Predicting TF Binding Sites

Combined Project Report for COMP 561 & 598
Jaspal Singh, 260727323, jaspal.singh2@mail.mcgill.ca
Ramchalam Kinattinkara Ramakrishnan, 260711189, ramchalam.kinattinkararamakrishn@mail.mcgill.ca

***Abstract*— This paper presents the basic methodology and results obtained for the prediction of Transcription Factor binding locations within the Human Genome. Machine learning techniques have been implemented for the prediction of these binding sites. DNA shape data has also been utilized and fed into the algorithms to find specific patterns to predict TF binding sites. Finally, an ensemble of both structure and sequence (using 1 hot encoding and PWM) has been used for prediction. The best accuracy achieved for the 4 TFs chosen for this paper are E2F4:99.68 %, UAK21: 99.86 %,ZNF263:99.99 % MAX:100 %. The results suggest that applying DNA shape data to predict TF binding sites is encouraging.**

I. INTRODUCTION

Transcription factors are proteins involved in the process of converting, or transcribing, DNA into RNA.[1] Transcription factors include a wide number of proteins, excluding RNA polymerase, that initiate and regulate the transcription of genes. One distinct feature of transcription factors is that they have DNA-binding domains that give them the ability to bind to specific sequences of DNA called enhancer or promoter sequences. Some transcription factors bind to a DNA promoter sequence near the transcription start site and help form the transcription initiation complex. The action of transcription factors allows for unique expression of each gene in different cell types and during development.

Recent studies have indicated that the structure of DNA plays a crucial role in the prediction of Transcription Factor Binding Sites. For this project, we have utilized the software DNAShapeR[2] which is an open source package written in R which predicts the shape of the DNA. DNAshapeR predicts DNA shape features in an ultra-fast, high-throughput manner from genomic sequencing data. The package takes either nucleotide sequence or genomic intervals as input, and generates various graphical representations for further analysis. The crux of DNAShape program uses a sliding pentamer window where structural features unique to each of the 512 distinct pentamers define a vector of minor groove width (MGW), Roll, propeller twist (ProT), and helix twist (HelT) at each nucleotide position. MGW and ProT define base-pair parameters whereas Roll and HelT represent base pair- step parameters. It predicts the four DNA shape features MGW, HelT, ProT, and Roll, which were observed in various co-crystal structures as playing an important role in specific protein-DNA binding.

Our core idea was to integrate the structural and sequential features to predict TF binding sites. It is noted recently,[3] that the accuracy obtained using only DNAShapeR data improves prediction of the TF binding sites. Our aim was to improve on this accuracy by creating an ensemble of structural and sequential features. We have essentially utilized Random Forests as our primary Machine Learning Algorithm.

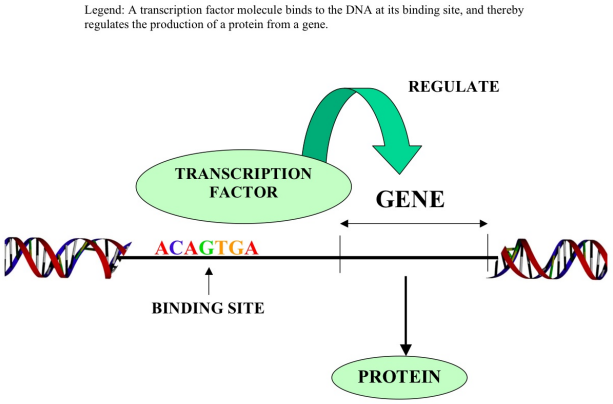

Fig. 1: Binding of Transcription Factor

II. FEATURE SELECTION

II-A. Methods

Specific methodologies were followed while choosing the TF and other hyper parameters.

II-A.1. TF Selection: Recent published papers have shown that DNAShape features work best for specific TFs which belong to E2F and MADS-domain family. [3] Hence these families were chosen along with two other TFs (selected based on the number of positive examples available). Therefore E2F4, MAX, UAK21 and ZNF263 were the TFs selected.

II-A.2. DNAShapeR: DNAShapeR returns 4 values for a given sequence - MGW, HelT, ProT, and Roll. These values were used individually as a feature set and were later also combined to be used as a feature set.

DNAShape Vector	Description	Feature Category
MGW	Minor Groove Width	Float
Prot	Propeller Twist	Float
Helt	Helix Twist	Float
Roll	Roll	Float

Fig. 2: DNAShape Features

II-A.3. PWM: The provided position Weight Matrix was used to calculate a score for each k-mer. This score was calculated by multiplying the PWM value for each nucleotide at given positions in the k-mer. It was observed that for k-mers belonging to non binding sites this score would more often than not turn to 0 because pwm matrix contained 0s for these nucleotides.

II-A.4. 1-hot encoding of k-mers: Each sequence representing a k-mer of the prospective binding or non binding sites was encoded using a technique called one-hot encoding. This encoding technique is used to represent each nucleotide with a binary number of 4 digits. (For Ex: A was represented with 1000, C with 0100, G with 0010 and T with 0001). So for a k-mer of 20 nucleotides a feature set of (20*4) 80 integers was created.

Feature Name	Description	Feature Category
DNAShape	The shape vectors	Float
K-mer	The one hot encoding of k-mer data	Bits
PWM	Score based on Position Weight Matrix for the TF	Float

Fig. 3: Features used in the algorithms

II-B. Feature Extraction

Although the complete human genome was provided, extracting useful data was a major challenge. The major feature extraction methods which were employed are listed below:

II-B.1. Data Extraction: DNAShapeR program takes in a DNA sequence as input and returns the output shape vectors. Although DNAShape doesnt consider all the neighbors of each position, we chose a method to overcome this pseudo problem. The idea was to generate the 4 output vectors for each of the chromosomes of the human genome. As processors and time were a premium, this was one of the trade-offs of the method. Each DNAshape vector generation took close to 3-4 hours and with the limited computational capabilities, only chr10 to chr22 were generated using DNAshape R. One of the drawbacks of DNAShapeR program was that it failed to separate certain parts of the output with commas. An additional burden of modifying these huge files was added. Once this was completed, the next step was to generate the positive and negative samples. The positive samples were generated as per the indices mentioned in the file. For each of the output vectors generated, the corresponding indices were mapped and each of MGW, Helt, Prot and Roll files were generated. The same was followed for the negative samples and another set of vector files were generated. Although this was excruciatingly slow, it was the most logical method to do.

For the second method of extraction, the idea was to generate the DNAShape data on the fly. The respective indices were scanned in the files and it was then extended to around 50 nucleotides on both sides and then provided as input to the DNAShapeR program. This would essentially return the output shape vectors for the corresponding indices. This method returned faster output when compared to the previous one albeit with minute discrepancies in the output data. Although this method was not flawless, taking time into consideration, it was swift in the execution. Moreover, this facilitated in extraction of a higher number of features and thus was indicative of the true nature of the Human Genome. During the positive and negative extraction of the data, the 1 hot encoding of kmer features was also initiated in parallel, along with the PWM vector.

After extraction of these data, inconsistent data such as non integer values present in the DNAShapeR (Ex: 'NA') output, k-mer contains unknown nucleotides (N), etc were removed from the training set. Providing clean data to the classifier was of utmost importance and it was logical to cleanse the data before hand.

II-B.2. Positive and Negative Examples: The positive examples were chosen from each of the chr files of the human genome. For each of the TF chosen, the positive examples were extracted from the indices mentioned in the files. This was quite straightforward. The challenge was to mimic the actual biological scenario of data imbalance and training the model to learn for such a case. This required negative examples at a much higher value and hence window sliding method was used to generate these negative features. The indices were mentioned for the negative examples and based on the TF binding length, a sliding window was used to generate all these features. This took a huge toll on the processing time and memory but the objective of the project depended on resembling this data imbalance problem to reflect the true capability of the model using these features.

II-C. Feature Selection

The objective was to make sure the results obtained were truly indicative of what happens within the genome. It is known that there is 1 TF binding site for approximately 100000 non-binding sites. Taking this into consideration, our algorithm has been designed to execute on different types of features. As discussed in feature selection methods, the classifier was trained with these 4 feature sets:

- Feature Set A: All of DNAShape Vectors
- Feature Set B: Individual DNAShape Vectors
- Feature Set C: Only k-mer and PWM
- Feature Set D: Ensemble of all above sets

III. IMPLEMENTATION

Machine Learning techniques have been used to predict the TF binding sites. 3 different machine learning algorithms were used in this project. In all the models, a Train-Test split of 70-30 % was used for the total training data. A 10 fold Cross Validation was used to calculate the final accuracy. The machine learning algorithms used are described below:

III-.1. Random Forests: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.[4] Random decision forests correct for decision trees' habit of over-fitting to their training set. It was one of simplest and basic algorithm that tends to perform consistently better in computational biology tasks.

For the current scenario, the hyper-parameters were chosen based on the 10 fold Cross-Validation. The best n-estimators(number of decision trees) value was 20 with the best accuracy. The input was fed into the classifier. The number of samples in training and test data for each TF after the split is mentioned in figure 4

TF Type	Total Train Set Size	Total Test Set Size
E2F4	111914	55123
UAK21	96288	47426
MAX	105819	52120
ZNF263	122179	60179

Fig. 4: Test-Train Split

III-.2. Extremely Randomized Trees: In Extremely Randomized Trees, randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

The model was trained with the same number of features and input data and tested for various hyperparameters. This was only an extension of Randomized Trees and hence was expected to give at-least the same results as Randomized Trees. Again a 10 fold Cross Validation was used to predict the optimal n-estimators (the number of decision trees). The similarity to Randomized trees was evident and the number came up-to around 16.

III-.3. Artificial Neural Networks: An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information[5]. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

Compared to the previous 2 methods selected, this was a lot complex and time consuming. The accuracy was dependent on the number of hidden layers and the neuron count within each layers. The optimal number of neurons within a layer for one hidden layer came upto around 2000 through a 10 fold Cross Validation.[Figure 5]

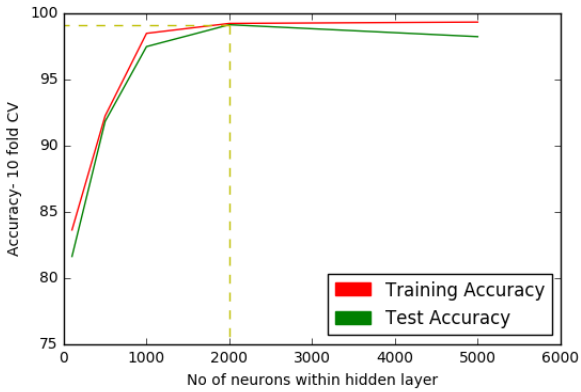

Fig. 5: Accuracy vs No of neurons in a hidden layer. E2F4: Neural Network using Set D

All the machine learning algorithms were implemented using scikit-learn [6]

IV. RESULT

The results of the various models and their accuracies are mentioned below.

IV-A. E2F4

E2F4 is one of the TFs which would plausibly perform well with shape data. The results are quite evident. For E2F4 using Neural nets, the results (table 6) show that the ensemble of all features (Set D) performs the best. The kmer + PWM (Set C) feature set also performs well in this case. It could possibly be due to the class imbalance problem as there are a lot more negative cases than positive cases. The sensitivity is much lower compared to Set D. Set B performs poorly as expected. This is primarily due to the fact that one shape vector alone is not sufficient to train the classifier. In all cases, Helt shape vectors have been used since they proved to be the best of the lot.

Feature Set Used	**Set A**	**Set B**	**Set C**	**Set D**
Total Positive	24002	24002	24002	24002
Total Negative	143035	143035	143035	143035
Accuracy	99.49	88.09	99.2	99.677
Sensitivity	0.9971	0.3018	0.9792	0.9969
Specificity	0.9946	0.9767	0.9947	0.9967
Total Error Count	277	6562	410	178
False Positive	255	1100	248	154
False Negative	22	5462	162	24

Fig. 6: Results E2F4: Neural Networks

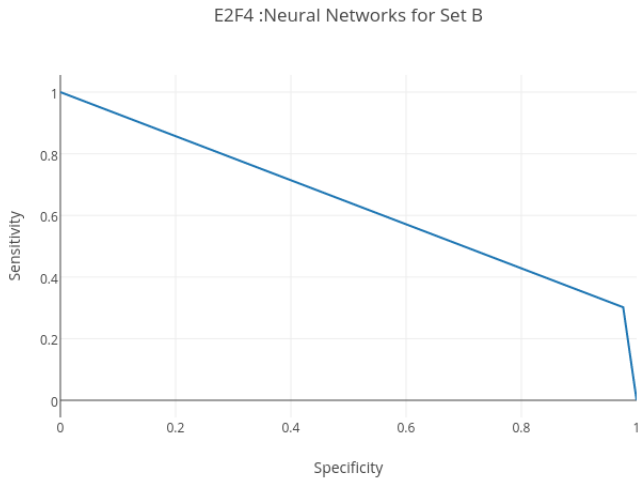

Fig. 7: Specificity vs Sensitivity for E2F4: Neural Network using Set B

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	24002	24002	24002	24002
Total Negative	143035	143035	143035	143035
Accuracy	99.61	99.55	99.60	99.68
Sensitivity	0.9957	0.9911	0.9956	0.9960
Specificity	0.9961	0.9963	0.9961	0.9970
Total Error Count	174	244	182	171
False Positive	137	175	144	140
False Negative	37	69	38	31

Fig. 8: Results E2F4: Random Forest

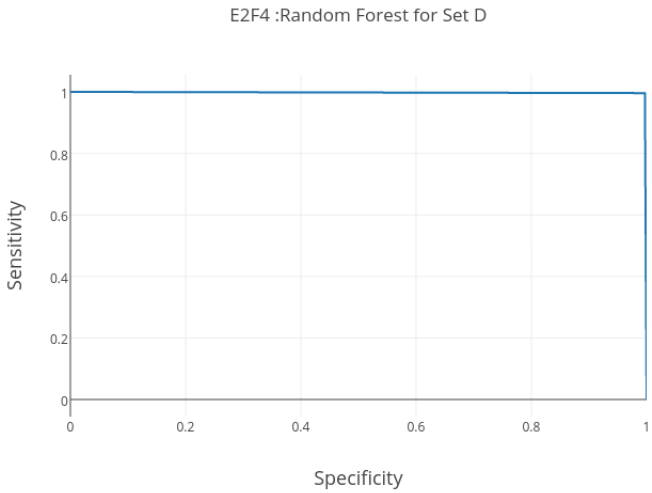

Fig. 9: Specificity vs Sensitivity for E2F4: Random Forest using Set D

E2F4 with random forests (refer table 8) follows more or less the same trend albeit it being more accurate. One difference to be noted is that, Random Forest is able to garner enough pattern for Set B and it performs much better than neural nets. The Specificity vs Sensitivity graph is plotted for Set D (Figure 11)and it indicates high accuracy since the area under the curve is almost equal to 1.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	24002	24002	24002	24002
Total Negative	143035	143035	143035	143035
Accuracy	99.63	99.50	99.55	99.677
Sensitivity	0.9952	0.9860	0.9909	0.9960
Specificity	0.9968	0.9965	0.9967	0.9969
Total Error Count	186	274	225	175
False Positive	147	165	154	137
False Negative	39	109	71	41

Fig. 10: Results E2F4: Extremely Randomized Trees

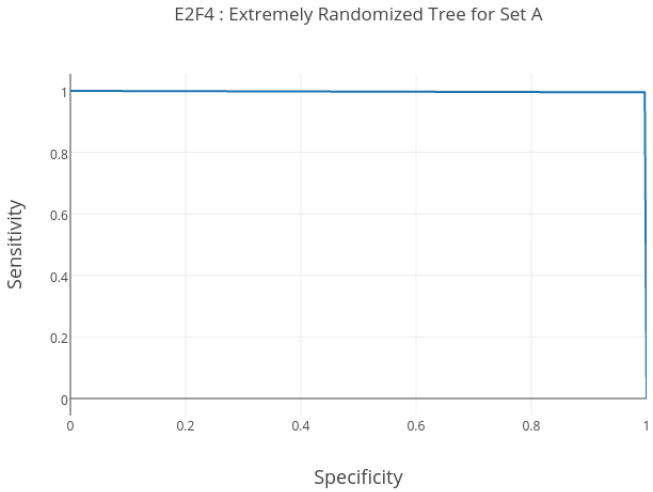

Fig. 11: Specificity vs Sensitivity for E2F4: Extremely Randomized Tree using Set A

Finally, for E2F4 with Extremely Randomized Trees, the table 11 follows the same trend as Random Forests albeit a sightly lower accuracy compared to Random Forests.

IV-B. UAK21

UAK21 is a TF which is not one of the TF Families used for Shape Data Training. For Neural Nets, it can be seen in table 12 that set B has 0 sensitivity. The set A shows that the impact of using shape data on this TF does not give substantial accuracy(figure 13) Moreover, even the ensemble set D has a lower Sensitivity compared to the previous values. This could point out 2 things: One being the class imbalance problem- The total number of positive cases is quite low. Two being, UAK families do not result in high accuracies when using Shape Data with Neural Nets.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	679	679	679	679
Total Negative	143035	143035	143035	143035
Accuracy	99.68	99.47		99.71
Sensitivity	0.601	0	0.5384	0.801
Specificity	0.998	1.0	0.9978	0.998
Total Error Count	142	247	214	137
False Positive	24	0	100	88
False Negative	118	247	114	49

Fig. 12: Results UAK21: Neural Networks

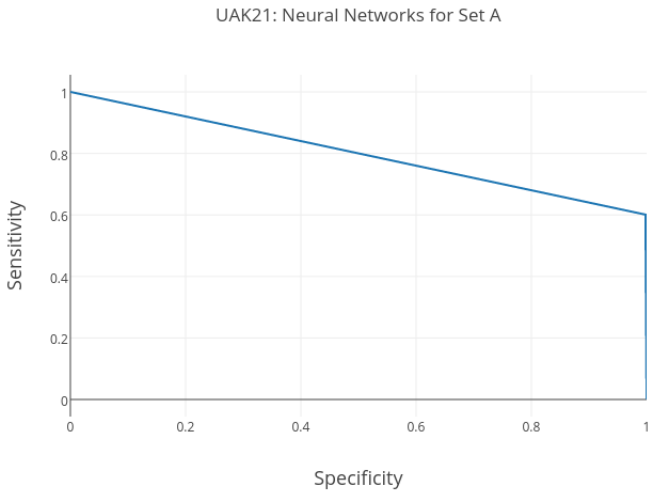

Fig. 13: Specificity vs Sensitivity for UAK21: Neural Networks using Set A

Using Random Forest (RT)(table 14) and Extremely Randomized Trees (ERT)(table 16) gives homogeneous solutions. This could be attributed to the fact that the algorithms used underneath are somewhat similar in their approach. Overall random forests performs slightly better for set B compared to ERT. Moreover, the positive examples being dwarfed by the negative examples could be one of the reasons attributed for the lower sensitivity.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	679	679	679	679
Total Negative	143035	143035	143035	143035
Accuracy	99.81	99.80	99.2	99.83
Sensitivity	0.796	0.7246	0.7004	0.797
Specificity	0.9993	0.9995	0.9997	0.9994
Total Error Count	81	87	100	78
False Positive	23	19	24	28
False Negative	58	68	76	50

Fig. 14: Results UAK21: Random Forest

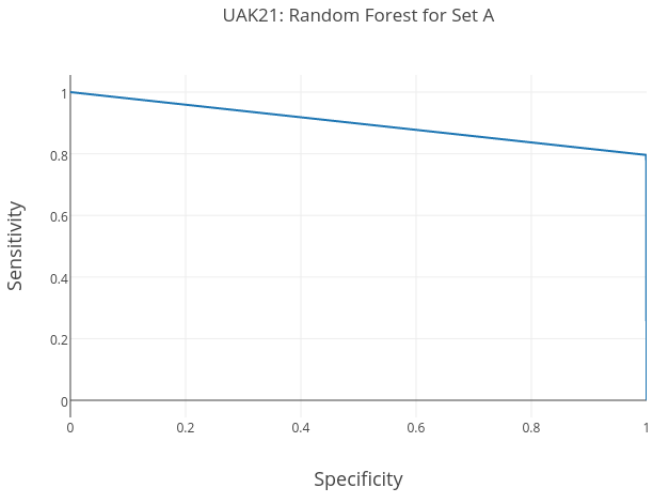

Fig. 15: Specificity vs Sensitivity for UAK21: Random Forest using Set A

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	679	679	679	679
Total Negative	143035	143035	143035	143035
Accuracy	99.74	99.60	99.6	99.86
Sensitivity	0.5465	0.2550	0.6599	0.821
Specificity	0.9998	0.9999	0.9997	0.9995
Total Error Count	119	187	96	65
False Positive	7	3	12	21
False Negative	112	184	84	44

Fig. 16: Results UAK21: Extremely Randomized Trees

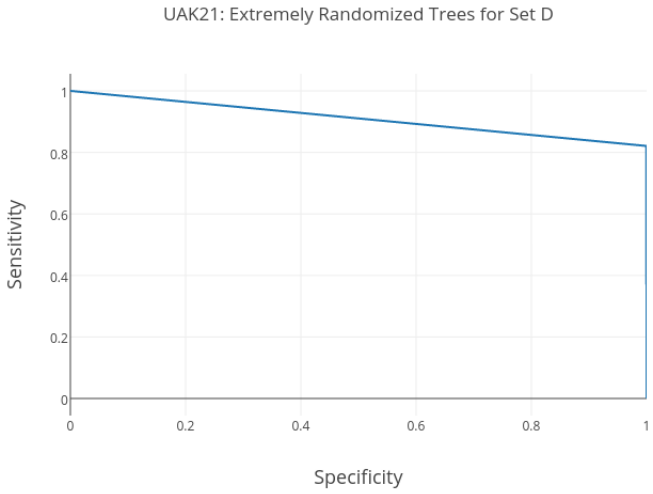

Fig. 17: Specificity vs Sensitivity for UAK21: Extremely Randomized Tree using Set D

IV-C. MAX

MAX families of TFs are again renowned for performing well with Shape Data. The accuracy with Neural Networks is mentioned in table 18. The area under the curve (figure 19) plotted for set B proves that set B is not an appropriate feature set used for classification with neural networks.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	14903	14903	14903	14903
Total Negative	143035	143035	143035	143035
Accuracy	99.87	90.43	99.8	99.92
Sensitivity	0.9997	0.007	0.9997	0.9960
Specificity	0.9986	0.9961	0.9988	0.9993
Total Error Count	65	4987	68	49
False Positive	64	182	40	30
False Negative	1	4805	15	19

Fig. 18: Results MAX: Neural Networks

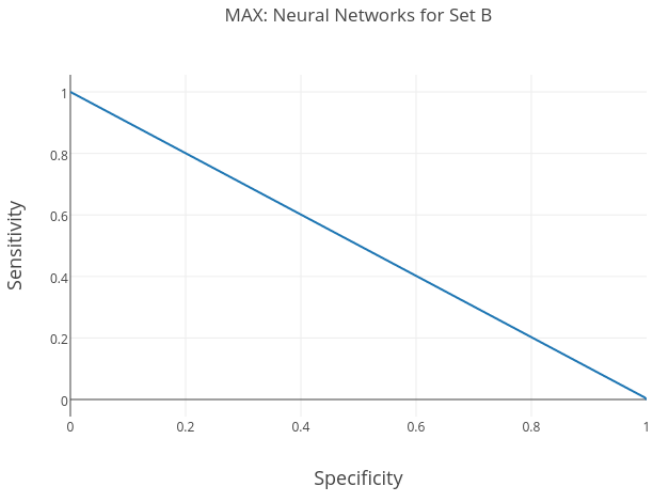

Fig. 19: Specificity vs Sensitivity for MAX: Neural Networks using Set B

The results using Random Forests is shown in table 20. Figure 21 shows that set B, which was not useful for classification using Neural Networks, gets a huge accuracy boost, thereby proving that Random Forests are better at this task than Neural Networks.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	14903	14903	14903	14903
Total Negative	143035	143035	143035	143035
Accuracy	99.94	99.94	99.9	100
Sensitivity	0.9989	0.9989	0.9986	1.0
Specificity	0.9995	0.9995	0.9995	1.0
Total Error Count	28	29	32	0
False Positive	23	24	26	0
False Negative	5	5	6	0

Fig. 20: Results MAX: Random Forest

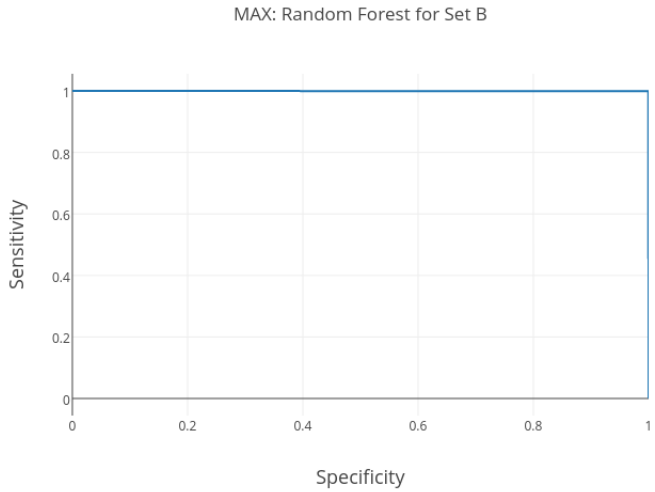

Fig. 21: Specificity vs Sensitivity for MAX: Random Forests using Set B

The results using Extremely Random Trees are mentioned in table22. Figure 23 shows the specificity vs sensitivity graph for Feature Set D using Extremely Random Trees.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	14903	14903	14903	14903
Total Negative	143035	143035	143035	143035
Accuracy	99.95	99.95	99.94	99.96
Sensitivity	0.9991	0.9989	0.9988	0.9993
Specificity	0.9995	0.9995	0.9994	0.9996
Total Error Count	23	24	29	20
False Positive	19	19	18	17
False Negative	4	5	11	3

Fig. 22: Results MAX: Extremely Randomized Trees

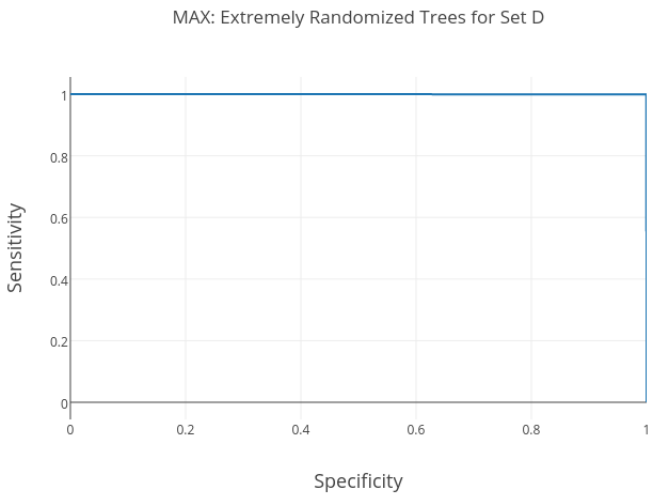

Fig. 23: Specificity vs Sensitivity for MAX: Extremely Randomized Trees using Set D

IV-D. ZNF263

ZNF263 is the TF with the most number of positive examples among the TFs chosen for this project. The results for Neural networks, Random Forest and Extremely Random Trees are depicted in table 24, 26 and 28 respectively. The results show that the machine learning classifier is robust to both kinds of data where there is a class imbalance problem and where there is no class imbalance problem.

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	39322	39322	39322	39322
Total Negative	143035	143035	143035	143035
Accuracy	99.43	94.00	99.46	99.73
Sensitivity	0.9981	0.9155	0.9941	0.9984
Specificity	0.9932	0.9466	0.9949	0.9970
Total Error Count	342	3610	317	159
False Positive	318	2522	241	154
139 Negative	24	1088	76	20

Fig. 24: Results ZNF263: Neural Networks

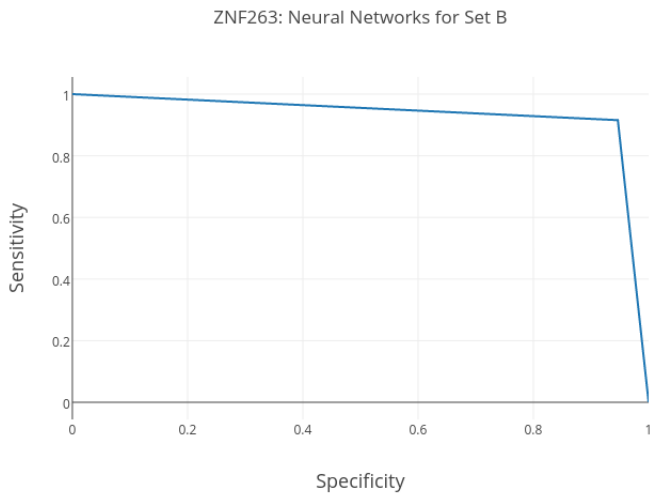

Fig. 25: Specificity vs Sensitivity for ZNF263: Neural Network using Set B

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	39322	39322	39322	39322
Total Negative	143035	143035	143035	143035
Accuracy	99.59	99.52	99.6	99.99
Sensitivity	0.9958	0.9920	0.9959	0.9996
Specificity	0.9966	0.9970	0.9967	1.0
Total Error Count	213	244	200	5
False Positive	160	141	151	0
False Negative	53	103	49	5

Fig. 26: Results ZNF263: Random Forest

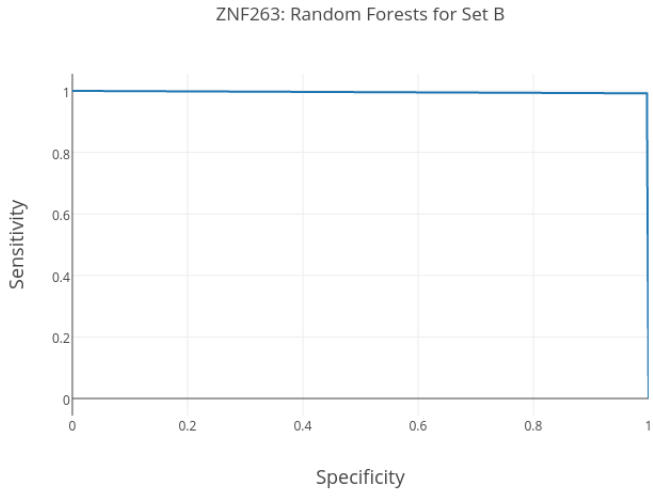

Fig. 27: Specificity vs Sensitivity for ZNF263: Random Forests using Set B

Feature Set Used	Set A	Set B	Set C	Set D
Total Positive	39322	39322	39322	39322
Total Negative	143035	143035	143035	143035
Accuracy	99.64	99.58	99.62	99.78
Sensitivity	0.9956	0.9909	0.9955	0.9975
Specificity	0.9969	0.9972	0.9967	0.9973
Total Error Count	202	248	206	158
False Positive	146	131	150	127
False Negative	56	117	56	31

Fig. 28: Results ZNF263: Extremely Randomized Trees

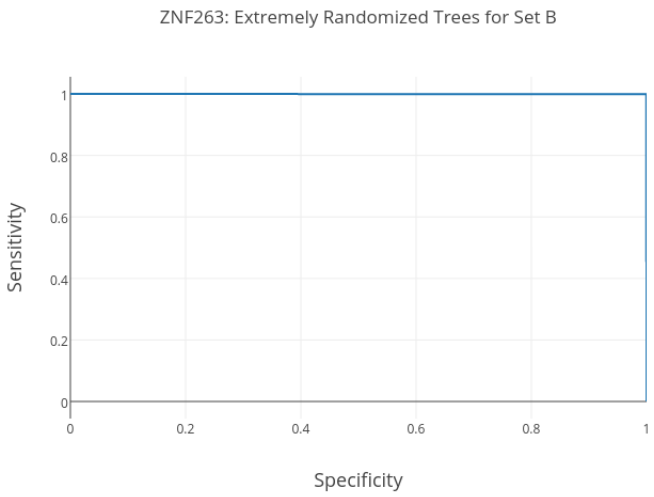

Fig. 29: Specificity vs Sensitivity for ZNF263: Extremely Randomized Trees using Set B

V. Discussion & Future Work

The following observations could be made from the results obtained:

- It is observed that the ensemble of all features sets (Sets D) perform the best and provide the highest accuracy and sensitivity.
- Although Shape Data generally tends to perform better, the tried and tested kmer-PWM combination performs equally good.
- Moreover, the classification scores indicate there could be a plausible one to one relationship between set A and C for some TFs.
- In most cases, Set B has performed the lowest and this is evident from the fact that Helt vectors individually do not provide any pattern for the algorithm to exploit.
- Additionally, in all cases the ensemble of Shape and Seqeunce features provide the best accuracy which indicates there may not necessarily a direct one to one mapping between shape only and sequence only features in all TFs.
- Despite the class imbalance problem, the high sensitivity and specificity values indicate the algorithm being able to exploit the dataset and classify both classes correctly.
- Assessing the 4 TFs used, ZNF263 had the highest number of positive samples and UAK21 had the least. MAX TF Family for Set D with Random Forests provided perfect accuracy with 0 False positives and 0 False negatives.
- Random Forests and Extremely Random Trees performed better than Neural Networks. This could possibly be due to the fact that Neural Networks was not completely optimized. The optimal number of hidden layers and neurons need to be obtained to improve the accuracy further.

Even though the accuracy and results are in the high nineties, these numbers can be misleading because of the following reasons:

- The ratio of negative samples compared to positive samples is very high in the human genome,i.e. approximately in the ratio of 100000:1. The data used in the experiments above is in the range of 10:1 for most of the TF's shown above.
- The experiments were only conducted for some specific TF's. Also the positive and negative samples were taken from a subset of the entire human genome.

In addition to finding ways to obviate the problems stated above, the below mentioned points state some of the factors which when accounted for could improve the accuracy of the classifiers.

- Considering TF-TF interaction sites, to predict the TF binding sites.
- Factor in some algorithms to consider the shape of the nearby nucleotides.
- Use the patterns of the neighboring areas of the TF/k-mer match with the sequence and use it as a potential feature for the model.
- Higher computational capabilities would help ease the problem of time consumption. Despite running on a system which includes NVIDIA Graphics Card 940M, the DNAShapeR feature generation was extremely time consuming. Additionally, lack of memory also played a crucial role. The chromosome data had to be split into smaller subsets and done separately. Ultimately the total time for generating the negative and positive indices easily exceeded 2-3 days. Adequate machinery would go a long way in improving the overall time taken and assist in the optimisation of the code.

Overall, the belief is that the feature extraction methods employed is fair enough to reflect the high accuracies of the models. We believe this can be extended to other Transcription Factors to predict the prospective binding sites.

VI. Statement of Contributions

We hereby state that all the work presented in this report is that of the authors. All authors contributed equally to the approach, design of feature sets, implementing the machine learning classifiers and the written report.

REFERENCES

[1] http://www.nature.com/scitable/definition/general-transcription-factor-transcription-factor-167

[2] Tianyin Zhou , Lin Yang , Yan Lu , Iris Dror , Ana Carolina Dantas Machado, Tahereh Ghane , Rosa Di Felice, and Remo Rohs DNAshape: a method for the high-throughput prediction of DNA structural features on a genomic scale, Nucleic Acids Research, 2013, Vol. 41

[3] Anthony Mathelier, Beibei Xin, Tsu-Pei Chiu, Lin Yang, Remo Rohs, Wyeth W. Wasserman: Article DNA Shape Features Improve Transcription Factor Binding Site Predictions In Vivo http://www.cell.com/cell-systems/pdf/S2405-4712(16)30218-6.pdf

[4] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

[5] Christos Stergiou and Dimitrios Siganos. Neural Networks https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[6] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E.Scikit-learn: Machine Learning in Python Journal of Machine Learning Research, volume=12,pages=2825–2830, year=2011