HyperParameters

In Machine Learning

What is a Hyperparameter?

- A **hyperparameter** is a parameter whose value is set before the learning process begins.
- Unlike model parameters, which are learned from the training data during the model fitting process (such as weights in a linear regression or coefficients in a decision tree), hyperparameters are specified by the practitioner.
- These hyperparameters govern the training process and the structure of the machine learning model.

Why Use Hyperparameters?

- **Model Performance Optimization**
- **Control Overfitting and Underfitting**
- **Training Efficiency**
- **Model Interpretability:**
  - Certain hyperparameters can make models more interpretable. For instance, limiting the depth of a decision tree can produce a simpler, more understandable model at the cost of potentially lower performance.

Tuning Hyperparameters

- Finding the best hyperparameters is often done through a process called **hyperparameter tuning**.
- Common techniques include:
  - **Grid Search**: Systematically tries every combination of hyperparameters from a predefined set of values.
  - **Random Search**: Samples hyperparameters randomly from a specified range.
  - **Bayesian Optimization**: Uses probabilistic models to find the optimal hyperparameters by learning from previous evaluations.
  - **Cross-Validation**: Often used in conjunction with the above methods to evaluate the performance of different hyperparameter settings on different subsets of the training data.

Linear Regression

- **fit_intercept**: Boolean; whether to calculate the intercept for the model. If False, no intercept will be used.
  - Controls whether the intercept term should be included in the model. Use True for most cases unless you have reasons to exclude the intercept.
- **normalize**: Boolean; if True, the regressors X will be normalized before regression.
  - Controls whether the features should be normalized before fitting. Use True if your features have different scales and you haven't already normalized them, to ensure better model performance and numerical stability.

Logistic Regression

- **penalty**: Type of regularization to be used (e.g., 'l1', 'l2', 'elasticnet', 'none').
- **C**: Inverse of regularization strength; smaller values specify stronger regularization.
- **solver**: Algorithm to use in the optimization problem ('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga').
- **max_iter**: Maximum number of iterations to converge

Decision Tree

- **criterion**: Function to measure the quality of a split (e.g., 'gini', 'entropy').
- **max_depth**: Maximum depth of the tree.
- **min_samples_split**: Minimum number of samples required to split an internal node.
- **min_samples_leaf**: Minimum number of samples required to be at a leaf node.
- **max_features**: Number of features to consider when looking for the best split.

Random Forest

- **n_estimators**: Number of trees in the forest.
- **criterion**: Function to measure the quality of a split (e.g., 'gini', 'entropy').
- **max_depth**: Maximum depth of the tree.
- **min_samples_split**: Minimum number of samples required to split an internal node.
- **min_samples_leaf**: Minimum number of samples required to be at a leaf node.
- **max_features**: Number of features to consider when looking for the best split.
- **bootstrap**: Whether bootstrap samples are used when building trees.

Support Vector Machine (SVM)

- **C**: Regularization parameter; trade-off between achieving a low error on the training data and minimizing the norm of the weights.
- **kernel**: Specifies the kernel type to be used (e.g., 'linear', 'poly', 'rbf', 'sigmoid').
- **degree**: Degree of the polynomial kernel function (if 'poly' is chosen as kernel).
- **gamma**: Kernel coefficient for 'rbf', 'poly', and 'sigmoid'.

k-Nearest Neighbors (k-NN)

- **n_neighbors**: Number of neighbors to use.
- **weights**: Weight function used in prediction (e.g., 'uniform', 'distance').
- **algorithm**: Algorithm used to compute the nearest neighbors (e.g., 'auto', 'ball_tree', 'kd_tree', 'brute').
- **leaf_size**: Leaf size passed to BallTree or KDTree.
- **p**: Power parameter for the Minkowski metric (1 for Manhattan, 2 for Euclidean).

Gradient Boosting

- **n_estimators**: Number of boosting stages to be run.
- **learning_rate**: Shrinks the contribution of each tree by learning_rate.
- **max_depth**: Maximum depth of the individual regression estimators.
- **min_samples_split**: Minimum number of samples required to split an internal node.
- **min_samples_leaf**: Minimum number of samples required to be at a leaf node.
- **subsample**: Fraction of samples to be used for fitting the individual base learners.

XGBoost

- **n_estimators**: Number of boosting rounds.
- **learning_rate**: Step size shrinkage used in update to prevents overfitting.
- **max_depth**: Maximum depth of a tree.
- **subsample**: Subsample ratio of the training instances.
- **colsample_bytree**: Subsample ratio of columns when constructing each tree.
- **gamma**: Minimum loss reduction required to make a further partition on a leaf node

Neural Networks (e.g., using Keras/ Tensor Flow)

- **hidden_layer_sizes**: Number of neurons in the hidden layers.
- **activation**: Activation function for the hidden layer (e.g., 'relu', 'tanh', 'logistic').
- **solver**: The solver for weight optimization (e.g., 'adam', 'sgd', 'lbfgs').
- **alpha**: L2 penalty (regularization term) parameter.
- **learning_rate**: Learning rate schedule for weight updates (e.g., 'constant', 'adaptive').
- **max_iter**: Maximum number of iterations.

Gaussian Naive Bayes (GNB)

- This is used for continuous data, assuming that the features follow a normal distribution.
- **var_smoothing**: This is a small positive value added to the variances to prevent division by zero and handle numerical stability issues. It can be thought of as a form of regularization.

Multinomial Naive Bayes (MNB)

- This is used for discrete data, such as word counts in text classification.
- **alpha**: Smoothing parameter (Laplace smoothing). This parameter helps in handling zero probabilities by adding a small positive number to each count.

Bernoulli Naive Bayes (BNB)

- This is used for binary/boolean features.
- **alpha**: Smoothing parameter (Laplace smoothing), similar to Multinomial Naive Bayes.
- **binarize**: Threshold for binarizing (mapping to 0 or 1) the features. If None, the input is assumed to be binary.

k-Means

- **n_clusters**: Number of clusters to form.
- **init**: Method for initialization (e.g., 'k-means++', 'random').
- **n_init**: Number of time the k-means algorithm will be run with different centroid seeds.
- **max_iter**: Maximum number of iterations of the k-means algorithm for a single run.
- **tol**: Relative tolerance with regards to inertia to declare convergence.

DBSCAN

- **eps**: The maximum distance between two samples for one to be considered as in the neighborhood of the other.
- **min_samples**: The number of samples in a neighborhood for a point to be considered as a core point.

Dimensionality Reduction

- **Principal Component Analysis (PCA)**
- **n_components**: Number of components to keep.
- **svd_solver**: The algorithm to use in the decomposition ('auto', 'full', 'arpack', 'randomized').