.021}
Configuration	{ ⋯15 }
FinalTrainingSize	176

```
,
```

Configurations →

Value	Options
NeuralNetwork	NetworkType
	NetworkDepth
	NumberOfParameters
	ActivationFunction
	L2Regularization
	Dropout
	NetInitializationMethod
	OptimizationMethod
	MaxTrainingRounds
	ValidationSet
	EarlyStopping
	TrainingProgressReporting
	NetTrainOptions
	LossFunction

showing 1–1 of 10

PreprocessorMemory → 232 640, InputDimension → 76, OutputDimension → 1,
BaselineLogProbability → −0.68549, VariableBudget → True,
CheckpointingInfo → ⟨|Checkpointing → False|⟩, UserStop → False,
NaturalStop → True, AbortStop → False, LastReportingTime → $3.8064774021455742 \times 10^9$,

RoundPartitioning →

TrainingSizes	TimeBudgets	ElapsedTimes	ExperimentCounts
8	5.53699	7.02299	8
30	7.90998	10.0742	9
141	11.3	14.5943	7

|⟩,

Log → ⟨| Example →

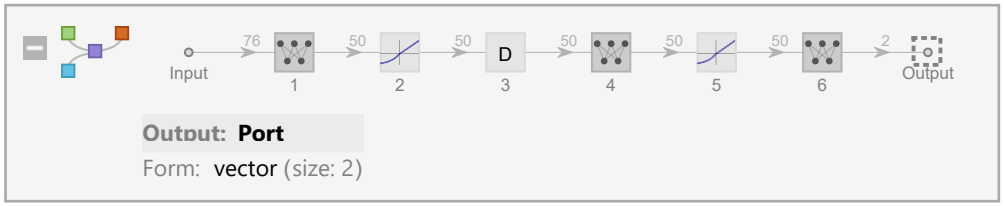
	Type	Weight	Values
f1	Numerical	1	{67}
f2	Nominal	1	{male}
f3	Numerical	1	{−0.100623}
f4	Numerical	1	{−0.0985011}
f5	Numerical	1	{−0.279312}
f6	Numerical	1	{0.093508}
f7	Numerical	1	{0.00443616}
f8	Numerical	1	{−1.23278}
f9	Numerical	1	{1.66531}
f10	Numerical	1	{−0.119824}
f11	Numerical	1	{−1.18998}
f12	Numerical	1	{−0.308749}
f13	Numerical	1	{−0.741803}
f14	Numerical	1	{−0.417476}
f15	Numerical	1	{0.487395}
f16	Numerical	1	{−0.190604}
⋮ 59			

1 examples | no weights | grouped examples | no density weights

, TrainingTime → 35.2581, MaxTrainingMemory → 11 988 608,
DataMemory → 345 960, FunctionMmemory → 676 256, LanguageVersion → {12., 0},
Date → Sat 15 Aug 2020 10:50:04 GMT−4., ProcessorCount → 4, ProcessorType → x86−64,
OperatingSystem → Windows, SystemWordLength → 64, Evaluations → {} |⟩ |⟩

In[ ]:= **c[[1]][[11]][[2]] // NetGraph**

Out[ ]= NetGraph[



]

example of scoring function based on age

In[ ]:=
```
{age, Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
  Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[]} // Length
```

Out[ ]= 75

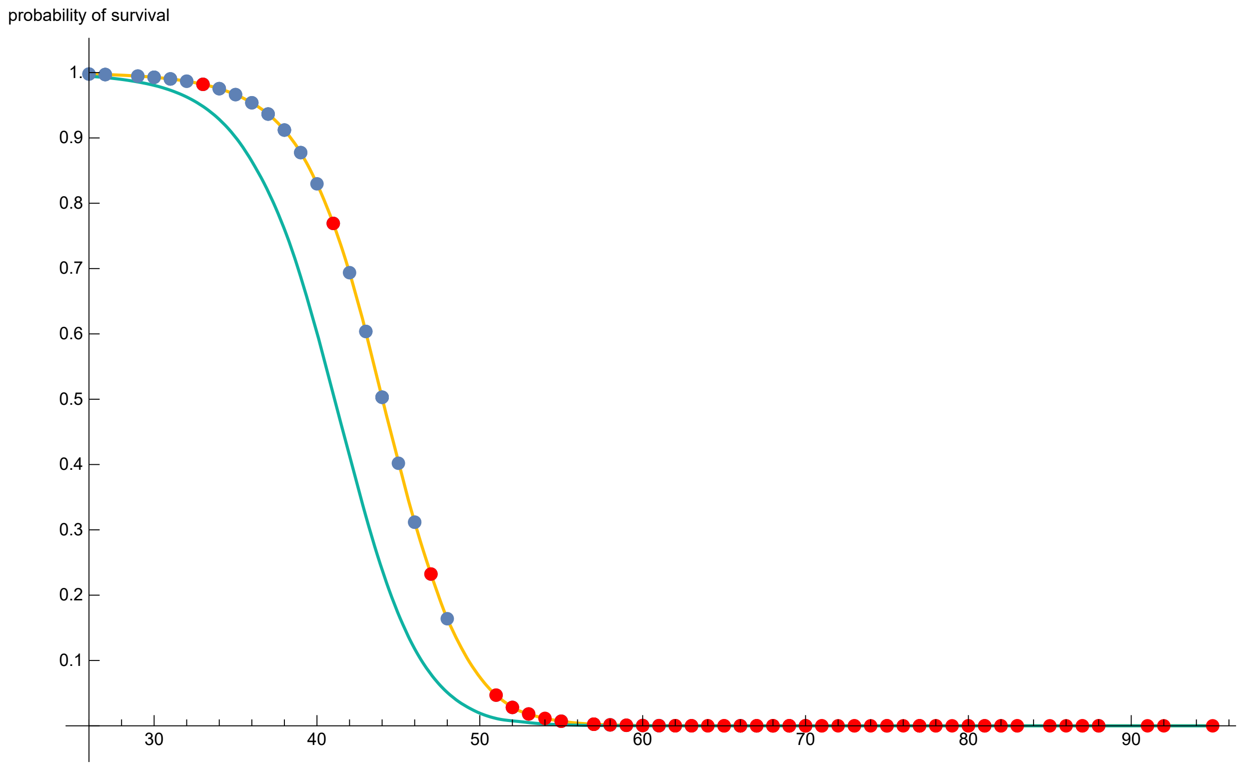
In[ ]:=
```
f[age_] :=
  Values[c[{age, "male", Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[]}, "Probabilities"]][[2]];
```

In[ ]:=
```
g[age_] := Values[
    c[{age, "female", Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
      Missing[], Missing[], Missing[], Missing[]}, "Probabilities"]][[2]];
```

In[•]:=
```
Show[Plot[{f[age], g[age]}, {age, 26, 95}, PlotStyle → {■, ■},
   AxesLabel → {"age", "probability of survival"}, Ticks → {Automatic, Range[0, 1, 0.1]}],
  ListPlot[Transpose[{data1[[All, 1]][[1 ;; 99]], f /@ data1[[All, 1]][[1 ;; 99]]}],
   PlotStyle → PointSize[Large]],
  ListPlot[Transpose[{data1[[All, 1]][[100 ;; 176]], f /@ data1[[All, 1]][[100 ;; 176]]}],
   PlotStyle → {PointSize[Large], Red}]]

SwatchLegend[{■, ■, ■, ■},
 {"survived", "died", "female", "male"}, LegendFunction → "Frame"]
```

Out[•]=



Out[•]=

■ survived
■ died
■ female
■ male

In[•]:=
```
ArrayReshape[{age, "male", Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[], Missing[], Missing[],
    Missing[], Missing[], Missing[], Missing[], Missing[]}, {75, 1}] // MatrixForm
```

```
( age    )
( "male" )
```

```
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
Missing[]
```

```
{{Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]},
 {Missing[]}}
```

In[ ]:=
```
Missing[]
```

Out[ ]=
```
Missing[]
```