ADIKAVI NANNAYA UNIVERSITY
UNIVERSITY COLLEGE OF ENGINEERING

INTRUSION DETECTION SYSTEM

PROJECT GUIDE,
DR.D.LATHA,
ASSISTANT PROFESSOR.

PRESENTED BY,
CH.B.NEERAJ SINGH,
178297601003(CSE).

CONTENTS

- Introduction
- Objective.
- Requirements.
- Implementation.
- Conclusion

INTRODUCTION

- What is Intrusion ?
- Types of intruders
  - Masquerader (Not authorized)
  - Misfeasor(Misuses privileges)
  - Clandestine(Seizes Supervisory Control)

- Types of Intrusion detection system:
  - **Network IDS**
  - Host IDS
  - Protocol-based IDS
  - Application protocol –based IDS
  - Hybrid IDS

Continue…

- Classification of IDS:
  - **Signature based IDS**
  - Anomaly based IDS

OBJECTIVE

- To reduce the human intervention.
- To detect intrusions.
- To experiment machine learning algorithm in the domain of Cyber security.

REQUIREMENTS

- Software Requirements:
  - Python programming language.
  - Jupyter Notebook (Python editor).
  - Windows OS/unix.
- Hardware Requirements:
  - Fluently working Laptops (64 bit preferable).
  - RAM minimum 6GB .

IMLEMENTATION

- The system under following steps:

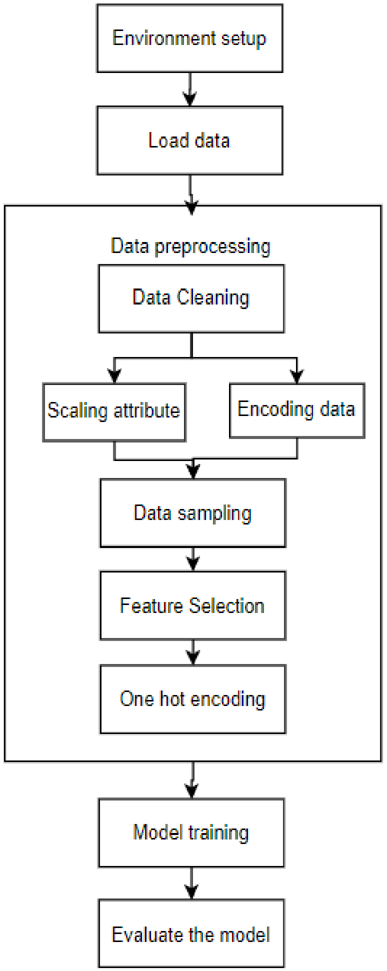
Environment Setup

Load Data

Data Preprocessing

Model Training

Evaluate the model



Continue….

- Environment Setup:
  - Setting Jupyter notebook
  - Installing and importing the packages
    - Numpy (Numerical and Mathematical Operations)
    - Seaborn (Data visualization library)
    - Padas (Data Frames)
    - Matplotlib (plots)
    - Sklearn (tools for machine learning)
    - Imblearn (re-sampling techniques)

Ex: import Numpy

- Load Dataset:
  - Kaggel
  - Test Dataset (125973 *42)
  - Train Dataset(22544 *42)
  - Some attributes are
    - count
    - dst_host_srv_count
    - Src_bytes
    - Srv_count
    - dst_host_serror_rate etc………

Pd.read_csv(‘Train.csv’)

Continue…….

- Data Preprocessing
  - Data Cleaning
    - Missing values
    - Dummy attributes

drop()

  - Scaling numerical attribute
    - Standardization (sklaern)
    - Mean=0 and Standard deviation =1
    - Z=x-M/SD

```
scaler = StandardScaler()
 scaler.fit_transform(df_train.select_dtypes(include=['float64','int64']))
```

  - Encoding the categorical data
    - Label Encoder (sklearn)
    - Categorical values to numerical labels
    - Alphabetical order

Labels	Dos	Normal	Probe
Label after encoding	0	1	2

```
encoder = LabelEncoder()
cattrain = df_train.select_dtypes(include=['object']).copy()
traincat = cattrain.apply(encoder.fit_transform)
```

Continue……….

- Data Sampling
  - Solve the class imbalance problem
  - Random oversampling (imblearn)

```
ros = RandomOverSampler(random_state=42)
X_res, y_res = ros.fit_sample(X, y)
```

- Feature Selection
  - Random forest
  - Important features only
  - Performance
  - Recursive feature elimination

```
rfe = RFE(rfc, n_features_to_select=10)
rfe = rfe.fit(X_res, y_res)
```

  - 'src_bytes', 'dst_bytes', 'logged_in', 'count', 'srv_count', 'dst_host_srv_count', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_serror_rate', 'service'

Continue….

- One hot Encoding
  - Draw back on label encoding
  - Interger label to binary label

Label encoding

Dos	0
Normal	1
Probe	2

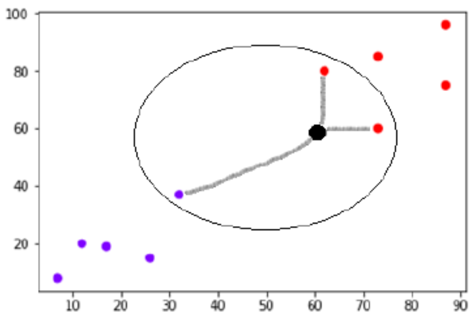
one hot encoding

Dos	Normal	Probe
1	0	0
0	1	0
0	0	1

- Model training
  - K-NearestNeighbour classifier
  - Logistic Regression

Continue….

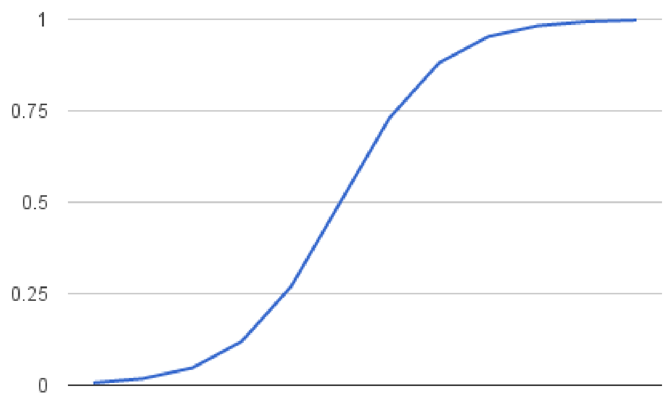
- K-Nearest Neighbor:
  - Non parametric ,Lazy learning
  - Distance Euclidean ,Hamming distance etc……….
  - Algorithm:
    - Initialize the value of k
    - for i=0 to m: Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method.
    - Sort the calculated distances in ascending order based on distance values
    - Get top k rows from the sorted array
    - Return the majority label among S.




```
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);
```

Continue…

- Logistic Regression:
  - Probability based (0-1)
  - Sigmoid function: 1/1+e^-x
  - Developed from linear regression.




```
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);
```

- Evaluating the model
  - Accuracy Score:
    - Number of correct predictions / Total number of predictions

accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))

  - Confusion matrix:
    - Accuracy=(true postive +true negative)*100/total samples

confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))

  - Classification Report:
    - Precission: percent of correct prediction by model =TP/(TP+FP)
    - Recall:fraction of positive that are correctly identified by the classifier =TP/(TP+FN)
    - F1-Score:weighted mean of persision and recall = (2*Recall*percision)/(recall+Percission)

classification = metrics.classification_report(Y_test, v.predict(X_test))

```
============================== Model Evaluation =========

Model Accuracy:
 0.8946356805871046

Confusion matrix:
 [[5893 1565]
 [ 244 9467]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.96      0.79      0.87      7458
         1.0       0.86      0.97      0.91      9711

    accuracy                           0.89     17169
   macro avg       0.91      0.88      0.89     17169
weighted avg       0.90      0.89      0.89     17169
```

```
============================== Model Evaluation ==========

Model Accuracy:
 0.8525248995282194

Confusion matrix:
 [[5843 1615]
 [ 917 8794]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.86      0.78      0.82      7458
         1.0       0.84      0.91      0.87      9711

    accuracy                           0.85     17169
   macro avg       0.85      0.84      0.85     17169
weighted avg       0.85      0.85      0.85     17169
```

CONCLUSION

– The system got the accuracy of 89% through KNN and 85% through Logistic regression

KNN classifier	89% accuracy
Logistic Regression	85% accuracy

– A similar system is implemented by J.S.Sirisha (178297601004) and obtained an accuracy of 82.88% using decision tree algorithm and 86.97% using Naive Bayes algorithm.

– From the results KNN got the higher accuracy.

- What I look forward to use:
  - Boosting techniques
  - Implementation is done using real life packets.
  - Advance algorithms and technologies.
  - Mainly your suggestions

Thank you