PyFeat: A Python-based Effective Features Generation Tool from DNA, RNA, and Protein Sequences

Supplementary Material

PyFeat Version 1.0

Contents

1 Introduction

The PyFest is an extensive Python-based tool for generating various numerical feature presentation schemes from DNA, RNA and protein sequences. This tool is also able to select the best features among from previously generated vast amount of features. After that, it can train model, to evaluate model using various machine learning techniques.

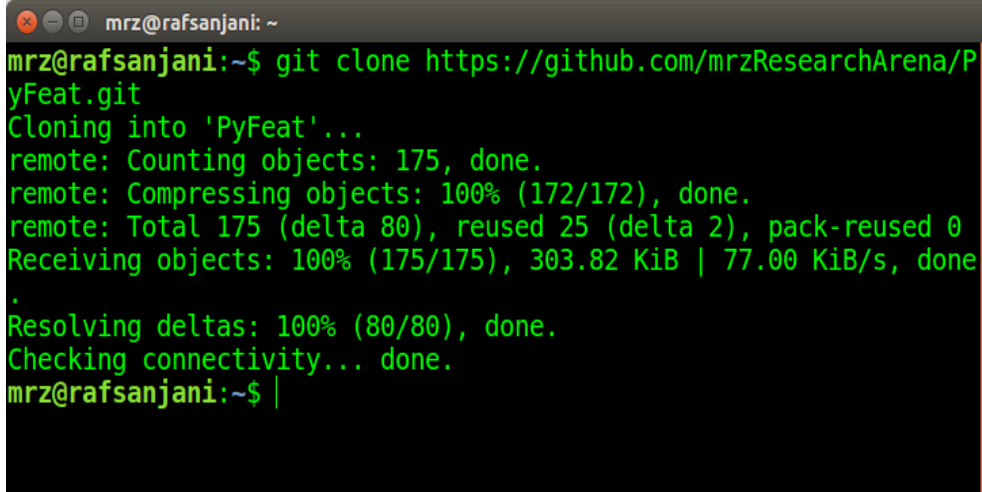# 2 Download Package

2.1 Direct Download:

PyFeat package can be **downloaded** by clicking the link. This package will be download in zip (.zip) format named PyFeat-master.zip.

2.2 Clone a GitHub Repository (Optional)

Cloning a repository syncs it to our local machine. After clone, we can add and edit files and then push and pull updates. We can follow any one procedure that given below with illustration (a) clone over HTTPS, (b) or clone over SSH.
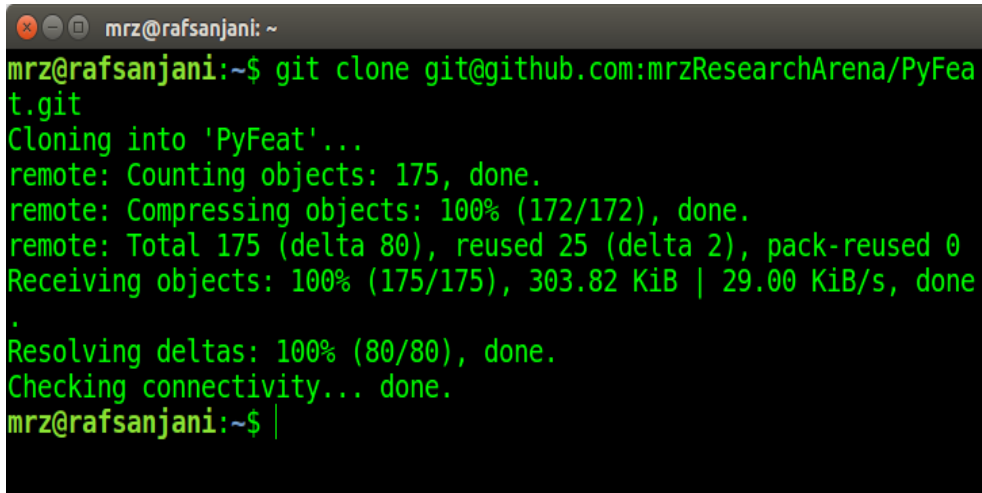
2.2.1 Clone over HTTPS

user@machine:~$ git clone https://github.com/mrzResearchArena/PyFeat.git

Figure 1: Clone over HTTPS

2.2.2 Clone over SSH

user@machine:~$ git clone git@github.com:mrzResearchArena/PyFeat.git

Figure 2: Clone over SSH

Note: If the clone was successful, a new sub-directory appears on our local drive. This directory has the same name (PyFeat) as the github repository that we cloned.

3 Installation Process

PyFest is an open-source Python-based tool, which operates depending on the Python environment (Python version 3.5 or above) and can be run on multi-OS systems (such as
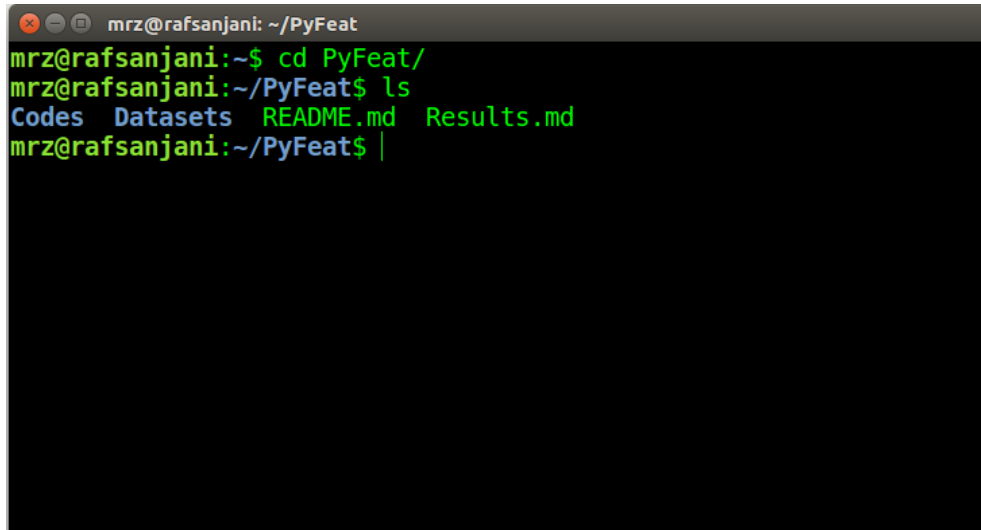
Linux-based OS, Mac OS, and Windows OS), but we will recommend you to use Linux-based OS. Before running PyFeat, a user should make sure all the following packages are installed in their Python environment:

1. Generate Features:

   - python (version $\geq$ 3.5), and
   - numpy (version $\geq$ 1.13.0).

2. Performance Measures (Optional):

   - sklearn (version $\geq$ 0.19.0),
   - pandas (version $\geq$ 0.21.0), and
   - matplotlib (version $\geq$ 2.1.0).

For convenience, we strongly recommended users to install the Anaconda (Python version $\geq$ 3.5) on your local computer. This software can be freely downloaded from https://www.anaconda.com/download/. We can also follow from https://github.com/mrzResearchArena/PyFeat/blob/master/README.md (Section 2.).

4   Package Description

PyFeat tool mainly contains two directory named **'Codes'**, and **'Datasets'**.
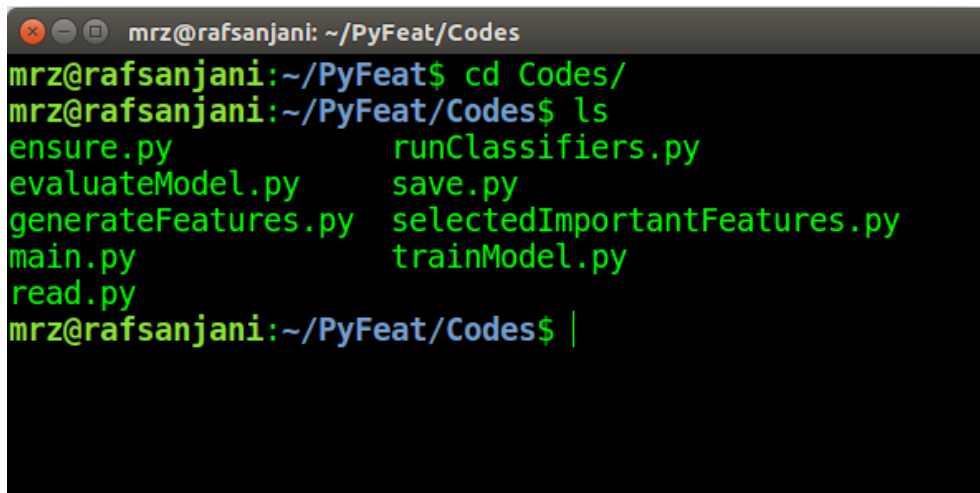
Figure 3: Enter 'PyFeat' directory and seeing all the files

The **'Codes'** directory/folder contains all the codes **(*.py files)**. PyFeat includes four main programs: **'main.py'**, **'runClassifiers.py'**, **'trainModel.py'**, and **'evaluate-Model.py'**.

4
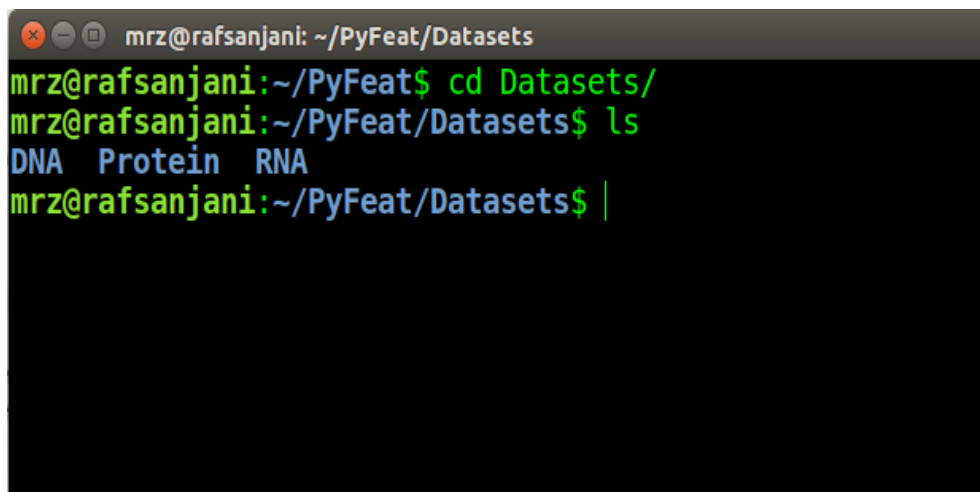
- **'main.py'** is the main programme, it will generate datasets.

- **'runClassifiers.py'** programme, it will work for n-fold cross-validation with different machine learning classifiers and also provide the classifications results.

- **'trainModel.py'** programme, it able to train an individual model.

- **'evaluateModel.py'** programme, it able to evaluate the trained model.

Figure 4: Enter 'Codes' directory and seeing all the .py files

The **'Datasets'** directory/folder contains another three directory/folder named 'DNA', 'RNA', and 'Protein' where we will find DNA, RNA, and protein sequences respectively.

Figure 5: Enter the 'Datasets' directory

5 Working Procedure
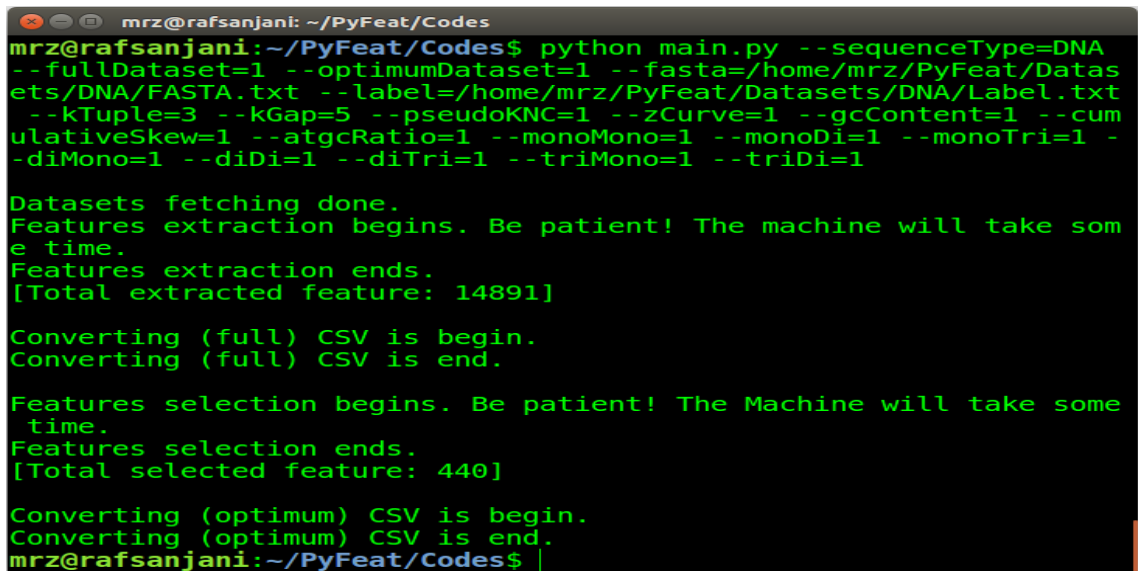
5.1 Generate Features

5.1.1 Training Purpose

Generate datasets for training purpose. Unix command line given below.

We can use full argument:
user@machine:~/PyFeat/Codes$ python main.py --sequenceType=DNA
--fullDataset=1 --optimumDataset=1
--fasta=/home/user/PyFest/Datasets/DNA/FASTA.txt
--label=/home/user/PyFest/Datasets/DNA/Labels.txt
--kTuple=3 --kGap=5
--pseudoKNC=1 --zCurve=1 --gcContent=1 --cumulativeSkew=1 --atgcRatio=1 --monoMono=1
--monoDi=1 --monoTri=1 --diMono=1 --diDi=1 --diTri=1 --triMono=1 --triDi=1

Figure 6: Training with Arguments

or, we can also use corresponding optional argument:
user@machine:~/PyFeat/Codes$ python main.py -seq=DNA
-full=1 -optimum=1
-fa=/home/user/PyFeat/Datasets/DNA/FASTA.txt
-la=/home/user/PyFeat/Datasets/DNA/Label.txt
-ktuple=3 -kgap=5
-pseudo=1 -zcurve=1 -gc=1 -skew=1 -atgc=1 -f11=1 -f12=1 -f13=1 -f21=1 -f22=1 -f23=1
-f31=1 -f32=1

Figure 7: Training with Corresponding Optional Arguments

It will generate a full dataset named **fullDataset.csv** (if -full=1 or, --fullDataset=1), and it will also generate a selected features dataset named **optimumDataset.csv** (if -optimum=1 or, --optimumDataset=1). On the contrary, if you don't want to generate full dataset simply set -full=0 or, --fullDataset=0, and if you don't want to generate optimum dataset simply set -optimum=0 or, --optimumDataset=0. The equal sign (=) is optional. To know more details about arguments: https://github.com/mrzResearchArena/PyFeat/blob/master/README.md (Table 1).

5.1.2 Evaluation Purpose

Generate datasets for evaluation purpose. Unix command line given below.

We can use full argument:
user@machine:∼/PyFeat/Codes$ python main.py --sequenceType=Protein --testDataset=1
--fasta=/home/user/PyFeat/Datasets/Protein/independentFASTA.txt
--label=/home/user/PyFeat/Datasets/Protein/independentLabel.txt
--kTuple=3 --kGap=5
--pseudoKNC=1 --zCurve=1 --gcContent=1 --cumulativeSkew=1 --atgcRatio=1 --monoMono=1
--monoDi=1 --monoTri=1 --diMono=1 --diDi=1 --diTri=1 --triMono=1 --triDi=1

or, we can also use corresponding optional argument:
user@machine:∼/PyFeat/Codes$ python main.py -seq=Protein -test=1
-fa=/home/user/PyFeat/Datasets/Protein/independentFASTA.txt
-la=/home/user/PyFeat/Datasets/Protein/independentLabel.txt
-ktuple=3 -kgap=5
-pseudo=1 -zcurve=1 -gc=1 -skew=1 -atgc=1 -f11=1 -f12=1 -f13=1 -f21=1 -f22=1 -f23=1
-f31=1 -f32=1

7
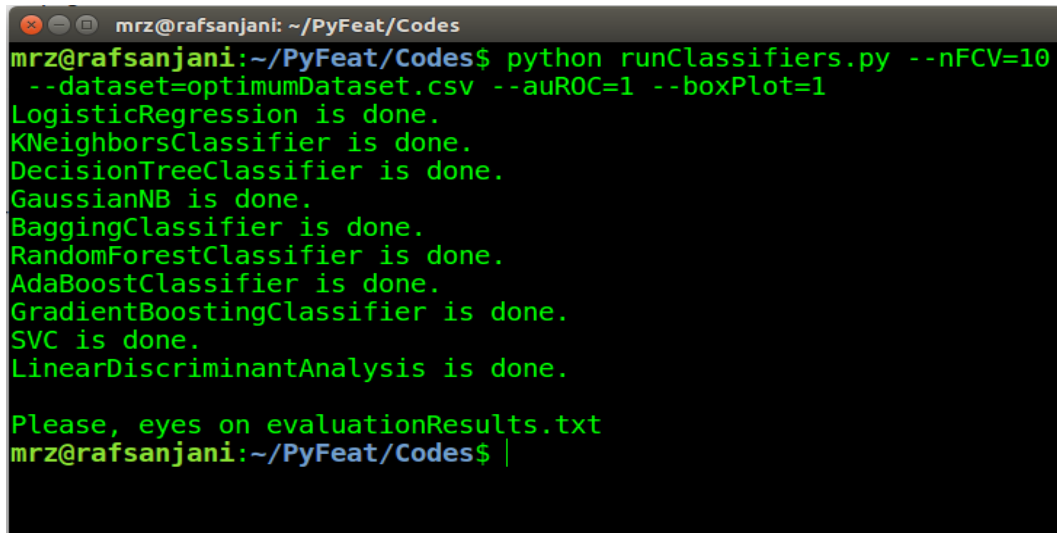
It will generate a full testing dataset named **testDataset.csv** (if -test=1 or, –testDataset==1). To know more details about arguments: `https://github.com/mrzResearchArena/PyFeat/blob/master/README.md` (Table 1).

The process will run smoothly for **valid FASTA** sequences and row-wise binary class label {0, 1}, or {-1, +1}.

5.2 Run Machine Learning Classifiers

Running machine learning classifiers with n-fold cross-validation. Unix command line given below.

user@machine:∼/PyFeat/Codes$ python runClassifiers.py `--nFCV=10` `--dataset=optimumDataset.csv` `--auROC=1` `--boxPlot=1`


```
mrz@rafsanjani: ~/PyFeat/Codes
mrz@rafsanjani:~/PyFeat/Codes$ python runClassifiers.py --nFCV=10
 --dataset=optimumDataset.csv --auROC=1 --boxPlot=1
LogisticRegression is done.
KNeighborsClassifier is done.
DecisionTreeClassifier is done.
GaussianNB is done.
BaggingClassifier is done.
RandomForestClassifier is done.
AdaBoostClassifier is done.
GradientBoostingClassifier is done.
SVC is done.
LinearDiscriminantAnalysis is done.

Please, eyes on evaluationResults.txt
mrz@rafsanjani:~/PyFeat/Codes$
```

Figure 8: Running different machine learning classifiers

It will provide classification results **(evaluationResults.txt)** from the user provides binary class dataset (.csv format), and it will also generate a ROC Curve **(auROC.png)**, and an accuracy comparison via boxPlot **(AccuracyBoxPlot.png)**. To know more details about arguments: `https://github.com/mrzResearchArena/PyFeat/blob/master/README.md` (Table 3).

5.3 Training Model

Training model with single classifier. Unix command line given below.

user@machine:∼/PyFeat/Codes$ python trainModel.py `--dataset=optimumDataset.csv` `--model=LR`

Figure 9: Training dataset with a single model

It will provide a **dumpModel.pkl** from the user provides binary class dataset (.csv format). To know more details about arguments: https://github.com/mrzResearchArena/PyFeat/blob/master/README.md (Table 4).

5.4 Evaluation Model

Evaluation model from the previously trained model. Unix command line given below.

user@machine:~/PyFeat/Codes$ python evaluateModel.py
`--optimumDatasetPath=optimumDataset.csv` `--testDatasetPath=testDataset.csv`

Here, **optimumDataset.csv**, and **testDataset.csv** using as a traing dataset and test dataset respectively. To know more details about arguments: https://github.com/mrzResearchArena/PyFeat/blob/master/README.md (Table 5).

6 Features Description

6.1 zCurve

Z-curve theory is often used in genomic sequence analysis. It has got three components in three axis. They are defined as following.

$$\begin{cases} x \ axis = (\sum A + \sum G) - (\sum C + \sum T) \\ y \ axis = (\sum A + \sum C) - (\sum G + \sum T) \\ z \ axis = (\sum A + \sum T) - (\sum G + \sum C) \end{cases} \quad (1)$$

Three features will generate using the Z-Curve method.

Figure 10: zCurve features

6.2   gcContent

In general, GC-content is expressed as a percentage value (%).

$$GC\ Content = \frac{\sum G + \sum C}{\sum A + \sum C + \sum G + \sum T} \times 100\% \qquad (2)$$

DNA with high GC-content is more stable than DNA with low GC-content. One feature will generate using the GC-content method.

Figure 11: GC Content feature

6.3 atgcRatio

Single feature will generate using the AT/GT Ratio method. The equation is given below.

$$AT/GCRatio = \frac{\sum A + \sum T}{\sum G + \sum C} \tag{3}$$

Figure 12: AT/GC ratio features

6.4 cumulativeSkew

Due to deamination process there is a difference of the count of G and T in forward and reverse strands. The forward strand often have more G and T. The cumulative skew is defined formally as:

$$GC\ skew = \frac{\sum G - \sum C}{\sum G + \sum C};\ AT\ skew = \frac{\sum A - \sum T}{\sum A + \sum T} \tag{4}$$

Figure 13: Cumulative skew features

Here as $\sum A$ represents the total number of A, $\sum C$ represents the total number of C from the sequence and so forth. Two features will generate using the cumulative skew method.

6.5 pseudoKNC

When k=n then the $\sum_{i=1}^{n} 4^i$ features will exist for DNA and RNA, but $\sum_{i=1}^{n} 20^i$ features will exist for protein.
When k=1, feature structure will be **X**.
When k=2, feature structure will be **X**, and **XX**.
When k=3, feature structure will be **X**, **XX**, and **XXX**.

Described with appropriate examples:
When k=1 then only four (4) features will exist for DNA and RNA, but twenty (20) features will exist for protein. Features will be numbers of A, C, G and T/U of the whole sequence of DNA and RNA respectively.

When k=2 then only twenty (20) features will exist for DNA and RNA, but four hundred and twenty (420) features will exist for protein. Features will be numbers of A, C, G, T, AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, and TT of the whole sequence of DNA respectively.

When k=3 then only eighty four (84) features will exist for DNA and RNA, but eight thousand four hundred and twenty (8,420) features will exist for protein. Features will be numbers of A, C, G, T, AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, TT, AAA, AAC, AAG, AAT, ACA, ACC, ACG, ACT, AGA, AGC, AGG, AGT, ATA, ATC, ATG, ATT, CAA, CAC, CAG, CAT, CCA, CCC, CCG, CCT, CGA, CGC, CGG,

12

CGT, CTA, CTC, CTG, CTT, GAA, GAC, GAG, GAT, GCA, GCC, GCG, GCT, GGA, GGC, GGG, GGT, GTA, GTC, GTG, GTT, TAA, TAC, TAG, TAT, TCA, TCC, TCG, TCT, TGA, TGC, TGG, TGT, TTA, TTC, TTG, and TTT of the whole sequence of DNA respectively.

Figure 14: PseudoKNC features with -ktuple=3

6.5.1 important definitions

$$\mathbf{X} = \begin{cases} \{A,C,T,G\}, & \text{if the problem involves DNA sequences} \\ \{A,C,T,U\}, & \text{if the problem involves RNA sequences} \\ \{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y\}, & \text{if the problem involves protein sequences} \end{cases}$$

$x_i \in \mathbf{X}$ where i specifies the position of x in some subsequence. Counts of such subsequences of varying lengths is regarded as features in our method.

$j \in \{1, 2, 3...k\}$ where j specifies the number of gaps (don't care) in a subsequence.

6.6 monoMonoKGap

When -kgap=n then the $(4) \times (4) \times n$ features will exist for DNA and RNA but $(20) \times (20) \times n$ features will exist for protein.
When -kgap=1, feature structure will be **X_X**.
When -kgap=2, feature structure will be **X_X**, and **X__X**.
When -kgap=3, feature structure will be **X_X**. **X__X**, and **X___X**.

Described with appropriate examples:

13

When -kgap=1 then only sixteen (16) features will exist for DNA and RNA but four hundred (400) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A, C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G, and T_T of the whole sequence of DNA respectively.

When -kgap=2 then only thirty two (32) features will exist for DNA and RNA but eight hundred (800) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A, C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G, T_T, A__A, A__C, A__G, A__T, C__A, C__C, C__G, C__T, G__A, G__C, G__G, G__T, T__A, T__C, T__G, and T__T of the whole sequence of DNA respectively.

When -kgap=3 then only forty eight (48) features will exist for DNA and RNA, but one thousand and two hundred (1,200) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A, C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G, T_T, A__A, A__C, A__G, A__T, C__A, C__C, C__G, C__T, G__A, G__C, G__G, G__T, T__A, T__C, T__G, T__T, A___A, A___C, A___G, A___T, C___A, C___C, C___G, C___T, G___A, G___C, G___G, G___T, T___A, T___C, T___G, and T___T of the whole sequence of DNA respectively.

Figure 15: monoMonoKGap features when -kgap=1

6.7 monoDiKGap

When -kgap=n then the $(4) \times (4 \times 4) \times n$ features will exist for DNA and RNA, but $(20) \times (20 \times 20) \times n$ features will exist for protein.
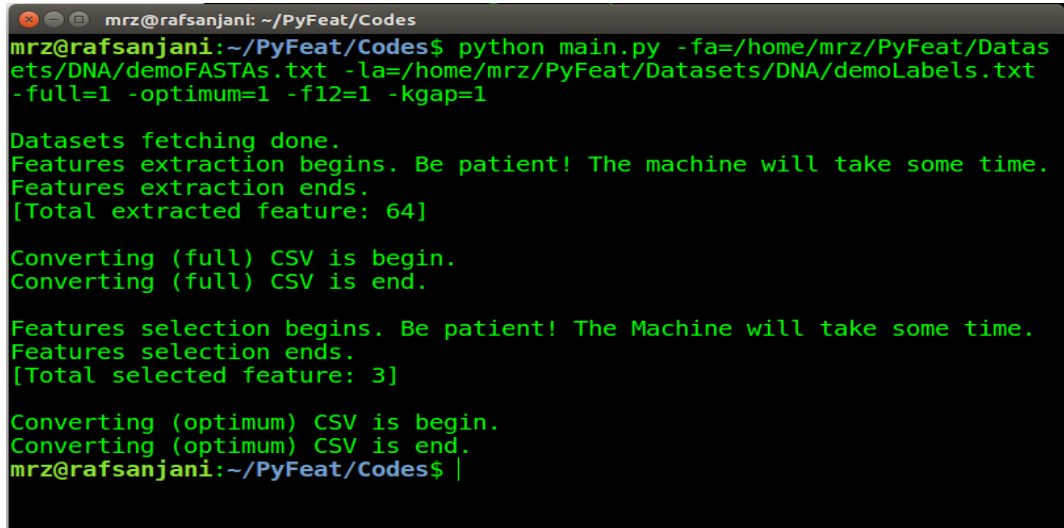When -kgap=1, feature structure will be **X_XX**.
When -kgap=2, feature structure will be **X_XX**, and **X__XX**.
When -kgap=3, feature structure will be **X_XX**, **X__XX**, and **X___XX**.

14

Described with appropriate examples:

When -kgap=1 then only sixty four (64) features will exist for DNA and RNA, but eight thousand (8,000) features will exist for protein. Features will be numbers of A_AA, A_AC, A_AG, A_AT, A_CA, A_CC, A_CG, A_CT, A_GA, A_GC, A_GG, A_GT, A_TA, A_TC, A_TG, A_TT, C_AA, C_AC, C_AG, C_AT, C_CA, C_CC, C_CG, C_CT, C_GA, C_GC, C_GG, C_GT, C_TA, C_TC, C_TG, C_TT, G_AA, G_AC, G_AG, G_AT, G_CA, G_CC, G_CG, G_CT, G_GA, G_GC, G_GG, G_GT, G_TA, G_TC, G_TG, G_TT, T_AA, T_AC, T_AG, T_AT, T_CA, T_CC, T_CG, T_CT, T_GA, T_GC, T_GG, T_GT, T_TA, T_TC, T_TG, and T_TT of the whole sequence of DNA respectively.

When -kgap=2 then only hundred and twenty eight (128) features will exist for DNA and RNA, but sixteen thousand (16,000) features will exist for protein. Features will be numbers of A_AA, A_AC, A_AG, A_AT, A_CA, A_CC, A_CG, A_CT, A_GA, A_GC, A_GG, A_GT, A_TA, A_TC, A_TG, A_TT, C_AA, C_AC, C_AG, C_AT, C_CA, C_CC, C_CG, C_CT, C_GA, C_GC, C_GG, C_GT, C_TA, C_TC, C_TG, C_TT, G_AA, G_AC, G_AG, G_AT, G_CA, G_CC, G_CG, G_CT, G_GA, G_GC, G_GG, G_GT, G_TA, G_TC, G_TG, G_TT, T_AA, T_AC, T_AG, T_AT, T_CA, T_CC, T_CG, T_CT, T_GA, T_GC, T_GG, T_GT, T_TA, T_TC, T_TG, T_TT, A__AA, A__AC, A__AG, A__AT, A__CA, A__CC, A__CG, A__CT, A__GA, A__GC, A__GG, A__GT, A__TA, A__TC, A__TG, A__TT, C__AA, C__AC, C__AG, C__AT, C__CA, C__CC, C__CG, C__CT, C__GA, C__GC, C__GG, C__GT, C__TA, C__TC, C__TG, C__TT, G__AA, G__AC, G__AG, G__AT, G__CA, G__CC, G__CG, G__CT, G__GA, G__GC, G__GG, G__GT, G__TA, G__TC, G__TG, G__TT, T__AA, T__AC, T__AG, T__AT, T__CA, T__CC, T__CG, T__CT, T__GA, T__GC, T__GG, T__GT, T__TA, T__TC, T__TG, and T__TT of the whole sequence of DNA respectively.

Figure 16: monoDiKGap features when -kgap=1

6.8 monoTriKGap

When -kgap=n then the $(4) \times (4 \times 4 \times 4) \times n$ features will exist for DNA and RNA, but $(20) \times (20 \times 20 \times 20) \times n$ features will exist for protein.

When -kgap=1, feature structure will be **X_XXX**.

When -kgap=2, feature structure will be **X_XXX**, and **X__XXX**.

When -kgap=3, feature structure will be **X_XXX**, **X__XXX**, and **X___XXX**.

Described with appropriate examples:

When -kgap=1 then only two hundred and fifty six (256) features will exist for DNA and RNA, but hundred and sixty thousand (160,000) a huge amount of features will exist for protein. Features will be numbers of A_AAA, A_AAC, A_AAG, A_AAT, A_ACA, A_ACC, A_ACG, A_ACT, A_AGA, A_AGC, A_AGG, A_AGT, A_ATA, A_ATC, A_ATG, A_ATT, A_CAA, A_CAC, A_CAG, A_CAT, A_CCA, A_CCC, A_CCG, A_CCT, A_CGA, A_CGC, A_CGG, A_CGT, A_CTA, A_CTC, A_CTG, A_CTT, A_GAA, A_GAC, A_GAG, A_GAT, A_GCA, A_GCC, A_GCG, A_GCT, A_GGA, A_GGC, A_GGG, A_GGT, A_GTA, A_GTC, A_GTG, A_GTT, A_TAA, A_TAC, A_TAG, A_TAT, A_TCA, A_TCC, A_TCG, A_TCT, A_TGA, A_TGC, A_TGG, A_TGT, A_TTA, A_TTC, A_TTG, A_TTT, C_AAA, C_AAC, C_AAG, C_AAT, C_ACA, C_ACC, C_ACG, C_ACT, C_AGA, C_AGC, C_AGG, C_AGT, C_ATA, C_ATC, C_ATG, C_ATT, C_CAA, C_CAC, C_CAG, C_CAT, C_CCA, C_CCC, C_CCG, C_CCT, C_CGA, C_CGC, C_CGG, C_CGT, C_CTA, C_CTC, C_CTG, C_CTT, C_GAA, C_GAC, C_GAG, C_GAT, C_GCA, C_GCC, C_GCG, C_GCT, C_GGA, C_GGC, C_GGG, C_GGT, C_GTA, C_GTC, C_GTG, C_GTT, C_TAA, C_TAC, C_TAG, C_TAT, C_TCA, C_TCC, C_TCG, C_TCT, C_TGA, C_TGC, C_TGG, C_TGT, C_TTA, C_TTC, C_TTG, C_TTT, G_AAA, G_AAC, G_AAG, G_AAT, G_ACA, G_ACC, G_ACG, G_ACT, G_AGA, G_AGC, G_AGG, G_AGT, G_ATA, G_ATC, G_ATG, G_ATT, G_CAA, G_CAC, G_CAG, G_CAT, G_CCA, G_CCC, G_CCG, G_CCT, G_CGA, G_CGC, G_CGG, G_CGT, G_CTA, G_CTC, G_CTG, G_CTT, G_GAA, G_GAC, G_GAG, G_GAT, G_GCA, G_GCC, G_GCG, G_GCT, G_GGA, G_GGC, G_GGG, G_GGT, G_GTA, G_GTC, G_GTG, G_GTT, G_TAA, G_TAC, G_TAG, G_TAT, G_TCA, G_TCC, G_TCG, G_TCT, G_TGA, G_TGC, G_TGG, G_TGT, G_TTA, G_TTC, G_TTG, G_TTT, T_AAA, T_AAC, T_AAG, T_AAT, T_ACA, T_ACC, T_ACG, T_ACT, T_AGA, T_AGC, T_AGG, T_AGT, T_ATA, T_ATC, T_ATG, T_ATT, T_CAA, T_CAC, T_CAG, T_CAT, T_CCA, T_CCC, T_CCG, T_CCT, T_CGA, T_CGC, T_CGG, T_CGT, T_CTA, T_CTC, T_CTG, T_CTT, T_GAA, T_GAC, T_GAG, T_GAT, T_GCA, T_GCC, T_GCG, T_GCT, T_GGA, T_GGC, T_GGG, T_GGT, T_GTA, T_GTC, T_GTG, T_GTT, T_TAA, T_TAC, T_TAG, T_TAT, T_TCA, T_TCC, T_TCG, T_TCT, T_TGA, T_TGC, T_TGG, T_TGT, T_TTA, T_TTC, T_TTG, and T_TTT of the whole sequence of DNA respectively.

When -kgap=2 then only hundred and twenty eight (512) features will exist for DNA and RNA, but sixteen thousand (320,000) a huge amount of features will exist for protein. Features will be numbers of A_AAA, A_AAC, A_AAG, A_AAT, A_ACA, A_ACC, A_ACG, A_ACT, A_AGA, A_AGC, A_AGG, A_AGT, A_ATA, A_ATC, A_ATG, A_ATT, A_CAA, A_CAC, A_CAG, A_CAT, A_CCA, A_CCC, A_CCG, A_CCT, A_CGA, A_CGC, A_CGG, A_CGT, A_CTA, A_CTC, A_CTG, A_CTT, A_GAA, A_GAC, A_GAG, A_GAT, A_GCA,

A_GCC, A_GCG, A_GCT, A_GGA, A_GGC, A_GGG, A_GGT, A_GTA, A_GTC, A_GTG,
A_GTT, A_TAA, A_TAC, A_TAG, A_TAT, A_TCA, A_TCC, A_TCG, A_TCT, A_TGA,
A_TGC, A_TGG, A_TGT, A_TTA, A_TTC, A_TTG, A_TTT, C_AAA, C_AAC, C_AAG,
C_AAT, C_ACA, C_ACC, C_ACG, C_ACT, C_AGA, C_AGC, C_AGG, C_AGT, C_ATA,
C_ATC, C_ATG, C_ATT, C_CAA, C_CAC, C_CAG, C_CAT, C_CCA, C_CCC, C_CCG,
C_CCT, C_CGA, C_CGC, C_CGG, C_CGT, C_CTA, C_CTC, C_CTG, C_CTT, C_GAA,
C_GAC, C_GAG, C_GAT, C_GCA, C_GCC, C_GCG, C_GCT, C_GGA, C_GGC, C_GGG,
C_GGT, C_GTA, C_GTC, C_GTG, C_GTT, C_TAA, C_TAC, C_TAG, C_TAT, C_TCA,
C_TCC, C_TCG, C_TCT, C_TGA, C_TGC, C_TGG, C_TGT, C_TTA, C_TTC, C_TTG,
C_TTT, G_AAA, G_AAC, G_AAG, G_AAT, G_ACA, G_ACC, G_ACG, G_ACT, G_AGA,
G_AGC, G_AGG, G_AGT, G_ATA, G_ATC, G_ATG, G_ATT, G_CAA, G_CAC, G_CAG,
G_CAT, G_CCA, G_CCC, G_CCG, G_CCT, G_CGA, G_CGC, G_CGG, G_CGT, G_CTA,
G_CTC, G_CTG, G_CTT, G_GAA, G_GAC, G_GAG, G_GAT, G_GCA, G_GCC, G_GCG,
G_GCT, G_GGA, G_GGC, G_GGG, G_GGT, G_GTA, G_GTC, G_GTG, G_GTT, G_TAA,
G_TAC, G_TAG, G_TAT, G_TCA, G_TCC, G_TCG, G_TCT, G_TGA, G_TGC, G_TGG,
G_TGT, G_TTA, G_TTC, G_TTG, G_TTT, T_AAA, T_AAC, T_AAG, T_AAT, T_ACA,
T_ACC, T_ACG, T_ACT, T_AGA, T_AGC, T_AGG, T_AGT, T_ATA, T_ATC, T_ATG,
T_ATT, T_CAA, T_CAC, T_CAG, T_CAT, T_CCA, T_CCC, T_CCG, T_CCT, T_CGA,
T_CGC, T_CGG, T_CGT, T_CTA, T_CTC, T_CTG, T_CTT, T_GAA, T_GAC, T_GAG,
T_GAT, T_GCA, T_GCC, T_GCG, T_GCT, T_GGA, T_GGC, T_GGG, T_GGT, T_GTA,
T_GTC, T_GTG, T_GTT, T_TAA, T_TAC, T_TAG, T_TAT, T_TCA, T_TCC, T_TCG,
T_TCT, T_TGA, T_TGC, T_TGG, T_TGT, T_TTA, T_TTC, T_TTG, T_TTT, A__AAA,
A__AAC, A__AAG, A__AAT, A__ACA, A__ACC, A__ACG, A__ACT, A__AGA, A__AGC, A__AGG,
A__AGT, A__ATA, A__ATC, A__ATG, A__ATT, A__CAA, A__CAC, A__CAG, A__CAT, A__CCA,
A__CCC, A__CCG, A__CCT, A__CGA, A__CGC, A__CGG, A__CGT, A__CTA, A__CTC, A__CTG,
A__CTT, A__GAA, A__GAC, A__GAG, A__GAT, A__GCA, A__GCC, A__GCG, A__GCT,
A__GGA, A__GGC, A__GGG, A__GGT, A__GTA, A__GTC, A__GTG, A__GTT, A__TAA,
A__TAC, A__TAG, A__TAT, A__TCA, A__TCC, A__TCG, A__TCT, A__TGA, A__TGC, A__TGG,
A__TGT, A__TTA, A__TTC, A__TTG, A__TTT, C__AAA, C__AAC, C__AAG, C__AAT, C__ACA,
C__ACC, C__ACG, C__ACT, C__AGA, C__AGC, C__AGG, C__AGT, C__ATA, C__ATC, C__ATG,
C__ATT, C__CAA, C__CAC, C__CAG, C__CAT, C__CCA, C__CCC, C__CCG, C__CCT, C__CGA,
C__CGC, C__CGG, C__CGT, C__CTA, C__CTC, C__CTG, C__CTT, C__GAA, C__GAC, C__GAG,
C__GAT, C__GCA, C__GCC, C__GCG, C__GCT, C__GGA, C__GGC, C__GGG, C__GGT,
C__GTA, C__GTC, C__GTG, C__GTT, C__TAA, C__TAC, C__TAG, C__TAT, C__TCA, C__TCC,
C__TCG, C__TCT, C__TGA, C__TGC, C__TGG, C__TGT, C__TTA, C__TTC, C__TTG, C__TTT,
G__AAA, G__AAC, G__AAG, G__AAT, G__ACA, G__ACC, G__ACG, G__ACT, G__AGA,
G__AGC, G__AGG, G__AGT, G__ATA, G__ATC, G__ATG, G__ATT, G__CAA, G__CAC, G__CAG,
G__CAT, G__CCA, G__CCC, G__CCG, G__CCT, G__CGA, G__CGC, G__CGG, G__CGT,
G__CTA, G__CTC, G__CTG, G__CTT, G__GAA, G__GAC, G__GAG, G__GAT, G__GCA,
G__GCC, G__GCG, G__GCT, G__GGA, G__GGC, G__GGG, G__GGT, G__GTA, G__GTC,
G__GTG, G__GTT, G__TAA, G__TAC, G__TAG, G__TAT, G__TCA, G__TCC, G__TCG, G__TCT,
G__TGA, G__TGC, G__TGG, G__TGT, G__TTA, G__TTC, G__TTG, G__TTT, T__AAA,
T__AAC, T__AAG, T__AAT, T__ACA, T__ACC, T__ACG, T__ACT, T__AGA, T__AGC, T__AGG,
T__AGT, T__ATA, T__ATC, T__ATG, T__ATT, T__CAA, T__CAC, T__CAG, T__CAT, T__CCA,
T__CCC, T__CCG, T__CCT, T__CGA, T__CGC, T__CGG, T__CGT, T__CTA, T__CTC, T__CTG,

T__CTT, T__GAA, T__GAC, T__GAG, T__GAT, T__GCA, T__GCC, T__GCG, T__GCT, T__GGA, T__GGC, T__GGG, T__GGT, T__GTA, T__GTC, T__GTG, T__GTT, T__TAA, T__TAC, T__TAG, T__TAT, T__TCA, T__TCC, T__TCG, T__TCT, T__TGA, T__TGC, T__TGG, T__TGT, T__TTA, T__TTC, T__TTG, and T__TTT of the whole sequence of DNA respectively.

Figure 17: monoTriKGap features when -kgap=1

6.9 diMonoKGap

When -kgap=n then the $(4 \times 4) \times (4) \times n$ features will exist for DNA and RNA but $(20 \times 20) \times (20) \times n$ features will exist for protein.

When -kgap=1, feature structure will be **XX_X**.

When -kgap=2, feature structure will be **XX_X**, and **XX__X**.

When -kgap=3, feature structure will be **XX_X**, **XX__X**, and **XX___X**.

Figure 18: diMonoKGap features when -kgap=1

6.10   diDiKGap

When -kgap=n then the $(4 \times 4) \times (4 \times 4) \times n$ features will exist for DNA and RNA but $(20 \times 20) \times (20 \times 20) \times n$ features will exist for protein.

When -kgap=1, feature structure will be **XX_XX**.

When -kgap=2, feature structure will be **XX_XX**, and **XX__XX**.

When -kgap=3, feature structure will be **XX_XX**, **XX__XX**, and, **XX___XX**.
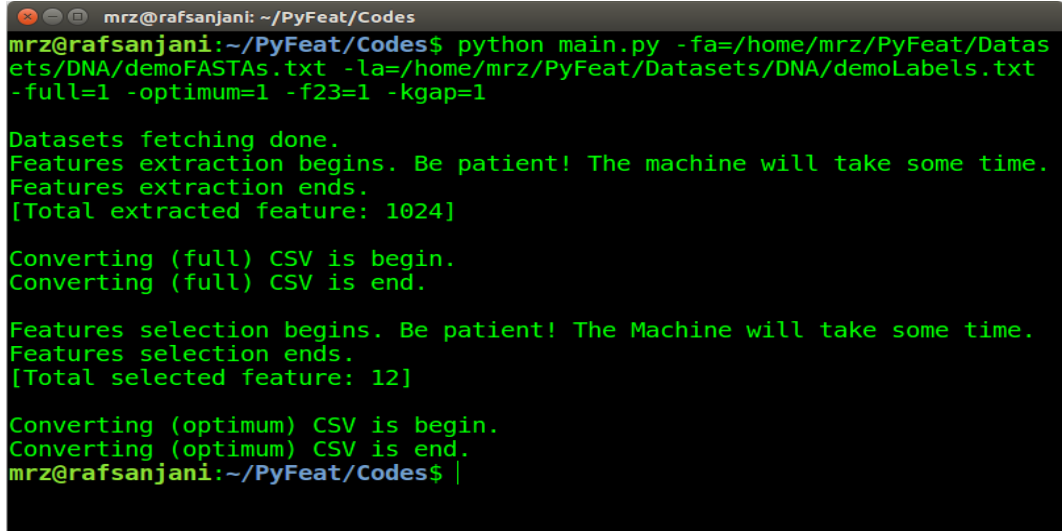

Figure 19: diDiKGap features with -kgap=1

6.11   diTriKGap

When -kgap=n then the $(4 \times 4) \times (4 \times 4 \times 4) \times n$ features will exist for DNA and RNA but $(20 \times 20) \times (20 \times 20 \times 20) \times n$ features will exist for protein.
When -kgap=1, feature structure will be **XX_XXX**.
When -kgap=2, feature structure will be **XX_XXX**, and **XX__XXX**.
When -kgap=3, feature structure will be **XX_XXX**, **XX__XXX**, and **XX___XXX**.

Figure 20: diTriKGap features with -kgap=1

6.12   triMonoKGap

When -kgap=n then the $(4 \times 4 \times 4) \times 4 \times n$ features will exist for DNA and RNA but $(20 \times 20 \times 20) \times 20 \times n$ features will exist for protein.
When -kgap=1, feature structure will be **XXX_X**.
When -kgap=2, feature structure will be **XXX_X**, and **XXX__X**.
When -kgap=3, feature structure will be **XXX_X**, **XXX__X**, and **XXX___X**.

Figure 21: triMonoKGap features with -kgap=1

6.13 triDiKGap

When -kgap=n then the $(4 \times 4 \times 4) \times (4 \times 4) \times n$ features will exist for DNA and RNA but $(20 \times 20 \times 20) \times (20 \times 20) \times n$ features will exist for protein.

When -kgap=1, feature structure will be **XXX_XX**.

When -kgap=2, feature structure will be **XXX_XX**, and **XXX__XX**.

When -kgap=3, feature structure will be **XXX_XX**, **XXX__XX**, and **XXX___XX**.

Figure 22: triDiKGap features with -kgap=1

References

[1] Bin Liu, Fule Liu, Longyun Fang, Xiaolong Wang, and Kuo-Chen Chou. repdna: a python package to generate various modes of feature vectors for dna sequences by incorporating user-defined physicochemical properties and sequence-order effects. *Bioinformatics*, 31(8):1307–1309, 2014.

[2] Dong-Sheng Cao, Qing-Song Xu, and Yi-Zeng Liang. propy: a tool to generate various modes of chous pseaac. *Bioinformatics*, 29(7):960–962, 2013.

[3] Bin Liu. Bioseq-analysis: a platform for dna, rna and protein sequence analysis based on machine learning approaches. *Briefings in bioinformatics*, 2017.

[4] Zhen Chen, Pei Zhao, Fuyi Li, André Leier, Tatiana T Marquez-Lago, Yanan Wang, Geoffrey I Webb, A Ian Smith, Roger J Daly, Kuo-Chen Chou, et al. ifeature: a python package and web server for features extraction and selection from protein and peptide sequences. *Bioinformatics*, 1:4, 2018.

[5] Bin Liu, Hao Wu, Deyuan Zhang, Xiaolong Wang, and Kuo-Chen Chou. Pse-analysis: a python package for dna/rna and protein/peptide sequence analysis based on pseudo components and kernel methods. *Oncotarget*, 8(8):13338, 2017.

[6] Bin Liu, Fule Liu, Xiaolong Wang, Junjie Chen, Longyun Fang, and Kuo-Chen Chou. Pse-in-one: a web server for generating various modes of pseudo components of dna, rna, and protein sequences. *Nucleic acids research*, 43(W1):W65–W71, 2015.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[8] Bin Liu, Fule Liu, Longyun Fang, Xiaolong Wang, and Kuo-Chen Chou. reprna: a web server for generating various feature vectors of rna sequences. *Molecular Genetics and Genomics*, 291(1):473–481, 2016.