Supervised Learning

So what is supervised learning and how is it different from unsupervised learning? In supervised learning, we work with a set of labeled training data, where we have pairs of inputs and desired outputs. The goal here, is to infer a function from the labeled training data so that we can produce the desired output given an input, even for examples that the supervised learning algorithm has never seen before. This availability of labeled training data is what distinguishes supervised from unsupervised learning. There are two main types of supervised learning, classification and regression. In classification, we predict results in a discrete output. In other words, we use our training data to build a model that assigns unseen observations to one or more classes. Examples, here include deciding whether an email is spam or not, or classifying images as cat, dog, or fish images. In regression, on the other hand, we predict results within a continuous numerical output. Continues means that there aren't any gaps in the value that the output can take. Examples here include predicting the price at which a house will sell, or predicting the age of a person based on their image. Supervised learning has been successfully applied to biomedicine and healthcare, for both classification and regression problems. For instance, classification techniques have been employed to predict the occurrence of an asthma exacerbation based on data from proceeding telemonitoring reports. Another example, involves the employment of regression techniques on brain MRI images of patients with Alzheimer's to predict the clinical stage of the disease as expressed through continuous clinical variables. There is a wide range of supervised learning techniques available, such as decision trees, random forests, neural networks, support vector machines, K-Nearest neighbors, linear regression, logistic regression, and others. Let's look at a few of these in more detail. K-Nearest neighbors is one of the simplest machine learning algorithms used for classification and regression. The main idea here is that the prediction for a new data point, or in your case, is the mode or mean of the values of its K-closest data points. In other words, it's K-Nearest neighbors. Let's demonstrate this visually. Here, we have blue circles and red boxes. Now suppose that a mysterious grace star is introduced, and we want to predict whether it is a blue circle or red box. If we're following the K-Nearest neighbors algorithm, K equals one, then all we need to do is find the closest data point and assign its value to our star. In this case, we predict that our star should be a red box. We could try with different values for K, such as three. In this case, our star is predicted to be a blue circle, and that's all it is. All we do is look at the K-Closest data points and take their mode, in the case of classification or their mean, in the case of regression. And how do we decide what value K should take? We can split our training data set into different segments for which we built K-Nearest neighbors models using different values of K and choose the one that has the best accuracy. You might also be wondering, how do we measure the distance between different data points? A commonly used distance metric for continuous variables is Euclidean distance, while, for discrete

variables, the Hamming distance is used. Note that the classification example that we saw earlier contain two-dimensional training examples, but if we have more complex data with several features then the training examples are vectors in a multidimensional feature space, each with a class label. This could be the case of classification of high-risk and low-risk patients with asthma, where several risk factors such as smoking, air pollution, and allergies are used as features in the model. The K-Nearest neighbors algorithm is easy to understand and often gives reasonable performance without a lot of adjustments. Building the model is fast, but when a large number of features is involved, prediction can be rather slow. So this algorithm is worth trying as a baseline method before considering more advanced options. Decision trees allow us to predict the value of a target variable based on a set of input variables. Here's a fictitious example of a decision tree for predicting whether a flat is in Edinburgh or New York, and hence used for classification. As demonstrated in this example, decision trees have a tree like structure ,consisting of nodes and edges. Each node is associated with one of the input variables, while the edges coming from that node, are the total possible values of that note. Here for instance, the root node is about the flat price and the edges distinguish between two mutually exclusive cases, covering the entire range of possible values greater or lower than £400.000. When we are presented with a previously unknown case of a flat, all we need to do is follow the corresponding path down the tree to identify the predicted value of the leaf node. So, suppose that we want to predict the location of a flat with these characteristics. The corresponding path is the one highlighted here, and the prediction is that it's an Edinburgh flat. But how do we build a decision tree like that? Learning a decision tree is about learning the sequence of questions. In other words, the sequence of variables used in their split of possible values that get us to the correct answer more quickly. Let's explain this with the use of the flat example that we just saw. Suppose that we've been given a dataset about flats in Edinburgh and New York with these attributes. To learn a tree, the algorithm searches over all possible attributes or questions, and pick the one that is most informative about the target variable. In our example, price is the first attribute chosen to split the tree in two subsets as it is a big determinant of flat location. If the price of a flat is greater than 400K, then it's most likely to be based in New York and the other way round. However, there are still some cases of Edinburgh flats that cost more than 400K, as well as cases of New York flats that are cheaper than 400K. We can build a more accurate model by repeating the search for the next attribute for each of the two subsets. The most informative attribute for the subset on the left-hand side of the tree, is the number of bedrooms. while, for the right hand side of the tree, it's the year in which the flat was built. This new splits improve the prediction accuracy of the tree. We could have kept going to introduce further attributes until the tree's accuracy is a 100%. However, this is not recommended, as our extended decision tree would then not perform well on previously unseen test data. This would be due to over-fitting to the training data, a phenomenon that may occur in supervised learning, and which reduces the ability of the model to generalize from the training to

the test data. Decision trees have been used in the medical field to diagnose blood infections or even predict heart attack outcomes in chest pain patients. An extension of decision trees has also been used called random forests, in which several decision trees are built from separate subsets of the training data set and then are combined to improve the prediction accuracy. The main advantage of decision trees is that they can handle large amounts of data while remaining easy to read and communicate. On the other hand, the may lead to overly complex models and overfitting could be an issue. Neural networks or artificial neural networks as they're also called, are another technique that is used for both classification and regression problems. A neural network is an interconnected group of nodes called neurons, which are captured here as circles. The arrows here represent connections between the neurons which can transmit a signal from one neuron to another. The neuron that receives the signal can process it and then signal neurons that are connected to it. Neural networks are typically organized in layers. There is an input and an output layer and the maybe one or more hidden layers. Before looking at how neural networks work, let's explain how a single neuron processing works. A neuron calculates a weighted sum of its inputs and then applies a threshold or activation function to determine whether the signal will be sent or not. In a neural network, this process is repeated several times. First, computing hidden units as an intermediate processing step, which are again combined to yield the final result. So, if we know the inputs, the weights between the different layers in the threshold functions, then we can compute the output even manually. However, all we know in supervised learning is the inputs and outputs. So the purpose of neural networks is to learn the weights and the threshold functions with the use of existing data so as to make predictions about previously unseen data. We should note that an important parameter that needs to be set by the user is the number of nodes in the hidden layer. It is also possible to add more layers as shown here. Having large neural networks that consist of many of these layers is referred to as deep learning. Deep learning has attracted a lot of attention lately, as it has brought significant advances in several challenging domains, such as, image analysis and natural language processing. Neural networks and deep learning are extremely powerful. They can deal with large amounts of data and build incredibly complex models. This however, comes at the cost of interpretability. As the more hidden layers are added, the harder it is to understand what is happening inside the neural network. Training large and powerful neural networks can also be a challenging task.