Disease Prediction

A network theory + ML approach

University of Pavia
Financial Data Science course

Andrea Alberti
Davide Ligari
Cristian Andreoli
Matteo Scardovi

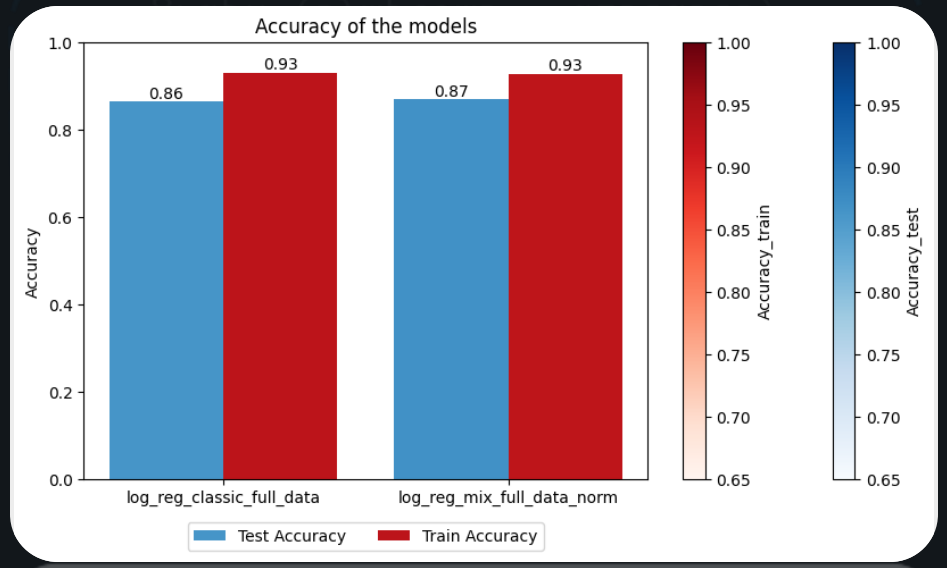
Dataset

- **Kaggle** dataset
- 773 unique diseases
- 377 unique symptoms
- 246.945 samples
- **Artificially** generated

Our Objectives

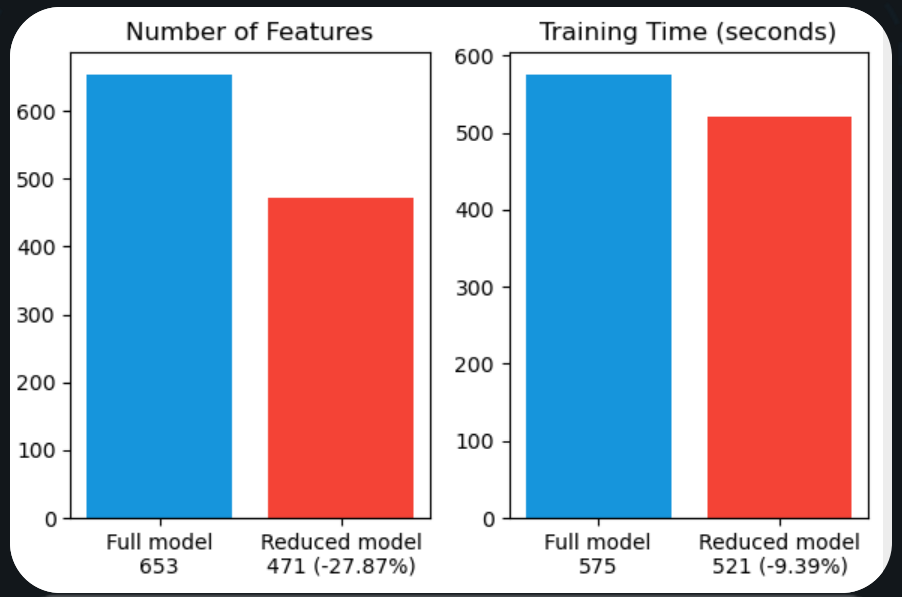
GOAL 1

Evaluate the effectiveness on diseases prediction models of new features extracted from a bipartite graph (symptoms – diseases)

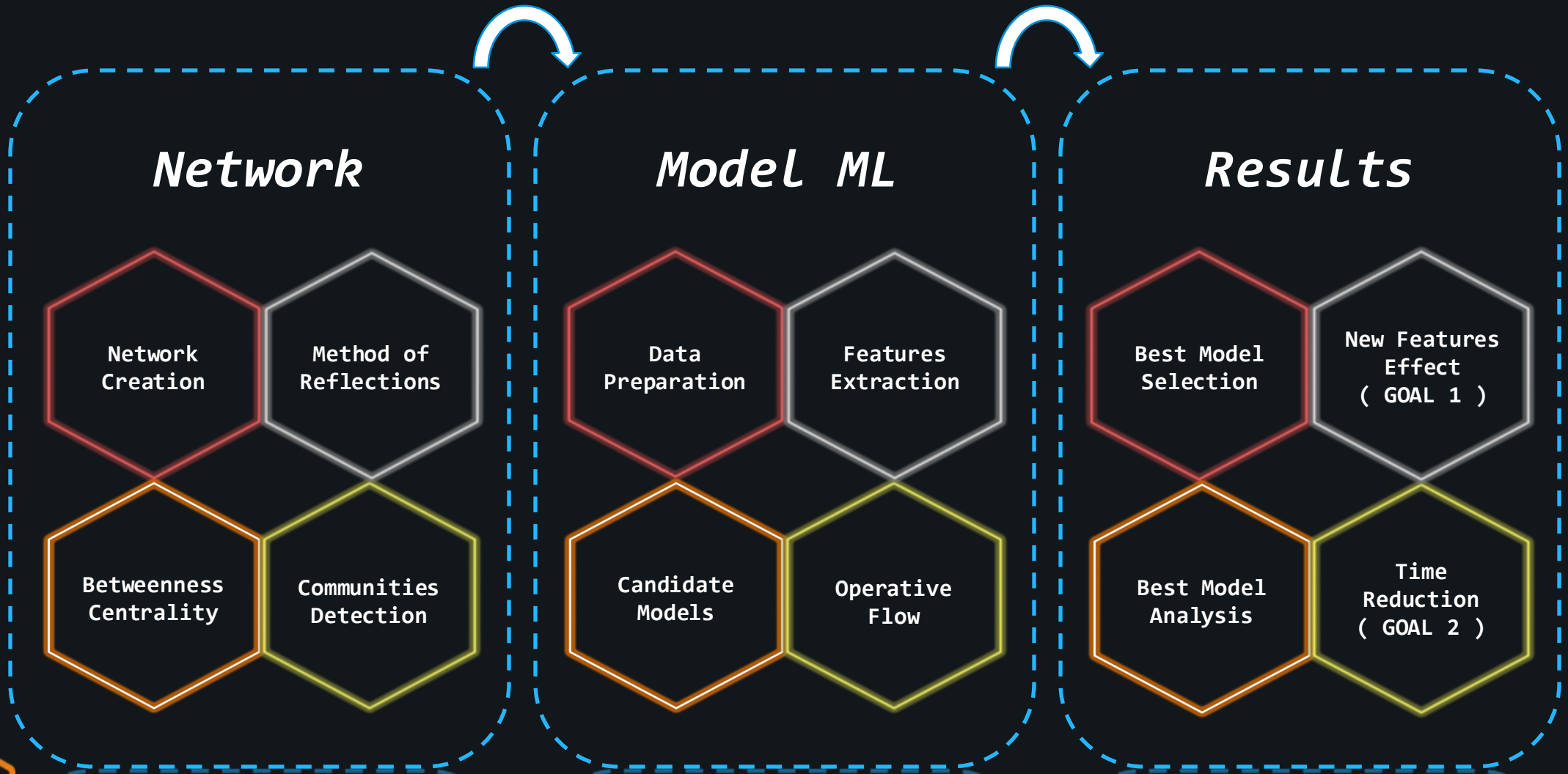


GOAL 2

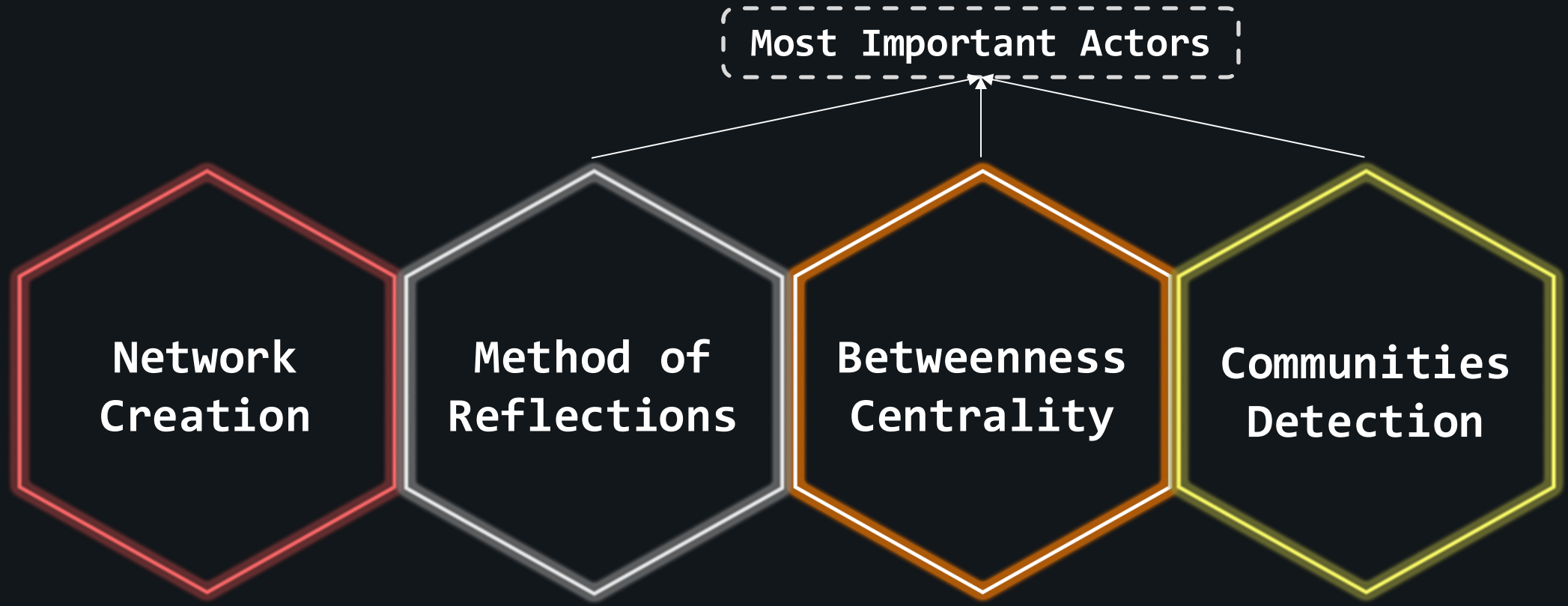
Evaluate the effectiveness of graph-based solutions in improving the prediction models computational efficiency



Summary

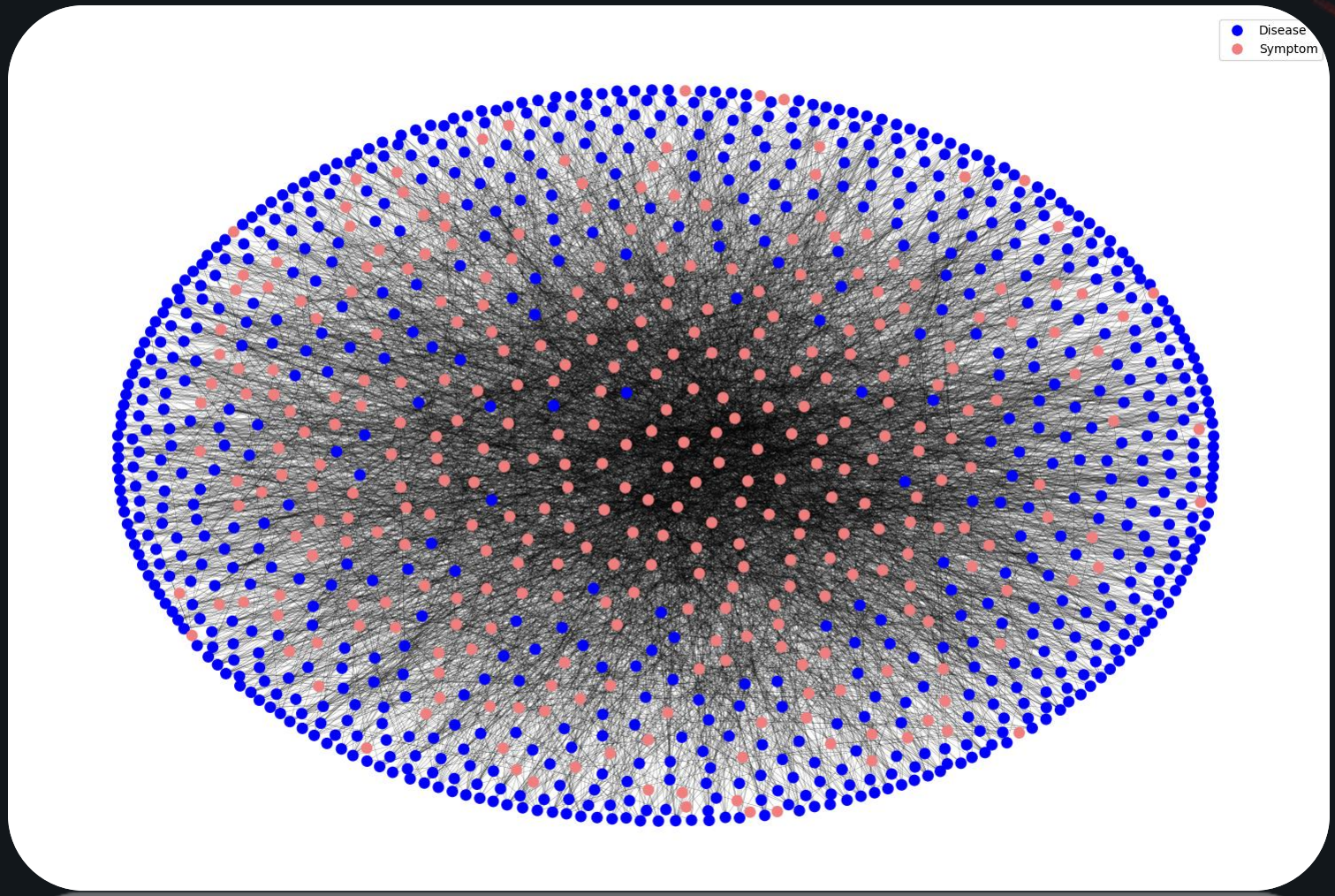

Network
Network Creation
Method of Reflections
Betweenness Centrality
Communities Detection
Model ML
Data Preparation
Features Extraction
Candidate Models
Operative Flow
Results
Best Model Selection
New Features Effect (GOAL 1)
Best Model Analysis
Time Reduction (GOAL 2)

Network Methodology and Results



Network Creation

- Bipartite network
- unweighted
- Removed isolated nodes (52 symptoms)



Method of reflection

2 Indexes

- SI index: related to symptom nodes

$$\mathrm{SI}_{\mathrm{v,N}} = \frac{1}{\mathrm{SI}_{\mathrm{v,1}}} \sum_{\mathrm{u}} W(v,u)\, DI_{u,N-1}$$

- DI index: related to disease nodes

$$DI_{u,N} = \frac{1}{DI_{u,1}} \sum_{v} W(v,u) SI_{v,N-1}$$

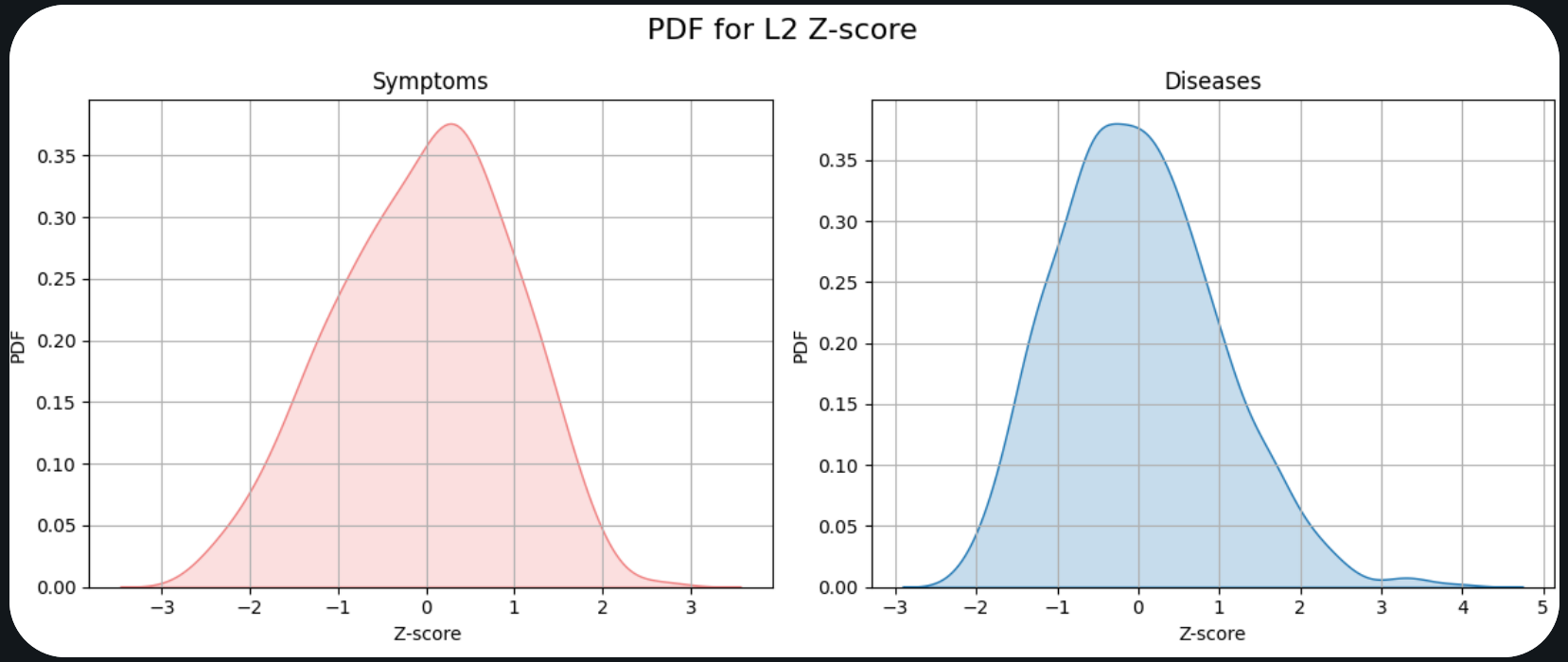
2 Levels

- **Level 1: Degree of the node**

$$SI_{v,1} = \sum_{u} W(v,u)$$

- **Level 2: a symptom is present in diseases affected by numerous other symptoms (SI)**
  disease exhibits symptoms that affect many other diseases (DI)

Method of reflection



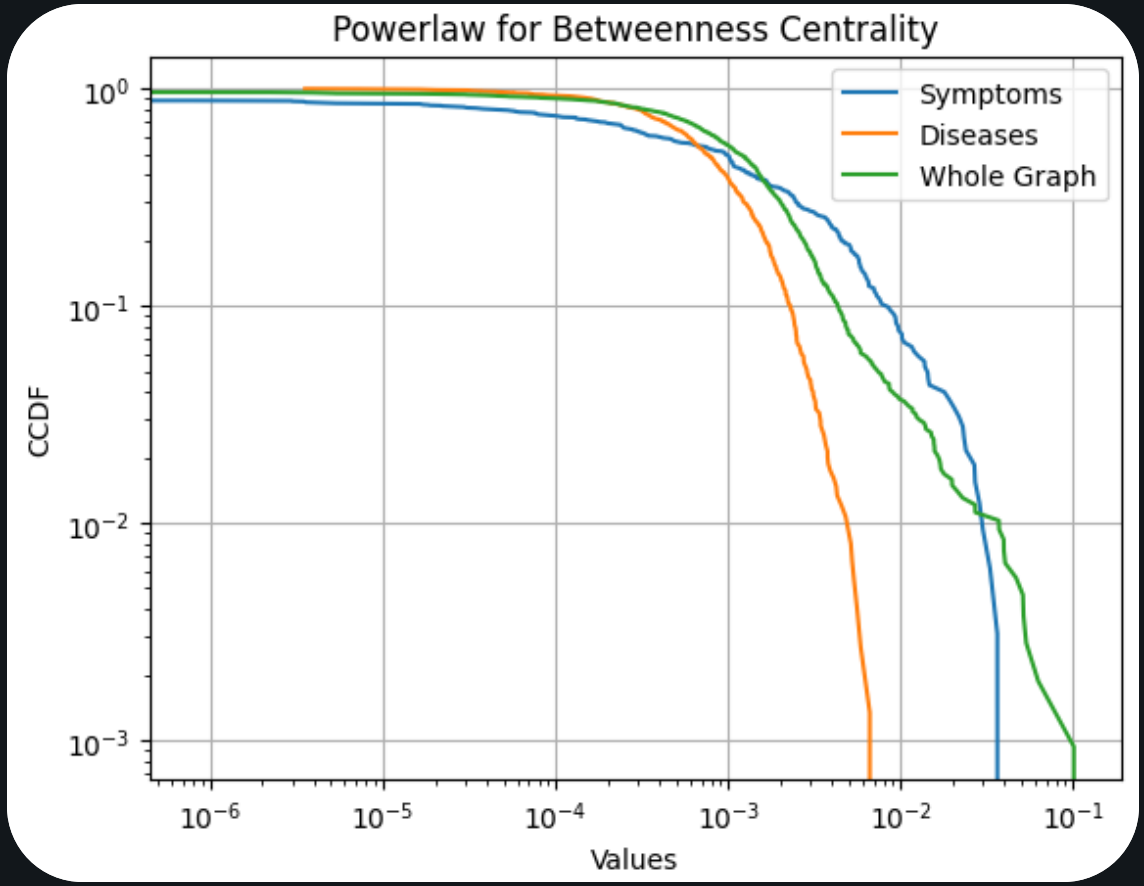
Significance test

- Employed null models
- Mean Close to 0 *and Variance too high*
- *H0 Rejected*

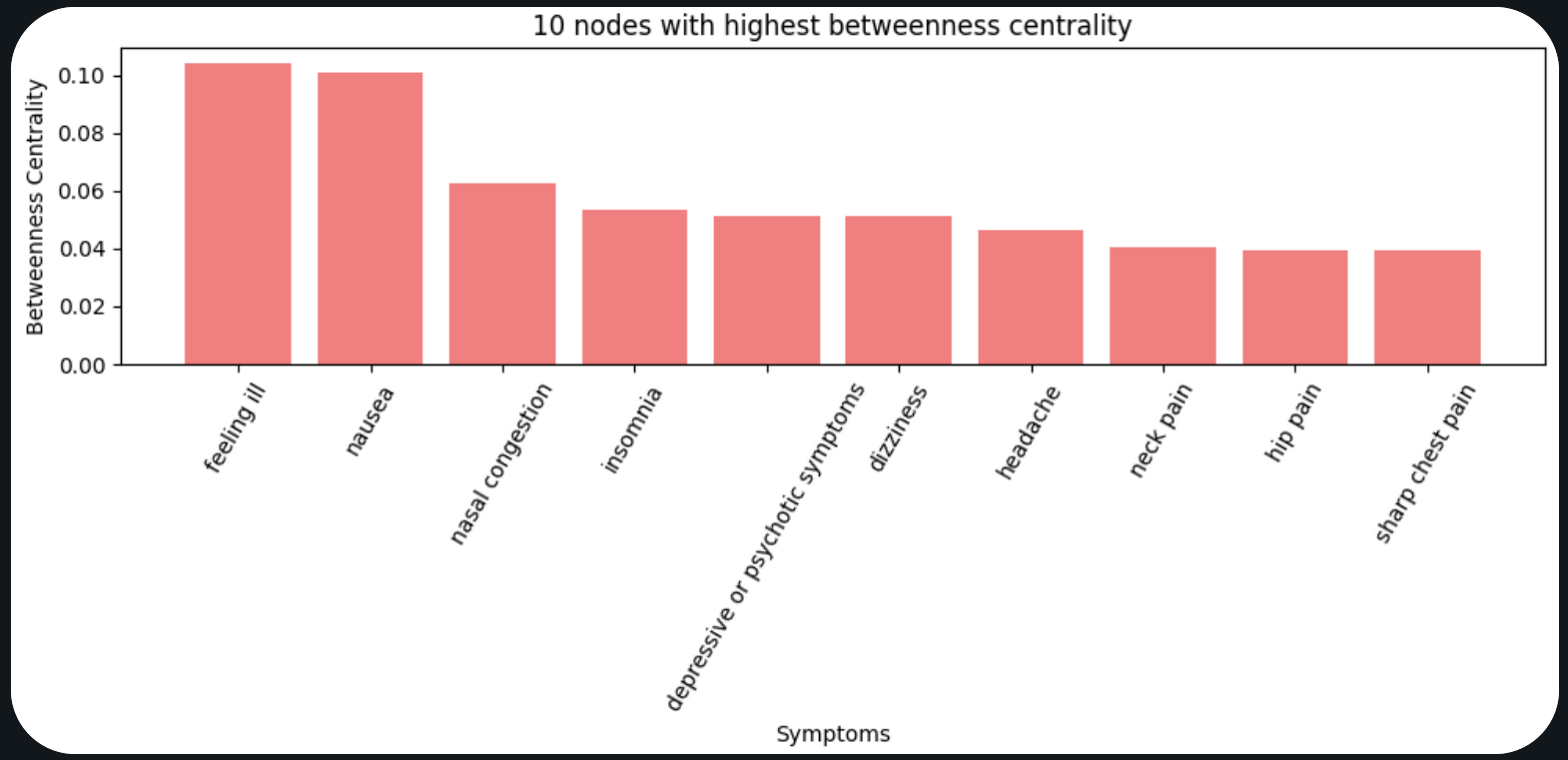
Betweenness Centrality

Power Law Distribution

- Scale-free network with few Hubs
- Symptoms have higher betweenness than diseases
- Symptoms tends to have higher degrees
- Bad under predictive standpoint



Betweenness Centrality



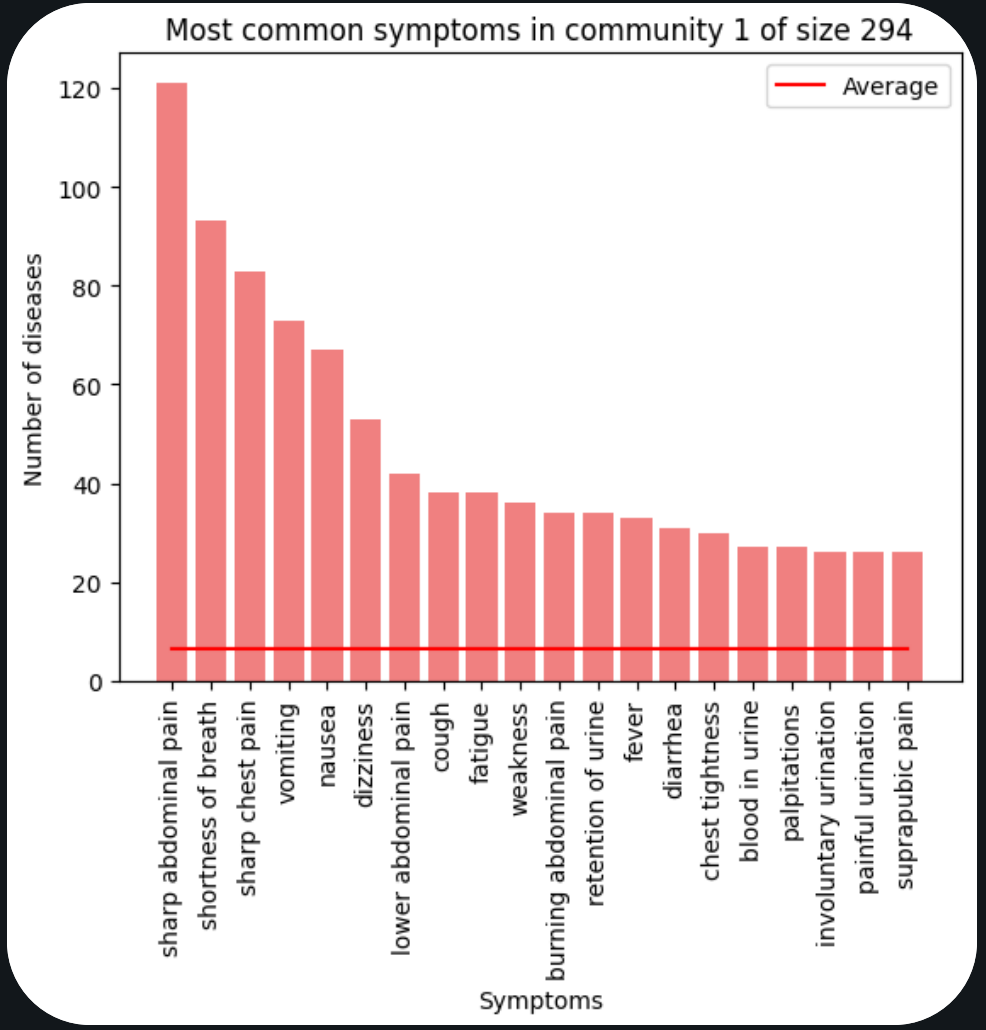
Most influential nodes

- They are all symptoms
- Very commonly present

Communities Detection

Co-occurrence similarity

- Greedy Modularity Maximization
- 3 Communities each
- INFO 1: symptoms in same communities frequently co-occur within same diseases
- INFO 2: symptoms specificity for a given community
- INFO 3: diseases specificity for a given community



Communities Detection

Features

Community Count

- How many symptoms are from a given community
- Each symptom community has different common diseases.
- Model can learn prioritizing diseases from community with highest count

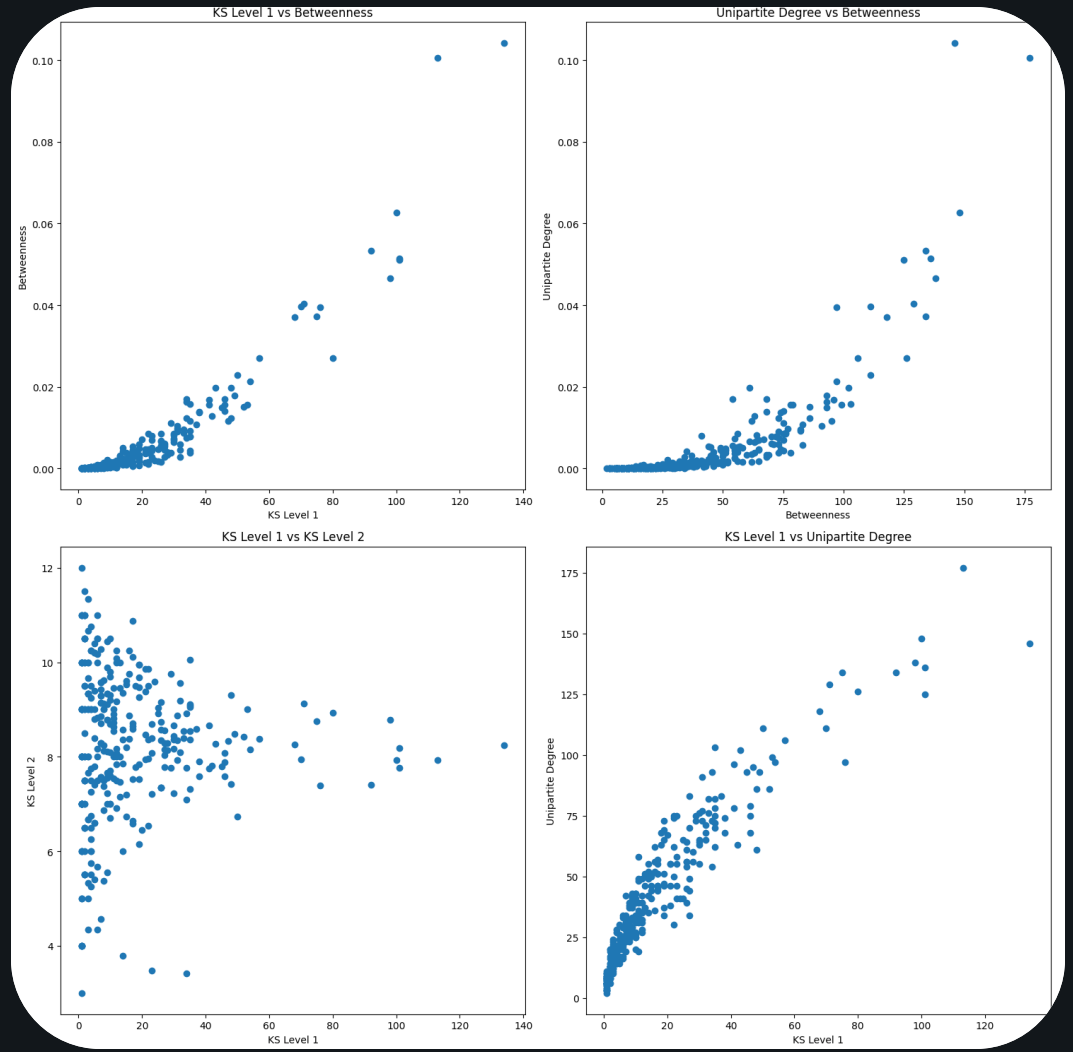
Community Size

- Replace symptom with size of its community
- Each symptom belongs to community of a given size
- Model can distinguish symptoms from large or small communities
- If many symptoms from small community, the diseases of that community may be more likely

Most Important Actors

Features Reduction

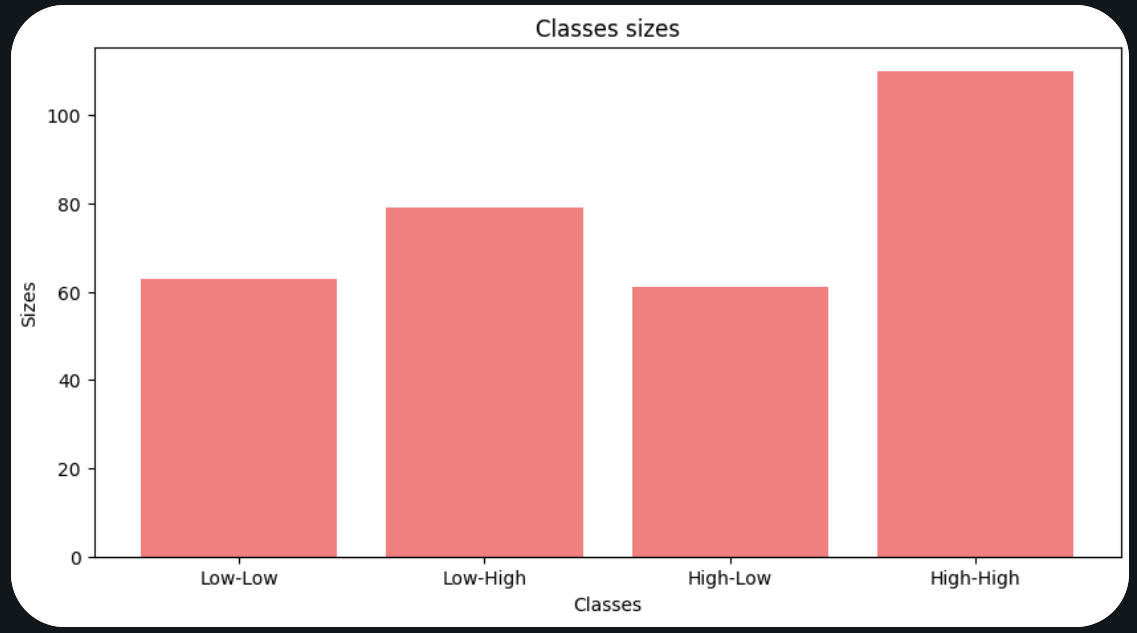
- Various combinations options
- Take the most uncorrelated
- Classification based on thresholds (0.5 * avg deg)



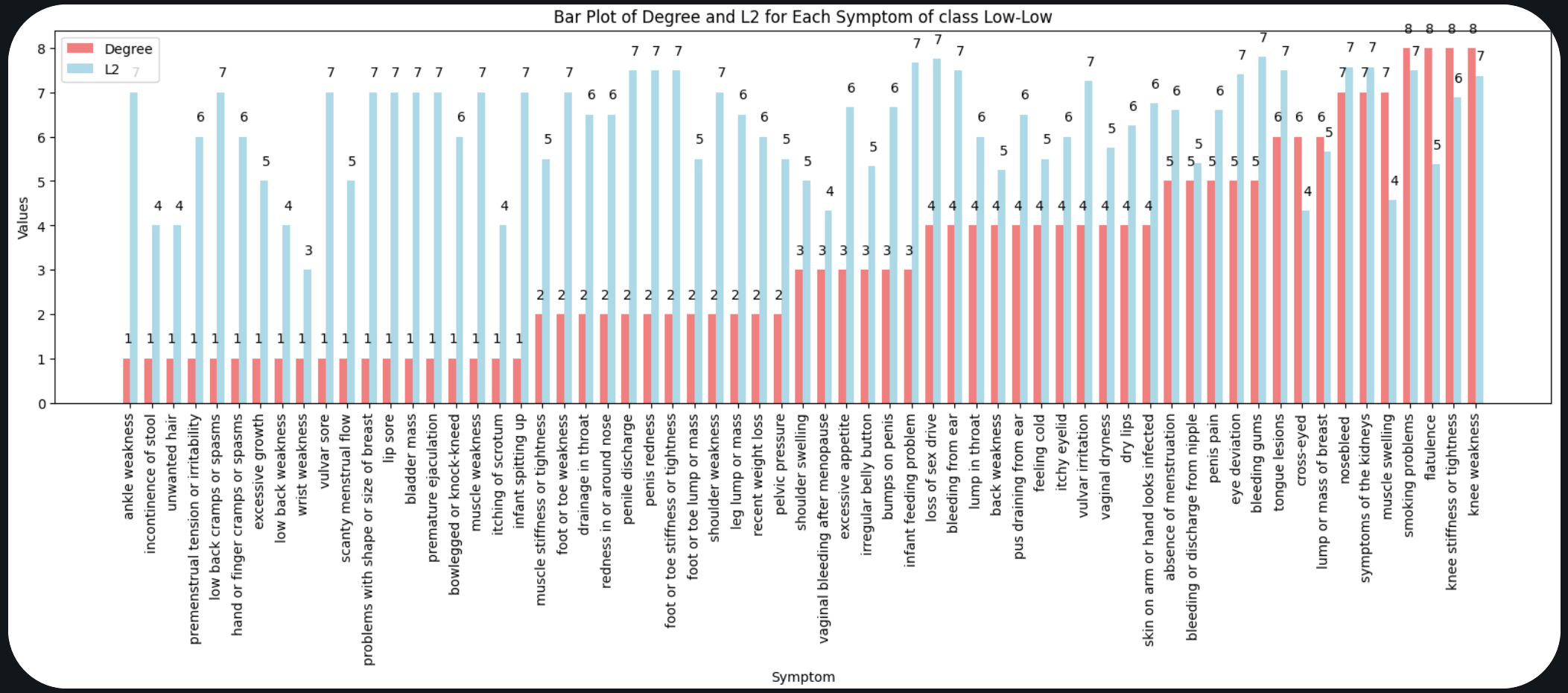
Most Important Symptoms

Provided Insights

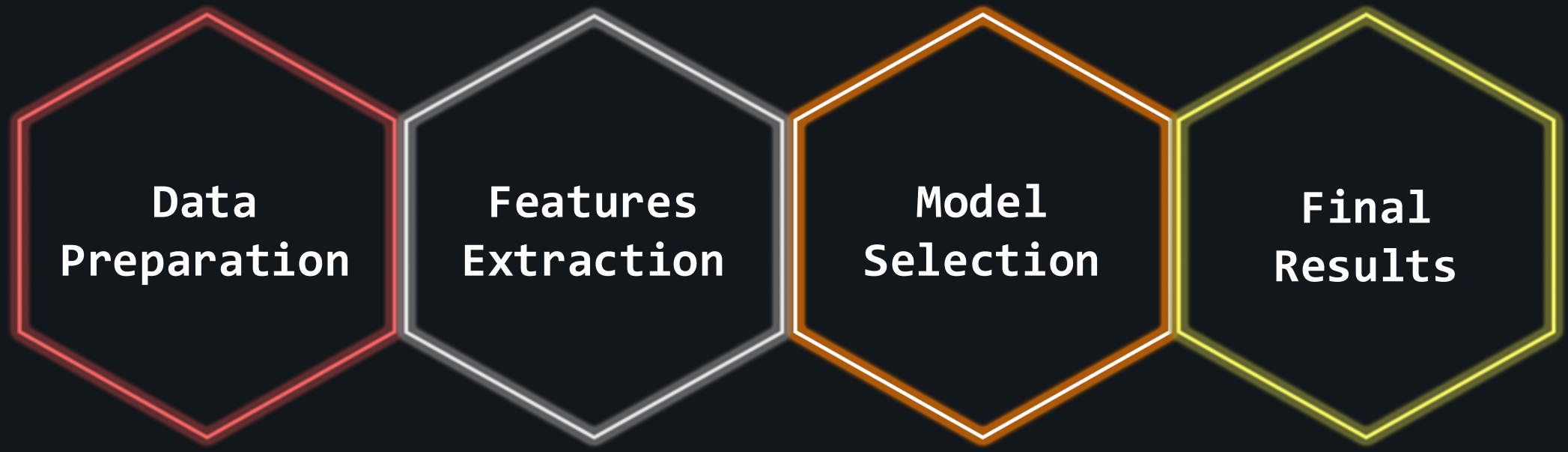
- **Low-Low:** very important for prediction of specific diseases
- **Low-High:** less specific than the first class
- **High-Low:** important in general
- **High-High:** important for overall
- Same analysis done for diseases to find the most symptomatologically complex



Most Important Symptoms



ML Models


Data Preparation
Features Extraction
Model Selection
Final Results

Operative Flow

Core Operations

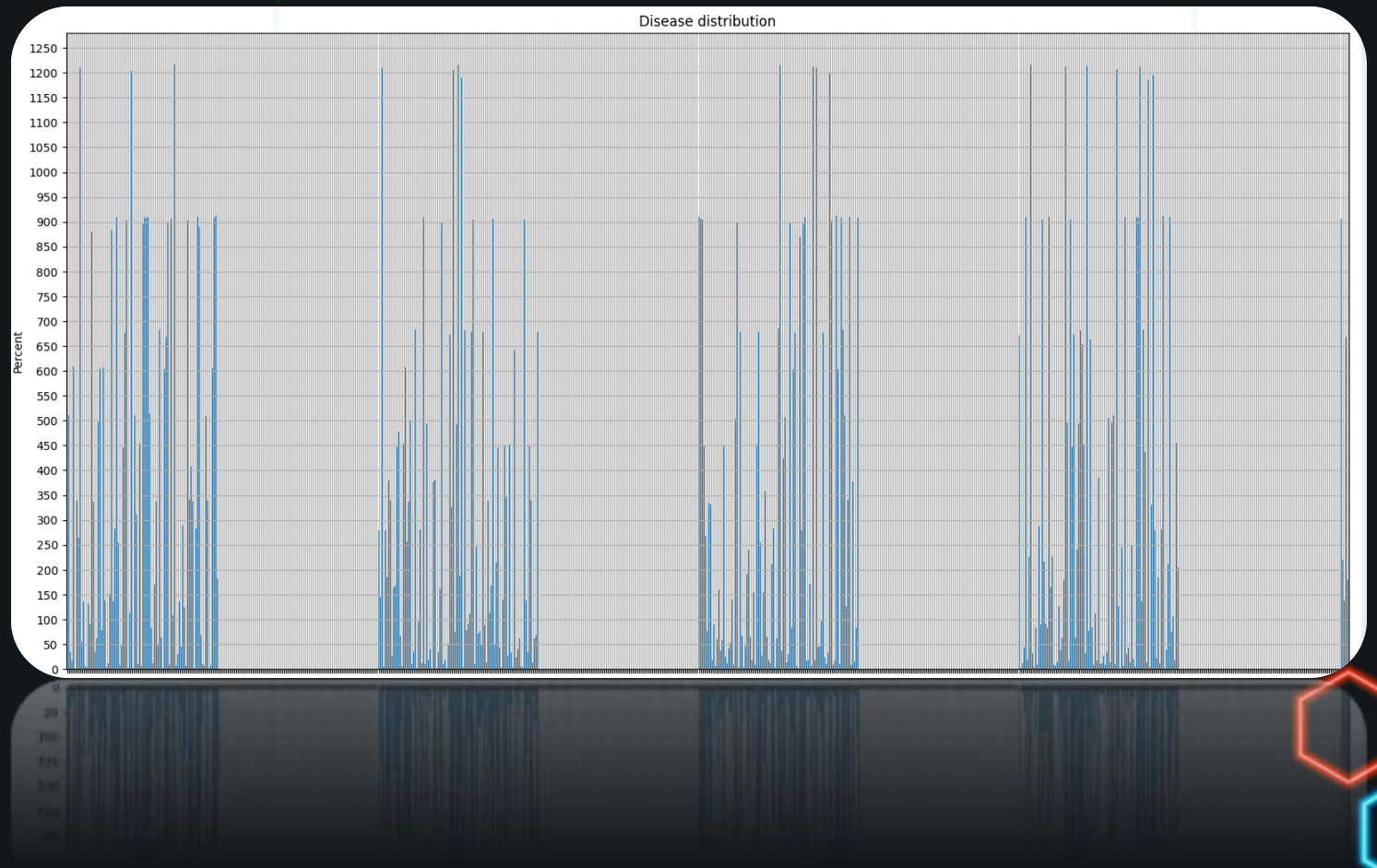
- Sampling + Balancing
- Features Combination
- Hyperparameters Choice
- Model Selection
- Features Reduction

Data Preparation

Random Sampling

- Dataset larger than 250k
- Many operations to be performed
- Random Sampling of 10% of data

Unbalanced Classes



Data Preparation

Oversampling and Undersampling

- Classes with more than 1200 samples
- Classes with less than 10 samples
- Very high delta
- Gain more than 5% accuracy

Balancing Function

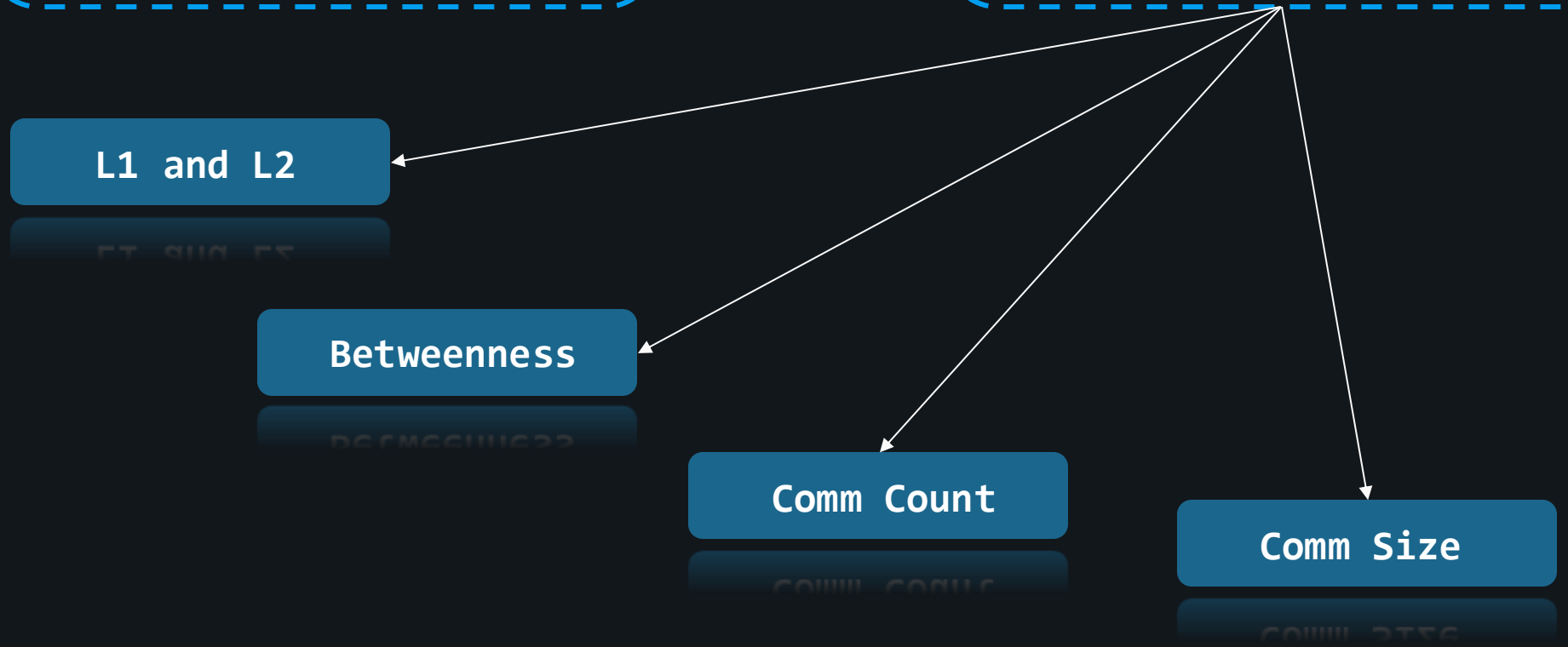
```python
def balanceSampling(features, labels, threshold=35):
    # Over-sample
    original_samples_per_class = {
        label: np.sum(labels == label) for label in np.unique(labels)
    }
    sampling_strategy = {
        label: max(threshold, original_samples)
        for label, original_samples in original_samples_per_class.items()
    }
    ros = RandomOverSampler(random_state=42, sampling_strategy=sampling_strategy)
    oversampled_features, oversampled_labels = ros.fit_resample(features, labels)
    # Under-sample
    updated_samples_per_class = {
        label: np.sum(oversampled_labels == label) for label in np.unique(labels)
    }
    sampling_strategy = {
        label: min(threshold, original_samples)
        for label, original_samples in updated_samples_per_class.items()
    }
    rus = RandomUnderSampler(random_state=42, sampling_strategy=sampling_strategy)
    undersampled_features, labels = rus.fit_resample(
        oversampled_features, oversampled_labels
    )

    return undersampled_features, labels
```

Features Extraction

Classic Features

- **Symptoms** one hot encoding

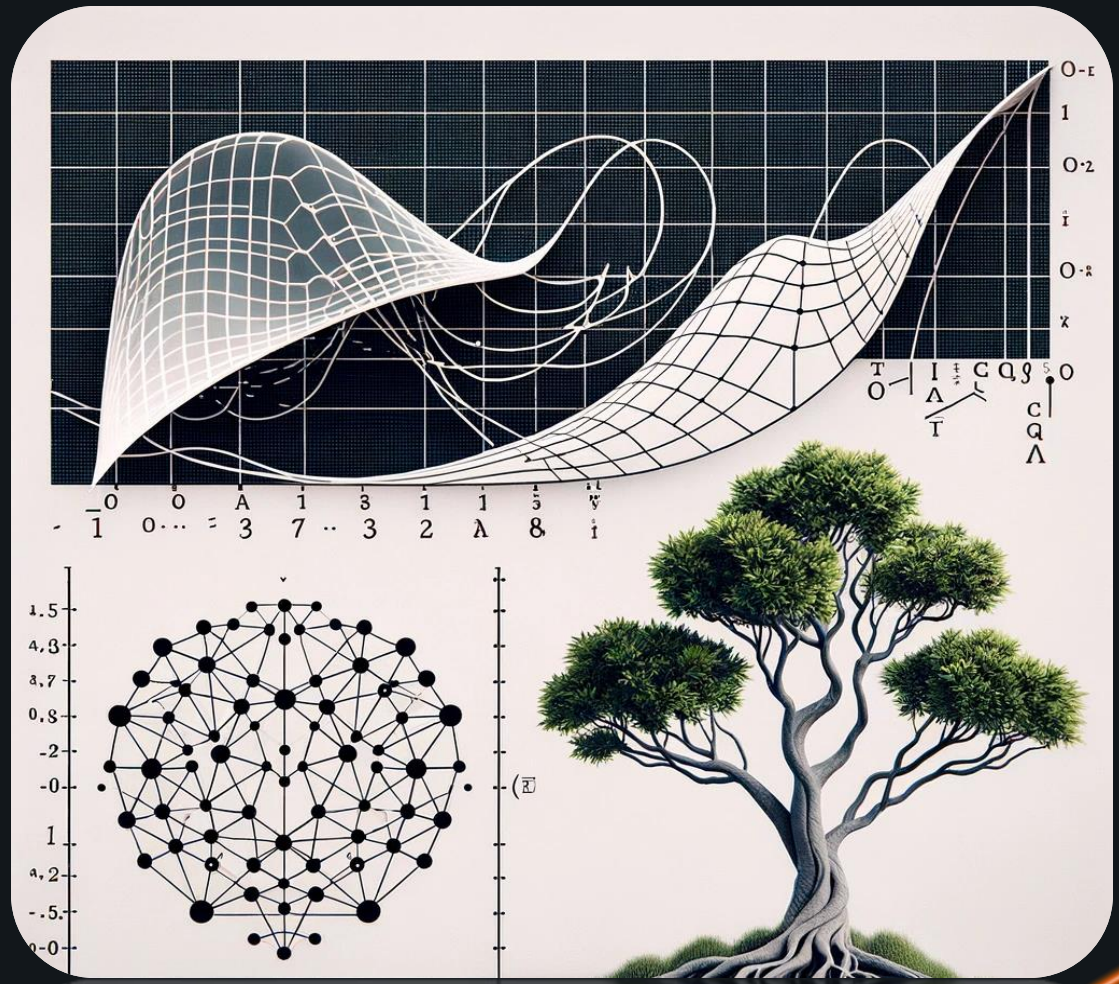
Network Features



Model Selection - Candidate Models

Model Choice

- Random Forest
- Logistic Regression
- Multi-Layer-Perceptron



Model Selection – Features Selection

More complex isn't always better

- Forward stepwise feature selection
- Accuracy maximization

Features to be added (Starting from an empty model)	Beetweenness	Community count	Community size	Symptoms L1	Symptoms L2
1st iteration	29.98%	1.51%	88.62%	57.26%	87.93%
2nd iteration	88.52%	88.64%		88.43%	89.19%
3rd iteration	89.13%	89.21%		89.13%	
		Stopping point			
4th iteration	89.21%			89.13%	
5th iteration				89.06%	

Example: logistic regression

Model Selection - Parameters Tuning

Greedy Approach

- Unfeasible GridSearch approach
- Tuning just one parameter at time
- No best absolute combination
- CrossValidation

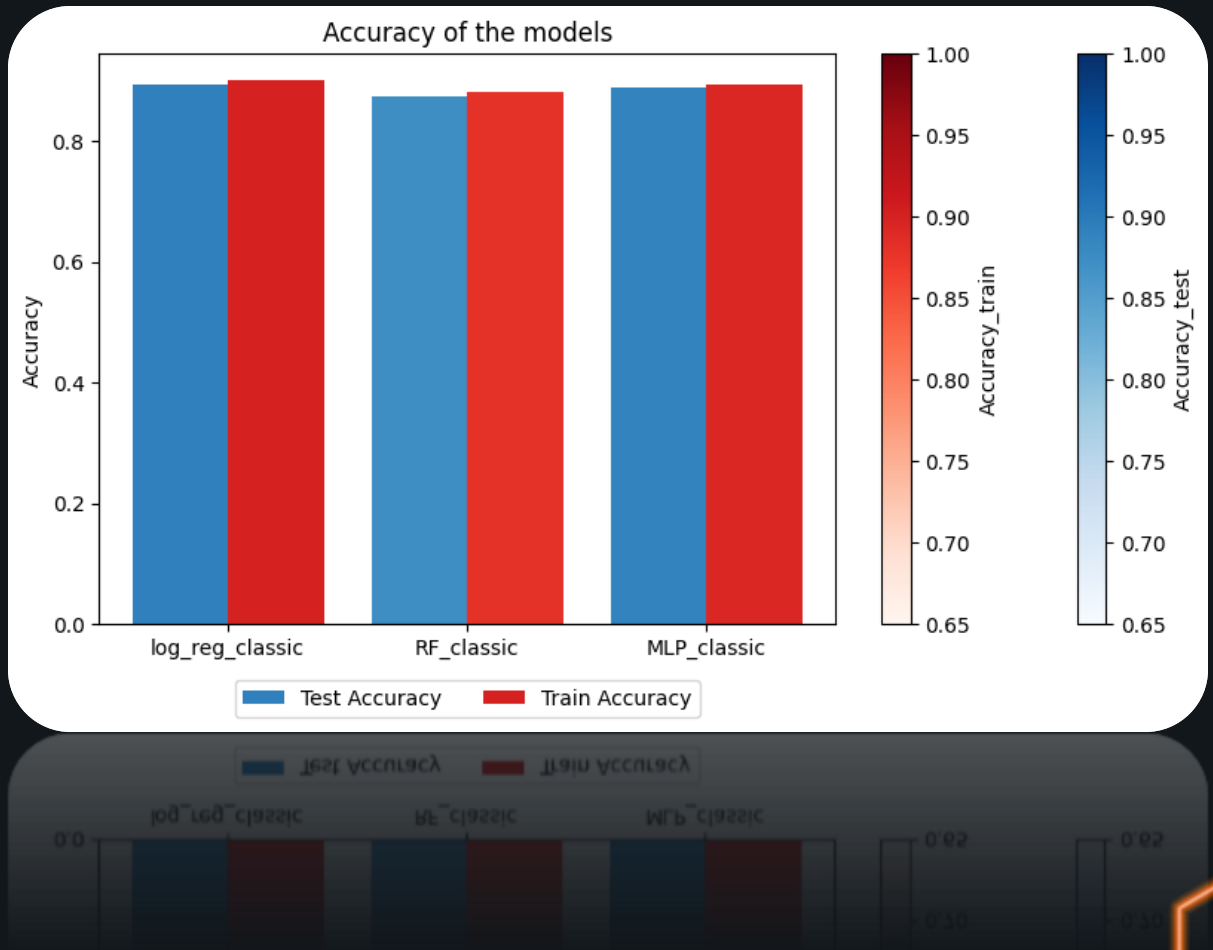
Tuning Process

```python
# Define the parameter grid to search for Random Forest
param_grid = {
    "n_estimators": [100,200,300,500,600],
    "max_depth": [25,50,75,100],
    "min_samples_split": [2,5,10],
    "min_samples_leaf": [1,2,5],
}

# Create the GridSearchCV object
grid_search = GridSearchCV(
    random_forest, param_grid, verbose=3, cv=3, scoring="accuracy", n_jobs=-1
)

# Fit the GridSearchCV object to the data
grid_search.fit(X_train, y_train)
```

Model Selection – Symptoms only

Trained Models

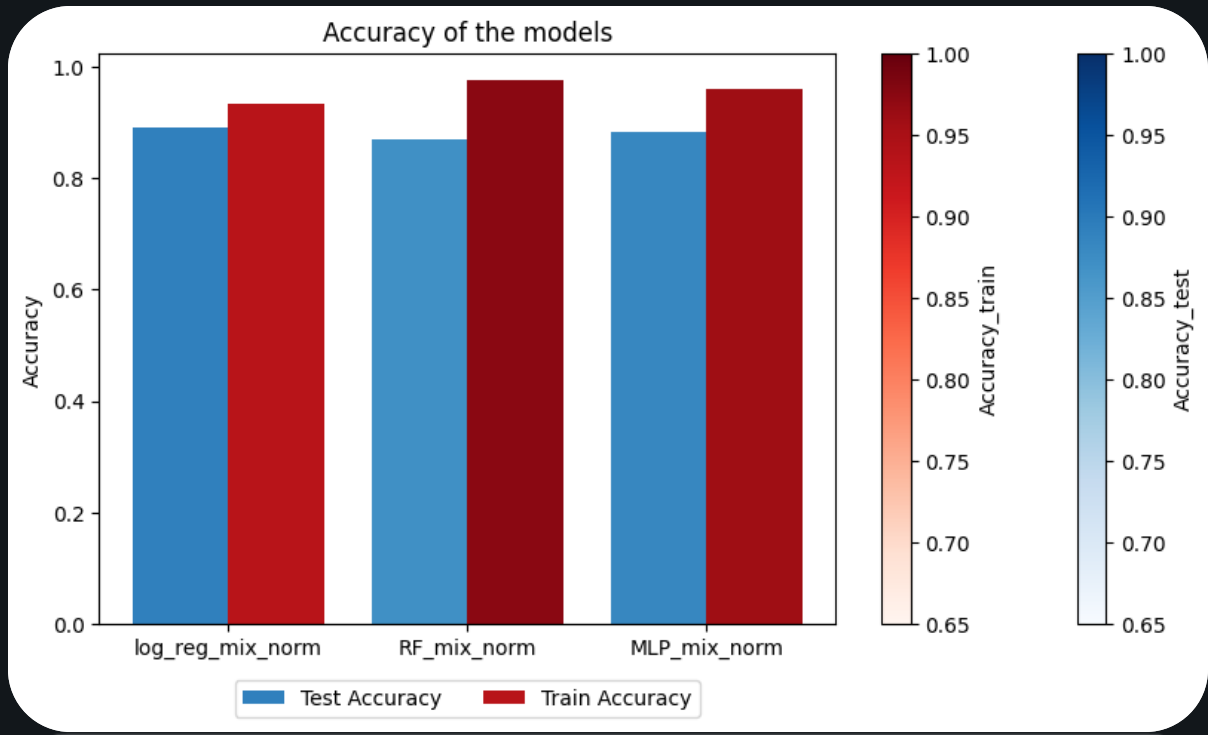
- Logistic Regression
- Random Forest
- MLP Neural Network



Model Selection – New Features

Trained Models

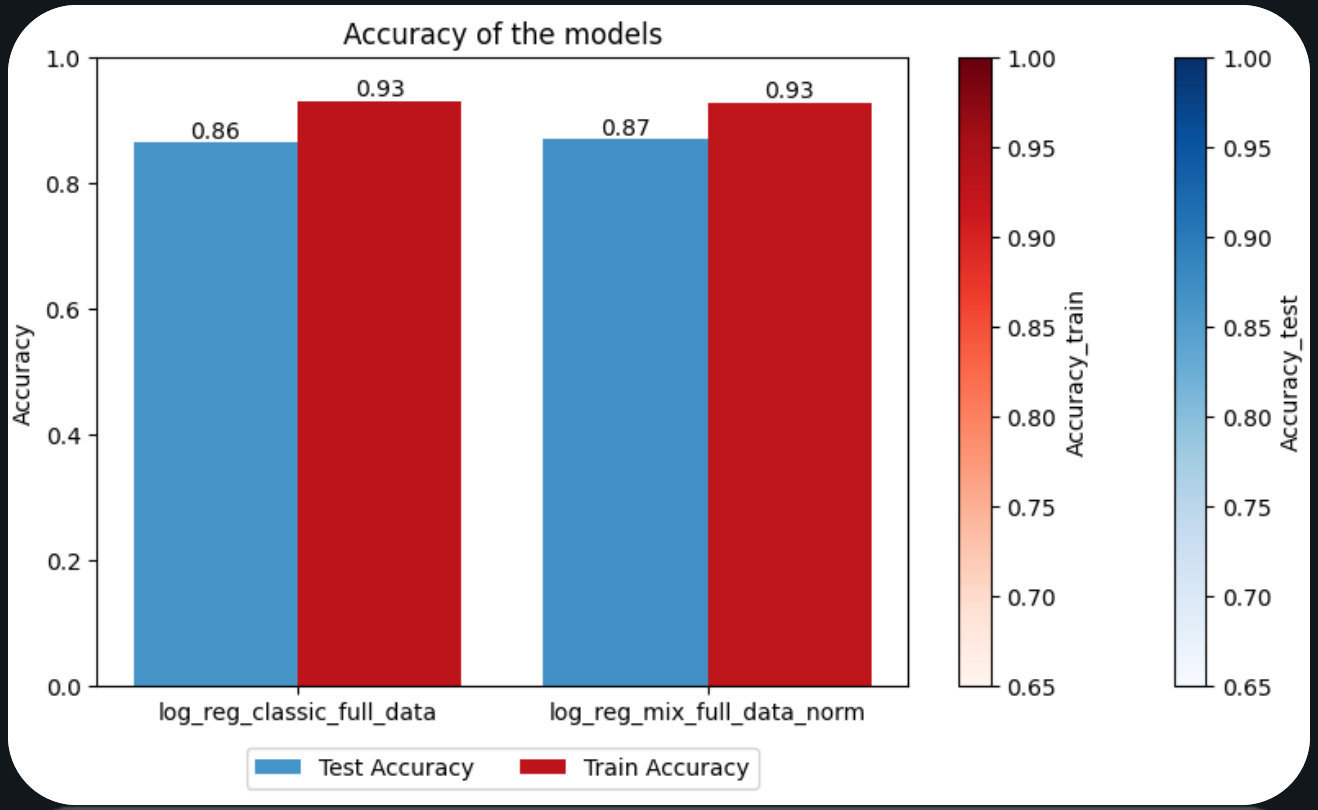
- Logistic Regression
  - ❖ Betweenness, Count, Size
- Random Forest
  - ❖ Betweenness, Count, Size
- MLP Neural Network
  - ❖ Count, Size



Final Results - Network Features Effect

GOAL 1

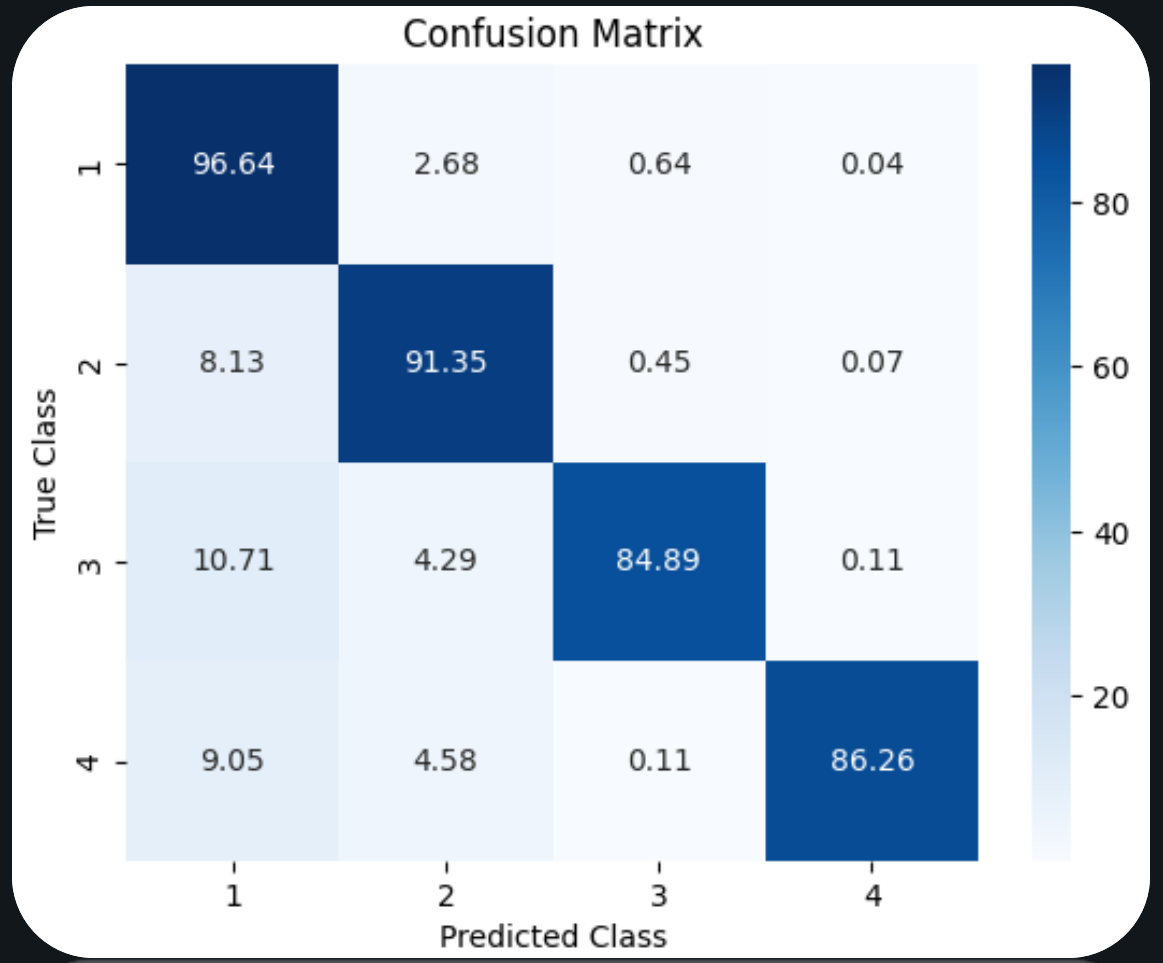
- Equal performance as substitute
- More features thus complexity
- Simplicity of the dataset



Final Results - Best Model Analysis

Performance Analysis

- Classes based on the Disease Influence indexes
- Diseases with low diagnostic accuracy
- Most impactful symptoms



Final Results - Best Model Analysis

Performance Analysis

- Symptoms overlap

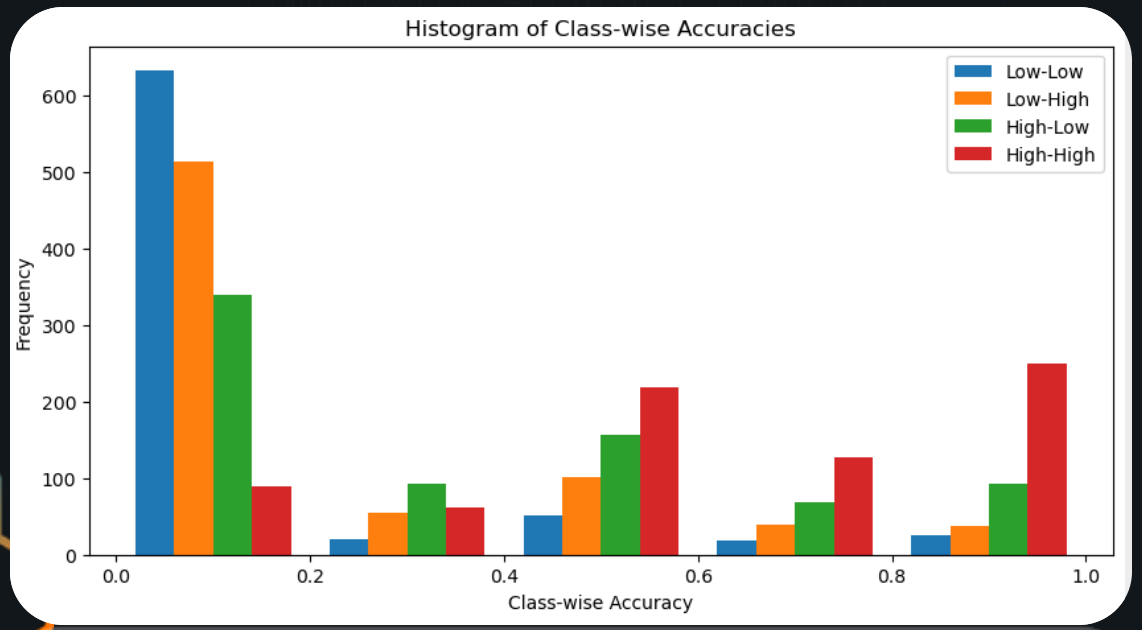
Disease	Accuracy	f1-score
premature ventricular contractions	0.500000	0.666667
histoplasmosis	0.498876	0.560252
hemiplegia	0.483908	0.496462
acute bronchiolitis	0.473684	0.562500
poisoning due to antimicrobial drugs	0.467849	0.567968
open wound of the mouth	0.394890	0.564315
acute otitis media	0.383938	0.468456
vitamin b12 deficiency	0.333333	0.071429
bladder cancer	0.288740	0.378102
otitis media	0.250000	0.181818

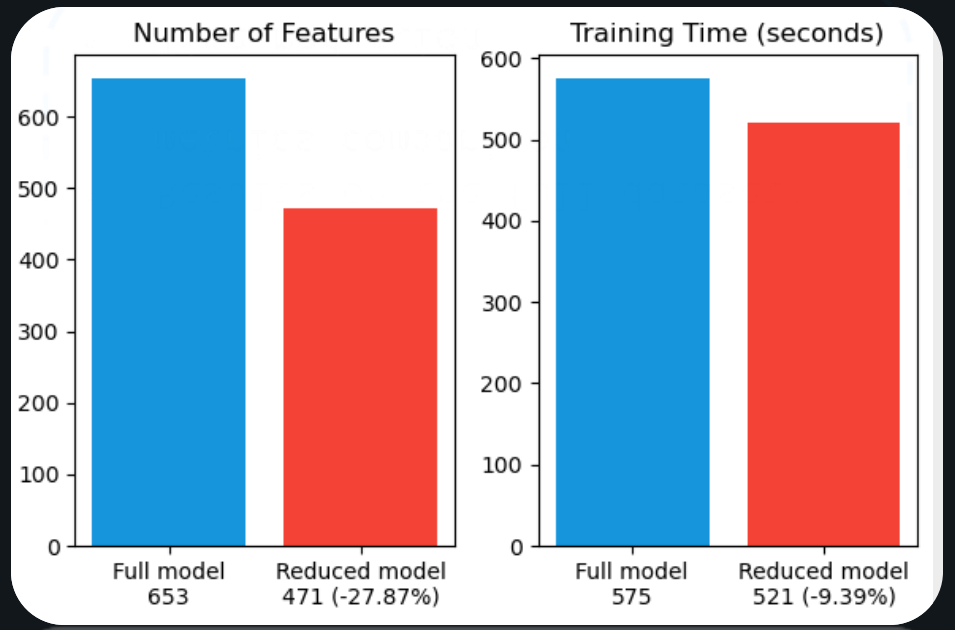
Final Results - Complexity Reduction

- Division based on SI indexes
- Complexity-Accuracy tradeoff

GOAL 2

- Results on the full dataset: metrics comparison
- Time reduction





Conclusion

Achievements:

- Network models have a similar performance with respect to symptoms models
- A good balance between features reduction and model performance was achieved

Limits:

- **Feature selection**
- **Hyperparameters tuning**

A detailed and complete explanation of all the limits can be consulted in the report

Authors

AndreaAlberti07

DavideLigari

CristianAndreoli

Andrea Alberti

TeoScardov