INTRUSION DETECTION MODEL USING MACHINE LEARNING ALGORITHM

TABLE OF CONTENTS

OBJECTIVE:

The objectives of the intrusion detection model using ml on big data environment are:

- To provide spark-chi-svm model for intrusion detection that can deal with big data
- To provide a high performance and reducing the training time and efficient algorithm for big data
- Reduce the computations, to provide high speed and a low of false positive alarms rate
- To provide optimal algorithm for the considered dataset to get the highest precision.

SUCCESS CRITERIA:

A multi-class model that can identify different attack types and combine network and host-based intrusion detection systems for better detection can be added to the model in the future.

TARGET DATA:

- We have acquired the dataset from common websites like Kaggle (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html).
- One of the few methods used to acquire the dataset is web scraping, an automated method of gathering data from the web is called web scraping.
- The dataset considered is kdd99 (knowledge discovery in databases).

- The core of the kdd approach is machine learning and data mining, which uses inference of algorithms to analyze the data, build the model, and discover previously undetected patterns
- Kdd99 is a relational data set which contains multiple values that has different fields containing categorical features, binary features, discrete features, and continuous features.
- A relational database makes it straightforward to view and understand how different data formats connect to one another by storing data in one or more tables (or "relations") of columns and rows.
- Relational databases use preestablished relationships to arrange data.

STATEMENT:

- Information security and data analysis systems for big data have recently changed in relevance due to the enormous amounts of data and their gradual growth.
- A system called an intrusion detection system (ids) monitors and analyses data to find any incursion in The network or system.
- The network generates data at a high rate, volume, and variety, making it highly challenging to identify assaults using conventional methods.
- Ids uses big data approaches to handle big data for precise and effective data analysis. The spark-chi-svm model for intrusion detection was introduced in this paper.
- On the Apache spark big data platform, we constructed an intrusion detection model utilizing a support vector machine (svm) classifier and chi-square selector for feature selection.
- Kdd99 was used to train and evaluate the model. We included a comparison between the chi-svm classifier and the chi-logistic regression classifier in the experiment.
- The experiment's findings demonstrated the excellent performance, shorter training time, and effectiveness of the spark-chi-svm model for big data.

BACKGROUND INFORMATION

Big data are the kind of data that are challenging to handle, store, and analyze with conventional database and software methods. Big data encompasses a huge volume and velocity of data as well as a variety of data that requires novel handling methods. A hardware or software monitor called an intrusion detection system (ids) examines data to find any attacks on a system or network. When dealing with big data, traditional intrusion detection system approaches make the system more complicated and inefficient since its analytical properties process is difficult and time-consuming. The system is vulnerable to damage for a while before receiving any alerts due to the lengthy data analysis process. Therefore, analyzing and storing data in intrusion detection systems utilizing big data tools and techniques helps speed up calculation and training.

The ids have three traditional methods for detecting attacks:

- Signature- based detection.
- Anomaly- based detection, and
- Hybrid – based detection.

Signature - based detection:

By utilizing the signatures of those assaults, signature-based detection is intended to identify known attacks. It works well for identifying pre-loaded known assaults in the ids database. As a result, it is frequently thought that an intrusion attempt or recognized assault is considerably more accurate. The databases are regularly updated in order to improve their efficacy of detections, however new forms of attacks cannot be found since their signature is not displayed. To resolve this issue to find aberrant behaviors that might represent intrusions, anomaly-based detection is utilized to evaluate current user activity against predetermined profiles.

Anomaly - based detection

Without requiring system upgrades, anomaly-based detection is effective against unknown or zero-day assaults. However, the false positive rates for this approach are frequently substantial. To find aberrant behaviors that might represent intrusions, anomaly-based detection is utilized to evaluate current user activity against predetermined profiles.

Hybrid - based detection

In order to get beyond the drawbacks of using just one intrusion detection approach while maximizing the benefits of using two or more, hybrid-based detection combines two or more methods. 4 for intrusion detection, several studies have suggested machine learning algorithms to lower false positive rates and create accurate ids. The typical machine learning approaches, however, require a long time to learn and categorize data when dealing with big data. Ids may overcome numerous difficulties, such as speed and computing time, by using big data approaches and machine learning.

MACHINE LEARNING:

Machine learning refers to a system that can learn from examples by getting better on its own and without having to be explicitly programmed. The innovation is based on the notion that a machine can create correct results just by learning from the data (i.e., examples). To forecast an outcome, machine learning integrates data with statistical methods. Corporate uses this output to provide insights that can be put into practice. The computer takes data as input and generates replies using an algorithm. Machine learning is also utilized for a wide range of tasks, including automating tasks, optimizing portfolios, detecting fraud, and more. Learning and inference are at the heart of machine learning. The first way the machine learns is by identifying patterns. The data allowed for this finding to be made. The data scientist's ability to carefully select the data to provide the computer is one of their most important skills. A feature vector is a set of qualities that is used to solve an issue. A feature vector may be thought of as a subset of data that is utilized to solve a problem. The machine simplifies reality using some sophisticated algorithms, turning this discovery into a model. As a result, the data are described and condensed into a model during the learning step.

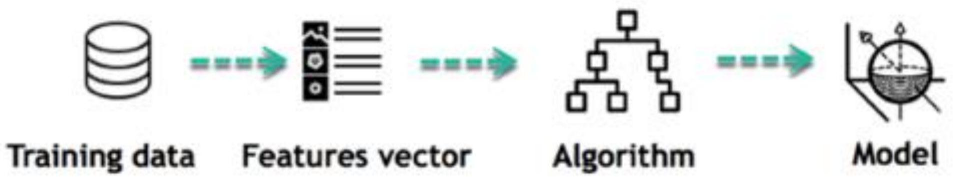


Fig: machine learning phases

MACHINE LEARNING ALGORITHMS: -

We are using 3 Algorithms to model and implement the data set they are: -

1. Support Vector Machine Classifier
2. Random Forest Classifier
3. Logistic Regression Classifier

Support Vector Machine Classifier:

Support Vector Machine Classifier also called as SVM this is a supervised ML algorithm. This can be used for Regression and classification both but we are using it to classify. In SVM whatever you want to classify is represented as a point in n-dimension space and then SVM classifies it by drawing a hyperplane that is a line in 2d or a plane in 3d. In such a way that all points of one category or the one side of hyperplane and the other categories are on the other side of the hyperplane. There can be many Hyperplanes that divide all the points but the best one is chosen in such a way by SVM thus the distance between them is more this distance is called margin. To implement this, we need a training data set that is already classified and labeled. In background SVM solves Converse Optimization Problem that maximizes the margin and also which side contains which part of the category. But you need not worry about all these problems in implementation because implementing SVM is as easy as implementing libraries. The Biggest Pros are they are easy to understand, implement, use and interpret.

Random Forest Classifier:

Random forest Classifier is a supervised Machine learning algorithm that generates and combines many decision trees into a single forest. Let's understand Random Forest Classifier in Layman's terms. Suppose you want to go somewhere nice for your winter break. To decide on a place where you can go you can either search on net or ask a friend or pick a random spot. Let's assume that you decide to ask your friends then they recommended couple of places that they have been to. Now you make a list of those recommendations and ask everyone to vote and the place with most votes is the place you decide to visit. In summary you are going through 2 steps here

Step 1 is asking your friends for recommendations this part is like using the Decision Tree

Step 2 is after collecting recommendations the voting process.

This whole process of getting recommendations and voting on this is called Random Forest Classifier.

Logistic Regression Classifier:

Logistic Regression is a technique we usually use in binary classification now what is binary classification. A binary classification output can only take one of these two possible outcomes they can be anything from 0/1, True/False, if a mail is Spam/Not or even if a transaction is Fraud/Not.

I'll take an easy example with only one feature to explain Logistic Regression.

On X axis we'll have age that's our feature and on Y axis we have heart disease probability that is o/1.

Now how to make new predictions using this dataset.

We draw an approximate straight line that fits the data set and we consider that as probability.

Consider x as the threshold or tipping value so every output more than these x is considered as 1 and every output less than this x is considered as 0.

Sometimes this process gives wrong output so instead of straight line we draw a curve that's called sigmoid curve which suits for the data better and gives us more precision outputs.

SCOPE OF THE STUDY:

- The Traditional detection of intrusion detection became very difficult due to high volume, variety of data that is being generated. So, we are using big data in IDS to get better effective results.
- The growth of Intrusion Prevention.
- Advances in Source Determination.

DATA SOURCES:

The Dataset we are using is KDD99 Dataset it is used to evaluate the proposed model. The variety of instances which are used are same to 494,021 and it has 41 attributes/functions and the class attributes shows if a given example is normal or an attack.

Data exploration and visualization: -

For organizations, gathering and storing data about every aspect of their operations is now simpler than ever. Understanding the consequences and opportunities hidden in that data rapidly is a difficulty faced by business leaders.

In data science, data exploration is a crucial step. In order to help businesses and organizations grasp insights and adopt new policies, analysts examine a dataset to highlight specific patterns or traits.

Although data exploration doesn't always uncover every little nuance, it does provide researchers a better overall picture of certain trends or subject areas. Users examine data to choose the optimal model or algorithm for further data analysis processes using both human and automated methods and tools.

Users can utilize manual data exploration approaches to pinpoint specific areas of interest, which is useful but falls short of more thorough research. Here is where machine learning can advance your data analysis.

Automated exploration software or machine learning algorithms can quickly find links between different data variables and dataset structures to check for outliers and provide data values that can draw attention to patterns or interesting areas.

Machine learning and data exploration both have the ability to find interesting patterns in datasets and assist in making inferences. However, users may quickly and accurately extract information from massive databases using machine learning.

Many businesses are struggling with an abundance of data but not enough resources to evaluate and process it as a result of the greater availability of data than ever before. Machine learning can help in this situation.

Exploratory data analysis using machine learning aids data scientists in keeping track of their data sources and sifting thru data for big analyses. Manual data exploration is exquisite for focusing in on precise datasets of interest, however ML gives a wider lens and actionable insights which can revolutionize your company's knowhow of styles and trends.

Additionally, ML tools can make data much simpler to understand. Companies may quickly extract valuable information from data points by exporting them to DV displays like scatter plots or bar charts, saving time on understanding and challenging results.

You can gain in-depth insights that help you make better decisions when you start exploring your data with automated data exploration tools. Open-source tools with regression capabilities and visualization techniques that use programming languages like Python for data preparation are part of today's machine learning solutions.

Finding the variables: -

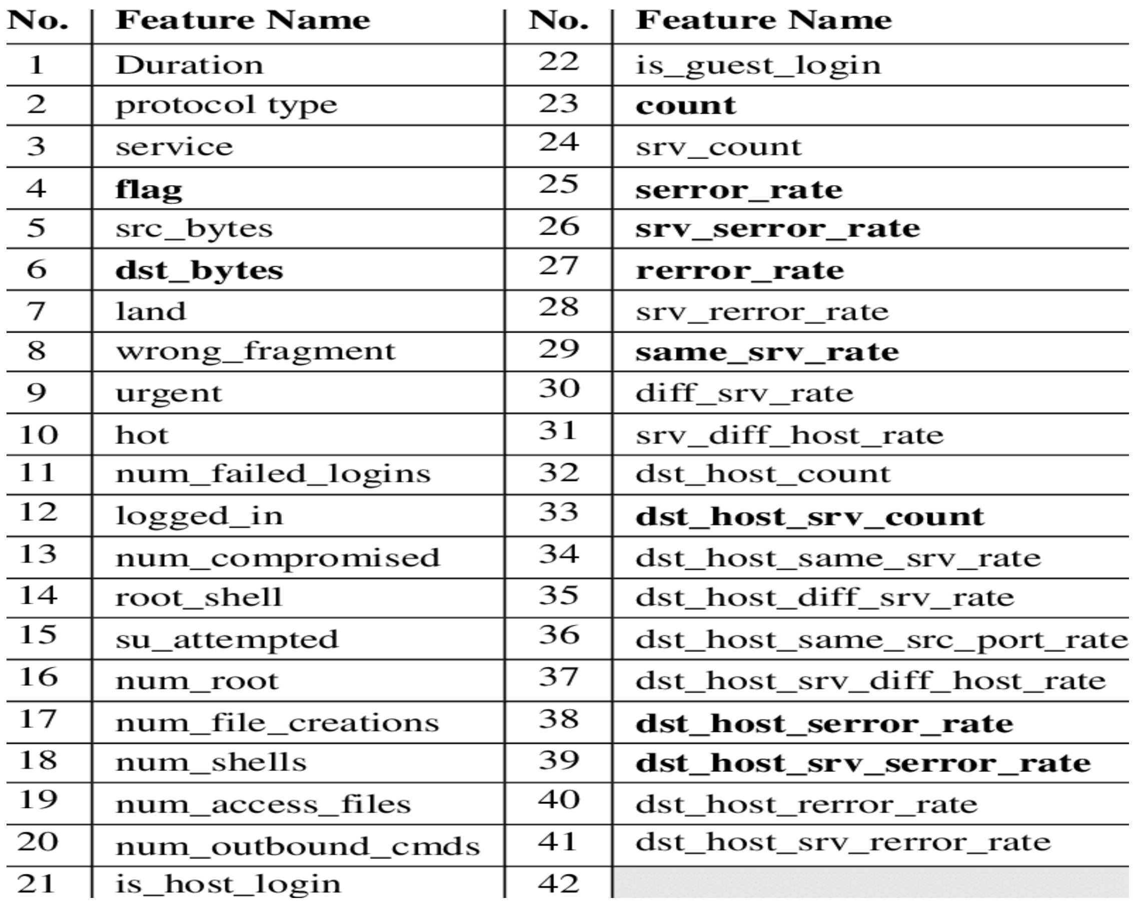
Data scientists will first identify the variables that vary or have the potential to change. The variables' categorization and data type will then be determined by scientists.

KDD99 is the dataset being examined (Knowledge Discovery in Databases).

Machine Learning and Data Mining, which employ inference of algorithms to evaluate the data, create the model, and identify previously unnoticed patterns, are the foundation of the KDD approach.

KDD99 is a relational data collection that includes several values in addition to categorical, binary, discrete, and continuous features in several fields. By storing data in one or more tables (or "relations") of columns and rows, relational databases make it simple to observe and comprehend how various data types relate to one another.

Relational databases arrange data using pre-existing relationships.



No.	Feature Name	No.	Feature Name
1	Duration	22	is_guest_login
2	protocol type	23	**count**
3	service	24	srv_count
4	**flag**	25	**serror_rate**
5	src_bytes	26	**srv_serror_rate**
6	**dst_bytes**	27	**rerror_rate**
7	land	28	srv_rerror_rate
8	wrong_fragment	29	**same_srv_rate**
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	**dst_host_srv_count**
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	**dst_host_serror_rate**
18	num_shells	39	**dst_host_srv_serror_rate**
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login	42	▨

Fig: Dataset

df.shape (494020, 42)

The Data types of our data set consists of

- Int64

- Object

- Float64

Understanding your data through descriptive statistical analysis is crucial for machine learning. This is as a result of machine learning's focus on prediction. Statisticians, on the other hand, focus on making inferences from data, which is an essential first step.

Normal Distribution: -

Since nearly all statistical tests require normally distributed data, one of the most crucial notions in statistics is the normal distribution. It essentially defines the appearance of big samples of data when they are plotted. The "Gaussian curve" or "bell curve" are two more names for it.

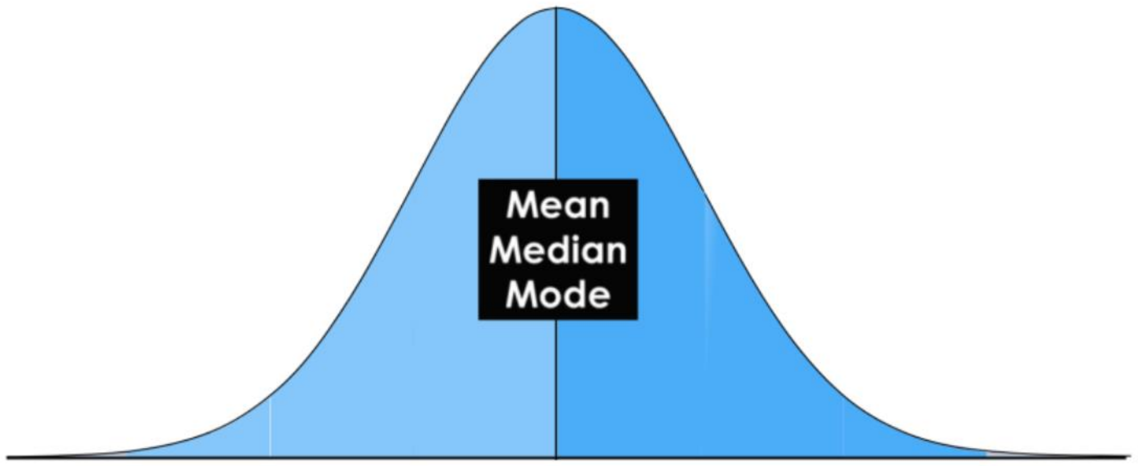


Fig: Normal distribution

A normal distribution must be provided in order to calculate probabilities and use inferential statistics. In essence, this means that you should be extremely cautious about the statistical tests you use to data that is not regularly distributed because they may result in incorrect results.

If the data is symmetrical, bell-shaped, centered, and unimodal, a normal distribution is implied.

These measurements all fall at the same midpoint in a normal distribution. The median, mode, and mean are therefore all equivalent.

Central Tendency:

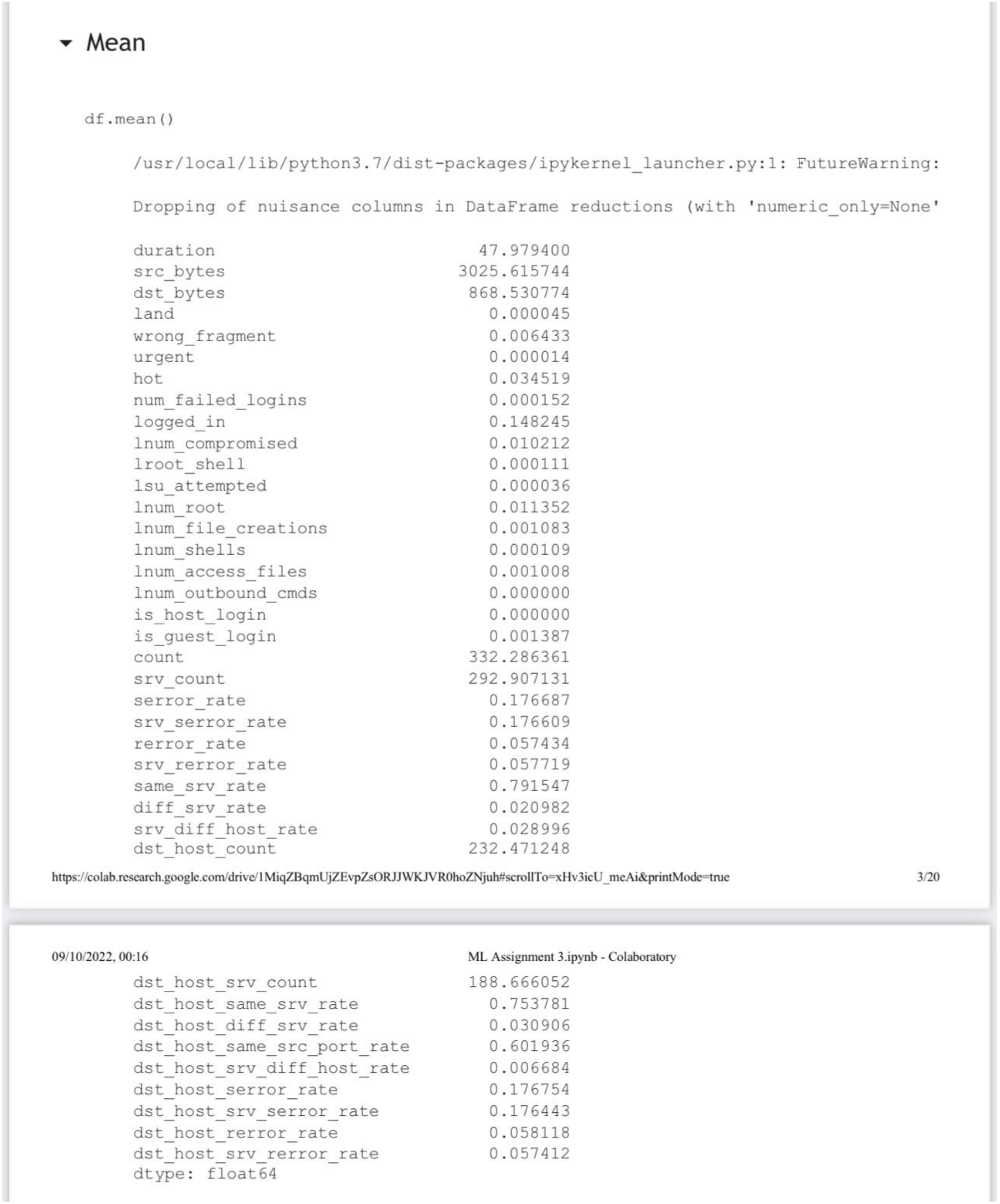
The mean, which is just the average, is thought to be the most accurate indicator of central tendency for drawing conclusions about the entire population from a small sample. The tendency for your data's values to cluster around its mean, mode, or median is known as central tendency. The mean is calculated by dividing the total number of values by the sum of all the values.

The value or category that appears the most frequently in the data is the mode. Therefore, if no number is repeated or if no category is the same, a dataset does not have a mode. A dataset could have more than one mode.

The only measure of central tendency that can be used to categorical data is the mode.

The median, which is also known as the "50th percentile," is the "middle" value or midpoint in your statistics. The median is significantly less impacted by outliers and skewed data than the mean, it should be noted.

MEAN:



▾ Mean

```
df.mean()

    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:

    Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None'

    duration                         47.979400
    src_bytes                      3025.615744
    dst_bytes                       868.530774
    land                              0.000045
    wrong_fragment                    0.006433
    urgent                            0.000014
    hot                               0.034519
    num_failed_logins                 0.000152
    logged_in                         0.148245
    lnum_compromised                  0.010212
    lroot_shell                       0.000111
    lsu_attempted                     0.000036
    lnum_root                         0.011352
    lnum_file_creations               0.001083
    lnum_shells                       0.000109
    lnum_access_files                 0.001008
    lnum_outbound_cmds                0.000000
    is_host_login                     0.000000
    is_guest_login                    0.001387
    count                           332.286361
    srv_count                       292.907131
    serror_rate                       0.176687
    srv_serror_rate                   0.176609
    rerror_rate                       0.057434
    srv_rerror_rate                   0.057719
    same_srv_rate                     0.791547
    diff_srv_rate                     0.020982
    srv_diff_host_rate                0.028996
    dst_host_count                  232.471248
```

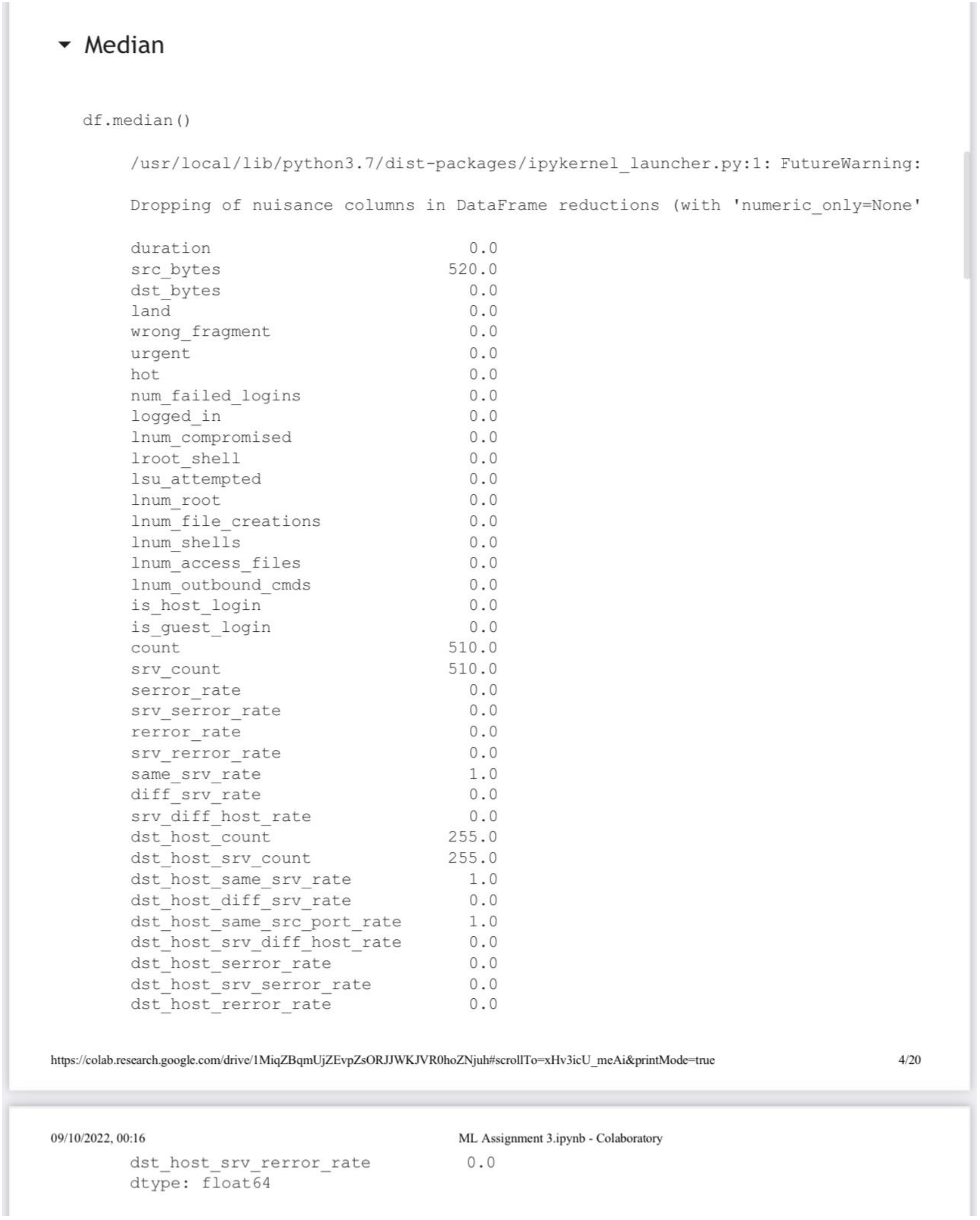
https://colab.research.google.com/drive/1MiqZBqmUjZEvpZsORJJWKJVR0hoZNjuh#scrollTo=xHv3icU_meAi&printMode=true 3/20

09/10/2022, 00:16 ML Assignment 3.ipynb - Colaboratory

```
    dst_host_srv_count              188.666052
    dst_host_same_srv_rate            0.753781
    dst_host_diff_srv_rate            0.030906
    dst_host_same_src_port_rate       0.601936
    dst_host_srv_diff_host_rate       0.006684
    dst_host_serror_rate              0.176754
    dst_host_srv_serror_rate          0.176443
    dst_host_rerror_rate              0.058118
    dst_host_srv_rerror_rate          0.057412
    dtype: float64
```

Fig : Mean

Median:



▾ Median

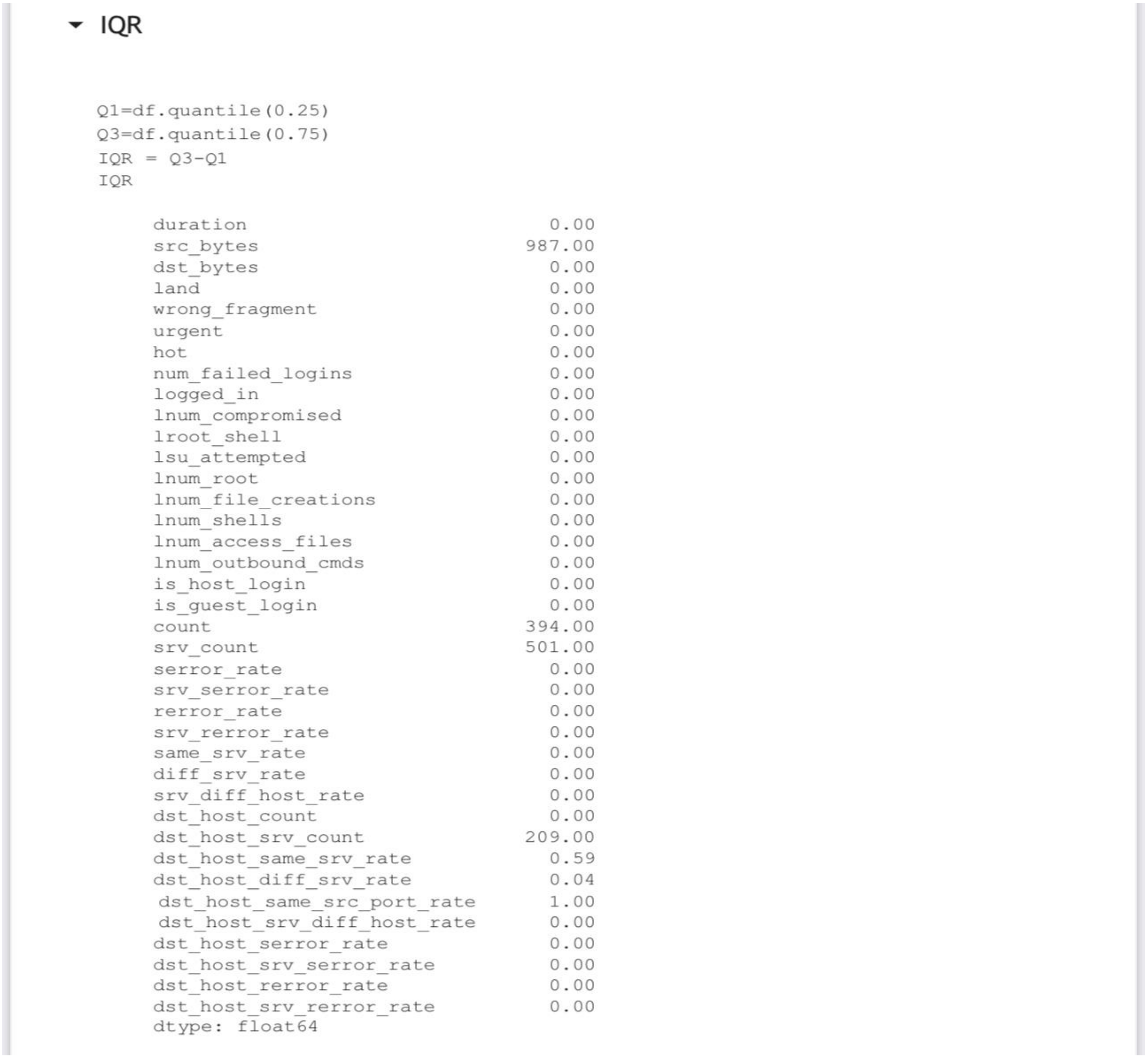
```
df.median()

    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:

    Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None'

    duration                         0.0
    src_bytes                      520.0
    dst_bytes                        0.0
    land                             0.0
    wrong_fragment                   0.0
    urgent                           0.0
    hot                              0.0
    num_failed_logins                0.0
    logged_in                        0.0
    lnum_compromised                 0.0
    lroot_shell                      0.0
    lsu_attempted                    0.0
    lnum_root                        0.0
    lnum_file_creations              0.0
    lnum_shells                      0.0
    lnum_access_files                0.0
    lnum_outbound_cmds               0.0
    is_host_login                    0.0
    is_guest_login                   0.0
    count                          510.0
    srv_count                      510.0
    serror_rate                      0.0
    srv_serror_rate                  0.0
    rerror_rate                      0.0
    srv_rerror_rate                  0.0
    same_srv_rate                    1.0
    diff_srv_rate                    0.0
    srv_diff_host_rate               0.0
    dst_host_count                 255.0
    dst_host_srv_count             255.0
    dst_host_same_srv_rate           1.0
    dst_host_diff_srv_rate           0.0
    dst_host_same_src_port_rate      1.0
    dst_host_srv_diff_host_rate      0.0
    dst_host_serror_rate             0.0
    dst_host_srv_serror_rate         0.0
    dst_host_rerror_rate             0.0
```

```
    dst_host_srv_rerror_rate         0.0
    dtype: float64
```

IQR:

An indicator of statistical dispersion between the upper (75th) and lower (25th) quartiles is the interquartile range (IQR).

The interquartile range is a measurement of where the bulk of the values reside, whereas the range determines the starting and stopping points of each data point.



▾ IQR

```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR = Q3-Q1
IQR
```

```
duration                       0.00
src_bytes                    987.00
dst_bytes                      0.00
land                           0.00
wrong_fragment                 0.00
urgent                         0.00
hot                            0.00
num_failed_logins              0.00
logged_in                      0.00
lnum_compromised               0.00
lroot_shell                    0.00
lsu_attempted                  0.00
lnum_root                      0.00
lnum_file_creations            0.00
lnum_shells                    0.00
lnum_access_files              0.00
lnum_outbound_cmds             0.00
is_host_login                  0.00
is_guest_login                 0.00
count                        394.00
srv_count                    501.00
serror_rate                    0.00
srv_serror_rate                0.00
rerror_rate                    0.00
srv_rerror_rate                0.00
same_srv_rate                  0.00
diff_srv_rate                  0.00
srv_diff_host_rate             0.00
dst_host_count                 0.00
dst_host_srv_count           209.00
dst_host_same_srv_rate         0.59
dst_host_diff_srv_rate         0.04
 dst_host_same_src_port_rate   1.00
 dst_host_srv_diff_host_rate   0.00
dst_host_serror_rate           0.00
dst_host_srv_serror_rate       0.00
dst_host_rerror_rate           0.00
dst_host_srv_rerror_rate       0.00
dtype: float64
```

Skewness :

A distribution's skewness can be measured in terms of its symmetry.

As a result, it specifies how far a distribution deviates to the left or right from the normal distribution. It is possible for the skewness value to be positive, negative, or zero. Because the mean and median are equal, a perfect normal distribution would have a skewness of zero.

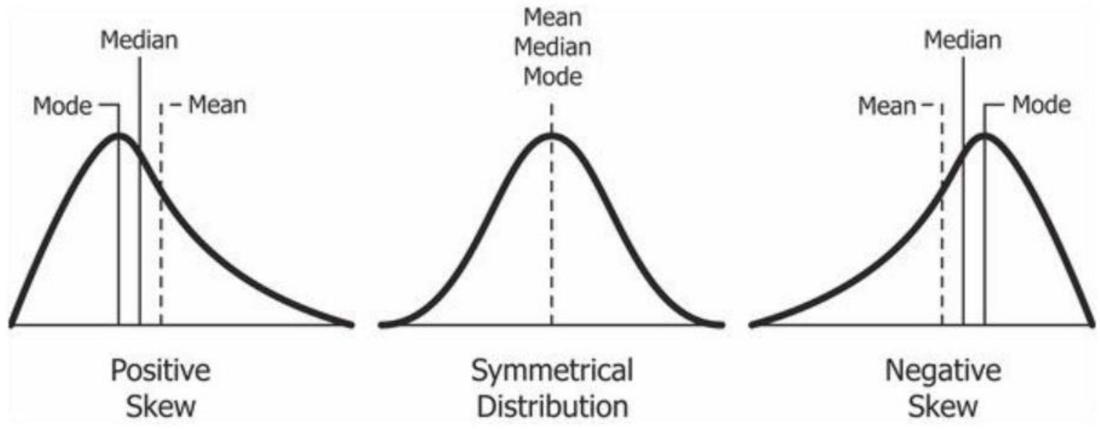
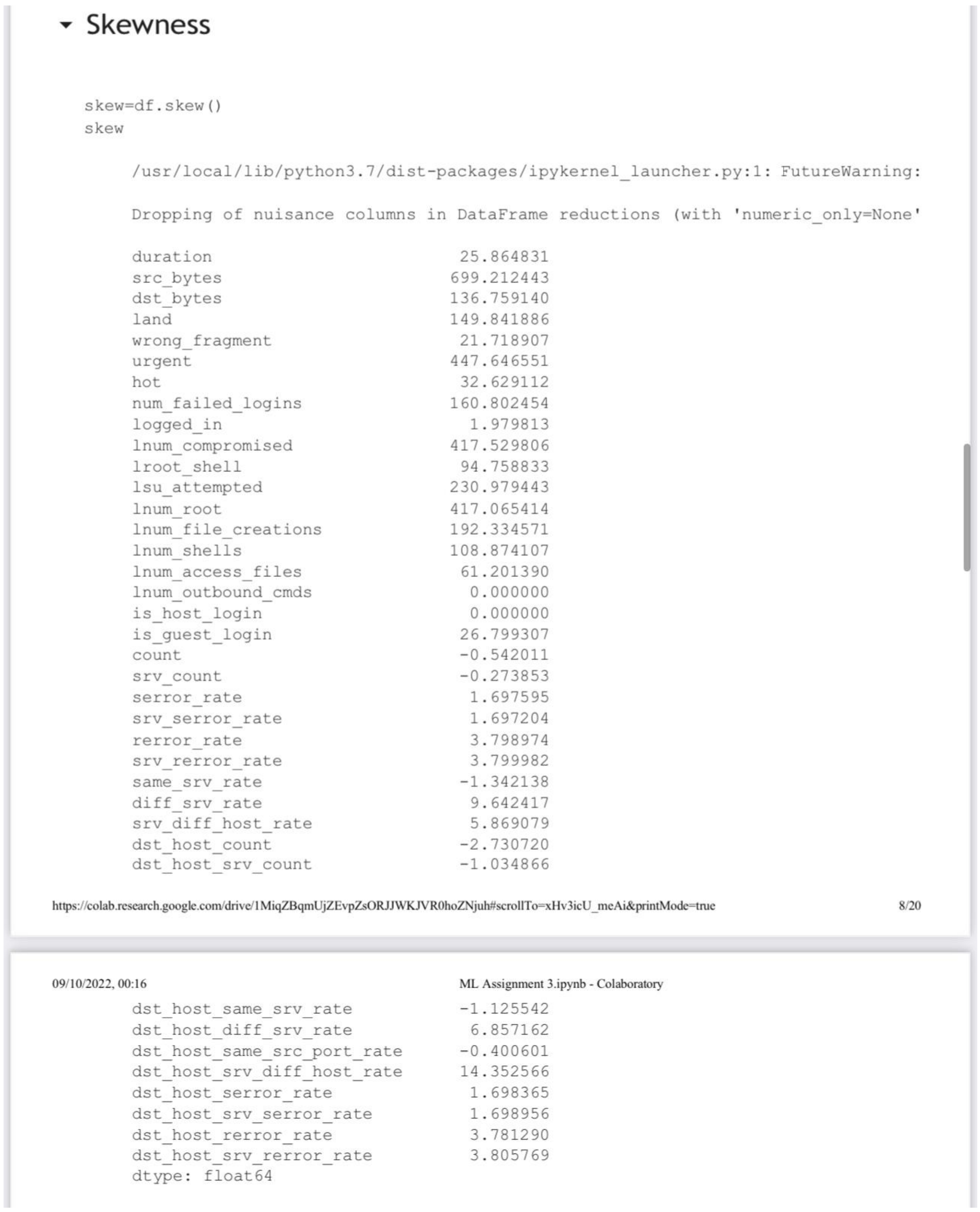


Fig: skewness

If the data are stacked to the left with the tail pointing to the right, this is referred to as a positive skew.

If the data are stacked up to the right, the tail will point to the left and there will be a negative skew. You should take note that positive skews are more common than negative ones.

The Pearson's skewness coefficient, which offers a rapid calculation of a distribution's symmetry, is a useful indicator of a distribution's skewness. You only need to use the "skew()" method in Pandas to calculate skewness.



Skewness

```
skew=df.skew()
skew
```

```
    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:

    Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None'

    duration                        25.864831
    src_bytes                      699.212443
    dst_bytes                      136.759140
    land                           149.841886
    wrong_fragment                  21.718907
    urgent                         447.646551
    hot                             32.629112
    num_failed_logins              160.802454
    logged_in                        1.979813
    lnum_compromised               417.529806
    lroot_shell                     94.758833
    lsu_attempted                  230.979443
    lnum_root                      417.065414
    lnum_file_creations            192.334571
    lnum_shells                    108.874107
    lnum_access_files               61.201390
    lnum_outbound_cmds               0.000000
    is_host_login                    0.000000
    is_guest_login                  26.799307
    count                           -0.542011
    srv_count                       -0.273853
    serror_rate                      1.697595
    srv_serror_rate                  1.697204
    rerror_rate                      3.798974
    srv_rerror_rate                  3.799982
    same_srv_rate                   -1.342138
    diff_srv_rate                    9.642417
    srv_diff_host_rate               5.869079
    dst_host_count                  -2.730720
    dst_host_srv_count              -1.034866
```

https://colab.research.google.com/drive/1MiqZBqmUjZEvpZsORJJWKJVR0hoZNjuh#scrollTo=xHv3icU_meAi&printMode=true 8/20

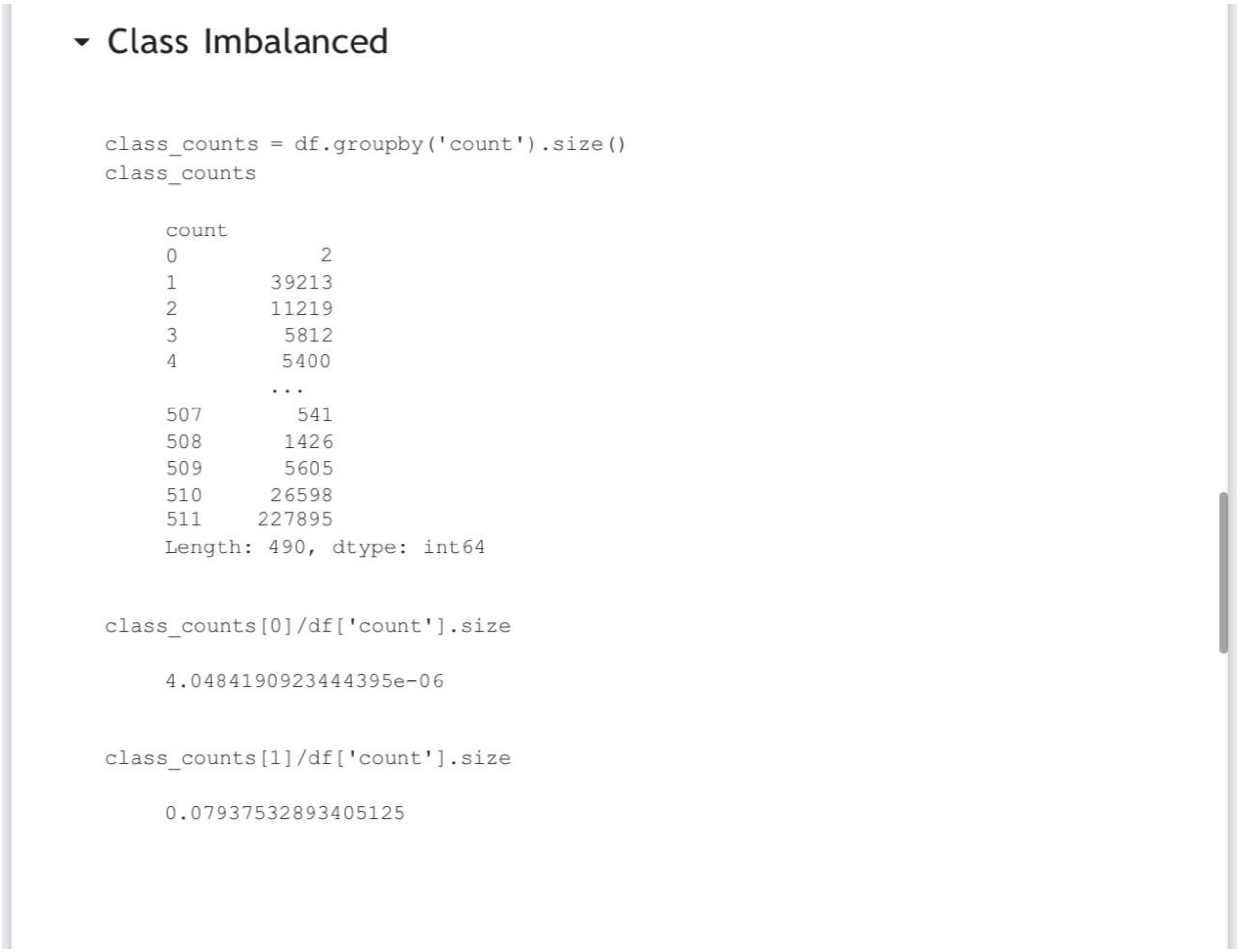
09/10/2022, 00:16 ML Assignment 3.ipynb - Colaboratory

```
    dst_host_same_srv_rate          -1.125542
    dst_host_diff_srv_rate           6.857162
    dst_host_same_src_port_rate     -0.400601
    dst_host_srv_diff_host_rate     14.352566
    dst_host_serror_rate             1.698365
    dst_host_srv_serror_rate         1.698956
    dst_host_rerror_rate             3.781290
    dst_host_srv_rerror_rate         3.805769
    dtype: float64
```

Fig: skewness

Class imbalance :

An imbalanced classification data set has uneven class proportions. Majority classes are those that represent a sizable share of the data set. Minority classes are those that make up a smaller percentage. Most learning algorithms presumptively distribute data equally. Due to this bias towards the dominant class, the machine learning classifier performs poorly when there is a class imbalance, misclassifying the minority class.



Class Imbalanced

```
class_counts = df.groupby('count').size()
class_counts

     count
     0           2
     1       39213
     2       11219
     3        5812
     4        5400
             ...
     507       541
     508      1426
     509      5605
     510     26598
     511    227895
     Length: 490, dtype: int64


class_counts[0]/df['count'].size

     4.0484190923444395e-06


class_counts[1]/df['count'].size

     0.07937532893405125
```

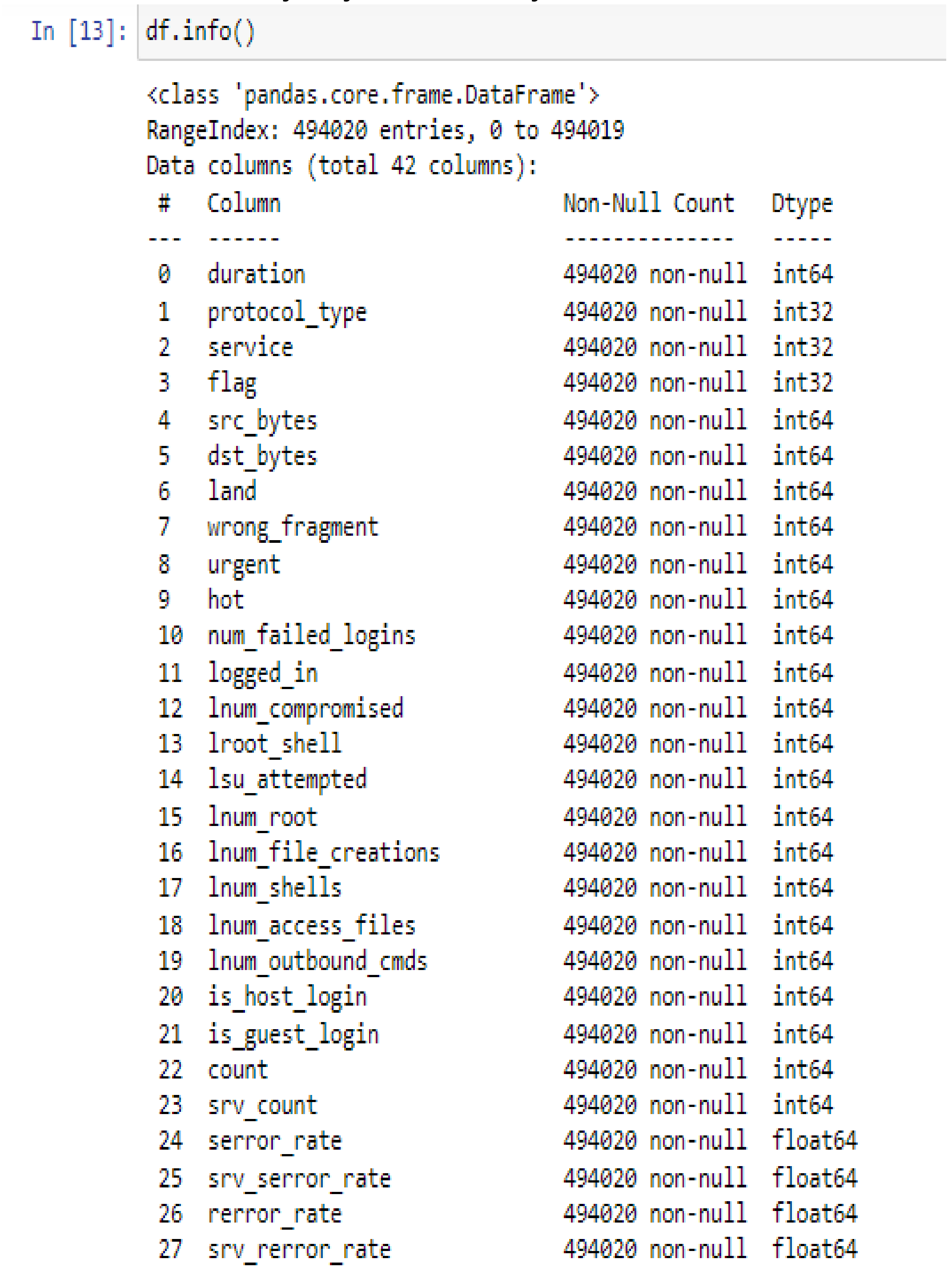
Fig: Class imbalance

Correlation:

One or more traits are related to one another according to correlation. These attributes may be ones that were used as input data and forecasted our goal attribute. The statistical method of correlation shows how one attribute moves or changes in connection to another one. It provides us with a general understanding of how closely the two traits are related. For numerical and categorical features, respectively, the chi-square test and Pearson correlation coefficients are frequently utilized.

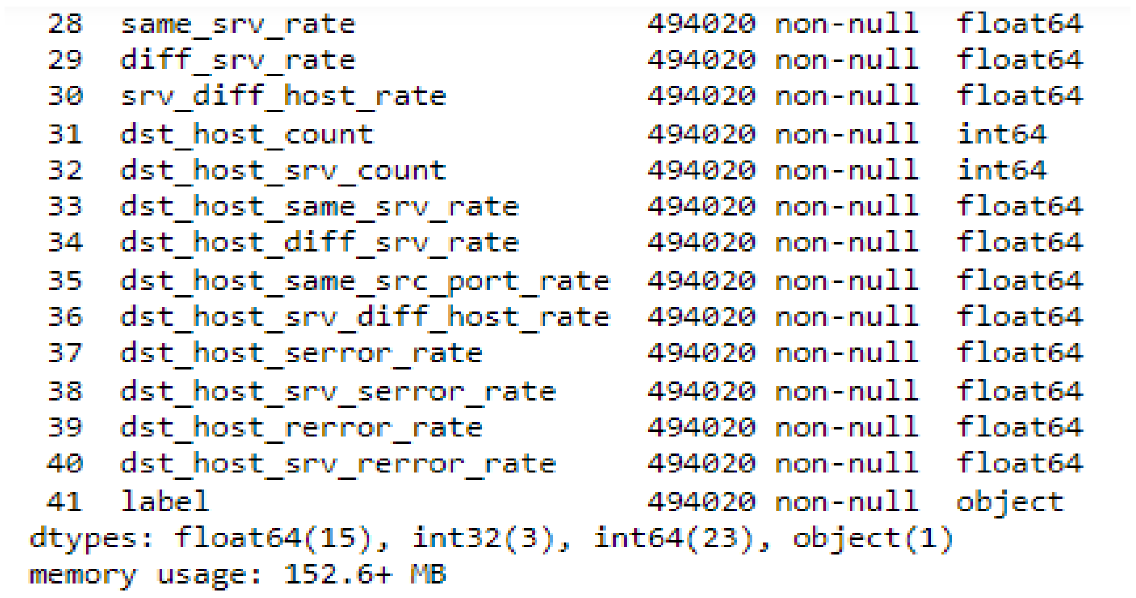
Missing values:

Missing values or data are a regular occurrence in datasets. Finding informational gaps increases the accuracy of your data analysis as a whole.



```
In [13]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 494020 entries, 0 to 494019
Data columns (total 42 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   duration               494020 non-null  int64
 1   protocol_type          494020 non-null  int32
 2   service                494020 non-null  int32
 3   flag                   494020 non-null  int32
 4   src_bytes              494020 non-null  int64
 5   dst_bytes              494020 non-null  int64
 6   land                   494020 non-null  int64
 7   wrong_fragment         494020 non-null  int64
 8   urgent                 494020 non-null  int64
 9   hot                    494020 non-null  int64
 10  num_failed_logins      494020 non-null  int64
 11  logged_in              494020 non-null  int64
 12  lnum_compromised       494020 non-null  int64
 13  lroot_shell            494020 non-null  int64
 14  lsu_attempted          494020 non-null  int64
 15  lnum_root              494020 non-null  int64
 16  lnum_file_creations    494020 non-null  int64
 17  lnum_shells            494020 non-null  int64
 18  lnum_access_files      494020 non-null  int64
 19  lnum_outbound_cmds     494020 non-null  int64
 20  is_host_login          494020 non-null  int64
 21  is_guest_login         494020 non-null  int64
 22  count                  494020 non-null  int64
 23  srv_count              494020 non-null  int64
 24  serror_rate            494020 non-null  float64
 25  srv_serror_rate        494020 non-null  float64
 26  rerror_rate            494020 non-null  float64
 27  srv_rerror_rate        494020 non-null  float64
```



```
 28  same_srv_rate                494020 non-null  float64
 29  diff_srv_rate                494020 non-null  float64
 30  srv_diff_host_rate           494020 non-null  float64
 31  dst_host_count               494020 non-null  int64
 32  dst_host_srv_count           494020 non-null  int64
 33  dst_host_same_srv_rate       494020 non-null  float64
 34  dst_host_diff_srv_rate       494020 non-null  float64
 35  dst_host_same_src_port_rate  494020 non-null  float64
 36  dst_host_srv_diff_host_rate  494020 non-null  float64
 37  dst_host_serror_rate         494020 non-null  float64
 38  dst_host_srv_serror_rate     494020 non-null  float64
 39  dst_host_rerror_rate         494020 non-null  float64
 40  dst_host_srv_rerror_rate     494020 non-null  float64
 41  label                        494020 non-null  object
dtypes: float64(15), int32(3), int64(23), object(1)
memory usage: 152.6+ MB
```

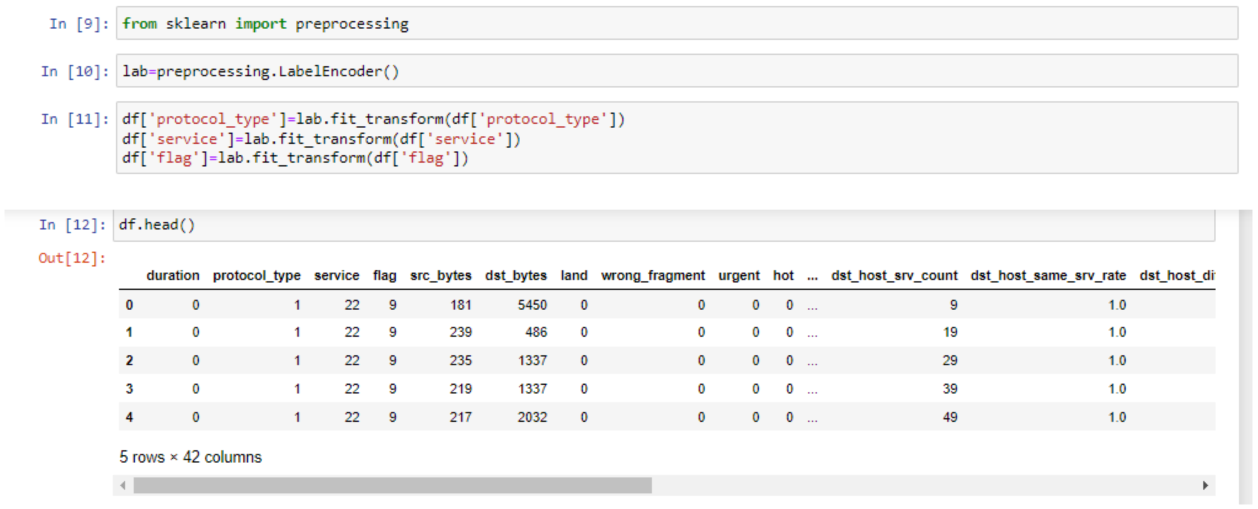
DATA PREPROCESSING: -

The data cleaning task is performed on the dataset initially. υ A preprocessing method, called Label Encoder is used to convert the categorical data to numerical data. υ

Standardization: It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1. υ from sklearn import preprocessing

labelEncode=preprocessing.LabelEncoder() standardisation = preprocessing.StandardScaler()

Protocol	**LabelEncoding**	**src_bytes**	**Standardization**
tcp	0	239	0.65098039
udp	1	177	0.43586695



```
In [9]: from sklearn import preprocessing
```

```
In [10]: lab=preprocessing.LabelEncoder()
```

```
In [11]: df['protocol_type']=lab.fit_transform(df['protocol_type'])
         df['service']=lab.fit_transform(df['service'])
         df['flag']=lab.fit_transform(df['flag'])
```

```
In [12]: df.head()
```

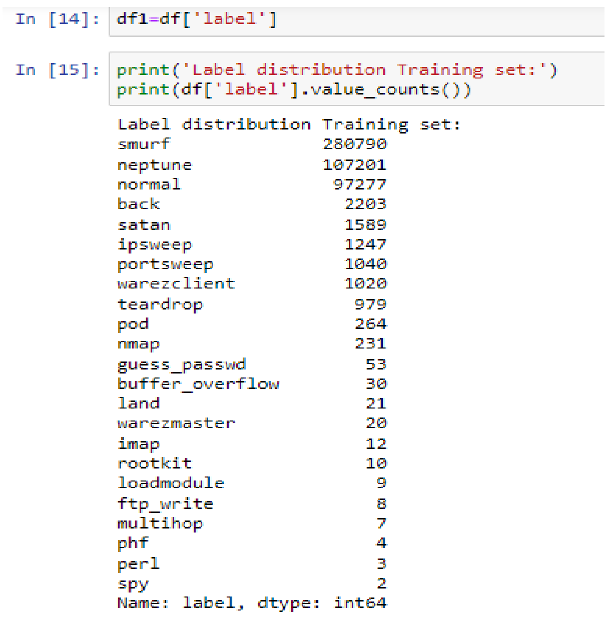
Out[12]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_di
0	0	1	22	9	181	5450	0	0	0	0	...	9	1.0	
1	0	1	22	9	239	486	0	0	0	0	...	19	1.0	
2	0	1	22	9	235	1337	0	0	0	0	...	29	1.0	
3	0	1	22	9	219	1337	0	0	0	0	...	39	1.0	
4	0	1	22	9	217	2032	0	0	0	0	...	49	1.0	

5 rows × 42 columns

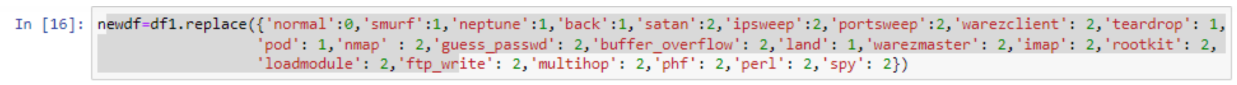
NORMALIZE DATA: -

Data translation into the [0, 1] (or any other range) or simple data transformation onto the unit sphere are both examples of normalization in machine learning. Standardization and normalizing are advantageous for several machine learning methods, especially when Euclidean distance is employed.



```
In [14]: df1=df['label']
```

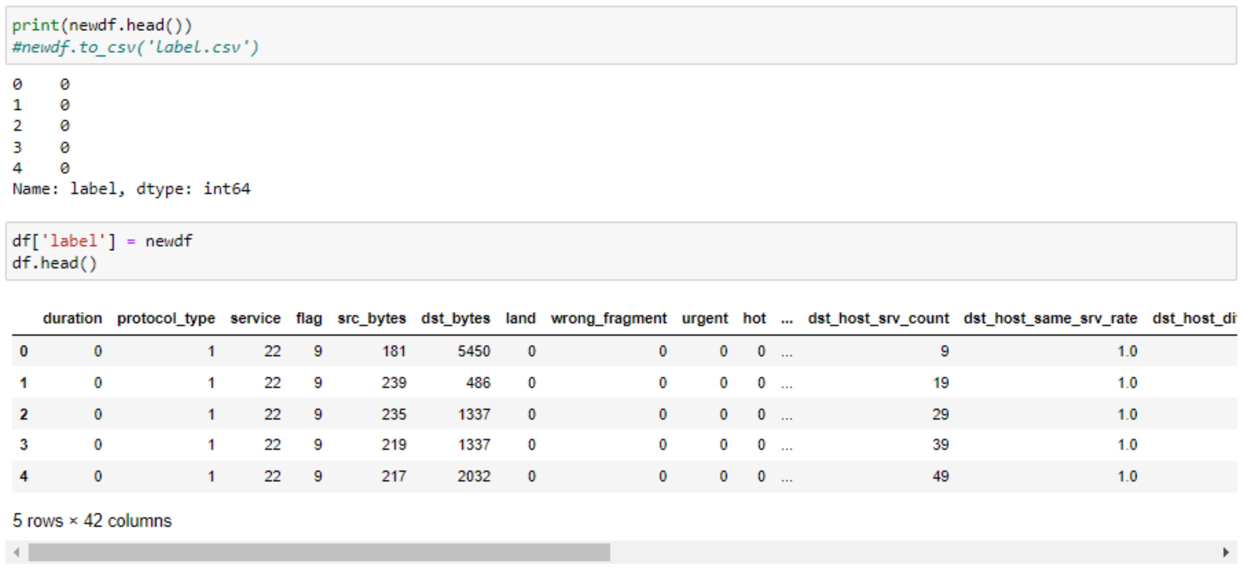
```
In [15]: print('Label distribution Training set:')
         print(df['label'].value_counts())

         Label distribution Training set:
         smurf              280790
         neptune            107201
         normal              97277
         back                 2203
         satan                1589
         ipsweep              1247
         portsweep            1040
         warezclient          1020
         teardrop              979
         pod                   264
         nmap                  231
         guess_passwd           53
         buffer_overflow        30
         land                   21
         warezmaster            20
         imap                   12
         rootkit                10
         loadmodule              9
         ftp_write               8
         multihop                7
         phf                     4
         perl                    3
         spy                     2
         Name: label, dtype: int64
```



```
In [16]: newdf=df1.replace({'normal':0,'smurf':1,'neptune':1,'back':1,'satan':2,'ipsweep':2,'portsweep':2,'warezclient': 2,'teardrop': 1,
                        'pod': 1,'nmap' : 2,'guess_passwd': 2,'buffer_overflow': 2,'land': 1,'warezmaster': 2,'imap': 2,'rootkit': 2,
                        'loadmodule': 2,'ftp_write': 2,'multihop': 2,'phf': 2,'perl': 2,'spy': 2})
```

DISCRETIZATION DATA: -

The process of transforming or partitioning continuous attributes, features, variables, or intervals into discretized or nominal attributes, features, variables, or intervals is known as discretization in statistics and machine learning. This can be helpful for developing probability mass functions, or more specifically, when estimating densities.



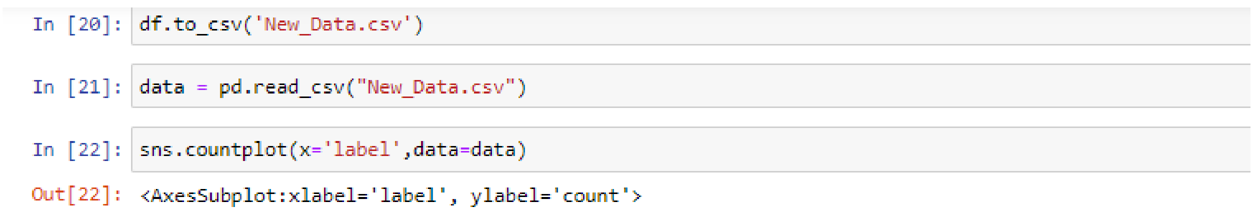
```
print(newdf.head())
#newdf.to_csv('label.csv')
```

```
0    0
1    0
2    0
3    0
4    0
Name: label, dtype: int64
```

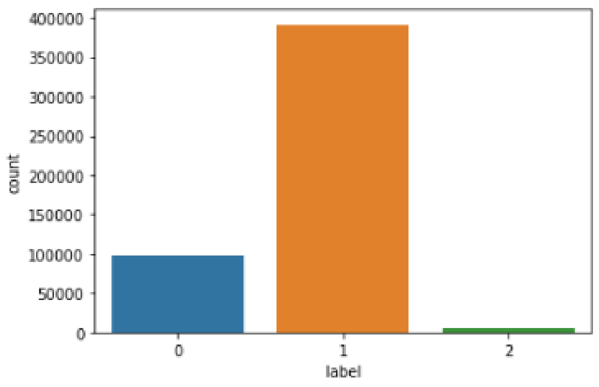
```
df['label'] = newdf
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_di
0	0	1	22	9	181	5450	0	0	0	0	...	9	1.0	
1	0	1	22	9	239	486	0	0	0	0	...	19	1.0	
2	0	1	22	9	235	1337	0	0	0	0	...	29	1.0	
3	0	1	22	9	219	1337	0	0	0	0	...	39	1.0	
4	0	1	22	9	217	2032	0	0	0	0	...	49	1.0	

5 rows × 42 columns



```
In [20]: df.to_csv('New_Data.csv')

In [21]: data = pd.read_csv("New_Data.csv")

In [22]: sns.countplot(x='label',data=data)

Out[22]: <AxesSubplot:xlabel='label', ylabel='count'>
```



FEATURE SELECTION using Chi-Sq: -

The best features for a particular dataset may be found using the Chi-SQ test by identifying the characteristics that the output class label is most dependent on. The chi-square is computed for each feature in the dataset, the higher the value of chi-sq, the more dependent the output label is on the feature.

Protocol	Service	Class
tcp	http	normal
tcp	http	normal
tcp	smtp	smurf
udp	domain_u	smurf
udp	domain_u	normal

The Formula for Chi Square Is

$$\chi^2_c = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

$c =$ degrees of freedom

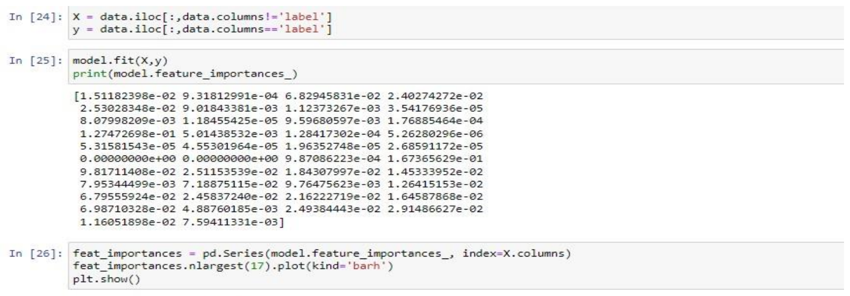
$O =$ observed value(s)

$E =$ expected value(s)

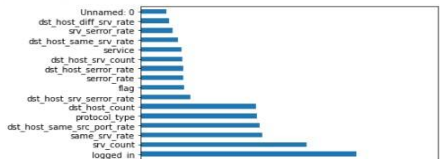
Contingency table	normal	smurf
tcp	2	1
udp	1	1

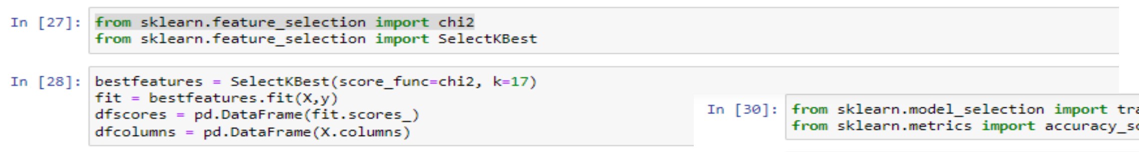
The expected value for the cell (tcp,normal) is calculated as $\frac{3}{5} \times 3 = 1.8$

$$\chi^2_{(\text{Protocol})} = \frac{(2-1.8)^2}{1.8} + \frac{(1-1.2)^2}{1.2} + \frac{(1-1.2)^2}{1.2} + \frac{(1-0.8)^2}{0.8} = 0.13$$

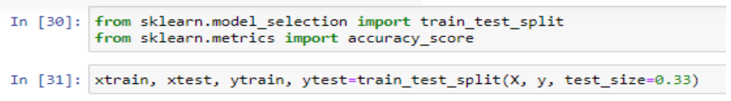


```
In [24]: X = data.iloc[:,data.columns!='label']
         y = data.iloc[:,data.columns=='label']

In [25]: model.fit(X,y)
         print(model.feature_importances_)

         [1.51182398e-02 9.31812991e-04 6.82945831e-02 2.40274272e-02
          2.53028348e-02 9.01843381e-03 1.12373267e-03 3.54176936e-05
          8.07998209e-03 1.18455425e-05 9.59680597e-03 1.76885464e-04
          1.27472698e-01 5.01438532e-03 1.28417302e-04 5.26280296e-06
          5.31581543e-05 4.55301964e-05 1.96352748e-05 2.68591172e-05
          0.00000000e+00 0.00000000e+00 9.87086223e-04 1.67365629e-01
          9.81711408e-02 2.51153539e-02 1.84307997e-02 1.45333952e-02
          7.95344499e-03 7.18875115e-02 9.76475623e-03 1.26415153e-02
          6.79555924e-02 2.45837240e-02 2.16222719e-02 1.64587868e-02
          6.98710328e-02 4.88760185e-03 2.49384443e-02 2.91486627e-02
          1.16051898e-02 7.59411331e-03]

In [26]: feat_importances = pd.Series(model.feature_importances_, index=X.columns)
         feat_importances.nlargest(17).plot(kind='barh')
         plt.show()
```



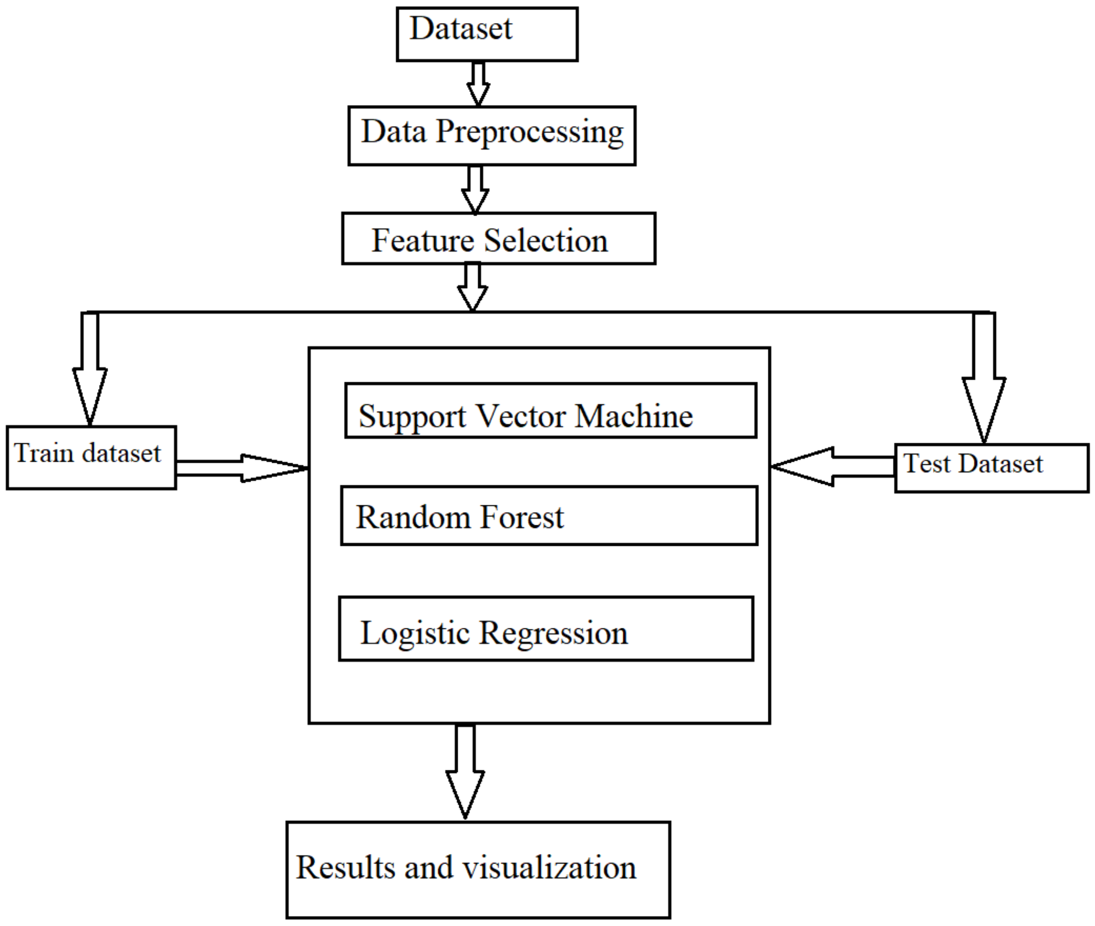


```
In [27]: from sklearn.feature_selection import chi2
         from sklearn.feature_selection import SelectKBest

In [28]: bestfeatures = SelectKBest(score_func=chi2, k=17)
         fit = bestfeatures.fit(X,y)
         dfscores = pd.DataFrame(fit.scores_)
         dfcolumns = pd.DataFrame(X.columns)
```



```
In [30]: from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score

In [31]: xtrain, xtest, ytrain, ytest=train_test_split(X, y, test_size=0.33)
```

MODELING AND IMPLEMENTATION:



1.Dataset: The KDD99 data set is used to evaluate the proposed model. The number of instances that are used are equal to 494,021. The KDD99 dataset has 41 attributes and the 'class' attributes which indicates whether a given instance is a normal instance or an attack.

2.Data pre-processing: Large-scale datasets usually contain noisy, redundant and different types of data which present critical challenges to knowledge discovery and data modelling. Generally, the intrusion detection algorithms deal with one or more of the raw 26 input data types such as SVM algorithm that deals with numerical data only. Hence, we prepare data and convert categorical data in the dataset to numerical data.

3.Feature Selection:
In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons:
Simplification of models to make them easier to interpret by researchers/users, Shorter training times, to avoid the curse of dimensionality, enhanced generalization by reducing over fitting (formally, reduction of variance).
The central premise when using a feature selection technique is that the data contains some features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information.

4.Train and Test Datasets:
After feature selection, 80% of the dataset is used for training purpose and 20% of the dataset is used for evaluation and test the model. So, the given dataset is divided into two subsets. The training subsets are given to the classifiers for classification and training accuracy is measured. The test subsets after training are given to models for evaluating the performance.

5.Classifier Models:
We have 3 classifiers used in our proposed model. Support vector machine, Random Forest, and Logistic Regression. But our main Classifier is Support Vector machine. Remaining two will be used to compare the accuracy results with SVM. So that, we can learn about the competency between the 3 classifiers with the given dataset.
6.Results & Visualization:
After Training of model, test set is given to the model without the class label. The test set is the ‘unseen dataset’ to it. It predicts the output which will be the expected output and it will be compared to the actual output which we have. Then the accuracy is measured by evaluating the matched and mismatched outputs.

Accuracy
The degree to that the info is getting ready to verity values. whereas process all attainable valid values permits invalid values to be simply spotted, it doesn't mean that they're accurate. a sound address mightn’t truly exist. a sound person’s eye color, say blue, can be valid, however not true (doesn’t represent the reality). Another issue to notice is that the distinction between accuracy and precision. locution that you just survive the world is, true. But, not precise. wherever on the earth? locution that you live at a specific street address is additional precise.

Completeness
The degree to which all required facts is known. Missing facts goes to appear for diverse reasons. One can mitigate this trouble with the aid of using thinking the unique source, if possible, say re-interviewing the issue. Chances are that the issue is both going to provide a special solution or can be difficult to attain again.

Consistency
The degree to which the information is consistent, withinside the identical information set or throughout a couple of information sets. Inconsistency happens while values withinside the information set contradict every other. A legitimate age, say 10, mightn't suit with the marital status, say divorced. A purchaser is recorded in distinct tables with distinct addresses.

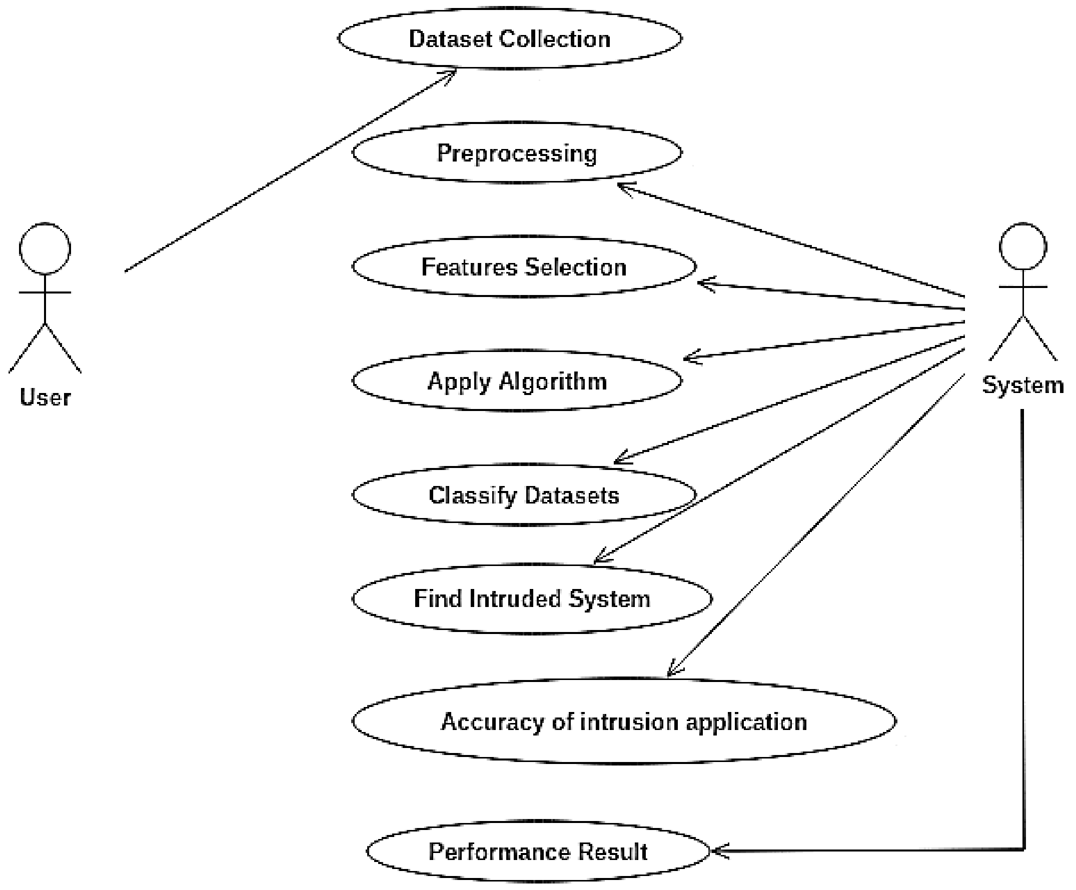
Uniformity

The Degree to which the information is certain the use of the identical unit of degree. The weight can be recorded both in kilos and kilos. The date would possibly comply with the us layout or European layout. The forex is occasionally in USD and occasionally in YEN. And so, information should be transformed to a Single degree unit.

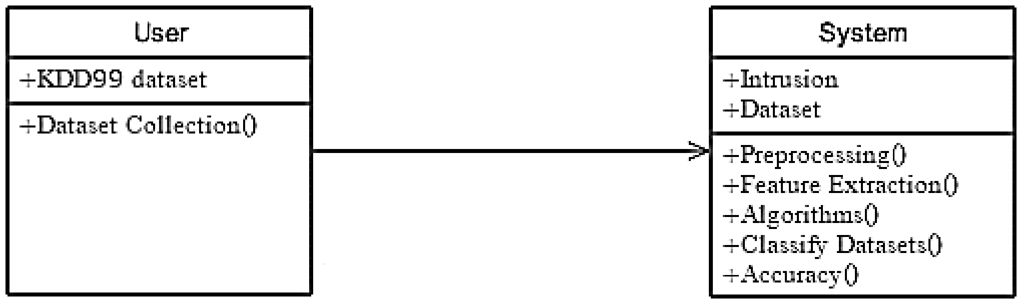
Standardization
Standardization comes below Feature Scaling. Feature Scaling is a way to standardize the unbiased functions gift withinside the records in a set range. It is completed at some point of the records pre-processing to deal with exceedingly various magnitudes or values or units. If function scaling isn't done, then a system gaining knowledge of set of rules tends to weigh more values, better and take into account smaller values because the decrease values, irrespective of the unit of the values.

Example:
If an algorithm is not using feature scaling method, then it can consider the value 3000 meter to be greater than 5 km but that's actually not true and, in this case, the algorithm will give wrong predictions. So, we use Feature Scaling to bring all values to same magnitudes and thus, tackle this issue.

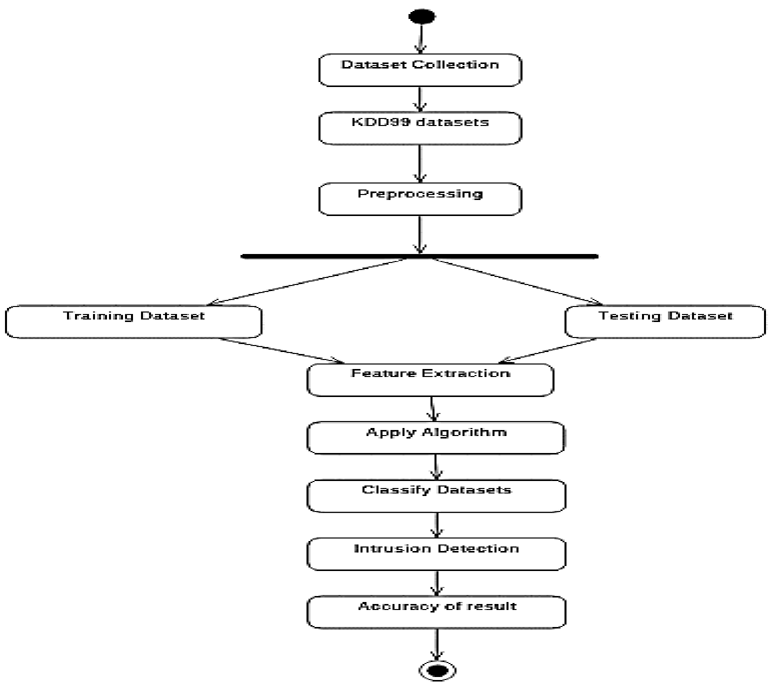
Use Case diagram:



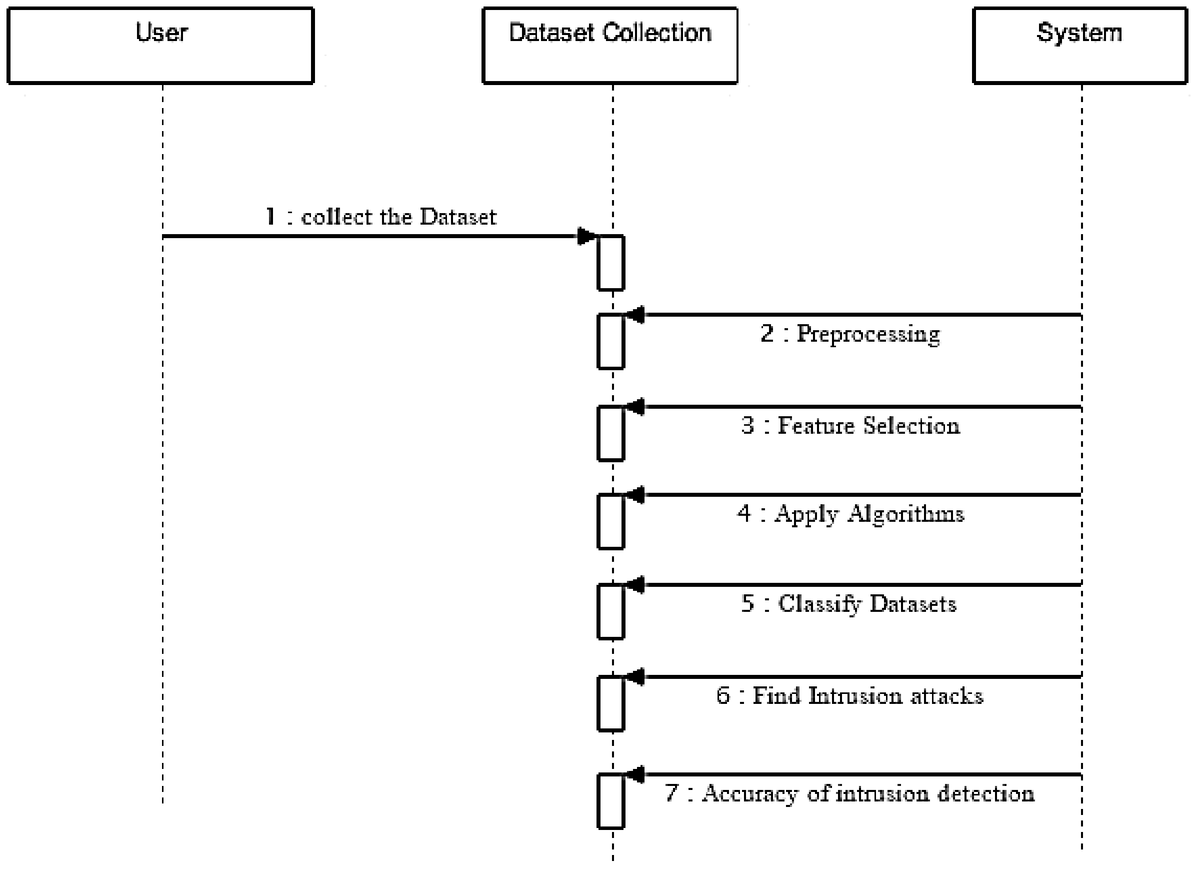
Class Diagram:



Activity Diagram:



Sequence Diagram:



IMPLEMENTATION

Source Code

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report
df=pd.read_csv('kddcup99_csv.csv') #1
df.head()
df.columns
df.info()
df.shape
df.isna().sum()
print('Data set:')
for col_name in df.columns:
if df[col_name].dtypes == 'object' :
unique_cat = len(df[col_name].unique())
print("Feature '{col_name}' has {unique_cat} categories".format(col_name=col_name,
unique_cat=unique_cat))
print()
print('Distribution of categories in service:')
print(df['service'].value_counts().sort_values(ascending=False).head())
from sklearn import preprocessing
ab=preprocessing.LabelEncoder()

        df['protocol_type']=lab.fit_transform(df['protocol_type'])

df['service']=lab.fit_transform(df['service'])
df['flag']=lab.fit_transform(df['flag'])
df.head()
df.info()
df1=df['label']
rint('Label distribution Training set:')
print(df['label'].value_counts())
newdf=df1.replace({'normal':0,'smurf':1,'neptune':1,'back':1,'satan':2,'ipsweep':2,'portswe
ep':2,'warezclient': 2,'teardrop': 1,'pod': 1,'nmap' : 2,'guess_passwd': 2,'buffer_overflow':
2,'land': 1,'warezmaster': 2,'imap': 2,'rootkit': 2, 'loadmodule': 2,'ftp_write': 2,'multihop':
2,'phf': 2,'perl': 2,'spy': 2})
print(newdf.head())
#newdf.to_csv('label.csv')
df['label'] = newdf
```

```
df.head()
df.info()
#df.to_csv('New_Data.csv')
data = pd.read_csv("New_Data.csv")
sns.countplot(x='label',data=data)
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
X = data.iloc[:,data.columns!='label']
y = data.iloc[:,data.columns=='label']
model.fit(X,y)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest

        bestfeatures = SelectKBest(score_func=chi2, k=10)\

fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
featureScores.nlargest(10,'Score')
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
xtrain, xtest, ytrain, ytest=train_test_split(X, y, test_size=0.33)
from sklearn import svm
sv=svm.LinearSVC()
sv.fit(xtrain,ytrain)
predic1=sv.predict(xtest)
acc2=accuracy_score(predic1,ytest)
acc2
clf1=classification_report(predic1,ytest)
print(clf1)
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(xtrain,ytrain)
predic=rf.predict(xtest)
acc1=accuracy_score(predic,ytest)
acc1
clf=classification_report(predic,ytest)
print(clf)
from sklearn.linear_model import LogisticRegression
lg=LogisticRegression()
```

```
lg.fit(xtrain,ytrain)
predic2=lg.predict(xtest)
acc3=accuracy_score(predic2,ytest)
acc3 clf2=classification_report(predic2,ytest)
print(clf2)
import matplotlib.pyplot as plt;plt.rcdefaults()
objects = ('Support Vector','Random forest','LogisticRegression')
y_pos = np.arange(len(objects))
performance = [acc1,acc2,acc3]
plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy')
plt.title('SVM vs Decision Tree')

        plt.show()
```

Steps:

The steps that are been included for finding the approximate value, intrusion detection from the dataset and comparing the dataset values using the algorithms such as Support vector, Random Forest and Logistic Regression.

#1. Input the Data Set
Here first we are reading the data sets given(kddcup99_csv.csv) and analysing the values of the values in the data set and checking the all-column values from the dataset, where all the table is read and the shape is defined with the **(494020, 42)** values.

#2. Distribution of categories in service

- Feature 'protocol type' has 3 categories
- Feature 'service' has 66 categories
- Feature 'flag' has 11 categories
- Feature 'label' has 23 categories

We have features where we consider these features and do the distribution of categories in service.

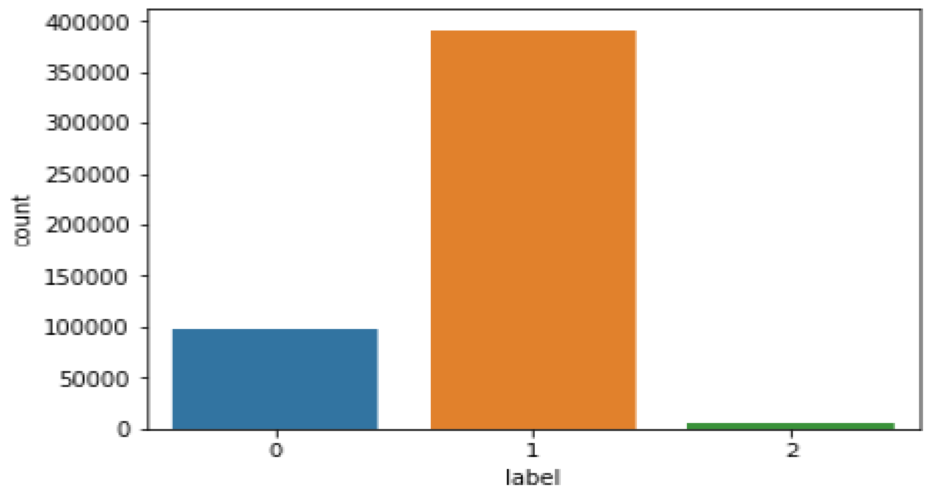
SERVICE NAME	DATA TYPE(int64)
ecr_i	281400
Private	110893
http	64292
Smtp	9723
Others	7237

Label distribution training set

When we want to import the pre-processor technique then we should analyse the protocol type, service, and the flag values. All the information will be gathered, and the data will be trained and distributed.

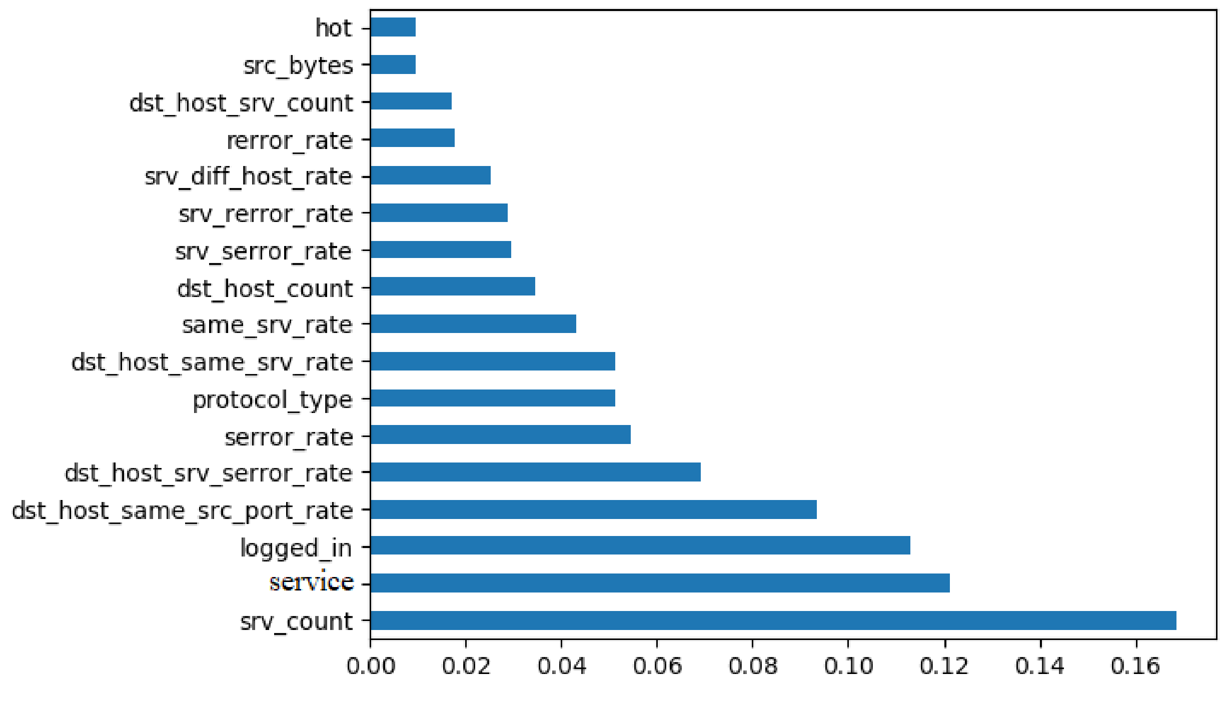
LABEL NAME	DATATYPE
SSmurf	280790
Neptune	107201
Normal	97277
Back	2203
Satan	1589
Ipsweep	1247
Portsweep	1040
Warezclient	1020
Teardrop	979

Now comparing the data with the dataset(new_data.csv) with the label values.

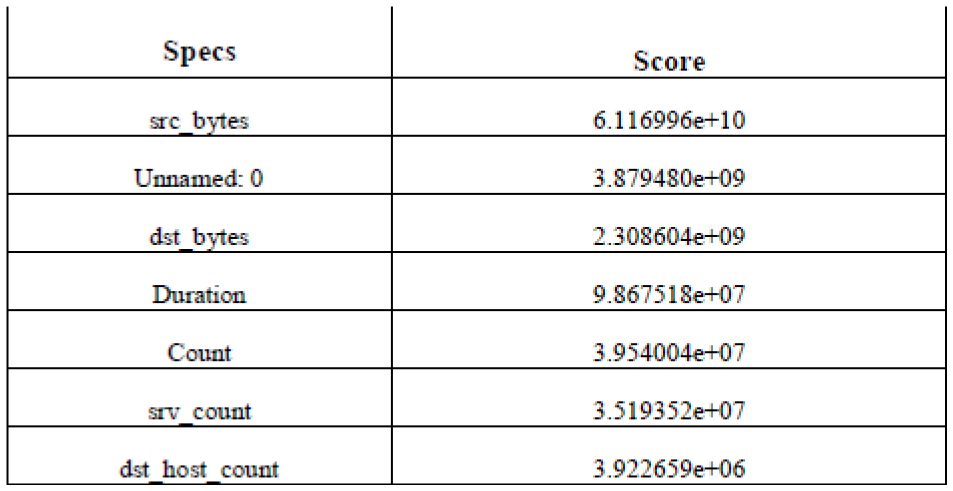


Feature Selection

For feature selection we should import ExtraTreesClassifier and we will solve the model feature importance. We have 42 feature selections attributes where we only consider the best 17 attributes whose chi-square value is the highest, which are classified and used and are shown in below figure.



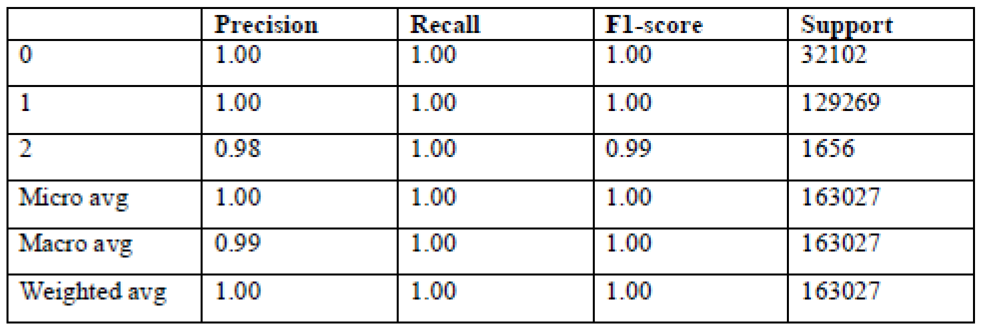
And we find the data score using the feature selection method with the scores using the specification.



Specs	Score
src_bytes	6.116996e+10
Unnamed: 0	3.879480e+09
dst_bytes	2.308604e+09
Duration	9.867518e+07
Count	3.954004e+07
srv_count	3.519352e+07
dst_host_count	3.922659e+06

Support Vector Machine (SVM)

In the support vector machine (SVM)we train and test the data using the accurate score with **test size=0.33**
We predict the test cases using the accurate score and get the value of accuracy as **acc1= 0.9997669097756813.**
The classification is done, and the classification report is given as

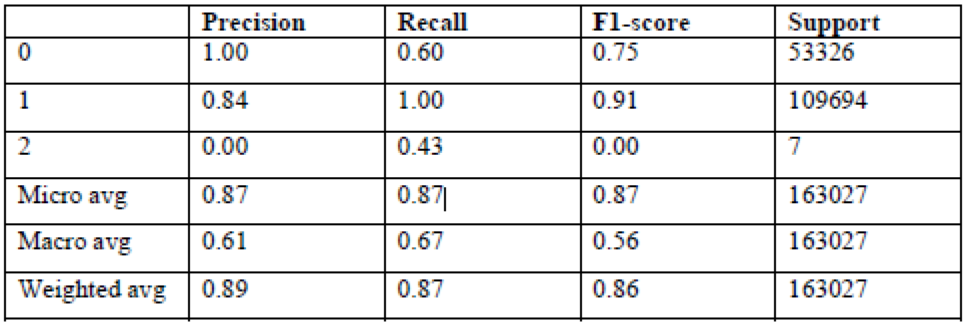


	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	32102
1	1.00	1.00	1.00	129269
2	0.98	1.00	0.99	1656
Micro avg	1.00	1.00	1.00	163027
Macro avg	0.99	1.00	1.00	163027
Weighted avg	1.00	1.00	1.00	163027

Random Forest Classifier

In the Random Forest classifier, we train and test the data using the accurate score with **test size=0.33.**
We predict the test cases using the accurate score and get the value of accuracy as **Acc2=0.8658688438111478.**

The classification is done, and the classification report is given as



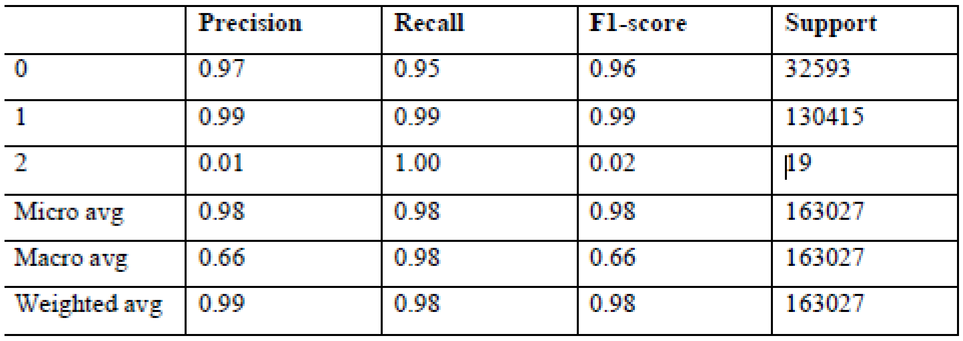
	Precision	Recall	F1-score	Support
0	1.00	0.60	0.75	53326
1	0.84	1.00	0.91	109694
2	0.00	0.43	0.00	7
Micro avg	0.87	0.87	0.87	163027
Macro avg	0.61	0.67	0.56	163027
Weighted avg	0.89	0.87	0.86	163027

Logistic Regression

Using the Logistic regression algorithm, we train and test the data using the accurate score with **test size=0.33.**

We predict the test cases using the accurate score and get the value of accuracy as **Acc3=0.978316475185092.**

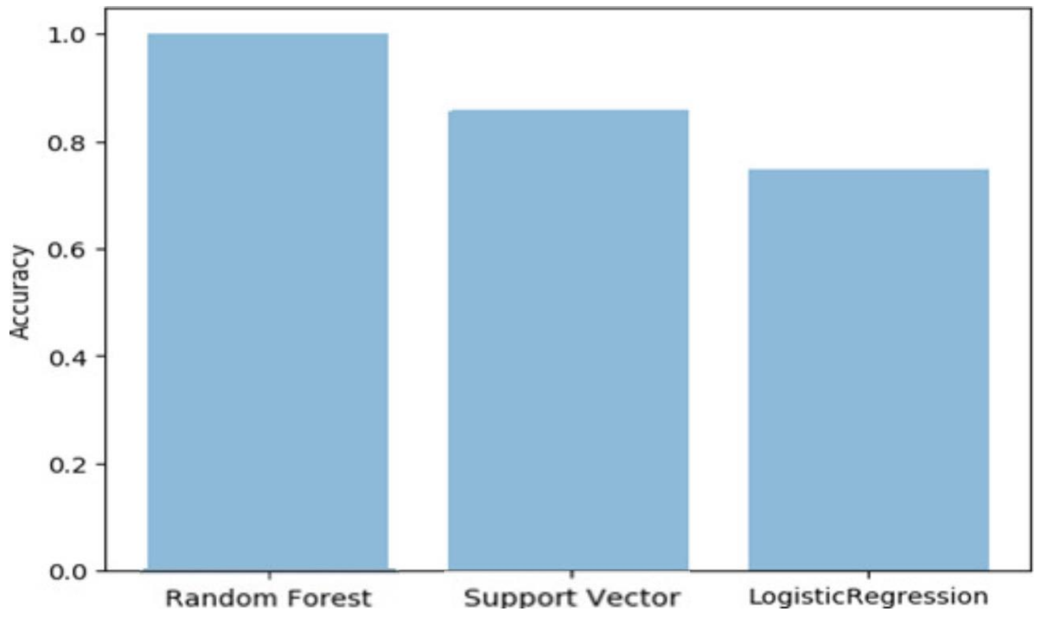
The classification is done, and the classification report is given as



	Precision	Recall	F1-score	Support
0	0.97	0.95	0.96	32593
1	0.99	0.99	0.99	130415
2	0.01	1.00	0.02	19
Micro avg	0.98	0.98	0.98	163027
Macro avg	0.66	0.98	0.66	163027
Weighted avg	0.99	0.98	0.98	163027

Comparison between the Algorithms

The values are compared with the accuracy found by which we are analysing the best method to solve and find the intrusion.



ANALYSIS OF RESULTS:

After training, a test set without a class label is provided to the model. Its "unseen dataset" is the test set. It forecasts the outcome, which will be the anticipated outcome, and it will be contrasted with the outcome that we really have. The accuracy is then assessed by comparing the outputs that were matched and those that weren't. The ultimate outcomes are presented as a graph. The graph compares the accuracy of the three classifiers. To compare it to other systems, we can see which classifier attained the highest level of classification accuracy and its prediction time. The experiment's findings demonstrated that the model has high. The results of the experiment showed that Spark-Chi model has high performance, reduces the training time and is efficient for Big Data.

SUMMARY:

The team developed the Big Data-capable Spark-Chi Random Forest model for intrusion detection system. The proposed model made advantage of the Spark Big Data platform, which has a fast data processing and analysis speed. Big data contain great degree of dimensions, which increase the complexity and length of the categorization process. To classify the data into normal and attack in the proposed model, the researchers took the Chi-Square Selector method to choose relevant features with SGD. The experiment's findings demonstrated the model's superior performance and quickness. Future research can expand the model to a multi-class model that can recognize different attack kinds.

REFERENCES

1 Author: r.a. Kemmerer; g. Vigna

Title: intrusion detection: a brief history and overview

Publishing source data: ieee

Date of publication: computer (volume: 35, issue: 4, April 2002)

Url: https://ieeexplore.ieee.org/document/1012428

Text: This paper studies an improved k-dependency Bayesian network (kdbn) structural model that can accurately describe the dependence relationships among system variables and reduce the complexity of the Bayesian network structure by reducing the directed edges of weak dependence.

2 Author: dewan md. Farid, mohammad zahidur rahman

Title: anomaly network intrusion detection based on improved self-adaptive Bayesian algorithm

Publishing source data: journal of computers

Date of publication: vol. 5, no. 1, January 2010

Url:https://citeseerx.ist.psu.edu/document?Repid=rep1&type=pdf&doi=240bf5d70f2439075a9a80150df64f49eb1c715d

Text: By enabling a direct comparison of different security solutions with respect to their relative effectiveness, a network security metric may provide quantifiable evidence to assist security practitioners in securing computer networks. However, research on security metrics has been hindered by difficulties in handling zero-day attacks exploiting unknown vulnerabilities. In fact, the security risk of unknown vulnerabilities has been considered as something unmeasurable due to the 8 less

predictable nature of software flaws. This causes a major difficulty to security metrics, because a more secure configuration would be of little value if it were equally susceptible to zero-day attacks.

3 Author: qinglei zhang, gongzhu hu & wenying feng

Title: design and performance evaluation of a machine learning-based method for intrusion detection

Publishing source data: ieee

Date of publication: part of the studies in computational intelligence book series (sci,volume 295)

Url: https://link.springer.com/chapter/10.1007/978-3-642-13265-0_6

Text: this research uses artificial intelligence methods for computer network intrusion detection system modeling. Primary classification is done using self-organized maps (som) in two levels, while the secondary classification of ambiguous data is done using sugeno type fuzzy inference system (fis). Fis is created by using adaptive neuro-fuzzy inference system (anfis). The main challenge for this system was to successfully detect attacks that are either unknown or that are represented by very small percentage of samples in training dataset.

4 Author: ashara banu mohamed; norbik bashah idris; bharanidharan shanmugum

Title: alert correlation using a novel clustering approach

Publishing source data: ieee

Date of publication: 17 may 2012

Url: https://ieeexplore.ieee.org/abstract/document/6200725

Text: This paper is regarding present a novel graphics processing unit (gpu)-accelerated implementation of a monte carlo markov chain-based algorithm for learning bns that is up to 7.5-fold faster than current general-purpose processor (gpp)-based implementations. The gpu-based implementation is just one of several implementations within the larger application, each optimized for a different input or machine configuration

5 Author: hung-jenliaochun-hungrichard linying-chihlin kuang-yuantung

Title: intrusion detection system: a comprehensive review

Publishing source data: journal of network and computer applications

Date of publication: volume 36, issue 1, January 2013, pages 16-24

Url:https://www.sciencedirect.com/science/article/pii/s108480451200194

Text: hybrid model with test data achieved acceptable dr value 0.8883 and far value 0.2415. The objectives were successfully achieved as it is presented (compared with the similar researches on nsl kdd dataset). Proposed model can be used not only in further research related to this domain, but also in other research areas.

6 Author: Ben Lutkevich

Title: Intrusion detection system (ids)

Publishing source data: tech target

Date of publication: 6 August 2015

Url:https://www.techtarget.com/searchsecurity/definition/intrusion-detection-system

Text: Security threats and economic loss caused by network attacks, intrusions and vulnerabilities have motivated intensive studies on network security. Normally, data collected in a network system can reflect or can be used to detect security threats. We define these data as network security-related data. Studying and analyzing security-related data can help detect network attacks and intrusions, thus making it possible to further measure the security level of the whole network system. Obviously, the first step in detecting network attacks and intrusions is to collect security-related data.