Integrative Network Analysis: Unveiling Symptom-Disease Interactions and Enhancing Predictive Models

Andreoli C. • Ligari D. • Alberti A. • Scardovi M. [1]

[1] *Department of Computer Engineering, Data ScienSickness predictionce, University of Pavia, Italy*
Course of Financial Data Science

Github repository: https://github.com/AndreaAlberti07/enhancing-disease-prediction

Abstract

We will write it once we have the results.

Keywords— *Graph theory • Features Engineering • Community detection • Null models • Random forest • MLP*

Contents

1 Introduction

In the dynamic landscape of healthcare, understanding the intricate interplay between symptoms and diseases is paramount for effective diagnosis and prediction. This report embarks on a comprehensive journey through the realms of network analysis, leveraging both theoretical foundations and empirical data to unravel the complexities of symptom-disease interactions. Our dual-fold objective is to provide a nuanced descriptive analysis of these interactions while identifying key features to bolster predictive models.

The foundation of this endeavor lies in an extensive review of existing literature, drawing insights from seminal works on network theory and disease prediction. By establishing a baseline through prior research, we pave the way for a deeper understanding of the subject matter and ensure the relevance of our findings in the broader context of scientific inquiry.

Guided by insights gleaned from the literature, our exploration extends to the realm of data, where we meticulously curate and analyze datasets of varying sizes. Through a systematic process of exploratory data analysis and cleaning, we prepare the groundwork for constructing meaningful networks that encapsulate the relationships between

symptoms and diseases.
The heart of our analysis lies in the creation of intricate network structures, employing bipartite models and non-weighted links to distill meaningful patterns. We delve into a spectrum of network metrics, from fundamental measures like degree distribution and clustering coefficients to more nuanced assessments of node importance and betweenness centrality. Statistical significance is rigorously assessed through the lens of a null model, ensuring that our observations transcend mere chance.
Community detection algorithms further dissect the network, revealing hidden structures and relationships between diseases. This not only enriches our understanding but also lays the groundwork for subsequent analyses. As we traverse the terrain of network analysis, we introduce novel metrics inspired by the Hidalgo-Hausmann framework, stratifying symptoms and diseases based on their predictive importance. These metrics, coupled with traditional measures like betweenness centrality, contribute to the definition of features that fuel our predictive models.
With a robust foundation established, we transition to the realm of predictive modeling, where our feature-rich approach promises to enhance the performance of established models. Logistic regression, random forest, and multi-layer perceptron models are trained, tested, and validated, with a keen eye on feature importance and model improvement strategies.
This report unfolds as a holistic exploration, weaving together theoretical frameworks, empirical analyses, and predictive modeling into a cohesive narrative. As we traverse the intricate web of symptom-disease interactions, our aim is not only to elucidate the underlying dynamics but also to pave the way for more accurate and insightful predictive models in the realm of healthcare.

2 Dataset

The dataset used for this project is obtained from Kaggle and is available at the following link. It comprises disease names along with the symptoms reported by the respective patients.

Overview: The dataset encompasses 773 unique diseases and 377 symptoms, resulting in approximately 246,000 rows. It was artificially generated while preserving Symptom Severity and Disease Occurrence Possibility.

Data Encoding: To facilitate model training, the dataset utilizes one-hot encoding for each symptom, transforming categorical symptom data into a binary format.

Class Imbalance: The original dataset exhibited significant class imbalance, with some classes having only one sample and others containing thousands. We addressed this problem using Oversampling and Undersampling techniques, and the details are further elaborated in Section **??**.

Data Cleaning: To allow consistent Oversampling, classes (diseases) with fewer than three symptoms were excluded from the dataset, resulting in the removal of 25 classes. Additionally, diseases with no symptoms and symptoms with no associated diseases were deleted as well.

3 Goals

The primary objective of this project is to develop a robust **machine learning model** capable of predicting diseases based on reported symptoms. In pursuit of this goal, two specific objectives are outlined, both centered around **leveraging the network** structure:

1. **Feature Extraction:** Exploiting the network characteristics and metrics, aiming to extract novel features that can enhance the predictive capabilities of the model.
2. **Computational Efficiency Improvement:** Leveraging network information, we aim to reduce the number of symptoms, retaining only the most relevant ones. This strategic reduction aims to decrease training time while preserving the accuracy of the model.

By integrating these two objectives, the project aspires to not only advance the predictive capabilities of the machine learning model but also optimize its efficiency in handling the complexities inherent in disease prediction based on symptoms.

4 Network Methodology

In this section we technically describe the methodology used to create the network and to compute metrics on it.

4.1 Network Creation (Not Weighted - Bipartite)

4.2 L1 and L2 measures

4.3 Betweenness Centrality

The betweenness centrality of a node v, as defined by Brandes [2], is calculated as the sum of the fraction of all-pairs shortest paths that pass through v:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{1}$$

where:

- V: The set of nodes.
- $\sigma(s,t)$: The number of shortest paths from node s to node t.
- $\sigma(s,t|v)$: The number of those shortest paths from node s to node t that pass through some node v other than s and t.
- If $s = t$, then $\sigma(s,t) = 1$.
- If $v \in \{s,t\}$, then $\sigma(s,t|v) = 0$.

To compute the betweenness centrality, the NetworkX function *nx.bipartite.betweenness_centrality* was utilized. This function implements the algorithm proposed by Brandes [1], specifically designed for bipartite graphs, and includes proper normalization for accurate results.

4.4 Communities Detection

Prior to applying any community detection algorithm, two crucial steps must be performed:

- **Graph Projections:** The bipartite graph needs to be projected into two separate graphs, one for each set of nodes. In our case, the two sets represent symptoms and diseases. To achieve this, the NetworkX function *nx.bipartite.projected_graph* is employed, returning the projection of the bipartite graph onto the specified nodes.

- **Compute Similarity:** The similarity between nodes needs to be computed. For our purposes, a co-occurrence matrix is created for each set of nodes. Taking the example of the co-occurrence matrix for symptoms, each entry s_{ij} represents the number of times the symptom i and the symptom j co-occur in the same disease.

Once the two graphs, with links weighted by node similarity, are obtained, the community detection algorithm can be applied. We utilized the Clauset-Newman-Moore greedy modularity maximization algorithm [3], implemented in the NetworkX function *nx.algorithms.community.greedy_modularity_communities*. This algorithm aims to find the partition of the graph that maximizes modularity, defined by Newman [4] as:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{2}$$

where:

- Q: Modularity of the network.
- A_{ij}: Element of the adjacency matrix representing the connection between nodes i and j.
- k_i and k_j: Degrees of nodes i and j, respectively.
- m: Total number of edges in the network.
- $\delta(c_i, c_j)$: Kronecker delta function, which is 1 if c_i is equal to c_j (i.e., nodes i and j belong to the same community) and 0 otherwise.
- The sum is taken over all pairs of nodes i and j.

5 Network Results

5.1 L1 and L2 Metrics

5.2 Betweenness Centrality

The examination of betweenness centrality in our bipartite network, as depicted in Figure 1, reveals a Power Law Distribution, indicative of a scale-free structure. This implies the presence of a few central nodes that act as pivotal connectors, while the majority of nodes exhibit lower betweenness centrality.

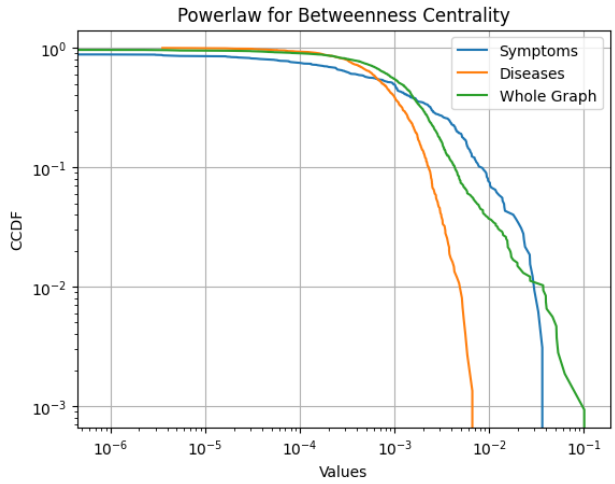

Figure 1. Betweenness Centrality CDFs

Upon dissecting the centrality values into symptoms and diseases (see Figures 2 and 3), a notable observation emerges: symptoms tend to have higher betweenness centrality compared to diseases. To decipher the significance of this result, it's essential to delve into the interpretation of betweenness centrality.

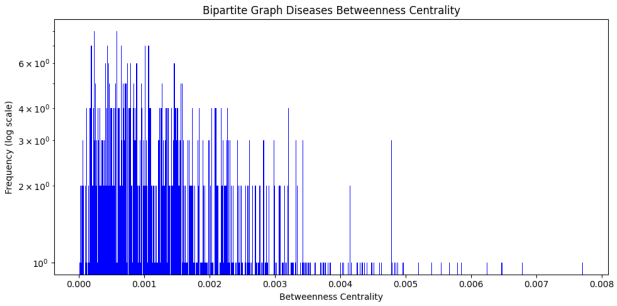

Figure 2. Betweenness Centrality of the diseases

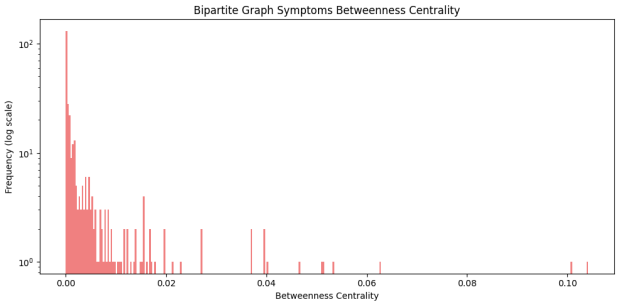

Figure 3. Betweenness Centrality of the symptoms

In general, a symptom exhibits high betweenness centrality when it is linked to numerous diseases, and these diseases, in turn, are connected to a relatively limited set of symptoms. Conversely, a disease attains high betweenness centrality when it connects to numerous symptoms, and these symptoms are associated with relatively few diseases.

Analyzing our results (L1 and L2), it becomes evident that the higher betweenness centrality of symptoms is attributed to their connections with a multitude of diseases, while diseases, on the contrary, are linked to a relatively limited number of symptoms. From a predictive standpoint, this outcome presents a challenge as each symptom is not sufficiently specific, contributing to a broad array of disease classes.

Figure 4 highlights the top 10 nodes with the highest betweenness centrality, all of which are symptoms. As anticipated, these symptoms are more generic in nature, aligning with their central role in connecting various diseases.

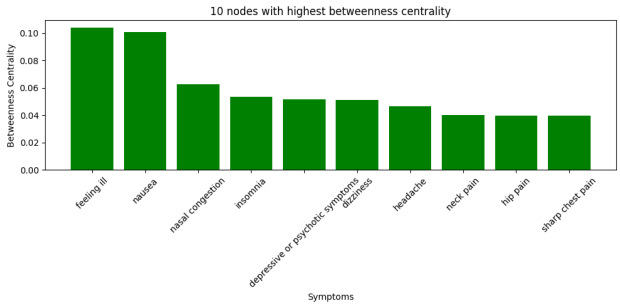

Figure 4. Top 10 nodes with the highest betweenness centrality

5.3 Communities

The identification of communities within the network serves a dual purpose – facilitating network interpretation and enhancing the capabilities of our ML prediction model.

From a network interpretation perspective, communities offer insights into disease-symptom relationships. A community of symptoms signifies a set of symptoms that frequently co-occur within the same diseases, while a community of diseases identifies a set of diseases often co-occurring within the same symptoms. The sizes of different communities are illustrated in Figure 5.

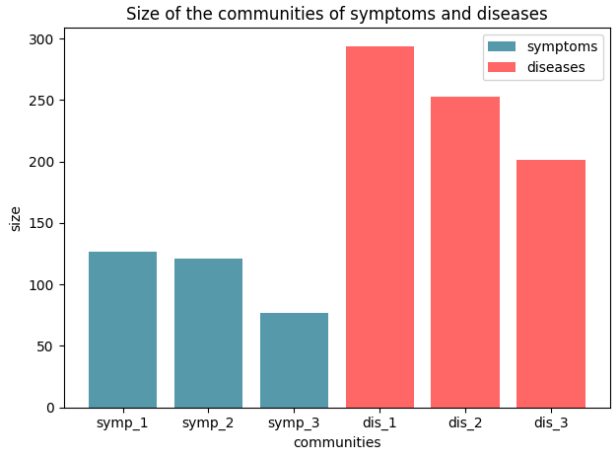

Figure 5. Sizes of the communities of symptoms and diseases

For clinical relevance, examining symptoms communities provides valuable information about diseases associated with these symptoms. This is exemplified in Figures 6, 10, and 11. As an illustration, in the community 1 of symptoms (Figure 6), 'herniated disk' is the third most pointed disease by the symptoms of the community, with each symptom pointing, on average, to three diseases.

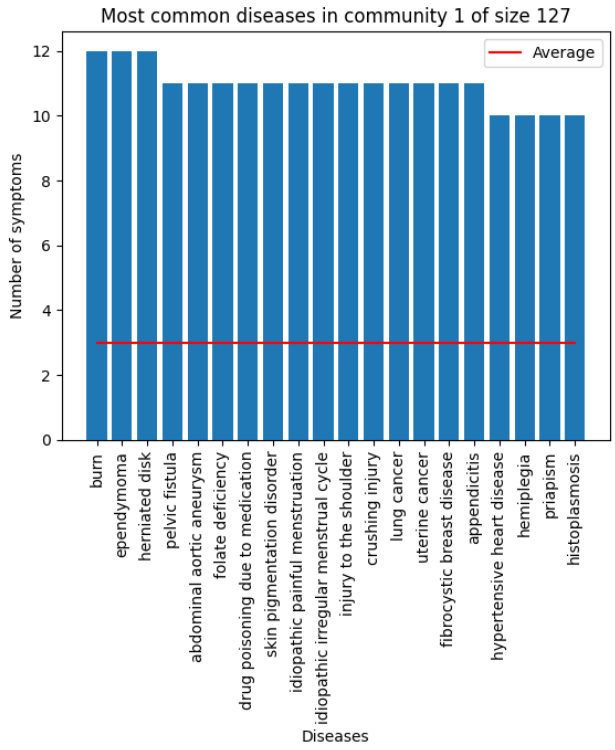

Figure 6. Community 1 of symptoms

A similar study can be conducted for communities of diseases, as depicted in Figures 7, 12, and 13. This information aids in profiling diseases and understanding the significance of each symptom. For instance, in community 1 of diseases (Figure 7), the symptom 'sharp abdominal pain' is present in almost half of the diseases in the community, indicating its generic nature and limited discriminatory value.

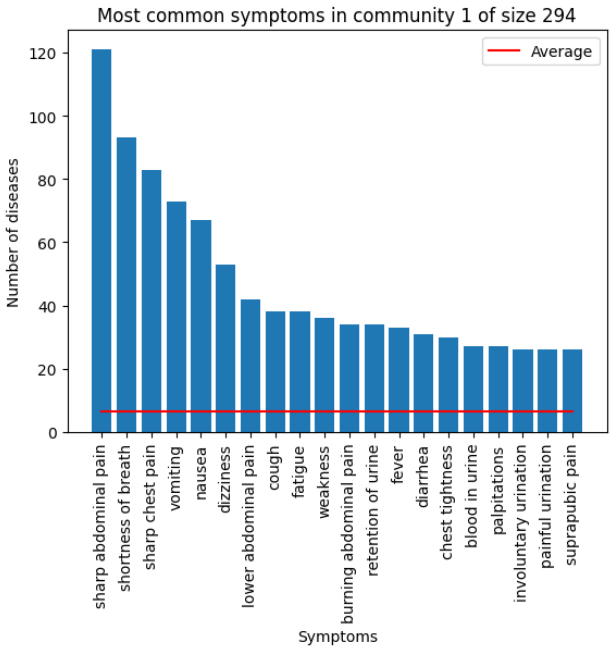

Figure 7. Community 1 of diseases

Transitioning to the creation of features for the ML model, two types of features were developed:

- **Community Count:** This feature counts how many symptoms of the symptom vector belong to each community. Each symptom community is characterized by different pointed diseases. The model can learn to prioritize diseases associated with the community with the highest count.
- **Community Size:** This feature replaces each symptom in the symptom vector with the size of the community to which the symptom belongs. It enables the model to distinguish between symptoms belonging to small and large communities, injecting community information into the model beyond basic one-hot encoding of symptoms.

It is noteworthy that communities can also contribute to improving the computational efficiency of the model. For example, a symptom associated with many diseases may be less informative and could potentially be removed from the symptom vector. However, we opted for a comprehensive approach using a combination of L1 and L2 measures to address this issue.

5.4 Most Important Actors

As previously mentioned, our objective extends beyond feature extraction; we aim to leverage network information to enhance the computational efficiency of the model. The strategy involves reducing the number of symptoms, retaining only the most significant ones, to decrease training time while maintaining high accuracy. Various approaches were tested, including L1, L2, betweenness centrality, and the degree of the unipartite projection of symptoms. We performed a graphical analysis of their correlation (Figure 8), favoring the most uncorrelated features to provide the most complementary information.

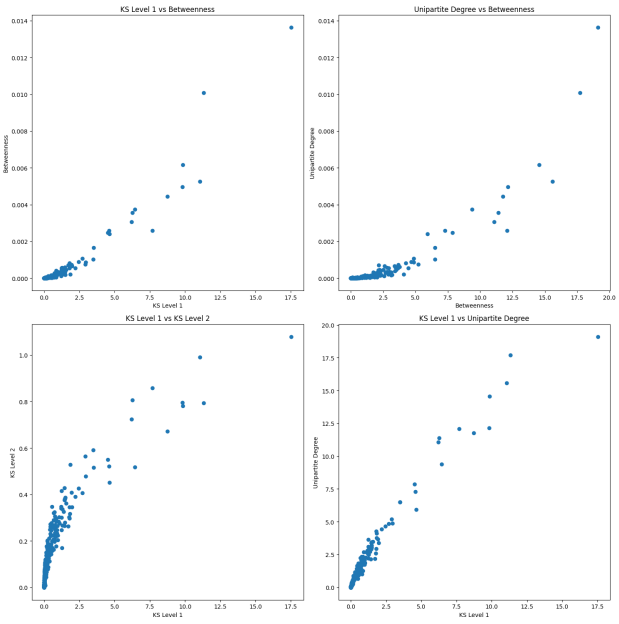

Figure 8. Correlation between features

Despite a non-negligible positive correlation in all cases, we opted for L1 and L2 due to their interpretability and the less pronounced positive correlation at lower L1 values. This allows for a substantial division of symptoms for the same value of L1 but different values of L2. Figure 9 illustrates the possibility of placing a proper threshold on L1 and L2 to obtain four classes of symptoms, each with a non-null number of symptoms.

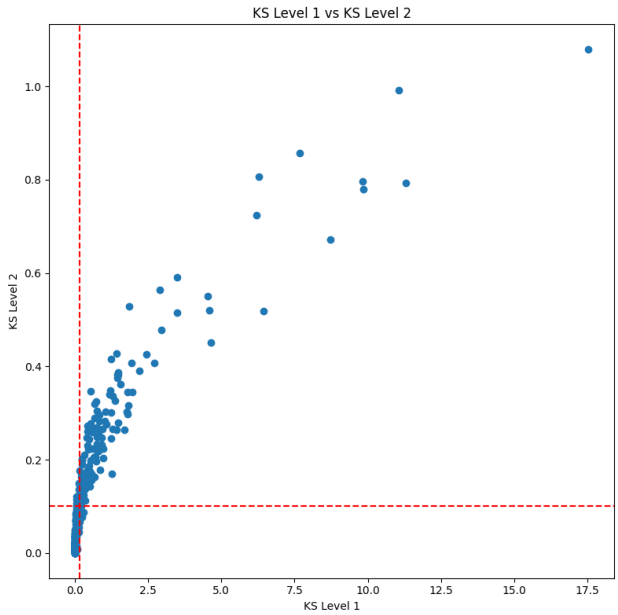

Figure 9. Division based on L1 and L2 values

To interpret these classes, we briefly recall the meanings of L1 and L2. The former represents the number of diseases associated with a symptom, while the latter measures the number of symptoms associated with those diseases. Consequently, we define the following classes:

- **High L1 - High L2**: Symptoms with high degree and high L2. These symptoms are less crucial for prediction as they contribute to many classes (diseases), which are also connected to many other symptoms.
- **High L1 - Low L2**: Symptoms with high degree and low L2. These symptoms should not be removed a priori since they can be useful. For instance, a symptom may be associated with many diseases, but those diseases may only be associated with that symptom. In this case, the symptom is very important for prediction.
- **Low L1 - High L2**: Symptoms with low degree and high L2. These symptoms may be important for prediction, as they contribute to few diseases.
- **Low L1 - Low L2**: Symptoms with low degree and low L2. These symptoms are the most important for prediction since they contribute to few classes (diseases),

and those classes are also connected to few other symptoms.

According to the above considerations, we can start from the last class and iteratively add symptoms from the other classes, monitoring the impact on both accuracy and training time. The results are analyzed in Section **??**.

It is worth saying that all the other features (betweenness centrality, community count, and community size) represent a valuable alternative to L1 and L2. Their use in computational time reduction is not explored in this work, but it is a possible future development.

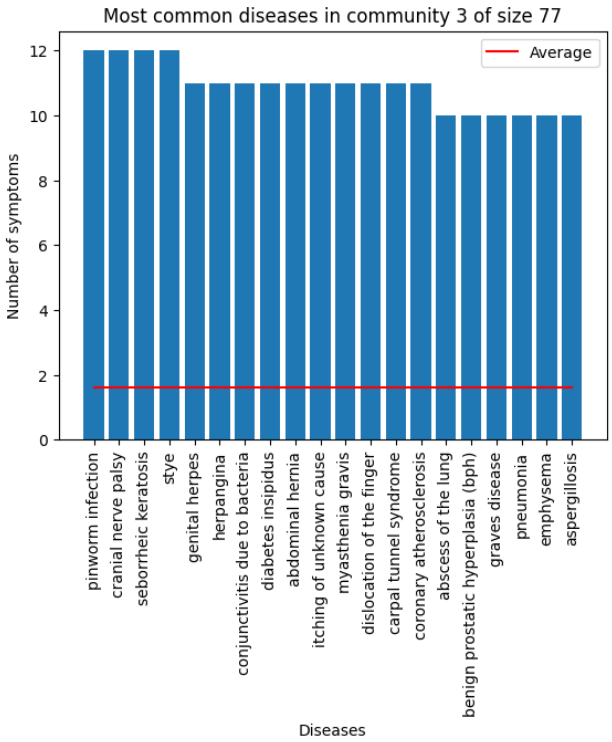

Figure 11. Community 3 of symptoms

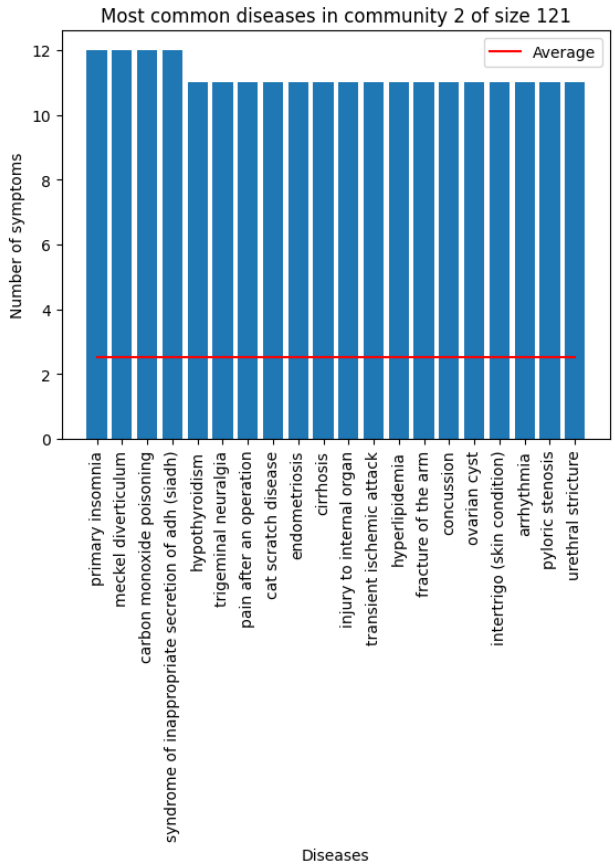

Figure 10. Community 2 of symptoms

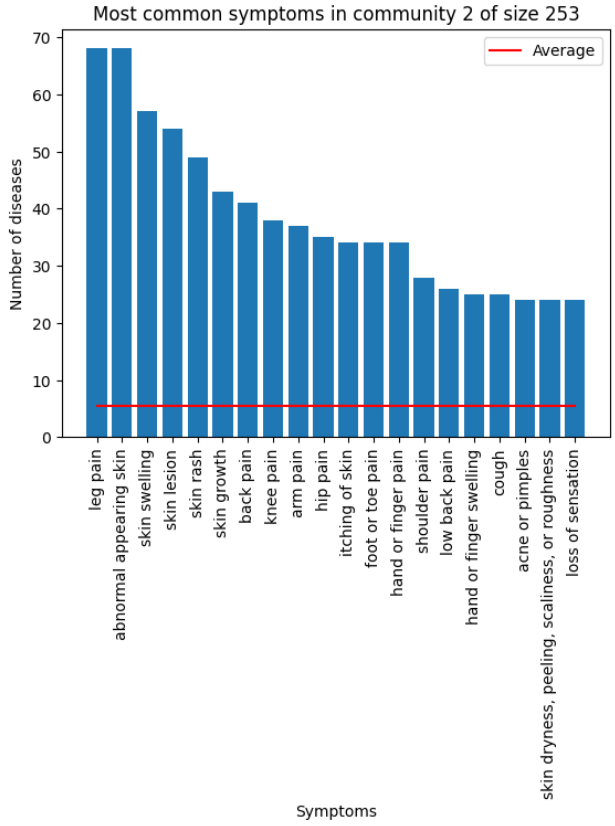

Figure 12. Community 2 of diseases

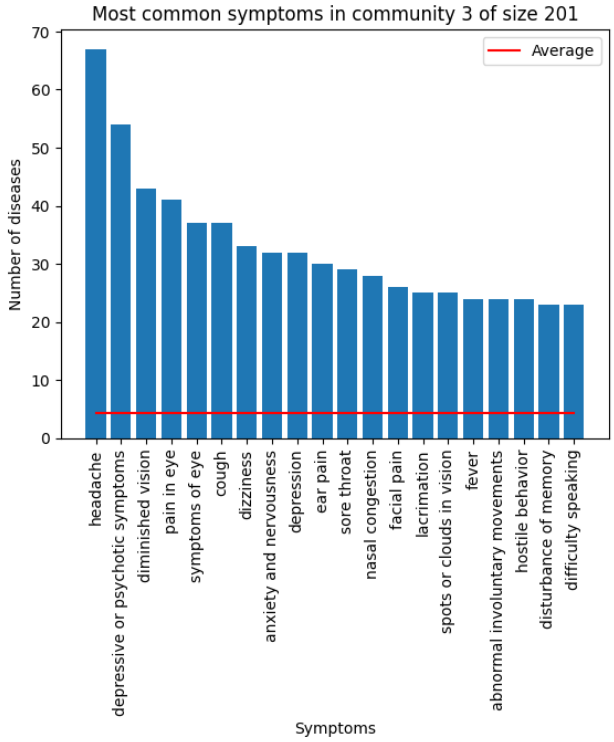

Figure 13. Community 3 of diseases

6 ML Model Methodology

This section provides a comprehensive overview of the methodologies employed in the construction of the machine learning model. The discussion encompasses various techniques designed to handle the intricacies of model building, coupled with a logical flow that guides the entire process.

6.1 Data Balancing

6.2 Feature Extraction

A pivotal phase in constructing a machine learning model is feature extraction. In addition to the one-hot vector representation of symptoms, the network analysis affords us the following features:

- **L1 and L2 Measures**: A vector with values representing the L1 and L2 measures for each symptom.
- **Betweenness Centrality**: A vector with values denoting the betweenness centrality of each symptom.
- **Community Count**: A vector indicating the number of symptoms belonging to each community.
- **Community Size**: A vector replacing symptoms with

the size of the community to which they belong.

Given the diverse scales of these features, normalization becomes imperative for their cohesive integration into the model without introducing biases. To achieve this, we opted for *MaxAbs* normalization. This normalization scales each feature individually, ensuring that the maximal absolute value of each feature in the training set becomes 1.0, while preserving the sparsity of data.

6.3 Operative Flow

Once the features are ready, the core part of the model building process can begin. We trained three different models: a Logistic Regression, a Random Forest, and a Multi Layer Perceptron (MLP). For each model we had to address two problems: the selection of the best parameters and the selection of the best features. The problem is that the two things are not independent from each other, in other words, changing the parameters leads to different features being selected and vice versa. The unique solution to this problem is to test all the parameters combinations and for each of them test all the features combinations. This approach is not feasible in terms of computational effort. For this reason we decided for a greedy approach. First of all we split the features in two groups: the symptoms one hot vector and the other features. The former is used to train a base model with random parameters, the latter is used to investigate the improvement that may be brought by the new features. For each of the two groups we get the best feature combination using Algorithm 1, then given the best combination of features we get the best combination of parameters as shown in Algorithm 2. Now we train each model with the best parameters and the best features combination and we pick the best model according to the accuracy. Finally the best model overall is trained with the whole dataset.

Algorithm 1 Feature Selection Algorithm

```
BestFeatureComb ← EmptySet
for each model do
    CurrentModel ← EmptyModel
    BestModAccuracy ← 0
    Parameters ← InitializeRandomParameters
    for i = 1 to NumberOfFeatures do
        BestAccuracy ← -1
        for each feature do
            TrainModel(CurrentModel, Parameters)
            CurrentAccuracy ← GetAccuracy(CurrentModel)
            if CurrentAccuracy >= BestAccuracy then
                BestAccuracy ← CurrentAccuracy
                BestFeature ← feature
            end if
            UpdateModel(CurrentModel, BestFeature)
        end for
        FreezeFeatures(CurrentModel)
        ModelAccuracy[i] ← GetAccuracy(CurrentModel)
        FeaturesComb[i] ← GetFeat(CurrentModel)
    end for
    BestComb ← FeaturesComb[Max(ModelAccuracy)]
    BestFeatureComb ← BestComb ∪ BestFeatureComb
end for
return BestFeatureComb
```

Algorithm 2 Best Model Selection Algorithm

```
CurrentAccuracy ← 0
for each model do
    CreateGrid(Parameters)
    BestParameters ← GridSearchCV(model)
    TrainModel(model, BestParameters)
    CurrentAccuracy ← GetAccuracy(model)
    if CurrentAccuracy >= BestAccuracy then
        BestAccuracy ← CurrentAccuracy
        BestModel ← model
        BestParameters ← Parameters
    end if
end for
FullDataTrain(BestModel, BestParameters)
BestAccuracy ← GetAccuracy(BestModel)
ReduceFeatures(BestModel, BestParameters)
return BestParameters, BestModel, BestAccuracy
```

At the end of these two algorithm we have the following two models:

- **Symptoms Model**: Best model with the best parameters and the symptoms as features

- **Other Features Model**: Best model with the best parameters and the best features combination

At this point we can compare the two models in term of prediction performance to see whether the network features are capable of improving the accuracy of the model. As regard the second goal of the project, we can apply the feature reduction technique discussed in Section **??** to both models, to see whether the network features can reduce the computational complexity of the model without affecting the accuracy.

7 ML Model Results

8 Conclusion

9 Future Works

References

[1] Ulrik Brandes. “A Faster Algorithm for Betweenness Centrality”. In: *The Journal of Mathematical Sociology* 25 (Mar. 2004). DOI: 10.1080/0022250X.2001.9990249.

[2] Ulrik Brandes. “On variants of shortest-path betweenness centrality and their generic computation”. In: *Social Networks* 30.2 (May 2008), pp. 136–145. ISSN: 0378-8733. DOI: 10.1016/j.socnet.2007.11.001.

[3] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. “Finding community structure in very large networks”. In: *Physical Review E* 70.6 (Dec. 2004). arXiv:cond-mat/0408187, p. 066111. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.70.066111.

[4] M. E. J. Newman. “Modularity and community structure in networks”. In: *Proceedings of the National Academy of Sciences of the United States of America* 103.23 (June 2006), pp. 8577–8582. ISSN: 0027-8424. DOI: 10.1073/pnas.0601602103.