Missing Link Prediction in Ecological Environment

Project report in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering (Artificial Intelligence and Machine Learning)

Submitted By

Arijit Das	Enrollment No. 12021002028129
Ankan Paul	Enrollment No. 12021002028098
Trina Chowdhury	Enrollment No. 12021002028100
Madhushree Pramanik	Enrollment No. 12021002028101
Spandan Moyra	Enrollment No. 12021002028001

Under the guidance of

Prof. Sramana Mukherjee

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning)

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700160.

CERTIFICATE

This is to certify that the project titled **“Missing Link Prediction in Ecological Environment”** submitted by **Arijit Das (University Roll No. 12021002028129), Ankan Paul (University Roll No. 12021002028098)**, **Trina Chowdhury (University Roll No. 12021002028100), Spandan Moyra (University Roll No. 12021002028001)**, **Madhushree Pramanik(University Roll No. 12021002028101)** students of University Of Engineering And Management, Kolkata, in partial fulfillment of requirement for the degree of Bachelor of Computer Science and Engineering (Artificial Intelligence and Machine Learning), is a bonafide work carried out by them under the supervision and guidance of Prof. Sramana Mukherjee during 6th Semester of academic session of 2023 - 2024. The content of this report has not been submitted to any other university or institute. I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

____

Signature of Guide

____________________________________ ____________________________________

Signature of Head of the Department
Department of CSE(AI & ML)

Signature of External Guide

ACKNOWLEDGEMENT

We would like to express our profound gratitude to Prof. Sramana Mukherjee, Project Mentor, Prof. (Dr.) Sudipta Sahana, Head of the Department of CSE (AI & ML), and Prof. (Dr.) Sajal Dasgupta, Vice Chancellor, University of Engineering And Management, Kolkata for their contributions to the completion of my project titled **"Missing Link Prediction in Ecological Environment"**. I would like to express my special thanks to our project mentor Prof. Sramana Mukherjee for his/her time and efforts he/she provided throughout the year. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you. I would like to acknowledge that this project was completed entirely by me and not by someone else. Last but not least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Arijit Das
Ankan Paul
Trina Chowdhury
Madhushree Pramanik
Spandan Moyra

TABLE OF CONTENTS

ABSTRACT

Predicting missing links in ecological networks is a critical task with far-reaching implications for ecosystem understanding and conservation. This research delves into the application of machine learning techniques to address this challenge within ecological environments. By leveraging ML algorithms, we aim to develop robust models capable of predicting unobserved interactions among species or ecological entities. The project focuses on optimizing ML methods to handle the complexity and sparsity of ecological data, incorporating network topology, node attributes, and ecological principles. We demonstrate the effectiveness of our framework using real-world ecological datasets, showcasing its ability to accurately predict missing links across different types of ecological networks. The outcomes of this research are expected to significantly enhance network reconstruction, improve ecological modelling accuracy, and contribute to better-informed conservation decisions. This interdisciplinary approach at the intersection of ecology and machine learning promises to advance our understanding of ecosystem dynamics and resilience in the face of environmental changes.

CHAPTER – 1: INTRODUCTION

Ecosystems are intricate networks of interacting species and their physical environments. Understanding the relationships and interdependencies within these ecological networks is crucial for preserving biodiversity, maintaining ecosystem services, and mitigating environmental impacts. However, our knowledge of ecological interactions is often incomplete due to the challenges of directly observing and quantifying all the connections within a complex ecosystem. This lack of complete information represents a significant obstacle in developing accurate ecological models and making informed conservation decisions.

Missing link prediction aims to address this problem by leveraging existing knowledge about ecological networks to infer potential missing interactions or connections. By utilizing computational techniques and data-driven approaches, missing link prediction methods can provide insights into the unobserved or undocumented relationships between species, habitats, and environmental factors. This predictive capability holds immense value for uncovering hidden ecological patterns, identifying potential species introductions or invasions, and anticipating the cascading effects of environmental changes.

In the context of ecological environments, missing link prediction can be applied to various types of networks, such as food webs, mutualistic networks (e.g., plant-pollinator interactions), and habitat connectivity networks. By accurately predicting missing links, researchers and conservationists can gain a more comprehensive understanding of ecosystem dynamics, enabling them to develop more effective management strategies, prioritize conservation efforts, and assess the potential impacts of human activities or environmental disturbances.

This introduction sets the stage for discussing the importance of missing link prediction in ecological contexts, highlighting its potential to fill knowledge gaps and support informed decision-making for environmental conservation and management.

CHAPTER – 2: LITERATURE SURVEY

1. **Link Prediction Under Imperfect Detection: Collaborative Filtering for Ecological Networks (-** Xiao Fu, Eugene Seo, Justin Clarke, and Rebecca A. Hutchinson**): -**

This paper proposes a computational framework for link prediction in ecological networks (e.g., plant-pollinator, host-parasite) under imperfect detection. The key challenges are:

a. Observations of ecological interactions are subject to systematic undercounting/missed detections due to factors like limited sampling time and conditions.
b. Traditional collaborative filtering methods for link prediction assume observed data is noise-free, violating the imperfect detection in ecological data.

The main contributions are:

a. A statistical model combining a Poisson N-mixture model for the true interaction counts and a binomial-linear regression model for the imperfect detection process.
b. An effective optimization algorithm based on block coordinate descent to estimate the model parameters.
c. Evaluations on synthetic data and real-world ecological networks (plant-pollinator, host-parasite) for link prediction, showing the proposed method outperforms baselines.

The work addresses a key issue in ecological data (imperfect detection) and develops a link prediction framework tailored for such data, bridging statistical ecology and matrix factorization techniques.

2. **Ecological network metrics: opportunities for synthesis** (- MATTHEW K. LAU, STUART R. BORRETT, BENJAMIN BAISER, NICHOLAS J. GOTELLI, AND AARON M. ELLISON)

3. **Joining the dots: An automated method for constructing food webs from compendia of published interactions** (- Clare Gray, David H. Figueroa, Lawrence N. Hudson, Athen Mae, Dan Perkins, Guy Woodward)

CHAPTER – 3: PROBLEM STATEMENT

Problem Statement:

Expanding on the refined problem statement for the missing link prediction in ecological networks, we can look into objectives, and methodologies that underscore this area. This expanded explanation aims to provide a more detailed roadmap for addressing the problem, highlighting its complexities and the multifaceted approach needed to tackle it.

Some Notable Challenges:

1. **Incomplete Data and Network Fragmentation:**
   - Ecological networks are often constructed from fragmented and incomplete datasets, leading to a partial view of the ecosystem's interaction web. This limitation obscures our understanding of ecological dynamics and impedes accurate modeling and prediction efforts.
2. **Complexity of Ecological Interactions:**
   - The interactions within ecological networks span various types, including predation, competition, mutualism, and more. Each type has its own set of rules and dynamics, adding layers of complexity to the prediction of missing links.
3. **Temporal and Spatial Variability:**
   - Ecological networks are not static; they evolve over time and vary spatially. This dynamic nature requires models that can adapt to and predict changes in network structure due to seasonal variations, migration patterns, and environmental changes.
4. **High Dimensionality and Scalability:**
   - Ecological datasets can be vast and high-dimensional, challenging computational models' ability to scale effectively and maintain high accuracy in missing link prediction.

Objectives and Methodological Approaches:

1. **Data Integration and Standardization:**
   - Combining diverse data sources, including observational data, experimental data, and environmental variables, and standardizing these datasets to form a cohesive foundation for modeling ecological networks.
2. **Model Development and Validation:**
   - Developing advanced computational models, such as machine learning algorithms and network analysis tools, tailored to the specific characteristics of ecological networks. This includes leveraging techniques for handling sparse

data, modeling non-linear interactions, and validating predictions through empirical data or expert knowledge.

3. **Dynamic Modeling for Temporal and Spatial Analysis:**
   - Creating models capable of incorporating temporal and spatial variability to predict how networks might change over time or across different geographic landscapes, thereby understanding migration patterns, seasonal changes in interactions, and the impact of habitat alterations.
4. **Scalability and High-Dimensional Data Analysis:**
   - Employing and refining algorithms that can effectively process and analyze high-dimensional data, ensuring that models are both accurate and scalable across large ecological networks.

Expected Outcomes and Impacts:

1. **Enhanced Ecological Network Reconstructions:**
   - By accurately predicting missing links, the reconstructed networks will more closely mirror real-world ecosystems, facilitating improved ecological modeling, biodiversity assessments, and conservation planning.
2. **Insights into Ecosystem Dynamics and Functioning:**
   - Improved understanding of the underlying mechanisms driving ecosystem stability, resilience, and functionality, enabling better predictions of how ecosystems respond to environmental changes.
3. **Advancements in Computational Ecology:**
   - The development and refinement of computational methods for ecological research will not only aid in missing link prediction but also contribute to the broader field of computational ecology, enhancing data analysis techniques and modeling approaches.
4. **Informing Conservation and Management Strategies:**
   - With a more complete understanding of ecological networks, conservationists and environmental managers can make more informed decisions regarding species protection, habitat restoration, and biodiversity conservation, potentially mitigating the impacts of human activities and climate change.

This expanded explanation underscores the interdisciplinary nature of missing link prediction in ecological networks, requiring a blend of ecological theory, computational science, and data analysis to address the complexities of real-world ecosystems.

CHAPTER – 4: PROPOSED SOLUTION

1. Data Preprocessing and Network Representation

The first step involves collecting and preprocessing the available ecological data, which may include species interactions, environmental factors, and attributes related to the network entities (e.g., species characteristics, habitat information). This data can be stored in a structured format, such as CSV files.

Once the data is pre-processed, we can represent the ecological network as a directed graph (DI graph) using Python libraries like NetworkX. In this graph representation, nodes represent entities (e.g., species, habitats), and edges represent known interactions or relationships between them.

2. Centrality Measures

To gain insights into the network structure and identify potentially important nodes, we can calculate various centrality measures. These measures quantify the importance or influence of a node within the network based on its connectivity patterns. Some commonly used centrality measures include:

- Degree Centrality: Measures the number of connections a node has.
- Betweenness Centrality: Quantifies the number of shortest paths that pass through a node.
- Closeness Centrality: Measures the average distance of a node from all other nodes in the network.
- Eigenvector Centrality: Considers not only the number of connections but also the importance of the connected nodes.

These centrality measures can be computed using the NetworkX libraries in Python.

3. Network Visualization and Heat Map

To visualize the ecological network and gain insights into its structure, we can use graph visualization tools like Matplotlib or Plotly in Python. These tools allow us to create interactive graphs where nodes and edges can be colored or sized based on their attributes or centrality measures.

Additionally, we can create heat maps to visualize the relationships or interactions between different entities in the network. Heat maps can help identify clusters or groups of strongly connected nodes, which may indicate potential missing links within those groups.

4. Bipartite Graph and Clustering

Ecological networks often involve interactions between different types of entities, such as species and habitats, or plants and pollinators. In such cases, we can represent the network as a bipartite graph, where nodes are divided into two disjoint sets, and edges only connect nodes from different sets.

For clustering within the bipartite graph, we can employ techniques like:

- Spectral Clustering: This method partitions the graph based on the eigenvectors of the Laplacian matrix, which captures the connectivity patterns in the network.
- Modularity-based Clustering: This approach aims to maximize the modularity score, which measures the density of edges within clusters compared to edges between clusters.

These clustering techniques can help identify potential missing links within or between clusters, as entities within the same cluster are more likely to interact or share connections.

5. Bi-Clustering

In addition to clustering the nodes, we can also perform bi-clustering, which simultaneously clusters the rows and columns of the data matrix (e.g., species-by-habitat matrix). This approach can reveal patterns and potential missing links based on the co-occurrences or interactions between different entities.

One possible bi-clustering algorithm that can be used is the Spectral Co-Clustering algorithm, which leverages the spectral properties of the data matrix to identify coherent clusters.

6. Link Prediction Algorithms

After extracting relevant features from the network and ecological data (e.g., topological properties, node attributes, domain-specific knowledge), we can employ various link prediction algorithms and machine learning models to predict missing links.

Some potential approaches include:

- Neighbourhood-based algorithms (e.g., Common Neighbours, Adamic/Adar, Resource Allocation)
- Path-based algorithms (e.g., Katz, Rooted PageRank)
- Probabilistic models (e.g., Stochastic Block Models, Hierarchical Random Graphs)
- Machine learning techniques (e.g., Random Forests, Gradient Boosting, Neural Networks)

These algorithms can be implemented using Python libraries like NetworkX, scikit-learn, or dedicated network analysis libraries.

7. Evaluation and Validation

To evaluate the performance of the link prediction algorithms, we can use appropriate evaluation metrics, such as precision, recall, F1-score, and area under the ROC curve (AUC-

ROC). Cross-validation techniques can be employed to ensure robust and unbiased performance estimates.

Additionally, it is crucial to validate the predicted missing links against known interactions or empirical data, as well as incorporate domain expertise and ecological principles to refine the predictions.

8. Implementation in Python and Google Colab

The entire solution pipeline, including data preprocessing, network representation, centrality calculations, clustering, link prediction algorithms, and evaluation, can be implemented in Python. Google Colab, a cloud-based Jupyter notebook environment, can be leveraged to execute the code and perform computations seamlessly.

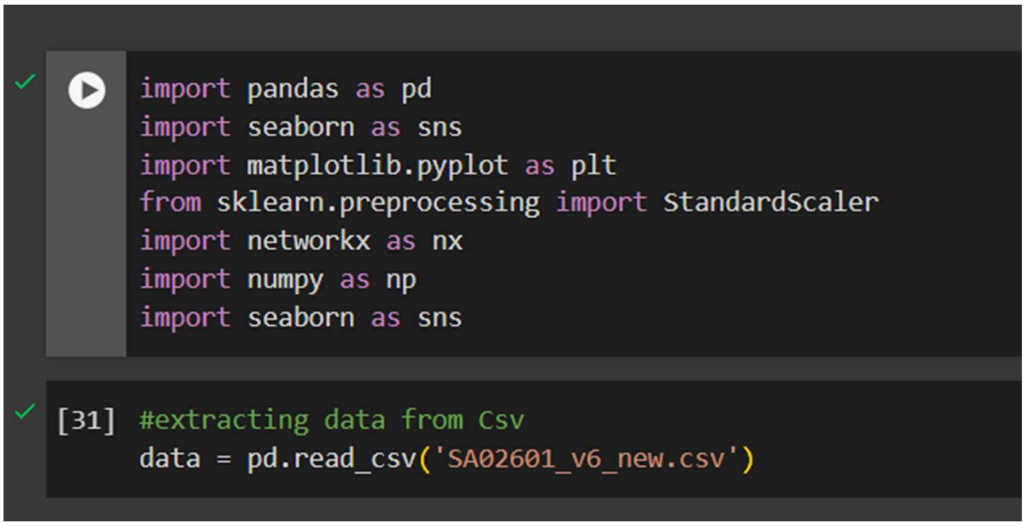
Python libraries such as NetworkX, scikit-learn, Matplotlib, Seaborn, and Pandas can be utilized for various tasks, including graph operations, machine learning, data manipulation, and visualization.

This proposed solution report outlines a comprehensive approach to predicting missing links in ecological networks, leveraging various techniques and methodologies. By combining graph theory, machine learning, data visualization, and ecological domain knowledge, this solution aims to enhance our understanding of ecological networks, support informed conservation decisions, and contribute to the advancement of computational ecology.

CHAPTER – 5: EXPERIMENTAL SETUP AND RESULT ANALYSIS

In this project we are using Google collab and python as our programming language.

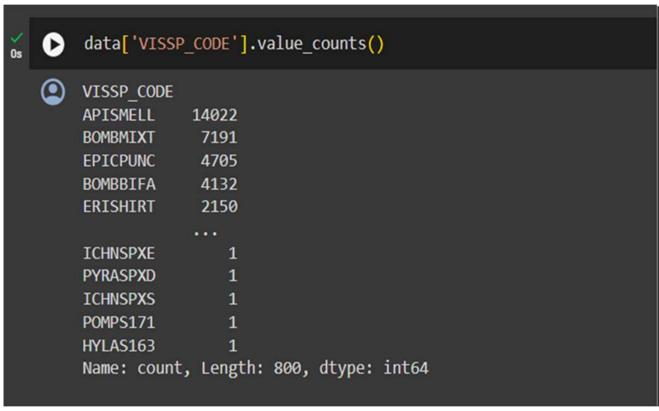
At first libraries are imported, data is extracted from the csv dataset.



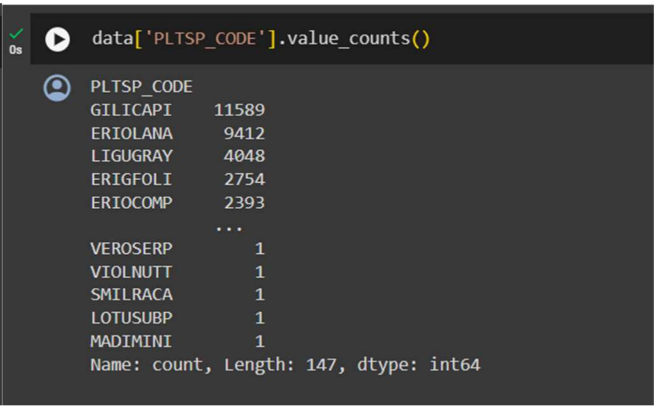
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import networkx as nx
import numpy as np
import seaborn as sns
```

```
[31] #extracting data from Csv
     data = pd.read_csv('SA02601_v6_new.csv')
```

The two columns we have analysed are ‘PLTSP_CODE’ and ‘VISSP_CODE’ which defines Plant species and Visitor species(pollinator).

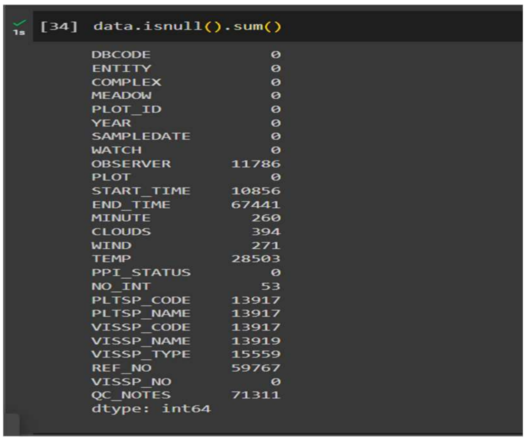


```
data['VISSP_CODE'].value_counts()

VISSP_CODE
APISMELL    14022
BOMBMIXT     7191
EPICPUNC     4705
BOMBBIFA     4132
ERISHIRT     2150
            ...
ICHNSPXE        1
PYRASPXD        1
ICHNSPXS        1
POMPS171        1
HYLAS163        1
Name: count, Length: 800, dtype: int64
```



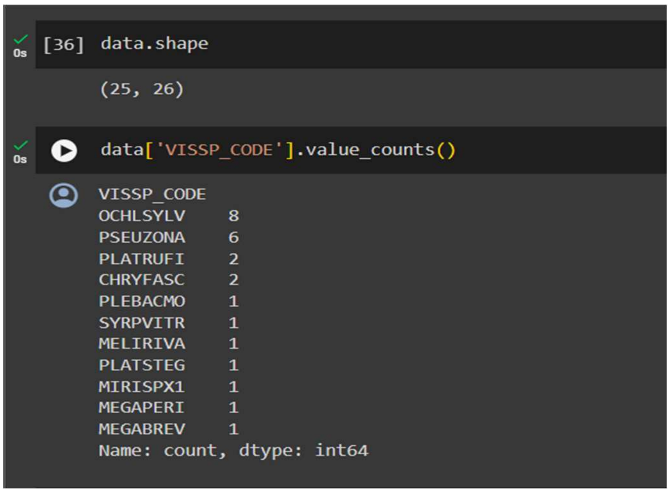
```
data['PLTSP_CODE'].value_counts()

PLTSP_CODE
GILICAPI    11589
ERIOLANA     9412
LIGUGRAY     4048
ERIGFOLI     2754
ERIOCOMP     2393
            ...
VEROSERP        1
VIOLNUTT        1
SMILRACA        1
LOTUSUBP        1
MADIMINI        1
Name: count, Length: 147, dtype: int64
```

Then we checked for the number of null values in a DataFrame called data. The isnull() method is used to check for null values in the DataFrame and the sum() method is used to count the number of null values in each column.

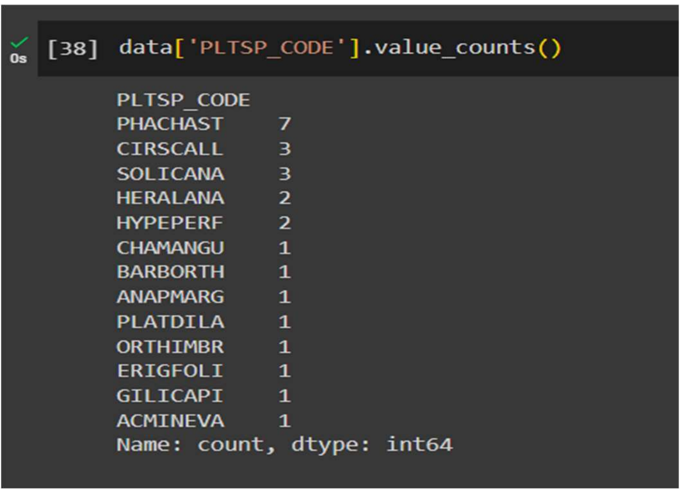


```
[34] data.isnull().sum()

DBCODE              0
ENTITY              0
COMPLEX             0
MEADOW              0
PLOT_ID             0
YEAR                0
SAMPLEDATE          0
WATCH               0
OBSERVER        11786
PLOT                0
START_TIME      10856
END_TIME        67441
MINUTE            260
CLOUDS            394
WIND              271
TEMP            28503
PPI_STATUS          0
NO_INT             53
PLTSP_CODE      13917
PLTSP_NAME      13917
VISSP_CODE      13917
VISSP_NAME      13919
VISSP_TYPE      15559
REF_NO          59767
VISSP_NO            0
QC_NOTES        71311
dtype: int64
```

After removing null values from the dataset we again checked the values of two columns.

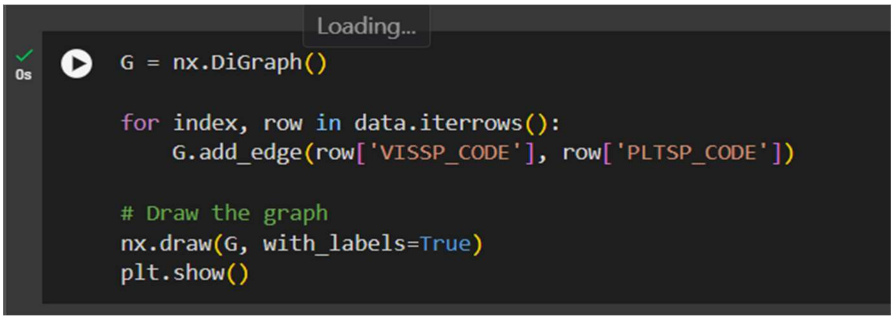


```
[36] data.shape

     (25, 26)

     data['VISSP_CODE'].value_counts()

     VISSP_CODE
     OCHLSYLV    8
     PSEUZONA    6
     PLATRUFI    2
     CHRYFASC    2
     PLEBACMO    1
     SYRPVITR    1
     MELIRIVA    1
     PLATSTEG    1
     MIRISPX1    1
     MEGAPERI    1
     MEGABREV    1
     Name: count, dtype: int64
```

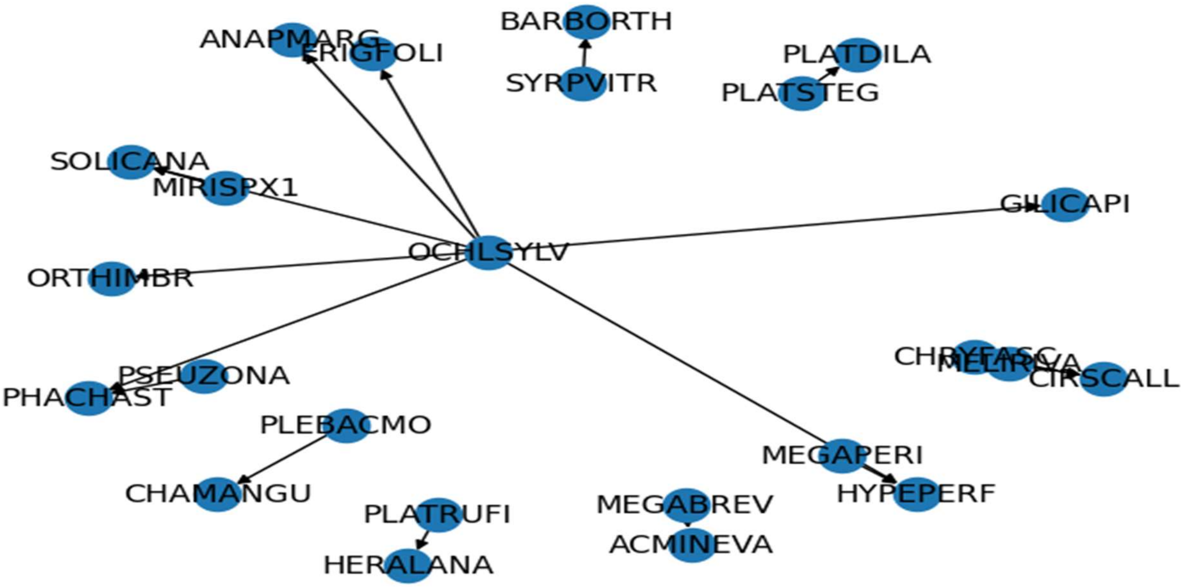


```
[38] data['PLTSP_CODE'].value_counts()

     PLTSP_CODE
     PHACHAST    7
     CIRSCALL    3
     SOLICANA    3
     HERALANA    2
     HYPEPERF    2
     CHAMANGU    1
     BARBORTH    1
     ANAPMARG    1
     PLATDILA    1
     ORTHIMBR    1
     ERIGFOLI    1
     GILICAPI    1
     ACMINEVA    1
     Name: count, dtype: int64
```

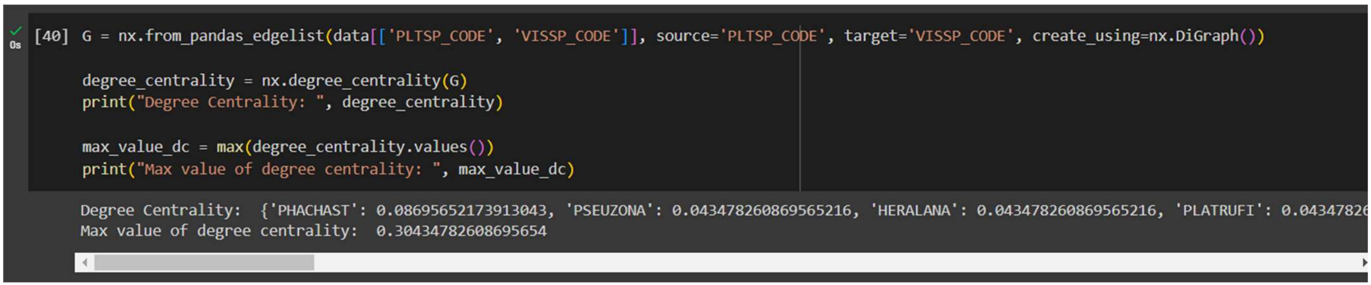
Next the plotting of DAG graph is needed to check the interactions happened between the species.



```
G = nx.DiGraph()

for index, row in data.iterrows():
    G.add_edge(row['VISSP_CODE'], row['PLTSP_CODE'])

# Draw the graph
nx.draw(G, with_labels=True)
plt.show()
```

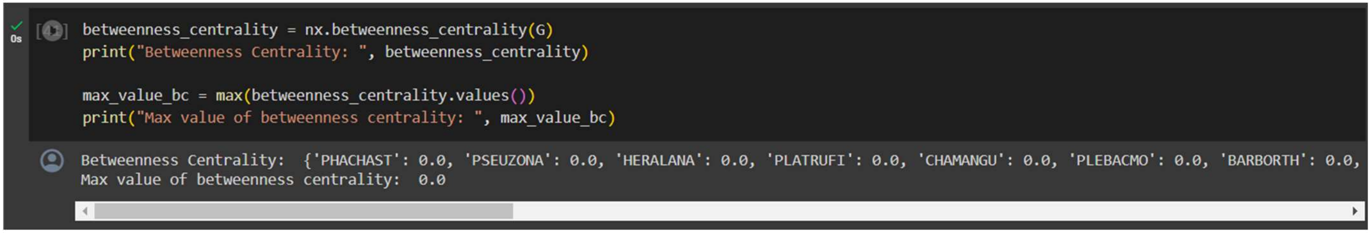


Then we measure the different centralities to find things like most interactive species , least interactive species and keystone species etc. After measuring the different centralities we got a brief solution of two species.



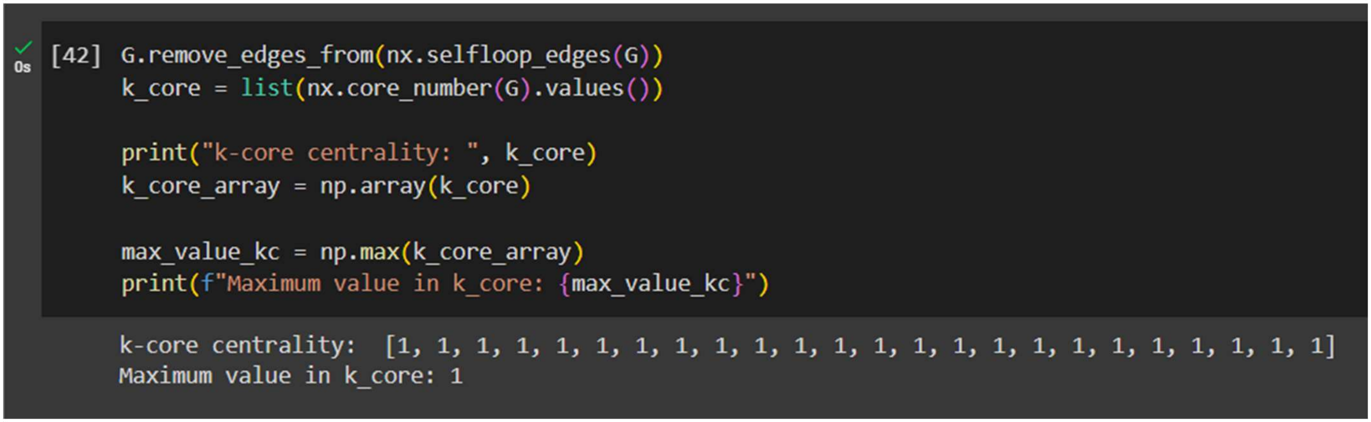
```
[40] G = nx.from_pandas_edgelist(data[['PLTSP_CODE', 'VISSP_CODE']], source='PLTSP_CODE', target='VISSP_CODE', create_using=nx.DiGraph())

     degree_centrality = nx.degree_centrality(G)
     print("Degree Centrality: ", degree_centrality)

     max_value_dc = max(degree_centrality.values())
     print("Max value of degree centrality: ", max_value_dc)
```

```
Degree Centrality:  {'PHACHAST': 0.08695652173913043, 'PSEUZONA': 0.043478260869565216, 'HERALANA': 0.043478260869565216, 'PLATRUFI': 0.0434782б
Max value of degree centrality:  0.30434782608695654
```



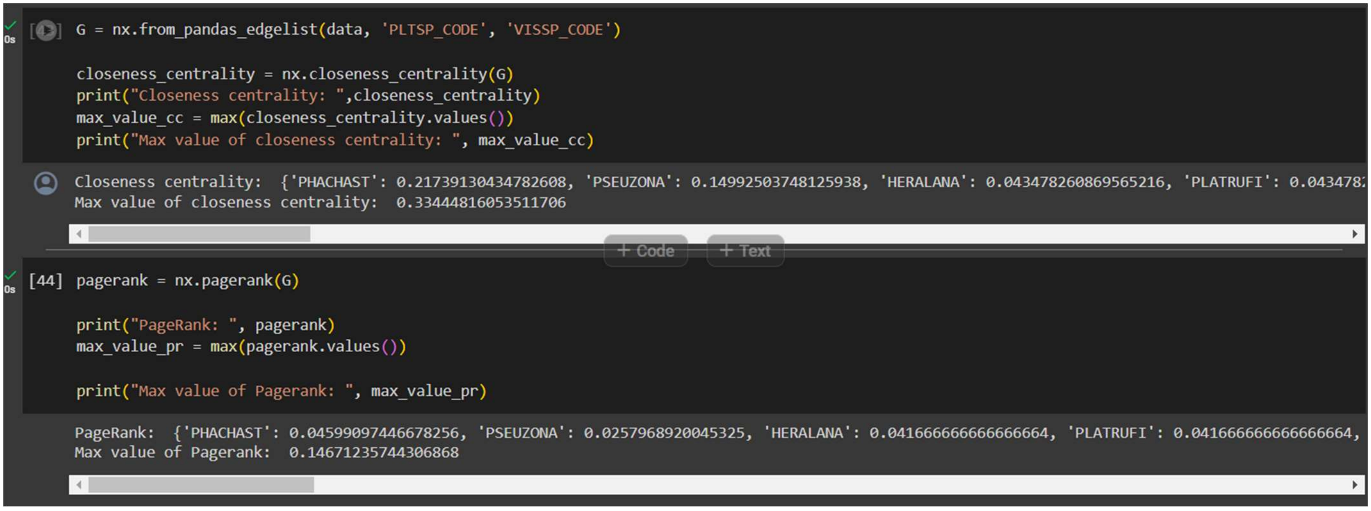
```
betweenness_centrality = nx.betweenness_centrality(G)
print("Betweenness Centrality: ", betweenness_centrality)

max_value_bc = max(betweenness_centrality.values())
print("Max value of betweenness centrality: ", max_value_bc)
```

```
Betweenness Centrality:  {'PHACHAST': 0.0, 'PSEUZONA': 0.0, 'HERALANA': 0.0, 'PLATRUFI': 0.0, 'CHAMANGU': 0.0, 'PLEBACMO': 0.0, 'BARBORTH': 0.0,
Max value of betweenness centrality:  0.0
```



```
[42] G.remove_edges_from(nx.selfloop_edges(G))
     k_core = list(nx.core_number(G).values())

     print("k-core centrality: ", k_core)
     k_core_array = np.array(k_core)

     max_value_kc = np.max(k_core_array)
     print(f"Maximum value in k_core: {max_value_kc}")
```

```
k-core centrality:  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Maximum value in k_core: 1
```



```
G = nx.from_pandas_edgelist(data, 'PLTSP_CODE', 'VISSP_CODE')

closeness_centrality = nx.closeness_centrality(G)
print("Closeness centrality: ",closeness_centrality)
max_value_cc = max(closeness_centrality.values())
print("Max value of closeness centrality: ", max_value_cc)
```

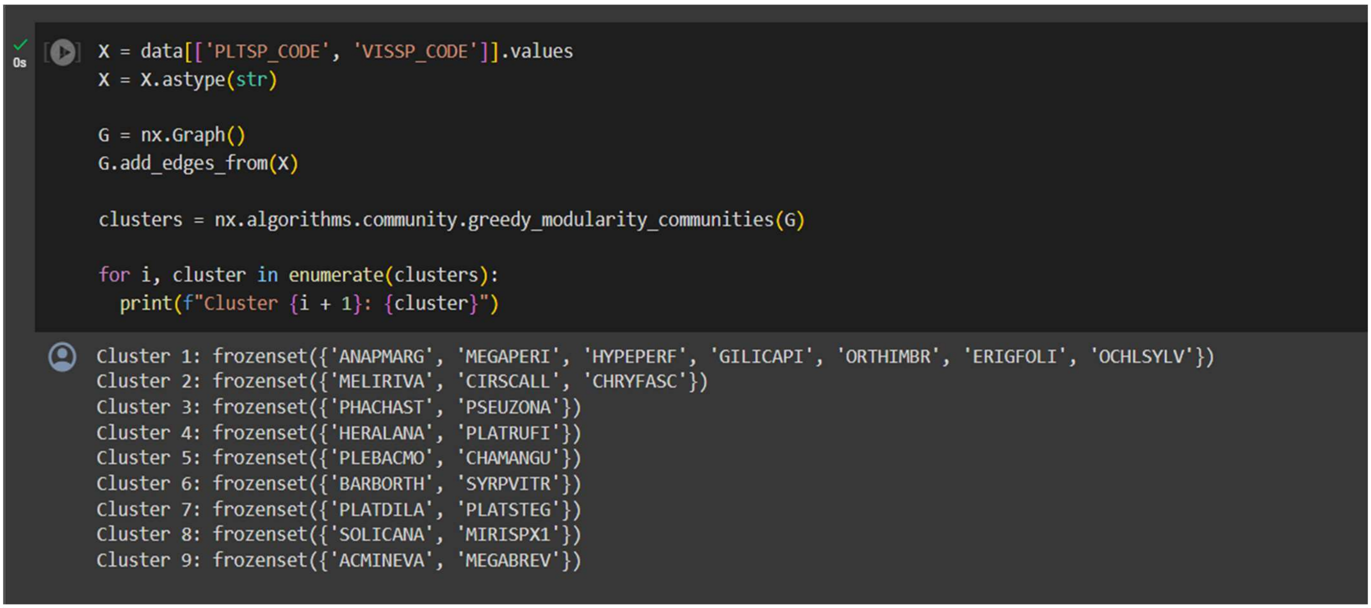
```
Closeness centrality:  {'PHACHAST': 0.21739130434782608, 'PSEUZONA': 0.14992503748125938, 'HERALANA': 0.043478260869565216, 'PLATRUFI': 0.043478
Max value of closeness centrality:  0.33444816053511706
```

```
[44] pagerank = nx.pagerank(G)

     print("PageRank: ", pagerank)
     max_value_pr = max(pagerank.values())

     print("Max value of Pagerank: ", max_value_pr)
```

```
PageRank:  {'PHACHAST': 0.04599097446678256, 'PSEUZONA': 0.0257968920045325, 'HERALANA': 0.041666666666666664, 'PLATRUFI': 0.041666666666666664,
Max value of Pagerank:  0.14671235744306868
```

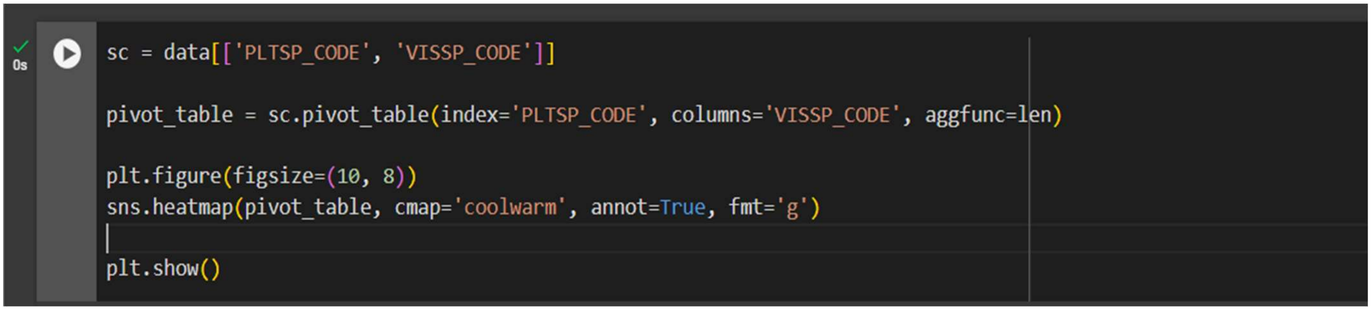
Then we extract two columns and try to get the informations and also applied clustering upon the two columns.we got the solution but it was not too evident.

We got 9 clusters from the clustering operation and tried to plot a heatmap for better understanding . The heatmap is shown below.

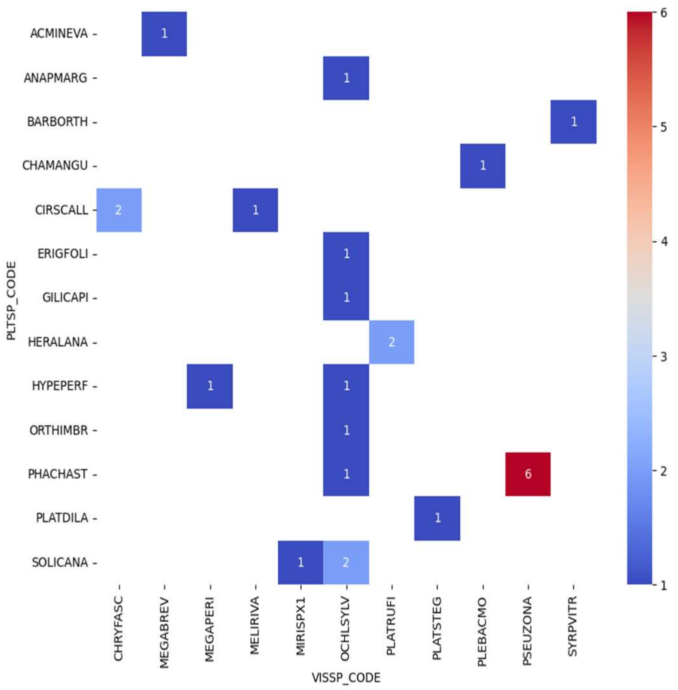


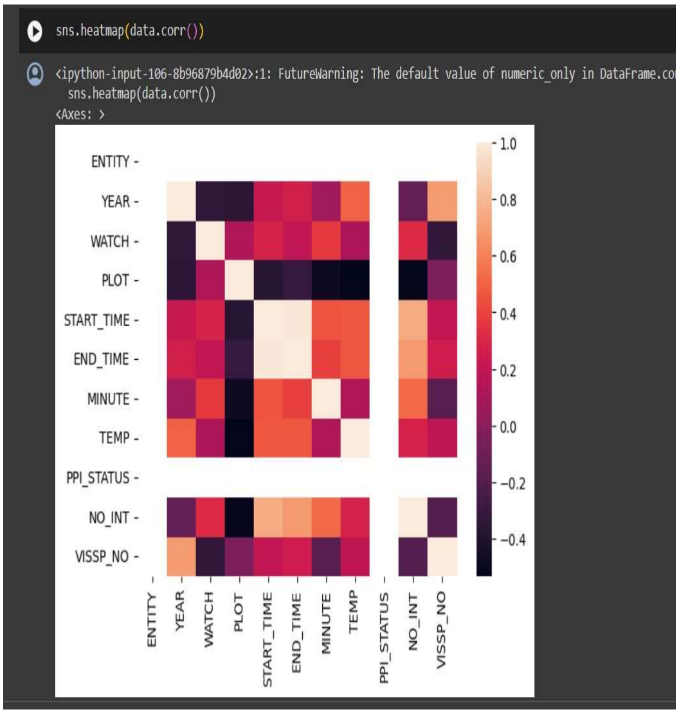
```
X = data[['PLTSP_CODE', 'VISSP_CODE']].values
X = X.astype(str)

G = nx.Graph()
G.add_edges_from(X)

clusters = nx.algorithms.community.greedy_modularity_communities(G)

for i, cluster in enumerate(clusters):
  print(f"Cluster {i + 1}: {cluster}")
```

```
Cluster 1: frozenset({'ANAPMARG', 'MEGAPERI', 'HYPEPERF', 'GILICAPI', 'ORTHIMBR', 'ERIGFOLI', 'OCHLSYLV'})
Cluster 2: frozenset({'MELIRIVA', 'CIRSCALL', 'CHRYFASC'})
Cluster 3: frozenset({'PHACHAST', 'PSEUZONA'})
Cluster 4: frozenset({'HERALANA', 'PLATRUFI'})
Cluster 5: frozenset({'PLEBACMO', 'CHAMANGU'})
Cluster 6: frozenset({'BARBORTH', 'SYRPVITR'})
Cluster 7: frozenset({'PLATDILA', 'PLATSTEG'})
Cluster 8: frozenset({'SOLICANA', 'MIRISPX1'})
Cluster 9: frozenset({'ACMINEVA', 'MEGABREV'})
```



```
sc = data[['PLTSP_CODE', 'VISSP_CODE']]

pivot_table = sc.pivot_table(index='PLTSP_CODE', columns='VISSP_CODE', aggfunc=len)

plt.figure(figsize=(10, 8))
sns.heatmap(pivot_table, cmap='coolwarm', annot=True, fmt='g')

plt.show()
```

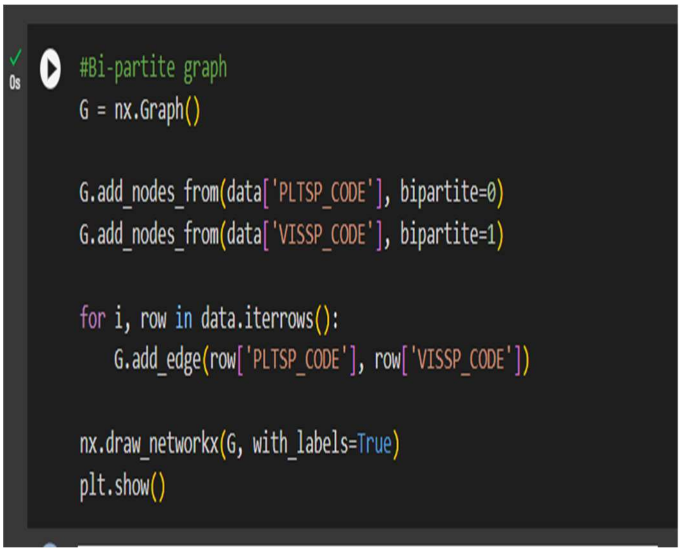




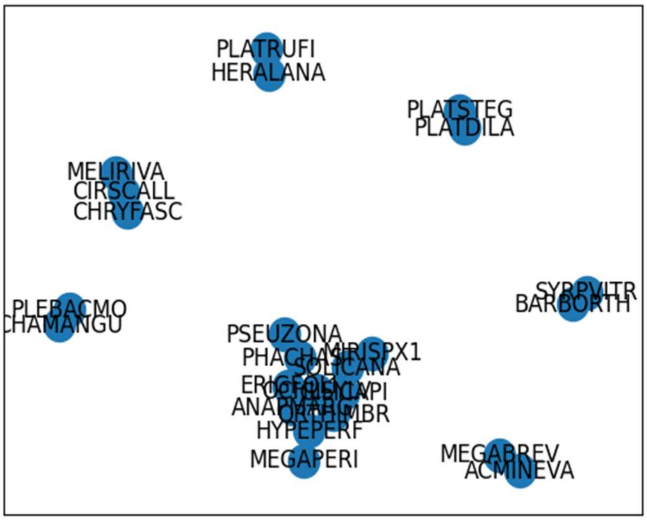
```
sns.heatmap(data.corr())
```

```
<ipython-input-106-8b96879b4d02>:1: FutureWarning: The default value of numeric_only in DataFrame.co
  sns.heatmap(data.corr())
<Axes: >
```

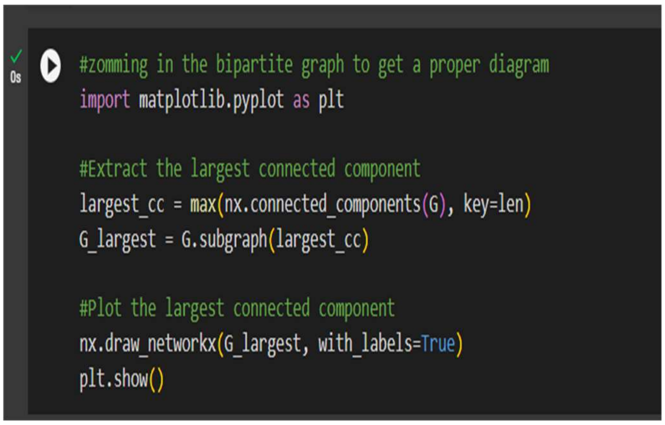
Then we plotted a Bipartite graph for this two columns. Bipartite graphs are useful for analysing and visualizing relationships between two distinct sets of entities. The graph is shown below.



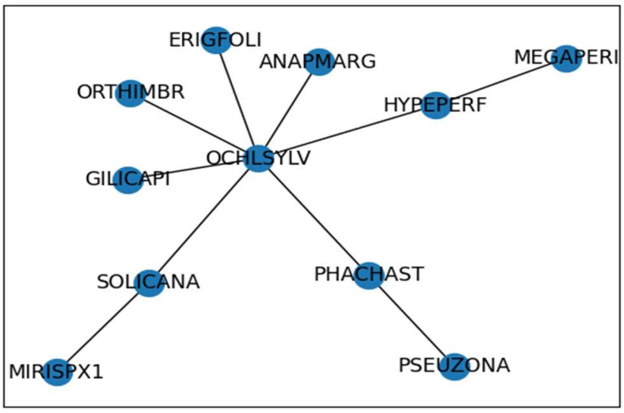
```
#Bi-partite graph
G = nx.Graph()

G.add_nodes_from(data['PLTSP_CODE'], bipartite=0)
G.add_nodes_from(data['VISSP_CODE'], bipartite=1)

for i, row in data.iterrows():
    G.add_edge(row['PLTSP_CODE'], row['VISSP_CODE'])

nx.draw_networkx(G, with_labels=True)
plt.show()
```



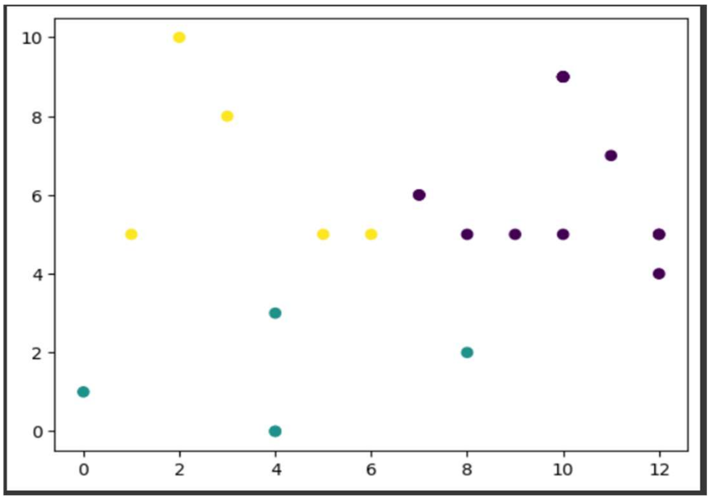
After plotting Bipartite graph we got the solution but the graph is not accurate to work on so we did Zooming on this graph for better accessiblity.The graph is shown below

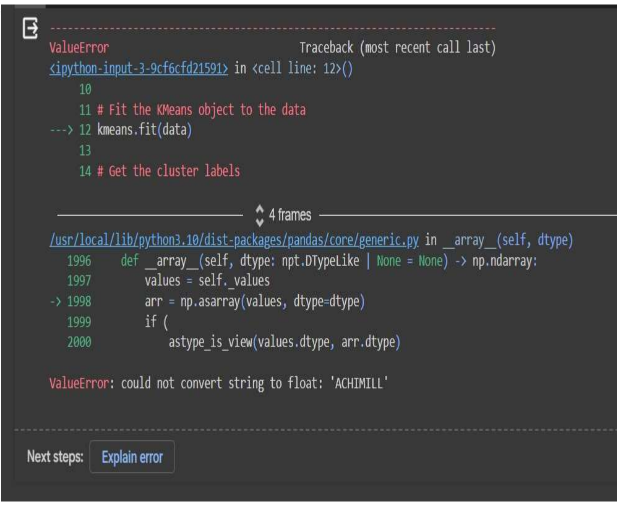


```
#zomming in the bipartite graph to get a proper diagram
import matplotlib.pyplot as plt

#Extract the largest connected component
largest_cc = max(nx.connected_components(G), key=len)
G_largest = G.subgraph(largest_cc)

#Plot the largest connected component
nx.draw_networkx(G_largest, with_labels=True)
plt.show()
```



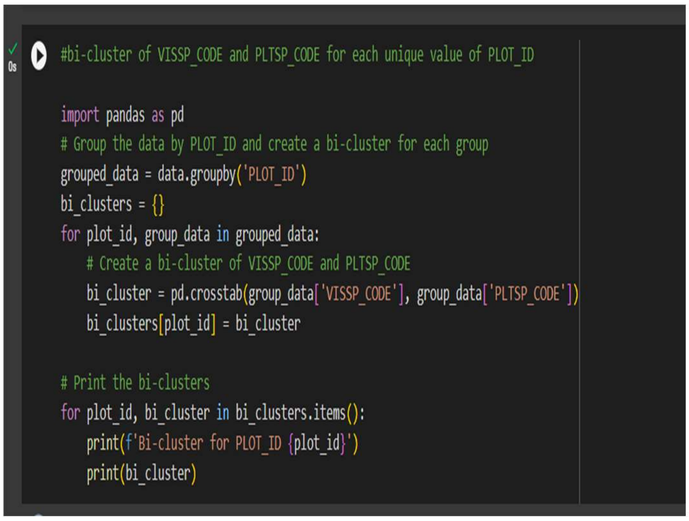
We wanted to apply K-means clustering on two columns but faced some error due to the datatypes. As the columns contains string as k- means clustering is only applicable for numerical values.



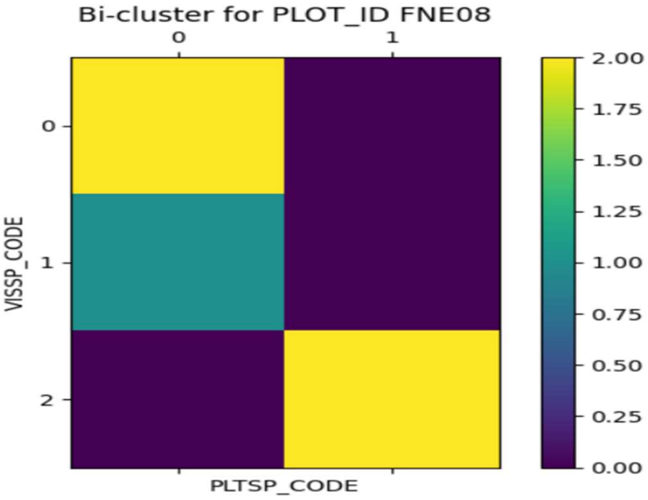


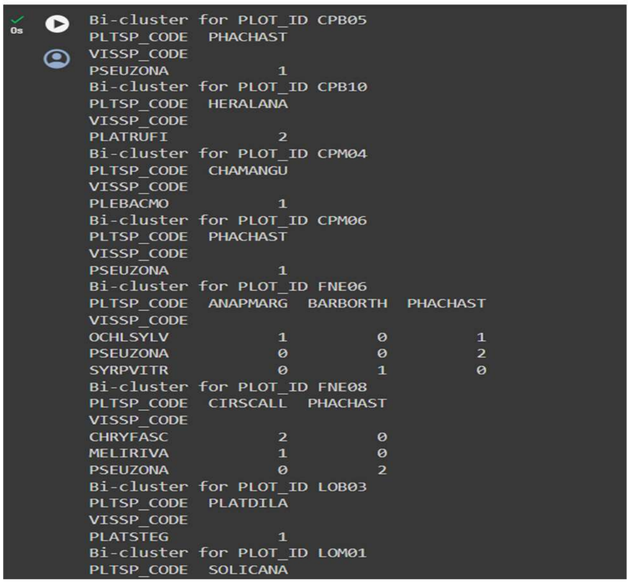
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-9cf6cfd21591> in <cell line: 12>()
     10
     11 # Fit the KMeans object to the data
---> 12 kmeans.fit(data)
     13
     14 # Get the cluster labels

4 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __array__(self, dtype)
   1996     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
   1997         values = self._values
-> 1998         arr = np.asarray(values, dtype=dtype)
   1999         if (
   2000             astype_is_view(values.dtype, arr.dtype)

ValueError: could not convert string to float: 'ACHIMILL'
```

So we tried to apply Bi-clustering on the two columns with the respect of PLOT_ID which are actually regions to work on.

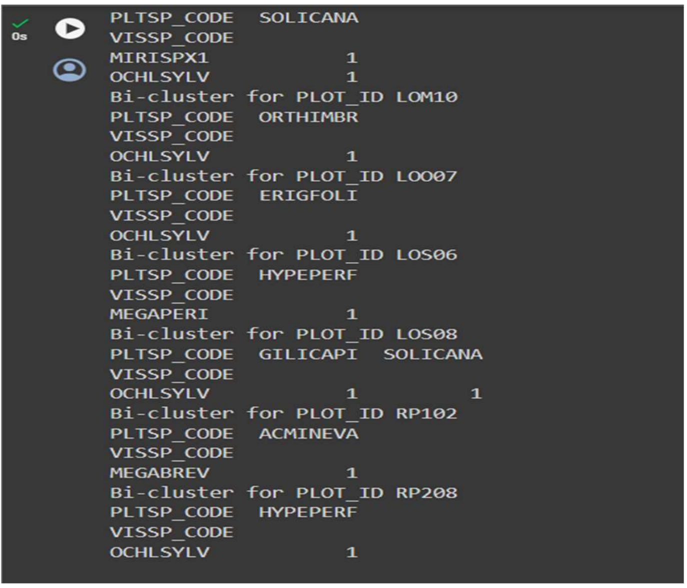


```
#bi-cluster of VISSP_CODE and PLTSP_CODE for each unique value of PLOT_ID

import pandas as pd
# Group the data by PLOT_ID and create a bi-cluster for each group
grouped_data = data.groupby('PLOT_ID')
bi_clusters = {}
for plot_id, group_data in grouped_data:
    # Create a bi-cluster of VISSP_CODE and PLTSP_CODE
    bi_cluster = pd.crosstab(group_data['VISSP_CODE'], group_data['PLTSP_CODE'])
    bi_clusters[plot_id] = bi_cluster

# Print the bi-clusters
for plot_id, bi_cluster in bi_clusters.items():
    print(f'Bi-cluster for PLOT_ID {plot_id}')
    print(bi_cluster)
```



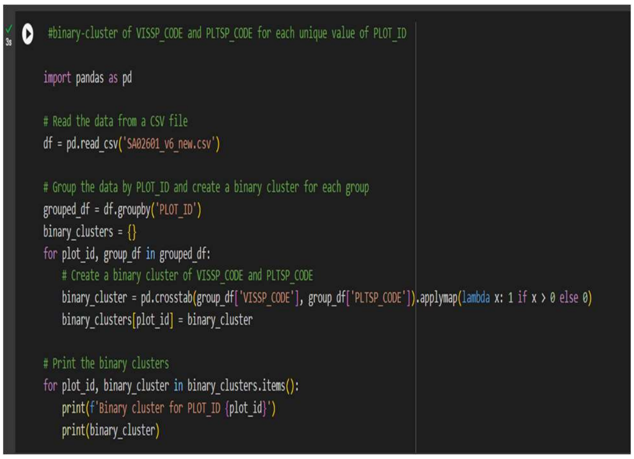


```
Bi-cluster for PLOT_ID CPB05
PLTSP_CODE  PHACHAST
VISSP_CODE
PSEUZONA           1
Bi-cluster for PLOT_ID CPB10
PLTSP_CODE  HERALANA
VISSP_CODE
PLATRUFI           2
Bi-cluster for PLOT_ID CPM04
PLTSP_CODE  CHAMANGU
VISSP_CODE
PLEBACMO           1
Bi-cluster for PLOT_ID CPM06
PLTSP_CODE  PHACHAST
VISSP_CODE
PSEUZONA           1
Bi-cluster for PLOT_ID FNE06
PLTSP_CODE  ANAPMARG  BARBORTH  PHACHAST
VISSP_CODE
OCHLSYLV           1         0         1
PSEUZONA           0         0         2
SYRPVITR           0         1         0
Bi-cluster for PLOT_ID FNE08
PLTSP_CODE  CIRSCALL  PHACHAST
VISSP_CODE
CHRYFASC           2         0
MELIRIVA           1         0
PSEUZONA           0         2
Bi-cluster for PLOT_ID LOB03
PLTSP_CODE  PLATDILA
VISSP_CODE
PLATSTEG           1
Bi-cluster for PLOT_ID LOM01
PLTSP_CODE  SOLICANA
```

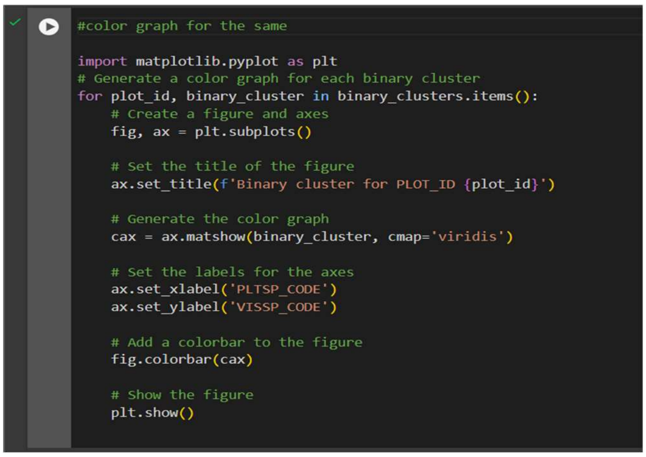


```
PLTSP_CODE  SOLICANA
VISSP_CODE
MIRISPX1           1
OCHLSYLV           1
Bi-cluster for PLOT_ID LOM10
PLTSP_CODE  ORTHIMBR
VISSP_CODE
OCHLSYLV           1
Bi-cluster for PLOT_ID LOO07
PLTSP_CODE  ERIGFOLI
VISSP_CODE
OCHLSYLV           1
Bi-cluster for PLOT_ID LOS06
PLTSP_CODE  HYPEPERF
VISSP_CODE
MEGAPERI           1
Bi-cluster for PLOT_ID LOS08
PLTSP_CODE  GILICAPI  SOLICANA
VISSP_CODE
OCHLSYLV           1         1
Bi-cluster for PLOT_ID RP102
PLTSP_CODE  ACMINEVA
VISSP_CODE
MEGABREV           1
Bi-cluster for PLOT_ID RP208
PLTSP_CODE  HYPEPERF
VISSP_CODE
OCHLSYLV           1
```

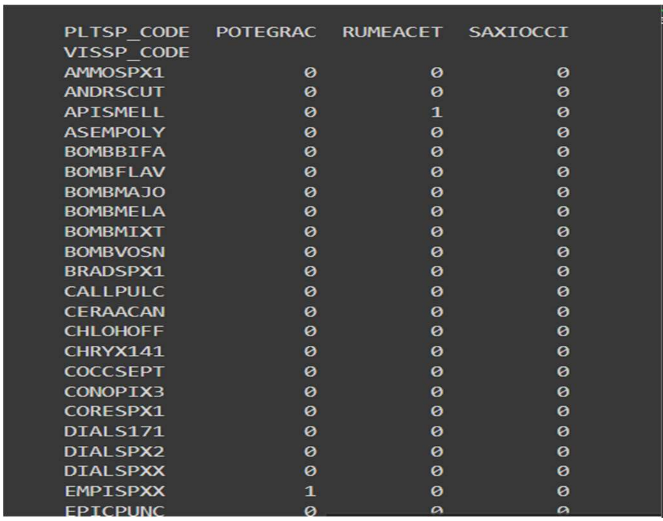
Binary clustering is a clustering technique that assigns data points to one of two clusters. Binary clustering simplifies the clustering process by dividing data points into only two clusters, making it easier to interpret and analyse the results. In some cases, binary clustering can be used to make binary decisions. So we also used this technique and plotted a graph for this two columns and based on Each region too. The colour graph is shown below.



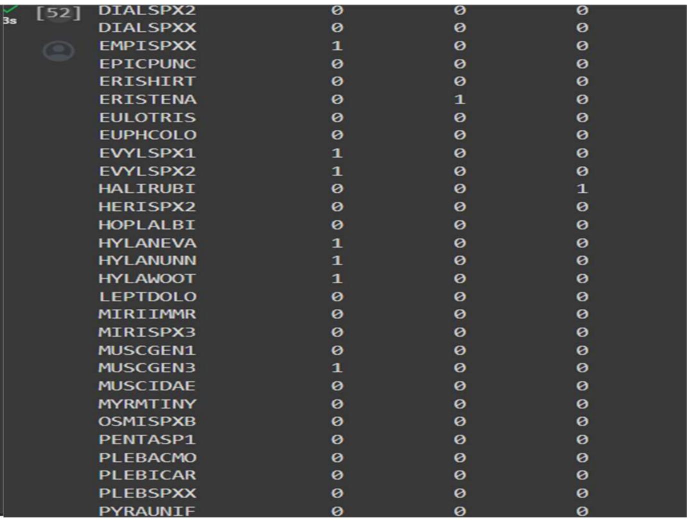
```
#binary-cluster of VISSP_CODE and PLTSP_CODE for each unique value of PLOT_ID

import pandas as pd

# Read the data from a CSV file
df = pd.read_csv('SA02601_v6_new.csv')

# Group the data by PLOT_ID and create a binary cluster for each group
grouped_df = df.groupby('PLOT_ID')
binary_clusters = {}
for plot_id, group_df in grouped_df:
    # Create a binary cluster of VISSP_CODE and PLTSP_CODE
    binary_cluster = pd.crosstab(group_df['VISSP_CODE'], group_df['PLTSP_CODE']).applymap(lambda x: 1 if x > 0 else 0)
    binary_clusters[plot_id] = binary_cluster

# Print the binary clusters
for plot_id, binary_cluster in binary_clusters.items():
    print(f'Binary cluster for PLOT_ID {plot_id}')
    print(binary_cluster)
```



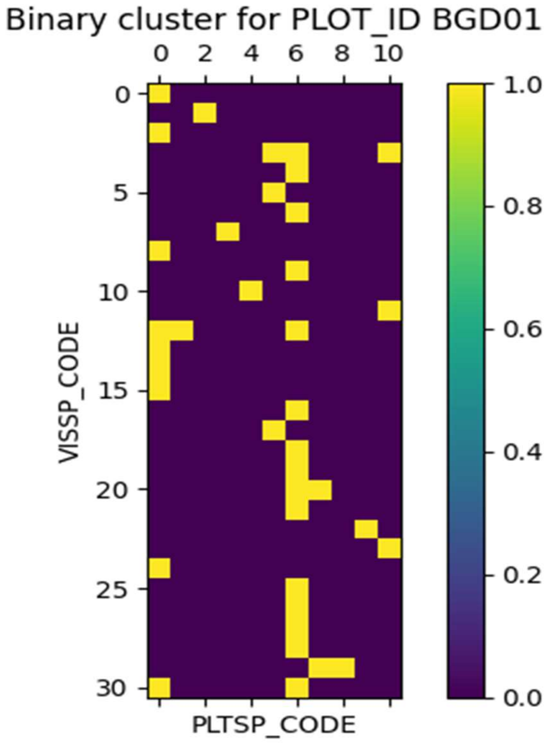
```
#color graph for the same

import matplotlib.pyplot as plt
# Generate a color graph for each binary cluster
for plot_id, binary_cluster in binary_clusters.items():
    # Create a figure and axes
    fig, ax = plt.subplots()

    # Set the title of the figure
    ax.set_title(f'Binary cluster for PLOT_ID {plot_id}')

    # Generate the color graph
    cax = ax.matshow(binary_cluster, cmap='viridis')

    # Set the labels for the axes
    ax.set_xlabel('PLTSP_CODE')
    ax.set_ylabel('VISSP_CODE')

    # Add a colorbar to the figure
    fig.colorbar(cax)

    # Show the figure
    plt.show()
```

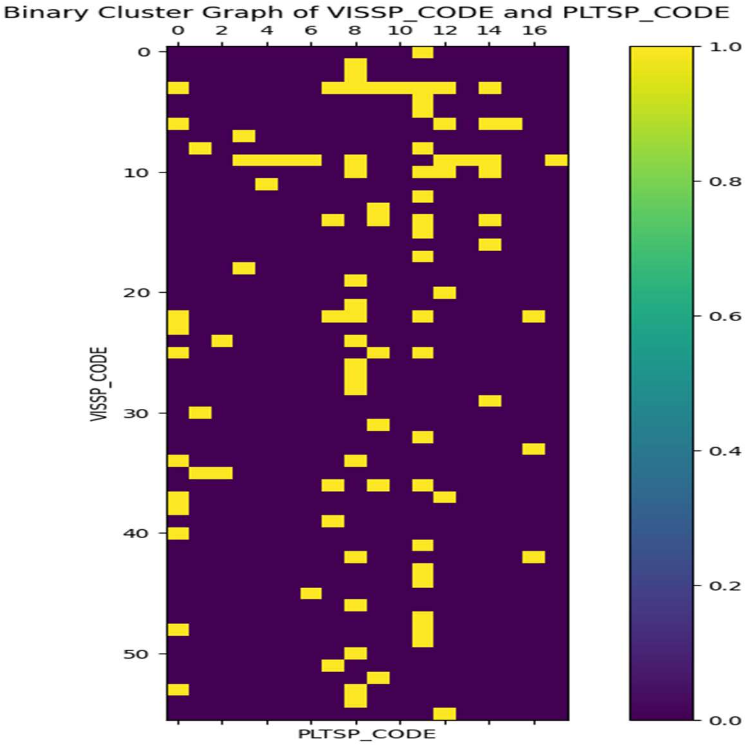


```
PLTSP_CODE   POTEGRAC   RUMEACET   SAXIOCCI
VISSP_CODE
AMMOSPX1            0          0          0
ANDRSCUT            0          0          0
APISMELL            0          1          0
ASEMPOLY            0          0          0
BOMBBIFA            0          0          0
BOMBFLAV            0          0          0
BOMBMAJO            0          0          0
BOMBMELA            0          0          0
BOMBMIXT            0          0          0
BOMBVOSN            0          0          0
BRADSPX1            0          0          0
CALLPULC            0          0          0
CERAACAN            0          0          0
CHLOHOFF            0          0          0
CHRYX141            0          0          0
COCCSEPT            0          0          0
CONOPIX3            0          0          0
CORESPX1            0          0          0
DIALS171            0          0          0
DIALSPX2            0          0          0
DIALSPXX            0          0          0
EMPISPXX            1          0          0
EPICPUNC            0          0          0
```



```
[52] DIALSPX2            0          0          0
     DIALSPXX            0          0          0
     EMPISPXX            1          0          0
     EPICPUNC            0          0          0
     ERISHIRT            0          0          0
     ERISTENA            0          1          0
     EULOTRIS            0          0          0
     EUPHCOLO            0          0          0
     EVYLSPX1            1          0          0
     EVYLSPX2            1          0          0
     HALIRUBI            0          0          1
     HERISPX2            0          0          0
     HOPLALBI            0          0          0
     HYLANEVA            1          0          0
     HYLANUNN            1          0          0
     HYLAWOOT            1          0          0
     LEPTDOLO            0          0          0
     MIRIIMMR            0          0          0
     MIRISPX3            0          0          0
     MUSCGEN1            0          0          0
     MUSCGEN3            1          0          0
     MUSCIDAE            0          0          0
     MYRMTINY            0          0          0
     OSMISPXB            0          0          0
     PENTASP1            0          0          0
     PLEBACMO            0          0          0
     PLEBICAR            0          0          0
     PLEBSPXX            0          0          0
     PYRAUNIF            0          0          0
```





For detailed work we are dividing each region into a csv file . We got 180 unique plot id that i.e, regions ,so we got 180 csv files.

CHAPTER – 6: CONCLUSION & FUTURE SCOPE

In conclusion, our project has made significant progress in implementing a missing link prediction model in an ecological environment using Python. We have successfully conducted initial analysis and training of the model, leveraging machine learning algorithms and network analysis techniques to predict missing **interactions between species in the ecosystem with promising results. Moving forward, further refinement of the model, validation with real-world data, and integration of additional features could enhance its predictive power and applicability in ecological research and environmental management. Continued work on this project holds great potential for improving our understanding of ecological networks and guiding conservation efforts in the future.**

The future scope of missing link prediction in ecological networks using Graph Convolutional Networks (GCN) and Graph Neural Networks (GNN) holds great potential. Here are some exciting directions for further research and development:

1. Integration **with Environmental Data**:
   - Incorporate additional environmental data (such as climate, habitat, and species interactions) into the network representation.
   - Explore how environmental factors impact missing link prediction accuracy.
2. **Dynamic Networks**:
   - Extend the models to handle temporal and dynamic ecological networks.
   - Investigate how GCN and GNN can adapt to changing ecological interactions over time.
3. **Multi-Modal Networks**:
   - Combine different types of ecological data (e.g., genetic, spatial, and behavioral) to create multi-modal networks.
   - Develop novel architectures that can effectively leverage diverse data sources.
4. **Interpretable Models**:
   - Enhance model interpretability to understand the ecological mechanisms behind missing link predictions.
   - Visualize feature importance and network embeddings.

5. **Transfer Learning**:
    - Investigate transfer learning techniques to improve performance on smaller ecological networks.
    - Pre-train models on larger networks and fine-tune on specific ecological datasets.
6. **Uncertainty Estimation**:
    - Quantify uncertainty in missing link predictions.
    - Explore Bayesian approaches or ensemble methods.
7. **Domain-Specific Metrics**:
    - Develop evaluation metrics tailored to ecological networks (e.g., considering trophic levels, species interactions, and ecosystem stability).
8. **Data Augmentation**:
    - Generate synthetic missing links to augment training data.
    - Investigate how data augmentation affects model performance.
9. **Collaboration with Ecologists**:
    - Collaborate closely with domain experts to identify relevant features and validate model predictions.
    - Bridge the gap between machine learning and ecological research.
10. **Real-World Applications**:
    - Apply missing link prediction models to real-world ecological conservation efforts, invasive species management, and ecosystem restoration.

In summary, the integration of GCN and GNN with ecological data has immense potential for advancing our understanding of ecological networks and improving conservation strategies. Researchers and practitioners should continue exploring these avenues to address critical ecological challenges

APPENDIX

https://portal.edirepository.org/nis/metadataviewer?packageid=knb-lter-and.5216.8

BIBLIOGRAPHY

1. https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/Proximus
2. https://www.researchgate.net/post/How_to_do_Binary_data_Clustering_using_Machine_Learning
3. https://scikit-learn.org/stable/modules/biclustering.html
4. https://github.com/padilha/biclustlib
5. https://github.com/Hutchinson-Lab/Poisson-N-mixture