Deep phenotyping is defined as learning about genotype-phenotype associations and the history of human illness by analyzing phenotypic anomalies. It is significant to investigate the association between phenotype and genotype. Machine learning approaches are good at predicting the associations between abnormal human phenotypes and genes. A novel framework based on machine learning is proposed to estimate the links between human phenotype ontology (HPO) and genes. Orphanet’s annotation parses the human phenotype-gene associations. An algorithm node2vec generates the embeddings for the nodes (HPO and genes). It performs node sampling on the graph using random walks and learns features on these sampled nodes for embedding. These embeddings were used downstream to predict the link between these nodes by supervised classifiers. Results show the gradient boosting decision tree model (LightGBM) has achieved an optimal AUROC of 0.904 and an AUCPR of 0.784, an optimal weighted F1 score of 0.87. LightGBM can detect more accurate interactions and links between human phenotypes and gene pairs.

1. Introduction

Today, many humans have diseases with abnormalities in the genome, and because of their nonuniformity, diseases are missed or undiagnosed. The analysis of human phenotypes plays a crucial function in clinical practice [1]. Phenotype Ontology (HPO) applies precision medicine into clinical practice in deep phenotyping, which is an in-depth analysis of phenotypic abnormalities with precision where phenotype components are investigated and interpreted [2]. HPO is a resource to systematically define and logically organize human phenotypes [3]. It is obtained from medical literature and knowledge resources such as the database of chromosomal imbalance and human phenotypes using DECIPHER [4], OMIM [5], and Orphanet [6]. The majority of research denotes relevant phenotypes mainly based on topological and ancestral relationships between any two nodes in the directed acyclic graphs. However, they fail to use feature learning through embeddings of different nodes to detect associations which is difficult to infer through graph structure. The traditional classification methods were applied to predict the interactions and associations between HPO-gene terms. These methods introduce inconsistencies because target values are predicted without considering inherited relationships within the ontology. For instance, if we are trying to predict the link between human phenotype gene, a normal classifier will associate the HPO term “Squamous Cell Carcinoma” with a gene, but it might not associate “Abnormality of the Skin,” hence
leading to an inaccurate prediction. To appropriately handle the hierarchical relationships between HPO terms that accurately characterize HPO, we use node embeddings. The node embeddings supply answers to transform graph nodes to distributed representations, allowing them to convert the relationship in a graph to an embedding domain. Node2Vec is a common method to create node embeddings [7]. It performs a flexible neighborhood sampling strategy using biased random walk and passes this sampling data to the word2vec model as input [8]. In this research, a framework was designed to estimate the interaction between human phenotype and gene. The dataset used for this study were taken from heterogeneous knowledge resources, specifically the Orphanet. We have converted this knowledge resource data to an undirected graph and are using this graph to create node embeddings to learn the features. These node features were used to build downstream machine learning models to predict interactions. We conducted a quantitative analysis on the output of these models using different metrics [9–16].

2. Related Work

Several studies have used the Human Phenotype Ontology graph structure to understand the association and interaction between different phenotype ontologies, genes, and proteins in the clinical service domain. The HPO2GO [17] study shows ways to predict associations between human phenotype terms in conjunction with genes [18]. OntoFUNC [19] integrated pharmacogenomics databases to identify associations between chemical pathways using analysis on chemical ontology. The HPOAnnotator [20] infers large-scale protein-human phenotype associations using PPI (protein-protein interaction) information and the HPO Graph. The low-rank approximation was employed to solve the sparsity problem in finding associations. However, none of these methods uses node embeddings and advanced machine learning models to extract appropriate node features for predicting human phenotype-gene associations [9, 21–26].

3. Methodology

3.1. Data Preparation. Link prediction intends to find the missing links or identify future link interactions between nodes based on the currently observed partial network. Predicting links between nodes has been the most significant topic in the field of graphs. To use machine learning for predicting the interaction between the unconnected pairs in the graph, we need to represent the graph with features. In Figure 1, (a) graph has seven nodes and unconnected node pairs AF, BD, BE, BG, and EG. Let us suppose we analyze the graph data at time $t$ and find new connections that have been formed in the graph at time $t + n$ (red links) in Figure 2. We extract unconnected node pairs A-F, B-D, B-E, B-G, and E-G, and then the graph is updated at time $t + n$ and these three new links are labeled (red lines in Figure 2) in the graph for the node pairs A-F, B-D, and B-E as 1, and the node-pairs BG and EG will be labeled as 0 because still at time $t + n$ no links were found for these nodes.

In this example, when we access the graph at time $t + n$, we can get the labels for the target variable. However, in real-world networks or graphs, we would have access to a single large graph. Therefore, we first need to understand that connections between nodes in the graph are developed gradually over time. Hence, we randomly hide some edges in the given graph and then creating labels would solve the problem. When we remove links or edges, we always avoid deleting edges which might lead to an isolated node, i.e., a node without any edge or an isolated network.

A significant part of the undirected graph obtained from the Orphanet HPO annotation dataset belongs to the negative population or unconnected node pairs. To find these node pairs, we create an adjacency matrix, as shown in Figure 3. This adjacency matrix is a square matrix where both columns and rows are defined by nodes of the graph. The values in the matrix denote edges or links. The value of one means an edge exists, and a value of zero means no edge was found between that node pair. In Figure 3, nodes HP: 0000951 and ORPHA:763 have a value of 0 at a cross junction in the adjacency matrix, and there is also no edge or link between them in the graph.

In this symmetric matrix, we only consider elements either above or below the diagonal. The traversal procedure finds the positions of the negative samples. A few edges are randomly dropped to obtain the positive samples. However, too much dropping might lose the connected fragments and nodes. To overcome it, we first check for the splitting of the graph when dropping a node pair. If these conditions are met, that node pair can be safely dropped. By following these steps, we found a total of 1665146 unconnected node pairs, out of which only 125304 were positive samples, i.e., around 7.5% (highly imbalanced dataset). The heterogeneous knowledge resource, Orphanet, contains 133733 associations between human phenotype and genes. The statistics for the undirected graph generated from this data can be found in Figure 4.

3.2. Feature Extraction. To extract features from graph $G$, we use the node2vec algorithm. It creates vector space embeddings to represent node features. Node2Vec starts with the weighted random walks from every node of the graph and interprets random walks as terms that are embedded by a skip-gram model into Euclidean space [21]. The aim of this approach is to maximize the node $n$’s probability in context $K$ within the contextual window of length $l$: \[
L(n) = \sum_{k \in K} \sum_{i=1}^{l} \sum_{j=1}^{l} \log p(k_i | k_j).
\] (1)

Here, $k_i$ denotes the $i^{th}$ term or node sequence generated by using a random walk. The output from the skip-gram method, which is being defined by the SoftMax function, is \( p(k_j | k_i) \), which is as follows:

\[
p(k_j | k_i) = \frac{\exp - (n_i n_j)}{\sum_{-l \leq j \leq l} \exp - (n_i n_j)}
\] (2)
where $\mathbf{n}_j, \mathbf{n}_i$ are vector representations of terms $k_j, k_i$ in the hidden layer of the skip-gram model. The node2vec library was used to train a model with 30 nodes in each walk and 200 walks per node with 128 embedding dimensions. We will apply this node2vec model to every node pair in the dataset.

### 3.3. Downstream Prediction Using ML Algorithms

The node features from the node2vec are fed to machine learning models. To verify their performance, we split the complete dataset into training (80%) and testing (20%). We used the following methods for this supervised link prediction work.

#### 3.3.1. Logistic Regression

Logistic regression classifies binary outcomes borrowed from the field of statistics. It uses the sigmoid activation function to restrict the outcome between 0 and 1. The coefficients are estimated from our training set using a maximum-likelihood learning algorithm. It is a common algorithm that makes assumptions about the distribution of data. For this study, we used the L-BFGS solver [22].

#### 3.3.2. Random Forest

Random forest is a widely used bagging method for supervised machine learning tasks. The decision trees are ensembled together to generate forest
training using bagging methods. The bagging here refers to building multiple decision trees together and later merging them to get a more accurate prediction. This algorithm can be used for both classifications as well as a regression task. In our case, we will be training the random forest model for binary classification to predict the link between two nodes. The maximum depth used to train this model was 12 [23–29].

3.3.3. Neural Network. Neural networks have revolutionized the field of machine learning with recent advancements. This algorithm tries to simulate the human brain to find patterns in information. Neural networks can be used for a wide variety of tasks, including similar clustering of data, classifying different objects, and many more. The neural network is initiated with random weights and thresholds. The training data are being fed as vectors to the input layer and letting it pass through the succeeding hidden layers by complexly adjusting these weights and thresholds until we yield similar outputs as true labels. For this study, we are using a neural network with two hidden layers, each with the ReLU activation, and using sigmoid activation at the output layer to get the output values between 0 and 1. The Adam optimizer is used with a learning rate value of 1e-3, which is an extension to SGD (stochastic gradient descent), which helps us to update the network weight based on training data.

3.3.4. XGBoost. XGBoost (eXtreme gradient boosting) is a go-to algorithm in the machine learning community to solve most of the classification and regression problems. It uses a boosting technique by training models in isolation from each other by trying to correct mistakes from the previous models. These models are added up sequentially until we reach the point where no further mistakes can be made. Gradient boosting is a method that trains on the errors or residuals of previous models. This method is computationally efficient and fast as compared to other boosting methods. For this study, we have used a learning rate of 0.1, a maximum depth of 12, and a scale weight of 0.99 to handle the imbalanced nature of the data [24].

3.3.5. LightGBM. LightGBM is one of the gradient boosting methods that is very fast and computationally efficient. This framework is used for many downstream prediction tasks such as classification, regression, and ranking. It uses the same concepts as XGBoost except for one key difference: it splits trees on the leaf and not on the level. This helps to reduce more losses when growing on the same leaf. We use the maximum depth parameter value of 10 to avoid building a more complex model. The scale pos weight 99 was used to deal with the class imbalance problem for this study.

4. Results and Discussion

4.1. Evaluation Metrics. Due to the high imbalance, the model evaluation metrics we choose are important. In this study, we try to understand how each class is performing rather than focusing on overall metrics. Below are the metrics which we have used to validate the performance of these machine learning methods.

4.1.1. AUROC. AUROC stands for the area under the receiver operating characteristics (ROC) curve. In a binary classification task, a very common summary statistic to calculate the goodness of a predictor is defined by AUROC. In AUROC, the receiver operating characteristics is the probability curve, and the area under the curve is the degree of separability. It provides information about the class distinguishability in the model. This ROC curve is drawn with the true positive rate (sensitivity) against the false positive rate (1-specificity).
4.1.2. AUCPR. AUCPR stands for the area under the precision-recall curve which is the most widely used metric in the machine learning community for validating highly imbalanced data. This is used for evaluating binary classification model performance by plotting precision against recall. These metrics help us to see the performance of positive samples more closely as compared to AUROC.

\[
\text{sensitivity (TPR)} = \frac{\text{true positives}}{\text{true positive} + \text{false negatives}}; \quad \text{(3)}
\]

\[
\text{false positive rate (FPR)} = \frac{\text{false positives}}{\text{false positive} + \text{true negatives}}.
\]

4.1.3. Microaverage Precision/Recall/F1 Score. The microaverage metric helps us weigh each class instance equally. PrecisionMicroAvg can be defined as the sum of all true positives for all classes by all positive predictions.

\[
\text{Precision Micro Avg} = \frac{(TP1 + TP2 + \cdots + TPn)}{(TP1 + TP2 + \cdots + TPn + FP1 + FP2 + \cdots + FPn)} \quad \text{(5)}
\]

RecallMicroAvg can be defined as the sum of all true positives for all classes by all actual positives, i.e., true positives not the predicted positives.

\[
\text{Recall Micro Avg} = \frac{(TP1 + TP2 + \cdots + TPn)}{(TP1 + TP2 + \cdots + TPn + FN1 + FN2 + \cdots + FNn)} \quad \text{(6)}
\]

F1MicroAvg can be defined by calculating the harmonic mean for PrecisionMicroAvg and RecallMicroAvg.

\[
F1\text{Micro Avg} = 2 \times \frac{\text{Precision Micro Avg} \times \text{Recall Micro Avg}}{\text{Precision Micro Avg} + \text{Recall Micro Avg}} \quad \text{(7)}
\]

4.1.4. Macroaverage Precision/Recall/F1 Score. The macroaverage is used when we want to treat all classes equally when evaluating performance [25] concerning the most frequent class in the dataset.

PrecisionMacroAvg can be defined as the arithmetic mean of precision from all the classes. It can be mathematically defined as

\[
\text{Precision Macro Avg} = \frac{\text{Precision}_1 + \text{Precision}_2 + \cdots + \text{Precision}_n}{n} \quad \text{(8)}
\]

RecallMacroAvg can be defined as the arithmetic mean of recall from all the classes. It can be mathematically defined as
4.1.5. Weighted Average Precision/Recall/F1 Score. Precision Weighted Avg can be calculated by multiplying class weights based on true labels with its corresponding precision score.

\[
\text{Precision Weighted Avg} = \frac{\text{Precision}_1 \times W_1 + \text{Precision}_2 \times W_1 + \cdots + \text{Precision}_n \times W_n}{n}
\]

where \( W_i \) is class weight based on true labels.

Similarly, Recall Weighted Avg can be calculated by multiplying class weights based on true labels with its corresponding recall score.

\[
\text{Recall Weighted Avg} = \frac{(\text{Recall}_1 \times W_1 + \text{Recall}_2 \times W_1 + \cdots + \text{Recall}_n \times W_n)}{n}
\]

\[
F1Weighted Avg = \frac{1}{N} \sum_{i=0}^{N} F1_{\text{class } i} \times W_{\text{class } i}
\]

where \( W \) is class weight based on true labels.
4.2. Comparing Performances of Different Models. To evaluate the predictive performance of these five models, logistic regression, random forest, neural network, XGBoost, and LightGBM on the human phenotype-gene dataset, we use all the many different metrics, including AUROC, AUCPR, micro, macro, and weighted average precision, recall, and F1 score. To appropriately evaluate the imbalanced nature of the dataset, we calculate important metrics for each class instance. In our case, we have two classes, 0 and 1.

As shown in Table 1, it is much easier to identify which model is doing a great job in identifying each class. Based on these metrics, we can see that XGBoost and random forest perform well in identifying positive samples which are positive, i.e., when they predict a link between nodes, they are correct 99% and 100% of the time, respectively. Contrastingly, LightGBM beats all other methods in identifying correct actual positives—in other words, it correctly predicts 82% of all the links between these nodes. From Table 2, we can confer that LightGBM is better than all other models in terms of AUROC and AUCPR.

Figure 4 shows the graph statistics and degree distribution. Figure 5 denotes the AUROC and AUCPR for each classifier.

5. Conclusion

In this research, an approach is proposed to predict links between human phenotype and genes using heterogeneous knowledge resources, i.e., Orphanet. The most important part of this study is to represent data into a graph and then to find a way to represent this graph into an appropriate feature set which will allow us to use it for downstream tasks like prediction. In essence, we provided a way to get the embedding vectors by using an algorithm called node2vec and then using these embeddings to build five different machine learning models. We evaluated and compared the performances using different quantitative metrics, including AUROC, AUCPR, micro, macro, and weighted precision, recall, and F1 score. Some of these metrics were calculated for each class instance to better understand the situation for imbalanced class, in our case, positive samples. Based on these metrics, we found very interesting results. Suppose we want to just focus on positive samples meaning the measure of the link that we correctly identify having associations of all the actual associations in the graph (we refer to it as precision). In that case, we may either use XGBoost or random forest algorithm. On the other hand, if we just want to focus on accurately identifying positives from true positives, i.e., actual links in the graph (recall), then use LightGBM.

Data Availability

The data supporting this research are available from the corresponding author on reasonable request.

Disclosure

A preprint has been published previously [21] (http://export.arxiv.org/list/cs/new?show=1000&skip=0).

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] P Owen, “Clinical practice and medical research: bridging the divide between the two cultures,” British Journal of General
mathematical problems in engineering


