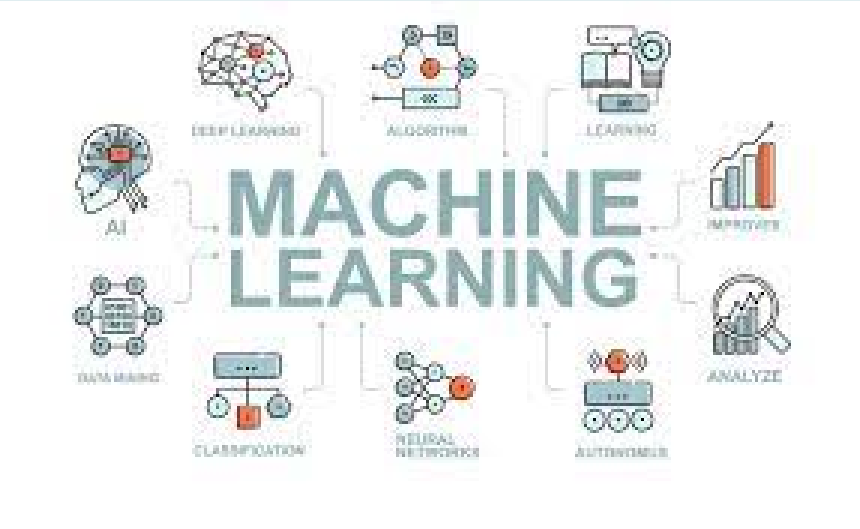
MACHINE LEARNING PROJECT

INTRUSION DETECTION MODEL USING MACHINE LEARNING ALGORITHM



ABSTRACT

- Information security for Big data environments has recently evolved due to the enormous volumes of data and their gradual evolution. Inorder to handle Big Data accurately and effectively, Intrusion Detection Systems employ machine learning algorithms.

- Inthis project, evaluations of the support vector machine, logistic regression, and Spark-Chi-Random Forest models for intrusion detection were established.

BUSINESS OBJECTIVES

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.

EXISTING SYSTEM

- Cluster machine learning technique, the k-Means method in the machine learning libraries for Intrusion detection.
- Optimization algorithm for feature selection, Hadoop-based parallel Binary Bat algorithm method for intrusion detection.

DISADVANTAGES :

- There is no feature selection strategy to choose the associated features, which makes dealing with Big Data security more complicated and inefficient.
- It requires longer calculation period.

PROPOSED METHOD

- Comparison among Random Forest classifier, SVM classifier and Logistic Regression.

ADVANTAGES :

- The model is trained and tested using the KDD99.
- Compared to Hadoop, Apache Spark's big data capabilities are 100 times quicker.
- Good speed, high performance, and a low risk of false positive alarms.
- Using Random Forest on KDD Datasets helps speed up computation.

PROCEDURE

- Data Preprocessing
- Feature Selection
- Train with classifiers
- Evaluation and Test
- Visualization of Results

SWOT ANALYSIS

STRENGTHS • This model produces high performance and speed capabilities • Easy to use once it is deployed	**WEAKNESS** • Model making requires highly skilled staff • Collecting huge data sets and preprocessing can be tedious.
OPPORTUNITIES • This model can be extended to multi-classes model that could detect types of attacks. • High end security can be achieved.	**THREATS** • Noises can severely limit the efficiency.

List the dataset acquired

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.

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	

Type of Data Set:

- 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.

Feature Selection

- Feature selection is crucial for designing the intrusion detection models. In this process, only the most relevant capabilities are extracted from the whole dataset.
- This prevents the inappropriate capabilities from inflicting noise with inside the categorization of the connections.
- Currently, there exist two main approaches to carry out feature selection: filter and wrapper.
- Wrapper is more complex in terms of computing than the filter approach but gives better results.

Proposed Model Feature Selection

- The function choice set of rules proposed right here is proven. Initially, the facts gain ratio for every of the 41 features of KDD99 are computed, and then ranked in accordance with their values.

- In the sequence, after every iteration, the k-means classifier extracts the function with the maximum IGR from the dataset and assesses the detection price of the top-quality subset of features.

- Additionally, the accuracy level in detecting the right category for the connection in the top quality subset is verified. The selecting manner stops when both the classifier accuracy is above the adjusted threshold, or the accuracy fee is under the preceding calculated fee.

- The IGR metric was used right here specially because of its good results proven in the filter approach, as well as its low computational cost.

DATA PREPROCESSING:

- The data cleaning task is performed on the dataset initially.

- A preprocessing method, called Label Encoder is used to convert the categorical data to numerical data.

- **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.

  from sklearn import preprocessing.

LabelEncoder = preprocessing.LabelEncoder()

standardization = preprocessing.StandardScaler()

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

IMPORTING NECESSARY LIBRARIES:

```
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

NON-NULL VALUES:

```
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
```

NORMALIZE DATA:

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})
```

DISCRETIZE DATA:

```
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	

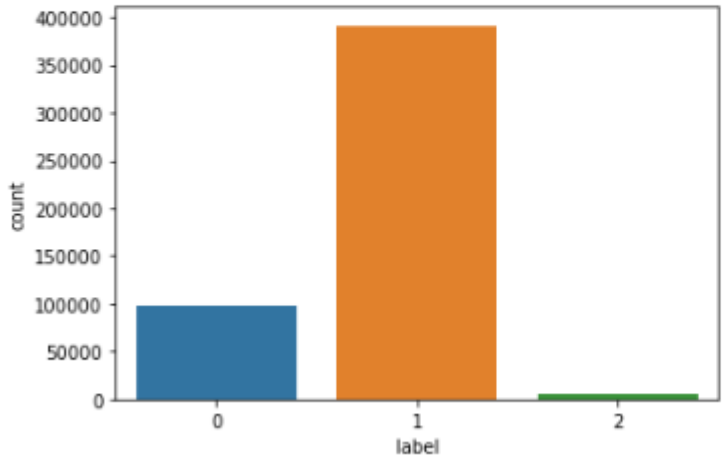
5 rows × 42 columns

```
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

Chi-SQ test can be used to determine the best features for a given dataset by determining the features on which the output class label is most dependent on. For each feature in the dataset, the chi-sq is calculated, 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

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

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 $\frac{3}{5} \times 3 = 1.8$

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

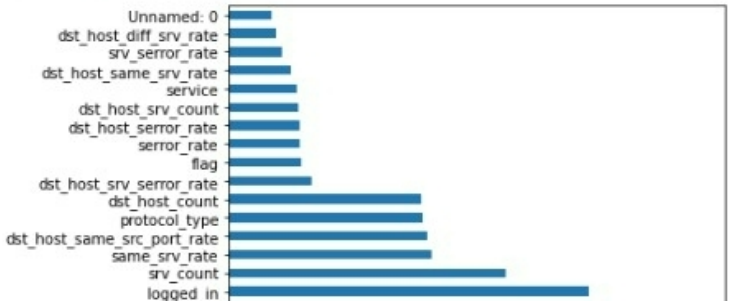
CODE:

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
```

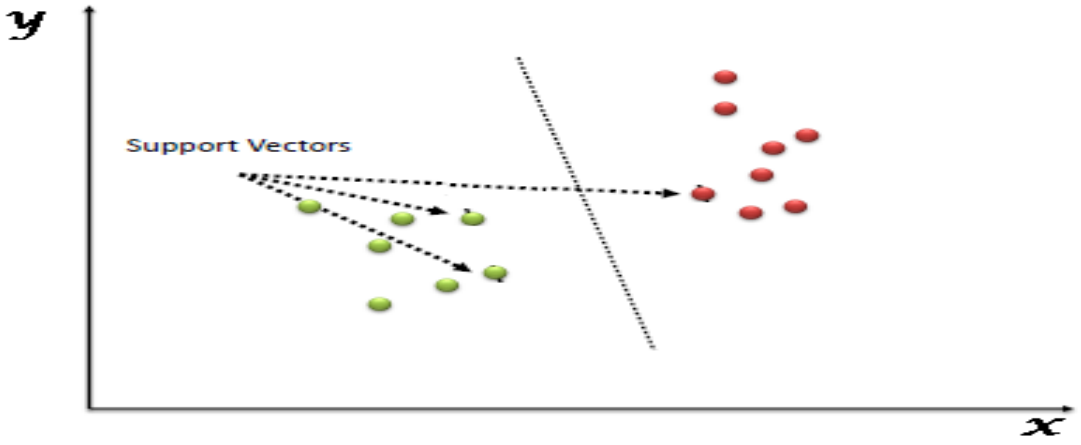
In [31]:
```
xtrain, xtest, ytrain, ytest=train_test_split(X, y, test_size=0.33)
```

Here we are using 3 Modeling techniques namely : -

Support Vector Machine (SVM)
Random Forest Classifier
Logistic Regression Classifier

SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM) is a supervised machine language algorithm which can be used for both classification and regression challenges.
However, it is mostly used in classification problems.
In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.



RANDOM FOREST CLASSIFIER

Random Forest is a robust machine learning algorithm that can be used for a variety of tasks including regression and classification.
It is an ensemble method, meaning that a random forest model is made up of a large number of small decision trees, called estimators, which each produce their own predictions.
The random forest model combines the predictions of the estimators to produce a more accurate prediction.

Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes.
Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Sigmoid activation: - In order to map predicted values to probabilities, we use the sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

$$f(x) = \frac{1}{1+e^{-(x)}}$$

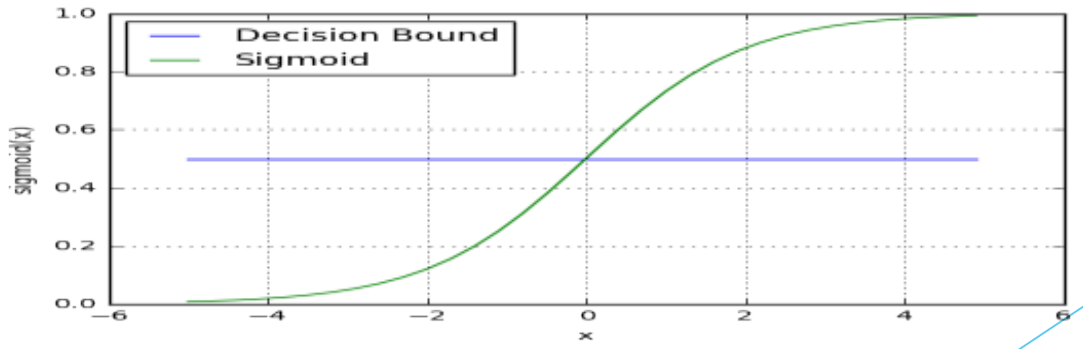
Logistic Regression

Our current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class, we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2.

$P \geq 0.5$, class=1

$P < 0.5$, class=0

For example, if our threshold was .5 and our prediction function returned .7, we would classify this observation as positive. If our prediction was 0.2, we would classify the observation as negative. For logistic regression with multiple classes we could select the class with the highest predicted probability.



List of Selected Features

As we discussed in the previous PPT we use Chi-Square method for feature selection
A chi-square test is used in statistics to test the independence of two events. Given the data of two variables, we can get observed count O and expected count E. Chi-Square measures how expected count E and observed count O deviates each other.

There will be a threshold value ‘k’, according to the given k value, the top k highest chi-squared valued features are selected.
For us it’s 17 so we consider 17 highest Chi-Square valued features

The Formula for Chi Square Is

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

where:

c = degrees of freedom

O = observed value(s)

E = expected value(s)

Here, we got 17 attributes after feature selection is done.

Label No.	FeatureName
5	src_bytes
0	Unnamed:0
6	dst_bytes
1	Duration
23	Count
24	srv_count
32	dst_host_count
33	dst_host_srv_count
10	Hot
12	logged_in
31	srv_diff_host_rate
36	dst_host_same_src_port_rate
35	dst_host_ diff_srv_rate
37	dst_host_ srv_diff_host_rate
30	diff_srv_rate

Mechanism to test the models' quality and validity

Testing is a process, which reveals errors in the program. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

Testing is the process of executing a program with the intent of finding an error.

During testing, the program is executed -with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

A test case contains all the information necessary to verify some particular functionality of the software.

We must employ different test techniques to execute a program or application with the intent of finding software bugs (errors or other defects).

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

❑*Unit Testing*

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

❑*Black Box Testing*

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

❑*White Box testing*

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases: Guarantee that all independent paths have been executed. Execute all logical decisions on their true and false Sides.

❑*Performance Testing*

It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software. There are different causes that contribute in lowering the performance of a software

- Network delay
- Client-side processing
- Database transaction processing
- Load balancing between servers
- Data rendering
- Load Testing

It is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating large input data.

❑*Stress Testing*

Stress Testing is a software testing technique that determines the robustness of software by testing beyond the limits of normal operation. Stress testing is particularly important for critical software but is used for all types of software. Stress testing emphasizes on robustness, availability and error handling under a heavy load rather than on what is correct behavior under normal situations.

LABEL DISTRIBUTION TRAINING SET

When we want to import the preprocessor technique then we should analyze 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

CRITERIA

Spark-Chi-SVM model for intrusion detection that can deal with Big Data. The proposed model used Spark Big Data platform which can process and analyze data with high speed. Big data have a high dimensionality that makes the classification process more complex and takes a long time. Therefore, in the proposed model, the researchers used Chi-Square Selector to select related features and SVM with SGD to classify data into normal or attack. The results of the experiment showed that the model has high performance and speed. In future work, the researchers can extend the model to a multi-classes model that could detect types of attack.

The work	Training time (s)	Predict time (s)
Spark-Chi-SVM	10.79	1.21
Clustering approach based on mini batch K-means	530.45	26.369
Real-time support vector machine based network	561.04	19.02
Hadoop based parallel binary bat algorithm	38.91	1.56

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 **testsize=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

Testing

S.No	Test cases	Output	Result	Resolution
1.	Verification of reading the input dataset	Positive	Dataset is displayed without any errors.	NA
2.	Verify the data cleaning process	Positive	Dataset is cleaned without any null values.	NA
3.	Verify the label encoding process	Positive	The categorical values have changed to numerical values.	NA
4	Using 10 features instead of 17 for 3 classifiers.	Positive	Still SVM has the highest accuracy.	NA
5.	Using 5 features instead of 17 for 3 classifiers.	Negative	Logistic Regression, Random Forest accuracies have been increased instead of SVM.	Minimum no.of features will benefit Logistic and Random Forest, but has the problem of under fitting. So, choose max no. of features.
6.	Training 80% dataset and testing 20% dataset	Positive	SVM has the highest accuracy.	NA
7.	Training 60% dataset and testing 40% dataset	Negative	The 3 classifiers have shown under fitting problem.	This type of dataset requires more percent for training.
8.	Processing time for each classifier	Positive	SVM has the least prediction time.	NA

Experience and lessons

We have Selected the machine learning use cases with care and established success measures. We have Started by defining our company's problems, identifying those that can be solved with machine learning, and developing clear success indicators. Data readiness, business effect, and machine learning applicability are things to consider. We understood that domain specialists and technology experts must collaborate. The correct team is essential for selecting the best machine learning use case and ensuring the project is implemented successfully. This aids in bridging cultural divides.

We concentrated on developing a machine learning architecture that works regardless of the knowledge level and takes advantage of current tools. Identify and address any skills gaps To finish the assignment effectively, we concentrated on educating ourselves and staying current with new versions and technology. We made sure that we had quick access to the data we needed and a well-rounded data strategy .A solid data strategy is the foundation of effective machine-learning solutions. "Data is frequently identified as the top barrier to adopting machine learning since our machine learning model is only as good as the data it's trained on.

Future work

The continued reduction in reliance on signatures in intrusion detection

- The growth of intrusion prevention
- Advances in data correlation and alert correlation methods
- Advances in source determination
- Inclusion of integrated forensics functionality in IDSs and IPSs

Conclusion

Spark-Chi-SVM model for intrusion detection that can deal with Big Data. The proposed model used Spark Big Data platform which can process and analyze data with high speed. Big data have a high dimensionality that makes the classification process more complex and takes a long time. Therefore, in the proposed model, the researchers used Chi- Square Selector to select related features and SVM with SGD to classify data into normal or attack. The results of the experiment showed that the model has high performance and speed. In future work, the researchers can extend the model to a multi-classes model that could detect types of attack.

The work	Training time (s)	Predict time (s)
Spark-Chi-SVM	10.79	1.21
Clustering approach based on mini batch K- means	530.45	26.369
Real-time support vector machine based network	561.04	19.02
Hadoop based parallel binary bat algorithm	38.91	1.56

Thank You