Wolfram Engine 12.0 demo on Jupyter

created 2019 by Jürgen Kanz, www.juergen-kanz.de

In [1]:
```
Clear["Global`*"]
```

Launch Kernels for Parallel Computing

In [2]:
```
LaunchKernels[]
```

Out[2]:
```
{KernelObject[1, local], KernelObject[2, local], KernelObject[3, local], KernelObject[4, local]}
```

In [3]:
```
ParallelTable[i^2, {i, 100}]
```

Out[3]:
```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 32
4, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 108
9, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 21
16, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3
481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041,
5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889, 7056,
7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409,
9604, 9801, 10000}
```

In [4]:
```
CloseKernels[];
```

Load an Image

In [5]:
```
img=Import["e:\horse.jpg"]
```

Out[5]:

Transform horse image to zebra image

In [6]:
```
net=NetModel["CycleGAN Horse-to-Zebra Translation Trained on ImageNet \
Competition Data"]
```

Out[6]:

NetChain

Input port:	image
Output port:	image
Number of layers:	31

In [7]:
```
netEnc = NetEncoder[{"Image", ImageDimensions[img]}]
```

Out[7]:

NetEncoder

Type:	Image
Output:	array (size: 3 × 379 × 673)

In [8]:
```
resizedNet =
 NetReplacePart[
  net, {"Input" -> netEnc, "Output" -> NetDecoder[{"Image"}]}]
```

Out[8]:

NetChain

Input port:	image
Output port:	image
Number of layers:	31

In [9]:
```
resizedNet[img]
```

Out[9]:

Machine Learning: predict the propability to survive as TITANIC passenger

In [10]:
```
dataset = ExampleData[{"MachineLearning", "Titanic"}, "Data"];
c = Classify[dataset, Method -> "LogisticRegression"]
```

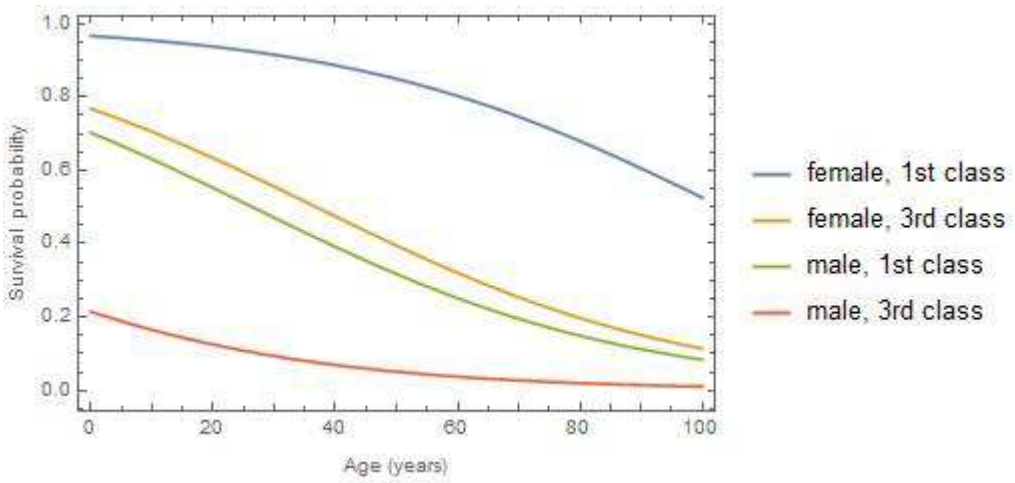
```
 | Time elapsed: | Training example used: | Current best method: | Current\

>   accuracy: | Current loss: |
 |     1.1s      |        50/1309         |  LogisticRegression  |      \

>   0.750       |     0.546     |
 |     1.7s      |       1047/1309        |  LogisticRegression  |      \

>   0.774       |     0.497     |
```

Out[10]:

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

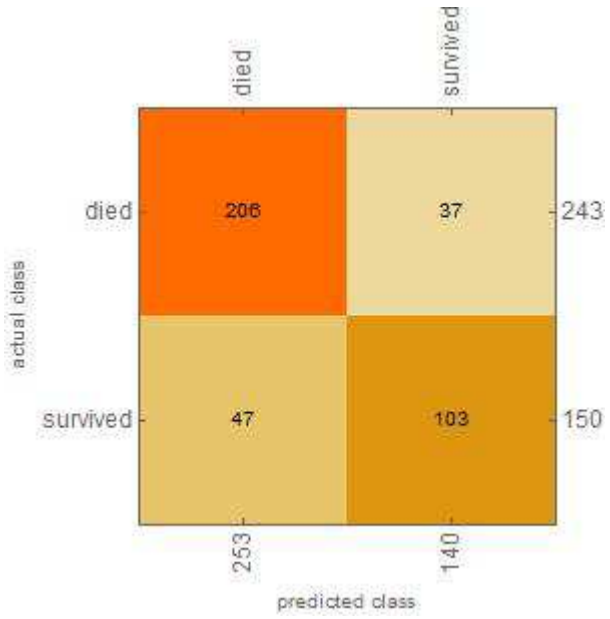
In [12]:
```
p[class_, age_, sex_] :=
  c[{class, age, sex}, {"Probability", "survived"}];
Plot[{
  p["1st", x, "female"], p["3rd", x, "female"],
  p["1st", x, "male"], p["3rd", x, "male"]},
 {x, 0, 100},
 PlotLegends -> {
   "female, 1st class", "female, 3rd class",
   "male, 1st class", "male, 3rd class"},
 Frame -> True,
 FrameLabel -> {"Age (years)", "Survival probability"}]
```

Out[12]:



In [14]:
```
cmo=ClassifierMeasurements[c,ExampleData[{"MachineLearning", "Titanic"}, "TestData"
]];
cmo["ConfusionMatrixPlot"]
```

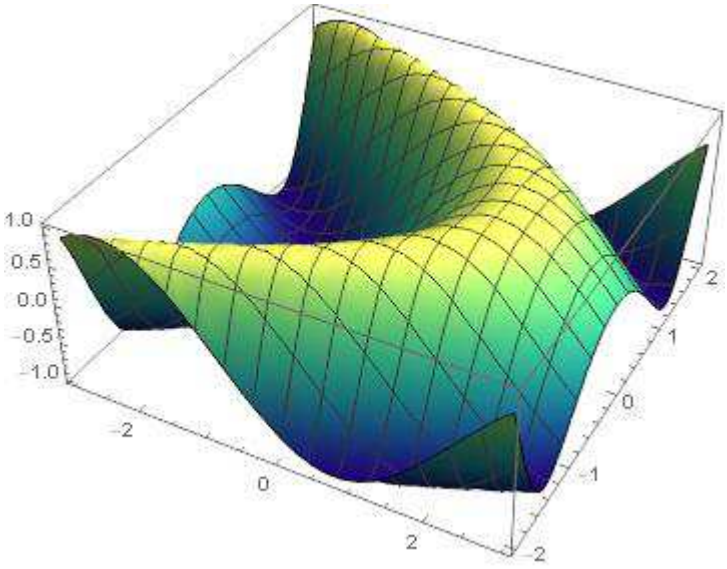
Out[14]:



Plot of 3D function

In [16]:
```
Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2},ColorFunction -> "BlueGreenYellow"]
```

Out[16]:



In [ ]: