Machine Learning Algorithm Parameters Cheat Sheet

Linear Models

1. **Linear Regression**

   - Key Parameters:

     - `fit_intercept`: Whether to calculate the intercept for the model.

       - Example: `LinearRegression(fit_intercept=True)`

     - `normalize`: Normalize input features.

       - Example: `LinearRegression(normalize=True)`

     - `alpha` (Ridge/Lasso): Regularization strength (L2 for Ridge, L1 for Lasso).

       - Example: `Ridge(alpha=1.0)` / `Lasso(alpha=0.1)`

2. **Logistic Regression**

   - Key Parameters:

     - `penalty`: Regularization type (`l1`, `l2`, `elasticnet`, `none`).

       - Example: `LogisticRegression(penalty='l2')`

     - `C`: Inverse of regularization strength.

       - Example: `LogisticRegression(C=1.0)`

     - `solver`: Optimization algorithm (`liblinear`, `saga`, `lbfgs`).

       - Example: `LogisticRegression(solver='liblinear')`

Tree-Based Models

1. **Decision Tree**

   - Key Parameters:

     - `max_depth`: Maximum depth of the tree.

       - Example: `DecisionTreeClassifier(max_depth=10)`

     - `min_samples_split`: Minimum samples required to split a node.

       - Example: `DecisionTreeClassifier(min_samples_split=5)`

     - `min_samples_leaf`: Minimum samples required at a leaf node.

       - Example: `DecisionTreeClassifier(min_samples_leaf=2)`

2. **Random Forest**

   - Key Parameters:

     - `n_estimators`: Number of trees in the forest.

       - Example: `RandomForestClassifier(n_estimators=100)`

     - `max_features`: Number of features to consider for the best split.

       - Example: `RandomForestClassifier(max_features='sqrt')`

     - `bootstrap`: Whether to use bootstrapped samples for training.

       - Example: `RandomForestClassifier(bootstrap=True)`

3. **Gradient Boosting**

   - Key Parameters:

     - `learning_rate`: Shrinks the contribution of each tree.

       - Example: `GradientBoostingClassifier(learning_rate=0.1)`

     - `n_estimators`: Number of boosting stages.

       - Example: `GradientBoostingClassifier(n_estimators=100)`

     - `subsample`: Fraction of samples used for fitting each base learner.

       - Example: `GradientBoostingClassifier(subsample=0.8)`

Support Vector Machines (SVM)

1. **SVM (Classification)**
   - Key Parameters:
     - `C`: Regularization parameter.
       - Example: `SVC(C=1.0)`
     - `kernel`: Type of kernel function.
       - Example: `SVC(kernel='rbf')`
     - `gamma`: Kernel coefficient.
       - Example: `SVC(gamma='scale')`

2. **SVM (Regression)**
   - Key Parameters:
     - `epsilon`: Margin of tolerance.
       - Example: `SVR(epsilon=0.1)`

Clustering

1. **K-Means**
   - Key Parameters:
     - `n_clusters`: Number of clusters to form.
       - Example: `KMeans(n_clusters=3)`
     - `init`: Initialization method.

    - Example: `KMeans(init='k-means++')`

  - `max_iter`: Maximum iterations for a single run.

    - Example: `KMeans(max_iter=300)`

2. **DBSCAN**

   - Key Parameters:

     - `eps`: Radius of neighborhood for clustering.

       - Example: `DBSCAN(eps=0.5)`

     - `min_samples`: Minimum number of points to form a dense region.

       - Example: `DBSCAN(min_samples=5)`

Neural Networks

1. **MLP (Multilayer Perceptron)**

   - Key Parameters:

     - `hidden_layer_sizes`: Number and size of hidden layers.

       - Example: `MLPClassifier(hidden_layer_sizes=(100, 50))`

     - `activation`: Activation function.

       - Example: `MLPClassifier(activation='relu')`

     - `solver`: Optimization algorithm.

       - Example: `MLPClassifier(solver='adam')`

     - `learning_rate`: Learning rate schedule.

       - Example: `MLPClassifier(learning_rate='constant')`

Regularization and Optimization

1. **Dropout (NN)**

   - Key Parameters:

     - `rate`: Fraction of neurons to drop.

       - Example: Use in frameworks like PyTorch: `torch.nn.Dropout(rate)`

2. **L1/L2 Regularization**

   - Key Parameters:

     - `alpha`: Strength of regularization.

       - Example: `Ridge(alpha=1.0)` / `Lasso(alpha=0.1)`

3. **Adam Optimizer**

   - Key Parameters:

     - `learning_rate`: Step size for updating weights.

       - Example: Use in PyTorch or TensorFlow: `torch.optim.Adam(lr=0.001)`