Network Classification and Categorization Paper

James P. Canning, Emma E. Ingram, Adriana M. Ortiz and Karl R. B. Schmitt

June 2017

Abstract

Networks are often labeled according to the underlying phenomena that they represent, such as re-tweets, protein interactions, or web page links. Our research seeks to determine if we can use machine learning techniques to gain a better understanding of the categories of networks on the Network Repository (www.networkrepository.com) and then classify the unlabeled networks into categories that make sense. It is generally believed that networks from different categories have inherently unique network characteristics. Our research provides conclusive evidence to validate this belief by presenting the results of global network clustering and classification into common categories using machine learning algorithms. The machine learning techniques of Decisions Trees, Random Forests, Linear Support Vector Classification and Gaussian Naïve Bayes were applied to a 14-feature 'identifying vector' for each graph. During cross-validation, the best technique, Gaussian Naïve Bayes, achieved an accuracy of 92.8%. After training the machine learning algorithm it was applied to a collection of initially unlabeled graphs from the Network Repository (www.networkrepository.com). Results were then manually checked by determining (when possible) original sources for these graphs. We conclude by examining the accuracy of our results and discussing how future researchers can make use of this process.

1 Introduction

Network science is a field which seeks to understand the world by representing interactions and relationships between objects as graphs. By organizing data into networks, we can better understand how objects of interest are connected. However, what if we want to know how the networks themselves are connected? Networks are often labeled according to the underlying phenomena that they represent, such as re-tweets, protein interactions, or web page links; therefore, there are many different types of networks, some more similar than others. Given any graph, how can we tell to which category it belongs? The problem of network classification has been studied from different perspectives for years. There has been work done on calculating the similarity between graphs [8], and classifying a graph based on topological features, both global and local [9][19]. These studies, however, focus only on the classification of networks within specific fields, such as distinguishing a network of cancerous brain cells from a network of cancerous breast cells [19], or distinguishing between structures arising from different types of social graphs [28]. Network classification on a broader scale, such as distinguishing between a collaboration network and an infrastructure network, has not yet been investigated.

We want to classify networks across disciplines according to their features to increase interdisciplinary work in the world of network science. Some researchers have been using networks to represent data for a long time and have developed useful ways of working with their networks to obtain meaningful results, for example, sociologists have been using ideas from graph theory to understand social organizations for years, as described in a 1969 paper by J.A. Barnes [6]. In contrast, others have just began exploring networks as ways to represent their information. By finding a way to classify graphs we hope to help guide individuals and institutions that are newer to the realm of network science by showing them other, potentially more understood networks that have similar features to theirs. We hope our research will foster collaboration and understanding across fields that use networks.

Section 2 begins with an overview of what other researchers have done in this field. Section 3 discusses the data that we chose to use and our method for cleaning it. Section 4 describes the methods used in this research and how they were used, including: unsupervised machine learning techniques (Section 4.3), supervised machine learning techniques (section 4.4) and different variations done with the supervised

learning models (section 4.5). Section 5 analyses all of the results and scores obtained. Finally, Sections 6 and 7 summarize and conclude this work.

The major contributions of our work include:

1. Presented a method for accurately classifying networks according to their discipline of origin using machine learning models.
2. Discussed how incorrectly classified graphs can actually be useful for the research community.
3. Proposed combinations of categories that seem to be similar
4. Proposed new categories to be added in the Network Repository
5. Identified collections that could be useful with sufficient data points

2 Related work

The problem of classifying networks is one interlaced with the problem of determining network similarity. Both of these research areas have far-reaching impacts not only in the field of network science, but in any field that uses networks such as biology and cheminformatics.

Network similarity has been accomplished through many different strategies including Graph FingerPrinting [8], and analyzing sub-graphs to extract features that highlight the underlying structure of a network [14]. Some researchers have even used image similarity techniques on graphs' adjacency matrices to compare them [31]. The results from these network similarity works show that there are many different graph features that could be used to compare networks, and that quantifying similarity can aid the process of using machine learning to classify a graph. There are so many different network similarity measures that works have been created to specifically analyze the differences between them and guide researchers to similarity measures that would work best for them [26]. In addition, in multiple works on network science ([8][26]) we have found the use of the Canberra distance metric (defined in [17]) to compute the similarity between two vectors. We used this finding later in our research when we investigated a metric to use in one of our processes.

The Graph FingerPrint strategy uses feature vectors including both global and local graph features to measure the similarity between two graphs. In this paper we use a similar strategy, namely combining a variety of features into an identifying vector of a graph and using this to classify the networks.

One of the challenges of network classification is collecting a sufficient amount data to classify. For this reason, research on network similarity and classification often uses synthetically generated graphs [8] [21], as these can easily be created and customized. The work that has been done on real-world networks has largely used benchmark data sets consisting of graphs pulled from the same field, such as chemical compounds or protein interactions [14] [19]. However, there has been no research showing that this type of network classification can be done using real-world networks from a variety of different fields. In this research we address that question in order to confirm the widely held belief among network scientists that networks from different disciplines have different characteristics.

2.1 Feature Comparison

The work by Li, Semerci, Yener, and Zaki [19] addresses network classification within data sets using topological graph features. This work is successful in its classification, using feature vectors and a support vector machine classifier, showing that feature based classification is possible when classifying chemical compounds, proteins, and cell-graphs. However, like other work in network classification, they did not implement their process using graphs across disciplines. The 23 features identified in this paper especially helped our work, as we also used feature vectors for classification. When investigating which features were the most important for our models, we looked at the features that this work identified as being widely important and compared them to the ones we found. The most important features found in this paper are number of nodes, number of edges, average degree, average clustering coefficient, number of eigenvalues and energy.

Bonner's paper "Efficient Comparison of Massive Graphs Through the Use of Graph Fingerprints" [8] addresses the problem of graph similarity, especially focusing on large graphs. It presents an approach called

"Graph FingerPrint Comparison" (GFPC) which enables the comparison of large graphs by extracting both local and global level features, creating a compact abstraction of a graph. This approach was developed to investigate new techniques to study small changes of graphs over time, so it is very sensitive to very small changes. Bonner found that GFPC outperformed NetSimile on all metrics of the ones he used. The results have shown the GFPC approach to be an accurate but compact representation of a graph. Bonner suggested that a potential direction for future work would be the classification of networks by comparing their "fingerprint" vectors to a "labeled ground-truth" fingerprint for a certain domain. We were inspired by this idea of creating an identifying vector for each network and comparing them to classify networks.

Another approach made by Bonner was using node level topological features and global features to classify massive networks using deep neural networks [9]. Deep neural networks are algorithms that are modeled on the human brain, and are used to identify patterns. This approach of classifying networks was called Deep Topology Classification (DTP). Two data sets were generated synthetically, using multi classification and binary classification, to test this new classification method. DTP then was compared with a Support Vector Machine (SVM) classifier, proving to be more accurate than the SVM.

2.2 Subgraphs

Many researchers have approached the problem of classifying networks by analyzing smaller sections of graphs instead of looking at the network as a whole. This is often done by working with "subgraphs", which are simply graphs contained in another graph. These subgraphs give insight into the structure and behavior of the entire network. Also, analyzing the frequency and number of subgraphs provide metrics to distinguish graphs with different qualities.

The paper by Ugander, Backstom, and Kleinberg [28] uses subgraph frequency and collections of "neighborhoods" to examine networks. This work then uses the subgraph frequency to classify between different types of social networks on Facebook, discerning between communities, groups, and events.

The work by Guo and Zhu [14] also analyzed subgraphs in a network, but instead of focusing on subgraph frequencies they generated subgraph features, and used these features for classification. The graphs used in this work came from benchmark data sets on chemical compounds, protein structures, and citation networks.

Also looking at subgraphs, Ahmed, Neville, and Rossi [3] presented an efficient method to count the number of k-sized graphlets (another word for subgraphs) in a network. This work showed how the number of graphlets can be used with an SVM to distinguish between different types of protein and chemical compound graphs.

The work by Milenković, Lai, and Pržulj [21], motivated by the large amounts of biological network data now available, builds on the concepts of network motifs (small over-represented subgraphs) and graphlets (small induced subgraphs). It introduces GraphCrunch (GC), a software tool that can compute local network features based on motifs and graphlets. GC uses graph comparison techniques to compare large real-world networks to random graph models. We investigated GraphCrunch because it can calculate computationally difficult network properties like the relative graphlet frequency distance and the graphlet degree distribution agreement.

Although graphlets and the graph characteristics calculated using them can be powerful tools for comparing networks we ultimately decided not to use these tools in our research thus far. We explored the literature around these graph features because we believe that they are worth considering when dealing with the problem of graph comparison and similarity. However, because we already had data with global and local features calculated, in the interest of time it was best to focus on the methods that used these types of features.

Through our literature search we gained insight into what makes specific networks distinct, and how network comparison and classification is implemented. We saw evidence that machine learning could be used to classify networks in few, related categories, which motivated our research question. It also gave us an outline of where to begin our process, and which tools to explore.

3 Data

3.1 Python Packages for Data Manipulation

Python is a powerful programming language that we chose to use for our project because of its ability to work well on data science problems. We used DataCamp www.datacamp.com, an online data science learning website, to learn about the Python language and a few key packages we would use. One of the important packages we use is called pandas. We used pandas [27] to store our data in an easy to manipulate object called a DataFrame [27]. For visualizations the package Bokeh [25] was used. Although not a python package, IDEAR [4], a tool built by Microsoft for use in data science, was used for exploratory data analysis.

3.2 Data Source

With the aid of two of our collaborators, Dr. Ryan Rossi and Dr. Nesreen Ahmed, we obtained feature data from the Network Repository [24] for 1241 labeled graphs as well as 60 unlabeled miscellaneous graphs (See Figure 1).

The first problem we encountered was that when we read the values from the Network Repository data file into our pandas DataFrame we found that all of the numerical values were being stored as objects instead of floating-point values. To solve this problem we had to coerce the values to a numeric data type. Once this operation was performed on each feature the data was ready to be manipulated. The features in the data are listed in Table 1. Each of these features is discussed in Section 3.5.

Number of Nodes	Average Degree	Average Clustering Coefficient
Number of Edges	Assortativity	Fraction of Closed Triangles
Density	Total Triangles	Maximum Clique (lower bound)
Minimum Degree	Maximum Triangles	Maximum K-Core
Maximum Degree	Average Triangles	Chromatic Number

Table 1: Features calculated by the Network Repository

3.3 Data Cleaning

3.3.1 Removing Less Useful Collections

Upon investigating the collections of graphs in the data set we noticed that there were some collections that did not seem to work with our research question. For example, the collection "Temporal Reachability" was made up of graphs that represented slices of networks over time [30]. Since the graphs in this collection may not all correspond to the same discipline and represented changes in graphs over time, not the graphs themselves, this collection will not be used in this research. A similar collection that we decided not to use is "Dynamic Networks." As the name suggests, this section represents graphs that change, so the data might include the same graph multiple times with some changes. Due to these characteristics, these graphs would not be useful for our classification process.

Three other collections had to be removed from our data sets. The Center for Discrete Mathematics and Theoretical Computer Science (abbreviated DIMACS) has created many collections of network data to be used as benchmark data sets for specific problem solving[1]. Two such data sets appear in our data, "DIMACS" and "DIMACS10." Due to their purpose as data for implementation challenges, these graphs do not all have similar sources so are not used for training or testing the models in this project. A similar collection, the Benchmarks with Hidden Optimized Solutions Library [32](appearing in our data as "BHOSLIB"), was created for testing solutions to a specific problem and is thus excluded from use in this project as well.

3.3.2 Removing Missing Features

In addition to cleaning the overall collection data by removing collections that were less relevant to our research we also cleaned up the feature data. Some graphs had "Not a Number" (NaN) entries, but one

specific feature, chromatic number had the most NaN entries; more than half of our graphs did not have a value for 'Chromatic Number'[7]. Instead of dropping all of the graphs that had a missing value for chromatic number and lose too much information, we decided to drop this feature from consideration in our research.

After getting rid of Chromatic Number, only 8 graphs still had data issues including at least one NaN entry. Some of the missing values were for Fraction of Closed Triangles, Assortativity and Maximum k-core. We considered generating the missing values for these graphs, but instead we decided against it because of how big these graphs are and how time consuming it would be. Instead we dropped these 8 graphs from our data. These graphs and details of their missing features are shown in Table 2.

Graph Name	Missing Features
scc-rt-lebanon	Assortativity and Fraction of Closed Traingles
scc-rt-obama	Assortativity and Fraction of Closed Traingles
tech-p2p	All
soc-LiveJournal1	All except Nodes, Edges and Degree (Maximum, Minimum and Average)
soc-sinaweibo	All except Nodes, Edges and Total Traingles
soc-student-coop	Assortativity and Fraction of Closed Triangles
soc-friendster	Assortativity and Average Clustering Coefficient
soc-twitter	All except Nodes, Edges and Total Traingles

Table 2: Graphs with Missing Features

Another task we performed as a part of our data cleaning process was investigating some of the zero values in our data. Sometimes researchers represent missing data as a zero instead of a 'NaN' value, so we looked for zeros that did not make sense. The features that wouldn't be a problem when they are equal to zero include:

1. Minimum Degree - there could be nodes that are not connected to any other node
2. Total Triangles - a graph could have zero triangles
3. Maximum Triangles - if total triangles is zero, the maximum will be zero
4. Average Triangles - if total triangles is zero, the average will be zero
5. Assortativity - when the network is non-assortative
6. Average Clustering Coefficient - if total triangles is zero
7. Maximum k-core - if zero is the maximum degree k for which there exists a maximal induced subgraph which has minimum degree greater than or equal to k
8. Maximum clique - if there are no cliques in the network

Some of the values for average triangles were zero even though the total triangles were not zero, so we recalculated these values using the data we had.

3.4 Data Preprocessing

3.4.1 Balancing Data

After trimming down our data set and analyzing the file, the features still had a wide range of values across many magnitudes, for example, some graphs had fewer than 30 nodes while some graphs had millions of nodes. In addition, some graphs had a density of 0.1 while others had a density of less than 0.0001. To make these features more comparable, we decided to scale the data by feature. Python's scikit-learn package [22] includes a few different scalers and normalizers that we could use, but we had to narrow it down to the one that worked best for our data. First of all, we wanted a tool that would scale the data along the columns of our dataframe, by feature, for example number of nodes, instead of by graph. We eliminated

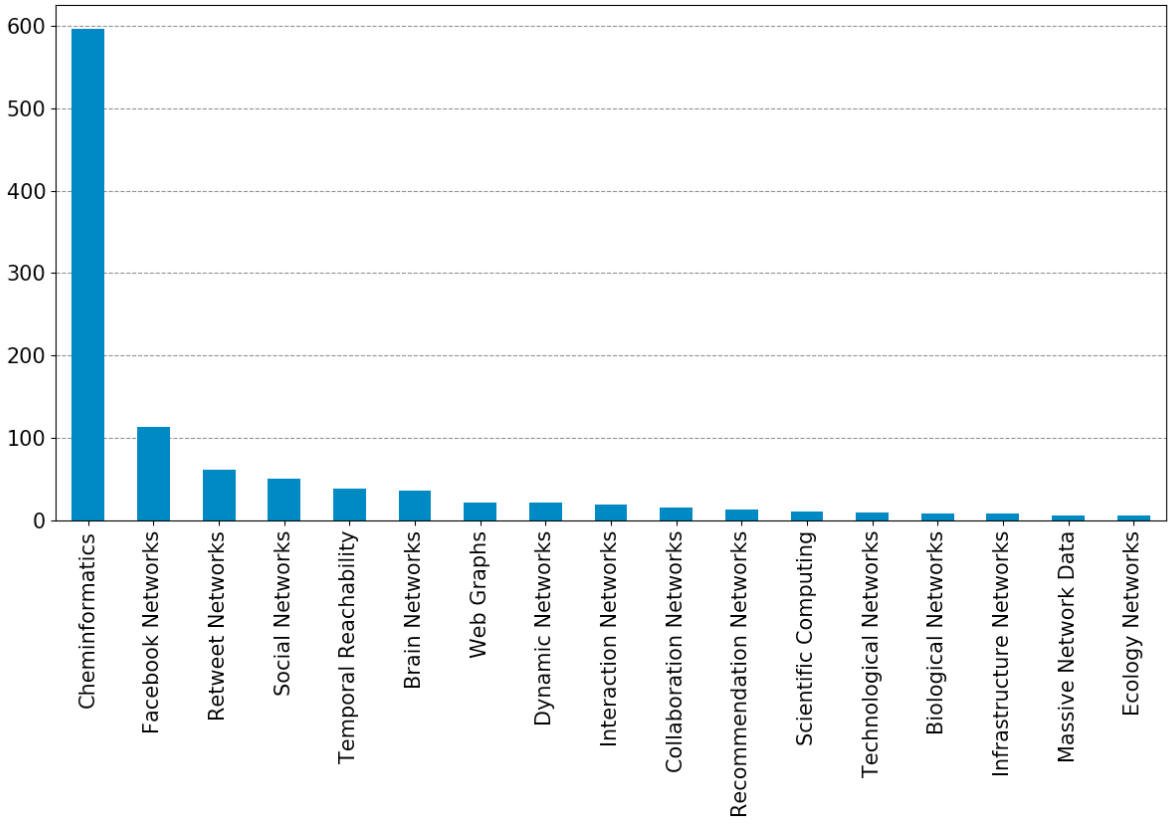


Figure 1: Data after removing less useful collections and graphs with missing features

the 'Normalizer' tool from scikit-learn because it normalized the data by graph. The other two scalers that we were familiar with were the 'Min-Max Scaler' and 'Standard Scaler.' The Min-Max Scaler puts the data in the column in the range of 0 to 1, while the Standard Scaler normalized the data in the column to have a mean of zero and a unit variance. To find out which of these two scaling tools would work the best for our project we ran k-means clustering (see section 4.3) on our data twice: once using each scaler. We then examined the cross-tabulations of the results and found that the Min-Max Scaler led to more graphs being clustered with other graphs from their own collection. Henceforth we will use the Min-Max Scaler on our data.

3.4.2 Downsampling Data

The collections of networks need to be relatively balanced because the goal is to classify networks into categories. (See [10] for a discussion of dealing with imbalanced data). We found that 57% of our data was from the collection Cheminformatics (600 graphs), with the next largest collection of Facebook Networks being 23% of our data (114 graphs), see Figure 1. To better represent all of the collections without getting rid of too much data, we decided to downsample the Cheminformatics collection.

We hypothesized that there could be patterns within the Cheminformatics graphs that we did not want to get rid of through our downsampling process, so we decided to run k-means clustering (see Section 4.3) to see if there were any distinct clusters. First, we generated a plot of the inertia vs. the number of clusters used in the k-means (Figure 2). Although there was not an incredibly distinct elbow in this plot, the marginal change in inertia started to decrease at k=4, so we decided to run k-means clustering on Cheminformatics using 4 clusters.

From each cluster we then randomly selected 20% of the networks to include in our experimental data set. We then combined these smaller slices of data into a new, downsampled Cheminformatics section. After this downsampling process we were left with 119 Cheminformatics graphs, much more comparable to our 114 Facebook networks.

After the data cleaning process we were left with 494 graphs and 15 collections (See Figure 3).

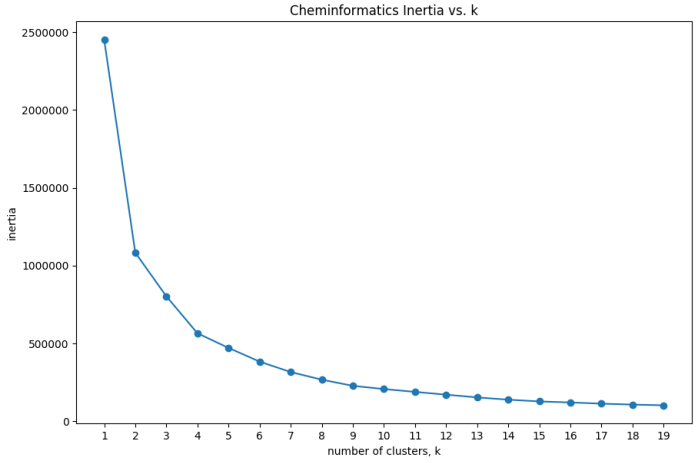


Figure 2: Inertia Plot for Cheminformatics Collection

3.5 Feature definitions

Formal feature definitions can be found in Maarten van Steen's book *Graph Theory and Complex Networks*[29] and were calculated by the Network Repository [24]. Let $G = (N, E)$ be a graph with N nodes and E edges.

- Nodes - Number of nodes in a graph, N

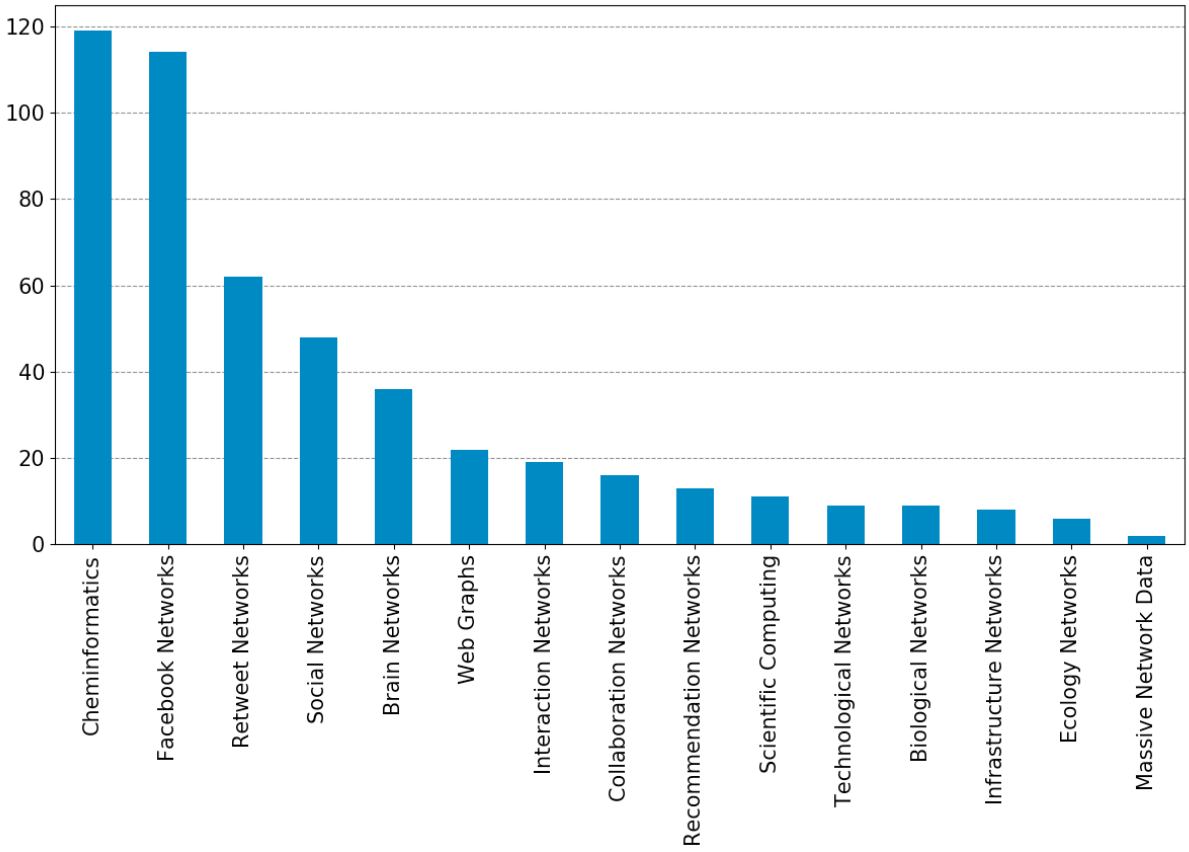


Figure 3: Data after downsampling Cheminformatics

- Edges - Number of edges in a graph, E
- Density - The ratio of the amount of edges in a graph to the amount of possible edges. This is calculated using the formula $\frac{N(N-1)}{2}$.
- Maximum degree - The degree of a node is the amount of neighbors connected to itself. The maximum degree is the largest node degree.
- Minimum degree - The lowest node degree in a graph. If there are nodes not connected to any other, the minimum degree is 0.
- Average degree - The average degree of all nodes in a graph.
- Assortativity - Tendency of nodes to connect to other nodes with similar degree, or in contrast, the tendency of dissimilar nodes to connect.
- Total triangles - A triangle is a complete subgraph of G with exactly three vertices. The total number of triangles in a graph is the sum of all triangles.
- Average triangles - Average number of triangles formed by an edge.
- Maximum triangles - Maximum number of triangles in a graph.
- Average clustering coefficient - The clustering coefficient of a graph quantifies how a node in a graph tends to cluster together.

- Fraction of closed triangles (Global clustering coefficient)- It is defined as the ratio of the number of triangles and the number of possible triangles in a graph.
- Maximum k-core - A k-core of G is a maximal subgraph of G such that for all vertices in the subgraph, the degree is greater or equal to k. Maximum k-core is the largest k.
- Maximum clique (lower-bound) - Largest complete sub-graph of G, such that the subgraph is not contained in a any other complete subgraph of G.

4 Methods

4.1 Python Packages for Analysis and Machine Learning

One of the Python packages mentioned before that was used in this project is the scikit-learn package. All of the necessary machine learning algorithms we have worked with this summer are included in this package so we have used it in almost all of our scripts. Another tool we used is Orange [11], a Python data mining library with a graphical user interface. We only used Orange to create some visualizations but we found it useful for these purposes. For example, Orange has a colorful, easy to follow visualization of a decision tree and how it makes splits of the data. We used this image of the decision tree (see Figure 5) to better understand how the model works as well as to help others to learn about the model. All of these tools were used together to aid this research project.

4.2 Exploratory Data Analysis

To better understand the data that we collected from the Network Repository, we decided to do some Exploratory Data Analysis (EDA). EDA is the process of gaining a better understanding of a data set through summarizing its main characteristics, often using visualization tools. Our EDA process consisted of looking at some statistics (mean and range) of the features by the graph collections, then analyzing a correlation heat map of the features. Once we got a better sense of our data we moved on to some machine learning techniques.

When looking at the features of our graphs, we noticed that some graphs had a number of nodes and edges less than 100, but we also had graphs that had nodes and edges up to one million. Because of this, we decided to look at distribution plots of each of the features by collection, so that we could start understanding how each feature is distributed in each collection. By looking at these, we found that some collections had outliers that were either too big or too small to compare to the other networks. To get a better idea of how our data was arranged, we made tables (see Tables 3 and 4) expressing both the ranges and the means for the features across the collections. These summaries of the data led to a better overall understanding and insights that motivated future work. (See Section 5.1 for a further analysis).

A correlation heat map of all features, shown in Figure 10, can show how these can be correlated with each other. This correlation map was generated using the pearson correlation method, which measures the linear correlation between every two features. It gives a score between -1 and 1, where 1 is positive linear correlation, 0 is no linear correlation and -1 is negative linear correlation. For our project, we are looking for features that could differentiate one collection from the other, meaning features that have correlations closer to zero. If two features have a correlation greater or less than 0, we have redundant information and we can consider discarding one or more of those features. For this analysis, see Section 5.1.

4.3 Unsupervised Learning

Unsupervised learning makes use of unlabeled data to make predictions. Unsupervised learning methods are often used for clustering tasks as well as to understand the underlying patterns in data. The algorithms and techniques that we used in our research were:

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)** - Technique used to visualize high-dimensional data in 2 dimensions. It calculates the distance between the data in all 14 dimensions,

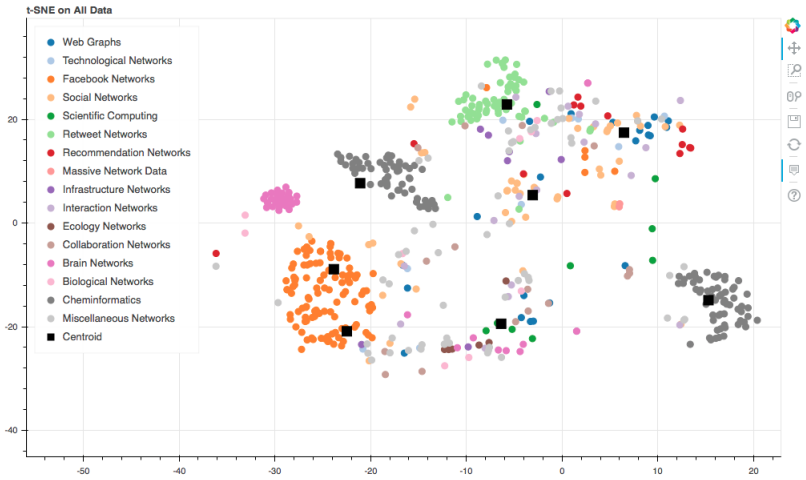


Figure 4: t-SNE on all data

and the reduces then dimensions to plot the approximate nearness of the data in an xy grid. [20] See Figure 13 for an example.

- **K-Means Clustering** - Algorithm that looks for similarities in the data. This algorithm creates k clusters and assigns each data point to a single cluster based on the features provided. [15]
- **Inertia plot** - A plot we use to determine the correct number of clusters, k, for our data when we run K-Means Clustering. We plot the inertia vs. the number of clusters and choose the value at the "elbow" of the plot (see figures 12 and 14 for examples). We choose to use the k value that corresponds to this shape in the plot because after the "elbow", the inertia value does not significantly decrease when adding more clusters.

T-SNE can be a helpful tool to visualize the structure of a data set, so we decided to run it on our data to see if any distinct clusters emerged. We started by running our data file through t-SNE with a learning rate parameter value of 1000, which we decided upon after a few experiments with different values (see Section 5.2.1). We then ran k-means clustering with 5 clusters on this plot to see how the data was arranged into groups. This first t-SNE was done before we downsampled the "Cheminformatics" section so we repeated this process on the data after we downsampled that collection as well. This second t-SNE was also generated using a learning rate of 1000 but this time we ran k-means with 8 clusters on it. These 8 clusters represented how the data was arranged in high dimensional space so it gave us some insight into how our supervised learning models might classify networks. A third t-SNE plot was generated, but in this case, k-means was calculated in all 14 dimensions of the data. The data including the clusters in 14 dimensions was passed through t-SNE and plotted (see Figure 18). This plot was not informative for our project, so we decided to not calculate k-means in all dimensions, but in the t-SNE generated 2 dimensional data (see Section 5.2 for further analysis).

After visualizing the clusters of the networks that were in a collection, we decided to run t-SNE again, but this time adding the 60 miscellaneous graphs that we had in our file (See Figure 4). By doing this process we wanted to start seeing how some of these miscellaneous networks would be clustered with other networks that already had a collection.

4.4 Supervised Learning

Supervised learning algorithms make use of a target variable to classify data into different categories. In our project, the target variable is be the network's collection. The models used in this project are:

- Decision Tree - Generates a series of binary splits using one feature at a time to classify data (See Figure 5)[13]

- Random Forest (RF) - An ensemble classifier that generates many Decision Trees, randomly selecting a subset of features to use in each one, and takes the mode of all of the trees' predictions. [13]
- Gaussian Naïve Bayes (GNB) - Generates probabilities for each possible classification of a data point, using the highest one as the prediction. This model assumes that all of the features used are independent and that the data is normally distributed between classes. [13]
- Support Vector Classification (SVC) - A support vector machine which finds the hyperplane that maximizes the distance to the nearest point in each class [13]
- Linear Support Vector Classifier (Linear SVC) - A support vector machine that uses linear algebra to transform and separate the data into distinct groups. It uses a one-against-all comparison, where it predicts a network's collection by asking if that network is from collection A or all of the other collections, and it does this with all of our collections. [22]
- Logistic Regression - a linear model for binary classification (in the multi-class case it uses a one-versus-rest scheme), minimizes a logistic function [23]

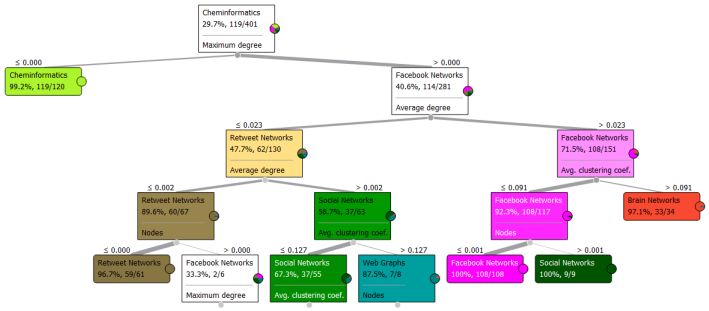


Figure 5: Example of a Decision Tree

To understand our models' predictions better, we looked at the overall accuracy as well as the accuracy within collections. When measuring the accuracy within a collection we used three types of scores: precision, recall, and f1-score. The precision in a category is the number of graphs correctly predicted into this category divided by the total number of graphs predicted into this category (also called positive predictive value). Recall is the number of graphs correctly predicted into this category divided by the total number of graphs of this category present in the model (also called sensitivity). The f1-score is the harmonic mean of precision and recall:

$$\text{f1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

To divide our data into train and test sets we used two different methods: Stratified K-Fold Cross Validation and Leave One Out Cross Validation (LOOCV). Cross Validation divides the data into different subsets, where one subset is defined as a test set and all the others as train sets. It then runs a model on these subsets giving an accuracy score, and changes the test and train subsets to again run the model. It does this until all subsets have been used as a test set, and then takes the average of all of the accuracy scores to give the overall accuracy. Stratified K-Fold Cross Validation divides the data in K subsets, while keeping the ratio of collections in each subset similar to the overall data's distribution. For this research we used $K = 5$, therefore the data was divided in 5 subsets and approximately 20% of the graphs from each collection were included in each subset. LOOCV takes only one data point as a test set and all the other data points as the train set in every iteration. LOOCV is better for our purposes because we have a small enough number of data points that it does not take too much time to run. Also, since we do not have many data points, using all but one of them to train the model ultimately leads to a model that can better classify the one test point. Although running this type of cross validation is time intensive, it often leads to a better evaluation of a model when sufficient data is available.

4.4.1 Additional Data Preparation

One problem we recognized was that we had many collections that contained only a few graphs. Since a machine learning model needs enough data points to properly represent the categories, we decided to only use the collections with size greater than 20 in our supervised machine learning models. With at least 20 instances in each collection, we believe the model has enough data points to learn how to accurately classify graphs into one of the categories. The six collections we used in our models are Cheminformatics, Facebook Networks, Retweet Networks, Social Networks, Brain Networks, and Web Graphs, seen in the blue box in Figure 6.

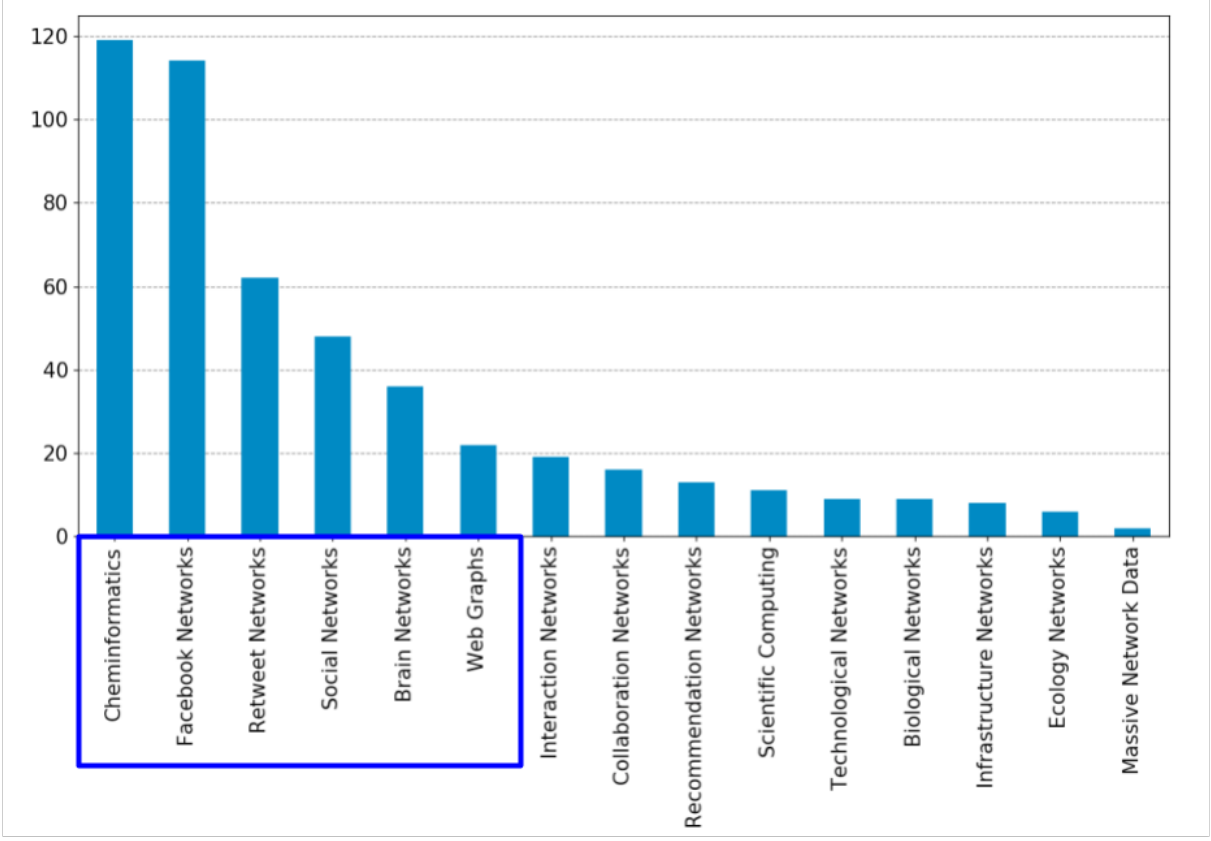


Figure 6: Data After Cleaning: blue box indicates collections used in supervised machine learning models

4.4.2 Feature Selection Algorithms

Before running these models we looked at different feature selection algorithms to find the most important features in our data, as well as the optimal amount of features to use in each model. We hypothesized that some of the features may be providing redundant information. This hypothesis was motivated through looking at the correlation matrix between the features (Figure 10and noticing that some of them are correlated so they are not independent. These correlated features could therefore be making our data set noisy and could be eliminated to lead to better results.

We first investigated Stability Selection, a type of feature selection algorithm that is given a model as input and outputs the most important features in that model. This algorithm takes a model, for example Randomized Logistic Regression, and runs it on a subset of the data and a subset of the features. It runs the model a few times changing the subset of the data and the subset of the features, deciding which are

the most important features in each iteration. After running the model many times, it looks for the most important features throughout the runs and assigns a score between 0 and 1 to each feature. A score of 1 indicates that the feature was the most important and a score of 0 means that it was the least important.

Next we used Lasso Regression, a machine learning model that can be used for feature selection because it shrinks the coefficient of the least important features to be exactly zero and gives a coefficient with a greater absolute value to those that were categorized as the most important.

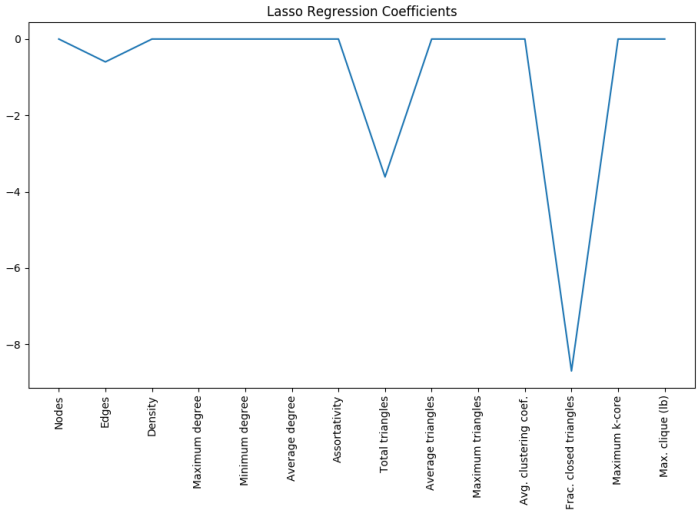


Figure 7: Lasso Regression Coefficients

Principal Component Analysis (PCA) is a dimension reduction technique that can be used to find the intrinsic dimension of a data set. It is similar to t-SNE in that it can be used to express data in a more compact way. Unlike t-SNE, which is primarily used for visualizations, PCA can eliminate noisy features and potentially make classification better. We experimented a little with this application of PCA to see if we could express our data with fewer features to make improvements in our models. We first ran PCA and plotted the variance vs. PCA feature to find the intrinsic dimension of our data set (see Figure 8). The intrinsic dimension is the number of PCA features with a significant variance. We found the intrinsic dimension of the data to be 5 features, but when we actually reduced the features we found that we lost too much information, so we decided not to use PCA further in our research. See Section 5.3.1 for more analysis.

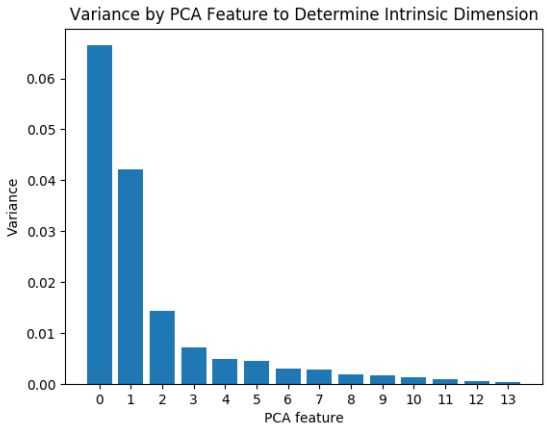
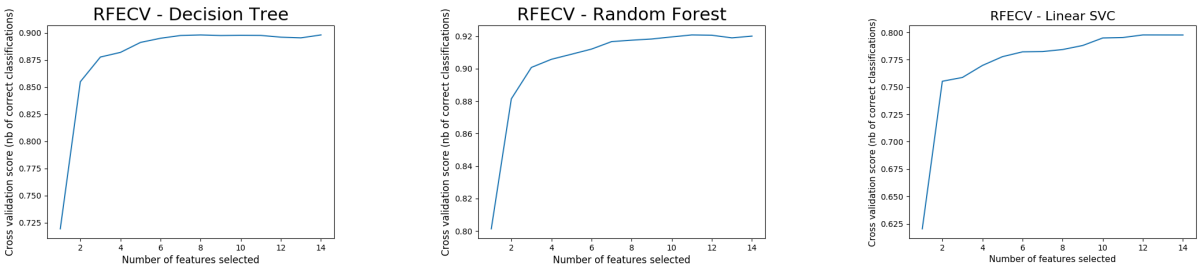


Figure 8: Variance by PCA Feature to Determine Intrinsic Dimension



(a) Decision Tree (b) Random Forest (c) Linear SVC

Figure 9: RFECV for different models

Recursive Feature Elimination (RFE) is a feature selection algorithm used with different models to obtain the most important features in each model. RFE takes a model, for example Decision Tree, and runs the model using all 14 features, then it eliminates the least important feature. RFE then runs the model again with 13 features and eliminates the least important feature. It keeps doing this until it has k most important features, where k is a given parameter. By setting $k = 1$, we can obtain a ranking of all the features, from the least important being ranked as 14, and the most important as 1. Some models used with RFE were: Decision Tree, RF, Linear SVC and Logistic Regression. Even though GNB is one of the models we use to train our data, this model couldn't be used with RFE because GNB assumes all features are independent, making them all 'important features'.

Recursive Feature Elimination with Cross Validation (RFECV) uses the same technique as RFE, the difference is that, every time it runs the model, it uses cross validation and assigns an accuracy score to that amount of features. This will also give us a ranking of the features, but this time it will report how accurately it predicts labels using that amount of features. This analysis will be conducted in Section 5.3.1.

Because we weren't able to run RFECV on GNB, we decided to run the models using different combinations of features to see if by reducing some of the features, the accuracy score improved. We wrote our own script that This is discussed in section 4.5.2.

4.5 Experiments

4.5.1 Experimental setup & data

Using the clean, scaled, downsampled data we began to test the following supervised machine learning classifiers: Decision Tree, RF, GNB, Support Vector Classifier (SVC), Linear Support Vector Classifier (Linear SVC), and Logistic Regression. As noted in section 4.4.1, we only included collections with more than 20 graphs.

4.5.2 Fine-Tuning Models

We adjusted the parameters of the models in order to improve their accuracy. For the RF classifier, we tested the parameter that defines how many trees are generated, and found the best accuracy when the model generated around 20 to 40 trees. The change of accuracy within this range was minimal, so we decided to use the RF model with 30 trees.

Unlike the other classifiers, GNB performed better when we reduced the number of features used. After manually testing the models while leaving out one feature at a time, we found that the GNB model improved when we removed minimum degree, maximum degree, assortativity, and total triangles. Starting by removing these features and calculating the resulting average accuracy, we wrote code that recursively tests the model by adding or removing each feature one at a time. It was a greedy optimization process by keeping (or losing) the feature that improves the accuracy the most. Through this process, we found that the best score came from only including nodes, average clustering coefficient, fraction of closed triangles, maximum k-core, density, maximum clique, average degree, and assortativity. See Section 5.3.1 for further analysis.

4.5.3 Testing Models

To test the accuracy of these models we implemented stratified 5-fold cross validation, stratifying over the graph categories. We also used leave one out cross validation, which almost always resulted in higher accuracy. Because of the relatively small amount of data, we ran each cross validation 100 times and took the average score to account for variability in results (see section 5.4.1). The best results came from the GNB, RF, Decision Tree, and Linear SVC models. SVC and Logistic Regression models did consistently poorly.

We originally tested the models using a variety of collections. We used all collections first, then removed collections with less than 10 graphs. See Section 5.4.1 for further analysis.

4.5.4 Synthetic Networks

Previous research has been done using machine learning to classify synthetic networks [8] [9] [21]. Although our approach is different because of our use of real-world networks, we wanted to test if our models are capable of discerning between synthetic networks as well. We generated 125 graphs: 50 using the Barabasi-Albert (BA) model [5] and 75 using the Erdős-Rényi (ER) model [12]. We selected a few different graph sizes that would be similar to the sizes of graphs already in our data set. 25 of the Barabasi graphs had 1000 nodes and the other 25 had 10000 nodes. The Erdős-Rényi graphs had 25 graphs of each of those sizes, as well as another 25 of size 100000. We selected values for the parameters of these graph models in order to create graphs that would not be obviously completely different from the real-world networks in our data. To select p, the probability that an edge exists between two nodes in the ER model, we looked at the densities of different sizes of graphs and chose p values that would generate graphs of similar densities. For graphs with 1000 nodes we used p values of 0.05, 0.1, and 0.2; for graphs with 10000 nodes we used p values of 0.0005, 0.005, and 0.001; and for graphs with 100000 nodes we used p values of 0.0005, 0.00005, and 0.000005. To select values for m, the number of edges that should exist from a node to the other nodes in the graph in the BA model, by looking at the average degrees and number of edges in our real-world data and choosing values that would make the synthetic networks "blend" in with the data. For graphs with 1000 nodes we used m values of 10, 40, and 60; and for graphs with 10000 nodes we used m values of 40, 60, and 100.

We added these synthetic networks to our data as two categories: Synthetic Barabasi and Synthetic Erdős Rényi. We used LOOCV to test the accuracy of the RF and GNB classifiers with this new data. Results from this experiment are in Section 5.4.3.

4.5.5 Combining Collections

In order to include more collections in our models, we tested grouping smaller, similarly labeled collections together. We combined Web Graphs and Technological Networks into a new collection called Web-Tech. We did the same with Brain and Biology Networks into Brain-Bio, and Scientific Computing and Ecology Networks into Sci-Eco. We tested the RF and GNB classifiers each with combination of these new collections using LOOCV. Results from this experiment are in Section 5.4.4.

4.5.6 Miscellaneous Networks

To be able to test how accurately RF and GNB could predict the collections of the miscellaneous networks, a collection hypothesis was created for each of graph in the miscellaneous collection. Of the 60 miscellaneous networks, one had NaN entries, so it was removed from the file. All 59 of the networks were searched for in the Network Repository to find a description of the graph and its' features. Some of the graphs found were well described in the NR, but some of them were either very vague descriptions, or there was no description at all. Because of this, our search was extended into other repositories online, to see if some new information of the graph would be found. In this process the repositories used were Stanford Network Analysis Project (SNAP) ([18]), Konect [16] and Index of Complex Networks (ICON) [2]. These repositories had some of the miscellaneous graphs categorized into a collection, including some collections not included in our data. This was taken into consideration when creating the collection hypothesis for these graphs.

After analyzing and going through all of the miscellaneous graphs, we found that some of the graphs were duplicated, but had a different graph name. One copy of these duplicated graphs was kept and the

others removed. We also found some miscellaneous graphs that were duplicates of graphs that were already categorized into a given collection on the NR. We removed these graphs from the Miscellaneous Networks collection, ending with 50 miscellaneous graphs, all with a collection hypothesis.

Training the RF and GNB models on the graphs with specific collection labels, including all three combined collections mentioned in Section 4.5.5, we predicted the collection labels of the Miscellaneous Networks and recorded accuracy based on the collection hypotheses. Results from this experiment are in Section 5.4.5.

5 Analysis

5.1 Exploratory Data Analysis

At the beginning of our exploratory data analysis process we analyzed tables of the features across the different collections and noticed some patterns in the data. For example, Brain networks have many more edges than the other collections, but the assortativity feature was similarly distributed among the collections. These findings indicate that features that have a lot of variation, like edges for example, will be useful to separate the different collections, while the features that are similar across collections might not be as useful to our algorithms.

Collection	Nodes	Edges	Density	Maximum Degree	Minimum Degree	Average Degree	Assortativity
Brain Networks	29.0 - 810505.0	44.0 - 171005692.0	0.0004 - 0.9983	9.0 - 17749.0	1.0 - 98.0	2.0 - 1073.0	-0.8447 - 0.5876
Cheminformatics	3.0 - 74.0	6.0 - 240.0	0.0889 - 2.0	2.0 - 9.0	0.0 - 0.0	2.0 - 4.75	-0.5 - 0.0
Facebook Networks	769.0 - 59216211.0	2981.0 - 92522012.0	0.0 - 0.057	248.0 - 8246.0	1.0 - 1.0	2.0 - 116.0	-0.6682 - 0.1969
Retweet Networks	96.0 - 1112702.0	117.0 - 2278852.0	0.0 - 0.0257	17.0 - 31569.0	1.0 - 1.0	2.0 - 6.0	-0.8061 - -0.0182
Social Networks	16.0 - 4033137.0	58.0 - 117185083.0	0.0 - 0.4833	10.0 - 106218.0	0.0 - 3.0	3.0 - 76.0	-0.8764 - 5.4684
Web Graphs	643.0 - 2140198.0	2280.0 - 17783332.0	0.0 - 0.011	59.0 - 127090.0	1.0 - 1.0	3.0 - 181.0	-0.3041 - 0.9969

Collection	Total Triangles	Average Triangles	Maximum Triangles	Avg. Clustering Coef.	Frac. Closed Triangles	Maximum k-Core	Maximum Clique
Brain Networks	3.0 - 109030181352.0	0.0622 - 722311.8606	1.0 - 17512230.0	0.0213 - 3.8729	0.0047 - 0.7675	3.0 - 2195.0	2.0 - 255.0
Cheminformatics	1.0 - 159.0	0.27 - 4.31	0.5 - 11.5	0.034 - 0.759	0.092 - 0.758	2.0 - 4.0	1.0 - 2.0
Facebook Networks	273.0 - 166819284.0	0.0945 - 2353.5101	52.0 - 198991.0	0.0272 - 0.4093	0.0004 - 0.2913	4.0 - 117.0	4.0 - 24.0
Retweet Networks	6.0 - 525939.0	0.0013 - 20.4267	1.0 - 47410.0	0.0006 - 0.1274	0.0 - 0.0742	3.0 - 80.0	2.0 - 44.0
Social Networks	135.0 - 1882752543.0	0.219 - 3717.0742	17.0 - 3723314.0	0.0144 - 0.5706	0.0006 - 0.5271	5.0 - 569.0	5.0 - 87.0
Web Graphs	3144.0 - 2513657160.0	0.7361 - 19390.7149	103.0 - 1341850.0	0.0708 - 0.9884	0.0012 - 0.9997	7.0 - 18135.0	4.0 - 500.0

Table 3: Table of Ranges

Analyzing the correlations between the features can provide insights into our data and what features may be more important than others. Figure 10 shows a correlation heat map of all features in this project. If two features are highly correlated they may contain redundant information. For example, the edges feature is highly correlated with total triangles, as we see in the red box in the matrix location for “Edges” and “Total triangles”. The dark red color shows that the correlation between these edges is high. Out of all 14 features, the ones that were most highly correlated with other features were: edges, average degree, average triangles, total triangles, maximum triangles, average clustering coefficient and maximum clique. These had correlation greater than 0.5 with at least two other features. It makes sense that these features would be correlated, as a graph with more edges should also have a higher average degree, given that number of nodes is constant. Also, networks with large clustering coefficients and maximum cliques should contain more triangles, thus leading to a correlation between those features and the features that have to do with triangles. This finding suggests that if features should be removed, we can remove some of the highly correlated ones because they contain similar information.

Collection	Nodes	Edges	Density	Maximum Degree	Minimum Degree	Average Degree	Assortativity
Brain Networks	532992.4444	108536303.3889	0.1129	7688.5833	4.7778	335.8611	0.22
Cheminformatics	32.7479	126.3697	0.3128	6.1176	0	3.8507	-0.0464
Facebook Networks	1101107.7281	2494041.8684	0.0112	1932.2807	1	72.6754	0.0509
Retweet Networks	30265.8387	55532.0484	0.0009	2428.3871	1	2.1774	-0.3266
Social Networks	574078.8542	7801877.5312	0.0202	15316.6875	0.9375	19.3929	0.0688
Web Graphs	553148.9091	4394887.3636	0.0011	26775.0909	1	20.1364	0.0351

Collection	Total Triangles	Average Triangles	Maximum Triangles	Avg. Clustering Coef.	Frac. Closed Triangles	Maximum k-Core	Maximum Clique
Brain Networks	43386366506.9722	79596.5309	4204822.5556	0.7075	0.3406	889.4722	129.5833
Cheminformatics	70.7647	2.2764	5.6723	0.3962	0.3887	2.9916	1.7731
Facebook Networks	14086131.7193	873.373	38401.9386	0.2336	0.1564	67.4561	11.1667
Retweet Networks	21419.9839	0.5414	1119.6452	0.0182	0.0048	6.7419	4.7581
Social Networks	159581861.3438	277.7122	385206.7083	0.1753	0.0796	111.4792	26.5
Web Graphs	220324837.3182	1088.2134	289793.0909	0.4141	0.2405	976.7727	76.7727

Table 4: Table of Means

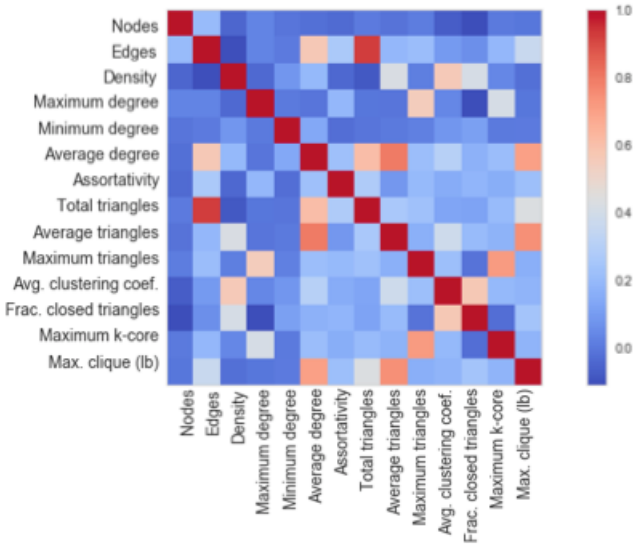
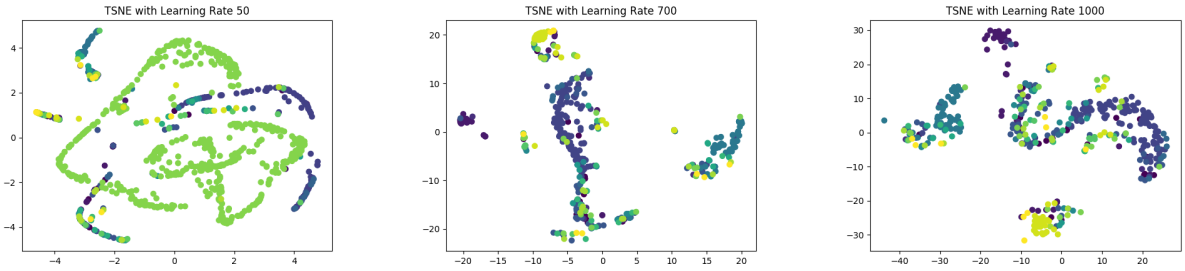


Figure 10: Heat Map of feature correlations



(a) Learning Rate = 50 (b) Learning Rate = 700 (c) Learning Rate = 1000

Figure 11: t-SNE plots with different learning rates

5.2 Analysis of Unsupervised Learning

5.2.1 t-SNE Cluster Analysis

T-SNE plots can be generated using different learning rates to get different pictures. To better understand which was the optimum learing rate we created a few different t-SNE plots (see Figure 11). We used different learning rates, ranging from 50 to 1,000, so that we could see how different the plots were from one another. The learning rate in t-SNE can be an important parameter that will vary depending on the data. We decided that for further plotting with t-SNE we would use a learning rate of 1,000 because it created what we understood to be the best visualization.

We also decided to explore using a different distance metric in our t-SNE to see if we could get a better visualization. We had read about people using the Canberra distance metric to compare the feature vectors of graphs [8] so we decided to try it with our data. When we ran the t-SNE with the Canberra distance as the metric used in the algorithm we did not get better results than when we ran it with the default Euclidean distance metric. In fact, we found the visualizations with the default metric showed more easily the clustered collections in the data, so we decided to use that metric going forward.

After settling on a learning rate and a distance metric for use in our t-SNE plots we ran our data through t-SNE again and saw a few clusters emerge in the plot. To understand the clusters in the t-SNE plot we decided to run k-means on it and add the cluster centroids to the plot. To determine how many clusters to use in K-Means we created an inertia plot of inertia vs. k to get the optimal amount of clusters (see Figure 12). Inertia can be recognized as a measure of how coherent the clusters are. A lower inertia value indicates that the items in the cluster are very close together, so that cluster is very internally coherent.

We then used 5 clusters in K-Means and plotted the data to visualize the clusters. In Figure 13, we can see how the majority of the data points are from Cheminformatics, and there are a few other clusters of distinct collections, like Facebook and Brain Networks. An interesting finding was that the Cheminformatics graphs were being divided into multiple distinct clusters. This plot showed us the overwhelming amount of Cheminformatics graphs in our data set.

Analyzing this plot partially motivated us to remove many of the Cheminformatics graphs. After we downsampled this collection, we decided to run t-SNE using that data. The inertia plot for this new data is presented in Figure 14. We then used 5 clusters in K-Means and plotted the data. We found that only using 5 clusters didn't really determine all of the visibly different clusters shown in figure 15. Because of this, we decided to use 8 clusters instead of 5, and that way we could verify if every visible cluster was divided by K-Means (see Figure 16). Looking at this picture we can see that collections like Facebook and Brain were being clustered very tightly together, while others, like Web and Technology, were being clustered together. This finding led us to understand that there are some underlying patterns in our data that are actually dividing some of the collections from the others. We also experimented on obtaining the 8 k-means clusters on the original 14 dimension data, and then plotting in 2 dimensions using t-SNE. Figure 18 shows how the clusters seemed to be a bit distorted, and the k-means centroids (black squares) weren't as clear as before. Because of this unclear picture, we decided against obtaining the k-means clusters in all dimensions.

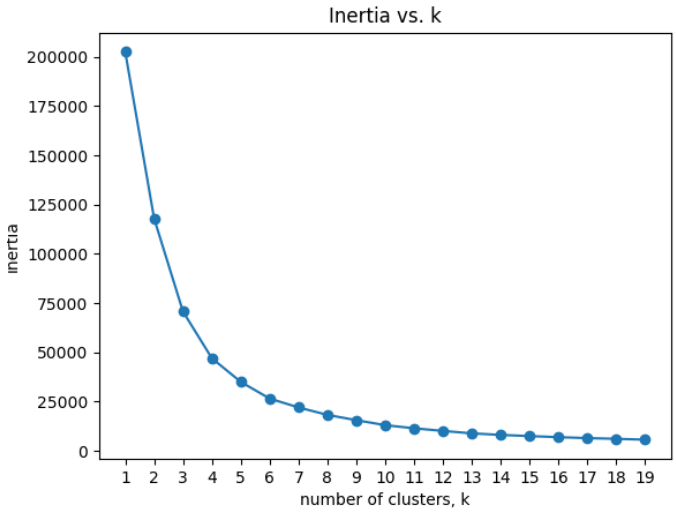


Figure 12: Inertia Plot for data with all of Cheminformatics

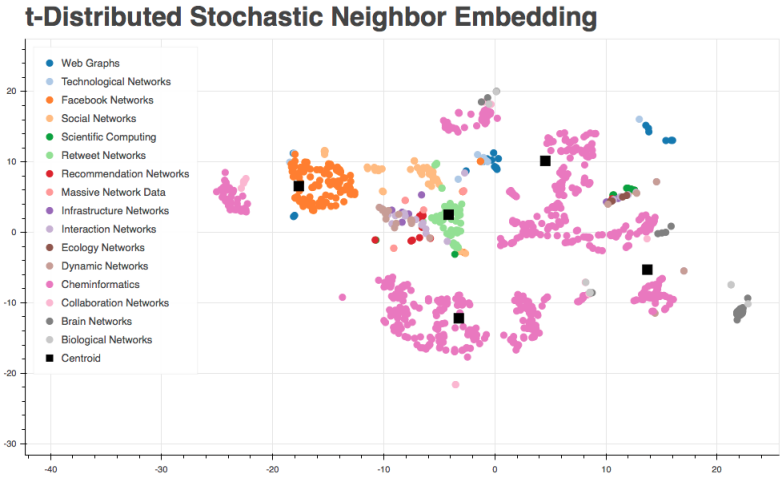


Figure 13: t-SNE using all of Cheminformatics

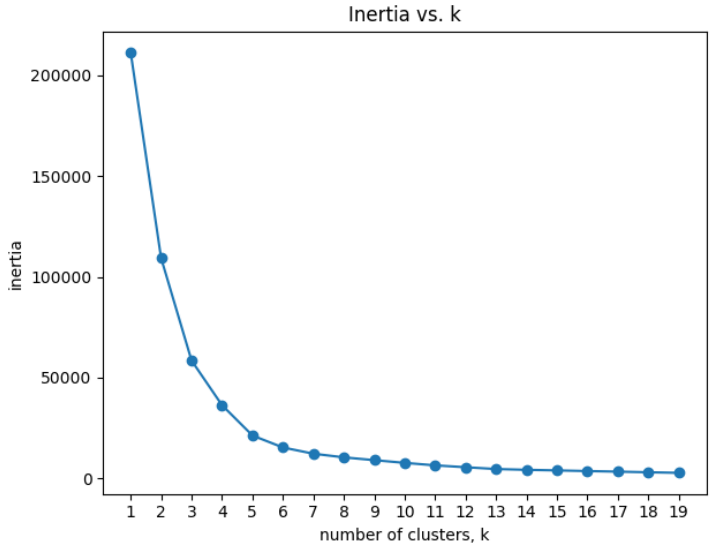


Figure 14: Inertia Plot for data with Downsampled Cheminformatics

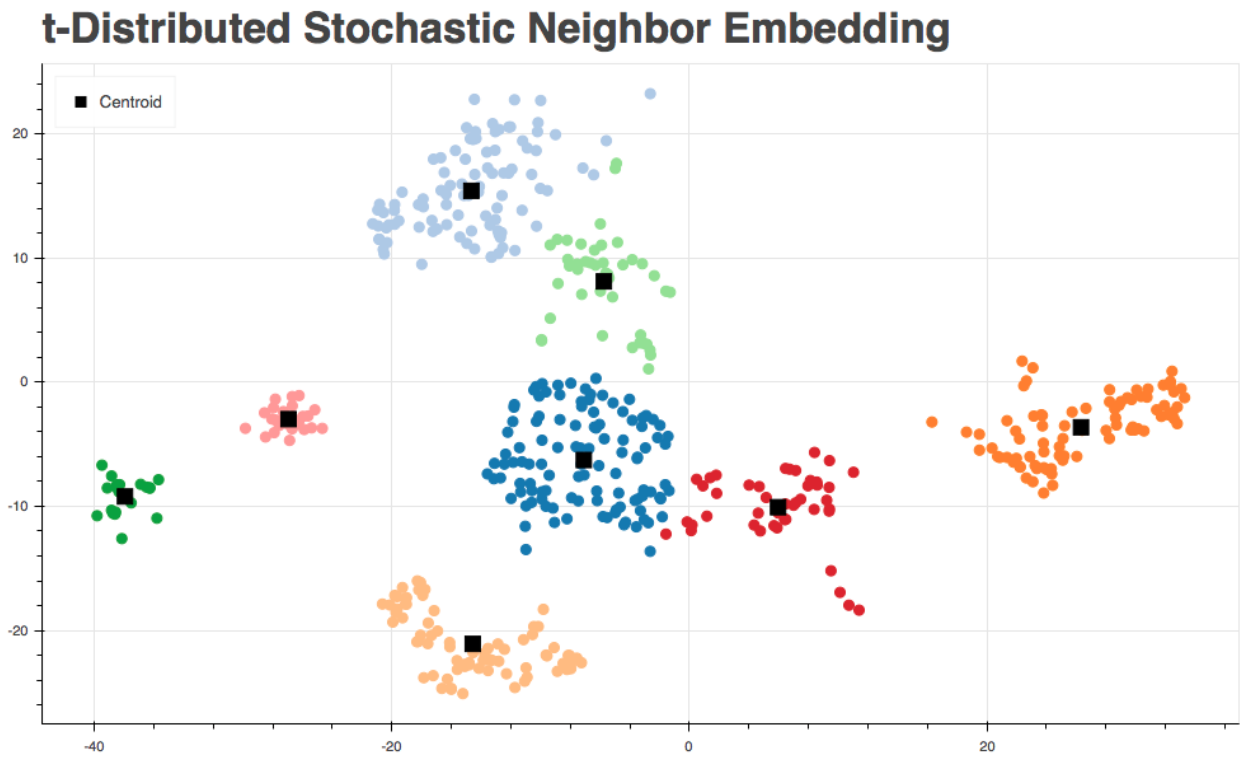


Figure 15: t-SNE plot colored by cluster

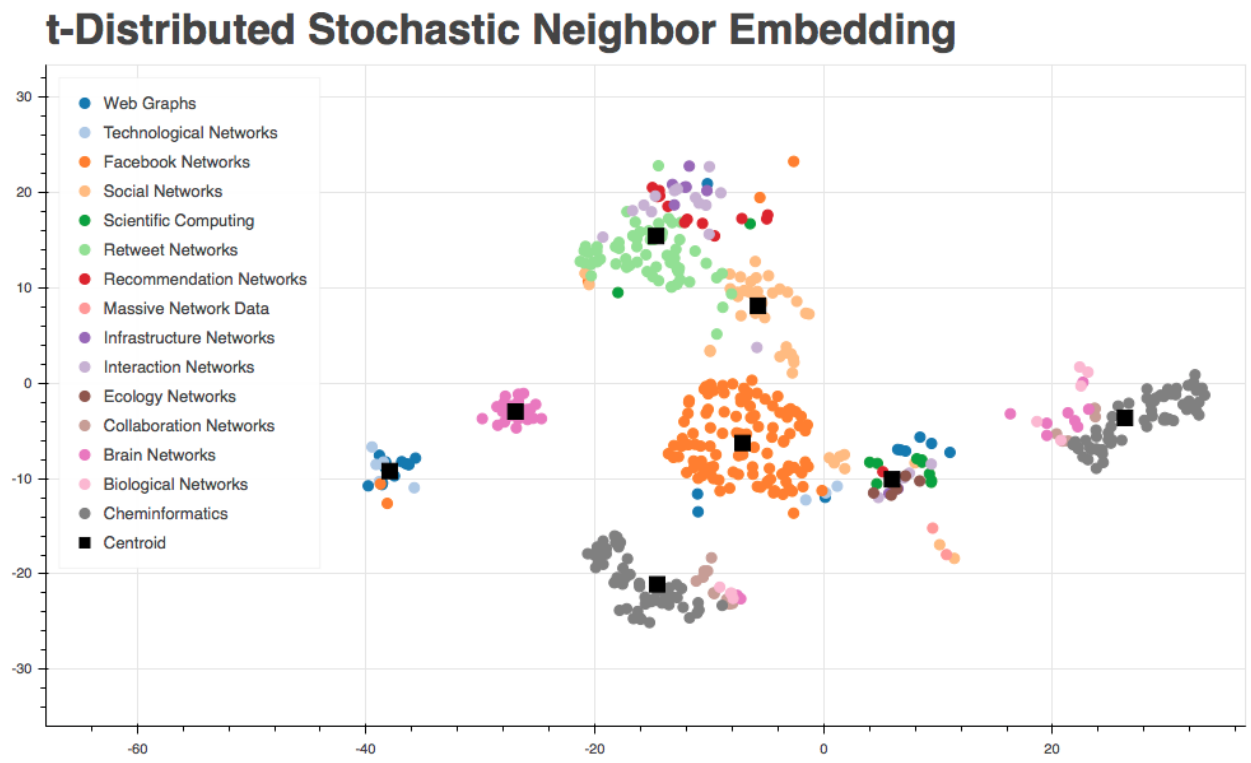


Figure 16: t-SNE using Downsampled Cheminformatics

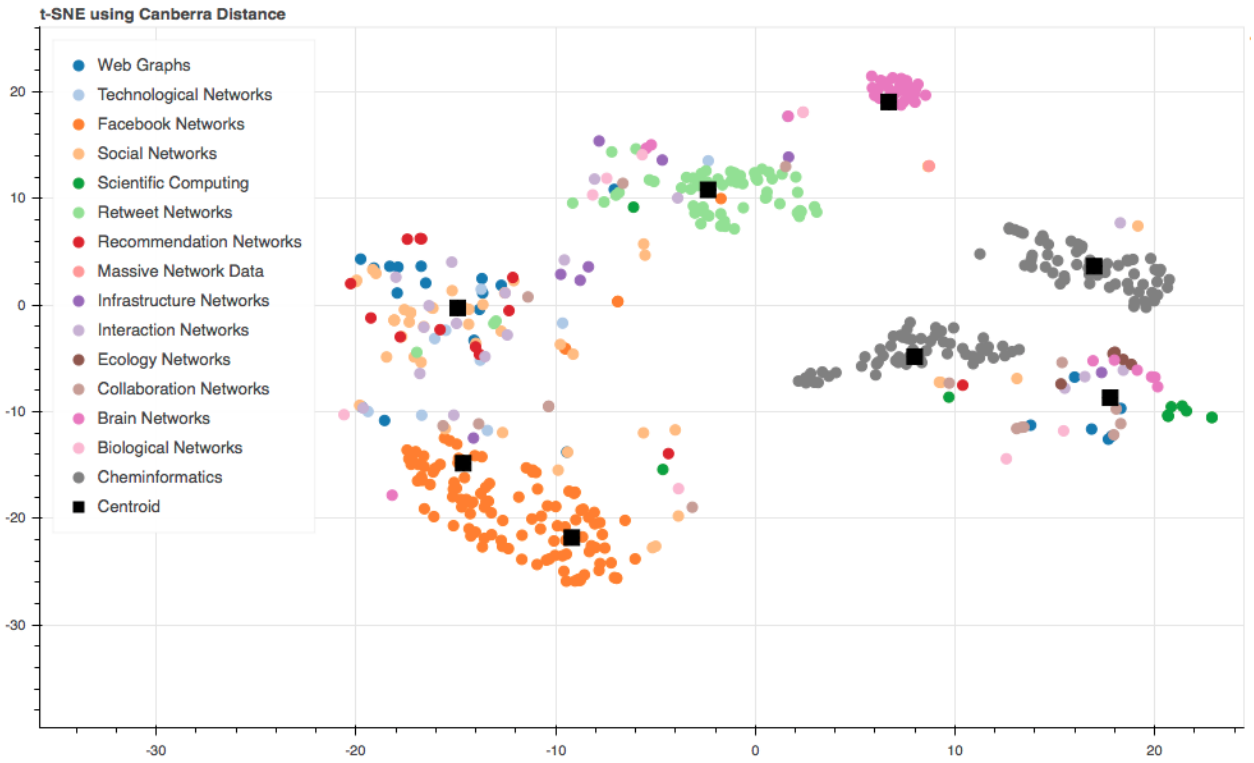


Figure 17: t-SNE using Canberra distance

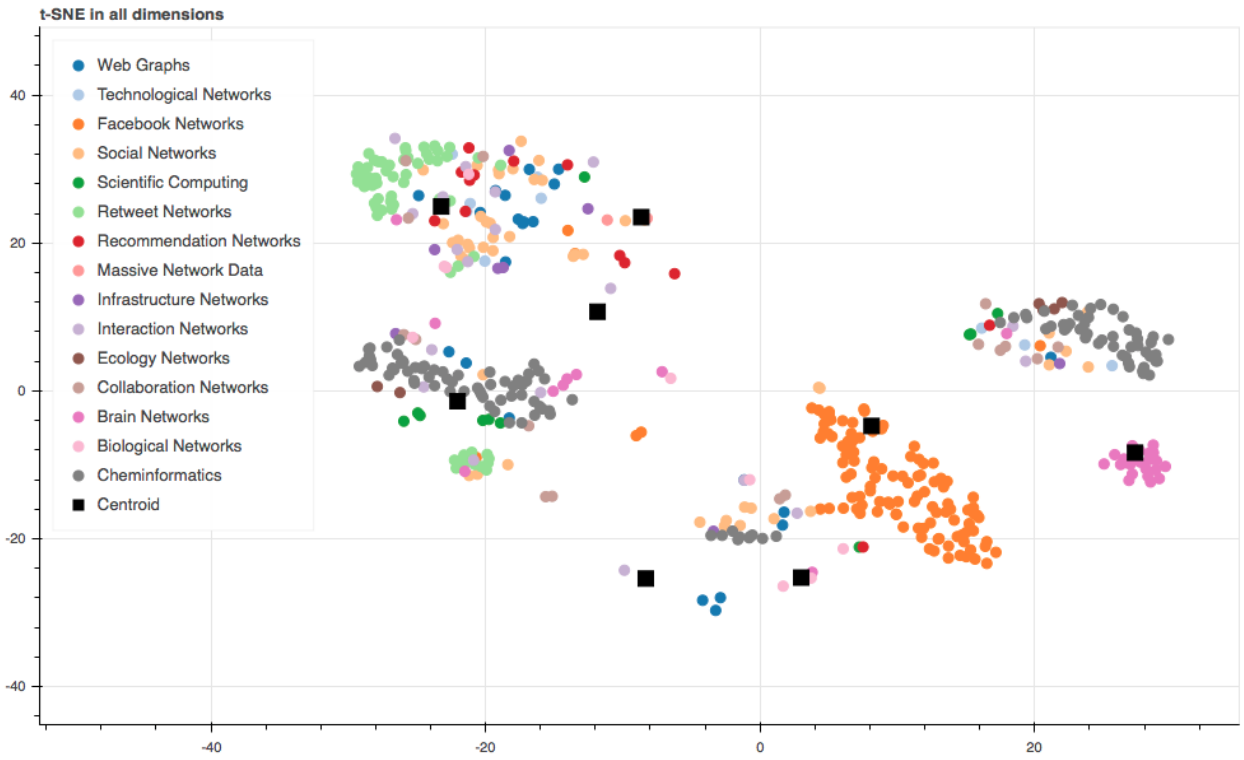


Figure 18: t-SNE using k-means in 14 dimensions

Looking back at figure 16, this figure shows a plot which includes 8 different clusters. To better understand what is going on inside of each one of these clusters, we decided to create box and whisker plots for each cluster (see Figure 19). These plots will help us understand why different collections were clustered together, and why others were completely isolated of every other data point. Figure 5 shows how the graphs were divided into the different clusters.

- Cluster 0 - Composed only of 26 Brain Networks. Had the highest median for edges than any other cluster, highest median for average degree and the only distinctive range for all features with triangles.
- Cluster 1 - Composed of 108 Facebook Networks and 2 Web Graphs. Had the second highest median from average degree and assortativity.
- Cluster 2 - Composed mainly of 57 Retweet Networks, with a few graphs from 7 other collections. Had the lowest range of scores for average clustering coefficient, fraction of closed triangles and average degree. Had the widest range of scores for assortativity.
- Cluster 3 - Composed of 62 Cheminformatics networks, about half of them, and a few Brain and Biology Networks. Had no range for minimum degree because all values were 0, and had the highest median for average clustering coefficient and fraction of closed triangles.
- Cluster 4 - Composed mainly of 57 Cheminformatics netwokrs, about half of them, and a few Brain and Biology Networks. Had no range for miminum degree because all values were 0, and had the second highest median for density.
- Cluster 5 - Composed mainly of 37 social networks and a few other graphs from 3 different collections. Had the second lowest median for average clustering coefficient.
- Cluster 6 - Composed of about 12 Web Graphs and 2 Facebook Networks. Had the widest range of values for nodes and maximum degree.

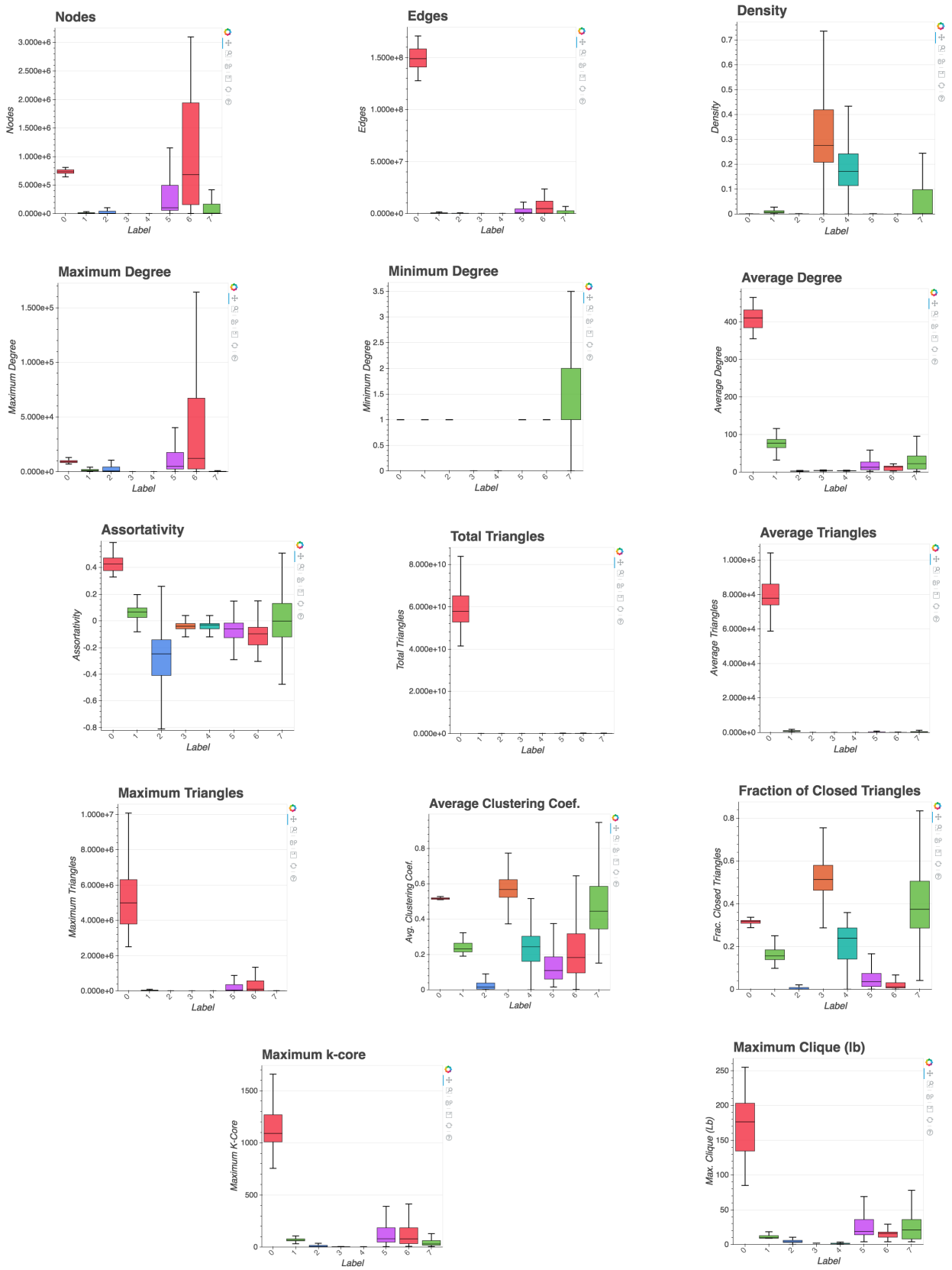


Figure 19: Boxplots of Features by k-Means Label

	Labels							
	0	1	2	3	4	5	6	7
Biological Networks	0	0	0	5	4	0	0	0
Brain Networks	26	0	0	8	2	0	0	0
Cheminformatics	0	0	0	62	57	0	0	0
Collaboration Networks	0	0	0	6	10	0	0	0
Ecology Networks	0	0	0	0	0	0	0	6
Facebook Networks	0	108	3	0	0	0	2	1
Infrastructure Networks	0	0	6	0	0	0	0	2
Interaction Networks	0	0	13	0	0	1	0	5
Massive Network Data	0	0	0	0	0	0	0	2
Recommendation Networks	0	0	10	0	0	2	0	1
Retweet Networks	0	0	57	0	0	5	0	0
Scientific Computing	0	0	2	0	0	0	0	9
Social Networks	0	0	3	0	0	37	0	8
Technological Networks	0	0	0	0	0	0	6	3
Web Graphs	0	2	1	0	0	0	12	7

Table 5: Cross tabulation of K-Means with t-SNE data

- Cluster 7 - Composed mainly of a few graphs from 10 different collections, without there being one dominant collection. Had the widest range of values for minimum degree, average clustering coefficient and fraction of closed triangles.

5.2.2 t-SNE including Miscellaneous Networks Analysis

T-SNE was used again to find underlying patterns in the data, but this time, the miscellaneous networks were included in the file. Figure 4 shows that the clusters are a bit more distorted than when we ran t-SNE without this addition; however, some of the distinct clusters of collections, like Facebook and Brain networks, are still clear. One interesting finding is how some miscellaneous networks are being clustered with different collections, indicating that those networks belong to those collections. In section 5.4.4 a collection hypothesis was created for each miscellaneous graph.

5.3 Analysis of Supervised Learning

5.3.1 Feature Selection Analysis

In this section we discuss the analysis that led to our decisions in feature selection. First, in examining a plot of the coefficients for the features we can see that the Lasso Regression identified the features fraction of closed triangles, total triangles, and edges as the most important features in our data set)See Figure 7).

In order to see whether or not we should reduce features using PCA we looked at the plot of the variances vs. PCA features (see Figure 8). We decided that the cut off point for significant variance was about 0.005, so the intrinsic dimension of our data set is 5. To check if our data set could be represented in only 5 dimensions, we ran PCA on our data and reduced it to 5 dimensions. We do not have the ability to visualize 5 dimensions at once, so we decided to run t-SNE on the PCA-reduced features to see if we got similar clustering behavior (See Figure 20). Although this plot still had a few clusters, this t-SNE image was much less distinctly clustered than our original, so we lost a lot of the structure of the data through reducing dimension using PCA. We also tried plotting two PCA features at once to see if any patterns emerged, but the plots were not highly informative so we abandoned them (See Figure 21). In the end we decided not to use PCA to reduce the features in our data set.

Instead of PCA, we decided to use the feature selection tool RFECV to further analyze whether or not to remove certain features. We ran it 100 times with each of the models (Decision Tree, RF and Linear SVC). We were not able to run GNB with RFECV because this probabilistic model does not have the attributes that RFECV needs to rank each feature and give an acuracy score for each amount of features. When running it 100 times with all three of the models stated above, we can see an increase in the accuracy score

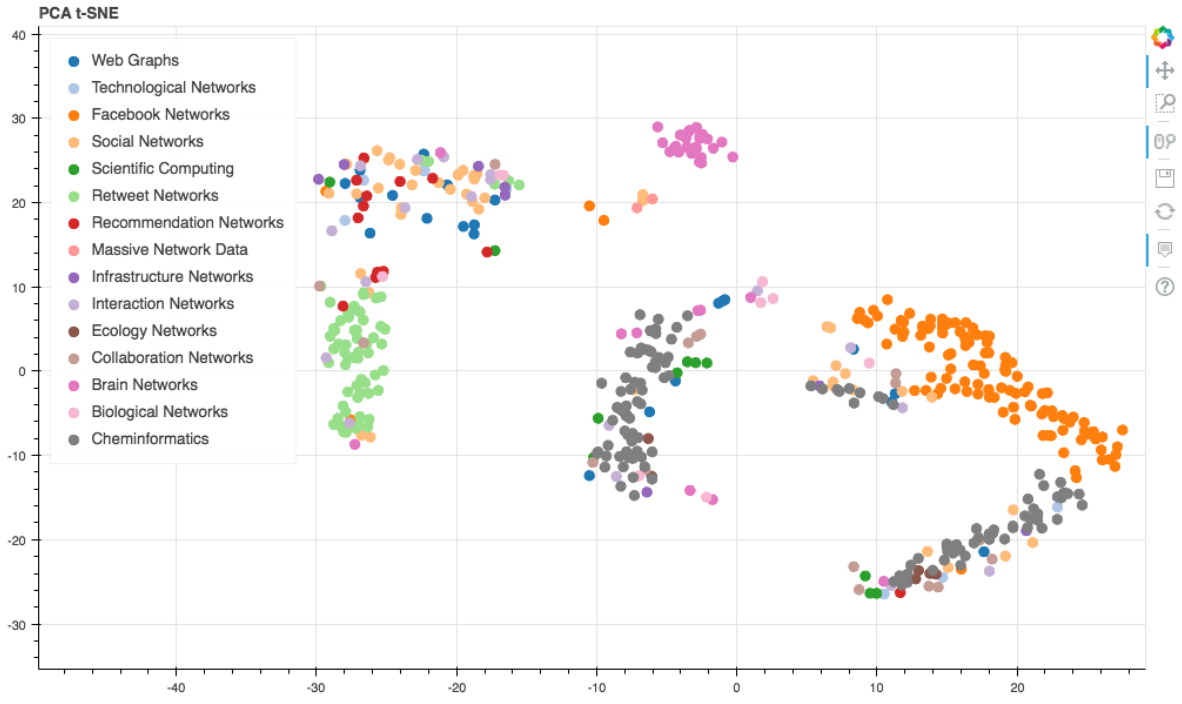


Figure 20: t-SNE with 5 PCA Reduced Features

every time we add one more feature (see Figure 4.4.2, 4.4.2 and 4.4.2). Even though the line starts to plateau after 4 or 5 features in the model, we decided to use all 14 features to obtain the best accuracy score for our models.

By running our recursive function on the GNB model, we found that removing the features number of edges, maximum degree, minimum degree, total triangles, minimum triangles, and average triangles resulted in the best accuracy score.

This specific group of features intuitively makes sense to eliminate because of how the GNB classifier works. In the process of generating the probability of classification predictions, this model assumes that all features used are independent. By removing the features listed above, only one feature related to degree remains in the model (average degree), and one feature related to triangles remains (fraction of closed triangles) in the model. It is interesting to note that our recursive code did not come to this result every time that we ran it. If we started with too few features, or too many, the code would finish with a different list of features and a worse final cross validation score. This suggests that specific combinations of features together have a significant impact on how well the GNB classifier works.

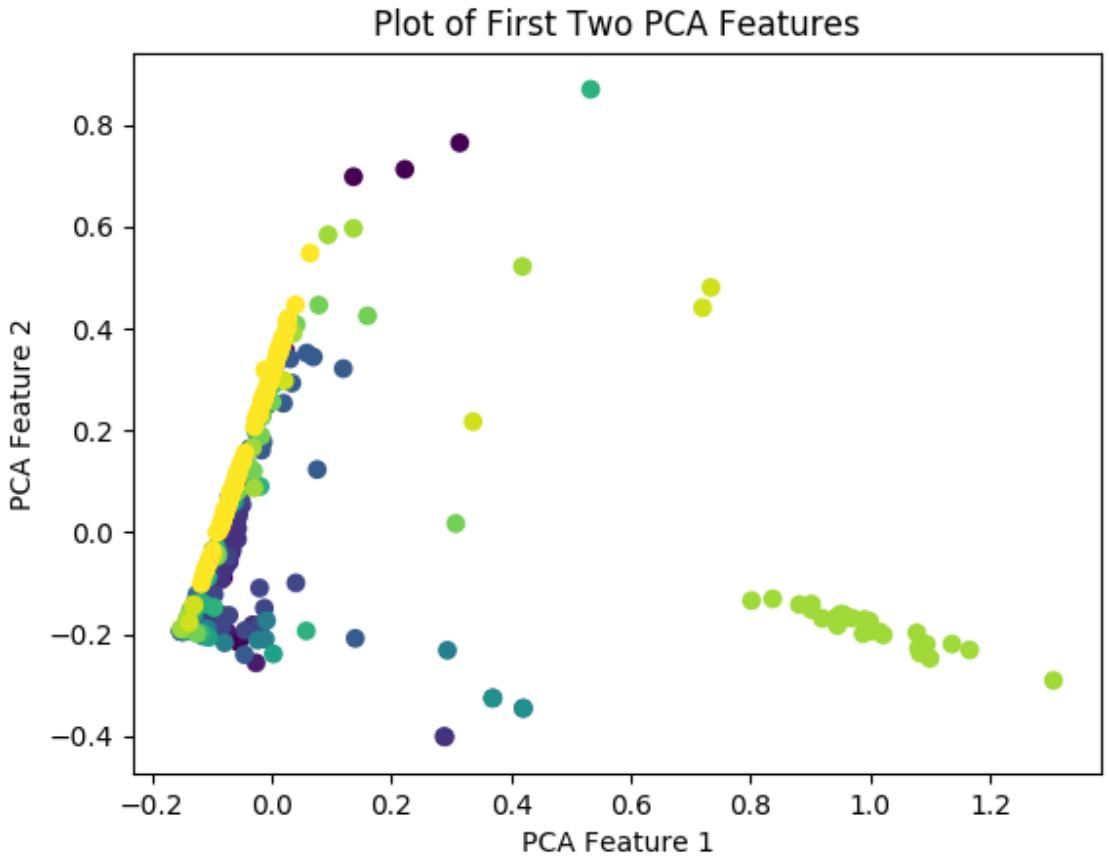


Figure 21: PCA Feature 2 vs. PCA Feature 1

5.4 Analysis of Experiments

5.4.1 Testing Models Analysis

Table 6 shows the average stratified 5-fold and Leave One Out cross validation scores for the six models after testing each model 100 times, along with the standard deviations of these scores. For all models except SVC, using LOOCV resulted in a higher accuracy than stratified 5-fold cross validation. The best results come from GNB, RF, Decision Tree, and Linear SVC, while Logistic Regression and SVC do poorly. A possible reason for the disparity between the Linear SVC and SVC scores is that Linear SVC implements a One-Against-All classification method, while SVC uses a One-Against-One method. It is also important to note that Decision Tree and RF implement randomness while training, which is why these two models have a non-zero standard deviation using LOOCV.

		Cross Validation	
		Stratified 5-fold	Leave One Out
Models: Gaussian Naïve Bayes	Accuracy	91.5%	92.8%
	Standard Deviation	0.794%	0.000%
Random Forest	Accuracy	91.7%	92.3%
	Standard Deviation	0.750%	0.489%
Decision Tree	Accuracy	89.6%	91.4%
	Standard Deviation	1.164%	0.554%
Linear SVC	Accuracy	79.8%	80.3%
	Standard Deviation	0.453%	0.000%
Logistic Regression	Accuracy	63.9%	64.8%
	Standard Deviation	0.345%	0.000%
SVC	Accuracy	55.7%	52.1%
	Standard Deviation	0.330%	0.000%

Table 6: Overall accuracy scores for each model using collections with more than 20 instances

In order to understand how the models were classifying each specific collection, we tested the top four models 100 times and took the average of the resulting f1-scores from each collection using LOOCV (Table 7). The f1-scores for Brain, Cheminformatics, Facebook, and Retweet collections are consistently strong, while the models do a poorer job in the Social and Web collections. This could be partially due to the fact that the Web Graphs category only contains 22 networks, barely making the 20 graph cut-off. Note that the score of 0% for Web Graphs in Linear SVC means that this classifier never correctly labeled a graph in that category.

	f1-score					
Models	Brain	Cheminformatics	Facebook	Retweet	Social	Web
Gaussian Naïve Bayes	90.4%	99.6%	97.3%	92.9%	72.9%	61.1%
Random Forest	92.3%	99.5%	97.8%	92.9%	76.4%	56.3%
Decision Tree	86.5%	99.1%	97.0%	92.4%	75.4%	62.2%
Linear SVC	87.5%	92.5%	88.8%	78.5%	11.3%	0%

Table 7: f1 scores for top models

We noticed that the RF and GNB classifiers generally performed very well, with similar accuracy scores. In order to get a better picture of how these two models were behaving we looked into the confusion matrices. Table 8 and Table 9 display the confusion matrices for one Leave One Out cross validation run of both models. It is important to note that these tables only show one instance of the cross validation, and that the RF results vary with each run. We can see that the two models make similar predictions for each collection, except GNB performs better with Web Graphs.

We also trained and tested the top four models using some of the smaller collections to compare results. We first used all collections, and then only used collections containing more than 10 graphs. The resulting accuracies are in Table 10. It is interesting to note that while the GNB model performs as well as or better

Actual \ Predicted	B	C	F	R	S	W
Brain	32	1	1	1	1	0
Chem	0	119	0	0	0	0
Facebook	0	0	112	1	1	0
Retweet	0	0	0	57	3	2
Social	0	1	1	1	40	5
Web	1	0	1	1	10	9

Table 8: Random Forest Confusion Matrix

Actual \ Predicted	B	C	F	R	S	W
Brain	33	0	0	1	2	0
Chem	1	118	0	0	0	0
Facebook	0	0	111	1	2	0
Retweet	0	0	0	59	3	0
Social	1	0	1	3	40	3
Web	2	0	2	1	6	11

Table 9: Gaussian Naïve Bayes Confusion Matrix

than the other models on collections with more than 20 graphs, the RF and Decision Tree models perform better than the other models when including smaller collections. This suggests that GNB is a stronger classifier when using clear, coherent categories, while RF and Decision Tree classifiers handle diverse or smaller collections better. This trend is continued in Sections 5.4.4 and 5.4.5.

Models	All Collections	Collections > 10
Gaussian Naïve Bayes	69.4%	78.3%
Random Forest	80.1%	84.4%
Decision Tree	77.7%	80.2%
Linear SVC	66.0%	70.4%
Number of Collections	15	10

Table 10: Leave One Out cross validation accuracy including smaller collections

5.4.2 Analysis of Mislabeled Networks

Although the GNB and RF models were able to predict the collection of networks with very high accuracy, we analyzed the graphs that were consistently misclassified to gain insight into why this was happening. We ran both models 100 times and chose to look into the graphs that were mislabeled over 50% of the time. Table 11 shows the graphs that the RF classifier mislabeled, along with the first two predictions that GNB gave the graphs. Table 12 shows the graphs that GNB mislabeled, along with the RF predictions. It is important to note that the probabilities of 0.000 were too small to show in these tables.

When looking into the second most likely prediction from GNB, we found that for 16 of the 29 mislabeled graphs in Table 12, the second most likely prediction is the correct label. When considering the second prediction, GNB is able to predict the collection of our data in LOOCV with 96.0% accuracy .

Graph	Actual Collection	RF Prediction	GNB Prediction	Probability	2nd Prediction	2nd Probability
web-EPA	Web Graphs	Retweet Networks	Retweet Networks	1.000	Social Networks	0.0000
web-baidu-baike-related	Web Graphs	Social Networks	Social Networks	0.958	Web Graphs	0.0420
web-baidu-baike	Web Graphs	Social Networks	Social Networks	0.999	Web Graphs	0.0009
web-frwikinews-user-edits	Web Graphs	Social Networks	Brain Networks	1.000	Cheminformatics	0.0000
web-google-dir	Web Graphs	Social Networks	Web Graphs	0.623	Social Networks	0.3764
web-google	Web Graphs	Social Networks	Web Graphs	0.995	Social Networks	0.0040
web-hudong	Web Graphs	Social Networks	Web Graphs	1.000	Brain Networks	0.0000
web-polblogs	Web Graphs	Social Networks	Social Networks	0.971	Facebook Networks	0.0295
web-sk-2005	Web Graphs	Social Networks	Web Graphs	0.969	Social Networks	0.0289
web-spam	Web Graphs	Social Networks	Facebook Networks	0.949	Social Networks	0.0501
web-uk-2005	Web Graphs	Brain Networks	Brain Networks	1.000	Web Graphs	0.0000
web-webbase-2001	Web Graphs	Social Networks	Social Networks	0.956	Facebook Networks	0.0259
web-wiki-ch-internal	Web Graphs	Social Networks	Social Networks	0.997	Web Graphs	0.0027
web-wikipedia2009	Web Graphs	Social Networks	Facebook Networks	0.642	Social Networks	0.3569
socfb-B-anon	Facebook Networks	Social Networks	Social Networks	0.718	Facebook Networks	0.3786
socfb-OR	Facebook Networks	Social Networks	Facebook Networks	0.993	Social Networks	0.0064
socfb-nips-ego	Facebook Networks	Retweet Networks	Retweet Networks	1.000	Social Networks	0.0000
soc-FourSquare	Social Networks	Web Graphs	Social Networks	0.998	Web Graphs	0.0022
soc-advogato	Social Networks	Web Graphs	Social Networks	0.680	Facebook Networks	0.3095
soc-douban	Social Networks	Retweet Networks	Retweet Networks	1.000	Social Networks	0.0000
soc-gplus	Social Networks	Retweet Networks	Social Networks	0.969	Web Graphs	0.0294
soc-hamsterster	Social Networks	Web Graphs	Social Networks	0.562	Web Graphs	0.4363
soc-wiki-Talk-dir	Social Networks	Web Graphs	Social Networks	0.997	Web Graphs	0.0032
soc-wiki-Vote	Social Networks	Web Graphs	Social Networks	0.974	Facebook Networks	0.0252
soc-wiki-elec	Social Networks	Facebook Networks	Facebook Networks	0.977	Social Networks	0.0225
rt-higgs	Retweet Networks	Social Networks	Retweet Networks	1.000	Social Networks	0.0000
rt-pol	Retweet Networks	Social Networks	Social Networks	0.986	Web Graphs	0.0140
rt-retweet-crawl	Retweet Networks	Social Networks	Social Networks	0.982	Web Graphs	0.0175
rt-retweet	Retweet Networks	Brain Networks	Social Networks	1.000	Brain Networks	0.0000
bn-fly-drosophila-medulla-1	Brain Networks	Facebook Networks	Brain Networks	1.000	Web Graphs	0.0000
bn-macaque-rhesus-brain-2	Brain Networks	Social Networks	Brain Networks	1.000	Social Networks	0.0000
bn-mouse-visual-cortex-1	Brain Networks	Cheminformatics	Social Networks	1.000	Brain Networks	0.0000
bn-mouse-visual-cortex-2	Brain Networks	Retweet Networks	Retweet Networks	1.000	Social Networks	0.0000

Table 11: Random Forest Mislabel

Graph	Actual Collecion	GNB Prediction	Probability	2nd Prediction	2nd Probability	RF Prediction
web-EPA	Web Graphs	Retweet Networks	1.000	Social Networks	0.0000	Retweet Networks
web-baidu-baike-related	Web Graphs	Social Networks	0.958	Web Graphs	0.0420	Social Networks
web-baidu-baike	Web Graphs	Social Networks	0.999	Web Graphs	0.0009	Social Networks
web-frwikinews-user-edits	Web Graphs	Brain Networks	1.000	Cheminformatics	0.0000	Social Networks
web-italycnr-2000	Web Graphs	Social Networks	0.813	Web Graphs	0.1870	Web Graphs
web-polblogs	Web Graphs	Social Networks	0.971	Facebook Networks	0.0295	Social Networks
web-spam	Web Graphs	Facebook Networks	0.949	Social Networks	0.0501	Social Networks
web-uk-2005	Web Graphs	Brain Networks	1.000	Web Graphs	0.0000	Brain Networks
web-webbase-2001	Web Graphs	Social Networks	0.956	Facebook Networks	0.0259	Social Networks
web-wiki-ch-internal	Web Graphs	Social Networks	0.997	Web Graphs	0.0027	Social Networks
web-wikipedia2009	Web Graphs	Facebook Networks	0.642	Social Networks	0.3570	Social Networks
socfb-B-anon	Facebook Networks	Social Networks	0.718	Facebook Networks	0.2810	Social Networks
socfb-Caltech36	Facebook Networks	Social Networks	0.999	Brain Networks	0.0011	Facebook Networks
socfb-nips-ego	Facebook Networks	Retweet Networks	1.000	Social Networks	0.0000	Retweet Networks
soc-douban	Social Networks	Retweet Networks	1.000	Social Networks	0.0000	Retweet Networks
soc-flickr-und	Social Networks	Web Graphs	1.000	Social Networks	0.0000	Social Networks
soc-livejournal	Social Networks	Web Graphs	0.761	Social Networks	0.2390	Social Networks
soc-tribes	Social Networks	Brain Networks	1.000	Social Networks	0.0000	Social Networks
soc-twitter-follows-mun	Social Networks	Retweet Networks	1.000	Social Networks	0.0002	Social Networks
soc-twitter-follows	Social Networks	Retweet Networks	1.000	Social Networks	0.0000	Social Networks
soc-twitter-higgs	Social Networks	Web Graphs	0.646	Social Networks	0.3530	Social Networks
soc-wiki-elec	Social Networks	Facebook Networks	0.977	Social Networks	0.0225	Facebook Networks
rt-pol	Retweet Networks	Social Networks	0.986	Web Graphs	0.0140	Social Networks
rt-retweet-crawl	Retweet Networks	Social Networks	0.982	Web Graphs	0.0175	Social Networks
rt-retweet	Retweet Networks	Social Networks	1.000	Brain Networks	0.0000	Brain Networks
bn-macaque-rhesus-brain-1	Brain Networks	Social Networks	0.952	Brain Networks	0.0482	Social Networks
bn-mouse-visual-cortex-1	Brain Networks	Social Networks	1.000	Brain Networks	0.0000	Cheminformatics
bn-mouse-visual-cortex-2	Brain Networks	Retweet Networks	1.000	Social Networks	0.0000	Retweet Networks
ENZYMES-g136	Cheminformatics	Brain Networks	1.000	Cheminformatics	0.0000	Cheminformatics

Table 12: Gaussian Naïve Bayes Mislabel

5.4.3 Synthetic Networks Analysis

Table 13 shows the average LOOCV accuracy from the RF and GNB classifiers using the original collections with size greater than 20 as well as the 50 Synthetic Barabasi graphs and 75 Synthetic Erdős-Rényi graphs. Adding these networks caused decent increase to the RF score, while the GNB score decreased slightly

		Leave One Out Cross Validation
Gaussian Naïve Bayes	Accuracy	92.0%
	Standard Deviation	0.000%
Random Forest	Accuracy	94.2%
	Standard Deviation	0.418%

Table 13: Accuracy scores including synthetic networks

Table 14 and Table 15 show confusion matrices for one cross validation prediction of the two best models. We notice that both models successfully classify all 50 Barabasi graphs, with minimal loss of recall. However, the GNB model does not do as well with the Erdős-Renyí graphs, predicting 12 of them as Retweet Networks. Ten out of twelve of these mislabeled graphs were the larger, less dense ER graphs with 100,000 nodes and a p-value of $5.0 * 10^{-6}$.

		Predicted							
		B	C	F	R	S	W	Bar	ER
Actual	Brain	31	0	1	2	1	0	1	0
	Chem	0	119	0	0	0	0	0	0
	Facebook	0	0	113	1	0	0	0	0
	Retweet	0	0	0	59	3	0	0	0
	Social	0	0	1	1	40	5	0	1
	Web	1	0	0	1	10	10	0	0
	Barabasi	0	0	0	0	0	0	50	0
	Erdős-Renyí	0	0	0	0	0	0	0	75

Table 14: Random Forest Confusion Matrix with Synthetic Networks

		Predicted							
		B	C	F	R	S	W	Bar	ER
Actual	Brain	33	0	0	1	2	0	0	0
	Chem	1	118	0	0	0	0	0	0
	Facebook	0	0	111	1	2	0	0	0
	Retweet	0	0	0	59	3	0	0	0
	Social	1	0	1	3	39	3	1	0
	Web	2	0	2	1	6	11	0	0
	Barabasi	0	0	0	0	0	0	50	0
	Erdős-Renyí	0	0	0	12	0	0	0	63

Table 15: Gaussian Naïve Bayes Confusion Matrix with Synthetic Networks

5.4.4 Combining Collections Analysis

The original models that were trained and tested using collections with more than 20 graphs only included 6 collections out of the initial 15 (after data cleaning) in the file. Revisiting the t-SNE plot (Figure 16) generated in section 4.3, collections that were clustered together were identified. Web and Technology Networks seemed to be clustered together, and based on the descriptions, some of the technological graphs represented phenomena similar to the web graphs. Because of this we combined these two collections into one, called Web-Tech. Brain and Biology Networks were also combined into Brain-Bio because of how tightly clustered together they were, as well as the similarities in their underlying phenomena. Furthermore, Scientific Computing and Ecology Networks were combined into Sci-Eco. Although these two collections do not seem related by nature, they were both labeled almost exclusively in the same cluster in the t-SNE plot. Table 16 shows the LOOCV scores resulting from adding these combined collections to the models.

	Combinations Used	Graphs Added	Accuracy Score
Random Forest	Web-Tech	9	91.6%
	Brain-Bio	9	91.5%
	Sci-Eco	17	91.2%
	Web-Tech and Brain-Bio	18	90.9%
	Sci-Eco and Brain-Bio	26	90.2%
	Sci-Eco and Web-Tech	26	90.7%
	All Three	35	89.7%
Gaussian Naive Bayes	Web-Tech	9	90.7%
	Brain-Bio	9	91.7%
	Sci-Eco	17	89.2%
	Web-Tech and Brain-Bio	18	89.7%
	Sci-Eco and Brain-Bio	26	88.3%
	Sci-Eco and Web-Tech	26	87.8%
	All Three	35	87.8%

Table 16: Accuracy Scores using Combined Collections for top models

5.4.5 Miscellaneous Networks Analysis

RF and GNB were the models that most accurately predicted the collections of graphs. These models were trained with graphs that were already labeled, and tested on the miscellaneous networks. As described in section 4.5.6, the miscellaneous file was cleaned and the collections of 50 miscellaneous networks were predicted by the two top models. Table 17 shows the accuracy scores, showing that when using all the miscellaneous networks, the accuracy score is a very low 39.8%. Even though this number is not ideal, some analysis of these results showed that some of the collection hypothesis made weren't collections which the model was trained on. The models where trained using 7 collections (including the three combined collections proposed in section 4.5.5), and the collection hypotheses of the miscellaneous networks include 11 different collections (see Table 18). Becuase of this, there are 18 graphs that will always be mislabeled, therefore obtaining a very low accuracy score.

To more accurately test our best models, the graphs with collection hypotheses not included in training the model were removed, therefore leaving the file with only 32 graphs. Looking again at Table 17, the resulting RF accuracy score was 60.1%.

	Collections Used	Graphs Used	Accuracy
Random Forest	All Collections	50	38.5%
	Only collections the model was trained on	32	60.1%
Gaussian Naïve Bayes	All Collections	50	20.0%
	Only collections the model was trained on	32	31.25%

Table 17: Accuracy Scores for Miscellaneous Networks

Collection	Is it in the model?
Facebook Networks	✓
Web Networks	✓
Technological Networks	✓
Lexical Networks	✗
Recommendation Networks	✗
Social Networks	✓
Affiliation Networks	✗
Collaboration Networks	✗
Biological Networks	✓
Infrastructure Networks	✗
Citation Networks	✗

Table 18: List of Collections included in the Miscellaneous Networks

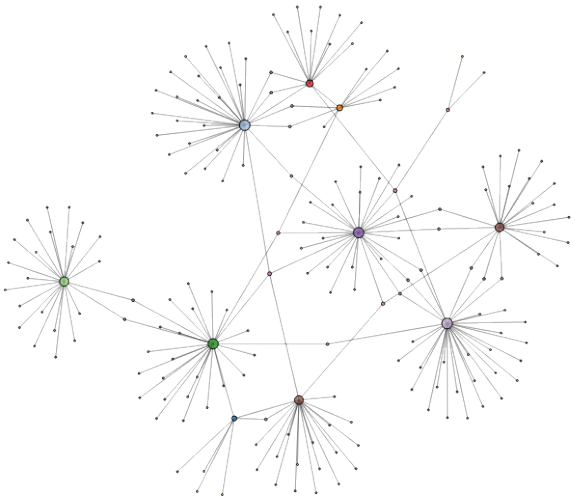


Figure 22: Brain-Bio Network - A mouse's visual cortex

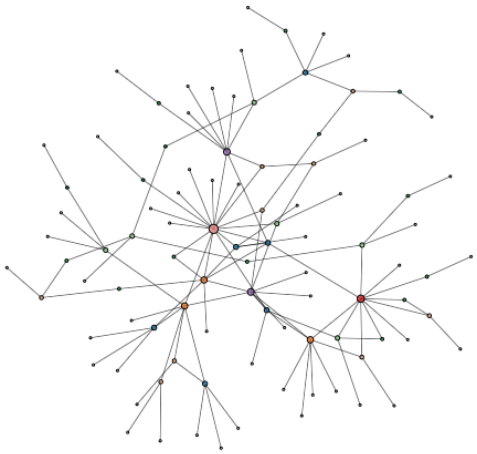


Figure 23: Retweet Network

6 Results and Discussion

We began with the hypothesis that real-world networks from different disciplines have unique characteristics that can be identified through machine learning. Our results support this hypothesis, as the Gaussian Naïve Bayes and Random Forest classifiers were able to predict the collections of real-world networks with 92.8% and 92.3% accuracy respectively. After conducting experiments reducing the features, combining collections, adding synthetic networks, and predicting the collections of miscellaneous networks, our results suggest that although the GNB classifier performs better when using large, clear categories, the RF classifier performs better when including smaller and less concise categories. When adding synthetic networks to the data, both models were able to classify both Barabasi and Erdős-Renyí graphs rather with high accuracy, suggesting that these graphs are easier to classify.

Tables 8 and 9 show that both RF and GNB models tend to label Web Graphs as Social Networks, and vice versa. This could be because the websites that the Web Graphs represent are organized based on social relationships. For example, Table 12 shows that the graph 'web-polblogs' was mislabeled by the GNB classifier as a Social Network. Furthermore, the second highest probability from the prediction was for Facebook Networks. This graph represents the hyperlinks between political blogs, and it would make sense that these links connect blogs promoting similar opinions, just as people tend to have relationships with other like-minded people. Thus, this Web Graph might mimic the relationships represented by the Social Networks.

Figure 22 shows a Brain Network representing a mouse's visual cortex. This graph was mislabeled 100% of the time by both RF and GNB as a Retweet Network (an example of a Retweet Network is shown in Figure 23). Visually comparing both graphs, a similarity can be seen in the structure, and even though the phenomena that generated these graphs is distinctly different, there seem to be similar characteristics between them. This finding is an example of how our research can promote interdisciplinary communication. In this case, this mouse's brain network can potentially be studied by looking into established methods researchers use to study Retweet Networks, and applying them to this particular network. This type of analysis can be done with many of the mislabeled graphs, thereby giving insight into other unexpected similarities between collections.

Another interesting finding is that of all the Brain Networks, the networks representing human brains were correctly classified, while the misclassified Brain Networks all represented animal brains. This classification makes sense, as different species have different levels of brain complexity. However, both this finding and the previous one would require further research in order to make any definitive conclusions.

We have shown that some collections contain outlier graphs that have different qualities. On the other

hand, we have also shown that some collections are similar enough that they can be combined. Table 16 shows that adding the three proposed combined collections does not decrease the accuracy of the models significantly. This suggests that combining collections is a viable option, and can be further explored. Furthermore, the miscellaneous graphs with the collection hypothesis of Lexical Networks were most often labeled as Social Networks. This could also imply that these collections could be combined, thereby enabling the models to recognize more collections. When analyzing the miscellaneous networks, both RF and GNB obtained a moderately low accuracy score. Adding more combined collections to the models would allow them to identify more of the miscellaneous graphs, thus improving the accuracy score. However, it is important to note that the miscellaneous score depends on the collection hypotheses, which are at best educated guesses, and could be flawed.

7 Conclusion

In conclusion, this work provides evidence for the hypothesis that networks from different disciplines have distinct characteristics. It also provides a method to train a machine learning algorithm to use these characteristics to classify an unlabeled network into a field of origin. We finish with a discussion of how to analyze graphs that are consistently labeled into an unexpected category and a summary of potential future research.

The connections between network categories identified through the mislabel analysis could help researchers gain insights on their own graphs. The unexpected potential similarities in networks across different fields of study ideally will lead to increased interdisciplinary communication and collaboration with networks.

Networks that were mislabeled can also give some new insights. A mislabeled graph, even though it means the model incorrectly classified it, helps us understand the connections between seemingly unrelated network collections.

7.1 Future Work

Some extensions of this work could include collecting more graphs to include more categories in the training set. This would greatly improve the models' ability to classify a miscellaneous graph. Another possible extension of this research is further investigating the similarity discovered between collections, such as Social and Web graphs or possibly Brain and Biology networks. Finally, we hope to extend this work by deploying a version of the algorithm as part of the Network Repository's upload process to provide other researchers suggestions for further research.

References

[1] Dimacs implementation challenges, 1990-2012.

[2] E. T. Aaron Clauset and M. Sainz.

[3] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. Efficient graphlet counting for large networks. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 1–10. IEEE, 2015.

[4] Azure. Azure tdsp utilities, jan 2017.

[5] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[6] J. A. Barnes. Graph theory and social networks: A technical comment on connectedness and connectivity. *Sociology*, 3(2):215–232, 1969.

[7] C. Berge and E. Minieka. Graphs and hypergraphs. 1973.

[8] S. Bonner, J. Brennan, I. Kureshi, G. Theodoropoulos, and A. McGough. Efficient comparison of massive graphs through the use of graph fingerprints. In *Twelfth Workshop on Mining and Learning with Graphs (MLG) Workshop at KDD'16*, 2016.

[9] S. Bonner, J. Brennan, G. Theodoropoulos, I. Kureshi, and A. S. McGough. Deep topology classification: A new approach for massive graph classification. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 3290–3297. IEEE, 2016.

[10] N. V. Chawla, N. Japkowicz, and A. Kotcz. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[11] J. Demšar, T. Curk, A. Erjavec, Č. Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, et al. Orange: data mining toolbox in python. *The Journal of Machine Learning Research*, 14(1):2349–2353, 2013.

[12] P. Erdos and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[13] J. Grus. *Data science from scratch: First principles with Python.* " O'Reilly Media, Inc.", 2015.

[14] T. Guo and X. Zhu. Understanding the roles of sub-graph features for graph classification: an empirical study perspective. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 817–822. ACM, 2013.

[15] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[16] J. Kunegis. Konect - the koblenz network collection. In *Proceedings of Int. Web Observatory Workshop*, pages 1343–1350, 2013.

[17] G. N. Lance and W. T. Williams. Mixed-data classificatory programs i - agglomerative systems. *Australian Computer Journal*, 1(1):15–20, 1967.

[18] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[19] G. Li, M. Semerci, B. Yener, and M. J. Zaki. Effective graph classification based on topological and label attributes. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(4):265–283, 2012.

[20] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[21] T. Milenković, J. Lai, and N. Pržulj. Graphcrunch: a tool for large network analyses. *BMC bioinformatics*, 9(1):70, 2008.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[23] S. Raschka. Logistic regression, 2014-2017.

[24] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[25] M. M. M. P. P. R. C. T. B. V. d. V. Sarah Bird, Luke Canavan. Welcome to bokeh, 2015.

[26] S. Soundarajan, T. Eliassi-Rad, and B. Gallagher. A guide to selecting a network similarity method. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 1037–1045. SIAM, 2014.

[27] W. M. . P. D. Team. pandas: powerful python data analysis toolkit, jul 2017.

[28] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1307–1318. ACM, 2013.

[29] M. van Steen. *Graph Theory and Complex Networks*. Maarten van Steen, first edition, apr 2010.

[30] J. Whitbeck, M. Dias de Amorim, V. Conan, and J.-L. Guillaume. Temporal reachability graphs. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 377–388. ACM, 2012.

[31] K. Wu, P. Watters, and M. Magdon-Ismail. Network classification using adjacency matrix embeddings and deep learning. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 299–306. IEEE, 2016.

[32] K. Xu. Bhoslib: Benchmarks with hidden optimum solutions for graph problems (maximum clique, maximum independent set, minimum vertex cover and vertex coloring)–hiding exact solutions in random graphs. web site. *Web site, http://www. nlsde. buaa. edu. en/˜ kexu/benchmarks/graphbenchmarks. htm.*