Predicting Protein Residue-Residue Contacts

Reshma Sri Challa
Computer Science
University of Ottawa
Ottawa, Canada
rchal050@uottawa.ca

Abstract. Residue-residue contact prediction is predicting which amino acid residues in the structure of a protein are "in contact". Typically, residues are defined to be in contact when the distance between their α-carbon atoms is smaller than 8Å. Usually, contact prediction methods are structured as either sequence-based or template-based. Sequence-based contact prediction normally uses machine learning methods. This project comprises of residue-residue contact prediction using machine learning models - random forest and logistic regression classifier. The primary goal of contact prediction here is typically to produce predictions that correctly label a pair of amino acids in a protein as "in contact" or "not in contact". The ability to predict which pairs of amino acid residues in a protein are in contact with each other offers many advantages for various areas of research that focus on proteins. There have been several approaches to achieve this, but we research on defining one such framework that will predict the contact using one hot encoding obtained from the window of amino acids and contact maps.

1 Introduction

Protein contacts contain crucial information for the understanding of protein structure and function and therefore, contact prediction from sequence is a significant problem. Recently exciting progress has been made on this problem. It has been approached by many researches with various techniques, we will try with the information around each protein, that is, considering a window of sequence before and after the amino acid in each pair of amino acids.

Predicting protein contacts also has wide applications in the field of bioinformatics. Predicted contact pairs are used for fold prediction, as a primary constraint for molecular modelling. As contact prediction has progressed it has become more widespread for 3D structure predictors to combine contact prediction into structure building and even to help discover functionally important regions of proteins. These predictions are becoming particularly significant given the relatively low number of experimentally determined protein structures compared to the quantity of accessible protein sequence data.

Our major focus here in this project is "if the information from the sequence of amino acids positioned around an amino acid in the pair of amino acid can influence the contact of pair of amino acids". This was inspired from [2], which is the base paper for this project. We define a framework to test using one hot encoding applying on a window of size 10. There are many publicly available PDB files, which can provide us with lots of data for this problem, but we use only a few files to define if this technique is valuable or not.

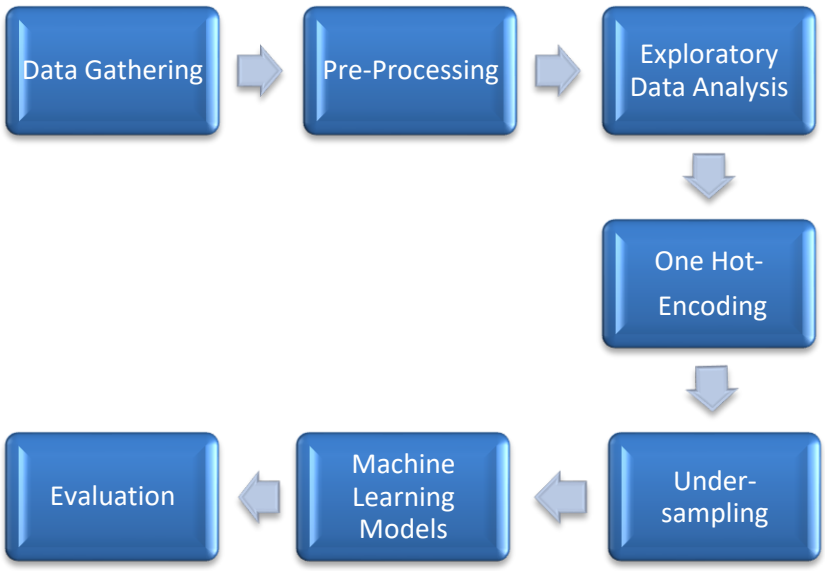

Figure 1: Framework

2 Dataset

Dataset used here is composed of X-ray determined protein structures of the Protein Data Bank [3]. It provides the structural information of proteins, nucleic acids, and complex assemblies that help us understand all aspects of biomedicine from protein synthesis to health and disease. The RCSB PDB curates and annotates PDB data.We download the PDB files of proteins from Protein Data Bank. We have listed the name of all the PDB files in a text file in the local folder. Every file is named as "https://files.rcsb.org/download/" plus the PDB file, as this is the https services they provide. For example, the link for PDB file named "4fek" is https://files.rcsb.org/download/4fek. Using this information, we downloaded the files using "urllib" and unzipped using "gzip" libraries in python into a local folder named train dataset and test dataset for both training and testing. We use 50 PDB files for train and 2 PDB files for test with 3369431 and 40410 amino acid pairs respectively.

Table 1: Feature Description

Feature Name	Description	Value
Label (classification)	α-carbon distance < 8Å	Contact
	α-carbon distance > 8Å	No Contact
Range	| j-i | <10 Å	Short
	10 <= | j-i | < 20 Å	Medium
	| j-i | > 20 Å	Long

3 Preprocessing

We Parse the PDB file using PDB parser in python to find the position of amino acids in the protein ,calculate the possible amino acid residue pairs and calculate the distance between Cα atoms in each pair of residues ,that is, amino acids in each PDB file and if the distance is less than 8 Angstroms we name it "Contact" else "No Contact". This is done using Contact Maps. A contact map is usually defined as a binary matrix with the rows *i* and columns *j* representing the residues of two different chains. Each matrix element *(i, j)* is one/True if the α-carbon distance between the associated residues is less than some threshold (8 angstroms), or zero/False if the residues are far away. This will be calculated using the Bio.PDB in python. We generate the target value for each pair of amino acids for each PDB file from its contact map, for the values true and false we label "Contact" and "No Contact" respectively, which is clearly represented in table 1. This generates a binary classification problem. We calculate another feature named range, describing how far these elements at *(i, j)* are in the sequence, which is clearly mentioned in table 1.This feature is used to check if contact depends on the range of position of both amino acid in the molecular structure of the protein.

For each value of *i* and *j* in the contact map , we find the amino acid at these positions and extract the window of 10 amino acids around the amino acid at these positions ,centering the amino acid at positions *i* and *j*. Now let's understand this in detail, for each amino acid in the pair of amino acids *(i,j),* let us consider amino acid at position *i* in the residue, we run a window of size 10 around each amino acid and get the first letter of each amino acids which are until 5 positions before and 5 positions after the required amino acid at position *i*. In our data we call this residue_one_window_10 and residue_two_window_10 for amino acid at position *j*. We can clearly observe that amino acids at the first few positions of residue in PDB file do not have 5 amino acids before the amino acid at position *i*, so we use padding here. To maintain all the sequence window of the same length, we add padding ("_") for each amino acid at position *i* to make it of length 10 and also same repeats for each amino acid at *j*. We get a string of sequence of length 10 of each amino acid at position *i* and *j*, now we combine this to get a collective sequence containing the information for the pair of amino acid *(i,j)* of length 20. In our data we call this Window_10. In the coming sections we will refer this to feature Window_10. On this we apply one hot encoding to get the feature vector. We have repeated the same process for window size of 20 (sequence of length 40) and window of size 30 (sequence of length 60) as well and named then Window_20 and Window_30 respectively.

4 Exploratory Data Analysis

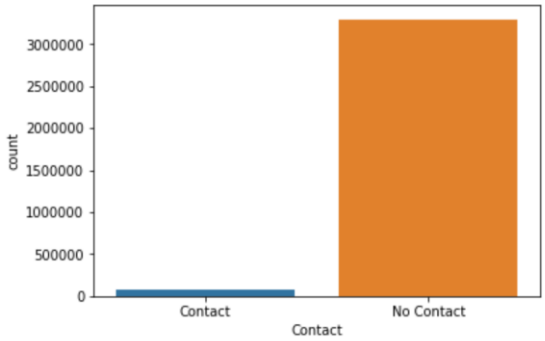

Figure 2: Contact label in train data

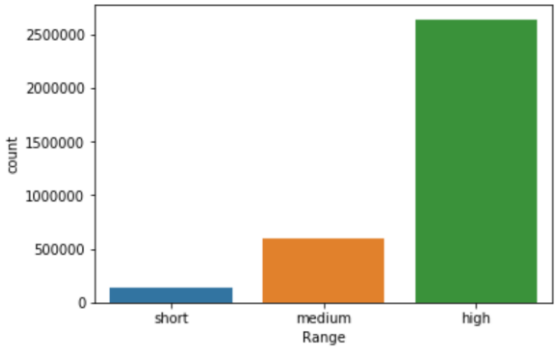

Figure 3: Range label in train data

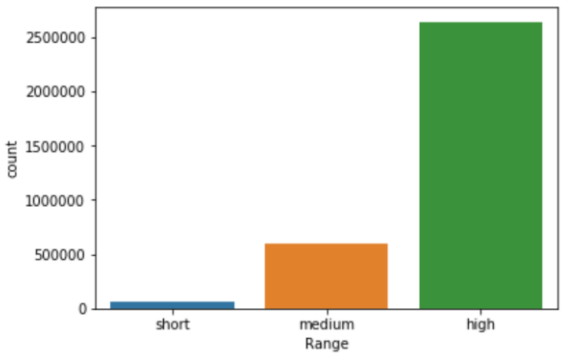

Figure 4: Range label with no contact in train data

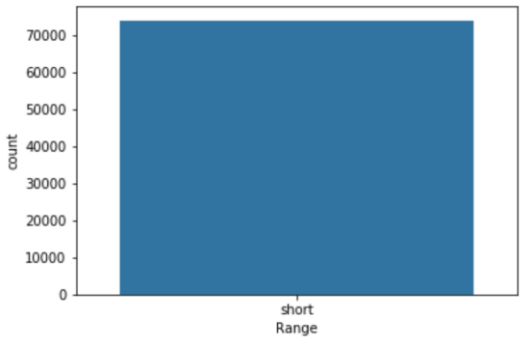

Figure 5: Range label with contact in train data

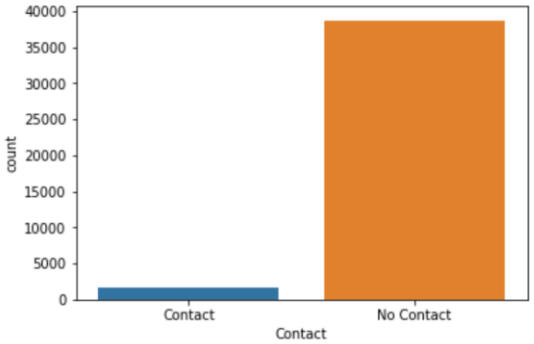

Figure 6: Contact label in test data

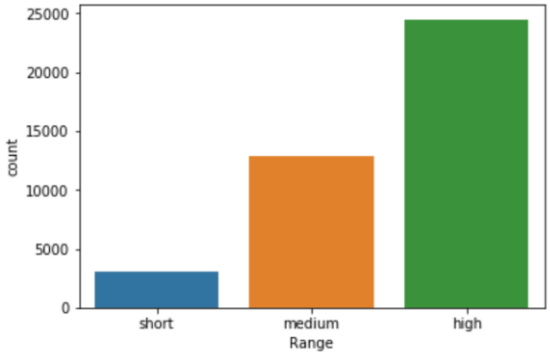

Figure 7: Range label in test data

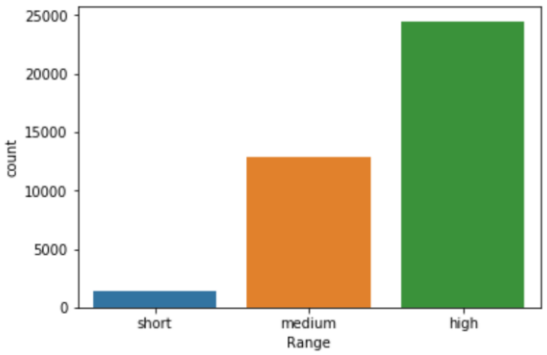

Figure 8: Range label with no contact in test data

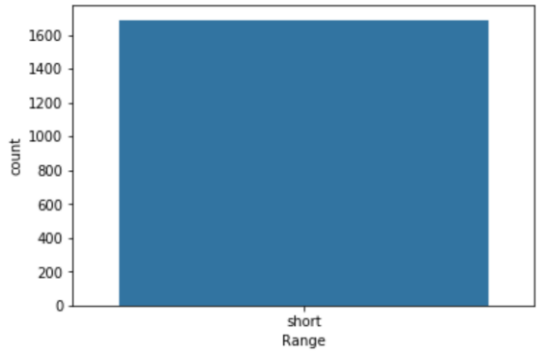

Figure 9: Range label with contact in test data

The above figures clearly depict that there is a clear imbalance in the data, also the amino acid pairs in contact and no contact belong to short range but only no contact amino acid pairs belong to medium and high range.

5 Methods

5.1 Under Sampling

It is ideal to apply under-sampling on train data but not on test data. From exploratory analysis on train data we can conclude that label no-contact contains 3295496 instances while label contact contains only 73935 instances. This is an example of Binary (as there are 2 labels, 0 & 1) Imbalanced Data. A simple under-

sampling technique is to under-sample the majority class ,class 0, randomly and uniformly, was introduced into this framework to reduce imbalance to a great extent. After under sampling using RandomUnderSampler in python our data reduces to size 147870,which consists of 73935 of contact and no contact labels.

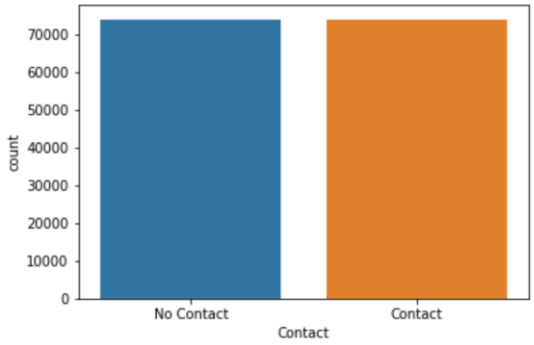

Figure 10: Contact label after Undersampling in train data

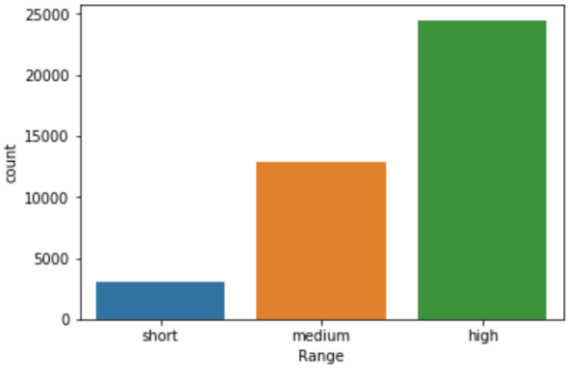

Figure 11: Range label after Undersampling in train data

5.2 One Hot Encoding Vector

We have started the implementation using the one hot encoding in scikit learn but we met with some challenges which will be listed in the discussion section in detail, instead we created a one hot encoding vector of our own. We create a one hot encoding vector for each amino acid with respect to the naturally occurring 20 amino acids and in our case also including padding which makes it 21 so each amino acid, represented as a vector of size 21. Our feature Window_10 is of length 20 and each amino acid will be represented as length of 21, which makes each instance of feature

to be represented as an array of shape 420.This feature is to be reshaped before it is passed into our models therefore we use a function called numpy.row_stack in python to create a single array of size 147870 ,as this stacks each row of our feature into a single array. The total number of features are 147870 therefore shape of our feature is (147870,420). Since this iterates our feature Window_10 , an array of size 147870, one by one increasing our processing time. So, our primary step was under sampling our data and then creating a single array of the feature vector. We have repeated the steps for features Window_20 with shape of (147870,840) and Window_30 with shape of (147870,1260).

5.3 Machine Learning Models

Here, we are trying to predict the contact label which is a binary classification problem ,that is, if the pair of amino acids are in contact or not. Classification problem is a Supervised Learning technique. In supervised learning, an algorithm is used to learn the mapping function from the input variable (x) to the output variable (y) i,e. $y = f(x)$. Here the classifies we will use are Random forest Classifier and Decision Tree. Evaluation of these models is based on the accuracy, F1 score(the harmonic mean of precision and recall) and area under the curve, these are produced from the classification report in scikit learn. These are discussed below in detail. Confusion matrix depicts the true positives, true negatives, false positives and false negatives as shown in the table 2.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Where,

$$True\ Negative\ Rate = TNR = \frac{TN}{TN+FP} \quad (2)$$

Table 2: Confusion Matrix for classification

	Predicted(+)	Preddicted(-)
Actual (POS)	TP	FN
Actual (NEG)	FP	TN

$$precision = \frac{TP}{TP+FP} \quad (3)$$

$$True\ Positive\ Rate/Recall = TPR = \frac{TP}{TP+FN} \quad (4)$$

$$F1\ Score = \left(\frac{\frac{1}{recall} + \frac{1}{precision}}{2}\right)^{-1} \quad (5)$$

AUC - ROC curve is a performance measurement for classification problem. ROC is a probability curve and AUC signifies the degree or measure of separability. It tells how much our model can differentiate between classes. Higher the AUC, better the model is at predicting class 0 as 0 and class 1 as 1 for a binary classification problem. The ROC curve is plotted with TPR on y-axis and FPR on the x-axis. In our case, higher the AUC, better the model is at distinguishing between amino acid pairs with contact and no contact.

6 Results

Our intial approch was only on two PDB files with no undersampling and got an accurracy of 96% with only feature Window_10 using random forest and when we added range as another feature, we got an accuracy of 98%, this was for a total of 128911 instances and 2969 contacts among them. We tried projecting this problem on 330 PDB files, the processing time was around 11 hrs and while saving(a single array of feature vector) it resulted in memory issues in the browser.We did try with 100 PDB files but faced the same issue again so settled for 50 PDB files. All this was executed with a windows laptop (core i7) of 16 GB RAM and a 4 GB graphics card. 50 PDB files contained 3369431 instances among which 73935 are not in contact and 73935 are in contact. The count of PDB files might be less but the amino acid pairs (instances) for each PBD file are very high, so we have good amount of data as well,even with high data imbalance. After undersampling this data we get 147870 instances among which 73935 are in contact and 73935 are not in contact. Then the results of random forest and logistic Regression are discussed in table 3 and the below figures.

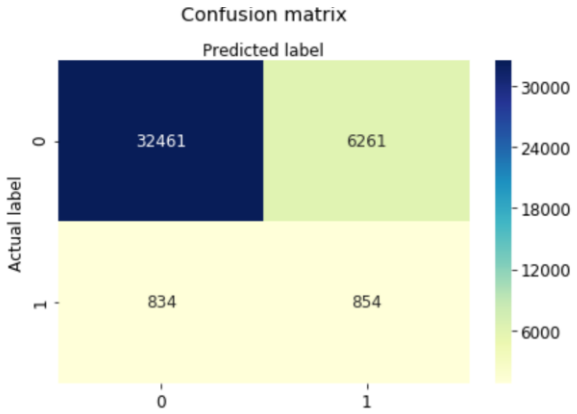

Figure 12:Confusion Matrix for Random Forest ,Window Size 10

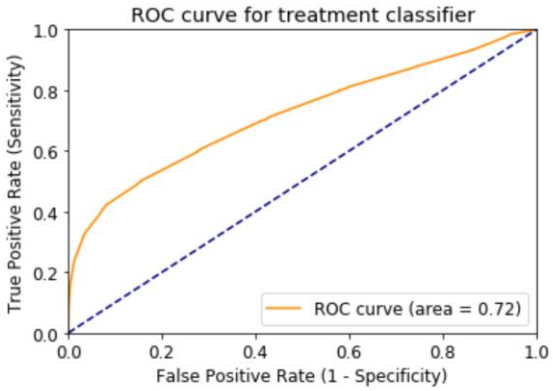

Figure 13: ROC Curve for Random Forest ,Window Size 10

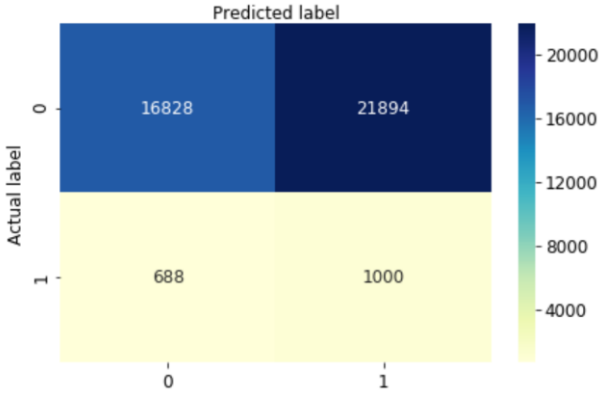

Figure 14:Confusion Matrix for Logistic Regression,Window Size 10

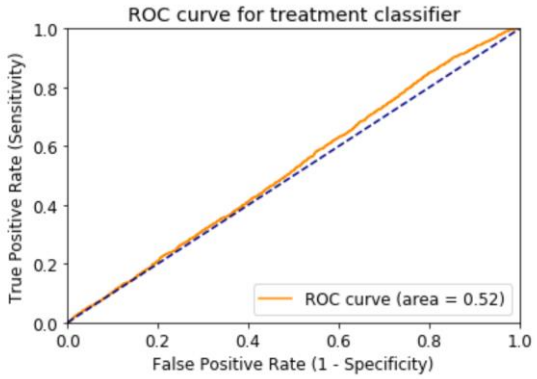

Figure 15: ROC Curve for Logistic Regression,Window Size 10

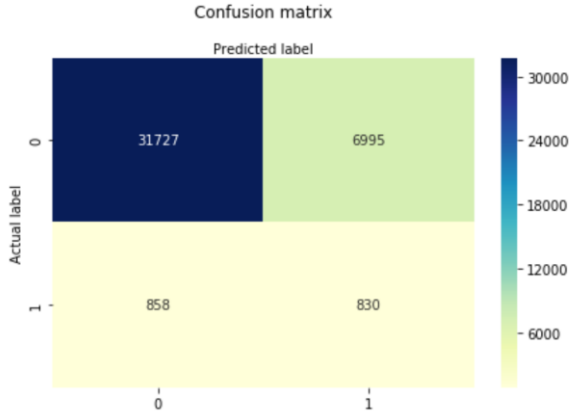

Figure 16:Confusion Matrix for Random Forest ,Window Size 20

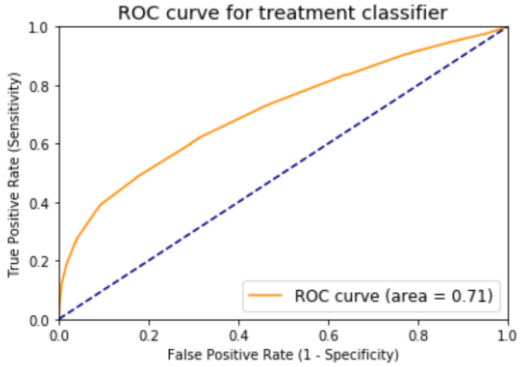

Figure 17: ROC Curve for Random Forest,Window Size 20

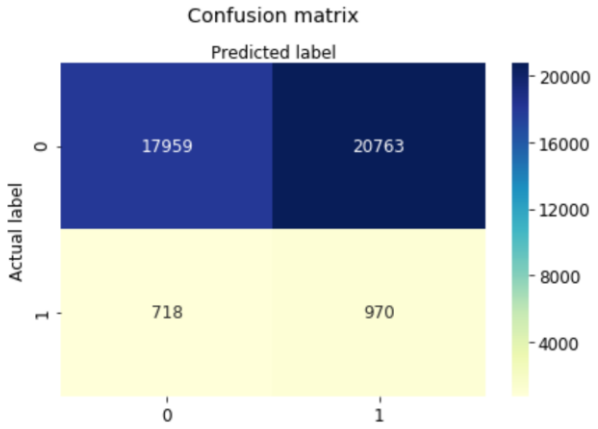

Figure 18:Confusion Matrix for Logistic Regression ,Window Size 20

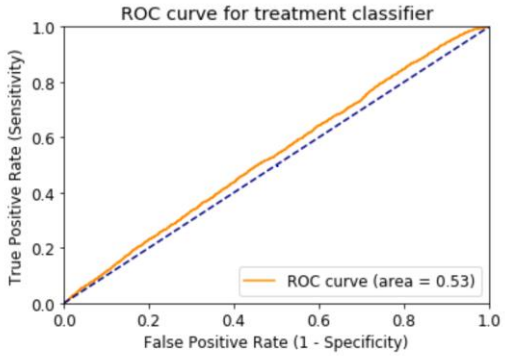

Figure 19: ROC Curve for Logistic Regression,Window Size 20

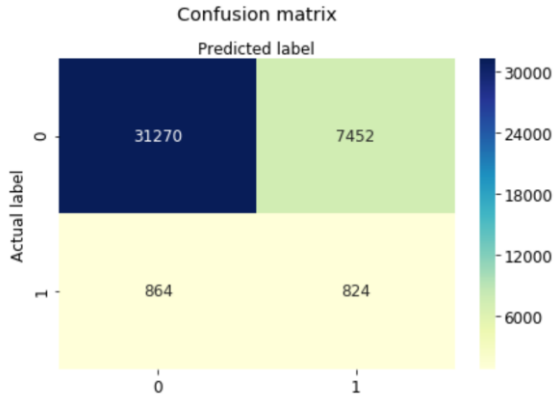

Figure 20:Confusion Matrix for Random Forest ,Window Size 30

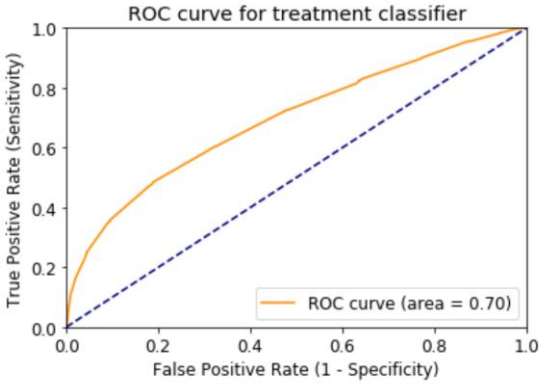

Figure 21: ROC Curve for Random Forest ,Window Size 30

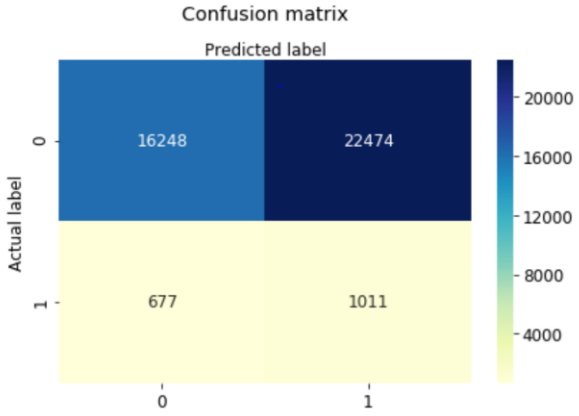

Figure 22:Confusion Matrix for Logistic Regression,Window Size 30

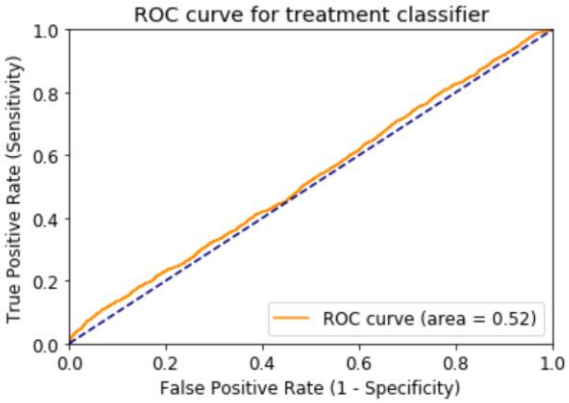

Figure 23: ROC Curve for Logistic Regression,Window Size 30

Table 3: Evaluation of Classifiers

Classifier	Window_10			Window_20			Window_30		
Random forest	0.82	0.87	0.72	0.80	0.86	0.71	0.79	0.85	0.70
Desicion Tree	0.44	0.58	0.52	0.46	0.60	0.53	0.42	0.56	0.52

Using the feature, range label with each window size, produced around 96% accuracy for both logistic regression and random forest and all window sizes ,my observation here is since we obtained range label from the distance, this contains direct information of the contact label, we also have stated in exploratory analysis that range label of medium and high do not have any contact. So, considering the range label as a fetaure is not a good idea.We did not add these results here as this was not a valid metric (can refer to the code pdf in the submission if reqiuired).

From table 3, we can distinctly potray that Window_10 has the highest accuracy for each classifier, this proves that the window of size 10 determines the most contacts in amino acids pairs. Also we can see that Random forest has better accuracy and ROC than Decision Tree , therefore random forest is better suitable for predicting protein residue-residue contact. This is what was used in the base paper and has similar results as in the paper.All these results are with no under sampling on test data ,which mean it is higly imbalanced as shown in figures 6,7.

7 Discussion

In this section we will discuss the challenges faced and the future work for our framework.

The biggest challenge here is the relationship of time and data. For each PDB file we have around 800 amino acids which makes it 319600 amino acid pairs, but only a countable are in contact as the distance between most of them is greater than 10, that is they belong to either medium or high range. We did handle this class imbalance problem with under sampling but we picked up only 50 PDB files for training and 2 PDB files for testing as this had a processing time of 4 hrs for each window size and choosing more data resulted in memory issues and crashing. Another important challenge to discuss is the one hot encoding in scikit learn, this created a sparse matrix for each instance and its result was an array of sparse matrixes which was not in the right shape for the input into machine learning models. So, we created our own simple one hot encoding method, which resulted in a sparse array for each instance of the feature. The future work would be using this

approach on a higher number of PDB files (typically all available files) and also with other classifiers, maybe be deep learning approaches.

8 Conclusion

Our early focus for this project was clear, we tackled some challenges during the project but after good research and Professor's guidance we conclude with a framework that can predict the protein residue-reside contacts using amino acid sequence information around each amino acid, for which the best would be using window of size 10 around the amino acids and classifier Random Forest, which had the best results in our case. Further we can experiment with more classifiers and also with more data that is more PDB files for a deep research on this problem.

References

1. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov
2. Joseph Luttrell, 4th, Tong Liu, Chaoyang Zhang, and Zheng Wang. Predicting protein residue-residue contacts using random forests and deep networks. BMC Bioinformatics, 20(Suppl 2):100, Mar 2019
3. https://PDB101.rcsb.org/learn/guide-to-understanding-PDB-data/primary-sequences-and-the-PDB-formatb
4. The PDB module, biopython-cn.readthedocs
5. biopython.org
6. Biostars
7. warwick.ac.uk