INTRO

Abstract:

- ❏ An important component of cyber security is intrusion detection. One way to perform intrusion detection is through anomaly detection: using algorithms to differentiate between normal and abnormal, potentially malicious, network usage in network traffic data.
- ❏ Many types of anomaly detection algorithms exist. Algorithms such as Local Outlier Factor, DBSCAN, and K-Means use distance measures to identify anomalies, while algorithms such as Isolation Forest use a decision tree to accomplish the same task. Choosing the best algorithm is hard enough; however, each algorithm requires user-input parameters, and often algorithms do not offer straight-forward, objective methods for choosing the best parameter. Thus, in this project, we focused on parameter selection and, after many experiments, verified the best parameters for using anomaly detection algorithms on the NSL-KDD dataset.
- ❏ However, the network traffic data to be analyzed presents its own set of issues. For example, the IDS 2017 dataset contains missing values that must be filled, and the NSL-KDD dataset has categorical data that must be converted into numerical data before a distance-based clustering algorithm can be implemented. Thus, in this project, we further focused on experimentally evaluating the most effective preprocessing techniques.
- ❏ Despite focusing on dataset preprocessing and algorithm parameters, the larger goal of this project is evaluating clustering-based anomaly detection techniques, and this project we accomplished by comparing the clustering results K-Means and DBSCAN experiments.

- ❏ Cyber Security : Parameters and Preprocessing for Cluster Based Anomaly Detection
  - ❏ Dataset Generation
    - ❏ Hard to create or find a dataset with up-to-date attacks for quality training and testing (CIC)
      - ❏
    - ❏ Generated datasets that we used:
      - ❏ NSL-KDD
        - ❏ Updated and improved version of KDD 99
        - ❏ Though it "may not be a perfect representative of existing real networks," it "still can be applied as an effective benchmark dataset" for Intrusion Detection models (CIC).
        - ❏ Small enough of a dataset so that one doesn't need to "randomly select a small portion" to run tests (CIC).
        - ❏ We chose this dataset because it is popular and updated.
        - ❏ 125973 instances
        - ❏ 43 columns: 41 columns of features, 1 column of difficulty level (how difficult it is for a computer to correctly label this specific instance, used for supervised learning), and 1 column of attack/normal usage labels

      - ❏ IDS 2017
        - ❏ Generated by the Canadian Institute for Cyber Security

- ❑ This dataset “contains benign and the most up-to-date common attacks, which resembles the true real-world data” (CIC).
- ❑ We chose this dataset because it is a recently generated set and therefore likely more similar to attacks that happen today, and also because it is larger than NSL-KDD but still small enough that our computers can handle analyzing it in a reasonable amount of time (at least for K-Means Algorithm).
- ❑ 5 day data capture
    - ❑ We used data from Wednesday night, which has six labels: five types of attacks plus a “benign” category for normal usage.
- ❑ 692703 instances
- ❑ 79 columns: 78 columns of features and 1 column of attack/normal usage labels

- ❑ Anomaly Detection vs. Misuse Detection
    - ❑ “While misuse-based detection is generally favored in commercial products due to its predictability and high accuracy, in academic research anomaly detection is typically conceived as a more powerful method due to its theoretical potential for addressing novel attacks” (Tavallaee, Bagheri, Lu, and Ghorbani, 2009).
    - ❑ Misuse detection
        - ❑ Compares incoming signatures to known attacks or “rules” (Craiger, 2014)
        - ❑ Can be implemented immediately because IDS simply compares all incoming packets with a set of preloaded signatures (Craiger, 2014)
    - ❑ Anomaly detection
        - ❑ Unlike misuse detection, able to detect previously unknown attacks based off of model generated by an algorithm (Omar, Ngadi, and Jebur, 2013)
        - ❑ Disadvantage compared to misuse detection is relatively high false-positive rates (Omar, Ngadi, and Jebur, 2013)
        - ❑ Establishes a baseline for “normal” activity and flags any activity outside of the “norm” (Craiger, 2014)
        - ❑ Supervised / Semisupervised
            - ❑ Requires training period for the computer to learn labeled data (Davis)
            - ❑ Issues: “shortage of training data set that covers all areas,” and “obtaining accurate labels is a challenge and the training sets usually contain some noises that result in higher false alarm rates” (Omar, Ngadi, and Jebur, 2013)
        - ❑ Unsupervised
            - ❑ No training period needed (Davis, 2017), and no training data needed (Omar, Ngadi, and Jebur, 2013)

            - Algorithm/program runs data and decides what is an anomaly and what is normal (Davis, 2017)
            - Assumes normal data is much more common than anomalous data (Davis, 2017) and that "malicious traffic is statistically different from normal traffic" (Omar, Ngadi, and Jebur, 2013)
- We chose to work on unsupervised methods
    - Cluster-Based Anomaly Detection
        - Cluster dataset using the clustering algorithm
        - Data Points are grouped into clusters
        - Clusters then have corresponding attack types or normal labels based on the most prevalent cluster in data
        - Labels are not needed to cluster data but are needed for comparing clusters found by the algorithm to the true clusters known before clustering.
    - A few existing clustering algorithms: (descriptions copied from our github README that we wrote after beginning the outline)
        - K-Means
            - Works by simply "computing the distances between points and group centers; very few computations! It thus has a linear complexity O(n)" (Seif, 2018).
            - We used K-Means for its computational simplicity. It is "one of the simplest unsupervised learning algorithms" used for solving "known clustering problem[s]" (Kuman, Chauhan, adn Panwar, 2013).
            - Because K-Means clusters using centroids, it assumes that clusters are circular-shaped
            - K-Means clusters data by starting with user-specified K initial cluster centroids, and assigning all points to the nearest centroid. Based on the assignments, the algorithm recalculates the cluster centers and reassigns all points to the nearest cluster center. The algorithm repeats this process for a default of 300 iterations. When the process ends, K-Means has clustered data into K clusters. SciKitLearn's K-Means algorithm offers the option for the user to also specify the method for initialization, the way that the algorithm chooses which points to use as initial cluster centroids. In this project, the user specifies K, the number of initial cluster centroids and eventual clusters. A typical way of choosing K is often by the elbow method. The implementation of K-Means in this project reports the sum of squared distances to cluster centers (or squared sum of errors, SSE) needed in the elbow plot, so a user can run tests with different values of K and plot that against the SSE for each K value. A user can then subjectively choose the elbow point on such a plot to determine the best K, and can then conduct tests with this K. The researchers suggest using a few values of K

around the elbow and comparing the evaluation metric scores generated for each K in order to determine the best value of K.

- ❏ DBSCAN
    - ❏ Unlike K-Means, which “assumes that all directions are equally important for each cluster” (Pierobon, 2018) and therefore assumes that clusters are circular-shaped (or hyper-spherical shaped for multi-dimensional data), DBSCAN instead takes the density of data into account for clustering.
    - ❏ Density-Based Spacial Clustering of Applications with Noise, or DBSCAN, relies on two user-input parameters, namely epsilon and minimum samples. Epsilon denotes the neighborhood of density to be explored for each data point, and minimum samples denote the minimum number of samples needed to be within a point’s epsilon neighborhood for said point to be considered a core point. Points within another core point’s epsilon neighborhood, but not core points themselves, are considered border points. Meanwhile, points that are not within another core point’s epsilon neighborhood, and that are not core points themselves, are considered anomalous points or noise. DBSCAN finds clusters of core points and border points and reports those clusters along with a group of all of the anomalous points. SciKitLearn's DBSCAN offers the user other parameters to manipulate the specific way that DBSCAN calculates the clusters; this project uses all default parameters except for the algorithm parameter, for which the project specifies the 'brute' option in order to reduce run time. **DBSCAN run time will depend of how big the dataset is and what resources your computer has. Since "DBSCAN groups together points that are close to each other based on a distance measurement," it is slower than K-means algorithm (Salton do Prado, 2017). The experiments on DBSCAN were made on a Macbook Pro 2.6 GHz i7 with 16 GB of RAM memory and using the Brute parameter for the algorithm. The average time for these experiments was 3 minutes. DBSCAN tests were attempted on a Macbook air 1.6 GHz i5 with 8GB of RAM, but after 30 minutes never finished due to the processing capacity of the computer. Before running experiments with DBSCAN make sure the computer can handle it.**
- ❏ Isolation Forest
    - ❏ Similarly to Local Outlier Factor, Isolation Forest returns for each point a score representing the probability of that particular point being an anomaly, and the user must choose a threshold for which scores will indicate an anomaly and which

will indicate a normal instance. The algorithm generates the probability scores for each instance by the following process: *First, randomly choose a feature (column). Next, randomly choose a value between the min and max of that feature. Partition, or split the data into two groups: those with values in that feature above the randomly chosen value, and those with values below. Now, choose one of the two groups again and split the data on a random point. Repeat until a single point is isolated. Obtain the number of splits required to isolate that point. Repeat this process, eventually isolating all points across many features, and obtain for each specific point the average number of splits required for that point to be isolated*. The theory behind Isolation Forest states that anomalies occur less frequently and differ more greatly than normal points, and therefore will require fewer partitions, or splits, to isolate them than normal points would require. Thus, a score based on the average number of splits, also known as the average path length, denotes the probability that a particular point is an anomaly. The score is adjusted such that a a score near 1 denotes a likely anomaly, and a score near 0.5 denotes a likely normal point. Again, the user can set the contamination factor to indicate the threshold for scores labeled as anomaly and as normal.

- ❏ Local Outlier Factor
  - ❏ Local Outlier Factor, or LOF, begins with the parameter K, a default-set or user-chosen integer. For a specific point, the algorithm calculates the reach-distance to each point, which is essentially the distance from a specific point to another point with a small smoothing caveat for close points. The algorithm then takes the average of the reach-distances for a specific point to each of that point's k-nearest neighbors. The inverse of this average is called the Local Reachability Distance, or LRD. A point's high LRD indicates that the point exists in a highly dense neighborhood and does not have to travel far to encounter all K nearest neighbors, and a point's low LRD indicates the opposite, a low-density neighborhood. The algorithm calculates the LRDs for each point, and finds the average of all LRDs. Finally, the algoirthm calculates the Local Outlier Factor for each point by dividing that point's LRD by the average LRD of all points. An LRD around 1 indicates a point with average density, and an LRD much greater than 1 indicates a point in a much lower-density neighborhood than the average point, and therefore a point that is likely an anomaly. SciKitLearn's LOF algorithm returns the negative of each point's local outlier factor. In this code, one can choose an

Offset value such that all points with an LOF more negative than that Offset value are labeled as anomalous points, and all points equal to or more positive than that Offset value are labeled as normal points.

METHODS

- ❏ Preprocessing
  - ❏ Dealing with categorical features (NSL-KDD dataset)
    - ❏ Omit entirely
      - ❏ In the NSL-KDD dataset, only 3 features out of the total 40 data features are categorical; the remaining 37 are numerical. The unsupervised clustering algorithms used in this experiment rely on numerical feature values, which means that categorical features must be encoded in some way. Omitting the relatively small ratio of features that are categorical eliminates the burden of finding the best encoding method, thus simplifying the preprocessing algorithm. We do not expect that omitting the small number of categorical features will have a large effect on clustering results. Indeed, our experiment results show that removing categorical features effects the F-Score of K-Means clustering by no more than 0.13%.
    - ❏ One Hot Encode
      - ❏ Each possible category in categorical features becomes a separate feature, and each data point is either labeled as 0 (does not have that categorical feature) or 1 (does have that categorical feature). While this process lets each category be "represented by a separate feature" and therefore "build[s] more fine-grained models," this method of encoding also increases the dimensionality of the data by the number of categories, resulting in "an even higher memory consumption" (Mittag, Romer, and Zell, 2015).
    - ❏ One Hot Encode Protocol Type and Service Flag; Omit Service Type
      - ❏ The protocol type feature has 4 appearing categories and service flag type has 11 appearing categories, but service type has 69 appearing categories. Because significantly more category types appear within the service type feature, adding
    - ❏ Replace with associated risk value
      - ❏ Juanchaiyaphum, Arch-Int, N. Arch-Int, S., and Saiyod, of Khon Kaen University in Thailand, performed experiments to analyze categorical information and associated data label (normal usage or attack). The researchers assigned calculated risk values to each service type and status flag, and found through their tests that replacing categorical features with associated risk values effectively improved the F-Score and other metrics when compared with omitting categorical features, and effectively reduced run time and data dimensionality when compared to One Hot Encoding the data (Juanchaiyaphum, Arch-Int, N. Arch-Int, S., and Saiyod, 2014).

            - Because the study offered no risk values for protocol type, we tried assigning risk values to those with risk values while: 1) omitting protocol type and 2) one hot encoding protocol type.
- Scaling/standardizing data
    - Important for Euclidean distance-based algorithms like K-Means because, when calculating the statistical distance between data points, certain features with greater ranges will have greater weight in the calculation, simply because of the range of the data and not because that feature should be considered with extra weight than others.
    - We used MinMaxScaler in SciKit Learn
        - Subtracts minimum value of feature from each value in feature, then divides by range (minimum value in feature subtracted from maximum value in feature) (Hale, 2019).
        - Function "preserves the shape of the original distribution," does not "meaningfully change the information embedded in the original data," and does not "reduce the importance of outliers" (Hale, 2019).
- Dimensionality reduction
    - The network traffic datasets used in this study, namely NSL-KDD and IDS 2017, have 41 and 78 features respectively. Fodor writes that the "traditional statistical methods," such as clustering algorithms, "break down… mostly because of the increase in the number of variables associated with each observation" (Fodor, 2002). The sheer high dimensionality makes analyzing and clustering network traffic data a difficult chore indeed. However, because some of the variables, or dimensions, are simply not "important" to the task at hand, namely identifying attacks (Fodor, 2002), and because others are redundant, many dimensionality reduction techniques exist to narrow down the data to only the most informative and necessary features for the task at hand.
    - Feature Selection
        - Drops features that are unimportant or redundant.
        - Low variance filter
            - Assigns a value to each feature that shows the variance of data with feature and without feature. The value indicates how much each feature differentiates one datapoint from another.
            - User picks threshold of low variance value and drops a certain number of features with low variance from the original dataset.
    - Dimensionality Reduction
        - Combines dimensions to generate a new data set. Features lose real-world significance but retain large degrees of variance with significantly fewer dimensions.
        - PCA
            - This dimensionality reduction technique is "the best, in the mean-square error sense, linear reduction technique" (Fodor, 2002).

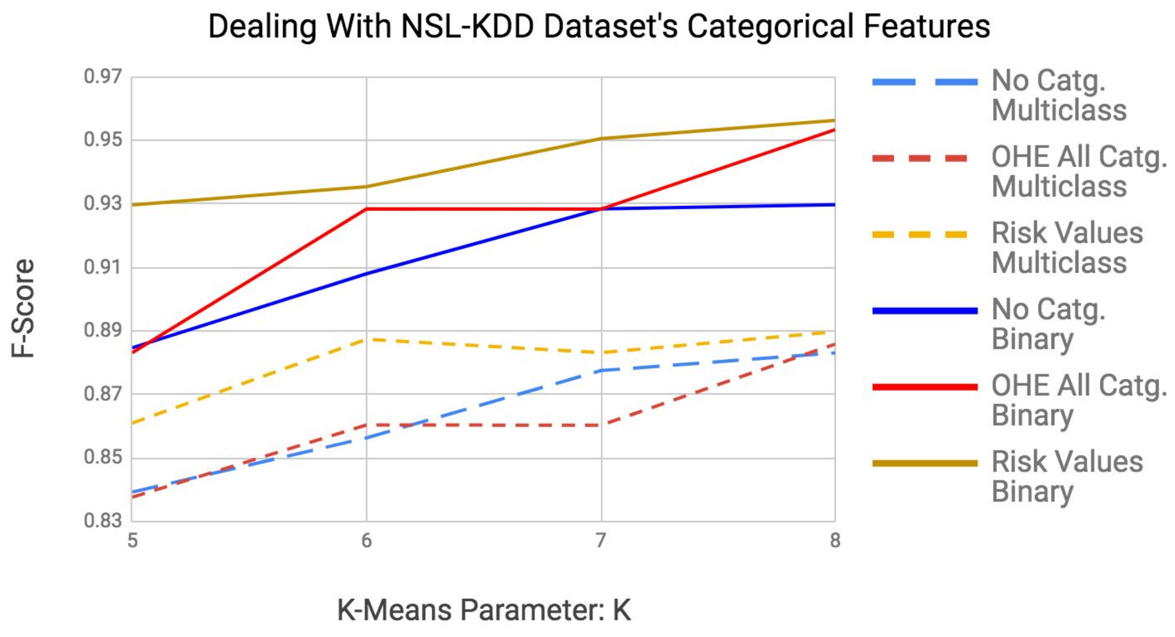
                - ❏ Works by “finding a few orthogonal linear combinations (the PCs) of the original variables with the largest variance” (Fodor, 2002).
                - ❏ The first PC is the “linear combination with the largest variance,” and each successive PC is the next linear combination with the next biggest variance. The algorithm iteratively finds PCs until the resulting linear combinations describe X% of the variance, where X is a user-set threshold typically between 95 and 99.

- ❏ Clustering
    - ❏ K-Means
        - ❏ The user chooses the number of clusters, K.
            - ❏ Elbow method
                - ❏ is typically used, but it is subjective and still relies on the user interpreting elbow graph.
                - ❏ Run K-Means on data with different values of K, starting at K=2 and increasing by 1 for each test. Calculate “percentage of variance explained by the clusters” for each test with each different value of K (Bholowalia and Kumar). Plot percentage of variance against K. Find “elbow point,” where variance stops changing at a significant rate; find corresponding K value for this point and use that as K for clustering the data.
            - ❏ Instead of using the elbow method, we ran tests with a variety of K values and calculated F-Scores for each value of K, noting what K values give the best F-Scores. Keep in mind that as K approaches the number of data points (aka increases) data becomes overfitted. Focus on K values around the number of expected clusters, aka the number of attacks/normal usage labels present in the dataset.
        - ❏ The algorithm begins with K randomly chosen centroids for the data. Each data point is assigned to the nearest centroid, and the centroid is recalculated for each cluster. This process is repeated with the new centroids, and repeated for X number of iterations chosen by the user or until the clusters don’t change very much.
        - ❏ Initial Seed
            - ❏ K-Means ++
            - ❏ Random
    - ❏ DBSCAN
        - ❏ DBSCAN relies on two user-input parameters, namely epsilon and minimum samples. Epsilon denotes the neighborhood of density to be explored for each data point, and minimum samples denote the minimum number of samples needed to be within a point’s epsilon neighborhood for said point to be considered a core point.
        - ❏ Points within another core point’s epsilon neighborhood, but not core points themselves, are considered border points. Meanwhile, points that are *not*

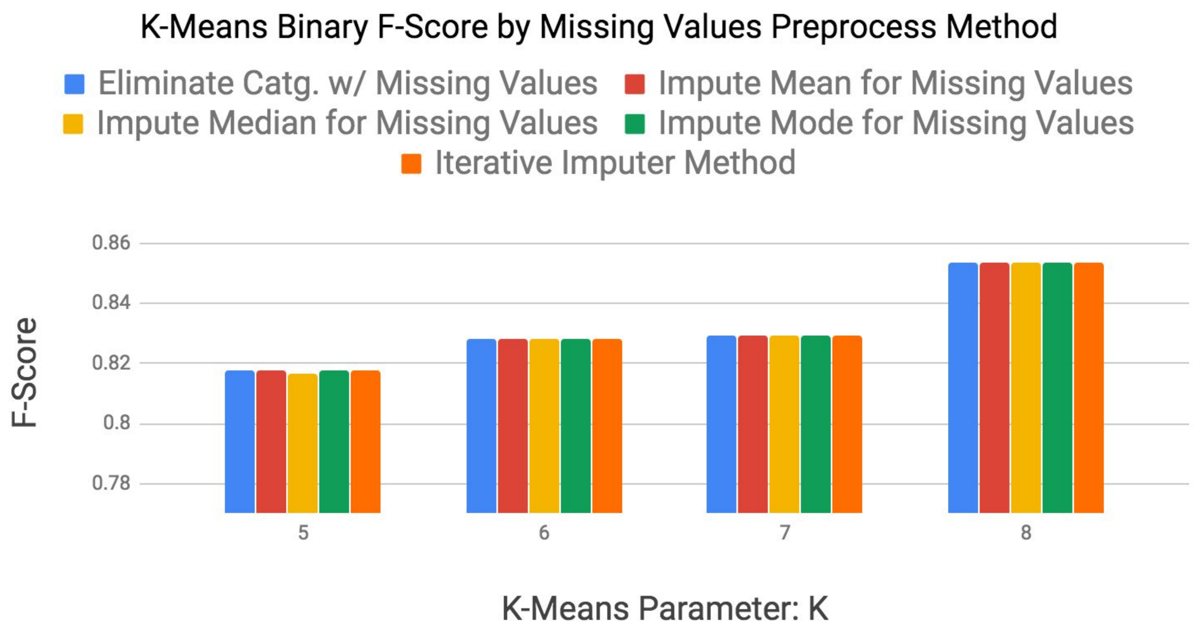
    within another core point’s epsilon neighborhood, and that are not core points themselves, are considered anomalous points or noise.
  - ❑ DBSCAN finds clusters of core points and border points and reports those clusters along with a group of all of the anomalous points.
  - ❑ Parameters
    - ❑ Run tests on data with different parameters and calculate F-Scores for each clustering. Focus on parameters that yield highest F-scores, denoting best clustering as compared to True cluster labels.
      - ❑ Choosing esp
      - ❑ Choosing minimum samples
      - ❑ Choosing the algorithm parameter
- ❑ Isolation Forest
  - ❑ Similarly to Local Outlier Factor, Isolation Forest returns for each point a score representing the probability of that particular point being an anomaly, and the user must choose a threshold for which scores will indicate an anomaly and which will indicate a normal instance.
  - ❑ The algorithm generates the probability scores for each instance by the following process: *First, randomly choose a feature (column). Next, randomly choose a value between the min and max of that feature. Partition, or split the data into two groups: those with values in that feature above the randomly chosen value, and those with values below. Now, choose one of the two groups again and split the data on a random point. Repeat until a single point is isolated. Obtain the number of splits required to isolate that point. Repeat this process, eventually isolating all points across many features, and obtain for each specific point the average number of splits required for that point to be isolated.*
  - ❑ The theory behind Isolation Forest states that anomalies occur less frequently and differ more greatly than normal points, and therefore will require fewer partitions, or splits, to isolate them than normal points would require. Thus, a score based on the average number of splits, also known as the average path length, denotes the probability that a particular point is an anomaly. The score is adjusted such that a a score near 1 denotes a likely anomaly, and a score near 0.5 denotes a likely normal point. Again, the user can set the contamination factor to indicate the threshold for scores labeled as anomaly and as normal.
- ❑ Local Outlier Factor
  - ❑ Begins with the parameter K, a default-set or user-chosen integer. For a specific point, the algorithm calculates the reach-distance to each point, which is essentially the distance from a specific point to another point with a small smoothing caveat for close points.
  - ❑ The algorithm then takes the average of the reach-distances for a specific point to each of that point's k-nearest neighbors. The inverse of this average is called the Local Reachability Distance, or LRD. A point's high LRD indicates that the point exists in a highly dense neighborhood and does not have to travel far to encounter all K nearest neighbors, and a point's low LRD indicates the opposite, a low-density neighborhood. The algorithm calculates the LRDs for each point, and finds the average of all LRDs.

        - ❑ Finally, the algoirthm calculates the Local Outlier Factor for each point by dividing that point's LRD by the average LRD of all points. An LRD around 1 indicates a point with average density, and an LRD much greater than 1 indicates a point in a much lower-density neighborhood than the average point, and therefore a point that is likely an anomaly.
        - ❑ SciKitLearn's LOF algorithm returns the negative of each point's local outlier factor. In this code, one can choose an Offset value such that all points with an LOF more negative than that Offset value are labeled as anomalous points, and all points equal to or more positive than that Offset value are labeled as normal points.
- ❑ Evaluating
    - ❑ Comparing against "true" clusters
        - ❑ Binary (normal vs. abnormal)
        - ❑ Multiclass (Specific attacks)
            - ❑ NSL-KDD: Normal (not an attack), DoS, Probe, R2L, U2R
            - ❑ IDS 2017: BENIGN,DoS slowloris,DoS Slowhttptest,DoS Hulk,DoS GoldenEye,Heartbleed
        - ❑ For NSL-KDD:
            - ❑ The original dataset contains 22 attacks; each particular attack belongs to a broader attack group, either DoS, Probe, R2L, & U2R
    - ❑ F-Score
        - ❑ The harmonic mean of precision and recall.
        - ❑ Precision is the ratio of correctly predicted positive values to all values predicted to be positive. AKA, how often the clustering algorithm identified a true positive vs. how often the clustering algorithm identified false positives and true positives. AKA, how sure are you that the found positive values are actually positive.
        - ❑ Recall is the ratio of correctly predicted positive values to all values that are actually positive. AKA, how often the clustering algorithm identified a true positive vs. how many true positives there really are. AKA, how sure you are that you didn't miss any positive values when you labeled values as positive.
    - ❑ NMI
        - ❑ Entropy measure
    - ❑ ARS
        - ❑ Similarity measure

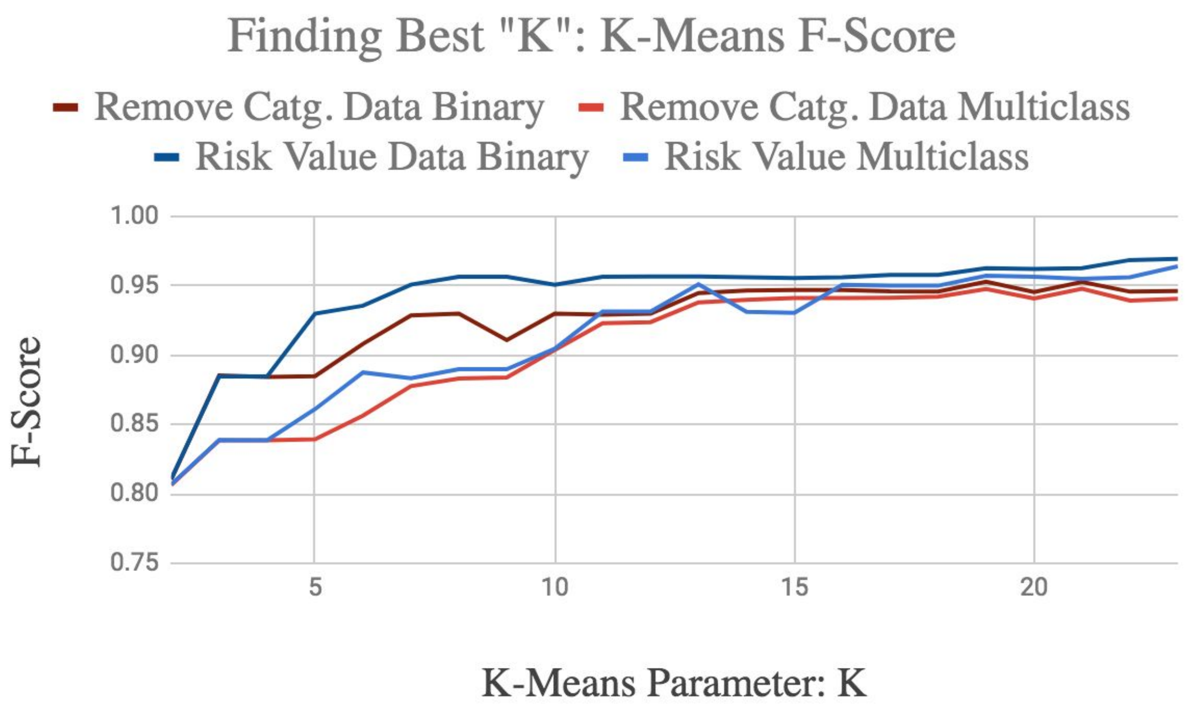
RESULTS
- ❑ Preprocessing
    - ❑ Dealing with categorical features

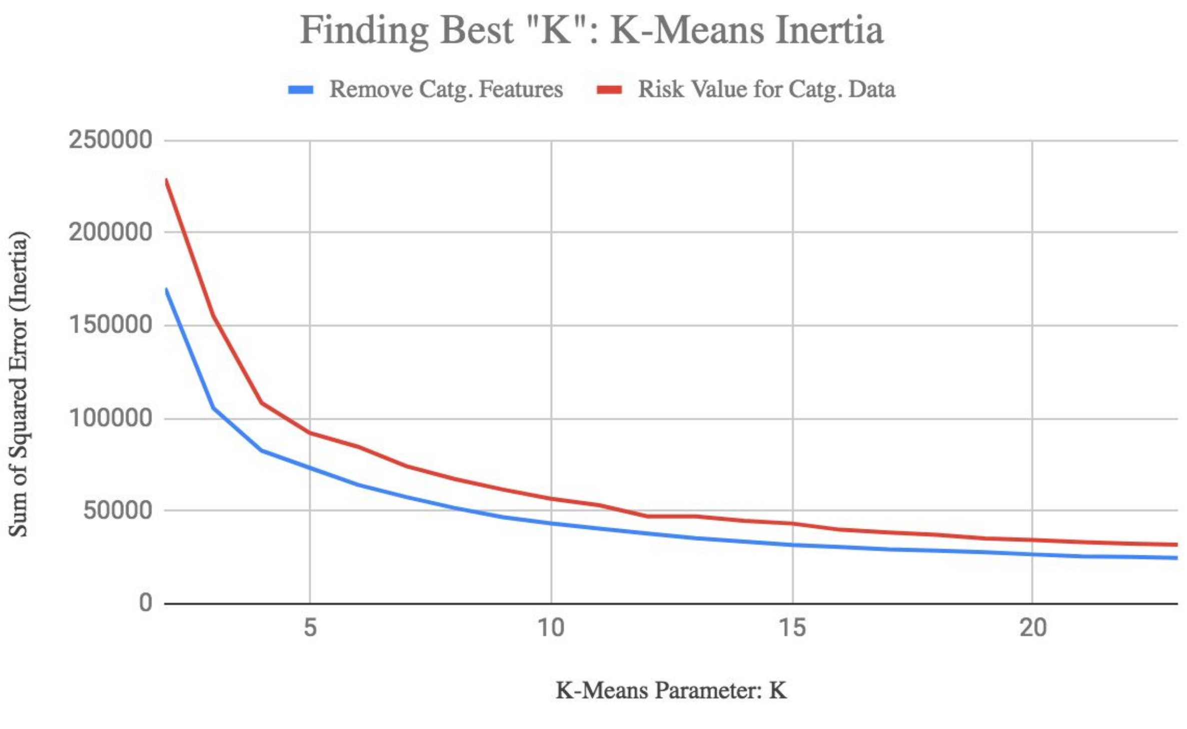


- ❑ Dealing with missing values

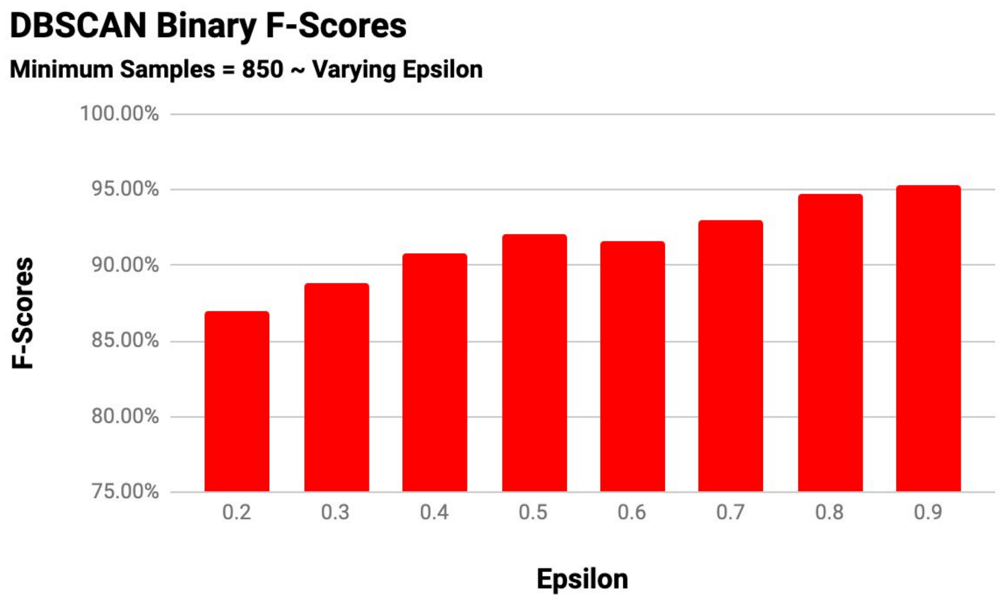


- ❑ Clustering
    - ❑ K-Means
        - ❑ Choosing the best K

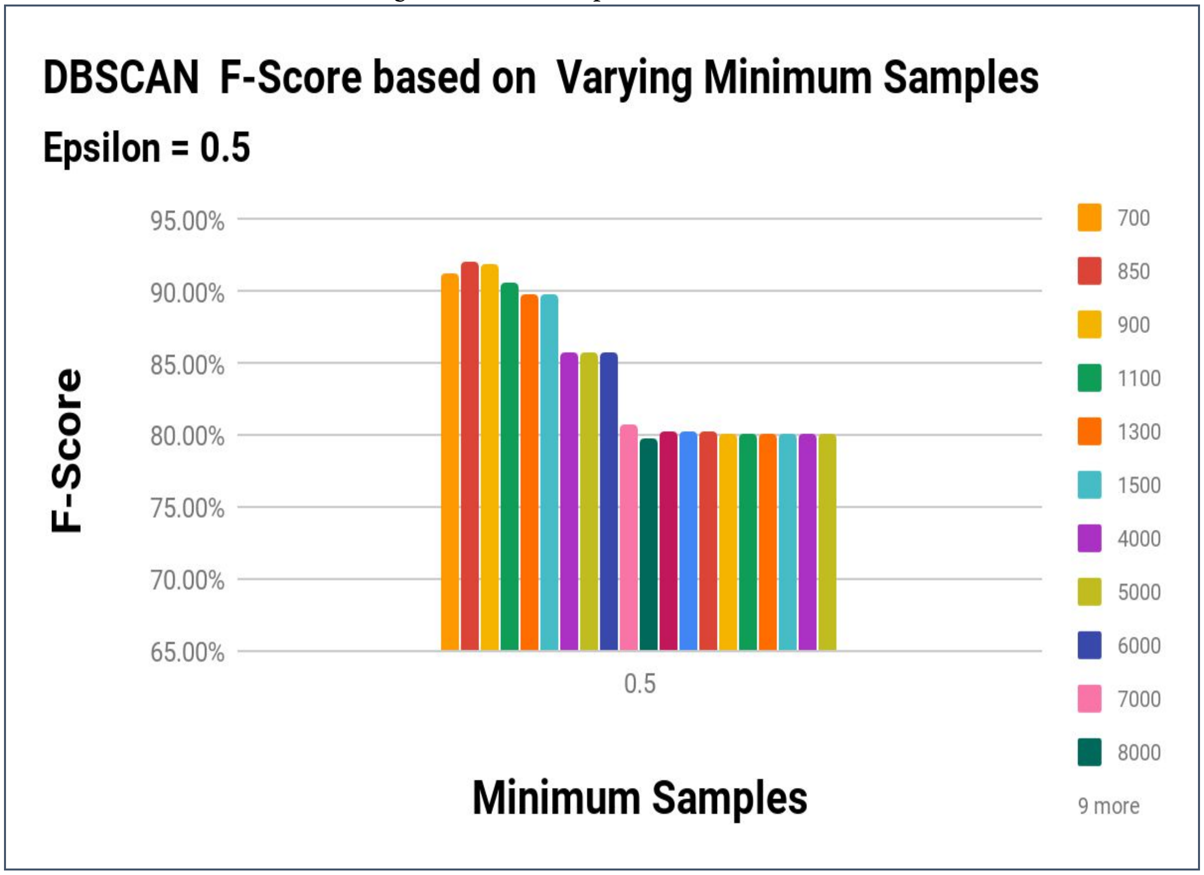




- ❏ DBSCAN
  - ❏ Choosing esp



- ❏ Choosing minimum samples



- ❏ Isolation Forest

        - Local Outlier Factor

Comparison of F-Scores and Run Times: Best Algorithm Parameters					
Dataset	**Algorithm**	**Parameter(s)**	**Binary F-Score**	**Multiclass F-Score**	**Run Time (seconds)**
NSL-KDD	K-Means	K = 8	95.63%	88.98%	**0.0000188**
	DBSCAN	Esp = 0.8 ~ MS = 650	**96.19%**	**92.09%**	214.2
	LOF	Contamination = 0.3	54.19%	53.46%	259.8
	ISOForest	Contamination = 0.25	63.66%	55.79%	24.3

- Evaluating
    - Comparing against "true" clusters
        - Binary (normal vs. abnormal)
        - Multiclass (Specific attacks, eg DoS, Probe, etc.)
    - F-Score, NMI, and ARS
        - While F-Scores for quality DBSCAN and K-Means clustering fell above 88%, corresponding NMI and ARS scores were never above 70% and rarely above 60%. We are unsure why this happened so consistently.

SIGNIFICANCE

- After extensively testing K-Means and DBSCAN parameters, we found that the best F-Scores can be obtained by using K = 8 (K-Means) and eps = 0.8 with ms = 650 (DBSCAN). The respective F-Scores are 95.63% and 96.19%. We ran a parameter sweep on K-Means and found an F-Score asymptote as K increases beginning at K = 8. We further found that the K-Means F-Score plot results aligned with the popular K-Means sum of squares (elbow plot) results. Because of this, we are confident in using K = 8 for further tests on the NSL-KDD Dataset. However, because DBSCAN requires two parameters, one of which is boundless (namely minimum samples), we were unable to test every possible parameter combination. Thus, there possibly exists other parameters that would produce higher F-Scores and better clustering for DBSCAN.
- The preprocessing experiment on the NSL-KDD dataset, containing 7.3% of categorical data, showed that the best F-Score can be obtained by encoding categorical data with the risk values given in [5]; thus, our experiment verified the risk value experiment conducted by [5]. We also ran the missing values experiment on the IDS 2017 dataset, in which we found that whether one drops the categories with missing values entirely or imputes a statistically relevant value, the F-Score remains unaffected. However, only 0.0047% of the dataset contains missing values; thus, the missing values affect the IDS dataset much less than the categorical values affect the NSL-KDD dataset.
- Finally, we compared clustering algorithms K-Means and DBSCAN with non-clustering algorithms Local Outlier Factor and Isolation Forest and found the clustering algorithms to

produce significantly higher F-Scores. K-Means proved to be significantly faster than DBSCAN, with an average run time of 0.0000188 seconds compared respectively to 214.2 seconds, as expected due to the complexity of the respective algorithms. As mentioned above, the best F-Score from DBSCAN is 96.19% and from K-Means is 95.63%, showing that DBSCAN is slightly more accurate than K-Means. While DBSCAN and K-Means obtained F-Scores above 95%, Local Outlier Factor and Isolation Forest both obtained F-Scores below 60%. Because DBSCAN and K-Means are both clustering algorithms, these results favoring DBSCAN and K-Means indicate a large potential for clustering algorithms in network data analysis applications.

FUTURE WORK

- ❑ **Improving Code**
  - ❑ User Interface
    - ❑ Currently the code asks the user whether he or she wants to analyze the NSL-KDD dataset or the IDS 2017 dataset. The code is limited to these two datasets because of the specific way it deals with the categorical features, missing values, and labels. A significant improvement to the code would be to automate these processes.
      - ❑ Categorical features: Automatically check the data for features with categorical data, report which features have categorical data as well as the percentage of categorical data in the whole dataset, and ask the user if he or she wants to One Hot Encode or omit each feature individually. Provide an option to One Hot Encode or omit all features at once, for the user's ease.
      - ❑ Missing Values: Automatically check the data for features with missing values, report which features have missing values as well as the percentage of missing values in the dataset, and ask the user if he or she wants to remove features with missing values, impute mean, median, or mode for missing values, or use the smart imputer provided by python.
      - ❑ Labels: Ask user what column in the dataset that the labels are located in. Remove this column from the experimental data. Return the attributes of this column. Ask user how he/she would like to encode these attributes: for each feature, allow user to input desired label (for example, interface would say "normal: " and user would type "normal," or interface would say "neptune: " and user would type "DoS" for multiclass or "Attack" for binary labels.
    - ❑ Benefit: code could be used on many different datasets, not just NSL-KDD and IDS 2017
    - ❑ Drawback: the label encoding portion would take significantly more user input, increasing the complexity of running a test and requiring users to have a good grasp on the datasets they are using as well as how the code works.
- ❑ **Preprocessing**
  - ❑ **Dimensionality Reduction**

        - ❑ Because IDS 2017 was too large to be successfully clustered by DBSCAN, we think that reducing unnecessary dimensions could reduce the size of the dataset enough to allow it to be clustered by DBSCAN.
        - ❑ Dimensionality Reduction not only decreases the number of features and thus decreases the complexity of the clustering problem, it can also increase the accuracy of clustering by getting rid of data that has no pertinent information and is thus just noise (Shilpa, et. al., 2010).
        - ❑ Feature Selection
            - ❑ Low variance filter
        - ❑ Dimensionality Reduction
            - ❑ PCA
- ❑ **Stream data mining**
    - ❑ Looking at real-world, commercial applications of this theoretical project

Works Cited

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. "A Detailed Analysis of the KDD CUP 99 Data Set," *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

CIC: Canadian Institute for Cyber Security. "NSL-KDD Dataset." University of New Brunswick.

Craiger, Philip [Computer and Network Security]. (2014 June 25). *Introduction to Intrusion Detection* [Video file]. Retrieved from https://www.youtube.com/watch?v=VPLSIsRegFI.

Davis, Michael. (2017 March 5.) *Algorithms for Anomaly Detection - Lecture 1* [Video file]. Retrieved from https://cds.cern.ch/record/2254858?ln=en.

Omar, S., Ngadi, A., and Jebur, H. (2013). "Machine Learning Techniques for Anomaly Detection: An Overview." *International Journal of Computer Applications*, Vol. 79, No. 2. Retrieved from https://pdfs.semanticscholar.org/0278/bbaf1db5df036f02393679d485260b1daeb7.pdf.

Seif, George. (2018). "The 5 Clustering Algorithms Data Scientists Need to Know." *Towards Data Science*. Retrieved from: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68.

Pierobon, Gabriel. (2018). "DBSCAN clustering for data shapes k-means can't handle well (in Python). *Towards Data Science*. Retrieved from: https://towardsdatascience.com/dbscan-clustering-for-data-shapes-k-means-cant-handle-well-in-python-6be89af4e6ea.

Juanchaiyaphum, J., Arch-Int, N. Arch-Int, S., Saiyod, S. (2014). "Symbolic Data Conversion Method Using the Knowledge-Based Extraction in Anomaly Intrusion Detection System." *Journal of Theoretical and Applied Information Technology,* Vol. 65, No. 3. Retrieved from http://www.jatit.org/volumes/Vol65No3/13Vol65No3.pdf.

Mittag, F., Romer, M., and Zell, A. (2015). "Influence of Feature Encoding and Choice of Classifier on Disease Risk Prediction in Genome-Wide Association Studies." *PLOS One: A Peer Reviewed, Open Access Journal*, 10(8): e0135832. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4540285/.

Fodor, I. (2002). "A Survey of Dimension Reduction Techniques." Lawrence Livermore National Lab., CA (US). Retrieved from https://www.osti.gov/biblio/15002155.

Bholowalia, P. and Kumar, A. (2014). "EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN." *International Journal of Computer Applications*, Vol. 105, No. 9. Retrieved from https://pdfs.semanticscholar.org/5771/aa21b2e151f3d93ba0a5f12d023a0bfcf28b.pdf.

Kumar, V., Chauhan, H., Panwar, D. (2013). "K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset." *International Journal of Soft Computing and Engineering*, Vol. 3, Issue 4. Retrieved from https://pdfs.semanticscholar.org/7cc4/0f2da30d539dd86d5529d8cbca3afc4a4d2a.pdf.

Hale, J. (2019). "Scale, Standardize, or Normalize with Scikit-Learn." *Towards Data Science*. Retrieved from https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02.

Pedregosa *et al.* (2011). Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

Shilpa, L., Sini, J., Bhupendra, V.. (2010). Feature Reduction using Principal Component Analysis for Effective Anomaly–Based Intrusion Detection on NSL-KDD. International Journal of Engineering Science and Technology. 2. Retrieved from https://www.researchgate.net/publication/50281791_Feature_Reduction_using_Principal_Component_Analysis_for_Effective_Anomaly-Based_Intrusion_Detection_on_NSL-KDD/citation/download.