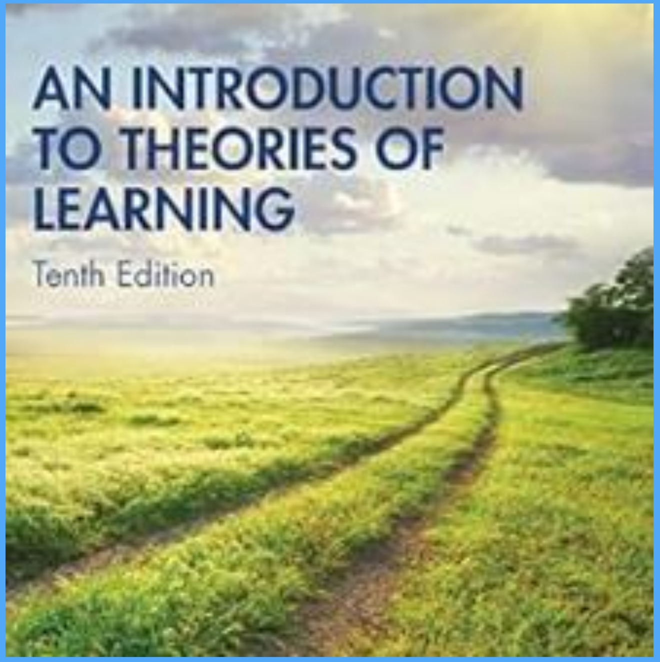
Introduction to Scientific Computation

Halil Bayraktar

Lecture 11 –Deep learning



Machine learning with Matlab

It teaches the computer to think like humans. The data is provided and interpret to build a model

Supervised learning

Classification

- Nearest Neighor
- Naïve Bayes
- Support Vector machines
- Random Forest
- Neuronal Networks

Regression

- Linear Reg
- Logistic Reg
- Gaussian model

Unsupervised learning

- Kmeans
- Hidden markov model
- Hieracrhical model

Regression

It is used to predict continuous values

- Linear
- Logistic models

Examples,

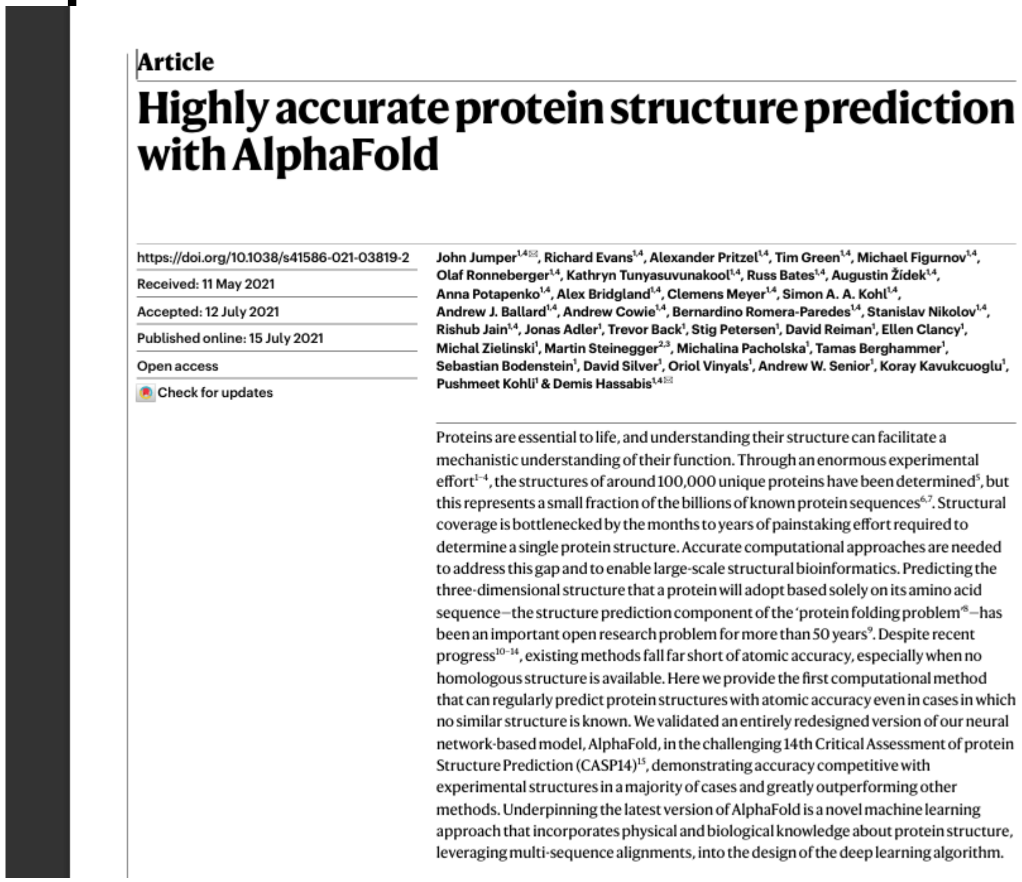
- Predict if a person have a risk of disease
- The price of an apartment
- The expression levels of a protein

Classification

- Used to predict the label of given data,
- Single-class or multi-class models

- What is the label of a given handwritten number?
- Is the good healthy or not?
- Dou you get a low and high grade?

Machine learning for biology
Alphafold



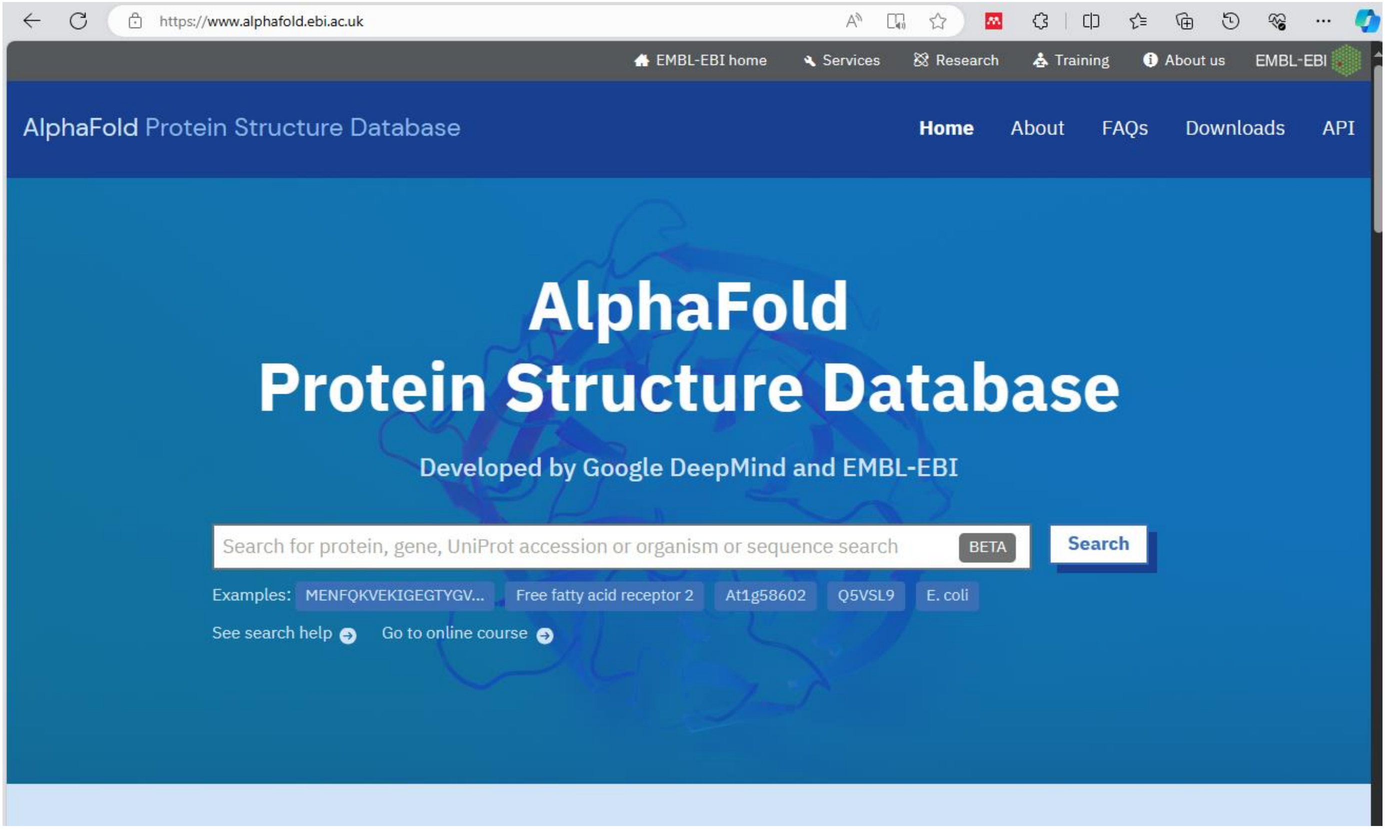
Article

Highly accurate protein structure prediction with AlphaFold

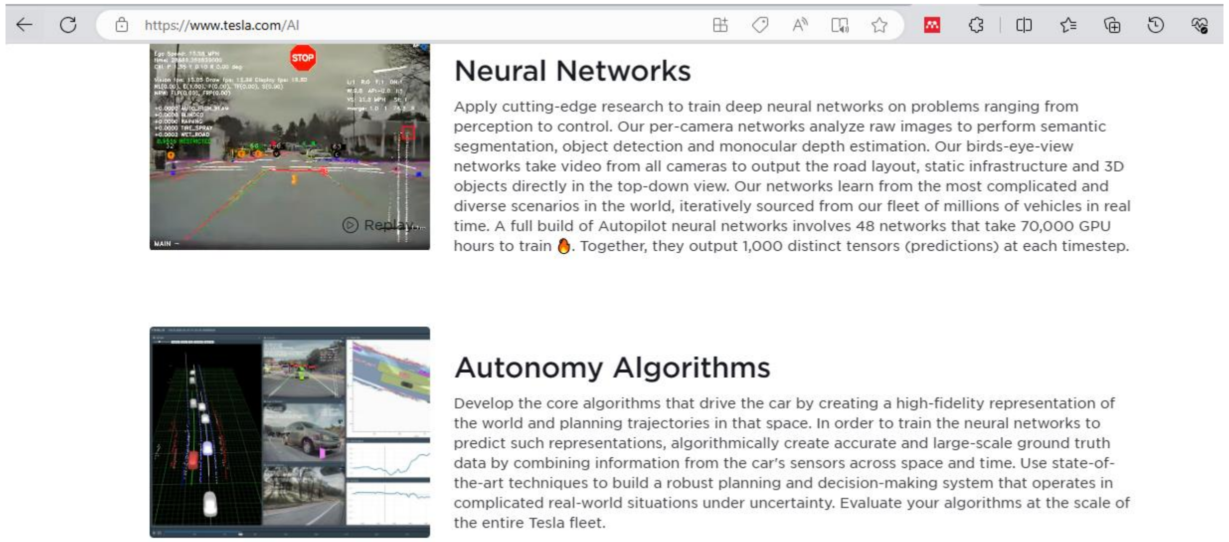
https://doi.org/10.1038/s41586-021-03819-2
Received: 11 May 2021
Accepted: 12 July 2021
Published online: 15 July 2021
Open access
Check for updates

John Jumper[1,4✉], Richard Evans[1,4], Alexander Pritzel[1,4], Tim Green[1,4], Michael Figurnov[1,4], Olaf Ronneberger[1,4], Kathryn Tunyasuvunakool[1,4], Russ Bates[1,4], Augustin Žídek[1,4], Anna Potapenko[1,4], Alex Bridgland[1,4], Clemens Meyer[1,4], Simon A. A. Kohl[1,4], Andrew J. Ballard[1,4], Andrew Cowie[1,4], Bernardino Romera-Paredes[1,4], Stanislav Nikolov[1,4], Rishub Jain[1,4], Jonas Adler[1], Trevor Back[1], Stig Petersen[1], David Reiman[1], Ellen Clancy[1], Michal Zielinski[1], Martin Steinegger[2,3], Michalina Pacholska[1], Tamas Berghammer[1], Sebastian Bodenstein[1], David Silver[1], Oriol Vinyals[1], Andrew W. Senior[1], Koray Kavukcuoglu[1], Pushmeet Kohli[1] & Demis Hassabis[1,4✉]

Proteins are essential to life, and understanding their structure can facilitate a mechanistic understanding of their function. Through an enormous experimental effort[1–4], the structures of around 100,000 unique proteins have been determined[5], but this represents a small fraction of the billions of known protein sequences[6,7]. Structural coverage is bottlenecked by the months to years of painstaking effort required to determine a single protein structure. Accurate computational approaches are needed to address this gap and to enable large-scale structural bioinformatics. Predicting the three-dimensional structure that a protein will adopt based solely on its amino acid sequence—the structure prediction component of the 'protein folding problem'[8]—has been an important open research problem for more than 50 years[9]. Despite recent progress[10–14], existing methods fall far short of atomic accuracy, especially when no homologous structure is available. Here we provide the first computational method that can regularly predict protein structures with atomic accuracy even in cases in which no similar structure is known. We validated an entirely redesigned version of our neural network-based model, AlphaFold, in the challenging 14th Critical Assessment of protein Structure Prediction (CASP14)[15], demonstrating accuracy competitive with experimental structures in a majority of cases and greatly outperforming other methods. Underpinning the latest version of AlphaFold is a novel machine learning approach that incorporates physical and biological knowledge about protein structure, leveraging multi-sequence alignments, into the design of the deep learning algorithm.



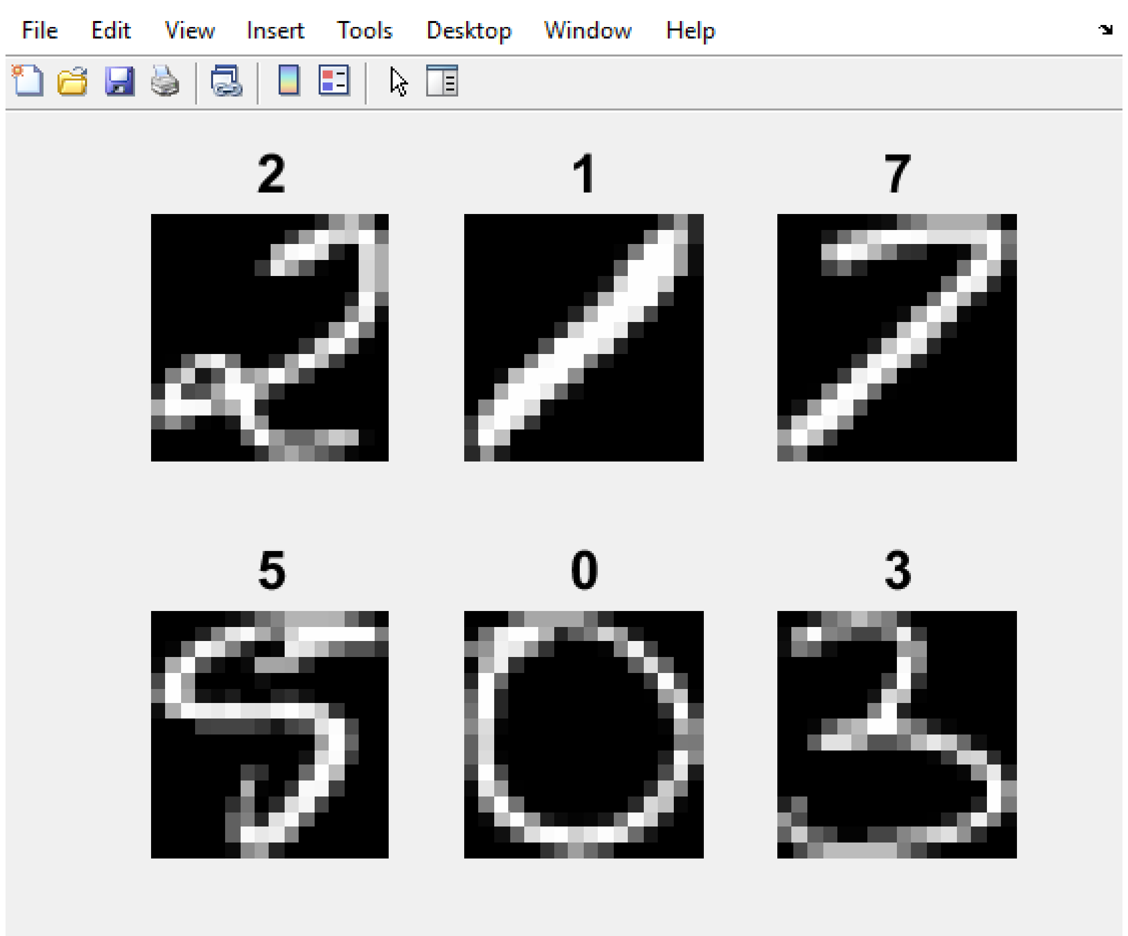
Machine learning for self driving cars

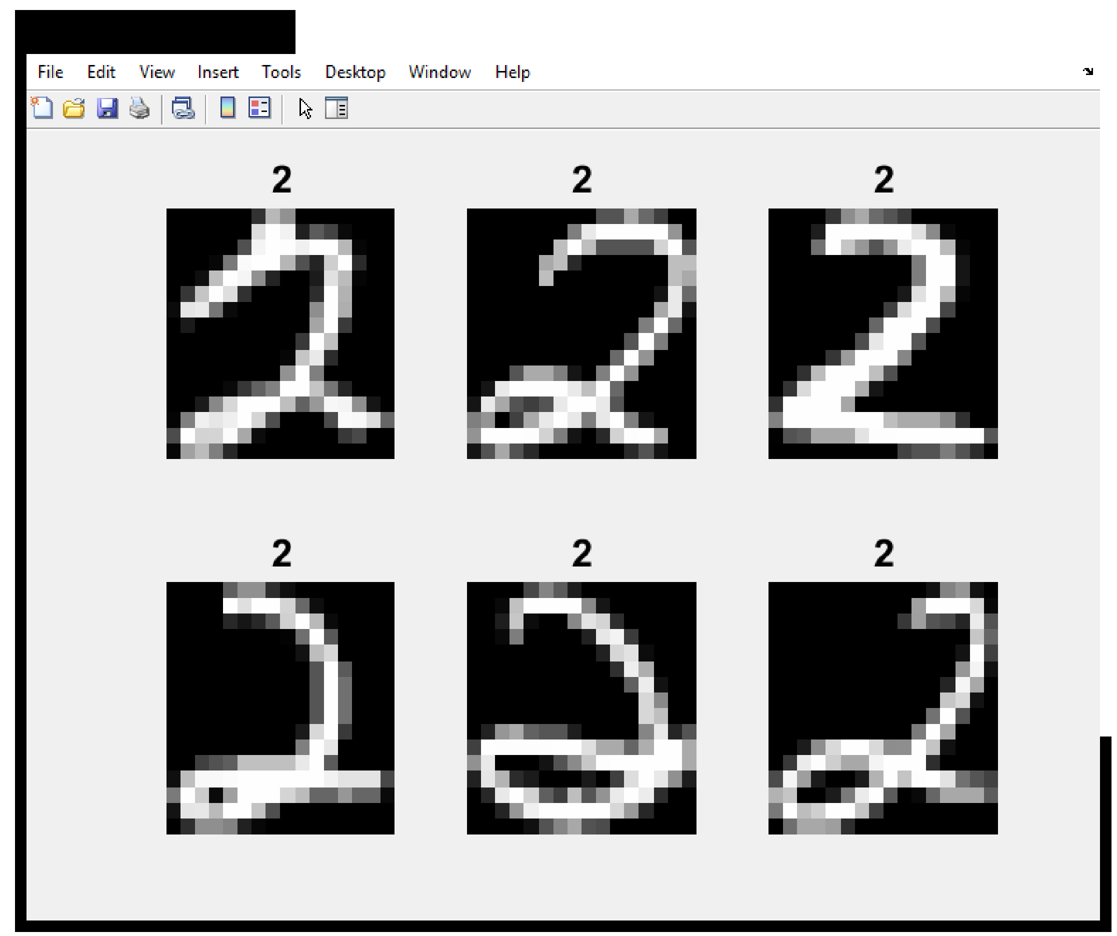


Example: Hand written recognition

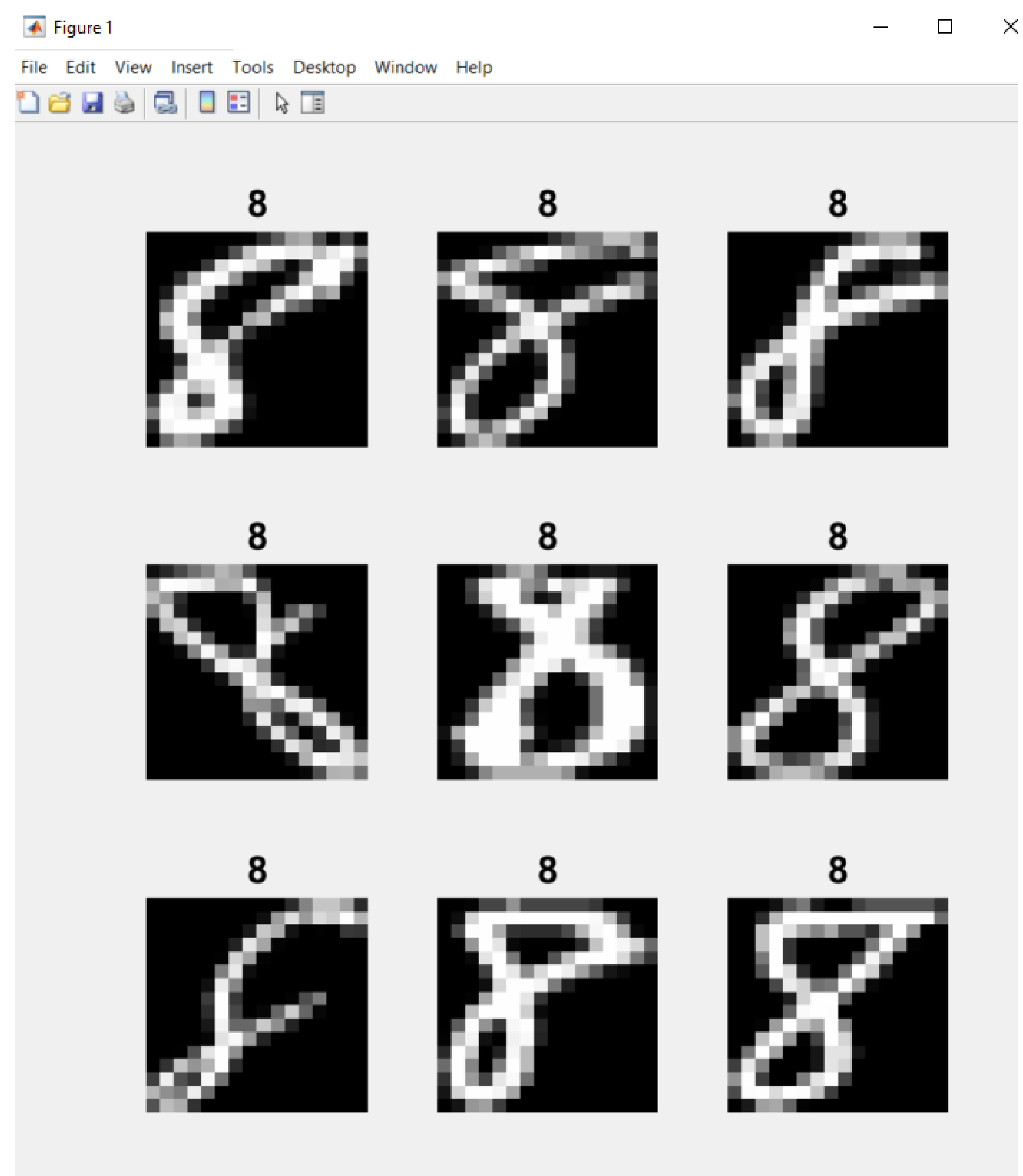
Classic problem in machine learning

Problem: Can we teach the computer to read the hand written digits ?

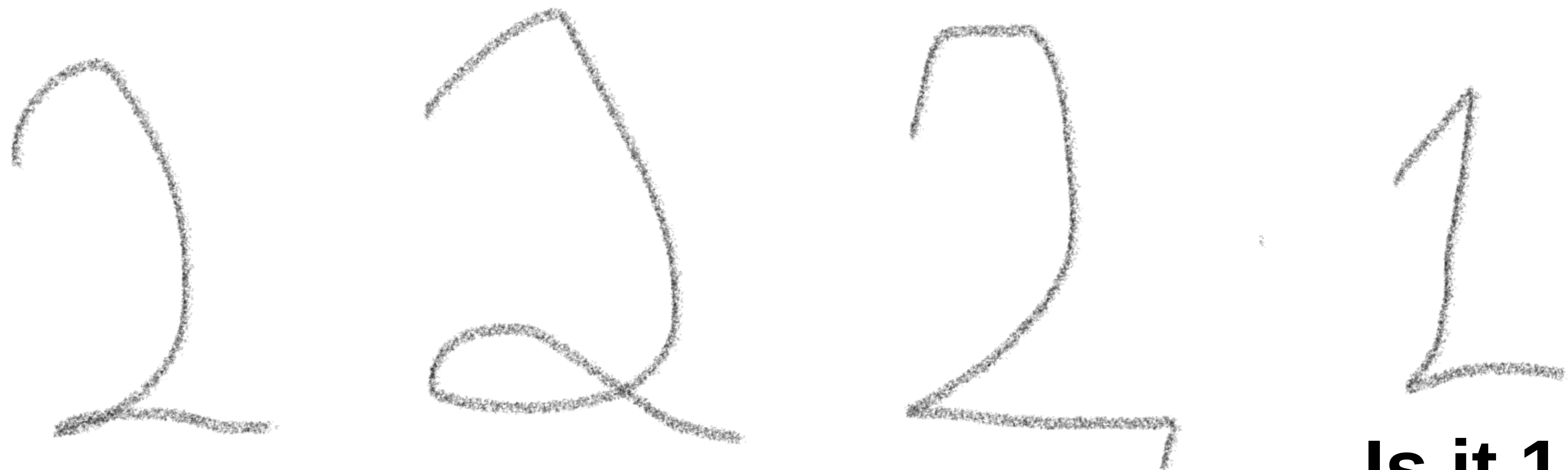




```matlab
%%
% Randomly pick the images and their labels
for ii = 1:6
    subplot(2,3,ii)
    rand_num = randperm(11000,1);
    imshow(alldigitaldata{rand_num,1},[])
    title((y(rand_num)),'FontSize',20)
    axis off
end



%%
% lets pick the selected values
% here we find the index of selelected values
selectnumber=8
[r,c,logic]=find(y==selectnumber)

for i = 1:9
    subplot(3,3,i)
    selectnum=randi([1,1100],1,1)

    image(reshape(X(r(selectnum),:),16,16))

    title(string(selectnumber),'FontSize',20)
    axis off
end
```



Can you predict the following hand written digit? Is it 1 or 2?



Is it 1 or 2?

Labels

2 2 **2**

As we humans, computers also make mistakes!
How to reduce error rate?

1. Use many training samples
2. Use many features

Step 1:
Convert the images into a linear form

11000x256 double

	70	71	72	73	74	75	76	77	78	79
1	0	0	0	39	216	255	245	98	3	0
2	0	0	0	117	255	255	255	255	255	255
3	0	0	0	0	0	27	231	255	255	114
4	0	0	0	0	5	75	238	255	250	222
5	0	0	0	0	11	215	224	40	0	0
6	0	0	0	0	93	255	255	255	231	69
7	0	0	64	103	255	255	255	255	255	255
8	0	0	0	0	0	54	226	255	255	255
9	0	0	0	0	0	99	255	255	194	9
10	0	0	0	0	71	235	234	16	0	158
11	0	0	0	0	19	163	252	255	229	70
12	0	0	0	0	0	0	212	255	255	255
13	0	0	0	0	0	48	230	255	254	112
14	0	0	0	0	16	210	255	249	129	0
15	0	0	0	16	154	255	255	156	13	0
16	0	0	0	0	0	72	250	90	0	0
17	0	0	0	0	17	218	255	255	91	0
18	0	0	0	0	131	255	255	253	160	16
19	255	255	255	255	255	255	255	255	255	249
20	0	106	222	255	255	255	255	255	255	72
21	0	0	0	0	67	214	229	91	0	0
22	0	0	0	99	229	255	255	255	255	178
23	0	68	189	255	255	255	255	255	255	255
24	0	0	0	0	131	255	255	222	55	0
25	255	255	255	221	162	162	83	0	0	0

```
%%
alldigilinear=zeros(11000,256)
for i=1:11000;
  alldigilinear(i,1:256) =reshape(alldigitaldata{i,:},1,256);
end


%%
X=alldigilinear
cv = cvpartition(y, 'holdout', .5);
Xtrain = X(cv.training,:);
Ytrain = y(cv.training,1);
Xtest = X(cv.test,:);
Ytest = y(cv.test,1);
```

Step 2:
Separate data into test and test set

Step 2:
Separate data into train and test set

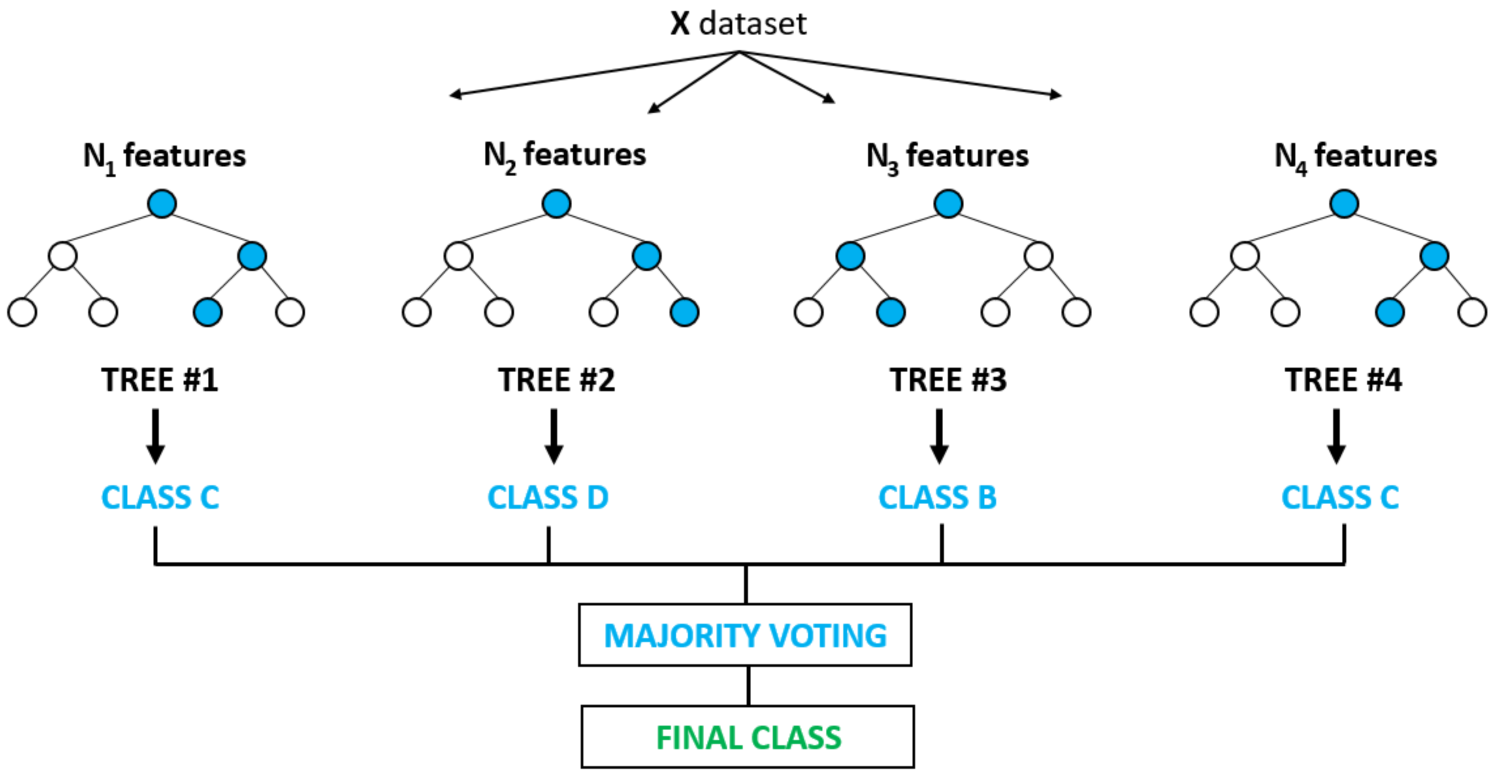
```
end


%%
X=alldigilinear
cv = cvpartition(y, 'holdout', .5);
Xtrain = X(cv.training,:);
Ytrain = y(cv.training,1);
Xtest = X(cv.test,:);
Ytest = y(cv.test,1);
```

Xtest	double	*5500x256 double*
Xtrain	double	*5500x256 double*
y	double	*11000x1 double*
ylabel	double	[1,2,3,4,5,6,7,8,9,0]
ypred	double	*5500x1 double*
Ytest	double	*5500x1 double*
Ytrain	double	*5500x1 double*

Classification Tree

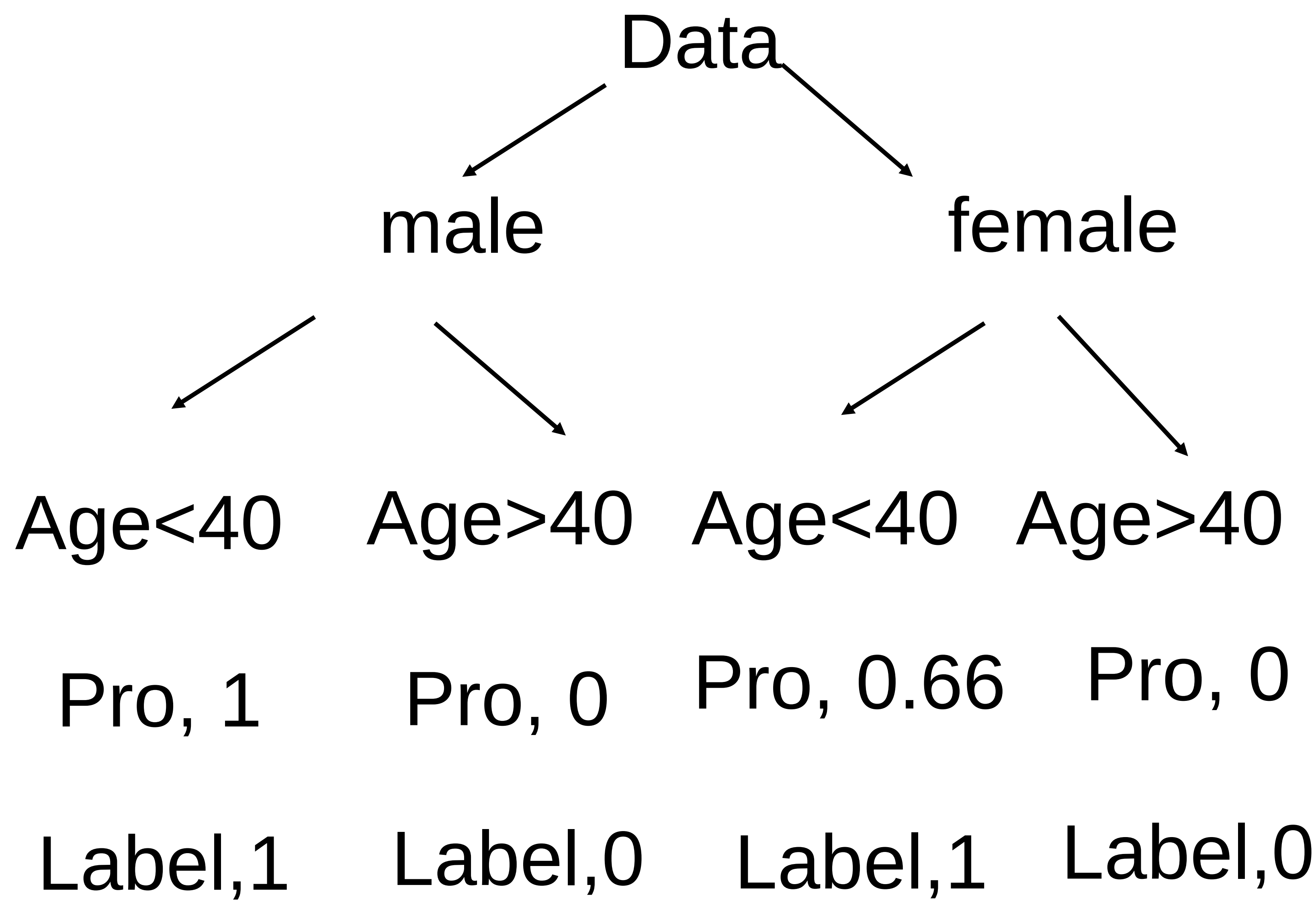
- Used for multiclass classification.
- It is an iterative process for splitting data into partitions and split them further into branches

- The method based on finding features that splits data.
- We create a model that predicts the label of a target variable by learning decision rules extracted from the data features.



Build a simple Classification Tree for fail or pass the course

people	gender	Age <40		Pass or fail
1	1	1		1
2	1	1		1
3	1	0		0
4	1	1		0
5	0	1		1
6	0	0		0
7	0	1		1
8	0	0		0



Feature1:
Female=1
Male=0

Feature 2:
Age<40=1
Age>40=0

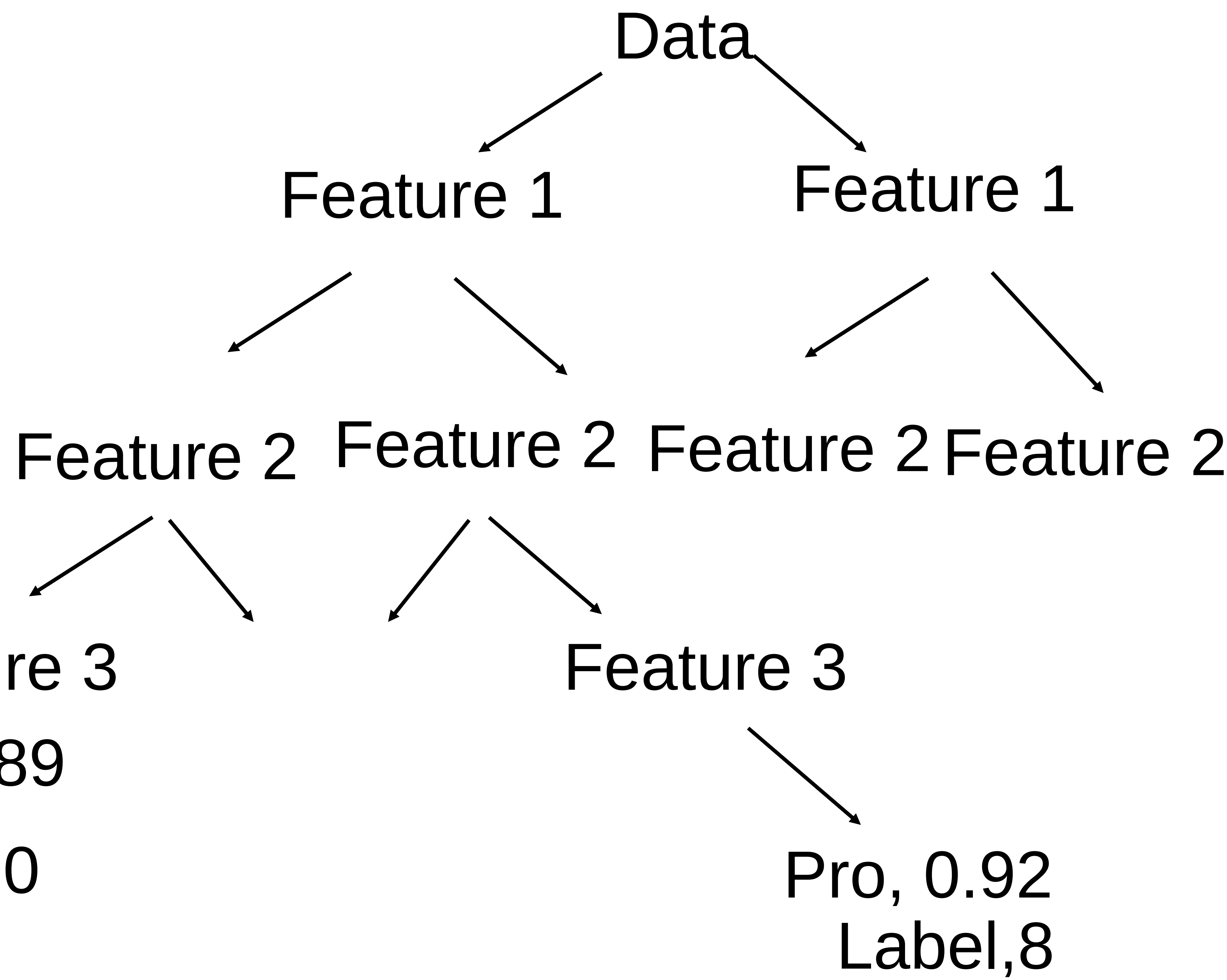
Test samples: a) male, age>24
b) Female, age

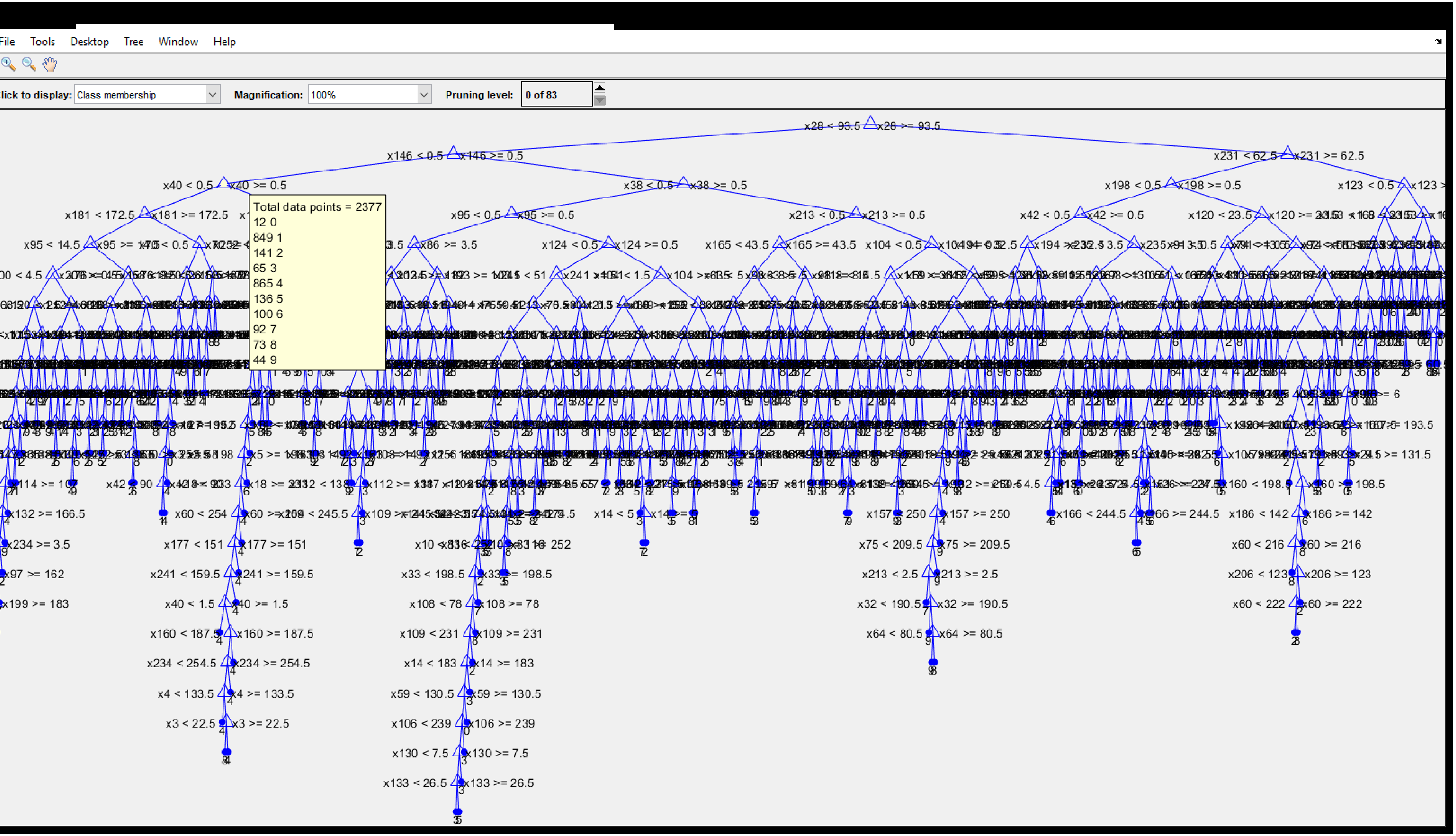
Features

11000x256 double

	70	71	72	73	74	75	76	77	78	79
1	0	0	0	39	216	255	245	98	3	0
2	0	0	0	117	255	255	255	255	255	255
3	0	0	0	0	0	27	231	255	255	114
4	0	0	0	0	5	75	238	255	250	222
5	0	0	0	0	11	215	224	40	0	0
6	0	0	0	0	93	255	255	255	231	69
7	0	0	64	103	255	255	255	255	255	255
8	0	0	0	0	0	54	226	255	255	255
9	0	0	0	0	0	99	255	255	194	9
10	0	0	0	0	71	235	234	16	0	158
11	0	0	0	0	19	163	252	255	229	70
12	0	0	0	0	0	0	212	255	255	255
13	0	0	0	0	0	48	230	255	254	112
14	0	0	0	0	16	210	255	249	129	0
15	0	0	0	16	154	255	255	156	13	0
16	0	0	0	0	0	72	250	90	0	0
17	0	0	0	0	17	218	255	255	91	0
18	0	0	0	0	131	255	255	253	160	16
19	255	255	255	255	255	255	255	255	255	249
20	0	106	222	255	255	255	255	255	255	72
21	0	0	0	0	67	214	229	91	0	0
22	0	0	0	99	229	255	255	255	255	178
23	0	68	189	255	255	255	255	255	255	255
24	0	0	0	0	131	255	255	222	55	0
25	255	255	255	221	162	162	83	0	0	0

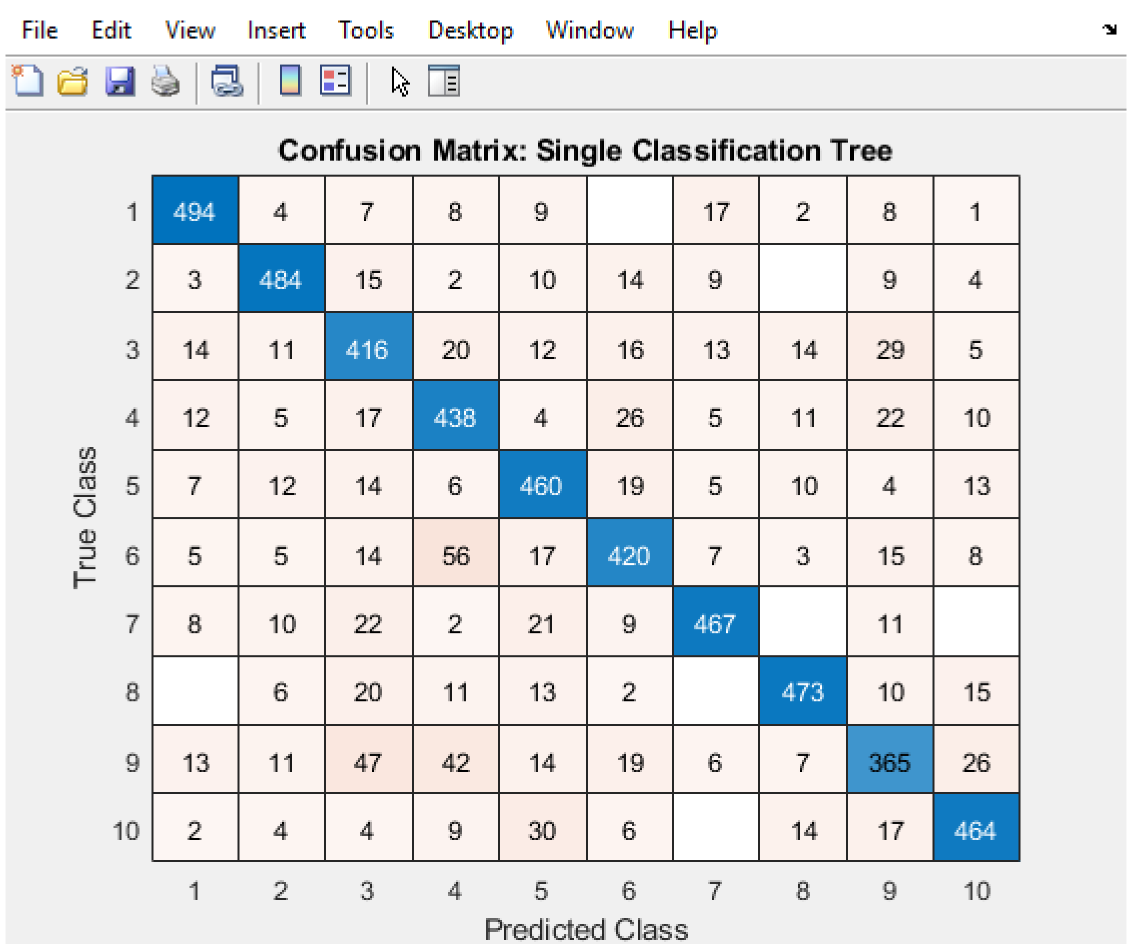
Command History



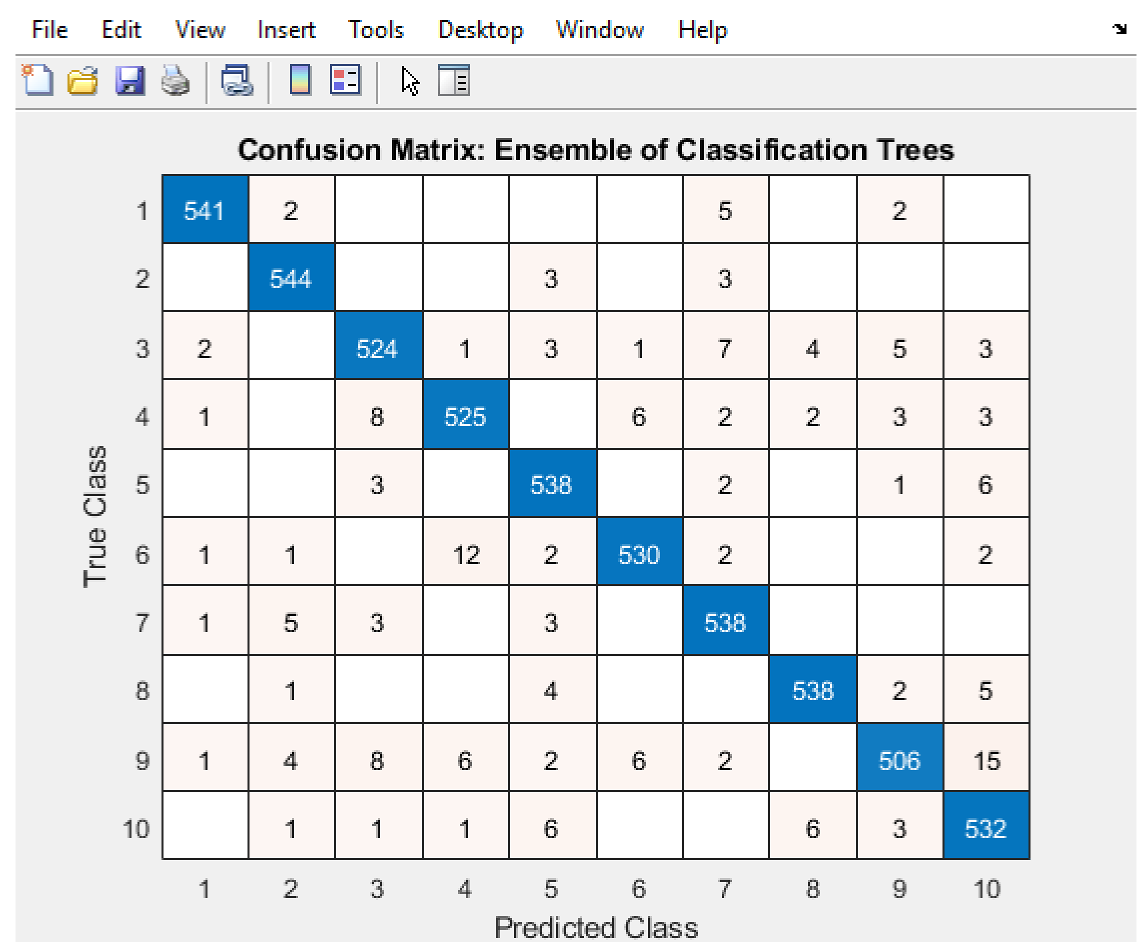

File Tools Desktop Tree Window Help
Click to display: Class membership
Magnification: 100%
Pruning level: 0 of 83
x28 < 93.5
x28 >= 93.5
x146 < 0.5
x146 >= 0.5
x231 < 62.5
x231 >= 62.5
x40 < 0.5
x40 >= 0.5
x38 < 0.5
x38 >= 0.5
x198 < 0.5
x198 >= 0.5
Total data points = 2377
12 0
849 1
141 2
65 3
865 4
136 5
100 6
92 7
73 8
44 9

Compare predicted and true labels

```
%%
% Train and Predict Using a Single Classification Tree
mdl_ctree = ClassificationTree.fit(Xtrain,Ytrain);
ypred = predict(mdl_ctree,Xtest);
Confmat_ctree = confusionmat(Ytest,ypred);

%
%Train and Predict Using Bagged Decision Trees
mdl = fitensemble(Xtrain,Ytrain,'bag',200,'tree','type','Classification');
ypred = predict(mdl,Xtest);
Confmat_bag = confusionmat(Ytest,ypred);
```



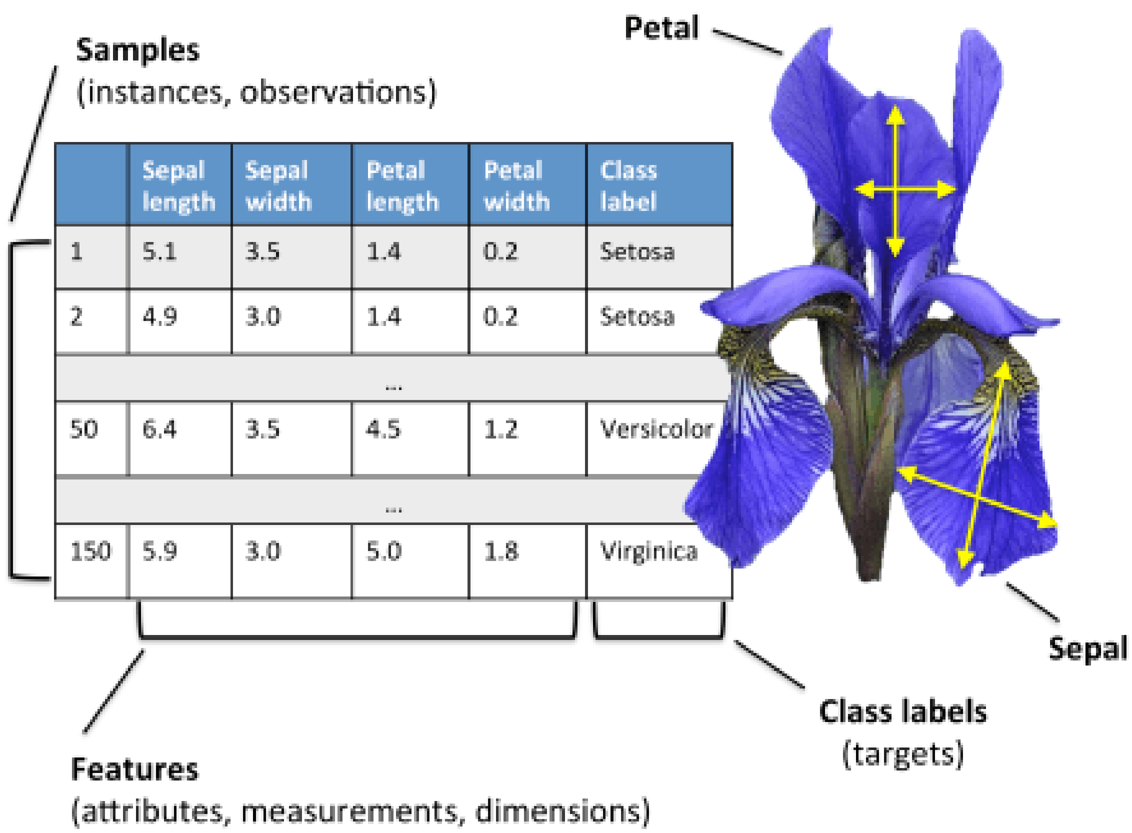
True Class \ Predicted Class	1	2	3	4	5	6	7	8	9	10
1	494	4	7	8	9		17	2	8	1
2	3	484	15	2	10	14	9		9	4
3	14	11	416	20	12	16	13	14	29	5
4	12	5	17	438	4	26	5	11	22	10
5	7	12	14	6	460	19	5	10	4	13
6	5	5	14	56	17	420	7	3	15	8
7	8	10	22	2	21	9	467		11	
8		6	20	11	13	2		473	10	15
9	13	11	47	42	14	19	6	7	365	26
10	2	4	4	9	30	6		14	17	464

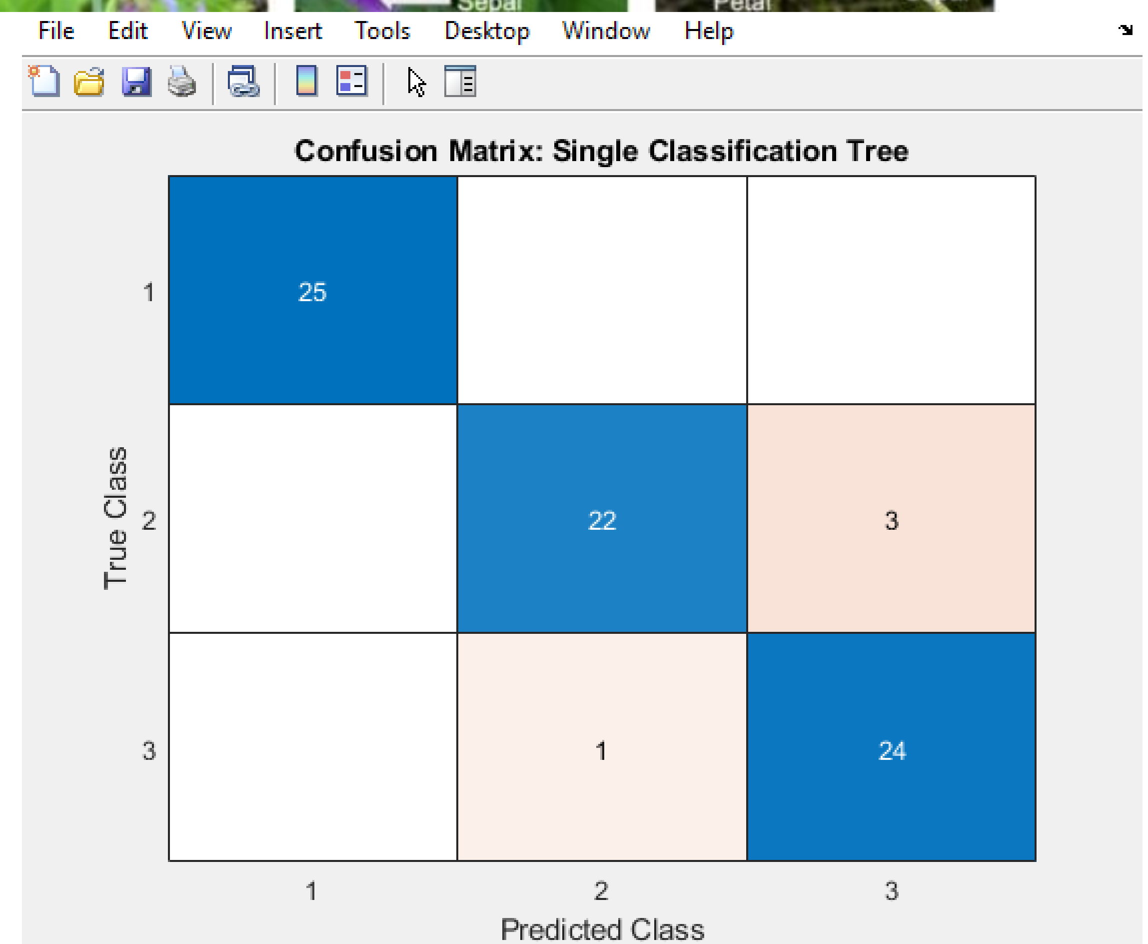


True Class \ Predicted Class	1	2	3	4	5	6	7	8	9	10
1	541	2					5		2	
2		544			3		3			
3	2		524	1	3	1	7	4	5	3
4	1		8	525		6	2	2	3	3
5			3		538		2		1	6
6	1	1		12	2	530	2			2
7	1	5	3		3		538			
8		1			4			538	2	5
9	1	4	8	6	2	6	2		506	15
10		1	1	1	6			6	3	532

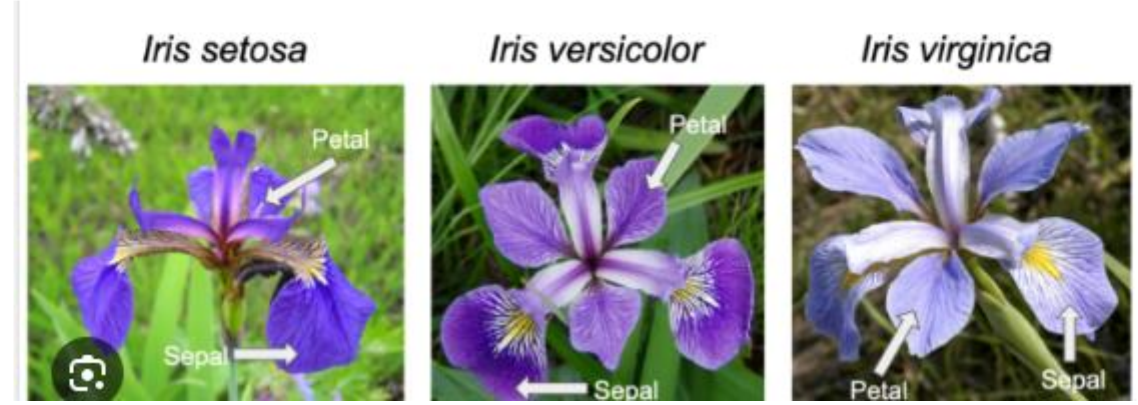
Other examples for decision tree

- A decision tree is a set of simple rules, for example if the sepal length is less than 5.00, classify the specimen as setosa.
- Decision trees are nonparametric model because they do not require any assumptions about the distribution of the variables in each class.





Data:



Variables - meas

species | meas

150x4 double

	1	2	3	4	5	6	7	8	9
1	5.1000	3.5000	1.4000	0.2000					
2	4.9000	3	1.4000	0.2000					
3	4.7000	3.2000	1.3000	0.2000					
4	4.6000	3.1000	1.5000	0.2000					
5	5	3.6000	1.4000	0.2000					
6	5.4000	3.9000	1.7000	0.4000					
7	4.6000	3.4000	1.4000	0.3000					
8	5	3.4000	1.5000	0.2000					
9	4.4000	2.9000	1.4000	0.2000					
10	4.9000	3.1000	1.5000	0.1000					
11	5.4000	3.7000	1.5000	0.2000					
12	4.8000	3.4000	1.6000	0.2000					
13	4.8000	3	1.4000	0.1000					
14	4.3000	3	1.1000	0.1000					
15	5.8000	4	1.2000	0.2000					
16	5.7000	4.4000	1.5000	0.4000					
17	5.4000	3.9000	1.3000	0.4000					
18	5.1000	3.5000	1.4000	0.3000					
19	5.7000	3.8000	1.7000	0.3000					
20	5.1000	3.8000	1.5000	0.3000					
21	5.4000	3.4000	1.7000	0.2000					
22	5.1000	3.7000	1.5000	0.4000					
23	4.6000	3.6000	1	0.2000					
24	5.1000	3.3000	1.7000	0.5000					
25	4.8000	3.4000	1.9000	0.2000					
26	5	3	1.6000	0.2000					
27	5	3.4000	1.6000	0.4000					

Command History

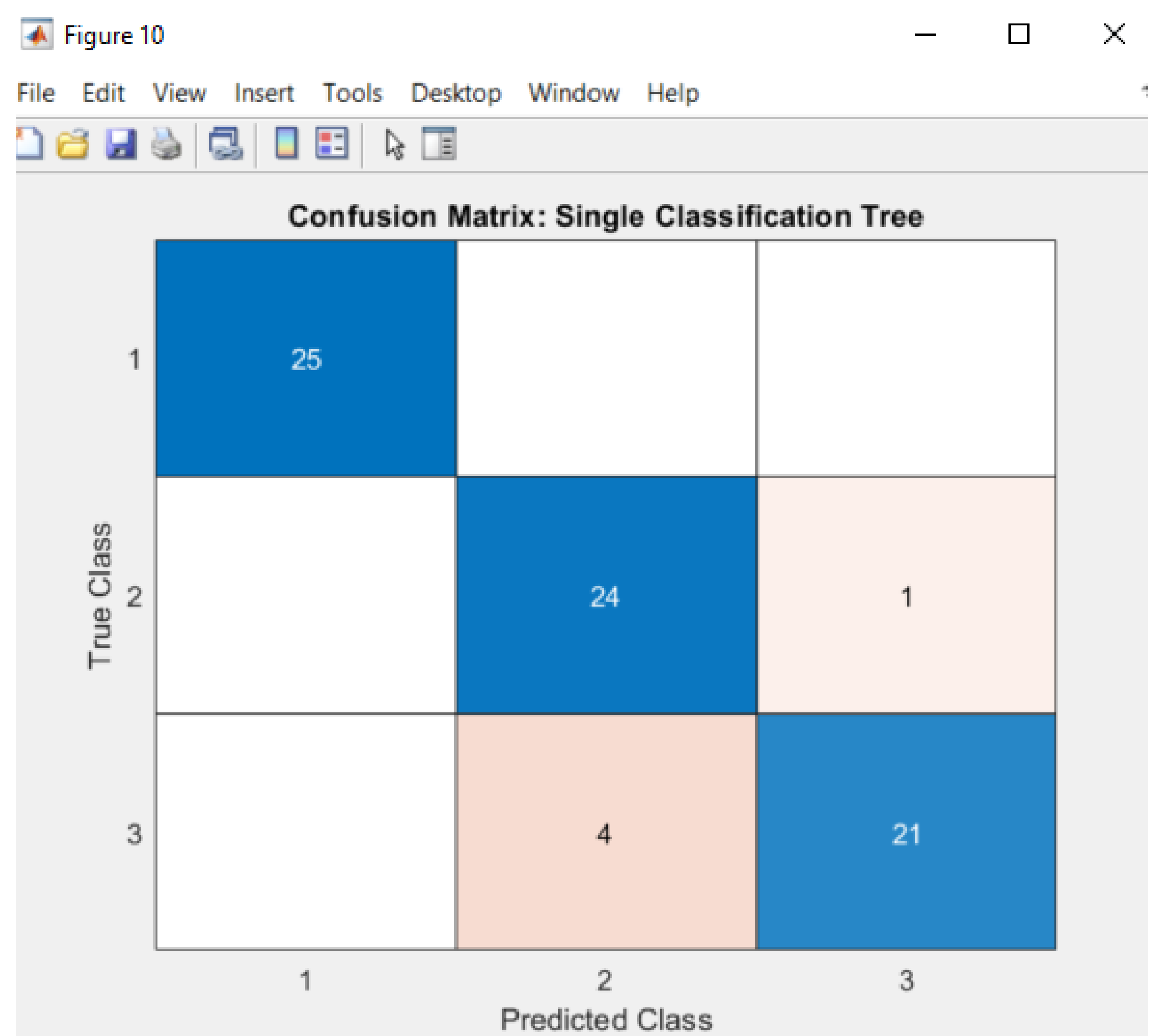
Variables - species

species

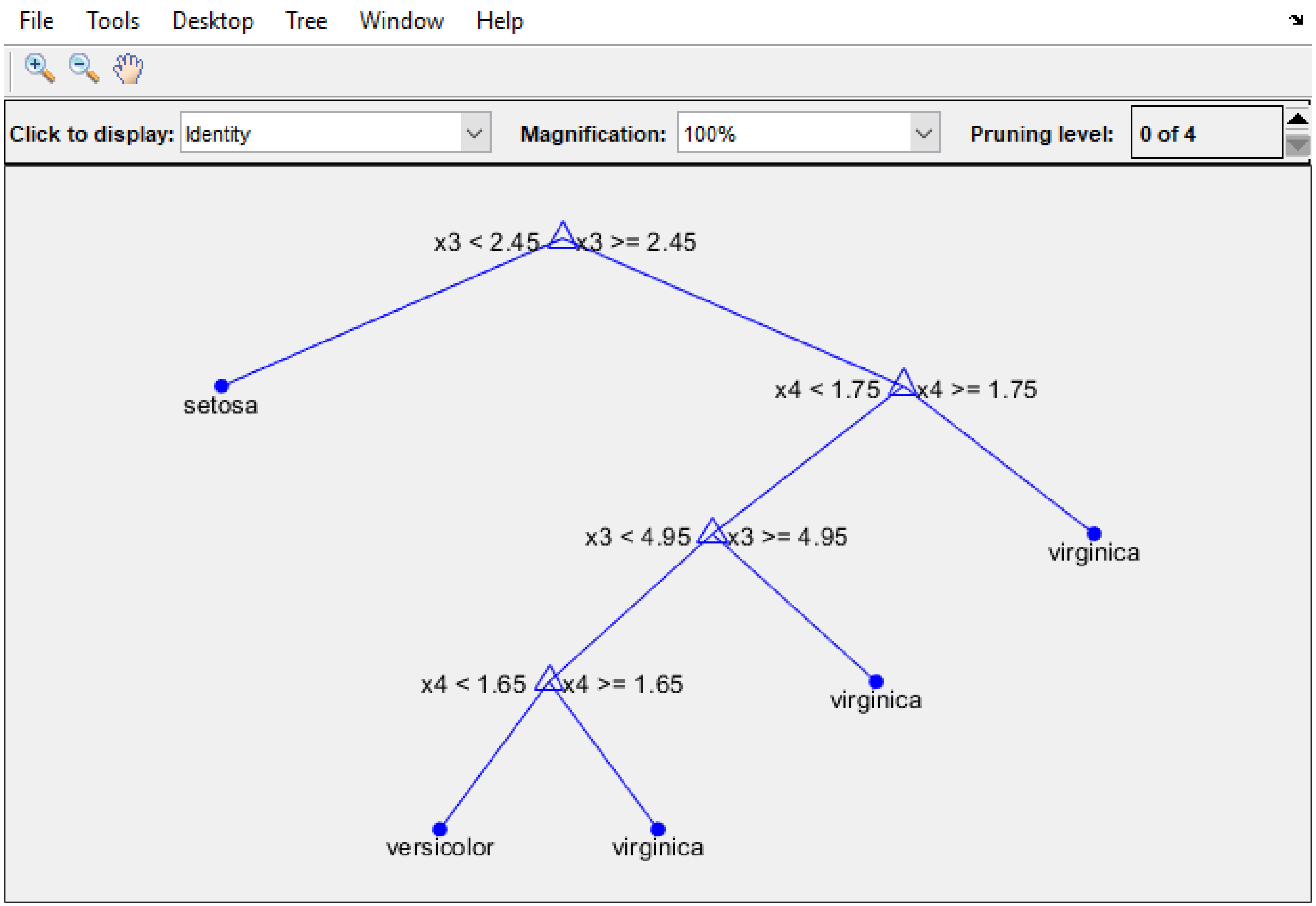
150x1 cell

	1	2	3	4	5
1	setosa				
2	setosa				
3	setosa				
4	setosa				
5	setosa				
6	setosa				
7	setosa				
8	setosa				
9	setosa				
10	setosa				
11	setosa				
12	setosa				
13	setosa				
14	setosa				
15	setosa				
16	setosa				
17	setosa				
18	setosa				
19	setosa				
20	setosa				
21	setosa				
22	setosa				
23	setosa				
24	setosa				
25	setosa				
26	setosa				
27	setosa				

```
view(ctree,'mode','graph')
%%

cv = cvpartition(species, 'holdout', .50);
Xmeastrain = meas(cv.training,:);
Ymeastrain = species(cv.training,1);
Xmeastest = meas(cv.test,:);
Ymeastest = species(cv.test,1);

mdl_ctree = ClassificationTree.fit(Xmeastrain,Ymeastrain);
ypred = predict(mdl_ctree,Xmeastest);
Confmat_ctree = confusionmat(Ymeastest,ypred);

%Train and Predict Using Bagged Decision Trees
mdl = fitensemble(Xmeastrain,Ymeastrain,'bag',200,'tree','type','Classification');
ypred = predict(mdl,Xmeastest);
Confmat_bag = confusionmat(Ymeastest,ypred);


figure(10)
confusionchart(Confmat_ctree)
title('Confusion Matrix: Single Classification Tree')
figure(11)
confusionchart(Confmat_bag)
title('Confusion Matrix: Ensemble of Classification Trees')
```




Iris setosa
Iris versicolor
Iris virginica
Petal
Sepal
Petal
Sepal
Petal
Sepal


File
Tools
Desktop
Tree
Window
Help
Click to display:
Identity
Magnification:
100%
Pruning level:
0 of 4
x3 < 2.45
x3 >= 2.45
setosa
x4 < 1.75
x4 >= 1.75
x3 < 4.95
x3 >= 4.95
virginica
x4 < 1.65
x4 >= 1.65
virginica
versicolor
virginica

ELSEVIER

Computers in Biology and Medicine
Volume 150, November 2022, 106193

Adaptive tracking algorithm for trajectory analysis of cells and layer-by-layer assessment of motility dynamics

Mohammad Haroon Qureshi [a b], Nurhan Ozlu [a], Halil Bayraktar [c]

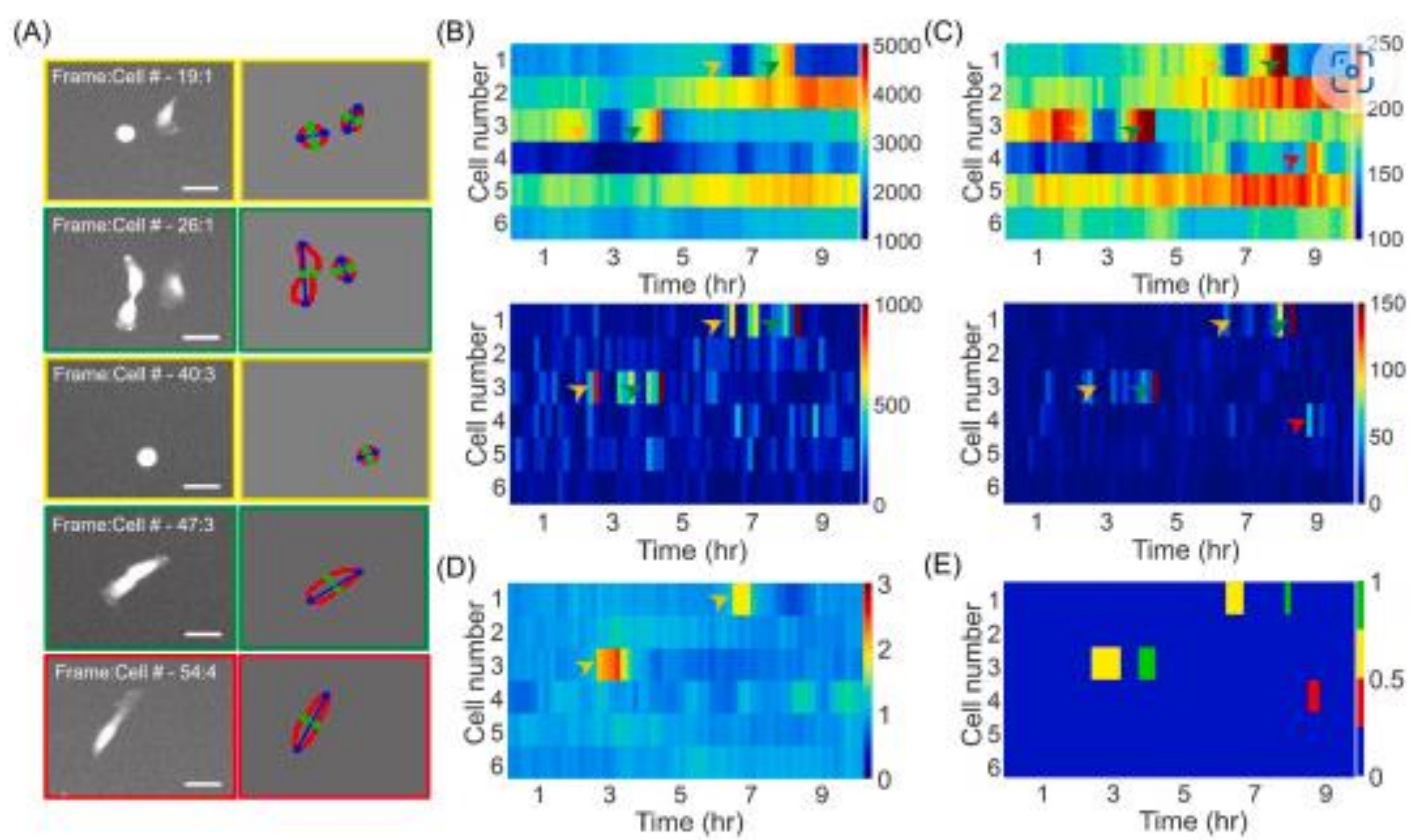
Show more

Add to Mendeley Share Cite

https://doi.org/10.1016/j.compbiomed.2022.106193

Abstract

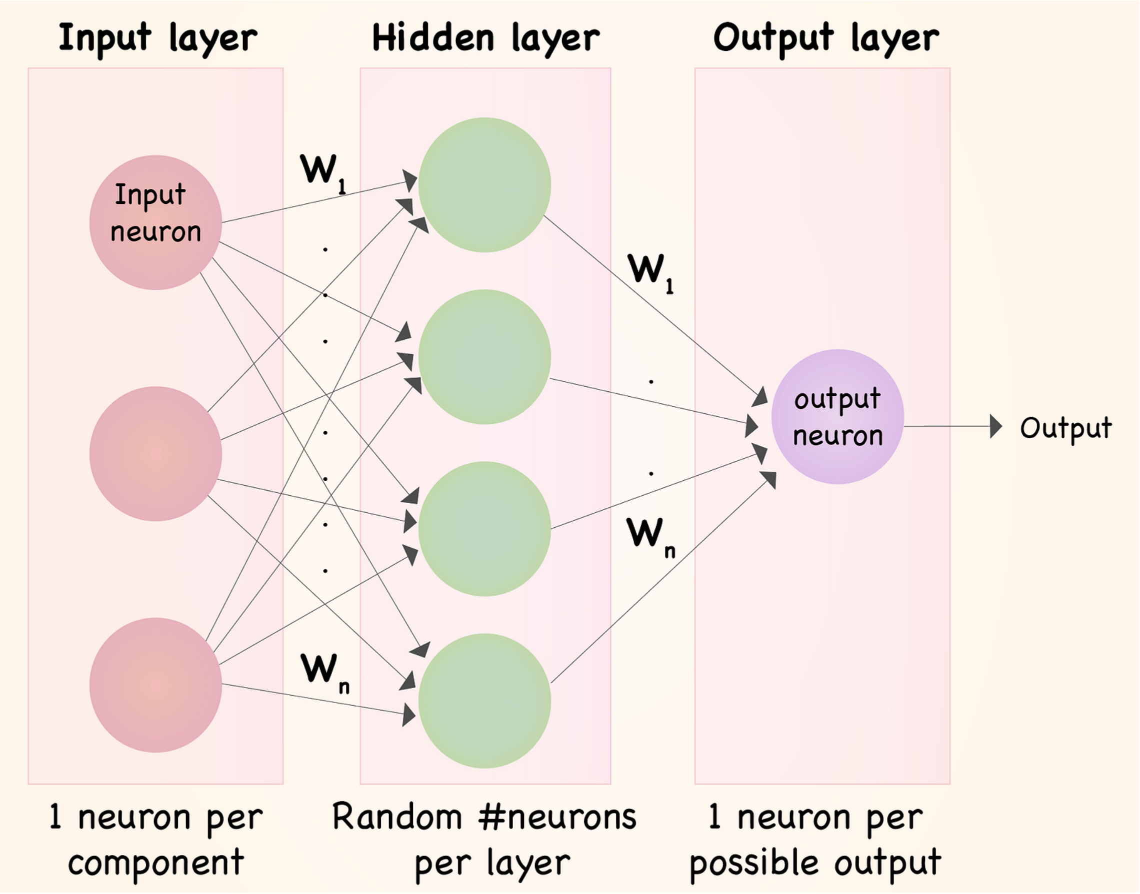
Tracking biological objects such as cells or subcellular comp
with time-lapse microscopy enables us to understand the m
about the dynamics of cell behaviors. However, automatic o
segmentation and extracting trajectories remain as a rate-li
intrinsic challenges of video processing. This paper present
tracking algorithm (Adtari) that automatically finds the opti



Deep learning

- Based on the principles of learning
- Composed of linked neurons
- İncludes input, hidden and output layers

Represented as network diagrams



Applications

- Medical diagnosis (flu, cold, bacterial)
- Fraud detection in banking (valid or fraud transactions)
- Image classification (cat, dog, cow,…)
- Drug discovery (inhibitor or not)
- Chemical synthesis (route selection or organic synthesis)
- Genome analysis (cancer risk or not)
- Spam filter (spam email or normal)
- Language models, (What does it say?)

- Suppose we want to classify the students if they have low 0 or high grade 1.

features

diet Study/ week Sports Income Living nearby Homeworks

labels

AA,BA,BB, CB 1

Less than CB 0

observations

1									0
2									1
3									0
4									1
5									1
6									1
7									1
8									0

Label={l1,… ln}

Label={0,1}

Label={1,2,3,4,5,6,7,8,9,0}

Label={Turkiye, France,…}

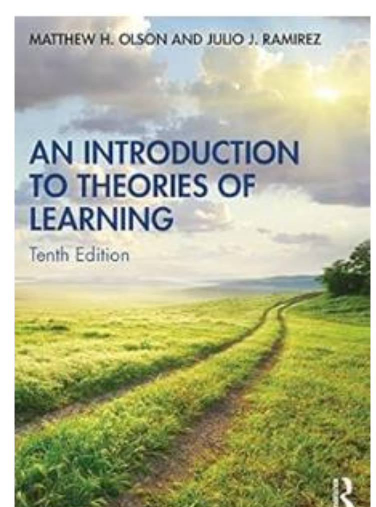
- Binary or Boolean classification when labels=2
- Multi-class classification when label>2

Another example,

Is it cat or not?

How do we learn objects at early age?

- Learning theory states that as we learn things it strengths the link between neurons.
- Deep learning was inspired from this principle that help us to learn things around us.



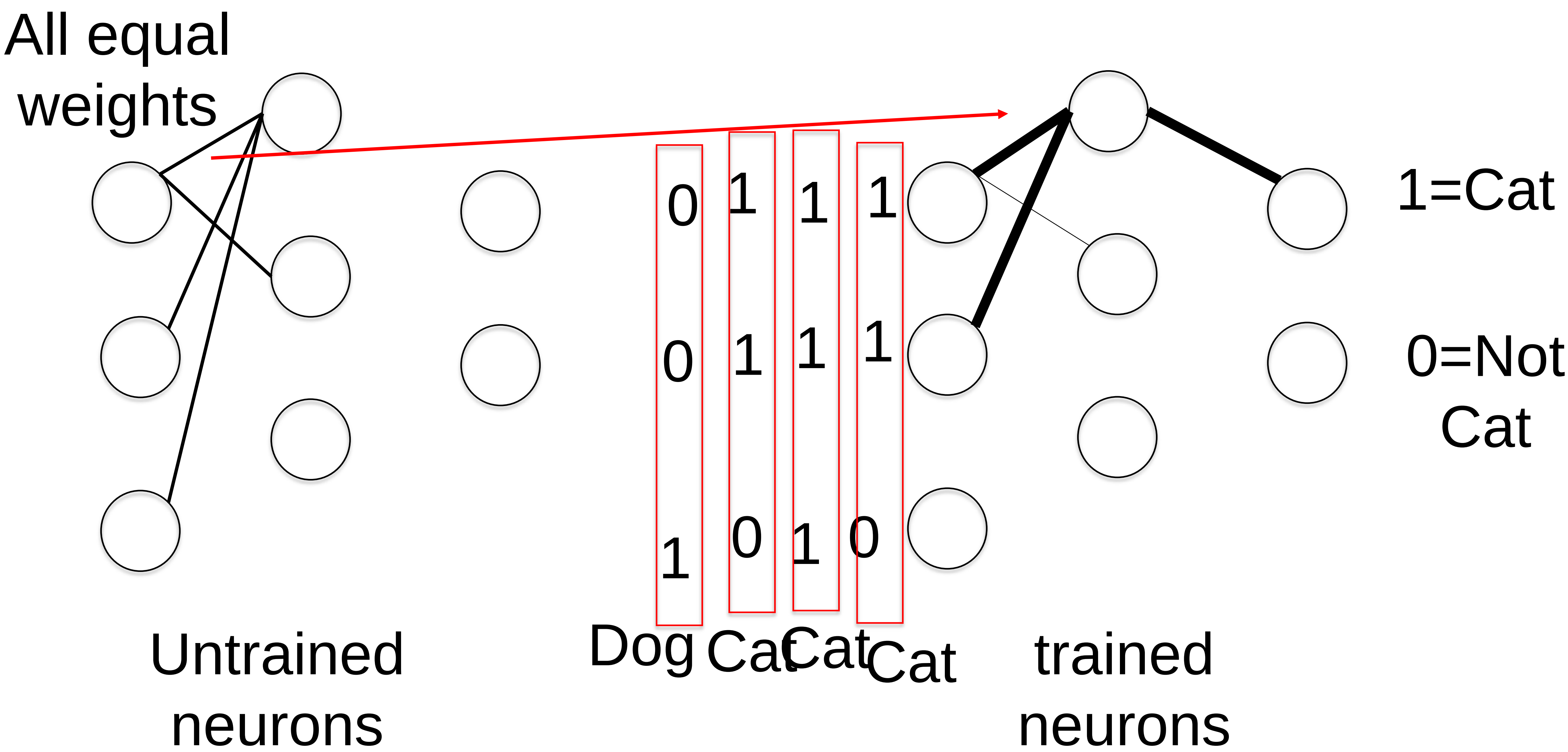
Covert 2d Image to 1d Array

Used to convert 2D to 1D array

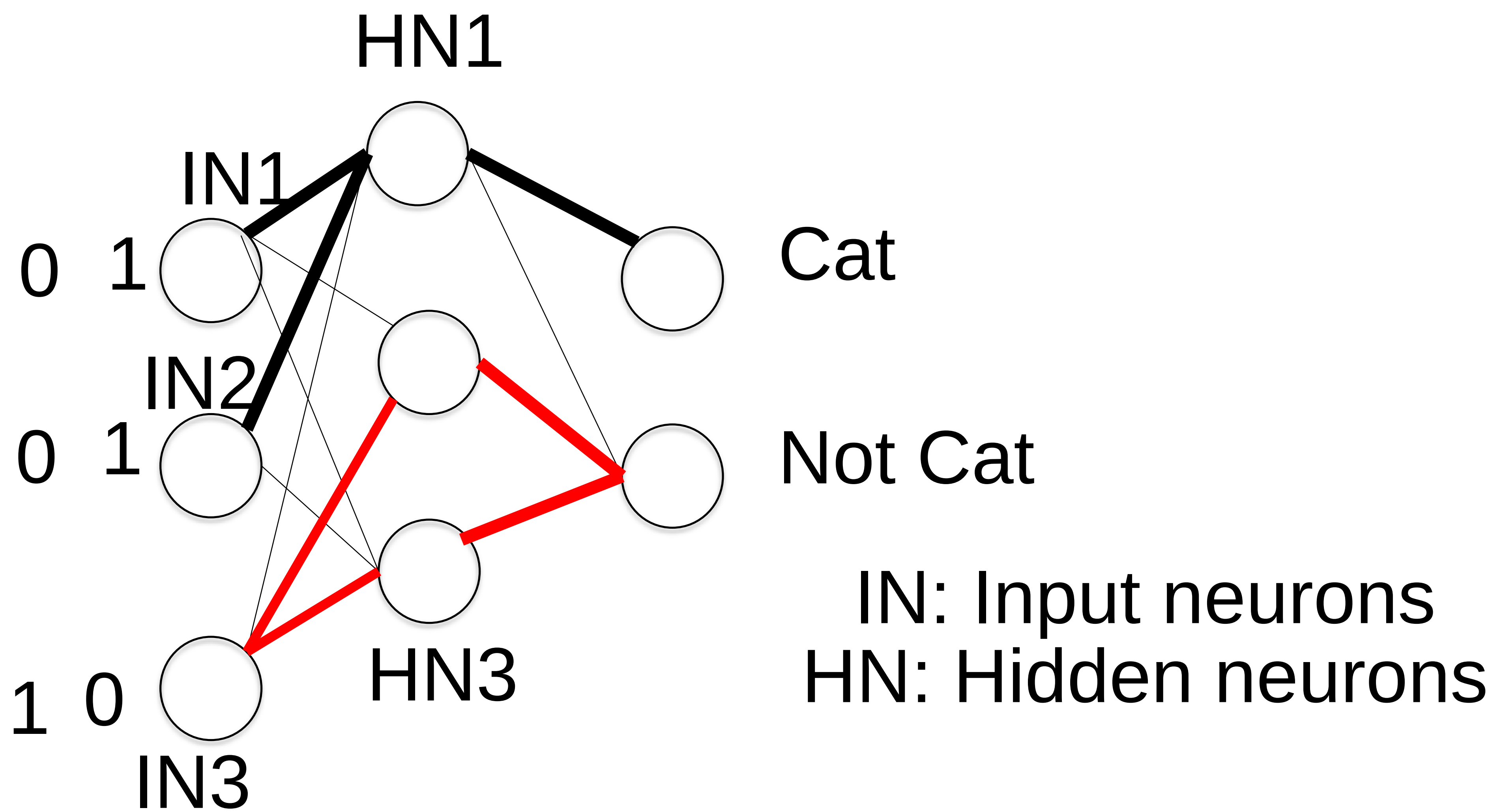
- y = reshape(repmat(ylabel,1100,1),11000,1);

Example

How do you learn if the object on the Picture is cat?

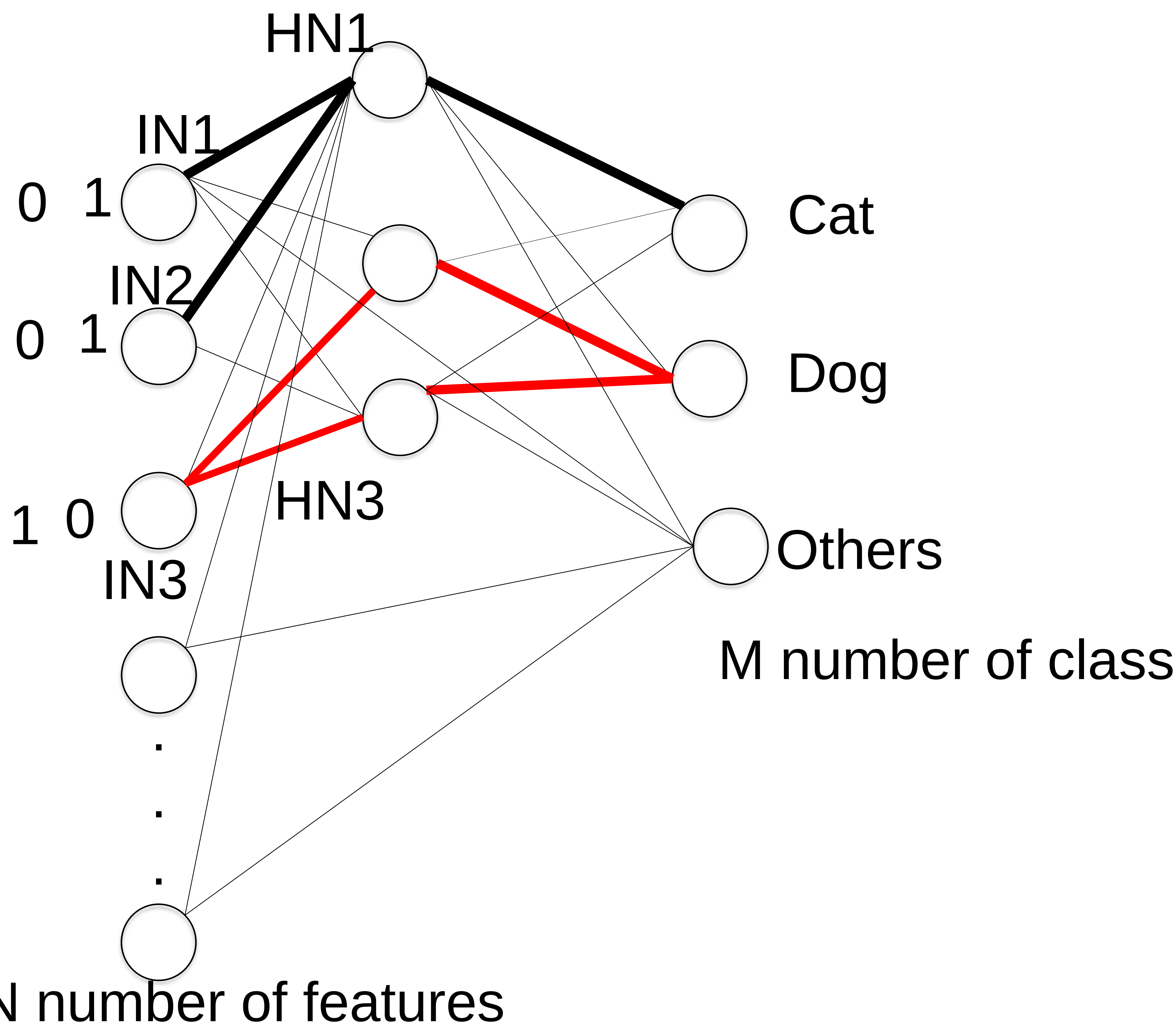


The connection between the 1st neuron in the input layer and 1st layer in hidden layer gets stronger

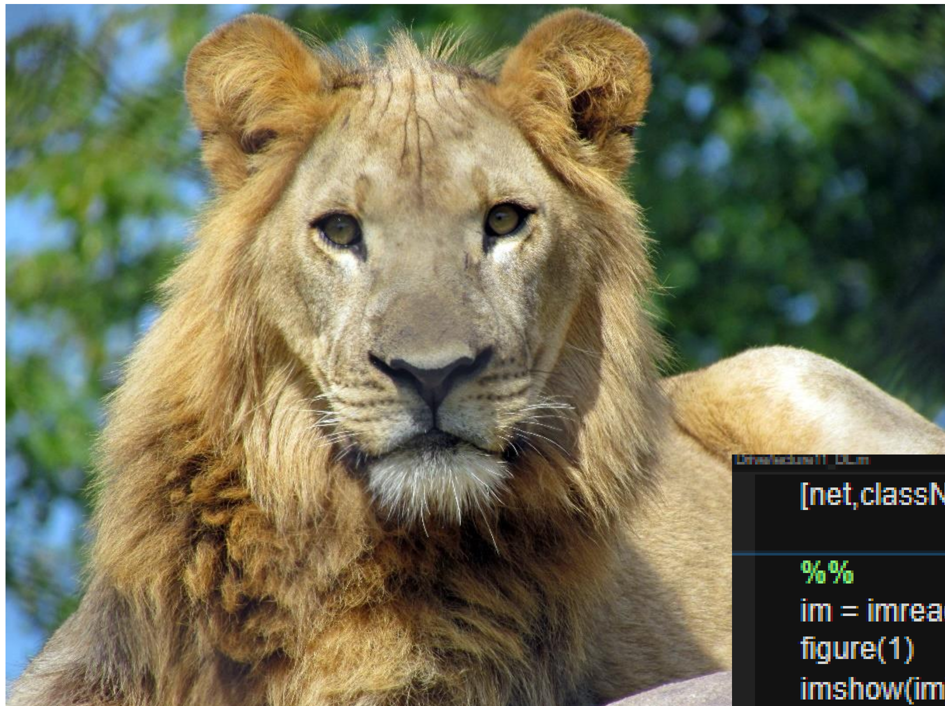


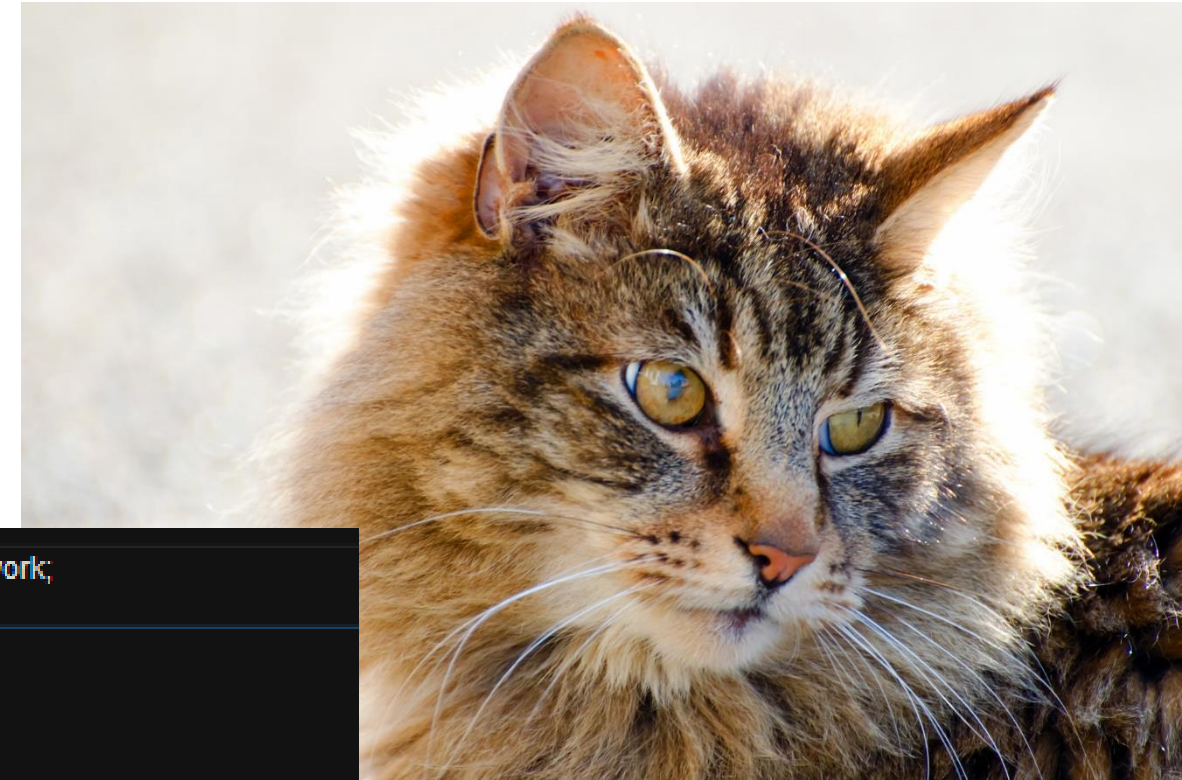
- Observing more cats strengten the connections between some neurons.
- The strength in deep learning are represented by weigths (w). When the neuron in the hidden layer receives enough input

Adding more features to DL model

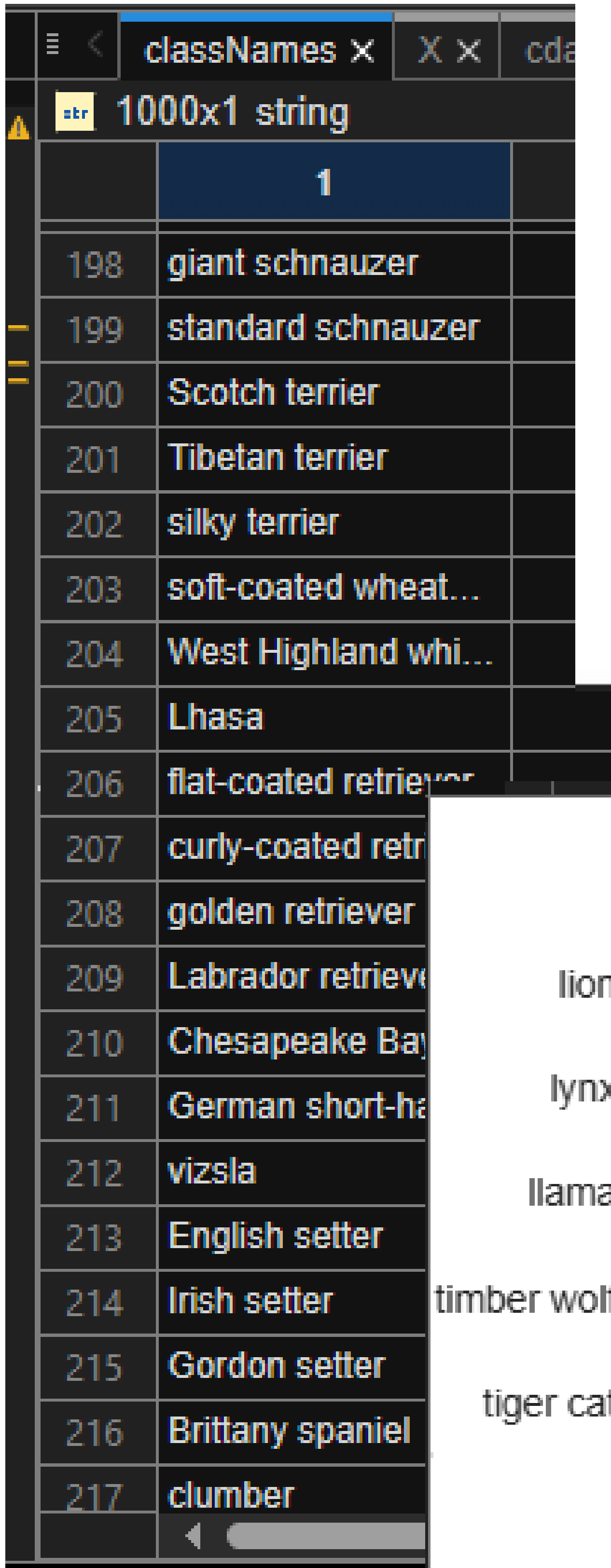


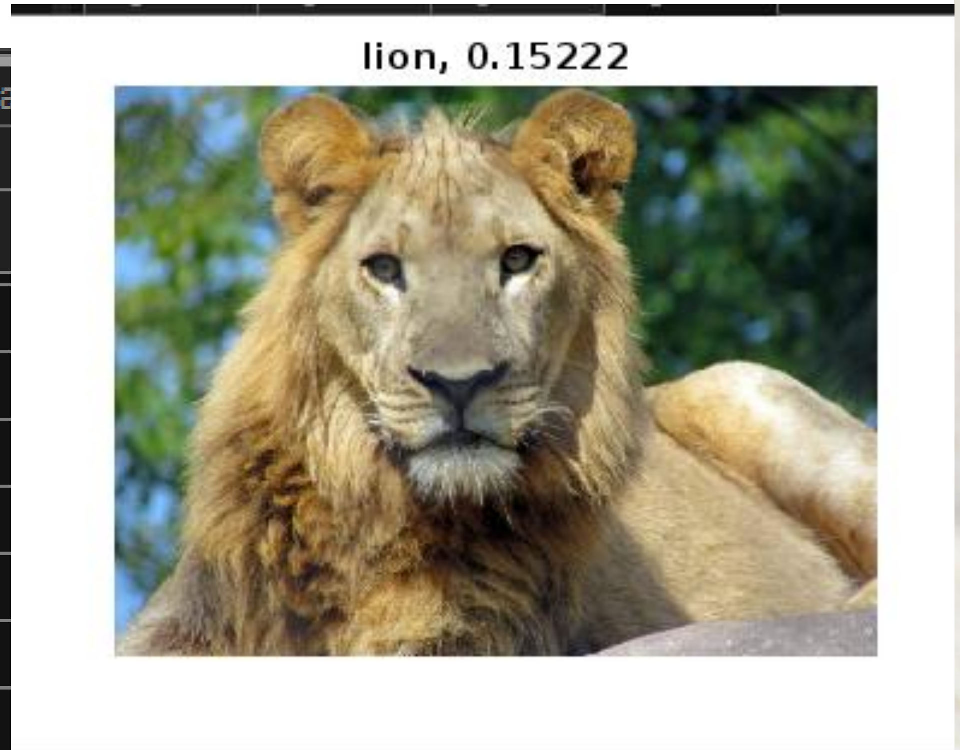
Example:

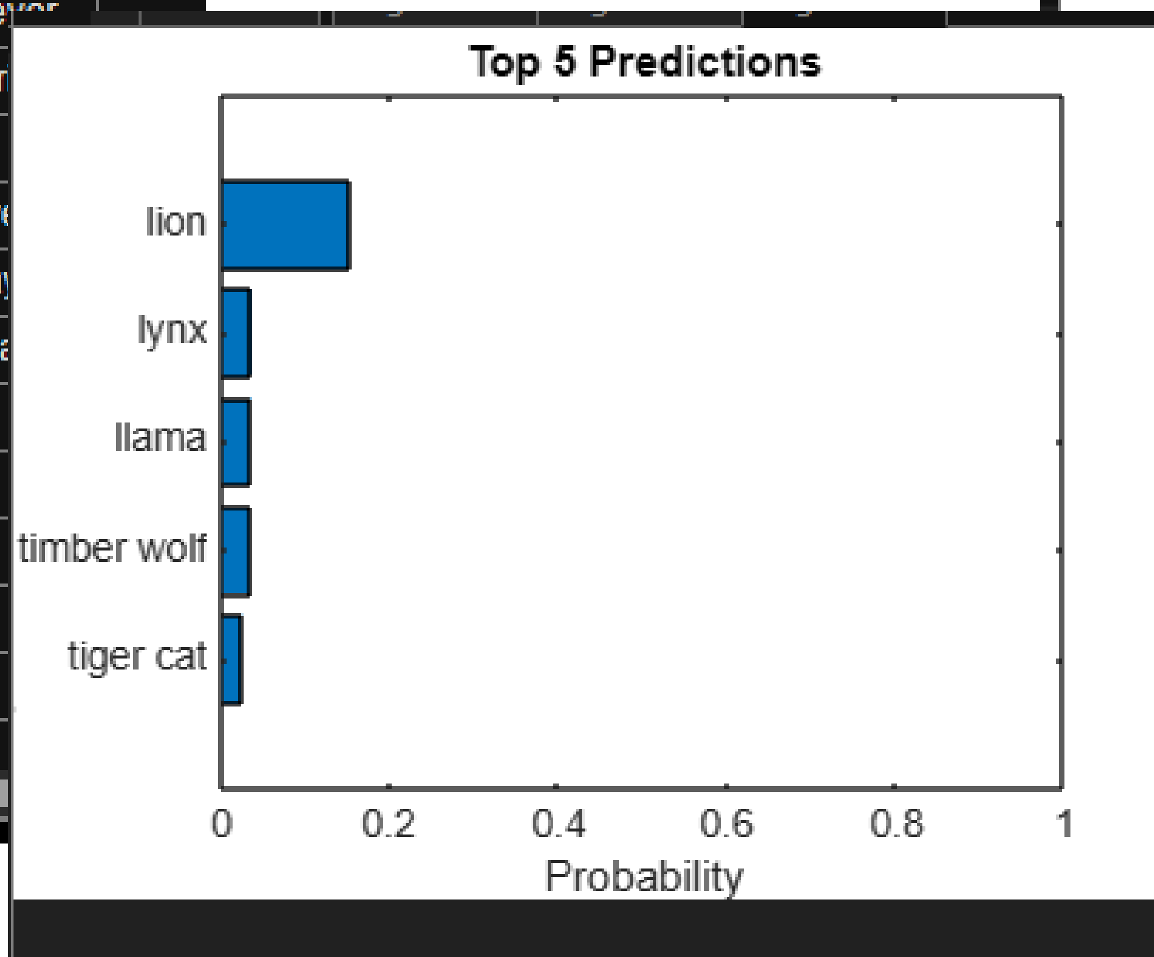


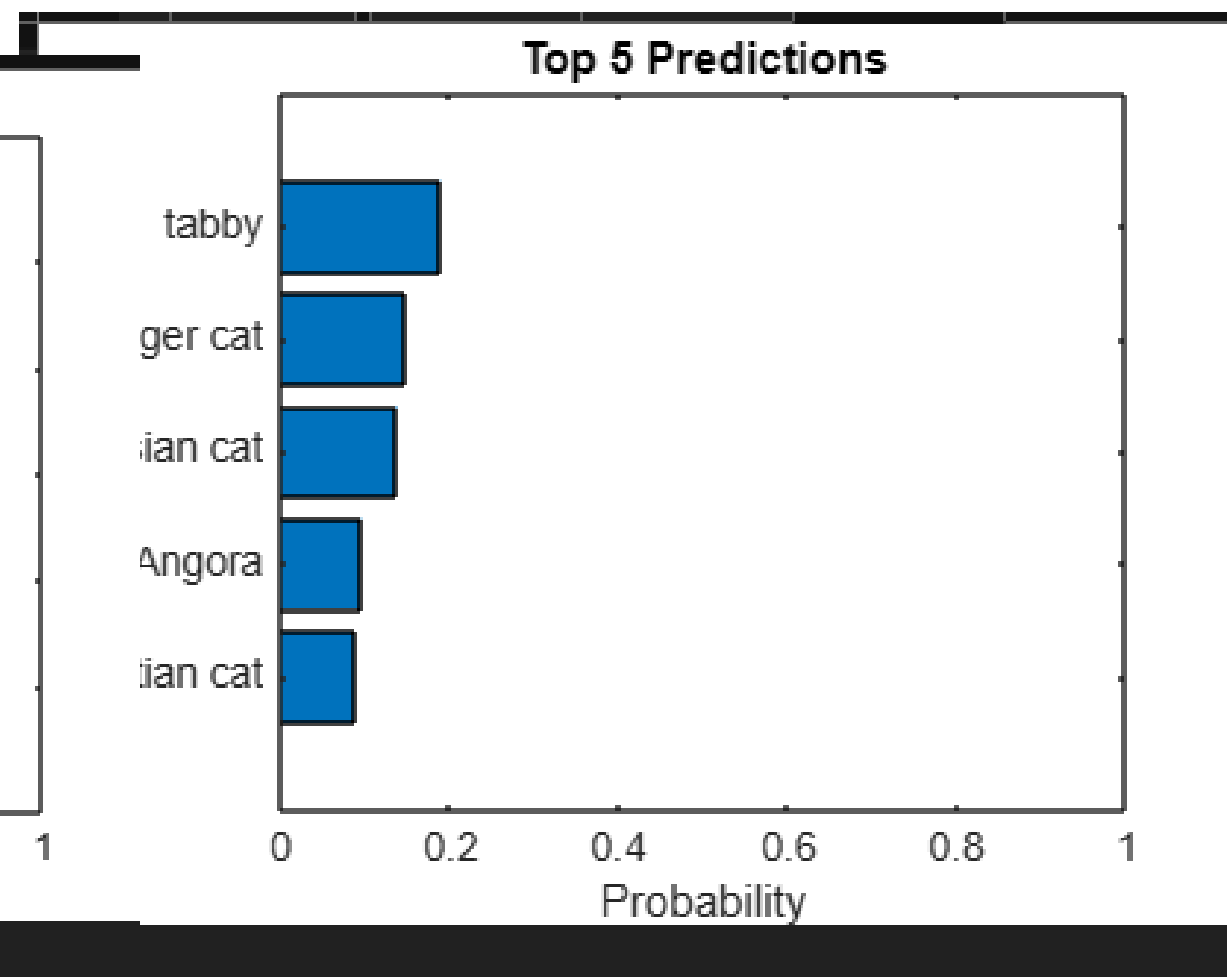


```
[net,classNames] = imagePretrainedNetwork;

%%
im = imread("peppers.png");
figure(1)
imshow(im)
%%
X = single(im);
scores = predict(net,X);
[label,score] = scores2label(scores,classNames);
%%
for i=1:1000;
  if classNames(i,1)=='lion';
       disp(i)
  end
end
%%
figure(1)
imshow(cdata)
figure(2)
imshow(cdata_1)
%%
cdata1 = single(cdata);
cdata2 = single(cdata_1);
%%
scores = predict(net,cdata2);
[label,score] = scores2label(scores,classNames);
disp(label)
disp(score)
%%
```

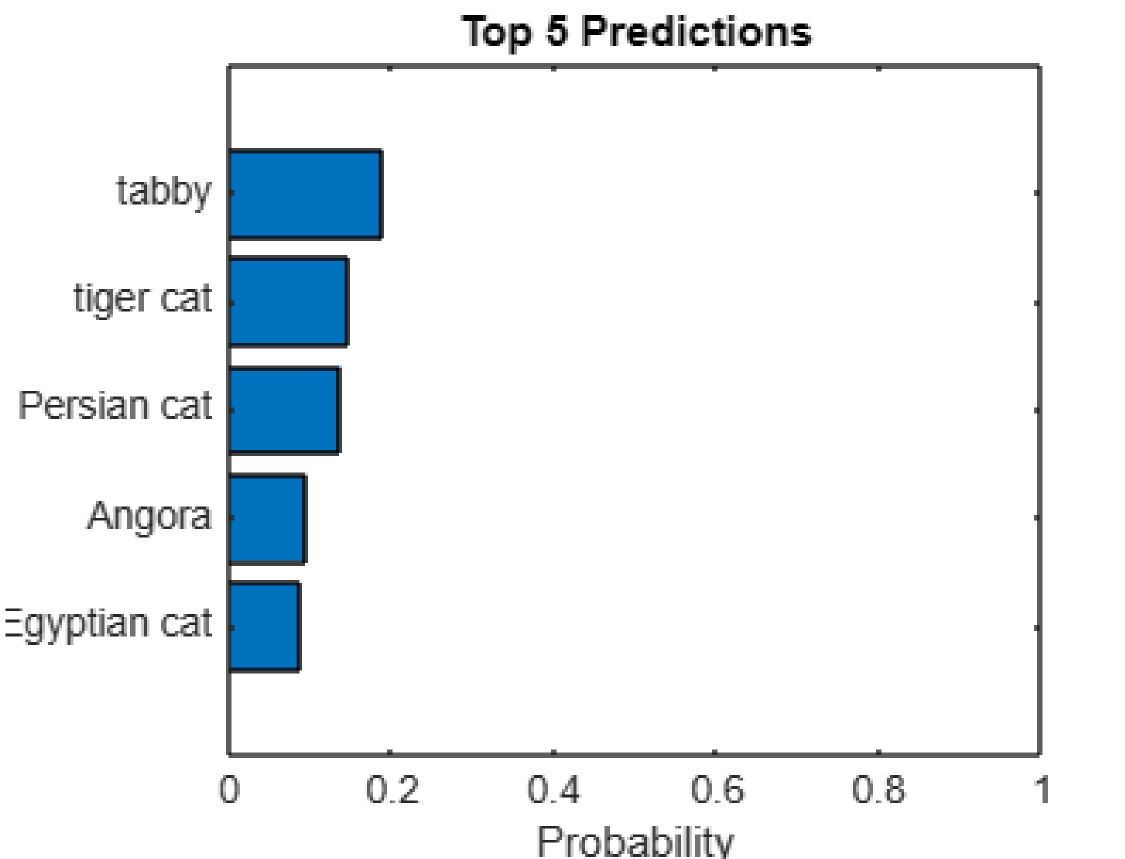

classNames
X
cda
1000x1 string
1
198 giant schnauzer
199 standard schnauzer
200 Scotch terrier
201 Tibetan terrier
202 silky terrier
203 soft-coated wheat...
204 West Highland whi...
205 Lhasa
206 flat-coated retriever
207 curly-coated retr
208 golden retriever
209 Labrador retrieve
210 Chesapeake Bay
211 German short-ha
212 vizsla
213 English setter
214 Irish setter
215 Gordon setter
216 Brittany spaniel
217 clumber

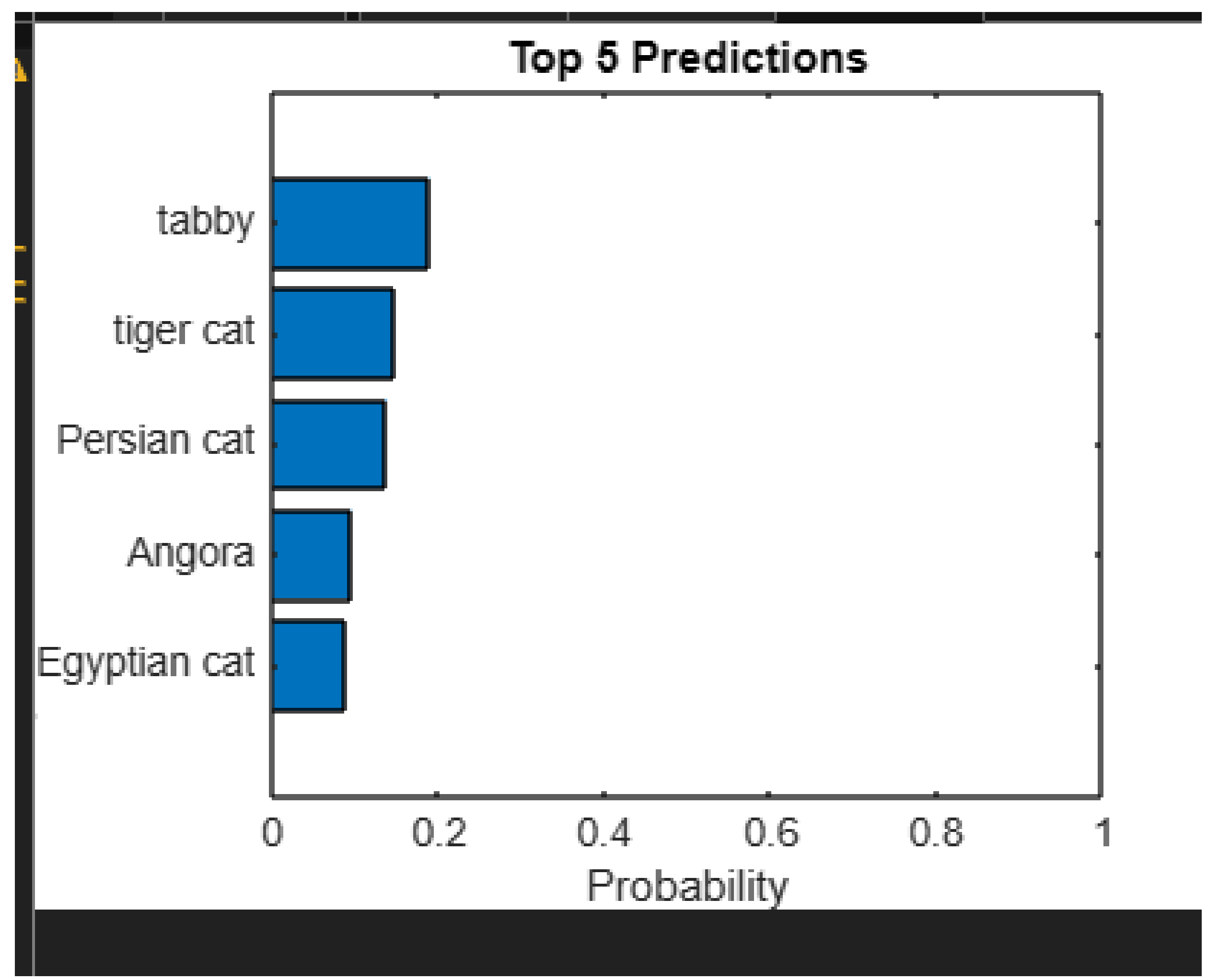

lion, 0.15222


Top 5 Predictions
lion
lynx
llama
timber wolf
tiger cat
0
0.2
0.4
0.6
0.8
1
Probability

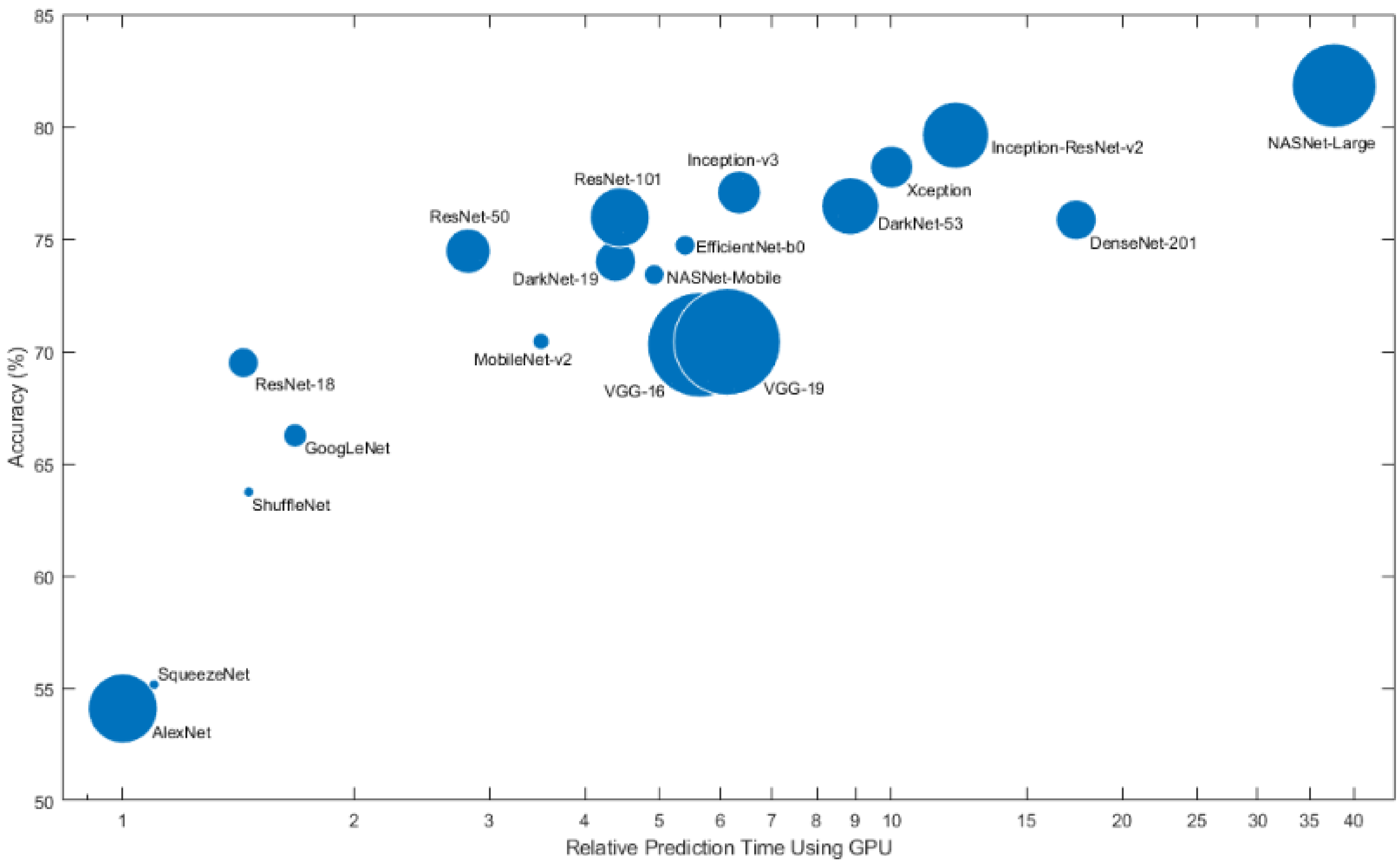

Top 5 Predictions
tabby
ger cat
ian cat
Angora
ian cat
0
0.2
0.4
0.6
0.8
1
Probability


tabby, 0.18962


Top 5 Predictions
tabby
tiger cat
Persian cat
Angora
gyptian cat
0
0.2
0.4
0.6
0.8
1
Probability


Top 5 Predictions
tabby
tiger cat
Persian cat
Angora
Egyptian cat
0
0.2
0.4
0.6
0.8
1
Probability

Pretrained networks



Load Pretrained Neural Networks

To load the SqueezeNet neural network, use the `imagePretrainedNetwork` function.

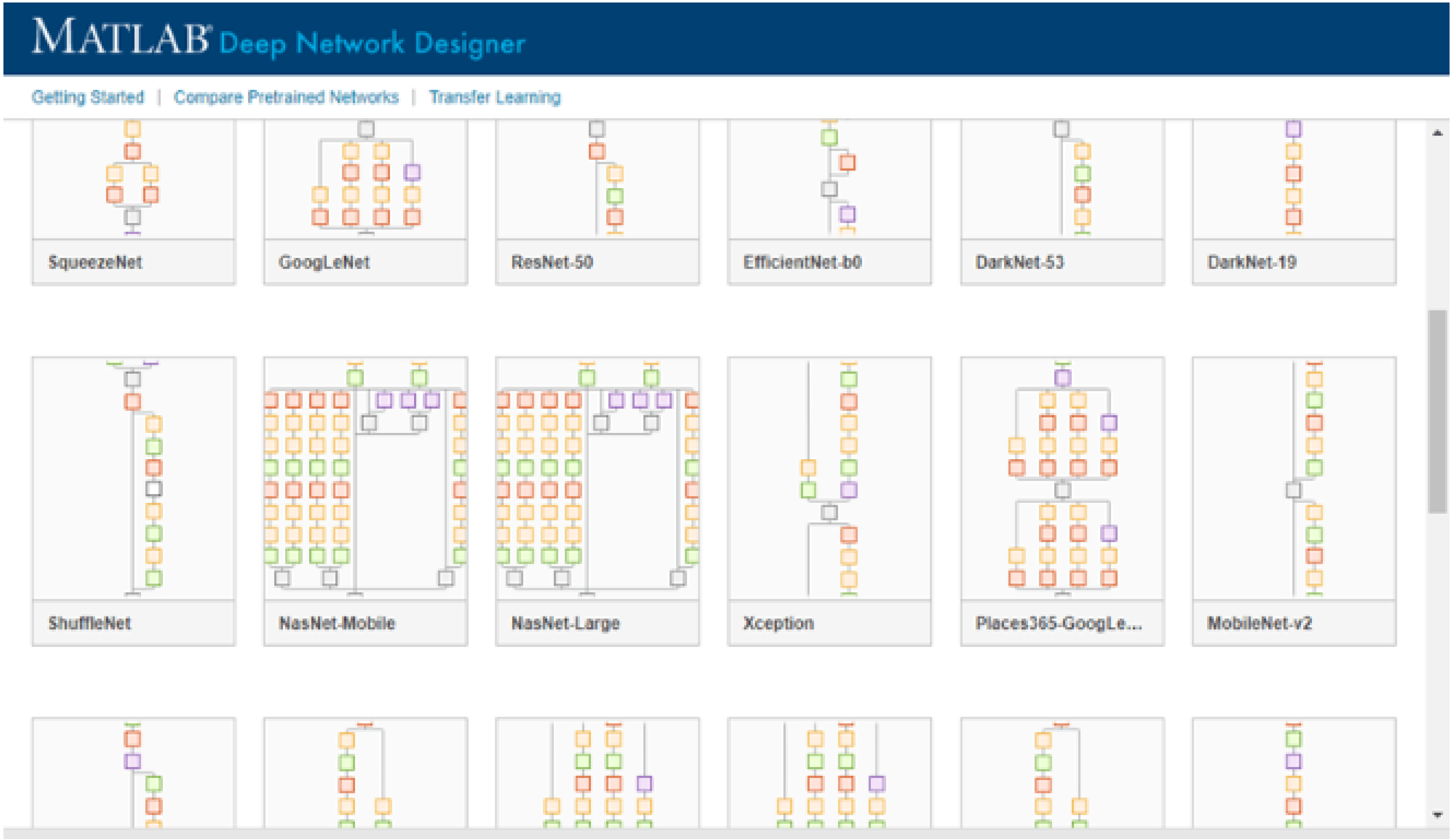
```
[net,classNames] = imagePretrainedNetwork;
```

For other neural networks, specify the model using the first argument of the `imagePretrainedNetwork` function. If you do not have the required support package for the network, the function provides a link to download it. Alternatively, you can download the pretrained neural networks from the Add-On Explorer.

This table lists the available pretrained neural networks trained on ImageNet and some of their properties. The neural network depth is defined as the largest number of sequential convolutional or fully connected layers on a path from the network input to the netwo output. The inputs to all neural networks are RGB images.

`imagePretrainedNetwork` Model Name Argument	Neural Network Name	Depth	Size	Parameters (Millions)	Image Input Size	Required Support Package
"squeezenet"	SqueezeNet [2]	18	5.2 MB	1.24	227-by-227	None
"googlenet" "googlenet-places365"	GoogLeNet [3][4]	22	27 MB	7.0	224-by-224	Deep Learning Toolbox Model for *GoogLeNet Network*
"inceptionv3"	Inception-v3 [5]	48	89 MB	23.9	299-by-299	Deep Learning Toolbox Model for *Inception-v3 Network*
"densenet201"	DenseNet-201 [6]	201	77 MB	20.0	224-by-224	Deep Learning Toolbox Model for *DenseNet-201 Network*
"mobilenetv2"	MobileNet-v2 [7]	53	13 MB	3.5	224-by-224	Deep Learning Toolbox Model for *MobileNet-v2 Network*
"resnet18"	ResNet-18 [8]	18	44 MB	11.7	224-by-224	Deep Learning Toolbox Model for *ResNet-18 Network*
"resnet50"	ResNet-50 [8]	50	96 MB	25.6	224-by-224	Deep Learning Toolbox Model for *ResNet-50 Network*
"resnet101"	ResNet-101 [8]	101	167 MB	44.6	224-by-224	Deep Learning Toolbox Model for *ResNet-101 Network*
"xception"	Xception [9]	71	85 MB	22.9	299-by-299	Deep Learning Toolbox Model for *Xception Network*
"inceptionresnetv2"	Inception-ResNet-v2 [10]	164	209 MB	55.9	299-by-299	Deep Learning Toolbox Model for *Inception-ResNet-v2 Network*
"shufflenet"	ShuffleNet [11]	50	5.4 MB	1.4	224-by-224	Deep Learning Toolbox Model for *ShuffleNet Network*
"nasnetmobile"	NASNet-Mobile [12]	*	20 MB	5.3	224-by-224	Deep Learning Toolbox Model for *NASNet-Mobile Network*
"nasnetlarge"	NASNet-Large [12]	*	332 MB	88.9	331-by-331	Deep Learning Toolbox Model for *NASNet-Large Network*
"darknet19"	DarkNet-19 [13]	19	78 MB	20.8	256-by-256	Deep Learning Toolbox Model for *DarkNet-19 Network*
"darknet53"	DarkNet-53 [13]	53	155 MB	41.6	256-by-256	Deep Learning Toolbox Model for *DarkNet-53 Network*
"efficientnetb0"	EfficientNet-b0 [14]	82	20 MB	5.3	224-by-224	Deep Learning Toolbox Model for *EfficientNet-b0 Network*
"alexnet"	AlexNet [15]	8	227 MB	61.0	227-by-227	Deep Learning Toolbox Model for *AlexNet Network*

Explore other pretrained neural networks in Deep Network Designer by clicking **New**.



If you need to download a neural network, pause on the desired neural network and click **Install** to open the Add-On Explorer.