Introduction to Quantitative Biology
Assignment-2

In our code we have included the following functions:

1) array_of_features_of_sequence(seq):

    This function helps in providing all the features that we need to include finally in our all_features array of size 420.

2) ratio_amino_acid(seq):

    This function takes in the sequence as it's argument in which we traverse through the sequence to find the protein present at the particular position is of which type.

3) dipeptide_comp(seq):

    This function also takes a sequence as it's only argument.This function looks at the type of bond present between two entities , which could have 400 different types.Hence this function uses an array of size 400 to encounter and take care of each of these 400 different possibilities.

4) merge(arr1,arr2):

    This merges 2 different array in to single larger array by simply traversing through them.

- Our main code include the part after all the functions, we start by reading the "train.csv" file . As we know cannot direclty typecast the numpy type array, hence we have used two different arrays of features_array and label_array which later on we have added values by simply iterating into the data received from our "train.csv" file.

- The research paper provided contained input method named "Amino Acid Composition-based model" , that's what we have used. By using this idea we have calculated the ratio of each amino acid type within a peptide using the first function that we have written.

- We have finally used svm to build our model and obtain the final output.

- We also used the property GridSearchCV which was imported from the module sklearn , to predict the values of "gamma" and "C" and finally used them in the constrcutor of svm to optimise our model built using properties of svm.

- For further optimisation there are method:
    1. ***Random Forest Classifier:***

       The term *random* stems from the fact that we randomly sample the training set, and since we have a collection of trees, it's natural to call it a forest — hence Random Forest. To build the root node or any node in the tree, a random subset of features is selected. For each of these selected features, the algorithm searches for the optimal cutting point to determine the split for the given feature. The feature from the randomly selected subset that produces the purest split is then used to create the root node. The tree is grown to a depth of one, and the same process is repeated for all other nodes in the tree, until the desired depth of the tree is reached. Finally, it's important to note that each tree is built separately using a different bootstrap, which introduces variation among the trees. Consequently, each tree makes different mistakes and when combined a strong classifier can be built.

2. *Extra Trees Classifier:*

Similar to a Random Forest classifier we have the Extra Trees classifier.
Each decision stump will be built with the following criteria:

1.All the data available in the training set is used to built each stump.

2.To form the root node or any node, the best split is determined by searching in a subset of randomly selected features of size sqrt(number of features). The split of each selected feature is chosen at random.

3.The maximum depth of the decision stump is one.

Notice that the features and splits are selected at random. Since splits are chosen at random for each feature in the Extra Trees Classifier, it’s less expensive than a Random Forest.

- ➢ We also used cross validation techniques and the above explained method , we printed accuracy , we found out that the method of extra trees was superior and thus gives a better accuracy , also because it gives low variance.
- ➢ Hence we used the Extra Trees method to optimise our model.

➔ ***Note :*** Source of definition for optimisation methods of Extra trees and Random trees clasifiers was: https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b