**Hyperparameter tuning with GridSearchCV**

Hugo demonstrated how to use to tune the n\_neighbors parameter of the KNeighborsClassifier() using GridSearchCV on the voting dataset. You will now practice this yourself, but by using logistic regression on the diabetes dataset instead!

Like the alpha parameter of lasso and ridge regularization that you saw earlier, logistic regression also has a regularization parameter: CC. CC controls the *inverse* of the regularization strength, and this is what you will tune in this exercise. A large CC can lead to an *overfit* model, while a small CC can lead to an *underfit* model.

The hyperparameter space for CC has been setup for you. Your job is to use GridSearchCV and logistic regression to find the optimal CC in this hyperparameter space. The feature array is available as X and target variable array is available as y

**Hyperparameter tuning with RandomizedSearchCV**

GridSearchCV can be computationally expensive, especially if you are searching over a large hyperparameter space and dealing with multiple hyperparameters. A solution to this is to use RandomizedSearchCV, in which not all hyperparameter values are tried out. Instead, a fixed number of hyperparameter settings is sampled from specified probability distributions. You'll practice using RandomizedSearchCV in this exercise and see how this works.

Here, you'll also be introduced to a new model: the Decision Tree. Don't worry about the specifics of how this model works. Just like k-NN, linear regression, and logistic regression, decision trees in scikit-learn have .fit() and .predict() methods that you can use in exactly the same way as before. Decision trees have many parameters that can be tuned, such as max\_features, max\_depth, and min\_samples\_leaf: This makes it an ideal use case for RandomizedSearchCV.

As before, the feature array X and target variable array y of the diabetes dataset have been pre-loaded. The hyperparameter settings have been specified for you. Your goal is to use RandomizedSearchCV to find the optimal hyperparameters. Go for it!