iProps: A cross-platform protein recognition software

Instruction Manual

Author: Changli Feng et al.

Institute: Taishan University & Institute of Fundamental and Frontier Sciences

Date: Oct. 07, 2023

Version: 1.0

Bio: Information

Contents

Chapter 1 Introduction

This paper introduces iProps, a newly developed pure Python package specifically designed to address these aforementioned functionalities. iProps excels in feature extraction, evaluation, automated machine learning, and classification model interpretation. Moreover, it also offers extensive reduction information through 568 reduction schemes, supporting feature combinations and evaluations. iProps efficiently identifies the most suitable protein class recognition feature among thousands of candidates, employs three automatic machine learning methods to search for optimal classifiers and parameters, and generates a comprehensive explanatory report comprising 23 graphs derived from three interpretable models. Additionally, iProps provides a file interface enabling the importation of supplementary numerical feature files from other software. iProps not only aids bioinformatics research but also extends its utility by providing robust data analysis tools applicable to diverse fields through its automatic machine learning and model interpretation modules. iProps is an open-source computational platform that is freely accessible. It offers a user-friendly interface design that allows for effortless navigation, enabling individuals without programming experience to use it seamlessly. The source code of the software is available for download at the following website: https://github.com/LigosQ/iProps.

For any other question, suggestion or comment, feel free to contact us on GitHub issues or email us at `tafchl(at)mail.nankai.edu.cn`.

Note *To avoid harassment, @ in the above email address has been replaced with (at). If you want to contact me, you can replace (at) with @.*

Repository information:

- GitHub: https://github.com/LigosQ/iProps
- Gitee: https://gitee.com/LigosQ/iProps

(Cite our paper)

Our papers are being submitted. Please quote our previous papers if you need to quote them. Feng Changli, Wu Jin, Wei Haiyan, Xu Lei, Zou Quan. CRCF: A Method of Identifying Secretory Proteins of Malaria Parasites. IEEE/ACM Trans Comput Biol Bioinform. 2022 Jul-Aug;19(4):2149-2157. doi: 10.1109/TCBB.2021.3085589. Epub 2022 Aug 8. PMID: 34061749.

In building our code, we refer to the results of the following authors:

(Our Citation)

1. *Liu, B. , Wu, H. and Chou, K. (2017) Pse-in-One 2.0: An Improved Package of Web Servers for Generating Various Modes of Pseudo Components of DNA, RNA, and Protein Sequences. Natural Science, 9, 67-91. doi: 10.4236/ns.2017.94007.*
2. *Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ, Chou KC, Song J. iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics. 2018 Jul 15;34(14):2499-2502. doi: 10.1093/bioinformatics/bty140. PMID: 29528364; PMCID: PMC6658705.*
3. *Lei Zheng, Shenghui Huang, Nengjiang, Mu, Haoyue, Zhang, Jiayu Zhang, Yu Chang, Lei Yang, Yongchun Zuo*, RAACBook: a web server of reduced amino acid alphabet for sequence-dependent inference by using the Chou's 5-steps rule, Database, 2019, DOI: 10.1093/database/baz131.*

Anaconda3-2023.03-Windows-x86_64.exe	786.0 MiB	2023-03-21 01:57
Anaconda3-2023.07-0-Linux-aarch64.sh	711.9 MiB	2023-07-12 02:23
Anaconda3-2023.07-0-Linux-ppc64le.sh	468.7 MiB	2023-07-12 02:23
Anaconda3-2023.07-0-Linux-s390x.sh	336.1 MiB	2023-07-12 02:23
Anaconda3-2023.07-0-Linux-x86_64.sh	1010.4 MiB	2023-07-12 02:23
Anaconda3-2023.07-0-MacOSX-arm64.pkg	628.2 MiB	2023-07-12 02:23
Anaconda3-2023.07-0-MacOSX-arm64.sh	629.9 MiB	2023-07-12 02:23
Anaconda3-2023.07-0-MacOSX-x86_64.pkg	593.8 MiB	2023-07-12 02:24
Anaconda3-2023.07-0-MacOSX-x86_64.sh	595.4 MiB	2023-07-12 02:24
Anaconda3-2023.07-0-Windows-x86_64.exe	893.9 MiB	2023-07-12 02:24
Anaconda3-2023.07-1-Linux-aarch64.sh	711.9 MiB	2023-07-14 04:37
Anaconda3-2023.07-1-Linux-ppc64le.sh	468.7 MiB	2023-07-14 04:37
Anaconda3-2023.07-1-Linux-s390x.sh	336.1 MiB	2023-07-14 04:37
Anaconda3-2023.07-1-Linux-x86_64.sh	1010.4 MiB	2023-07-14 04:37
Anaconda3-2023.07-1-MacOSX-arm64.pkg	628.1 MiB	2023-07-14 04:37
Anaconda3-2023.07-1-MacOSX-arm64.sh	629.9 MiB	2023-07-14 04:37
Anaconda3-2023.07-1-MacOSX-x86_64.pkg	593.8 MiB	2023-07-14 04:38
Anaconda3-2023.07-1-MacOSX-x86_64.sh	595.4 MiB	2023-07-14 04:38
Anaconda3-2023.07-1-Windows-x86_64.exe	893.8 MiB	2023-07-14 04:38

Figure 1.1: Software versions for different platforms

1.1 Basic Environment

Note *If you're familiar with configuring your Python environment, you can skip this section and move on to the following chapters.*

In order to reduce problems such as package missing during the use of our software, we suggest users to build a basic development environment using Anconda platform. If you already have Anaconda installed on your computer or you are familiar with the pip command to manage packages, you can skip the Anaconda installation steps. For faster download, we recommend using a domestic image source to download the Anaconda installation package. For example, Tsinghua mirror source can be accessed via the following link: Download.

As shown in the figure 1.1, a variety of software package versions for different platforms are available in the above web page. Users can download and install software packages based on their platform types. You can download the installation package with the "Mac OSX-ARM64" logo for Mac M1 and M2 series computers, download the installation package with the "MAC OSX-X86_64" logo for MAC Intel series computers. Linux device Select the installation package whose name contains "Linux". The name of the installation package for Windows series computers contains "Windows".

Note *This software is developed based on the version Anaconda3-2023.07-0, which can be seen in the figure 1.1. This version uses Python 3.11. The compatibility of the code to other versions has not been tested. If you want to run this version of the code more smoothly, it is recommended to use the version number in the figure. Compatibility with other versions of Python can also be contacted by email.*

1.2 Software Download

You can easily download our latest software version on GitHub or Gitee.

Because of Python's cross-platform nature, this software script should be available across devices. Further, our code has been tested on Windows 11 and Mac OS devices based on the M1 chip. On Linux devices, the terminal-based code has not yet been developed. Compatibility with other system versions or device types can also be communicated to us.

1.3 Software Installation

Before using the software, run the "setupList_linux.sh" script in the Python environment to install the basic software package.

If you are familiar with Python development, you can pick your machine's uninstalled packages from the following list of packages. Then, use the pip command to install the picked packages.

Mac Os:

```
pip install applescript==2021.2.9
pip install FLAML==2.0.2
pip install flet==0.11.0
pip install imbalanced_learn==0.10.1
pip install lazypredict==0.2.12
conda install -c conda-forge lightgbm
pip install nest_asyncio==1.5.6
pip install nicegui
pip install prettytable==3.9.0
pip install shap==0.42.1
pip install sweetviz==2.2.1
pip install TPOT==0.12.1
pip install umap_learn==0.5.3
pip install xgboost==1.7.6
pip install yellowbrick==1.5
pip install pywebview==4.4.1
```

Window OS:

```
pip install applescript==2021.2.9
pip install FLAML==2.0.2
pip install flet==0.11.0
pip install imbalanced_learn==0.10.1
pip install lazypredict==0.2.12
pip install nest_asyncio==1.5.6
pip install nicegui
pip install prettytable==3.9.0
pip install shap==0.42.1
pip install sweetviz==2.2.1
pip install TPOT==0.12.1
pip install umap-learn==0.5.3
pip install xgboost==1.7.6
pip install yellowbrick==1.5
pip install pywebview==4.4.1
pip install lightgbm
pip install pypiwin32
python -m pip install --user --upgrade pywin32
```

You can use "Python –version" from the command line to see which version of Python is installed. You can also use "pip list" to see if the above Python libraries are installed on your machine. This applies if you only have one Python installed on your machine. If you have more than one Python environment installed on your

computer, this command will need to be adjusted accordingly.

Chapter 2 How to Use

Introduction

- ❏ *Main window*
- ❏ *Feature evaluation*
- ❏ *progress information*
- ❏ *Results*
- ❏ *Visualization*
- ❏ *Performance*
- ❏ *Extract Files*
- ❏ *Help information*

2.1 Main Window

- A. Unzip the archive of this code into a folder or directory. It is important to note that all directory names in the path of this directory cannot appear Spaces, special symbols and other special symbols. The current version only supports directory names with characters and numbers.
- B. Open a terminal window under Mac/Linux OS (or a cmd window under a Windows OS), Use the “cd“ command to navigate to the unzipped directory.
- C. Type ”python AutoML_Plus.py” to open the main program.
- D. The fig. 2.1 shows the main window of our software.

Figure 2.1: The main window of the software

- You can click on the first plot to run the automatic feature evaluation.
- You can click on the second graph to set custom parameters.
- You can click on the third plot to get the optimal classifier and generate the machine learning model explanation.

2.2 Feature Evaluation

The user gets a new window which is shown in Fig. 2.3 when he/she clicked the image button ”Auto Feature Evaluation” in the main window. This window briefly introduces the main functions of the feature evaluation section as well as our paper information.

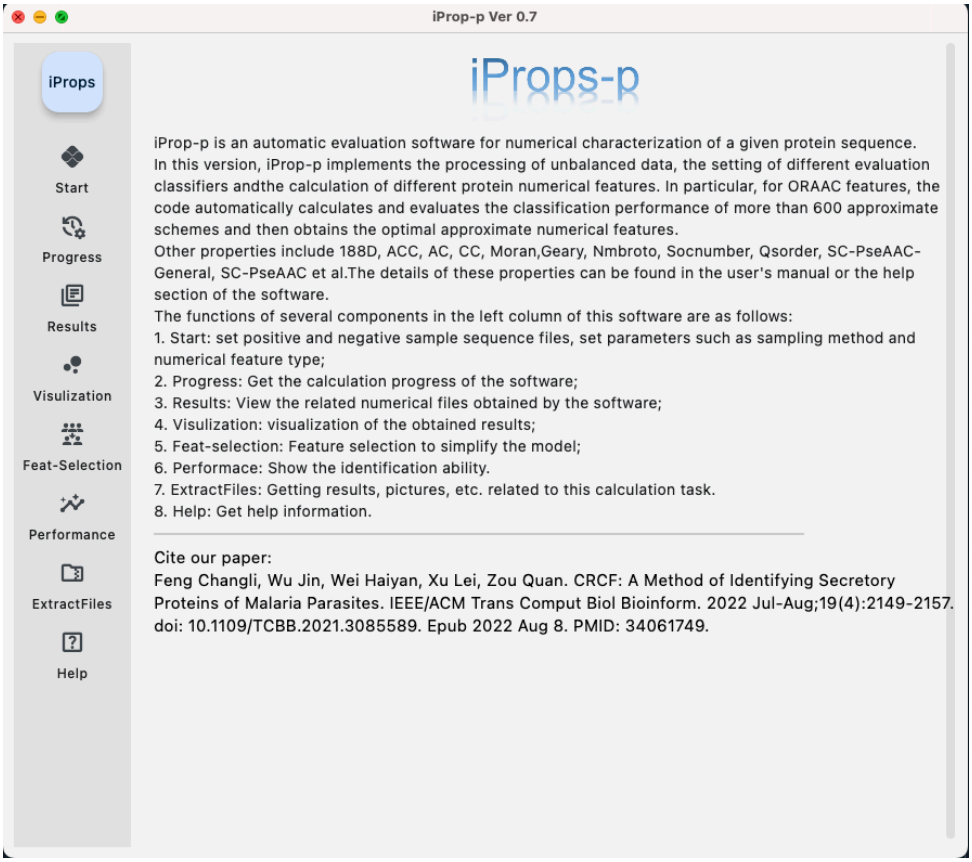


Figure 2.2: The feature evalution script(Home page)

- A. Click the ”Start” button on the sidebar .
- B. Click two file buttons to select the fasta files.(See Fig. 2.3)
- C. Numerical data resampling method selected via drop-down menu (necessary especially for severe data imbalance)
- D. Selection of classifiers to be used for cross-sectional comparisons between features(See Fig.2.4). It is worth noting that the parameters of the four classifiers are set to default values here in order to simplify comparing model complexity.

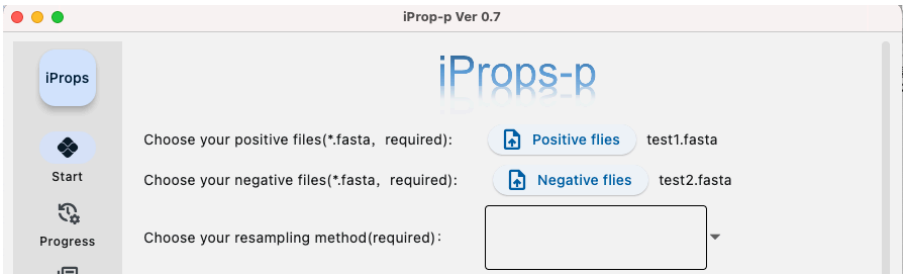


Figure 2.3: Set the fasta file

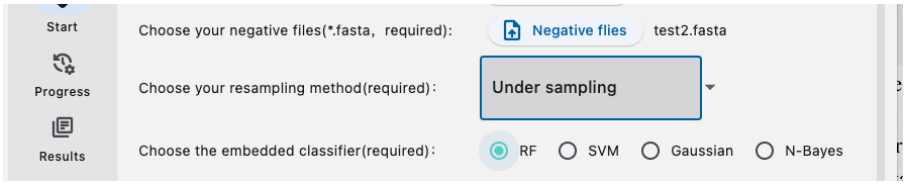


Figure 2.4: Set the resampling method and classifier

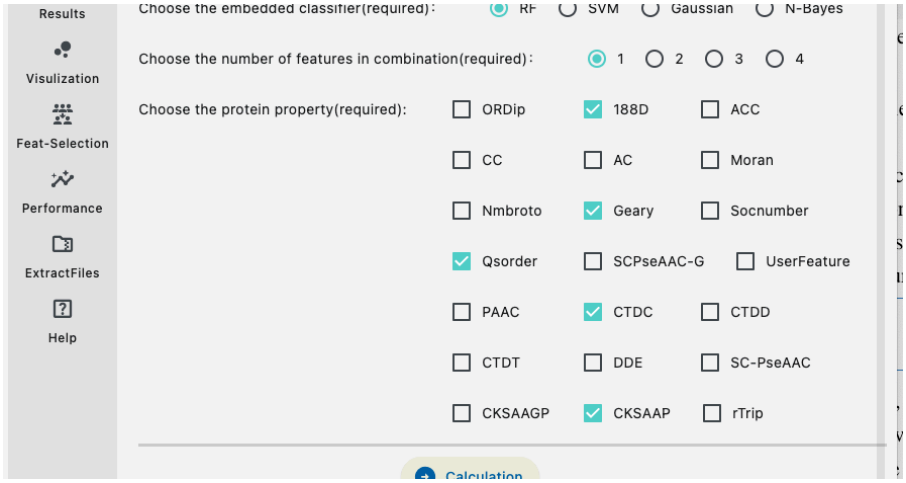


Figure 2.5: Select the features

- E. Sets the number of feature combinations. If you do not want to perform feature combinations and only evaluate certain features provided by this software, the number can be set to one. If you are evaluating two-by-two combinations of certain features, the number can be set to two. And so on.
- F Select the type of numerical features involved in the evaluation in this computational task. Use the mouse to select the corresponding feature by clicking the radio button in front of the feature. The selected feature will be presented as highlighted. (See Fig.2.5)
- G. Click on the "Calcluation" button at the bottom for calculations and evaluations.

Note *All of the above settings are required, and the absence of any of them will prevent subsequent calculations from being performed.*

Note *It is noteworthy that iProps offers a feature concerning the Position Specific Scoring Matrix (PSSM), designated as SR-PSSM. Due to the specialized nature of its computation, this feature is not integrated into the main interface of the software. It can be executed via the command line or imported within a Python (Py) file. Additionally, SR-PSSM generates numerical features in CSV format, which, when integrated into a single CSV file in the original sequence order, can be utilized as numerical features corresponding to UserFeature in the main window of iProps. For instructions on configuring the UserFeature option in the main interface and associating it with the CSV file of numerical features, please refer to "Chapter 3: Calculation Parameters Setting." Furthermore, for details on calculating the SR-PSSM feature using the command line, please consult the "Appendix A" of this document.*

Remark In addition, the software also provides hints. Missing settings will be indicated in the bottom left corner of this page, and the user can modify your settings according to the information. When the settings are complete click on the button again for calculation and evaluation. (See Fig.2.6)

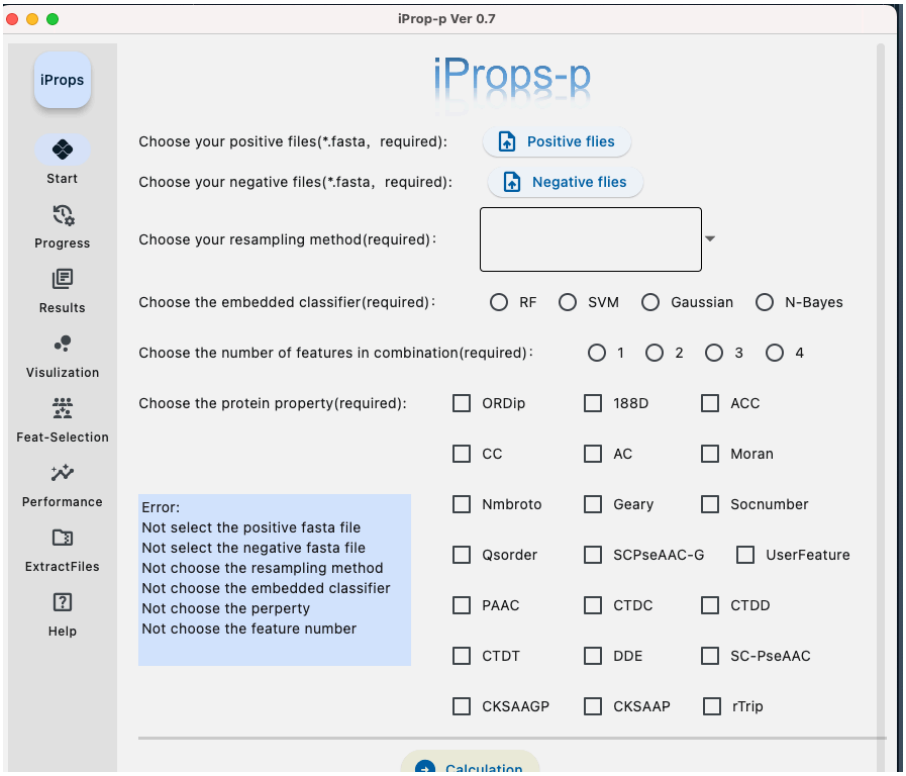


Figure 2.6: The hint information

2.3 Progress Information

When you click the ”Calculation” button in ”start” and the code starts to calculate, you can click ”Progress” in the left column to view the calculation results of the code.

As seen from Fig. 2.7, if there are no features being calculated, there is no information in the software window.

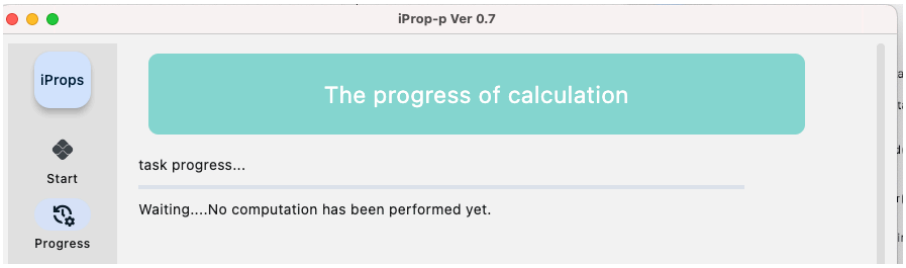


Figure 2.7: The initial status of progress

When a feature is computed, it is updated in the progress bar and the feature name is displayed in the text below.(Fig. 2.8)

2.4 Results

During the calculation of the program, click on the ”Results” button to see the csv file save path of the relevant numerical features. It can be seen in Fig. 2.9.

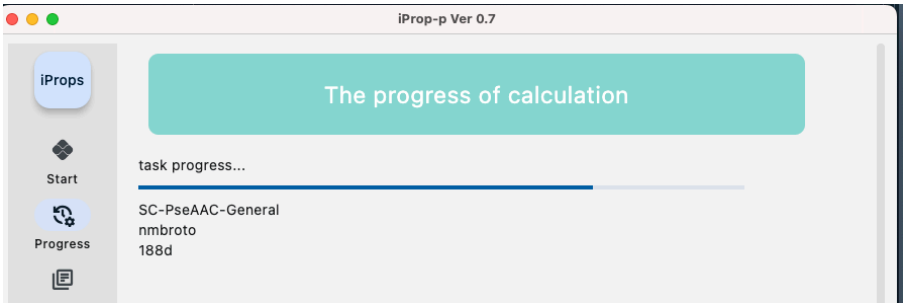


Figure 2.8: The information of calculated features

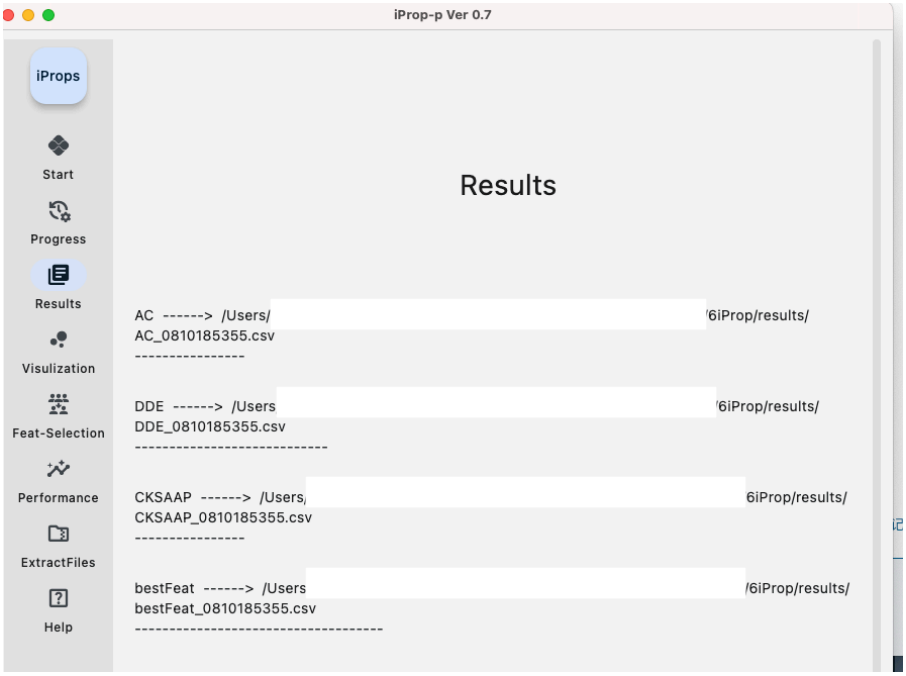


Figure 2.9: The result information

2.5 Visualization

2.5.1 Feature Performance Rank

The protein recognition ability of all features as well as combined features will be ranked. The ranking is based on the recognition ability score of the machine learning algorithm selected by the user. The feature with the highest recognition score will be selected as the optimal numerical feature to recognize the current protein. A simple example can be seen in the first subfigure of Fig. 2.10.

In addition, the plotted diagrams will be adjusted accroding to the following rules:

a) when the total number of all computed features as well as combined features exceeds 15, the top 15 with the best recognition performance will be displayed.

b) If ORDip features are selected, the code will pick the best three numerial reduced dipeptide component features from all reduced schemes as the result of this feature.

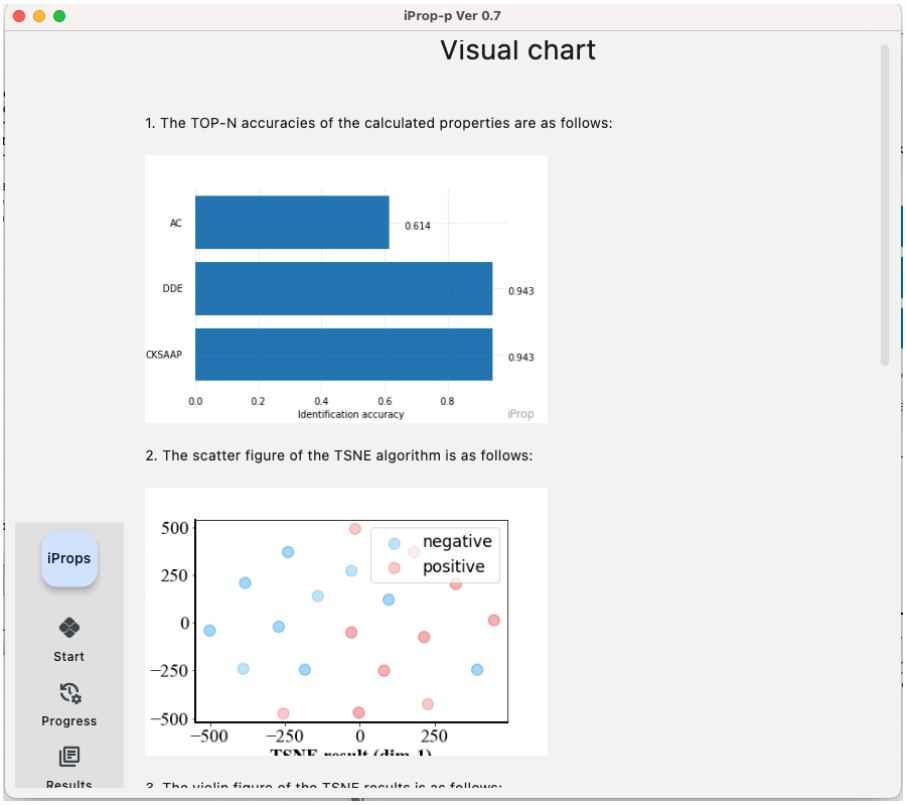


Figure 2.10: The rank of selected features

2.5.2 Scatter Plot(I)

The scatterplot will be processed using the T-SNE algorithm. Then, we labelled different protein classes by using different colors. From the scatter figure, we can visualize the classification ability of the optimal numerical features with the classifier set by the user. The example of the scatter figure is shown in Fig. 2.10.

Note *While the graphic display in the software window is incomplete, specific details can be viewed in full graphic through its high-resolution image.*

2.5.3 Compositional Analysis(I)

In order to identify the reasons why the positive and negative data in the scatterplot can have a better separation of the campus, we analyze in this section the distribution of the variables corresponding to the two axes of the scatterplot in the positive and negative samples. The relevant graphs are presented using Violin plots.

2.5.4 Scatter Plot(II)

This software also provides another scatterplot effect graph. The data of this graph is the result of UMAP algorithm processing. We use two colors to distinguish different protein classes, and the protein recognition ability of the current optimal numerical features can be seen from the separation state of the two data in the graph. An example can be seen in Fig. 2.12.

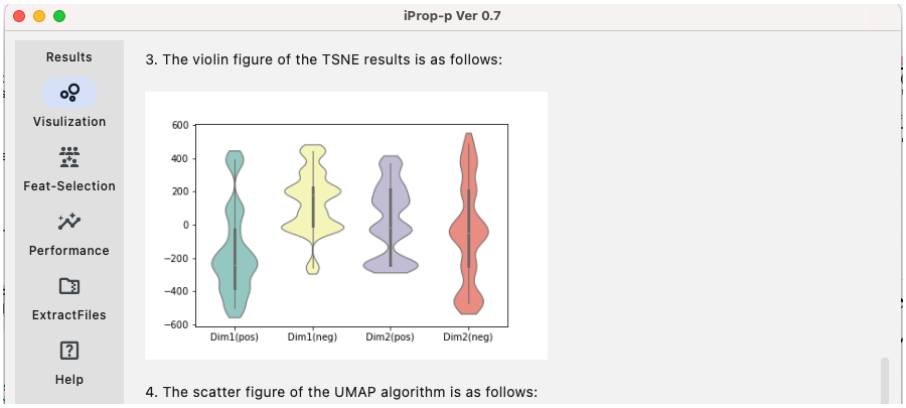


Figure 2.11: The analysis results

2.5.5 Compositional Analysis(II)

UMAP processed two-dimensional data were also analyzed. The data distribution of each dimension of data on positive and negative samples was also shown in comparison using Violin plots. In general, the greater the difference between the graphs of the data on the positive and negative samples, the greater the variability of the positive and negative samples. Also, large variability is the reason why positive and negative samples are successfully recognized by the classifier. An example can be seen in Fig. 2.12.

Figure 2.12: The second type of scatterplot and analytic graphs

Note *This software provides two different types of scatterplots as mentioned above, and the user can choose one of them according to the specific task in the specific use.*

2.6 Feature Selection

In order to simplify the protein recognition model and improve the computational efficiency of the algorithm, this algorithm provides two feature selection algorithms. These two algorithms are PCA and LDA, respectively.To use this feature you can click on the ”Feat-selection” button on the left sidebar after the computational task described in the previous section. An example of parameterization can be seen in Fig. 2.13.

a) Click the first drop-down menu to select the feature selection algorithmic.

b) Click the second drop-down menu to set the inner classified used in performance comparison.

c) Click the third drop-down menu to set the resampling strategy. This item serves the same purpose as it does on the ”Start” screen.

d) Click the forth drop-down menu to set the proportion of test set to all numerical features. For example, a value of 0.1 means that the test set is 10% and the training set is 90%.

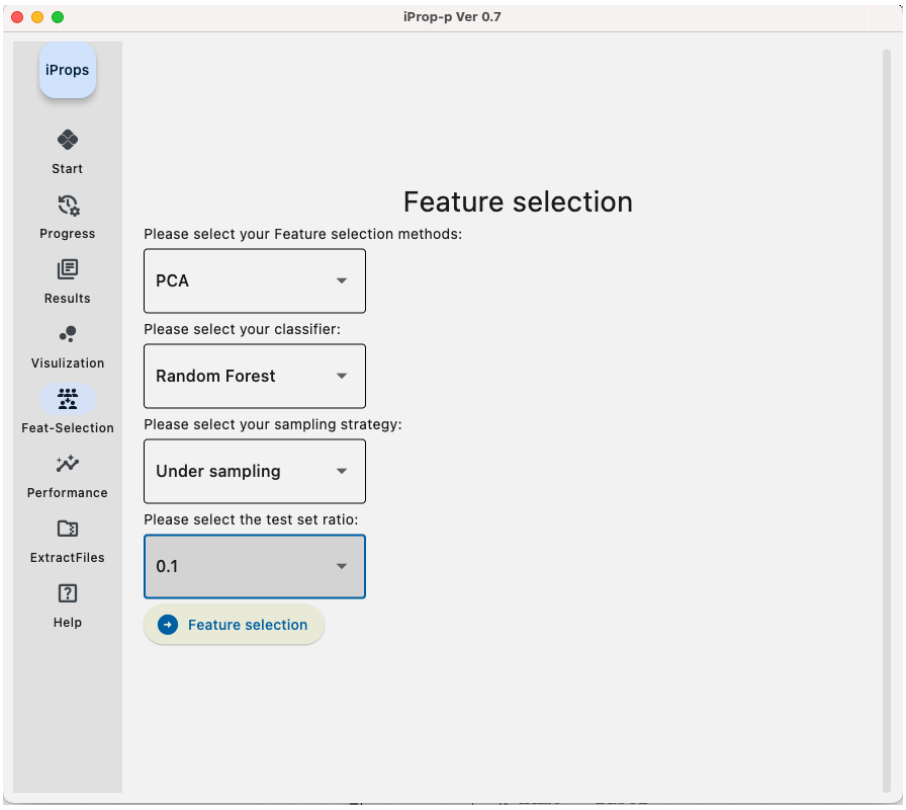


Figure 2.13: The feature selection setting

2.7 Performance

The results of the performance evaluation will be displayed at the end of the feature selection algorithm, otherwise a blank page will be displayed with a hint message. The prompt message display window is shown in Fig. 2.14(1).

When the feature selection and performance evaluation are finished, the results are displayed in the right window. There are four types of results are displayed in the window.

(1) The confusion matrix. (Fig. 2.14(2))

Figure 2.14: The hint information of performance

(2) The identification performance metrics. Four metrics including SN, SP, ACC and MCC are plotted in this graphic.(Fig. 2.14(2))

(3) The receiver operating characteristic. Using the default parameters, ten classifier were trained and tested. The output was drawn in the curve.(Fig. 2.14(3))

(4) The explained variance rate curve. This indicator is able to show the ability of the selected principal components to explain the whole information (Fig. 2.14(3)). It's worth to note that this figure is valid as far as the PCA algorithm is used.

(5) The classifier performance table. (Fig. 2.14(4))

(6) The above four figures are drawn with the LDA algorithm (Fig. 2.14(5)-(6)).

Remark If you select PCA in the Feat-Selection window, the relevant results based on the PCA algorithm will be displayed first in the performance window. Conversely, the LDA algorithm will be displayed first.

2.8 Extract Files

If you click the "ExtractFiles" icon in the sidebar, the right window will generate a new window to let user save all generated files and figures. Users can click the bottom "Save file" button to select a destination folder and set a file name. All files and graphics associated with this calculation task will be packed and saved in a user-specified directory. The user can more easily and conveniently use these files for calculations or writings.

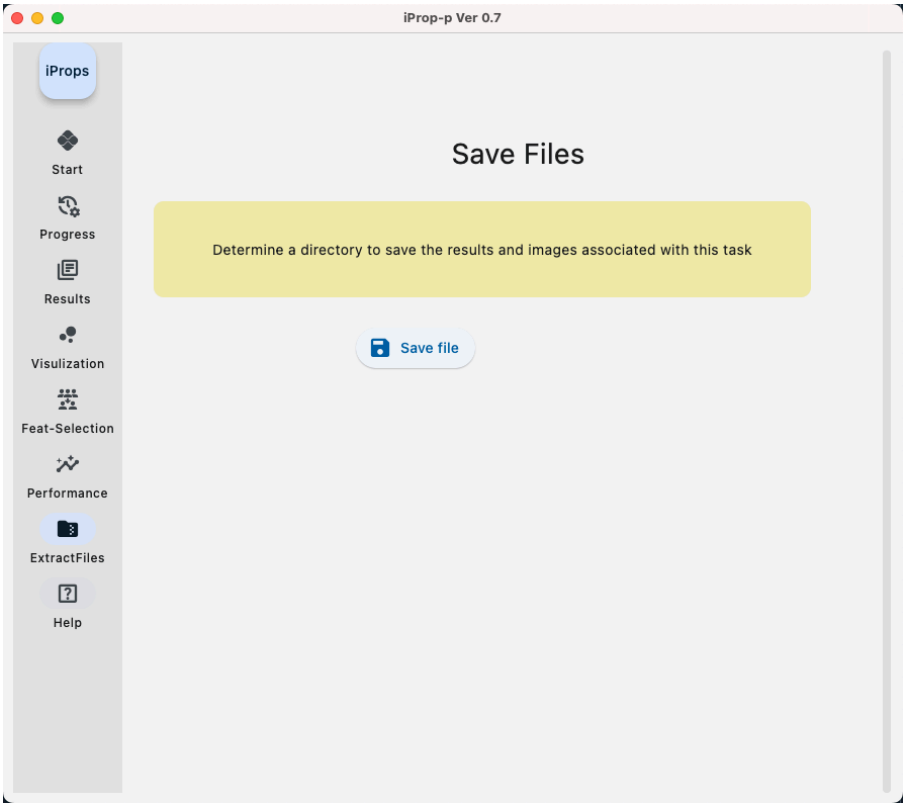


Figure 2.15: Extract Files

The files included in the zip package include:

- All figures in "Visualization" page.
- The CSV format files of all single features.
- The JSON file which includes the identification result records of all evaluated features.

- The identification ability table of ”Performance” page.

2.9 Help Information

In this section the user can find detailed instructions on how to use the software and the author’s contact information.

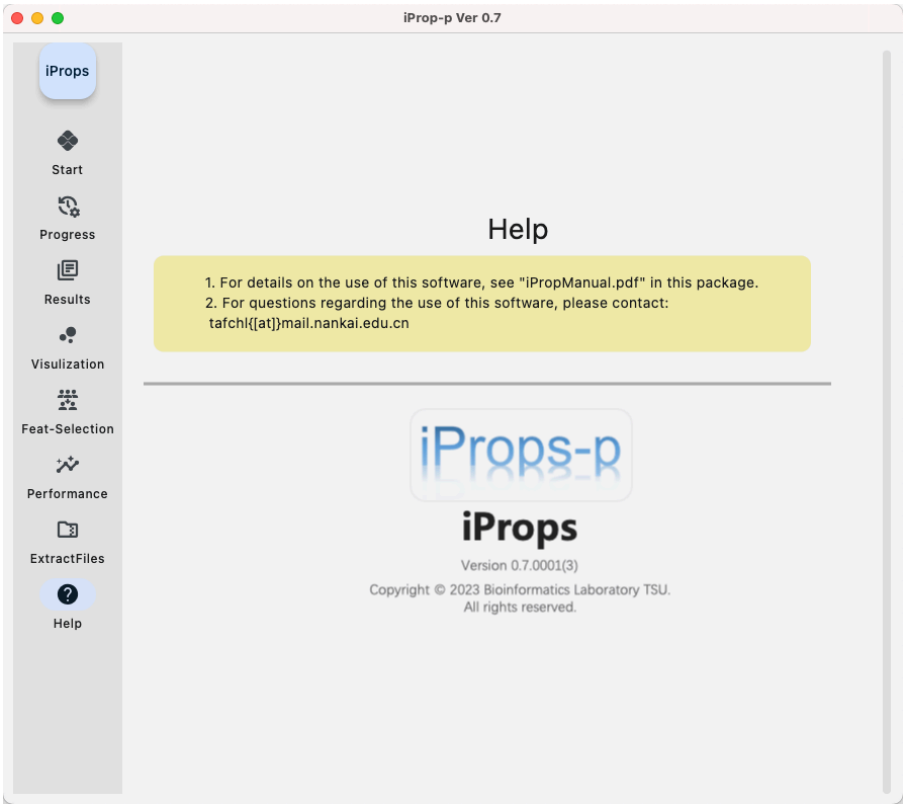


Figure 2.16: The help window

Chapter 3 Calculation Parameters Setting

Introduction

❑ *Calculation Parameters Setting*

The numerical features in Chapter 2 were computed based on default parameters, but in some tasks these default parameters do not satisfy user needs. To solve this problem, we provide the function of parameter setting for feature calculation.

3.1 Calculation Parameters Setting

If you need to specify the parameters in the numerical feature calculation formula by yourself, the user needs to click the second icon at the bottom of main window.

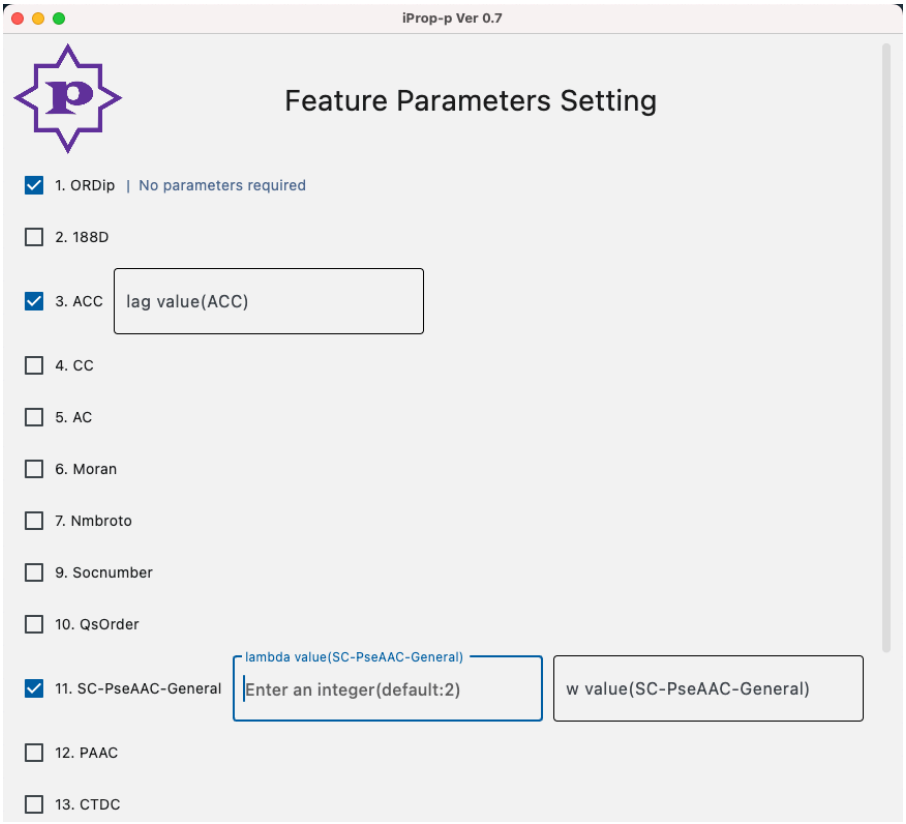


Figure 3.1: Parameter Setting Window

As seen from Fig. 3.1, this program window provides the function of setting parameters for 24 numerical features. If a numerical feature is calculated without parameters, the words ”No parameters required” will be displayed to the right of the feature. If a feature has a parameter, a text input box for setting the parameter will appear to the right of the feature when it is selected. The name of the parameter is displayed in the upper left corner of the text box. After clicking the mouse in the text input box, another text will appear showing the default value of the current parameter. It is worth noting that if a feature has no parameters to set, the feature will not appear in the Fig. 3.1.

Once the parameters have been modified, click the set parameters button at the bottom of this page to modify the corresponding parameters. It is worth noting that this software will not modify the calculation parameters for features not selected by the user in this window.

When the user wants to reset all the parameters to the default parameters, the user can reset all the parameters to the default values by not selecting any of the buttons and clicking the initalize button at the bottom of the page.

Note *Once the parameters of this software are modified, it will affect the next calculation, so it is recommended to set or reset the calculation parameters of this software before each calculation.*

Chapter 4 Auto Machine Learning and Model Interpretation

Introduction

- *Auto Machine Learning*
- *Machine learning model interpretation*

The protein recognition ability evaluation function provided in Chapter 2 is also based on the default parameters of the machine learning method. The classifiers constructed based on these default parameters are often not optimal classifiers. In order to obtain the optimal classifier, an automated machine learning approach is used to obtain the optimal protein class classifiers. Meanwhile, Many models are black box models, meaning their inner workings are hard to understand and explain. This makes it difficult to understand the decision-making process of the model and thus to trust the results of the model. To solve this problem and obtain more model details to give more clues to the protein classification results, we provide the model interpretation function.

4.1 Auto Machine Learning

In order to do get the best classifier through automatic machine learning, the user can click the third button in the main window. Then, the program window will be opened, which can be seen in Figure 4.1.

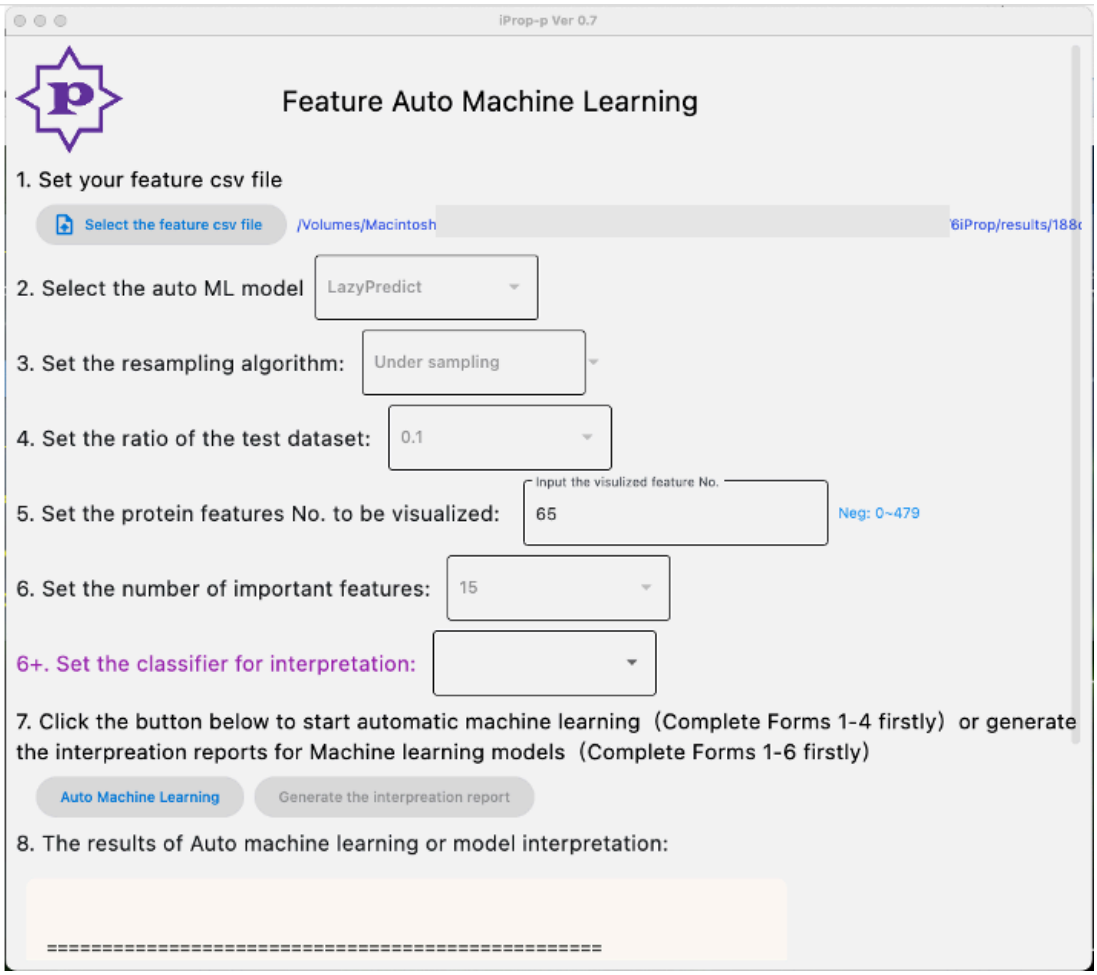


Figure 4.1: The AutoML window

In the new window, you need to follow the steps below:

- (1) Click the ”Select the feature csv file” button and select the csv format file in your computer.
- (2) Choose the automatic machine learning model you need.
- (3) Select the resampling strategy from the three options ”under sampling”, ”over sampling” or ”using orignial data”.
- (4) Select the ratio of the test set in all data from the drop-down menu.
- (5) We will visualize one of the copies during model interpretation. Therefore, the user needs to enter the number of samples you want to see in the text field. In order to allow the user to enter the correct value, the numbering range of all samples will be prompted to the user in the text box. At the same time, the right side of the text box will also be prompted for the number covered by the negative sample.
- (6) When the model is explained, the program will visualize the N most important features that affect protein identification. Users can select the value of N through the drop-down menu on the right.
- (7) Click ”Auto Machine Learning” button and waiting for the program to calculate. Because the algorithm needs to evaluate different parameter settings, each calculation task takes a long time, please be patient.
- (8) When the calculation is finished, the eighth part will be displayed in the window. This section will show information about the optimal classifier, optimal parameter settings, computation time, accuracy on the test set, etc. The Fig. 4.2 shows the results of three models.

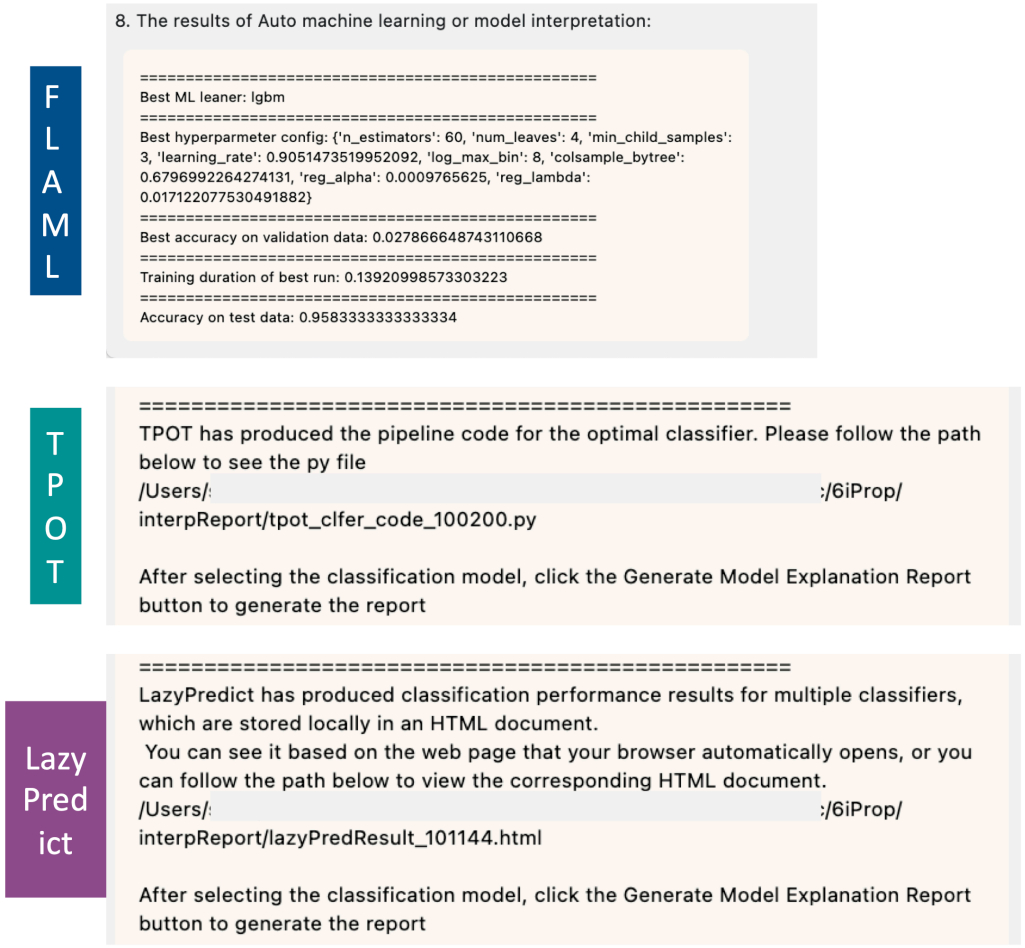
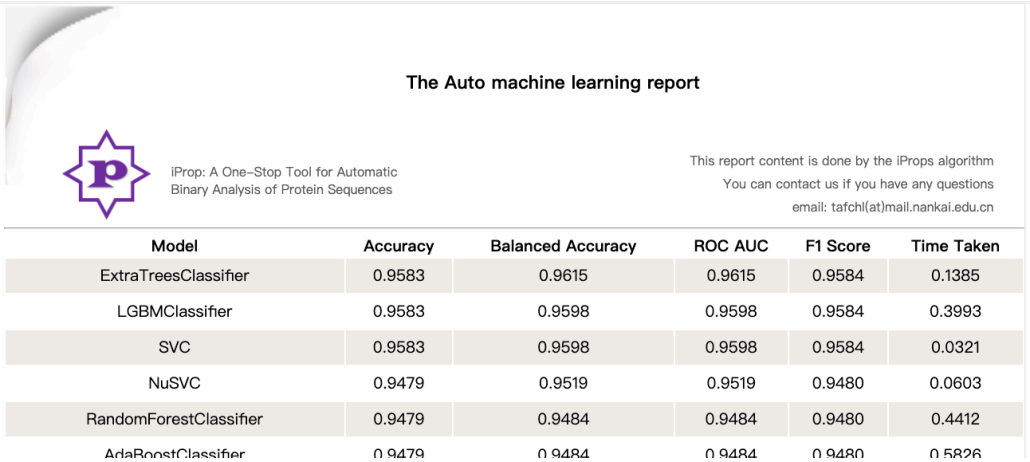


Figure 4.2: The results of the auto machine learning model

From Fig. 4.2, it is known the Flaml model gives the parameters of the computed optimal classification

model; the TPOT model generates a .py file which can build the computed optimal classification; the Lazy prediction model provides a html report which contains a table comparing the performance of multiple classifiers which can be found in Fig. 4.3.


The Auto machine learning report

iProp: A One-Stop Tool for Automatic Binary Analysis of Protein Sequences

This report content is done by the iProps algorithm
You can contact us if you have any questions
email: tafchl(at)mail.nankai.edu.cn

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
ExtraTreesClassifier	0.9583	0.9615	0.9615	0.9584	0.1385
LGBMClassifier	0.9583	0.9598	0.9598	0.9584	0.3993
SVC	0.9583	0.9598	0.9598	0.9584	0.0321
NuSVC	0.9479	0.9519	0.9519	0.9480	0.0603
RandomForestClassifier	0.9479	0.9484	0.9484	0.9480	0.4412
AdaBoostClassifier	0.9479	0.9484	0.9484	0.9480	0.5826

Figure 4.3: The report of lazy prediction model

Note *Lazy prediction can be used as the auto ml tool if you want a fast automatic machine learning speed.*

4.2 Machine learning model interpretation

In this section, we make comprehensive use of three types of model interpretation libraries. The used libraries contains ”yellowbrick”, ”shap” and ”sweetviz“. After the calculation of these libraries, the user obtain a Html page, which contains 23 pages. These pictures can explain the performance of the user-specified classifier model in protein identification from different perspectives.

The user steps are as follows:

1. Analyze the results obtained in Section 4.1 and determine the name of the optimal classifier.
2. Set the classifier for model interpretation in the window of Fig. 4.1. The optimal classifier should be the one obtained from the previous analysis step, which can be set in ”6+ Set the classifier for interpretation”(Fig. 4.1). If the optimal classifier from the previous step does not appear in ”6+...” , the user can replace others. For example, if the lazyPredict model is used, the user can select the second-ranked classifier.
3. Click the button ”Generate the interpretation report” to run the program.
4. When the program is finished, an HTML page will be generated. The system’s browser will then automatically open the document. In addition, in ”8. The results of...” of Fig.4.1 will display the path to the HTML document.
5. The following figure 4.4 gives all visualizations explained by the model interpretation.

Note *It is worth noting that 23 images in high definition (300dpi) are also saved in the same directory as the HTML. You can find a subdirectory through the sub-string ”test?” in the name of the HTML document. This folder contains the original images, which can be used in your papers directively.*

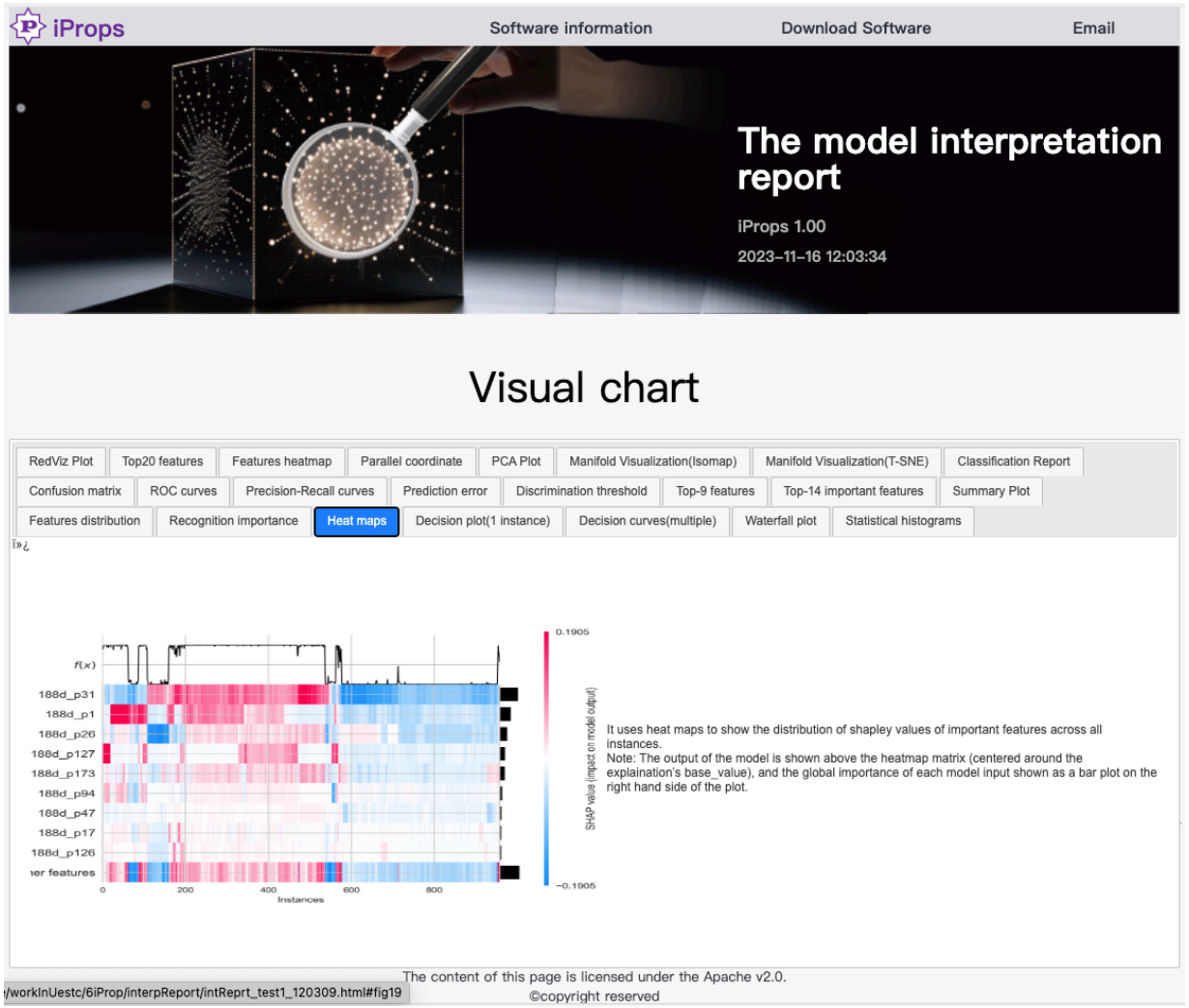


Figure 4.4: The HTML report of model interpretation

Appendix A Protein Features

This appendix covers some of the basic definition of numerical features used in this code package.

A.1 188D

This method can extract a total of 188 feature dimensions, so it is also called 188D (Li et al., 2020b). The 188D top 20 extraction dimension vectors were used to calculate the frequency of the arrangement for 20 kinds of natural amino acids (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y) (Zheng et al., 2020). Mainly refer to the following equation for calculation:

$$(v_1, v_2, \cdots, v_{20}) = \frac{N_i}{L_{seq}} \tag{A.1}$$

In Formula A.1, N_i represents the number of the i-th amino acid present in the protein sequence, and L represents the total number of amino acids contained in the sequence. The next 168 features are associated with eight physicochemical properties, all represented by descriptors C (composed of amino acids), T (transition), and D (distribution). These three properties are made up of numbers, where C is composed of 3, representing the frequency of amino acids in a particular class; T is made up of three and represents the percentage of amino acids in the two different categories; D is made up of 15, representing the chain length ratios of the first, quarter, half and last amino acids in a given category, and then expanding the calculation by another hundred times. In this way, we extracted 168-dimensional features later:

$$(C + T + D) \times 8 = 168 \tag{A.2}$$

This process encompasses the entire process for the extraction of 188 dimension features and the meaning of each feature.

Reference: I. Dubchak, I. Muchnik, S. R. Holbrook, and S. H. Kim, "PREDICTION OF PROTEIN-FOLDING CLASS USING GLOBAL DESCRIPTION OF AMINO-ACID-SEQUENCE," Proceedings of the National Academy of Sciences of the United States of America, vol. 92, no. 19, pp. 8700- 8704, Sep 1995, doi: 10.1073/pnas.92.19.8700.

A.2 Cross covariance (CC))

Given a protein sequence P (Eq. 72), the CC (12,13,31) approach measures the correlation of two different properties between two residues separated by a distance of lag along the sequence, which can be calculated by:

$$CC(u_1, u_2, lag) = \sum_{i=1}^{L-lag} (P_{u_1}(R_i) - \bar{P_{u_1}})(P_{u_2}(R_{i+lag}) - \bar{P_{u_2}})/(L - lag) \tag{A.3}$$

where u_1, u_2 are two different physicochemical indices, L is the length of the protein sequence, $P_{u_1}(R_i)$ and $P_{u_2}(R_{i+lag})$ is the numerical value of the physicochemical index $u_1(u_2)$ for the amino acid $R_i(R_{i+lag})$ at position i or $(i + lag)$. $\bar{S}_{u_1}$ and $\bar{S}_{u_2}$is the average score for amino acid u_1 and u_2 along the sequence. $\bar{S}_{u_i}(i =$

$1, 2)$ can be calculated by the following formula:

$$\bar{P}_{u_i} = \sum_{j=1}^{L} \frac{P_u(R_j)}{L} \tag{A.4}$$

In such a way, the length of the CC feature vector is $N \cdot (N-1) \cdot LAG$, where N is the number of physicochemical indices (Table 7) and LAG is the maximum of $lag(lag = 1, 2, \cdots, LAG)$. For more information of this approach, please refer to (12,13).

Reference:

1. B. Liu, F. Liu, L. Fang, X. Wang, and K. C. 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," (in eng), Bioinformatics, vol. 31, no. 8, pp. 1307-9, Apr 15 2015, doi: 10.1093/bioinformatics/btu820.
2. Dong, Q., Zhou, S. and Guan, J. (2009) A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. Bioinformatics, 25, 2655-2662.
3. Guo, Y., Yu, L., Wen, Z. and Li, M. (2008) Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. Nucleic Acids Research, 36, 3025-3030.
4. Cao, D.-S., Xu, Q.-S. and Liang, Y.-Z. (2013) propy: a tool to generate various modes of Chou's PseAAC. Bioinformatics, 29, 960-962.

A.3 Auto covariance (AC)

Suppose a protein sequence P with L amino acid residues; i.e.

$$P = P_1 P_2 P_3 \cdots P_L \tag{A.5}$$

where R_1 represents the amino acid residue at the sequence position 1, R_2 the amino acid residue at position 2 and so forth. The AC (12,13,31) approach measures the correlation of the same property between two residues separated by a distance of lag along the sequence, which can be calculated as:

$$CC(i, lag) = \sum_{j=1}^{L-lag} (P_u(R_j) - \bar{P}_u)(P_u(R_{j+lag}) - \bar{P}_u)/(L - lag) \tag{A.6}$$

where u is a physicochemical index, L is the length of the protein sequence, $P_u(R_j)$ means the numerical value of the physicochemical index u for the amino acid R_j at position j, $\bar{P}_u$ is the average value for physicochemical index u along the whole sequence:

$$\bar{P}_u = \sum_{j=1}^{L} P_u(R_j)/L \tag{A.7}$$

In such a way, the length of AC feature vector is N· LAG, where N is the number of physicochemical indices (Table 7) extracted from AAindex (32); LAG is the maximum of $lag(lag = 1, 2, \cdots, LG)$. For more information of this approach, please refer to (12,13).

Reference:

1. B. Liu, F. Liu, L. Fang, X. Wang, and K. C. 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," (in eng), Bioinformatics, vol. 31, no. 8, pp. 1307-9, Apr 15 2015, doi: 10.1093/bioinformatics/btu820.
2. Dong, Q., Zhou, S. and Guan, J. (2009) A new taxonomy-based protein fold recognition approach based

on autocross-covariance transformation. Bioinformatics, 25, 2655-2662.

3. Guo, Y., Yu, L., Wen, Z. and Li, M. (2008) Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. Nucleic Acids Research, 36, 3025-3030.
4. Cao, D.-S., Xu, Q.-S. and Liang, Y.-Z. (2013) propy: a tool to generate various modes of Chou's PseAAC. Bioinformatics, 29, 960-962.

A.4 Auto-cross covariance (ACC)

ACC (12,13,31) is a combination of AC and CC. Therefore, the length of the ACC feature vector is N*N*LAG, where N is the number of physicochemical indices (Table 7) and LAG is the maximum of $lag (lag = 1, 2, \cdots, LAG)$.

Reference: B. Liu, F. Liu, L. Fang, X. Wang, and K. C. 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," (in eng), Bioinformatics, vol. 31, no. 8, pp. 1307-9, Apr 15 2015, doi: 10.1093/bioinformatics/btu820.

A.5 Moran

The autocorrelation descriptors are defined based on the distribution of amino acid properties along the sequence (68-70). The amino acid properties used here are different types of amino acids index, which is retrieved from the AAindex Database (40) available at http://www.genome.jp/dbget/aaindex.html/. All the amino acid indices are centralized and standardized prior to the calculation:

$$p_r = \frac{p_r - \bar{p}}{\sigma} \tag{A.8}$$

where $\bar{p}$ is the average of the properties of the 20 amino acids and σ is the standard deviation of the properties of the 20 amino acids. $\bar{p}$ and σ can be calculated as follows:

$$\bar{P} = \frac{\sum_{r=1}^{20} p_r}{20} \tag{A.9}$$

$$\sigma = \sqrt{\frac{1}{20}\sum_{r=1}^{20}(p_r - \bar{p})^2} \tag{A.10}$$

The Moran autocorrelation descriptors (68,71) can thus be defined as:

$$I(d) = \frac{\frac{1}{N-d}\sum_{i=1}^{N-1}\left(P_i - \bar{P}\right)\left(P_{i+d} - \bar{P}\right)}{\frac{1}{N}\sum_{i=1}^{N}\left(P_i - \bar{P}\right)^2}, \quad d = 1, 2, 3 \ldots, \text{ nlag }, \ldots \tag{A.11}$$

where d is the lag of the autocorrelation, nlag is the maximum value of the lag, P_i and P_{i+d} are the properties of the amino acids at positions i and $i + d$, respectively. $\bar{P}$ is the average of the considered property P over the entire sequence of length N and is calculated as:

$$\bar{P}' = \frac{\sum_{i=1}^{N} P_i}{N} \tag{A.12}$$

The Moran descriptor has been successfully applied to membrane protein type prediction (68) and protein secondary structural content prediction (71).

Reference:

1. Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T. and Kanehisa, M. (2008) AAindex: amino acid index database, progress report 2008. Nucleic Acids Res, 36, D202–205.

2. Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. J Protein Chem, 19, 269–275.
3. Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. Biopolymers, 27, 451–477.
4. Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an example from an Amerindian tribal population. Am J Phys Anthropol, 129, 121–131.
5. Lin, Z. and Pan, X.M. (2001) Accurate prediction of protein secondary structural content. J Protein Chem, 20, 217–220.

A.6 Geary

The Geary autocorrelation descriptors for a protein or peptide sequence are defined as:

$$C(d) = \frac{\frac{1}{2(N-d)} \sum_{i=1}^{N-1} (P_i - P_{id})^2}{\frac{1}{N-1} \sum_{i=1}^{N} \left(P_i - \bar{P}'\right)^2}, \quad d = 1, 2, \ldots, \text{ nlag }, \tag{A.13}$$

where nlag have the same definitions as described above. The Geary descriptor has been successfully applied to population structure inferring (70).

Reference: Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an example from an Amerindian tribal population. Am J Phys Anthropol, 129, 121–131.

A.7 NMBroto (Normalized moreau-broto autocorrelation)

The Moreau-Broto autocorrelation descriptors are defined as follows:

$$AC(d) = \sum_{i=1}^{N-1} P_i \times P_{i+d}, \quad d = 1, 2, \ldots, \text{ nlag} \tag{A.14}$$

The normalized Moreau-Broto autocorrelation descriptors are thus defined as:

$$ATS(d) = \frac{AC(d)}{N-d}, \quad , d = 1, 2, \ldots, \text{nlag} \tag{A.15}$$

The NMBroto descriptor has been successfully applied to protein helix content prediction (69).

Reference: Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. Biopolymers, 27, 451–477.

A.8 SOCNumber (Sequence-Order-Coupling Number)

The d-th rank sequence-order-coupling number is defined as:

$$\tau_d = \sum_{i=1}^{N-d} (d_{i,i+d})^2, \quad d = 1, 2, \ldots, \text{ nlag }. \tag{A.16}$$

where $d_{i,i+d}$ is the entry in a given distance matrix describing a distance between the two amino acids at position i and $i + d$, nlag denotes the maximum value of the lag (default value: 30) and N is the length of a protein or peptide sequence. As distance matrix both the Schneider-Wrede physicochemical distance matrix (79) used by Kuo-Chen Chou, and the chemical distance matrix by Grantham (80) are used. Accordingly, the descriptor dimension will be nlag 2. The quasisequence-order descriptors described next also utilizes the two matrices. An illustrated example of this encoding scheme is provided in the following Figure A.1.

Note *The length of the protein must be not less than the maximum value of nlag.*

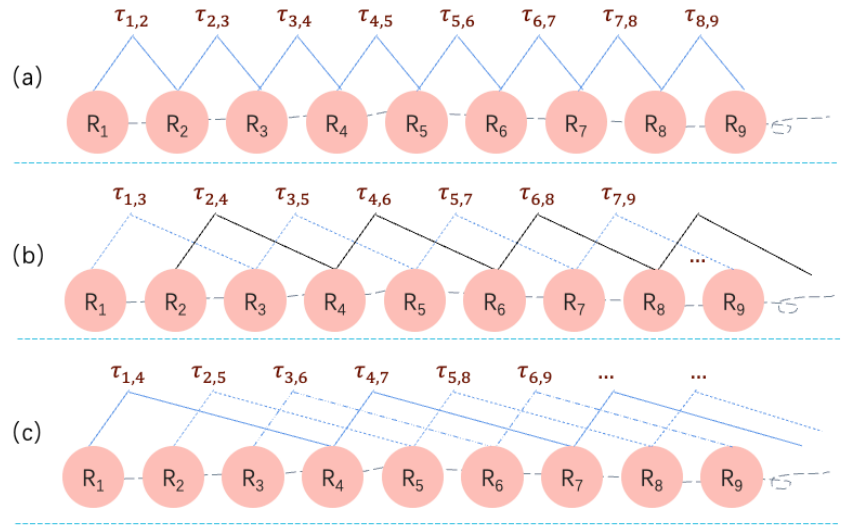


Figure A.1: The schematic diagram of SOCNumber.A schematic drawing to show (a) the 1st-rank, (b) the 2nd-rank, and (3) the 3rd-rank sequence-order-coupling mode along a protein sequence. (a) reflects the coupling mode between all the most adjacent residues, (b) shows the coupling between the adjacent plus one residue, and (c) shows the coupling between the adjacent plus two residues. This figure is adapted from (81).

Reference:

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res, 12, 2825–2830.
2. Schneider, G. and Wrede, P. (1994) The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site. Biophys J, 66, 335–344.
3. Grantham, R. (1974) Amino acid difference formula to help explain protein evolution. Science, 185, 862–864.
4. G Chou, K.C. (2000) Prediction of protein subcellular locations by incorporating quasisequence-order effect. Biochem Biophys Res Commun, 278, 477–483.

A.9 QSOrder (Quasi-sequence-order)

For each amino acid type, a quasi-sequence-order descriptor can be defined as:

$$X_r = \frac{f_r}{\sum_{r=1}^{20} f_r + w \sum_{d=1}^{n} \mathrm{t}_d}, \quad r = 1, 2, \ldots, 20 \tag{A.17}$$

where f_r is the normalized occurrence of amino acid type r and w is a weighting factor (w = 0.1), nlag and have the same definitions as described above. These are the first 20 quasi-sequenceorder descriptors. The other 30 quasi-sequence-order descriptors are defined as:

$$X_v = \frac{w\tau_d - 20}{\sum_{r=1}^{20} f_r + w \sum_{d=1}^{nlag} \tau_d}, \quad d = 21, 22, \ldots, 20 + \text{ nlag}\,. \tag{A.18}$$

The SOCNumber and QSOrder descriptors have been successfully applied to protein subcellular location prediction (79,82).

Reference:

1. Schneider, G. and Wrede, P. (1994) The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site. Biophys J, 66, 335–344.
2. Chou, K.C. and Cai, Y.D. (2004) Prediction of protein subcellular locations by GO-FunDPseAA predictor. Biochem Biophys Res Commun, 320, 1236–1239.

A.10 SC-PseAAC-General

SC-PseAAC-General approach was proposed by Chou [27, 28]. Compared with the traditional amino acid composition, it contains more sequence-length and sequence-order information. In this work, the physicochemical indices of eight amino acids were selected from the AAIndex [29]. The SC-PseAAC-General feature vector of S is defned by:

$$S = [s_1 \quad s_2 \cdots s_{20} \quad s_{20+1} \cdots s_{20+\lambda} \quad s_{20+\lambda+1} \cdots s_{20+\lambda\Lambda}]^T \tag{A.19}$$

The variable s_i is defined as follows:

$$\begin{cases} \frac{\xi_u}{\sum_{i=1}^{20}\xi_i + \frac{w}{L-1}\sum_{j=1}^{\lambda\Lambda}\sum_{i=1}^{L-1}H_{i,i+1}^{j}} & (1 \le u \le 20) \\ \frac{w\eta_{u-20}}{\sum_{i=1}^{20}\xi_i + \frac{w}{L-1}\sum_{j=1}^{\lambda\Lambda}\sum_{i=1}^{L-1}H_{i,i+1}^{j}} & (20+1 \le u \le 20+\lambda\Lambda) \end{cases} \tag{A.20}$$

where ξ(1, 2, , 20) represents the normalized frequency of the twenty natural amino acids in protein S. The parameter λ is an integer, which expresses the rank with the highest correlation along the sequence. Λ is the number of physicochemical indices, and details are shown in Table 1. ω is a weighting factor, ranging from 0 to 1; η_μ represents the sequence correlation factor between adjacent residues in the u-tier. $H_{i,i+n}^{\tau}$ is the correlation function and it can be calculated as:

$$\begin{cases} H_{i,i+n}^{\tau} = H^{\tau}(R_i)H^{\tau}(R_{i+n}) \\ \tau = 1, 2, \cdots, \Lambda; m = 1, 2, \cdots, \lambda; i = 1, 2, \cdots, L-n. \end{cases} \tag{A.21}$$

where $H^{\tau}(R_i)$ is the τ-th original physicochemical property value for the i-th (i = 1, 2, … , L) amino acid in S, and can be calculated by formula A.21.

Reference:

1. Chou, K.C. and Cai, Y.D. (2004) Prediction of protein subcellular locations by GO-FunDPseAA predictor. Biochem Biophys Res Commun, 320, 1236–1239.
2. Cao DS, Xu QS, Liang YZ (2013) propy: a tool to generate various modes of Chou's PseAAC. Bioinformatics 29:960–962
3. Kawashima S, Ogata H, Kanehisa M (1999) AAindex: amino acid index database. Nucleic Acids Res 27:368–369

A.11 SC-PseAAC

SC-PseAAC (36) is a variant of PC-PseAAC. Given a protein sequence P (Eq. 72), the SC-PseAAC feature vector of P is defined:

$$P = [p_1 \quad p_2 \cdots p_{20} \quad p_{20+1} \cdots p_{20+\lambda} \quad p_{20+\lambda+1} \cdots p_{20+2\lambda}]^T \tag{A.22}$$

In the above equation, p_i can be written as:

$$\begin{cases} \frac{f_u}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{2\lambda}\tau_j}, & (1 \le u \le 20) \\ \frac{w\tau_{u-20}}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{2\lambda}\tau_j}, & (20+1 \le u \le 20+2\lambda) \end{cases} \tag{A.23}$$

where f_i (i = 1, 2,..., 20) is the normalized occurrence frequency of the 20 native amino acids in the protein P; the parameter λ is an integer, representing the highest counted rank (or tier) of the correlation along a protein sequence; w is the weight factor ranging from 0 to 1; j-tier sequence-correlation factor that reflects the sequence-order correlation between all the most contiguous residues along a protein sequence, which is defined:

$$\begin{cases} \tau_1 = \frac{1}{L-1}\sum_{i=1}^{L-1} H_{i,i+1}^1 \\ \tau_2 = \frac{1}{L-1}\sum_{i=1}^{L-1} H_{i,i+1}^2 \\ \tau_3 = \frac{1}{L-2}\sum_{i=1}^{L-2} H_{i,i+2}^1 \\ \tau_4 = \frac{1}{L-2}\sum_{i=1}^{L-2} H_{i,i+2}^1 \\ \cdots \\ \tau_{2\lambda-1} = \frac{1}{L-\lambda}\sum_{i=1}^{L-\lambda} H_{i,i+\lambda}^1 \\ \tau_{2\lambda} = \frac{1}{L-\lambda}\sum_{i=1}^{L-\lambda} H_{i,i+\lambda}^2 \end{cases} \tag{A.24}$$

where $\lambda < L-1$. $H_{i,j}^1$ and $H_{i,j}^2$ are the hydrophobicity and hydrophilicity correlation functions given by

$$\begin{cases} H_{i,j}^1 = h^1(R_i) \cdot h^1(R_j) \\ H_{i,j}^2 = h^2(R_i) \cdot h^2(R_j) \end{cases} \tag{A.25}$$

where $h^1(R_i)$ and $h^2(R_i)$ are, respectively, the hydrophobicity and hydrophilicity values(Table 9) for the i-th (i = 1, 2, . . . ,L) amino acid in Eq. 72, and the dot ($\cdot$) means the multiplication sign. Note that before substituting the values of hydrophobicity and hydrophilicity into Eq. 85, they are all subjected to a standard conversion as described by the following equation:

$$\begin{cases} h^1\left(\mathrm{R}_i\right) = \frac{h_0^1(\mathrm{R}_i) - \sum_{k=1}^{20}\frac{h_0^1(\mathbb{R}_k)}{20}}{\sqrt{\frac{\sum_{u=1}^{20}\left[h_0^1(\mathbb{R}_u) - \sum_{k=1}^{20}\frac{h_0^1(\mathbb{R}_k)}{20}\right]^2}{20}}} \\ h^2\left(\mathrm{R}_i\right) = \frac{h_0^2(\mathrm{R}_i) - \sum_{k=1}^{20}\frac{h_0^2(\mathbb{R}_k)}{20}}{\sqrt{\frac{\sum_{u=1}^{20}\left[h_0^2(\mathbb{R}_u) - \sum_{k=1}^{20}\frac{h_0^2(\mathbb{R}_k)}{20}\right]^2}{20}}} \end{cases} \tag{A.26}$$

where we use the R_i (i = 1, 2, . . . , 20) to represent the 20 native amino acids. The symbols h_0^1 and h_0^2 represent the original hydrophobicity and hydrophilicity values of the amino acid in the brackets right after the symbols. For more information of the SC-PseAAC, please refer to (36).

Reference: Chou, K.-C. (2005) Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. Bioinformatics, 21, 10-19.

A.12 ASDC(Adaptive skip dinucleotide composition)

The adaptive skip dipeptide composition is a modified dinucleotide composition, which sufficiently considers the correlation information present not only between adjacent residues but also between intervening residues (31). For given a sequence, the feature vector for ASDC is represented by:

$$ASDC = (f_{v_1}, f_{v_2}, f_{v_3}, \cdots, f_{v_1 6}) \tag{A.27}$$

where f_{v_i} is calculated by the following formula:

$$f_{v_i} = \frac{\sum_{g=1}^{L-1} O_i^g}{\sum_{i=1}^{16} \sum_{g=1}^{L-1} O_i^g} \tag{A.28}$$

where f_{v_i} denotes the occurrence frequency of all possible dinucleotide with $\leq$ L-1 intervening nucleotides. The ASDC descriptor has been successfully applied to anti-cancer peptide (31) and cell-penetrating peptide predictions (32).

Reference:

1. Wei, L., Zhou, C., Chen, H., Song, J. and Su, R. (2018) ACPred-FL: a sequence-based predictor using effective feature representation to improve the prediction of anti-cancer peptides. Bioinformatics, 34, 4007–4016.
2. Wei, L., Tang, J. and Zou, Q. (2017) SkipCPP-Pred: an improved and promising sequence- based predictor for predicting cell-penetrating peptides. BMC Genomics, 18, 742.

A.13 PAAC (Pseudo-Amino Acid Composition)

This group of descriptors has been proposed in (83,84). Let $H_1^0(i)$, $H_2^0(i)$, $M^0(i)$for i = 1, 2, 3, … 20 be the original hydrophobicity values, the original hydrophilicity values and the original side 67 chain masses of the 20 natural amino acids, respectively. They are converted to the following quantities by a standard conversion:

$$H_1(i) = \frac{H_1^0(i) - \frac{1}{20}\sum_{i=1}^{20} H_1^0(i)}{\sqrt{\frac{\sum_{i=1}^{20}[H_1^0(i) - \frac{1}{20}\sum_{i=1}^{20} H_1^0(i)]^2}{20}}} \tag{A.29}$$

where $H_2^o(i)$ and $M^o(i)$ are normalized as $H_2(i)$ and $M(i)$ in the same manner. An example of the correlation function is provided in the following Figure A.2.

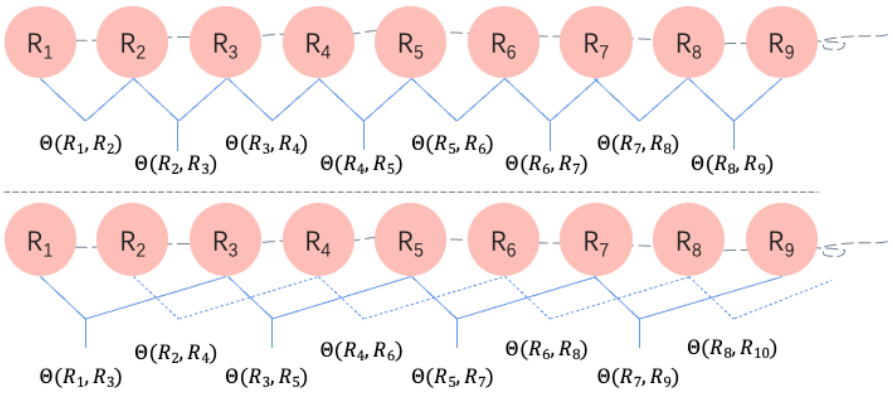


Figure A.2: A schematic illustration showing (a) the first-tier, (b) the second-tier, and (3) the thirdtier sequence order correlation mode along a protein sequence. (a) reflects the coupling mode between all the most adjacent residues, (b) shows the coupling between the adjacent plus one residue, and (c) shows the coupling between the adjacent plus two residues.

Then, a correlation function can be defined as follows:

$$\Theta(R_i, R_j) = \frac{1}{3}[H_1(R_i) - H_1(R_j)]^2 + [H_2(R_i) - H_2(R_j)]^2 + [M(R_i) - M(R_j)]^2 \tag{A.30}$$

For one amino acid property, the correlation can be defined as

$$\Theta(R_i, R_j) = [H_1(R_i) - H_1(R_j)]^2 \tag{A.31}$$

where $H(R_i)$ is the amino acid property of amino acid Ri after standardization. For a set of n amino acid

properties, it can be calculated as:

$$\Theta(R_i, R_j) = \frac{1}{n}\sum_{n=1}^{n}[H_k(R_i) - H_k(R_j)]^2 \tag{A.32}$$

where $H_k(R_i)$ is the k-th property in the amino acid property set for amino acid R_i. A set of descriptors called sequence order-correlated factors are defined as:

$$\begin{aligned}
\theta_1 &= \tfrac{1}{N-1}\textstyle\sum_{i=1}^{N-1}\Theta(R_i, R_{i+1}) \\
\theta_2 &= \tfrac{1}{N-2}\textstyle\sum_{i=1}^{N-2}\Theta(R_i, R_{i+2}) \\
\theta_3 &= \tfrac{1}{N-3}\textstyle\sum_{i=1}^{N-3}\Theta(R_i, R_{i+3}) \\
&\cdots \\
\theta_\lambda &= \tfrac{1}{N-\lambda}\textstyle\sum_{i=1}^{N-\lambda}\Theta(R_i, R_{i+\lambda})
\end{aligned} \tag{A.33}$$

where $\lambda(\lambda < N)$ is an integer parameter to be chosen. Let f_i be the normalized occurrence frequency of amino acid i in the protein sequence. Then, a set of 20 + λ descriptors called the pseudo-amino acid composition for a protein sequence can be defines as:

$$\begin{aligned}
X_c &= \frac{f_c}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{\lambda}\theta_j}, \quad (1 < c < 20), \\
X_c &= \frac{w\theta_{c-20}}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{\lambda}\theta_j}, \quad (21 < c < 20 + \lambda),
\end{aligned} \tag{A.34}$$

where w is the weighting factor for the sequence-order effect and is set $w = 0.05$ as the default value.

Reference:

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res, 12, 2825–2830.
2. Chou, K.C. (2005) Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. Bioinformatics, 21, 10–19.
3. Chou, K.C. (2001) Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins, 43, 246–255.

A.14 APAAC (Amphiphilic Pseudo-Amino Acid Composition)

Amphiphilic Pseudo-Amino Acid Composition (APAAC) was proposed in (83,84). The definition of this set of features is similar to the PAAC descriptors. Using $H_1(i)$ and $H_2(j)$ as previously defined, the hydrophobicity and hydrophilicity correlation functions are defined as:

$$\begin{aligned}
H^1_{i,j} &= H_1(i)H_1(j) \\
H^2_{i,j} &= H_2(i)H_2(j)
\end{aligned} \tag{A.35}$$

respectively. An illustrated example of the correlation functions is provided in the following Figure S27. Thus, sequence order factors can be defined as:

$$\begin{cases} \tau_1 = \frac{1}{N-1}\sum_{i=1}^{N-1} H_{i,i+1}^1 \\ \tau_2 = \frac{1}{N-1}\sum_{i=1}^{N-1} H_{i,i+1}^2 \\ \tau_3 = \frac{1}{N-2}\sum_{i=1}^{N-2} H_{i,i+2}^1 \\ \tau_4 = \frac{1}{N-2}\sum_{i=1}^{N-2} H_{i,i+2}^1 \\ \cdots \\ \tau_{2\lambda-1} = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^1 \\ \tau_{2\lambda} = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^2 \end{cases} \tag{A.36}$$

Then, Amphiphilic Pseudo-Amino Acid Composition (APAAC) is defined as:

$$\begin{cases} P_c = \frac{f_c}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{2\lambda}\tau_j}, & (1 < c < 20), \\ P_c = \frac{w\tau_u}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{2\lambda}\tau_j}, & (21 < c < 20 + 2\lambda), \end{cases} \tag{A.37}$$

where w is the weighting factor. This factor is set to w = 0.5 as described in Chou's work (84). The PAAC and APAAC have been successfully applied to protein cellular attributes prediction (84) and enzyme subfamily classes prediction (83).

Reference:

1. Chou, K.C. (2005) Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. Bioinformatics, 21, 10–19.
2. Chou, K.C. (2001) Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins, 43, 246–255.

A.15 DDE(Dipeptide Deviation from Expected Mean)

The Dipeptide Deviation from Expected Mean feature vector is constructed by computing three parameters, i.e. dipeptide composition (Dc), theoretical mean (Tm), and theoretical variance (Tv). The above three parameters and the DDE are computed as follows. Dc(r,s), the dipeptide composition measure for the dipeptide 'rs', is given as

$$D_c(r,s) = \frac{N_{rs}}{N-1}, \qquad r,s \in \{A, C, D, \cdots, Y\} \tag{A.38}$$

where N_{rs} is the number of dipeptides represented by amino acid types r and s and N is the length of the protein or peptide. $T_m(r,s)$, the theoretical mean, is given by:

$$T_m(r,s) = \frac{C_r}{C_N} \times \frac{C_s}{C_N} \tag{A.39}$$

where C_r is the number of codons that code for the first amino acid and C_s is the number of codons that code for the second amino acid in the given dipeptide 'rs'. C_N is the total number of possible codons, excluding the three stop codons (i.e., 61). $T_v(r,s)$, the theoretical variance of the dipeptide 'rs', is given by:

$$T_v(r,s) = \frac{T_m(r,s)(1 - T_m(r,s))}{N-1} \tag{A.40}$$

Finally, $DDE(r,s)$ can be calculated by the following formula:

$$DDE(r,s) = \frac{D_c(r,s) - T_m(r,s)}{\sqrt{T_v(r,s)}} \tag{A.41}$$

Reference:

1. Saravanan, V. and Gautham, N. (2015) Harnessing Computational Biology for Exact Linear B-Cell Epitope Prediction: A Novel Amino Acid Composition-Based Feature Descriptor. OMICS, 19, 648–658.
2. White, G. and Seffens, W. (1998) Using a neural network to backtranslate amino acid sequences. Electronic Journal of Biotechnology. Vol 1 Num 3, 1.

A.16 ORDip

We organized, analyzed and de-deplicated the published group schemes. Finally, we obtained 568 group schemes. The details of these schemes can be found in Appendix B. The feature are calculated successively using two steps. The first step is to iterate through all 568 schemes and evulate the identification performance of each scheme. For a given group scheme, it will be used to recode the protein sequence. Then, the frequency of all amino acid pairs seperated by k residues will counted. The obtained value is divided by the number of all this kind of pairs to from the feature. Therefore, the numerical feature vector of a given sequence can be expressed as follows:

$$(\frac{N_{g_1g_1}}{N^0_{total}}, \frac{N_{g_1g_2}}{N^0_{total}}, \cdots, \frac{N_{g_ng_n}}{N^0_{total}}, \frac{N_{g_1{*}g_1}}{N^1_{total}}, \frac{N_{g_1{*}g_2}}{N^1_{total}}, \cdots, \frac{N_{g_n{*}g_n}}{N^1_{total}}, \frac{N_{g_1{**}g_1}}{N^2_{total}}, \frac{N_{g_1{**}g_2}}{N^2_{total}}, \cdots, \frac{N_{g_n{**}g_n}}{N^2_{total}}) \tag{A.42}$$

where N^k_{total} denotes the number of all amino acid pairs when $k = 0, 1, 2$. $g_1{**}g_1$ denotes a reduced amino acid pair seperated by any two reduced amino acids.

Once the features of all proteins were computed, a built-in forest classifier is used to evaluate the identification performance of the current group scheme. Further, we used the mean accuracy on the test data as a metric to access the numerical feature.

In the second step, when all schemes have been evaluated, the codes will rank all schemes based on the mean accuracy value. The user can set a integer value N, the schemes with the top-N accuracy will be extracted as the optimal group/reduce schemes. Meanwhile, the "csv" format files will be generated according to the obtianed optimal group/reduce schemes. Therefore, this feature can produce from one to 568 different numerical features.

Note *The default value of N is ten. The user can modify this parameter by the second function of the main window.*

Reference: H. Y. Zhang, Q. Xi, S. H. Huang, L. Zheng, W. Yang, and Y. C. Zuo, "iSP-RAAC: Identify secretory proteins of malaria parasite using reduced amino acid composition," Comb. Chem. High Throughput Screen, vol. 23, no. 6, pp. 536–545, 2020.

A.17 RTC(Reduced Tripeptide component)

For a group scheme set by the user, the protein sequence will be recoded fristly. This feature wil calculate the frequency of tripeptides spaced by $k(k = 0, 1, 2)$ amino acids in the encoded sequence. Only six types of tripeptides are supported in the current version. These six types are as follows:

Assuming the current group scheme divided all 20 amino acids into m groups, the relevant component of this featuer can be expressed as follows:

$$(\frac{N_{g_1g_1g_1}}{N_{trip}}, \frac{N_{g_1g_1g_2}}{N_{trip}}, \cdots, \frac{N_{g_1g_1g_m}}{N_{trip}}, \frac{N_{g_1g_2g_1}}{N_{trip}}, \frac{N_{g_1g_2g_2}}{N_{trip}}, \cdots, \frac{N_{g_1g_2g_m}}{N_{trip}}, \cdots, \frac{N_{g_mg_mg_1}}{N_{trip}}, \frac{N_{g_mg_mg_2}}{N_{trip}}, \cdots, \frac{N_{g_mg_mg_m}}{N_{trip}}) \tag{A.43}$$

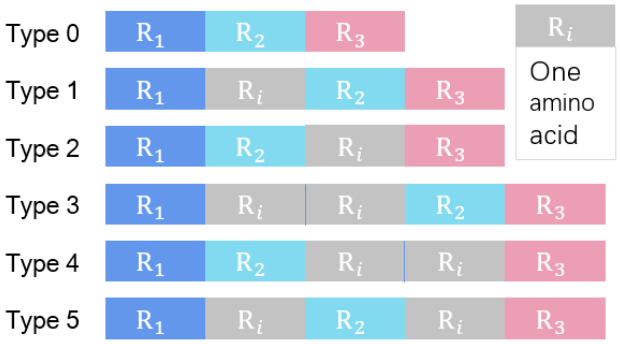


Figure A.3: The schematic illustration of RTC

Where N_{trip} denotes the number of all tripeptides in the recoded sequence. Once the type of tripeptide is determined(Type=k, k$\in \{0,1,2,3,4,5\}$) , the tripeptide "$g_i g_j g_s$" will be searched as represented in Fig. A.3.

Note *The used group scheme in this feature is as the same as in Appendix B.*

A.18 CTDC-E

This feature is an extended version of the CTDC feature. In the original CTDC features, all amino acids are divided into three groups: polar, neutral and hydrophobic (Table S6). The precentage of these three types are used to build the elements of the CTDC feature. Users can get the feature value by the following equation

$$C(r) = \frac{N(r)}{N}, r \in \{\text{ploar, neutral, hydrophobic}\} \tag{A.44}$$

where N is the length of the sequence. $N(r)$ denotes the frequency of amino acid r in the encoded sequence.

CTDC used 13 group schemes to recode the sequence successively. Thus, these schemes are as follows:

```
hydrophobicity_PRAM900101(redSchm_417), hydrophobicity_ARGP820101(redSchm_560),
hydrophobicity_ZIMJ680101(redSchm_561), hydrophobicity_PONP930101(redSchm_563),
hydrophobicity_CASG920101(redSchm_564), hydrophobicity_ENGD860101(redSchm_565),
hydrophobicity_FASG890101(redSchm_566), normwaalsvolume(redSchm_567),
polarity(redSchm_419), polarizability(redSchm_420), charge(redSchm_421),
secondarystruct(redSchm_422), solventaccess(redSchm_423)
```

In our codes, the number of group schemes is added to 46. The added 33 group schemes are as follows:

```
redSchm_2, redSchm_33, redSchm_44, redSchm_80, redSchm_105, redSchm_126,
redSchm_149, redSchm_166, redSchm_182, redSchm_183, redSchm_189, redSchm_192,
redSchm_194, redSchm_196, redSchm_199, redSchm_216, redSchm_234, redSchm_248,
redSchm_262, redSchm_292, redSchm_317, redSchm_382, redSchm_400, redSchm_426,
redSchm_442, redSchm_460, redSchm_478, redSchm_498, redSchm_499, redSchm_503,
redSchm_520, redSchm_529, redSchm_546
```

Remark The details of the above "redSchm_*" can be found in Appendix B. For example, "redSchm_2" corresponding to "(2) ACGILMPSTV#DEHKNQR#FWY".

The feature vector can be expressed as:

$$(C_{g1}^1, C_{g2}^1, C_{g3}^1, \cdots, C_{g1}^i, C_{g2}^i, C_{g3}^i, \cdots, C_{g1}^{54}, C_{g2}^{54}, C_{g3}^{54}), i = 1, 2, 3, \cdots, 54 \tag{A.45}$$

Therefore, the dimension of CTDT-E feature is $54 \times 3 = 162$.

Reference: Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ* , Chou KC* , Song J* . *iFeature*: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics, 2018, doi: 10.1093/bioinformatics/bty140.

A.19 CTDT-E

This feature is an extended version of CTDT. Comparing with the CTDT featue, we increased the number of group schemes from 13 to 54. The added schemes are the same as expressed in Feature "CTDC-E".

This feature consists of three types of transitions(72,73): (I) transitions between polar and neutral groups; (ii) transitions between neutral and hydrophobic groups; and (iii) transitions between hydrophobic and polar. In this regard, a transition between a polar group and a neutral group refers to a polar amino acid immediately followed by a neutral amino acid or a neutral amino acid immediately followed by a polar amino acid. The remaining two transitions are defined similarly to the first. The descriptor is defined as:

$$T(r,s) = \frac{N(r,s) + N(s,r)}{N-1}, \quad r,s \in \{\text{(polar,neutral),(neutral,hydrophobic),(hydrophobic,polar)}\} \tag{A.46}$$

where $N(r,s)$ and $N(s,r)$ are the frequency of dipeptides with "rs" or "sr" forms in the encoded sequence, N is the length of the sequence.

The feature vector of this feature can be expressed as:

$$(T^1_{p,n}, T^1_{n,h}, T^1_{h,p}, \cdots, T^i_{p,n}, T^i_{n,h}, T^i_{h,p}, \cdots, T^{54}_{p,n}, T^{54}_{n,h}, T^{54}_{h,p}), i = 1,2,3,\cdots,54 \tag{A.47}$$

where $T^i_{p,n}$ denotes the value of $T(p,n)$ corresponding to the group scheme i.

Reference: Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ* , Chou KC* , Song J* . *iFeature*: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics, 2018, doi: 10.1093/bioinformatics/bty140.

A.20 CTDD-E

This feature is an extended version of the CTDD feature. As in CTDC-E and CTDT-E, we made the same expansion in the group schemes.

In this feature, we extended the number of group scheme from 13 to 54. Each scheme divided 20 amino acids into three groups. For each scheme, we calculated five descriptors. That is, the position at which the first residue of each grouping occurs, and the positions at which 25%, 50%, 75%, and 100% occur. The above five descriptors are integers and will be converted to a fraction by dividing by the length of the sequence. This feature used the same group schemes as the Feature "CTDC-E".Therefore, this feature contains 54×5=220 dimensions.

Reference: Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ* , Chou KC* , Song J* . *iFeature*: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics, 2018, doi: 10.1093/bioinformatics/bty140.

A.21 CKSAAP(Composition of k-spaced amino acid pairs)

This feature calculates the frequency of amino acid pairs seperated by k residues. k can take values from zero to five. For all 20 amino acids, the number of all amino acid pairs is 400. The value of each amino acid pair is the number of this pair in the sequence divided by the total number of all pairs in the protein sequence.

Using $k = 1$ as an example, the feature can be expressed in the following form:

$$(\frac{N_{A*A}}{N_{total}}, \frac{N_{A*C}}{N_{total}}, \frac{N_{A*D}}{N_{total}}, \cdots, \frac{N_{Y*Y}}{N_{total}})_{400} \tag{A.48}$$

where $A * A$ denotes an amino acid pair seperated by arbitrary an amino acid, N_{total} is the total number of the amino acid pair in the sequence.

Reference: Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ* , Chou KC* , Song J* . *iFeature*: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics, 2018, doi: 10.1093/bioinformatics/bty140.

A.22 CKSAAGP-E (Composition of k-spaced amino acid group pairs Extension)

In the original CKSAAGP feature, authors used only one grouped scheme: GAVLMI#FYW#KRH#DE#STCPNQ. The group scheme divides amino acids into five groups: g_1, g_2, g_3, g_4, g_5. This feature calculate the frequency of amino acid pairs seperated by any k residues. Taking $k = 2$ for example, there are 25 zero-spaced group pairs: g_1**g_1, g_1**g_2, g_1**g_3,$\cdots$,g_5**g_5, in which g_1**g_1 denotes the reduced amino acid pair g_1g_1 sperated by any two amino acids. The value of this featue can be calculated as follows:

$$(\frac{N_{g_1**g_1}}{N_{total}}, \frac{N_{g_1**g_2}}{N_{total}}, \frac{N_{g_1**g_3}}{N_{total}}, \cdots, \frac{N_{g_5**g_5}}{N_{total}})_{25} \tag{A.49}$$

In our extended version, the number of the group scheme was increased to 568. The added 567 schemes can be found in Appendix B. The user can set the serical number of the scheme used in calculation of this feature through "paraSetting.py" script. The value ranges from 1 to 568.

Note *The number of amino acid group is different for all 568 schemes. The number varies from two to 19. Therefore, the feature dimension of this feature will vary according to different schemes.*

Reference: Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ* , Chou KC* , Song J* . *iFeature*: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics, 2018, doi: 10.1093/bioinformatics/bty140.

A.23 EAAC-E

The original EAAC divided 20 amino acids into five groups: Aliphatic (g1: GAVLMI), Aromatic (g2: FYW), Positively Charged (g3: KRH), Negatively Charged (g4: DE), and Uncharged (g5: STCPNQ). The elements of EAAC is the frequency of these five groups. The numerical feature vector can be calculated as follows:

$$\begin{aligned} f(g) &= \tfrac{N(g)}{N}, g \in \{g_1, g_2, g_3, g_4, g_5\} \\ N(g_t) &= \textstyle\sum N(t), t \in g \end{aligned} \tag{A.50}$$

In our extended version, all 568 group scheme are used to compute the EAAC-E features. Each group scheme produces a feature vector similiar to the variable of Eq. A.50. Then, the features of each scenario were aggregated to obtain a large vector, whose dimension is 5264.

Reference: Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ* , Chou KC* , Song J* . *iFeature*: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics, 2018, doi: 10.1093/bioinformatics/bty140.

A.24 User given feature

This code calculates a limited number of features and may not be able to calculate some features that the user finds more important. In order to enhance the extensibility of this code, this feature supports user-supplied numerical feature files to participate in the computation, comparison and evaluation.

The user needs to ensure three things before using this feature:

- the number of positive and negative samples should be the same as the number of features in the numerical feature file.
- in order to keep the code as simple and pure as possible, the code only supports class as a column name for the category column in the numerical feature file.
- Before performing the evaluation in Chapter 2, you need to set the value of ”user's Feature” using the ”paraSetting.py” script.(See Fig. A.4)

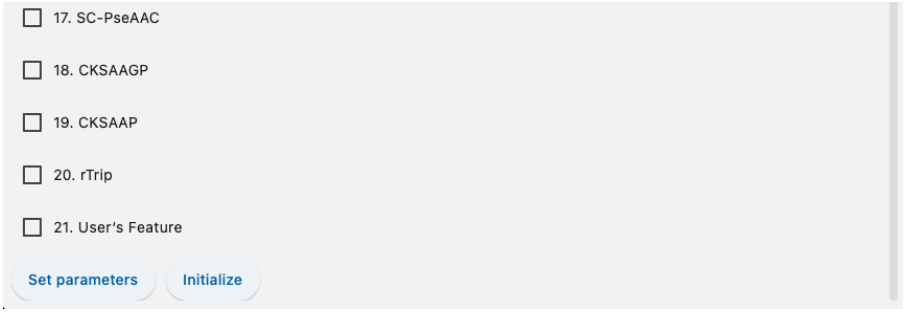


Figure A.4: The figure of setting user given feature

After comfirming the above details, the user can check the ”UserFeature” box in the interface of Chapter 2. The program will evaluate this feature in the subsequent calculations.

A.25 SR-PSSM

To effectively harness the evolutionary information within amino acid residue sequences, we propose a Position Specific Scoring Matrix (PSSM) feature descriptor that incorporates methods of sequence segmentation and amino acid reduction. The initial step involves data preprocessing, where each amino acid residue sequence, represented as Sp, is utilized to search for and align homologous sequences from the NCBI's Non-Redundant (NR) database using PSI-BLAST with standard parameter settings[38]. Through the analysis of this homologous data, an initial PSSM is constructed. Subsequently, the Mish function is implemented to scale the matrix's elements uniformly. The Mish function is expressed as:

$$f(x) = xtanh(\zeta(x)) \quad \zeta(x) = ln(1 + e^x). \tag{A.51}$$

Here, x signifies the original PSSM value.

To obtain more granular local information, the amino acid residue sequence is evenly divided into N non-overlapping segments, represented as $S_p = [seg_1, seg_2, \cdots, seg_N]$. To reduce the impact of segmentation on feature computation, each segment is uniformly extended in both directions from the original sequence by a predetermined length. This length is calculated based on the overlap ratio r and the length of each individual segment, $len(seq_i)$.

$$
\begin{aligned}
seg_i^p &= [AA_{-n}, AA_{-n+1}, \cdots, AA_{-1}], n = len(seq_i) * r, AA_j seg_{(i-1,} \\
seg_i^s &= [AA_0, AA_1, \cdots, AA_{n-1}], AA_j seg_{i+1} \\
seg'_{i+1} &= [seg_i^p, seg_{i+1}, seg_i^s]
\end{aligned} \tag{A.52}
$$

Here, $AA_{-n}, AA_{-n+1}, \cdots, AA_{-1}$ represent the n-th to the first elements in reverse order of seg_{i-1}, and $AA_0, AA_1, \cdots, AA_r$ represent the 1st to n-th elements of segi+1.

Based on the aforementioned settings, the new segments can be represented as $[seg'_1, seg'_2, seg'_3, \cdots, seg'_N]$, where each pair of adjacent segments has overlapping amino acid residues. On average, without considering the feature context, the overlap ratio is r times the length of each segment.

Next, the calculation of the feature descriptor is performed. Let $P = [p_1, p_2, p_3, \cdots, p_{20}]$ represent the PSSM of the queried sequence, where $p_j = [p_{1,j}, p_{2,j}, \cdots, p_{L,j}]$, and L is the length of the queried sequence. The feature descriptor for the PSSM matrix corresponding to each segment includes three types of sub-features.

The first type of sub-feature, f_{s1}, can be expressed as:

$$
f_j = \frac{1}{L}\sum_{i=1}^{L} p_{i,j}, \quad (j = 1, 2, 3, \cdots, 20), \quad f_{s1} = [f_1, f_2, \cdots, f_{20}] \tag{A.53}
$$

Here, $f_j (1 \leq j \leq 20)$ is the composition of the j-th type of amino acid and represents the average score of the amino acid in the protein sequence evolving into the j-th type of amino acid during the evolutionary process. The second type of sub-feature, f_{s2}, can be expressed as:

$$
s_k = \frac{1}{L}\sum_{j=1}^{L}(p_{i,j} - \bar{p_j})^2, \quad k = 1, 2, 3, \cdots, 20, \quad f_{s2} = [s_1, s_2, , s_{20}] \tag{A.54}
$$

Here, p_j denotes the mean of p_j, and $s_k (1 \leq k \leq 20)$ is the variance of the k-th type of amino acid, representing the degree of clustering of the average score of the amino acid in the protein sequence evolving into the k-th type of amino acid during the evolutionary process.

The third type of sub-feature, f_{s3}, can be expressed as:

$$
\begin{aligned}
d_{s,t} &= \frac{1}{L}\sum_{i=1}^{L-1}\frac{(p_{i,s} - p_{i+1,t})^2}{2}, \quad s, t = 1, 2, 3, \cdots, 20, \\
f_{s3} &= [d_{1,1}, d_{1,2}, \cdots, d_{1,20}, \cdots, d_{20,1}, d_{20,2}, \cdots, d_{20,20}]
\end{aligned} \tag{A.55}
$$

This represents the pairwise difference between adjacent amino acids of different types in the sequence, providing a measure of the local sequence variation. Each $d_{s,t}$ is calculated for all possible pairs of amino acid types s and t, and included in the feature vector f_{s3}.

After extracting the three types of sub-features discussed earlier, a reduced dictionary is then applied for the subsequent classification of these features. For instance, with the Hydrophobicity_PRAM900101 property [19] , the 20 native amino acids are categorized into three groups based on their physicochemical properties: RKEDQN are categorized as Polar, GASTPHY as Neutral, and CLVIMFW as Hydrophobic. We represent these three classes using the symbols R', N', and H', respectively.

As a result, the three types of sub-features are also correspondingly modified:

$$
\begin{aligned}
f'_{s1} &= [f_{R'}, f_{N'}, f_{H'}], f_{X'} = mean(f_x x X') \\
f'_{s2} &= [s_{R'}, s_{N'}, s_{H'}], s_{X'} = mean(s_x x X') \\
f_s 3' &= [d'_{R',R'}, d'_{R',N'}, d'_{R',H'}, \cdots, d'_{H',R'}, d'_{H',N'}, d'_{H',H'}]
\end{aligned} \tag{A.56}
$$

where $d_{X',Y'} = mean(d_{s,t} | s \in X', t \in Y')$. Therefore, for each segment seg'_i, its SR-PSSM feature can be

represented as: $f_{seg_i} = [f_s1', f_s2', f_s3']$ and the dimensionality of its feature is $2M + M^2$, where M is the total number of reduced amino acids.

Finally, the total SR-PSSM feature of the aforementioned queried sequence can be represented as:

$$[f_{seg'_1}, f_{seg'_2}, f_{seg'_3}, , f_{seg'_N}], \tag{A.57}$$

The total dimensionality of this feature is $(2M + M^2) \times N$, where N is the total number of segments.

It is noteworthy that the SC-PSSM feature, which necessitates computation across a series of PSSM files, exhibits a higher degree of complexity. Consequently, within iProps, the computation of this feature operates independently of the iProps graphical user interface (GUI). The output numerical feature file, formatted as CSV, can be utilized as a UserFeature-type feature file.

Note *The fundamental syntax for employing the SR-PSSM feature within a command-line environment is as follows:*

```
python SrPSSM.py -p infile.pssm -a DE#SF -o outfile.csv
```

Here, the -p parameter designates the path to the PSSM file, -a configures the string representation of the reduced dictionary (with relevant strings found in Appendix B), and -o specifies the path for the output CSV file.

Note *The fundamental syntax for employing the SR-PSSM feature within a .py file is as follows:*

```
from SrPSSM import SrPssm

...

p_pssmPth = 'xxx.pssm'
s_alpha = 'DE#SF'
d_srPssmFeat = SrPssm(p_pssmPth, s_alpha)
```

Appendix B Group Schemes

(1) ACFGILMPSTVWY#DEHKNQR
(2) ACGILMPSTV#DEHKNQR#FWY
(3) AGPST#CILMV#DEHKNQR#FWY
(4) AGPST#CILMV#DENQ#FWY#HKR
(5) AGST#CHP#DENQ#FWY#ILMV#KR
(6) AG#CILMV#DENQ#FWY#KR#ST
(7) DN#EQ#FY#ILMV#KR#ST
(8) FY#ILMV#KR
(9) AGPST#DEHKNQR#FILMVY
(10) AGPST#DEKNQR#FILMVY
(11) ADGNST#EKQR#FILMVY
(12) ADGNST#EKQR#FY#ILMV
(13) AGST#DN#EKQR#FY#ILMV
(14) AGST#DN#EQ#FY#ILMV#KR
(15) AST#DN#EQ#FY#ILMV#KR
(16) AST#DN#EQ#FY#IV#KR#LM
(17) AST#EQ#FY#IV#KR#LM
(18) AST#FY#IV#KR#LM
(19) FY#IV#KR#LM#ST
(20) IV#LM#ST
(21) IV#LM
(22) ADEGPST#FLMV#HNRWY#KQ
(23) AGPST#DEFKQRY#HNW#LMV
(24) AGSW#DEHNQRY#FIKLV#MP
(25) AT#DP#EGS#FHILMVY#KQ#NRW
(26) AFIMVY#DPR#EGLQS#HN#KT
(27) AEMQS#DKNR#FLV#GHW#PT
(28) ADEGKQS#FW#IM#LPTV#NR
(29) ALMVY#DKP#EFGNRS#HW
(30) AEPQ#DKR#GS#HW#ILMNTV
(31) AGS#DP#EK#FM#HNRWY#IV#LT
(32) ACDEFHIKLMNQRSTVWY#GP
(33) ACDEFHIKLMNQRSTVWY
(34) AEHKQRST#CFILMVWY#DN
(35) AEFHIKLMQRVWY#CT#DN
(36) AEFIKLMQRVWY#CH#DN
(37) AEFIKLMQRVWY#CH
(38) AEFIKLMQRVWY
(39) AEFIKLMQRV
(40) AEFIKLMQV

(41) AEFIKLMV
(42) AEFIKLV
(43) AFILMVWY#CDEGHKNPQRST
(44) AFILMVWY#CDEHKNQRST#GP
(45) ALM#CDEHKNQRST#FIVWY#GP
(46) ALM#CDHNST#EKQR#FIVWY#GP
(47) ALM#CHT#DNS#EKQR#FIVWY#GP
(48) CHT#DNS#EKQR#FIVWY#GP#LM
(49) CHT#DNS#EKQR#FIVWY#LM
(50) CHT#DNS#EKQR#FWY#IV#LM
(51) CH#DNS#EKQR#FWY#IV#LM
(52) CH#EKQR#FWY#IV#LM#NS
(53) EKQR#FWY#IV#LM#NS
(54) FY#IV#KQR
(55) FY#IV#QR
(56) FY#QR
(57) ADEGHKNQRST#FILMVY
(58) ADEGKNQRST#FILMVY
(59) ADEKNQRST#FILMVY
(60) AEKQRST#DN#FILMVY
(61) AEKQRST#DN#FY#ILMV
(62) DN#EKQRST#FY#ILMV
(63) DN#EKR#FY#ILMV#QST
(64) EKR#ILMV#QST
(65) EKR#ILV#QST
(66) EK#ILV#QST
(67) EK#ILV#ST
(68) ADEGHKNPQRST#FILMVY
(69) ADEHKNPQRST#FY#ILMV
(70) ADEHKNQRST#FY#ILMV
(71) AST#DEKNQR#FY#ILMV
(72) AST#DEKNQR#ILMV
(73) AST#DEN#ILMV#KQR
(74) AST#DE#ILMV#KQR
(75) AST#DE#ILM#KQR
(76) AST#ILM#KQR
(77) AST#ILM#KQ
(78) IL#KQ#ST
(79) AFILMVW#CDEGHKNPQRSTY
(80) AFILMVW#CGHKNPQRSTY#DE
(81) ACW#DE#FILMV#GHKNPQRSTY
(82) DE#FILMV#GHNPQSTWY#KR
(83) DE#FILMV#GNST#HPQWY#KR

(84) DE#FILMV#HPQWY#KR#NST
(85) DE#FM#HPQWY#ILV#KR#NST
(86) FM#HPQWY#ILV#KR#NST
(87) FM#HPQWY#IL#KR#NST
(88) FM#HPQWY#IL#KR#ST
(89) FM#HPQWY#KR
(90) FM#IL#KR#QWY
(91) FM#KR#QWY
(92) ACDEGHKNPQRST#FILMVWY
(93) ACPST#DEHKNQR#FILMVWY
(94) ACPST#DEHKNQR#FWY#ILMV
(95) ACST#DEHKNQR#FWY#ILMV
(96) ACST#DEHNQ#FWY#ILMV#KR
(97) ACST#DEHNQ#FWY#IMV#KR
(98) AST#DEHNQ#FWY#IMV#KR
(99) EKQR#FWY#HNST#IMV
(100) FWY#HNST#IMV#KQR
(101) FWY#HKQR#IMV#ST
(102) HKQR#IMV#ST#WY
(103) IMV#KQR#WY
(104) IMV#KQR
(105) AFILMVWY#DEGHKNPQRST
(106) AFILMVWY#DEKR#GHNPQST
(107) DE#FILMVWY#GHNPQST#KR
(108) DE#FILMVWY#HNPQST#KR
(109) DE#FILMV#HNPQST#KR#WY
(110) DE#FILMV#HNQST#KR#WY
(111) DE#FILMV#HNQ#KR#ST#WY
(112) DE#ILMV#NQ#ST
(113) AEIKLMQRV#CDFGHNPSTWY
(114) AEKQR#CFGPWY#DHNST#ILMV
(115) AEQ#CGP#DN#FWY#HST#ILMV#KR
(116) DN#EQ#FY#GP#IV#KR#LM#ST
(117) AG#CIV#DEKQR#FLMWY#HNPST
(118) AG#CIV#DE#FWY#HST#KQR#LM#NP
(119) AG#DE#FY#IV#KR#LM#NP#ST
(120) AGHPT#CMVWY#DEKNQRS#FIL
(121) AHT#CY#DEKN#FL#GP#MVW#QRS
(122) ALM#CHST#DN#EKQR#FWY#IV
(123) ALM#CT#DN#EKQR#FWY#HS#IV
(124) DN#EKQR#FWY#HS#IV#LM
(125) ACDEFHIKLMNPQRSTVWY
(126) AEFIKLMQRVWY#CDHNPST

(127) AEKLMQR#CDHNST#FIVWY
(128) ALM#CHST#DN#EKQR#FIVWY
(129) ALM#DN#EKQR#FIVWY#HST
(130) ALM#DN#FWY#IV#KQR#ST
(131) DN#FY#IV#KR#LM#ST
(132) AKPRSTV#CGHIL#DENQ#FMWY
(133) ACFGILMPV#DE#HKR#NQSTWY
(134) ACFGILMPV#DE#HKR#NQSTW
(135) ACFGILPV#DE#HKR#MY#NQSTW
(136) ACFGILPV#DE#HKR#MWY#NQST
(137) ACFGILV#DE#HKPR#MWY#NQST
(138) ANQST#CFGILV#DE#HKPR#MWY
(139) ANQST#CFGIL#DE#HKPRV#MWY
(140) ANQST#CFGIL#DEK#HPRV#MWY
(141) ANST#CFGIL#DEK#HPQRV#MWY
(142) APST#CGILMV#DENQ#FWY#HKR
(143) APST#CGILV#DENQ#FMWY#HKR
(144) APST#CGIL#DENQ#FMWY#HKRV
(145) APST#CGHIL#DENQ#FMWY#KRV
(146) APSTV#CGHILM#DENQ#FWY#KR
(147) APST#CILMV#DENQ#FWY#HKR
(148) LV
(149) AHPRT#CFILMVWY#DEGKNQS
(150) AHPRT#CILMV#DEGKNQS#FWY
(151) APQT#CILMV#DEGKNS#FWY#HR
(152) APT#CILMV#DE#FWY#GKNQS#HR
(153) AT#CILMV#DE#FWY#GNPQS#HR
(154) AT#DE#FWY#GNPQS#HR#ILMV
(155) DE#FWY#GNQS#HR#ILMV#PT
(156) DE#GNQS#HR#ILMV#PT#WY
(157) DE#HR#ILMV#NPQS#WY
(158) DE#HR#IL#MV#NPQS#WY
(159) DE#HR#IL#MV#NQS#WY
(160) DE#HR#IL#MV#NQS
(161) DE#IL#MV#NQS
(162) DE#IL#NQS
(163) DE#NQS
(164) DE#QS
(165) QS
(166) ACGPST#DEHKNQR#FILMVWY
(167) ACGST#DEHKNQR#FPWY#ILMV
(168) ACST#DEKNQR#FHWY#GP#ILMV
(169) ACST#DEN#FWY#HKQR#ILMV

(170) AST#DEN#FWY#HKQR#ILMV
(171) AST#DEN#FWY#ILMV#KQR
(172) AST#DE#FY#ILMV#KQR
(173) DEN#FY#ILMV#KR#ST
(174) FWY#ILMV#KR#ST
(175) FY#ILMV#KR#ST
(176) FY#ILV#KR#ST
(177) ILV#KR#ST
(178) ILV#KR
(179) AFHTY#CILMV#DEKPQ#GNRSW
(180) AIS#CHLV#DEPQ#FGMWY#KNRT
(181) ACFGHILMVWY#DEKNPQRST
(182) ACFILMVWY#DEKNPQR#GHST
(183) ACFILMVWY#DEKNQR#GHPST
(184) AMW#CLY#DEGHKNPQRST#FIV
(185) AM#CFIV#DEKNPQR#GHST#LWY
(186) AV#CGNP#EKQR#FHWY#ILM#ST
(187) AV#CGNP#EKR#FWY#ILM#ST
(188) AV#EKR#FWY#GNP#IL#ST
(189) AGHNPST#CFILMVWY#DEKQR
(190) ACHIMT#DEKPQ#FLVWY#GNRS
(191) ACIY#DESW#FLMV#GHNT#KPQR
(192) AGHPST#CFILMVWY#DEKNQR
(193) ACLM#DEKNPQRST#FIVW#GHY
(194) AGHST#CFILMVWY#DEKNPQR
(195) CFILMVWY#EK#GHST
(196) AGHPRT#CFILMVWY#DEKNQS
(197) AGST#CFIM#DENQ#HKPR#LVWY
(198) ADEGKNPQRST#CFHILMVWY
(199) ADEGNPST#CHKQRW#FILMVY
(200) AGNPST#CHWY#DEKQR#FILMV
(201) AGPST#CFWY#DEN#HKQR#ILMV
(202) APST#CW#DEGN#FHY#ILMV#KQR
(203) AGST#CW#DEN#FY#HP#ILMV#KQR
(204) AST#CG#DEN#FY#HP#ILV#KQR#MW
(205) AST#CW#DE#FY#GN#HQ#ILV#KR#MP
(206) AST#CW#DE#FY#GN#HQ#IV#KR#LM
(207) AST#DE#FY#GN#HQ#IV#KR#LM
(208) AST#DE#FY#HQ#IV#KR#LM
(209) AST#DE#FY#IV#KR#LM
(210) AST#DE#FL#IV#KR
(211) AST#DE#IV#KR
(212) AT#DE#IV#KR

(213) AT#DE#IV
(214) DE#IV
(215) ACDEFGHILMNPQRSTVWY
(216) ACDFGMPQRSTW#EHILNVY
(217) AGPT#CDFMQRSW#EHILNVY
(218) AGPT#CDQ#EHILNVY#FMRSW
(219) AG#CDQ#EFHILMNRSVWY#PT
(220) AG#CDQ#EHNY#FMRSW#ILV#PT
(221) AG#DQ#EHNY#FMRSW#ILV#PT
(222) AG#DQ#EHNY#FMW#ILV#PT#RS
(223) DQ#EHNY#FMW#ILV#PT#RS
(224) DQ#EHNY#FM#ILV#PT#RS
(225) DQ#EHNY#FM#IL#PT#RS
(226) DQ#FM#HNY#IL#PT#RS
(227) FM#HNY#IL#PT#RS
(228) FM#HNY#IL#PT
(229) HNY#IL#PT
(230) HNY#IL
(231) HNY
(232) HN
(233) ACFILMVWY#DEGHKNPQRST
(234) CFILMVWY#DEGHKNPQRST
(235) CFILMVWY#DEHKNPQRS
(236) CFILMVWY#DEKNPQRS
(237) CFILMVWY#DEKNQRS
(238) CILMV#DEKNQRS#FWY
(239) CILMV#DNS#EKQR#FWY
(240) DNS#EKQR#FWY#ILMV
(241) DN#EKQR#FWY#ILMV
(242) DN#EKQR#FWY#ILV
(243) EKQR#FWY#ILV
(244) EKQR#FWY#IV
(245) EKQR#IV#WY
(246) EKQR#IV
(247) IV#KR
(248) AG#CFILMVWY#DEHKNPQRST
(249) CFILMVWY#DEHKNPQRST
(250) CFILMVWY#DEHKNQRST
(251) CFILMVWY#DEHKNQR#ST
(252) DEHKNQR#FILMVWY#ST
(253) DEHKNQR#FWY#ILMV#ST
(254) DEKNQR#FWY#ILMV#ST
(255) DN#EKQR#FWY#ILMV#ST

(256) DN#EQ#FWY#ILMV#KR#ST
(257) DN#EQ#FWY#ILMV#KR
(258) DN#EQ#FWY#IV#KR#LM
(259) EQ#FWY#IV#KR
(260) EQ#FY#IV#KR
(261) FY#IV#KR
(262) ADEGHKNPQRST#FILMVWY
(263) ADGNST#EHKPQR#FILMVWY
(264) ADNST#EHKPQR#FILMVWY
(265) ADNST#EKQR#FILMVWY#HP
(266) AST#DN#EKQR#FILMVWY#HP
(267) AST#DN#EK#FILMVWY#HP#QR
(268) AST#DN#EK#FWY#HP#ILMV#QR
(269) DN#EK#FWY#HP#ILMV#QR#ST
(270) DN#EK#FWY#ILMV#QR#ST
(271) EK#FWY#ILMV#QR#ST
(272) EK#FWY#IV#LM#QR#ST
(273) EK#FWY#IV#LM#QR
(274) EK#FWY#IV#QR
(275) FWY#IV#QR
(276) IV#QR#WY
(277) IV#WY
(278) WY
(279) ADEFGIKLMNPQRSTVY
(280) ADEGIKLMNPQRSTV#FY
(281) ADEGIKLMNQRSTV#FY
(282) ADEIKLMNQRSTV#FY
(283) ADEKNQRST#FY#ILMV
(284) ADNST#EKQR#ILMV
(285) ANST#EKQR#ILMV
(286) ANST#EK#ILMV#QR
(287) AST#EK#ILV#QR
(288) AST#EK#ILV
(289) AST#ILV
(290) AST#IV
(291) AS#IV
(292) ADEGHKNPQRST#CFLMTVWY
(293) AY#CFLMTVW#DEGHKNPQRST
(294) AY#DEGHKNPQRST#FLMTVW
(295) AY#DEHKNPQRST#FLMTVW
(296) AH#DEKNPQRST#FLMTV#WY
(297) AH#DEKNQST#FLMTV#GP#WY
(298) DEHKNPST#FLMTV#WY

(299) DENQRST#FLMTV#WY
(300) DEKPQST#LMTV#NR
(301) DEKPST#FLMTV
(302) DEKPRS#LMTV
(303) DEKST#LMTV
(304) DEST#LMTV
(305) EST#LMTV
(306) DE#LMTV
(307) DE#LTV
(308) LTV
(309) AGST#CFMWY#DEHKNPQR#ILV
(310) ACPST#DEKNQR#FHWY#ILMV
(311) ACST#DEKNQR#FHWY#ILMV
(312) ACST#DENQ#FHWY#ILMV#KR
(313) AC#DE#FHWY#ILMV#KQR#NST
(314) AC#DE#FHWY#IV#KQR#LM#NST
(315) AC#EQ#FWY#HN#IV#KR#LM#ST
(316) FWY#HN#IV#LM
(317) ACGPSTWY#DEHKNQR#FILMV
(318) ACGMP#DEHNR#FILVWY#KQST
(319) AGILV#CM#DE#FWY#HKR#NQ#ST
(320) ACGS#DE#FWY#HKR#ILV
(321) ADEKNQRST#CFHILMVWY
(322) AE#DKR#FILMV#GP#NQST#WY
(323) ADNST#EKQR#FILVY
(324) AEFIKLMQRVW
(325) AM#DNS#EKQR#GP#HT#IV#LY
(326) DN#FWY#HS#IV#KQR
(327) ACKP#DFHMS#ELQR#GIV#NTWY
(328) AGPST#DENQ#FWY#HKR#ILMV
(329) DENQ#FWY#ILMV#KR#ST
(330) AC#DE#FWY#HN#IV#KQR#LM#ST
(331) AST#DN#EQ#FY#HW#ILMV#KR
(332) AST#DEN#FY#ILMV#KQR
(333) FWY#IV#KQR#LM#NS
(334) IV#KR#LM#ST
(335) AG#DEKNPQRST#FILMVWY
(336) ACILMV#DE#FHWY#GP#KR#NQST
(337) EKR#FY#ILMV#QST
(338) FY#IV#KQR#LM
(339) EK#ILV
(340) AFILMVWY#DE#GHNPQST#KR
(341) ACW#DE#FILMV#GHNPQSTY#KR

(342) FM#HPQWY#IL#KR
(343) HKQR#IMV#WY
(344) FY
(345) AV#CFILM#DKPQTW#EGHNRSY
(346) AH#CMVWY#DEGNPQRST#FIL
(347) AHT#CFILMVWY#DE#GP#KNQRS
(348) AFILMV#DEHKNQR#STWY
(349) AFV#DE#GW#HN#ILM#KPRY#QST
(350) ADEGHKNPQRST#CFILMVWY
(351) AGPST#CFWY#DEHKNQR#ILMV
(352) APST#CFWY#DEHKNQR#ILMV
(353) AST#CFWY#DEHKNQR#ILMV
(354) AST#CFWY#DEHNQ#ILMV#KR
(355) AST#CFWY#DEQ#HN#ILMV#KR
(356) AST#CFWY#DEQ#HN#IV#KR#LM
(357) AST#DEQ#FWY#HN#IV#KR#LM
(358) DEQ#FWY#HN#IV#KR#LM#ST
(359) EQ#FWY#HN#IV#KR#LM#ST
(360) EQ#FWY#HN#IV#KR#LM
(361) EQ#FWY#IV#KR#LM
(362) EQ#FWY#IV#LM
(363) EQ#FY#IV#LM
(364) FY#IV#LM
(365) FY#IV
(366) IV
(367) AG#CFILMVWY#DEHKNQRST
(368) AG#DE#FWY#HKR#ILV#NQ#ST
(369) ACDEGIKLMNPQRSTV#FHWY
(370) ACGILMNPQSTV#DE#FHWY#KR
(371) AGILPV#CMNQST#DE#FHWY#KR
(372) AGILPV#CMNQST#DE#FWY#KR
(373) AGILPV#CST#DE#FWY#KR#MNQ
(374) AGP#CST#DE#FWY#ILV#KR#MNQ
(375) AG#CST#DE#FWY#ILV#KR#NQ
(376) AG#DE#ILV#KR#NQ#ST
(377) AG#DE#ILV#NQ#ST
(378) AG#DE#ILV#NQ
(379) AG#DE#ILV
(380) DE#ILV
(381) DE#IL
(382) ADEFGHIKLMNPQRSTVY
(383) ADEGHIKMNPQRSTV#FLY
(384) ADEGHKNPQRST#FY#ILMV

(385) AGPS#CV#DEHKNQRT#FY#ILM
(386) AGP#CS#DEHNQ#FY#IKMRTV
(387) AG#CS#DENQ#FP#HY#IKMRTV
(388) AG#CSV#DENQ#HY#IKMRT
(389) AG#CSV#DENQ#HY#KMRT
(390) AG#CSV#DENQ#IR#KT#LM
(391) AFS#CV#DEQ#IM#KT#LN
(392) AFS#CV#DEQ#IM#LN
(393) AF#CS#DE#IV#LN#MQ
(394) AFS#CV#DE#LN
(395) AF#CV#DE#ST
(396) AF#DE#ST
(397) AF#DE
(398) AF
(399) ACDEGHIKMNPQRSTV#FLWY
(400) ACDEGHIKMNPQRSTV#FLY
(401) ADEGHKNPQRST#CIV#FLMY
(402) ADEGHKNPQRST#CV#FY#ILM
(403) ADEHKNPQRST#CV#FY#ILM
(404) ADEHKNQRST#CV#FY#ILM
(405) AHKQRST#CV#DEN#FY#ILM
(406) AGST#DEN#FLM#HKQR#IV
(407) AHKRS#DEN#FLM#ITV
(408) AST#DE#GN#HLM#IV#KQR
(409) ANST#DE#IV#KQR#LM
(410) AH#EQ#GN#IV#KR#LM#ST
(411) AH#EQ#GN#IV#LM#ST
(412) AH#EQ#GN#IV#ST
(413) AH#GN#IV#ST
(414) AH#GN#IV
(415) AH#GN
(416) GN
(417) AGHPSTY#CFILMVW#DEKNQR
(418) ADGPST#EILNQV#FHKMRWY
(419) AGPST#CFILMVWY#DEHKNQR
(420) ADGST#CEILNPQV#FHKMRWY
(421) ACFGHILMNPQSTVWY#DE#KR
(422) AEHKLMQR#CFITVWY#DGNPS
(423) ACFGILVW#DEKNPQ#HMPSTY
(424) AFILMPVW#CGNQSTY#DE#HKR
(425) ACFGHILMPSTVWY#DEKNQR
(426) AGHMPSTWY#CFILV#DEKNQR
(427) AGMSTW#CFILV#DEKNQR#HPY

(428) AGMSTW#CFILV#DNQ#EKR#HPY
(429) AMW#CFILV#DNQ#EKR#GST#HPY
(430) AMW#CFILV#DNQ#ER#GST#HPY
(431) AMW#CIV#DNQ#ER#FL#GST#HPY
(432) AMW#CIV#DNQ#ER#FL#GS#HPY
(433) AMW#CIV#DNQ#FL#GS#HPY
(434) AMW#CIV#DNQ#FL#HPY
(435) AMW#CIV#DNQ#FL#PY
(436) AMW#CIV#FL#NQ#PY
(437) AMW#CV#FL#NQ#PY
(438) AW#CV#FL#NQ
(439) AW#CV
(440) AW
(441) ACDEFHIKLMNQRTVWY#GPS
(442) AELM#CDFHIKNQRTVWY#GPS
(443) AELM#CDNTY#FHIKQRVW#GPS
(444) AELM#CDNTY#FHIKQRVW#GS
(445) AELM#CDT#FHIKQRVW#GS#NY
(446) AEM#CDT#FHIKQRVW#GS#NY
(447) AEM#CDT#FHIK#GS#NY#QRVW
(448) AEM#CDT#FHIK#NY#QRVW
(449) AEM#CDT#FHIK#NY#QV#RW
(450) CDT#EM#FHIK#NY#QV#RW
(451) DT#EM#FHIK#NY#QV#RW
(452) DT#EM#FIK#NY#QV#RW
(453) DT#EM#FIK#NY#RW
(454) DT#FIK#NY#RW
(455) DT#FIK#NY
(456) DT#IK#NY
(457) IK#NY
(458) NY
(459) ACDHLPQSV#EFGIKMNRTWY
(460) ACHLPV#DQS#EFGIKMNRTWY
(461) ACHLPV#DQS#EFGIKMNRTW
(462) ACHLPV#DQ#EFGIKMNRTW
(463) ACHLPV#DQ#EFGIMNRT#KW
(464) ACV#DQ#EFGIMNRT#HLP#KW
(465) ACV#DQ#EGNR#FIMT#HLP#KW
(466) ACV#EGNR#FIMT#HLP#KW
(467) ACV#EGNR#HLP#IMT#KW
(468) AC#EGNR#HLP#IMT#KW
(469) AC#ER#GN#HLP#IMT#KW
(470) AC#ER#GN#HP#IMT#KW

(471) AC#ER#GN#HP#IMT
(472) ER#GN#HP#IMT
(473) ER#GN#HP#MT
(474) ER#GN#MT
(475) ER#MT
(476) MT
(477) ADEFGIKLNPQRSTV#CHMWY
(478) ADEFGIKLNPQRSTV#CHMY
(479) AGLTV#CHMY#DEFIKNPQRS
(480) AGLTV#CHMY#DFKNQ#EIPRS
(481) AGLTV#CMY#DFKNQ#EIPRS
(482) CMY#DFKNQ#EIPRS#GLTV
(483) CMY#DFKNQ#GLTV#IPRS
(484) CMY#DFKNQ#GT#IPRS#LV
(485) CMY#DFKNQ#GT#IPR#LV
(486) CMY#DKN#FQ#GT#IPR#LV
(487) CM#DKN#FQ#GT#IPR#LV
(488) CM#DKN#FQ#IPR#LV
(489) CM#DKN#IPR#LV
(490) CM#DK#IPR#LV
(491) CM#DK#LV#PR
(492) CM#DK#PR
(493) CM#DK
(494) CM
(495) AHIPQY#CMNS#DTV#EL#FGKRW
(496) ADEHNPQT#CIKLMSV#FW
(497) ACDFMT#EHKNPQS#WY
(498) ACGILMPSTV#DENQ#FHKRWY
(499) ACGILMPSTV#DEHKNQ#FRWY
(500) ACGILMPSTV#DENQ#FWY#HKR
(501) ACGHNPQST#DE#FILMVWY#KR
(502) ACDEFGHKNPQRSTWY#ILMV
(503) ADEFGHKNPQRSTWY#ILMV
(504) ADEFGHNPQRSTWY#ILMV
(505) ADEFGHNPQSTWY#ILMV
(506) ADEFGNPQSTWY#ILMV
(507) ADEGNPQST#FWY#ILMV
(508) AGNPQST#DE#FWY#ILMV
(509) AGNQST#DE#FWY#ILMV
(510) AGNQST#DE#FW#ILMV
(511) AG#DE#FW#ILMV#NQST
(512) DE#FW#ILMV#NQST
(513) FW#ILMV#NQST

(514) ILMV#NQST
(515) ILMV#NQS
(516) ILMV#QS
(517) ILV
(518) ADEGLPSTV#CM#FWY#HKNQR
(519) AEFKLMNQRTV#CDGHIPSWY
(520) AFKMNQTV#CDGHIPSWY#ELR
(521) AKMQTV#CDGHIPSWY#ELR#FN
(522) AKMQTV#CGSY#DHIPW#ELR#FN
(523) AKMQTV#CS#DHIPW#ELR#FN#GY
(524) AKMQ#CS#DHIPW#ELR#FN#GY#TV
(525) AKMQ#CS#DHI#ELR#FN#GY#PW#TV
(526) AQ#CS#DHI#ELR#FN#GY#KM#PW#TV
(527) AQ#DHI#ELR#FN#GY#KM#PW#TV
(528) ADEFGHIKLMNPQRSTVWY
(529) AFGILMVWY#DEHKNPQRST
(530) AGWY#DE#FILMV#HKNPQRST
(531) AGWY#DE#FILMV#HNPQST#KR
(532) AGWY#DE#FILMV#HP#KR#NQST
(533) DE#FM#HP#ILV#KR#NQST#WY
(534) DE#FM#ILV#KR#NQST#WY
(535) DE#ILV#KR#NQST
(536) DE#ILV#NQ#ST
(537) DE#IL#NQ#ST
(538) IL#ST
(539) DN#EHKQ#IL
(540) IL
(541) ACDEGHIKLMNPQSTV#FY
(542) ADEGHIKLNPQRSTV#FY
(543) ADEGNPQST#FY#HIKL
(544) DN#EQ#FY#IL
(545) ACDEFGHIKLMNPQRSTVY
(546) ACDEGHIKLMNPQRSTV#FY
(547) ACDEGHIKLMNPQST#FY
(548) ADEGHIKLNPQST#FY
(549) DN#EHKQ#IL#PT
(550) DEHIKLMNPQRST#FWY
(551) DEHIKLNPQT#FY
(552) DN#EHKQ#FY#IL#PT
(553) DN#EHKQ#FY#IL
(554) DN#EQ#IL
(555) DEFHIKLMNPQRSTWY
(556) DENQT#FY#HIKLR

(557) EQ#FY#HIKL
(558) FY#IL
(559) AGV#C#DE#FILP#HNQW#KR#MSTY
(560) ACHKMRV#DEGNQST#FILPWY
(561) ADEGNQRSTW#CHKMV#FILPY
(562) AGILMV#CNPQST#DE#FWY#HKR
(563) AGHR#CFILMVWY#DEKNPQST
(564) AHLMVY#CFIW#DEGKNPQRST
(565) AGSTW#CFILMV#DEHKNPQRY
(566) ACFHILMVWY#DEKQRS#GNPT
(567) ACDGPST#EILNQV#FHKMRWY
(568) ACFGILVW#DEKNQR#HMPSTY