1. ***Solution***

* **C- Regularization parameter:**
* It tells the SVM optimisation how much is required to avoid misclassifying each training example.
* C parameter in SVM is considered as the penalty parameter of the error term and it is the degree of correct classification that the algorithm has to meet or the degree of optimization the SVM needs to meet.
* For large values of C, the optimization will choose a smaller-margin hyperplane for getting all the training points classified correctly.
* A very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if the hyperplane misclassifies more points.
* **Kernel**
* SVM algorithms use a set of mathematical functions that are defined as the kernel.
* The function of kernel is to take data as input and transform it into the required form thereby it helps to deal with the non-linear SVM problems.
* Different SVM algorithms use different types of kernel functions such as**linear, nonlinear, polynomial, Radial Basis Function (RBF), and sigmoid.**
* The most used type of kernel function is **RBF b**ecause it has localized and finite response along the entire x-axis and by default RBF is used in ML programming.
* **Degree**
* It is the **degree** of the polynomial kernel function ('poly') and is ignored by all other kernels.
* The default value is 3.
* **Gamma**
* Gamma is the hyperparameter of the Gaussian radial basis function to handle non-linear SVM.
* It is a hyperparameter which we have to set before training model.
* It decides the amount of curvature i.e., how much curvature we want in a decision boundary.
* High Gamma means more curvature and low Gamma means less curvature.
* **coef0**
* coef0 allows to adjust the independent term in the kernel function, but should also leave this alone most likely.
* It is only used in the polynomial and sigmoid kernels.
* **Shrinking**
* Shrinking Boolean in SVM is defaulted to be True.
* This has to do with whether or not we want a shrinking heuristic used in the optimization of the SVM, which is used in Sequential Minimal Optimization ([**SMO**](https://research.microsoft.com/pubs/69644/tr-98-14.pdf)).
* It should set as True, as it should greatly improve the performance, for very little loss in terms of accuracy in most cases.
* **Probability**
* Probability by default is set to be false.
* SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.
* When the constructor option probability is set to True, class membership probability estimates (from the methods predict\_proba and predict\_log\_proba) are enabled.
* **tol**
* The tol parameter is a setting for the SVM's tolerance in optimization.
* The equation Yi (Xi.W+b)-1 >= 0. For an SVM to be valid, all values must be greater than or equal to 0, and at least one value on each side needs to be "equal" to 0, which will be your support vectors.
* It is highly unlikely to actually get the values equal perfectly to 0, we set tolerance to allow a bit of wiggle room.
* The default tol with Scikit-Learn's SVM is 1e-3, which is 0.001.
* **cache\_size**
* By default, cache size=200
* It specifies the size of the kernel cache (in MB).
* A cache smooths the data exchange operations.
* If cache is too small and we have too many data, it becomes risky.
* **Class weight**
* The parameter C of class i is set to class weight[i]\*C for SVC.
* If not given, all classes are supposed to have weight one.
* The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data.
* **Verbose**
* Verbose is a general programming term for produce lots of [logging](http://en.wikipedia.org/wiki/Logfile) output.
* **Verbose by default, is false.**
* **It takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.**
* **max\_iter**
* The next important parameter is max\_iter, which is where we can set a maximum number of iterations for the quadratic programming problem to cycle through to optimize.
* The default is -1, which means there is no limit.
* **Decision\_-function\_-shape**
* The SVM optimization is really tasked to separate one group from another.
* In order to classify a total of 3 or more groups, the method, “One Verse Rest" or (OVR) is used.
* It separates each group from the rest. i.e., to classify three separate groups (1, 2, and 3), we would start by separating 1 from 2 and 3 and then separate 2 from 1 and 3.
* Then finally separate 3 from 1 and 2.
* Another method is One-vs-One (or OVO). In this case, three total groups and the way this works is to have a specific boundary that separates 1 from 3 and 1 from 2.
* This process repeats for the rest of the classes. In this way, the boundaries may be more balanced.
* **Break-\_ties**
* By default, break\_ties=false.
* If true, decision\_function\_shape='ovr', and number of classes > 2, [predict](https://scikit-learn.org/stable/glossary.html#term-predict) will break ties according to the confidence values of [decision function](https://scikit-learn.org/stable/glossary.html#term-decision_function)
* Otherwise, the first class among the tied classes is returned.
* The breaking ties comes at a relatively high computational cost compared to a simple predict.
* **random\_state**
* Regarding the random state, it is used in many randomized algorithms in sklearn to determine the random seed passed to the pseudo-random number generator and it is set as none.
* Ignored when probability is False.
* Pass an int for reproducible output across multiple function calls.

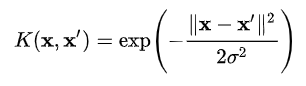
1. ***SOLUTION***

* **Kernels**
* SVM algorithms use a set of mathematical functions that are defined as the kernel.
* The function of kernel is to take data as input and transform it into the required form thereby it helps to deal with the non-linear SVM problems.
* Different SVM algorithms use different types of kernel functions such as**linear, nonlinear, polynomial, Radial Basis Function (RBF), and sigmoid.**
* **Linear kernel:**
* **Linear Kernel** is used when the data is Linearly separable, that is, it can be separated using a single Line.
* It is mostly used when there are a Large number of Features in a particular Data Set.
* One of the examples where there are a lot of features, is **Text Classification**, as each alphabet is a new feature, so we mostly use Linear Kernel in Text Classification.
* **Advantages** of using Linear Kernel:
* 1. Training a SVM with a Linear Kernel is **Faster** than with any other Kernel.
* 2. When training a SVM with a Linear Kernel, only the optimisation of the **C Regularisation** parameter is required.
* On the other hand, when training with other kernels, there is a need to optimise the **γ** parameter which means that performing a grid search will usually take more time.
* **Polynomial kernel**
* For degree-*d* polynomials, the polynomial kernel is defined as

{\displaystyle K(x,y)=(x^{\mathsf {T}}y+c)^{d}} 

where x and y are vectors in the input space.

* In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines and other kernelized models, that represents the similarity of vectors in a feature space over polynomials of the original variables, allowing learning of non-linear models.
* It is used in Image Processing.
* **RBF kernel**
* It is a general-purpose kernel; used when there is no prior knowledge about the data.
* The RBF kernel on two samples **x** and **x'**, represented as feature vectors in some input space, is defined as:

  
1. ‘σ’ is the variance and our hyperparameter  
2. ||x – x’|| is the Euclidean (L₂-norm) Distance between two points x and x’

* **Sigmoid kernel**
* The Sigmoid Kernel comes from the [Neural Networks](http://en.wikipedia.org/wiki/Neural_network) field, where the bipolar sigmoid function is often used as an [activation function](http://en.wikipedia.org/wiki/Activation_function) for artificial neurons.
* It is interesting to note that a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.
* This kernel was quite popular for support vector machines due to its origin from neural network theory.
* Also, despite being only conditionally positive definite, it has been found to [perform well in practice](http://perso.lcpc.fr/tarel.jean-philippe/publis/jpt-icme05.pdf).
* 
* There are two adjustable parameters in the sigmoid kernel, the slope **alpha** and the intercept constant **c**.
* A common value for alpha is 1/N, where N is the data dimension.