Hyperparameters are settings or configurations of a machine learning model that are tuned by the practitioner to optimize the model's performance. Here are examples of hyperparameters for various machine-learning models:

**Lasso Regression**:  
**'alpha' :[10,20,30,40,50] # Alpha is the regularization parameter (lambda)**

**Ridge Regression**:  
**'alpha' :[10,20,30,40,50] # Alpha is the regularization parameter (lambda)**

**Logistic Regression**:

**'C' : [0.1, 1.0, 10.0], # Regularization parameter**

**'penalty' : ['l1', 'l2'], # Penalty type**

**'max\_iter' : [100, 200, 300], # Maximum number of iterations**

**Support Vector Machines (SVM)**:

**'C' : [0.1, 1, 10, 100], #Regularization parameter**

**'kernel' : ['linear', 'poly', 'rbf', 'sigmoid'], # Kernel type**

**'gamma' : ['scale', 'auto'], # Kernel coefficient**

**'degree' : [2, 3, 4], # Degree for polynomial kernel**

**'coef0' : [0, 1] # Independent term in sigmoid kernel**

**Decision Trees**:

**'criterion' : ['gini', 'entropy'],**

**'max\_depth' : [None, 5, 10, 15],**

**'min\_samples\_split' : [2, 5, 10],**

**'min\_samples\_leaf' : [1, 2, 4],**

**'max\_features' : ['auto', 'sqrt', 'log2'],**

**'random\_state' : [42]**

**Random Forests**:

**'criterion' : ['gini', 'entropy'],**

**'n\_estimators' : [50, 100, 200],**

**'max\_depth' : [None, 5, 10],**

**'min\_samples\_split' : [2, 5, 10],**

**'min\_samples\_leaf' : [1, 2, 4],**

**'max\_features' : ['auto', 'sqrt', 'log2'],**

**'random\_state' : [42]**

**K-Nearest Neighbors (KNN)**:

**'n\_neighbors' : [3, 5, 7],**

**'weights' : ['uniform', 'distance'],**

**'algorithm' : ['auto', 'ball\_tree', 'kd\_tree', 'brute'],**

**'leaf\_size' : [30, 40, 50],**

**'p' : [1, 2],**

**'metric' : ['euclidean', 'manhattan', 'chebyshev']**

* **n\_neighbors**: Number of neighbors to consider.
* **weights**: Weight function used in prediction (e.g., 'uniform', 'distance').

**Gradient Boosting Machines (GBM)**:

* **n\_estimators**: Number of boosting stages.
* **learning\_rate**: Rate at which boosting adapts.
* **max\_depth**: Maximum depth of the individual trees.

**K-Means Clustering**:

* **n\_clusters**: Number of clusters to form.
* **init**: Method for initialization of centroids (e.g., 'k-means++', 'random').

**Neural Networks**:

* **learning\_rate**: Rate at which the model learns.
* **batch\_size**: Number of samples per gradient update.
* **activation**: Activation function used in hidden layers (e.g., 'relu', 'sigmoid', 'tanh').
* **hidden\_layer\_sizes**: Architecture of the neural network (number of neurons in each hidden layer).

These are just a few examples, and the choice of hyperparameters can greatly influence the performance of the model on a given task. Tuning hyperparameters is often an iterative process involving experimentation and evaluation.

1. **Criterion**: The function to measure the quality of a split. It can be "gini" for the Gini impurity or "entropy" for the information gain.
2. **Max Depth**: The maximum depth of the tree. Increasing this value allows the tree to grow deeper, potentially capturing more complex relationships in the data, but it also increases the risk of overfitting.
3. **Min Samples Split**: The minimum number of samples required to split an internal node. It controls the process of splitting nodes by preventing nodes from splitting if the number of samples is below this threshold.
4. **Min Samples Leaf**: The minimum number of samples required to be at a leaf node. Similar to min\_samples\_split, this parameter prevents the creation of nodes that represent fewer samples.
5. **Max Features**: The number of features to consider when looking for the best split. It can be an integer (representing the exact number of features) or a float (representing a fraction of the total features).
6. **Max Leaf Nodes**: The maximum number of leaf nodes in the tree. Limiting the number of leaf nodes can prevent overfitting by controlling the size of the tree.
7. **Min Impurity Decrease**: A node will be split if this split induces a decrease of the impurity greater than or equal to this value. It can be used to control the growth of the tree.
8. **Class Weight**: Weights associated with classes in the form **{class\_label: weight}**. It is useful for handling class imbalance.
9. **Random State**: The seed of the random number generator used to select features and make splits. Setting this parameter ensures reproducibility.
10. **n\_estimators**: Number of trees in the forest.
11. **max\_depth**: Maximum depth of each tree.
12. **min\_samples\_split**: Minimum number of samples required to split an internal node.
13. **max\_features**: Number of features to consider when looking for the best split in each tree.