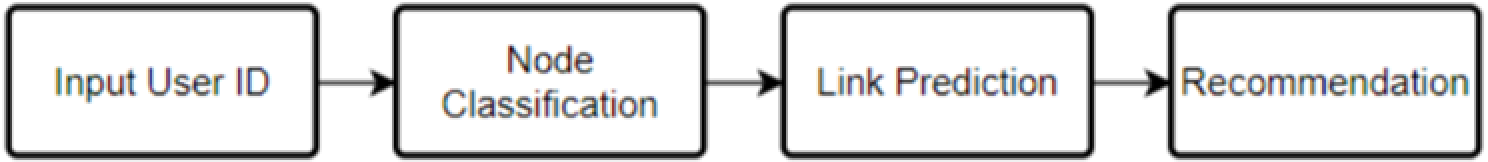
GitHub User Network Analysis and Recommendation System

Outline

- Motivation
- Data Analysis
- Node Classification
    - Node Features Extraction
    - Graph Neural Networks
- Link Prediction
    - Edge Features Extraction
    - Multi-layer Perceptron
- Conclusion

Motivation

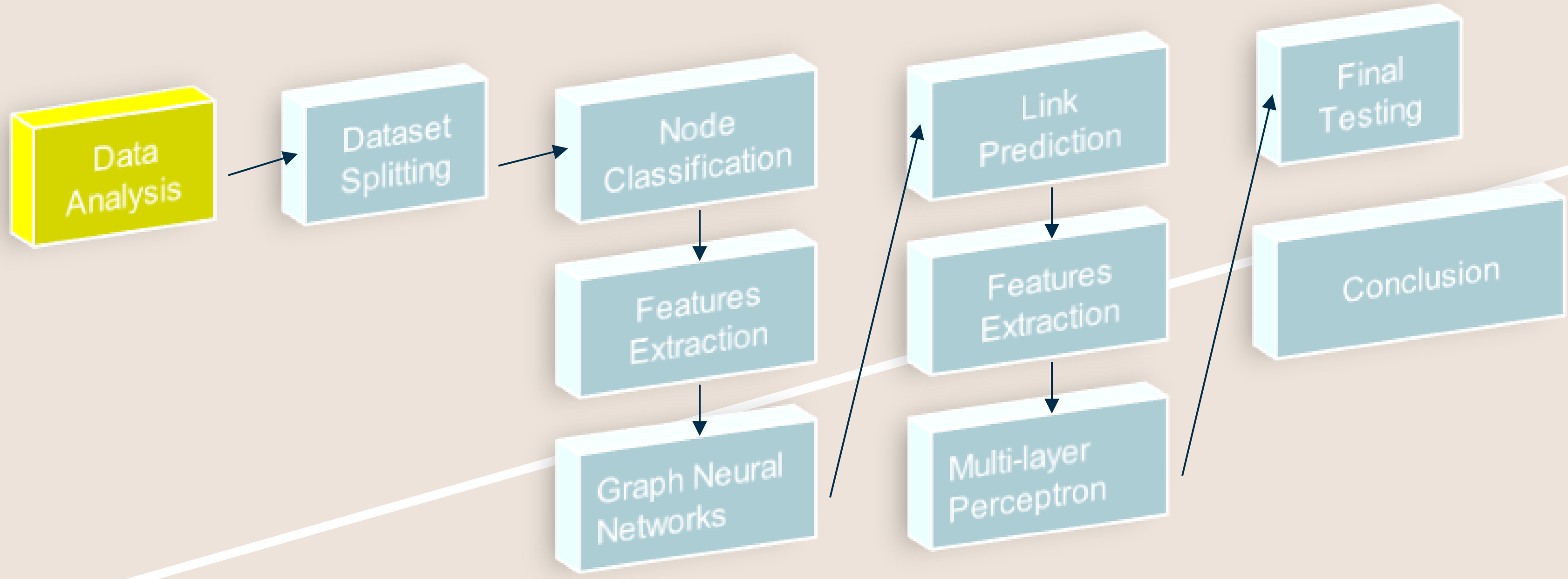
- GitHub recommendation system based on
  - Node classification
  - Link prediction
  - Combination of node classification and link prediction
  - Better link prediction results are expected with more node information



Dataset

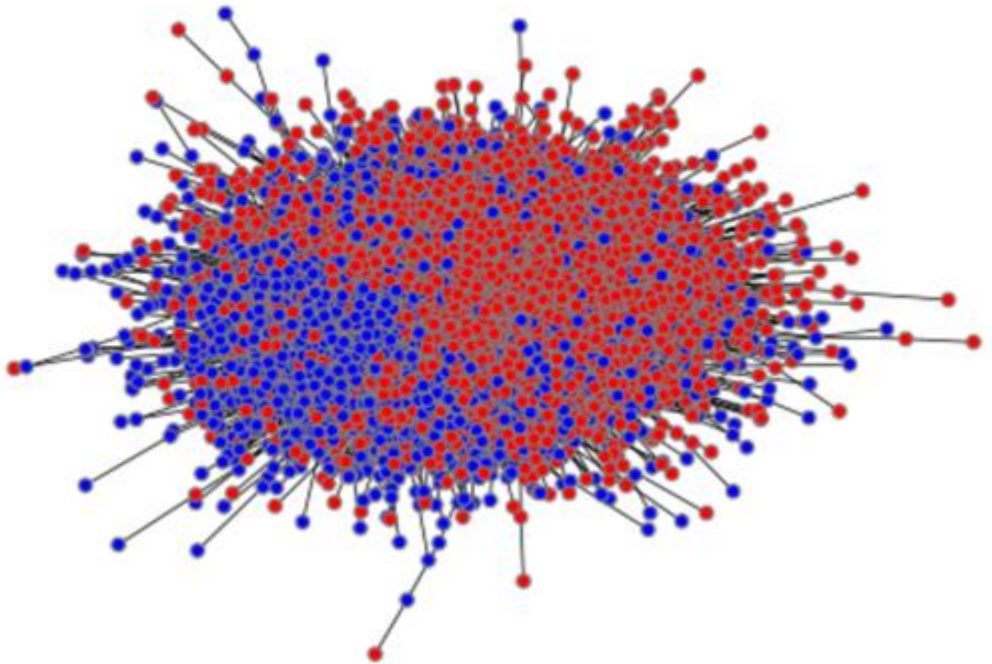
- A large social network of GitHub developers
- Collected in June 2019
- Nodes
  - Users stared at least 10 repositories
  - Either ML developer or Web developer
- Edges
  - Mutual follower relationships between users
- Vertex Features
  - Location
  - Repositories starred
  - Employer
  - E-mail address

Data Analysis



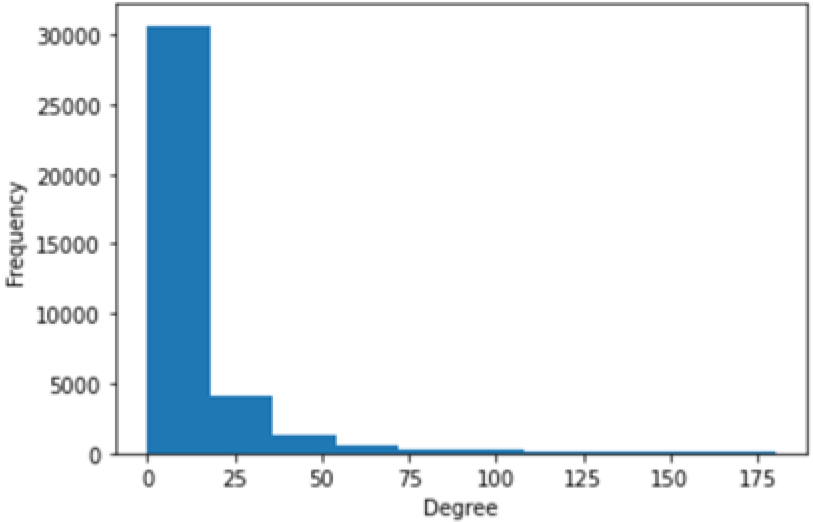
Visualization of the Network

- Library used:
  - Networkx
- Number of nodes: 37,700
- Number of edges: 289,003
- Node color:
  - Red: Machine Learning Developer
  - Blue: Web Developer



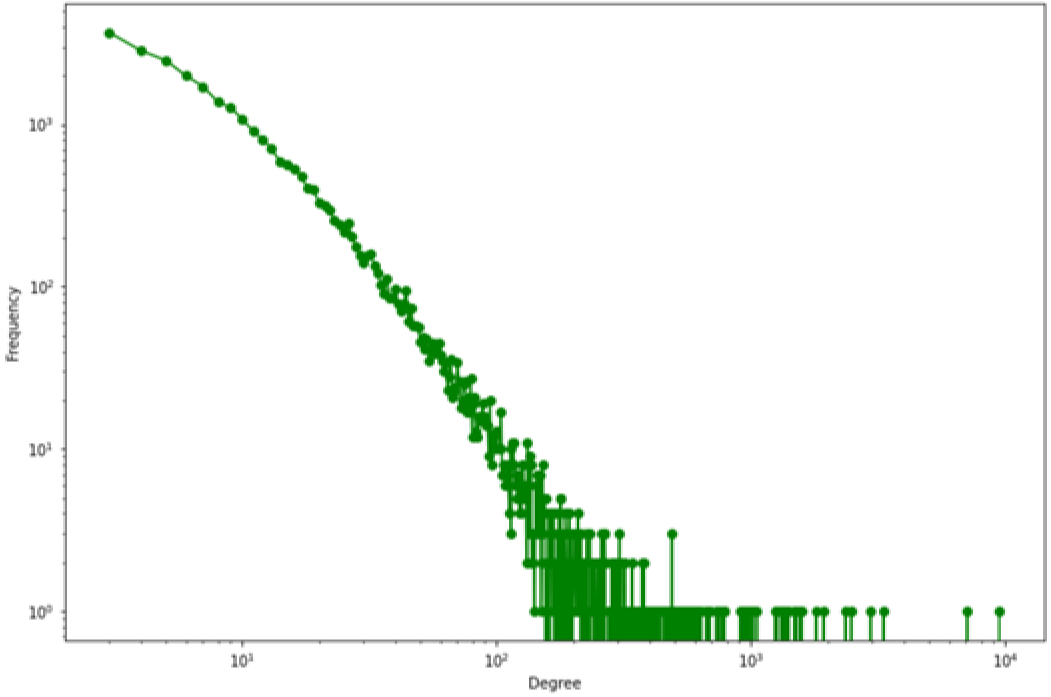
Degree Distribution

- Histogram
    - Most users has a low degree
    - Only very few users has a high degree



Degree Distribution

- Log-log plot
  - Has a long tail
- Follows power law



Average Shortest Path Length

- Networkx average_shortest_path_length function
- The average shortest path length is 3.2464
- Small world
- Users can reach to any other users easily

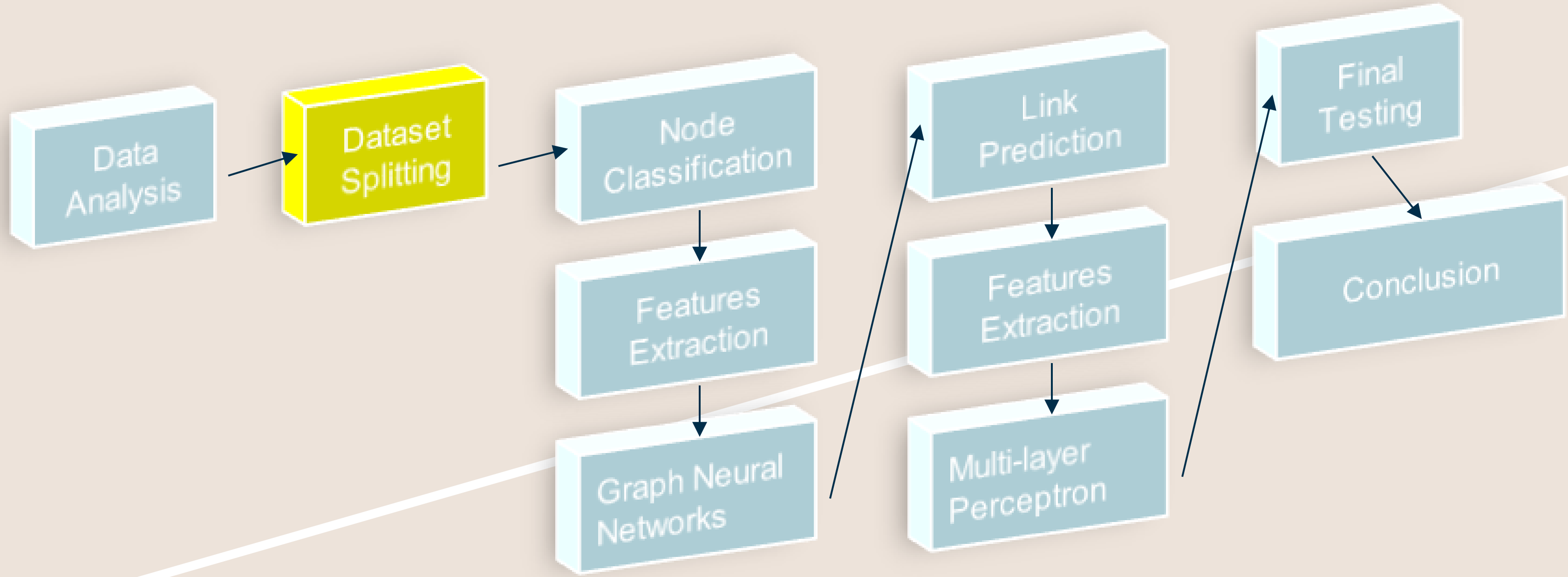
Betweenness Centrality

- Top 10 important nodes
- Normalized Betweenness Centrality
  - Networkx betweenness_centrality function

Betweenness Centrality (Top 10 important nodes)

Node id	User Name	Normalized Betweenness Centrality
31890	dalinhuang99	0.2695992678587705
27803	nfultz	0.2405414222349693
19222	Bunlong	0.0553227478416067
35773	addyosmani	0.0434081325634308
13638	gabrielpconceicao	0.035337494180908
36652	rfthusn	0.0308403065104948
10001	ronenhamias	0.0276116678118553
18163	nelsonic	0.0257991619039893
33671	shayan-taheri	0.0212605764505683
19253	JonnyBanana	0.0203152624455979

Dataset Splitting



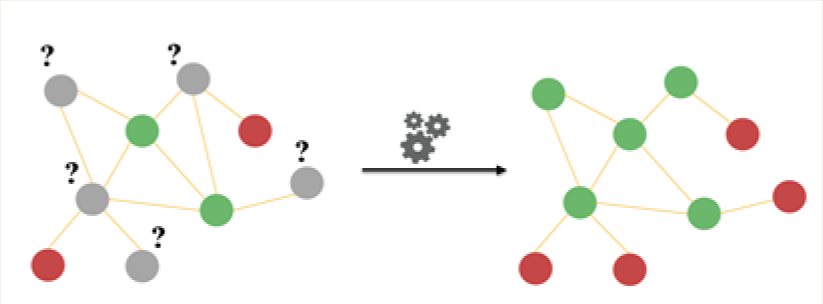
Dataset Splitting: Results

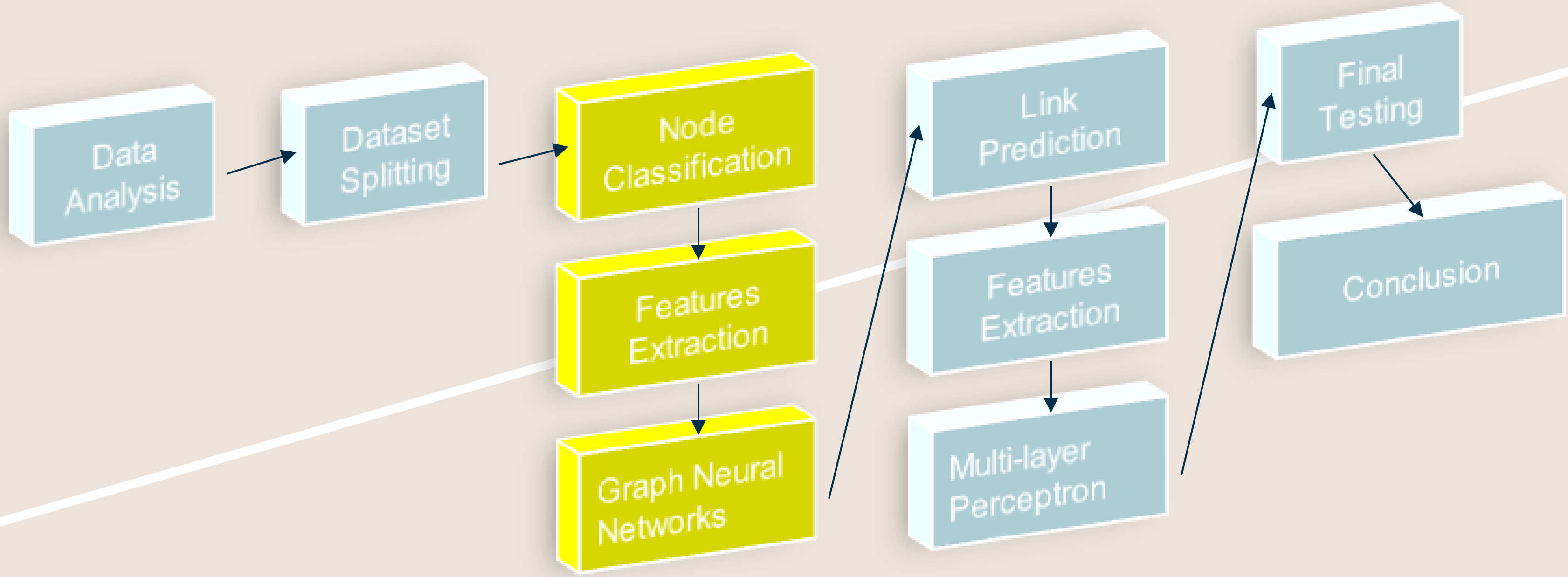
Graph	Number of Nodes	Positive Edges
Original Graph	37700	289000
Sub Graph	37700	234000

Node Set	Number of Nodes
Training Node Set	330160
Validation Node Set	3770
Testing Node Set	3770

Edge Set	Positive Edges	Negative Edges
Training Edge Set	193300	189600
Validation Edge Set	21400	21100
Testing Edge Set	23900	23300
Final Test Edge Set	5000	5600

Node Classification





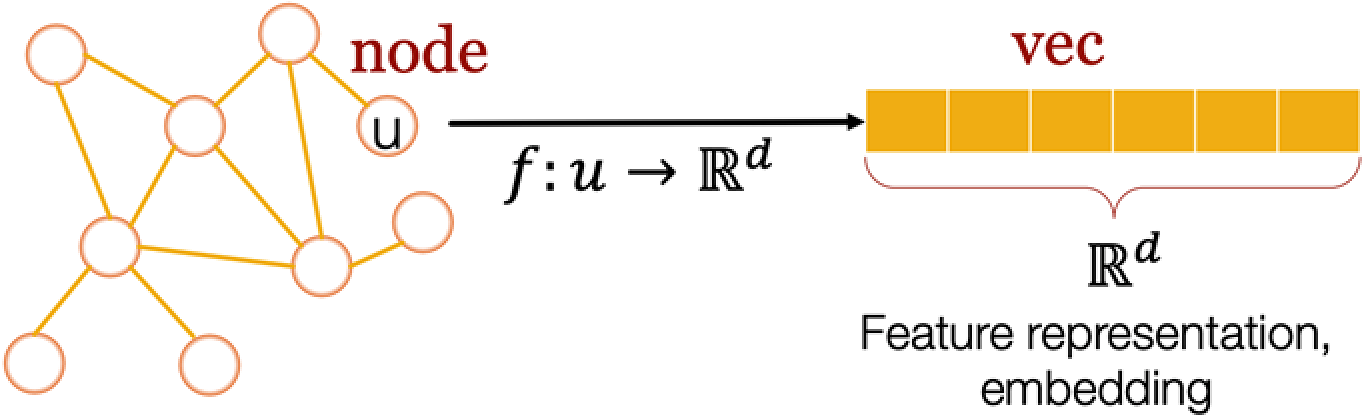
Node Features

Two methods:

1. Feature learning using node2vec
2. Dataset attributes

Node Features
Method 1 : Feature Learning

- Map nodes to low-dimensional space to represent network
- Random walk approach using node2vec
- Semi-supervised learning



Graph Neural Network

- Aggregate and combine node features of neighbours
- Two types of networks:
  1. Graph Isomorphism Network (GIN)
  2. Approximate Personalized Propagation of Neural Predictions (APPNP)

Traditional Graph Convolutional Network (GCN)

<u>Problem:</u>

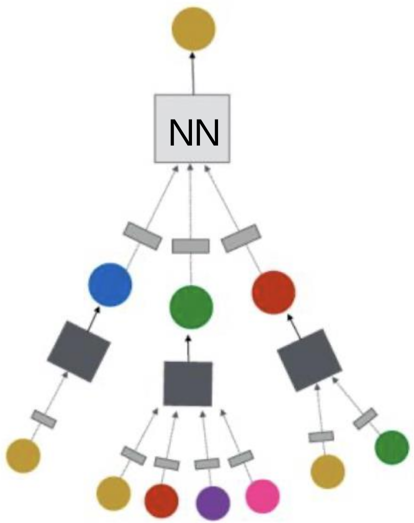
Increase size of neighbourhood

→ too many layers

→ converges to random walk's limit distribution

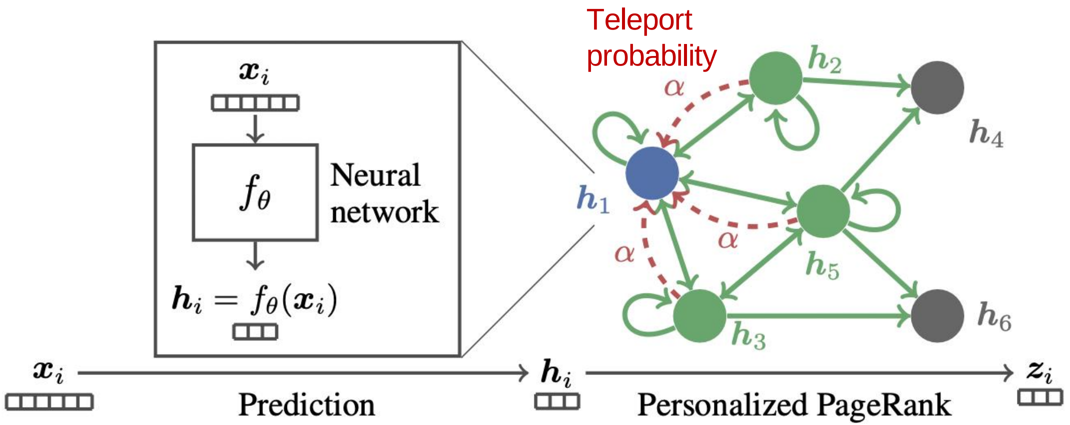
→ oversmoothing

- Feature becomes independent of the root node



Approximate Personalized Propagation of Neural Predictions (APPNP)

- From the paper “Predict then Propagate: Graph Neural Networks meet Personalized PageRank” (ICLR 2019)



$$
\begin{aligned}
\boldsymbol{Z}^{(0)} &= \boldsymbol{H} = f_\theta(\boldsymbol{X}), \\
\boldsymbol{Z}^{(k+1)} &= (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(k)} + \alpha\boldsymbol{H}, \\
\boldsymbol{Z}^{(K)} &= \operatorname{softmax}\left((1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(K-1)} + \alpha\boldsymbol{H}\right)
\end{aligned}
$$

- Teleport probability allows “restarts” → preserve local neighborhood

Experimental Results: GIN aggregator type

No.	Features	Network	Aggregator	Test Accuracy
1	node2vec	GIN	sum	0.8027
2			mean	0.8491
3			max	**0.8501**

- Mean and max aggregators perform better than sum aggregator

Experimental Results: Attribute Dimensionality Reduction

No.	Features	Network	Number of feature dimensions	Aggregator	Test Accuracy
1	Attributes	GIN	128	max	0.8531
2				mean	**0.8533**
3			256	max	0.8520
4				mean	0.8523

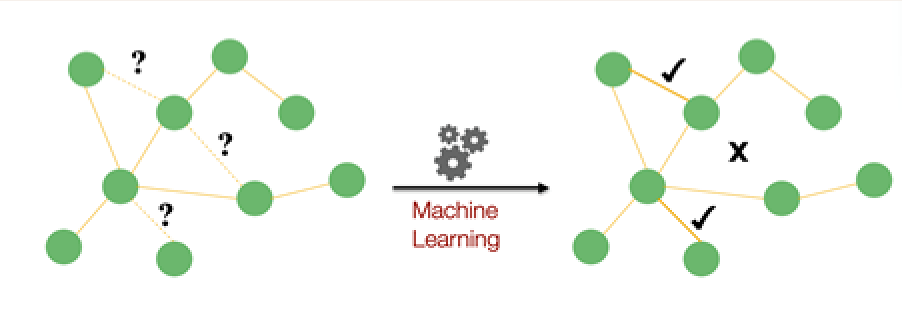
- Increasing number of dimensions does not help with node classification

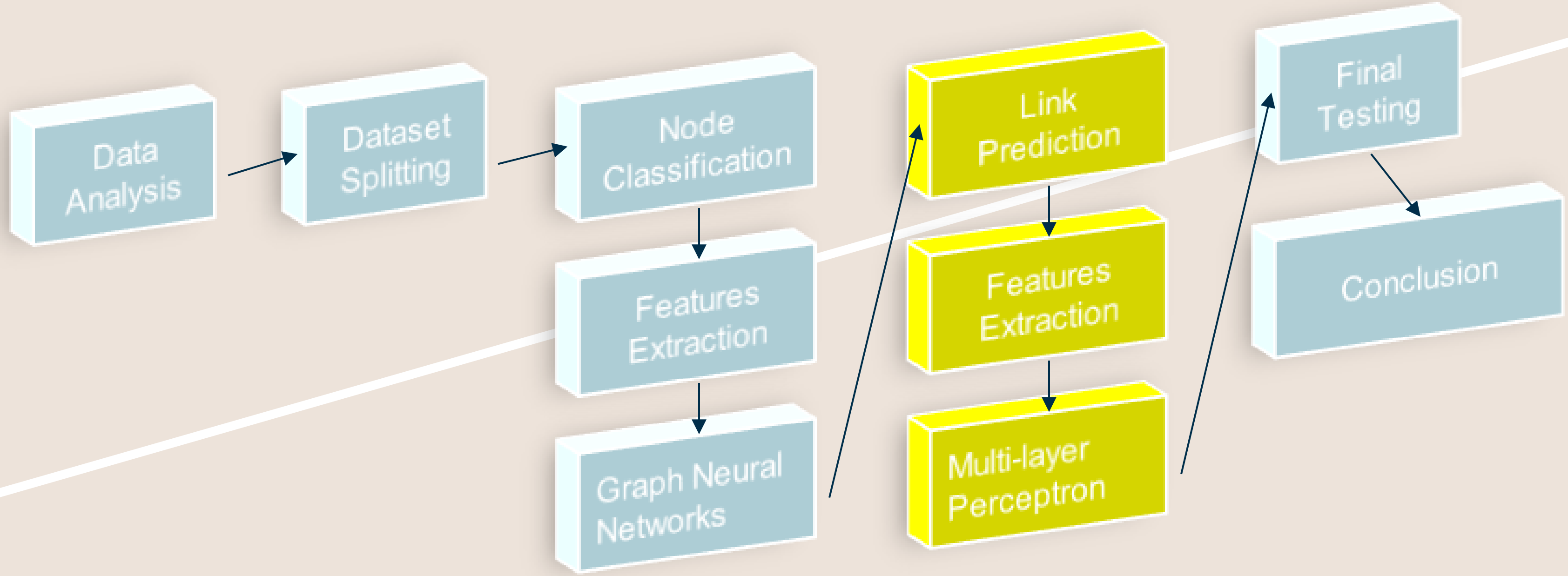
Experimental Results:
APPNP vs GIN | node2vec vs attributes

No.	Features	Network	Aggregator (if applicable)	Test Accuracy
1	node2vec	APPNP	-	0.8507
2		GIN	max	0.8501
3	attributes	APPNP	-	**0.8615**
4		GIN	max	0.8533

- APPNP performs better than GIN
- Using attributes is more effective than using node2vec node embeddings

Link Prediction





Edge Features

- Three set of edge features
- Indices and coefficients for edges
- Edge embeddings from node embeddings
- Feature from node labels

Indices and Coefficients for Edges

Resource Allocation Index of u and v:

$$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(w)|}$$

where $\Gamma(u)$ denote denotes the set of neighbors of u.

Jaccard coefficient of nodes u and v is defined as

$$\frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

where $\Gamma(u)$ denote denotes the set of neighbors of u.

preferential attachment score of nodes u and v:

$$|\Gamma(u)||\Gamma(v)|$$

where $\Gamma(u)$ denote denotes the set of neighbors of u.

$$|\Gamma(u) \cap \Gamma(v)| + \sum_{w \in \Gamma(u) \cap \Gamma(v)} f(w)$$

where $\Gamma(u)$ denote denotes the set of neighbors of u, and $f(w)$ equals to 1 if w belongs to the same community as u and v, 0 otherwise.

Resource Allocation Index Soundarajan Hopcroft

$$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{f(w)}{|\Gamma(w)|}$$

where $\Gamma(u)$ denote denotes the set of neighbors of u, and $f(w)$ equals to 1 if w belongs to the same community as u and v, 0 otherwise.

Ratio of within- and inter-cluster common neighbors for u and v: $\frac{wc}{ic}$

wc: within-cluster common neighbor w for u and v (w is in the same community as u and v)

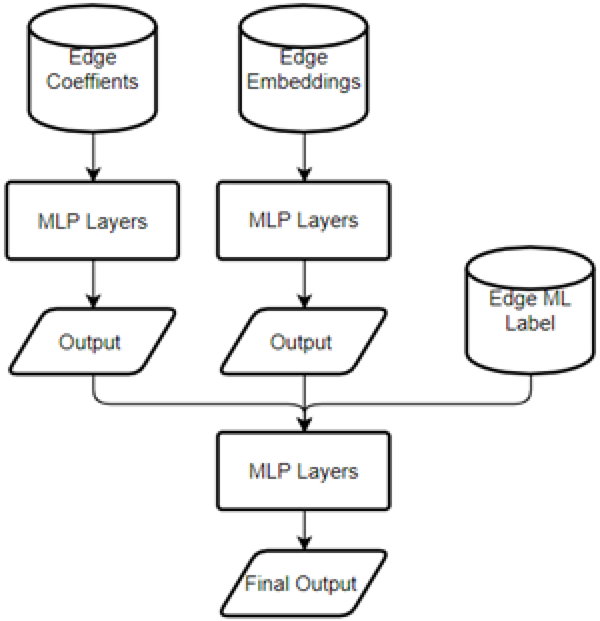
ic: inter-cluster common neighbor w for u and v (w is not in the same community as u and v)

Edge embeddings from Node Embeddings

- Four attempted binary operations
- Binary operations on the node embeddings to form edge embeddings
- Aims to capture the features or similarities of the two end nodes for an edge
- Element by element multiplication
- Element by element absolute difference
- Element by element square difference
- Element by element average

Link Prediction: Model Training

- MLP is used
- Cannot concatenate the features directly
- Two separate sets of layers for the first two sets of edge features
- Combined the layer outputs with the third feature



Link Prediction: Model Selection

Experiment results on model trained by extracting edges features from node embeddings obtained by random walk

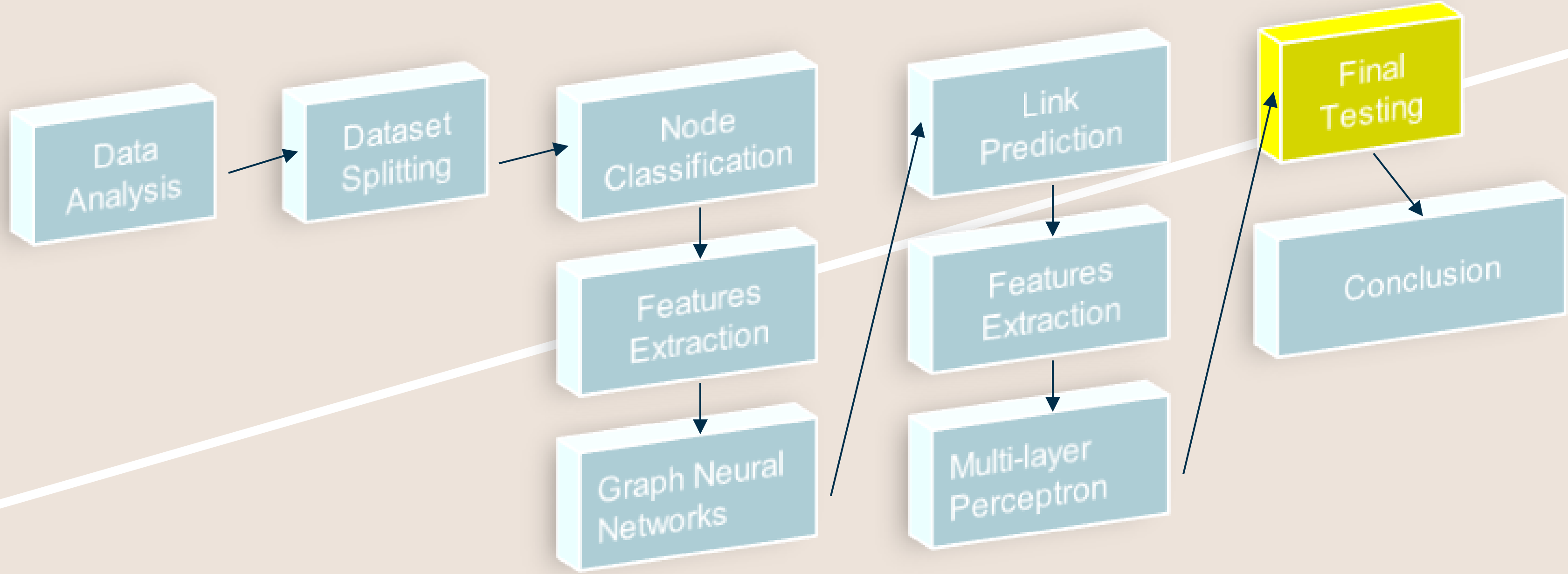
Binary Operations	Accuracy	Precision
multiplication	76.07%	81.13%
absolute difference	72.55%	78.87%
average	80.00%	55.30%
square difference	72.31%	78.87%

Experiment results on model trained by extracting edges features from original node features

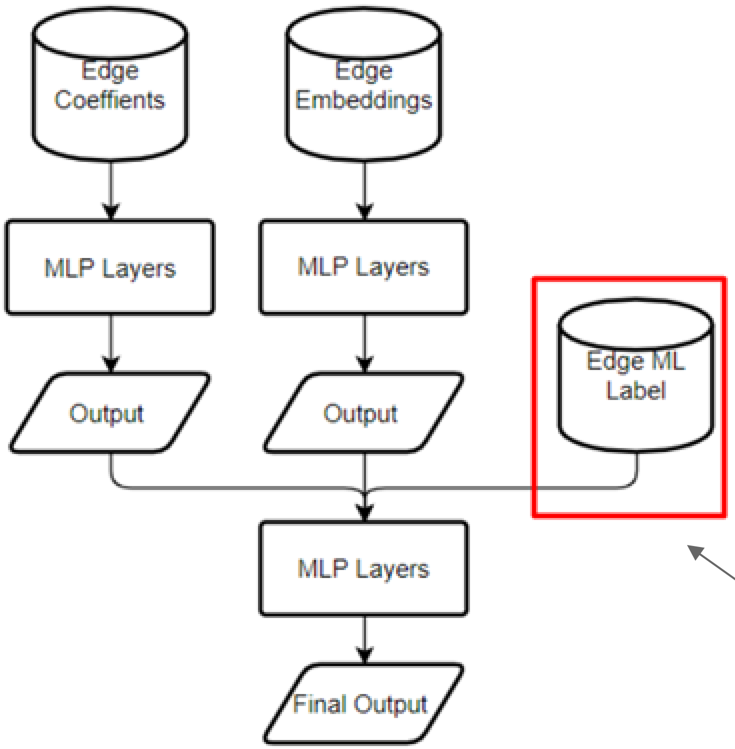
Binary Operations	Accuracy	Precision
multiplication	85.31%	85.31%
absolute difference	85.68%	85.68%
average	89.60%	89.60%
square difference	85.85%	85.85%

Best

Final Testing : Combining Node Classification and Link Prediction



Final Testing



- The prediction results of the node classification model are used for the link prediction
- Three link prediction models to be tested and compared

Obtained by performing absolution difference of the labels of two nodes

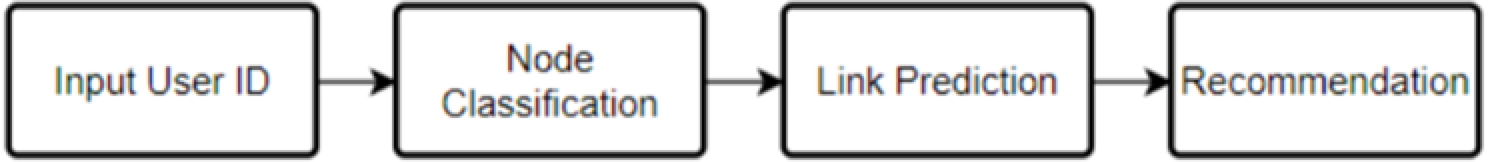
Final Testing

Model	Accuracy	Precision
Final Model (6 Indices for Edges + Original node features + Node label output from node classification)	**89.10%**	**89.12%**
Model Y (6 Indices for Edges + Original node features)	88.73%	88.79%
Model X (6 Indices for Edges + Node embedding by random walk + Node label output from node classification)	77.72%	81.69%

Final Model: the best since all features are used and the edge embeddings are obtained from original node features provided

Conclusion

- GitHub network follows Power Law and exhibits small world phenomenon
- Built recommendation system by training Graph Neural Networks for node classification and MLP for link prediction
- 86.15% node classification test accuracy
- Incorporate node classification results into link prediction task
- 89.10% link prediction test accuracy for the final testing



Thank You