UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

KDD CUP 1999 Dataset

Semester Project Report

Group-1

Kaushal Arvindbhai Dhanani- 2154027
Maheshwari Korra- 2188882
Nikhitha Maddineni- 2154366
Rushil Sai Daivala- 2150299
Yogendra Sai Vineel Gurvishetty- 2217981

INTRODUCTION:

The KDD Cup 99 dataset is a widely used dataset for the evaluation of **network intrusion detection** systems. It contains a wide variety of simulated network intrusions, including both attacks and normal connections, making it useful for training and evaluating predictive models for network security.
The KDD Cup 99 dataset is a multivariate dataset with more than **300,000** instances. It is a computer-related dataset, with attributes that are either categorical or integer in nature. The dataset contains **42 attributes** in total. It is primarily used for **classification** tasks. There are **no missing values** in the dataset.

The KDD Cup 99 dataset is a dataset used for network intrusion detection. The problem to be solved using this dataset is to develop a predictive model that can accurately classify network intrusions as either normal connections or attacks. This is important because network intrusions can have serious consequences, such as data loss or unauthorized access to sensitive information. By using machine learning techniques to analyze the KDD Cup 99 dataset, it is possible to develop a model that can identify potential intrusions and alert network administrators, allowing them to take appropriate action to prevent or mitigate the effects of an attack.

PRE-PROCESSING OF THE DATA:

The dataset is loaded using the pandas library, and the matplotlib and seaborn libraries are used for visualizing the data. Then we performed the following preprocessing operations:

1) Import the necessary libraries: The pandas, matplotlib, and seaborn libraries are imported at the beginning to enable the use of their functions and methods.
2) Load the dataset: The read_csv() method from the pandas library is used to load the dataset from a CSV file. The low memory parameter is set to False to prevent the parser from being overly conservative when parsing the data.
3) Remove the first column: The first column in the data frame is a row numbering column, so it is removed using the drop() method. The columns parameter is used to specify the columns to be dropped, and the axis parameter is set to 1 to indicate that the columns should be dropped, rather than the rows. The inplace parameter is set to True to modify the existing data frame, rather than creating a new one.
4) Check for duplicated rows: The duplicated() method is used to check for rows that are duplicates of other rows in the data frame. The sum() method is then used to count the number of duplicated rows. This can be useful for identifying and removing duplicate data, which can affect the accuracy of machine learning models.

5) Check the data types: The info() method is used to display information about the data frame, including the data type of each column. This can be useful for ensuring that each column has the appropriate data type for its values.

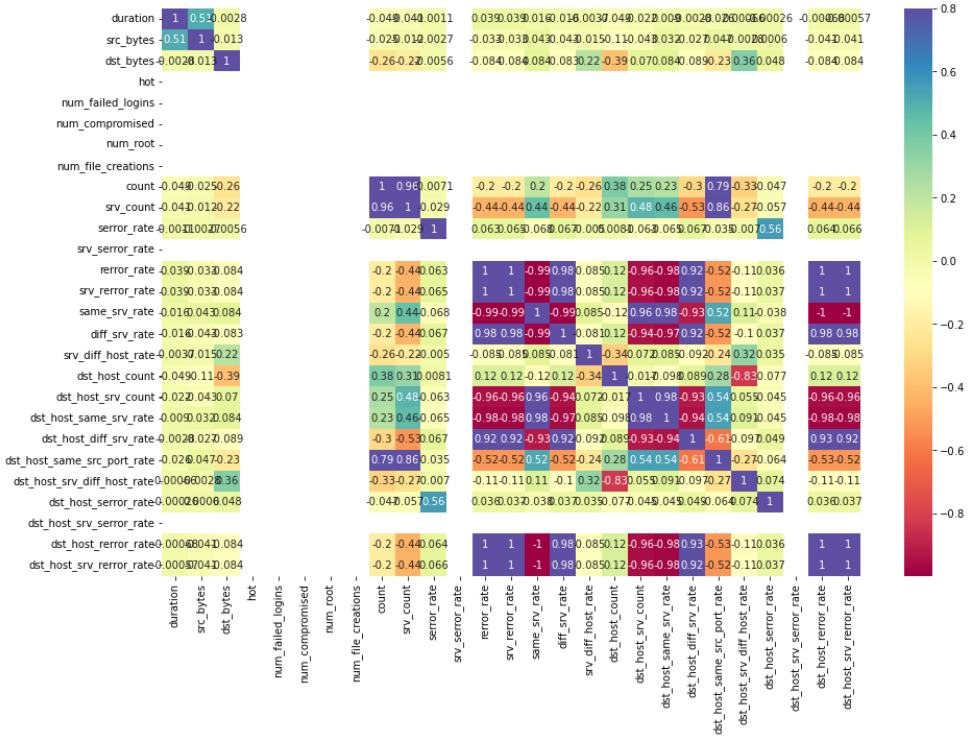
6) Convert the data in the "label" column: The "label" column contains the classification labels for the data, with "normal" values being labeled as 0 and other values being labeled as 1. The map() method is used to convert all "normal" values in the column to 0 and all other values to 1. This allows the data in the column to be treated as a binary classification target variable.

7) Convert some columns to categorical variables: Some of the columns in the data frame contain categorical data, so their data types are converted to the category type using the astype() method. This allows the data in these columns to be treated as categorical data, rather than numerical data.

8) Remove outliers: Outliers can have a negative effect. We remove any data points that are outside of 3 standard deviations from the mean for each of the numerical columns in the data set.

9) Save the cleaned data into various formats: We then saved the cleaned dataframe into three formats csv, pkl and npz.

Heatmap is used to show the correlation between different feature.



PERFORMANCE OF DIFFERENT BASE MODELS:

We have used Linear Ridge Classifier, KNearestNeighbors, Decision Trees, Random Forest, Support Vector Machine with non-linear kernel and Multi Layer Perceptron in our project.

Accuracy and Classification reports before Model Structure Selection:

1. Linear Classification:

A linear classifier is a model that determines the discrete class to which to assign a set of data points based on a linear combination of its explanatory variables. We have got 96.05% accuracy.

	precision	recall	f1-score	support
0	0.84	0.98	0.90	12941
1	1.00	0.96	0.98	55722
accuracy			0.96	68663
macro avg	0.92	0.97	0.94	68663
weighted avg	0.97	0.96	0.96	68663

2. KNN:

The k-nearest neighbors (KNN) algorithm is a data classification method for estimating the likelihood that a data point will become a member of one group or another. We got 97.53% accuracy for this.

	precision	recall	f1-score	support
0	0.97	0.89	0.93	12853
1	0.98	0.99	0.99	55810
accuracy			0.98	68663
macro avg	0.97	0.94	0.96	68663
weighted avg	0.98	0.98	0.97	68663

3. Random Forest:

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset. Accuracy = 97.93%

	precision	recall	f1-score	support
0	0.95	0.94	0.94	12881
1	0.99	0.99	0.99	55782
accuracy			0.98	68663
macro avg	0.97	0.97	0.97	68663
weighted avg	0.98	0.98	0.98	68663

4. Decision Trees:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. Accuracy of this model is 97.85%

	precision	recall	f1-score	support
0	0.95	0.94	0.94	12859
1	0.99	0.99	0.99	55804
accuracy			0.98	68663
macro avg	0.97	0.96	0.96	68663
weighted avg	0.98	0.98	0.98	68663

5. Support Vector Classifier with Non-linear kernel:

SVC works by mapping data points to a high-dimensional space and then finding the optimal hyperplane that divides the data into two classes. Accuracy = 96.29%

	precision	recall	f1-score	support
0	0.83	1.00	0.91	12718
1	1.00	0.95	0.98	55945
accuracy			0.96	68663
macro avg	0.92	0.98	0.94	68663
weighted avg	0.97	0.96	0.96	68663

6. Deep Learning: Multi Layer Perceptron

A Multilayer Perceptron has input and output layers, and one or more **hidden layers** with many neurons stacked together. Accuracy for this model is 96.61%

	precision	recall	f1-score	support
0	0.86	0.98	0.91	12718
1	0.99	0.96	0.98	55945
accuracy			0.97	68663
macro avg	0.93	0.97	0.95	68663
weighted avg	0.97	0.97	0.97	68663

MODEL PERFORMANCE AFTER MODEL STRUCTURE SELECTION AND HYPERPARAMETER TUNING:

We have used k-fold cross validation and GridSearchCV to find the values of hyperparameters.

- GridSearch involves using a different combination of all given hyperparameters and their values and find out the performance of each combination and selects the best value for hyperparameters.
- In a k-fold cross-validation, the data is broken into k-folds of almost equal sizes. In each step, $k - 1$ folds are used for training and the remaining fold is used for testing.

Algorithm	Accuracy
Linear Ridge Classifier	96.06%
KNN classifier	97.85%
Decision trees classifier	97.64%
Random forest classifier	97.93%
Support vector machine (non-linear kernel)	96.51%
Deep learning – Multi layer Perceptron	97.01%

Below charts depicts the classification reports after model structure selection:

Linear Ridge Classifier

	precision	recall	f1-score	support
0	0.83	0.98	0.90	12939
1	1.00	0.95	0.97	55724
accuracy			0.96	68663
macro avg	0.91	0.97	0.94	68663
weighted avg	0.96	0.96	0.96	68663

KNN

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.89	0.93	12853
1	0.98	0.99	0.99	55810
accuracy			0.98	68663
macro avg	0.97	0.94	0.96	68663
weighted avg	0.98	0.98	0.97	68663

Random Forest

	precision	recall	f1-score	support
0	0.95	0.94	0.94	12939
1	0.99	0.99	0.99	55724
accuracy			0.98	68663
macro avg	0.97	0.96	0.97	68663
weighted avg	0.98	0.98	0.98	68663

Decision Trees

	precision	recall	f1-score	support
0	0.95	0.94	0.94	12939
1	0.99	0.99	0.99	55724
accuracy			0.98	68663
macro avg	0.97	0.96	0.97	68663
weighted avg	0.98	0.98	0.98	68663

SVC (non-linear kernel rbf)

	precision	recall	f1-score	support
0	0.86	0.98	0.91	12718
1	0.99	0.96	0.98	55945
accuracy			0.97	68663
macro avg	0.93	0.97	0.95	68663
weighted avg	0.97	0.97	0.97	68663

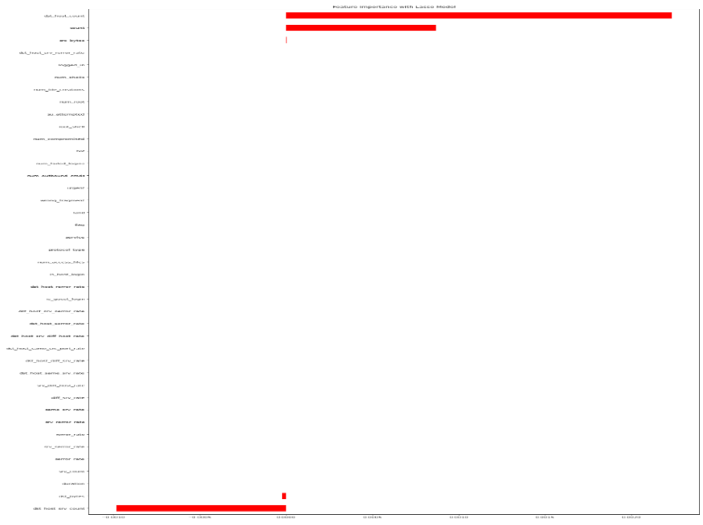
Deep learning: Multi layer perceptron

	precision	recall	f1-score	support
0	0.96	0.88	0.92	12884
1	0.97	0.99	0.98	55779
accuracy			0.97	68663
macro avg	0.97	0.93	0.95	68663
weighted avg	0.97	0.97	0.97	68663

Overall, comparing all the six modeling techniques after hyper parameter tuning using GridSearchCV , it is observed that Random Forest has highest accuracy of 97.93 and has highest precision and recall values. Hence, Random Forest modeling technique is chosen for this dataset which best fits the model.

FIRST VARIABLE SELECTION USING LASSOCV:

Lasso is a supervised algorithm where the process identifies the variables that are strongly associated with the response variable. This is called variable selection.
Random forest is the best model for our dataset. So, LassoCV is used to find the important features for Random forests modeling technique.
Below image shows the feature importance with Lasso Model.



From the graph, 'dst_host_count','count','src_bytes','dst_host_srv_count' are important features that are associated with target variable ‘Label’.
Performing LassoCV feature selection on Random Forest modeling technique yielded an accuracy of 97.21% .

```
print('Accuracy: {:.4f}'.format(accuracy_score(y_test, y_pred)))
```

```
Accuracy: 0.9721
```

```
#classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93     12951
           1       1.00      0.97      0.98     55712

    accuracy                           0.97     68663
   macro avg       0.94      0.98      0.96     68663
weighted avg       0.98      0.97      0.97     68663
```

BI-DIRECTIONAL ELIMINATION METHOD

Bidirectional Elimination is a method of fitting models in which the choice of predictive variables is carried out by an automatic procedure. In each step, a variable is considered for addition to or subtraction from the set of explanatory variables based on some prespecified criterion.
Bidirectional elimination is a combination of Forward selection and Backward elimination, testing at each step for variables to be included or excluded. We have selected 7 most important features, that we got from this method for training.

```
sffs.k_feature_names_
```

```
('duration',
 'src_bytes',
 'dst_bytes',
 'srv_count',
 'dst_host_count',
 'dst_host_srv_count',
 'dst_host_same_src_port_rate')
```

We have trained the model with these features on the best model, in our case ‘**Random Forest**’. We got the accuracy of 97.88% for this model.

```
print('Accuracy: {:.4f}'.format(accuracy_score(y_test, y_pred)))
```

```
Accuracy: 0.9788
```

```
#classification report
print(classification_report(y_test, y_pred))
```

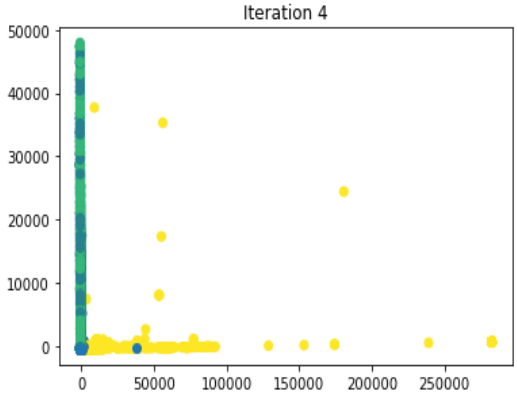
```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94     12913
           1       0.99      0.99      0.99     55750

    accuracy                           0.98     68663
   macro avg       0.97      0.96      0.97     68663
weighted avg       0.98      0.98      0.98     68663
```

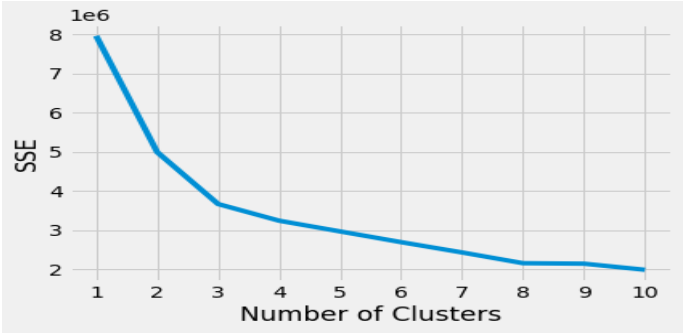
CLUSTERING:-

Clustering is the task of dividing the population or data points into several groups such that data points in the same groups are more like other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects based on similarity and dissimilarity between them.

K-means clustering is a clustering technique that is being used in our project. K-means clusters data points into unique, non-overlapping groupings. It halts creating and optimizing clusters when either the centroids have stabilized, or the defined number of iterations has been achieved.

We have visualized the clusters with the help of PCA and we found the optimum number of clusters as 3. We got the silhoute score of 0.72, so the cluster shows strong structure.



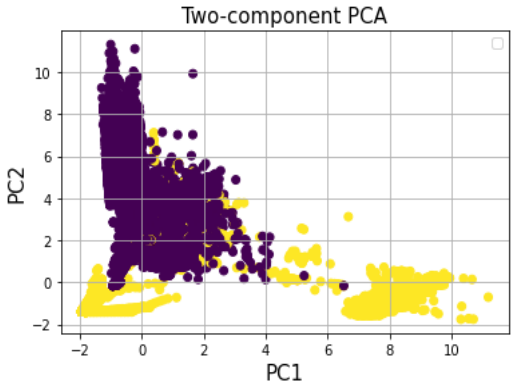


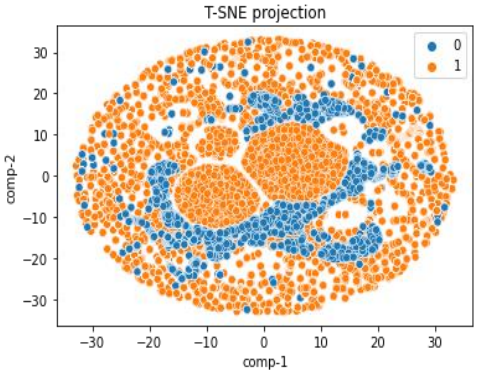
VISUALIZATION

Dimensionality reduction is a machine learning technique of reducing number of random variables in a problem by obtaining a set of principal variables and helps in reducing the system computation time.

In our project we have used **Principal Component Analysis (PCA) for linear dimensionality reduction** and **T-SNE (t Distributed Stochastic Neighborhood Embeded) technique for non linear dimensionality** reduction.

Below we have shown the visualization of our dataset after reducing its dimensions with PCA and t- SNE dimension reduction techniques.





One of the most important application of Dimensionality reduction is **Data Visualization.** We can drop the dimensions of high dimensional dataset and can visualize the dataset on a 2d or 3d plot. The goal of dimension reduction for data visualization is to take high dimensional data and project it down to 2 or 3 dimensions so that humans can understand its structure.

ENSEMBLE

It is a process of running two or more machine learning algorithms and then combine the results into single score in order to improvise the accuracy of model.

We have used Voting classifier for ensemble learning in our project. A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

It aggregates the findings of every classifier passed into Voting Classifier and predicts the output class based on the majority of voting.

We have taken the 20 different models in ensemble and predicted the outputs with Voting Classifier.

Below shows the accuracy score and classification report for ensemble.

```
print(classification_report(y_test, VC_pred), "\n")
```

```
Voting Classifier Classification Report on Test data using best model
              precision    recall  f1-score   support

           0       0.94      0.95      0.94     12896
           1       0.99      0.98      0.99     55767

    accuracy                           0.98     68663
   macro avg       0.96      0.97      0.96     68663
weighted avg       0.98      0.98      0.98     68663
```

```
: print('Accuracy: {:.4f}'.format(accuracy_score(y_test, VC_pred)))
```

```
Accuracy: 0.9776
```

After comparing the combined accuracy score with the individual performance of the model, we can say that the ensemble is better technique than individual models.

GENERAL DISCUSSION

If we could restart the project, we would have done better in data preprocessing. We would have taken the random sample of 100,000 observations from our dataset, so that the model can be trained faster. We would have used LabelEncoder to encode the data from starting of the project to reduce the time in hyperparameter training. It was taking lot of time to tune after creating dummies.

We can use more advanced machine learning algorithms such as ANN, CNN if we had more time. Also, we would have gone deeper into algorithms that we have used in the project and optimized them in more effective way. We would go in depth for Extreme Learning Machines, understand it completely and implement in our dataset.

Classification problem of our data was real world problem. Working on this project made us acquire more knowledge about implementing various machine learning algorithms and optimize their hyperparameters on real data. Also, during the project we learnt about organizational skills, teamwork and presentation skills.

CONCLUSION

The essence of this project was to learn and implement the different steps in a machine learning project. We can conclude after comparing all the results that Random Forest is the best model with best accuracy score of 97.93% for our dataset.

REFERENCES

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

https://towardsdatascience.com/what-why-and-how-of-t-sne-1f78d13e224d

https://www.analyticsvidhya.com/blog/

https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

https://towardsdatascience.com/