Human Gene Function Prediction: Machine-Learning Approaches

Haoxu Huang
huan1780@umn.edu
University of Minnesota - Twin Cities
Department of Computer Science and Engineering
Minneapolis, Minnesota, USA

Gustav Baumgart
baumg260@umn.edu
University of Minnesota - Twin Cities
Department of Computer Science and Engineering
Minneapolis, Minnesota, USA

Joshua Bernier
berni141@umn.edu
University of Minnesota - Twin Cities
College of Continuing and Professional Studies
Minneapolis, Minnesota, USA

Ingrid Rodriguez Aragon
rodri754@umn.edu
University of Minnesota - Twin Cities
Department of Physiology and Integrative Biology
Minneapolis, Minnesota, USA

Abstract

Gene function annotation is a great challenge as there is an expanding amount of biological information being released by genome-wide sequencing studies across organisms. A useful approach to tackle this large amount of new data is to apply machine learning methods that can predict gene function. In this study, support vector machine (SVM), k-nearest neighbor (KNN), and neural network (NN) were evaluated on their ability to categorize uncharacterized human cancer cell line genes. Gaussian Naive Bayes (GNB) was also tested based on a peer's observation that it yielded high performance on this dataset. Based on calculated ROC-AUC scores, the model that provided the most accurate gene function prediction was GNB, although predictive performance was improved via ensemble learning by blending all models. The top ranked GO-terms from our blending model included several annotations related to the cell cycle and DNA replication/repair. The GO-terms that were predicted with the lowest performance had more varied biological roles, such as those related to memory or response to chemicals such as ethanol. In summary, we found machine learning models can be useful tools to accurately predict gene functions, especially when multiple models are blended to improve performance.

Keywords

Bioinformatics, Gene Ontology, Machine Learning, Ensemble Learning

1 Introduction

The development of the CRISPR-Cas9 gene editing system has allowed for the study of genomes by observation of the effects of precise edits to an organism's genes. Even with the assistance of CRISPR-Cas9, studying the human genome is a challenging endeavor. However, with machine learning models scientists can make more informed decisions to guide genetic research, categorize genes, and identify gene targets of interest for disease treatment. We worked on the Human Gene Function Prediction challenge through Kaggle, using gene expression data from the Dependency Map project. This data was obtained through experiments on human cancer cell lines that involved altering or knocking-out specific genes using the CRISPR-Cas9 system to determine their function and importance to various biological processes. Through these experiments, these genes associated with human cancer cell lines could then be categorized with gene ontology (GO) annotations. The goal of this project was to develop machine learning models that would allow us to take gene expression data from these human cancer cell lines, categorize them by GO annotations, and then determine which model performed the best.

2 Methods

2.1 Performance Metrics

Since this dataset used a very sparse matrix with 0's and 1's to represent the prediction mapping from cell lines to every GO term, performance metrics like accuracy would not work because the classifiers can simply predict every element in the matrix to be 0 to achieve a perfect score. Hence, we decided to solely use ROC-AUC score as performance metric as it was evaluated by both True Positive Rate(TPR) and False Positive Rate(FPR), which could efficiently capture False Positive(FP) and False Negative(FN) to give a more comprehensive illustration for this imbalanced dataset prediction.

2.2 Models

2.2.1 Support Vector Machine

The first algorithm we used was Support Vector Machine(SVM). SVM is developed from the generic equation of a separating hyperplane

$$w^T x + b = 0 \tag{1}$$

and separate the dataset based on the hyperplane, where w is weight and b is bias. For generic SVM model with slack variable, x is replaced by $\varphi(x)$ to create the optimization problem to maximize margin between hyperplane and datapoints.

$$\min_{w,b,\xi_i} \frac{||w||^2}{2} + C \sum_{i=1}^{n} \xi_i \tag{2}$$

subject to

$$y_i(w^T \varphi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0$$

Then, by doing derivation to solve dual problem based on this formula, Eq(3) can be derived where kernel $K(u, v) = \langle \varphi(u), \varphi(v) \rangle = f(u, v)$ can be applied to address the non-linear predictions by

mapping to higher dimension where the dataset is linearly separable in the transformed space.[7]

$$\sum_{i=1}^{n} \lambda_i y_i \langle \varphi(x_i), \varphi(x) \rangle + b = 0 \tag{3}$$

In this project, radial basis function, polynomial with degree 2 and polynomial with degree 3 kernels were used for the prediction of original dataset.

2.2.2 K Nearest Neighbors

K Nearest Neighbors(kNN) is one of the simplest machine learning algorithms that essentially performs Majority Voting or Distance-Weighted Voting based on the majority class of its nearest neighbors with a proximity metrics. In this project, we used Majority Voting with equation

$$y = \arg\max_{v} \sum_{(x_i, y_i) \in D_z} I(v = y_i) \tag{4}$$

where I is an indicator function that returns 1 or 0 based on if the argument is True.[7] In this project, Euclidean and Manhattan distance were used as proximity metrics for kNN.

2.2.3 Random Forests

Random forests(RF) is a combination of tree predictors such that each tree depended on the values of a random vector sampled independently and with the same distribution for all trees in the forest.[6] Since it is a tree based classifier, we were able to rank the features based on their mean decrease impurity. In this project, we used Random Forests as feature selection check and we chose gini impurity as splitting criterion with equation

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2 \tag{5}$$

where p_i is the fraction of items labeled with class i in the set and 1 to J is the index number of all classes. Therefore, the impurity decreased from each feature can be averaged and the features are ranked according to this measure.[2]

2.2.4 Neural Network

We used full-connected feed forward neural networks in this project. Feed forward neural networks mean the connections between nodes proceed in a forward direction and do not form a circle.[3] Hence, each neuron from every layer will compute weighted sum from activated outputs of all previous neurons and non-linearity is added with an activation function. After the value of output layer is calculated, a loss function is applied to find the loss between predicted value and true value, and back-propagation algorithm[9] is applied to update all the weights in hidden layers. A simple deep feed forward neural network without regularizing layers can be illustrated as Fig. 1.

We created our NN model by stacking up these feed forward connections with ReLu activation function and dropout layers. The hyper-parameters such as number of hidden nodes for each layer were tuned by Bayesian Optimization. More details about our model's architecture are explained in the Experiments section.

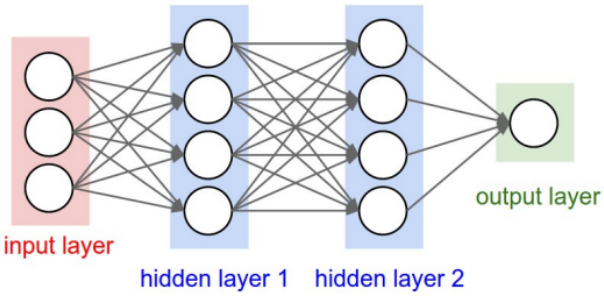


Figure 1: Feed Forward Neural Networks

As the figure showed, each neuron was connected with all neurons in next layers, which were the full-connected dense layers we used in our model.

2.2.5 Blending Predictions

For ensemble learning, we neither used the common stacking algorithm - "super learner algorithm"[8], which passes predictions from baseline models to a meta learner nor decided blending weights of each model simply by intuitions. Instead, because of the property of this dataset that it requires the algorithms to predict numerous labels, we used two methods that we constructed to modify the final predictions and decide weights that were used to stack models. First, we integrated the prediction power of our models by selecting the the predictions of GO terms with best mean 5-fold cross validation ROC-AUC among all models. Second, we integrated models by summing up predictions from all models with weights that were calculated by how many times these models won in the mean 5-fold cross validation ROC-AUC per GO term. Algorithm 1 in next page illustrates how we stacked predictions for all models and all GO terms with first method. Assuming we have n GO terms and store model data based on given indices from 0 to k total models, $Pred_j$ is the predictions for a specific model j, AUC_j^i is mean 5-fold cross validation AUC for ith GO term and model j, MaxIndex(lst) returns index of maximum number in list lst, and NeuralNet(df) returns predictions of our NN model based on input dataset df.

Algorithm 2 in next page illustrated how we stacked predictions for all models and all GO terms with second method. Notations are same as previous algorithm. Assume we have m observations on training set, Win_i represents number of wins for model i, Wins(i) returns number of wins for model i, $Weight_i$ represents weight for model i.

We attempted various combinations of blending from our six baseline models - NN, weighted kNN with k=150 and Manhattan distance, weighted kNN with k=100 and Euclidean distance, SVM with rbf kernal, SVM with degree 2 polynomial kernal, SVM with degree 3 polynomial kernel. GNB as a model that gets highest Kaggle score as single model was added later. After testing all combinations, the best blending model was selected based on their score on Kaggle.

3 Experiments and Results

3.1 Data Preprocessing

3.1.1 Dataset Normalization

We applied two methods to normalize the dataset according to

Algorithm 1 Predictions Blending By Mean ROC-AUC

Input: X_{test}, test dataset
Input: AUC_j, mean 5-fold cross validation ROC-AUC for model $j = M_0, M_1, \ldots, M_k$

$Pred_{stack} \leftarrow \text{NeuralNet}(X_{test})$
$ModelIndex \leftarrow []$
for $i = 0, 1, \ldots, n$ **do**
    $ModelsAUC \leftarrow []$
    for $j = 0, 1, \ldots, k$ **do**
        $ModelsAUC.\text{Append}(AUC_j^i)$
    end for
    $MaxModelIndex \leftarrow \text{MaxIndex}(ModelsAUC)$
    $ModelIndex.\text{Append}(MaxModelIndex)$
end for
for $i = 0, 1, \ldots, n$ **do**
    $BestModelIndex \leftarrow ModelIndex[i]$
    $Pred_{stack}[i] \leftarrow Pred_{BestModelIndex}[i]$
end for

Output: $Pred_{stack}$, final prediction from blending

Algorithm 2 Weighted Predictions blending

$Pred_{stack} \leftarrow 0_{m \times n}$
$Win_{i=0,1,\ldots,k} \leftarrow 0$
$Weight_{i=0,1,\ldots,k} \leftarrow 0$
for $i = 0, 1, \ldots, k$ **do**
    $Win_i \leftarrow \text{Wins}(i)$
    $Weight_i \leftarrow Win_i \div n$
end for
for $i = 0, 1, \ldots, k$ **do**
    $Pred_{stack}$ += $Weight_i \times Pred_i$
end for

Output: $Pred_{stack}$, final prediction from blending

their adaptation to different algorithms. We attempted both z-score normalization and quantile normalization [1] to all algorithms and observed that NN were able to perform better with quantile normalization and kNN/SVM were able to perform better with z-score normalization.

3.1.2 Feature Selection - Random Forest

In order to check if we need feature selection for this dataset, we applied Random Forest and visualized the feature importance based on gini impurity. The following is the hyper-parameters we used for training Random Forest - 100 trees in forest, gini impurity as metric to measure the quality of a split, 25 features to consider when looking for the best split and 15 as maximum depth of trees. From the sorted feature importance, we observed an anomaly that feature 'ACH.000677' had a significantly higher mean decrease gini impurity than rest of all features with 0.009764 score as rank one feature following by rank two feature with 0.005890 score. The feature importance was visualized without sorting in Fig.2.

We selected top 139 features with gini impurity higher than 0.002000 and retrained our NN model with these features. As a result, we yielded ROC AUC scores 0.6505 with 10-folds cross validation, which was lower than the score we got by training all

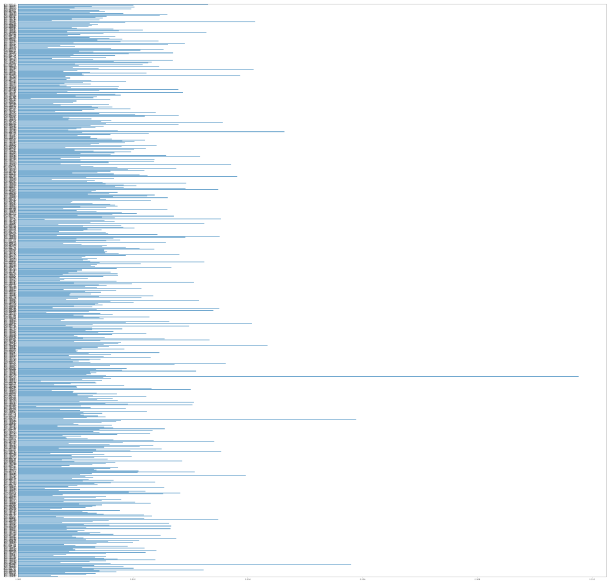


Figure 2: Random Forest Feature Importance
From the figure, we can observe that even though some features had much higher feature importance, most of them were not yielding significantly higher feature importance than each others.

features. These observations supported our decision that we were not doing feature selection for this dataset.

3.1.3 Feature Selection - Variance

After normalization by z-score across genes, we ranked features by standard deviation and selected the most variable features and tested them on a kNN weighted model using Euclidean distance. Features were ranked in descending order based on their variance after row standardization. The most variable features were selected for each point. Fig.3 shows how kNN performance varied for selecting subsets of features.

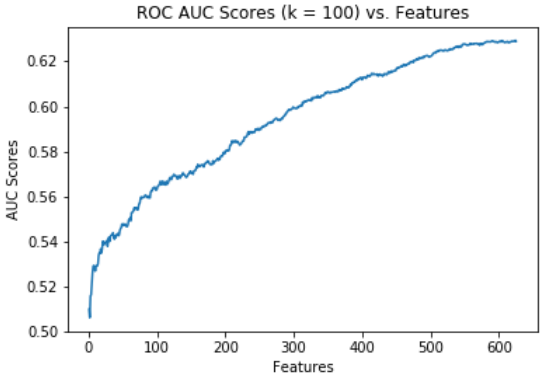


Figure 3: Variance by kNN
The figure did not support feature selection. The least variable 250 features contributed 0.02117 to the AUC score of the model. The highest point on the curve occurs when 624 out of 625 features were selected, which provided further evidence against feature selection

3.2 Model Selections

We tried various models before we decided to stack predictions with high performance classifiers. From here, we introduced our process of model selections.

3.2.1 K Nearest Neighbors

We tested the KNN model using 5-fold cross validation ROC-AUC scores for different values of k. The graph below shows the relationship between the number of neighbors and the performance for weighted Manhattan distance, weighted Euclidean distance, and uniform Euclidean models. In the weighted models, neighbors were assigned a weight by using the inverse of their distance. Fig.4 shows how the performance of these three kNN models changed with different k-values.

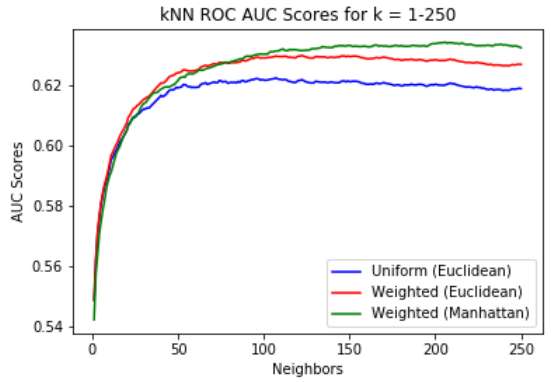


Figure 4: kNN Changes by Neighbors

From the figure, all kNN models slowed down the increase of performance after 50 neighbors and got the best performance after 100 neighbors.

The curves had their highest values at k = 107 (AUC = 0.6224) for the uniform model, k = 122 (AUC = 0.6298) for the weighted Euclidean model, and k = 206 (AUC = 0.6342) for the weighted Manhattan model. However, we determined an optimal k-value when using the complete dataset by using scores of submissions to Kaggle. With fewer tests, we determined the optimal values k = 150 (AUC = 0.6457) for the weighted Manhattan distance model and k = 100 (AUC = 0.6429) for the weighted Euclidean distance model.

3.2.2 Support Vector Machine

We did SVM in various ways with different kernels. Because SVM from scikit-learn library did not directly support multi-labels classification, we firstly tried to transform the dataset such that SVM can predict the the dataset by each single GO term. The methods we used to transform the dataset include Binary Relevance, Power Labelset and Classifier Chain. For simplicity of checking which way works best, we separated the dataset by 80 percent as training set and 20 percent as test set, and trained SVM with default hyper-parameters from scikit-learn. With Binary Relevance, the problem was broken into different single class classification for each GO term and we got an AUC score 0.6596. With Power Labelset, a unique class was given to every possible label combination such that the problem is treated as single class classification and we got an AUC score 0.5767. With Classifier Chain, the first classifier was trained just on the input data and then each next classifier was trained on the input space and all the previous classifiers in the chain such that it preserves label correlation. The AUC we got with Classifier Chain was 0.6615.[5] Even though Classifier Chain performed best as a dataset transformation method, we decided to use Binary Relevance for the rest of our evaluation because it was less computationally expensive with only a small decrease in performance.

We also tested SVM kernels by creating a separate model for each GO term on all 625 features using 5-fold cross-validation. Two of the kernels, sigmoid and linear, had an average ROC-AUC score lower than 0.6 (0.5936 and 0.5938 respectively). The rbf, polynomial with degree 2, and polynomial with degree 3 kernels had scores 0.6210, 0.6166, and 0.6288 respectively.

3.2.3 Neural Network

Firstly, we tried Deep Feed Forward NN as baseline model with Tensorflow Keras library. For our first model building attempt, we set the hyper-parameters manually to be two layers with 0.5 dropout rate, 160 nodes per layer, binary cross entropy loss function and Adam optimizer with 0.001 learning rate. By applying 10-folds cross validation, we got mean AUC score 0.6545. Then, we decided to apply hyper-parameters tuning based on Bayesian Optimization by using Keras Tuner library with 50 trials. Originally, we set the search space to be 1 to 10 hidden layers, 140 to 250 for each layers, drop out rate to be 0.2 to 0.5, learning rate to be 1e-4 to 1e-2 with ReLu activation function for each hidden layer, sigmoid activation function for output layer, Adam as optimizer, binary cross entropy as loss function, batch size to be 150 and epochs to be 80. By running these experiments, we observed that adding layers after two hidden layers did not improve the performance for this dataset. In contrast, the performance dropped rapidly as we kept adding layers. Then, we tried to change the hidden layers in search space to be 1 to 3 and 1 to 4. By re-tuning the hyper-parameters with these two search space, we found that 1 to 3 layers tuning yielded our best AUC score of 0.6619 with the following model architecture. A two-layer NN with hidden nodes 200, 180, drop out rate 0.350, 0.399 and learning rate 0.001414. After getting this hyper-parameter optimization, we attempted to optimize the architecture by adding batch normalization layers. However, with batch normalization layers added and re-tuned hyper-parameters by Bayesian Optimization, it was not able to beat our previous architecture with best score 0.6511 from Bayesian Optimization and 10-folds cross validation score 0.6443. Additionally, by examining the validation AUC score vs. training AUC score, we decided to choose epochs to be 52 as we observed that the model started to overfit the training set after approximately 52 epochs. Fig.5 in next page from one fold showed how the validation AUC and training AUC changes for each epoch until 100 epochs.

By applying 10-folds cross validation with this architecture, we got a mean AUC score of 0.6608, which was the best score we got by solely predicting the dataset with NN. In order to verify our assumption that adding more layers larger than 2 will decrease the model's performance rapidly, we trained models by 10-folds cross

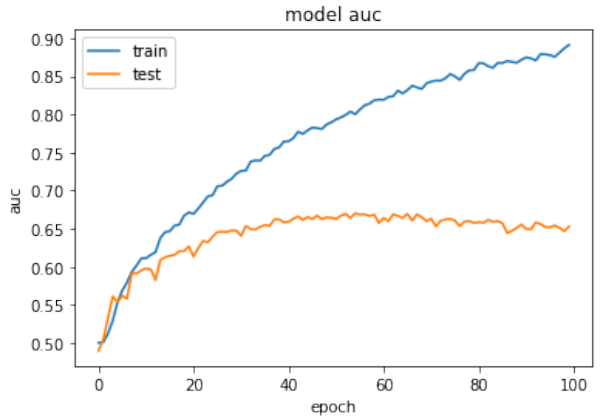


Figure 5: Validation ROC-AUC vs. Epochs

As this figure showed, an obvious overfitting effect started at around 45 to 55 epochs

validation with layers from 3 to 10 by adding layers with 100 hidden nodes and 0.35 dropout rate each time. Fig.6 in next page shows how mean AUC scores change when we keep adding more layers.

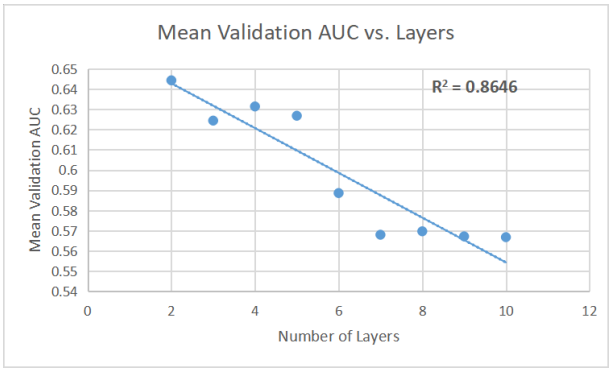


Figure 6: Validation AUC vs. Layers

As this figure showed, AUC scores performed a decreasing trend as number of layers increase with a linear regression R square value 0.8646

3.2.4 Blending Predictions

According to our observations by calculating per GO term AUC score with 5-fold cross validation for our best tuned NN and kNN with k=100 and Euclidean distance as the proximity metric, there were many common GO terms with similar performance (consistent performance on high or low AUC on both models). By sorting and comparing their GO terms for top 50 and worst 50 GO terms based on their mean AUC score, we found that they have 41 common GO terms in top 50 and 27 GO terms in the worst 50. Among these common GO terms, 28 of them had a higher AUC for kNN with k=100 and Euclidean distance as proximity metric from top 50, and 10 of them have higher AUC for kNN from worst 50. From the first attempt, we replaced prediction columns from NN with prediction columns from kNN for these kNN GO terms that have higher AUC. As a result, our AUC score in Kaggle was able to increase by 0.0026.

Then, we attempted to stack all models with decent performance from our experiments. The models included NN, weighted kNN with k=150 and Manhattan distance, weighted kNN with k=100 and Euclidean distance, SVM with rbf kernel, SVM with degree 2 polynomial kernel, SVM with degree 3 polynomial kernel. We stacked them by extracting the columns with best mean 5-fold cross validation AUC score per GO term among all of them. As a result, we achieved AUC score with 0.65116 on Kaggle. For the second blending method we attempted, we assigned a weight for the predictions of each model based on how many GO terms each model wins among 200 total GO terms and added them up by multiplying their corresponding fractions(number of wins divides by 200). As a result, we achieved AUC score with 0.67206 on Kaggle(increased by 0.02151). The number of wins for each model is shown in TABLE 1.

Table 1: Number of Wins Among All Models

Models	Number of Wins
kNN(Euclidean)	34
kNN(Manhattan)	33
SVM(RBF)	22
SVM(Poly 2)	25
SVM(Poly 3)	25
NN	**61**

As this table showed, NN got significantly higher wins among all models, followed by two kNN models.

Taking credit from Garrison Shea in this challenge, who found that GNB achieved best performance with 0.65653 Kaggle score as single model without blending, we tried to duplicate this observation and were able to get same score by training this dataset with z-score normalization and Binary Relevance dataset transformation. Then, we repeated our weighted blending for GNB with all previous six models. As a result, we achieved 0.67582 score on Kaggle, which was a noticeable improvement compared to the result of six-model blending. The number of wins for each model (later used for seven-model blending) is shown in TABLE 2.

Table 2: Number of Wins Among All Models

Models	Number of Wins
kNN(Euclidean)	27
kNN(Manhattan)	16
SVM(RBF)	19
SVM(Poly 2)	18
SVM(Poly 3)	19
NN	50
GNB	**51**

As this table showed, NN and GNB got a very close number of wins and outcompete rest of models

After we submitted all our attempted models, the Kaggle scores for all our submitted models with decent performance are shown in TABLE 3 in page 6.

To investigate how different models were performing with individual GO terms, a separate model was created for each one. We created histograms to show the distribution of AUC scores for some models we evaluated in Fig.7 in page 7.

3.3 Biological Result

In order to prove our assumption that there are GO terms that were easier or harder to predict across models, we calculated the Pearson correlation coefficient of ROC-AUC scores for each GO term between every pair of models. TABLE 4 in page 7 shows Pearson correlation coefficients from different combinations.

To show how the difficulty of predicting each GO term can vary, we showed top 10 AUC scores for our NN and kNN models in Fig.8 and Fig.9 in page 8. We then extracted the top and worst GO terms by averaging the ROC-AUC scores across all models for every GO term for the original top six models(exclude GNB). Fig.10 and Fig.11 in page 8 and 9 show their corresponding GO Annotations and Biological Process. We found that most top 20 annotations are associated with cellular processes and metabolic, and most worst 20 annotations have more variation of biological functions.

4 Discussion and Conclusion

Accurate annotation of human gene functions remains challenging as there is an abundant amount of data being released as a direct result of technological advances that allow for more rapid results of genome sequencing. Since the release of the human genome project, there is yet a vast need to directly study or characterize the function of a large number of genes. Although current methods use experimental approaches to validate gene function, these studies are both costly and require a lot of time to perform, especially with the large amounts of data that is currently available and for the data that is yet to be generated. In simple modeled organisms, these studies have suggested that genes that exhibit highly similar dependency profiles tend to be involved in the same protein complex, pathway or biological process[4]. In this study, we sought to test this concept by evaluating different supervised machine learning algorithms' to predict functional annotations using a set of human gene interaction profiles from cancer cell lines.

The results of this study showed that there were differences among the performances of the models across GO terms. For some GO terms, a proposed method showed better prediction accuracy, given by a higher AUC score, whereas for other GO terms, it gave less accurate predictions (see Fig.8 and Fig.9 — Top 10 NN AUCs and Top 10 kNN AUCs). Given this observation, we decided to take the top and worst 20 GO terms for our best performing model, the weighted stacked model, to determine which functional domains are among the best and worst ranked. As seen in the Top 20 table, almost all annotations are associated with cellular processes and the annotations also appear to skew towards metabolic processes, while the worst 20 slightly skew more towards biological regulation. Overall, the top ranked GO-terms from our blending model included several annotations related to the cell cycle and DNA replication/repair, while the GO-terms that were predicted with the lowest accuracy had more varied biological roles, such as those related to memory or response to chemicals such as ethanol.

As these results highlight, the increased performance from blending seven models verified the feasibility of our blending method and supervised machine learning methods hold a great potential to efficiently and accurately annotate unknown human gene functions. However, it is important to note the limitations of this strategy. Our models were limited to a subset of genomic expression profiles for a specific set of human cancer cell lines, future studies should include training using a multi-omics approach to integrate different sets of features, which will most likely strengthen the models performances and thus give better biological and computational predictions. Other factors to consider are the true recall of supervised machine learning methods, which may be lower or even higher than the estimated recall, as some false negatives or false positives may in fact represent errors in the underlying functional annotations. Therefore accuracy calculated may be higher or lower than the true values.

Although we did not expect supervised machine learning to immediately replace laboratory validation studies, these models served as powerful tools to shorten the time required for these analyses. Future studies should continue working on improving predictive performance through experimenting on other ensemble learning methods, attempting to stack more high performance models, exploring other feature selection combinations, modifying neural network architectures, and applying bootstrapping sampling techniques. In addition, the integration of more omics data for the characterization of unknown functional annotations will further help with the training of more accurate models. Other data that one might want to include stem from more exhaustive protein and mRNA expression data studies, such that there are a vast amount of stress conditions or experimentally derived protein-protein interaction data. Regardless, the main computational challenges that will remain in future studies are the following: 1) how to integrate these different data sources 2) how to deal with incomplete and noisy data, 3) how to interpret the results, 4) where to store all of these data.

References

[1] Dhammika Amaratunga and Javier Cabrera. Analysis of data from viral dna microchips. *Journal of the American Statistical Association*, 96(456):1161–1170, 2001.

[2] Ando Saabass. Selecting good features – Part III: random forests. https://blog.datadive.net/selecting-good-features-part-iii-random-forests/, 2014.

[3] Andreas Zell. *Simulation Neuronaler Netze*. Addison-Wesley, 1 edition, 1994.

[4] Costanzo et al. A global genetic interaction network maps a wiring diagram of cellular function. *Science*, 353:pii: aaf1420, 2016.

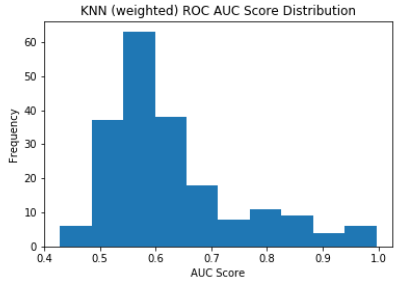
[5] Shubham Jain. Solving multi-label classification problems. https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/, 2017.

[6] Leo Breiman. Random Forest. *Machine Learning*, 45:5–32, 2001.

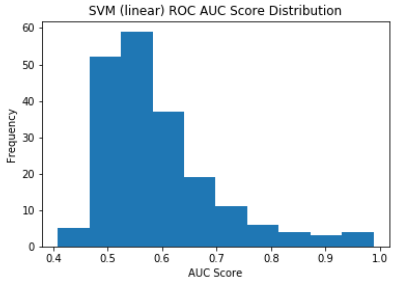
[7] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Anuj Karpatne. *Introduction to Data Mining*. Pearson, 2 edition, 2019.

[8] Polley, Eric C. and van der Laan, Mark J. Super Learner In Prediction. In *U.C. Berkeley Division of Biostatistics Working Paper Series*, Working Paper 266. May 2010.

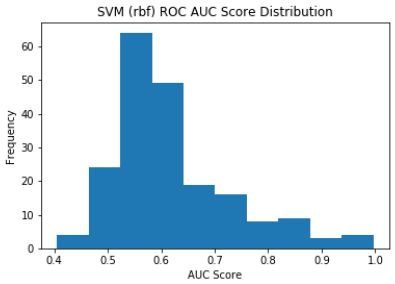
[9] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.



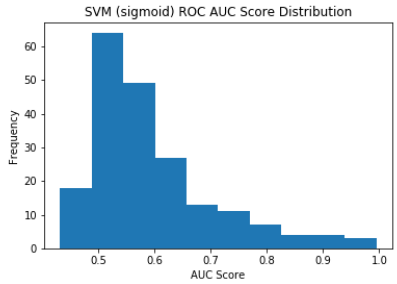
(a) weighted kNN (Euclidean)



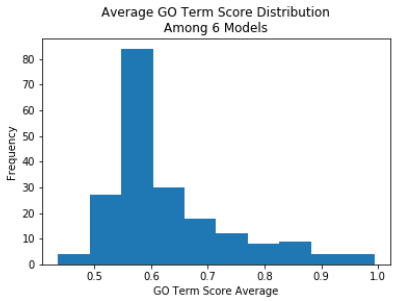
(b) SVM Linear



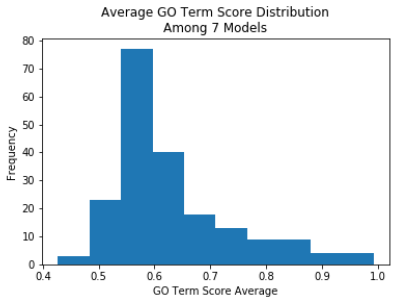
(c) SVM RBF



(d) SVM Sigmoid



(e) Average scores across six models



(f) Average scores across seven models

Figure 7: Histogram for ROC-AUC Distribution of all GO terms for different models

The six models we took average are kNN weighted Manhattan k=150, kNN weighted Euclidean k=100, SVM RBF, SVM polynomial of degree 2, SVM polynomial of degree 3, and NN. GNB was added for the average of seven models. As these figures showed, even though most GO terms distribute around ROC-AUC scores between 0.5 and 0.6, models with higher performance tend to have more GO terms that are close to ROC-AUC scores of 0.6.

GO Annotation	NN AUC score
mitochondrial translational termination (GO:0070126)	0.99453
mitochondrial translational elongation (GO:0070125)	0.99432
SRP-dependent cotranslational protein targeting to membrane (GO:0006614)	0.97559
viral transcription (GO:0019083)	0.95823
nuclear-transcribed mRNA catabolic process, nonsense-mediated decay (GO:0000184)	0.93833
translational initiation (GO:0006413)	0.91998
keratinization (GO:0031424)	0.91439
positive regulation of nucleic acid-templated transcription (GO:1903508)	0.90880
rRNA processing (GO:0006364)	0.89317

Figure 8: Top 10 NN AUCs

Table 3: Result of Kaggle Scores

Models	Kaggle Scores
kNN(Euclidean)	0.64292
kNN(Manhattan)	0.64569
SVM(RBF)	0.62534
SVM(Poly 2)	0.62246
SVM(Poly 3)	0.64162
NN	0.65055
GNB	0.65653
Blending(NN and kNN with Euclidean - Common Top and Worst 50 GO Terms)	0.65315
Blending(NN and kNN with Euclidean - All GO Terms)	0.65244
Blending(All Six Baseline Models - All GO terms)	0.65116
Blending(Weighted by Wins with Six Baseline Models - All GO terms)	0.67206
Blending(Weighted by Wins with Seven Baseline Models - All GO terms)	**0.67582**

As this table showed, blending weighted predictions based on cross validation ROC-AUC with seven models yielded best score on Kaggle. The bold font shows model with highest performance

Table 4: GO Term Scores Pearson Correlation Coefficients

	kNN Weighted Manhattan	kNN Weighted Euclidean	SVM Polynomial 2	SVM Polynomial 3	SVM RBF	NN	GNB
kNN Weighted Manhattan	–	**0.9615**	0.9478	0.9596	0.9400	0.8828	0.9097
kNN Weighted Euclidean	**0.9615**	–	0.9285	0.9492	0.9341	0.8869	0.9221
SVM Polynomial 2	**0.9478**	0.9285	–	0.8942	0.9282	0.8449	0.8887
SVM Polynomial 3	**0.9596**	0.9492	0.8942	–	0.9479	0.8855	0.9174
SVM RBF	0.9400	0.9341	0.9282	**0.9479**	–	0.8745	0.9128
NN	0.8828	**0.8869**	0.8449	0.8855	0.8745	–	0.8837
GNB	0.9097	**0.9221**	0.8887	0.9174	0.9128	0.8837	–

As this table showed, all combinations preserve strong Pearson correlations, which indicates that the difficulty of predicting each GO term can be very similar across models. The bold font shows model with highest correlation value with model in current row.

GO Annotation	kNN AUC score
mitochondrial translational elongation (GO:0070125)	0.99622
mitochondrial translational termination (GO:0070126)	0.99618
SRP-dependent cotranslational protein targeting to membrane (GO:0006614)	0.97065
viral transcription (GO:0019083)	0.96932
nuclear-transcribed mRNA catabolic process, nonsense-mediated decay (GO:0000184)	0.95798
translational initiation (GO:0006413)	0.94232
keratinization (GO:0031424)	0.93907
rRNA processing (GO:0006364)	0.91673
mRNA export from nucleus (GO:0006406)	0.89704

Figure 9: Top 10 kNN AUCs (Manhattan Distance, k=150)

Top 20 GO Terms, Top 6 Models	GO Annotations	Biological Process
GO:0070126	*mitochondrial translational termination*	cellular / metabolic
GO:0070125	*mitochondrial translational elongation*	cellular / metabolic
GO:0006614	*SRP-dependent cotranslational protein targeting to membrane*	cellular / localization
GO:0019083	*viral transcription*	interspecies interaction between
GO:0031424	*keratinization*	cellular / developmental process / multicellular organismal process
GO:0006413	*translational initiation*	cellular / metabolic
GO:0000184	*nuclear-transcribed mRNA catabolic process, nonsense-mediated decay*	cellular / metabolic / biological regulation
GO:0006364	*rRNA processing*	cellular / metabolic
GO:0006406	*mRNA export from nucleus*	cellular / metabolic / localization
GO:0031145	*anaphase-promoting complex-dependent catabolic process*	cellular / metabolic
GO:0006325	*chromatin organization*	cellular
GO:0070268	*cornification*	cellular / developmental process
GO:1903508	*positive regulation of nucleic acid-templated transcription*	cellular / metabolic / biological regulation
GO:0006412	*translation*	cellular / metabolic
GO:0006260	*DNA replication*	cellular / metabolic
GO:0000724	*double-strand break repair via homologous recombination*	cellular / response to stimulus / metabolic
GO:0006338	*chromatin remodeling*	cellular
GO:0010389	*regulation of G2/M transition of mitotic cell cycle*	cellular / biological regulation
GO:1901990	*regulation of mitotic cell cycle phase transition*	cellular / biological regulation
GO:1901796	*regulation of signal transduction by p53 class mediator*	cellular / response to stimulus / signaling / biological regulation

Figure 10: Top 20 GO terms

Top 20 out of six-model average (kNN weighted Manhattan k=150, kNN weighted Euclidean k=100, SVM RBF, SVM polynomial of degree 2, SVM polynomial of degree 3, and NN). Almost all annotations are associated with cellular processes (makes sense considering training data is from cell lines) and annotations also appear to skew towards metabolic processes. We see several involved in the cell cycle process and one involving p53 which makes sense as our models were trained on cancer cell line data.

Worst 20 GO Terms, Top 6 Models	GO Annotations	Biological Process
GO:0007613	*memory*	multicellular organismal / behavior
GO:0042981	*regulation of apoptotic process*	cellular / biological regulation
GO:0007605	*sensory perception of sound*	multicellular organismal
GO:0065003	*protein-containing complex assembly*	cellular
GO:0042127	*regulation of cell population proliferation*	cellular / biological regulation
GO:0000187	*activation of MAPK activity*	cellular / metabolic / biological regulation / response to stimulus
GO:0002576	*platelet degranulation*	cellular / localization
GO:0007015	*actin filament organization*	cellular
GO:0001934	*positive regulation of protein phosphorylation*	cellular / metabolic / biological regulation
GO:0030336	*negative regulation of cell migration*	cellular / biological regulation / localization
GO:0010976	*positive regulation of neuron projection development*	cellular / biological regulation / developmental / multicellular organismal
GO:0042493	*response to drug*	response to stimulus
GO:0030334	*regulation of cell migration*	cellular / biological regulation / localization / locomotion
GO:0031647	*regulation of protein stability*	biological regulation
GO:0006629	*lipid metabolic process*	metabolic
GO:0051289	*protein homotetramerization*	cellular
GO:0045471	*response to ethanol*	response to stimulus
GO:0030307	*positive regulation of cell growth*	cellular / growth / biological regulation
GO:0008584	*male gonad development*	developmental / multicellular organismal / reproduction / reproductive
GO:0071277	*cellular response to calcium ion*	cellular / response to stimulus

Figure 11: Worst 20 GO terms

Worst 20 out of six-model average (kNN weighted Manhattan k=150, kNN weighted Euclidean k=100, SVM RBF, SVM polynomial of degree 2, SVM polynomial of degree 3, and NN). Some skew towards biological regulation but it should be noted that even for our best model, these worst 20 had AUC scores only slightly better than 0.50, meaning the model predicted little better than random chance with these terms.