Predictor #2: k-nearest neighbour predictor

The second predictor we trained was the k-nearest neighbour predictor. The original data have been splitted into training data and testing data for predictor training purposes. The job is done in two phases, data preparation, parameter tuning and predictor evaluation.

Phase 1: Data preparation

First of all, we labeled both training data and testing data by their last column, +1 represents stable, -1 represents unstable. At this point, we trained our first demo predictor using the training data based on the *knn* scikit tool (with default parameters) in python, and testing the predictor on training data, the accuracy at this stage is 0.8621 and the F-*score* is 0.7969, which is not acceptable, and then we tried a range of k, from 1 to 50, and the predictors are tested on both training data (Figure 1) and testing data (Figure 2) by cross validation, the result in show below. By observing the original data, we found that the range of each column in the input data are distinct, i.e., the range of tau1 (column 1) is around 0 to 10, and the range of g2 (column 10) is around 0 to 1, in this case, g2 may accidentally become more uninformative than tau1, for example, the calculation of distance in *knn* is based on the input features, a larger range feature has a larger impact on the distance, i.e., the Euclidian distance equation is: D(X, Y) = $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$, we normalized all data by scaling them to the range of 0 to 1.

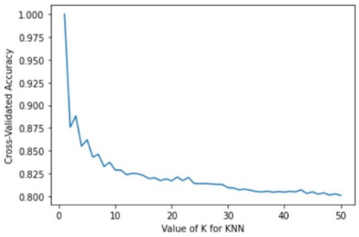


Figure 1 best k = 1, accuracy = 1 on training data

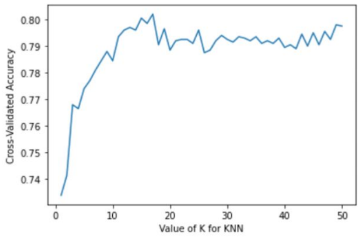


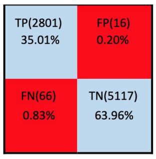
Figure 2 best k = 17, accuracy = 0.8020 on testing data

Phase 2: Parameter tuning and predictor evaluation

Parameter tuning is one of the most important processes that dominate the behaviour of a predictor, and ultimately resulting in a most optimal combination of parameters. According to the python sklearn's document, there are eight parameters, we only put three (underlined) of them into our tuning process:

- *n_neighbors*: decides the number of neighbors will be used to calculate kneighbors, 1 to 50 will be tested.
- *weights*: there are two choices, uniform or distance, uniform means all points in each neighborhood are weighted equally, and distance means the closer point is weighted more, we should test both cases.
- *algorithm*: (auto, ball_tree, kd_tree or brute), since run time is not in our consideration, they will not be tested.
- *leaf_size*: only related to the construction time of the predictor and the memory cost of the query, so we can just use the default value, which is 30.
- *p*: determines the distance calculation procedure that the predictor will use, 1 (Manhattan distance), 2 (Euclidean distance) and arbitrary number (Minkowski distance).
- *metric*: the distance metric to use for the tree, we can just accept the default algorithm.
- *metric_params*: additional keyword arguments for the metric function, we have no additional arguments.
- *n_jobs*: determines the number of jobs will run in parallel, we will not test it.

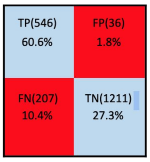
Among all possible combinations, the one that has the best accuracy on training data is { n_neighbors= 12, weights= distance, p = 1}, the accuracy is 0.8785, also avoiding overfitting, we train the model with cross-validation of n=10, and we got that The confusion matrices with *F1-score* on training data see Figure 3.



TP(2801) 35.01%	FP(16) 0.20%
FN(66) 0.83%	TN(5117) 63.96%

Figure 3:tn fp fn tp: 5117 16 66 2801, F1-SCORE = 0.985574

Applying the model to predict the test data and we got Final Evaluation on testing data set with Accuracy: 0.8785, and the confusion matrices with *F1-score* on test data see Figure 4.



TP(546) 60.6%	FP(36) 1.8%
FN(207) 10.4%	TN(1211) 27.3%

Figure 4: tn fp fn tp: 1211 36 207 546, F1-SCORE = 0.817978

Reference: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html