UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Tools for Artificial Intelligence with MATLAB, initiation (TAIM)

José Antonio Lázaro

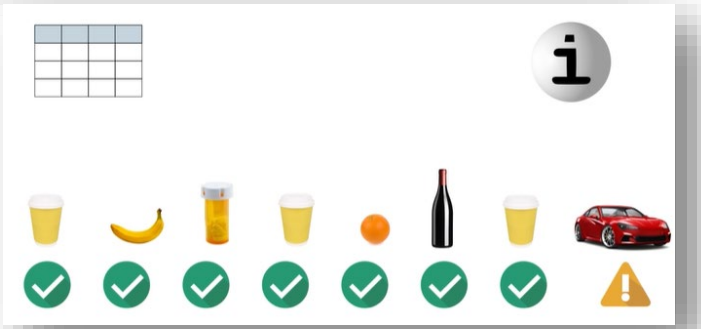
Machine Learning

Barcelona, 3, February 2025

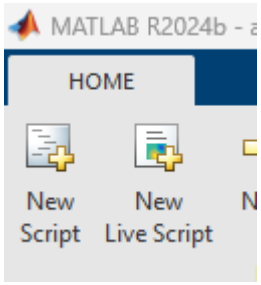
What for?

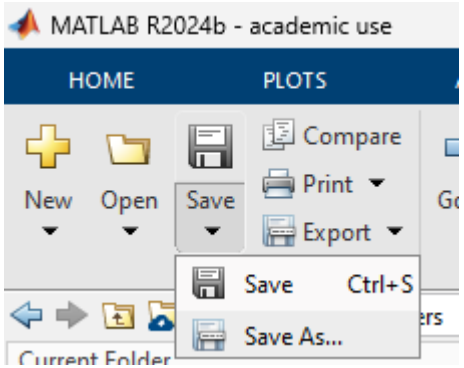
- Extracting information from data.

E.g.:
- Detection of fraudulent use of credit card.
- Maintenance prediction
- Hand-writen text recognition



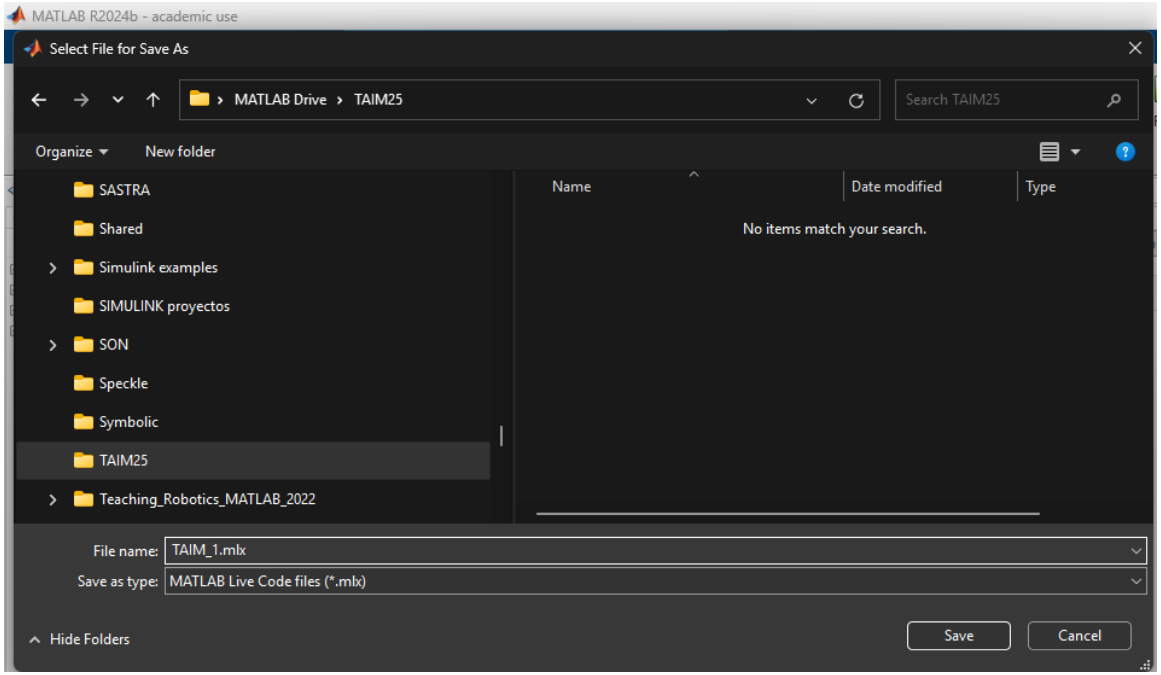
Let's go





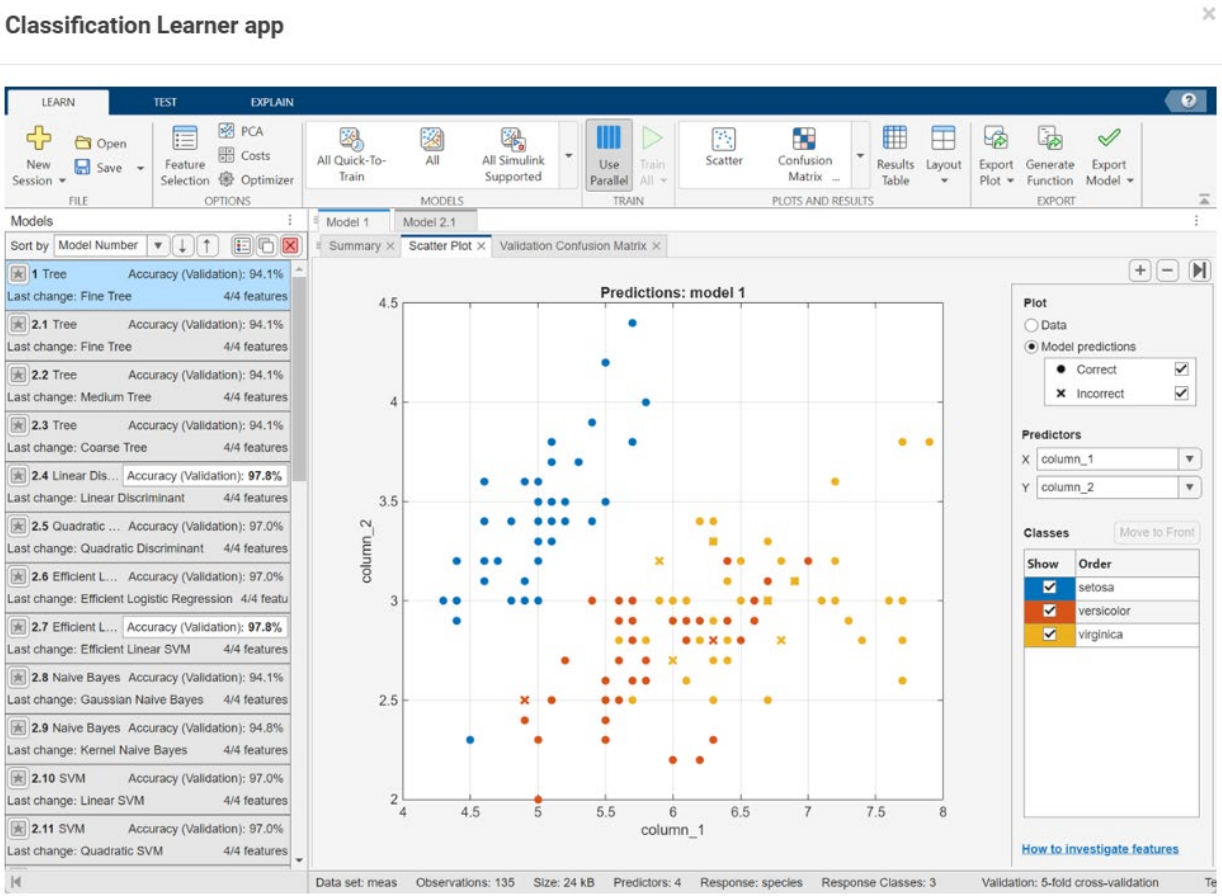
Open Matlab

- Do "New Live Script"
- Go "Live Editor"
- And "Save as"
- Create a Folder for your course "TAIM25"
- Select a Name and save it. (E.g. "**TAIM_2**")
- NO SPACES at the NAME



Matlab App: Classification Learner

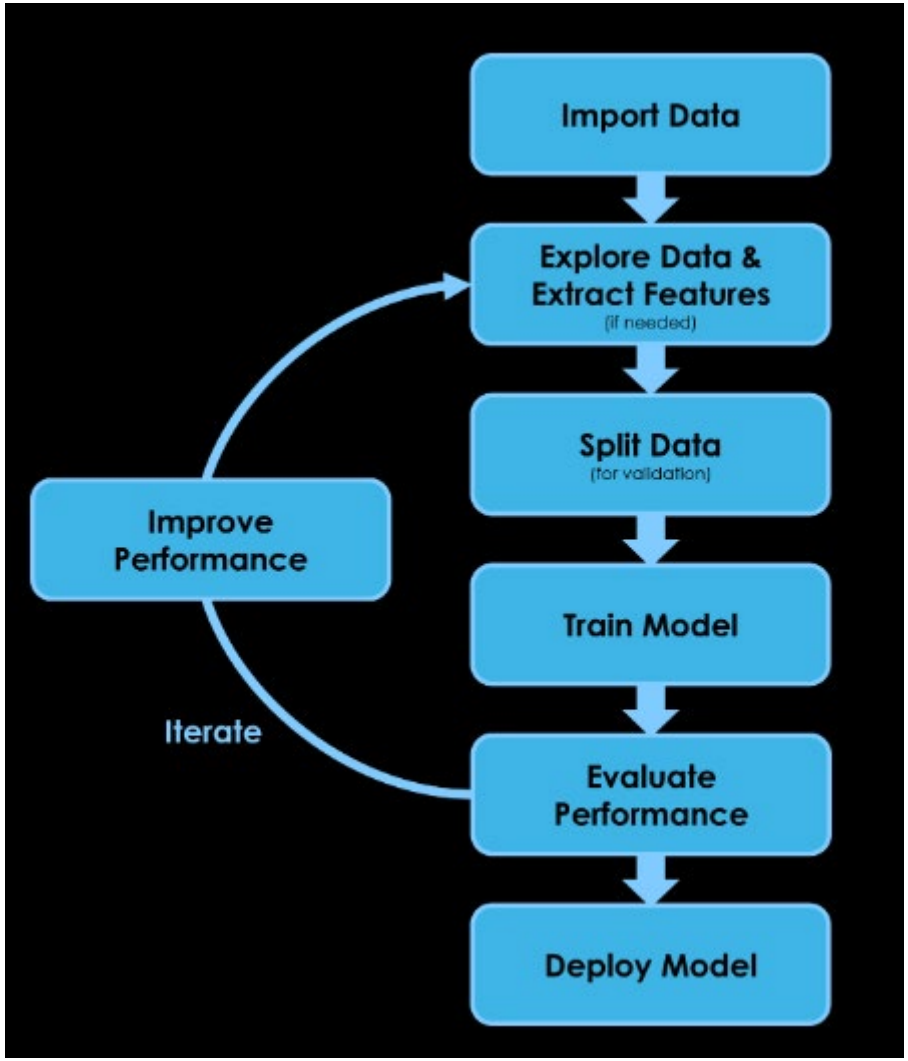
- Following similar approach as with the Curve Fitter App



TAIM - jose.antonio.lazaro@upc.edu

Machine Learning Workflow

- Iterative process



Machine Learning Workflow

- Exercise with handwriting data

Data for each handwritten letter stored in individual text files.

Each file is comma-delimited and contains four columns:

- a timestamp,
- the horizontal location of the pen,
- the vertical location of the pen,
- and the pressure of the pen.

Variables - letter

letter

49x4 table

	1 Time	2 X	3 Y	4 P
1	0	-0.4529	0.4102	0.0030
2	0.0340	-0.4529	0.3997	0.1690
3	0.0340	-0.4529	0.3997	0.1690
4	0.0580	-0.4529	0.3892	0.2470
5	0.0630	-0.4371	0.3892	0.2590
6	0.0830	-0.4371	0.3682	0.3410
7	0.1000	-0.4214	0.3262	0.4250
8	0.1150	-0.4214	0.2632	0.5170
9	0.1320	-0.3899	0.1582	0.6240
10	0.1490	-0.3584	0.0217	0.7000
11	0.1650	-0.3269	-0.1149	0.7280
12	0.1800	-0.3111	-0.2199	0.7390

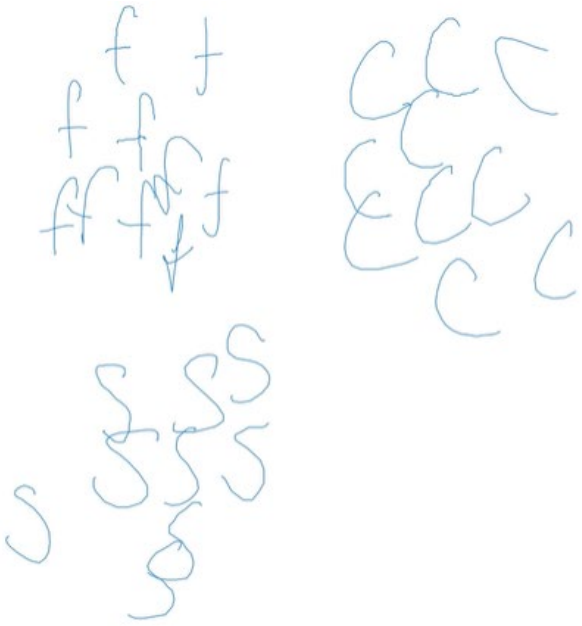
```
Time,X,Y,P
0,-0.401869018475355,-0.486529425034148,0.011
0.012,-0.401869018475355,-0.502604185773162,0.048
0.012,-0.401869018475355,-0.502604185773162,0.048
0.029,-0.401869018475355,-0.518678946512175,0.241
0.047,-0.377756877366834,-0.502604185773162,0.378
0.063,-0.353644736258316,-0.422230382078093,0.437
0.079,-0.25719617182423,-0.261482774687956,0.458
0.097,-0.13663546628163,0.0117881578752772,0.458
0.112,-0.0401869018475477,0.268984329699497,0.46
0.128,0.0803738036950561,0.526180501523716,0.466
0.146,0.176822368129138,0.751227151869908,0.499
0.162,0.249158791454699,0.863750477043004,0.528
0.178,0.273270932563221,0.847675716303991,0.56
0.195,0.297383073671742,0.735152391130895,0.638
0.211,0.321495214780264,0.574404783740757,0.72
0.228,0.321495214780264,0.349358133394565,0.86
0.246,0.297383073671742,0.10823672230936,0.94
0.262,0.273270932563221,-0.0846604065588051,0.964
0.279,0.225046650346178,-0.229333253209929,0.968
0.296,0.176822368129138,-0.341856578383025,0.958
0.312,0.128598085912099,-0.390080860600066,0.948
0.33,0.0803738036950561,-0.325781817644011,0.95
0.345,0.0562616625865346,-0.261482774687956,0.956
0.363,0.0321495214780132,-0.165034210253874,0.968
0.38,-0.0160747607390262,-0.0846604065588051,0.985
0.395,-0.0642990429560656,-0.0525108850807777,0.987
0.413,-0.0884111840645871,-0.00428660286373652,0.956
0.429,-0.0884111840645871,0.0117881578752772,0.728
0.445,0.00803738036949526,0.0439376793533046,0.153
0.453,0.0321495214780132,0.0439376793533046,0.036
0.455,NaN,NaN,NaN
```

Machine Learning Workflow

Organizing data:

- For machine learning, data is needed.

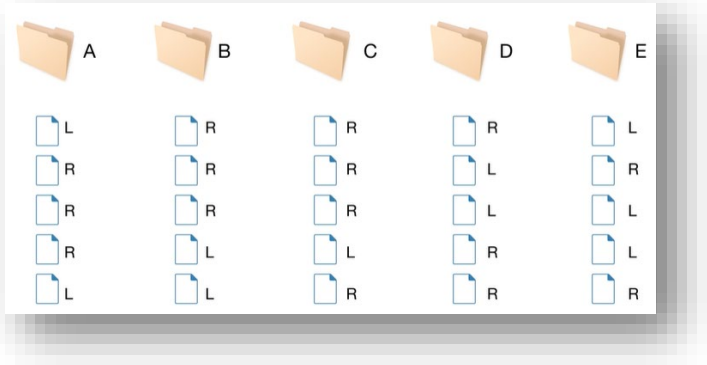
E,g.: A model to classify 26 separate characters → Multiple examples of each letter. (Maybe hundreds or even thousands of individual "observations") How to stored them?

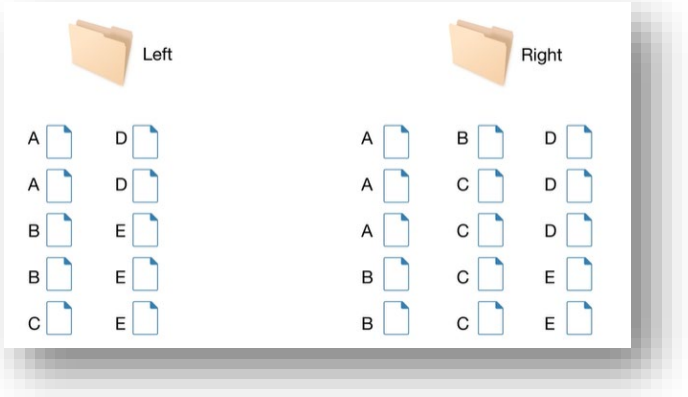


Machine Learning Workflow

Organizing data:

- For machine learning, data is needed.
- All data in a huge file? (Maybe not the best way).
- Better, each letter is stored in a separate file.
- How are those hundreds of files organized?
- E.g. a folder for each letter, or one folder with all the files with the letter in the file name.



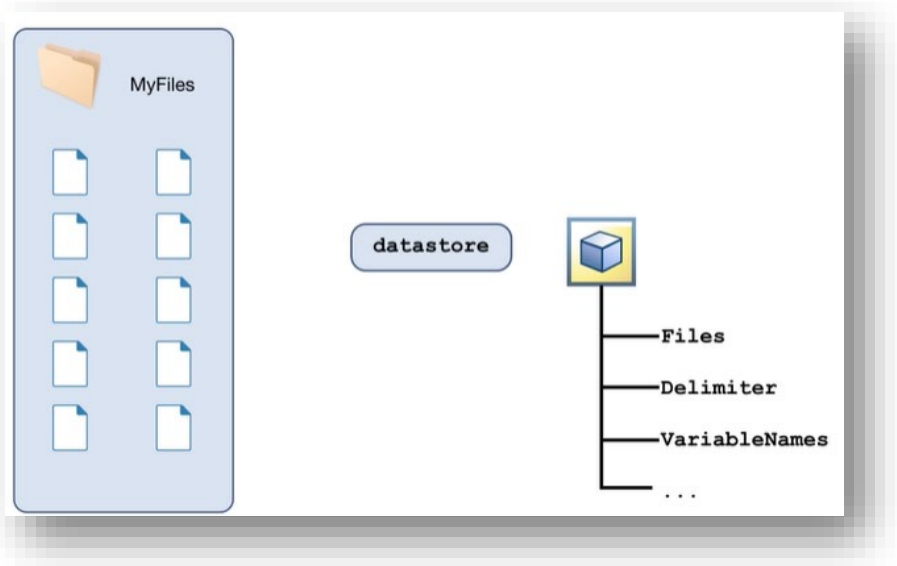


- If you're working with an existing data set, you have to work with the data as it's stored.
- If you can, → organization of your files for the most convenient arrangement. (Depends on the problem)

Machine Learning Workflow

Organizing data:

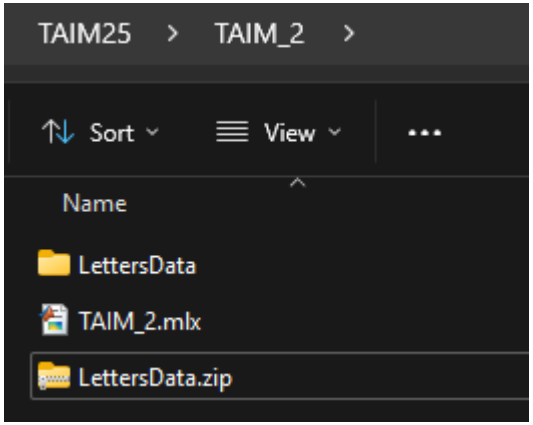
- However files are organized
- “datastores” variable provide a convenient way to access data stored in multiple files.
- MATLAB looks at all the data files in that location and returns a variable that contains information about the files and the format of their contents.
- Because there could be a lot of data, the data isn't imported until you ask for it.



Let's go to the Matlab implementation

Go to "My_TECH_SPACE"

1. Download the file: "LettersData.**zip**"
2. Copy them at your "TAIM_2" folder.
3. Uncompress it
4. It should look like this



Let's go to the Matlab implementation

At your "TAIM_2"

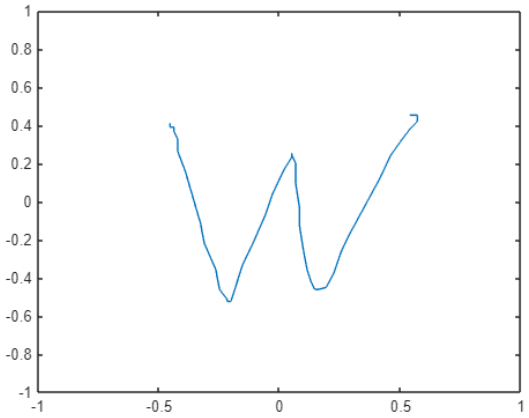
1. "addpath('./LettersData') % Add the relative path to "LettersData" folder to

% ... to the top of the search path for the current MATLAB session"

2. Let's read a 1st letter: "letter = readtable("user016_w_1.txt")"

3. Plot it: "plot(letter.X,letter.Y)"

4. Adjust axis: "axis([-1 1 -1 1])"



letter = 49×4 table

	Time	X	Y	P
1	0	-0.4529	0.4102	0.0030
2	0.0340	-0.4529	0.3997	0.1690
3	0.0340	-0.4529	0.3997	0.1690
4	0.0580	-0.4529	0.3892	0.2470
5	0.0630	-0.4371	0.3892	0.2590
6	0.0830	-0.4371	0.3682	0.3410
7	0.1000	-0.4214	0.3262	0.4250
8	0.1150	-0.4214	0.2632	0.5170
9	0.1320	-0.3899	0.1582	0.6240

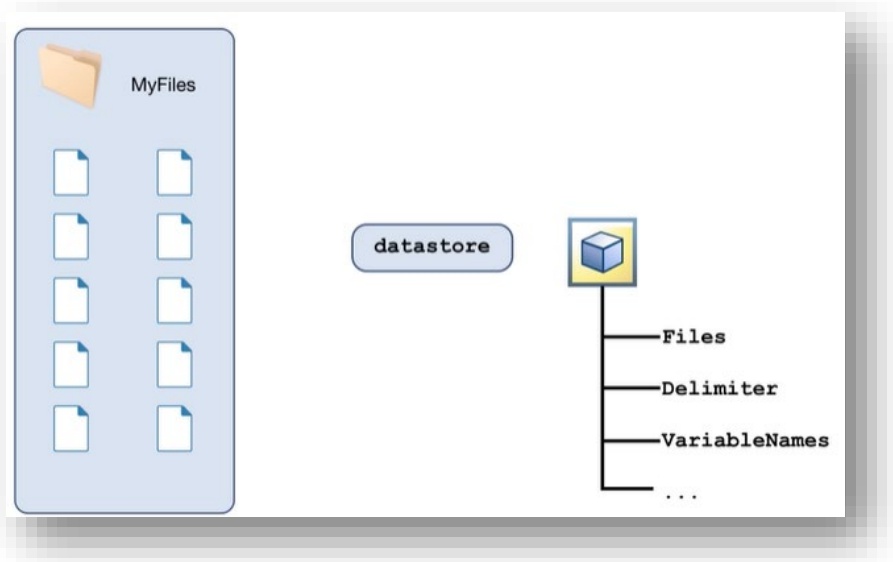
Let's go to the Matlab implementation

At your "TAIM_2"

1. Create the "datastore" (for exemple for all handwritten letters "r"):

"letterds = datastore("*_r_*.txt")"

¿?: You use * as a wildcard to make a datastore to files matching a particular pattern.
E.g.: the handwriting data files have names of the form user003_r_2.txt, that is XXX_r_XXX.txt, so two wildcards for this example are needed.

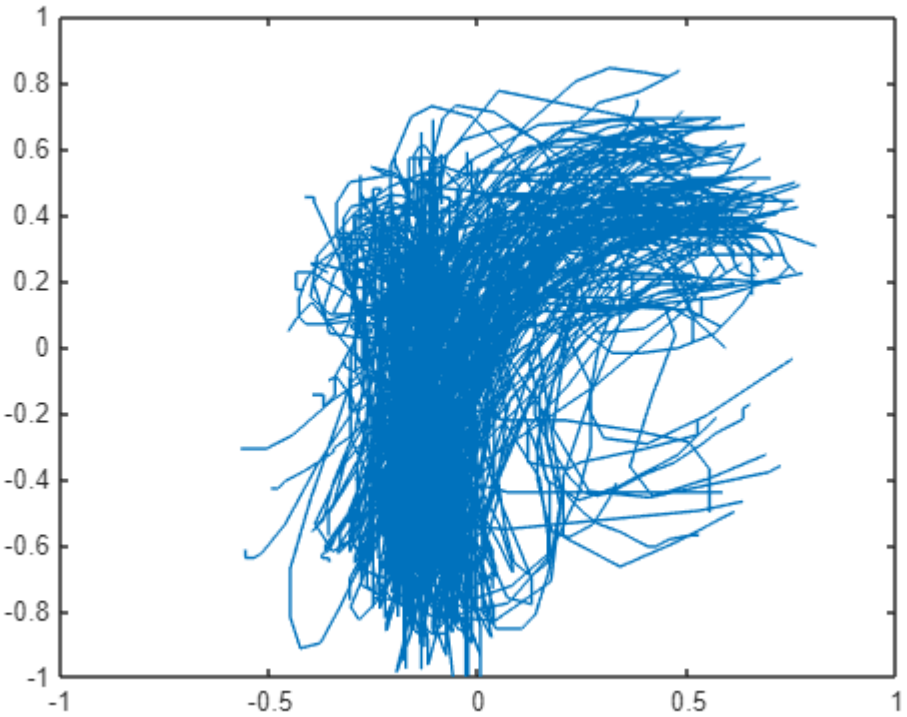


Let's go to the Matlab implementation

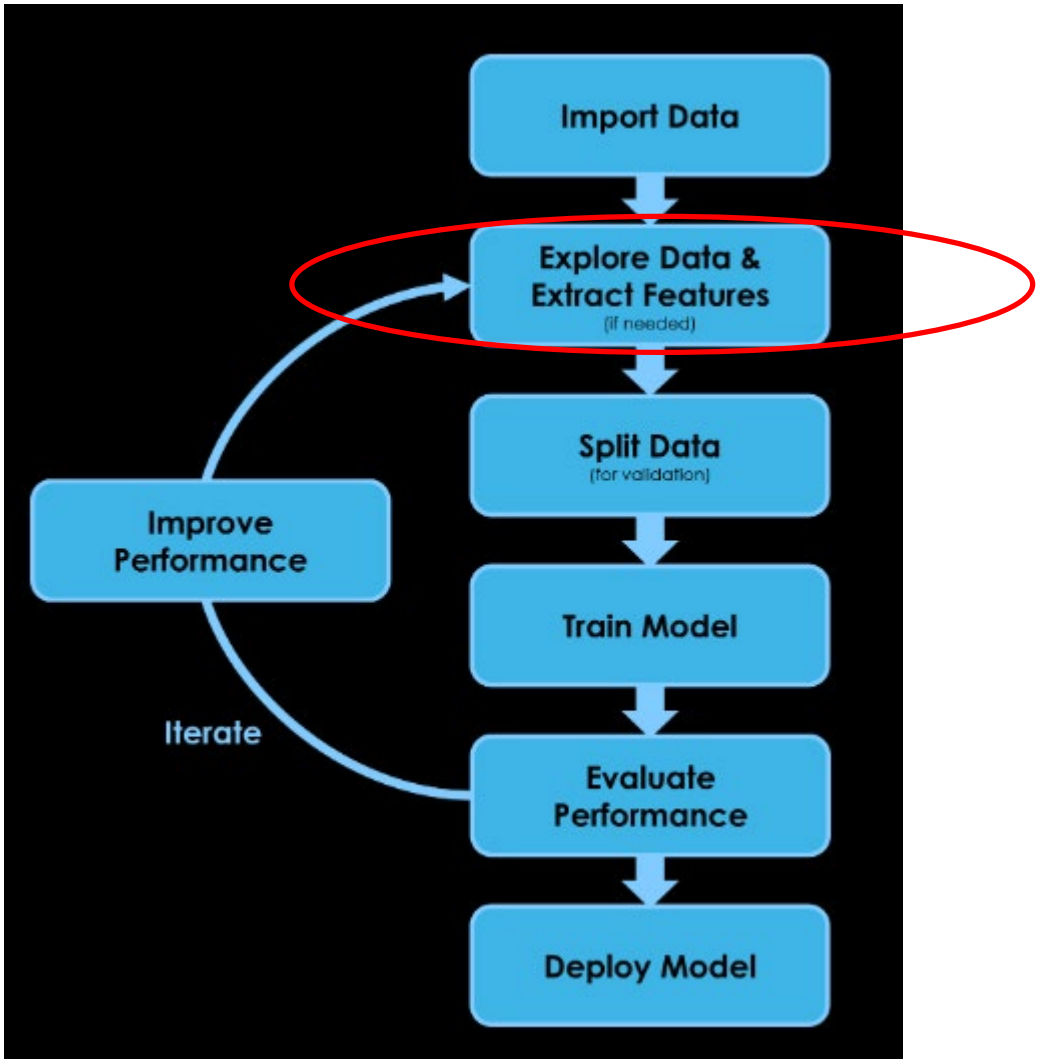
At your "TAIM_2"
2. Use the read function to read and show a 1st version of this letter:
"data = read(letterds)
plot(data.X,data.Y)
axis([-1 1 -1 1])"

3. A second one:
"data = read(letterds)
plot(data.X,data.Y)
axis([-1 1 -1 1])"

4. Now all of them
"data = readall(letterds)
plot(data.X,data.Y)
axis([-1 1 -1 1])"



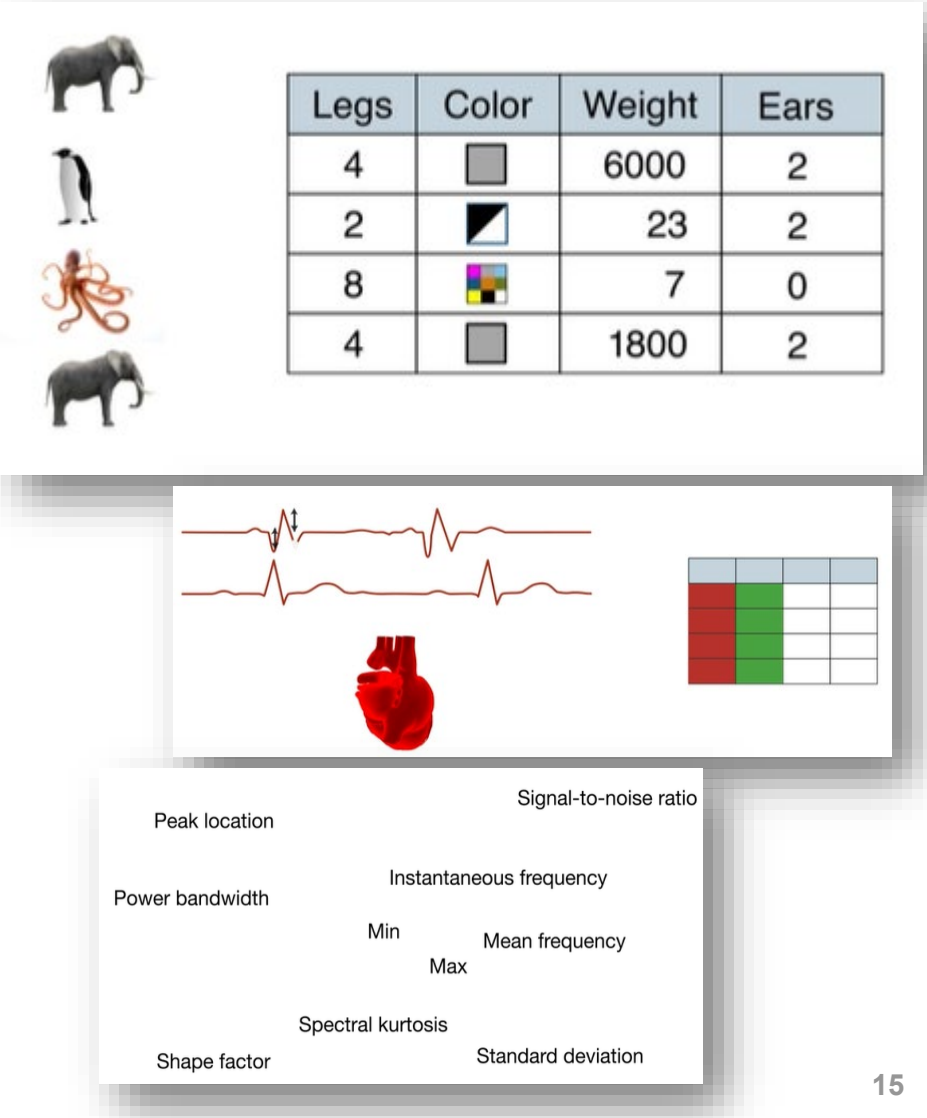
Next Step: Extra Features (What's that?)



TAIM - jose.antonio.lazaro@upc.edu

Next Step: Extra Features (What's that?)

- ML algorithms need data as a number of **observations** (Each observation consists of several features -> Row)
- Sometimes the data takes this form naturally. (E.g. each animal row)
- Not always. (Diagnosing a patient from a sensor like an EEG or EKG? Input is a signals through time.)
- 1st: transform the raw data into a set of features.



Next Step: Extra Features (What's that?)

- For our "letters" e.g.:
  - the letter "t" is likely to be tall and narrow, whereas the letter "m" is likely to be short and wide.
  - the letter "o" is quick to write, so the duration of the signal might also be a distinguishing feature.

Let's make an example of "observation table" with the letter "w".
"timeToWrite = letter.Time(end)
letterHeight = range(letter.Y)
letterWidth = range(letter.X)
features = table(timeToWrite, letterHeight, letterWidth)"

features = 1×3 table

	timeToWrite	letterHeight	letterWidth
1	0.8050	0.9767	1.0239

Next Step: Extra Features (What's that?)

- Perfect. Now we need to do this for ALL the letter examples that we have.
- 1st: Let's create a function to extract features:

"function features = extractLetterFeatures(letter)

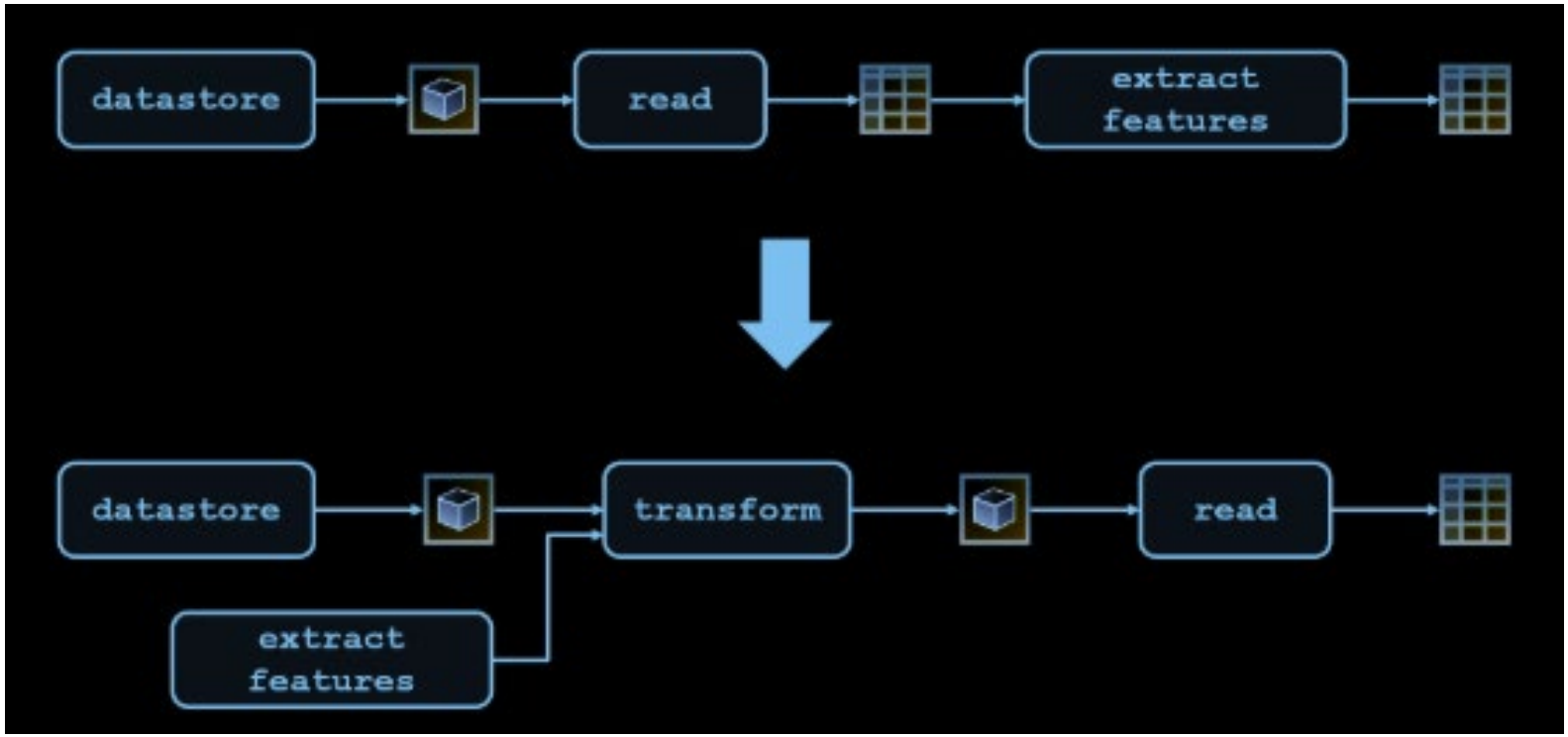
```
% Extract features
timeToWrite = letter.Time(end);
letterHeight = range(letter.Y);
letterWidth = range(letter.X);
firstXpos = letter.X(1);
lastXpos = letter.X(end-1);
firstYpos = letter.Y(1);
lastYpos = letter.Y(end-1);
numStrokes = sum(ismissing(letter.P));

% Combine features into a table
features = table(timeToWrite,letterHeight,letterWidth, ...
    firstXpos,lastXpos,firstYpos,lastYpos,numStrokes);
```

end"

Next Step: Transformed Datastores

- Instead of calculating the letter features one by one.
- Directly apply the same feature calculations to all data.
- By creating a transformed datastore (applies a function to all files in a datastore)



Next Step: Transformed Datastores

1. Let's display the text file names in the "LetttersData" folder: "ls("./LettersData/*.txt")"

2. Create a datastore for all of them: "letterds = datastore("./LettersData/*.txt")"

3. Use the transform function to apply the "extractLetterFeatures" function to each file in the "letterds" datastore:
"featds = transform (letterds, @extractLetterFeatures)"

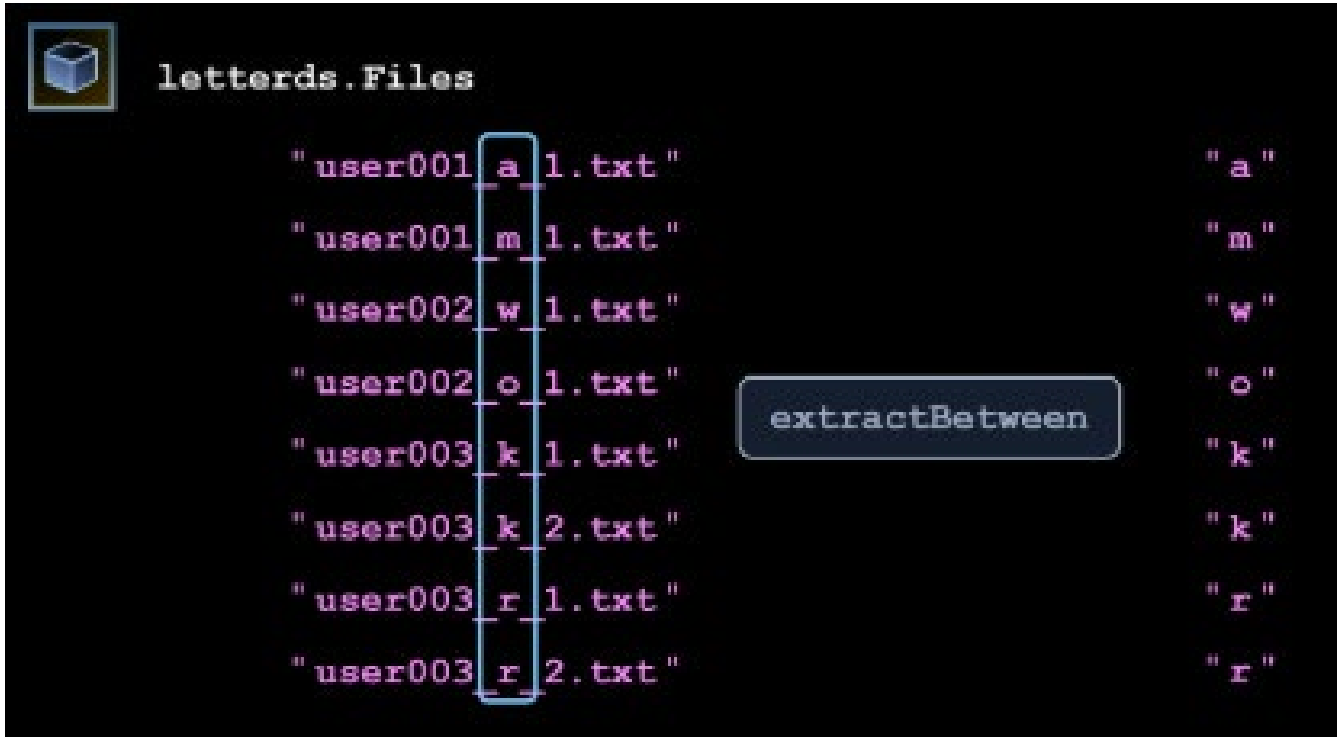
¿?: the @ symbol is "function handle" (A function handle is a reference to a function. Without the @ symbol, MATLAB interprets the function name as a call to that function, instead.) (Not needed to fully understand it now.)

```
featds =
  TransformedDatastore with properties:

      UnderlyingDatastores: {matlab.io.datastore.TabularTextDatastore}
    SupportedOutputFormats: ["txt"    "csv"    "dat"    "asc"    "xlsx"    "xls"
                Transforms: {[@extractLetterFeatures]}
               IncludeInfo: 0
```

Next Step: Labels

- We labels of the letter to classify the letters (to associate the features to the letters.
- The class labels are stored in the file names. (Let's use a text processing function to extract the letters from the file names.)



Next Step: Transformed Datastores

4. Use “readall” function to extract features from all the data files → into a table:
“data = readall(featds)”

5. Extract the text that occurs between given strings, by “extractBetween” function (extractedtxt = extractBetween(txt,"x","x")):
“c = extractBetween(letterds.Files,"_","_")”

¿?: The file names are stored in “letterds.Files”.

6. For classification problems, we prefer to represent the “known” label as a “categorical” variable:
“data.character = categorical(c)”

```
data.character = categorical(c)
```

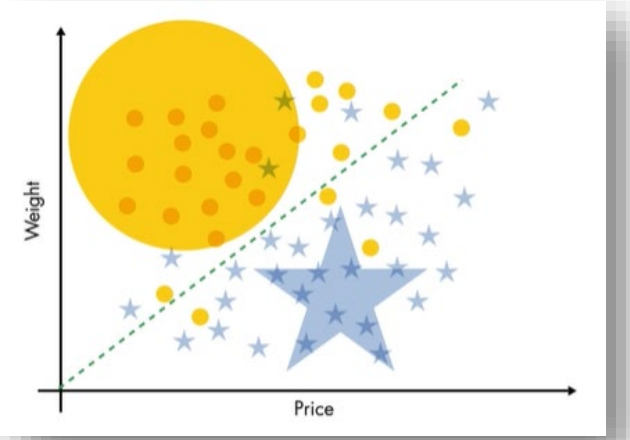
data = *1341×9 table*

	timeToWrite	letterHeight	letterWidth	firstXpos	lastXpos	firstYpos	lastYpos	numStrokes	character	
									1	2
1	0.5360	1.2517	0.7989	-0.0297	0.4097	0.1179	-0.6544	1	Drive\TAIM25\TAIM	a
2	0.3400	2.1106	0.4738	-0.1096	0.3426	1.0501	-1.0462	1	Drive\TAIM25\TAIM	h
3	0.4550	1.8313	0.5461	-0.0094	0.2885	0.9800	-0.6637	1	Drive\TAIM25\TAIM	k
4	0.4560	1.0518	0.9507	-0.3139	0.5559	0.2297	-0.3366	1	Drive\TAIM25\TAIM	m

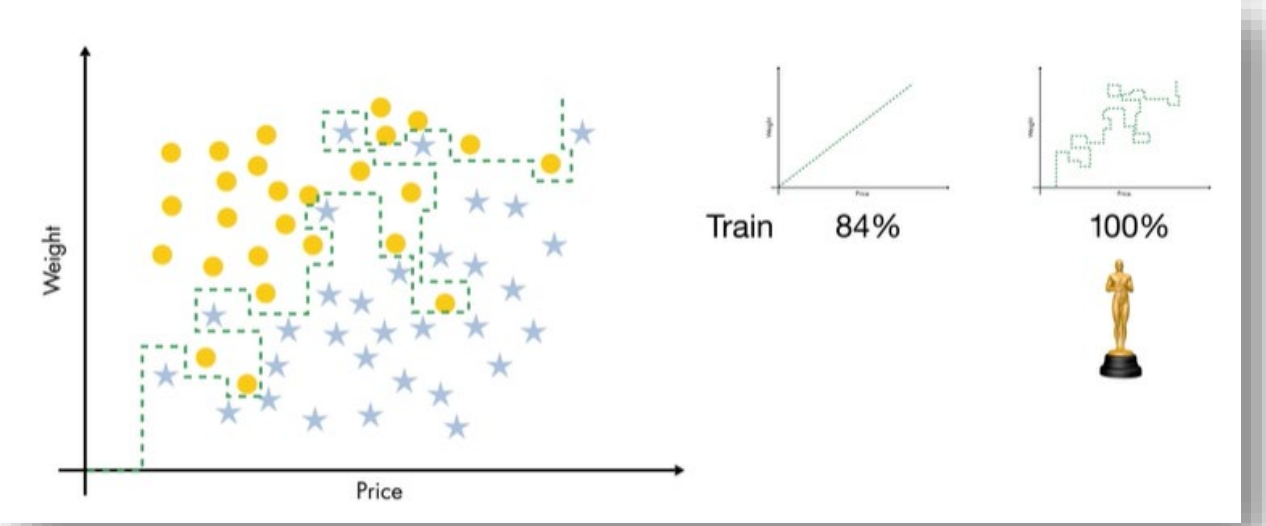
Next Step: Training your ML! (Things to think first)

1. Do you remember the problem of overfitting at “Curve Fitting”?

- Let’s suppose we have to classify data with 2 predictor variables and 2 output classes.
- Simple model? (If price is greater than weight, then star. Otherwise, circle.)
- What about a more complex model?

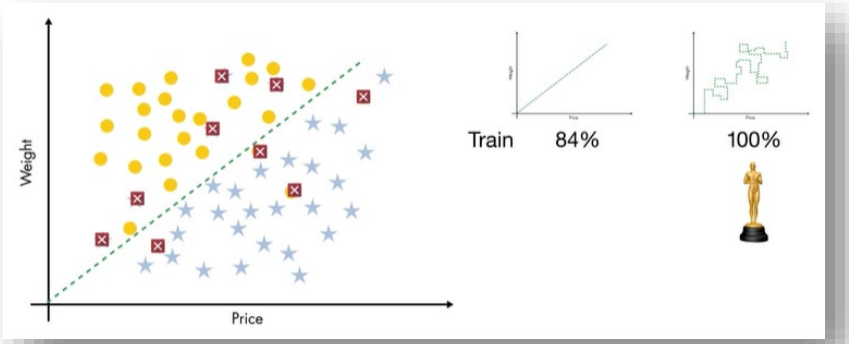


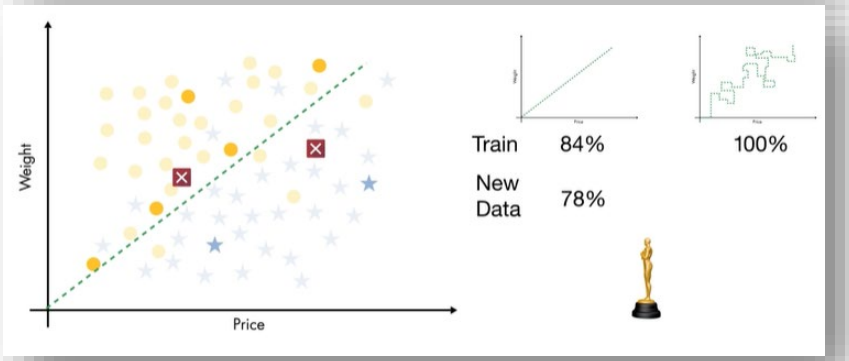
- If graded by their accuracy on the training data, the second model it's perfect.

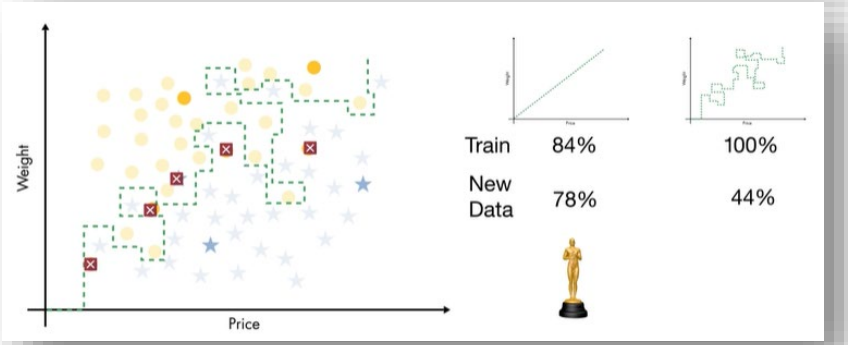


Next Step: Training your ML! (Things to think first)

- While the simple model makes some errors.
- The simple model on new data, is OK.
- And the complex model isn't as good as it promised to be.
- It's a common problem in machine learning, **overfitting**. (when the model performs really well on the data used to train it, but doesn't generalize well to new observations)



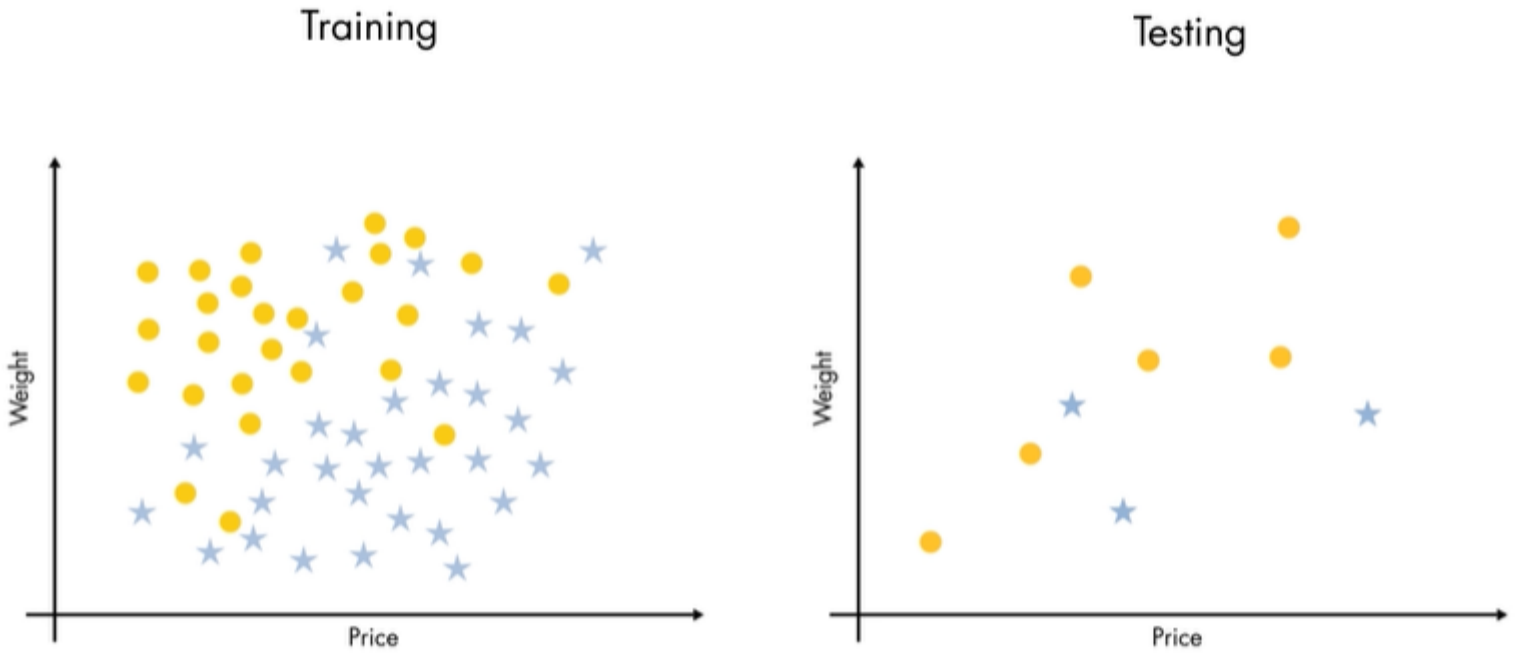




Next Step: Training your ML! (Things to think first)

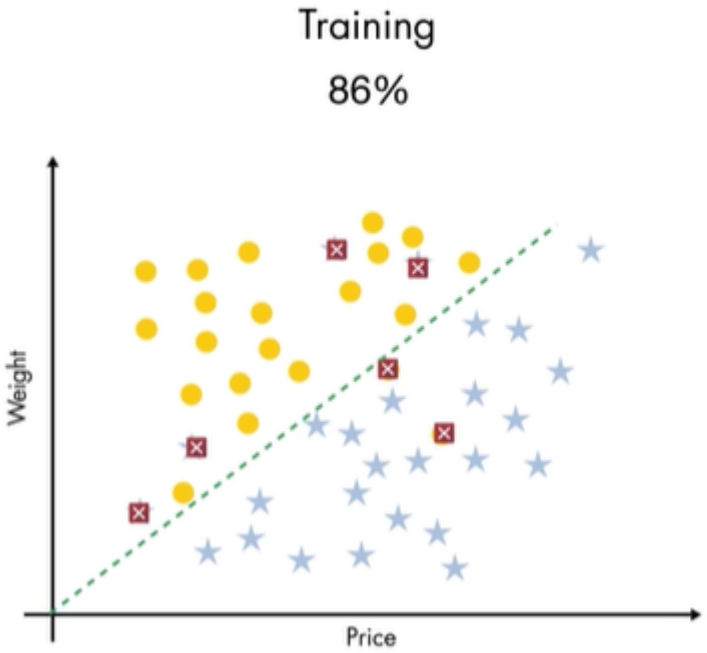
- How “to test” if overfitting on new data? (We have only “the data”)

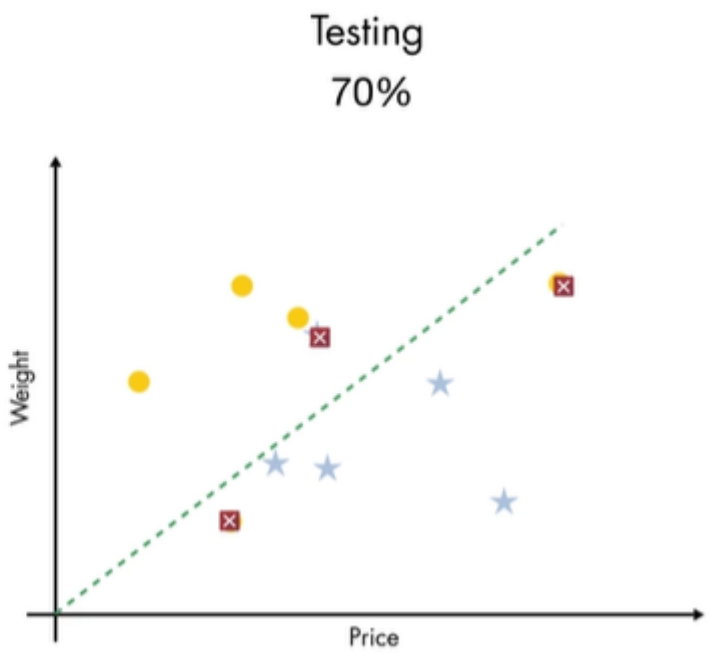
1. Separate “the data” into “Training” and “Testing”



Next Step: Training your ML! (Things to think first)

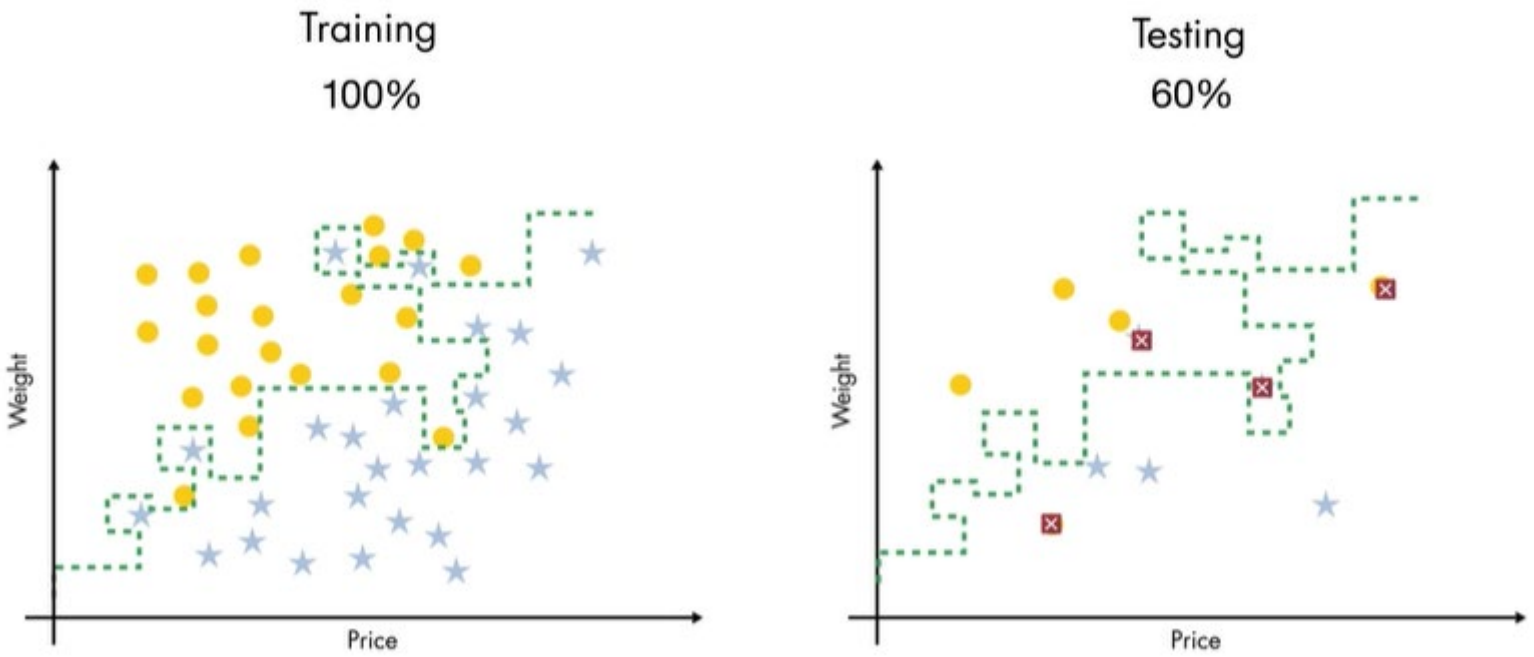
- Maybe the simple model has some limited performance.





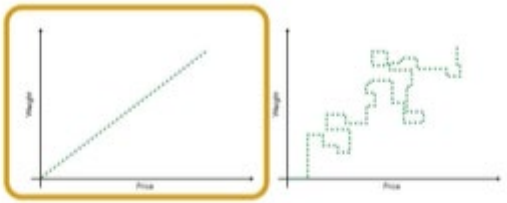
Next Step: Training your ML! (Things to think first)

- But maybe the more complex model is even worse.



Next Step: Training your ML! (Things to think first)

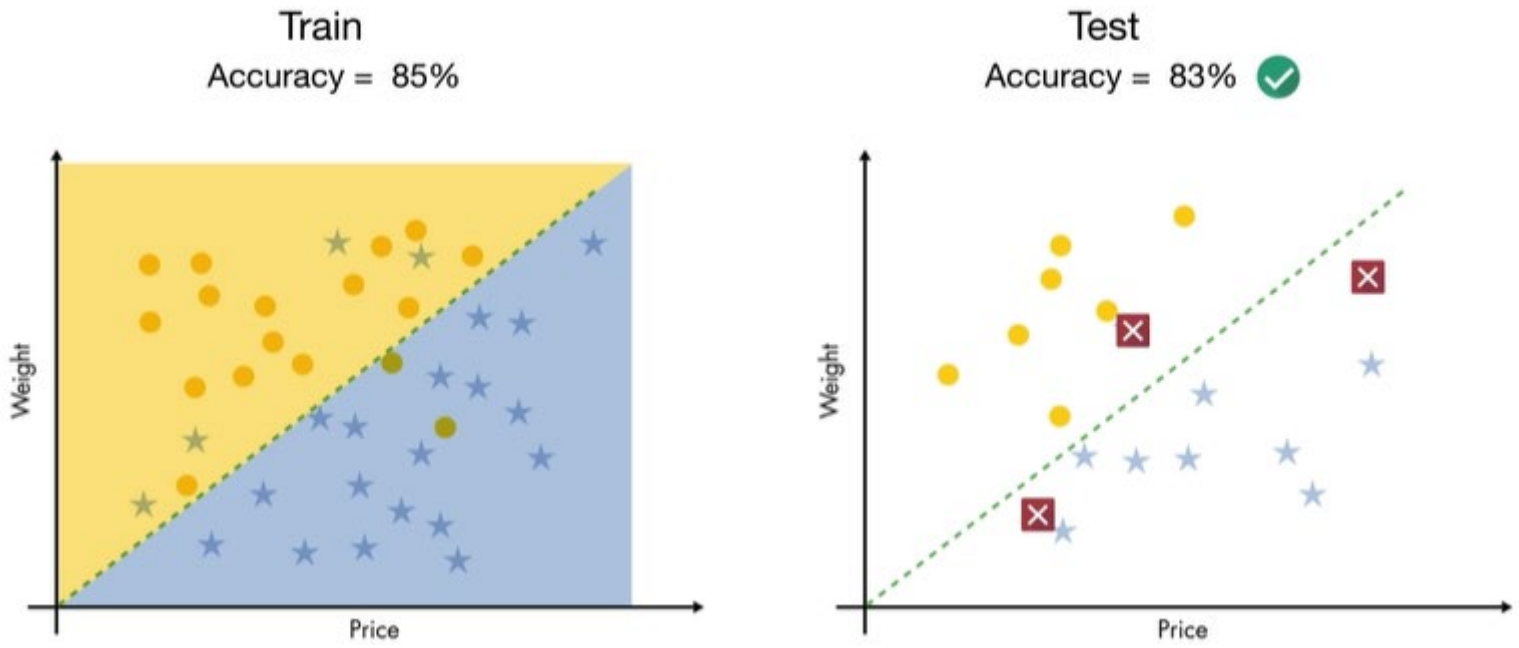
- At the end, maybe the simple model is the best?
- Well, we should have any model, someway.



Train	86%	100%
Test	70%	60%

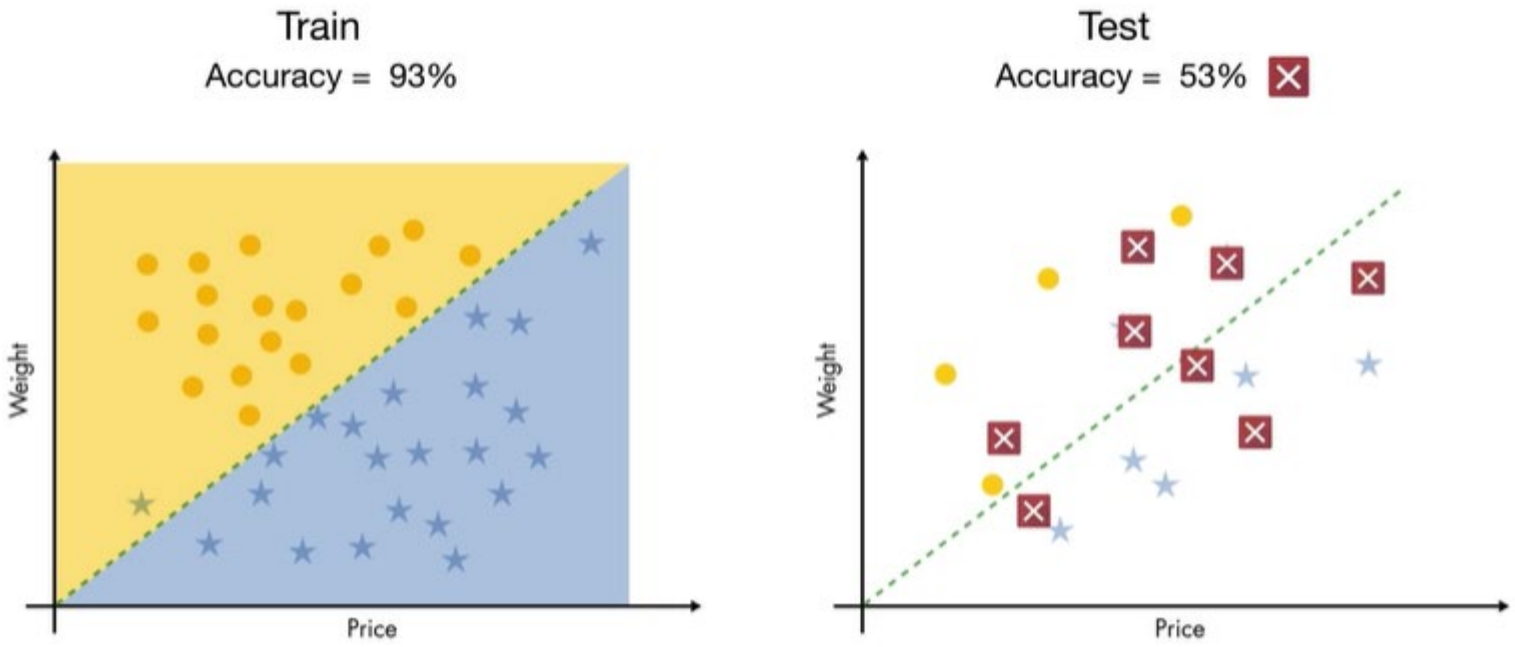
Next Step: Training your ML! (Things to think first)

- All this is OK if we have enough data & the test set is a good statistical representation of the whole data set.



Next Step: Training your ML! (Things to think first)

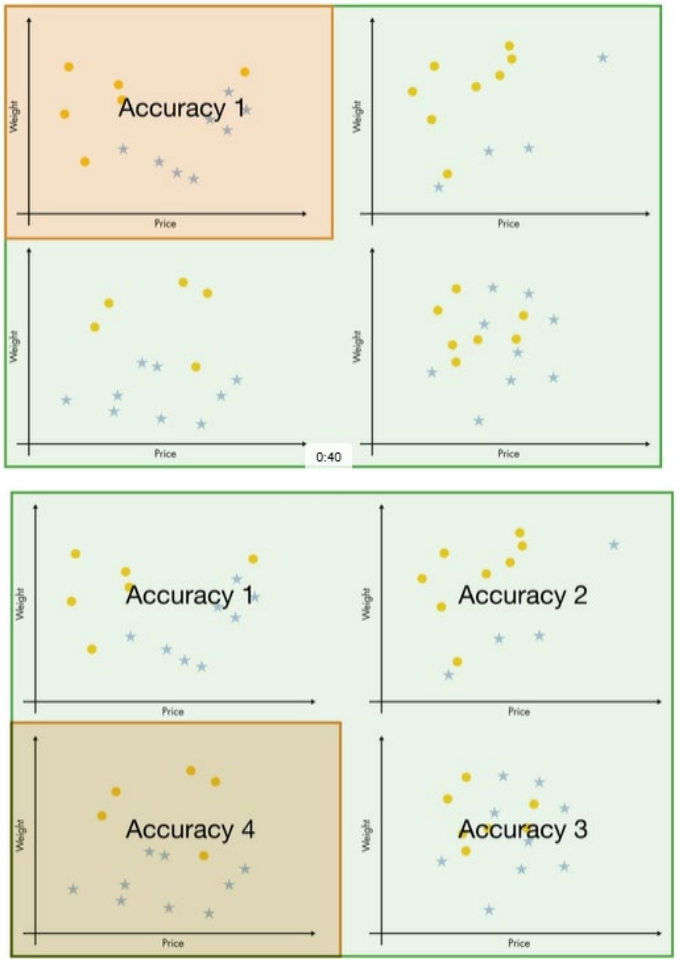
- It doesn't work when the test set contains data that doesn't reflect the rest of the data.



Next Step: Training your ML! (Things to think first)

Alternative solution: “k-fold validation”:

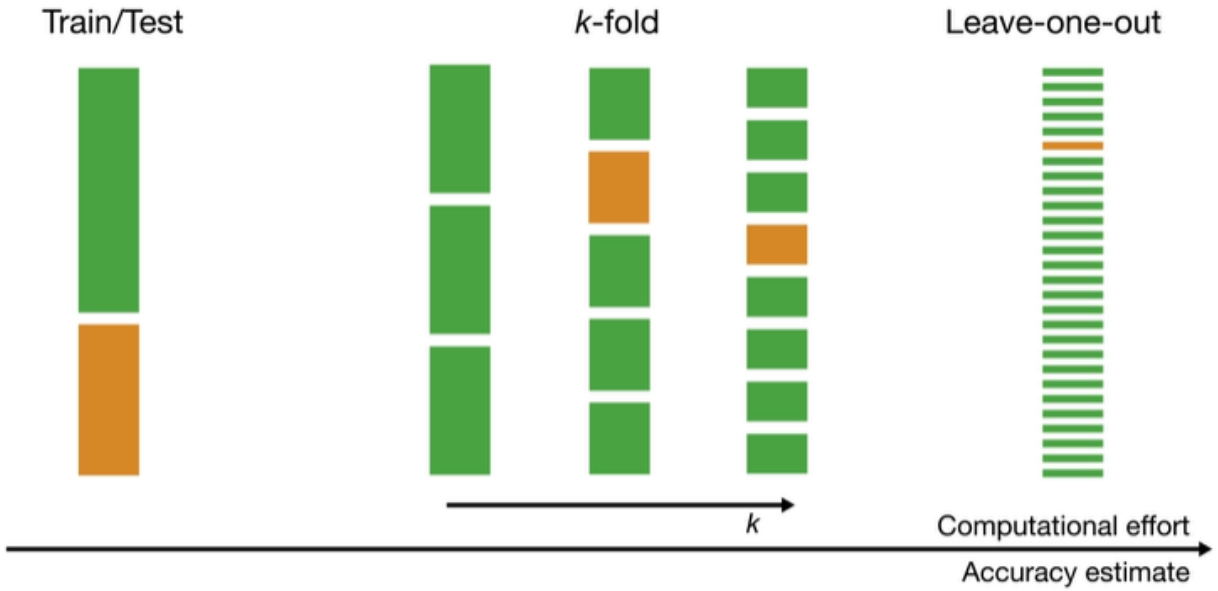
- Data randomly divided into k sets, known as folds.
- One of those folds is reserved as the validation set to measure performance & the rest of the data is used for training.
- Then, the process repeats with a different fold reserved for validation each time until all folds have been used once as the validation set.
- The average accuracy from all the folds is the validation accuracy.



Next Step: Training your ML! (Things to think first)

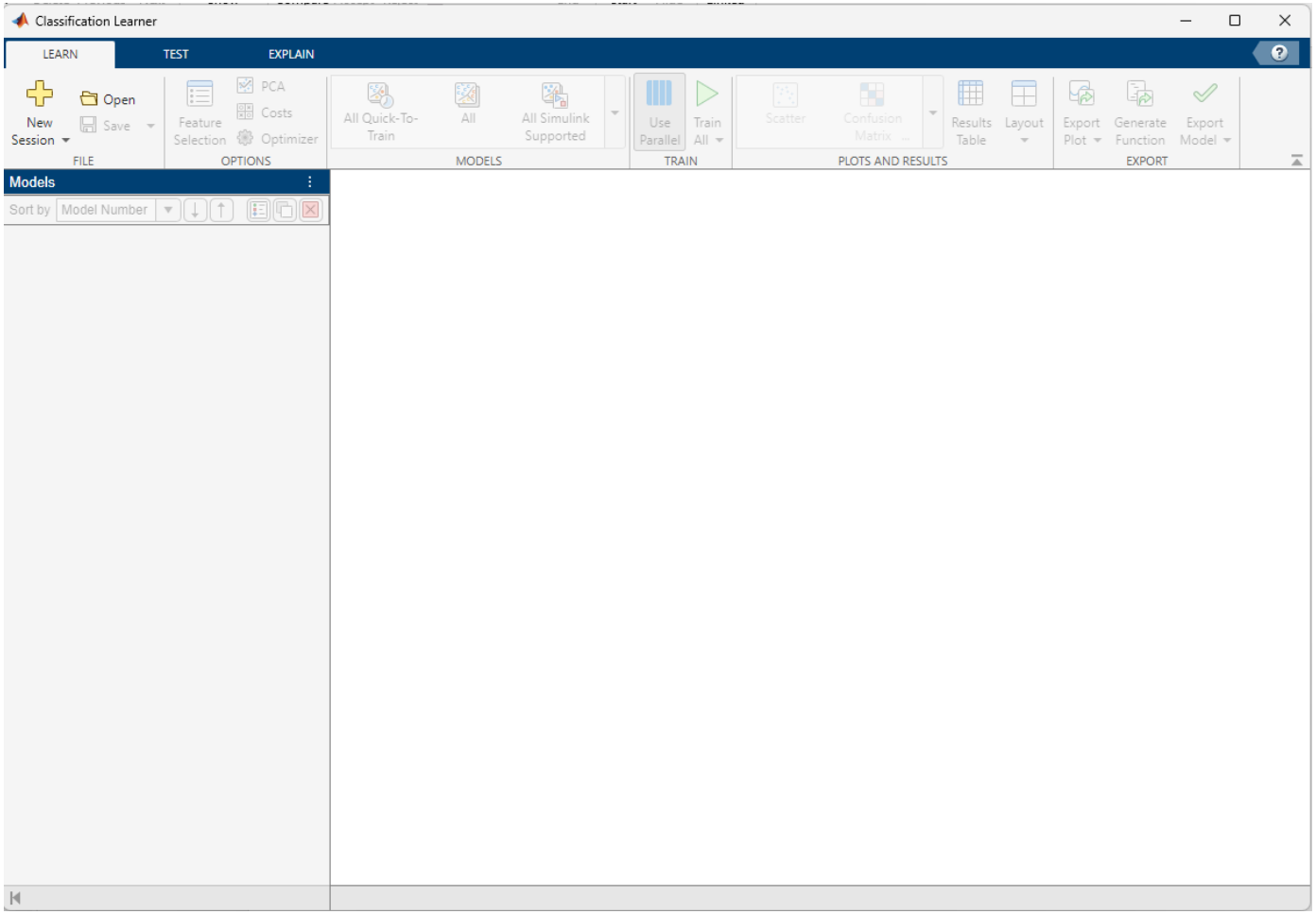
Alternative solution: “k-fold validation”:

- The average accuracy from all the folds is the validation accuracy.
- This process reduces the validation accuracy’s dependency on the particular way your data happened to be divided.
- But, more computational effort is required to fit and evaluate multiple models. (The bigger the value of k, the more computation, but the more robust the validation accuracy)



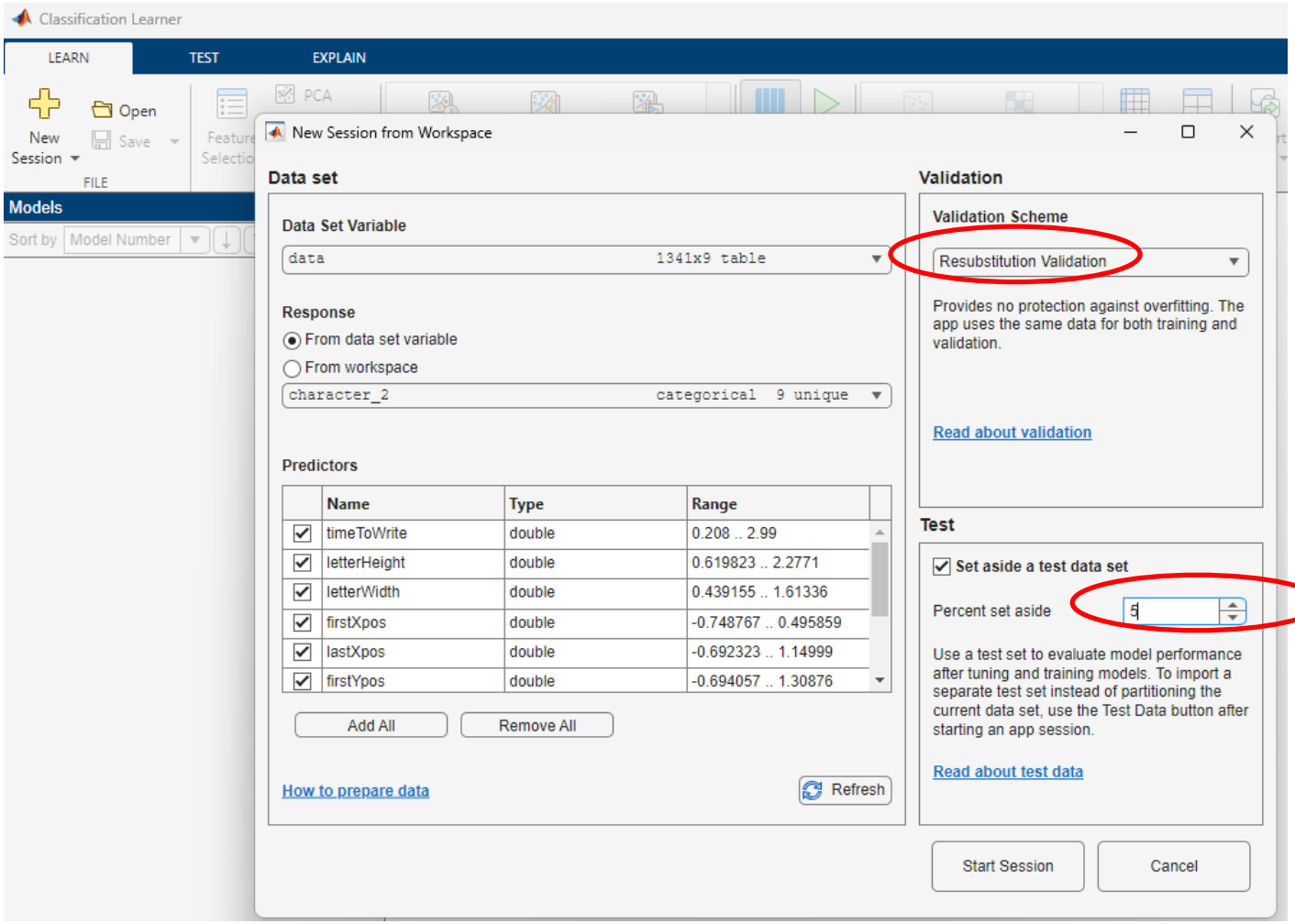
We have an App: Classification Learner

- Open the Classification Learner App



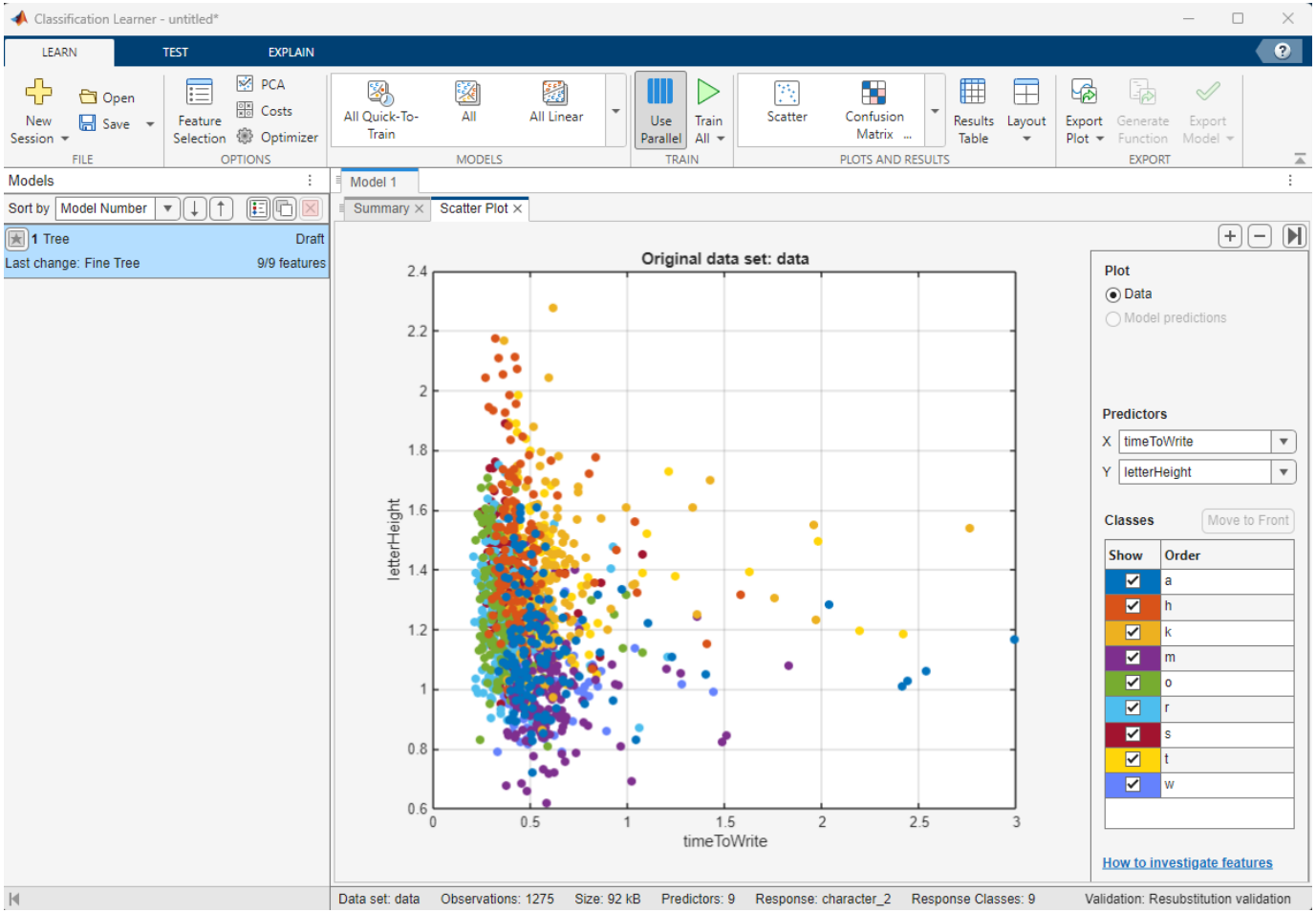
We have an App: Classification Learner

- Start “New Session” and load the “data”



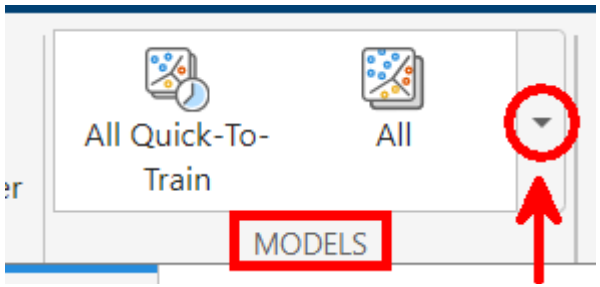
We have an App: Classification Learner

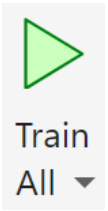
- By default a “Tree” algorithm is used



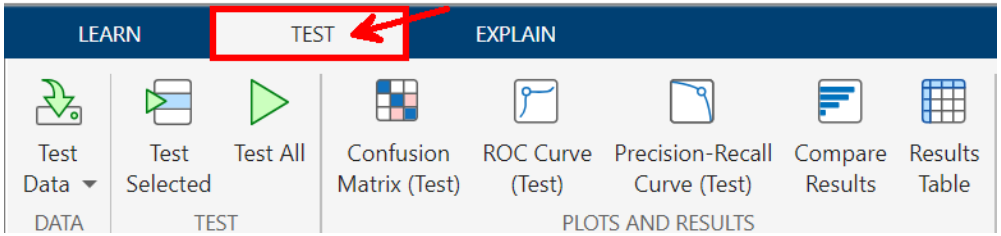
We have an App: Classification Learner

- Select the models you want to train from the Models section. (E.g. "Medium Gaussian")





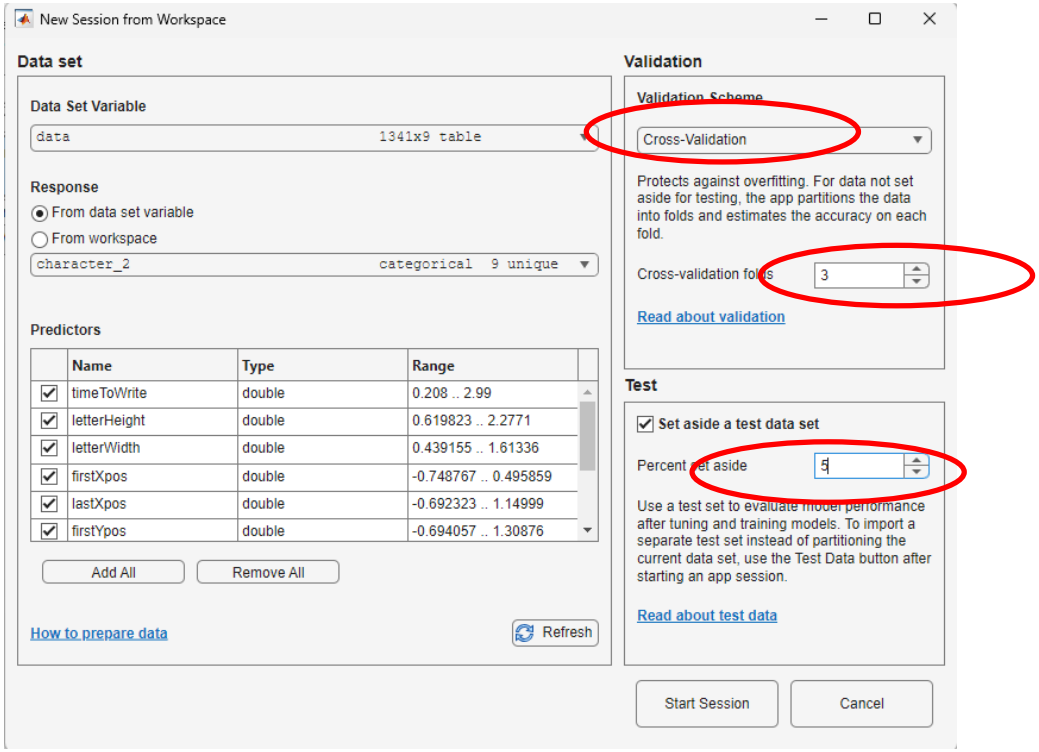
- To train the models, click **Train All**.
-
- If you have test data, you can access the test options from the Test Tab.



Do your own TEST

- Compare this training process with “Resubstitution Validation” (Train and Test data)
- With another “New Session” with “Cross-Validation” (k-fold validation)

- Compare for the same “Medium Gaussian”



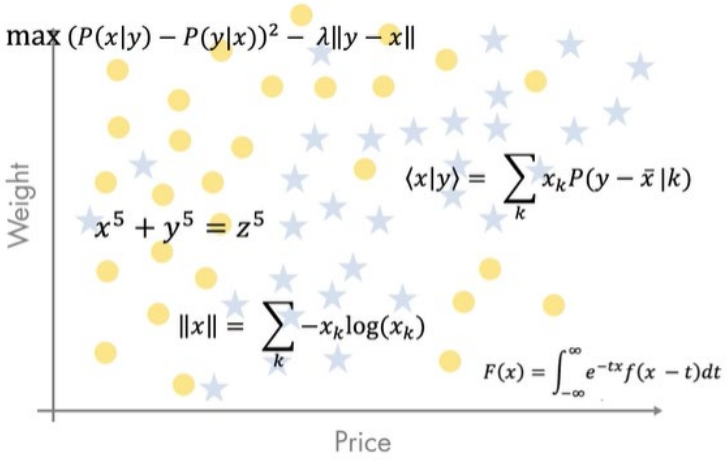
Do your own TEST

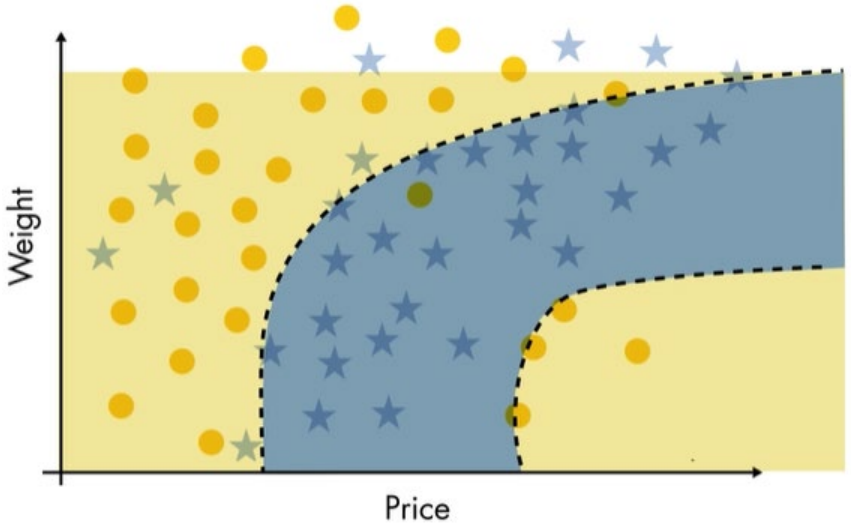
- Does the “Cross-Validation” (k-fold validation) works better than the with “Resubstitution Validation” (Train and Test data)?

- Document your results (Demonstrate with data results your answer)

How Classification Methods Work?

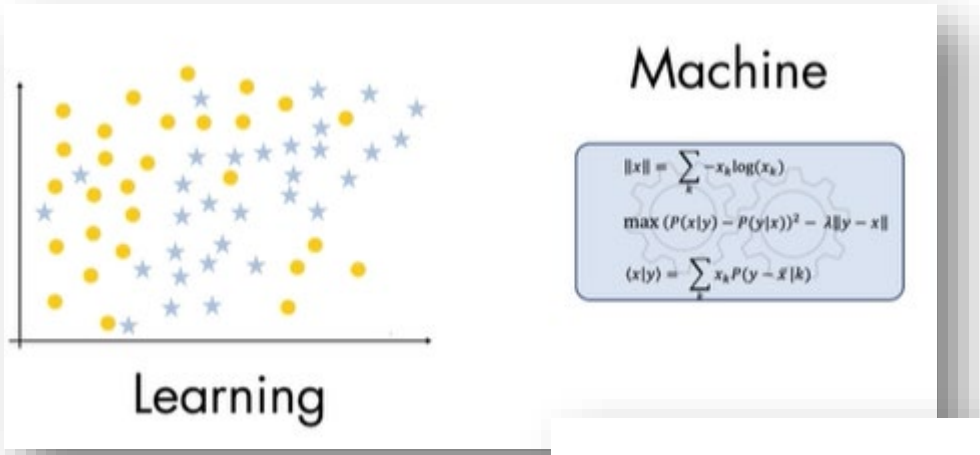
- A classification model is just a division of the feature space into regions, each labeled with one of the desired output categories.
- How is that division determined? By applying a given procedure, known as a machine learning method, **it's a recipe** that describes how to get a model from the given data.

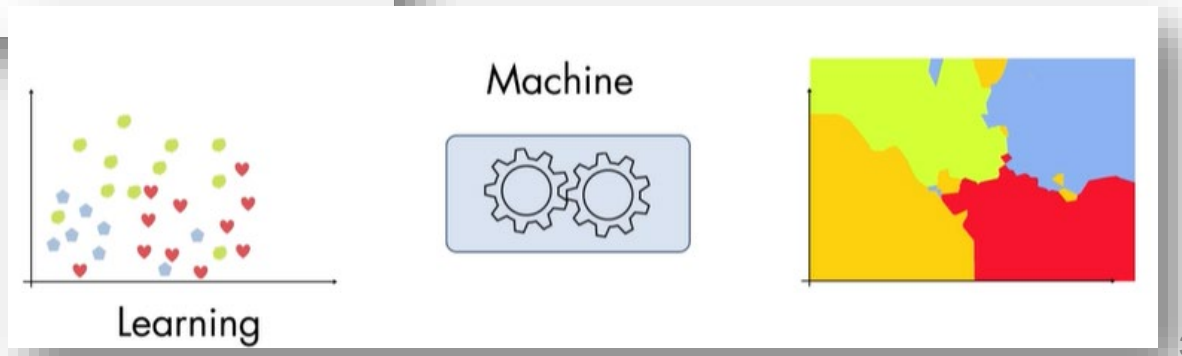




How Classification Methods Work?

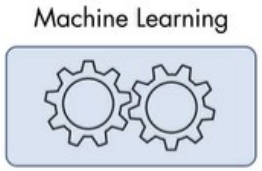
- That's makes “machine learning”
  - "Machine" because it's a computer following a **recipe**.
  - "Learning" because the end model depends on the training data it sees. (The machine has learned from the data.)

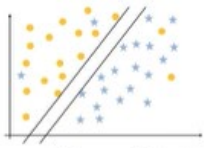




How Classification Methods Work?

- There are several machine learning methods to work well.
- There's no one method that's always the best.
- You don't need to know the details of how each method works.
- Usually the best approach is just to try them and see.

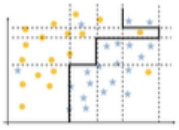



Support Vector Machine


k-Nearest Neighbors (kNN)


Naïve Bayes

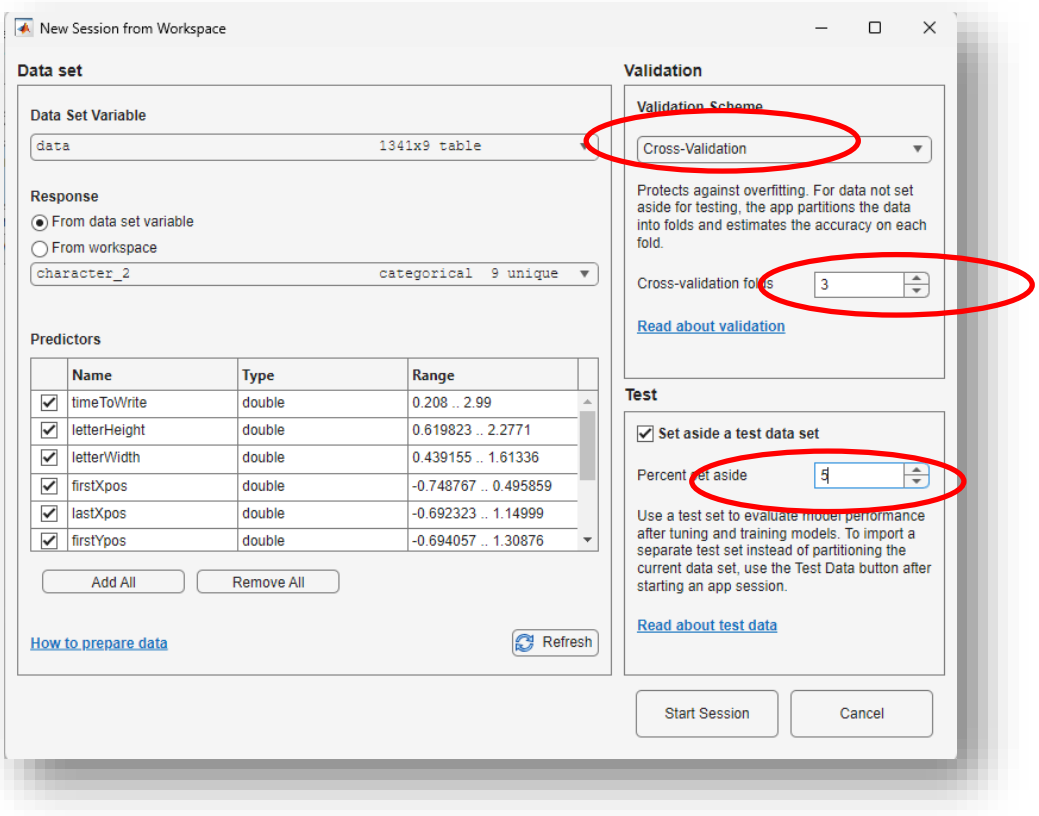

Discriminant Analysis

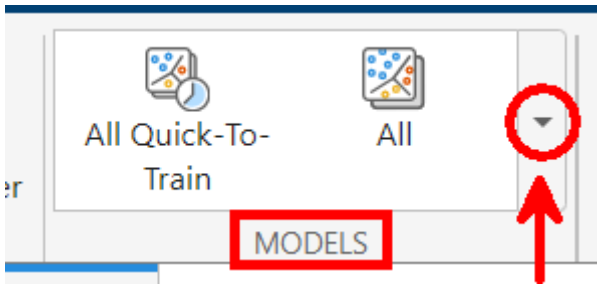


k-Nearest Neighbors (kNN) ✓
Support Vector Machine ✓
Naïve Bayes ✗
Discriminant Analysis ✓
Decision Tree ✗

Let's try to do it (Try them all!)

- Start a new Classification Learner session with 3-fold cross-validation

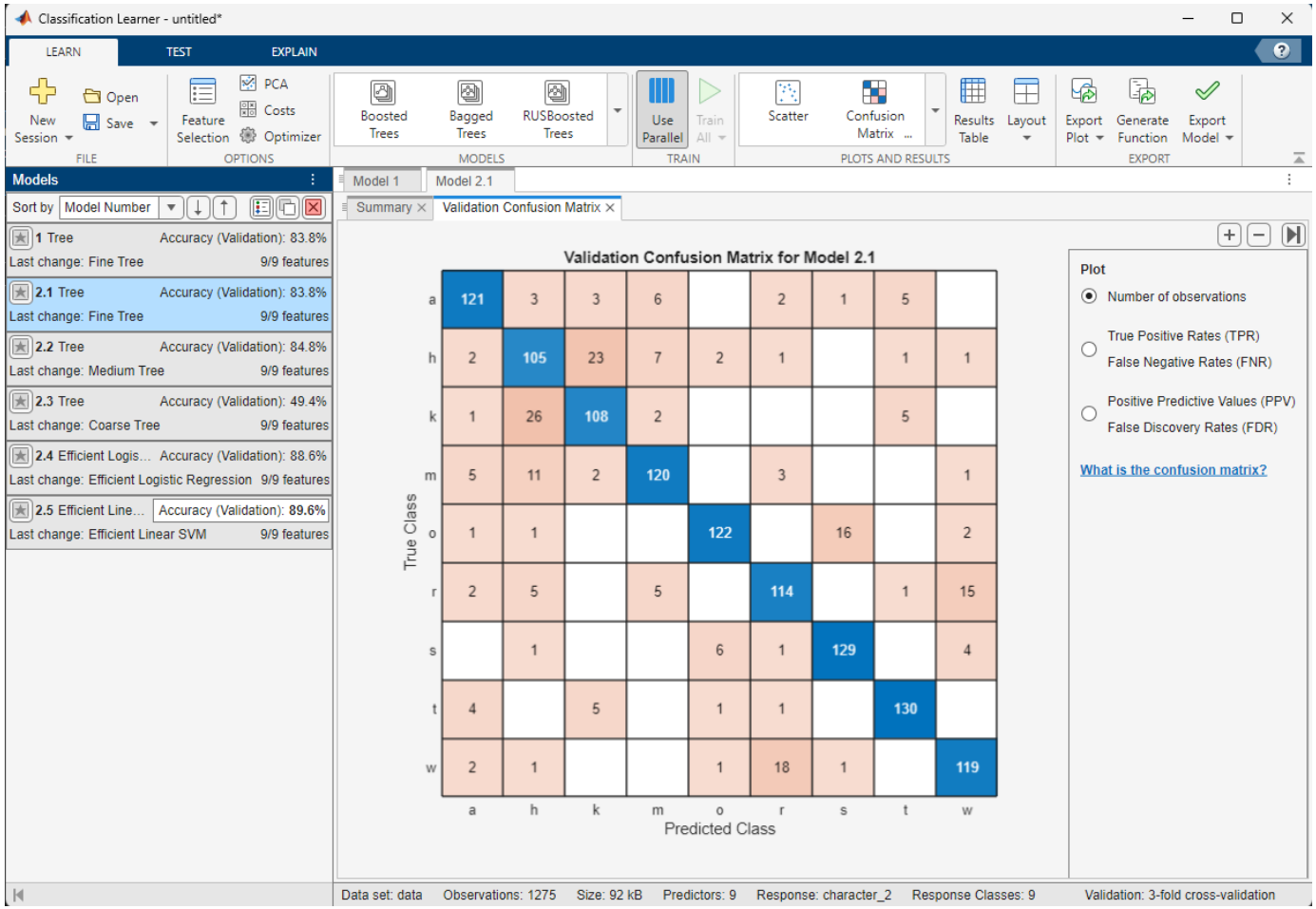




- And do a "All Quick-To-Train" & "Train All"

Let's try to do it (Try them all!)

- You will get something as:



How to measure the best method?

- We can: A) “Test All” (at “TEST”); B) Get the “Results Table”

Model 1 | Model 2.1 | Results Table ×

Session: untitled

Training Data: data Observations: 1275 Predictors: 9 Response Name: character_2 Response Classes: 9

Validation: 3-fold cross-validation

Test Data: data Observations: 66

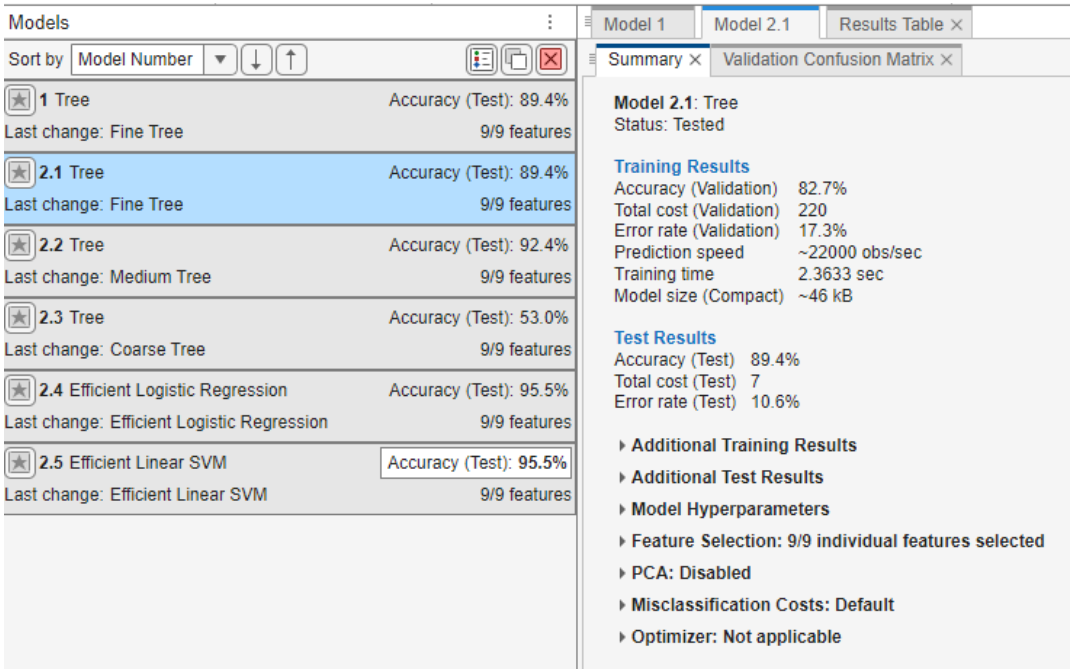
Select Columns | Plot Results | Export to Workspace | Export to

Favorite	Model Number	Model Type	Status	Accuracy (Validation)	Total Cost (Validation)	Accuracy (Test)	Total Cost (Test)
☐	1	Tree	Tested	82.75 %	220	89.39 %	7
☐	2.1	Tree	Tested	82.75 %	220	89.39 %	7
☐	2.2	Tree	Tested	81.88 %	231	92.42 %	5
☐	2.3	Tree	Tested	49.73 %	641	53.03 %	31
☐	2.4	Efficient Logistic Regression	Tested	88.39 %	148	95.45 %	3
☐	2.5	Efficient Linear SVM	Tested	89.10 %	139	95.45 %	3

- Which is the most accurate method?

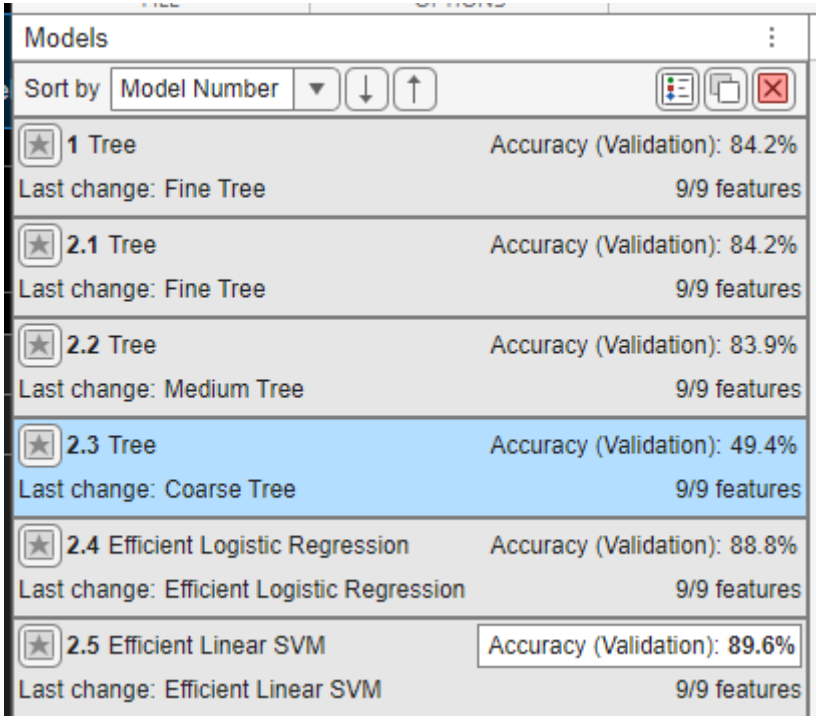
How to measure the best method?

- Could you guess which model was the fastest model to train?



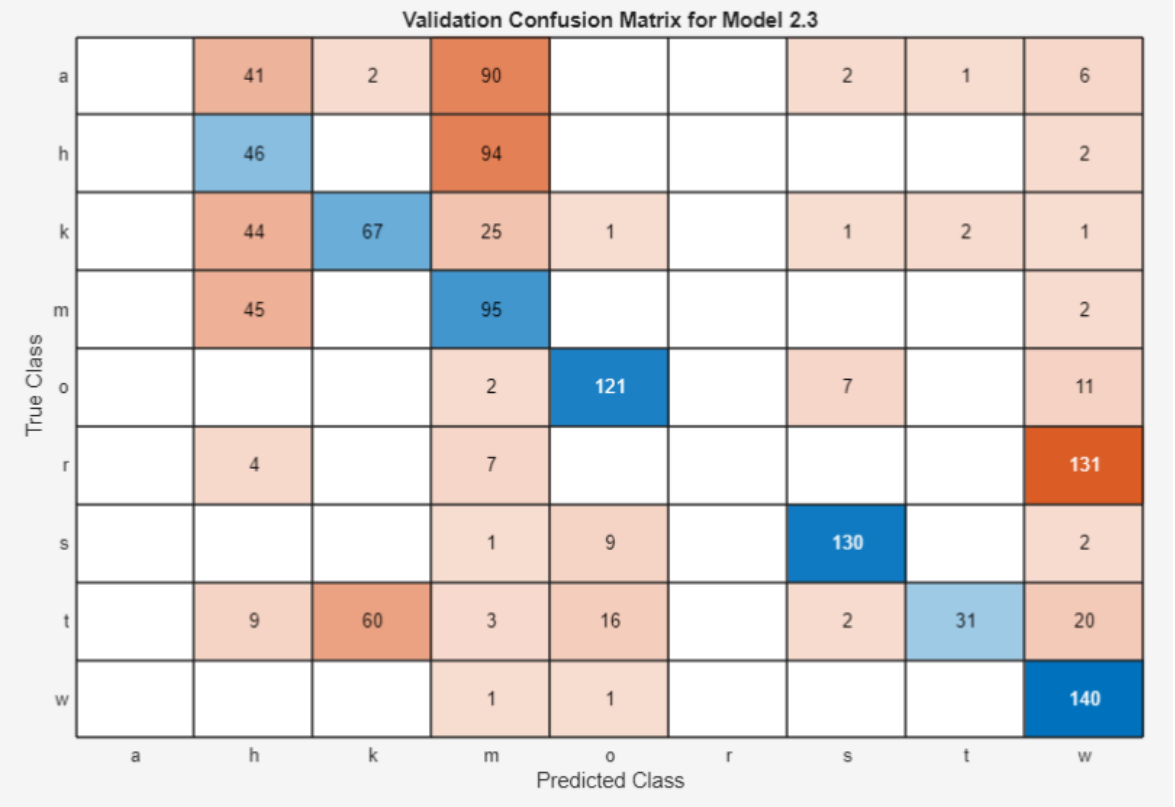
How to measure the best method?

- By the way, the 2.3 Tree shows the lowest Accuracy



How to measure the best method?

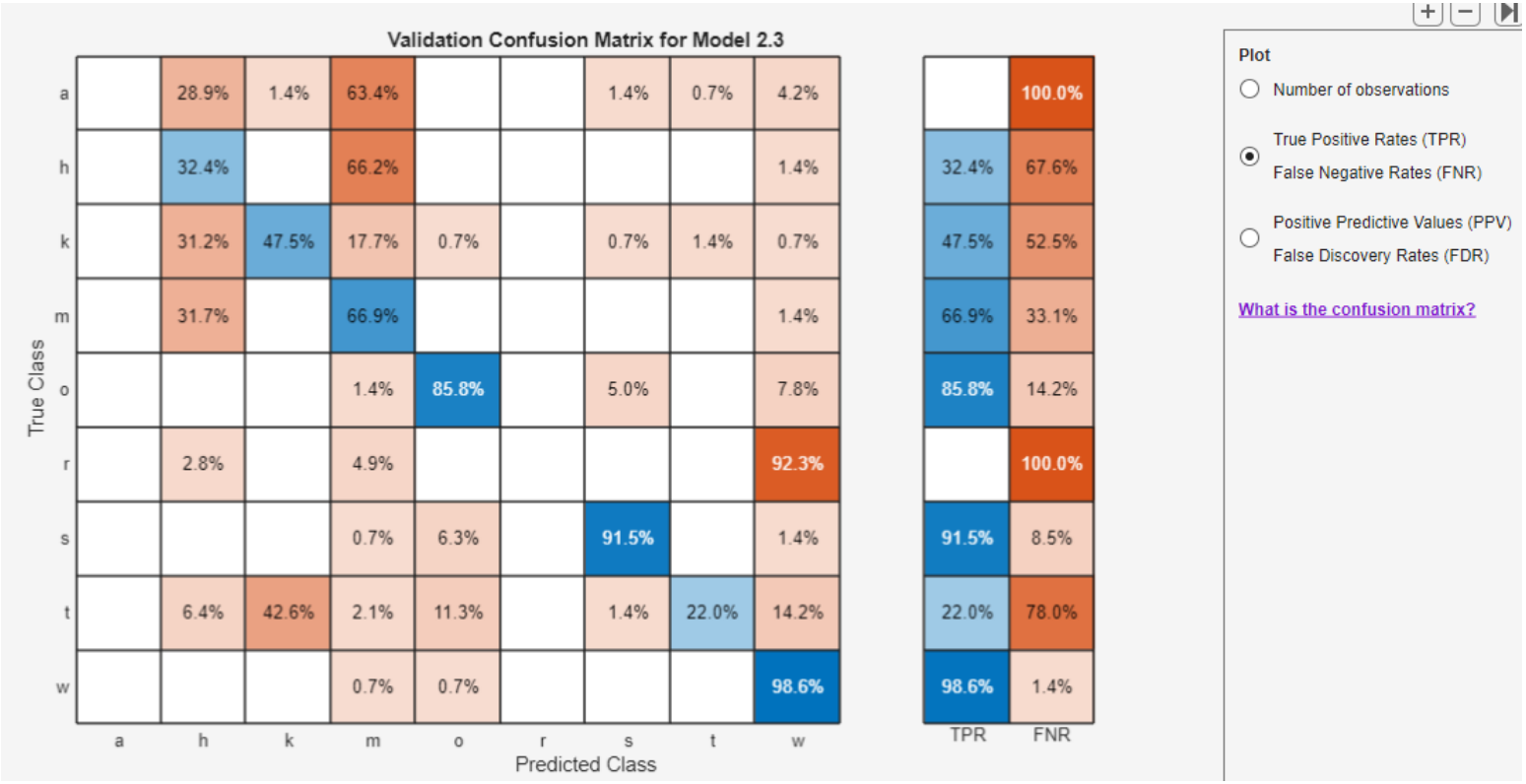
- To see more details on the 2.3 Tree Accuracy, we can see the "Confusion Matrix":



- What happened with "a" letter and "r" letter?

How to measure the best method?

- To see more details on the 2.3 Tree Accuracy, we can see the "Confusion Matrix":



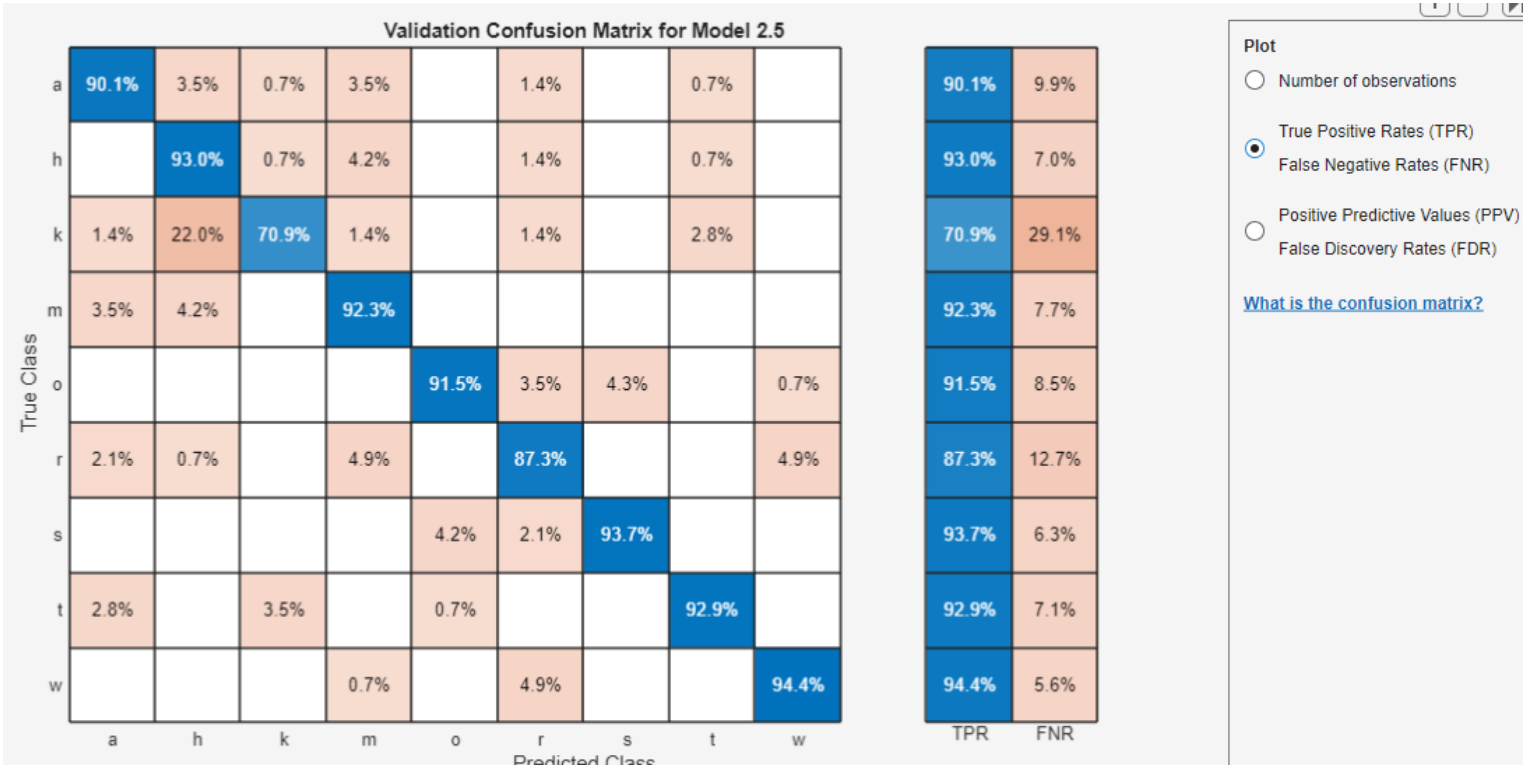
- What happened with "a" letter and "r" letter?

The Confusion Matrix

- For any response class X, you can divide a machine learning model's predictions into four groups:
  - True positives – predicted to be X and was actually X
  - True negatives – predicted to be not X and was actually not X
  - False positives – predicted to be X but was actually not X
  - False negatives – predicted to be not X but was actually X

How to measure the best method?

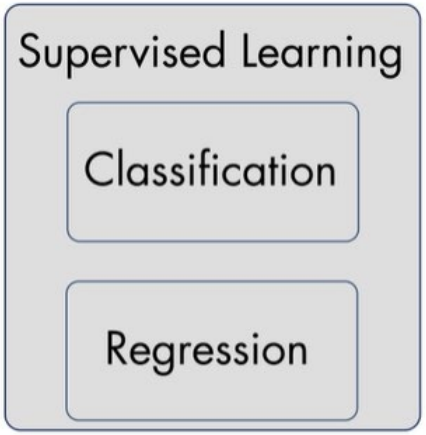
- NOW see more details on the highest accurate 2.5 “Efficient Linear SVM” “Confusion Matrix”:

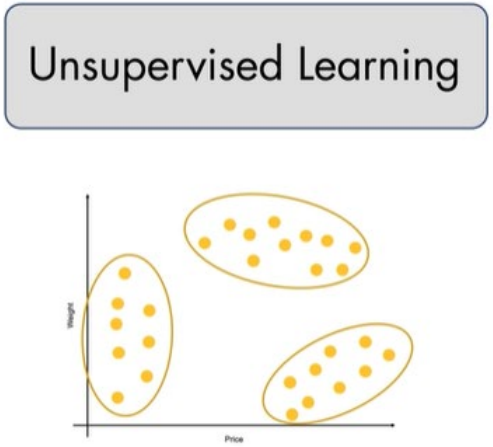


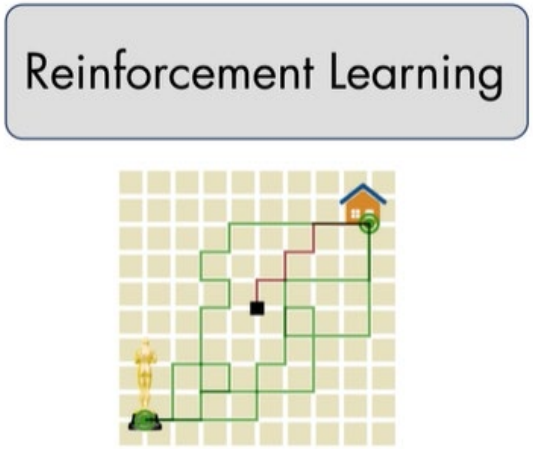
- Which are the letters that generates most confusion?

Summary

- We have focused on “Classification”
- Alternatively, e.g. trying to predict from the model is “Regression”
- All of the them as “supervised learning” (because your know the correct answer for the training.)
- If you just want to see if there's any structure or pattern in your data → Unsupervised Learning.







More resources (if you want)

- You have also examples at:
- https://uk.mathworks.com/help/stats/examples.html?s_tid=CRUX_topnav&category=classification

- If you are interested on trying to predict from the model, “Regression”, → There is also another App!

(This could be for another Seminar…
Not now!)