Identification of Disease-Associated Genes using ML

Nisarg Upadhyaya - 19CS30031
Neha Dalmia - 19CS30055

CBP Project [Group 14]
Academic Year 2023-24

1. Introduction

Understanding the link between genes and diseases is crucial for medical research. By identifying which genes are associated with certain illnesses, researchers can gain insights into the underlying causes of diseases. This knowledge allows for the development of targeted treatments and interventions, leading to more effective healthcare strategies. Additionally, studying gene-disease associations can help in early detection and diagnosis of conditions, enabling timely medical interventions and potentially improving patient outcomes. ML can sift through vast amounts of protein sequence data to pinpoint subtle correlations and associations that might not be apparent through traditional methods. This ability to extract intricate patterns allows for a deeper understanding of how proteins contribute to disease processes.

2. Dataset

The dataset is obtained from DisGeNET [1]. It has information on diseases associated with 21671 genes. Each gene can be associated with multiple diseases. The total number of such gene-disease interactions in the dataset are 3261324. The MeSH [2] class of each disease is also present in the database (if available). The gene sequences are downloaded from NCBI using their *datasets* utility. For each geneID we download all possible isoforms of the possible protein sequences. This is because different protein structures can be associated with the same disease and we must capture this information. We clean up the data and the final data statistics are available in the table below.

Statistic	Count
Total sequences	102827
Disease associated sequences	59967
Non disease associated sequences	42860
MeSH disease classes	26

3. Feature Extraction

Before proceeding with training our models on the sequences we need to create a feature representation for the sequences incorporating the sequence and physicochemical information. For this we use the FEPS [3] utility. The features extracted from FEPS do not require subsequent processing and are ready to be fed to the machine learning techniques as it provides various output formats as well as the ability to concatenate these generated features. The following features were included using FEPS:

1. Quasi and sequence-order-coupling: By incorporating information about the spatial relationships between amino acids, SOC features can improve the performance of machine learning algorithms in tasks such as protein classification, prediction, and function annotation. (full list in Appendix)
2. Autocorrelation descriptors: ACDs can be used to capture patterns or correlations in the properties of amino acids along the sequence. For example, they can measure how the properties of amino acids at one position in the sequence correlate with those at nearby positions. (full list in Appendix)
3. Entropy, relative entropy, and gain
4. Composition, transition and distribution: Composition reflects amino acid frequencies. Transition tracks changes between adjacent amino acids. Distribution assesses spatial patterns along the sequence. (full list in Appendix)
5. Conjoint triad: It involves grouping amino acids into overlapping triplets, known as triads, and encoding them based on their physicochemical properties. This representation captures spatial information and structural motifs.

Combining all the different feature vectors from above together we got a total of 1094 features per sequence. For traditional ML methods, this number is very large. Hence we incorporate PCA the feature vectors obtained above and select the top 10 and 20 principal components and report the results. With this the dimensionality is reduced to 43 and 83 columns respectively based on the number of components selected.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) involves transforming high-dimensional data into a reduced set of uncorrelated components (principal components) capturing the most significant variations. It identifies patterns by projecting data onto new axes, ordered by variance. PCA condenses information into fewer components, preserving essential features while reducing dimensionality.

4. Classifiers

Once we have extracted the relevant features, we train the model using the following approaches:

a. Support Vector Machine [4]
b. Gradient Boosting [5]
c. Random Forest [6]
d. Ada Boost [7]
e. Quadratic Discriminant Analysis [8]

For each classifier we split the dataset into 75-25 train-test data.

Further we also go one step ahead and try to build a neural network architecture which can predict the possible disease MeSH classes a gene might be linked to. To this end we develop a neural network with the following configuration. The input is a 1094 length feature vector and the output is the probability vector associated with each of the 26 MeSH classes.

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 1094)	0
dense_4 (Dense)	(None, 512)	560,640
dense_5 (Dense)	(None, 256)	131,328
dense_6 (Dense)	(None, 128)	32,896
dense_7 (Dense)	(None, 26)	3,354

Total params: 728,218 (2.78 MB)
Trainable params: 728,218 (2.78 MB)
Non-trainable params: 0 (0.00 B)

a. Support Vector Machine

Metric	43 Features	83 Features
Precision	0.70	0.73
Recall	0.66	0.70
F1-Score	0.61	0.67
Accuracy	0.66	0.70

b. Gradient Boosting

Metric	43 Features	83 Features
Precision	0.68	0.70
Recall	0.67	0.68
F1-Score	0.64	0.65
Accuracy	0.67	0.68

c. Random Forest

Metric	43 Features	83 Features
Precision	0.97	0.97
Recall	0.97	0.97
F1-Score	0.97	0.97
Accuracy	0.97	0.97

d. Ada Boost

Metric	43 Features	83 Features
Precision	0.61	0.63
Recall	0.62	0.63
F1-Score	0.60	0.61
Accuracy	0.62	0.63

e. Quadratic Discriminant Analysis

Metric	43 Features	83 Features
Precision	0.59	0.63
Recall	0.60	0.62
F1-Score	0.53	0.56
Accuracy	0.60	0.62

5. Neural Network Architecture

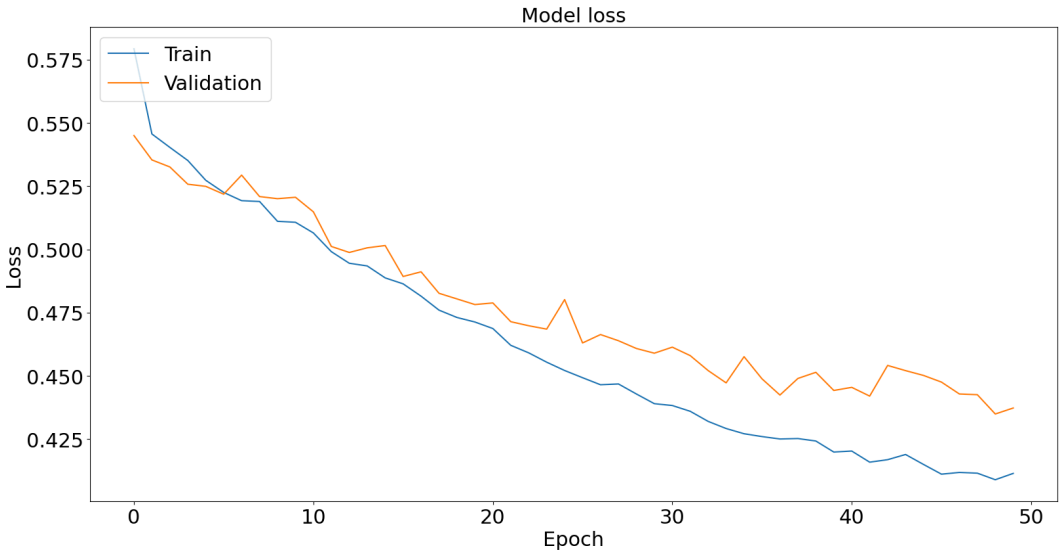
As the dimensionality of our data is very high and we want to do a multilabel classification it is an ideal choice to adapt a neural network based deep learning approach. First we identify the MeSH disease classes to which a gene is associated with. There are 26 such classes and hence we can use a one-hot encoding to map genes to disease classes. Each index of the encoding vector represents one disease class. The disease class to index mapping is given on the next page.

We then initialize a neural network with 3 hidden layers, with 512, 256 and 128 neurons respectively. All layers use the ReLU activation function. Further, as we have to predict for 26 classes we use a 26 neuron output layer with the sigmoid activation function to get the probabilities. The loss function used is BinaryCrossentropy. This combination of output activation and loss function is well suited for multi label classification. The model is trained for 50 epochs with the Adam optimizer.

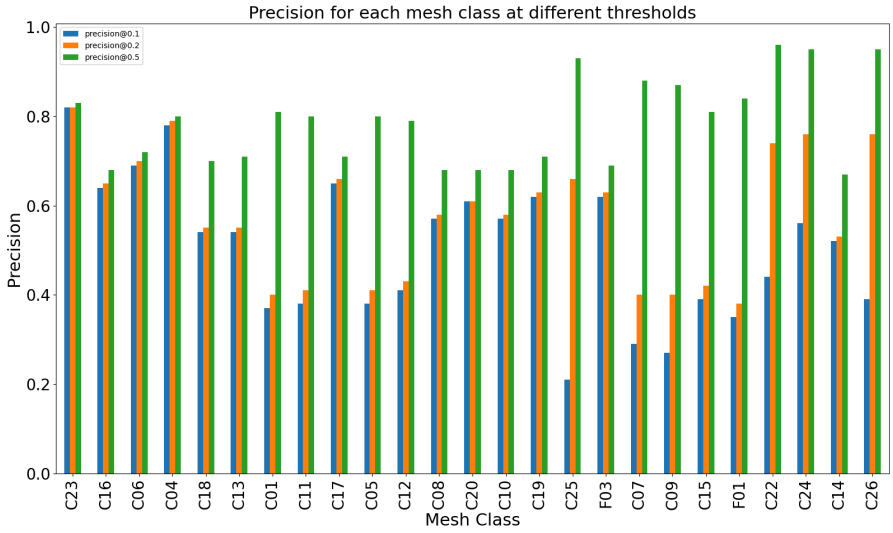
Index	MeSH Class	Class Name
0	C23	Pathological Conditions, Signs and Symptoms
1	C16	Congenital, Hereditary, and Neonatal Diseases and Abnormalities
2	C06	Digestive System Diseases
3	C04	Neoplasms
4	C18	Nutritional and Metabolic Diseases
5	C13	Female Urogenital Diseases and Pregnancy Complications
6	C01	Infections
7	C11	Eye Diseases
8	C17	Skin and Connective Tissue Diseases
9	C05	Musculoskeletal Diseases
10	C12	Male Urogenital Diseases
11	C08	Respiratory Tract Diseases
12	C20	Immune System Diseases
13	C10	Nervous System Diseases
14	C19	Endocrine System Diseases
15	C25	Chemically-Induced Disorders
16	F03	Mental Disorders
17	C07	Stomatognathic Diseases

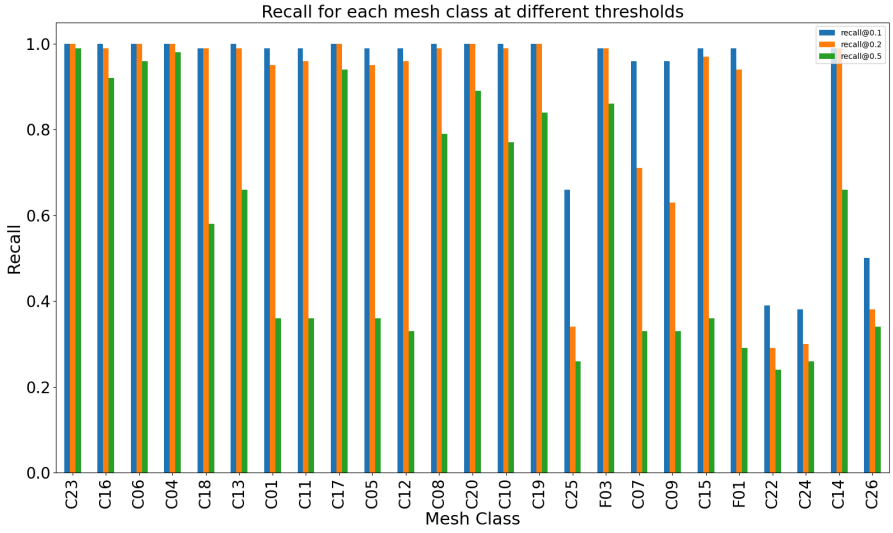
18	C09	Otorhinolaryngologic Diseases
19	C15	Hemic and Lymphatic Diseases
20	F01	Behavior and Behavior Mechanisms
21	C22	Animal Diseases
22	C24	Occupational Diseases
23	C14	Cardiovascular Diseases
24	C26	Wounds and Injuries
25	C21	Disorders of Environmental Origin

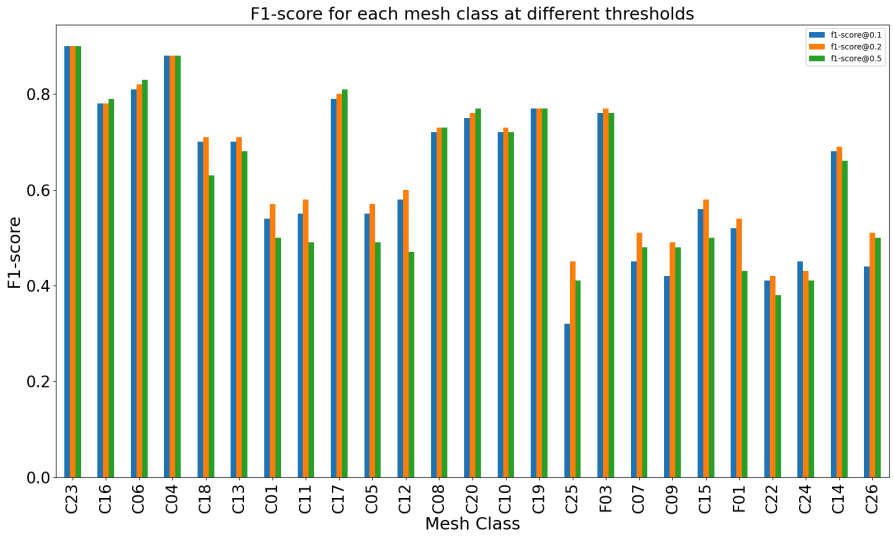
The loss vs epochs curve is given below. We report the weighted precision, recall and F1-score for three threshold values 0.1, 0.2 and 0.5. The per class scores are available in the appendix. Threshold values are the probability required for a gene to be assigned a certain MeSH class. So the gene is assigned those MeSH classes whose index positions in the output vector from the neural network contain a value greater than this threshold.



Metric	p > 0.1	p > 0.2	p > 0.5
Precision	0.57	0.60	0.75
Recall	0.98	0.96	0.72
F1-Score	0.71	0.72	0.70







6. Conclusion

In our study, the classifiers, particularly Random Forest, exhibited exceptional performance with precision and recall scores reaching 97% across varying feature sets. Additionally, the neural network architecture effectively predicted gene associations with MeSH disease classes, achieving a precision of 75% with a threshold of 0.5. These results underscore the robustness and versatility of our approach in accurately identifying disease-associated genes. Furthermore, the successful reduction of feature dimensionality through PCA highlights the efficacy of our methodology in handling high-dimensional data while maintaining predictive power.

Further for the neural network architecture note that when the threshold is low, the classifier is more likely to classify instances as positive, leading to higher recall but lower precision. This is because the classifier is capturing more true positives but also more false positives, resulting in a larger denominator for precision calculation. Conversely, as the threshold increases, the classifier becomes more conservative in predicting positive instances, resulting in fewer false positives but also fewer true positives. This leads to higher precision but lower recall because the true positives are a smaller proportion of the total positive instances.

Code: https://github.com/nisarg1631/CBP-Project

Appendix

1. **Classification Report for p > 0.1**

Classification report 0.1:

	precision	recall	f1-score	support
0	0.82	1.00	0.90	6567
1	0.64	1.00	0.78	4995
2	0.69	1.00	0.81	5426
3	0.78	1.00	0.88	6239
4	0.54	0.99	0.70	4022
5	0.54	1.00	0.70	4010
6	0.37	0.99	0.54	2706
7	0.38	0.99	0.55	2723
8	0.65	1.00	0.79	5112
9	0.38	0.99	0.55	2772
10	0.41	0.99	0.58	2933
11	0.57	1.00	0.72	4304
12	0.61	1.00	0.75	4698
13	0.57	1.00	0.72	4321
14	0.62	1.00	0.77	4805
15	0.21	0.66	0.32	882
16	0.62	0.99	0.76	4820
17	0.29	0.96	0.45	1818
18	0.27	0.96	0.42	1644
19	0.39	0.99	0.56	2776
20	0.35	0.99	0.52	2499
21	0.44	0.39	0.41	446
22	0.56	0.38	0.45	152
23	0.52	0.99	0.68	3910
24	0.39	0.50	0.44	736
25	0.00	0.00	0.00	4
micro avg	0.53	0.98	0.69	85320
macro avg	0.48	0.87	0.61	85320
weighted avg	0.57	0.98	0.71	85320
samples avg	0.55	0.98	0.65	85320

2. Classification Report for p > 0.2

Classification report 0.2:

	precision	recall	f1-score	support
0	0.82	1.00	0.90	6567
1	0.65	0.99	0.78	4995
2	0.70	1.00	0.82	5426
3	0.79	1.00	0.88	6239
4	0.55	0.99	0.71	4022
5	0.55	0.99	0.71	4010
6	0.40	0.95	0.57	2706
7	0.41	0.96	0.58	2723
8	0.66	1.00	0.80	5112
9	0.41	0.95	0.57	2772
10	0.43	0.96	0.60	2933
11	0.58	0.99	0.73	4304
12	0.61	1.00	0.76	4698
13	0.58	0.99	0.73	4321
14	0.63	1.00	0.77	4805
15	0.66	0.34	0.45	882
16	0.63	0.99	0.77	4820
17	0.40	0.71	0.51	1818
18	0.40	0.63	0.49	1644
19	0.42	0.97	0.58	2776
20	0.38	0.94	0.54	2499
21	0.74	0.29	0.42	446
22	0.76	0.30	0.43	152
23	0.53	0.99	0.69	3910
24	0.76	0.38	0.51	736
25	0.00	0.00	0.00	4
micro avg	0.57	0.96	0.71	85320
macro avg	0.56	0.82	0.63	85320
weighted avg	0.60	0.96	0.72	85320
samples avg	0.58	0.96	0.67	85320

3. Classification Report for p > 0.5

Classification report 0.5:

	precision	recall	f1-score	support
0	0.83	0.99	0.90	6567
1	0.68	0.92	0.79	4995
2	0.72	0.96	0.83	5426
3	0.80	0.98	0.88	6239
4	0.70	0.58	0.63	4022
5	0.71	0.66	0.68	4010
6	0.81	0.36	0.50	2706
7	0.80	0.36	0.49	2723
8	0.71	0.94	0.81	5112
9	0.80	0.36	0.49	2772
10	0.79	0.33	0.47	2933
11	0.68	0.79	0.73	4304
12	0.68	0.89	0.77	4698
13	0.68	0.77	0.72	4321
14	0.71	0.84	0.77	4805
15	0.93	0.26	0.41	882
16	0.69	0.86	0.76	4820
17	0.88	0.33	0.48	1818
18	0.87	0.33	0.48	1644
19	0.81	0.36	0.50	2776
20	0.84	0.29	0.43	2499
21	0.96	0.24	0.38	446
22	0.95	0.26	0.41	152
23	0.67	0.66	0.66	3910
24	0.95	0.34	0.50	736
25	0.00	0.00	0.00	4
micro avg	0.73	0.72	0.72	85320
macro avg	0.76	0.56	0.60	85320
weighted avg	0.75	0.72	0.70	85320
samples avg	0.69	0.71	0.65	85320

4. Complete set of features extracted using FEPS

AMINO ACID DESCRIPTORS

1. Composition Translation Distribution of AADs
2. Composition descriptors of AADs
3. Translation descriptors of AADs
4. Distribution descriptors of AADs
5. Composition descriptors based on charge of AADs
6. Composition descriptors based on hydrophobicity of AADs
7. Composition descriptors based on normalized VDWV of AADs
8. Composition descriptors based on polarity of AADs
9. Composition descriptors based on polarizability of AADs
10. Composition descriptors based on 2ndary structure of AADs
11. Composition descriptors based on solvent accessibility of AADs
12. Distribution descriptors based on charge of AADs
13. Distribution descriptors based on hydrophobicity of AADs
14. Distribution descriptors based on normalized VDWV of AADs
15. Distribution descriptors based on polarity of AADs
16. Distribution descriptors based on polarizability of AADs
17. Distribution descriptors based on 2ndary structure of AADs
18. Distribution descriptors based on solvent accessibility of AADs
19. Transition descriptors based on charge of AADs
20. Transition descriptors based on hydrophobicity of AADs
21. Transition descriptors based on normalized VDWV of AADs
22. Transition descriptors based on polarity of AADs
23. Transition descriptors based on polarizability of AADs
24. Transition descriptors based on 2ndary structure of AADs
25. Transition descriptors based on solvent accessibility of AADs

AUTOCORRELATION DESCRIPTORS

1. Geary autocorrelation
2. Moran autocorrelation descriptors
3. Normalized Moreau-Broto autocorrelation

SEQUENCE ORDER COUPLING

1. Quasi sequence order descriptors
2. Sequence order coupling numbers

References

[1] Janet Piñero, Josep Saüch, Ferran Sanz, & Laura I. Furlong (2021). The DisGeNET cytoscape app: Exploring and visualizing disease genomics data. Computational and Structural Biotechnology Journal, 19, 2960-2967.

[2] https://www.ncbi.nlm.nih.gov/mesh/

[3] Ismail, H., White, C., Al-Barakati, H., Newman, R. H., & Kc, D. B. (2022). FEPS: A Tool for Feature Extraction from Protein Sequence. Methods in molecular biology (Clifton, N.J.), 2499, 65–104. https://doi.org/10.1007/978-1-0716-2317-6_3

[4] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

[5] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, 1189–1232.

[6] Ho, T. K. (1995). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278–282).

[7] Schapire, R. E. (2013). Explaining adaboost. In Empirical inference (pp. 37–52). Springer.

[8] Benyamin Ghojogh, & Mark Crowley. (2019). Linear and Quadratic Discriminant Analysis: Tutorial.

Libraries Used

1. Scikit-learn for
    a. PCA
    b. Train Test Split
    c. Calculating metrics
    d. Support Vector Machine
    e. Gradient Boosting
    f. Random Forest
    g. Ada Boost
    h. Quadratic Discriminant Analysis
2. Numpy and Pandas for data handling
3. Pickle for saving/loading trained models
4. Matplotlib for plotting curves
5. Tensorflow for neural network architecture