ML-Driven Cybersecurity: Analyzing and Mitigating Digital Threats

BY

Srujan Putta [2020BCS0074]

Sankalp Sodagum [2020BCS0014]

Thirdhakshar Sai Ghanta [2020BCS0036]

Bhukya Akash [2020BCS0181]

Guided By,

Dr.Divya Sindhu Lekha

Indian Institute of Information Technology Kottayam

TABLE OF CONTENTS:

Introduction

- **Growing Cyber Threat Landscape**: In today's highly interconnected world, cyber threats ranging from malware to data breaches, have become increasingly prevalent.

- **Limitations of Traditional Cybersecurity**: Traditional cybersecurity measures are effective against known threats such as *RDP access vulnerability* and *established viruses*. However, they often fall short when faced with advanced threats like *fileless malware* and *zero-day exploits* delivered via email.

- **The Need for Advanced Solutions**:To effectively counter these evolving cyber threats, it is imperative to move beyond relying solely on *human-driven detection methods* and *traditional security measures*.

- **Augmentation with Machine Learning**: One of the key strategies to combat the changing cyber threat landscape is the integration of advanced solutions like *Machine Learning*. This approach allows for adaptation and learning, enhancing the overall security posture.

- **Adaptation to a Dynamic Environment**:By incorporating machine learning and other cutting-edge technologies, organizations can better adapt to the ever-changing cyber threat landscape, ensuring a more robust and proactive defense against emerging threats.

Significance of ML in Cyber Security

Indian Institute of Information Technology Kottayam

- **Protection of Sensitive Data:** The work in cybersecurity helps protect sensitive information such as personal data, financial records, and intellectual property from being compromised by cyberattacks, safeguarding individuals and organizations from potential financial and reputational damage.
- **Business Continuity:** Effective cybersecurity measures are crucial for ensuring the continuity of business operations. these efforts help organizations mitigate the risks associated with cyber threats, reducing downtime and financial losses.
- **National Security:** In an era where cyber-attacks can have significant geopolitical implications, the work in cybersecurity plays a role in preserving national security by protecting critical infrastructure, government systems, and sensitive defense information.
- **Technological Advancement:** As technology continues to advance, so do cyber threats. this work is essential for driving innovation in the cybersecurity field, developing new strategies and technologies to stay ahead of malicious actors and secure the digital future.

Literature Review 1/7

S. No.	Authors	Title	Findings	Challenges
1.link	Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A.V. and Di Franco, F., *(ACM Journal on Digital Threats, 2023)*	**The Role of Machine Learning in Cybersecurity**	1. Machine learning (ML) is a powerful tool that can improve cybersecurity. 2. ML can detect cyber threats, prevent attacks, and respond to incidents. 3. Issues involved: a. The need for labeled data, b. The difficulty of explaining ML decisions. c. The vulnerability of ML models to adversarial attacks. 4. To address these issues, it is important for researchers, practitioners, and policymakers to work together. 5. Real-world applications of ML in cybersecurity: a. Detecting malware b. Predicting attacks c. Responding to incidents.	1. Collaboration between academia and industry, 2. Evaluation and benchmarking, 3. Ethical and privacy considerations

Literature Review 2/7

S. No.	Authors	Title	Findings	Challenges
2.link	Chaofan Lu *(IEEEXplore,* *Intl. Conf. on Electronics and Devices, Computational Science (ICEDCS),* *France, 2022)*	**Research on the technical application of artificial intelligence in network intrusion detection system**	1. Network security is crucial in today's internet landscape. 2. NIDS structure and traditional NIDS models. 3. AI-powered Network Intrusion Detection Systems (**NIDS**) form a vital layer of defense against evolving threats. 4. NIDS use AI to monitor real-time network traffic and detect anomalies. 5. ANN-driven NIDS excel at recognizing complex intrusion patterns. 6. AI-powered NIDS form a crucial component of modern network security.	**Rapid information feedback is a double-edged sword:** A. It can be used to prevent the spread of viruses, but B. it can also pose a threat to users' privacy. AI technology needs to be developed to *protect personal privacy while still processing information at high speed.*

Literature Review 3/7

S. No.	Authors	Title	Findings	Challenges
3.Link	Zhang, Z., Al Hamadi, H., Damiani, E., Yeun, C.Y. and Taher, F., *(IEEE Access*, 2022)	**Explainable Artificial Intelligence(XAI) Applications in Cyber Security: State-of-the-Art in Research.**	1. XAI is a technique to make AI more understandable to humans. 2. AI in cybersecurity often lacks transparency, so XAI is needed to create accurate and explainable models. 3. AI models in cybersecurity can make costly errors, so developers often prioritize accuracy over interpretability. 4. The XAI domain is characterized by intelligibility, explainability, transparency, and interpretability. 5. Wireshark and WinDump are efficient tools for small-scale cybersecurity data capture. Datasets: Benchmark Cyber Security	1. XAI security, 2. Legal and Privacy Issues, 3. The trade-off between interpretability and accuracy.

Literature Review 4/7

S. No.	Authors	Title	Findings	Challenges
4.Link	Azam, Z., Islam, M.M. and Huda, M.N., *(IEEE Access, 2023)*	**Comparative Analysis of Intrusion Detection Systems and Machine Learning-Based Model Analysis Through Decision Tree**	1. Traditional IDS systems detect anomalies through a. predefined signatures or rules, b. employing pattern matching and c. rule-based methods to spot known attacks or suspicious activity. 2. ML-based IDS systems learn from data to detect anomalies and adapt to new attacks using semi-supervised learning, where a small labeled dataset teaches normal and abnormal behavior, while a larger unlabeled dataset fine-tunes the model for improved accuracy. a. ML-based IDS systems employ Decision Trees for b. feature selection, c. rule generation, d. pattern identification, and e. classification of network logs into normal or malicious instances, enhancing accuracy and interpretability.	**Real-time implementation of ML on mobile and wearable devices:** Deep learning algorithms used in IDS systems can be computationally intensive and require significant parameter tuning and initialization. Implementing these algorithms on mobile and wearable devices with limited resources is a challenge that needs to be addressed.

Literature Review 5/7

S. No.	Authors	Title	Findings	Challenges
5.Link	Sarker, I.H., Abushark, Y.B., Alsolami, F. and Khan, A.I., *(MDPI, Symmetry, 2020)*	**IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model**	1. The paper presents a machine learning-based intrusion detection model called ***IntruDTree***, which utilizes a tree-based generalized model to detect cyber intrusions. 2. The model ranks the security features according to their importance score and selects a set of significant features for further processing, reducing the dimensionality of the dataset without losing significant information. 3. Experimental results show that the IntruDTree model effectively detects anomalies or intrusions, with high precision, recall, fscore, and accuracy for both normal and anomaly classes. 4. The model outperforms all the traditional machine learning classification-based methods in terms of effectiveness. Datasets: Network Intrusion Detection,NSL-KDD	1. Handling Imbalanced Data, 2. Incorporating Real-Time Data, 3. Adapting to Evolving Cyber Threats, 4. Interpreting Model Decisions, 5. Generalization to Different Domains

Literature Review 6/7

SNO.	Authors	Title	Findings	Challenges
6.Link	Yang, Y., Li, H. and Jing, D., *(IEEEXplore, 8th Annual International Conference on Network and Information Systems for Computers (ICNISC),* 2022)	**A Phishing Website Detection Method Based on Multi-layer Perceptron with Mutual Information Feature Selection**	Method: The paper proposes a phishing website detection method called MI-MLP, which combines mutual information feature selection with a multi-layer perceptron model. Advantages: The method improves the accuracy of phishing website detection compared to other methods by selecting relevant features and training the model using a dataset. Experimental results: Experimental results demonstrate the superiority of the MI-MLP method in terms of 91.46% accuracy, 87.19 precision,87.19 recall, and 87.19 F1 score. Dataset used:-UCI Dataset	1. Models explanation for large datasets and real life exploration is missing, 2. scalability and applicability of this model is missing

Literature Review 7/7

SNO.	Authors	Title	Findings	Challenges
7.Link	Furat Turk *(Cankiri Karatekin University, Department of Computer Engineering, Cankiri/Turkey)*	**Analysis of Intrusion Detection Systems in UNSW-NB15 and NSL-KDD Datasets with Machine Learning Algorithms**	Method: The paper focuses on the analysis of intrusion detection using machine learning algorithms.Models like KNN, Logistic Regression (LR), LSTM, Decision Trees, Random Forest have been discussed in this paper. Deductions: In the UNSW-NB15 dataset, RF and LR algorithms gave the best results, respectively, and in the NSL-KDD dataset, the Decision Trees, Random Forest ,MLP and LSTM networks gave the best results. Experimental results: *For NSL-KDD:-* { Decision Tree and Random Forest:-0.96 MLP accuracy:-0.978 & LSTM accuracy:-0.975 } Dataset used:-UNSW-NB15 and NSL-KDD datasets	1. As attackers adapt and develop new techniques, the dataset used for training and testing intrusion detection systems may quickly become outdated. 2. Training these advanced models on large and diverse datasets, especially in real-time scenarios, demands substantial computing resources

Problem Statement

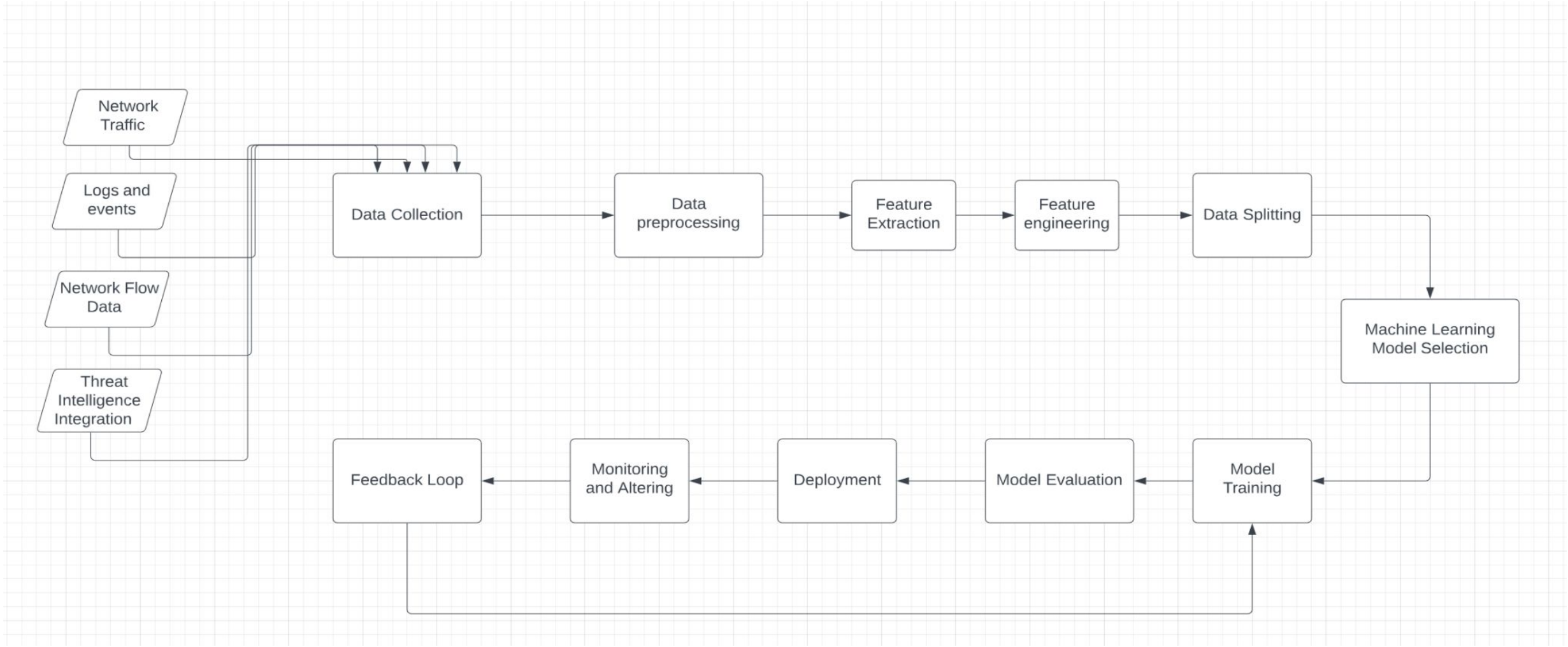
Detecting Evolving Cyber Threats with Network Intrusion Detection System (NIDS)

- **Background**: In an era of increasing connectivity and digital dependence, organizations face a constant barrage of cyber threats, ranging from traditional attacks to highly sophisticated, zero-day exploits. These threats pose significant risks to data security, privacy, and business continuity.

- **Challenges**: Traditional cybersecurity measures are often inadequate in detecting and preventing advanced and evolving cyber threats. This gap in security readiness highlights the need for an advanced Network Intrusion Detection System (NIDS) that can adapt to the changing threat landscape.

- **Scope**:Our project focuses on the design and implementation of a AI integrated NIDS capable of identifying and mitigating various types of network-based attacks, including malware, intrusions, and various other threats.

Architecture



Architecture

The firewall-NIDS serves as a critical defense mechanism against cyber threats by continuously monitoring network traffic, and taking appropriate actions to protect network assets and data.

Data Handling: Network traffic data is collected, cleaned, and transformed into a suitable format.

Model Selection and Training: Machine learning models are chosen, trained, and optimized for intrusion detection.

Deployment and Monitoring: Trained models are deployed in production, continuously monitored, and adjusted for real-time threats.

Feedback and Adaptation: Models learn from detected intrusions and false positives, adapting to changing network behaviors.

Feature Engineering: Involves developing domain-specific features that effectively capture network behavior and transforming categorical features into numerical representations. This step enhances the model's ability to detect and differentiate network intrusions.

Key Objectives:

1. **Zero-Trust Architecture (ZTA):**Implementing a Zero-Trust approach within NIDS which assumes that no user or system can be trusted by default, regardless of their location within the network. This approach requires continuous verification and validation of all network traffic and user behavior.
2. **Advanced Threat Detection**: Works on algorithms and techniques that can identify and classify known threats, as well as recognize and adapt to emerging and zero-day threats.
3. **Adaptability**: Implement machine learning and artificial intelligence capabilities to enable the NIDS to learn from new data and adjust its detection strategies accordingly.
4. **Context-Aware Threat Detection:** Enhancing with context-aware detection capabilities that take into account not only network traffic patterns but also contextual information such as user behavior, device profiles, and business processes.

5.Homomorphic Encryption for Data Privacy: Explore homomorphic encryption techniques to analyze network traffic for anomalies without decrypting sensitive data, thus ensuring data privacy and compliance with privacy regulations like GDPR.

6.Artificial Intelligence(XAI) : XAI in NIDS system to provide clear, interpretable insights into network security decisions, fostering transparency and building trust within organizations.

Expected Outcomes:

By successfully addressing these objectives, our project aims to provide organizations with a robust NIDS that enhances their cybersecurity posture, enabling them to detect and respond to cyber threats more effectively, ultimately safeguarding their data, operations, and reputation.

Significance:

The project holds significant importance in the context of modern cybersecurity, as it contributes to the development of proactive defense mechanisms against the constantly evolving landscape of cyber threats, helping organizations stay one step ahead of malicious actors.

Experimental Results

DataSet Used : NSL-KDD

1. The NDSL-KDD dataset is derived from the complete KDD dataset, offering an opportunity to correct errors present in the KDD99 cup dataset [7].

2. However, the NDSL-KDD dataset has certain limitations, including a lack of representation for low-footprint attacks [7].

3. Notably, the NDSL-KDD dataset boasts better reduction rates and an absence of duplicate records in the test set.[7]

4. With fewer data points than KDD-99, it proves to be a cost-effective choice for training machine learning models.[7]

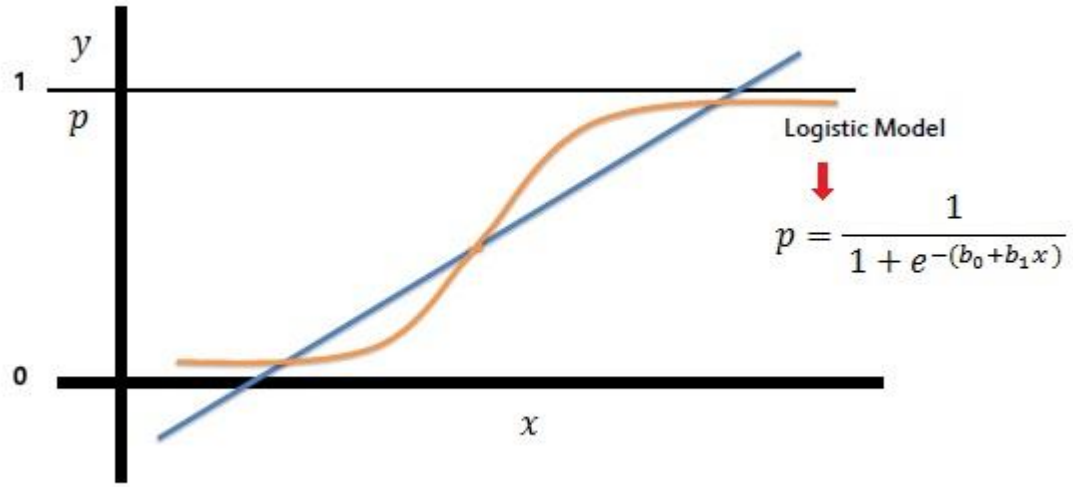
5. It supports binary classification (normal vs. abnormal) and multiclass classification (normal, DOS, R2L, U2R, and probe) and exhibits various data distribution patterns.[7]

Logistic Regression

This is a valuable machine learning technique used to model the relationship between class variables and features. It is commonly applied in binary classification problems where it calculates the probability of an observation belonging to a class, ranging between 0 and 1. With minor adjustments, LR can also be extended to handle multi-class classification tasks, making it a versatile tool for various predictive modeling scenarios.

$$\text{Logit}(pi) = 1/(1+ \exp(-pi))$$

$$\ln(pi/(1-pi)) = \text{Beta\_0} + \text{Beta\_1}*X\_1 + \ldots + B\_k*K\_k$$



Results

Indian Institute of Information Technology Kottayam

Training Accuracy Logistic Regression 86.13777824665048 Test Accuracy Logistic Regression 86.37130801687763

Training Precesion Logistic Regression 88.20464234959735 Test Precesion Logistic Regression 88.10426540284361

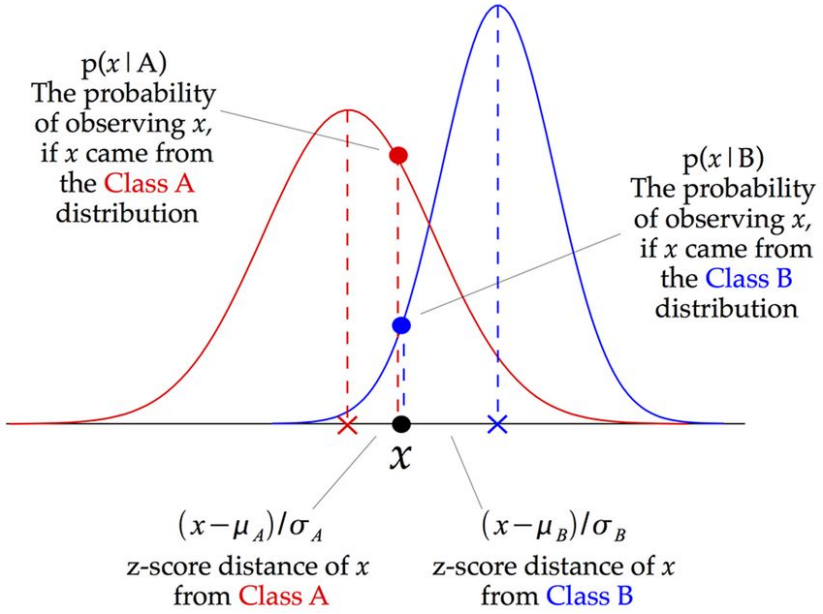
Training Recall Logistic Regression 95.90522791655937 Test Recall Logistic Regression 96.27136198860694

Training F1 Score Logistic Regression 91.89389265885256 Test F1 Score Logistic Regression 92.00692897797575

Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is another machine learning method, particularly useful for classification tasks. It's based on the Naive Bayes theorem and assumes that features are normally distributed within each class. GNB is commonly applied in binary or multi-class classification problems, where it calculates the likelihood of an observation belonging to a class based on the distribution of its features. This method is particularly effective when dealing with continuous and real-valued data, making it a valuable tool in a wide range of classification scenarios.

$$P(X|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}}$$

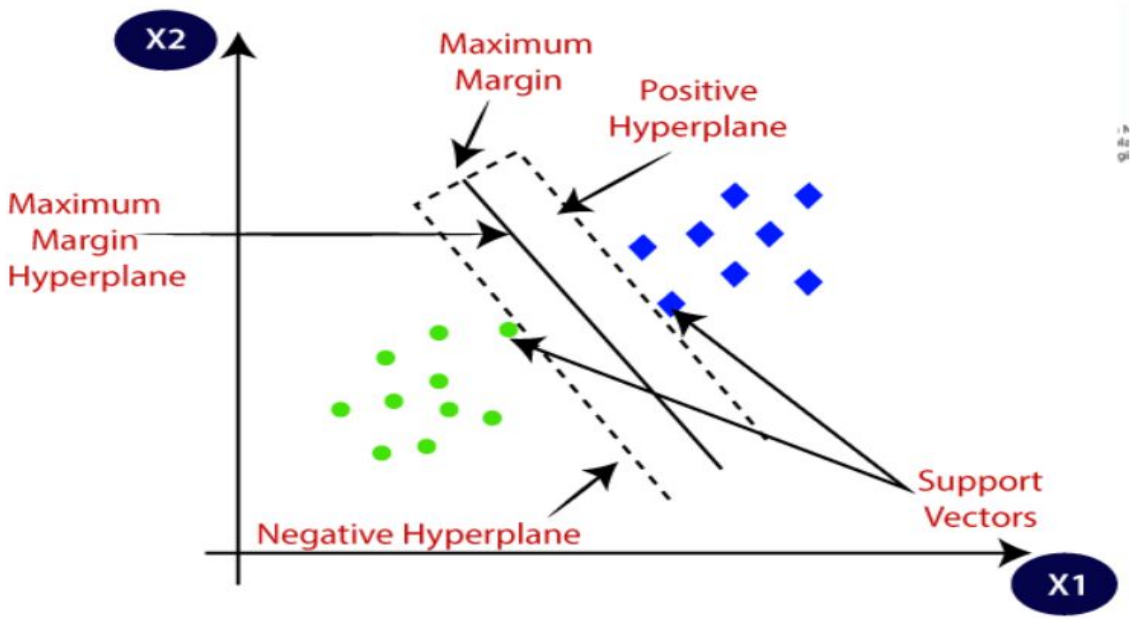


Results

Training Accuracy GaussianNB 79.71304989977845 Test Accuracy GaussianNB 79.7046413
5021096
Training Precesion GaussianNB 95.82745098039216 Test Precesion GaussianNB 95.19950
124688279
Training Recall GaussianNB 78.66340458408447 Test Recall GaussianNB 79.07819782496
117
Training F1 Score GaussianNB 86.4012446078778 Test F1 Score GaussianNB 86.39321074
96464

Linear Support Vector Classification

This is a powerful machine learning technique used for classification tasks. It works by finding a hyperplane that best separates data points into different classes. Linear SVC is particularly effective in binary and multi-class classification problems, offering robust performance when dealing with complex datasets. It is known for its ability to handle high-dimensional data and can be a valuable tool for various classification tasks, making it a popular choice in machine learning applications



Results

Training Accuracy Linear SVC(LBasedImpl) 97.25334153626324 Test Accuracy Linear SV
C(LBasedImpl) 97.15816630283787
Training Precesion Linear SVC(LBasedImpl) 96.09307970787034 Test Precesion Linear
SVC(LBasedImpl) 95.87296336117774
Training Recall Linear SVC(LBasedImpl) 98.07564981525384 Test Recall Linear SVC(LB
asedImpl) 98.1624184943687

K-Nearest Neighbours(KNN)

Indian Institute of Information Technology Kottayam

The k-Nearest Neighbors (KNN) algorithm is a versatile machine learning approach commonly used for classification and regression tasks. KNN relies on the idea of proximity, where an observation is classified based on the majority class of its k-nearest neighbors in the feature space. This algorithm is highly adaptable and applicable to both binary and multi-class classification problems. KNN is effective for a wide range of data types and can handle complex decision boundaries. It is a valuable tool for various classification and regression tasks in machine

```
Training Accuracy KNeighborsClassifier 99.05236313841452  Test Accuracy KNeighbors
Classifier 98.93629688430245
Training Precesion KNeighborsClassifier 99.22512234910276  Test Precesion KNeighbo
rsClassifier 99.05636317266003
Training Recall KNeighborsClassifier 98.73133850195424  Test Recall KNeighborsClas
sifier 98.67050554661698
```

Euclidean distance

Given two points A and B in d dimensional space such that $A = [a_1, a_2, \cdots, a_d]$ and $B = [b_1, b_2, \cdots, b_d]$, the Euclidean distance between A and B is defined as:

$$||A - B|| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \quad (1)$$

The corresponding cost function ϕ that is minimized when we assign points to clusters using the Euclidean distance metric is given by:

$$\phi = \sum_{x \in X} \min_{c \in C} ||x - c||^2 \quad (2)$$

Manhattan distance

Given two random points A and B in d dimensional space such that $A = [a_1, a_2, \cdots, a_d]$ and $B = [b_1, b_2, \cdots, b_d]$, the Manhattan distance between A and B is defined as:

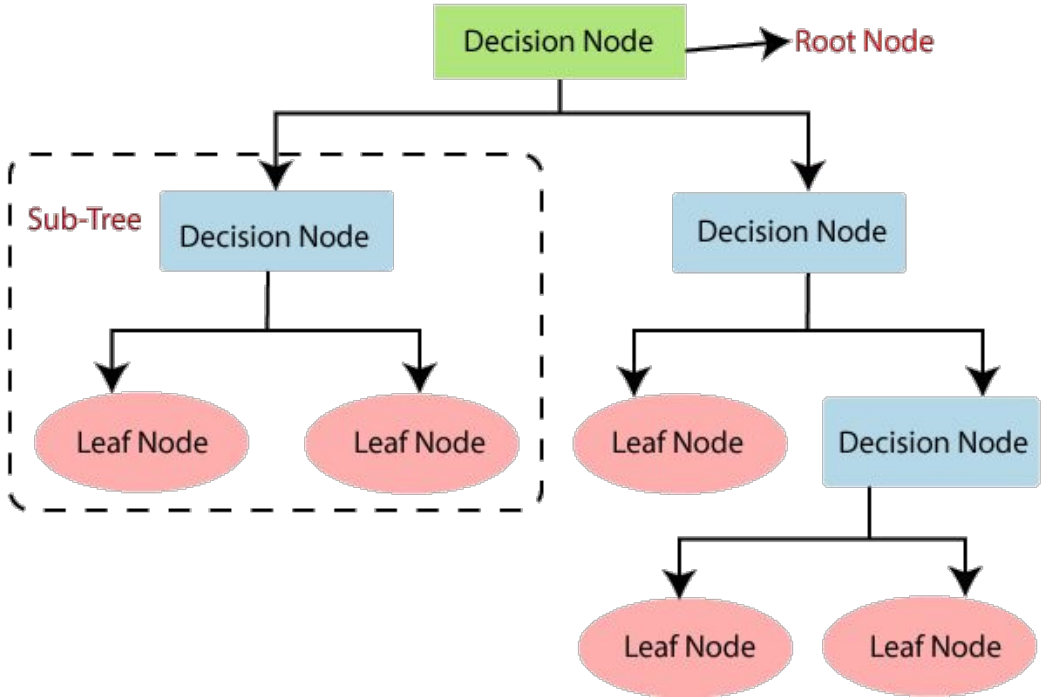
$$|A - B| = \sum_{i=1}^{d} |a_i - b_i| \quad (3)$$

The corresponding cost function ψ that is minimized when we assign points to clusters using the Manhattan distance metric is given by:

$$\psi = \sum_{x \in X} \min_{c \in C} |x - c| \quad (4)$$

Decision Tree

The Decision Tree algorithm is a fundamental machine learning method for predictive modeling. It creates a tree-like structure where each internal node represents a feature, each branch corresponds to a decision, and each leaf node is an outcome or class label. Decision Trees are versatile and can be used for both classification and regression tasks. They are particularly effective in scenarios involving binary and multi-class classification. Decision Trees are interpretable and can handle a mix of data types. They serve as valuable tools in various machine learning applications, making them a popular choice for decision-making and predictive modeling.learning.



Results

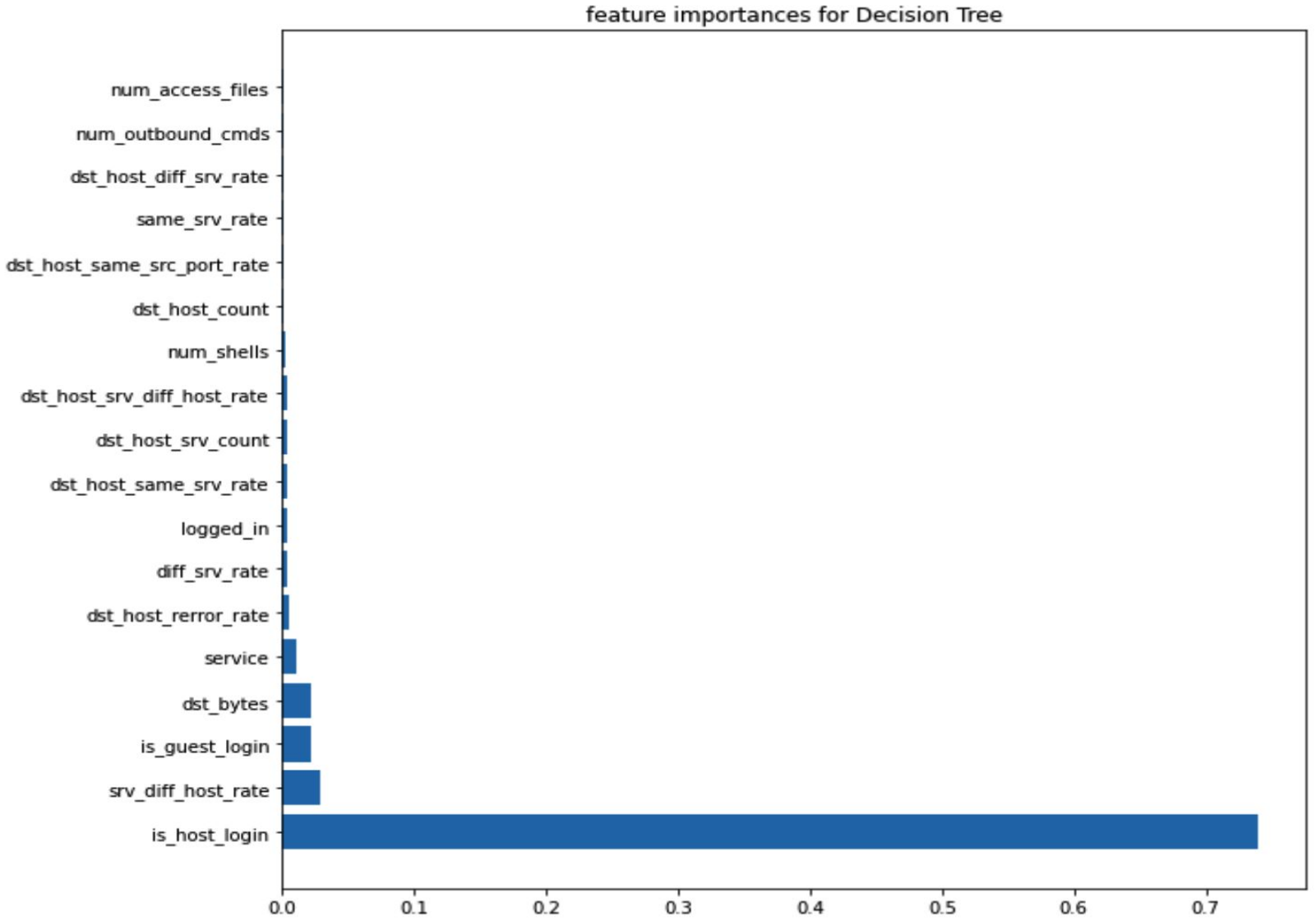
Training Accuracy DecisionTreeClassifier 99.99404626055548 Test Accuracy Decision TreeClassifier 99.8531454653701

Training Precesion DecisionTreeClassifier 100.0 Test Precesion DecisionTreeClassifier 99.8306806637318

Training Recall DecisionTreeClassifier 99.98718523739348 Test Recall DecisionTree Classifier 99.8560200186299

Feature Importances For Decision Tree

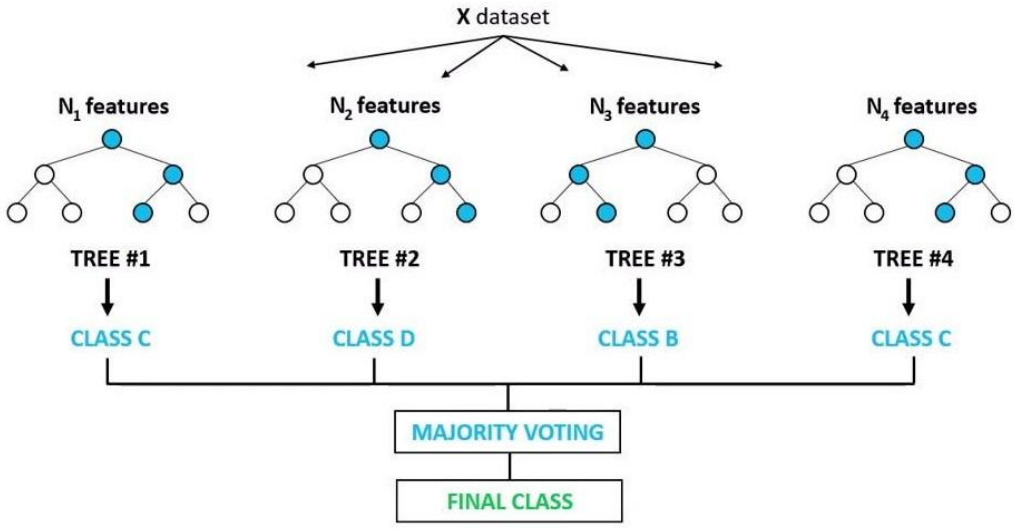
Indian Institute of Information Technology Kottayam



RandomForestClassifier

Indian Institute of Information Technology Kottayam

The RandomForestClassifier is a robust machine learning ensemble method that builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. It is widely used for both classification and regression tasks. The RandomForestClassifier is especially effective in binary and multi-class classification problems, providing robust results in a variety of scenarios. This method is known for its ability to handle complex, high-dimensional data, making it a valuable tool in machine learning applications. The ensemble nature of Random Forest enhances its predictive power and is a popular choice for decision-making and predictive modeling.

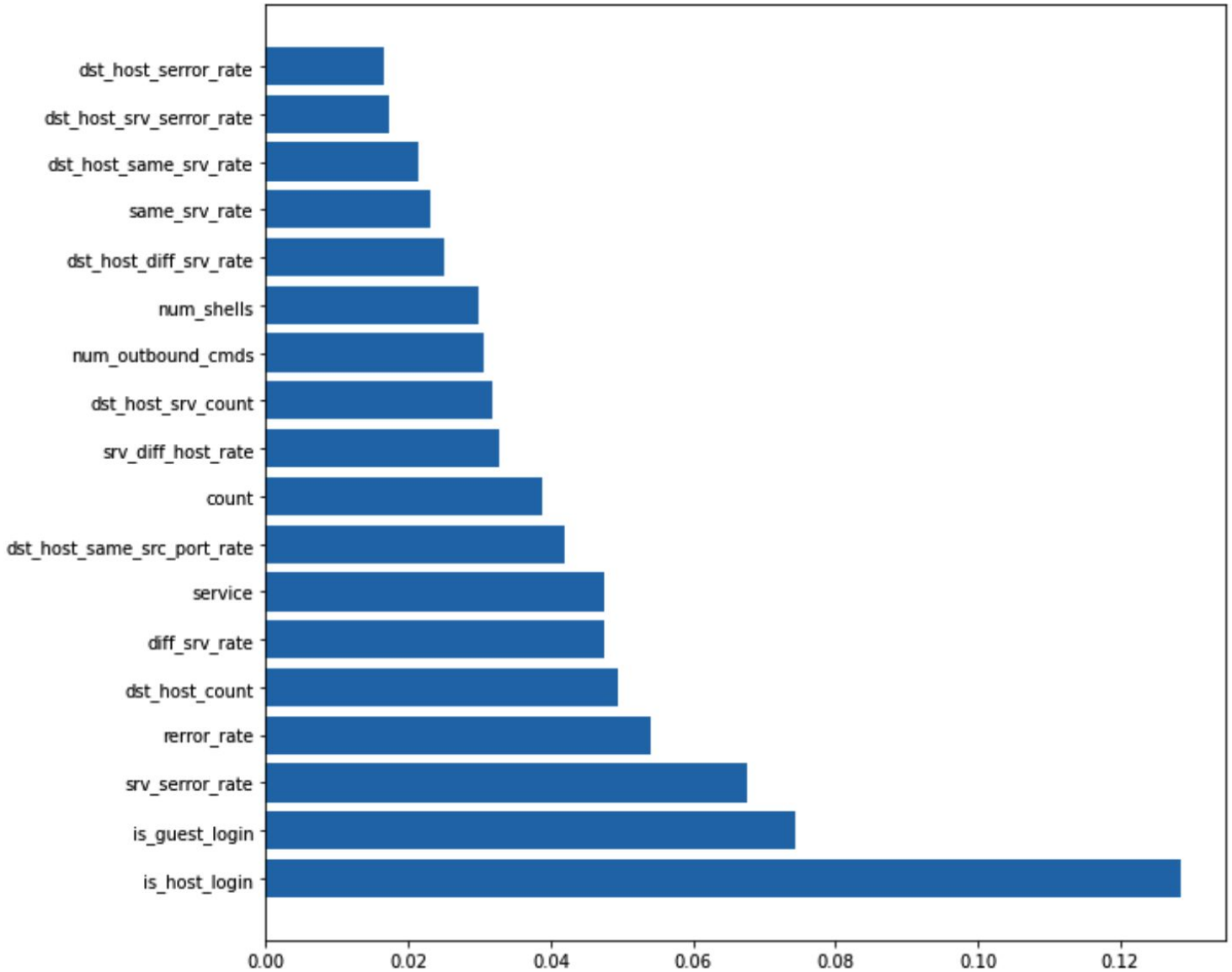


Results

Training Accuracy RandomForestClassifier 99.99404626055548 Test Accuracy RandomFo
restClassifier 99.89283588013494
Training Precesion RandomForestClassifier 99.99359261869674 Test Precesion Random
ForestClassifier 99.94912667458028
Training Recall RandomForestClassifier 99.99359261869674 Test Recall RandomForest
Classifier 99.8221695317131

Feature Importance For Random forest Classifier

Indian Institute of Information Technology Kottayam



Principal Component Analysis (PCA)

Indian Institute of Information Technology Kottayam

Principal Component Analysis (PCA) is a foundational machine learning technique used for dimensionality reduction and data transformation. It aims to capture the most significant variance in the data by identifying and creating a new set of uncorrelated variables called principal components. PCA is versatile and can be applied to various tasks, including classification and regression. It is effective in both binary and multi-class classification scenarios, providing a more compact representation of the data. PCA is particularly valuable for visualizing data and reducing computation complexity while preserving essential information, making it a popular choice in machine learning applications.

```
Training Accuracy PCA RandomForest 99.99404626055548  Test Accuracy PCA RandomForest 99.8293312165112
Training Precesion PCA RandomForest 99.9914571898426  Test Precesion PCA RandomForest 99.92362525458248
Training Recall PCA RandomForest 99.99572841246449  Test Recall PCA RandomForest 99.71208400372598
```

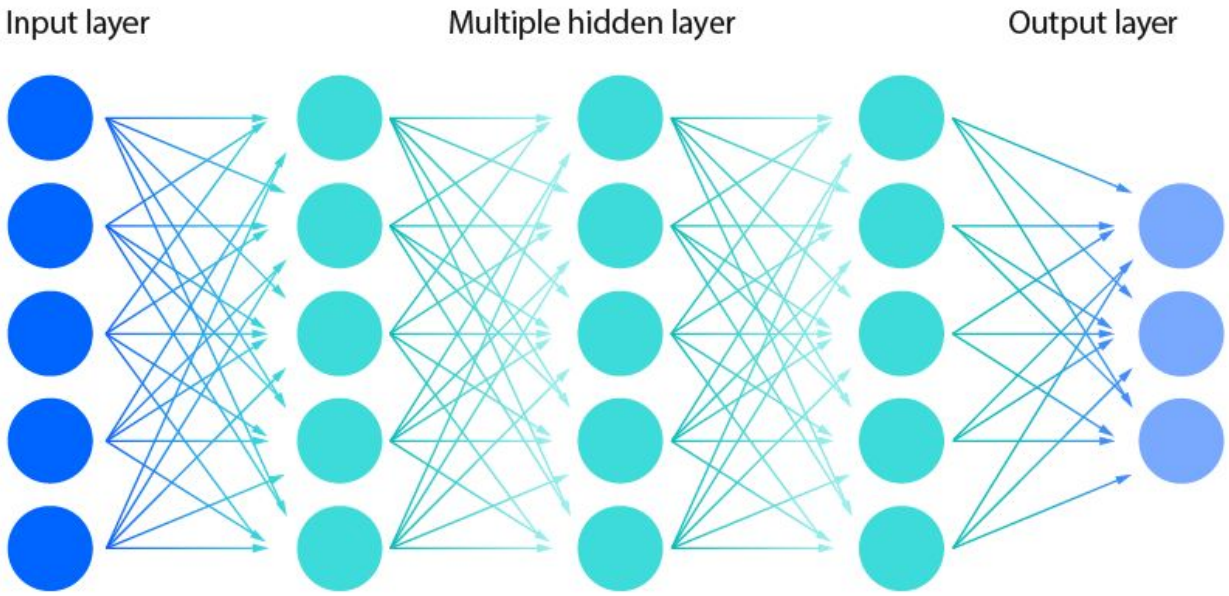
Neural Networks

Indian Institute of Information Technology Kottayam

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity.

$$\sum w_i x_i + bias = w_1x_1 + w_2x_2 + w_3x_3 + bias$$

$$output = f(x) = 1 \text{ if } \sum w_1x_1 + b \geq 0; \text{ 0 if } \sum w_1x_1 + b < 0$$



```
Epoch 1/10
3150/3150 [==============================] - 20s 6ms/step - loss: 1163.1600 - accuracy: 0.9538 - val_los
s: 425.0167 - val_accuracy: 0.9752
Epoch 2/10
3150/3150 [==============================] - 17s 5ms/step - loss: 307.9906 - accuracy: 0.9746 - val_los
s: 90.3703 - val_accuracy: 0.9849
Epoch 3/10
3150/3150 [==============================] - 17s 5ms/step - loss: 146.5123 - accuracy: 0.9790 - val_los
s: 22.0090 - val_accuracy: 0.9838
Epoch 4/10
3150/3150 [==============================] - 18s 6ms/step - loss: 92.3532 - accuracy: 0.9752 - val_loss:
6.3909 - val_accuracy: 0.9742
Epoch 5/10
3150/3150 [==============================] - 17s 5ms/step - loss: 20.2885 - accuracy: 0.9756 - val_loss:
0.8062 - val_accuracy: 0.9752
Epoch 6/10
3150/3150 [==============================] - 18s 6ms/step - loss: 3.4544 - accuracy: 0.9740 - val_loss:
0.4149 - val_accuracy: 0.9767
Epoch 7/10
3150/3150 [==============================] - 18s 6ms/step - loss: 2.3672 - accuracy: 0.9748 - val_loss:
0.1070 - val_accuracy: 0.9723
Epoch 8/10
3150/3150 [==============================] - 18s 6ms/step - loss: 0.1926 - accuracy: 0.9744 - val_loss:
0.0856 - val_accuracy: 0.9772
Epoch 9/10
3150/3150 [==============================] - 18s 6ms/step - loss: 0.0963 - accuracy: 0.9760 - val_loss:
8.5606 - val_accuracy: 0.9773
Epoch 10/10
3150/3150 [==============================] - 18s 6ms/step - loss: 1.6946 - accuracy: 0.9749 - val_loss:
0.0830 - val_accuracy: 0.9746
```

Demonstration

Indian Institute of Information Technology Kottayam

Exploring the dataset

In [2]:
```
# Read Train and Test dataset
data_train = pd.read_csv("../input/nslkdd/KDDTrain+.txt")
```

In [3]:
```
print(data_train)
```

```
        0  tcp  ftp_data   SF    491    0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  \
0       0  udp     other   SF    146      0    0    0    0    0    0    0    0
1       0  tcp   private   S0      0      0    0    0    0    0    0    0    0
2       0  tcp      http   SF    232   8153    0    0    0    0    0    1    0
3       0  tcp      http   SF    199    420    0    0    0    0    0    1    0
4       0  tcp   private  REJ      0      0    0    0    0    0    0    0    0
...    ..  ...       ...  ...    ...    ...  ...  ...  ...  ...  ...  ...  ...
125967  0  tcp   private   S0      0      0    0    0    0    0    0    0    0
125968  8  udp   private   SF    105    145    0    0    0    0    0    0    0
125969  0  tcp      smtp   SF   2231    384    0    0    0    0    0    1    0
125970  0  tcp    klogin   S0      0      0    0    0    0    0    0    0    0
125971  0  tcp  ftp_data   SF    151      0    0    0    0    0    0    1    0
```

In [5]:
```
columns = (['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot'
,'num_failed_logins','logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creations'
,'num_shells','num_access_files','num_outbound_cmds','is_host_login','is_guest_login','count','srv_count','serror_rate'
,'srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count'
,'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate'
,'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate','outcome','level'])
```

In [6]:
```
# Assign name for columns
data_train.columns = columns
```

In [7]:
```
data_train.head()
```

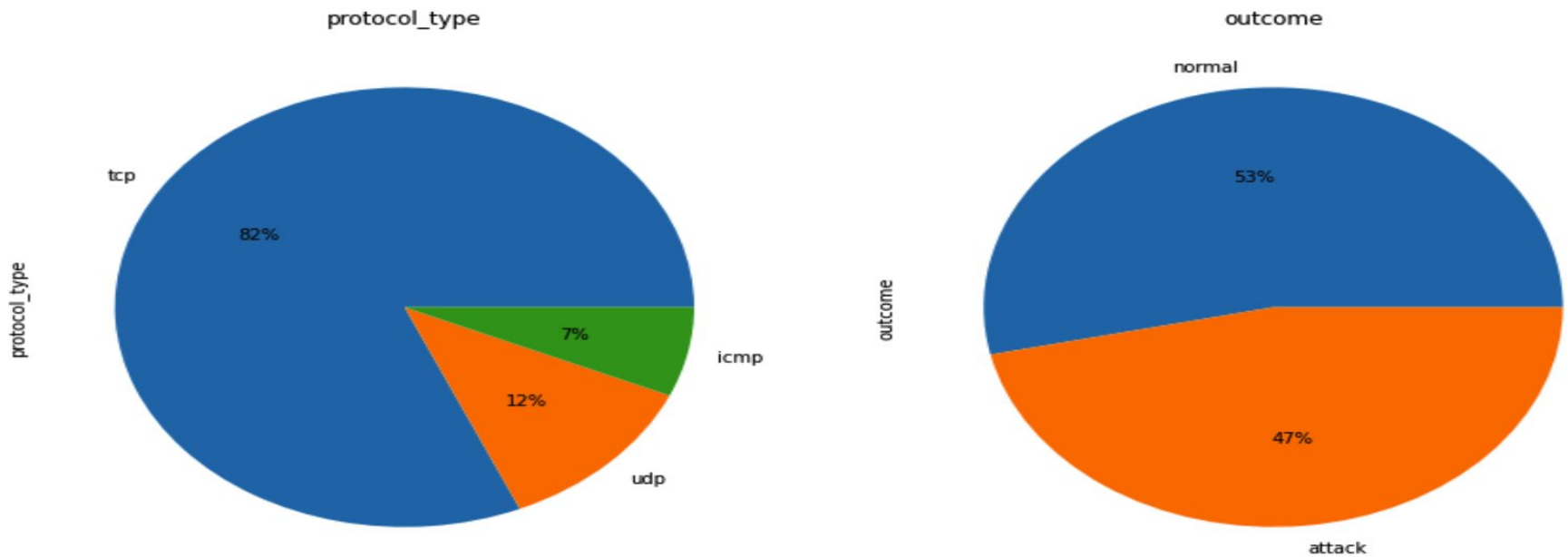
Out[7]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins
0	0	udp	other	SF	146	0	0	0	0	0	0
1	0	tcp	private	S0	0	0	0	0	0	0	0
2	0	tcp	http	SF	232	8153	0	0	0	0	0
3	0	tcp	http	SF	199	420	0	0	0	0	0
4	0	tcp	private	REJ	0	0	0	0	0	0	0

In [10]:
```
data_train.loc[data_train['outcome'] == "normal", "outcome"] = 'normal'
data_train.loc[data_train['outcome'] != 'normal', "outcome"] = 'attack'
```

In [11]:
```
def pie_plot(df, cols_list, rows, cols):
    fig, axes = plt.subplots(rows, cols)
    for ax, col in zip(axes.ravel(), cols_list):
        df[col].value_counts().plot(ax=ax, kind='pie', figsize=(15, 15), fontsize=10, autopc
t='%1.0f%%')
        ax.set_title(str(col), fontsize = 12)
    plt.show()
```

In [12]:
```
pie_plot(data_train, ['protocol_type', 'outcome'], 1, 2)
```



Preprocessing the data

Indian Institute of Information Technology Kottayam

[13]:
```
def Scaling(df_num, cols):
    std_scaler = RobustScaler()
    std_scaler_temp = std_scaler.fit_transform(df_num)
    std_df = pd.DataFrame(std_scaler_temp, columns =cols)
    return std_df
```

[14]:
```
cat_cols = ['is_host_login','protocol_type','service','flag','land', 'logged_in','is_guest_l
ogin', 'level', 'outcome']
def preprocess(dataframe):
    df_num = dataframe.drop(cat_cols, axis=1)
    num_cols = df_num.columns
    scaled_df = Scaling(df_num, num_cols)

    dataframe.drop(labels=num_cols, axis="columns", inplace=True)
    dataframe[num_cols] = scaled_df[num_cols]

    dataframe.loc[dataframe['outcome'] == "normal", "outcome"] = 0
    dataframe.loc[dataframe['outcome'] != 0, "outcome"] = 1

    dataframe = pd.get_dummies(dataframe, columns = ['protocol_type', 'service', 'flag'])
    return dataframe
```

[15]:
```
scaled_train = preprocess(data_train)
```

Advantages of PCA

There are two main advantages of dimensionality reduction with PCA.

- The training time of the algorithms reduces significantly with less number of features.
- It is not always possible to analyze data in high dimensions. For instance if there are 100 features in a dataset. Total number of scatter plots required to visualize the data would be 100(100-1)2 = 4950. Practically it is not possible to analyze data this way.

```
x = scaled_train.drop(['outcome', 'level'] , axis = 1).values
y = scaled_train['outcome'].values
y_reg = scaled_train['level'].values

pca = PCA(n_components=20)
pca = pca.fit(x)
x_reduced = pca.transform(x)
print("Number of original features is {} and of reduced features is {}".format(x.shape[1], x_reduced.sh

y = y.astype('int')
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
x_train_reduced, x_test_reduced, y_train_reduced, y_test_reduced = train_test_split(x_reduced, y, test_
x_train_reg, x_test_reg, y_train_reg, y_test_reg = train_test_split(x, y_reg, test_size=0.2, random_sta
```

```
Number of original features is 122 and of reduced features is 20
```

Modeling

The process of modeling means training a machine learning algorithm to predict the labels from the features, tuning it for the business need, and validating it on holdout data. The output from modeling is a trained model that can be used for inference, making predictions on new data points.

A machine learning model itself is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data. For example, let's say you want to build an application that can recognize a user's emotions based on their facial expressions. You can train a model by providing it with images of faces that are each tagged with a certain emotion, and then you can use that model in an application that can recognize any user's emotion

Model Evaluation Function

Indian Institute of Information Technology Kottayam

```
kernal_evals = dict()
def evaluate_classification(model, name, X_train, X_test, y_train, y_test):
    train_accuracy = metrics.accuracy_score(y_train, model.predict(X_train))
    test_accuracy = metrics.accuracy_score(y_test, model.predict(X_test))

    train_precision = metrics.precision_score(y_train, model.predict(X_train))
    test_precision = metrics.precision_score(y_test, model.predict(X_test))

    train_recall = metrics.recall_score(y_train, model.predict(X_train))
    test_recall = metrics.recall_score(y_test, model.predict(X_test))

    kernal_evals[str(name)] = [train_accuracy, test_accuracy, train_precision, test_precision, train_recall, test_recall]
    print("Training Accuracy " + str(name) + " {}  Test Accuracy ".format(train_accuracy*100) + str(name) + " {}".format(test_accuracy*1
    print("Training Precesion " + str(name) + " {}  Test Precesion ".format(train_precision*100) + str(name) + " {}".format(test_precisi
    print("Training Recall " + str(name) + " {}  Test Recall ".format(train_recall*100) + str(name) + " {}".format(test_recall*100))

    actual = y_test
    predicted = model.predict(X_test)
    confusion_matrix = metrics.confusion_matrix(actual, predicted)
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = ['normal', 'attack'])

    fig, ax = plt.subplots(figsize=(10,10))
    ax.grid(False)
    cm_display.plot(ax=ax)
```

Logistic Regression

```
lr = LogisticRegression().fit(x_train, y_train)
evaluate_classification(lr, "Logistic Regression", x_train, x_test, y_train, y_test)
```

k-nearest neighbors

```
knn = KNeighborsClassifier(n_neighbors=20).fit(x_train, y_train)
evaluate_classification(knn, "KNeighborsClassifier", x_train, x_test, y_train, y_test)
```

Naive Bayes

```
gnb = GaussianNB().fit(x_train, y_train)
evaluate_classification(gnb, "GaussianNB", x_train, x_test, y_train, y_test)
```

Support Vector Machines

```
lin_svc = svm.LinearSVC().fit(x_train, y_train)
evaluate_classification(lin_svc, "Linear SVC(LBasedImpl)", x_train, x_test, y_train, y_test)
```

Decision Tree

```
dt = DecisionTreeClassifier(max_depth=3).fit(x_train, y_train)
tdt = DecisionTreeClassifier().fit(x_train, y_train)
evaluate_classification(tdt, "DecisionTreeClassifier", x_train, x_test, y_train, y_test)
```

Random Forest

```
rf = RandomForestClassifier().fit(x_train, y_train)
evaluate_classification(rf, "RandomForestClassifier", x_train, x_test, y_train, y_test)
```

Measuring effect of PCA

Indian Institute of Information Technology Kottayam

```
rrf = RandomForestClassifier().fit(x_train_reduced, y_train_reduced)
evaluate_classification(rrf, "PCA RandomForest", x_train_reduced, x_test_reduced, y_train_reduced, y_te
```

Neural networks

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google’s search algorithm.

[44]:

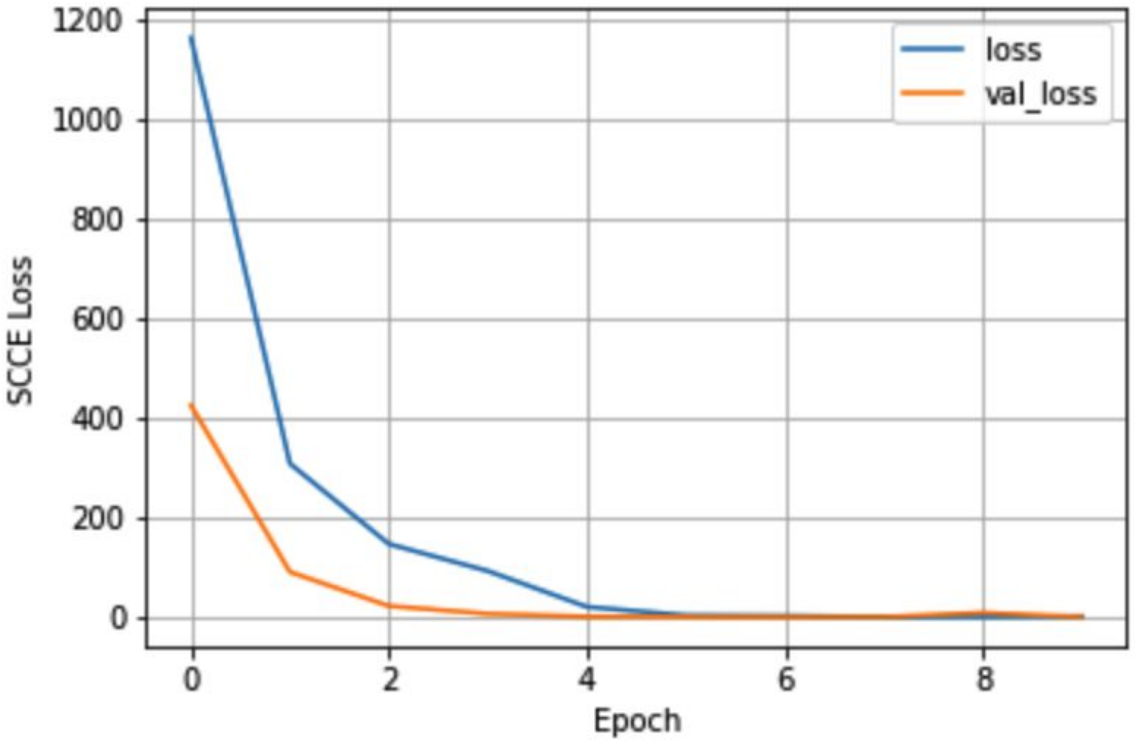
```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=64, activation='relu', input_shape=(x_train.shape[1:]),
                          kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
                          bias_regularizer=regularizers.L2(1e-4),
                          activity_regularizer=regularizers.L2(1e-5)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(units=128, activation='relu',
                          kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
                          bias_regularizer=regularizers.L2(1e-4),
                          activity_regularizer=regularizers.L2(1e-5)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(units=512, activation='relu',
                          kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
                          bias_regularizer=regularizers.L2(1e-4),
                          activity_regularizer=regularizers.L2(1e-5)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(units=128, activation='relu',
                          kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
                          bias_regularizer=regularizers.L2(1e-4),
                          activity_regularizer=regularizers.L2(1e-5)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(units=1, activation='sigmoid'),
])
```

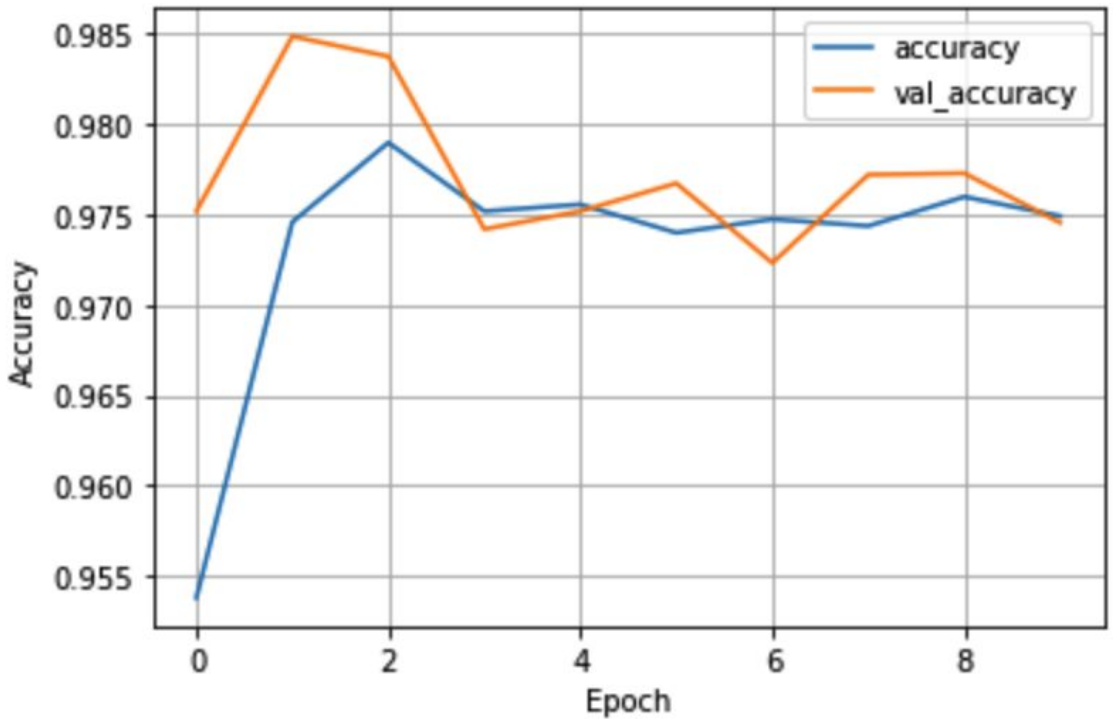
```
model.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(from_logits=True), metrics=['accuracy'])
```

```
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, verbose=1)
```

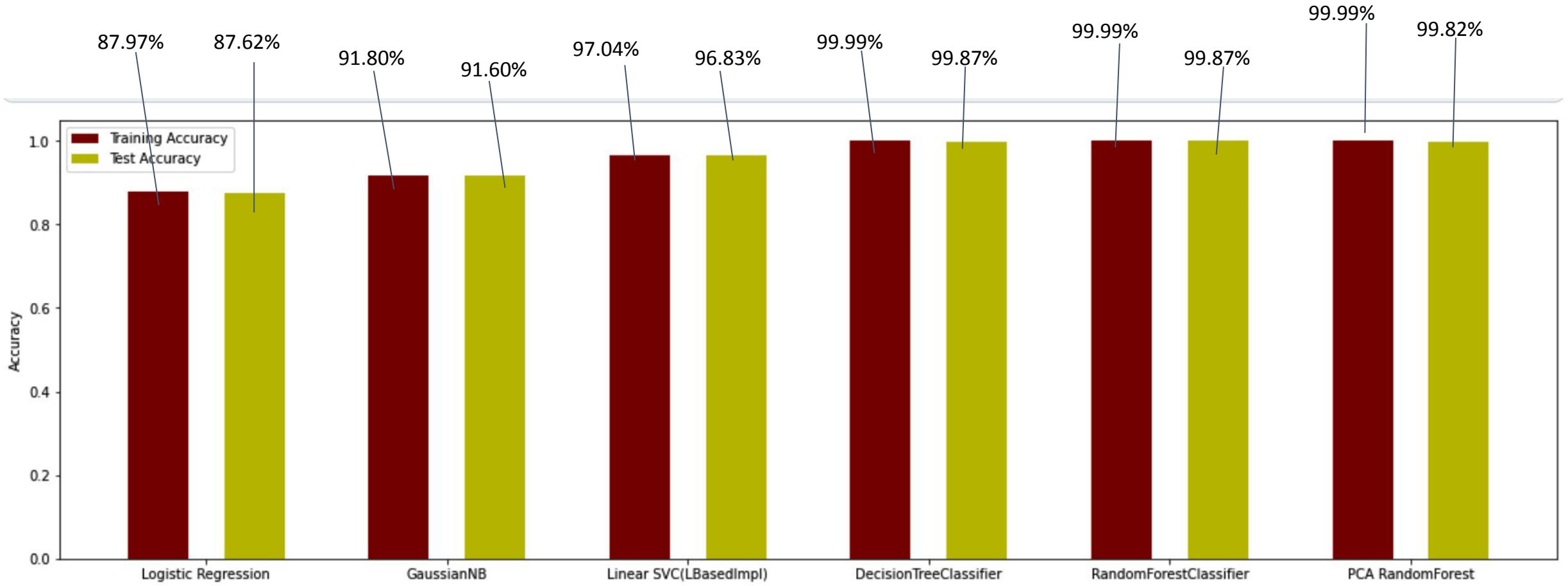
```
Epoch 1/10
3150/3150 [==============================] - 20s 6ms/step - loss: 1163.1600 - accuracy: 0.9538 - val_loss: 425.0167 - val_acc
uracy: 0.9752
Epoch 2/10
3150/3150 [==============================] - 17s 5ms/step - loss: 307.9906 - accuracy: 0.9746 - val_loss: 90.3703 - val_accur
acy: 0.9849
Epoch 3/10
3150/3150 [==============================] - 17s 5ms/step - loss: 146.5123 - accuracy: 0.9790 - val_loss: 22.0090 - val_accur
acy: 0.9838
Epoch 4/10
3150/3150 [==============================] - 18s 6ms/step - loss: 92.3532 - accuracy: 0.9752 - val_loss: 6.3909 - val_accurac
y: 0.9742
Epoch 5/10
3150/3150 [==============================] - 17s 5ms/step - loss: 20.2885 - accuracy: 0.9756 - val_loss: 0.8062 - val_accurac
y: 0.9752
Epoch 6/10
3150/3150 [==============================] - 18s 6ms/step - loss: 3.4544 - accuracy: 0.9740 - val_loss: 0.4149 - val_accurac
y: 0.9767
Epoch 7/10
3150/3150 [==============================] - 18s 6ms/step - loss: 2.3672 - accuracy: 0.9748 - val_loss: 0.1070 - val_accurac
y: 0.9723
Epoch 8/10
3150/3150 [==============================] - 18s 6ms/step - loss: 0.1926 - accuracy: 0.9744 - val_loss: 0.0856 - val_accurac
y: 0.9772
Epoch 9/10
3150/3150 [==============================] - 18s 6ms/step - loss: 0.0963 - accuracy: 0.9760 - val_loss: 8.5606 - val_accurac
y: 0.9773
Epoch 10/10
3150/3150 [==============================] - 18s 6ms/step - loss: 1.6946 - accuracy: 0.9749 - val_loss: 0.0830 - val_accurac
y: 0.9746
```

Indian Institute of Information Technology Kottayam





Indian Institute of Information Technology Kottayam



Conclusion

This sets the stage for increased adoption of ML solutions to safeguard present and Future systems of this Cyber World

Fine-Tuning Hyperparameters : We will conduct further experiments to optimize hyperparameters, ensuring that each model is performing at its best.

Incorporating Real-Time Learning : We will explore methods for real-time learning and adaptation to swiftly respond to emerging threats.

Threat Intelligence Integration : The integration of real-time threat intelligence feeds will be a priority, enabling our NIDS to remain up-to-date with the latest threat indicators.

Privacy-Preserving Techniques : We will investigate privacy-preserving methods to protect sensitive data while maintaining effective intrusion detection

References

[1] Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A.V. and Di Franco, F., 2023. The role of machine learning in cybersecurity. *Digital Threats: Research and Practice*, *4*(1), pp.1-38.

[2] Lu, C., 2022, September. Research on the technical application of artificial intelligence in network intrusion detection system. In *2022 International Conference on Electronics and Devices, Computational Science (ICEDCS)* (pp. 109-112). IEEE.

[3] Zhang, Z., Al Hamadi, H., Damiani, E., Yeun, C.Y. and Taher, F., 2022. Explainable artificial intelligence applications in cyber security: State-of-the-art in research. *IEEE Access*.

[4] Azam, Z., Islam, M.M. and Huda, M.N., 2023. Comparative Analysis of Intrusion Detection Systems and Machine Learning Based Model Analysis through Decision Tree. *IEEE Access*.

[5] Sarker, I.H., Abushark, Y.B., Alsolami, F. and Khan, A.I., 2020. Intrudtree: a machine learning based cyber security intrusion detection model. *Symmetry*, *12*(5), p.754.

[6] Yang, Y., Li, H. and Jing, D., 2022, September. A Phishing Website Detection Method Based on Multi-layer Perceptron with Mutual Information Feature Selection. In *2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC)* (pp. 117-123). IEEE.

[7] F. Türk , "Analysis of Intrusion Detection Systems in UNSW-NB15 and NSL-KDD Datasets with Machine Learning Algorithms", *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 12, no. 2, pp. 465-477, Jun. 2023, doi:10.17798/bitlisfen.1240469. IEEE

Thank You