A
PROJECT REPORT ON

Structural Classification Of Proteins Using Machine Learning

By

PRINCY GAJERA CE041 19CEUOS157

KRISHIL PATEL CE121 19CEUOG147

B. Tech. CE Semester – VI

Subject: System Design Practice

Guided by:
Prof. Apurva A. Mehta
Assistant Professor
Dept. of Computer Engineering

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of **System Design Practice** and recorded in this journal is the bonafide work of

PRINCY GAJERA CE041 19CEUOS157

KRISHIL PATEL CE121 19CEUOG147

of B.Tech semester **VI** in the branch of **Computer Engineering** during the academic year **2021-2022.**

Prof. Apurva A. Mehta
Assistant Professor,
Dept. of Computer Engineering
Faculty of Technology
Dharmsinh Desai University, Nadiad

Dr. C. K. Bhensdadia
Head of the Department,
Dept. of Computer Engineering
Faculty of Technology
Dharmsinh Desai University, Nadiad

Acknowledgement

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us during this project. This project would have been materialized without the help from many who asked us good questions and rescued from various red tape crisis.

Theoretical knowledge is of no importance if one doesn't know the way of its implementation. We are thankful to our institute that provided us an opportunity to apply our theoretical knowledge through the project. We feel obliged in submitting this project as part of our curriculum.

We would like to take the opportunity to express our humble gratitude to our guide **Prof. Apurva A Mehta**, under whom we undertook our project. His constant guidance and willingness to share his vast knowledge made us enhance our knowledge and helped us to complete the assigned tasks to perfection. Without his effort, support and an astonishing testing ability this project may not have succeeded.

With Sincere Regards,
Krishil Patel
Princy Gajera

Table Of Content

List of Figures

Abstract

We are planning to develop a Structural Classification of Protein Sequence (SCOP) application. The main purpose of this project is to predict classes of given protein sequences. SCOP app is used to predict the class of given protein seq which will be provided in the form of PDB Id or PSSM file.

We are predicting the top level of SCOP hierarchical classification(Protein fold classes), which is further classified into folds, super families, families.

A motivation for this classification is to determine the evolutionary relationship between proteins. Proteins with the same shapes but having little sequence or functional similarity are placed in different super families, and are assumed to have only a very distant common ancestor. Proteins having the same shape and some similarity of sequence and/or function are placed in "families", and are assumed to have a closer common ancestor. This is the top-level "root" of the SCOP hierarchical classification.

1. Introduction

1.1 Brief Introduction:

In this system, there is only one end user i.e. one who is going to use this application. Here, the user is required to give a PDB Id or PSSM file as an input and based on that classes are predicted from below

I. Alpha
II. Beta
III. Alpha +Beta
IV. Alpha /Beta
V. others

1.2 Tools, Technology and Platform Used:

I. Programming Language : Python

II. IDE :

- Google Colab : Used Cloud GPU for training model. Also deployed flask app.
- Visual Studio Code : Testing and creating flask app

III. Framework : Flask

IV. Python Libraries Used :

- XGBoost (Used XGBClassifier)
- Pandas
- Pyngrok
- flask_ngrok (Flask apps running on localhost available over the internet via the excellent ngrok tool)
- iFeature (Used to extract physicochemical Feature Descriptors from Protein and Peptide Sequences)
- Scikit learn
- rpy2.ipython (Used to run R code into python notebook)

V. RPackages : PSSMCOOL (Used to extract features from PSSM)

2. Software Requirement Specification

2.1 Description

Class prediction is the first step for the structural classification of protein sequences.

The basic steps of the system are

1. Preprocessing
2. Model Training
3. Class prediction

2.2 Scope

This system is designed to perform in various scenarios where structural classification predictions can be utilized. scope of this system is global and open for all the users.

2.3 Types of User

Here, there is only one end-user who is going to use this application.

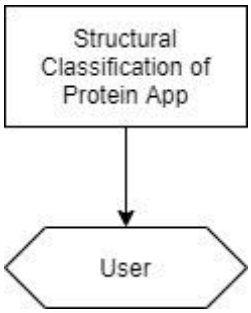


Fig. 2.1 Product Perspective Diagram

2.4 FUNCTIONAL REQUIREMENTS

2.4.1 Functional Requirements of features Dataset

R1. Data set must be unbiased to all classes(a, b, a+b, a/b).

R2. The dataset must not re-use the features of same proteins in training and testing phases.

2.4.2 Functional Requirements of system

R1. PDB Id to Class Prediction

Description: This requirement is for predicting class from the given PDB Id. System will extract the id from the data file and display it on the screen.

Input: PDB Id

Output: Displays the class on the screen

R2. PSSM File to Class Prediction

Description: This requirement is for predicting class from the given PSSM file. System will extract the features display class on the screen.

Input: PSSM File

Output: Displays the class on the screen

2.5 NON-FUNCTIONAL REQUIREMENTS

2.5.1 Performance :

The application should run very efficiently. it must be user friendly in nature and the prediction should be displayed to user as fast as possible.

2.5.2 Reliability :

The predictions returned by the program should be as accurate as possible.

2.6 HARDWARE REQUIREMENTS

Operating System : Windows 8 or Higher
Processor : Intel i3 or higher
Memory : 4GB or more
Graphics : CUDA() capable GUPs

3.Design

3.1 Use Case Diagram

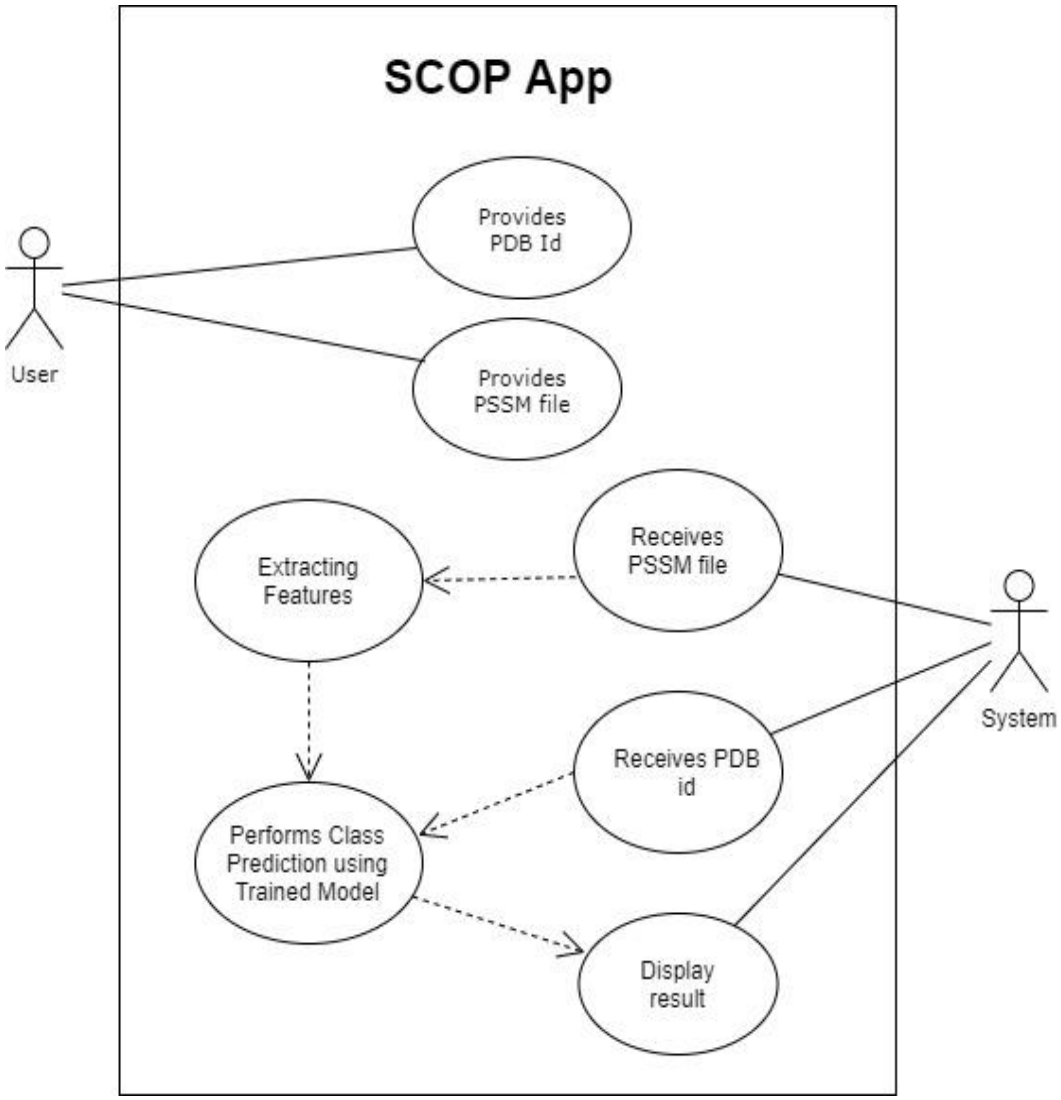


Fig. 3.1 Use Case Diagram

3.2 Activity Diagram

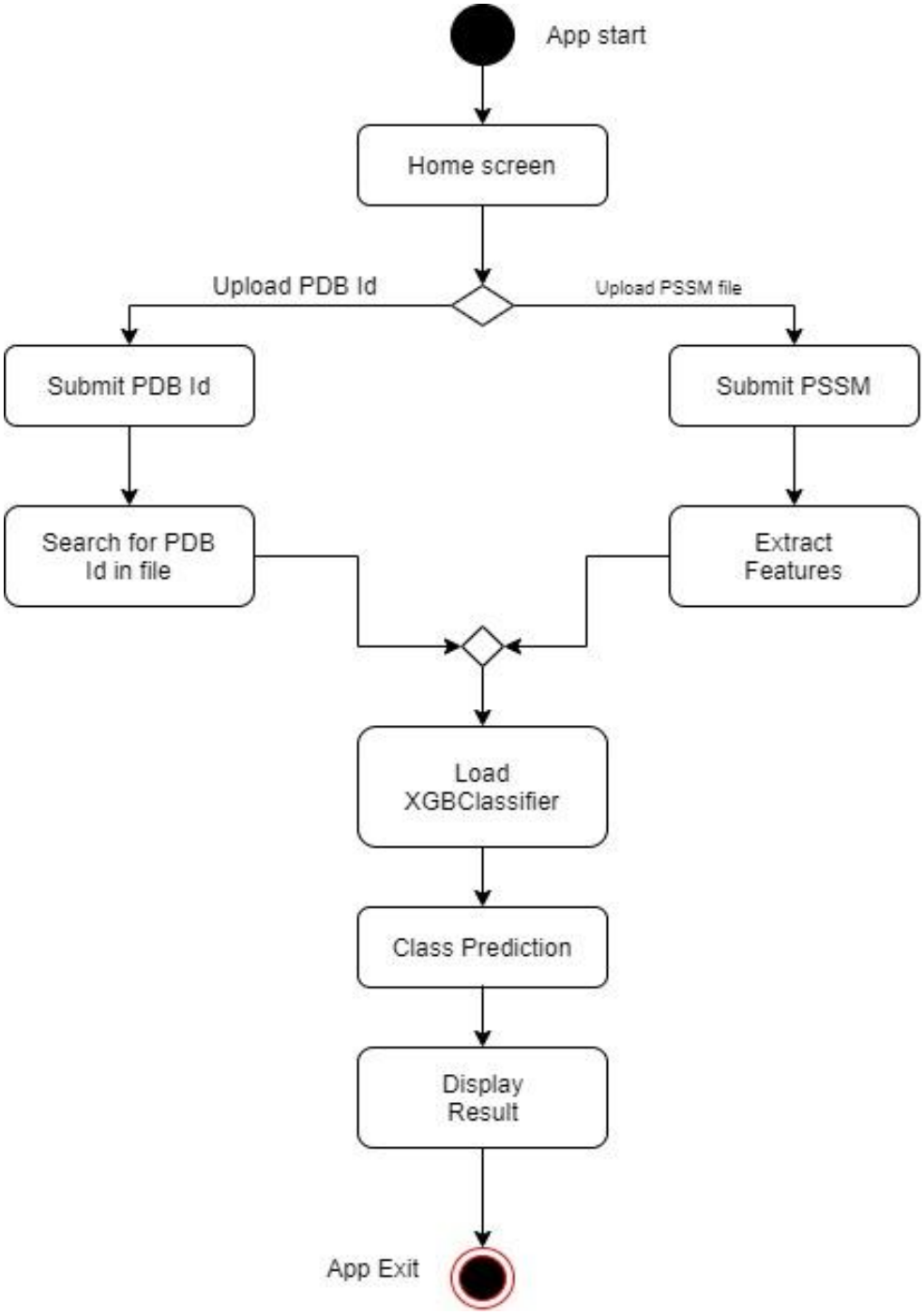


Fig. 3.2 Activity Diagram

3.3 Sequence Diagram

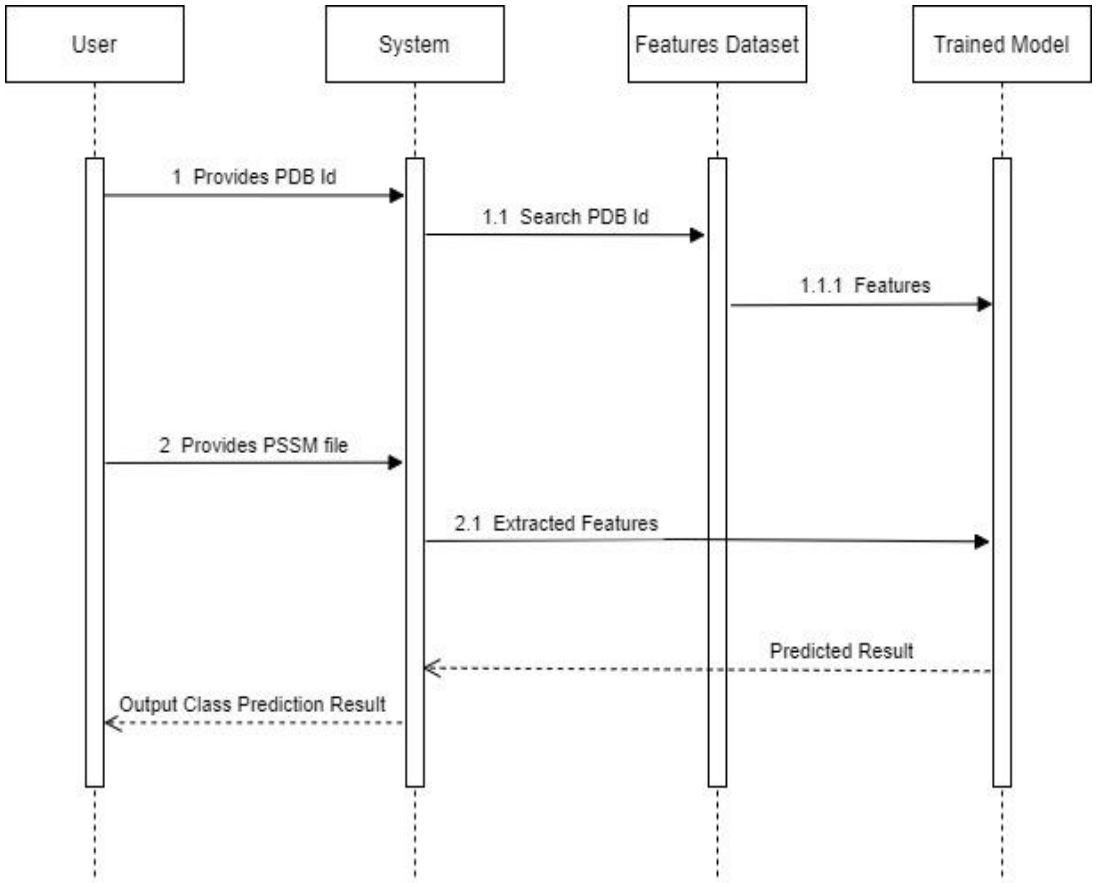


Fig. 3.3 Sequence Diagram

4.Analysis

4.1 Analysis of Accuracy

Different types of features: Sequential, Physico-chemical, Evolutionary, etc.Sequential and Physico-chemical includes features of one sequence. They are sequence specific.

4.1.1 Accuracy scores for physicochemical features

Sr.	Name	Accuracy
1	CTDC	0.5387953037263911
2	CTDT	0.5079122001020929
3	CTriad	0.501531393568147
4	AAC	0.5594691168963757
5	GAAC	0.41858090862685043
6	Combine Features	0.5959673302705462

4.1.2 Analysis of Accuracy for different PSSM features

PSSM is calculated using MSA so it includes the history of several sequences together. It highlights the evolution sequences experienced. Thus, features based on PSSM are more powerful than sequence specific features.

Sr.	Name	Accuracy
1	AATP_TPCC	0.7903349526975199
2	K_SEPERATED_BIGRAME	0.7310150856558425
3	CS_PSE_PSSM	0.8440296599335209
4	DWT_PSSM	0.7374073127077474
5	EDP_MEDP	0.8087445666070059
6	FPSSM	0.7333162873945283
7	PSSMBLOCK	0.7576067501917668
8	SCSH2	0.6576323190999744
9	AADP_PSSM	0.7772948095116339
10	DPC_PSSM	0.7816415239069292
11	DFMCA_PSSM	0.7642546663257479
12	LPC_PSSM	0.7164408079774993
13	MBMGAC_PSSM	0.8291996931731015
14	SOMA_PSSM	0.7576067501917668
15	SINGLE_AVERAGE	0.639989772436717
16	SVD_PSSM	0.5783687036563538
17	AATP_TPCC	0.7926361544362056
18	K_SEPERATED_BIGRAME	0.7320378419841472
19	CS_PSE_PSSM	0.8368703656353874
20	DWT_PSSM	0.7376630017898236
21	EDP_MEDP	0.8084888775249297
22	FPSSM	0.7305037074916901
23	PSSMBLOCK	0.7624648427512145
24	SCSH2	0.6486832012273076
25	TRIGRAME_PSSM	0.7995397596522629
26	combined_CS_PSE_MBMGAC	0.8476093070825875
27	combined_pssm(10,17,21,19,13)	0.8616722065967783
28	Combine_pssm(10,11,19,13)	0.8532344668882639

4.2 FUNDAMENTAL STEPS TO PERFORM FOR SCOP

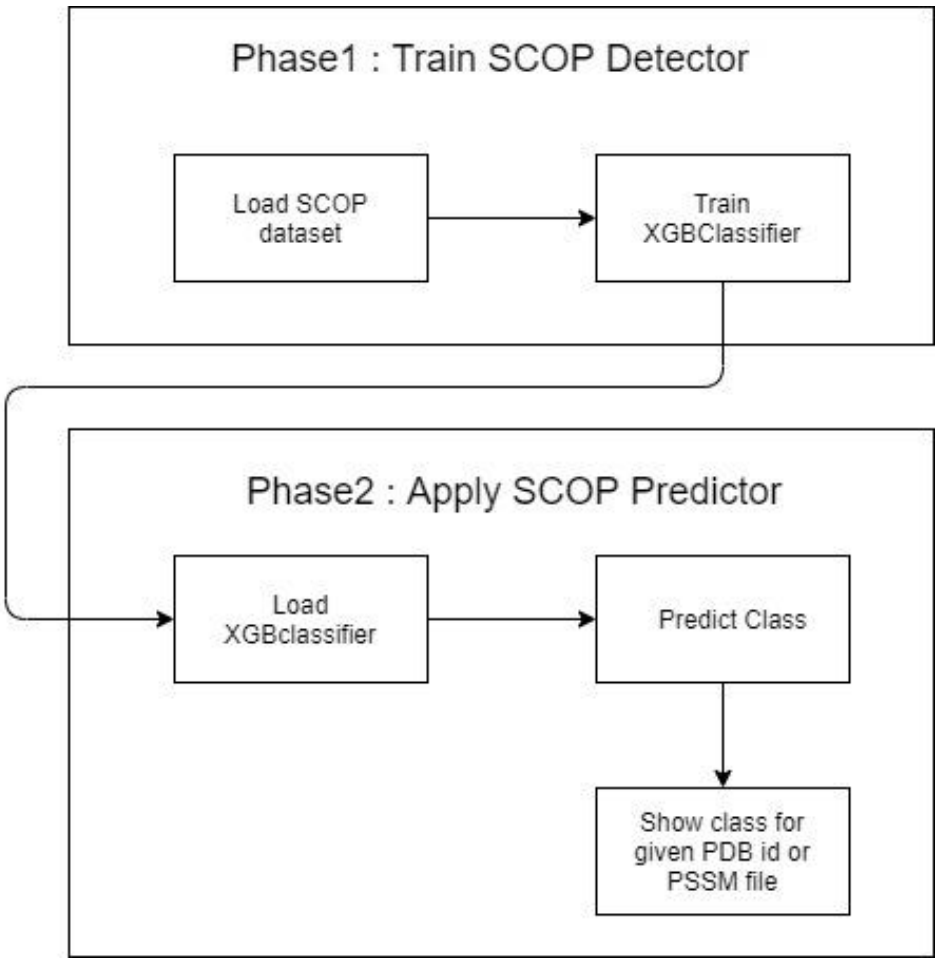


Fig. 4.1 Analysis of SCOP Diagram

5.Implementation Details

5.1 Front-end

5.1.1 Folder Structure

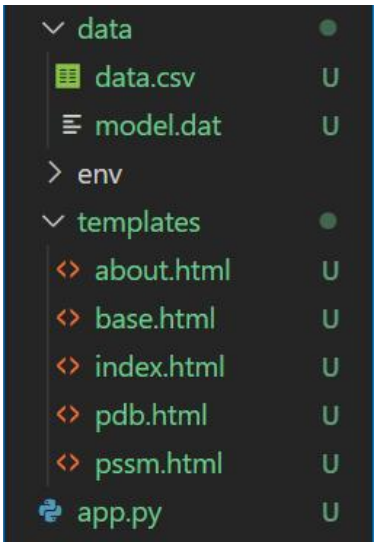


Fig. 5.1 Flask App's Folder Structure

5.1.2 Function for predict class from PDBId

```
index=-1
def predict(pdb):
    for i in range(0,13035):
        if(pdb==df.iloc[i,0]):
            index=i
            print(index)
            tmp=df.iloc[index,1:-1]
            result=pd.DataFrame([tmp])
            return model1.predict(result)
    return -1
```

Fig. 5.2 Predict from PDB Id Function

This function is used to predict class for given PDB Id which is given by the User.

```
def PredictFromPssm(filenm):
    f1(filenm)
    f2(filenm)
    f3(filenm)
    f4(filenm)

    df1=pd.read_csv("t1.csv")
    df1= df1.drop(df1.columns[0], axis=1)

    df2=pd.read_csv("t2.csv")
    df2= df2.drop(df2.columns[0], axis=1)

    df3=pd.read_csv("t3.csv")
    df3= df3.drop(df3.columns[0], axis=1)

    df4=pd.read_csv("t4.csv")
    df4= df4.drop(df4.columns[0], axis=1)

    df5 = pd.concat([df1,df2,df3,df4], axis=1)

    result=model2.predict(df5)
    return result
```

Fig. 5.3 Predict from PSSM File Function

5.2 Back-end

5.2.1. Feature Extraction

In First phase, features were extracted from a FASTA file (sequences.fasta) which contains protein sequences from nr-database, Features were extracted using IFeature (a python based toolkit)
link: https://github.com/Superzchen/iFeature/

```
(base) C:\Users\KRISHIL\Desktop\SDP proj\iFeature>python iFeature.py --help
usage: it's usage tip.

Generating various numerical representation schemes for protein sequences

optional arguments:
  -h, --help            show this help message and exit
  --file FILE           input fasta file
  --type {AAC,EAAC,CKSAAP,DPC,DDE,TPC,BINARY,GAAC,EGAAC,CKSAAGP,GDPC,GTPC,AAINDEX,ZSCALE,BLOSUM62,NMBroto,Moran,Geary,CT
OC,CTDT,CTDD,CTriad,KSCTriad,SOCNumber,QSOrder,PAAC,APAAC,KNNprotein,KNNpeptide,PSSM,SSEC,SSEB,Disorder,DisorderC,Disord
erB,ASA,TA}
                        the encoding type
  --path FILEPATH       data file path used for 'PSSM', 'SSEB(C)',
                        'Disorder(BC)', 'ASA' and 'TA' encodings
  --train TRAINFILE     training file in fasta format only used for
                        'KNNprotein' or 'KNNpeptide' encodings
  --label LABELFILE     sample label file only used for 'KNNprotein' or
                        'KNNpeptide' encodings
  --order {alphabetically,polarity,sideChainVolume,userDefined}
                        output order for of Amino Acid Composition (i.e. AAC,
                        EAAC, CKSAAP, DPC, DDE, TPC) descriptors
  --userDefinedOrder USERDEFINEDORDER
                        user defined output order for of Amino Acid
                        Composition (i.e. AAC, EAAC, CKSAAP, DPC, DDE, TPC)
                        descriptors
  --out OUTFILE         the generated descriptor file

(base) C:\Users\KRISHIL\Desktop\SDP proj\iFeature>python iFeature.py --file ../sequences.fasta --type GAAC --out output_
GAAC.tsv
Descriptor type: GAAC
```

```
PS C:\Users\KRISHIL\Desktop\SDP proj\iFeature>  python iFeature.py --file ..\sequences.fasta --type CTDT
Descriptor type: CTDT
PS C:\Users\KRISHIL\Desktop\SDP proj\iFeature>
```

Fig. 5.4 Extracting Features from iFeature.py

In second phase features extracted from pssm files were used which are extracted with the help of PSSMCOOL (A R package used to extract features from PSSM)

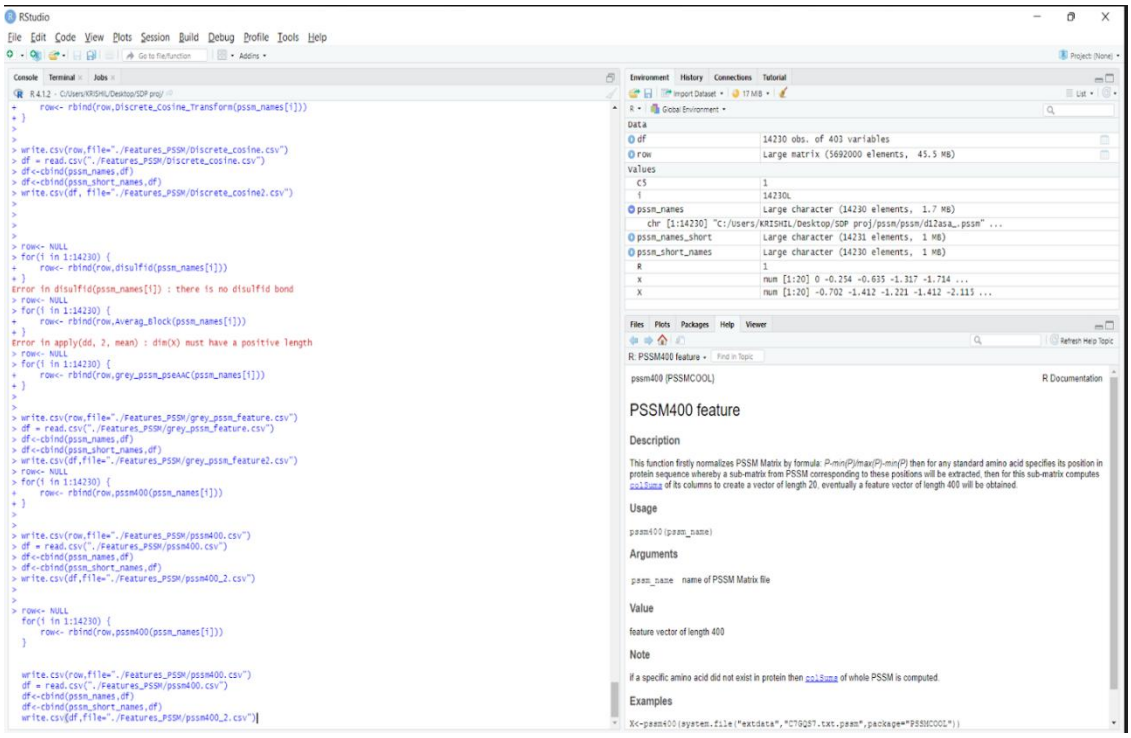


Fig. 5.5 Extracting Features using PSSMCOOL

PSSM(Position Specific Scoring Matrix) can be calculated with the help of psiblast tool and nr-database.

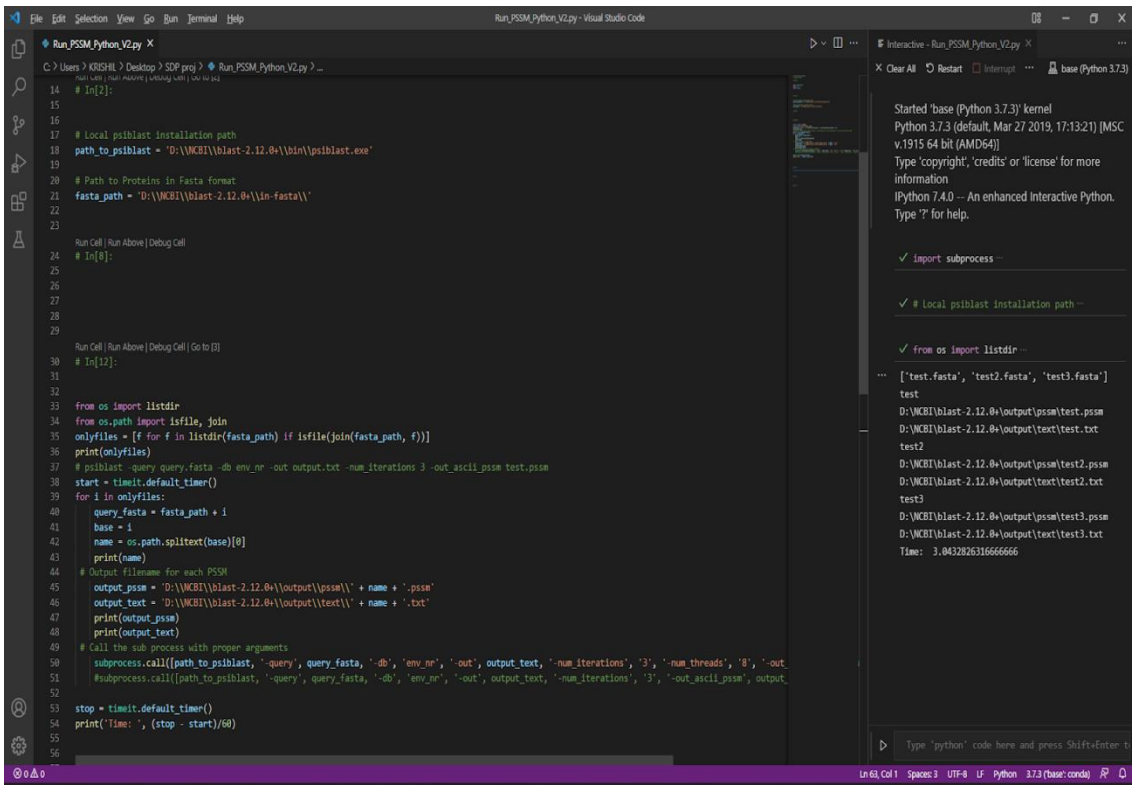


Fig. 5.6 Script to extract PSSM from fasta file using psiblast

5.2.2 Creating and Training of model:

During First phase we have used both RandomForest Classifier of sklearn.ensemble module and XGBClassifier of xgboost library on each extracted features from IFeature and also on combined features.

```
def classModel(data):
  X= data.iloc[:,:-1].values
  y= data.iloc[:,-1].values

  le = LabelEncoder()
  y= le.fit_transform(y)

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 5)

  model = XGBClassifier()
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)
  print('Confusion matrix: \n', confusion_matrix(y_test,y_pred))

  accuracy = accuracy_score(y_test,y_pred)
  # accuracy_scores.append(accuracy)
  print("\n\nAccuracy:",accuracy_score(y_test, y_pred))

  print("\n\nPrecision:",precision_score(y_test, y_pred, average=None))
  print("\nRecall:",recall_score(y_test, y_pred, average=None))
  print("\nF1 score:",f1_score(y_test, y_pred, average=None))
```

Fig. 5.7 Initial model

```
data = pd.read_csv('/content/drive/MyDrive/Project/output_csv_iFeature/CTDC_d.csv')
data.tail()
```

	hydrophobicity_PRAM900101.G1	hydrophobicity_PRAM900101.G2	hydrophobicity_PRAM900101.G3	hydrophobicity_ARGP820101.G1	hydrophobicity_ARGP820101.G2	hydrophobicity_ARGP820101.G3	hydrophobicity_ZI
13052	0.358140	0.400000	0.241860	0.409302	0.344186	0.246512	
13053	0.444730	0.282776	0.272494	0.426735	0.287918	0.285347	
13054	0.409160	0.282443	0.308397	0.387786	0.294656	0.317557	
13055	0.351767	0.334869	0.313364	0.331797	0.319508	0.348694	
13056	0.382609	0.347826	0.269565	0.330435	0.373913	0.295652	

```
classModel(data)

/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[14:04:22] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_m
Confusion matrix:
 [[421  48 164 149]
 [ 39 498 156 193]
 [ 63  78 886 158]
 [158 191 410 306]]


Accuracy: 0.5387953037263911


Precision: [0.61820852 0.61104294 0.54826733 0.37965261]

Recall: [0.53836317 0.56207675 0.74767932 0.28732394]

F1 score: [0.57552973 0.58553792 0.6326312  0.32709781]
```

Fig. 5.8 Classification report on features from iFeature

In second phase we have used XGBClassifier on features extracted by PSSMCOOL and also tried combination of features to select better model.

```
def classModel3(data):
  X= data.iloc[:,:-1].values
  y= data.iloc[:,-1].values

  le = LabelEncoder()
  print(y)
  y= le.fit_transform(y)
  print(y)

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 5)

  model = XGBClassifier(tree_method='gpu_hist',use_label_encoder=False)
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)

  print('\nClassification report:\n', classification_report(y_test,y_pred))
  print('Confusion matrix: \n', confusion_matrix(y_test,y_pred))

  accuracy = accuracy_score(y_test,y_pred)
  accuracy_scores.append(accuracy)
  print("\n\nAccuracy:",accuracy_score(y_test, y_pred))

  print("\n\nPrecision:",precision_score(y_test, y_pred, average=None))
  print("\nRecall:",recall_score(y_test, y_pred, average=None))
  print("\nF1 score:",f1_score(y_test, y_pred, average=None))
```

Fig. 5.9 Using GPU for Predictions

```
def automation(filenames):
  size= len(filenames)
  for i in range(size):
    print()
    print()
    print("Feature Name: ", filenames[i])
    print()
    data = pd.read_csv(filenames[i])
    classModel3(data)

files= ['AADP_PSSM.csv', 'DPC_PSSM.csv', 'DFMCA_PSSM.csv', 'LPC_PSSM.csv', 'MBMGAC_PSSM.csv',
        'SOMA_PSSM.csv', 'SINGLE_AVERAGE_PSSM.csv', 'SVD_PSSM.csv',
        'AATP_TPCC.csv',
              'K_SEPERATED_BIGRAME.csv',
              'CS_PSE_PSSM.csv',
              'DWT_PSSM.csv',
              'EDP_MEDP.csv',
              'FPSSM.csv',
              'PSSMBLOCK.csv',
              'SCSH2.csv',
              'TRIGRAME_PSSM.csv',
               'combined_CS_PSE_MBMGAC.csv',
        'combined_pssm.csv'
              ]
```

Fig. 5.10 Utility function

```
automation(files)



Feature Name:  AADP_PSSM.csv

['d' 'd' 'd' ... 'd' 'd' 'b']
[3 3 3 ... 3 3 1]
[16:33:31] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was chang

Classification report:
              precision    recall  f1-score   support

           0       0.80      0.77      0.79       772
           1       0.74      0.77      0.76       825
           2       0.84      0.89      0.87      1293
           3       0.70      0.63      0.66      1021

    accuracy                           0.78      3911
   macro avg       0.77      0.77      0.77      3911
weighted avg       0.77      0.78      0.78      3911

Confusion matrix:
 [[ 598   33   54   87]
 [  29  637   38  121]
 [  27   35 1157   74]
 [  91  152  130  648]]


Accuracy: 0.7772948095116339


Precision: [0.80268456 0.74329055 0.83901378 0.69677419]

Recall: [0.7746114  0.77212121 0.89481825 0.63467189]

F1 score: [0.78839815 0.75743163 0.86601796 0.66427473]
```

Fig. 5.11 Classification Report on features from PSSMCOOL

We saved 2 models using pickle module in .dat format. Both models can be used for prediction.

For model1:

Features Used:

- DPC_PSSM
- AATP_TPCC
- EDP_MEDP
- CS_PSE_PSSM
- MBMGAC_PSSM

```
 y_pred = model.predict(X_test)

print('\nClassification report:\n', classification_report(y_test,y_pred))
print('Confusion matrix: \n', confusion_matrix(y_test,y_pred))

accuracy = accuracy_score(y_test,y_pred)
accuracy_scores.append(accuracy)
print("\n\nAccuracy:",accuracy_score(y_test, y_pred))

print("\n\nPrecision:",precision_score(y_test, y_pred, average=None))
print("\nRecall:",recall_score(y_test, y_pred, average=None))
print("\nF1 score:",f1_score(y_test, y_pred, average=None))
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       772
           1       0.84      0.89      0.86       825
           2       0.92      0.91      0.91      1293
           3       0.80      0.77      0.78      1021

    accuracy                           0.86      3911
   macro avg       0.86      0.86      0.86      3911
weighted avg       0.86      0.86      0.86      3911

Confusion matrix:
 [[ 674   14   25   59]
 [  13  736    8   68]
 [  18   23 1179   73]
 [  58  106   75  782]]


Accuracy: 0.8619278956788545


Precision: [0.88335518 0.83731513 0.91608392 0.79633401]

Recall: [0.87305699 0.89212121 0.91183295 0.76591577]

F1 score: [0.8781759  0.86384977 0.91395349 0.78082876]
```

```
pickle.dump(model, open("model.dat", "wb"))
```

Fig. 5.12 Saving model1

For model2:

Features used:

- DPC_PSSM
- DFMCA_PSSM
- CS_PSE_PSSM
- MBMGAC_PSSM

```
X= df.iloc[:,:-1]
y= df.iloc[:,-1]
print(X)
print(y)

le = LabelEncoder()
print(y)
y= le.fit_transform(y)
print(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 5)

model = XGBClassifier(tree_method='gpu_hist',use_label_encoder=False)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('\nClassification report:\n', classification_report(y_test,y_pred))
print('Confusion matrix: \n', confusion_matrix(y_test,y_pred))

accuracy = accuracy_score(y_test,y_pred)
#accuracy_scores.append(accuracy)
print("\n\nAccuracy:",accuracy_score(y_test, y_pred))

print("\n\nPrecision:",precision_score(y_test, y_pred, average=None))
print("\nRecall:",recall_score(y_test, y_pred, average=None))
print("\nF1 score:",f1_score(y_test, y_pred, average=None))

pickle.dump(model, open("model2.dat", "wb"))
```

Fig. 5.13 Saving model2

Confusion Matrix for Model1:

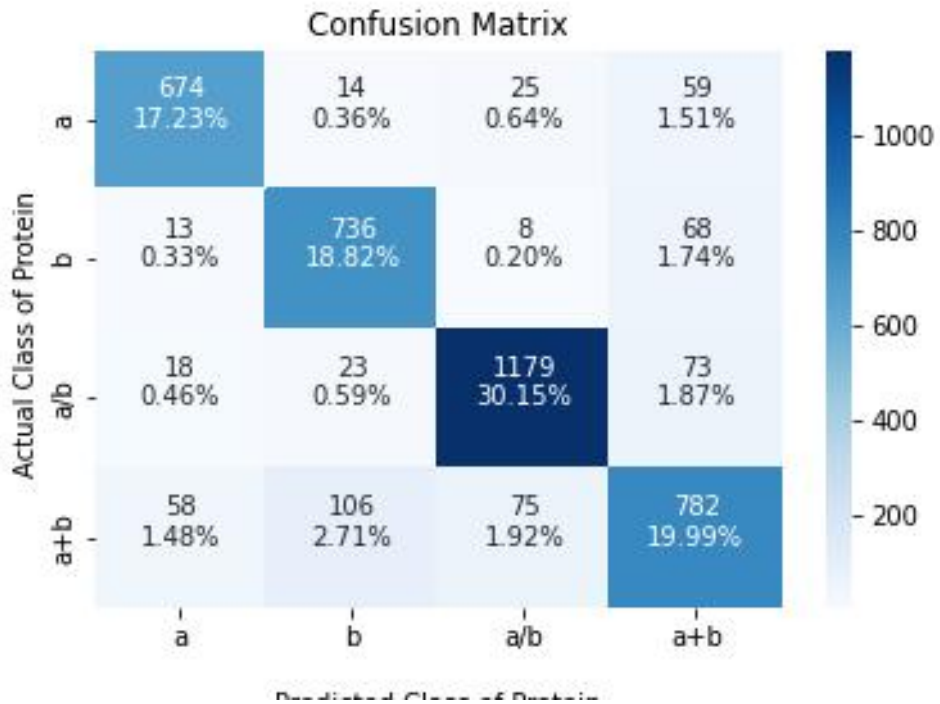


Fig. 5.14 Model1 Confusion Matrix

Confusion Matrix for Model2:

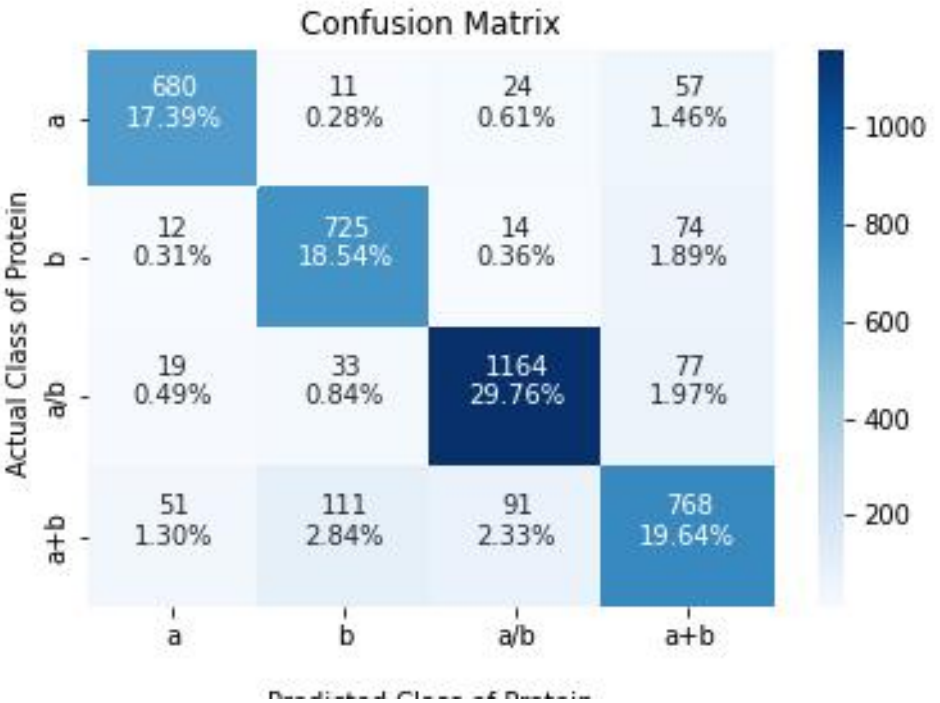


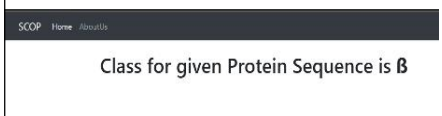
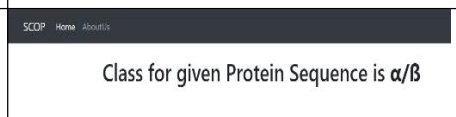
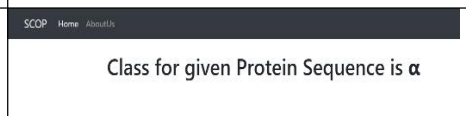
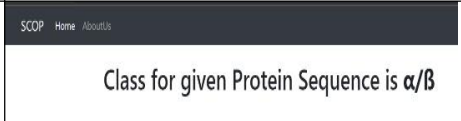
Fig. 5.15 Model2 Confusion Matrix

6. Testing

5.1 Testing Method Used

we have used black box testing method.For black box testing, we have designed the test cases and have tested it in our application.

5.2 Test Cases

Sr.	Name of file	Actual Target	Predicted Target	Pass/ Fail
1	d1q7ha1.pssm	a		Fail
2	d1daba_.pssm	b		Pass
3	d1knwa2.pssm	c		Pass
4	d1jfib_.pssm	c		Fail
5	d2gi3a1.pssm	a		Fail
6	d1q8ba_.pssm	d		Pass

7. Screen shots

7.1 Home Page

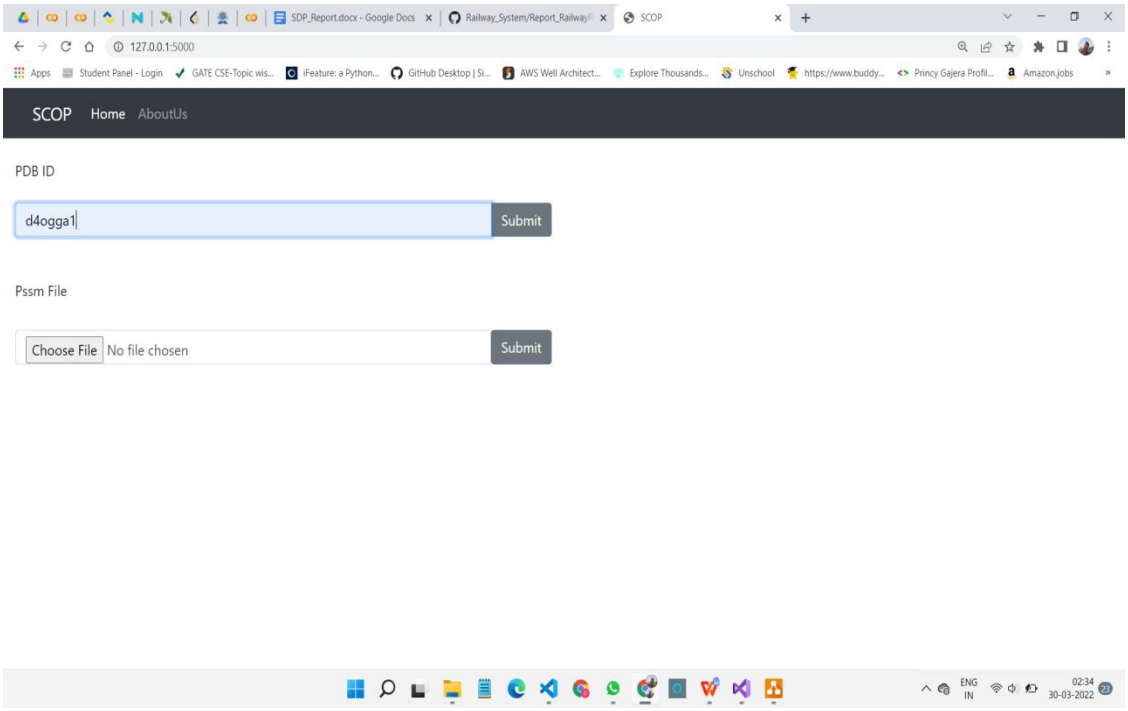


Fig. 7.1 Home page

7.2 About Page

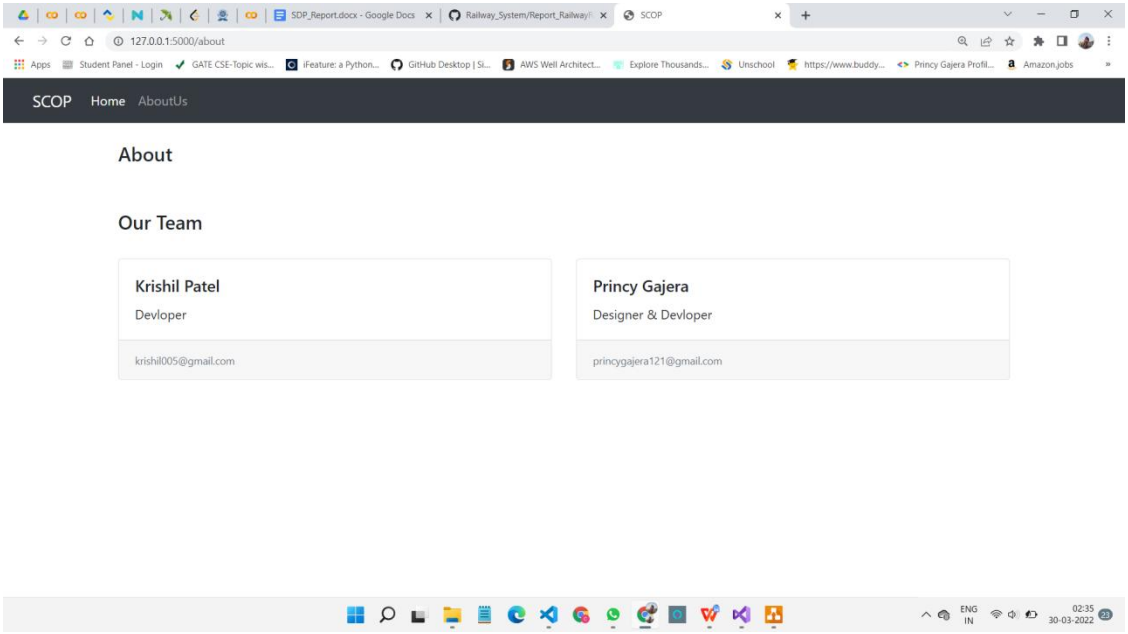


Fig. 7.2 About us

8. Conclusion

The broadest groups on SCOP are the protein fold classes. These classes group structures with similar secondary structure composition, but different overall tertiary structures and evolutionary origins. This is the top-level "root" of the SCOP hierarchical classification.
The project is developed using the python language and Flask. We implement this project in google colab.

We can predict up to 4 classes.

- All alpha proteins: Domains consisting of α-helices
- All beta proteins: Domains consisting of β-sheets
- Alpha and beta proteins (a/b): Mainly parallel beta sheets (beta-alpha-beta units)
- Alpha and beta proteins (a+b): Mainly antiparallel beta sheets (segregated alpha and beta regions)

So, after performing various comprehensive tests, we conclude that our project is working fine. But of course, there is always a scope for
improvement and learning. This was our first ML project but yes, in the end, we did learn something new from it. We are now looking forward to overcoming the existing limitations and adding new possible extensions which are discussed in the next section.

9. Limitation and Future Extension

9.1 Limitation

- The scope of our project is right now limited to predicting only one of the following classes: α, ß, α/ß, α+ß
- Users should already have a PSSM file with them to predict the class of protein sequence.

9.2 Future Extension

- We can extend this project to predict more classes and folds.
- We will improve the accuracy of this model by using some other features that separate classes in the best way.
- We will predict the class of protein directly using the fasta file.

10. Bibliography

https://scop.berkeley.edu/

https://ifeature.erc.monash.edu/

https://colab.research.google.com/

https://scholar.google.com/

https://cran.r-project.org/web/packages/PSSMCOOL/index.html

https://github.com/Alireza9651501005/PSSMCOOL

https://github.com/Superzchen/iFeature/

https://www.google.com/

https://stackoverflow.com/

https://docs.python.org

https://medium.com/@danielflor/using-xgboost-with-gpu-in-google-collab-4961999555f4

https://machinelearningmastery.com/

https://www.rdocumentation.org/packages/PSSMCOOL/versions/0.2.4

https://towardsdatascience.com/deploy-a-machine-learning-model-using-flask-da580f84e60c

https://medium.datadriveninvestor.com/machine-learning-model-deployment-using-flask-in-google-colab-1f718693a3c0