graph2class

▶ View Source

def calc_bc(G, return_dict):

▶ View Source

Parallel subprocess function to calculate the betweenness centrality.

Args

- **G ([networkx graph]):** graph

Returns

[dictionary]: betweeness centrality dictionary from multiple processes

def calc_shortest_pthlen(G, return_dict):

▶ View Source

Parallel subprocess function to calculate the average shortest path length.

Args

- **G ([networkx graph]):** graph

Returns

[dictionary]: average shortest path length dictionary from multiple processes

def calc_graph_features(G):

▶ View Source

Calculates several graph network features. If not connected, largest subgraph is used. Uses multiprocessing for parallelsim.

Args

- **G ([networkx graph]):** graph

Returns

[dictionary]: features dictionary

def similarity_measure(x1, x2):

▶ View Source

calculates the similarity between two feature values. similarity = 1 - the relative distance between features (x1 and x2)

Args

- **x1 ([float]):** feature from graph 1 (must range between 0,1)
- **x2 ([float]):** feature from graph 2 (must range between 0,1)

Returns

[float]: returns the relative similarity between 2 features

def calc_similarity_score(G1_dict, G2_dict, feature_list):

▶ View Source

calculates the similarity score of two graphs

Args

- **G1_dict ([dict] or [Pandas datafrane]):** graph 1 features dictionary or dataframe. must be able to use a key to access values
- **G2_dict ([dict] or [Pandas datafrane]):** graph 2 features dictionary or dataframe. must be able to use a key to access values
- **features_list ([list]):** list of graph features to compare. must be keys in graph features dictionary (above)

Returns

> [float]: similarity score (0,1) where 1 is an identical graph.

def process_graphs(graph_fnames):

▶ View Source

take a list of graph files, calculate their features, and return as a dataframe

Args

- **graph_fnames ([list]):** list of graph filenames to process

Returns

> [pandas dataframe]: dataframe containing graph features for each graph in filename list

def classify_graphs(class_file_list, sample_file_list, feature_list):

▶ View Source

Classifies a similarity score from a list of Class and Sample graphs

Args

- **class_file_list ([list]):** list of control/reference graph files (classes)
- **sample_file_list ([list]):** list of non-control/non-reference graph files (samples)
- **feature_list ([list]):** list of which features to use for similarity score. must be a valid key to the graph features dictionary/dataframe (above)

Returns

> [pandas dataframe]: each colummn is a class and each row is the similarity score of the sampled graph

def process_similarity_df(class_similarity_df):

▶ View Source

Generates y_true and y_pred based on the similarity score dataframe.

y_true is a list where each index is a class and each value is the class value. E.g., class 1 is y_true[1] = 1, class 2 is y_true[2]=2, etc.

y_pred is a list where each index is a sample and each value is the maximum similarity score for that sample.

Note: This assumes the correct classification is along the diagonal of the similarity matrix/dataframe.

ref: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report

Args

- **class_similarity_df ([pandas dataframe]):** each column is a class graph and each row is a sample graph. A_ij is the similarity score between graphs i and j. The exception is one column 'name' which contains the names of the sampled graphs for each row.

Returns

([tuple of lists]): y_true, y_pred