Classification of protein superfamily using deep learning (DL) models:

Proteins are the complex biomolecules that play important roles in the living organism, made up of a long chain of chemical units called amino acids simply represented by alphabets. There are 20 most common amino acids and 5 uncommon units arranged randomly in sequences which build millions of unique proteins. Functions of proteins depend on the number of amino acids and their conformations. The proteins with similar structural and functional domains are classified as members of a certain family. Traditional experimental methods to identify the functional relations of new proteins to its family is complicated and time consuming. Therefore, many machine learning (ML) and deep learning (DL) techniques have been developed to identify the respective families of novel proteins based on the available datasets. In this project, given the sequence of the amino acid residues (building block of proteins), we implement DL models and ML models to classify the protein functional domains based on the dataset extracted from the protein data bank (PDB). Names and symbols of 20 common amino acid units are shown in the table.

Amino Acid	**3-Letter Code**	**1-Letter Code**
Alanine	Ala	A
Cysteine	Cys	C
Aspartic acid or aspartate	Asp	D
Glutamic acid or glutamate	Glu	E
Phenylalanine	Phe	F
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Lysine	Lys	K
Leucine	Leu	L
Methionine	Met	M
Asparagine	Asn	N
Proline	Pro	P
Glutamine	Gln	Q
Arginine	Arg	R
Serine	Ser	S
Threonine	Thr	T
Valine	Val	V
Tryptophan	Trp	W
Tyrosine	Tyr	Y

One of the protein sequences from our dataset with 286 amino acid units that belongs to 'hydrolase' family is:

'TYTTRQIGAKNTLEYKVYIEKDGKPVSAFHDIPLYADKENNIFNMVVEIPRWTNAKLEITKEETLNPIIQDTK
KGKLRFVRNCFPHHGYIHNYGAFPQTWEDPNVSHPETKAVGDNEPIDVLEIGETIAYTGQVKQVKALGIMALLD
EGETDWKVIAIDINDPLAPKLNDIEDVEKYFPGLLRATNEWFRIYKIPDGKPENQFAFSGEAKNKKYALDIIKE
THDSWKQLIAGKSSDSKGIDLTNVTLPDTPTYSKAASDAIPPASLKADAPIDKSIDKWFFISGSV'

In this project, we applied a deep learning approach for classification of proteins based on only sequences. We applied characters to numeric vectors mapping using natural language processing (NLP) technique n-gram text frequency vectorizer to extract the features from the amino acid sequences. Large numbers of extracted features were subjected to the two different approaches of deep neural networks: convolutional neural network (CNN) and recurrent neural network (RNN). The model performances will be compared in terms of accuracy scores and confusion matrix scores for the multi-class classifications.

We also will apply various machine learning models to classify the protein families with the other features such as molecular weights, residual counts, Ph-value, length of the sequence, molecular densities, method of crystallizations, etc. By using these features, we will observe the performance of the ML models and compare the performances among the various ML models. However, we will not compare the performance of machine learning models to the deep learning models because the set of features used in ML and DL are completely different.

Data collection and cleaning:

We collected protein sequence data with corresponding classification from Kaggle. The data were taken from protein data bank and uploaded to Kaggle.com in the .csv format. The data were presented in two different files 'pdb_data_no_dups.csv' and 'pdb_data_seq.csv'. The former set contains 141, 401 samples of proteins with features listed as 'structureId', 'classification', 'experimentalTechnique', 'macromoleculeType', 'residueCount', 'resolution', 'structureMolecularWeight', 'crystallizationMethod', 'crystallizationTempK', 'densityMatthews', 'densityPercentSol', 'pdbxDetails', 'phValue', 'publicationYear', 'chainId'. The second file contains 104812 samples with columns: 'structureId', 'chainId', 'sequence', 'residueCount'. We counted the unique sequences and unique families and found 104812 unique sequences and 5050 unique classes. Observing the contents with a type of macromolecules, we found there are proteins, hybrid proteins, DNA, RNA, etc. We have shown the macromolecule type contents of the dataset in a pie chart Fig. 1. 87% samples belong to the protein and remaining sequences are hybrid proteins with DNA and RNA. We only keep the samples that belong to proteins.

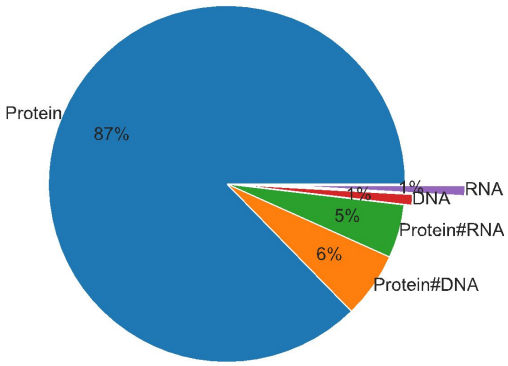


Figure 1: Showing the portion of samples belonging to the protein, protein+DNA, DNA, protein+RNA and RNA

Our goal was to predict the class of a protein given its sequence. Therefore, we observed the target variable 'classes. We found a total 5050 classes in the data set but all of them were not unique. There were several entries with classes labeled as different styles such as 'slashes, comma, parenthesis, etc.' We identified and removed the redundant labels. After removing the redundant entries, the total number of classes was reduced to 2684. We used seaborn heatmap to detect missed data. Since, a very small portion of data was missing in the dataset, we removed the samples with missing features. Also, we remove some columns such as 'chainId' which are completely independent of the classification. After cleaning the data, we got the single data file with 87098 samples with features such as 'experimentalTechnique', 'residueCount', 'resolution', 'structureMolecularWeight', 'densityMatthews', 'densityPercentSol', 'phValue', 'sequence' and a target column 'classes' with 2684 classes. Step by step data wrangling/cleaning process scripted in python code can be found in the jupyter notebook link here.

Exploratory Data Analysis and Feature Engineering:

Class Frequency:

We plotted the frequency of classes for 20 most common classes and found that the highest portion of the protein sequence was 'hydrolase' and the number of proteins belonging to the other classes were decreasing exponentially as shown in Fig. 2.

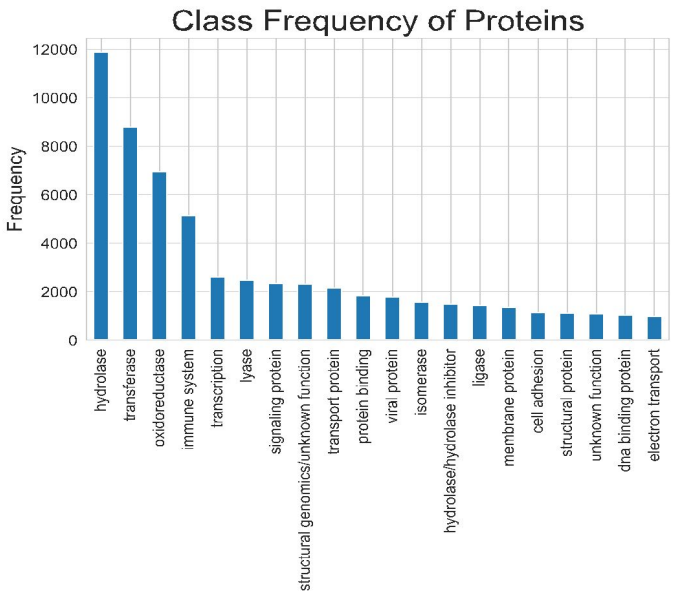


Figure 2: Frequency distribution of protein classes. 20 most common classes are plotted, 'hydrolase' is the most common class based on our dataset.

From the class frequency distribution, we came to know that there are very common classes. We filtered out the classes containing less than 1200 protein sequences. After such filtration, 15 unique protein families remained to be classified. Therefore, we applied multi-class classification models. Fifteen families of proteins were the target converted to the numerical labels before applied to the models. The classes to labels correspondence was:

Protein families	**Numerical Labels:**
{*'hydrolase'*	: 1,
'hydrolase/hydrolase inhibitor'	: 13,
'immune system'	: 4,
'isomerase'	: 12,
'ligase'	: 14,
'lyase'	: 6,
'membrane protein'	: 15,
'oxidoreductase'	: 3,
'protein binding'	: 10,
'signaling protein'	: 7,
'structural genomics/unknown function'	: 8,
'transcription'	: 5,
'transferase'	: 2,
'transport protein'	: 9,
'viral protein'	: 11}

We also observed the length and residue count distribution and found that the sequence length and residue counts have similar distribution.

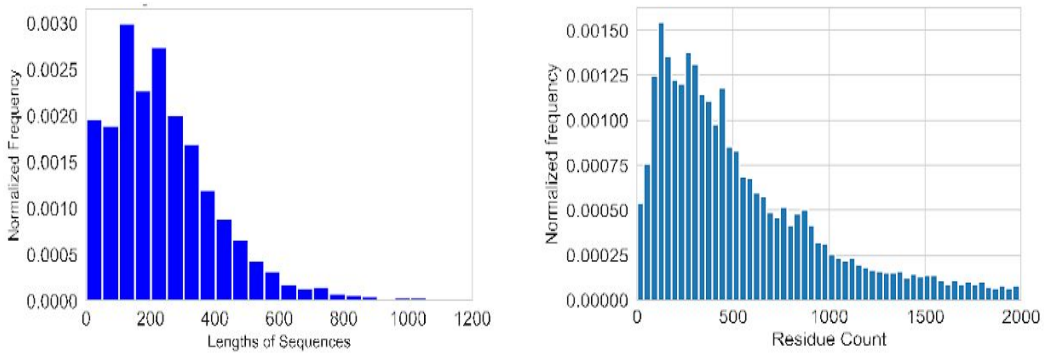


Figure 3: Sequence length and residue count distribution. Most of the proteins were of lengths less than 600 amino acid units

Number of letters representing the amino acid units of all sequences can be observed in Fig. 5. The appearance of all 20 common amino acids were almost equal as expected and the frequency of the 5 uncommon residues were found to be rare.

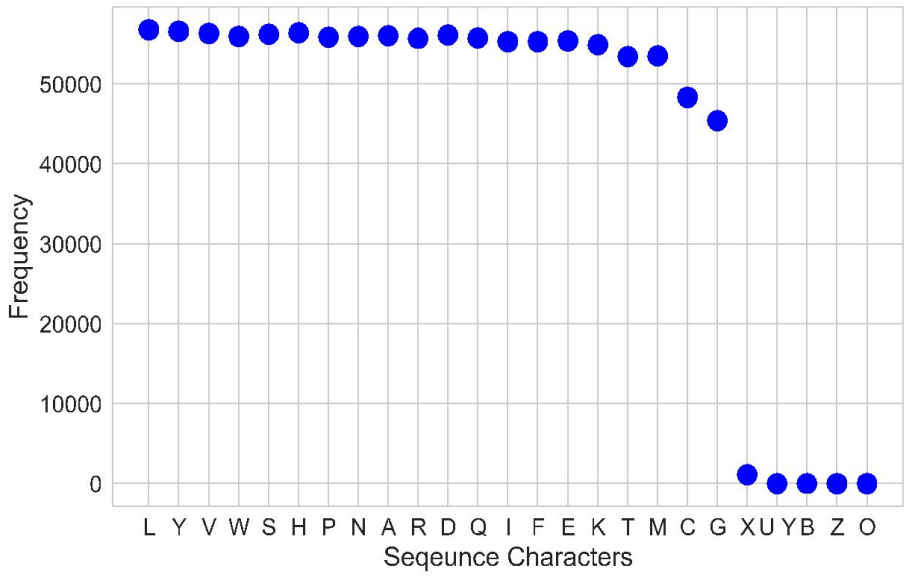
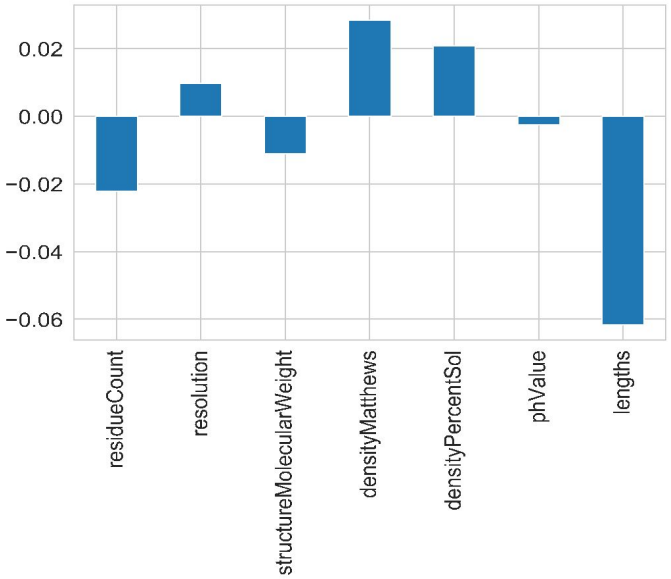


Figure 4: Frequency of amino acid units appearing in the dataset. Uncommon amino acids X, U, B, Z and O appear rarely in the sequences.

Feature Correlations:



We observed the feature correlations to the frequency of classes in figure 5. For the machine learning models, we only had seven features. We were applying various machine learning models to predict the protein classes. The results were compared to the different ML models but not with the DL models since in the later we extracted a huge number of features from the protein sequence. The python codes along with the visualization graphs for the EDA for protein sequences can be found in the link to jupyter notebook here.

Machine Learning Models:

Various machine learning models were applied: Random Forest (RF) classifier, Decision Tree (DT) classifier, Gradient Boosting (GB) classifier, Gaussian Naïve Bayes (GNB), Multinomial Naïve Bayes (MNB) classifier, Support Vector Machine (SVM), K-Nearest Neighbors (KNN) classifier, and OneVsRest classifier with Logistic Regression for multi-class classification, to classify the protein family based on the available features and target. As we mentioned in the exploratory data analysis section of this report, the target of our dataset had 2684 different classes. However, after frequency distribution of the protein classes, we found that most of the classes contained negligibly small numbers of proteins. We kept only the classes which contained more than 1200 proteins. We found 15 classes of such frequencies of protein sequences. Therefore, our final set of data had 15 classes with 53799 samples. The model parameters and the model evaluations for each of the ML models have been linked to jupyter notebook here.

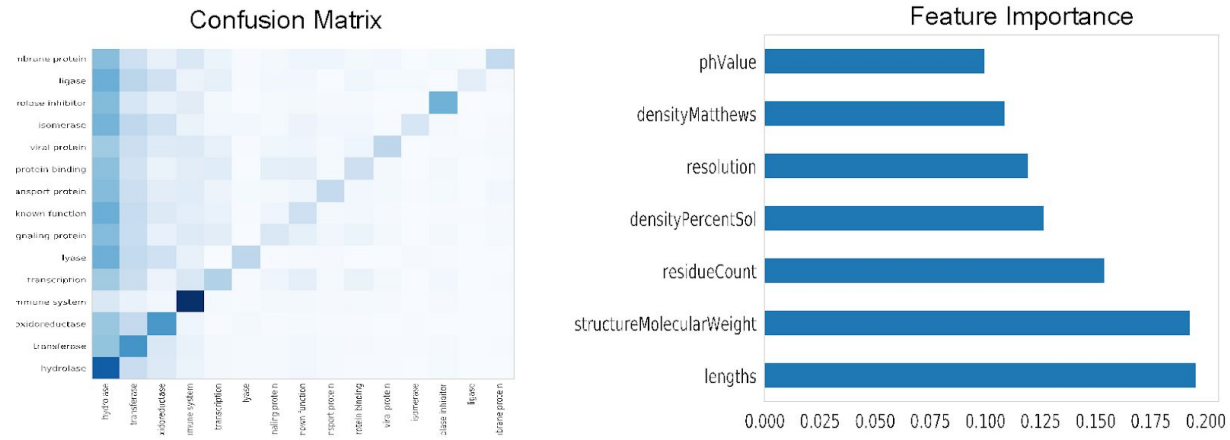
We applied tuned hyperparameters (by means of grid search cross validation) for each of the machine learning models and observed the test accuracy. After careful observation of the model outputs, we found that the ML models predicted the classes of proteins with accuracy less

than the random guess. The RF classifier performed better than other models. We presented the test accuracy obtained from various models in a table.

Models:	Test Accuracy
RF	0.40
DT	0.3
GB	0.22
GNB	0.12
MNB	0.09
SVM	0.15
KNN	0.32

Random Forest Classifier:

From the training and testing various machine learning models, RF classifier was found to be the best performer among ML predictors for the protein classification. We analyzed the classification output by calculating the confusion matrix and the feature importance. Figure 6 shows the confusion matrix in the heatmap. The darker color shows the higher score. From the heatmap, one can observe that most of the proteins were classified as 'hydrolase' (see the vertical axis corresponding to the 'hydrolase' along the horizontal axis). From the feature importance graph for RF classifier model, we didn't find significantly higher importance of any features, but the length of the sequence, molecular weights and residue counts were slightly more important than others.



Deep Learning Models:

Due to lack of a large number of features, the ML models were unable to precisely classify the protein classes available in the database. Therefore, we implemented DL models for

better predicting the protein classes. Protein sequences were taken as the input and we extracted features from the sequences. As we mentioned in the introductory section, protein is a random sequence of its 20 common building blocks and 5 uncommon. For each sequence, we extracted the maximum possible number of features by permutation/combination methods. For that purpose, we applied a bag of word vectorization technique (tf-idf) from Scikit-learn library. This method has been widely used to vectorize the texts (text mining) in Natural Language Processing (NLP). After converting the characters of the sequences to the numerical values, we subjected those vectors as inputs to the DL models for multi-class classification of protein families. In this project, we implemented 1 dimensional convolutional neural network (CNN) and bidirectional recurrent neural network (RNN) models. After careful tuning the layer parameters of the neural network, we observed that the one-dimensional CNN model outperformed over the RNN model.

Feature Extraction and target variable:

As we mentioned earlier, we converted the characters to numeric vectors by using 'tfidf' from 'Scikit-learn'. Each character representing the amino acids in the protein sequence was converted to the numerical feature vectors/matrices which then applied as input to the neural network.

Tf-idf (n-gram vectorizer):

It is well known that every element of the protein sequence is a character from the vocabulary of 26 alphabets. Each alphabet can be converted into numerical value on the basis of frequency of the character appearing in the whole database. According to this algorithm, the most common character will have small weight.

The protein family is generally based on the protein structure and their functionality. The structure of the protein is determined by the frequency of a group of amino acid units appearing in the sequence. In this project we counted all possible combinations of 3 adjacent amino acids.

During the process of converting the protein sequences to the vectors of equal size, "Tfidf" collected all the possible unique combinations of 3 adjacent letters from the whole datasets (that was counted 8944). For each sequence, it assigned an 8944-sized template and filled the normalized frequency of the trigram if found in the corresponding sequence and filled 0.0 otherwise. Thus, we found the vectors for each sequence were of equal size 8944. Each vector element was considered as a feature.

After getting the vector from the sequence, we concatenated the seven general features used to the ML models to which increased the number of total features equal to 8951. Classes of the protein family as target variables were converted to the numerical label as shown in the previous section (EDA).

Convolutional Neural Network:

Although convolutional neural networks have been famous in computer vision problems, one-dimensional convolutional networks can be used efficiently in the sequence detection problems. We applied a convolution filter of size equal to 3 amino acid units that scanned across the sequence and produced the activation across that sequence. After careful tuning of hyper parameters, we confirmed a simple single layer convolutional neural network with batch

normalization and dropout performed the best. The network architecture used in the model and their evaluations have been shown in figures 7 and 8.

Network Architecture:

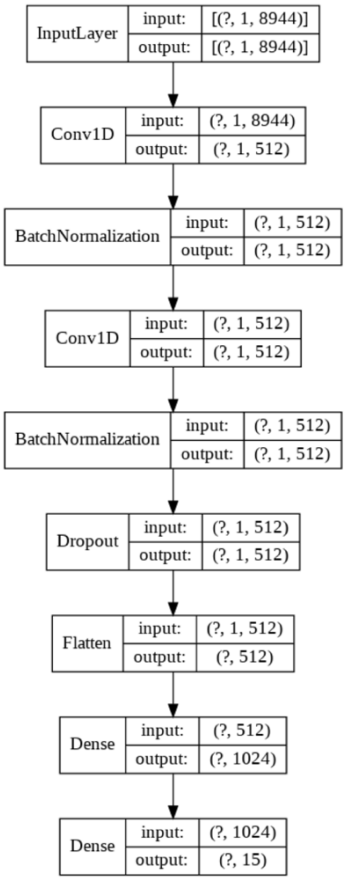


Figure 7: The one-dimensional convolutional network.

Model Evaluation:

The performance of the network was evaluated by plotting the accuracy and loss as a function of epochs. From the plots (figure 8), we found that the training accuracy increased very fast and remained constant at its maximum value of 0.96 whereas the test accuracy increased slowly and remained constant at around 0.67. During the training up to 20 epochs, the loss function for the validation and train data decreased to the values 1.6 and 0.4, respectively. Confusion matrices for the train and test set of data have been shown in figure 9.

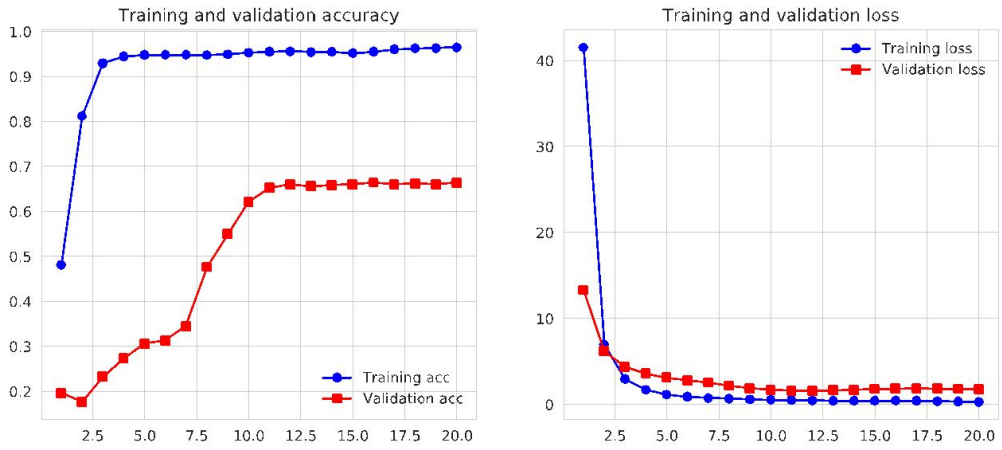


Figure 8: Accuracy and loss for train and validation data sets plotted against the epoch times.

Figure 9: Confusion matrices for the train and test data sets. The classes have been shown in the numerical labels. The corresponding protein families can be identified from the class dictionary given in the EDA section.

The model parameters and the used codes can be linked to the jupyter notebook here.

Bidirectional LSTM:

We used a bidirectional long short-term memory model based on the RNN. RNN has been popular to detect sequence-based problems such as speech recognition, language translation. LSTM is the most popular gated version of RNN model which keeps the memory of units far from the current unit in the sequence and forgets the unwanted units. To our knowledge, the confirmations of structured proteins are somehow related to the correlation of the units at various positions of the sequence. Therefore, we hypothesized LSTM as the best model to recognize the structure of proteins. In LSTM, output of the previous unit is generally used to feed to the next step in the forward direction. The sequential propagation with either exploding or diminishing gradient becomes problematic during optimization of unidirectional LSTM which might crash the model. The efficient way of removing such an effect is to propagate the gradient from both directions as in BiLSTM. Therefore, we chose the BiLSTM model.

Network Architecture:

Figure 10: Bidirectional LSTM model architecture.

Model Evaluation:

From the accuracy and loss plots below, it can be observed that the validation accuracy increases faster at early steps and remains saturated at the lower values ~0.55. The training

accuracy increases up to ~0.7. Similarly, the training loss decreased from large value at the beginning and went down to ~0.4 for our maximum epoch limit (20). The validation loss remains almost constant at ~1.8 throughout the training. This model, although being more complex than the one-dimensional CNN, performs poorly. However, one can observe the smaller gap between the train and test accuracy in the case of the BiLSTM model. We plotted a confusion matrix in the heatmap with scores in the color bar.

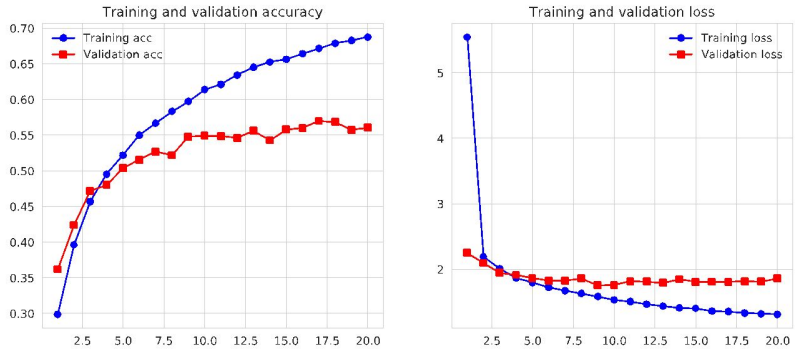


Figure 11: Train and test accuracy and loss plotted against epoch times.

Figure 12: Confusion matrix from the BiLSTM models for the 15 classes of protein families. The classes were represented by numerical labels; the corresponding protein families can be found in the class label dictionary provided in the EDA section.

The code used and the parameters used for the model evaluation can be linked to jupyter notebook here.

Conclusions:

We implemented ML and DL models to classify the protein family for the database extracted from the protein data bank. We applied the macroscopic features, such as molecular weights, length of amino acid units, total residue counts in each sequence, Ph-value, and molecular densities to the ML models for the classification process. Due to the limited features, the performance of any of the models were very poor (~0.4 for RF classifier, the best of all other ML models). This poor performance demanded the feature extraction from the protein sequence (sequence of 25 alphabets). We extracted features from protein sequences by implementing the methods used in text mining in natural language processing (NLP) such as, n-gram vectorization (we used 3-gram). We applied a huge set of features to the DL models which were able to predict the protein families with training accuracies up to 96%. Although we tuned the model parameters extensively, the gap between the train/test accuracy was very large (about 0.3). For the given dataset of protein sequences, designing model architectures that reduce the gap between the train/test accuracy can be the best subject for the future projects.