# Lecture\_7b

Hello and welcome to the second lecture in this week. We are still looking at machine learning implementation. In this lecture video we are going to be looking at classification models, particularly a typical dataset which can be used to build and train classification models. Demonstration in MATLAB, using the live script and the classification learner app. We are also going to be looking at regression models, specifically a dataset which can be used to build and train regression models.

Demonstration in MATLAB using live script and the regression learner app. As you can see, this is my current folder in MATLAB and this is the current path. This is the command window. Here is the workspace. I'm just going to open the dataset outside of MATLAB to view what we have. So as you can see, we have just one sheet. This is a well-organised dataset. We have to total of So, we have 4 predictors with variable names. Meas\_1, meas\_2, meas\_3, meas\_4 and these are the species for the Iris flowers and the species constitute the categories or the labels in this data. And as you can see as well, we have an equal number of observations in each category. We have a total of 3 categories with 50 observations in each category, giving us a total of As you can see, this is the table as well. The first thing that we want to do is to clear the workspace, clear the screen and close any opened window, any opened MATLAB window. We assign a variable name to our dataset, the file containing the dataset as you can see. Then we create a table using the built-in function readtable in MATLAB and if I do run on the classification app we can see, this is the table we have, which corresponds to the original dataset. We have hyperlink there that takes you to this example, the mathworks website. So, if I run the live script again. This time around, I'm opening the classification learner app in MATLAB by calling this.

Once the classification learner app is opened, we can start a new session because we now have our table in the workspace, so it should detect it. As you can see, we have 3 unique categorical variables and our predictors are meas\_1, meas\_2, meas\_3, meas\_4 and we are going to use all of the predictors. So, we add all and we are going to use cross-validation because we have just the 150 observations and we're going to use 5 folds because that's the very popular one. Now, we are going to build a very simple model to start with. So, we're going to use decision tree, fine tree. We're going to check the properties. We are not going to adjust this for now, we are not going to use any of the hyperparameters. You're just going to run. This is the scatterplot. So as you can see, it's finished training. The accuracy is 94%. That's the validation of prediction accuracy. The prediction speed is 3300 observations per second. The training time is 3.7324 seconds. We have not sure the hyperparameters. We have not used principal component analysis to reduce the dimensions.

Now, to analyse this model further, we can look at the confusion matrix because we have a total misclassification cost of 9, which means 9 observations were misclassified. And as you can see here, we have 6 observations that belong to versicolour label or category, classified as virginica, which is that predicted class, and we have 3 observations that have the true class of virginical, classified or predicted as versicolour. We can also look at the true positive rates and the false negative rates. We can also look positive predictive values and false discovery rates as well all from the confusion matrix or the error matrix. Importantly, we can look at the receiver operating characteristic curve. So for the setosa, if we go back to the confusion matrix, you can see this has all 50 observations correctly classified and that's why setosa's been the positive class here has an area under the curve to be equal to 1, which means the classifier is perfect when setosa is the positive class. Therefore versicolour, because we have 6 misclassifications there, the area under the curve is 0.94. That's the optimum operating point there. So this is still okay, and for virginica, the area under the curve is 0.95. So we can play around with the hyperparameters. For the decision tree, we can increase the number of splits for the model accuracy. We can also change the split criterion, accuracy is reduced. So we can also have surrogate decision split. So as you can tell, when we increase the number of splits, even though the validation accuracy us still the same, the predictions speed was much faster and the training time was much faster as well compared to when we had a maximum number of splits to be equal to 100. As you can see when the split criterion is also changed to twoing rule, prediction speed is much faster than when we used Gini's diversity index.

Okay. This is all you evaluated, typical classification model using the classification learner app and you can try other classifiers as well as the case may be. You may also decide to generate a function based on current classifier that you have and play around with the function as the case may be. However, remember that the table must be present in the workspace. This table has a variable for you to use the function. So the training data becomes the table and if I was to save this, if I call this function, command window, this gives me the result that I would have in the classification learner app. That is all for classification. If you have figures such as your ROC curve, this can also be exported. You can save as a jpeg file or a MATLAB figure or even a png file as the case may be to create your reports. It's the same for the confusion matrix. You can always export figures as well. You can also export the model, specific model and you can find this in the workspace, okay. And if you were to use this module, it's always a good practise to save it as a MATLAB file. You can always load it and simply use it in functions such as this and instead of having the partition model here, you can just replace this variable with the name of the variable that you have of your saved model. But take note that you need to have this now as another input into the function if you've decided to go this route. For the demonstration of regression, as you can see, I have my dataset there, which is an excel spreadsheet. I have the live script there.

This is my current path in MATLAB. That's the workspace. I'm just going to open the spreadsheet outside of MATLAB. So for this particular dataset, you can see we have categorical data. We have numeric data as well. So you can also use this dataset for classification. However, in this case, we are going to be using it for regression. And what I'm going to do is to use predictors such as acceleration of the car, the cylinders, displacement, the horsepower and the MPG. So you can see we have some missing data here. So how I will be using MPG as we need to use a dataset that is complete. We'll be using acceleration, cylinders, displacement and horsepower as my predictor variables and I'll try to predict the weight of the car. So, to do this, I'll use a lot live script. So you can see, we are clearing the workspace, clearing the screen, closing all other instances of applications that may be opened. We are assign a variable name to the file containing the dataset of the observations. We have how many observations? 100 observations. Then we create a table using our dataset, so we do not start regression learner yet. And that's our dataset. Unlock doc. Exactly the same as this. So, if we go back to the live script. You can also follow this link to view this example on the MathWorks website as well. So, I'm just going to run this to launch the regression learner app, because we have a table in the workspace already. Remember, the app is only going to recognise the table as the input. We can start a new session and automatically, it's picked the weight as my response variable. However, for my predictors I am not going to use this and I'm going to use, I am just going to use accelerations,

cylinders, displacement, horsepower. I'm also not going to use the model year to predict the weight. So I have 4 predictors. I'll also be using cross-validation five folds, because we only have 100 observations and I'll start the session. I'm also going to be using the regression tree, similar to what I've used for classification. And because the model has not yet been trained, I cannot see the regression line, so I'll just click on train. MATLAB is opening the parallel pool. So you do need the parallel computing tool box. So, it started to train, and it's finished training. As you can see here, we have the And as you can see here, we have the mean square error. We have the R-squared. We have the prediction speed and We have the training time. Here, we have not changed any of the hyperparameters. Because the R-squared value is quite close to 1, this suggests that the regression model explains the variability of the response data around its mean. When we can look at the regression line. However, the mean squared error is quite high. So we can try to optimise the model

by finding all decision splits an increase in the minimum leaf size. So it got worse. So we can try optimisable tree. An in this way, MATLAB is going to optamise the tree for us by tuning the hyperparameters, and you can see the minimum leaf size used is 7. And we have better R-squared value. The root mean squared error is also reduced. The prediction speed is much faster.

But the training time has cost us more because we've optimised this and you can tell by inspection we have more of the data cells around our regression line here. As before, we can export to figure. We can also generate a function as well. You can also export the model. Give it a name and just save it to workspace in MATLAB. It didn't save the function, so we can generate the function and save it. The input is always going to be the table. So if I want to try this, you can call this function window, and I can write this instead of using the classification learner app. In this lecture video, we've looked at classification models, particularly in typical datasets, which can be used to build and train classification models, a demonstration in MATLAB using a live script and the classification learner app. We've also looked at regression models, specifically typical dataset, which can be used to build and train regression models, demonstration in MATLAB using a live script and the regression learner app.