Predicting the Subcellular Locations of Eukaryotic Proteins

Kamakshi Bansal
16114486
kamakshi.bansal.16@ucl.ac.uk
University College London.

***Abstract*—To have the better understanding about protein and amino acid sequences, knowing protein subcellular location is crucial. It might give some clue about it's possible function. Powerful machine learning tools such as Logistic Regression, AdaBoost Classifier, XGBoost, Support Vector Machine, Neural Network and Random forest have been used to automate the process of assigning proteins to a certain subcellular location i.e classifying them into four subcellular locations - Cytosolic, Secreted, Nuclear, Mitochondrial.**

This paper is evaluating the performance of each model and finally ensembling them using Voting Classifier to reach the best accuracy of 67.92% . The source code can be found here.

I. Introduction

In recent years, there has been a lot of progress made in human genomes. More than 3000 genomes sequences has been discovered until now. Understanding these sequences on such vast dataset is very difficult for humans to decipher by naked eye. As well as it is very important to understand these sequences in order to perform ground breaking research in protein structures and build useful applications. The paper focuses on the eukaryotic proteins and predicting its subcellular location.

In this paper, I have implemented various machine learning classification techniques and made a comparison between each of their predictions. Classification algorithms like AdaBoost, Random Forest, Logistic Regression, XGBoost, SVM and Neural Networks have been used to train models. I have also performed 10-fold cross validation to combat overfitting in logistic regression along with L1 regularization. At the end I ensemble the predictions made out of all these classifiers and make a final prediction on the test set. The reason why these machine learning algorithms are used is that it is the most simple machine learning algorithm yet powerful and robust.

II. Data Set

Protein molecules are made of string of amino acids in a particular order. This string is called an amino acid sequence. Protein sequence is typically notated as a string of letters, listing the amino acids starting at the amino-terminal end through to the carboxyl-terminal end [9]. A single letter code is used to represent the 20 majorly occurring amino acids in the given data set along with other three amino acids. The given data set consist of amino acids shown in Table I. 9215 amino acid sequences from eukaryotic organisms in FASTA format are used as training and validating data. Out of 9215, 3002 sequences are the sequences taken from Cytosolic, 1296 from Mitochondrial, 1604 from Secreted, and 3313 from Nucleus.

TABLE I
Amino Acid in the given data set

Abbreviation	letter abbreviation	Amino acid name
Ala	A	Alanine
Arg	R	Arginine
Asn	N	Asparagine
Asp	D	Aspartic acid
Cys	C	Cysteine
Gln	Q	Glutamine
Glu	E	Glutamic acid
Gly	G	Glycine
His	H	Histidine
Ile	I	Isoleucine
Leu	L	Leucine
Lys	K	Lysine
Met	M	Methionine
Phe	F	Phenylalanine
Pro	P	Proline
Ser	S	Serine
Sec	U	Selenocysteine
Thr	T	Threonine
Trp	W	Tryptophan
Tyr	Y	Tyrosine
Val	V	Valine
Asx	B	Aspartic acid
Xaa	X	Any amino acid

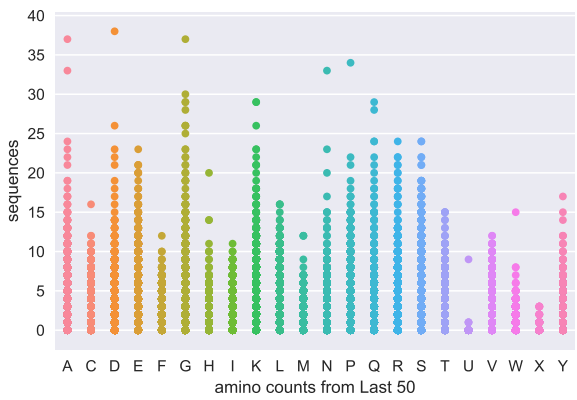

Fig. 1. Distribution of each amino acid for each sequence from last 50 for the whole data set.

III. Feature Extraction and Data Pre-processing

The data in its current form, is not very usable by an algorithm. Algorithms typically dont understand English letters i.e amino acid sequences. We need to design a model which precisely transforms the raw input into a form (features) that can be processed by an algorithm. To be more specific, we need an approach that will be flexible enough to handle potentially multiple algorithms. Some algorithms work better for some problems, and the only way to know is to try it out. We want an approach that enables us to plug in different algorithms, and compare their respective results.

Hence, to extract useful features, first of all each of the sequence from The fasta Format file is stored in the form of a list and it's labels are stored in another list. Then the features are extracted from each amino acid sequence and stored in the form of dictionary before vectorizing it. Basically, the training data is made in the form of list of dictionaries of features and then encoded to a vector.

Figure 1 represents the distribution of amino acid counts from last 50 for the whole of the training data set. Similarly, figure 2 and figure 3 represents the distribution of amino acid counts from top 50 and for the entire sequence respectively.

While evaluating the molecular weights of the sequences. the molecular weight of X is considered as 0 as it does not affect the feature value that much. Amino acid sequences are encoded to 69-dimensional feature vector. The details of feature vectors are shown in Table II. [8]

TABLE II
Detailed Description of Features used

Feature	Description	Bins
Isolelectric Point	Isolelectric Point of the amino acid sequence	1
Amino counts from Last 50	Count of 20 amino acids of the last 50 aminos	20
Amino counts from first 50	Count of 20 amino acids of the first 50 aminos	20
Amino counts	Count of 20 amino acids of the whole sequence	20
Molecular Weight	Molecular Weight of the amino acid sequence	1
Sequence Length	Length of the amino acid sequence	1

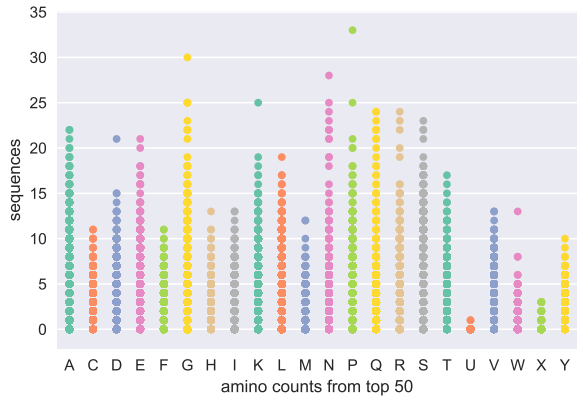

Fig. 2. Distribution of each amino acid for each sequence from top 50 for the whole data set.

Fig. 3. Distribution of each amino acid for each sequence for the whole data set.

IV. Models Used

A. Logistic Regression and Cross Validation

Logistic regression is a discriminative classifier that classifies an observation into two or more classes. It predicts the probability with more than two possible discrete outcomes i.e. for the given set of data,it gives the probabilities of the different possible outcomes of a categorically distributed dependent variable. It predicts the probability with the help of set of features extracted from the data and combining it linearly i.e multiplying each feature by some weight and then adding them up and then applying logistic function.Performing 10 fold cross validation made the accuracy score of 60.75%.

Logistic Function is :

$$f(x) = \frac{1}{1 + \exp(-x)}$$

Logistic regression works best with the binary data. Logistic regression is giving 62.31% accuracy for the given data set.

B. Support Vector Machine

Support vector machines is similar to logistic regression i.e. it is a classification algorithm but unlike logistic regression which provides probabilities, SVM is used to predicts the class identity using a linear function . If $W^T X + B$ is positive, SVM predicts the positive class else if $W^T X + B$ negative, it predicts the negative class. The key innovation associated with SVM is kernel trick. The kernel observes that many machine learning algorithm can be written exclusively as the dot product of examples. For eg, the linear function used by SVM can be re-written as :

$$W^T X + B = B + \sum_{i=1}^{m} \alpha_i x^T x^{(i)}$$

Where $x^{(i)}$ is the training example and α_i is the vector of coefficients. Rewriting the learning algorithm this way allows

us to replace x by the output of a given feature function $\phi(x)$ and the dot product with a function $k(x, x(i)) = \phi(x).\phi(x(i))$ called a kernel. The kernel-based function is exactly equivalent to preprocessing the data by applying $\phi(x)$ to all inputs, then learning a linear model in the new transformed space. For the given data set, it is giving the best accuracy of 65.38% [7]

C. AdaBoost Classifier

Boosting is an approach of machine learning to create a highly accurate prediction rule combining many weak learning rules. For example, we are studying for an exam using the past exam paper and we grade ourselves. We will tend to pay less attention to the questions we got right and study more for the questions we did wrong in order to get them correct in our next attempt i.e we are giving more weightage to the questions we did wrong and less weightage to the questions we solved correctly .We then reattempt the paper and grade ourselves again, we keep on repeating the process until we master that exam paper. The similar idea is used in Adaboost.
In adaboost, first of all each example is initialized with uniform distribution. Then a weak learner is trained on this weighted data. The weighted error rate with respect to the misclassified points is calculated which is used to determine the adaptive parameter α which controls how important weak learner classifier.

$$\alpha_t = \log\left(\frac{1 - e_t}{e_t}\right)$$

If the error e_t is is less than 50%, then $\alpha_t > 0$. After α is computed, the weight distribution is updated for the next iteration.
The examples which are correctly classified have their weights decreased multiplicatively and and the examples which are incorrectly classified have their weight increased multiplicatively.The final classifier returned by AdaBoost is a weighted vote of the individual classifiers, with weights given by the adaptive parameters. Adaboost Classifier is giving 60.75% accuracy for the given data set.

D. Random Forest

In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This strategy performs better than Logistic Regression, Adaboost Classifier, neural networks turns and many other classifiers. Random Forest is also robust against overfitting . In addition, it is very user-friendly in the sense that it has only two parameters (the number of variables in the random subset at each node and the number of trees in the forest), and is usually not very sensitive to their values. [1]
Random forest is giving 64.93% accuracy for the given data set.

E. Neural Networks

Neural Networks is a powerful machine learning tool that attempts to mimic the learning pattern of natural biological neural networks. Biological neural networks have interconnected neurons with dendrites that receive inputs, then based on these inputs they produce an output signal through an axon to another neuron. Perceptron is the most basic form of neural network which has one or more inputs, a bias, an activation function, and a single output. [2] Class MLPClassifier form nonlinear discriminant functions using multi-layer perceptrons with sigmoidal nonlinearities. They are trained with supervision, using gradient-descent training techniques, called backpropagation which minimize the squared error between the actual outputs of the network and the desired outputs. Patterns are applied to input nodes that have linear transfer functions.

It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. MLP trains on two arrays: array X which holds the training samples represented as floating point feature vectors; and array y of size, which holds the target values (class labels) for the training samples.MLPClassifier supports multi-class classification by applying Softmax as the output function.[3]
Neural Network is giving 63.73% accuracy for the given data set.

F. XGBoost Classifier

XGBoost is short for Extreme Gradient Boosting. XGBoost is used for supervised learning problems, where we use the training data (with multiple features) x_i to predict a target variable y_i. XGBoost is widely used among data scientist because of its execution speed and model performance. It's a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data.
The most important factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings [5]. It performs the best usually over any other machine learning tool. For the given data set, it is giving the best accuracy of 66.29%

G. Ensembling of Models using Voting Classifier

This is a simple and conservative approach of implementing a weighted majority rule ensemble classifier in scikit-learn that yielded remarkably good results i.e improved the accuracy to 67.92%.
The EnsembleVoteClassifier is a meta-classifier which combines similar or conceptually different machine learning classifiers for classification via majority or plurality voting. In the model, soft voting is used which predicts the class labels by averaging the class-probabilities. [4]

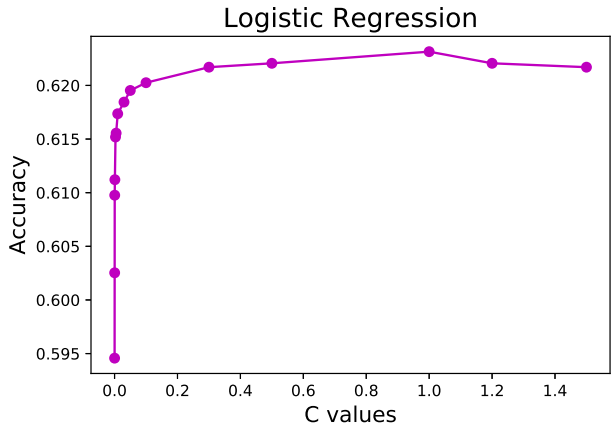

Fig. 4. Accuracy score on different C values.Logistic regression performs the best with C=1

V. Parameter Optimisation

Parameter optimisation is done for each and every model to find the parameters which gives the best accuracy score. From the graphs it can be seen that:

- Logistic regression performs the best with C=1.
- SVM performs the best with gamma=0.0001.
- XGboost performs the best with n_estimators=600 and learning rate=0.1.
- Adaboost Classifier performs the best with n_estimators=1000 and learning rate=0.01.
- Random Forest performs the best with n_estimators=2000.
- Neural Network performs the best with learning_rate_init=0.001.

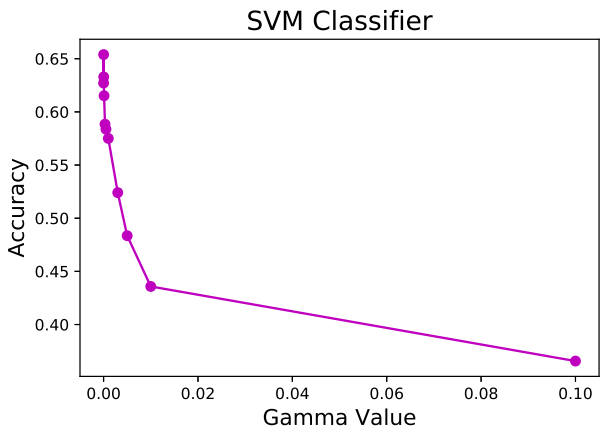

Fig. 5. Accuracy score on different C values.SVM performs the best with gamma=0.0001.

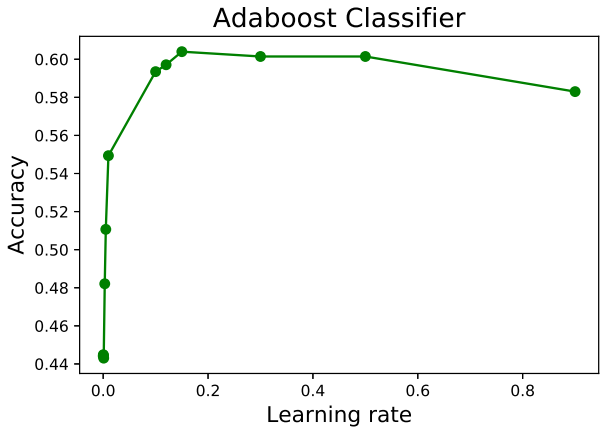

Fig. 6. Accuracy score on different Learning rate.Adaboost Classifier performs the best with n_estimators=1000 and learning rate=0.01.

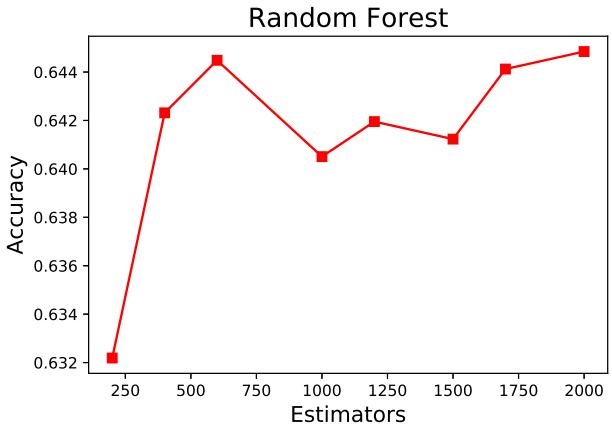

Fig. 7. Accuracy score on different values of estimators.Random Forest performs the best with n_estimators=2000.

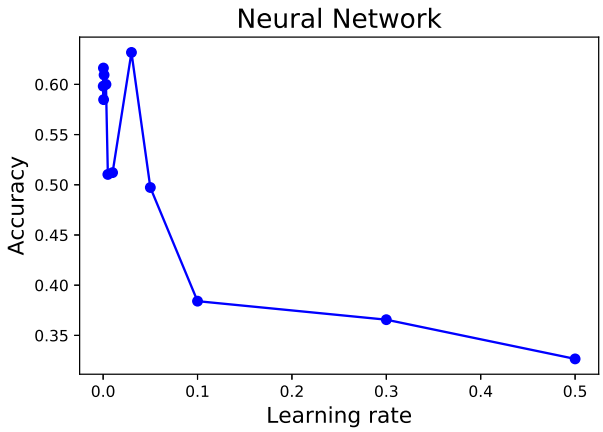

Fig. 8. Accuracy score on different Learning rate.Neural Network performs the best with learning_rate_init=0.001.

VI. Confusion Matrix

Confusion matrix is an error matrix which tells about the performance of machine learning algorithm with a table

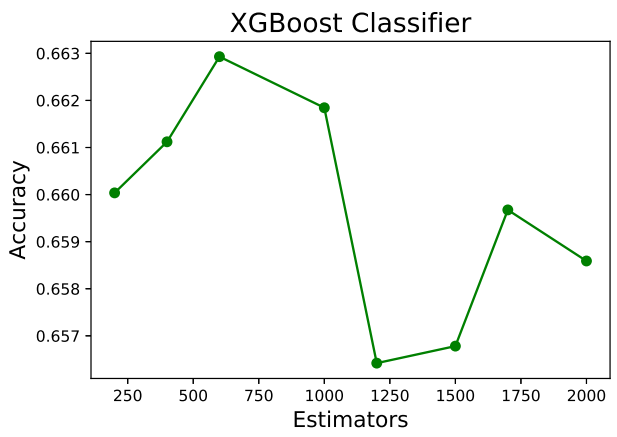

Fig. 9. Accuracy score on different values of estimators.XGboost performs the best with n_estimators=600 and learning rate=0.1.

layout, typically a supervised learning one. It is also known as matching matrix. It reports the number of false positives, false negatives, true positives, and true negatives. This gives more detailed analysis than mere proportion of correct guesses (accuracy). Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced [6]. The diagonal elements of the confusion matrix shows the elements which are correctly classified. Confusion Matrix is plotted on Validation set.

Logistic Regression classifies more number of sequences correctly for mito and secreted but performs poorly for the classification of cyto as compared to other models whereas Adaboost classifies the best for nucleus. Similarly, from the confusion matrices it can be seen XGBoost classifies the best for cyto.

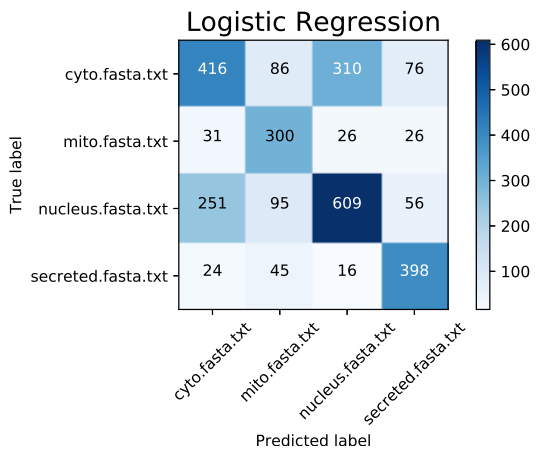

Fig. 10. Confusion Matrix of Logistic Regression shows that 416 sequences of cyto, 300 sequences of mito, 609 sequences of nucleus and 398 sequences of secreted as correctly classified. Whereas, 31 sequences of mito, 251 sequences of nucleus and 24 sequences of secreted are misclassified as cyto and so on.

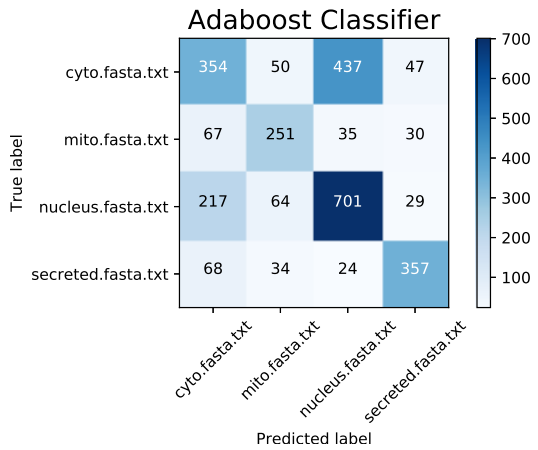

Fig. 11. Confusion Matrix of Adaboost Classifier shows that 354 sequences of cyto, 251 sequences of mito, 701 sequences of nucleus and 357 sequences of secreted as correctly classified.

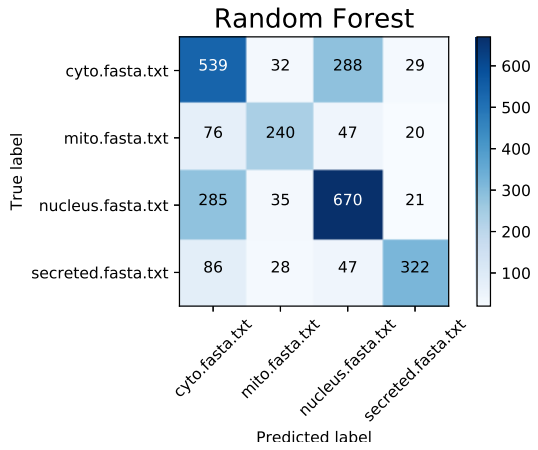

Fig. 12. Confusion Matrix of Random Forest shows that 539 sequences of cyto, 240 sequences of mito, 670 sequences of nucleus and 322 sequences of secreted as correctly classified.

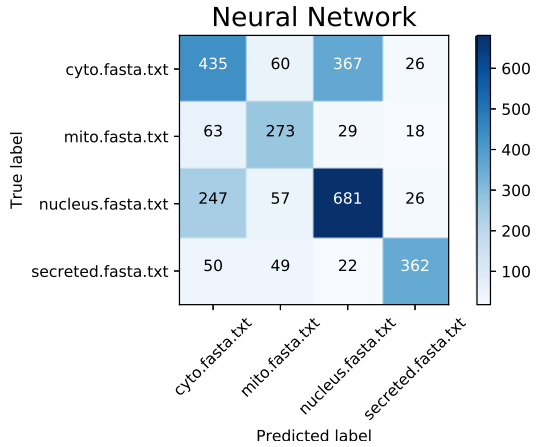

Fig. 13. Confusion Matrix of Neural Network shows that 435 sequences of cyto, 273 sequences of mito, 681 sequences of nucleus and 362 sequences of secreted as correctly classified.

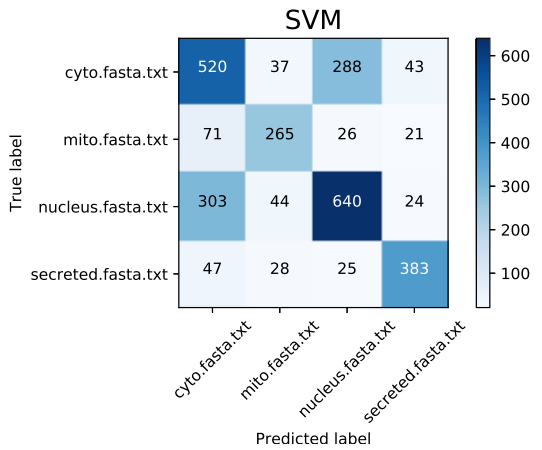

Fig. 14. Confusion Matrix of SVM shows that 520 sequences of cyto, 265 sequences of mito, 640 sequences of nucleus and 383 sequences of secreted as correctly classified.

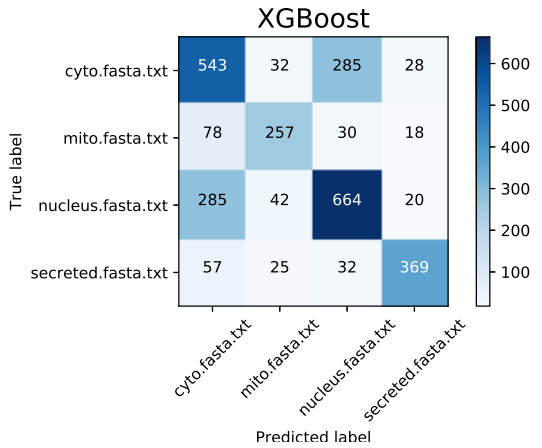

Fig. 15. Confusion Matrix of XGBoost shows that 543 sequences of cyto, 257 sequences of mito, 664 sequences of nucleus and 369 sequences of secreted as correctly classified.

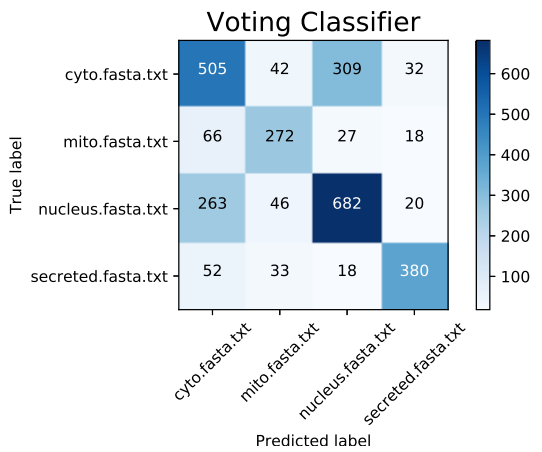

Fig. 16. Confusion Matrix of Ensembled models shows that 505 sequences of cyto, 272 sequences of mito, 682 sequences of nucleus and 380 sequences of secreted as correctly classified.

VII. Testing with Blind Data Set

To test the system with completely blind dataset, voting classifier is used to classify the 20 unknown sequences. Results of the system are illustrated in Table II.

TABLE III
PREDICTING RESULT OF BLIND DATASET

SEQ677	**cyto**	**Confidence**	**32.77%**
SEQ231	**secreted**	**Confidence**	**56.37%**
SEQ871	**secreted**	**Confidence**	**51.59%**
SEQ388	**cyto**	**Confidence**	**46.21%**
SEQ122	**nucleus**	**Confidence**	**46.82%**
SEQ758	**nucleus**	**Confidence**	**50.88%**
SEQ333	**cyto**	**Confidence**	**38.91%**
SEQ937	**cyto**	**Confidence**	**53.70%**
SEQ351	**cyto**	**Confidence**	**44.47%**
SEQ202	**cyto**	**Confidence**	**41.40%**
SEQ608	**mito**	**Confidence**	**62.83%**
SEQ402	**mito**	**Confidence**	**50.18%**
SEQ433	**secreted**	**Confidence**	**64.49%**
SEQ821	**cyto**	**Confidence**	**62.90%**
SEQ322	**nucleus**	**Confidence**	**66.30%**
SEQ982	**nucleus**	**Confidence**	**64.28%**
SEQ951	**nucleus**	**Confidence**	**47.61%**
SEQ173	**nucleus**	**Confidence**	**51.12%**
SEQ862	**mito**	**Confidence**	**42.19%**
SEQ224	**mito**	**Confidence**	**41.80%**

VIII. Conclusion

Some well known classification algorithms have been evaluated to correctly estimate the subcellular locations of protein sequences. The results obtained fall in line with the conventional understanding of classical machine learning algorithms. A brief summary of the findings is shared below.

Despite being a model of very limited capacity, logistic regression's performance is not far behind that of more complex algorithms. This indicates that the data is fairly well behaved in the input sample space. However, the data is not entirely linearly separable, as is indicated by a much better performance by SVM. SVM is known to address the issue of the input sample space not being linearly separable fairly well. Finally, owing to the generalization impact of using an ensemble, XGBoost ends up giving the best accuracy score.

Although the classical neural network used for this experiment does not offer any remarkable performance, it is the author's hope that leveraging a deep architecture for modeling the input data would result in a more precise classification. Having shown promising results on sequence data in general, a naturally well suited candidate for evaluation would be a Recurrent Neural Network (RNN).

However this analysis provides concrete evidence that machine learning algorithms would play a vital role in dealing with large sequences of amino acids to supplement bio-informatics research in the future.

IX. Acknowledgment

I would express my special thanks of gratitude to Professor David Jones for giving me an opportunity to work on this interesting project "Predicting the Subcellular Locations of Eukaryotic Proteins" and helping me with innovative ideas.

This project has helped me to get a deeper insight about protein sequences and various machine learning algorithms. I would also like to thank Dr.Kevin Bryson who is also the module tutor of Bioinformatics COMPGI10 module.

X. APPENDIX

Scikit Learn is used to implement the machine learning models.

DictVectorizer class is used to convert the list of features in the form of a vector.

First of all the training features are split into the training set and validation set. Then, the model is first trained on the X_train and Y_train and then accuracy is evaluated over the validation set. Finally, the labels are predicted over the test set.

Grid search method is used to find the optimal parameters of each model, which are then passed to the voting classifier where the final ensembling of all the models takes place. The source code can be found https://github.com/kamakshi22/BIOINFORMATICS/blob/master/src/ProteinFinal.ipynb.

The detailed description of the functions used in the code are as follows:

A. *processData*

This function removes the first line (information about sequence) of each sequence and extracts the amino acid sequence and stores it in the form of list. Along with it, this function also stores the labels for each and every sequence into another list.

B. *count_dict*

This function counts the number of each amino acid in the whole of the sequence and stores it in the form of dictionary.

C. *count_dict_from_Top50*

This function counts the number of each amino acid from the top 50 of the sequence and stores it in the form of dictionary.

D. *count_dict_from_Last50*

This function counts the number of each amino acid from the last 50 of the sequence and stores it in the form of dictionary.

E. *molecular_weight*

This function evaluates the molecular weight of the whole of the sequence and stores it in the form of dictionary.

F. *isoelctric_point*

This function evaluates the isoelectric point of the whole of the sequence using Biopython and stores it in the form of dictionary.

G. *Features*

This function calls all the fuctions which extract features and appends them to the list. It gives an output of list of dictionaries.

H. *printResults*

This function prints the labels on the test data along with their confidence.

REFERENCES

[1] Andy Liaw and Matthew Wiener.Classification and Regression by randomForest. http://ai2-s2-pdfs.s3.amazonaws.com/6e63/3b41d93051375ef9135102d54fa097dc8cf8.pdf
[2] Neural Network http://www.kdnuggets.com/2016/10/beginners-guide-neural-networks-python-scikit-learn.html
[3] Scikit Learn http://scikit-learn.org/stable/modules/neural_networks_supervised.html
[4] EnsembleVoteClassifier http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/.
[5] XGBoost Classifier http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf
[6] Confusion Matrix https://en.wikipedia.org/wiki/Confusion_matrix
[7] Ian Goodfellow and Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press,2016.
[8] Amino Acid http://www.ddbj.nig.ac.jp/sub/code-e.html#amino-1
[9] Amino Acid sequence https://en.wikipedia.org/wiki/Protein_primary_structure