

Machine Learning for Networking

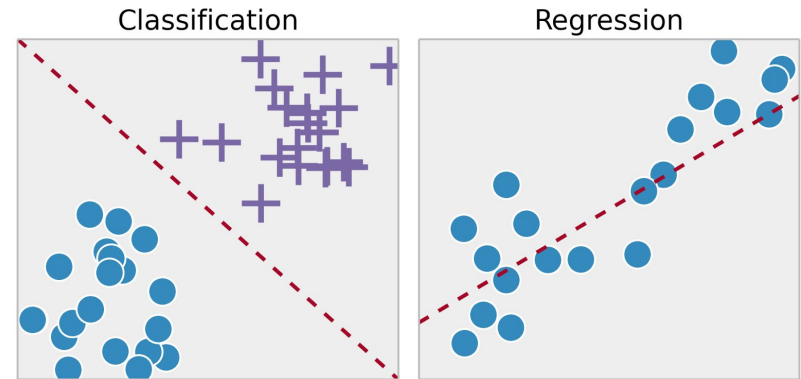
Seminar 3

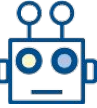
Marc Carrascosa Zamacois: marc.carrascosa@upf.edu



Classification

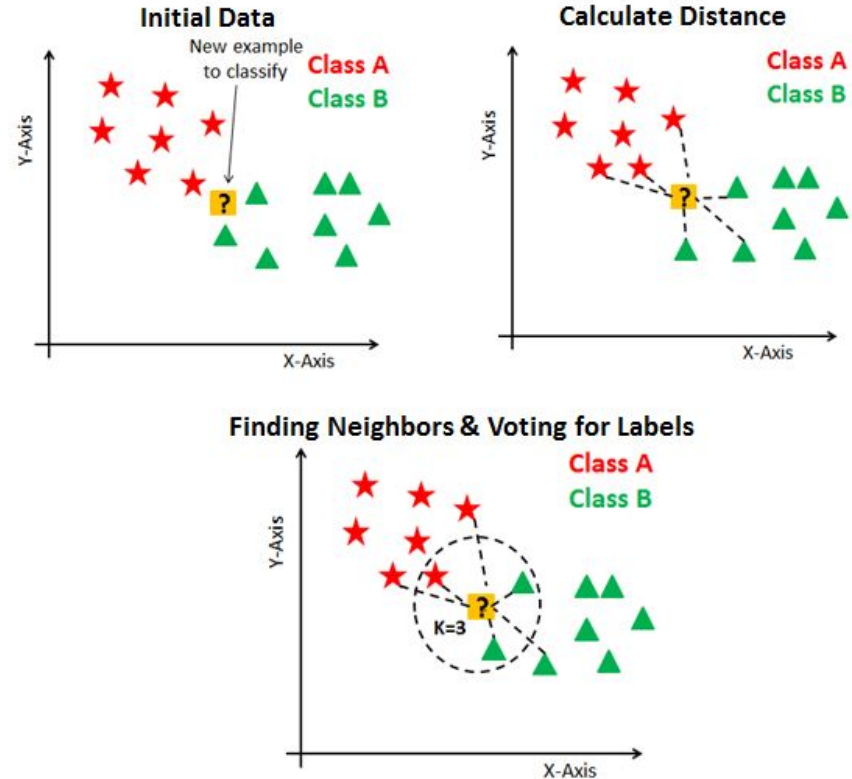
- Used when we have a finite output
- Instead of predicting a real value (like throughput in Mbps), we predict a class (is the user active? yes/no)
- Several methods available
 - K-Nearest Neighbours
 - Logistic Regression
 - Naive Bayes
 - Support Vector Machines
 - Decision Trees

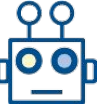




K-Nearest Neighbour

- The model are the training labels
- For each new data point we calculate the distance between it and every training point
- We then take the K data points that have the shortest distance
- We use the mode to choose the class





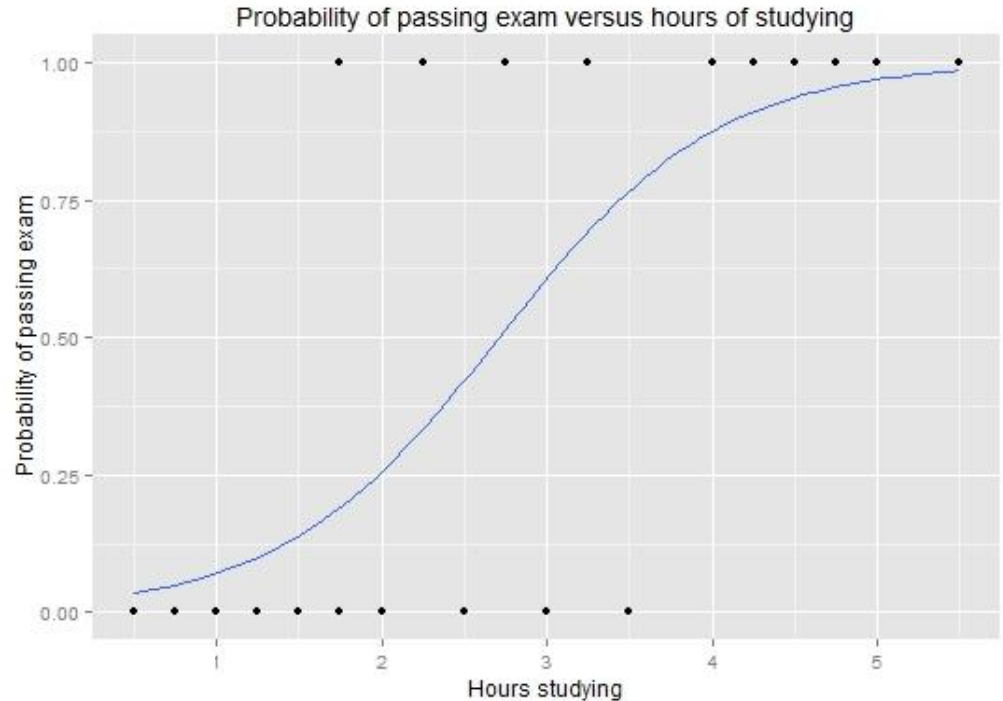
Probabilistic classifiers

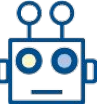
- Naive Bayes classifiers are based on Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Logistic Regression calculates the log odds of an event using a sigmoid function

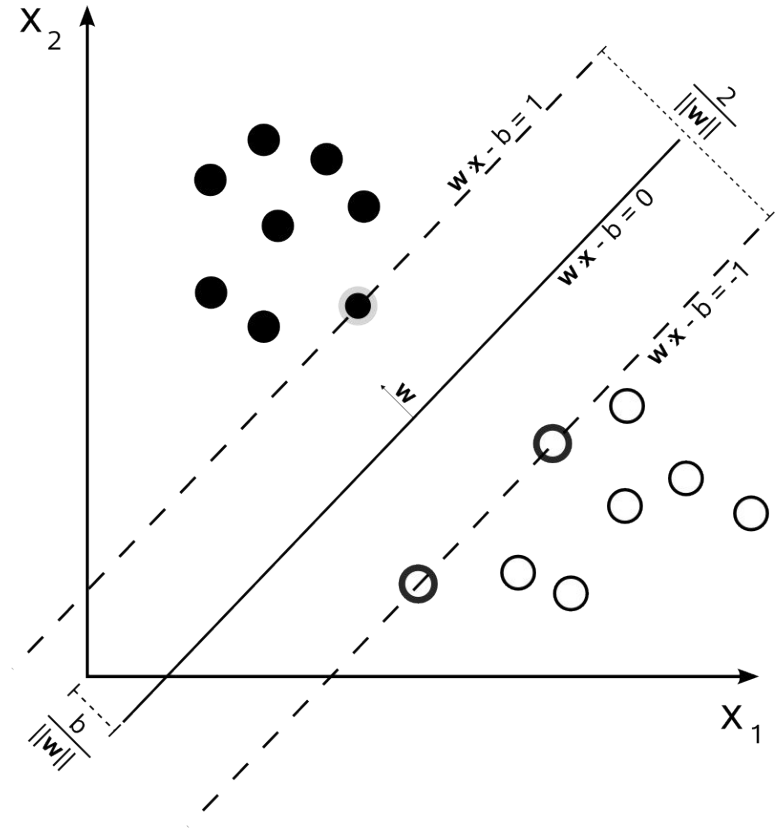
$$P(Y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}}$$

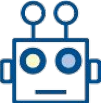




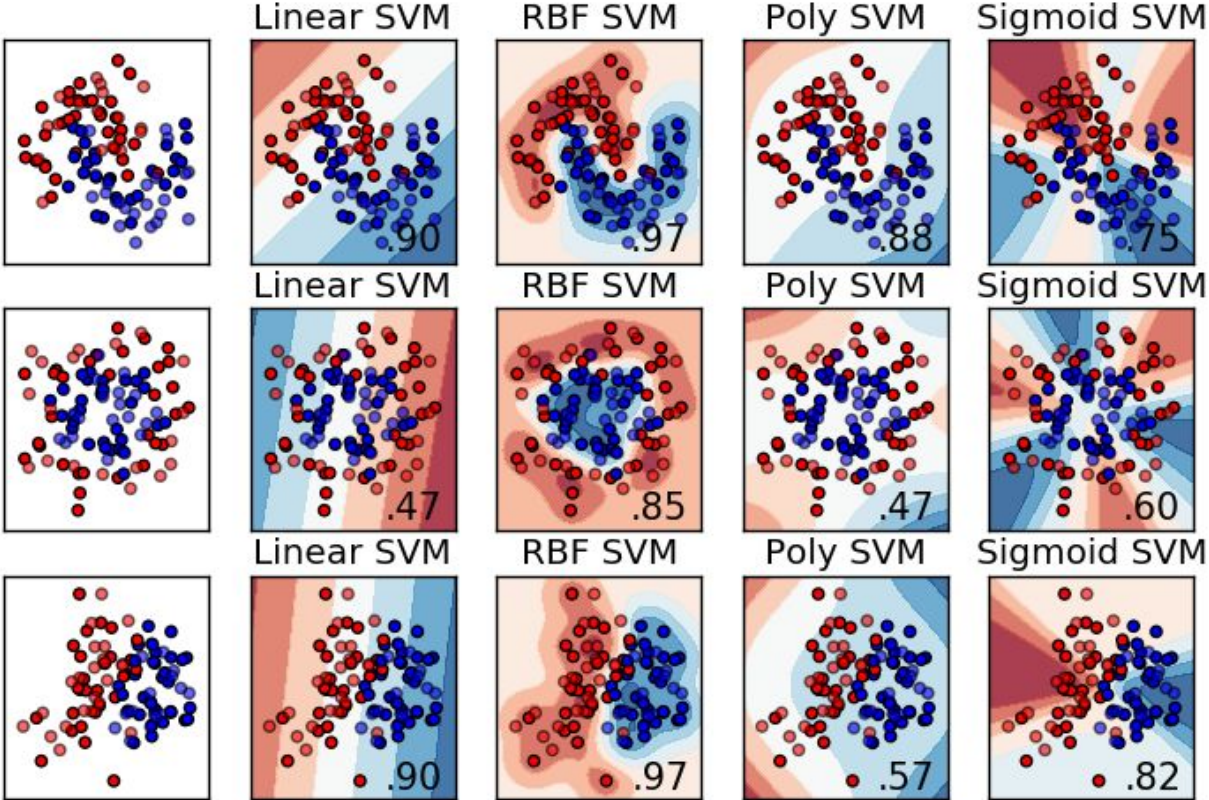
Support Vector Machines

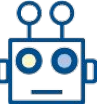
- There are multiple vectors that can separate the data
- SVM finds the one that maximizes the separation, minimizing the possibility of errors
- SVM are highly customizable





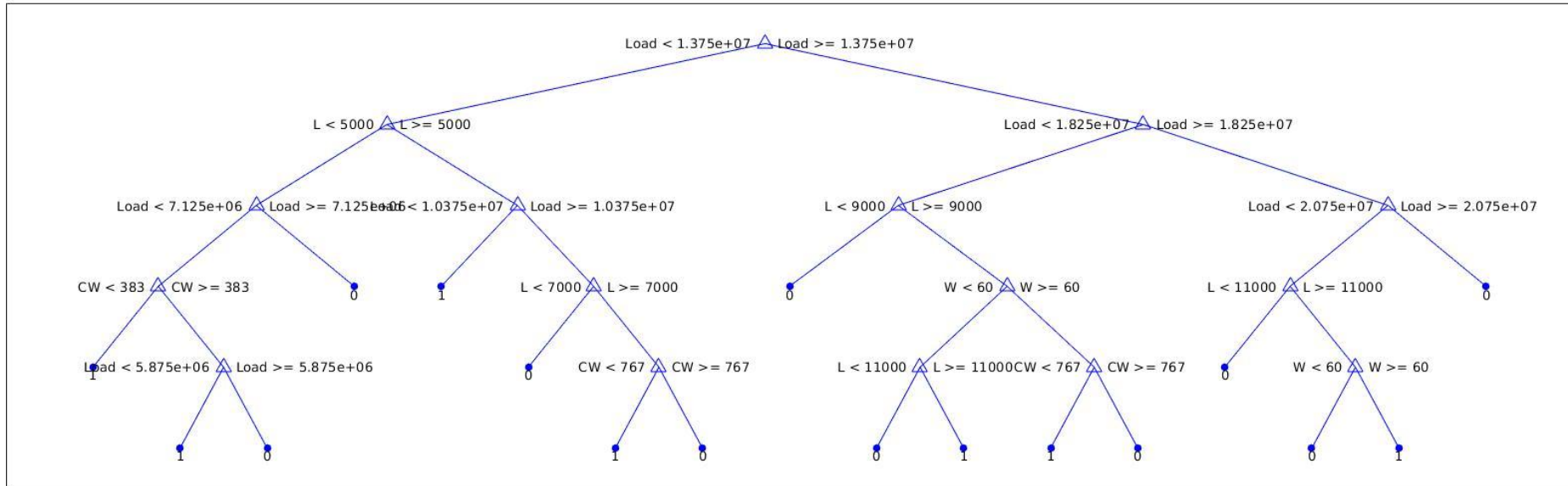
SVM kernels

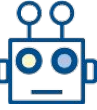




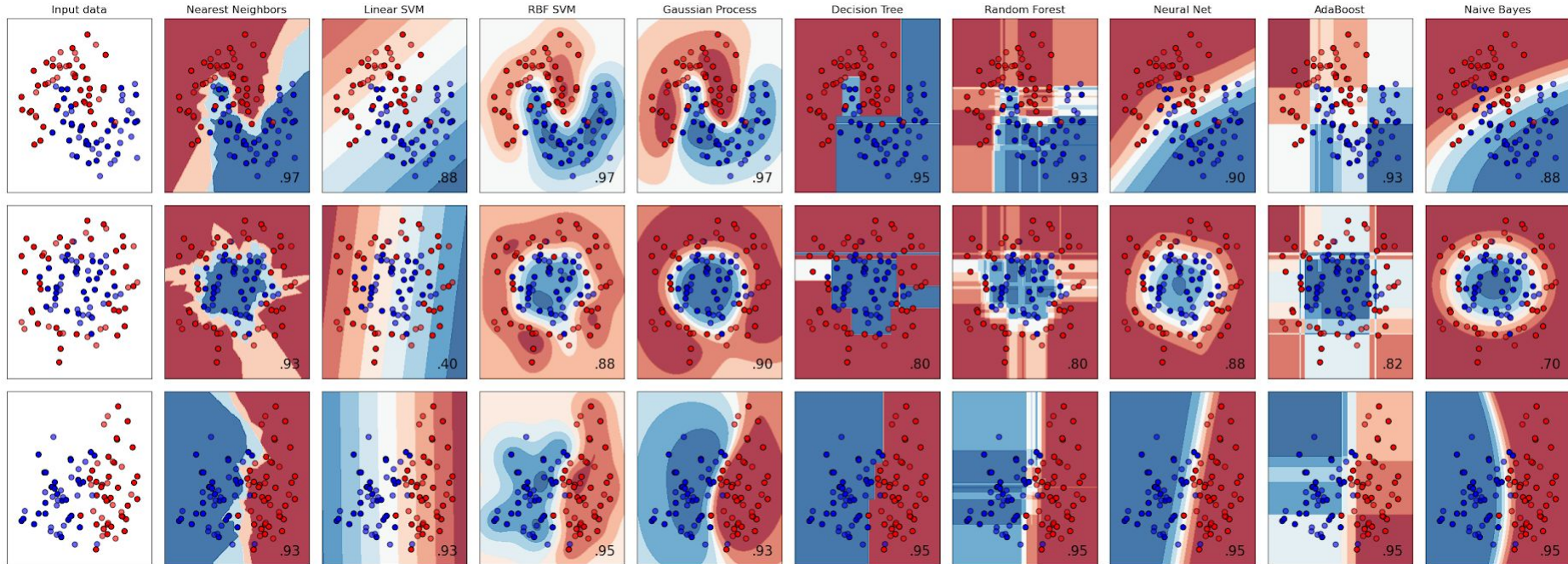
Decision Trees

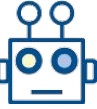
We saw them last seminar!





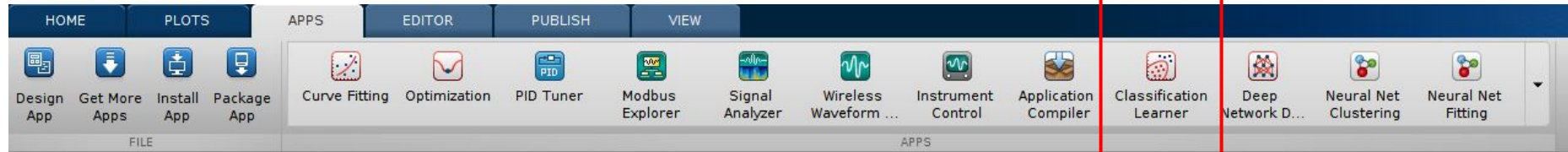
Classifiers

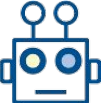




MATLAB classification learner

- MATLAB has a classification tool that simplifies the model selection/ training process
- It can be intensive on you RAM and CPU usage





Importing the dataset

Import the dataset from your workspace

Classification Learner

Classification Learner | **VIEW**

Classification Learner toolbar: **New Session** (highlighted), Feature Selection, PCA, All Quick-T...

Data Browser

New Session

Data set

Workspace Variable: NdsT 10000x12 double

Use columns as variables
 Use rows as variables

Response

column_12 double 0 .. 9

Predictors

	Name	Type	Range
<input checked="" type="checkbox"/>	column_1	double	2 .. 41
<input checked="" type="checkbox"/>	column_2	double	500000 .. 8.2e+07
<input checked="" type="checkbox"/>	column_3	double	1 .. 40
<input checked="" type="checkbox"/>	column_4	double	1 .. 40
<input checked="" type="checkbox"/>	column_5	double	1 .. 1600
<input checked="" type="checkbox"/>	column_6	double	3 .. 1023
<input checked="" type="checkbox"/>	column_7	double	20 .. 160
<input checked="" type="checkbox"/>	column_8	double	4000 .. 12000
<input checked="" type="checkbox"/>	column_9	double	-81.4052 .. -40.8636
<input checked="" type="checkbox"/>	column_10	double	-81.4052 .. -40.8636

Validation

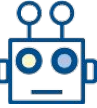
Cross-Validation
 Protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold.
 Cross-validation folds: 5 folds

Holdout Validation
 Recommended for large data sets.
 Percent held out: 30%

No Validation
 No protection against overfitting.

[Read about validation](#)

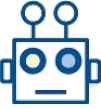
Response variable is numeric. Distinct values will be interpreted as class labels.



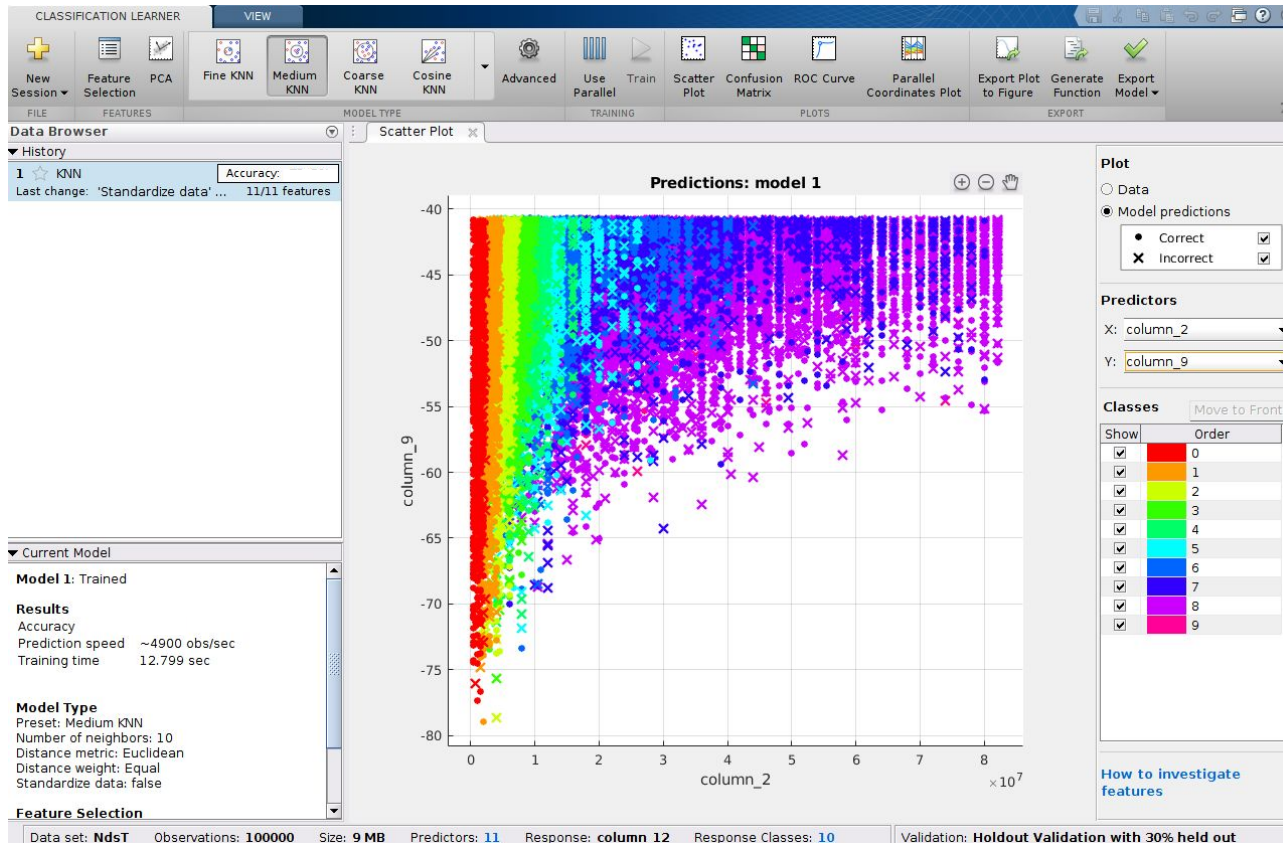
Classification learner

Disable parallel computing to improve performance!

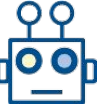
The screenshot shows the 'Classification Learner - Scatter Plot' window. The 'Use Parallel' button in the 'TRAINING' section is highlighted with a red box. The main plot area displays a scatter plot titled 'Original data set: NdsT' with 'column_9' on the y-axis and 'column_2' on the x-axis (scaled by $\times 10^7$). The plot shows a dense distribution of points colored by class. On the right, the 'Plot' settings show 'Data' selected, 'Predictors' set to 'column_2' and 'column_9', and 'Classes' listed from 0 to 7 with corresponding color swatches. The bottom status bar indicates: 'Data set: NdsT Observations: 100000 Size: 9 MB Predictors: 11 Response: column_12 Response Classes: 10 Validation: Holdout Validation with 30% held out'.



Classification learner



- Plots show where predictions failed once the model is trained
- You can also check the confusion matrix
- Models can be compared and once one is chosen exported to the workspace



Classification learner

Model 3

0	746	120								
1	321	1108	270	1						2
2	13	560	782	364	32		1		15	26
3	1	228	491	530	256	26	13	19	218	90
4		70	282	359	394	108	62	35	385	103
5		9	172	207	271	244	178	58	442	137
6			71	134	191	201	315	225	527	135
7			25	126	137	200	246	1078	2667	322
8			1	101	132	260	112	309	9417	1469
9			9	17	34	5			741	1044
	0	1	2	3	4	5	6	7	8	9

True class

Predicted class

10-Nearest Neighbour

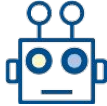
Model 6

0	786	80								
1	33	1517	148	3	1					
2		109	1453	217	3	6	1	4		
3			214	1239	374	11	25	8	1	
4			1	157	1350	234	40	13	3	
5				1	312	1152	182	56	15	
6					12	487	1028	177	95	
7					1	103	307	3452	934	4
8						11	74	501	10853	362
9						2	7	26	994	821
	0	1	2	3	4	5	6	7	8	9

True class

Predicted class

Decision tree



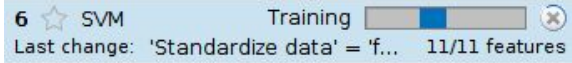
Classification learner

Gaussian Naive Bayes

Model 4: Trained

Results

Accuracy
Prediction speed ~580000 obs/sec
Training time **1.4087 sec**



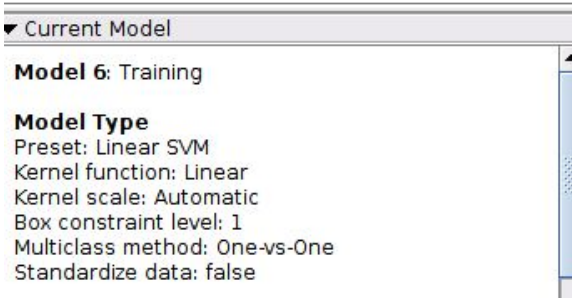
Training time: hours

Kernel Naive Bayes

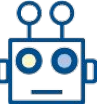
Model 5: Trained

Results

Accuracy
Prediction speed ~200 obs/sec
Training time **303.59 sec**



To reduce execution time, reduce the features used (based on what we saw in past seminars)



Commands

- `c = confusionmat(y_test,y_predicted);`
 - Get confusion matrix
- `confusionchart(c)`
 - Plot confusion matrix
- `fitcknn(x,y,'NumNeighbors',k)`
 - Fit k-nearest neighbours classifier
- `fitcnb(x,y)`
 - Fit naive bayes classifier
- `fitcecoc(x,y,'Learners','Linear')`
 - Fit SVM
- `loss(Model, x_test, y_test)`
 - Calculate error for classification model
 - Alternative `sum(ytest==ypred)/length(ytest)`
- `Fitctree (xtrain,ytrain,'MaxNumSplits',1000)`
 - Decision tree



Tasks

- Load the seminar 1 dataset.
- We will use the airtime (column 16) as the output. Separate the airtime into 10 classes (class 0: $[0, 0.1)$, class 1: $[0.1, 0.2)$, etc.)
- Open the MATLAB classifier app.
- Train a couple of models for this new dataset and check their accuracy (Modify the number of neighbors of KNN, disable the use of standardized data, try both types of Bayesian classifiers, change the depth of the decision trees, etc). How high can the accuracy go?
- Choose a new classification for the data, you can use the information from the previous testing, check the distribution of the data, or any method you prefer (hint: use less classes)
- Re-train the models with the new classification. Are the results better or worse than before?