NETWORK ATTACK DETECTION USING MACHINE LEARNING

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

**Bachelor of Technology**

***in***

**Computer Science Engineering**

*Submitted by*

Sonal Mishra :(Roll No. 200102191)

Adil Khan :(Roll No. 200102198)

Lanka Sreekar: (Roll No. 200102233)

## ***Under the Supervision of***

Dr. Bharti Sharma

Associate Professor

Head-BCA



Computer Science Engineering

2020-2024

**Declaration**

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed. The plagiarism check report is attached at the end of this document.

Name of the Student: Sonal Mishra Signature and Date:  06/05/24



Name of the Student: Adil Khan Signature and Date: 06/05/24



Name of the Student: Lanka Sreekar Signature and Date: 06/05/24

**Abstract**

In the digital age, the importance of cybersecurity cannot be overstated as organizations are continually exposed to a myriad of cyber threats. Network attack detection systems play a pivotal role in this scenario, analysing network traffic data to pinpoint potential threats. However, traditional methods often fail to adapt to the evolving strategies of cyber adversaries. This research addresses this issue by applying machine learning (ML) techniques to augment the capabilities of network attack detection.

The project initiates with the gathering of extensive datasets that encapsulate a wide array of network traffic parameters. A thorough data preprocessing phase is undertaken to prepare the dataset for ML model training. Advanced feature selection techniques, such as correlation analysis and the Random Forest Classifier, are utilized to identify key features for training the ML models.

Through continuous refinement and experimentation, the project aims to develop a versatile network attack detection system that can effectively counter cyber threats in real-world scenarios.

**Table Of Contents**

1. Introduction
2. Literature Review
3. Dataset Description
4. Data Preprocessing
5. Feature Selection And Working Block Diagram
6. Models
7. Results Discussion
8. Future Scope
9. References

**1. Introduction**

In the contemporary landscape of cybersecurity, organizations face a relentless onslaught of cyber threats within their interconnected digital environments. To fortify their defenses, network attack detection systems stand as indispensable tools, employing sophisticated machine learning (ML) algorithms to meticulously scrutinize network traffic data for signs of potential intrusion. Yet, the ever-evolving nature of cyber threats renders traditional approaches inadequate, necessitating a paradigm shift towards ML methodologies.

This project endeavors to harness the transformative potential of ML algorithms in bolstering cybersecurity efforts, specifically focusing on the development of an advanced network attack detection system. The primary objective is to engineer a robust and adaptive solution capable of discerning between benign network activity and malicious intrusions with a high degree of accuracy and efficiency. By leveraging comprehensive datasets containing a plethora of network traffic parameters, including protocol types, services, flags, and file sizes, we aim to train ML models capable of autonomously adapting to emerging threats.

The project's journey commences with a meticulous data preprocessing phase, wherein missing values, outliers, and categorical data are rigorously addressed to ensure the dataset’s integrity and readiness for model training. Employing advanced feature selection techniques such as correlation analysis and the Random Forest Classifier, we endeavor to identify and prioritize critical features essential for accurate model performance.

A diverse array of ML algorithms, spanning logistic regression, support vector machines (SVM), k-nearest neighbors (KNN), and decision trees, are methodically trained and rigorously evaluated using the prepared dataset. Model performance is comprehensively assessed using a suite of metrics, including accuracy, precision, recall, and F1 score, to ascertain their efficacy in detecting and mitigating network attacks.

Through iterative refinement and meticulous experimentation, this project aspires to deliver a state-of-the-art network attack detection system capable of proactively defending against cyber threats in real time. With a steadfast commitment to excellence and innovation, we aim to empower organizations with a formidable defense mechanism, safeguarding their digital assets against the relentless tide of cyber-attacks.

**2. Literature Review**

**3. Dataset Description**

Dataset gathered from Kaggle([Link](https://www.kaggle.com/code/pakinkitti/network-intrusion-detection/input)) which consists of network connection-related parameters like protocol, service, flag, file size of source and destination, etc. In dataset classification given in Normal or Anomaly mode of Attack. In this, we have a dimension of (25192,42) training and (22544,41) testing datasets (referred to in Table 3.1) that are used for training as well as testing of different ML Models. Initially, there is an imbalance in our training dataset (referred to in Table 3.2) due to which training of models gets biased towards normal results. So to remove this biased data gets balanced before performing model training. This helps to achieve the best model accuracy.

Important features are required to train the models on different corresponding values (referred to in Table 3.3), which depend upon the nature of the dataset. Such features help to maintain the accuracy of Prediction Models. Such tables are used here to compare the different features of the dataset and to establish a relation between them.

|  |  |  |
| --- | --- | --- |
| DATASET | ROWS | COLUMNS |
| Training | 25192 | 42 |
| Testing | 22544 | 41 |

(Table-3.1 Training and Testing

Data Shape)

|  |  |  |
| --- | --- | --- |
| DATA | NORMAL | ANOMALY |
| Initial | 13449 | 11743 |
| Final | 13449 | 13449 |

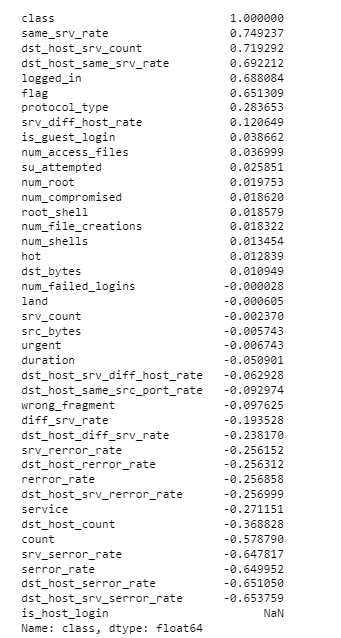
(Table-3.2 Class Labels of Dataset

Before and After Data Balancing)

|  |  |
| --- | --- |
| FEATURES | DATATYPE |
| protocol\_type | Object |
| service | Object |
| flag | Object |
| src\_bytes | Int64 |
| dst\_bytes | Int64 |
| count | Int64 |
| same\_srv\_rate | Float64 |
| diff\_srv\_rate | Float64 |
| dst\_host\_srv\_count | Int64 |
| dst\_host\_same\_srv\_rate | Float64 |

(Table-3.3 Important Features)

In the dataset to explore the relationship between variables correlation is applied which is a statistical measure to describe the strength and direction of the relationship between variables. Here considering class as the target variable correlation of different columns in the dataset shown in Table 3.4. Using this result feature value with respect to the target variable can be assumed.



(Table-3.4 Correlation between Columns)

**4. Data Preprocessing**

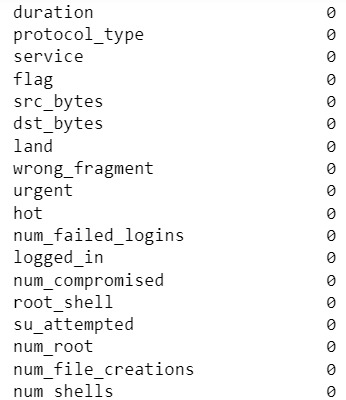
Data Preprocessing is a crucial step in data analysis and machine learning that involves

transformation of raw data into a clean and organized form that is suitable for analysis .

Usually it gets performed in many steps like data cleaning , data transformation etc.

After gathering of data from data source here following preprocessing steps are formed:

1. Missing Values and Outler Detection



Initially, in datasets, there are no missing values or duplicate values. Usually, if any missing values are there then such values can be replaced with mean/median/mode.

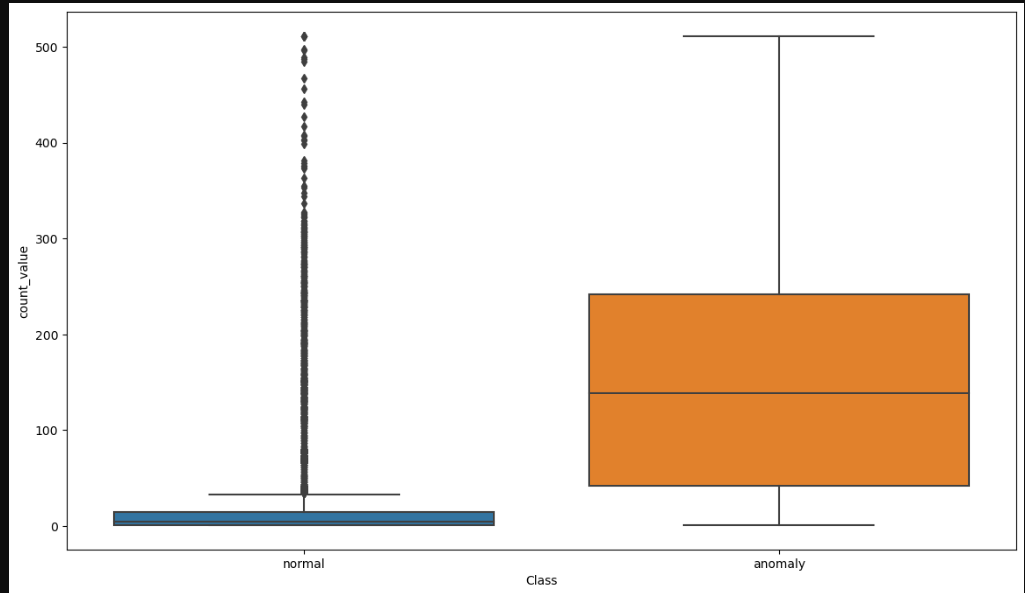
It helps to remove inconsistencies from the datasets.

A screenshot of a computer program

Description automatically generated

(Data-4.1 No Of Missing Values In Columns)

Outlier Detection



Outlier Detection also known as anomaly detection , is the process of identifying data points or observations that deviate from the rest of the dataset. It can be caused by many factors like Mathematical error or Experimental Error. Handling outliers is very important in data analysis and machine learning to ensure that they do not influence the results of the models.

Referred to Data-4.2 shows a boxplot visual of Class and Count Value.

Here to detect outlier boxplot is used to evaluate the Medians(Q2) , Quartiles(Q1,Q3) ,whiskers And outliers.

Data-4.2 Boxplot Between class vs Count\_value

1. Data Encoding

|  |  |
| --- | --- |
| Categorical Data | Assigned Integer Value |
| TCP | 1 |
| UDP | 2 |
| ICMP | 0 |
| Normal | 0 |
| Anomaly | 1 |

Data Encoding usually refers to the process of converting categorical data into numerical format that Can be used for analysis or input into machine learning algorithms. Categorical data represents categories or groups and can take on a limited, fixed number of values.

Data Encoding is implemented in many ways like Label Encoder, One hot Encoding, and Binary Encoding.

Referred to Table 4.3 shows some of the Categorical values Encoded to Integer Values.

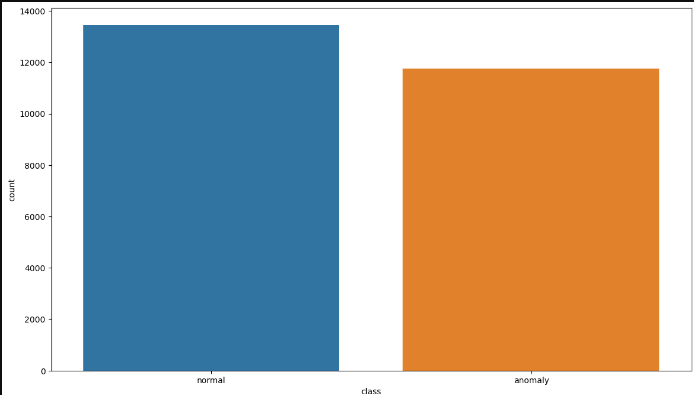
Based on the nature of the gathered datasets here Label Encoder is applied which assigns a unique integer to each Category in the dataset.

Table-4.3 Encoded Values

3. Data Imbalance Handling

Data imbalance refers to a situation where the distribution of classes within a dataset is not uniform, resulting in one or more classes being significantly more prevalent than others. It can occur in various types of datasets, mostly including those which is used for classification.

Here Resampling technique is used with random under-sampling and Synthetic Minority Over Sampling (SMOTE).

 (Data-4.4 Class Label Initial)

A blue and orange squares

Description automatically generated

Random under-sampling is a technique used to address class imbalance in a dataset by reducing the number of instances in the majority class to balance it with the minority class.

SMOTE- Synthetic Minority Oversampling used to address the class imbalance in the dataset. It works by generating synthetic samples for minority classes which effectively balance the class distribution.

(Data-4.5 Class Label After Data Balancing)

Referred to Table 3.2 shows the class label count in datasets before and after balancing . While

evaluating the dataset Class column has imbalanced data which is the Classification result outcome

column(rereferred to Data-4.4) having maximum count Value and (referred to as Data-4.5) shows

equal Count of Normal or Anomaly.

**5. Feature Selection**

Feature Selection is a process of machine learning where it can choose a subset of relevant features

to use in building a model. It helps to improve model performance by reducing overfitting and

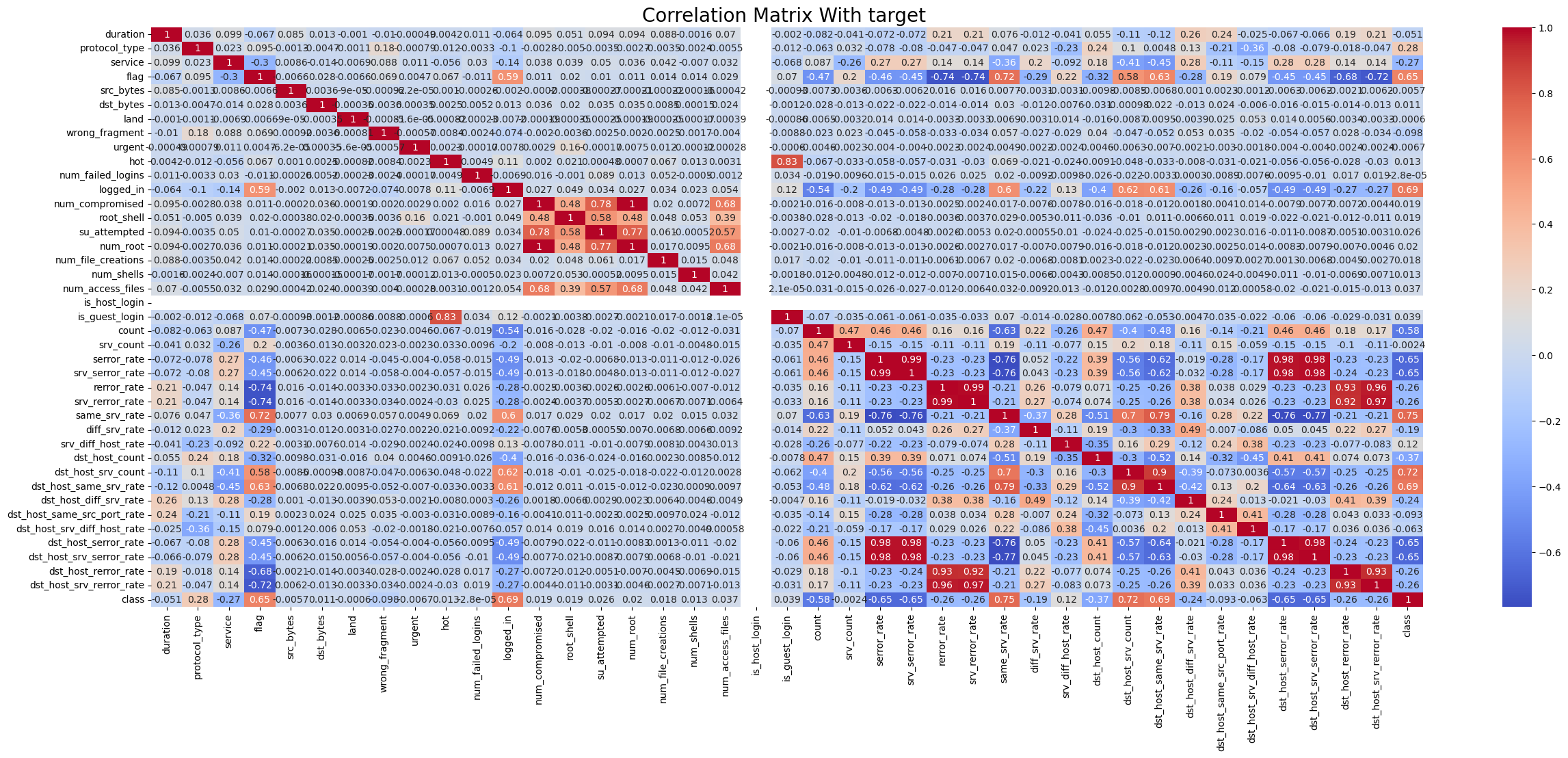
decreasing computational cost.

According to the nature of the gathered dataset Random Forest Classifier is implemented to extract important features for training and testing of Machine Learning Models.

The important features are listed in Table-1.3 and include protocol\_type, service, flag, src\_bytes,

dst\_bytes, count, same\_srv\_rate, diff\_srv\_rate, dst\_host\_srv\_count, and dst\_host\_same\_srv\_rate.

Feature selection is considered one of the primary tasks after data preprocessing. The Accuracy of Machine Learning Models depends upon the features used during the training and testing phase. The selection of important parameters is performed using a Random Forest Classifier Algorithm in three phases. In the first and second phases value of the feature to select is ten and seven, at the last phase its value becomes five. With the change in the number of features training and testing accuracy of different models changes(Referred to Table-5.1,5.2,5.3). With a Change in accuracy, the Precision, Recall, and F1 Score of every model gets changed which can be considered as a criterion to select best efficient model from all applied models.



(Data-5.0 Correlation Matrix With Target(Class)

* Accuracy of ML Models With Different Sample of Features

1. With All Features

|  |  |  |
| --- | --- | --- |
| ML MODEL | TRAINING ACCURACY | TESTING ACCURACY |
| Logistic Regression | 0.91826 | 0.917596 |
| Support Vector Machine | 0.901901 | 0.905328 |
| K Nearest Neighbours | 0.994211 | 0.991698 |
| Decision Tree | 0.999947 | 0.995043 |

Table-5.1

1. With 7 Features [protocol\_type , flag , src\_bytes ,dst\_bytes,count,same\_srv\_rate,dst\_host\_srv\_count]

|  |  |  |
| --- | --- | --- |
| ML MODEL | TRAINING ACCURACY | TESTING ACCURACY |
| Logistic Regression | 0.906416 | 0.905948 |
| Support Vector Machine | 0.868122 | 0.8772 |
| K Nearest Neighbours | 0.992564 | 0.988352 |
| Decision Tree | 0.999734 | 0.993556 |

Table-5.2

1. With 5 features [flag,src\_bytes,dst\_bytes,same\_srv\_rate,dst\_host\_same\_srv\_rate]

|  |  |  |
| --- | --- | --- |
| ML MODEL | TRAINING ACCURACY | TESTING ACCURACY |
| Logistic Regression | 0.863501 | 0.865304 |
| Support Vector Machine | 0.934846 | 0.931227 |
| K Nearest Neighbours | 0.994636 | 0.9943 |
| Decision Tree | 0.998672 | 0.992565 |

Table-5.3

The training and testing accuracy obtained using 30% data for testing and 70% for training. Using the above data with random state 42 ensures data splitting is reproducible to a specific value. Considering all features gives the best accuracy and every model performance gets improved.

* **Working Block Diagram**

Block Diagram gives the brief working of the complete project in simple steps which start from Data collection to result in outcome using different machine learning algorithms(referred to in fig-5.4). At result outcome is normal or anomaly classification of network attack.

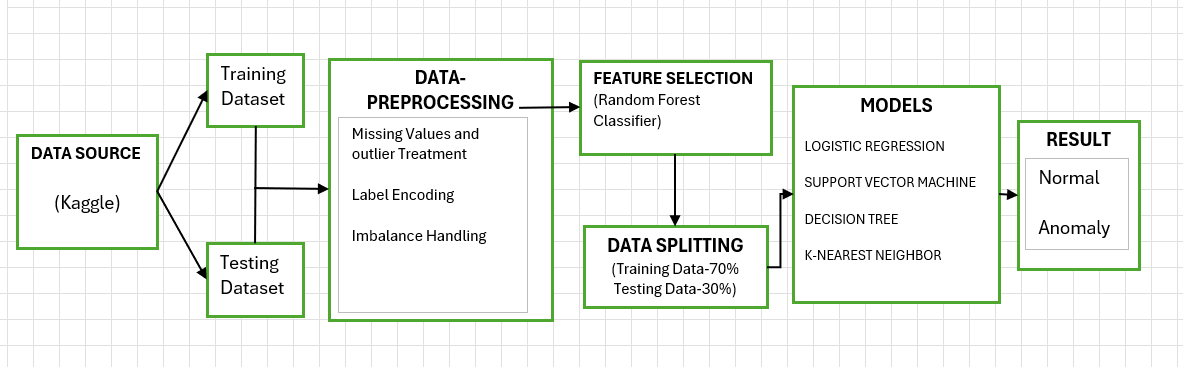


Fig-5.4 Block Diagram For Proposed Work Flow

**6. Models**

* Data Set Splitting

Using the Scikit-learn library dataset gets split into training and testing subsets which is crucial for evaluating the performance of machine learning models and ensuring that

models generalize well to unseen data.

It consists of subparts: Test size and Random State

While performing data splitting 70% of the dataset used for training the models and 30% Dataset used for testing of models.

Random State 42 gives reproducibility or consistent splitting. Reproducibility ensures that when we run the code with the same dataset and parameters will give the same output.

Consistent splitting ensures that data will be split in the same way whenever We run the code. This helps with debugging and reproducing results.

* Machine Learning Models
* Logistic Regression

Logistic Regression is mostly used for binary classification tasks where the primary goal is to predict the probability of a binary outcome(whether it is normal or Anomaly). It uses the logistic function (also called the sigmoid function) to model the relationship between target variables and the probability of the outcome.

The logistic function is defined as:

​

e denotes natural base algorithm.

Working Of Logistic Regression Model

1. Import the necessary libraries and load the training and testing dataset.
2. Apply data preprocessing and convert categorical variable into numeric using label encoder.
3. Initialize a Logistic Regression Classifier with a maximum number of iterations.
4. Train the Logistic Regression Classifier on the training data and Use the trained Logistic Regression Classifier to make predictions on the testing data.
5. Use the trained Logistic Regression Classifier to make predictions on the training data and calculate the accuracy.

In this model the iteration value used is 100 refers to the maximum number of iterations or steps that an iterative optimization algorithm, such as gradient descent, will perform during the training process.

(Referred To Table-6.0.0) shows the Training and Testing Accuracy of the Logistic Regression Model and (referred to Table 6.0.1) reflects the classification report using the Confusion Matrix.

|  |  |
| --- | --- |
| Training Accuracy | Testing Accuracy |
| 0.91826 | 0.0917596 |

(Table-6.0.0 Accuracy Of Logistic Regression Model)

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1 Score |
| Normal | 0.88 | 0.86 | 0.87 |
| Anomaly | 0.86 | 0.89 | 0.88 |

(Table-6.0.1 Classification Evaluation)

* Support Vector Machine

SVM is a powerful supervised learning algorithm that is used for classification, regression, and outlier detection. It works effectively in scenarios where data is not easily separable by a straight line or plane in its feature space. Its primary objective is to find a hyperplane (a decision boundary) that separates different classes in high-dimensional space. Data points closest to the hyperplane are called support vectors.

It works on the principle of :

a.x+b=0 , Where

a is the vector normal to the hyperplane

b is the offset

If the value of a.x+b>0

we can say it’s a positive point otherwise its a negative point.

In this model, Linear is used as kernel to perform linear classification and Linear kernel gives efficient results for large-scale data.

C generally denotes the regularization parameter in SVM. A smaller C value encourages a wider margin but may allow some misclassifications, while a larger C value seeks to classify all training examples correctly but may lead to a narrower margin. In this model, C is set to 2.5, indicating a moderate regularization strength.

Working Of SVM Model

1. Import necessary libraries and load the dataset.
2. Preprocess the data by converting categorical variables to numerical using label encoding and splitting the dataset into training and testing sets.
3. Initialize a Linear SVM Classifier with less C value(Parameter).
4. Train the SVM Classifier on the training data.
5. Using the trained SVM Classifier to make predictions on the training data .
6. Use the trained SVM Classifier to make predictions on the testing data and calculate the accuracy.

(Referred To Table-6.1.0) shows the Training and Testing Accuracy of the Support Vector Machine Model with linear classification and (referred to Table 6.1.1) reflects the classification report using the Confusion Matrix (Precision , Recall , F1 Score)

|  |  |
| --- | --- |
| Training Accuracy | Testing Accuracy |
| 0.869609 | 0.876084 |

(Table-6.1.0 Accuracy of SVM Model)

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1 Score |
| Normal | 0.94 | 0.91 | 0.93 |
| Anomaly | 0.92 | 0.94 | 0.93 |

(Table-6.1.1 Classification Evaluation)

* K-Nearest Neighbors

KNN is an effective algorithm for both classification and regression tasks in machine learning. It is nonparametric, meaning it doesn’t make any assumptions about the underlying data distribution and doesn’t learn the model during the training phase. Instead, it stores all available training data points and makes predictions based on the similarity of new data points.

This model is directly implemented using the Sklearn module. Here number of neighbors is taken as input for making a prediction. This is a hyperparameter we can tune based on the nature of your data and the problem at hand. A smaller Value of K means the model is more sensitive to noise and local variations, while a large value of K shows out the decision boundary but may overlook important patterns.

In this Euclidean distance is calculated between new data points and all other points,

Based on that data points are getting classified.

Referred Table 6.2.0 shows the training and testing accuracy of the KNN model with 5 number of neighbors.

|  |  |
| --- | --- |
| Training Accuracy | Testing Accuracy |
| 0.994211 | 0.991698 |

(Table-6.2.0 Accuracy of KNN Model)

Working Of KNN Model

1. Import the necessary libraries and load the dataset . Apply Preprocessing Steps.
2. Initialize a KNN Classifier with number of neighbors(n).
3. Train the KNN Classifier on the training data.
4. Use the trained KNN Classifier to make predictions on the testing data.
5. Use the trained KNN Classifier to make predictions on the training data.
6. Calculate the accuracy of the KNN Classifier .

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1 Score |
| Normal | 0.99 | 0.99 | 0.99 |
| Anomaly | 0.99 | 0.99 | 0.99 |

The complete evaluation of KNN classification is described using a confusion matrix which gives accuracy, precision, recall, and F1 score(referred to Table-6.2.1).

(Table-6.2.1 Classification Evaluation)

* Decision Tree

A Decision Tree is a popular machine-learning algorithm used for both classification and regression tasks. It's a predictive modelling tool that recursively partitions the data into subsets based on features that best separate the target variable. At each node of the tree, the decision is made based on feature value leading to a split in data. This splitting continues until the stopping criteria(Maximum Depth=3) is reached.

The factor on which the accuracy of the decision tree depends is the depth of the tree. An increase in maximum depth makes the mode more complex and prone to overfitting. Overfitting occurs when the model captures noise in training data rather than the underlying relationship, leading to poor generalization performance for unseen data.

Decision Trees are easy to understand and interpret. It can handle both numerical and categorical data and automatically handle missing values. It implicitly performs feature selection by selecting the most informative features for splitting.

Working Of Decision Tree Model

1. Import the necessary libraries and load the dataset.
2. Preprocess the data by converting categorical variables to numerical using Label encoding and splitting the dataset into training and testing sets.
3. Initialize a Decision Tree Classifier: Decision\_tree = DecisionTreeClassifier().
4. Train the Decision Tree Classifier on the training data: Decision\_tree.fit(x\_train, y\_train)
5. Use the trained Decision Tree Classifier to make predictions on the training data: y\_train\_pred = Decision\_tree.predict(x\_train).
6. Use the trained Decision Tree Classifier to make predictions on the testing data: y\_pred = Decision\_tree.predict(x\_test)
7. Calculate the accuracy of the Decision Tree Classifier

|  |  |
| --- | --- |
| Training Accuracy | Testing Accuracy |
| 0.999947 | 0.994796 |

The decision Tree has more training and Testing Accuracy than other models (referred to Table 6.3.0).

(Table-6.3.0 Accuracy of KNN Model)

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1 Score |
| Normal | 1.00 | 0.99 | 0.99 |
| Anomaly | 0.99 | 1.00 | 0.99 |

With depth value 3 classification report of the Decision Tree gives enhanced value for precision, recall, and F1 Score (referred to Table-6.3.1).

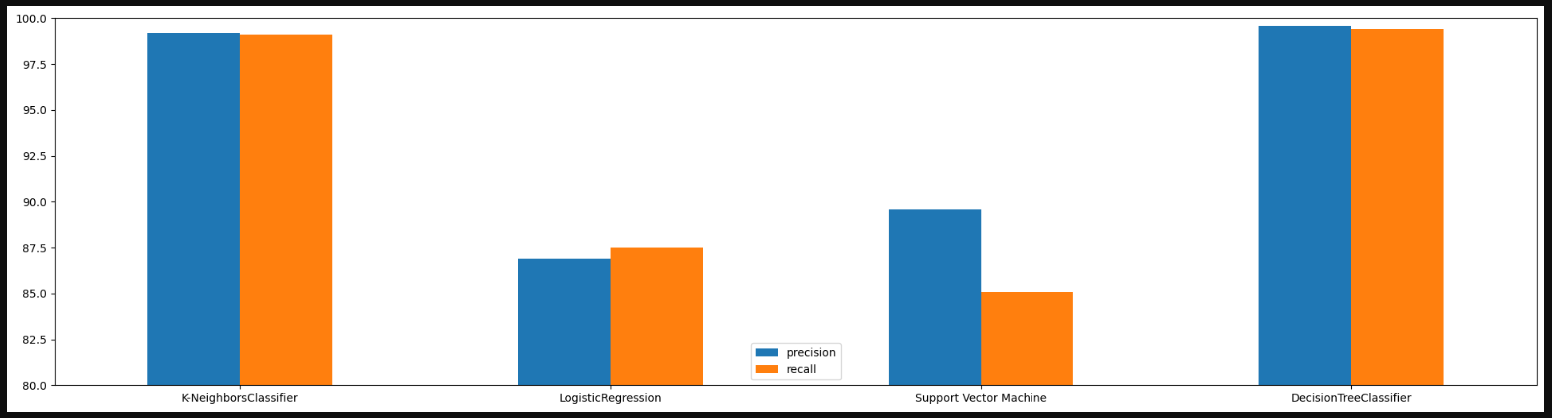
(Table-6.3.1 Classification Evaluation)

**7. Result Discussion**

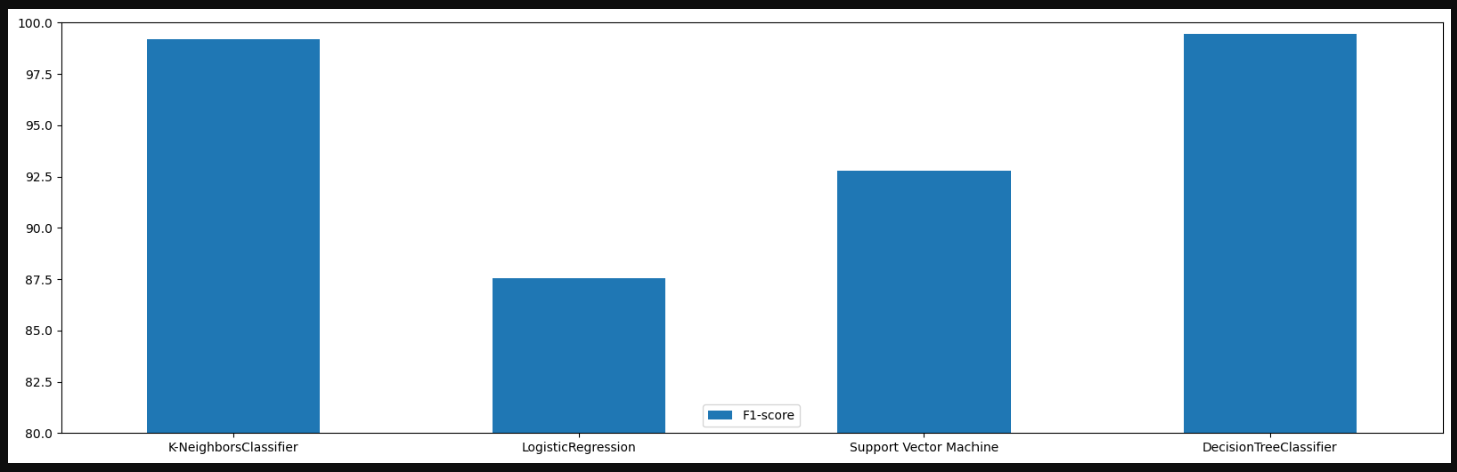
The evaluation of classification implemented using Machine Learning models(KNN, SVM, Logistic Regression, and Decision Tree) can be measured using a confusion matrix which gives accuracy, precision, recall, and F1 score. Based on the above matrix value models performance can be evaluated.

Referred to in Fig-7.0 shows the precision and recall of different models using training and testing data. The accuracy of every model is already shown in the model section which depends on the nature of the dataset used to train the model. Positive prediction accuracy is given by precision and Recall used to measure the ability of the model to correctly identify positive instances. Here KNN and Decision Tree have high precision and recall among all models and that also has high accuracy (referred to model section).

F1 Score is used to provide a balance measure to the model’s performance, and it combines precision and recall into a single value. It gives an overall understanding of model performance(referred to in Fig-7.1).



(Fig-7.0 Precision and Recall on different models)



(Fig-7.1 F1 Score of different models)

**8.Future Scope**

**9. Reference**