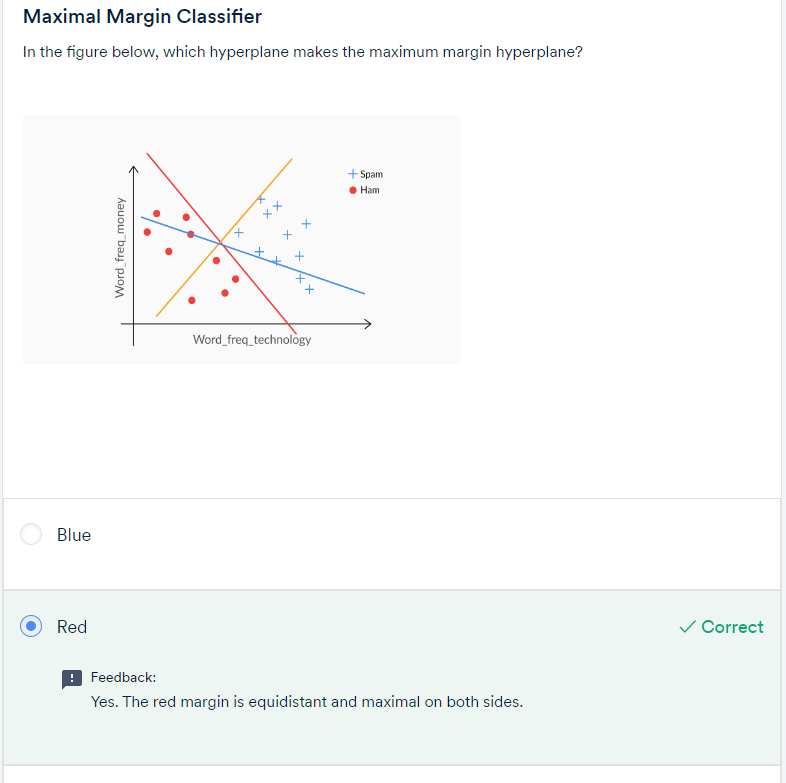
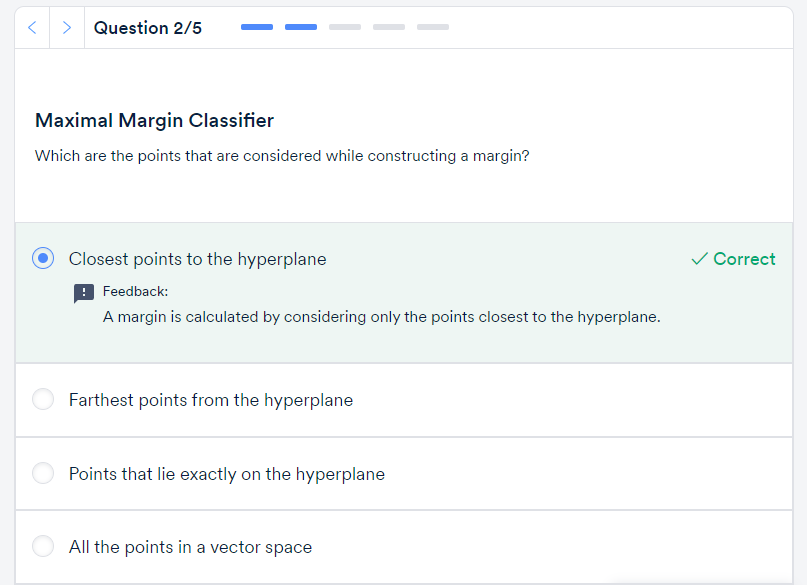


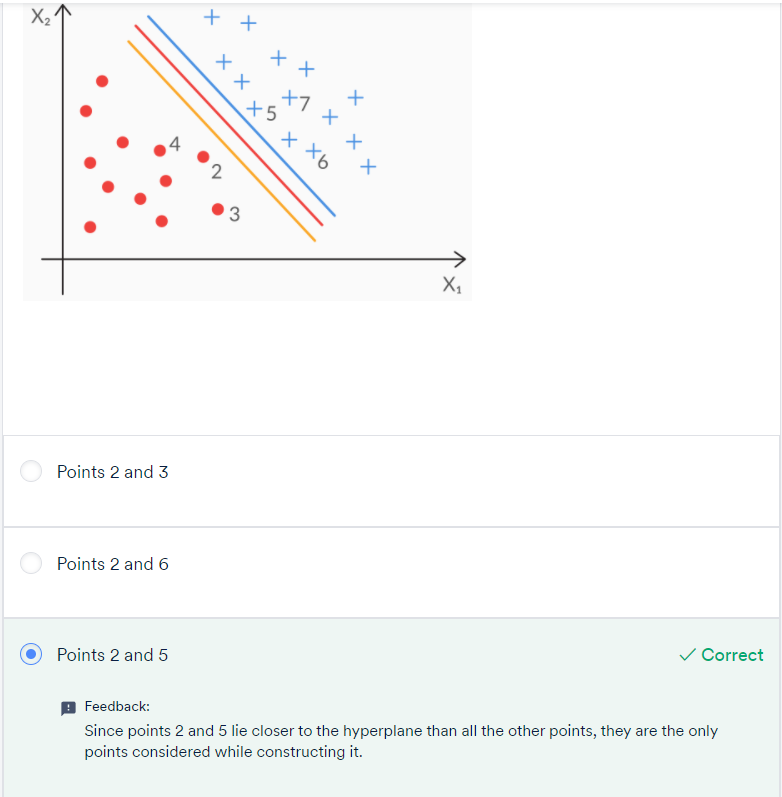


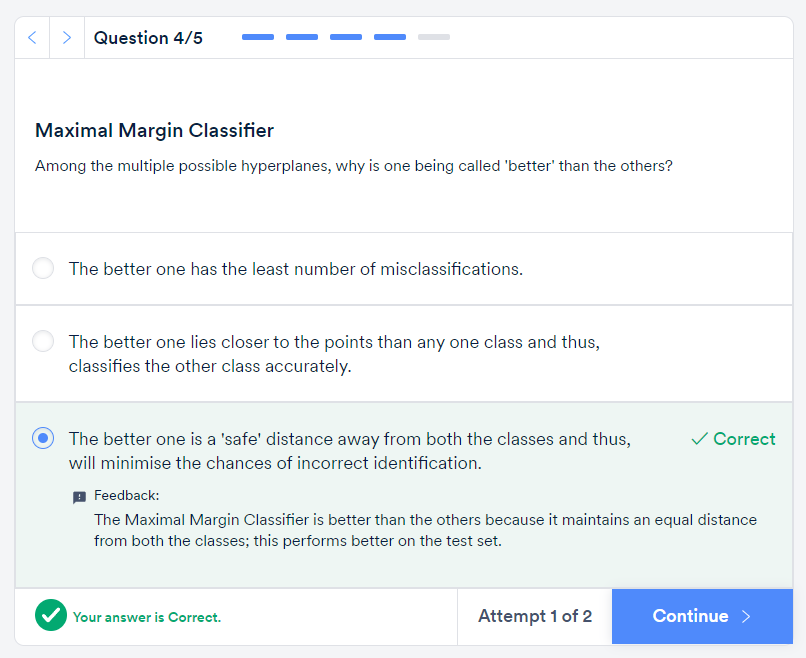


Questions:

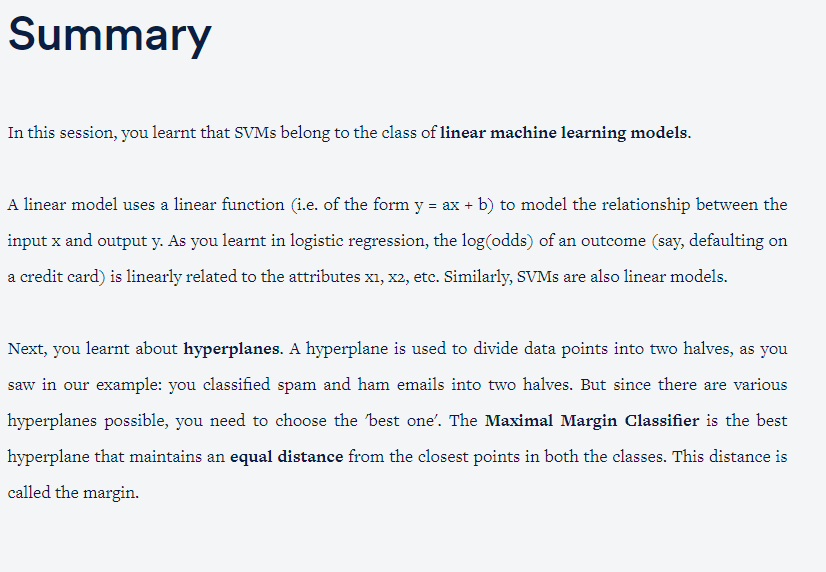




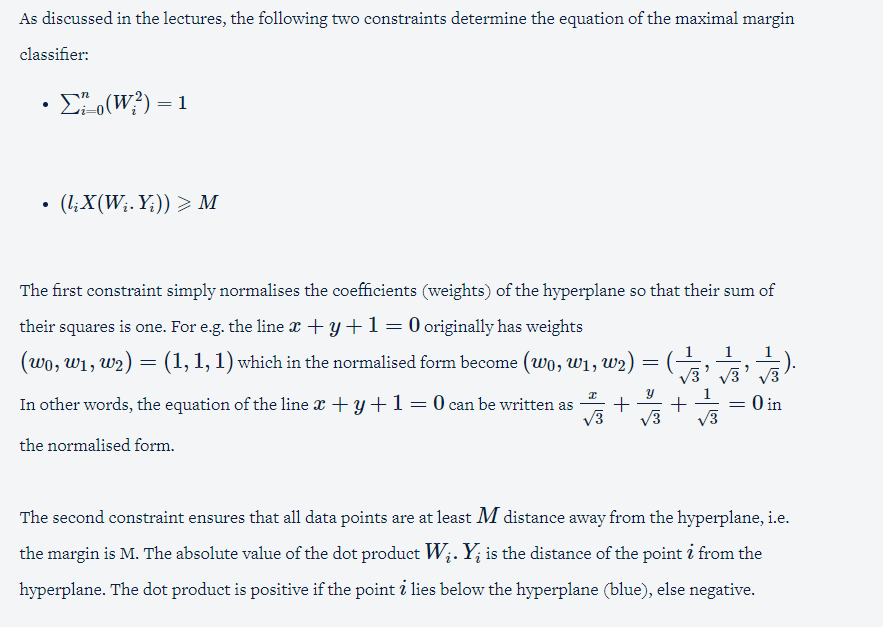


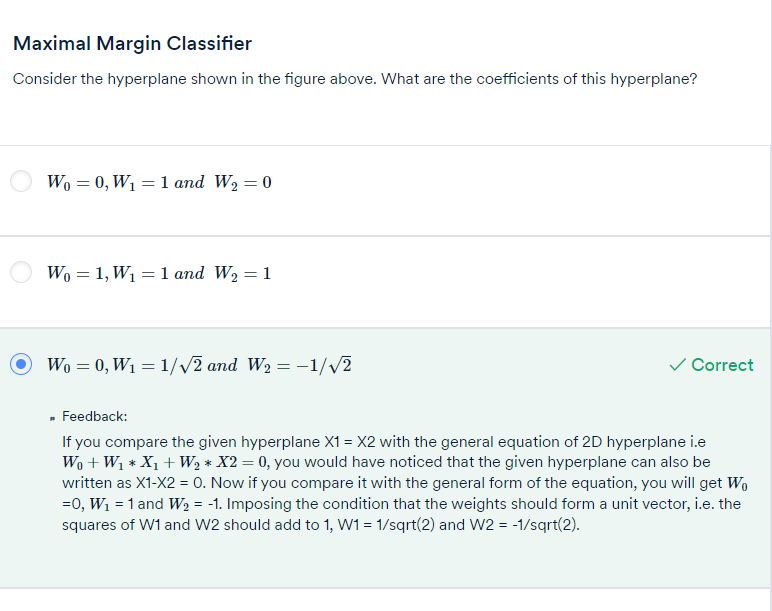


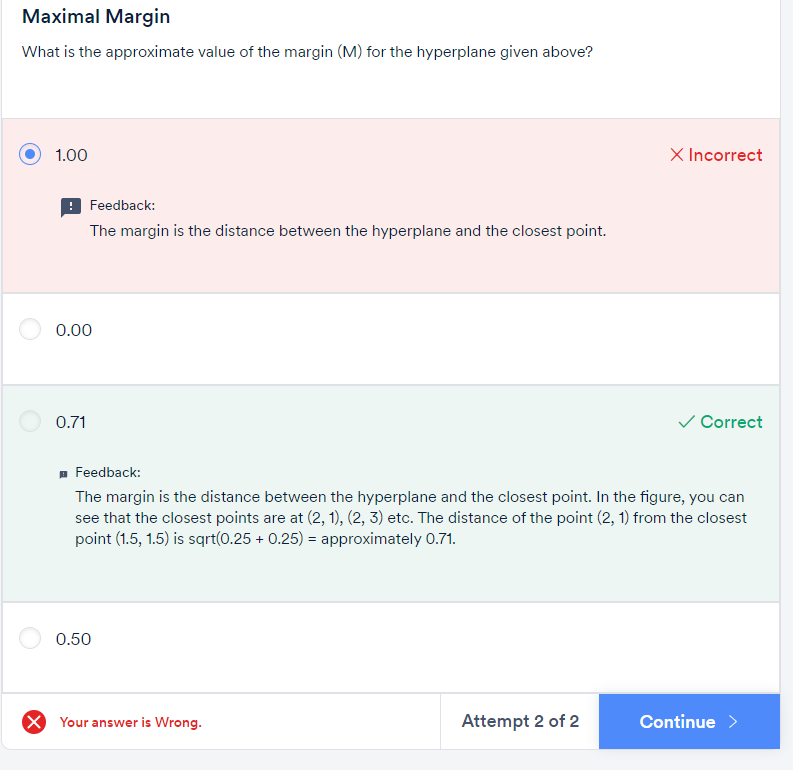




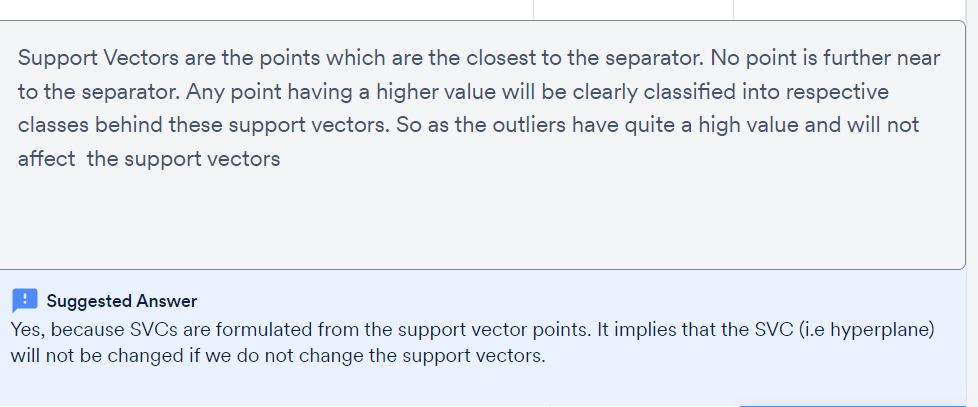
Practice Questions:

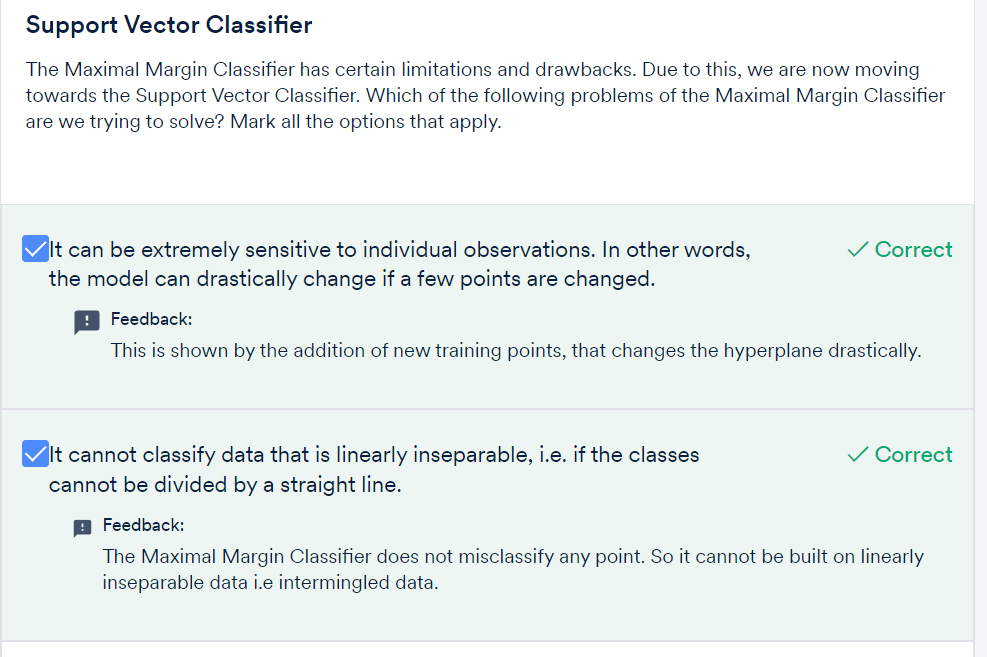




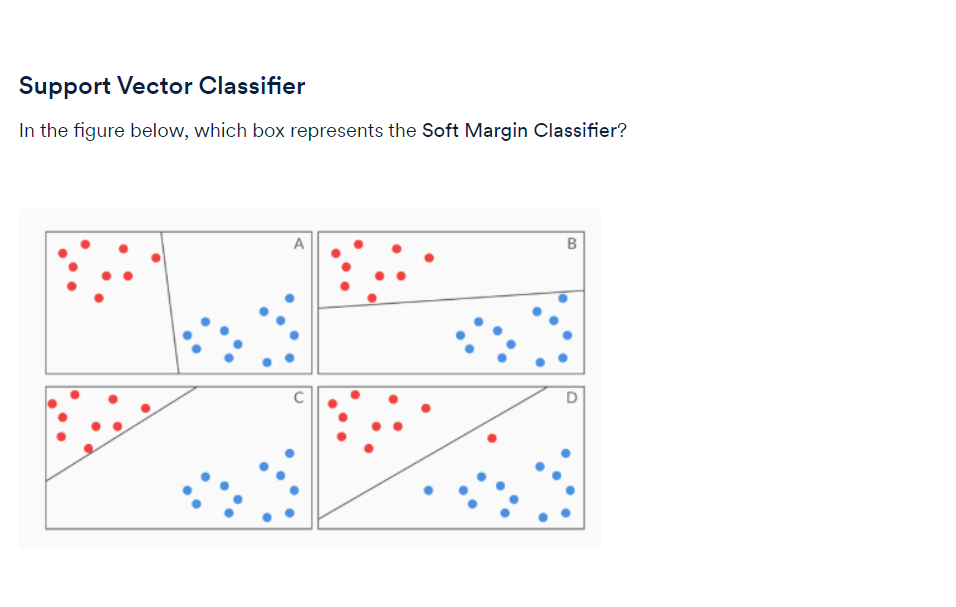


To summarise, **support vectors**are the points that **lie close to the hyperplane**. In fact, they are **the only points that are used in constructing the hyperplane**.

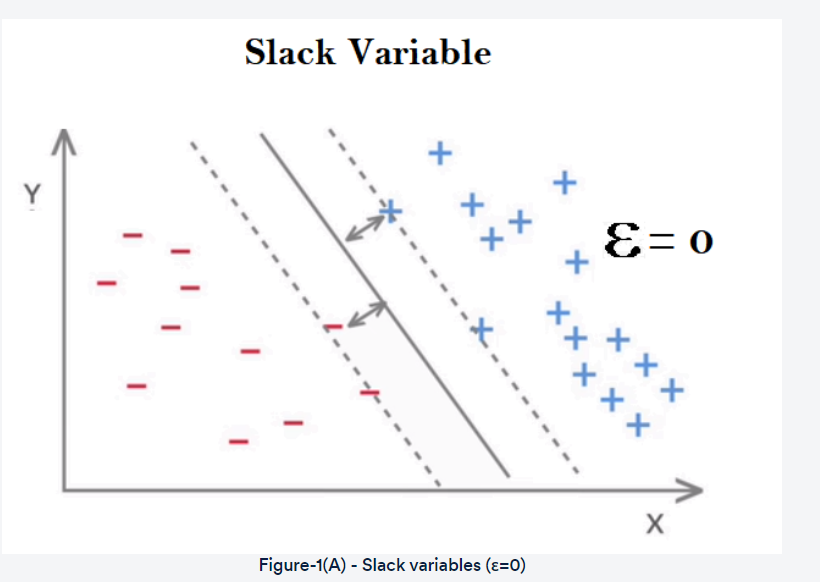


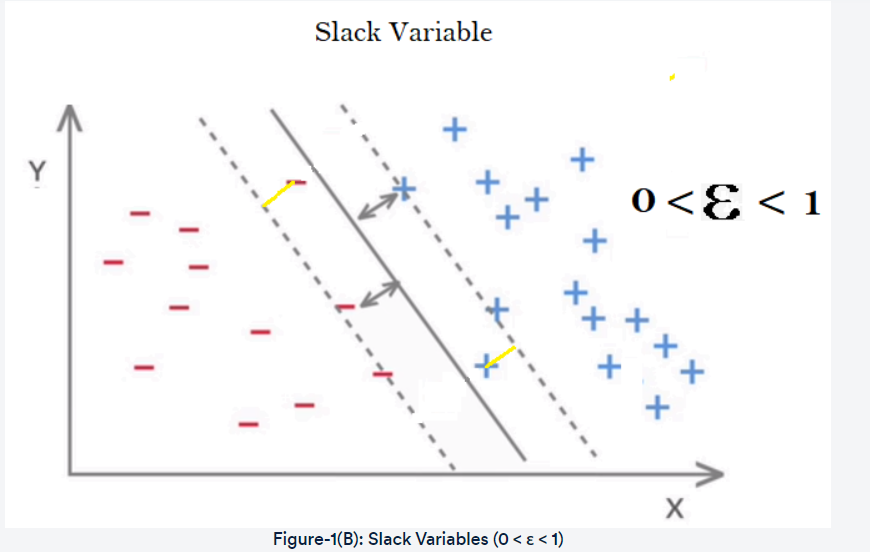


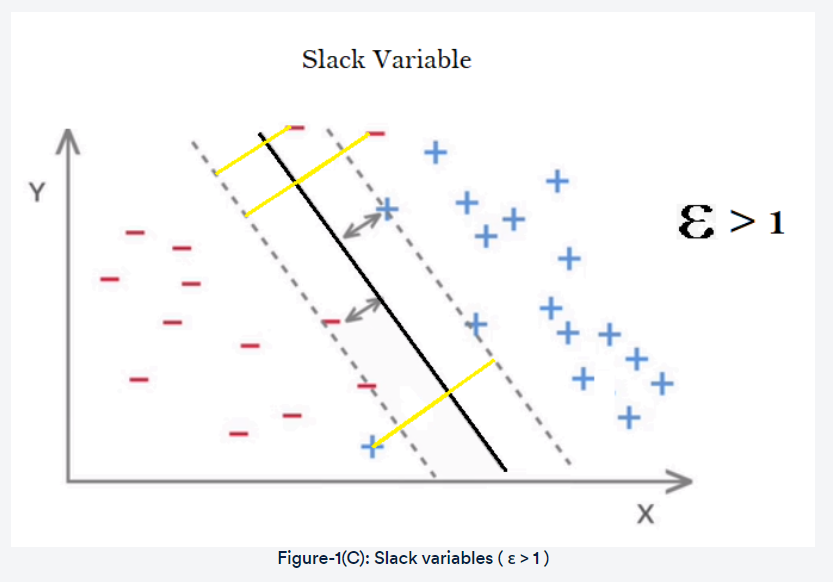




D

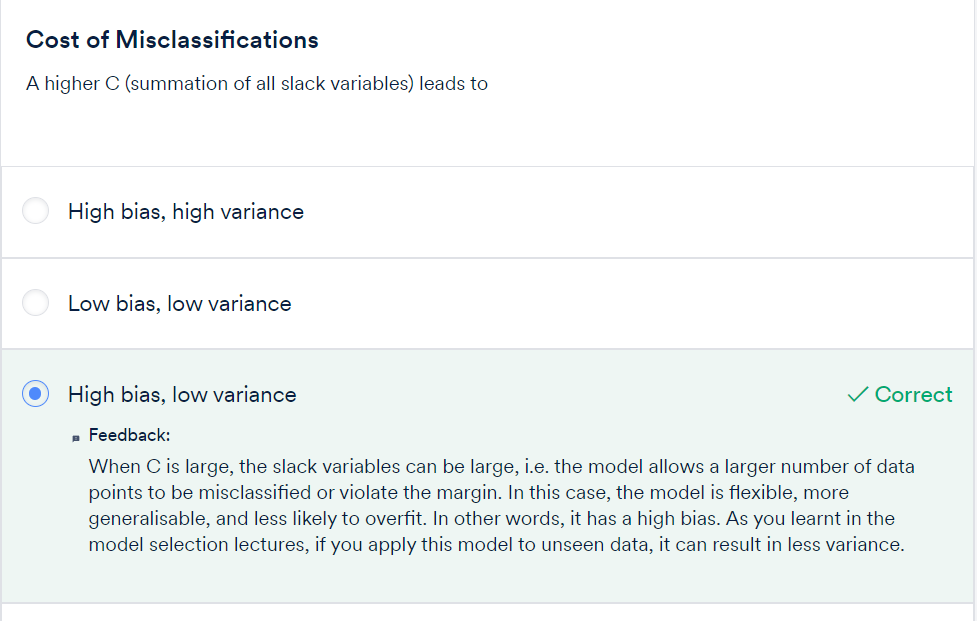


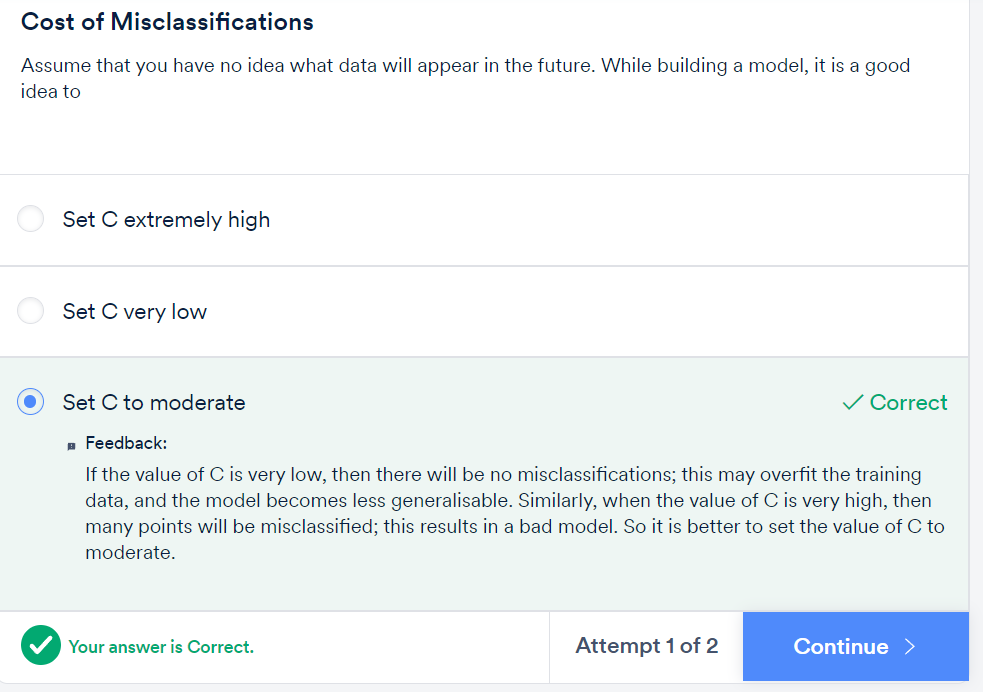


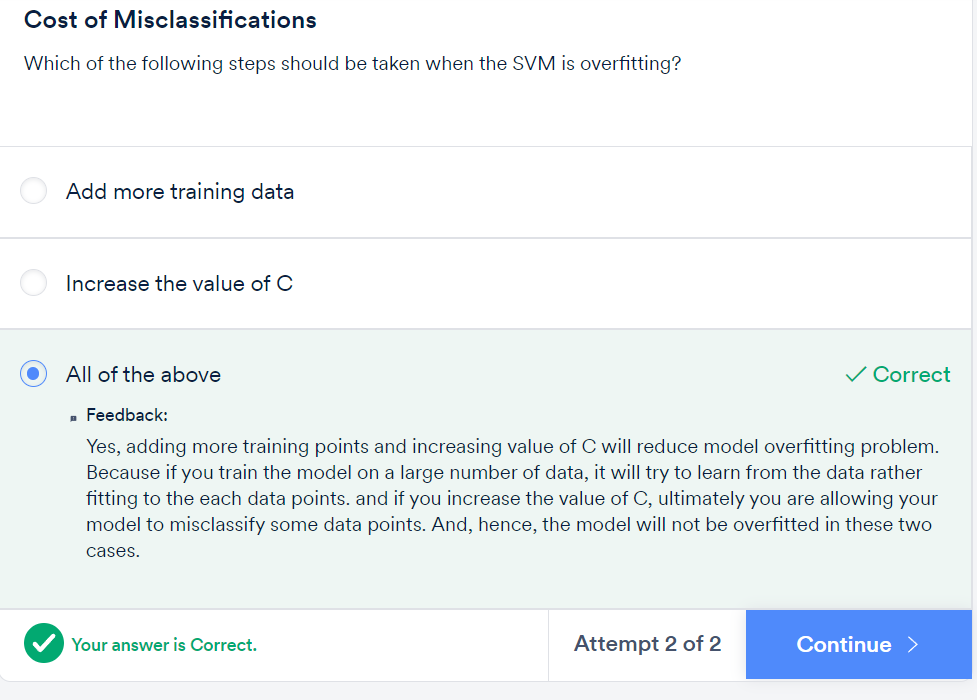


So you can see that

* **Each data point has a slack value** associated to it, according to where the point is located.
* The value of slack lies **between 0 and +infinity.**
* **Lower values of slack are better** than higher values (slack = 0 implies a correct classification, but slack > 1 implies an incorrect classification, whereas slack within 0 and 1 classifies correctly but violates the margin).







As you have already learnt, the mathematical formulation for the Maximal Margin Classifier can be expressed as

liX(W.Yi)>=M

where

* li represents the label of the ith observation (such as spam(+1) and ham(-1));
* **W** represents the vector of the coefficients (or weights) of each attribute (for example, if you have 3 attributes, W = [w0, w1, w2, w3]).
* Yi represents the vector of the attribute values for the ith row, e.g. Y = [1,y1, y2, y3] for 3 attributes.

Thus, the dot product W.Yi is simply the value of the expression obtained by putting the ith data point in the hyperplane equation, i.e. W.Yi=w0+w1y1+w2y2+w3y3.

Thus, W.Yi is lesser than, equal to or greater than 0, depending on the location of the **ith data point** with respect to the hyperplane. Also, note that the value of W.Yi gives you the distance of the**ith data point from the hyperplane**.

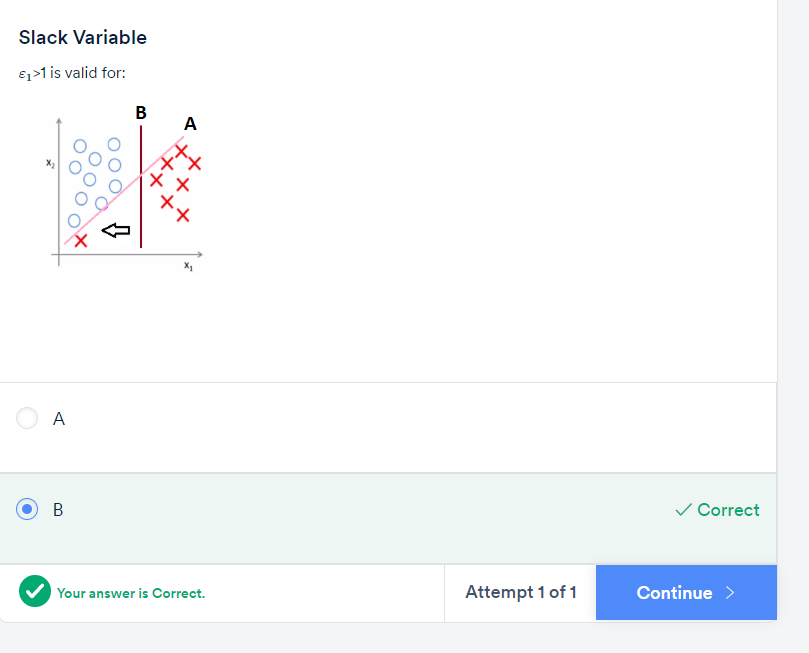
* **M** represents the margin, i.e. the distance of the closest data point from the hyperplane.

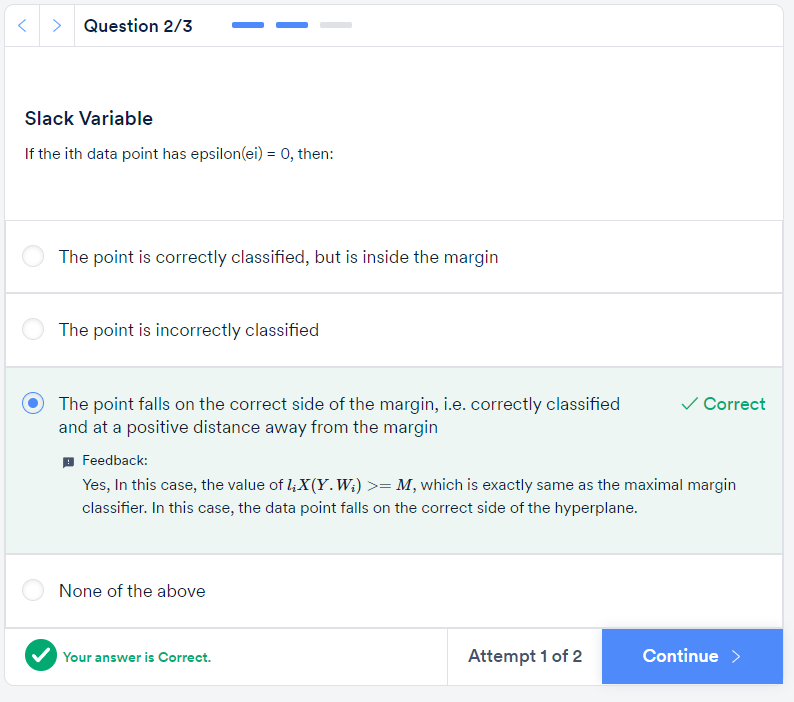
If you impose the condition (liX(W.Yi)>=M) on the model, then you are implying that you want each point to be at least **a distance M away from the hyperplane**. But unfortunately, few real datasets will be so easily, perfectly separable. Thus, to relax the constraint, you include a ‘**slack variable**’ εi for each data point **i**.

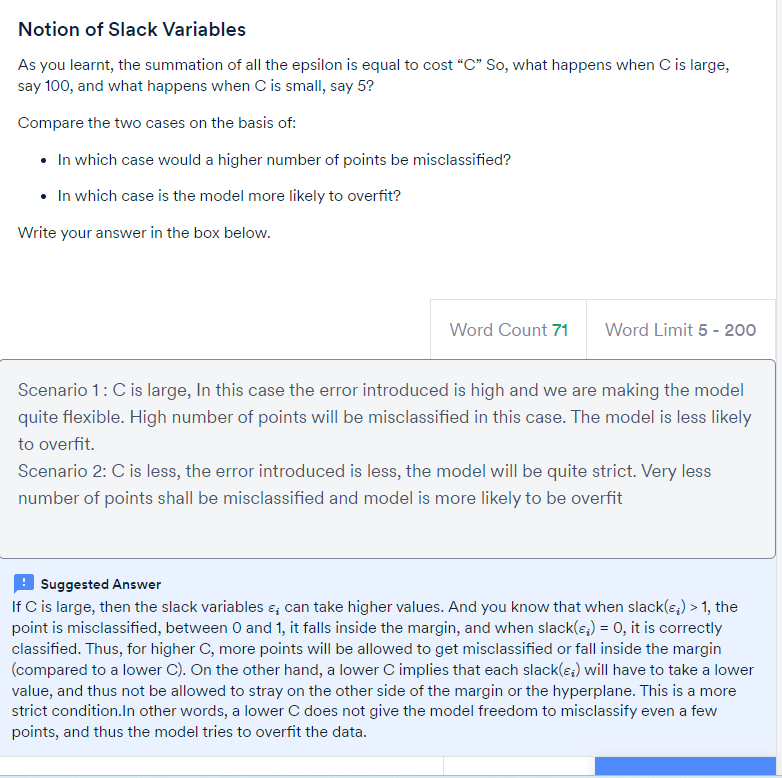
Thus, you modify the formulation to (liX(W.Yi)>=M(1−εi),

Where the **slack variable (**εi**)**takes a value between **0 to infinity**.

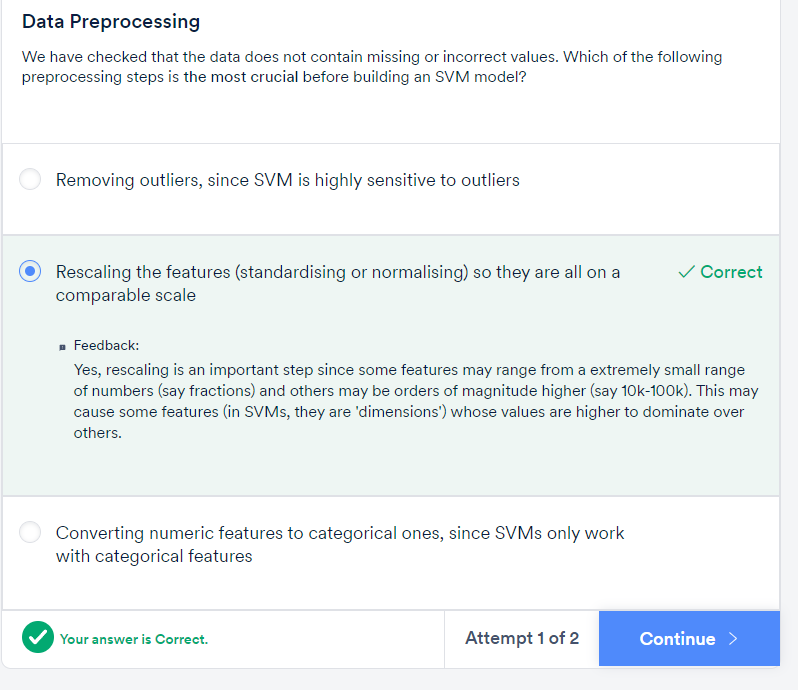
Depending on the value of εi, the ith data point can now take any position - it can fall on the**correct side of the margin** (and a safe distance away), or **inside the margin** (but still correctly classified), or **even stray on the wrong side of the hyperplane** itself.



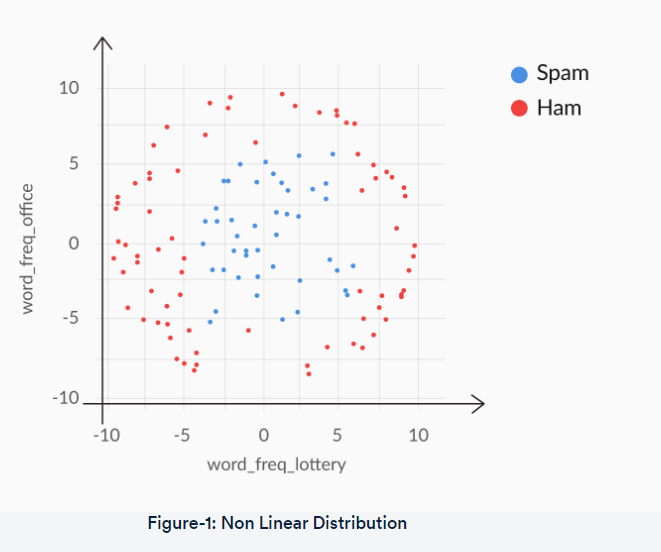




**Cost of Misclassification**

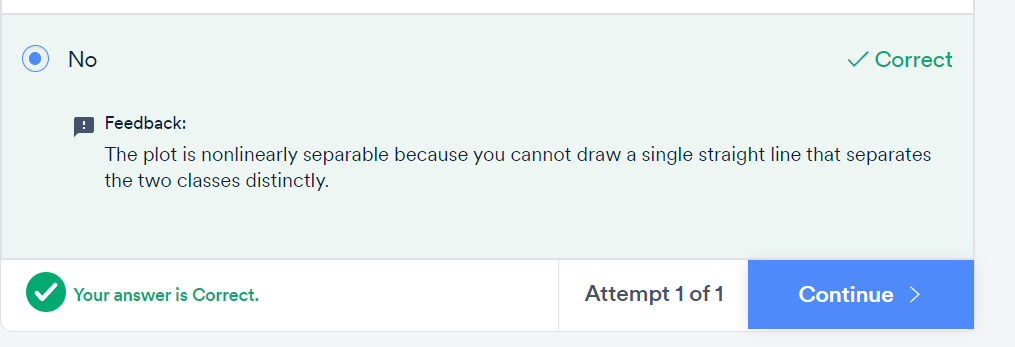


# Kernels :



the Maximal Margin Classifier, and the Support Vector Classifier. All of these are **linear models** (since they use linear hyperplanes to separate the classes).

**Kernels** serve this purpose — they enable the linear SVM model to separate nonlinearly separable data points.

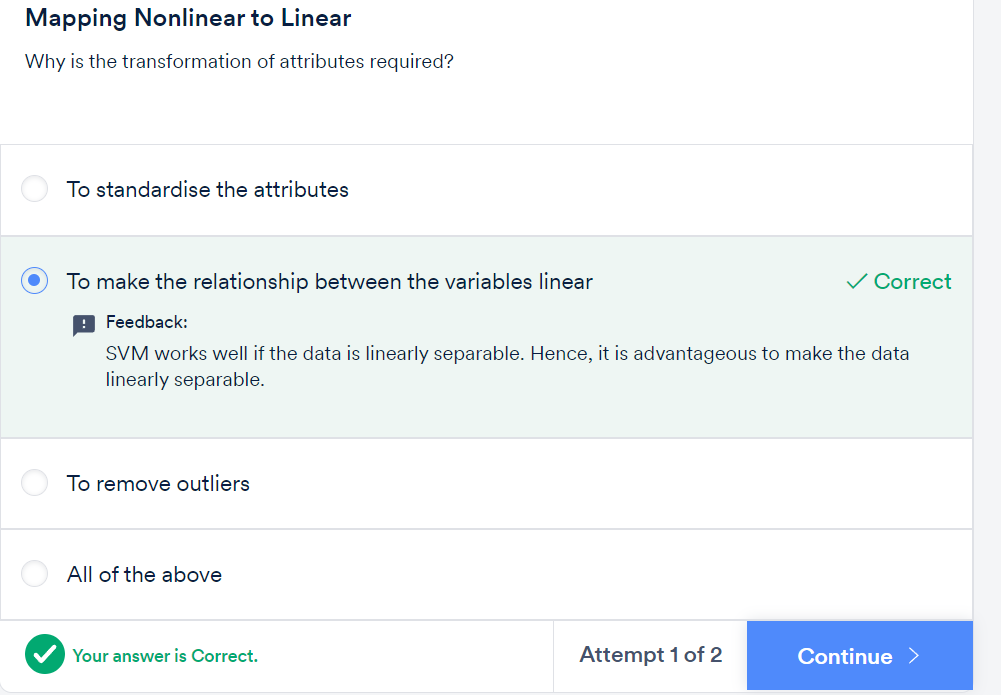






**Mapping Nonlinear Data to Linear Data**

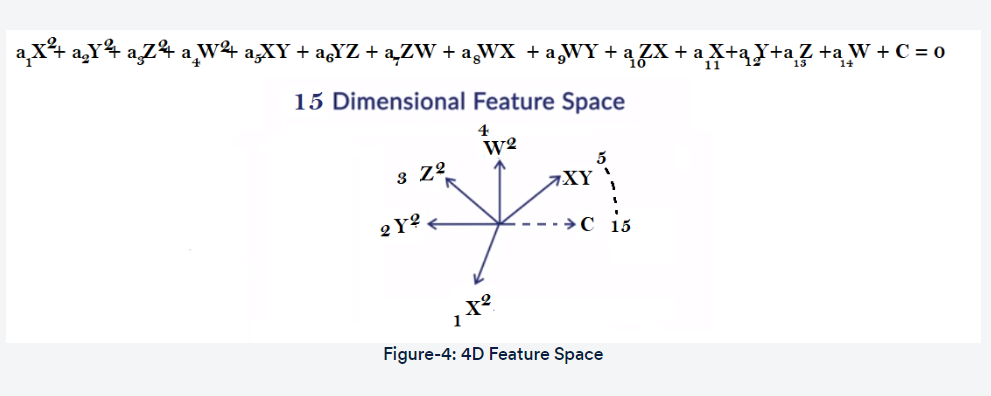
To summarize, you can transform nonlinear boundaries to linear boundaries by applying certain functions to the original attributes. The original space (X, Y) is called the original **attribute space,** and the transformed space (X’, Y’) is called the **feature space.**

****

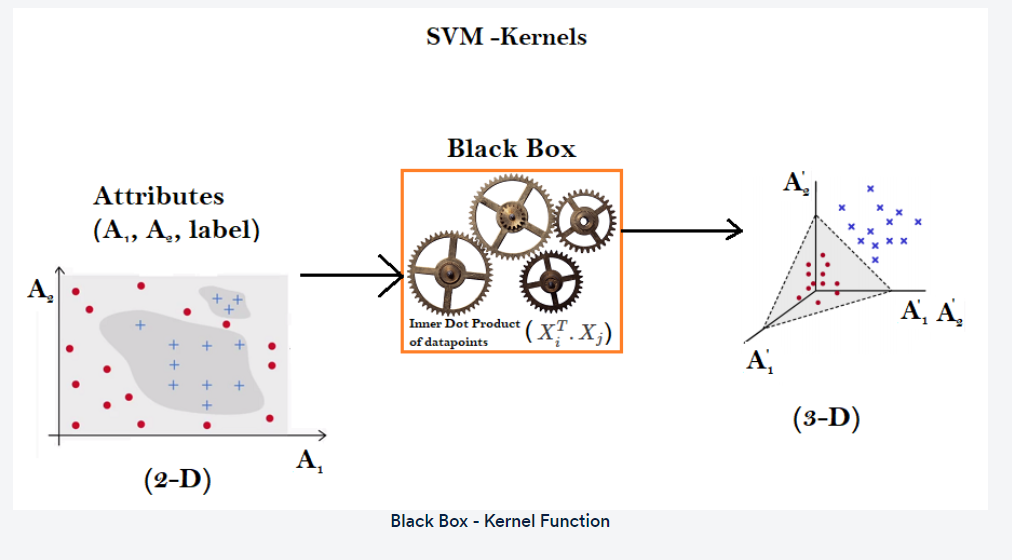
**Feature Transformation:**

You saw that as the number of attributes increases, there is an **exponential increase** in the number of dimensions in the transformed feature space. Suppose you have four variables in your original data set, then considering only a polynomial transformation with **degree 2**, you end up making **15 features**in the new feature space**,** as shown in the figure below.

Note that the terms xyz, xzw etc. do not appear here because it is a transformation of degree 2, and xyz, xzw etc. are third degree terms.

****

**The Kernel Trick:**

****

To summarise, the key fact that makes the kernel trick possible is that **to find a best fit model**, **the learning** **algorithm** **only needs the inner products**of the observations (XTi.Xj). It never uses the individual data points X1, X2 etc. in silo.

Kernel functions use this fact to **bypass the explicit transformation process** from the attribute space to the feature space, and rather **do it implicitly**. The benefit of implicit transformation is that now you do not need to:

* Manually find the mathematical transformation needed to convert a nonlinear to a linear feature space
* Perform computationally heavy transformations

In practice, you only need to know that **kernels are functions**which help you transform non-linear datasets. Given a dataset, you can try various kernels, and choose the one that produces the best model. The three most popular types of kernel functions are:

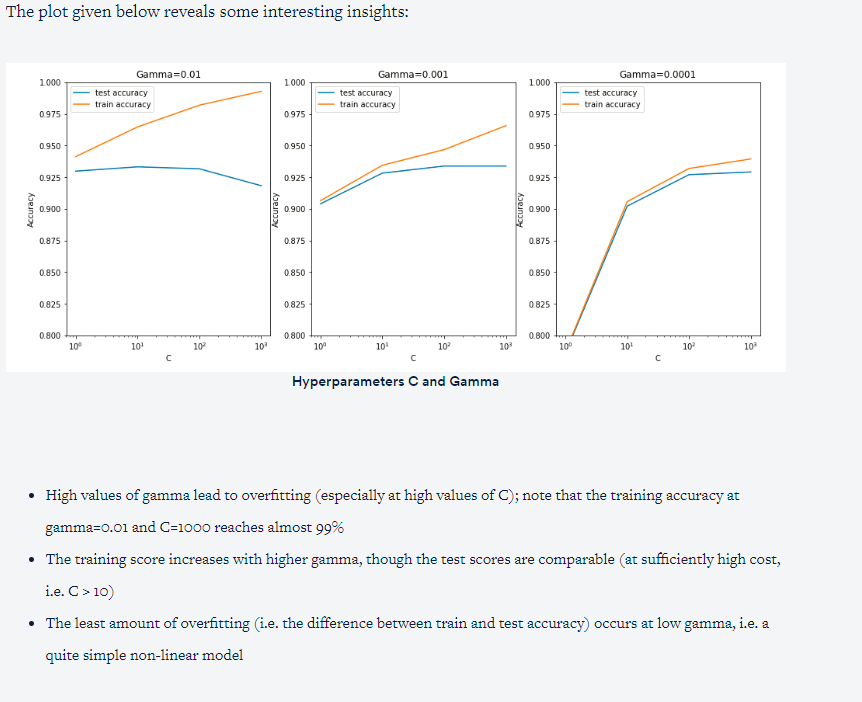
1. **The linear kernel:** This is the same as the support vector classifier, or the hyperplane, without any transformation at all
2. **The polynomial kernel:** It is capable of creating nonlinear, polynomial decision boundaries
3. **The radial basis function (RBF) kernel:**This is the most complex one, which is capable of transforming highly nonlinear feature spaces to linear ones. It is even capable of creating elliptical (i.e. enclosed) decision boundaries

As you have learnt, the **linear kernel,** also known as the hyperplane, requires only one tuning parameter, i.e. 'C' to select the best-fit linear model.

In a non-linear kernel, such as the RBF kernel, you'll need to choose two tuning parameters: **gamma** and '**C**'. The hyperparameter **gamma controls  the amount of non-linearity**in the model - as gamma increases, the model becomes more non-linear, and thus model complexity increases.

Also, now since you have two hyperparameters to optimise (C and gamma), you would need to define a range of values of 'C' and gamma. As you would have guessed, **grid search cross-validation** is the best way to choose the **best combination** of these hyperparameters.

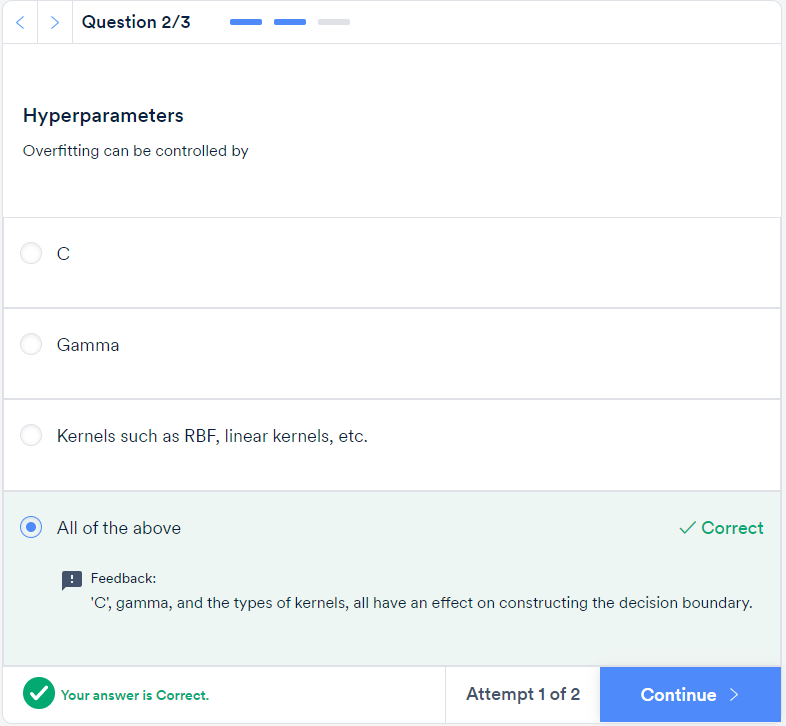


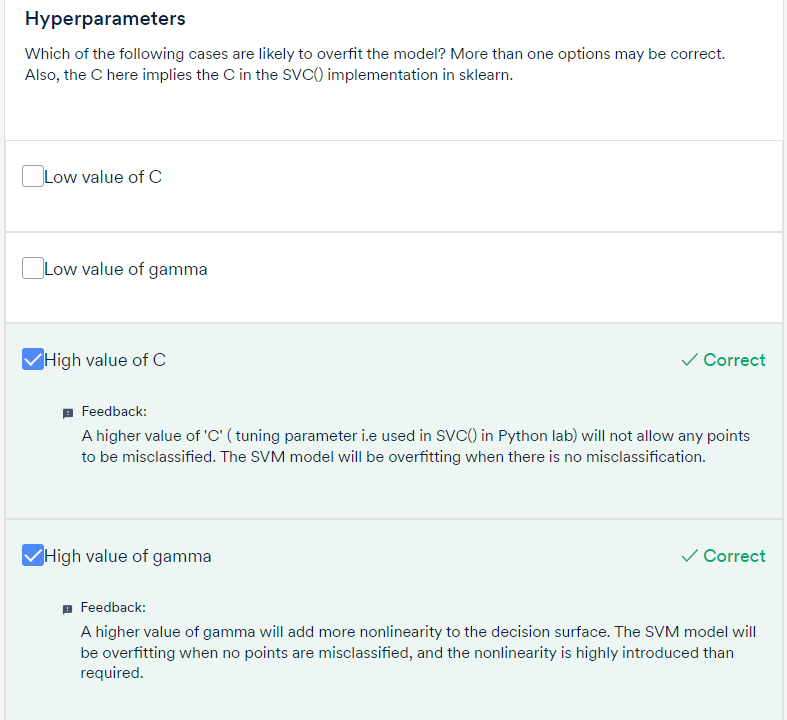


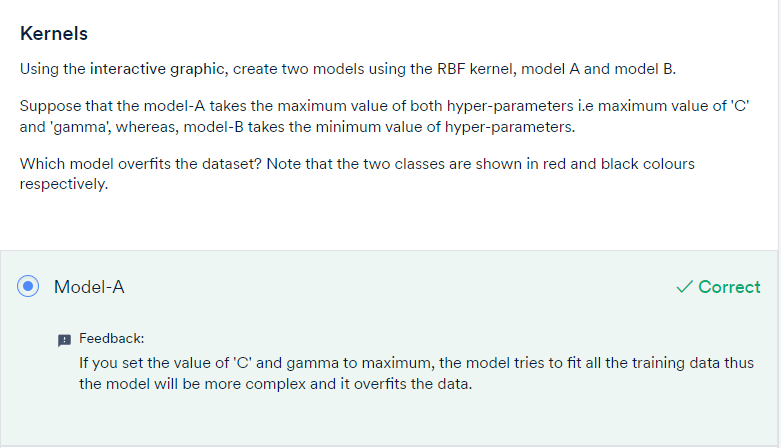
**Comprehension: Kernel Hyperparameters**

In nonlinear kernels such as the **RBF**, you use the parameter gamma to control the amount of nonlinearity in the model. The higher the value of gamma, the more is the nonlinearity introduced; the lower the value of gamma, the lesser is the nonlinearity. It is also denoted as sigma in some texts and packages.

Apart from gamma, you also have the hyperparameter C, or the cost (with all types of kernels).

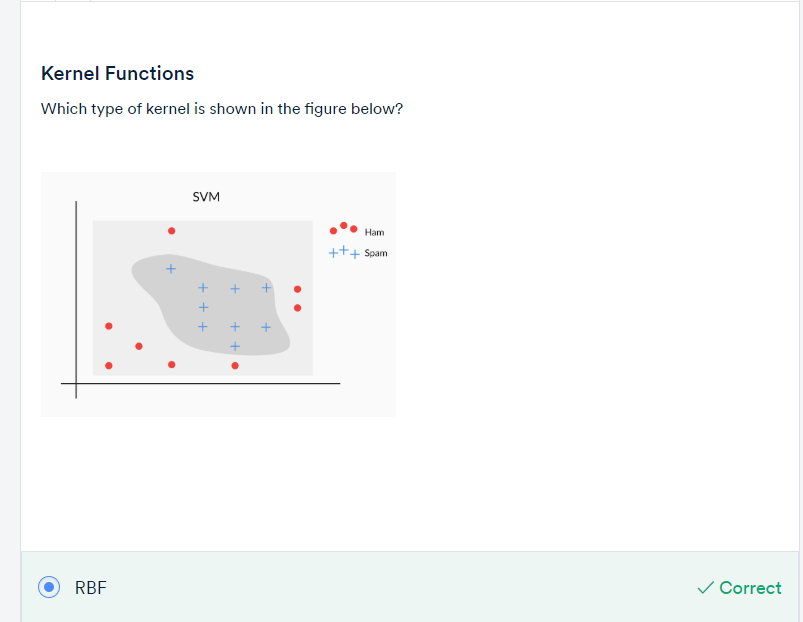






Choosing the appropriate kernel is important for building a model of optimum complexity. If the **kernel** is highly nonlinear, the model is likely to overfit. On the other hand, if the kernel is too simple, then it may not fit the training data well.

Usually, it is difficult to choose the appropriate kernel by visualising the data or using exploratory analysis. Thus, **cross-validation** (or hit-and-trial, if you are only choosing from 2-3 types of kernels) is often a good strategy.



# Summary

So far in this session, you learnt various tricks and tips related to the **transformation** of an attribute space to a feature space.

In 2D and 3D, it is very useful to visualise independent variables graphically; this can help you apply best-fit transformations that classify labels (remember the spam and ham example) correctly.

But the graphical representation of independent variables is not so easy for more than three attributes. Even the manual transformation of an attribute space to a feature space is a cumbersome process. To overcome this challenge, you use the **kernel trick** for transformation, which bypasses the process of converting an attribute space to a feature space explicitly. This means that now, you don't explicitly need to convert each attribute to a feature space. Instead, you need to find a function(∅) that achieves all the manual steps in one go.

Essentially, it helps you identify the best-fit linear separable boundaries by trying out various transformations within the function in the higher dimension.

