# Network Based Candidate Gene Predictor – Product Documentation

Author: Bhagya Wijeratne – 15219

Date: 31/3/2024

## Introduction

Welcome to Network Based Candidate Gene Predictor tool, an advanced bioinformatics tool designed to assist researchers in analyzing protein-protein interaction (PPI) networks and predicting candidate genes and gene functions. Our platform offers a range of powerful algorithms and user-friendly interfaces, providing researchers with valuable insights into complex biological networks.

## Key features and functionalities

1. **PPI Network Construction:** Easily create NetworkX graph objects from PPI network files (.tsv) obtained from the STRING database. This feature simplifies the initial setup for network analysis.
2. **PPI network Analytical Insights:** Quickly determine the number of proteins and interactions within a given PPI network. These basic metrics help researchers understand the structure of biological networks.
3. **Candidate gene prediction for single and multiple functions:** Predict candidate genes associated with specific biological functions using algorithms like Majority voting and Hishigaki. Whether it's predicting single or multiple functions, this tool provides accurate results for informed decision-making.
4. **Prediction of most accurate function for unknown proteins:** This feature employs both the majority voting algorithm and the Hishigaki algorithm to predict the most accurate function for unknown proteins. By considering multiple seed lists simultaneously, the platform provides predictions, enhancing researchers' understanding of the functional roles of unknown proteins.
5. **User-Friendly GUI for Intuitive Navigation:** Our platform features a user-friendly graphical interface, simplifying the process of protein function prediction. With intuitive controls and clear navigation, users can effortlessly interact with the application, whether predicting functions for a single protein or multiple proteins simultaneously.
6. **Command line implementation:** When necessary to call functions separately, the code compilation allows separate implementation using command line interface as well.
7. **Visualization of the graph data:** Command Line implementation has the ability to create a .gml network file that can be viewed on Cytoscape

## Scope and Restrictions of the Software

1. Restrictions in seed protein file format: The seed protein lists have to be in a specific format for the correct implementation (provided below in the user manual)
2. Taxonomy specificity: The program was designed mainly to analyse data concerning Arabidopsis thaliana hence this taxonomy specific
3. In cases where the highest Majority Voting score/ Hishigaki Score is similar for a protein is found for more than one function, both functions are predicted as most accurate functions.
4. Order of predictions changes when the scores are equal
5. Minor changes in the seed protein lists may impact the results of the Majority Voting Algorithmic predictions. Due to similar scores and high annotation dependency.

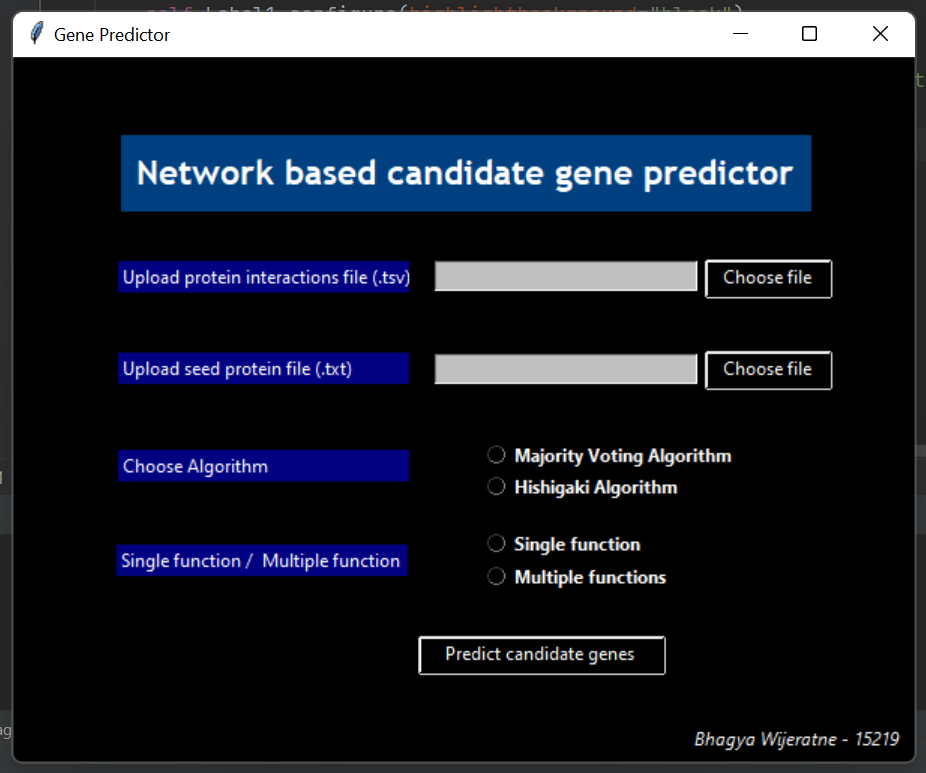
## Future Developments

1. Connect to GO annotations database and STRING database to make predictions directly without downloading data
2. Improving the restrictions around taxonomy specificity
3. Similar score predictions needs to be optimized using necessary methods – self consistency test.
4. Optimizing the algorithm: Optimizing the performance of the algorithm by code modification.
5. Graphical Visualization: Introducing graphical visualization of protein interaction networks and predicted functions can aid users in better understanding the results and exploring the relationships between proteins.
6. Accuracy evaluation: Employing a method to check the accuracy of the prediction results
7. A shortcoming of this approach is that within the n-neighborhood, proteins at different distances from p are treated in the same way: Chua et al (2006) try to tackle the second problem by investigating the relation between network distance and functional similarity. They focus on the 1- and 2-neighborhoods of a protein, and devise a functional similarity score that gives different weights to proteins according to their distances from the target protein.

## User Manual – Step by Step instructions

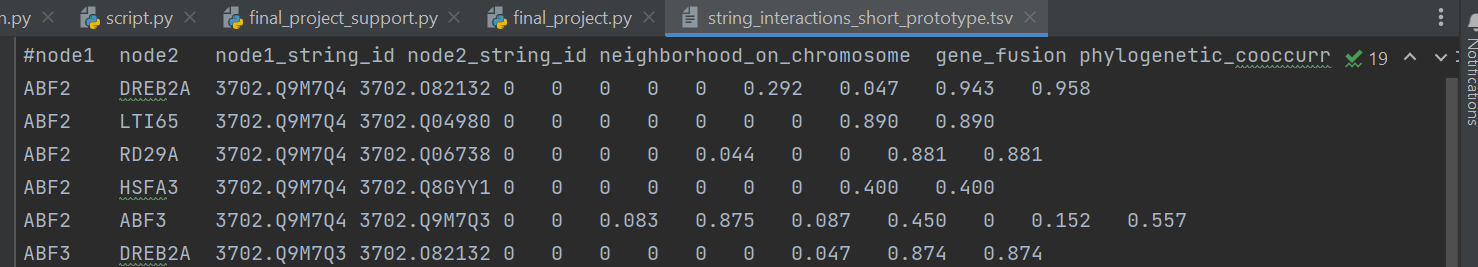
1. Download the program code
2. Extract and upload to python IDE as a python project
3. For command line implementation – run main.py file using python IDE
4. For GUI implementation - Run final\_project.py file

* The following dialog box will appear



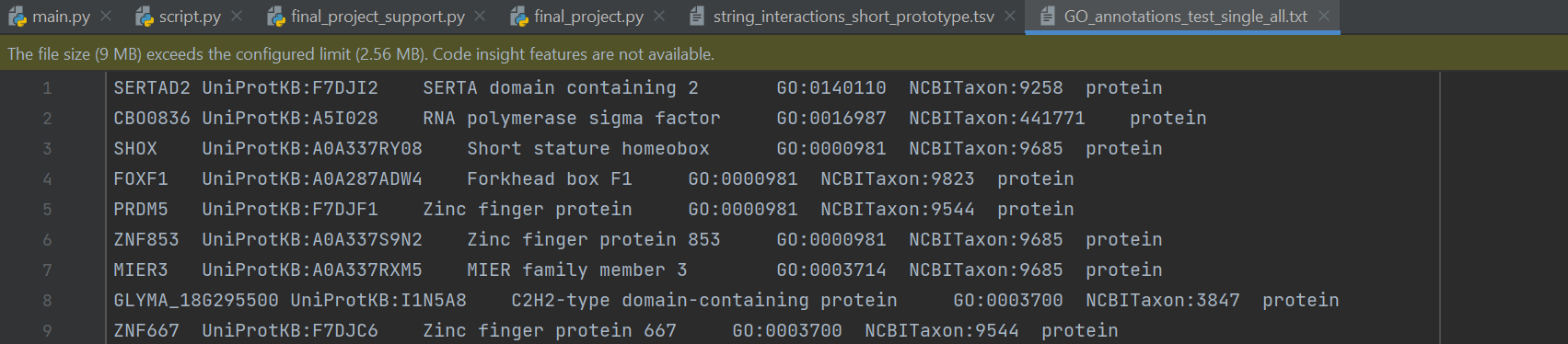
1. Choose the correct protein interactions file downloaded in the .tsv format from the STRING database.

Format – default .tsv format given by STRING database ( organism: *Arabidopsis thaliana*)

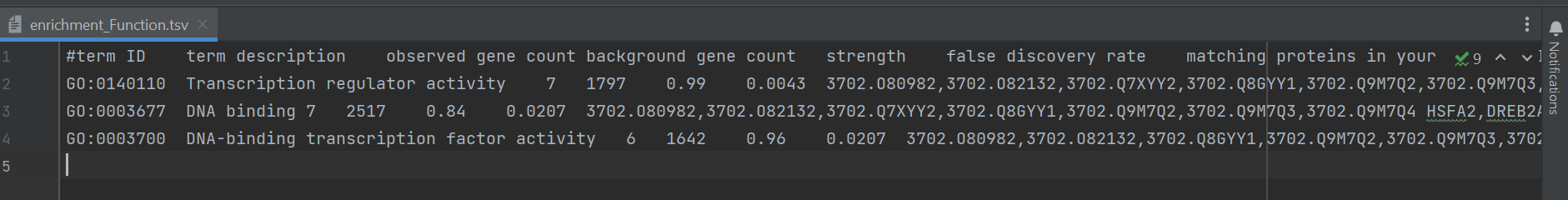


1. Choose the correct seed protein file in .txt format as well.

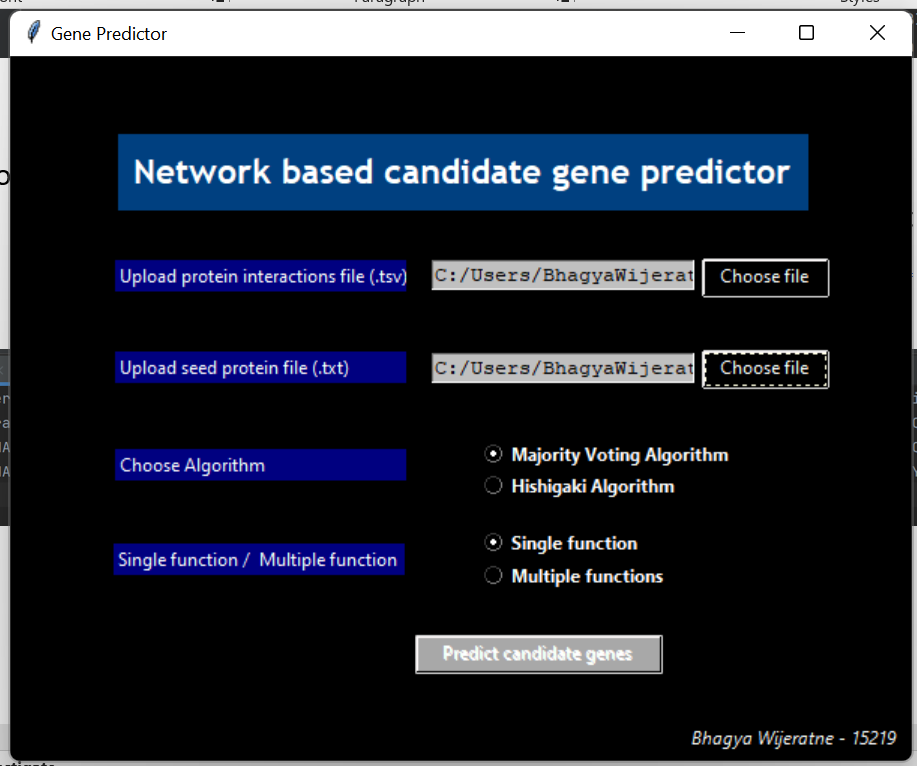
* Format for single function predictions
  + As long as the known proteins for the function are present in the 1st column of the seed protein list, it is suitable.



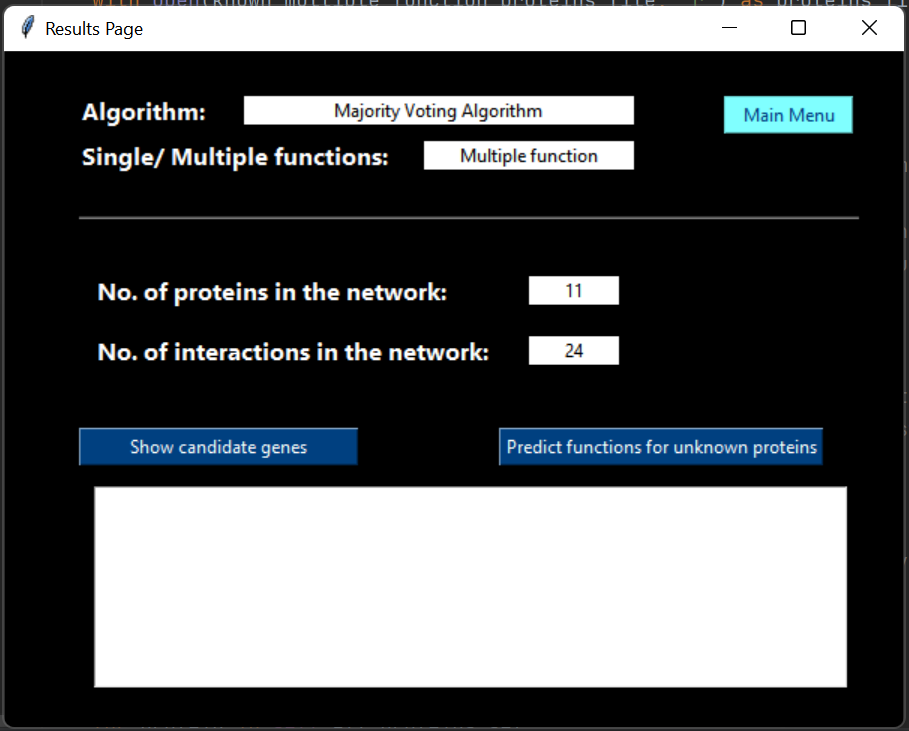
* Format for multiple function seed protein list
  + The function should be present in the second column and a list of comma separated proteins known for that function is found in the 8th column ( as downloaded using GO annotations)



1. Select the other options as press “Predict candidate genes” button.

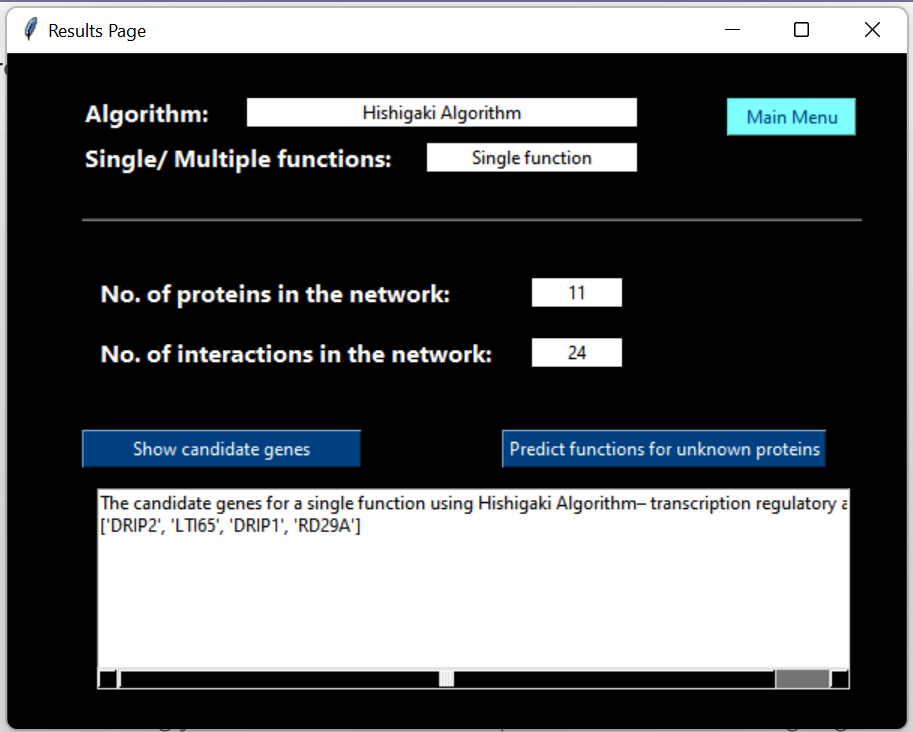


1. This leads to results page where the number of proteins in the network and number of interactions are displayed. The previously chosen Algorithm and the Single/ Multiple function prediction nature is also shown for clarity.

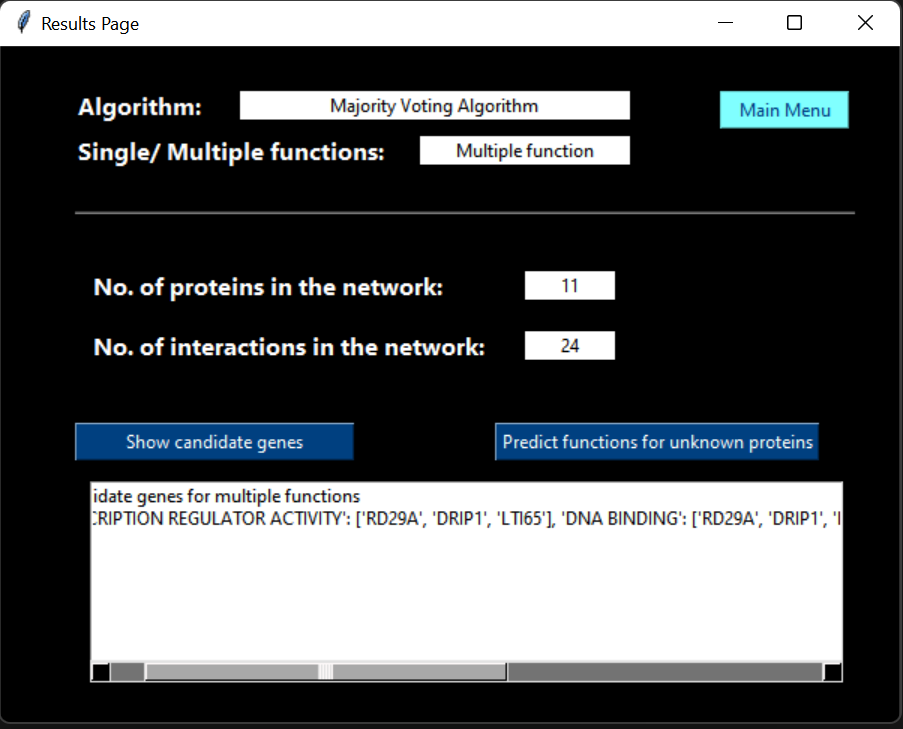


1. Hit “Show candidate genes” button to find candidate genes for Single/ Multiple functions.

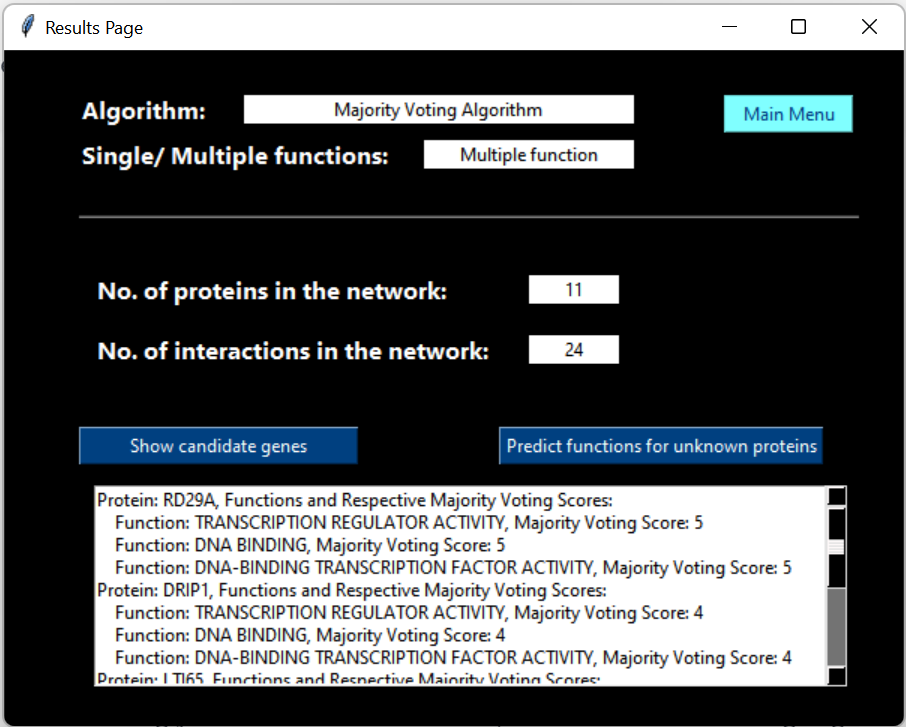
* If it is a single function, the candidate genes out of the unknown genes are displayed directly for that particular function



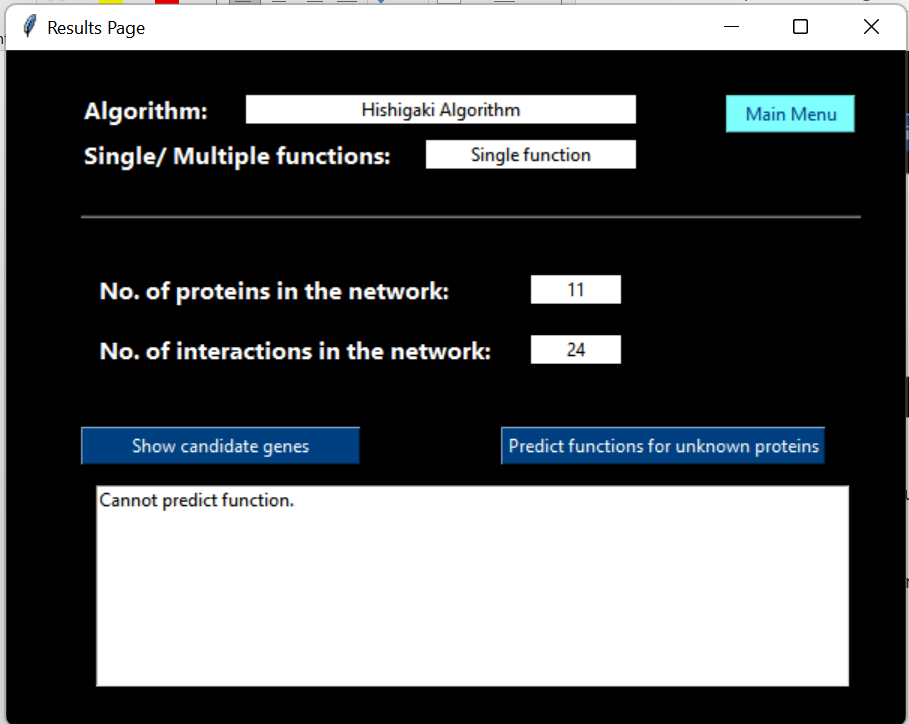
* If it is a multiple function prediction, candidate genes are predicted for each function based on the unknown genes for that particular function.



The threshold value for selecting the candidate genes can be changed by manipulating the code as necessary.

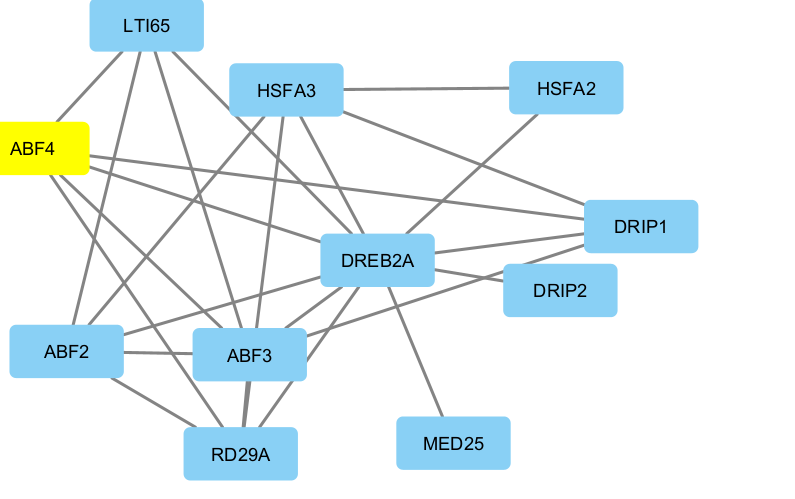
1. Hit the “Predict functions for unknown proteins” button to predict the most accurate function/ functions for all unknown proteins in the network.

This can be only carried out in the case of multiple function prediction. Since there is no function prediction to make if it is a single function, an error message will appear as follows



1. Use the main menu button to go back and change the options to make different predictions as necessary. The options can be changed if the uploaded files have the correct format specified above.

## Visualization of the .gml file created from the main.py script.



## Source code documentation

## <https://github.com/Bhagyaaa19/Network-based-candidate-gene-predictor.git>

## API documentation

### gene\_predict\_single\_function(known\_proteins\_file)

* **Description**: Predict candidate genes for a single function using the majority voting algorithm.
* **parameters**:
  + known\_proteins\_file: Path to the file containing GO annotations for the seed protein list for a single function.
* **returns**: None ( output is printed and captured to a variable)

### gene\_predict\_multiple\_function(known\_multiple\_function\_proteins\_file)

* **Description**: Predict candidate genes for multiple functions using the majority voting algorithm iteratively, one function at a time.
* **parameters**:
  + known\_multiple\_function\_proteins\_file: Path to the file containing GO annotations for multiple seed protein lists for three different functions.
* **return**: None ( output is printed and captured to a variable)

### function\_predict\_multiple\_function(known\_multiple\_function\_proteins\_file)

* **Description**: Predict the most accurate function for unknown proteins by considering all seed lists at one time using the majority voting algorithm.
* **parameters**:
  + known\_multiple\_function\_proteins\_file: Path to the file containing GO annotations for multiple seed protein lists for three different functions.
* **return**: None ( output is printed and captured to a variable)

### gene\_predict\_single\_function\_Hishigaki(known\_proteins\_file)

* **Description**: Predict candidate genes for a single function using the Hishigaki algorithm.
* **parameters**:
  + known\_proteins\_file: Path to the file containing GO annotations for the seed protein list for a single function.
* **return**: None (output is printed and captured to a variable)

### gene\_predict\_multiple\_function\_Hishigaki(known\_multiple\_function\_proteins\_file)

* **Description**: Predict candidate genes for multiple functions using the Hishigaki algorithm iteratively, one function at a time.
* **parameters**:
  + known\_multiple\_function\_proteins\_file: Path to the file containing GO annotations for multiple seed protein lists for three different functions.
* **return**: None ( output is printed and captured to a variable)

### function\_predict\_multiple\_function\_Hishigaki(known\_multiple\_function\_proteins\_file)

* **Description**: Predict the most accurate function for unknown proteins by considering all seed lists at one time using the Hishigaki algorithm.
* **paratemers**:
  + known\_multiple\_function\_proteins\_file: Path to the file containing GO annotations for multiple seed protein lists for three different functions.
* **returns**: None ( output is printed to a variable)

countNeighboursSingle(protein, known\_proteins)

* **Description**: Counts the number of edges between a given protein and known proteins in the network.
* **parameters**:
  + protein: Name of the protein for which neighbors are counted.
  + known\_proteins: List of known proteins to compare against.
* **returns**: Integer indicating the count of neighbors.

### extractSeedProteinsMultipleFunction(known\_multiple\_function\_proteins\_file)

* **Description**: Extracts functions and relevant proteins from multiple-function seed protein lists.
* **parameters**:
  + known\_multiple\_function\_proteins\_file: Path to the file containing GO annotations for multiple seed protein lists for different functions.
* **returns**: Updates the known\_multiple and unknown\_multiple attributes.

### extractSeedProteinsSingleFunction(known\_proteins\_file)

* **Description**: Extracts known proteins from the seed protein list for a single function.
* **parameters**:
  + known\_proteins\_file: Path to the file containing GO annotations for the seed protein list for a single function.
* **returns**: Updates the known\_set, unknown\_genes, and all\_proteins\_set attributes.

### calculateHishigakiScores()

* **Description**: Calculates the Hishigaki scores for each function and protein.
* **parameters**: None
* **returns**: Updates the hs\_dict\_multiple attribute.

## Attributes

* all\_proteins\_set: (Type: set): Set containing the names of all proteins in the network.
* known\_set: (Type: set): Set containing the names of known proteins.
* unknown\_genes: (Type: set): Set containing the names of unknown proteins.
* protein\_graph: (Type: network graph object): NetworkX graph object containing the PPI network information.
* n\_proteins: (Type: int): Number of proteins in the network
* n\_interactions: (Type: int): Number of interactions in the network
* known\_multiple: (Type: Dictionary): A dictionary containing GO annotations for multiple seed protein lists for different functions. Each function serves as a key, and the corresponding value is a list of known proteins associated with that function
* known\_proteins\_in\_network (Type: Dictionary): A dictionary containing the counts of known proteins for each function belonging to the network. Each function serves as a key, and the corresponding value is the count of known proteins for that function.
* hs\_dict\_multiple (Type: Dictionary): A dictionary containing the Hishigaki scores for multiple functions. Each function serves as a key, and the corresponding value is a nested dictionary where each protein is mapped to its Hishigaki score.
* n\_networks (Type: Integer): The number of networks present in the PPI network graph object.