Part 1

Cross-Validation with GridSearchCV Explain in your report what happens when we run clf.fit(X_train, Y_train)

The line —clf.fit(X_train, Y_train)— here uses the fit method on the object clf and takes as parameters the labels and features of the train sample. The fit method is training the model defined in the clf object. The object clf is from the class GridSearchCV which allows us to find the best hyperparameters among a fixed list we chose and perform a cross-validation. It is taking as parameters an object we named knn of the class KNeighborsClassifier(), a dictionary named parameters containing the number of neighbors to be tested in the knn algorithm (1 to 5 here) and the cv parameter referring to the number of folds to be used in the cross-validation. Basically it will perform a 3-folds cross-validation on a kNN model with 1 to 5 neighbors on the train sample and it will allow us to keep the best model. The kNN algorithm is parametered with the default metric which is the Euclidean distance : $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$. The functions are all part of the sklearn package.

What is the complexity for each of the three following cases?

Complexity can be divided into two kinds of complexity i.e: 1) time complexity, deal with how long the algorithm is executed, and 2) space complexity, deal with how much memory is used by this algorithm. table[ht] Complexity 0.5cm tabular—c—c—c—c—c—c—c—c— kNN Linear SVC Log Reg

With n : size of the training sample, d : dimension of the data, k : number of neighbors, m : number of features, l : support vectors.

What is the test accuracy? What would be the accuracy of random guess?

The test accuracy is the measure of how often the points are correctly classified in the test sample. In our case the accuracy is 0.875. It means that 87.5% of the time, the points are correctly classified on the test sample. It is computed as the number of well classified individuals over the sample size. If we did a random guess we would randomly choose an output in the range 0 to 9 so the accuracy would converge towards 110 according to the LLN.

What is LinearSVC() classifier? Which kernel are we using? What is C? (this is a tricky question, try to find the answer online)

LinearSVC means Linear Support Vector Classification, which is supervised learning methods used for classification. LinearSVC are classes capable of performing binary and multi-class classification on a dataset. This classifier is trained to separate the True labels from the False labels with a boundary compute with a kernel function. In our multi-class problem, the algorithm creates one classifier for each class and performs a one-vs-rest training by training the data of one class versus all the the other data as one other class.

What is the outcome of np.logspace(-8, 8, 17, base=2)? More generally, what is the outcome of np.logspace(-a, b, k, base=m)?

The outcome of np.logspace(-8, 8, 17, base=2) is a logarithmic space going from 2^{-8} to 2^8 with 17 numbers equally spaced on log scale. The logspace function from the numpy package will return k numbers going from m^{-a} to m^b spaced on a log scale with a log base m.

What is the meaning of the warnings? What is the parameter responsible for its appearance?

The warning tells us that the algorithm did not converge, it did not reach the stop criterionThe stop criterion here is the tol parameterefore the number of maximum iterations. The parameter responsible for its appearance is the max_iter parameter. Its value is not large enough for the algorithm to converge. The data variance is maybe too large for the algorithm to efficiently perform the SVM.

What did we change with respect to the previous run of LinearSVC()?

We added a pipeline which is a method of the ScikitLearn package that allows to streamline the data pre-processing. In the pipeline we added a —MaxAbsScaler()— method to scale the absolute data between 0 and 1 and thus reduce the variance of the data. We notice that the algorithm is not showing a convergence warning.

Explain what happens if we execute : verbatimpipe.fit(X_train, y_train)$pipe.predict(X_test, y_test)$

is the tol parameter b