Hyperparameters in Support Vector Machine

We take into account some essential hyperparameters for fine-tuning SVMs:

- **C: The regularization parameter that controls the trade-off between the margin and the number of training errors. A larger value of C penalizes training errors more heavily, resulting in a smaller margin but potentially better generalization performance. A smaller value of C allows for more training errors but may lead to overfitting.**
- **Kernel: The kernel function that defines the similarity between data points. Different kernels can capture different relationships between data points, and the choice of kernel can significantly impact the performance of the SVM. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid.**
- **Gamma: The parameter that controls the influence of support vectors on the decision boundary. A larger value of gamma indicates that nearby support vectors have a stronger influence, while a smaller value indicates that distant support vectors have a weaker influence. The choice of gamma is particularly important for RBF kernels.**

Hyperparameters in XGBoost

The following essential XGBoost hyperparameters need to be adjusted:

- **learning_rate: This hyperparameter determines the step size taken by the optimizer during each iteration of training. A larger learning rate can lead to faster convergence, but it may also increase the risk** of overfitting. A smaller learning rate may result in slower convergence but can help prevent overfitting.

- n_estimators: This hyperparameter determines the number of boosting trees to be trained. A larger number of trees can improve the model's accuracy, but it can also increase the risk of overfitting. A smaller number of trees may result in lower accuracy but can help prevent overfitting.
- max_depth: This hyperparameter determines the maximum depth of each tree in the ensemble. A larger max_depth can allow the trees to capture more complex relationships in the data, but it can also increase the risk of overfitting. A smaller max_depth may result in less complex trees but can help prevent overfitting.
- min_child_weight: This hyperparameter determines the minimum sum of instance weight (hessian) needed in a child node. A larger min_child_weight can help prevent overfitting by requiring more data to influence the splitting of trees. A smaller min_child_weight may allow for more aggressive tree splitting but can increase the risk of overfitting.
- subsample: This hyperparameter determines the percentage of rows used for each tree construction. A smaller subsample can improve the efficiency of training but may reduce the model's accuracy. A larger subsample can increase the accuracy but may make training more computationally expensive.

Some other examples of model hyperparameters include:

1. The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
2. Number of Trees and Depth of Trees for Random Forests.
3. The learning rate for training a neural network.
4. Number of Clusters for Clustering Algorithms.
5. The k in k-nearest neighbors.