Introduction

The main functions of the code are data conversion, marker selection and the construction and validation of the SVM(Support Vector Machine) model. All of the code is written in python and should work with version 2.7. Some parts of the code require matplotlib, numpy, scipy, and sklearn. We have packed the code as a *structural_protein_package.py* file, and you can use the following statement to import the package:

```
from mRMR_feature_package import *
```

All the data we used are stored in the *Task2 dataset* folder.

Data conversion

The datasets we obtained were stored in csv file at first. We need to convert csv file into a format that our Python script can recognize. You can call the following function to do the conversion.

```
csv_converter(csvfilename='orf_on_samples_filter_p_0.01 (大于
60%) .csv',outputfilename='dataset.txt')
```

The parameters of the function are *csvfilename* and *outputfilename* , which are two file names. Our script will read data from the *csvfilename* file(csv file) and output the conversion results to the *outputfilename* file(Conversion result).

Marker selection by mRMR

Patient discrimination gene markers were selected with a two-step scheme.

Step 1. All markers retained are first filtered by the mRMR algorithm. To run mRMR algorithm, we download a program written in C++ that can perform mRMR algorithm to filter the markers in given dataset. Then we compile the source code of the program and generate an executable file so that we can call the mRMR algorithm in python. The name of the executable file is *mrmr* and it is necessary for our Python script running. You can call the following function to perform mRMR algorithm on a given dataset.

```
run_MRMR(datasetFilename='dataset.txt',FEAS = 180)
```

The first parameter of the function is *datasetFilename*, which is the file name of the dataset converted from csv files. The second parameter of the function is *FEAS*, which is the number of markers we need to retain after running the mRMR algorithm. In detail, the *run_MRMR* function consist of 2 parts.

```
def run_MRMR(datasetFilename='dataset.txt',FEAS = 180):
    features=mRMR(datasetFilename,FEAS)   #Run mRMR algorithm
    OutputToFiles(features)     #Output the result to file
```

Note: When the *run_MRMR* function is called, the executable file of our mRMR algorithm must be placed in the same directory as the Python script and named of *mrmr*.

Step 2. In this step, we will perform an incremental search to select the optimal subset of markers. You can call the following function to perform an incremental search on a given dataset.

```
Fea_DataSetMaker(datasetFilename="dataset.txt",Best_FeasFilename='Best_Feas.txt')
```

The first parameter of the function is the same with that of the *run_MRMR* function. The second parameter of the function is *Best_FeasFilename*, which is the file name of the result of incremental search. This function will output the best markers' number to the *Best_FeasFilename* file. The function will also build the dataset of our model based on the result of incremental search, and the dataset will be output to *svmpos_in.txt* file and *svmneg_in.txt* file. In detail, the *Fea_DataSetMaker* function consist of 3 parts.

```
def Fea_DataSetMaker(datasetFilename="dataset.txt",Best_FeasFilename='Best_Feas.txt'):
    features=InputFromFiles(Best_FeasFilename)  #Read data from files
    Best_Feas = IncSearch(features,datasetFilename)  #Incremental search
    mkdataset(Best_Feas,datasetFilename, OutputToFile = 1)  #Output the data set to files
```

Model construction and validation

You can call the following function to construct and validate your SVM model.

```
svm_classifier(svmpos_inFilename="svmpos_in.txt",svmneg_inFilename="svmneg_in.txt")
```

The 2 parameters of the function are the file names of the positive dataset and the negative dataset. The function will construct an SVM model on the given dataset and evaluate the performance of the SVM model. The output of this function is the ROC curve of the SVM model built on the given dataset.