SHAHEED ZULFIKAR ALI BHUTTO
INSTITUTE OF SCIENCE AND TECHNOLOGY

Final Year Project Report

Detection And Prevention Of Machine Learning Attacks in Adversarial Setting

Project team:

Anzeela Fatima 1812257

Rumsha Khan 1812279

Project Supervisor:

Shahzad Haroon

Date: 24st July, 2022

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science in the

Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology (SZABIST) Karachi

Campus

Declaration of Authorship

We Anzeela Fatima (1812257) and Rumsha Khan (1812279) declare that this report "Detection And Prevention of Machine Learning Attacks in Adversarial Setting" and the work presented in this report is our own.

The work has been done completely while in the candidacy for a bachelor's degree at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology (SZABIST) Karachi, any report previously submitted on this topic in this university or any other institution is clearly mentioned in this report. Everything we used in this report which is submitted by others or belong to any other person or organization is stated in this report.

We have always cited the work we have used. We have acknowledged all source of help we used for this report. The report is based on the research work done by the team members with, we have clearly stated the sources where we took help from to conduct the research.

Signed: Anzeela Fatima (1812257)
Rumsha Khan (1812279)

Date:
24st July, 2022

Project Description

Over the last decade, we've seen a phenomenal rise, especially in artificial intelligence (AI) and deep learning (DL). The realization of deep learning as a machine learning (ML) classifier has recently received a lot of attention, especially in computer vision applications, telecommunications, and the control of autonomous systems. A classifier is an ML model that learns a mapping function between a set of inputs and classes. For example, an anomaly detector is a classifier that takes the characteristics of network traffic as input and assigns it to a normal or abnormal class. Deep learning is vulnerable to well-designed input examples. These samples are human-negligible low noise / noise and can easily trick high-performance deep learning models. Our project uses adversarial examples to train the model so that it can successfully prevent and detect attacks before they cause any actual damage.

Acknowledgement

In the name of ALLAH, the most beneficent and merciful who gave us the knowledge and courage to work on this research area. First of all, we would like to thank our supervisor Sir Shahzad Haroon. The field of study was new to us and we needed proper guidance and Sir Shahzad was there for us whenever we needed guidance or assistance in the project and supported us throughout the project. We thank all the teachers who have guided us with their knowledge and experience. We thank our parents for always supporting and encouraging us to make us better. Finally, We would like to thank the institution where we started our journey. Supportive faculty and a good environment helped us and helped us professionally and personally.

Contents

Project Proposal

1 Introduction

The previous decade has seen the incredible ascent of Artificial Intelligence (AI) and particularly Deep Learning (DL). The accomplishment of deep learning, as Machine Learning (ML) classifier, has drawn great attention in the recent, particularly in computer vision applications, telecommunications and control of autonomous systems. A classifier is an ML model that learns a mapping function between inputs and a set of classes. For instance, an anomaly detector is a classifier taking as inputs a network traffic feature and assigning them to the normal or abnormal class.
Deep learning is vulnerable against well-designed input samples. These samples can easily fool a well-performed deep learning model with little disturbance/perturbations negligible to humans. In our project we train aur model using adversarial examples so that when we encounter an attack our model can successfully prevent and detect it before it can cause any real damage.

2 Objective

We aim to implement a defense method for machine learning that would successfully detect and prevent any unexpected adversarial attack that could harm our data and protect us from any unwelcomed intruders.

3 Problem Description

Despite the success of ML in real time applications, it shows a vulnerability to data integrity threat. Such attacks are frequently incarnated by adversarial examples: legitimate inputs modified by adding small, often indistinguishable, perturbations to force an experienced classifier to misclassify the resulting adversarial inputs, while remaining correctly classified by the human observer. That is why knowing the insignificant perturbation gives us an idea of the level of robustness of Machine Learning model in the face of adversary attacks. When applied to machine learning based security products, these attacks can lead to a scathing security violation. Although a substantial number of studies has been directed on adversarial attacks in computer vision, there are very few studies on this matter of intrusion detection and intrusion prevention. Therefore, we aim to defend against such attacks by implementing a successful defense mechanism by using adversarial training so that it may detect an adversarial attack and help prevent it.

4 Methodology

Despite their popularity, DNNs have demonstrated to be at-risk to adversarial attacks in network traffic where, by introducing inconspicuous changes, an adversary can delude the classifier and as a result a malicious packet could be labeled as benign and vice versa. Therefore, in our project we study the effect of adversarial attacks on our ML model and then train our model on those adversarial examples so that it may detect and prevent adversarial attacks. We would first preprocess our benchmark dataset then train it on our model. After doing so, with the help of multiple adversarial attack we would train our dataset to generate an adversarial dataset. After that we would check accuracy of our dataset using different ML models. Then we implement a defense method to counter adversarial attacks by adversarial training method.

5 Project Scope

Recent studies in computer vision have shown that Deep Neural Networks can be deemed unsafe to adversarial attacks that are capable of misleading them into misclassification by injecting distinctively crafted data. In security- scathing areas, such attacks can cause major damage; therefore, in this project, we observe the effects of adversarial attacks on deep learning-based intrusion detection. In addition, we scrutinize the *efficiency* of adversarial training as a defense against such attacks. Experimental results show that with adequate disturbance, adversarial examples are able to deceive the detector and that the employment of adversarial training can improve the robustness of intrusion detection.

The idea behind adversarial training is to inject adversarial examples with their true labels into the training data so that the model learns how to manage them. To do this, we use various attacks to initiate adversarial samples before mixing them with the training data set. Here, we want to study two parameters of this defense: first, the effect of attack strength used to generate adversarial samples for the training, let's call it defense to avoid confusion with the strength of adversarial attack in the attack phase. Second, the proportion of adversarial training samples compared to clean training samples in the training data.

We first evaluate the result of adversarial attacks on a deep learning-based intrusion detection system. then, in the second part, we examine the efficiency of adversarial training as a means of making the system more robust against these attacks. we then summarized by discussing and analyzing the results obtained

6 Features

FYP 1:

1. Selecting Appropriate Benchmark
2. Preprocessing
a. Data Cleaning
b. **Encoding Categorical Dataset** Label Encoding
3. Create Tensorflow Based Model
4. Implementing Classifiers
a. Training Dataset (Train Test Validation)
b. Representation of categorical variable as binary vector (One Hot Encoding)
5. Training our Model
a. Creating a Confusion Matrix for original dataset
6. Implementing FGSM using Cleverhans Attack Module
a. Performing attack on our model with non-manipulated dataset
b. Creating a Confusion Matrix for Attacked dataset
7. Creating a dataset with adversarial example
8. Training our model with manipulated dataset
a. Creating a Confusion Matrix for Adversarial Trained Dataset
9. Implementing multiple attack from Cleverhans Attack Module

FYP 2:

1. Checking accuracy of our model with different attacks
2. Using different ML models to check accuracy of different attack
a. Comparing accuracy given by different models on multiple attacks
3. Implementing Defense Method (Adversarial Training)
4. Testing Defense Method
5. Using multiple benchmarks for our model
6. Preprocessing different benchmarks
7. Testing results given by different benchmarks
8. Working on GUI

7 Feasibility Study

Risks Involved:

- Update in versions of software (Python, TensorFlow, pip etc.) could cause us to lose some time in figuring an appropriate alternative or solution to the arising problem.
- Non-Technical Risk-

During the development of the project, the major risk is time. We have to complete the whole project in a limited time. Due to which the quality of the project may affect.

Resource Requirement: Computer or laptop with the specification of 8GB Ram and 500GB hard disk and need some software on which we will make our applications. Like: Anaconda, PyCharm, TensorFlow, and GitHub.

8 Solution Application Areas

Our project targets companies that are leaning toward ML for data processing and network flow. We provide a defense method against adversarial attacks that are a real threat to intrusion detection systems based on deep learning. By generating samples using adversarial attacks, an attacker can lead the system to misleading and, given adequate attack strength, the performance of the intrusion detection system can decline significantly. Our project can improve to some extent the robustness of deep learning-based intrusion detection systems.

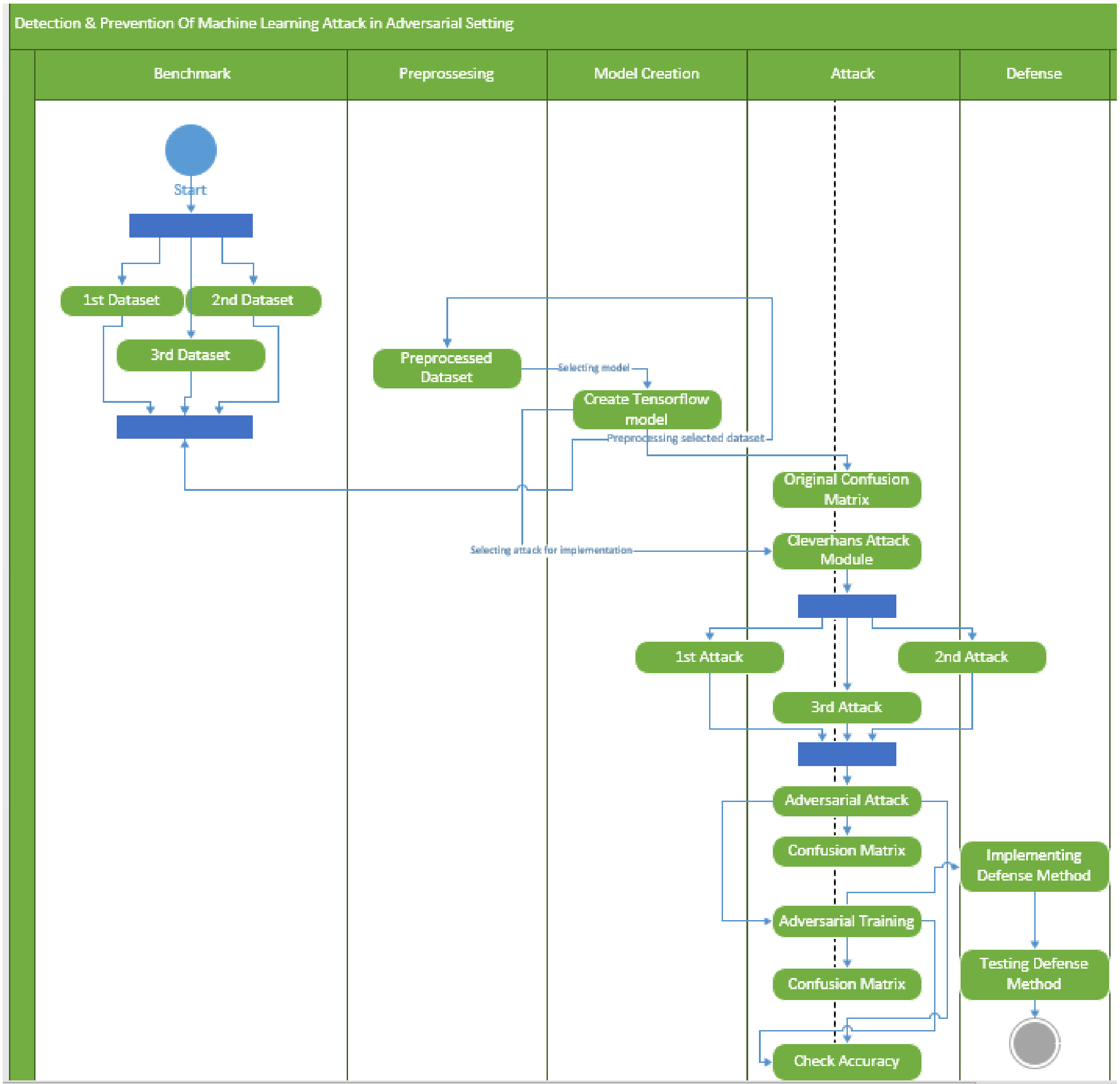
9 Tools/Technology

Python, Spyder, Tensorflow, Keras, Cleverhans, Pandas, Numpy, Scikit-learn, Tkinter, Anaconda.

10 Expertise of the Team Members

Both the team members have equal interest in the project and have studies AI in 6^{th} Semester. Both the team members have also opted for the Data Science elective which might be of use in this project. We gained interest in AI after working on the project for our AI course in which we scored first position for the project ranking. In our project we Dubbed and Translated a Video/Audio File from English to Urdu by displaying a GUI for user input of an audio or video file.

11 Milestone



Software Requirements Specification

12 Introduction

12.1 Purpose

The purpose of this document is to describe the requirements for the users to use our project i.e Detection and Prevention of Machine Learning Attack in Adversarial setting.

12.2 Document Conventions

We are using “Times New Roman” writing style. For headings we are using 24, for sub-headings we are using 18 font size in bold and 12 for Paragraphs.

12.3 Intended Audience and Reading Suggestions

The audience and beneficiaries from this project are the individuals who are interested or doing research in the field of security of applications built on Machine Learning algorithms

Our project also targets companies that are leaning toward ML for data processing and network flow. We provide an defense method against adversarial attacks. Our project can improve the robustness of DNN IDS to some extent. Someone interested in cybersecurity or ML would also find this interesting

12.4 Product Scope

It recently came to light that the Deep Neural Network has shown to be vulnerable to adversarial attacks in network traffic. Through initiating inconspicuous substitution, an adversary can successfully delude the classifier and as a result a malicious packet could be labelled as benign and vice versa. Therefore, in this paper we research about the impact of adversarial attacks on our ML model and then train our model on those adversarial examples so that it may detect and prevent adversarial attacks in future.

Results of experiments conducted shows that adversary samples can fool the detector with proper interference, and that adversarial training can be used to improve intrusion detection robustness

Advances in ML in real-time applications have proven vulnerable to integrity attacks. Such attacks are shown by adversarial example. The actual input is modified by adding subtle and imperceptible perturbations to claim that the trained classifier misclassifies subsequent adversarial inputs while remaining correctly classified by the human observer. To this end, the knowledge that the impact of confusion is minimal gives us an idea of how robust the ML model is in the area of adversarial attack. When applied to AI-based security elements, these attacks can lead to underlying security vulnerabilities. While considerable research has focused on adversarial attacks in computer vision, there is not much research on the subject of network traffic.

Firstly we examine the results of adversarial attacks on DNN based IDS. Then, after doing so we evaluate the efficiency of adversarial training to make the system more robust against these attacks. we then concluded the results by analyzing the results we obtained.

12.5 References

Ian Goodfellow, Nicolas Papernot, Ryan Sheatsley. “attacks module” .2017. 25 Oct 2021. https://cleverhans-nottombrown-fork.readthedocs.io/en/latest/source/attacks.html

Ansam Khraisat,Iqbal Gondal, Peter Vamplew, Joarder Kamruzzaman “Survey of intrusion detection systems: techniques, datasets and challenges” . 2019. 23 oct 2021. https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

Hatem Ibn-Khedher, Mohamed Ibn Khedher and Makhlouf Hadji. “Mathematical Programming Approach for Adversarial Attack Modelling”. 2021. 26 Oct 2021. https://www.scitepress.org/Papers/2021/103242/103242.pdf

13 Overall Description

13.1 Product Perspective

DNNs have proven vulnerable to attacks on network traffic, and attackers can trick classifiers by introducing unobtrusive changes and as a result a malicious packet could be labeled as benign and vice versa. Therefore, in our project we analyze the outcomes of adversarial attacks on our ML model and then train our model on those adversarial examples so that it may detect and prevent adversarial attacks. We would first preprocess our benchmark dataset then train it on our model. After doing so, with the help of multiple adversarial attack we would train our dataset to generate an adversarial dataset. After that we would check accuracy of our dataset using different ML models. Then we implement a defense method to counter adversarial attacks by adversarial training method.

13.2 Product Functions

We will develop an intrusion detection system for studying the efficiency of counterattacks. We focus on "white box" type theft attacks rather than targets. That is, the DNN's internal architecture used to find the attacker has prior knowledge, misleading the system and launching attacks during the predictive process. Then, in order to strengthen DNN's strength against these attacks, we will combine clean training data and counter samples during training to thoroughly consider reverse training as a defense against counterattacks.

13.3 User Classes and Characteristics

The Application will be used by “Experienced professionals those that have prior knowledge of machine learning algorithms and also deep neural network” and those individuals should have a networking background including knowledge about intrusion detection system. This mechanism will be used by organization that has an application that uses machine learning. This is not a userfriendly mechanism /interface and can only be used by experienced individuals.

13.4 Operating Environment

Programing Language (Python):

Python is the programming language we will be using.

Integrated Development Environment (Spyder):

Anaconda navigator provides multiple IDEs, from which w will be using Spyder.

13.5 Design and Implementation Constraints

2.5.1 Evaluating what has been learned:

Evaluation is the key to success in machine learning and data mining. Systematic diagnostic methods are used for the test set to determine which ranking algorithm is suitable for a particular ranking problem, or to estimate how well different algorithms work and compare them to each other. Why are training sets and other test sets evaluated? Model overfitting should be avoided. If the data is not distributed, the same data will be used for training and testing, and as a result, the model will repeat the labels of the samples seen during training during the initial testing phase and produce results. Perfect score for accuracy. I can predict the test set, but I can't predict the invisible data.

13.5.1 Feasibility Study

Risks Involved:

- Update in versions of software (Python, TensorFlow, pip etc.) could cause us to lose some time in figuring an appropriate alternative or solution to the arising problem.
- Non-Technical Risk-

During the development of the project, the major risk is time. We have to complete the whole project in a limited time. Due to which the quality of the project may affect.

Resource Requirement: Computer or laptop with the specification of 8GB Ram and 500GB hard disk and need some software on which we will make our applications. Like: Anaconda, PyCharm, TensorFlow, and GitHub.

13.6 User Documentation

The documentation for this project involves:

- System Requirement Specification (SRS)
- System Sequence Diagrams
- Sequence Diagram
- Dataset Details
- Gantt Chart
- Use case diagram
- Domain Model
- Use cases
- Test Cases
- Process / Model Details
- Iteration Plan

13.7 Assumptions and Dependencies

Machine Learning models are vulnerable to exploitation and we cannot ever attain 100% security. In our project we first attack the dataset then use different classifiers for classification. This will ensure that our defense mechanism will provide security against the attack that we implemented and would be somewhat successful in detection and preventing such attacks in near future before any harm is done

User should have knowledge of Deep Learning which comes under the umbrella of Machine Learning. User should also have knowledge about ML Classifiers.

2.7.1 Budget Assumption

- if processing cannot be done with the current laptop ram, then more ram would need to be added
- If faced with space issues, rom would need to me increased

2.7.2 Constraint:

- Must finish 25% of the project work within 8 weeks
- Must finish 50% of the project work within 16 weeks
- Must finish 75% of the project work till the mid of Spring Semester
- Must finish 100% of the project completed till the finals of Spring Semester • Must deliver the project within the deadline.

14 External Interface Requirements

14.1 User Interfaces

This is the base file that runs the software using the data, classifiers, and all the modules specified in the attack package. It also creates a command-line graphical user interface (GUI) that describes user interaction and software interaction, such as reading user input and displaying results.

14.2 Hardware Interfaces

The hardware interface for users is Computer, Laptop or any PC suitable for implementing Machine Learning Model.

14.3 Software Interfaces

We are using default library of Python in Spyder IDE. TO make our model we would use TensorFlow that is a framework and Keras that is a TensorFlow interface. We would also use CleverHans from which we call our attack modules. We would also need Anaconda.

14.4 Communications Interfaces

We will be using GUI which will have an option for the user to select the desired dataset out of three options the GUI will have an option to select a classifier. Based on the user selection defense mechanism would be trained and tested on benchmark and the classifier user selected

15 System Features

15.1 FYP 1

15.1.1 Selecting appropriate benchmark:

Description and Priority

Benchmark	Description	Priority
KDD’99	Dataset used for testing the performance of intrusion detection systems. Each instance is labeled with a normal or specific type of four categories: Probe, DoS, U2R, and R2L.	High

Stimulus/Response Sequences

Use case Name: Selecting Appropriate Benchmark **Summary:**
To select a benchmark to work on

Actors: System

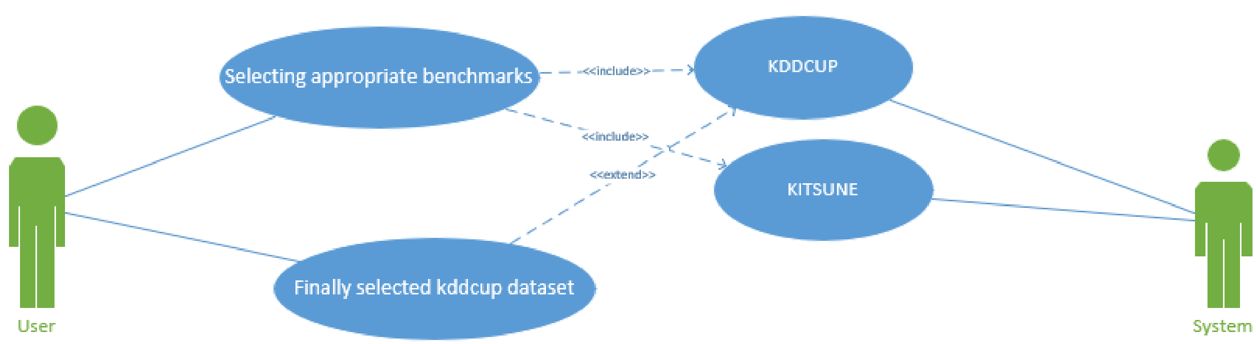
Pre-condition:

User should have a knowledge of network benchmarks and Intrusion Detection System **Basic courses of event/happy path:**

Actor Action	System Response
Select desired benchmark	System would fetch KDDCUP’99

Alternative Path: Default option is KDDCUP.

Post Condition: Name of the selected benchmark is displayed



Functional Requirements

REQ-1: System would take data from KDDCup'99 dataset as raw data and their labels.

15.1.2 Pre-Processing:

Description and Priority

		Description	Priority
Data Cleaning	Null Values	Find the null values and replace it with the mean of the column.	Medium
	Removing Outliers	It is the process in which we identify outliers and remove them.	Medium
Data Preparation	Feature Encoding	Categorical data is data which has some categories such as, in our dataset we had 5 categories.	High
	Feature Scaling	It is a technique of standardization of independent variables of the dataset in a specific range.	High

Stimulus/Response Sequences

Use case Name: Preprocessing

Summary: Preprocess KDDCup dataset

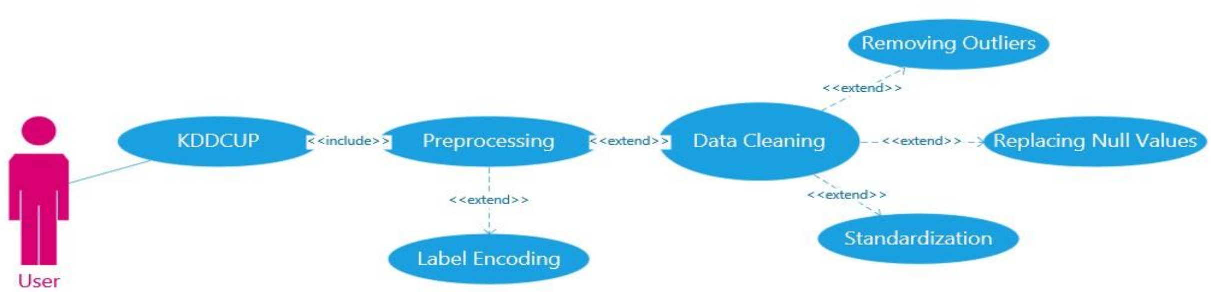
Actors: System

Pre-condition: System should receive an appropriate benchmark that could be preprocessed **Basic courses of event/happy path:**

Actor Action	System Response
Appropriate benchmark selection	System performs data cleaning, scaling and encoding

Alternative Path: None

Post Condition: Data is preprocessed



Functional Requirements

REQ-1: System would pre-process the given raw data.

15.1.3 Create Tensorflow Based Model

Description and Priority

	Description	Priority
TensorFlow Sequential Model	Use a sequential model with Keras as an interface to the TensorFlow library.	High

Stimulus/Response Sequences

Use case Name: Create Tensorflow based Model **Summary:**
Creating a tensorflow model

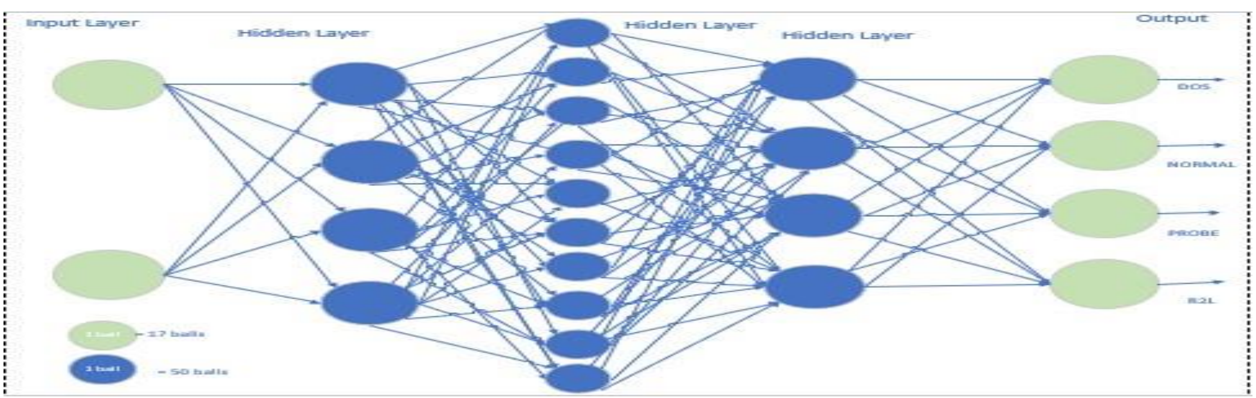
Actors: System

Pre-condition: System should receive an appropriate benchmark preprocessed **Basic courses of event/happy path:**

System Action	**System Response**
Getting parameters of preprocessed dataset	Create a DNN model with our preprocessed dataset using KerasAPI

Alternative Path: None

Post Condition: A tensorflow model created



Functional Requirements

REQ-1: Build Sequential Model

- Provide and input and output shape and hidden layers so that it can create our model - Provide appropriate activation function

REQ-2: configuration and compilation of our model with appropriate metrics.

15.1.4 Training our Model

Description and Priority

	Description	Priority
Model.fit	Adjusting models parameters and minimalizing loss.	High

Stimulus/Response Sequences

Use case Name: Training our model

Summary: Training our model with training dataset and compiling it.

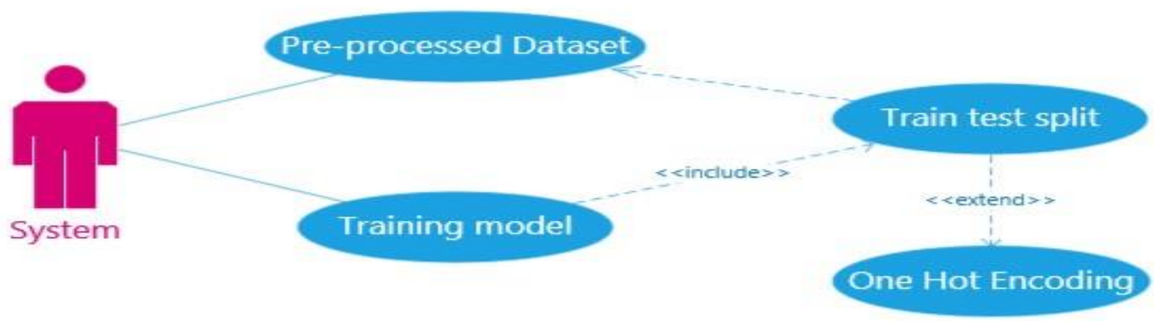
Actors: System

Pre-condition: A model that is created **Basic courses of event/happy path:**

Actor Action	**System Response**
Getting parameters after creating our model	After model is compiled, system would use .fit() method to adjust our model's parameter and loss functions to our needs

Alternative Path: None.

Post Condition: Model trained successfully



Functional Requirements

REQ-1: System would then train our model.

– Once the model is compiled, the system will train it by using using .fit() which is a built-in API for training

15.1.5 Implementing Classifiers

Description and Priority

	Description	Priority
Model.predict	.predict() generate predictions (probabilities -- the output of the last layer)on new data	High

` Stimulus/Response Sequences

Use case Name: Implementing Classifiers

Summary: Implementing a classifier for classification of our dataset

Actors: System

Pre-condition: System should have a trained model **Basic courses of event/happy path:**

Actor Action	System Response
Getting parameters from training our model	After training our model we would use .predict() so that we can get the predictions based on probabilities. We would use an ANN(MLP) model for classification.

Alternative Path: None

Post Condition: Classifier successfully implemented

Functional Requirements

REQ-1: System would then classify our model.

– Once the model is trained, the system will predict it's classes by using using .predict() which is a built-in API for classification.

15.1.6 Implementing FGSM Using Cleverhans Attack Module:

Description and Priority

	Description	Priority
Fast Gradient Sign Method (FGSM) Attack	An attack used to generate adversarial samples.	Medium

Stimulus/Response Sequences

Use case Name: Implementing FGSM using Cleverhans attack module **Summary:** Attacking our model with FGSM

Actors: System

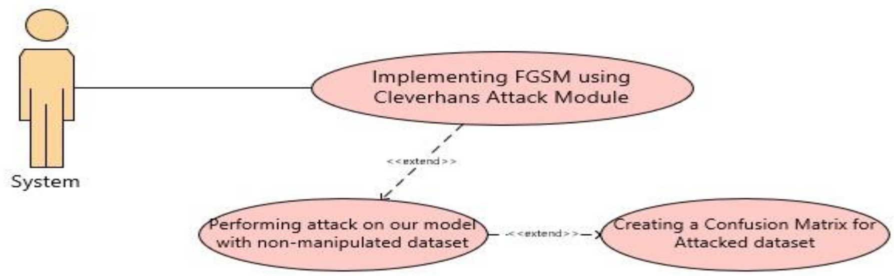
Pre-condition: System should have a classifier implemented

Basic courses of event/happy path:

Actor Action	**System Response**
Getting parameters after implementing classifier.	After our dataset is trained and a classifier is implemented, the system would attack our dataset using FGSM. The system would then create a confusion matrix based on the manipulated dataset.

Alternative Path: None

Post Condition: FGSM attack successfully implemented



Functional Requirements

REQ-1: After classification we would perform FGSM attack using Cleverhans Attack Module

15.1.7 Creating a dataset with adversarial examples:

Description and Priority

	Description	Priority
Adversarial Example	After the generation of adversarial samples, we would put it in a dataframe so that we could then convert it to csv format in order to save it in excel. We would inject adversarial examples in the nonmanipulated dataset with labels correspondingly.	Medium

Stimulus/Response Sequences

Use case Name: Creating a dataset with adversarial examples

Summary: Create a dataset after injecting adversarial examples

Actors: System

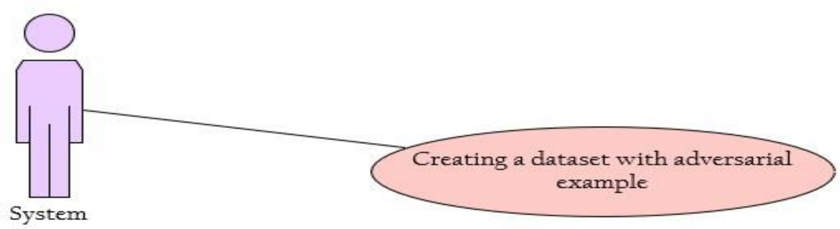
Pre-condition: System should have adversarial examples stored

Basic courses of event/happy path:

Actor Action	**System Response**
System would have adversarial examples after the attack.	System would firstly concatenate train and adversarial test set and then save it to dataframe so that it can then be converted to csv

Alternative Path: None

Post Condition: Dataset containing adversarial examples created



. Functional Requirements

REQ-1: System would allow the user to save an adversarial set that is resulted from attacking their chosen classifier.

15.1.8 Implementing multiple attacks from Cleverhans Attack Module

Description and Priority

	Description	Priority
Momentum Iterative Method	Enhanced version of FGSM. Another attack from Cleverhans	High
Projected Gradient Descent (PGD)	Another attack from Cleverhans Library	High

Stimulus/Response Sequences

Use case Name: Implementing multiple attacks from Cleverhans Attack Module **Summary:** Implementing multiple attacks.

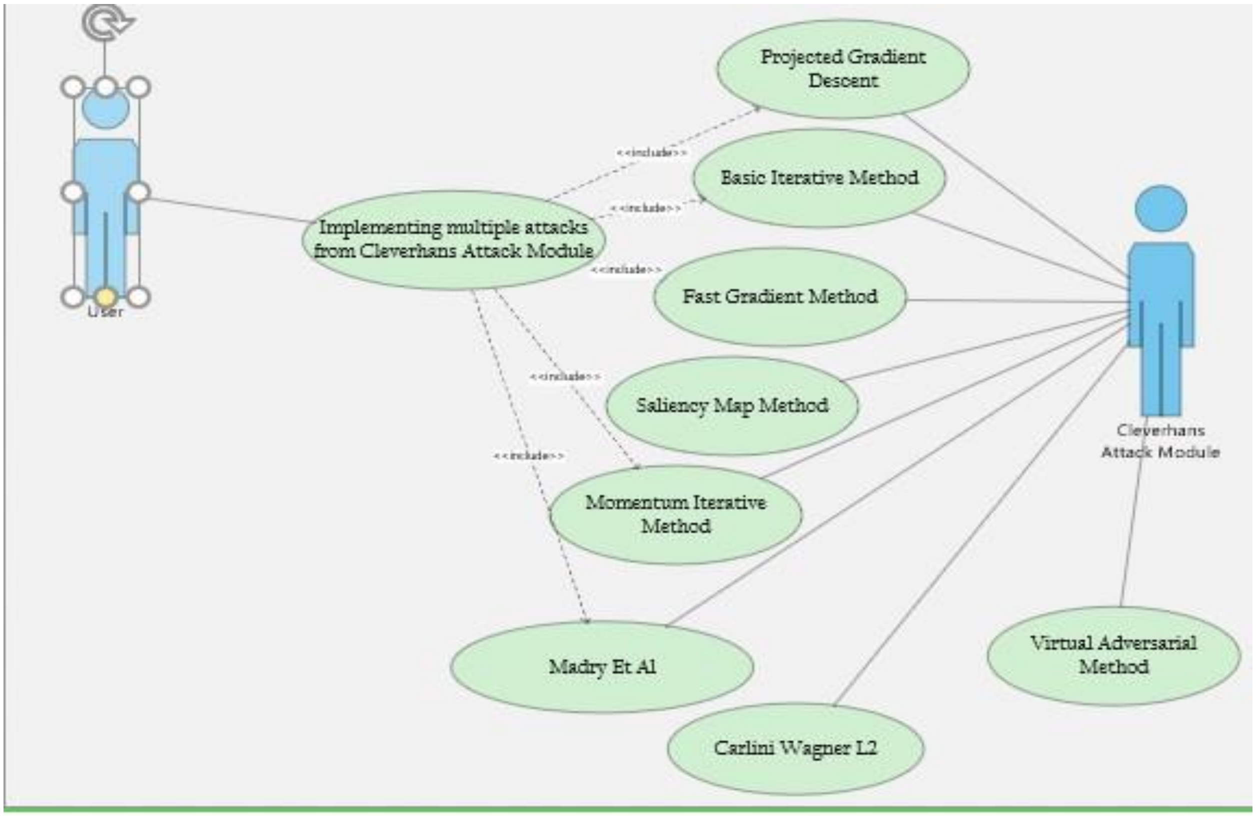
Actors: System

Pre-condition: Classifier should be properly implemented **Basic courses of event/happy path:**

Actor Action	System Response
Getting parameters after implementing classifier.	After classifier is implemented, the system would attack our dataset using **Momentum Iterative Method** After the attack system would have a manipulated dataset with values varying from the original dataset.
Getting parameters after implementing classifier.	After classifier is implemented, the system would attack our dataset using **Projected Gradient Descent**. After the attack system would have a manipulated dataset with values varying from the original dataset.

Alternative Path: None

Post Condition: Attack successfully implemented



Functional Requirements

REQ-1: Have a trained model and all required libraries. Have required python and tensorflow version to import cleverhans library.

15.1.9 Checking accuracy of our model with different attacks

Description and Priority

	Description	Priority
AUC - ROC	Measures the capability of our model to distinguish between classes.	High
F1Score	F1 score is a harmonic mean between precision and recall.	High
Accuracy	Calculated from the ratio of the number of correct predictions to the total number of predictions.	High

Stimulus/Response Sequences

Use case Name: Accuracy of different attacks

Summary: Calculate Accuracy, F1score and AUC-ROC after implementing different attacks

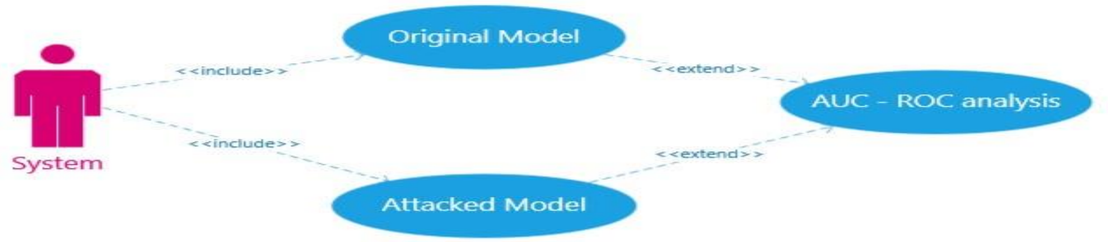
Actors: System

Pre-condition: Different attacks has to be implemented prior to checking accuracy **Basic courses of event/happy path:**

Actor Action	System Response
Parameters from FGSM attack	System calculates F1score, accuracy and AUC-ROC

Alternative Path: None

Post Condition: Evaluation metrics displayed



Functional Requirements

REQ-1: System would allow validating and evaluating trained classifiers before and after attacks.

– Having a trained classifier, the system will present the results from calculating the Accuracy, Precision and Recall scores of the Classifier before and after attacks.

15.2 FYP 2

15.2.1 Using different ML models to check accuracy of different attacks

Description and Priority

	Description	Priority
KNN	KNN (K-Nearest Neighbor) algorithm uses "feature matching" to predict the value of a new data point.	High
Decision Tree	Mostly adequate for large datasets. Detection accuracy of this classifier is high	High
Bayesian Network	It is based on a graphic model that creates potential relationships between variables of interest.	High

Stimulus/Response Sequences

Use case Name: Implementing Classifiers

Summary: To select a Classifier to work on

Actors: User

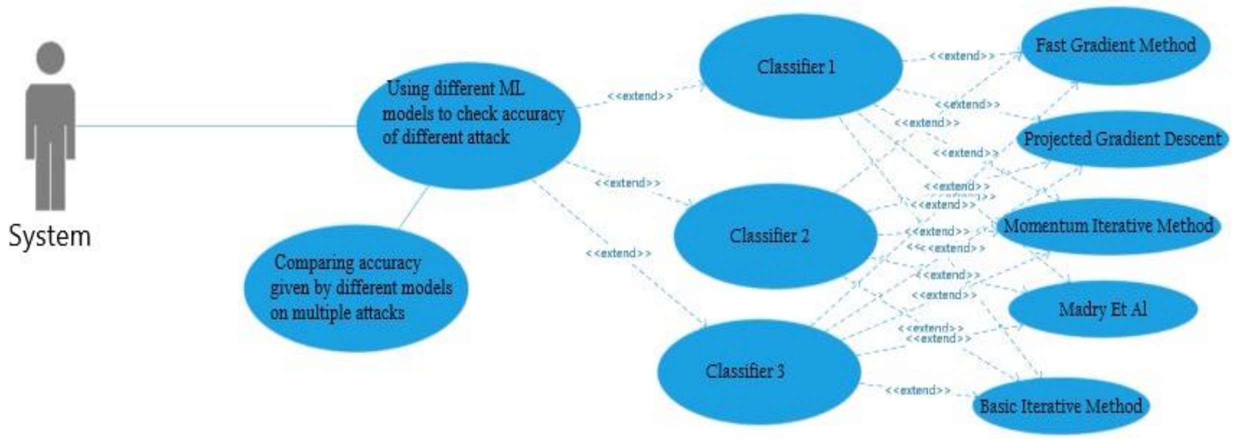
Pre-condition: User should have a knowledge of Machine learning models

Basic courses of event/happy path:

Actor Action	System Response
This use case initiates when a user is providing an option to select the desired classifier.	System prompts the user to select a classifier from three options provided namely KNN, Decision Tree, NB. System also calculate accuracy accordingly.

Alternative Path: Default option is MLP (Multi-Layer Perceptron).

Post Condition: Name of the selected classifier is displayed



Functional Requirements

REQ-1: System would provide different classification algorithms to create a Machine Learning model (Classifier).

– The system will have three different classifiers to create a classifier, KNN, Decision trees and Naïve Bayes. This is because classification algorithms differ in the following criteria which leads to different results i.e predictive accuracy, speed, robustness and scalability.

15.2.2 Implementing Defense Method (Adversarial Training)

Description and Priority

	Description	Priority
Adversarial Training	In method provides robustness against adversarial attacks by inserting the adversarial samples in clean training set and then passing it to the model and different classifiers to training using the training set containing adversarial sample with their true label	High

Stimulus/Response Sequences

Use case Name: Adversarial Training

Summary: Implementing defense against adversary

Actors: System

Pre-condition: Different attack and a model and classifier

Basic courses of event/happy path:

Actor Action	System Response
A model and an adversarial test set	Outputs result after adversarial training

Alternative Path: None

Post Condition: Adversarial Training successful

Functional Requirements

REQ-1: System would have an manipulated dataset

– Having a trained classifier, the system will present the results from adversarial training

15.2.3 Testing Defense Method

Description and Priority

	Description	Priority
Testing Adversarial Training	We test adversarial training by evaluating its evaluation metrics before and after defense mechanism is implemented	High

Stimulus/Response Sequences

Use case Name: Testing Adversarial Training

Summary: Testing defense mechanism by checking pre and post results

Actors: System

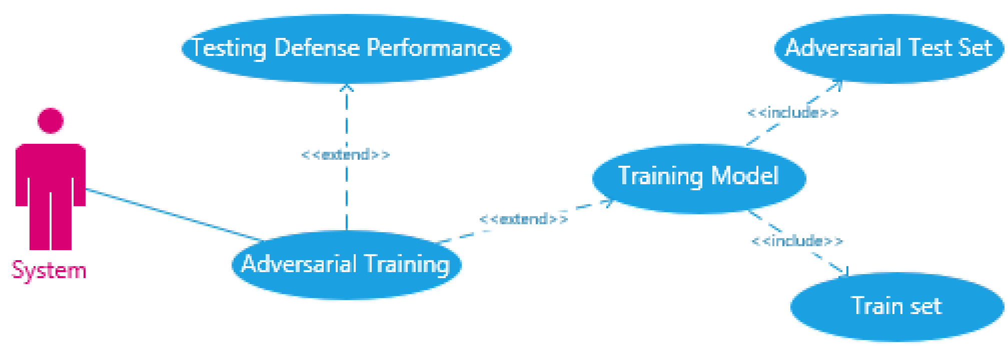
Pre-condition: Adversarial Training has to be implemented beforehand

Basic courses of event/happy path:

Actor Action	System Response
Adversarial training output	Test results after adversarial Training

Alternative Path: None

Post Condition: Testing of Adversarial Training successful



Functional Requirements

REQ-1: System would have adversarial training already implemented

– System would present test results for evaluation

15.2.4 Using Multiple Benchmarks

Description and Priority

	Description	Priority
Kitsune	This dataset has two labels in targets class, one benign and other malicious.	Medium

Stimulus/Response Sequences

Use case Name: Using multiple benchmarks for our model **Summary:** Give user multiple benchmarks to select from

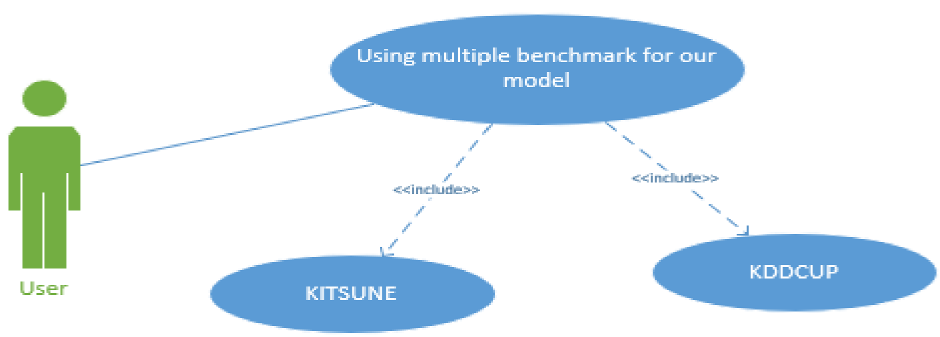
Actors: System

Pre-condition: System should receive an appropriate benchmark
Basic courses of event/happy path:

Actor Action	**System Response**
This use case initiates when a user is providing an option to select the desired benchmark	System prompts the user to select a benchmark from three options provided namely KDDCUP,KISTUNE.

Alternative Path: Default option is KDDCUP.

Post Condition: Benchmark is selected



Functional Requirements

REQ-1: System would provide different benchmarks.

– The system will have two different benchmarks, Kitsune and KDDCup.

15.2.5 Preprocessing different benchmarks

Description and Priority

	Description	Priority
Preprocessing selected benchmark	Preprocessing the benchmark the user selected from the provided options: KDDCup’99, Kitsune.	Medium

Stimulus/Response Sequences

Use case Name: Preprocessing different benchmarks **Summary:** Preprocess multiple datasets

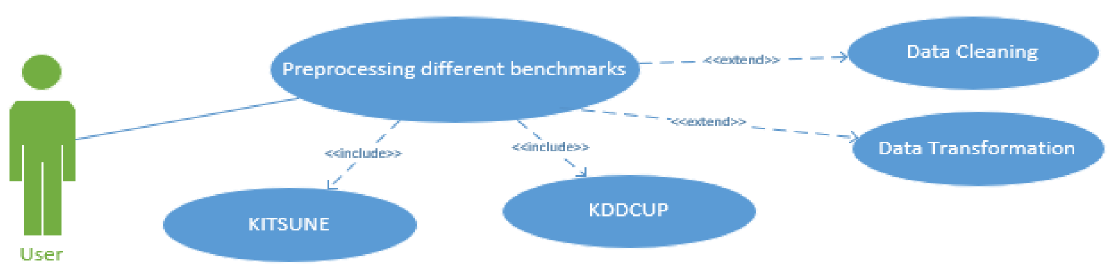
Actors: System

Pre-condition: System should receive an appropriate benchmark that could be preprocessed **Basic courses of event/happy path:**

Actor Action	**System Response**
System would have parameters from different benchmarks	System would replace null values and remove outliers After that system would prepare our data by feature encoding scaling. After that categorical data would be converted to label.

Alternative Path: Default option is KDDCUP.

Post Condition: Data is preprocessed



Functional Requirements

REQ-1: System would preprocess different benchmarks.

– The system will preprocess three different benchmarks, UNSWNB15, CICIDS2019 and KDDCup upon user request.

15.2.6 Testing results given by different benchmarks

Description and Priority

	Description	Priority
Testing results	After preprocessing we would implement adversarial training on both the dataset and then check the results.	Medium

Stimulus/Response Sequences

Use case Name: Testing results given by different benchmarks

Summary: Test the results of benchmarks after adversarial training

Actors: System

Pre-condition: System should have a pre-processed dataset **Basic courses of event/happy path:**

Actor Action	**System Response**
This use case initiates when a system receives preprocessed benchmark	System would the implement adversarial training on the benchmarks and test the results after implementation

Alternative Path: None

Post Condition: Results of two benchmarks after adversarial training

Functional Requirements

REQ-1: System would provide preprocessed benchmarks.

– The system will have three preprocessed benchmarks, Kitsune and KDDCup. The is to ensure the working efficiency of our IDS.

REQ-2: The system shall take data from the selected dataset as raw data and their labels for training and testing classifiers. It would then perform adversarial training

15.2.7 Working on GUI

Description and Priority

	Description	Priority
GUI	We would use this GUI library to give use options to select desired benchmarks or classifiers.	Medium

Stimulus/Response Sequences

Use case Name: Working on GUI

Summary: To display GUI for a friendly interface

Actors: User

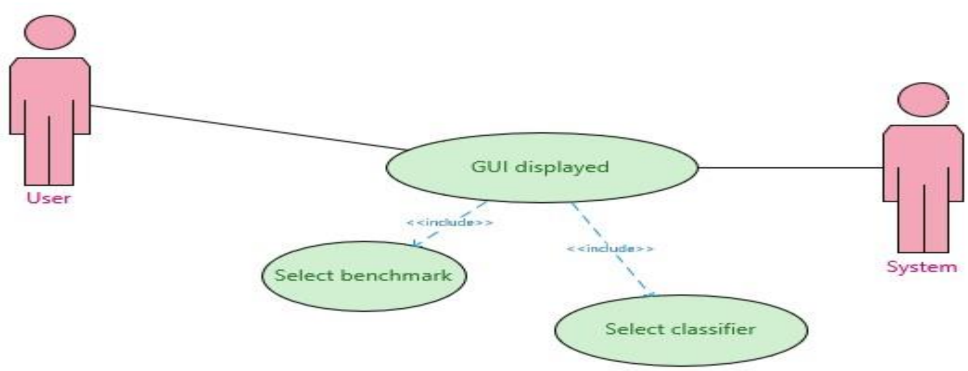
Pre-condition: Should have python

Basic courses of event/happy path:

Actor Action	**System Response**
User interaction with system	GUI is displayed showing options

Alternative Path: Interacting via command line/console

Post Condition: GUI displayed



Functional Requirements

REQ-1: System would display two options:

- To select desired benchmark
- To select the desired classifier

16 Other Nonfunctional Requirements

16.1 Performance Requirements

A good indicator of performance from the Confusion matrix would be having appropriate numbers in the diagonal elements and small numbers. In addition, these outcomes that are presented in the Confusion matrix are used by multiple assessment metrics such as, Accuracy, Recall and Precision to evaluate a classifier's performance.

The Scikit-learn library consists of a comprehensive and uniform interface to common machine learning algorithms, making machine learning easy on production systems. The library contains good documentation and standard code, combining ease of use with high performance.

16.2 Safety Requirements

If have to make sure that the datasets we are using are not illegally obtained. We also have to ensure that the dataset has correct parameters obtained through proper procedures and not just recklessly gathered and has miscalculation. We also have to make sure that the project should be in safe hands.

16.3 Security Requirements

Only personals that have fields directly related to cybersecurity can use this product. People with malicious intend should not be given authorization to use this defense mechanism.

Security issues arise from the Machine Learning model's adaptability in the presence of an adversary that can fool the DNN model by manipulating the input data in the test time of the model

16.4 Software Quality Attributes

• Usability – System will implemented as GUI software that will be easy to use and adjust.

• Speed – Pre-processing the datasets, creating the classifiers, testing and validating and evaluating the classifiers should be fast. multiple modules that can be reusable in other projects or systems.

• Re-usability – The implementation of the system will be broken into sub packages with

16.5 Business Rules

- No one is allowed to make changes in our project
- This project would be subjected to copyright
- This project should be used in an ethical manner without causing anyone any harm

17 Other Requirements

17.1 Test Cases

Test Case ID-1	Test Purpose: Training and Testing benchmark		
Preconditions: Implemented a dataset from network traffic and labels			
Test Case Steps:1			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	Input the benchmark when prompted...'Select benchmark'	A new screen should show up an option to test benchmark and the testing evaluation summary for Pass	Pass
Comments: This test case passed all steps			

Test Case ID-2	Test Purpose: Training Naïve Bayes classifier		
Preconditions: Implemented a dataset from network traffic and labels			
Test Case Steps:1			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	Input the classifier when prompted...'Select classifier'	A new screen should show up an option to train and test classifier and the testing evaluation summary for Pass	Pass
Comments: This test case passed all steps			

<table>
<tr><td>Test Case ID-3</td><td colspan="3">Test Purpose: Training Decision Tree classifier</td></tr>
<tr><td colspan="4">Preconditions: Implemented a dataset from network traffic and labels</td></tr>
<tr><td colspan="4">Test Case Steps:1</td></tr>
<tr><td>Step No</td><td>Procedure</td><td>Expected Response</td><td>Pass/Fail</td></tr>
<tr><td>1</td><td>Input the classifier when prompted...'Select classifier'</td><td>A new screen should show up an option to train and test classifier and the testing evaluation summary for Pass</td><td>Pass</td></tr>
<tr><td colspan="4">Comments: This test case passed all steps</td></tr>
</table>

<table>
<tr><td>Test Case ID-4</td><td colspan="3">Test Purpose: Training Artificial Neural Network classifier</td></tr>
<tr><td colspan="4">Preconditions: Implemented a dataset from network traffic and labels</td></tr>
<tr><td colspan="4">Test Case Steps:1</td></tr>
<tr><td>Step No</td><td>Procedure</td><td>Expected Response</td><td>Pass/Fail</td></tr>
<tr><td>1</td><td>Input the classifier when prompted...'Select classifier'</td><td>A new screen should show up an option to train and test classifier and the testing evaluation summary for Pass</td><td>Pass</td></tr>
<tr><td colspan="4">Comments: This test case passed all steps</td></tr>
</table>

<table>
<tr><td>Test Case ID-5</td><td colspan="3">Test Purpose: Attack a classifier using Fast Gradient Sign Method attack</td></tr>
<tr><td colspan="4">Preconditions: A classifier should be implemented</td></tr>
<tr><td colspan="4">Test Case Steps:1</td></tr>
<tr><td>Step No</td><td>Procedure</td><td>Expected Response</td><td>Pass/Fail</td></tr>
<tr><td>1</td><td>Call Fast Gradient Sign Method.</td><td>Results of testing the selected classifier in presence of fast gradient sign method attack displayed on console</td><td>Pass</td></tr>
<tr><td colspan="4">Comments: This test case passed all steps</td></tr>
</table>

Test Case ID-6	Test Purpose: Attack a classifier using Basic Iterative Method attack		
Preconditions: A classifier should be implemented			
Test Case Steps:1			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	Call Basic Iterative Method.	Results of testing the selected classifier in presence of basic iterative method attack displayed on console	Pass
Comments: This test case passed all steps			

Test Case ID-7	Test Purpose: Attack a classifier using Momentum Iterative Method attack		
Preconditions: A classifier should be implemented			
Test Case Steps:1			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	Call Momentum Iterative Method.	Results of testing the selected classifier in presence of momentum iterative attack displayed on console	Pass
Comments: This test case passed all steps			

Test Case ID-8	Test Purpose: Attack a classifier using Projected Gradient Descent attack		
Preconditions: A classifier should be implemented			
Test Case Steps:1			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	Call Projected Gradient Descent	Results of testing the selected classifier in presence of projected gradient descent attack displayed on console	Pass
Comments: This test case passed all steps			

Test Case ID-9	Test Purpose: Attack a classifier using MadryEtAl attack		
Preconditions: A classifier should be implemented			
Test Case Steps:1			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	Call MadryEtAl method	Results of testing the selected classifier in presence of MadryEtAl attack displayed on console	Pass
Comments: This test case passed all steps			

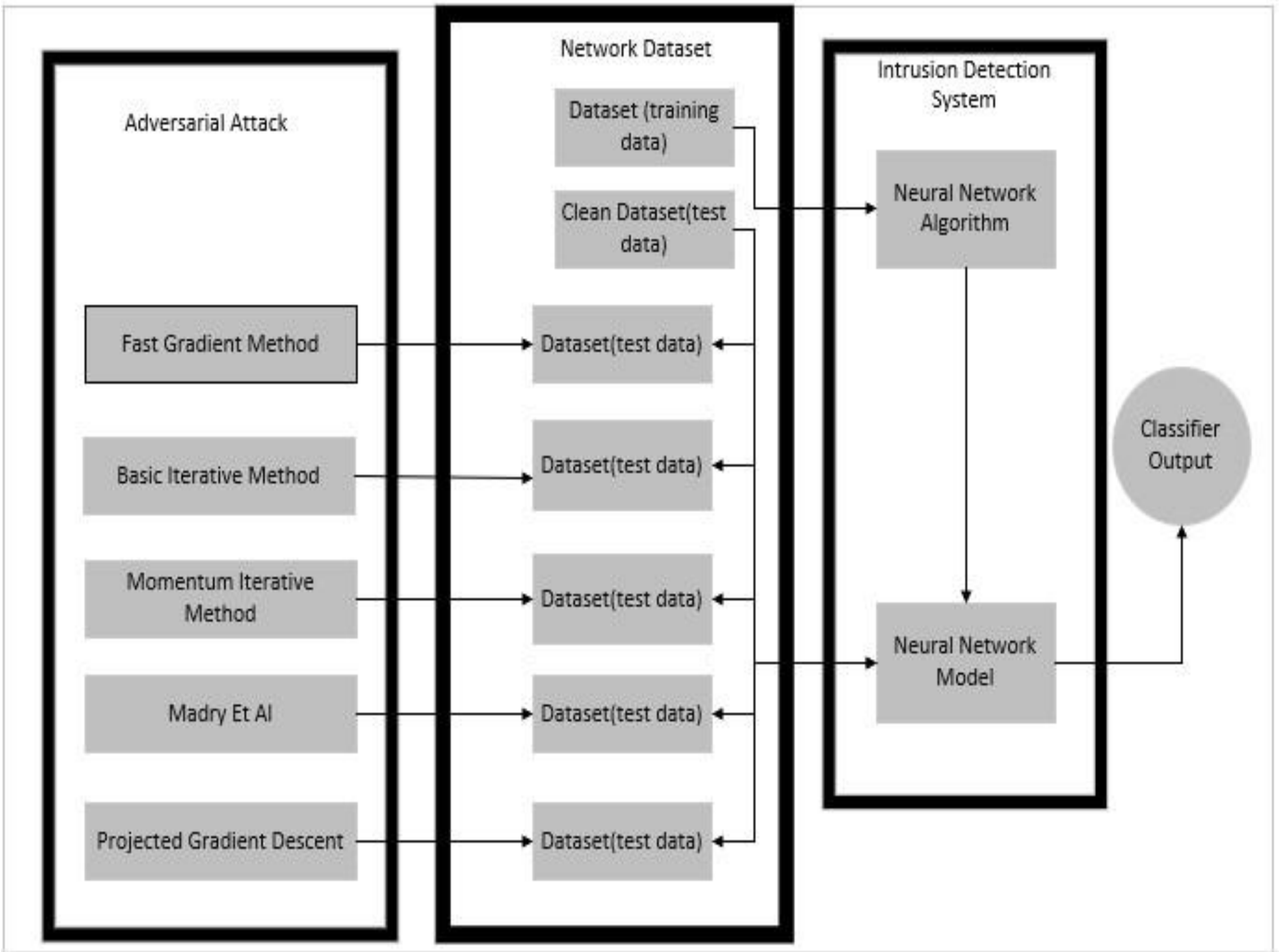
Test Case ID-10	Test Purpose: Save an adversarial set after attacking		
Preconditions: Attacks running any from previous tests			
Test Case Steps:2			
Step No	**Procedure**	**Expected Response**	**Pass/Fail**
1	When an attack is implemented, we save the result of that adversarial dataset	The adversarial dataset saved firstly in a dataframe to then be converted to a csv file	Pass
Comments: This test case passed all steps			

18 Appendix A: Glossary

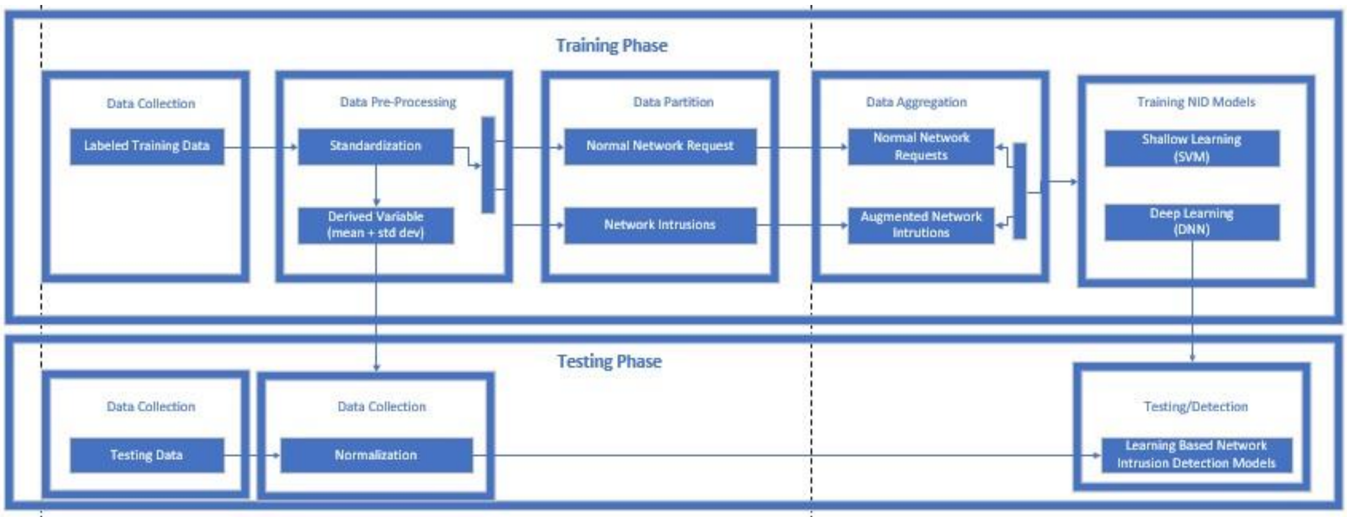
IDS: Intrusion Detection System
DL: Deep Learning
DNN: Deep Neural Network
R2L: Remote to Local
DoS: Denial Of Service
U2R: User To Root
FTP: File Transfer Protocol
UDP: User Datagram Protocol
ICMP: Internet Control Message Protocol

19 Appendix B: Analysis Models

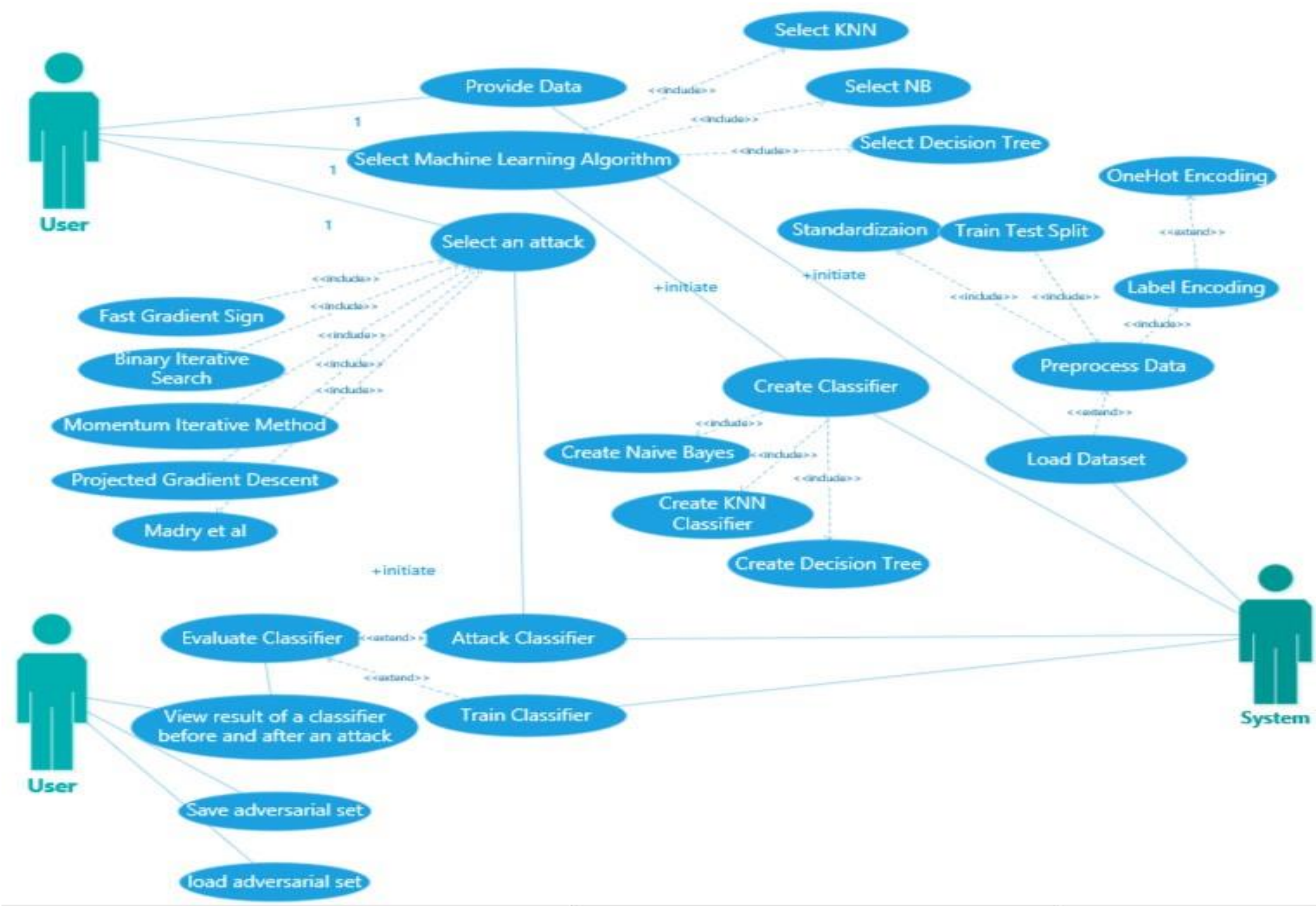
19.1 Domain Model



19.2 Data Flow Diagram



19.3 Use Case Diagram



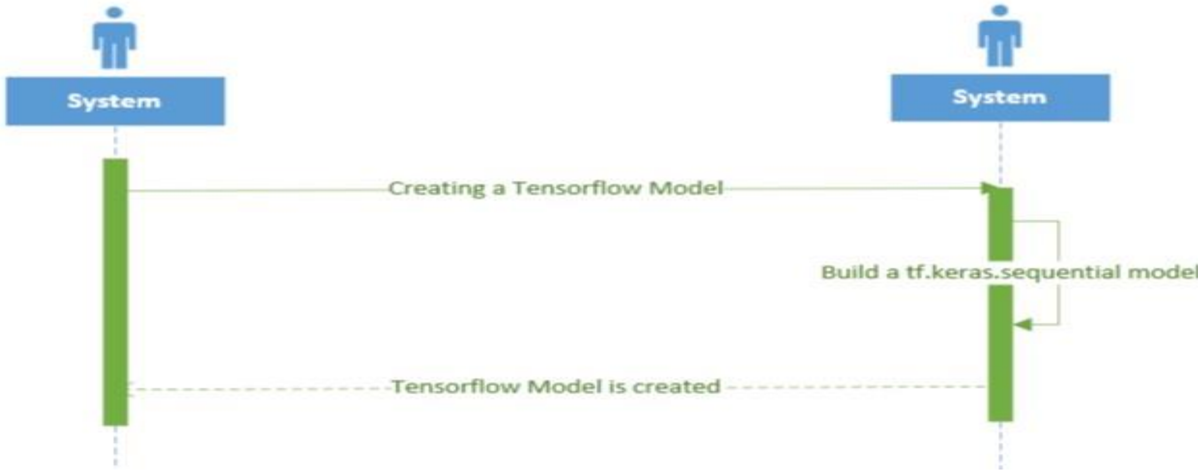
19.4 System Sequence Diagram

19.4.1 FYP 1

Selecting Appropriate Benchmark

Preprocessing

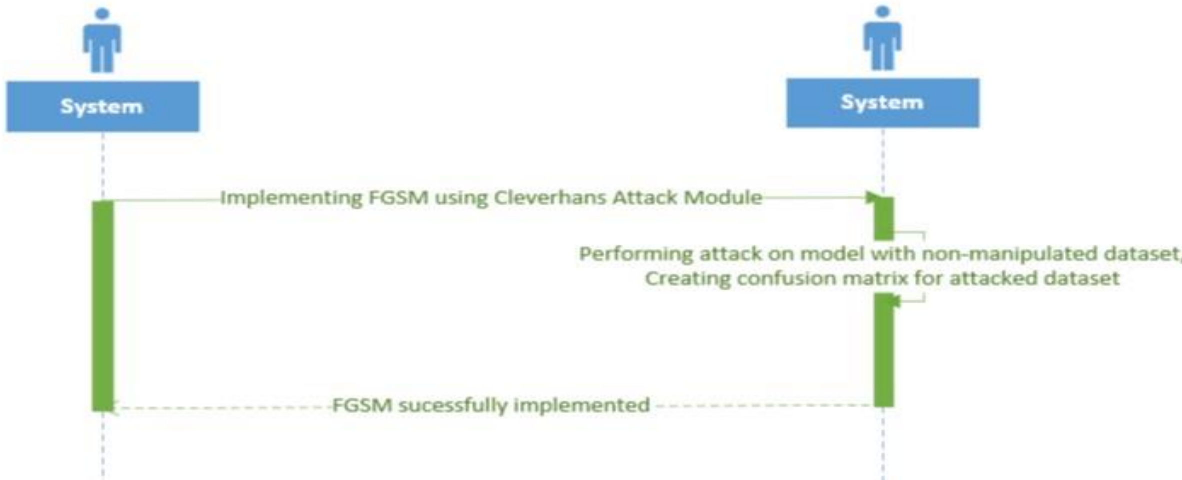
Create Tensorflow Based Model



Implementing Classifiers

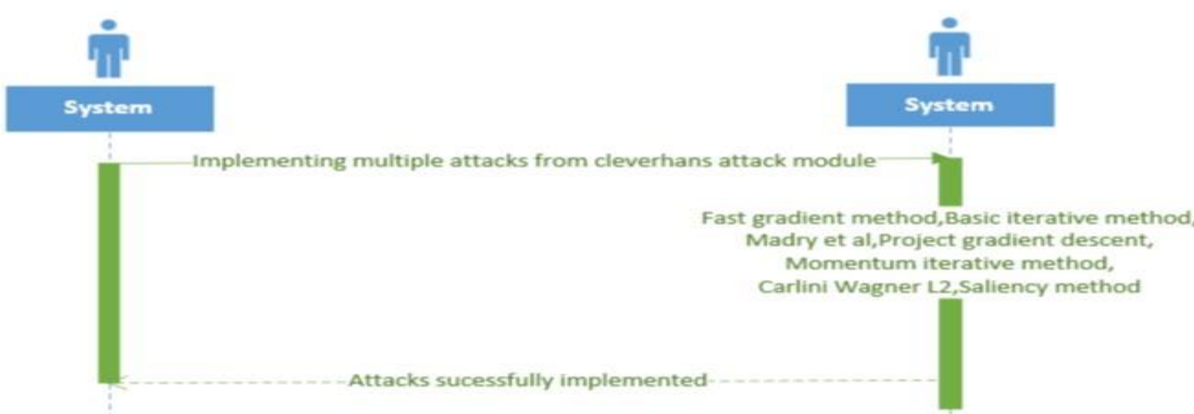
Training our Model

Implementing FGSM using Cleverhans Attack Module



Creating a dataset with adversarial example

Implementing multiple attack from Clever Hans Attack Module

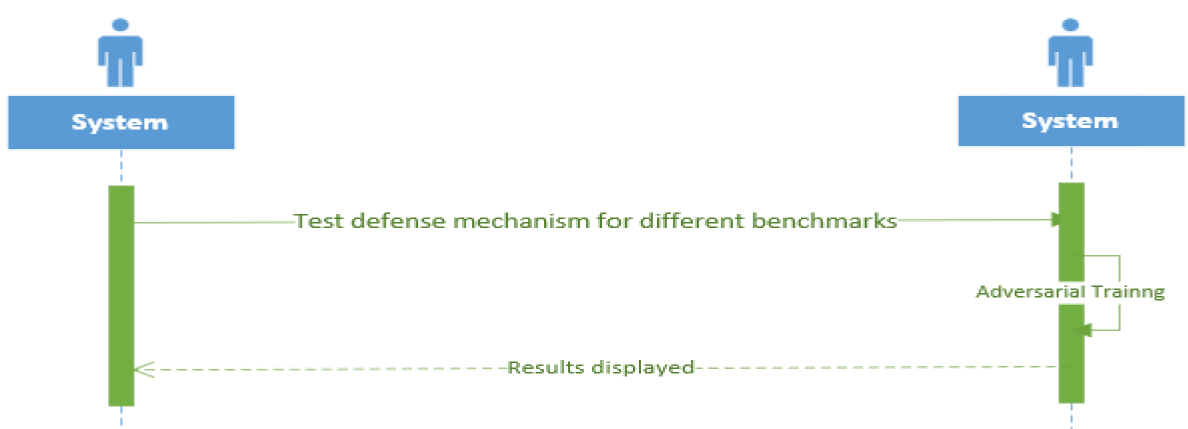


Checking accuracy of our model with different attacks

19.4.2 FYP 2

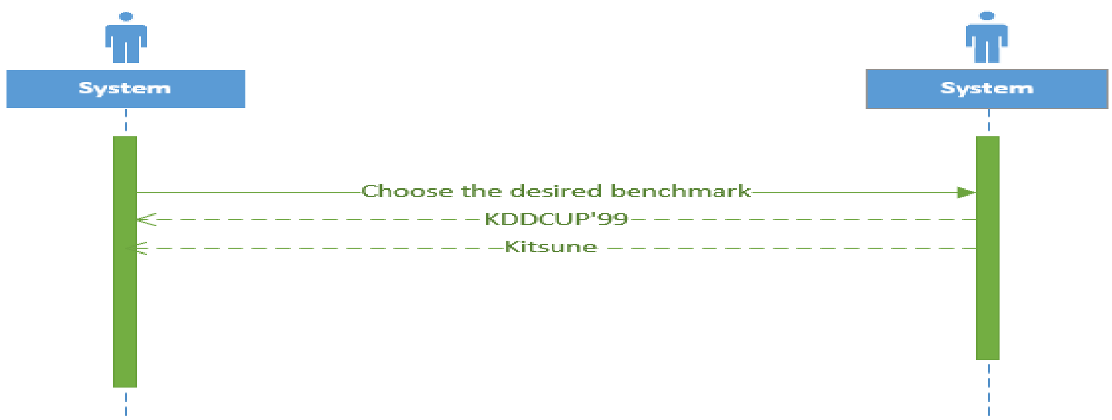
Using different ML models to check accuracy of different attack

Implementing Defense Method (Adversarial Training)



Testing Defense Method

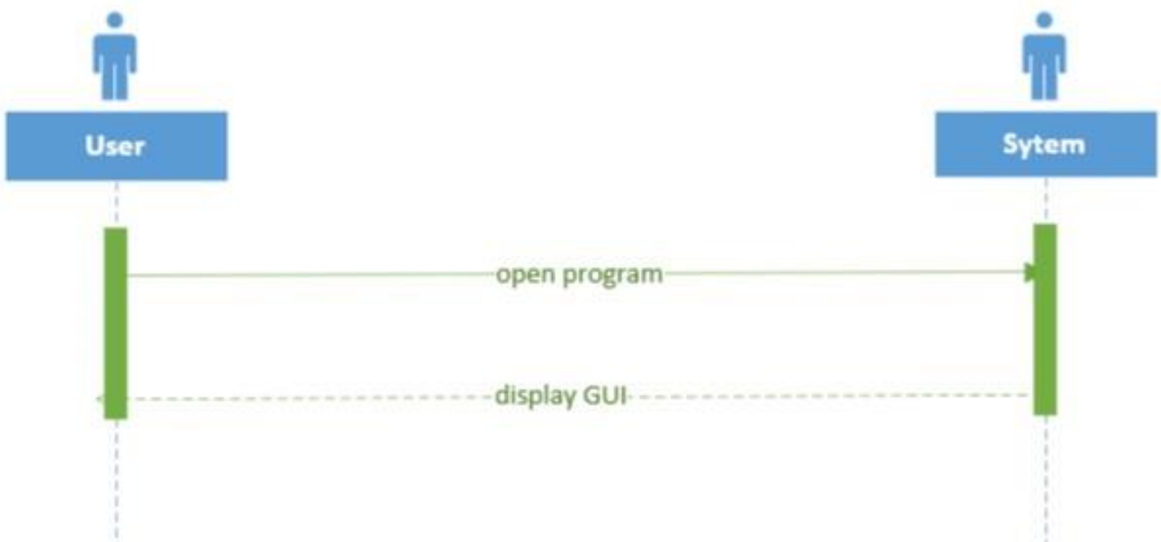
Using multiple benchmarks for our model



Preprocessing different benchmarks

Testing results given by different benchmarks

Working on GUI



20 Appendix C: To Be Determined List

1) None

Software Design Specification

21 Introduction

21.1 Purpose of this document

The Software Design Specification (SDS) is designed to provide a detailed overview of the software architecture and the design “**Detection and Prevention of Machine Learning Attack in Adversarial Settings**”. It specifies a general view of the system's architecture and the communication of user with the system. A software design specification is a design documentation that describes all the data, architecture, interface, and component-level design of our application.

21.2 Scope of the development project

It recently came to light that the Deep Neural Network has shown to be vulnerable to adversarial attacks in network traffic. Through initiating inconspicuous substitution, an adversary can successfully delude the classifier and as a result a malicious packet could be labelled as benign and vice versa. Therefore, in this paper we research about the impact of adversarial attacks on our ML model and then train our model on those adversarial examples so that it may detect and prevent adversarial attacks in future.
Results of experiments conducted shows that adversary samples can fool the detector with proper interference, and that adversarial training can be used to improve intrusion detection robustness Advances in ML in real-time applications have proven vulnerable to integrity attacks. Such attacks are shown by adversarial example. The actual input is modified by adding subtle and imperceptible perturbations to claim that the trained classifier misclassifies subsequent adversarial inputs while remaining correctly classified by the human observer. To this end, the knowledge that the impact of confusion is minimal gives us an idea of how robust the ML model is in the area of adversarial attack. When applied to AI-based security elements, these attacks can lead to underlying security vulnerabilities. While considerable research has focused on adversarial attacks in computer vision, there is not much research on the subject of network traffic.

Firstly we examine the results of adversarial attacks on DNN based IDS. Then, after doing so we evaluate the efficiency of adversarial training to make the system more robust against these attacks. we then concluded the results by analysing the results we obtained.

FYP 1:

1. Selecting Appropriate Benchmark
2. Preprocessing
    a. Data Cleaning
    b. **Encoding Categorical Dataset** Label Encoding
3. Create Tensorflow Based Model
4. Implementing Classifiers
    a. Training Dataset (Train Test Validation)
    b. Representation of categorical variable as binary vector (One Hot Encoding)
5. Training our Model
    a. Creating a Confusion Matrix for original dataset
6. Implementing FGSM using Cleverhans Attack Module
    a. Performing attack on our model with non-manipulated dataset
    b. Creating a Confusion Matrix for Attacked dataset
7. Creating a dataset with adversarial example
8. Feature Measurement
9. Implementing multiple attack from Clever Hans Attack Module

FYP 2:

1. Checking accuracy of our model with different attacks
2. Using different ML models to check accuracy of different attack
    a. Comparing accuracy given by different models on multiple attacks
3. Implementing Defense Method (Adversarial Training)
4. Testing Defense Method
5. Using multiple benchmarks for our model
6. Preprocessing different benchmarks
7. Testing results given by different benchmarks
8. Working on GUI

21.3 Definitions, acronyms, and abbreviations

Benchmark:

In our project we are using 2 network attack benchmark namely KDDCup'99 and Kitsune. These contain network packets and different network attacks. The benchmark is divided into two categories of network packet, one benign network packet and the other is malicious network packet.

Data preprocessing

After we have imported our dataset we would preprocess it. We world preprocess by first removing null values and outliers. After doing so we will standardize our dataset and then label encode and one hot encode our target class. After doing so we will split the dataset in a ratio of 80 to 20 so that it can then be passed in our model.

Intrusion Detection System (IDS):

IDS is an acronym for Intrusion Detection System. IDS's primary functions are host and network monitoring, computer system behavior analysis, alert generation, and suspicious behavior response.

21.4 References

Ian Goodfellow, Nicolas Papernot, Ryan Sheatsley. “attacks module” .2017. https://cleverhans-nottombrown-fork.readthedocs.io/en/latest/source/attacks.html

Ansam Khraisat,Iqbal Gondal, Peter Vamplew, Joarder Kamruzzaman “Survey of intrusion detection systems: techniques, datasets and challenges” . 2019. https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

Hatem Ibn-Khedher, Mohamed Ibn Khedher and Makhlouf Hadji. “Mathematical Programming Approach for Adversarial Attack Modelling”. 2021. https://www.scitepress.org/Papers/2021/103242/103242.pdf

Islam Debicha, Thibault Debatty, Jean-Michel Dricot, Wim Mees. “Adversarial Training for Deep Learning-based Intrusion Detection Systems” .20 Apr 2021. https://arxiv.org/pdf/2104.09852.pdf

21.5 Overview of document

In this documentation we give an overview our project, in which we analyze the outcomes of adversarial attacks on our ML model and then train our model on those adversarial examples so that it may detect and prevent adversarial attacks. We would first preprocess our benchmark dataset then train it on our model. After doing so, with the help of multiple adversarial attack we would train our dataset to generate an adversarial dataset. After that we would check accuracy of our dataset using different ML models. Then we implement a defense method to counter adversarial attacks by adversarial training method.

22 System architecture description

The system will be implemented as a main class that will include three sub classes, first class would contain preprocessed benchmark, the second class would have all the classifiers i.e Naïve Bayes, Decision Tree and KNN and the third would be responsible for GUI. The GUI would take user input to proceed with the user desired benchmark.

22.1 Section Overview

In our project, we are generating adversarial samples through adversarial training in Network Intrusion Detection Systems. We explore the use of Keras and TensorFlow library which is a deep learning framework to create our multi-layer perceptron DNN model. We would also use Cleverhans from which we call our attack from. For attack classification we used three classifiers. We used two benchmarks for our project.

22.2 General Constraints

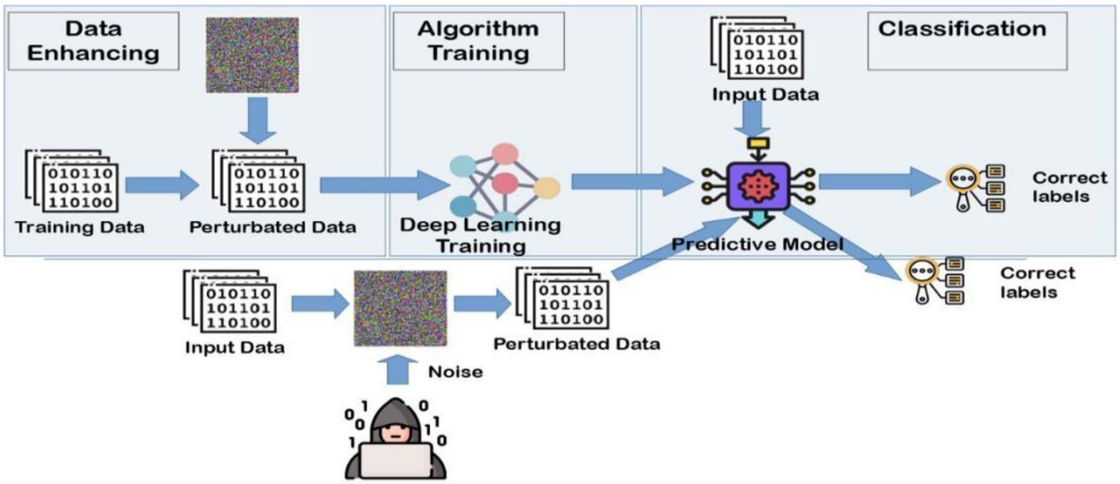
Detection and Prevention of Machine learning attacks in adversarial settings is basically made on Spyder (IDE) using the Python Frame-work, so no hardware will be required it’s completely based on software. It will be run by the users on their machine learning models. as a defense mechanism and a detection of adversarial attacks developed using adversarial training of adversarial samples which is used by users for the prevention of their machine learning model. The limitation of this defense mechanism is yet that the defense mechanism will not be able to prevent any attacks created in near future so this defense mechanism is basically for few adversarial attack that are known right now.

22.3 Data Design

We will have three datasets stored in local directory or called from cloud. These datasets would then be fetched upon user request.

22.4 Program Structure

The picture below illustrates the components that take part in our project and also the flow of data that enables us to achieve the completion of our tasks.



22.5 Alternatives Considered

Not Applicable

23 Detailed description of components

23.1 Section Overview

The system will be implemented as a main class that will include three sub classes, first class would contain preprocessed benchmark, the second class would have all the classifiers i.e Naïve Bayes, Decision Tree and KNN and the third would be responsible for GUI. Cleverhans is used for attack generation, matplotlib is used for visualization and sklearn library is used for calling classifiers.

23.2 Component and Detail

Component 1: CleverHans Library

Description

A use of CleverHans Library is the main component through which we will import our attacks so basically Cleverhans library is adversarial example library for constructing attacks ,benchmarking as well as building defenses.

Data Members

It's a type of library for constructing multiple attacks

Methods

1. We include this library in our project by writing "(' from cleverhans.tf2.attacks.fast_gradient_method import fast_gradient_method')" through which the fast gradient method will be implemented.
2. We Place this library at the top in our code for importing purpose
3. We will be using this library for multiple attacks.

Identification	CleverHans. Import from Github
Type	Library
Purpose	This library is used to import the usual cleverhans attack module to make an adversarial attack to the deep learning model.
Function	The library is used to call attacks
Subordinates	After classification we would perform FGSM attack using Cleverhans Attack. After testing with FGSM we would implement BIM, MIM, PGD and Madry elt
Dependencies	We should have TensorFlow 2 set in our environment to call the latest attacks from this library
Interfaces	After our dataset is trained and a classifier is implemented, the system would attack our dataset using Fast Gradient Sign Method. After the attack system would have a manipulated dataset with values varying from the original dataset. The system would then create a confusion matrix based on the manipulated dataset.
Resources	There is no usage of resources it's totally a software-based.

	https://arxiv.org/pdf/1706.06083
Processing	1. Import an attack from cleverhans library 2. Creating a function and initializing parameters required for the attacks. 3. Calling the function and giving model as an input parameter 4. Storing the manipulated dataset
Data	This library supports 3 frameworks: JAX, PyTorch, and TF2. We have install CleverHans using pip we can also clone this Github repository.

Component 2: Visualization of graphs

Description

A use of Matplotlib Library (matplotlib.pyplot as plt) is component to plot a graph. Through this library we will visualize the previously calculated AUC-ROC, F1 score and Accuracy.

Data Members

It's a type of library for visualizing graph and we are visualizing the results from the manipulated dataset of various attacks

Methods

1. We include this library in our project by writing "('matplotlib.pyplot as plt')" through which the visualization of various attacks and their accuracy will be implemented. 2. We Place this library at the top in our code for importing purpose
3. We will be using this library for visualization.

Identification	Visualization of Graphs
Type	Graph
Purpose	Purpose of Visualization is to visualize three accuracy measures imported from sklearn. We visualize the accuracies after different attacks and also after using different ML models for classification of the attacked dataset. Visualization helps us see what we cannot see using only numbers. It also shows us patterns
Function	Results for accuracy, f1 score and auc roc are plotted using this library
Subordinates	We will calculate the accuracy, f1 score and roc based on different epsilon of each attack and also after classification
Dependencies	We will be using Matplotlib library for graph visualization so the installation is given as: Pip install matplolib
Interfaces	Once the AUC-ROC is calculated then we will calculate the f1 score and accuracy of different attacks to compare the variation between attacks.

Resources	Accuracy metrics called via sklearn
Processing	There's no algorithm in this section
Data	We will visualize the results from the manipulated dataset of various attacks

Component 3: GUI

Description

A use of gui is to create user interface by using native elements for python application. We will use tkinter library, basically this is a module in python library which serves as an interface.

Data Members

It's a type of interface to communicate with user for easy interaction with machine

Methods

1. We include this Streamlit library in our project by writing "(import streamlit as st)". 2. We Place this library at the top in our code for importing purpose
3. We will be using this library for GUI purpose.

Identification	GUI
Type	Interface
Purpose	Help in the interaction between user and the system
Function	We would use this GUI library to give use options to select desired benchmarks or classifiers.
Subordinates	System shall display two options: - To select desired benchmark - To select the desired classifier and attack
Dependencies	We will be using streamlit library for GUI so the installation is given
Interfaces	The Interface will be GUI which shows that the system has two options to display Select the desired benchmark and select the desired classifier and then the results will be displayed.
Resources	Selectbox the process the user input out of options provided.
Processing	An instance of the library is used to perform functions provided by the library
Data	The first dropdown (benchmark). The second dropdown uses the (classifier) method to choose the classifier and third attack

Component 4: Benchmark

Description

The dataset we are using is KDD’99 and Kitsune. We would fetch and then preprocess them so that they can be proceeded to adversarial training

KDD’99: We are using KDD training dataset that contains 10% of original dataset. It has 41 features both numeric and categorical and 1 target class containing benign or malicious packet.

Kitsune: It contains benign and the malicious network traffic. Having 116 numeric features and 1 target column of

Data Members

Dataset which contains network traffic and labels

Methods

1. We would first preprocess our benchmark dataset then train it on our model. After doing so, with the help of multiple adversarial attack we would train our dataset to generate an adversarial dataset. After that we would check accuracy of our dataset using different ML models
2. When an attack is implemented, we save the result of that adversarial dataset

Identification	Benchmark
Type	Dataset
Purpose	The benchmark detail screen is to use for users to know about the benchmark in detail before selecting it.
Function	A user clicks on the benchmark so the screen will be redirected to the user to view the particular benchmark detail screen.
Subordinates	The following screen links to this screen • KDDCup’99 • Kitsune
Dependencies	There’s no dependencies of benchmark
Interfaces	We will be using GUI which will have an option for the user to select the desired dataset out of three options
Resources	if a user selects proper benchmark, then after selecting benchmark user will be able to select the desired classifier he wants to choose. The software we are using for this component is Spyder. IDE where we can use GUI.
Processing	The only type of processing is required is selecting benchmark Into button and navigating to other screens.
Data	Since we have 3 buttons in our benchmark form for selecting multiple benchmarks so each button takes in different types of data.and then select the classifier for defense mechanism.

Component 5: Classifier

Description

A classifier is an algorithm which maps the input data to specific category the classifiers we will be using

- KNN
- DECISION TREE
- NAÏVE BAYES

Data Members

After training our model we would use .predict() so that we can get the predictions based on probabilities. We would use an ANN(MLP) model for classification. After Classification from model we will implement other classifiers.

Methods

The system will have three different classification algorithms to create a classifier, Naïve Bayes, Decision trees and Naïve Bayes. This is because classification algorithms differ in the following criteria which leads to different results,

Classifier would then classify the manipulated dataset.

Identification	Classifier
Type	Machine Learning Classifier
Purpose	The Classifier detail screen is to use for users to know about the Classifier in detail before selecting it.
Function	A user clicks on the Classifier so the screen will be redirected to the user to view the particular Classifier detail screen.
Subordinates	The following screen links to this screen • KNN • DECISION TREE • NAÏVE BAYES
Dependencies	The system shall then classify our model. – Once the model is trained, the system will predict it's classes by using .predict() which is a built-in API for classification
Interfaces	We will be using GUI which will have an option for the user to select the desired classifier out of three options
Resources	if a user selects proper benchmark, then after selecting classifier user will be able to select the desired classifier he wants to choose.
Processing	The only type of processing is required is selecting classifier Into button and navigating to other screens.
Data	Since we have 3 buttons in our classifier form for selecting multiple classifier so each button takes in different types of data.and then results will be visible.

24 User Interface Design

24.1 Section Overview

This is the main file that uses all the defined modules in data, classifiers and attacks packages to run the software. It is also responsible of creating the command line GUI (Graphical User interface) and specifying the software interactions with the user actions such as, reading user input and presenting results.

24.2 Interface Design Rules

We will be using GUI which will have an option for the user to select the desired dataset out of three options the GUI will have an option to select a classifier. Based on the user selection defense mechanism would be trained and tested on benchmark and the classifier user selected

24.3 GUI Components

Graphical User Interface
Helps with the communication of user input with the system

24.4 Detailed Description

First GUI will show a window that will ask to select a benchmark out of three options. Once the benchmark is chosen another window will pop up for classifier that the user want from KNN, Decision Tree and Naïve Bayes. After that our system will take all the parameters and implement all attacks and performing adversarial training for the prevention and detection of adversarial attacks in future use.

25 Reuse and relationships to other products

<u>How reuse is playing a role in your product implementation:</u>
We are reusing using several cleverhans attacks which are used by many others in there implementation of an IDS against adversarial attacks. But most of those IDS provides protection from an adversarial image and nor network traffic as we are implementing. Work and research are being conducted on this topic.

26 Design decisions and tradeoffs

We are implementing several attacks and also giving classification options to the users

27 Pseudocode for components

27.1 Cleverhans Attack Module

1. Import an attack from cleverhans library
2. Make logits with input and output layer
3. Make a function and call the attack function from library and initialize required parameters
4. Call the attack function from our class and then provide it with the tensorflow model and x test.

27.2 Visualization

5. Import matplotlib library
6. Give title to the graph
7. Give label to the graph
8. Plot the graph by giving values of x and y axis and also design structure

27.3 GUI

9. Import streamlit
10. Call the required widget from the library

27.4 Benchmark

11. Fetch the required benchmark
12. Preprocess the benchmark
    a. Data Cleaning
    b. Standardaization
    c. Label Encoding
    d. One hot Encoding
13. Perform adversarial training

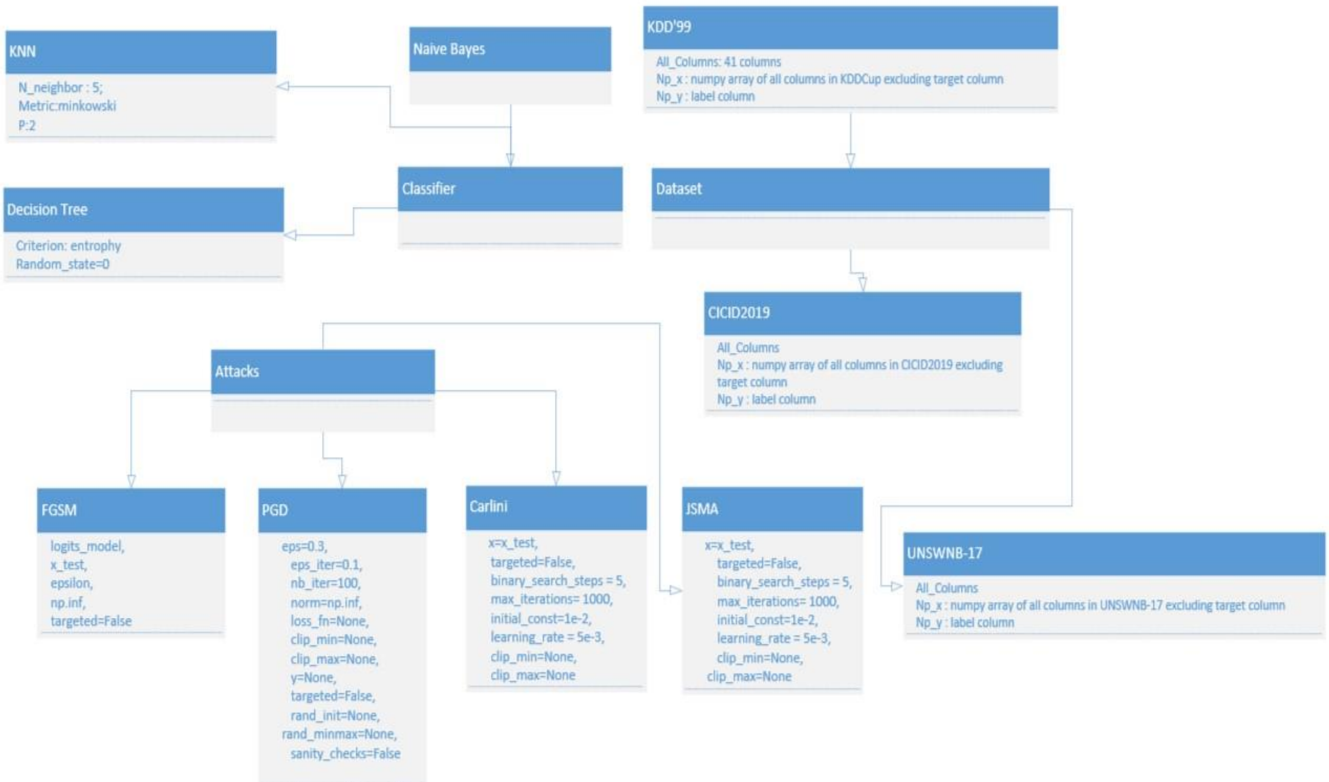
27.5 Classifier

1. Import classifier from sklearn library
2. Call the classifier and initialize its parameters
3. Compiliation
4. Predict x test
5. Check accuracy
6. Predict adversarial attacked dataset
7. Check accuracy, f1 score and roc auc score.

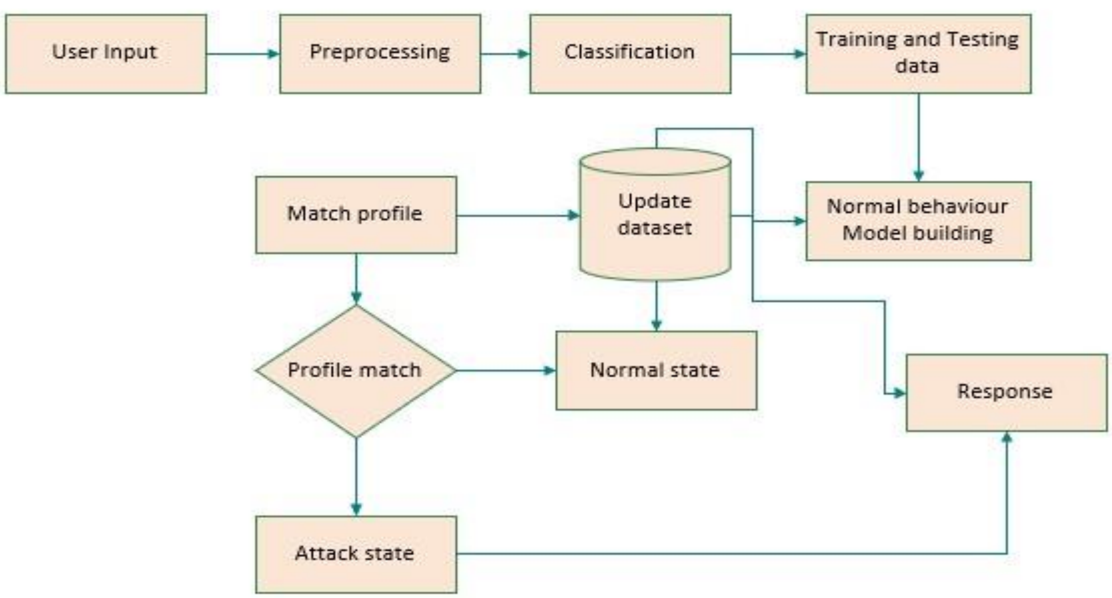
28 Appendices

28.1 Class Diagram

28.2 Object Diagram



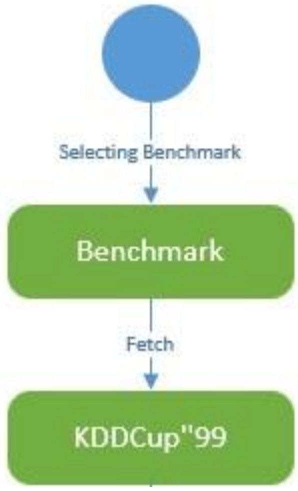
28.3 Statechart Diagram



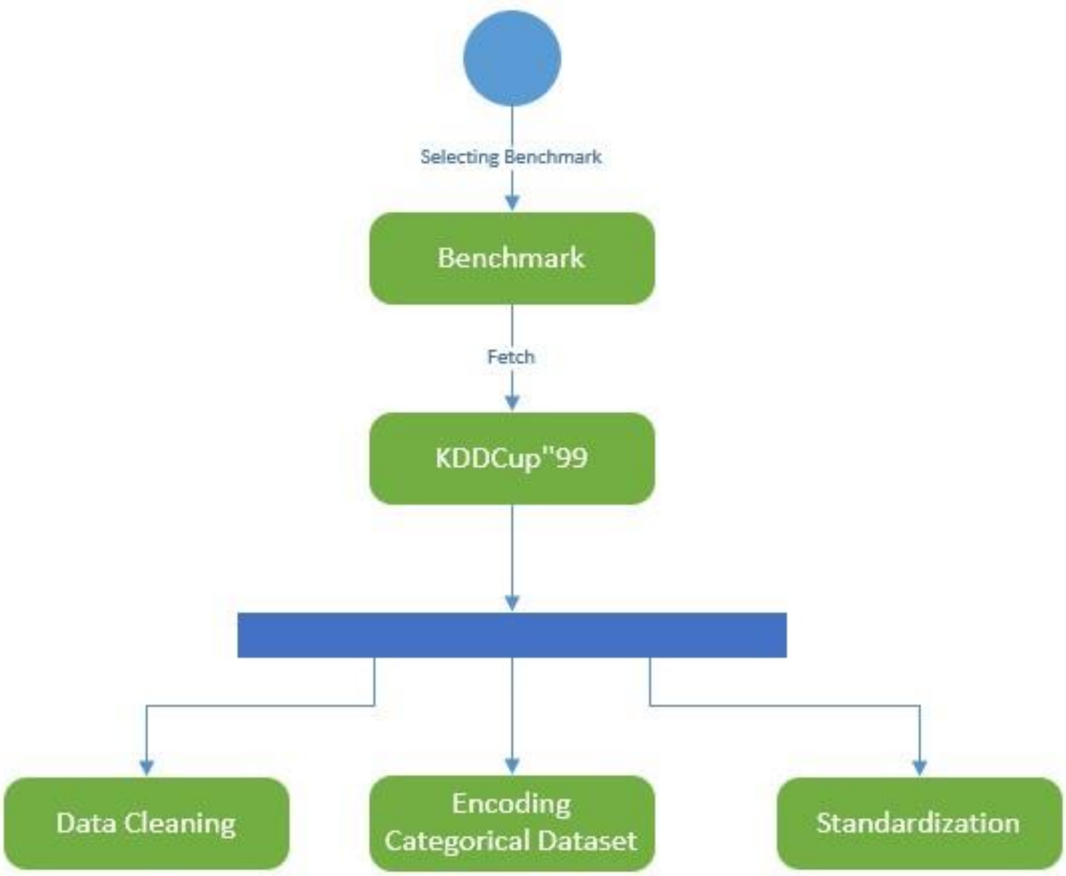
28.4 Activity Diagram

28.4.1FYP 1:

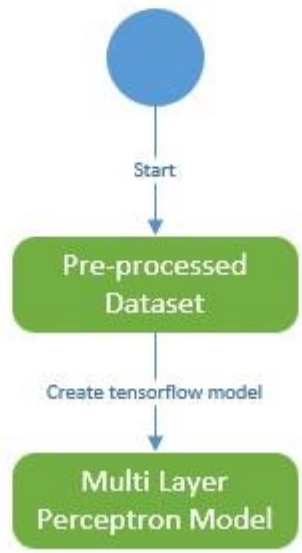
- Selecting Appropriate Benchmark



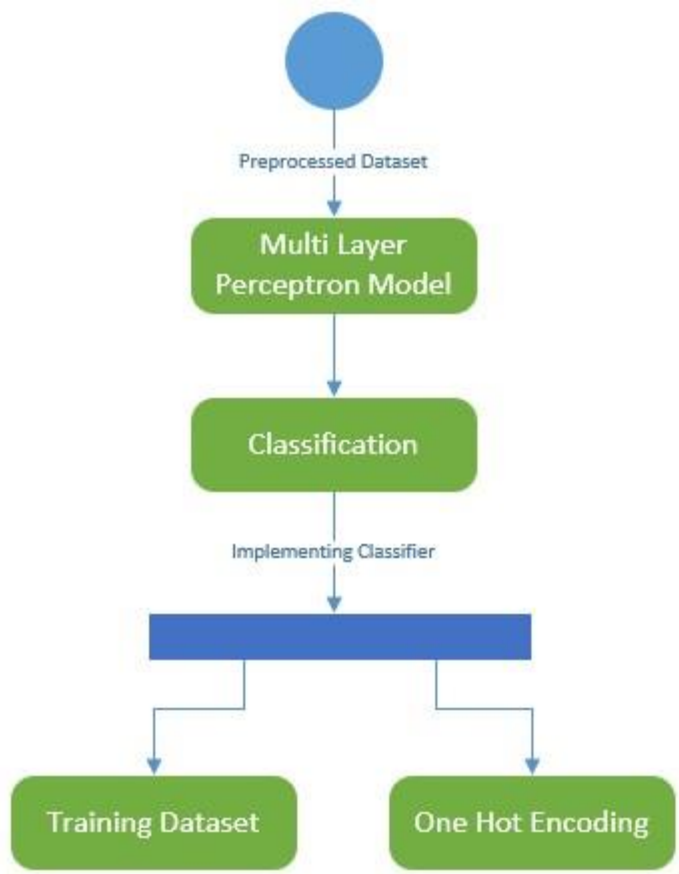
- Preprocessing



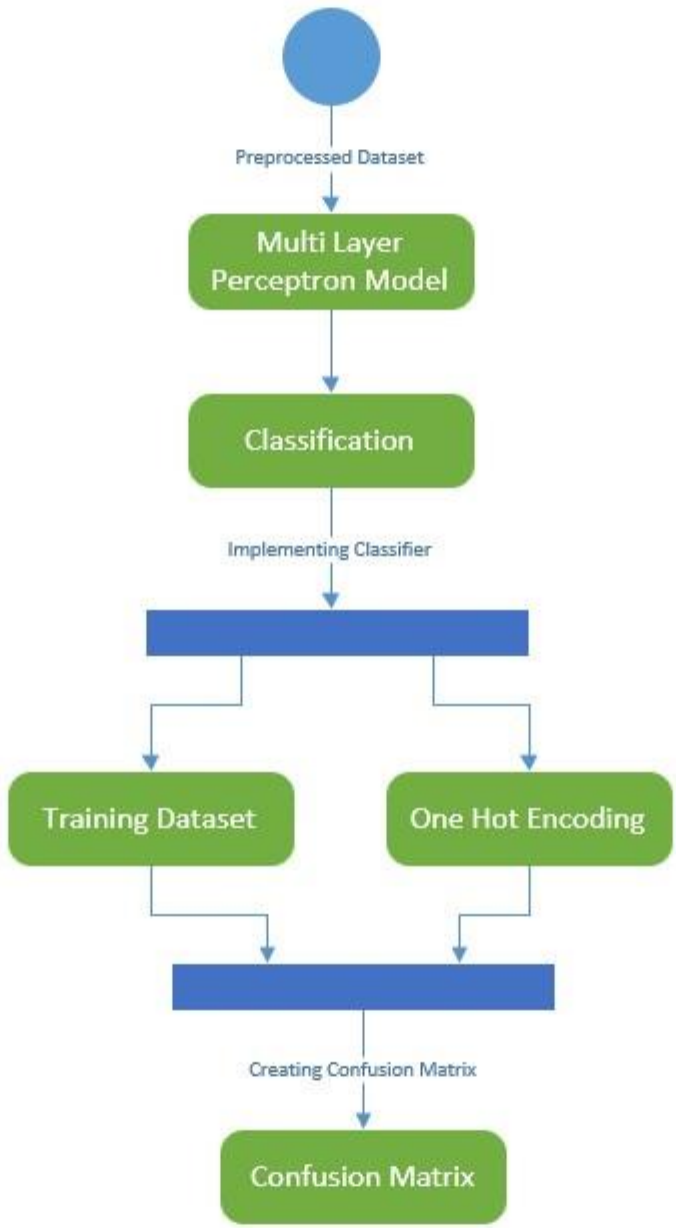
- Create Tensorflow Based Model



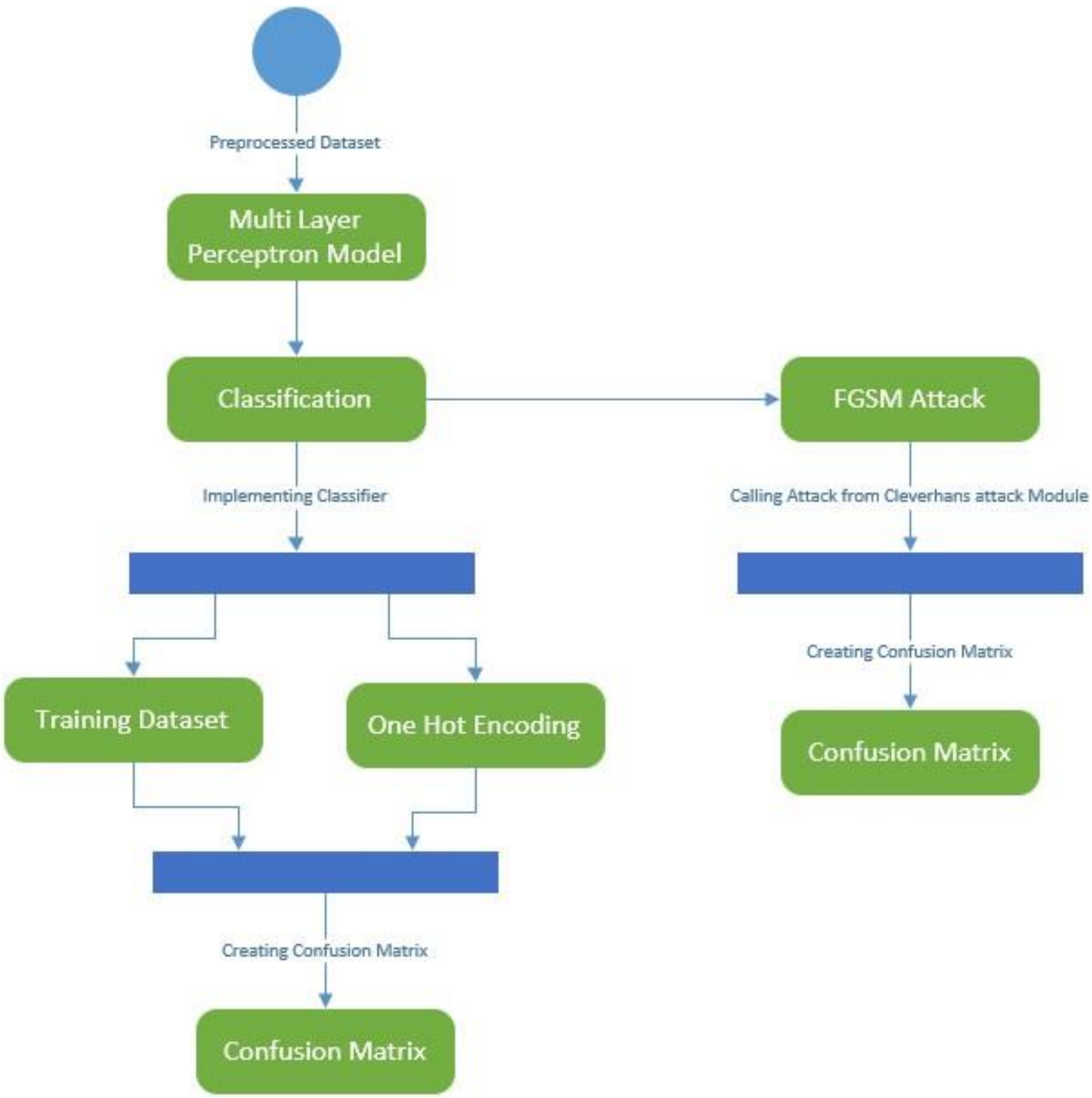
- Implementing Classifiers



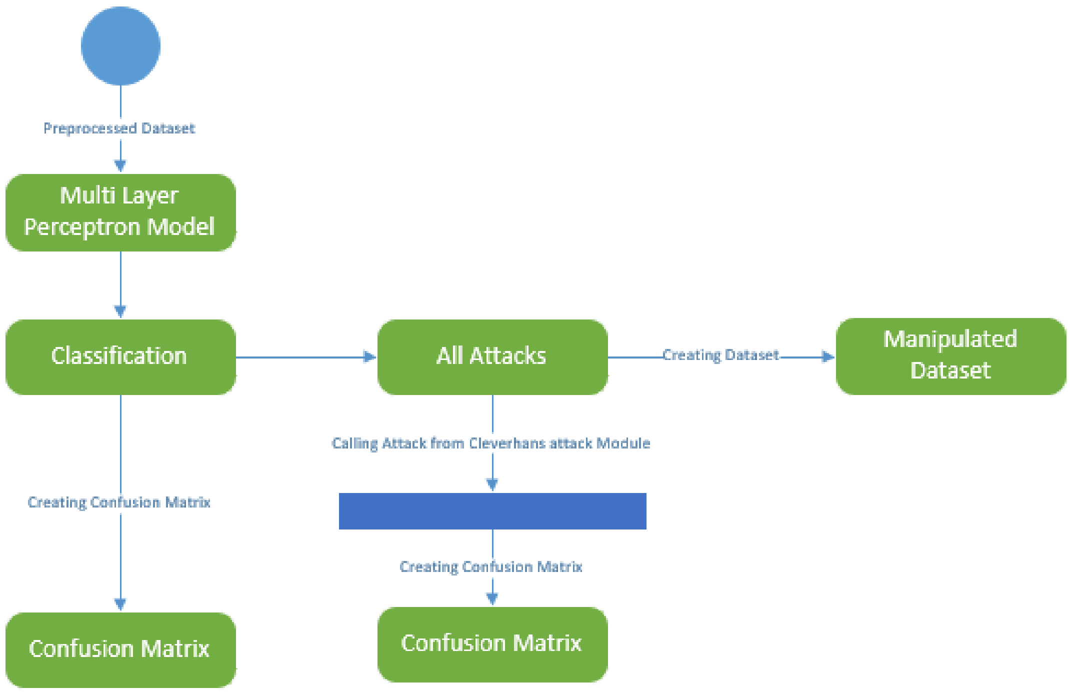
- Training our Model



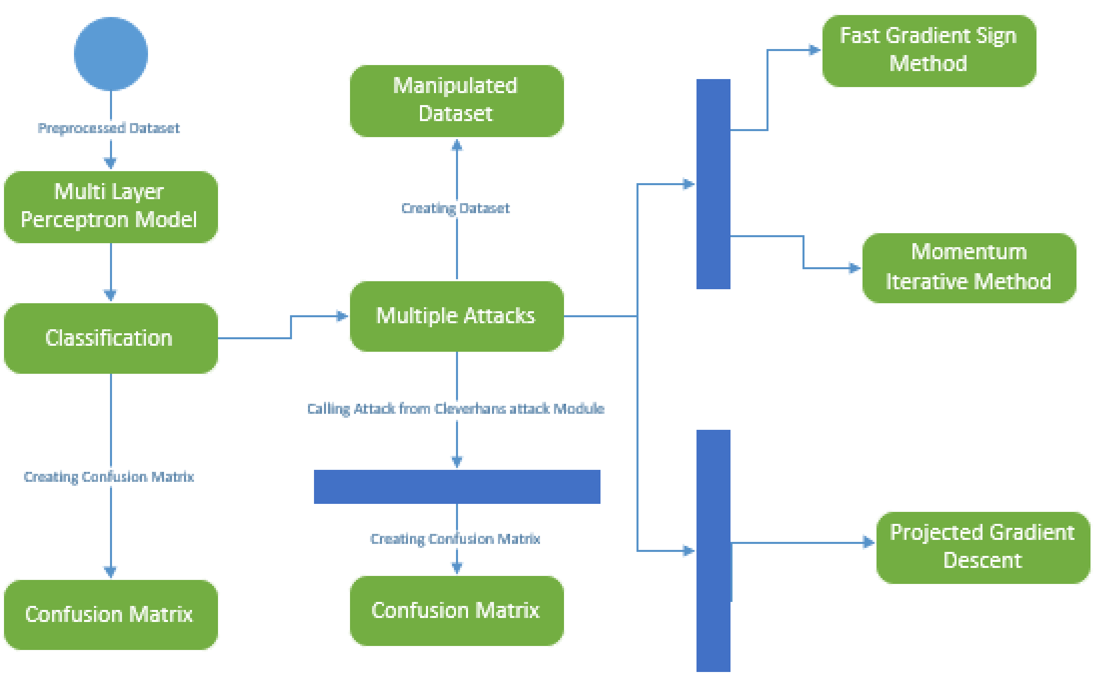
- Implementing FGSM using Cleverhans Attack Module



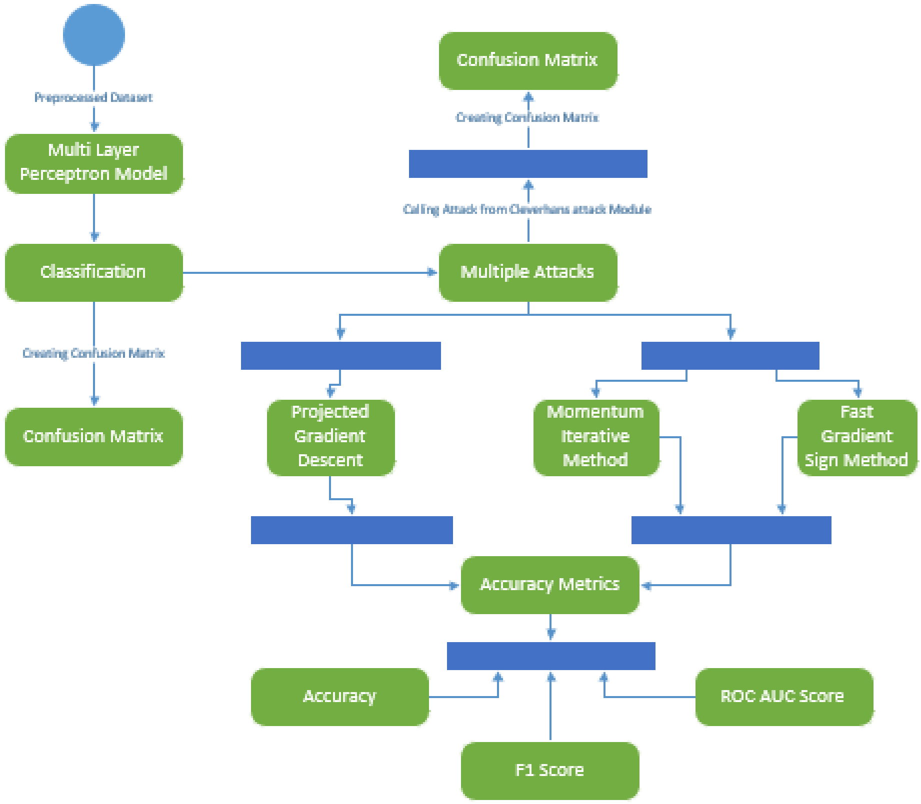
- Creating a dataset with adversarial example



- Implementing multiple attack from Clever Hans Attack Module

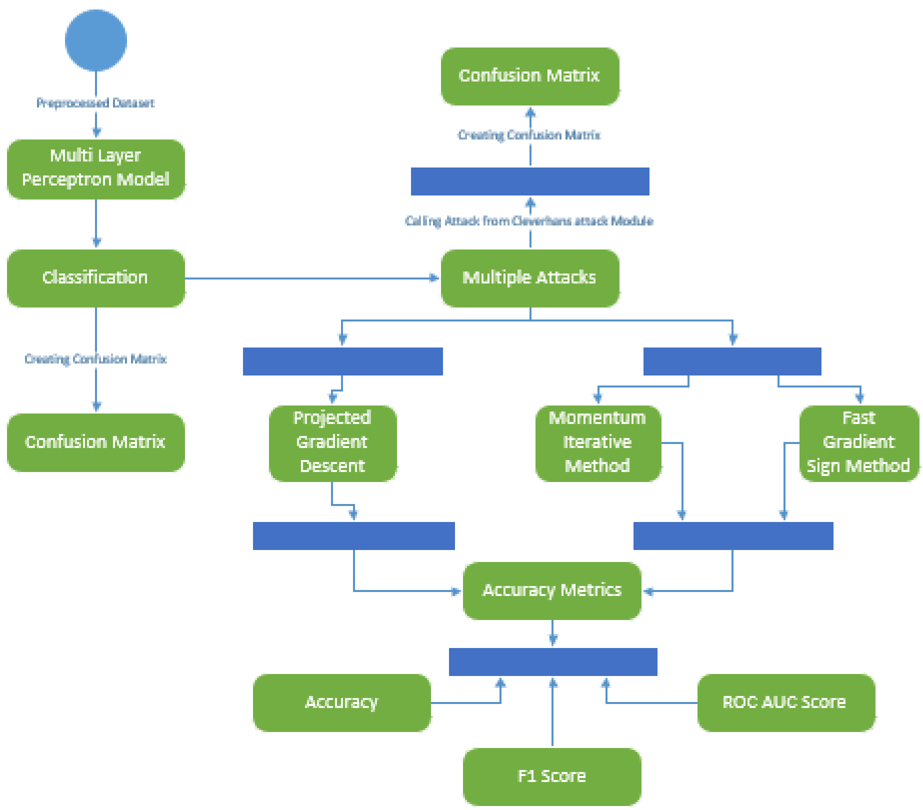


Checking accuracy of our model with different attacks

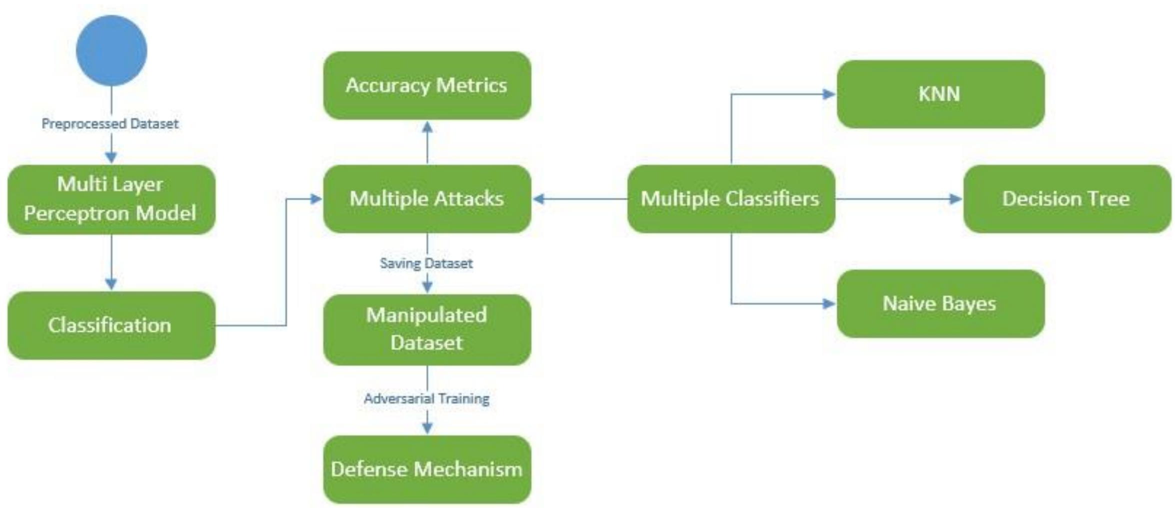


28.4.2 FYP 2:

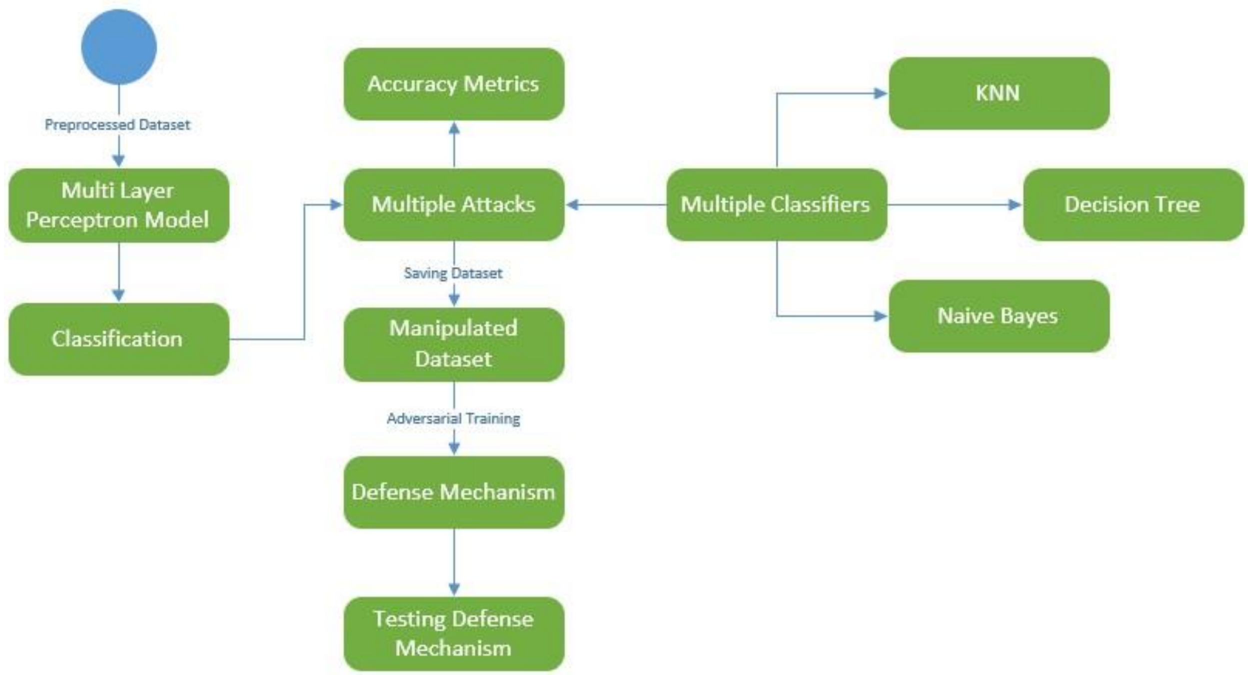
Using different ML models to check accuracy of different attack



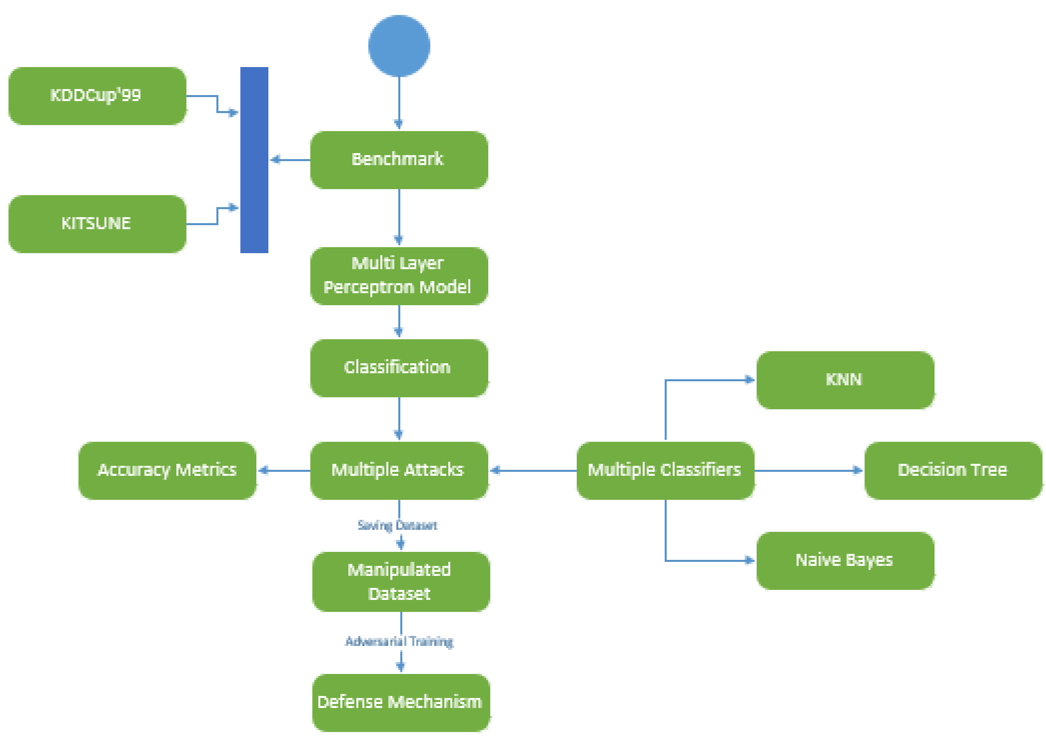
- Implementing Defense Method (Adversarial Training)



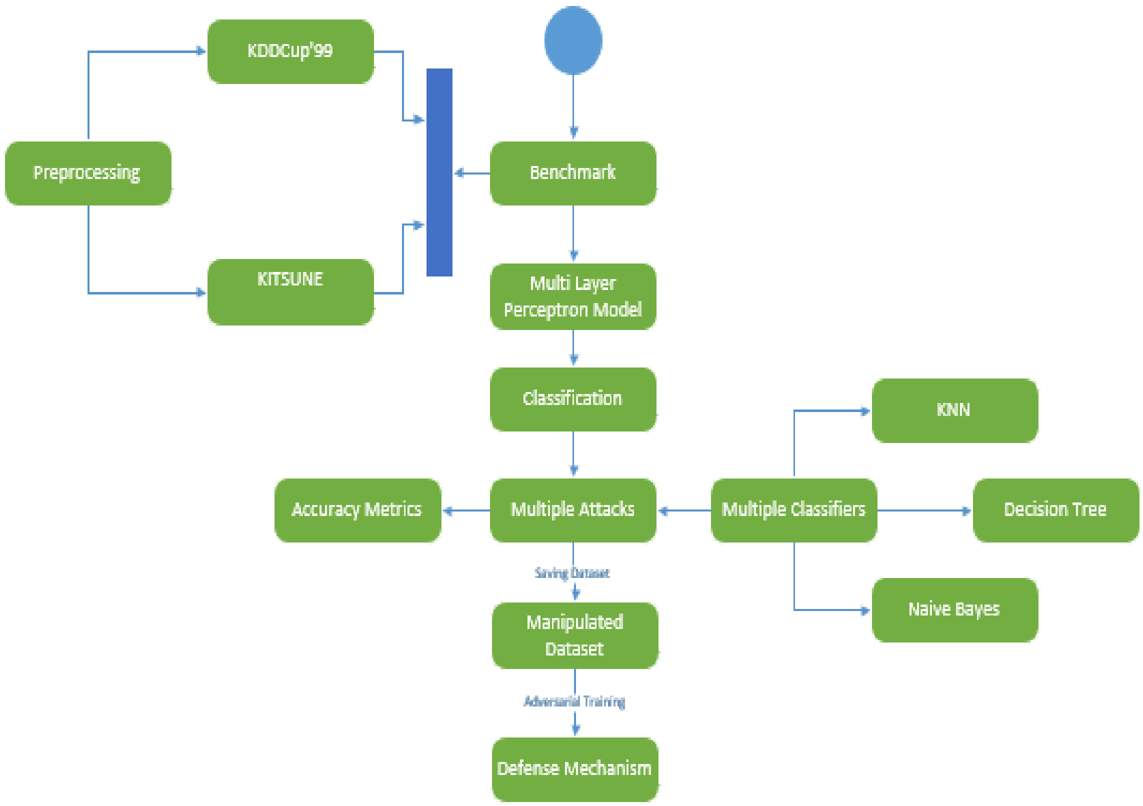
Testing Defense Method



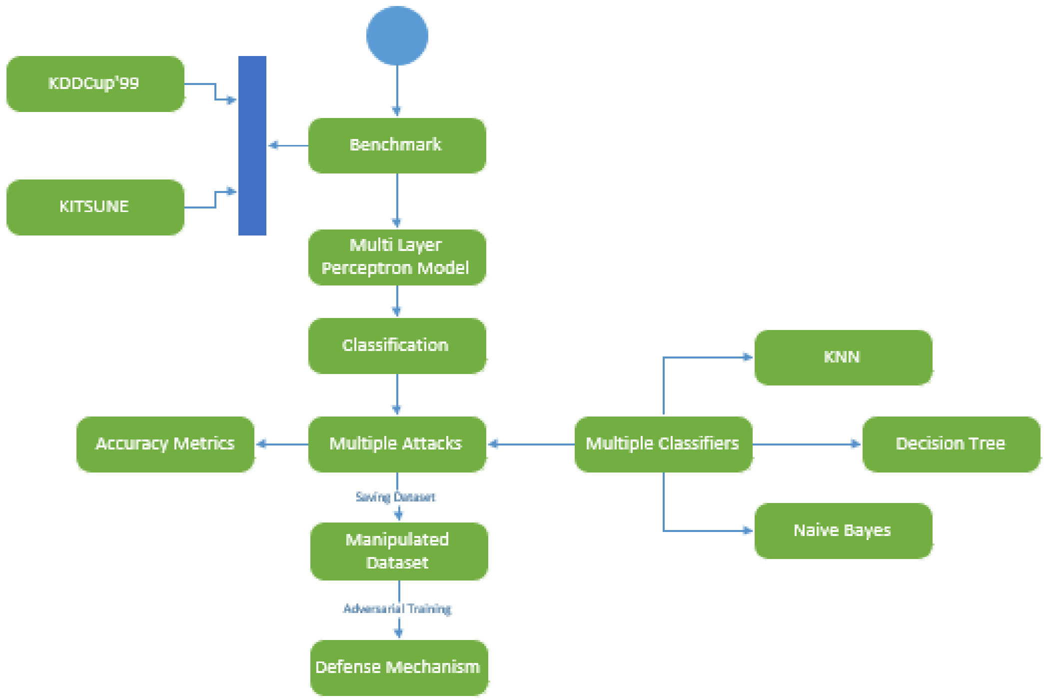
- Using multiple benchmarks for our model



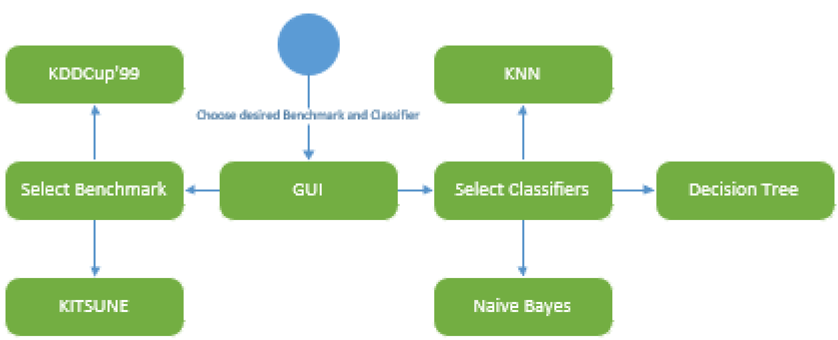
Preprocessing different benchmarks



Testing results given by different benchmarks



- Working on GUI



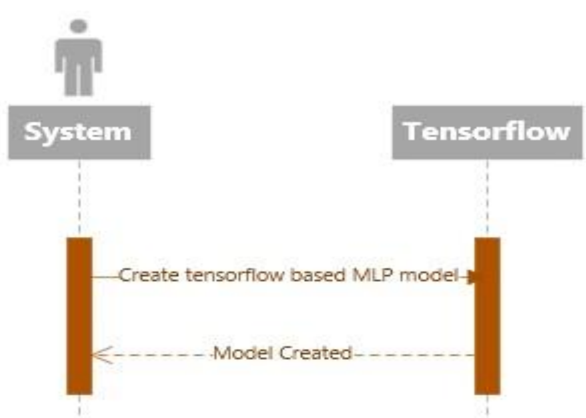
28.5 Sequence Diagram

28.5.1FYP 1:

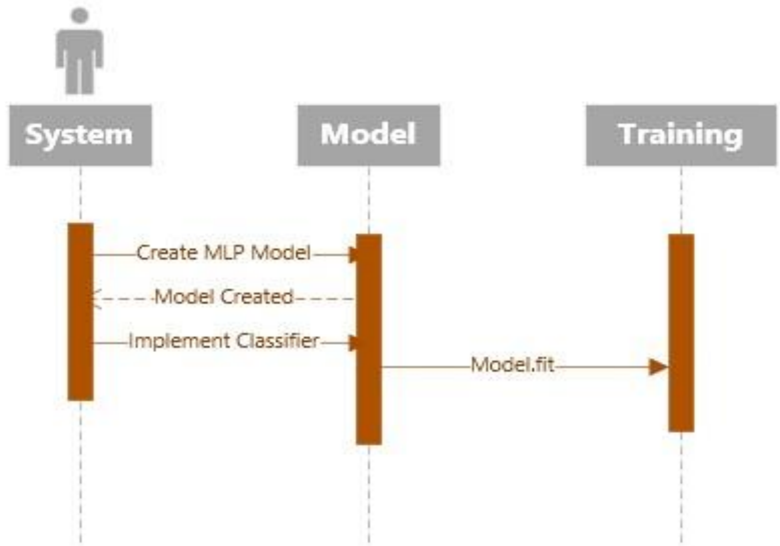
Appropriate Benchmark

Preprocessing

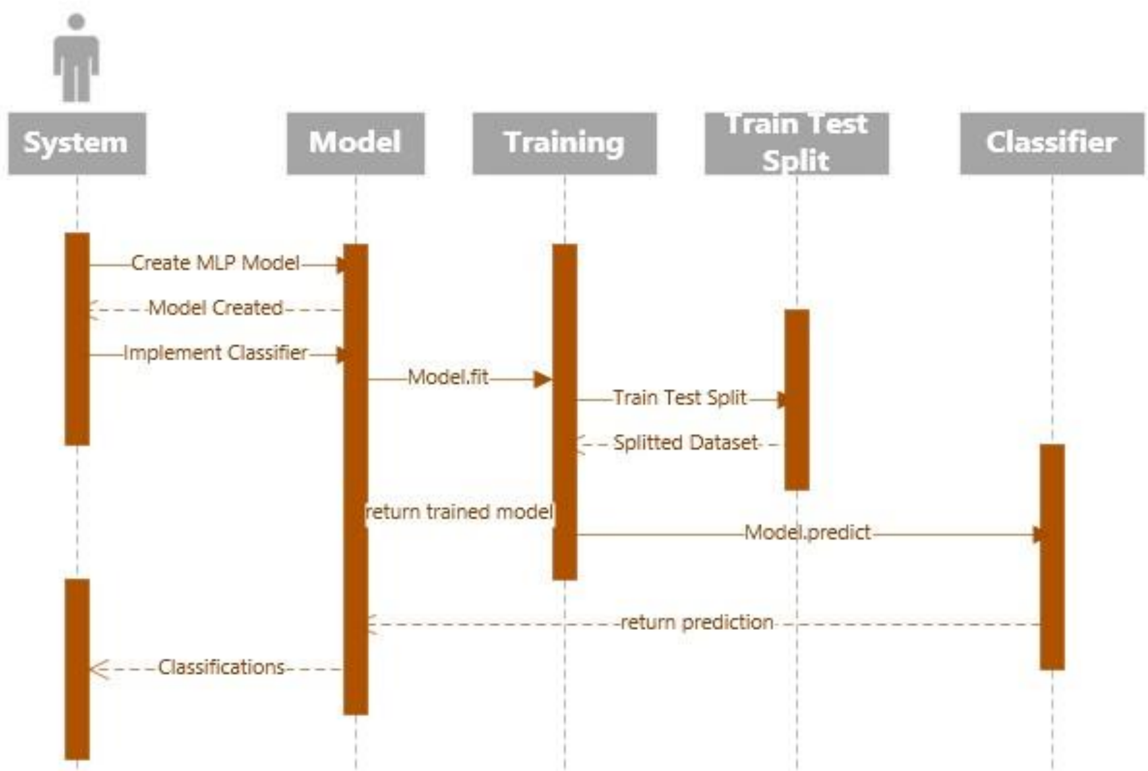
Create Tensorflow Based Model



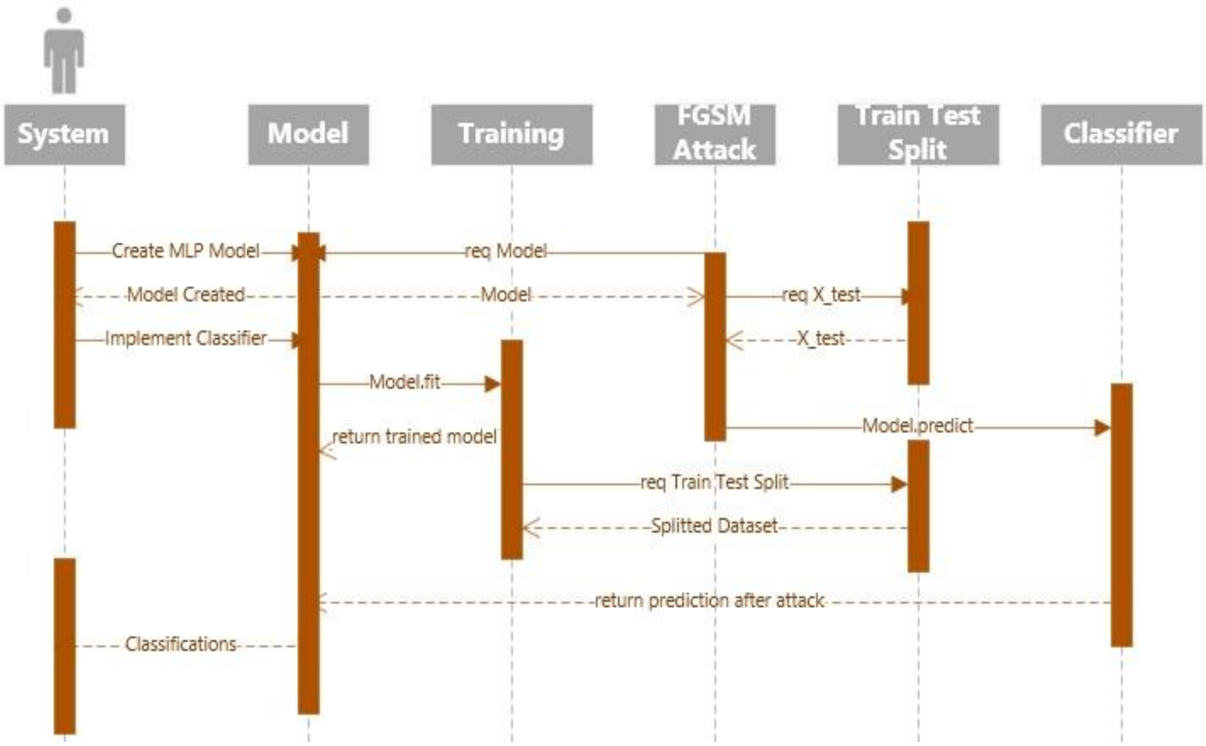
- **Implementing Classifiers**



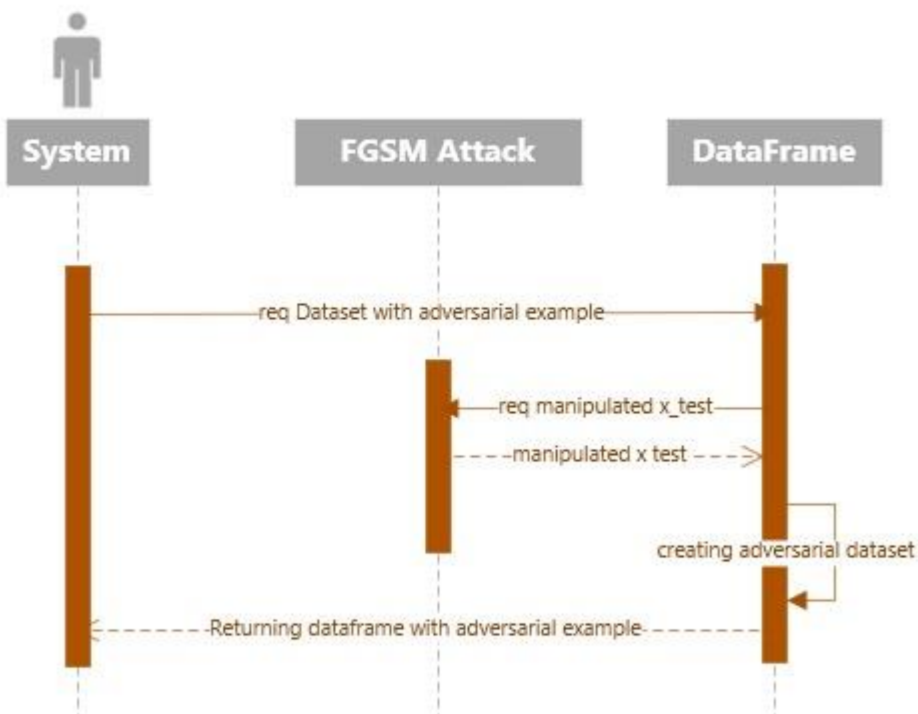
Training our Model



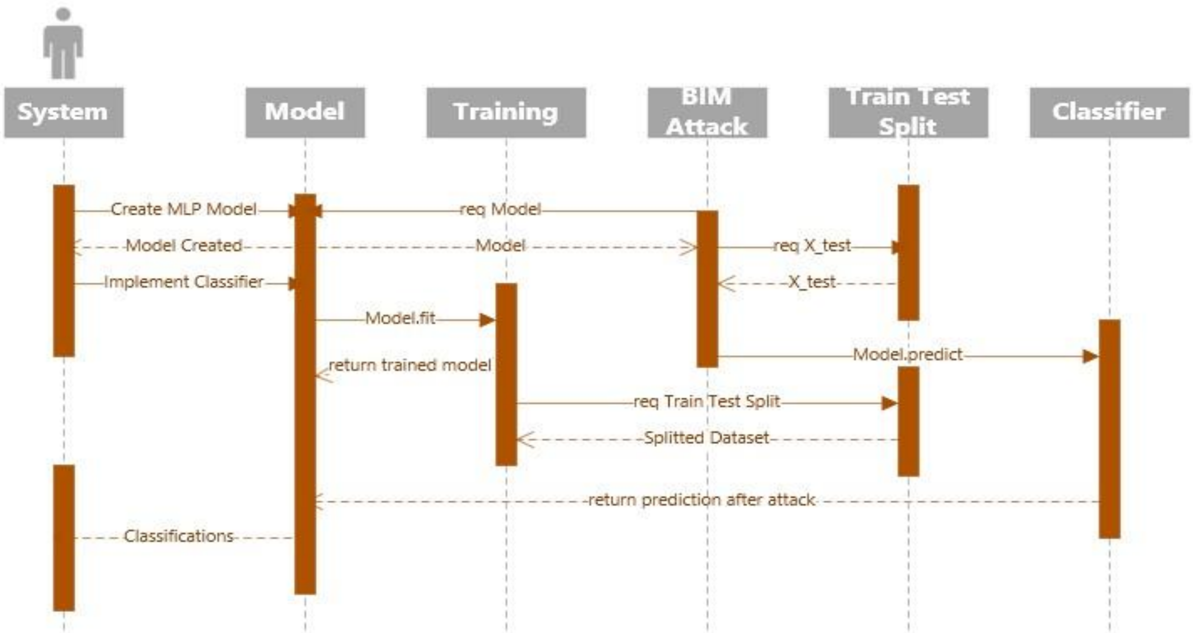
- **Implementing FGSM using Cleverhans Attack Module**

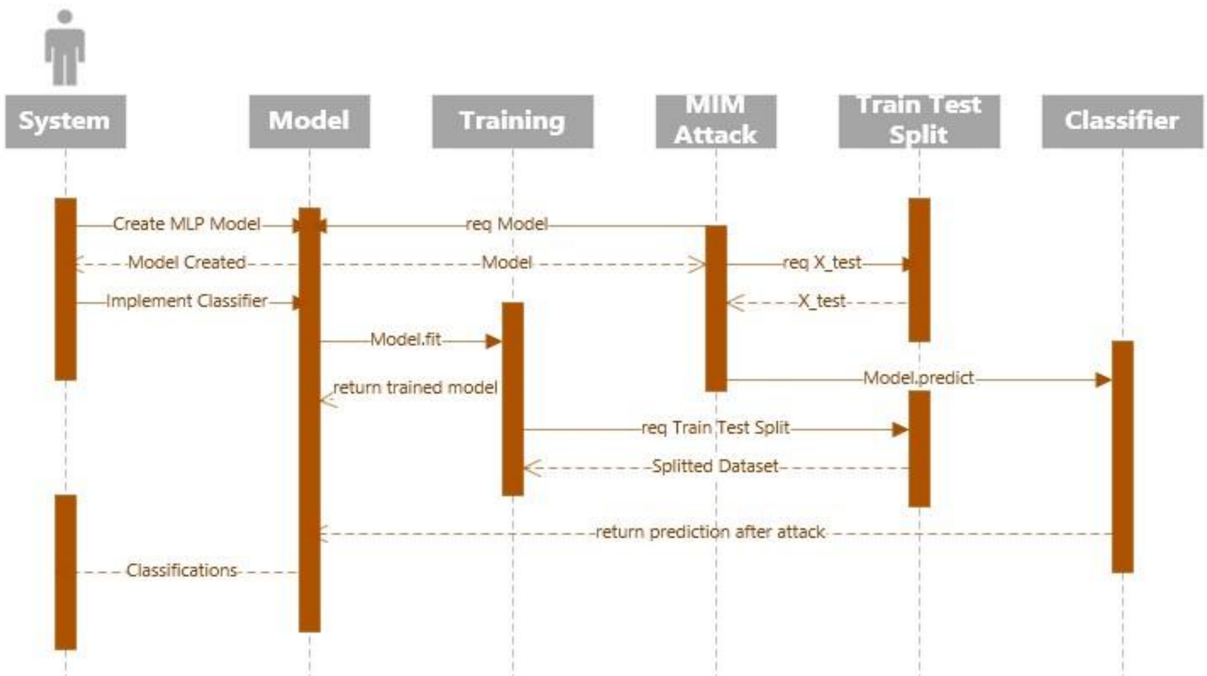


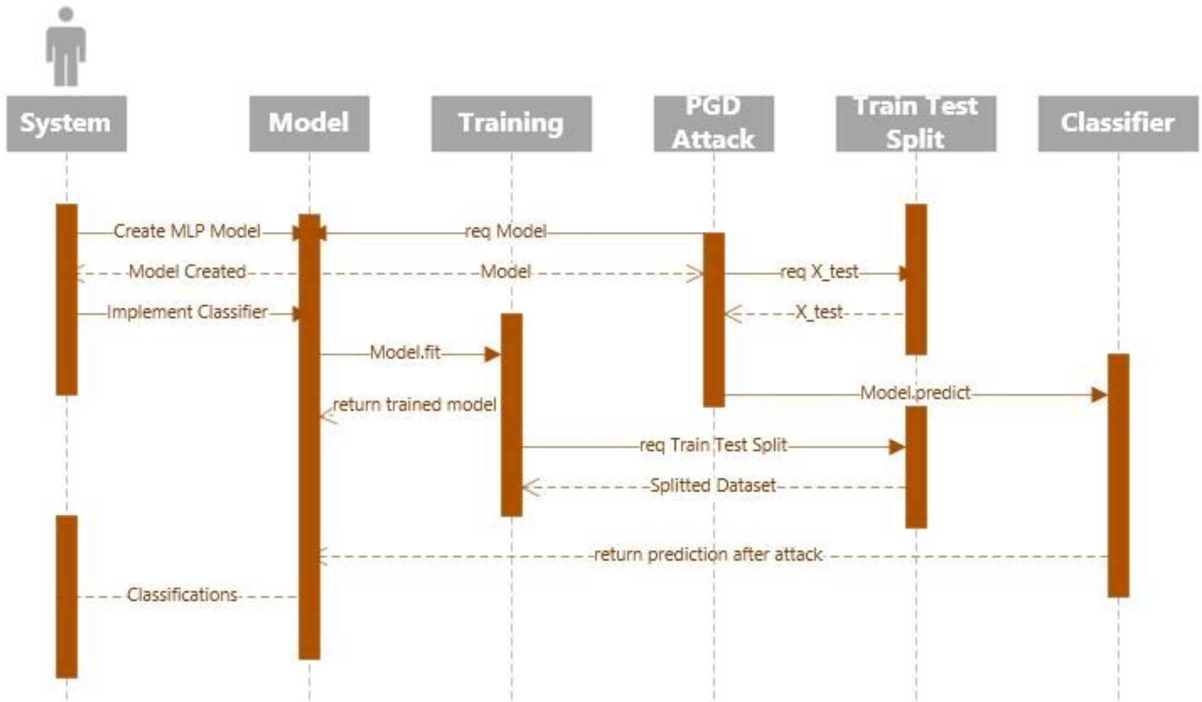
Creating a dataset with adversarial example

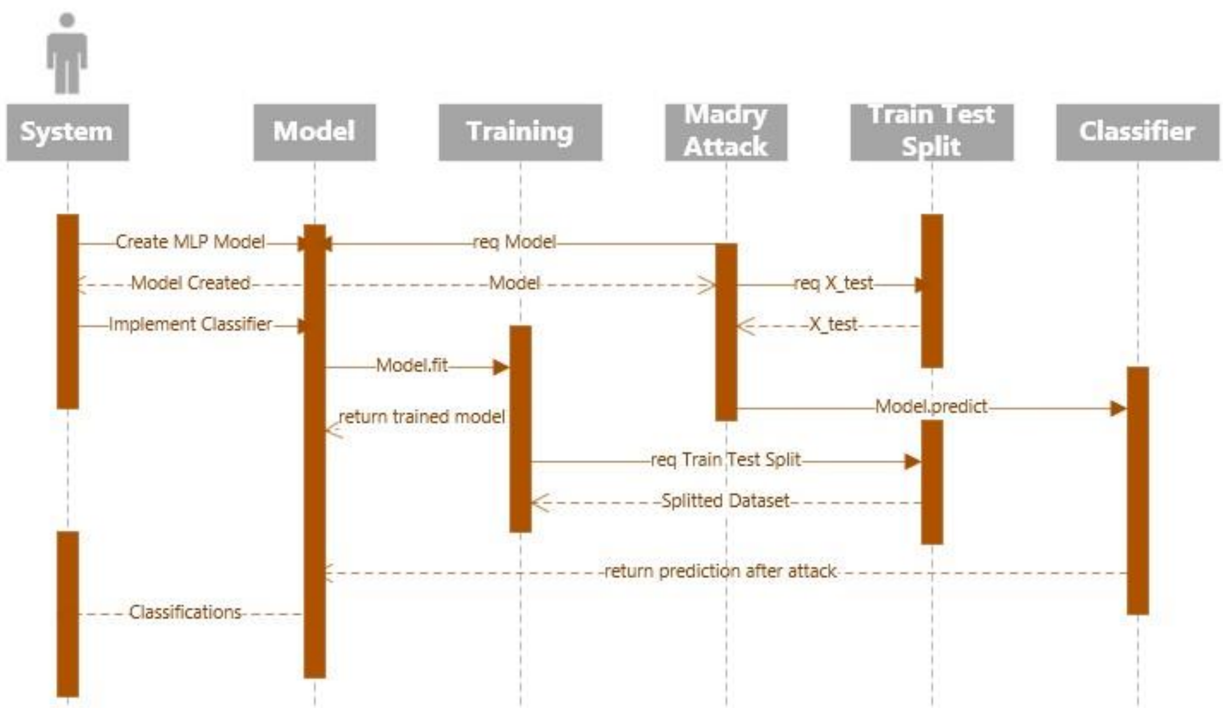


- **Implementing multiple attack from Clever Hans Attack Module**

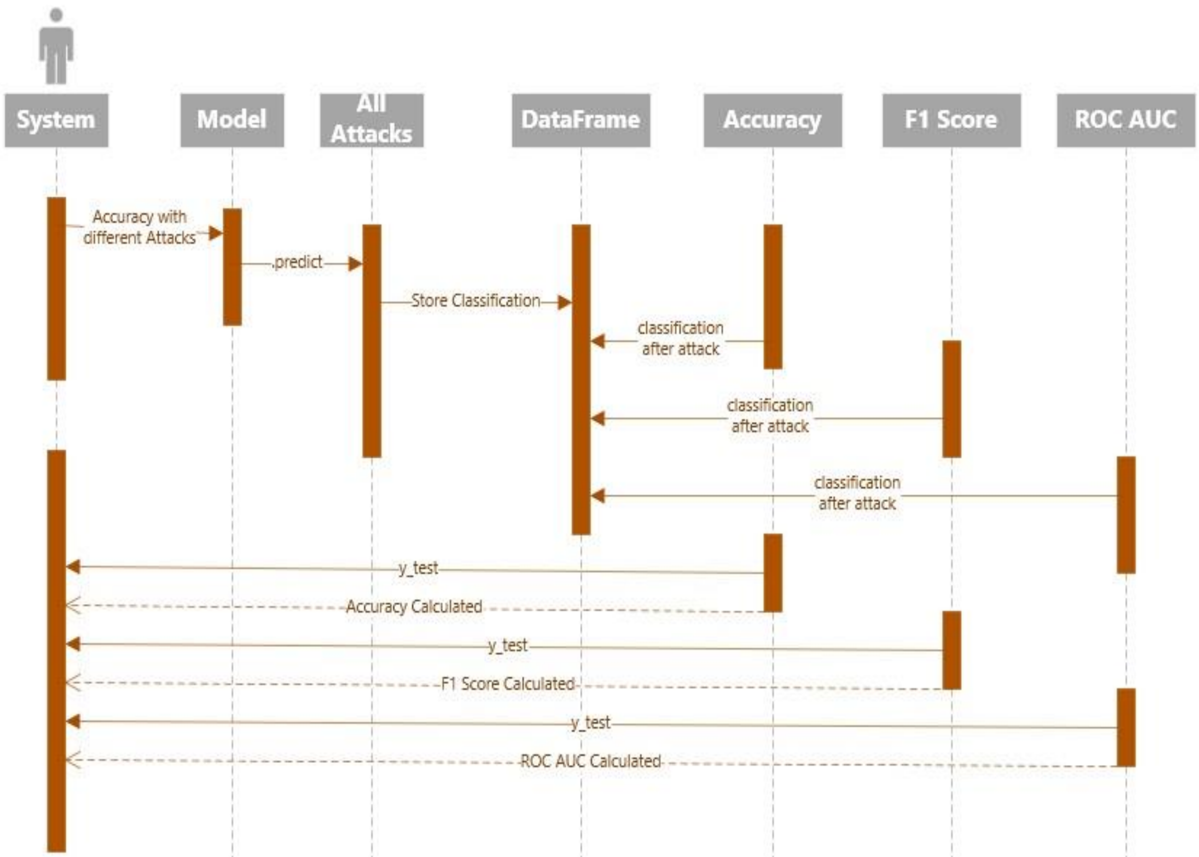





System
Model
Training
PGD Attack
Train Test Split
Classifier
Create MLP Model
req Model
Model Created
Model
req X_test
Implement Classifier
X_test
Model.fit
Model.predict
return trained model
req Train Test Split
Splitted Dataset
return prediction after attack
Classifications

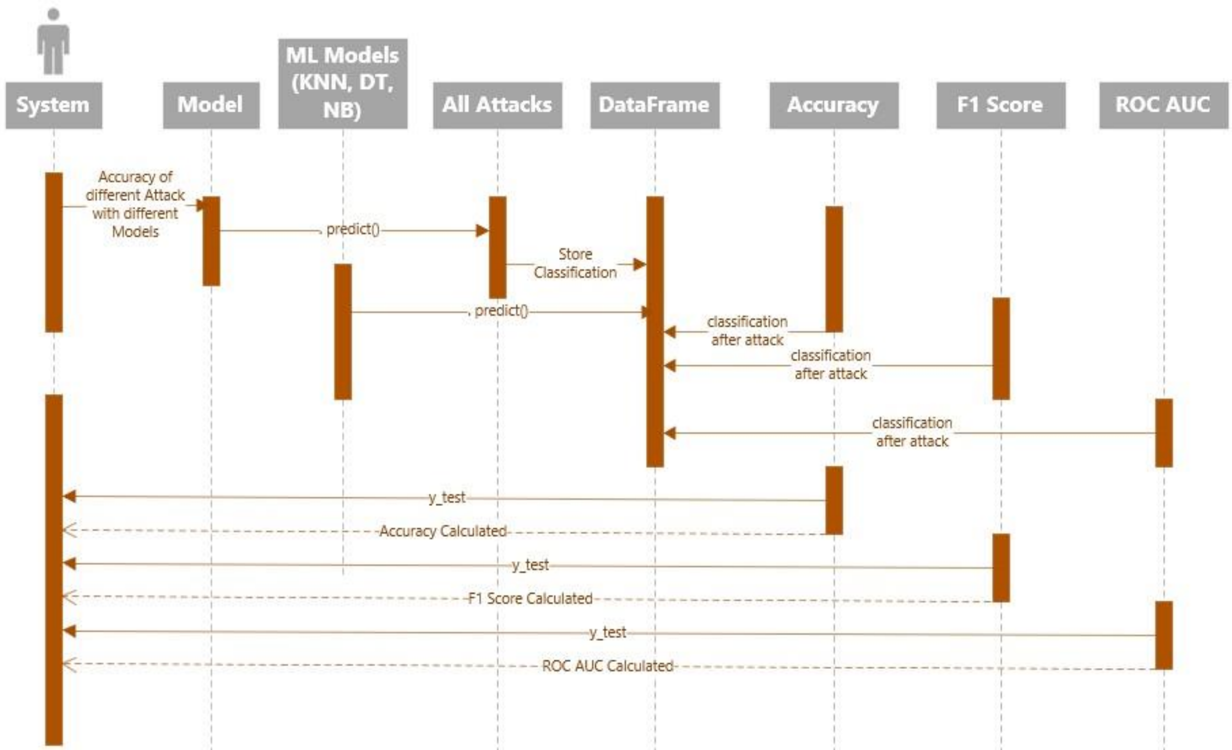

System
Model
Training
Madry Attack
Train Test Split
Classifier
Create MLP Model
req Model
Model Created
Model
req X_test
Implement Classifier
X_test
Model.fit
Model.predict
return trained model
req Train Test Split
Splitted Dataset
return prediction after attack
Classifications

Checking accuracy of our model with different attacks

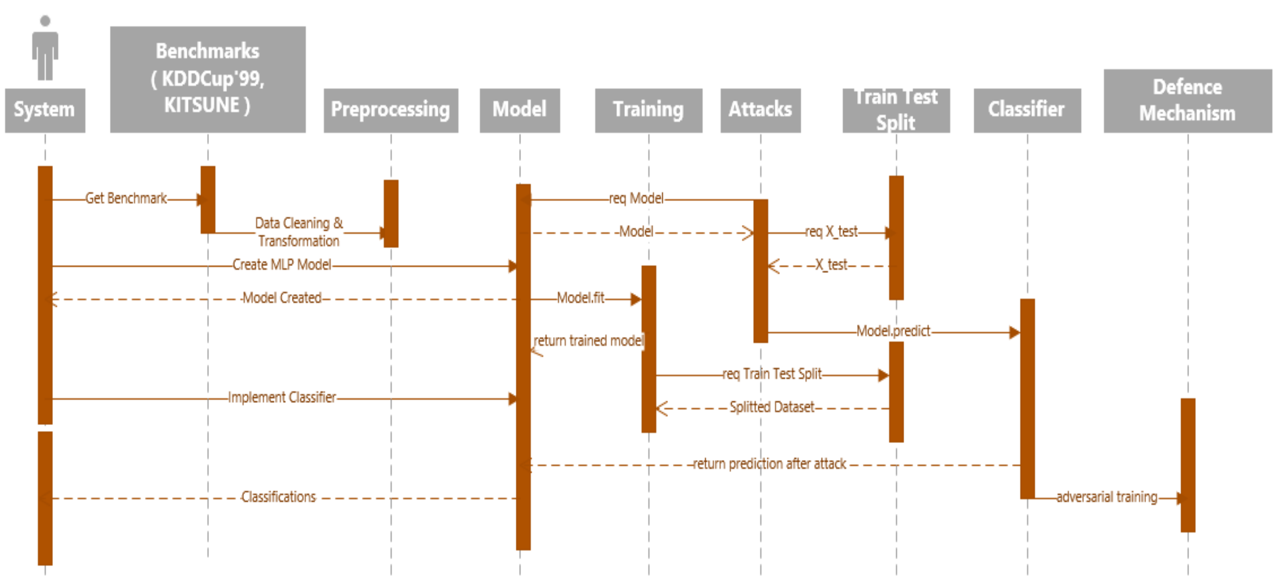


28.5.2FYP 2:

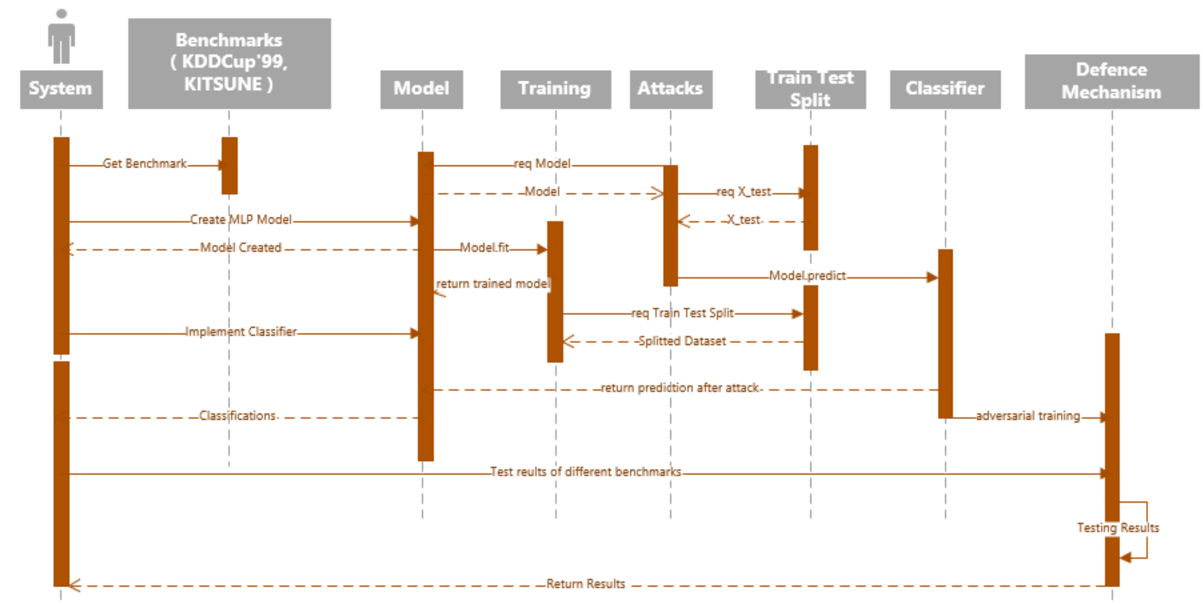
- **Using different ML models to check accuracy of different attack**



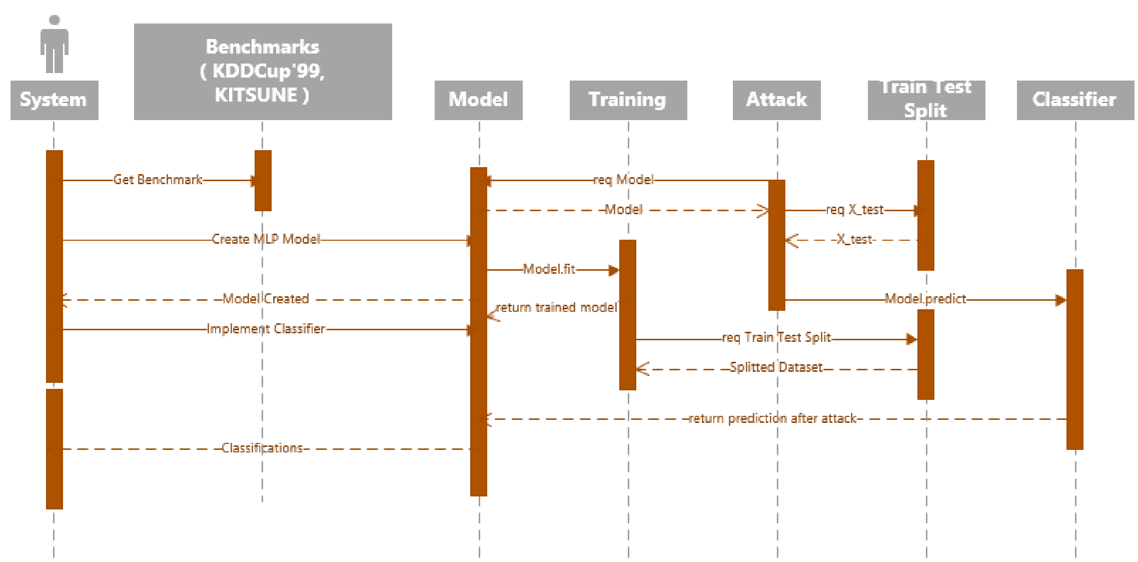
- **Implementing Defense Method (Adversarial Training)**



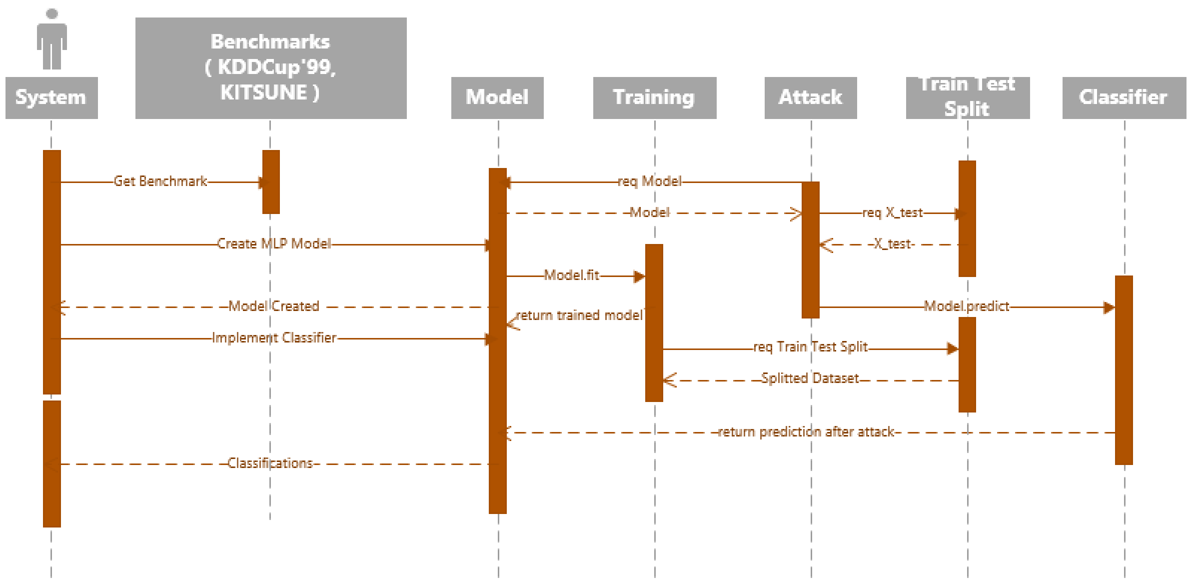
- **Testing Defense Method**



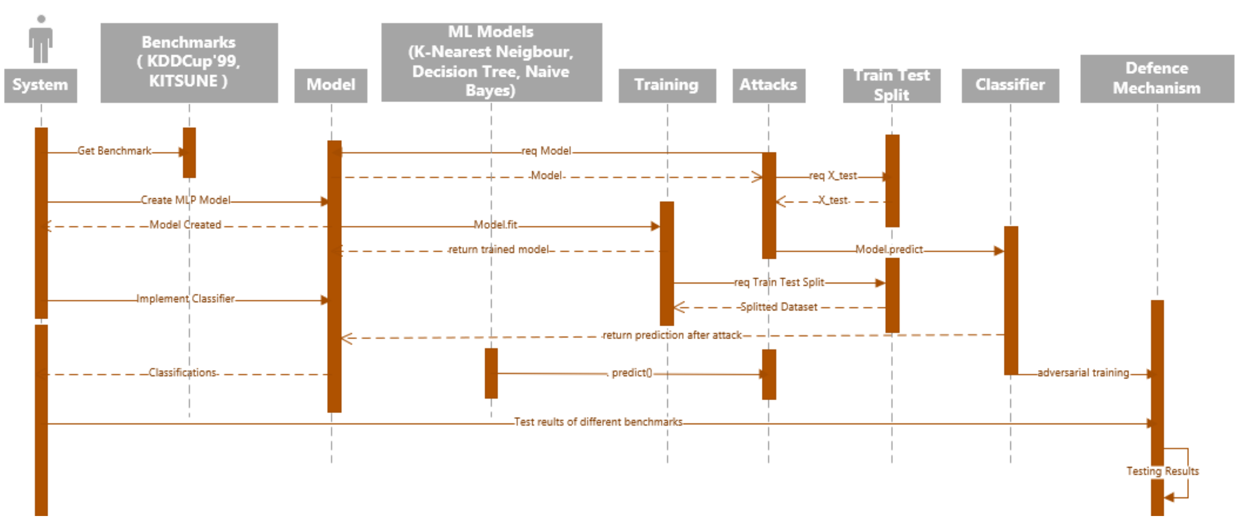
- **Using multiple benchmarks for our model**



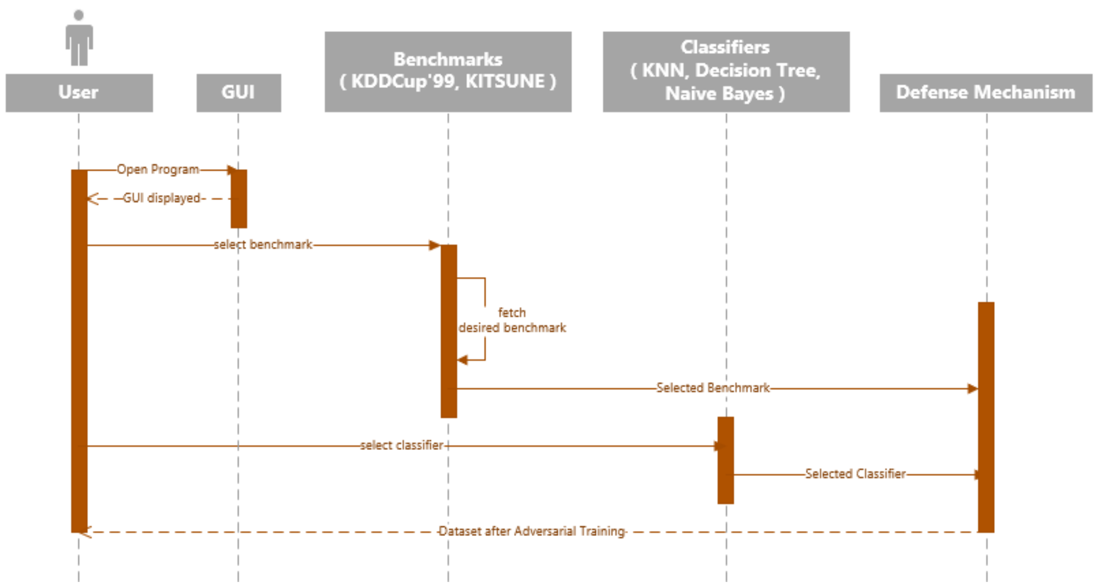
Preprocessing different benchmarks



Testing results given by different benchmarks



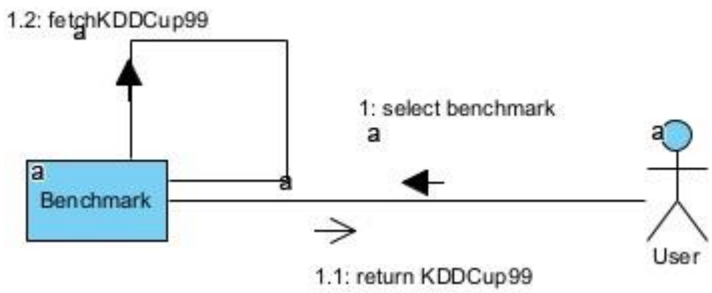
Working on GUI



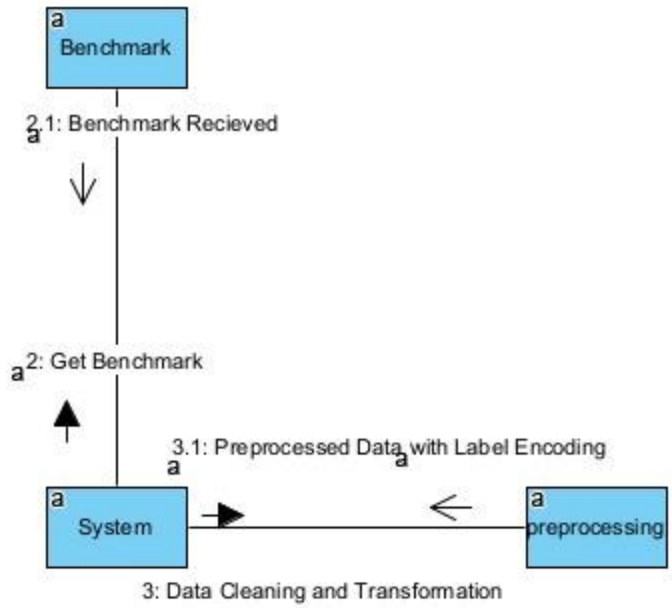
28.6 Collaboration Diagram

28.6.1 FYP 1:

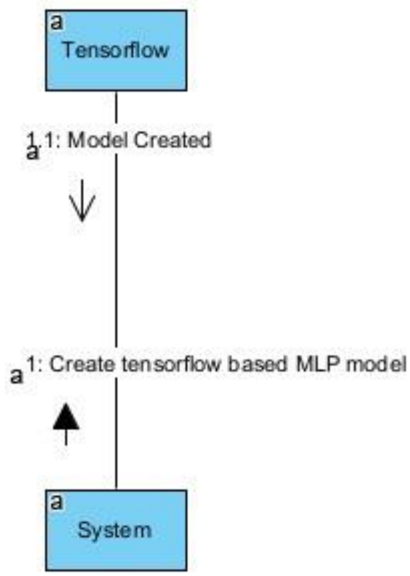
- **Selecting Appropriate Benchmark**



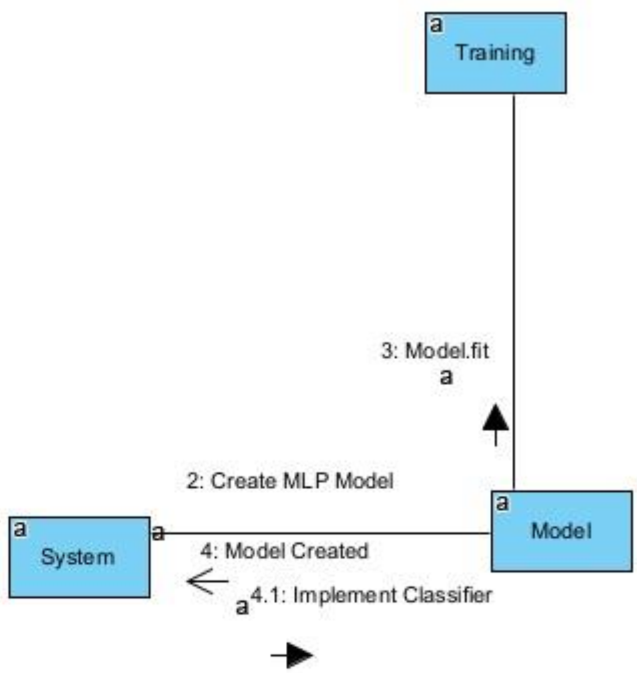
- **Preprocessing**



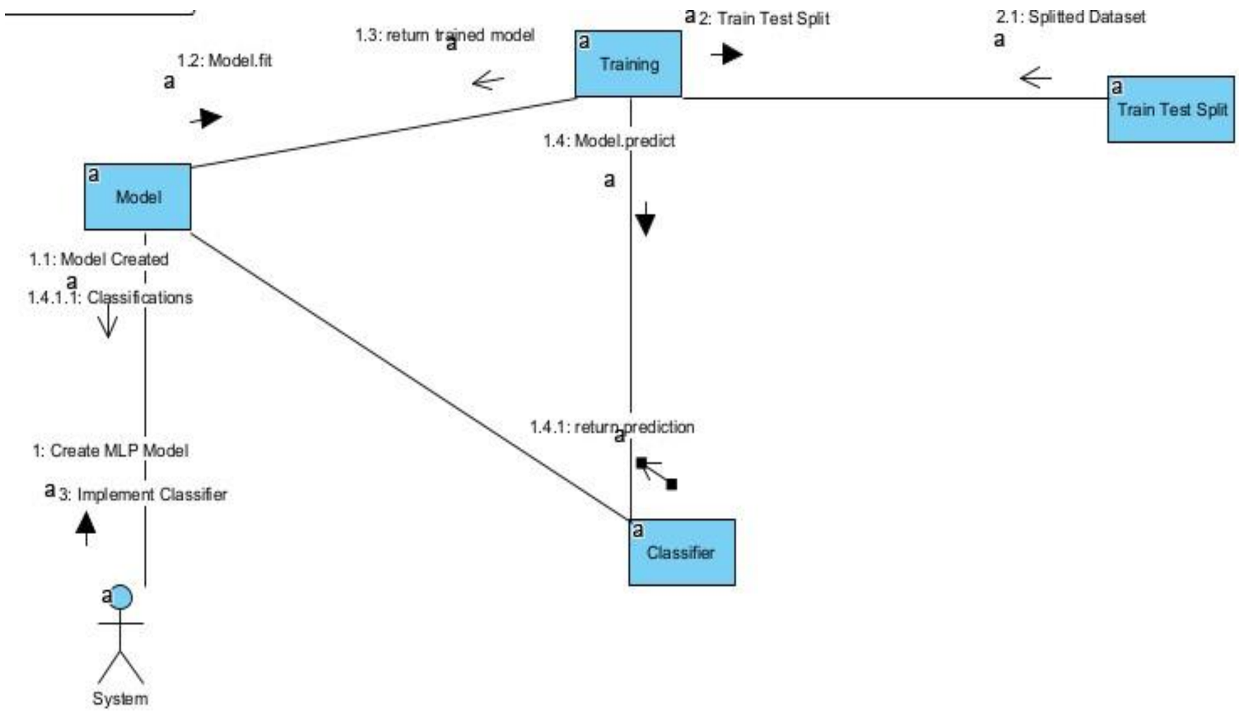
- **Create Tensorflow Based Model**



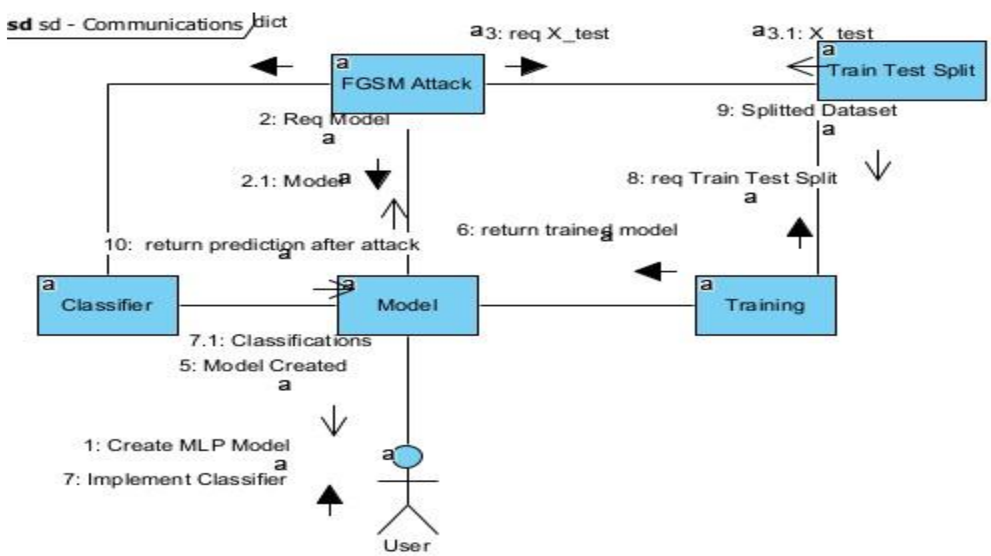
- **Implementing Classifiers**



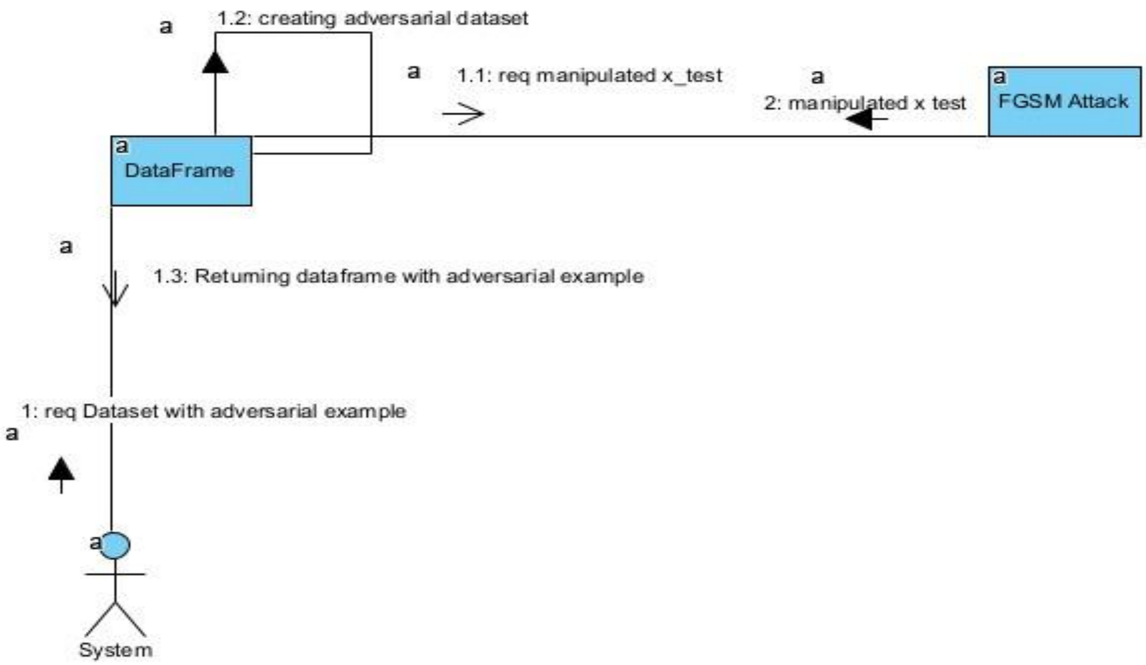
- **Training our Model**



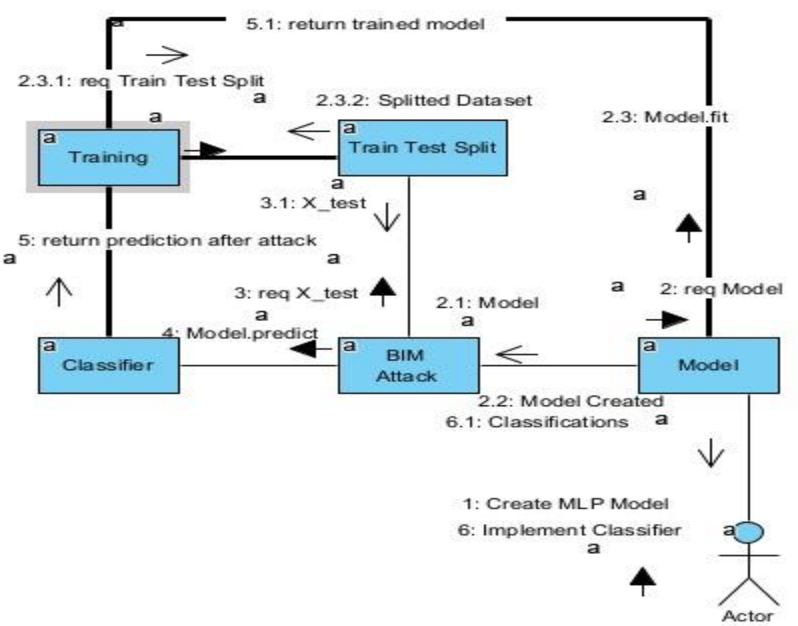
- **Implementing FGSM using Cleverhans Attack Module**

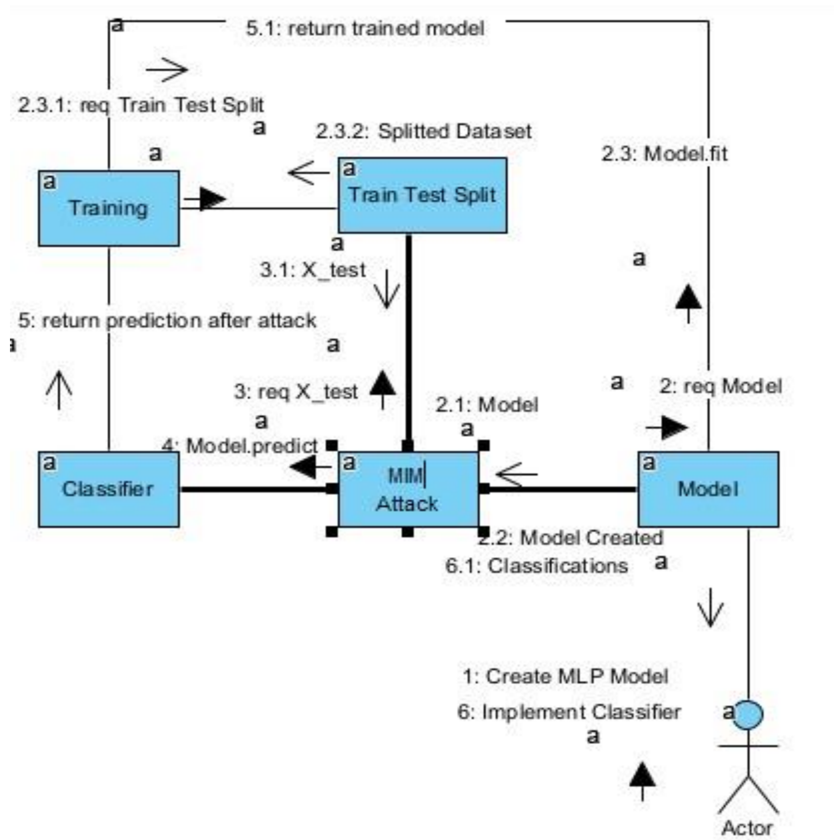


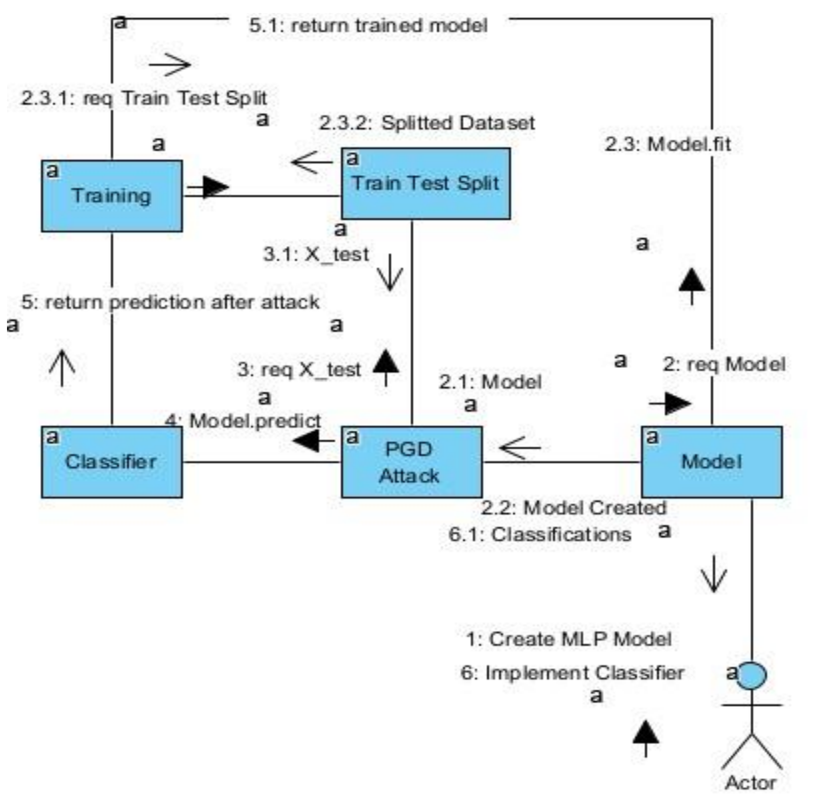
- **Creating a dataset with adversarial example**

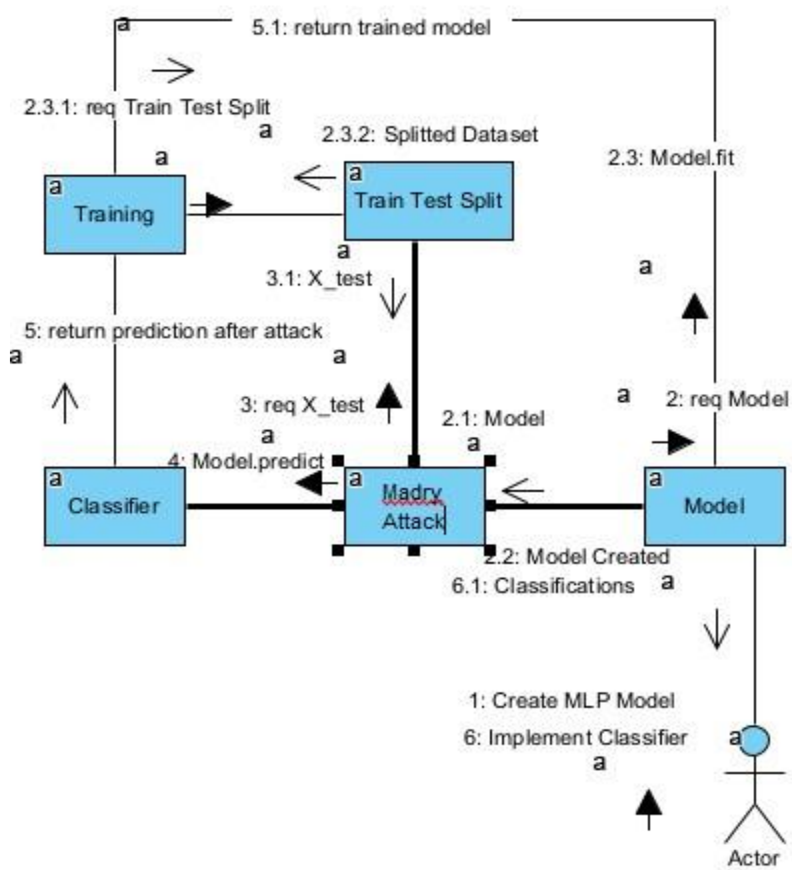


- **Implementing multiple attack from Clever Hans Attack Module**

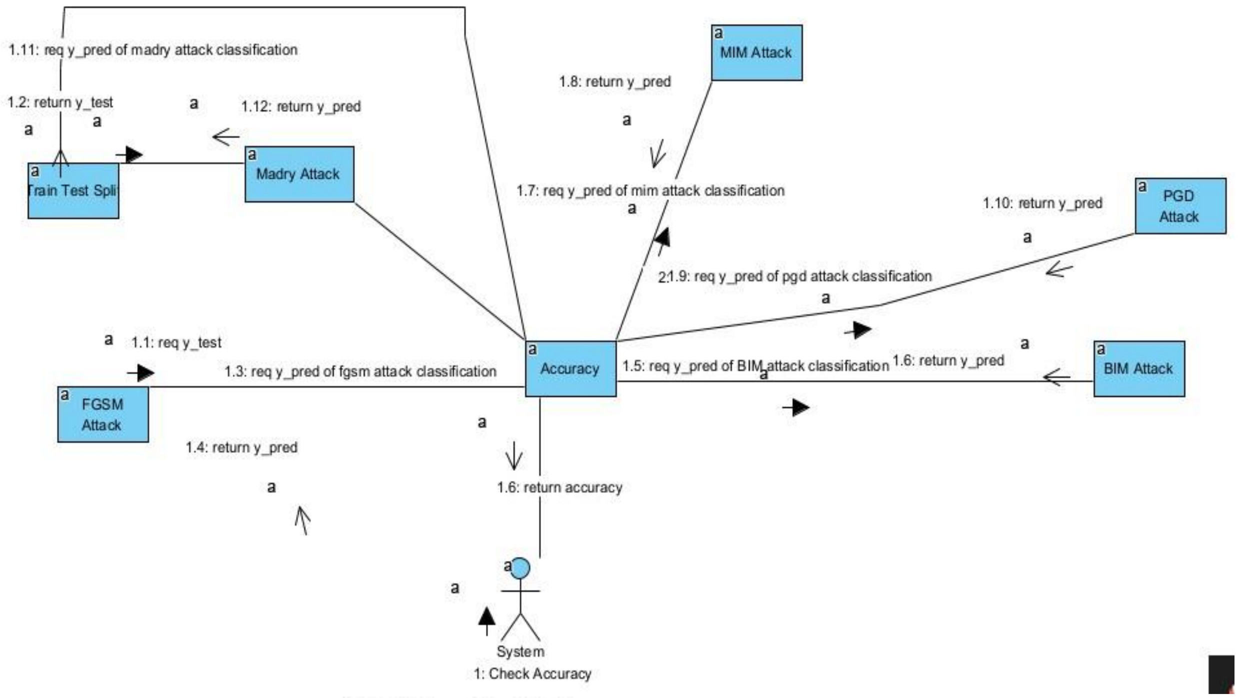


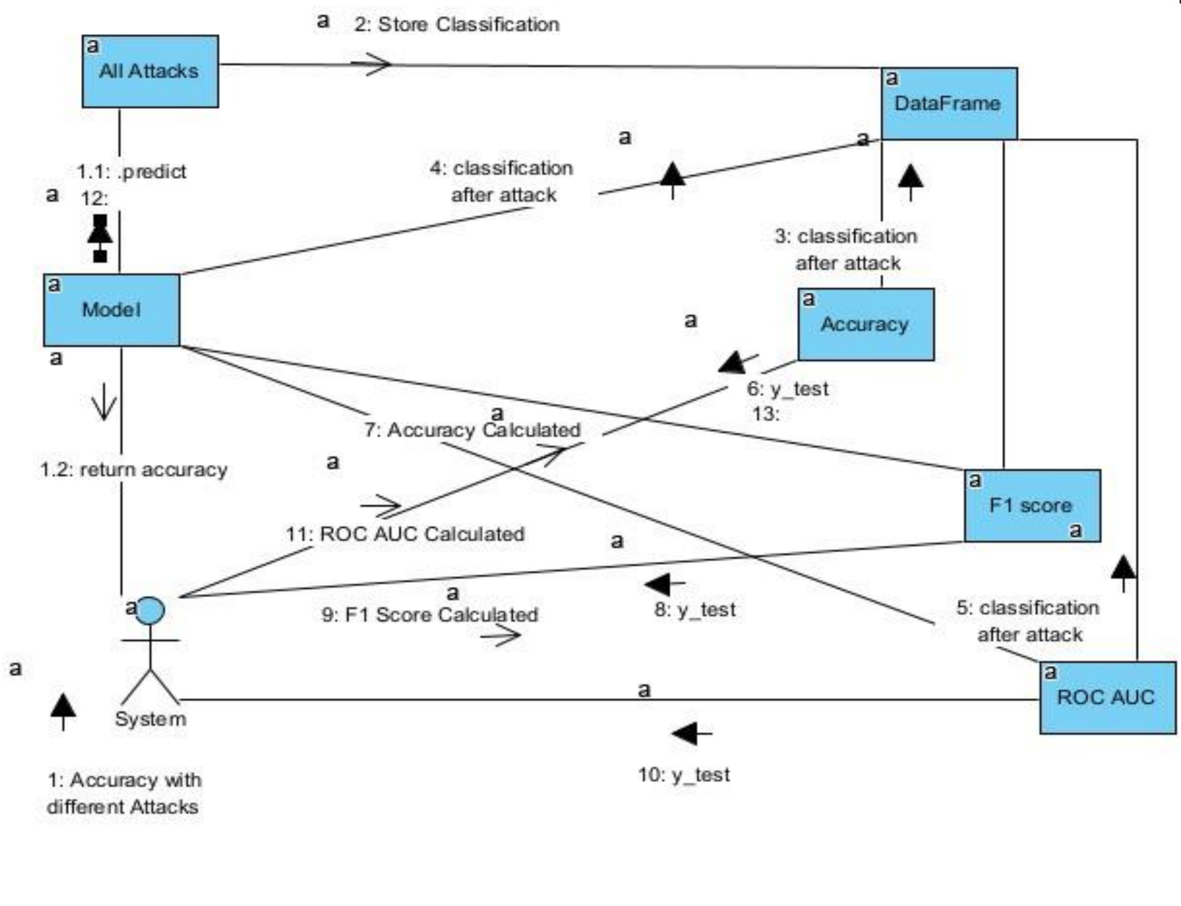

5.1: return trained model
2.3.1: req Train Test Split
2.3.2: Splitted Dataset
2.3: Model.fit
Training
Train Test Split
3.1: X_test
5: return prediction after attack
3: req X_test
2.1: Model
2: req Model
4: Model.predict
Classifier
MIM
Attack
Model
2.2: Model Created
6.1: Classifications
1: Create MLP Model
6: Implement Classifier
Actor


5.1: return trained model
2.3.1: req Train Test Split
2.3.2: Splitted Dataset
2.3: Model.fit
Training
Train Test Split
3.1: X_test
5: return prediction after attack
3: req X_test
2.1: Model
2: req Model
4: Model.predict
Classifier
PGD
Attack
Model
2.2: Model Created
6.1: Classifications
1: Create MLP Model
6: Implement Classifier
Actor



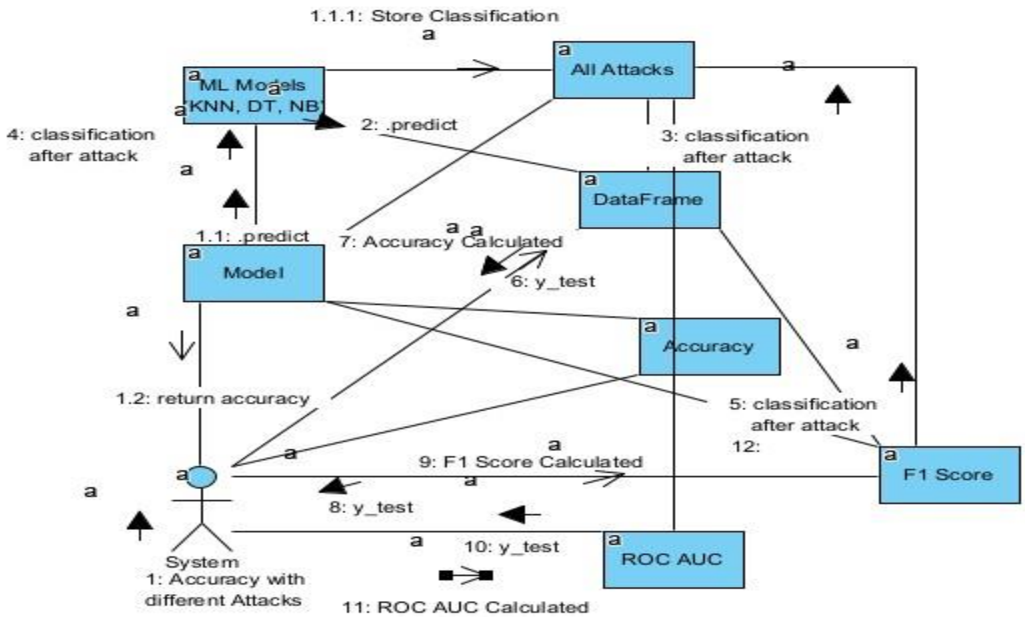
- **Checking accuracy of our model with different attacks**


1.11: req y_pred of madry attack classification
1.2: return y_test
1.12: return y_pred
Train Test Spli
Madry Attack
1.8: return y_pred
MIM Attack
1.7: req y_pred of mim attack classification
1.10: return y_pred
PGD Attack
21.9: req y_pred of pgd attack classification
1.1: req y_test
1.3: req y_pred of fgsm attack classification
Accuracy
1.5: req y_pred of BIM attack classification
1.6: return y_pred
BIM Attack
FGSM Attack
1.4: return y_pred
1.6: return accuracy
System
1: Check Accuracy

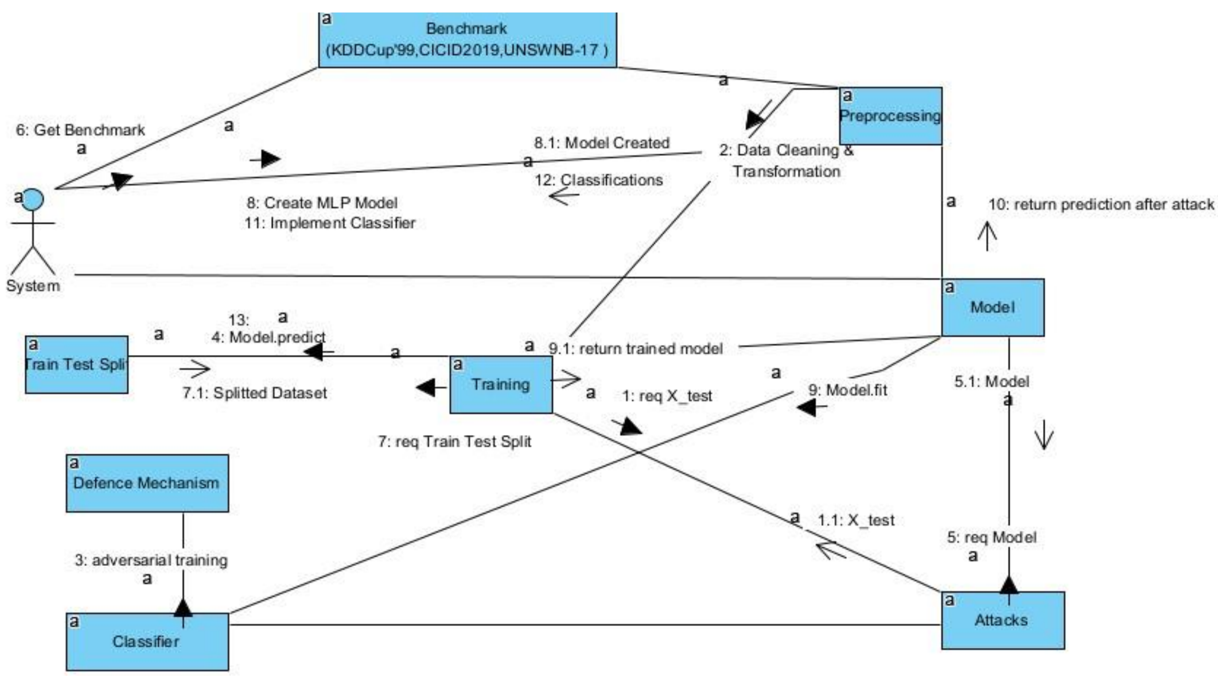

2: Store Classification
All Attacks
DataFrame
1.1: .predict
12:
4: classification after attack
3: classification after attack
Model
Accuracy
6: y_test
13:
7: Accuracy Calculated
1.2: return accuracy
F1 score
11: ROC AUC Calculated
8: y_test
9: F1 Score Calculated
5: classification after attack
System
ROC AUC
1: Accuracy with different Attacks
10: y_test

28.6.2FYP 2:

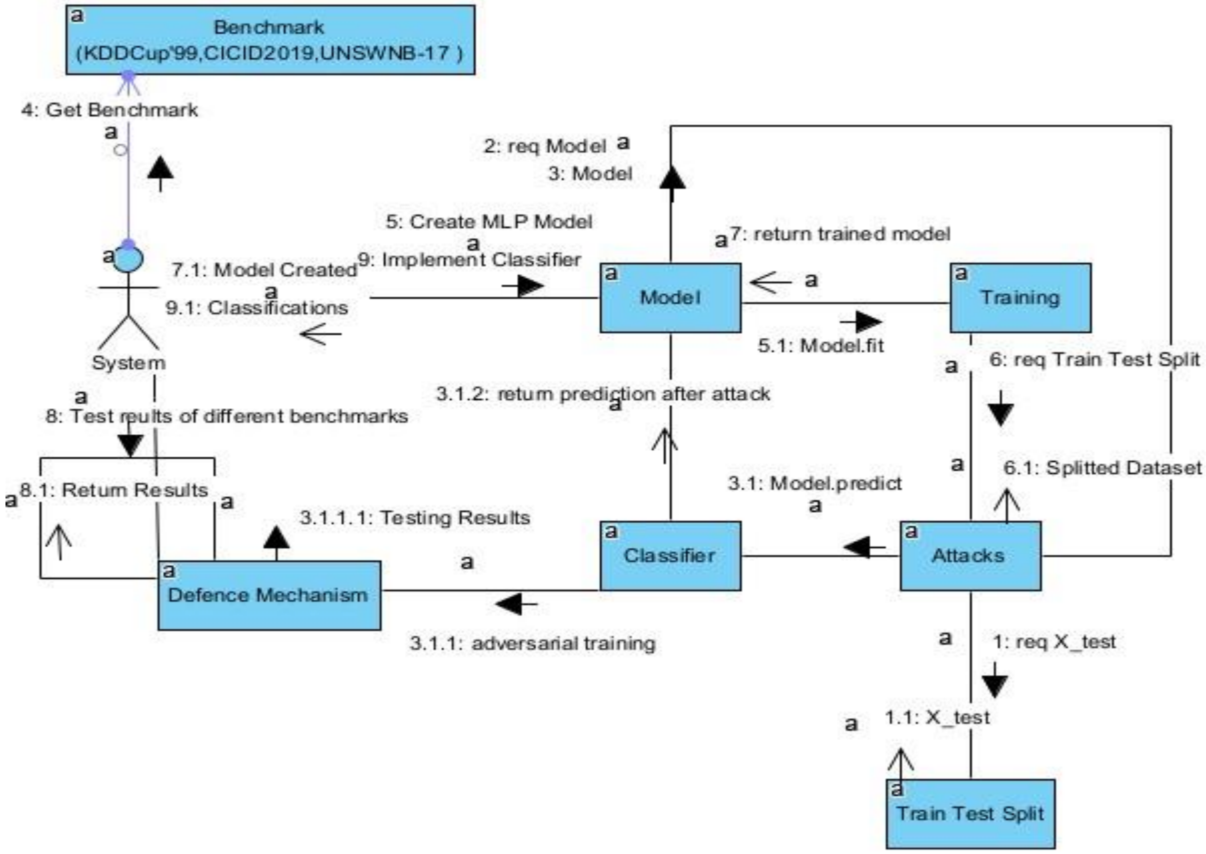
- **Using different ML models to check accuracy of different attack**



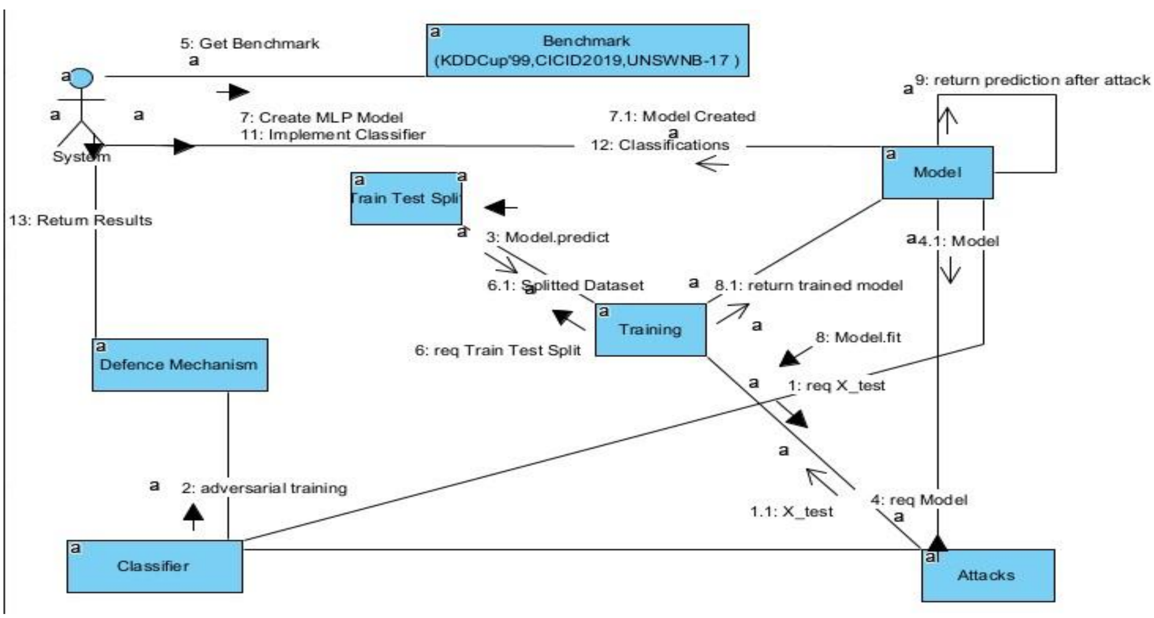
- **Implementing Defense Method (Adversarial Training)**



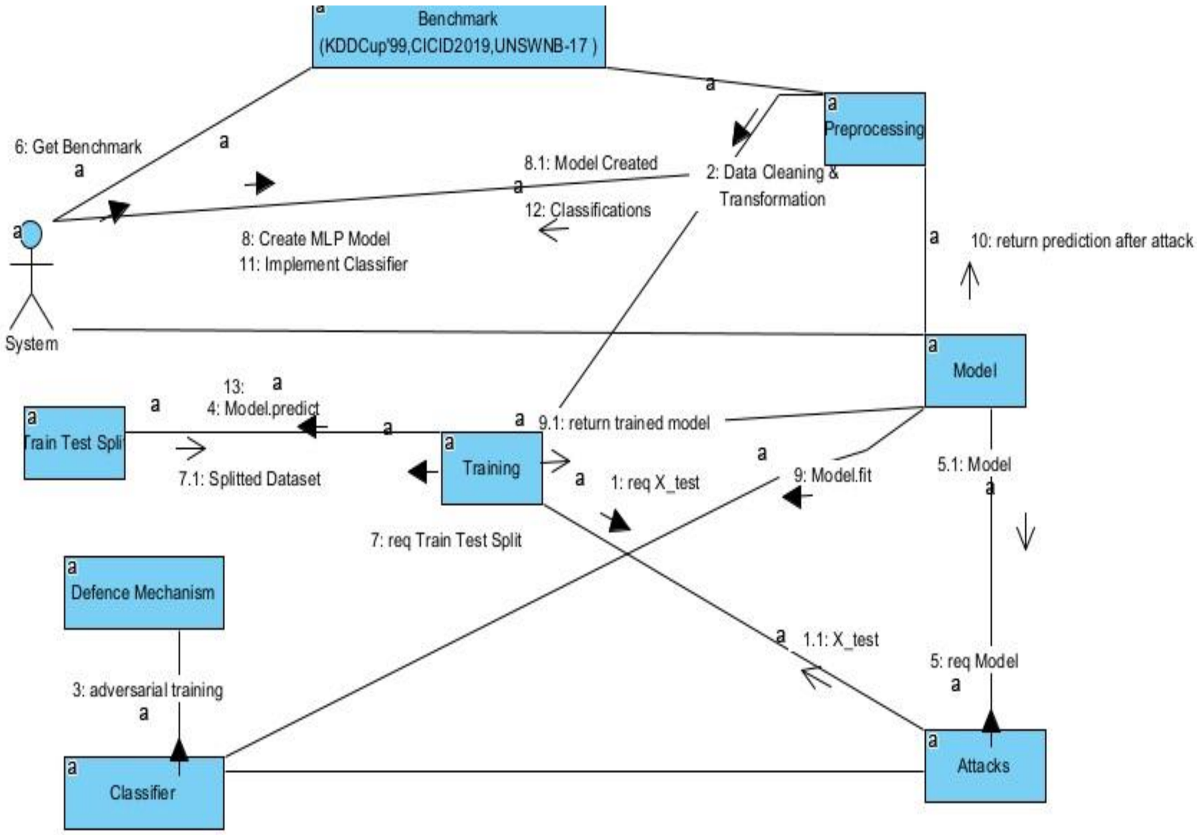
- **Testing Defense Method**



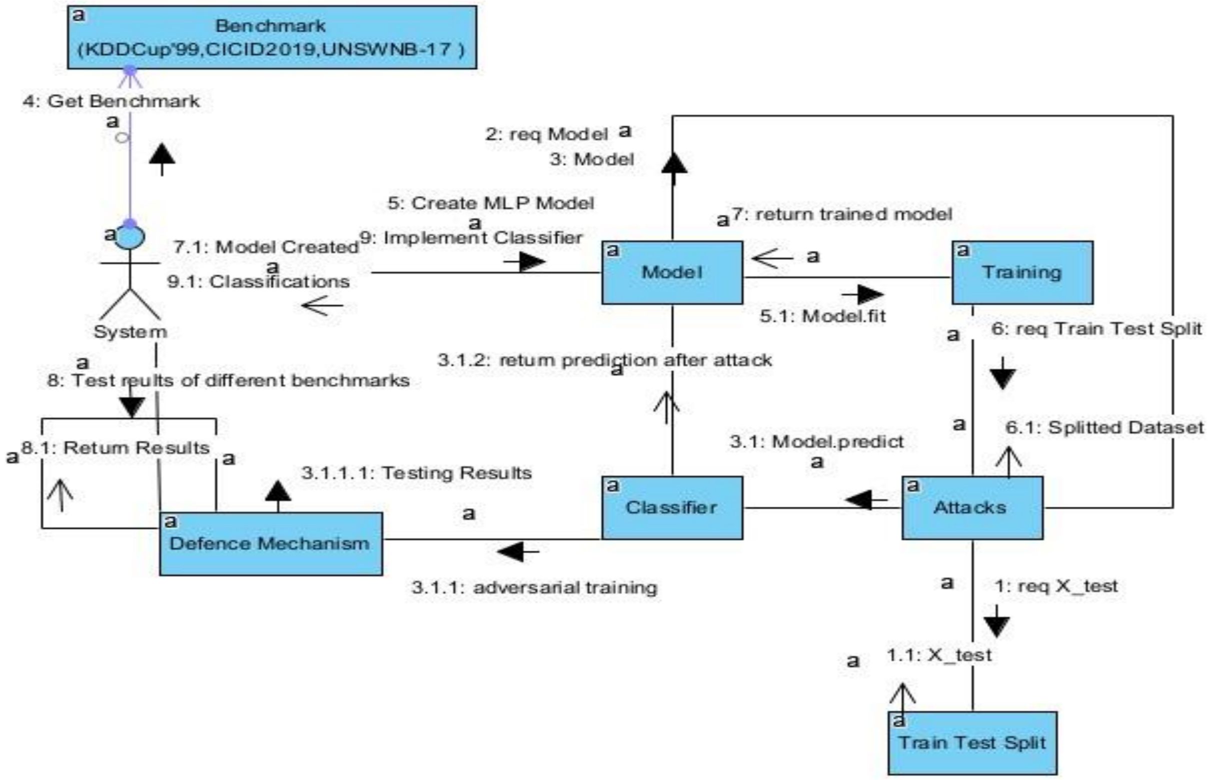
- **Using multiple benchmarks for our model**



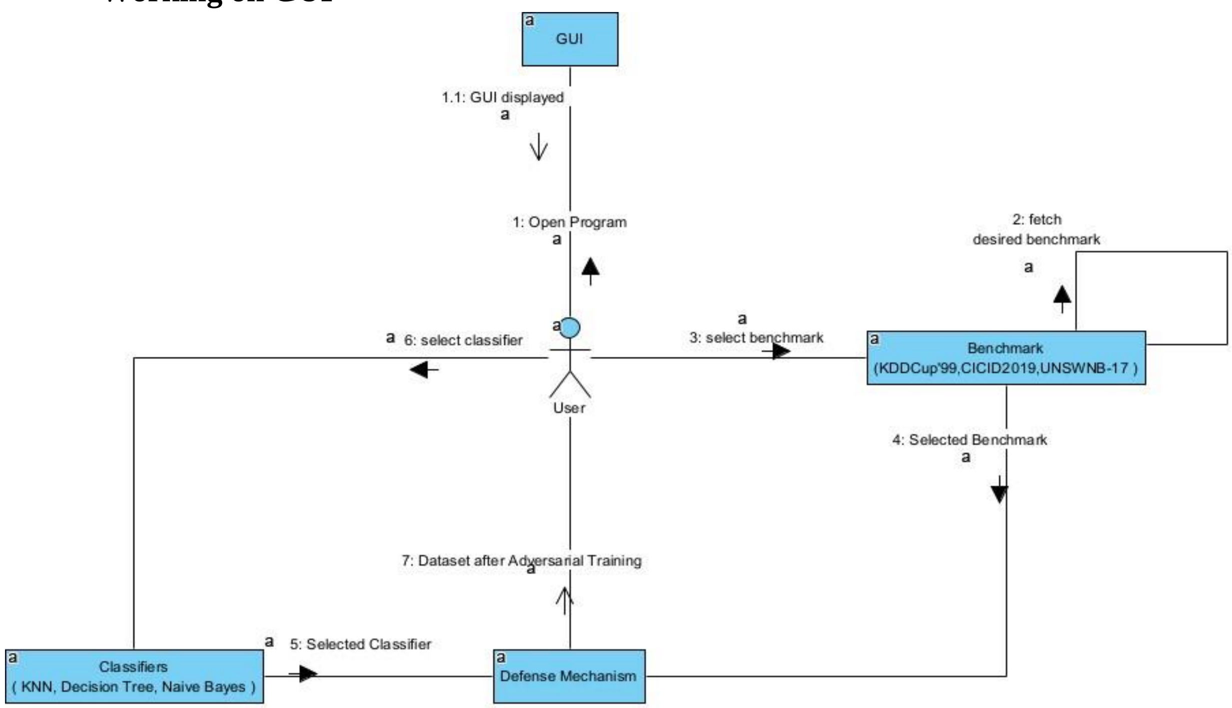
- **Preprocessing different benchmarks**



- **Testing results given by different benchmarks**

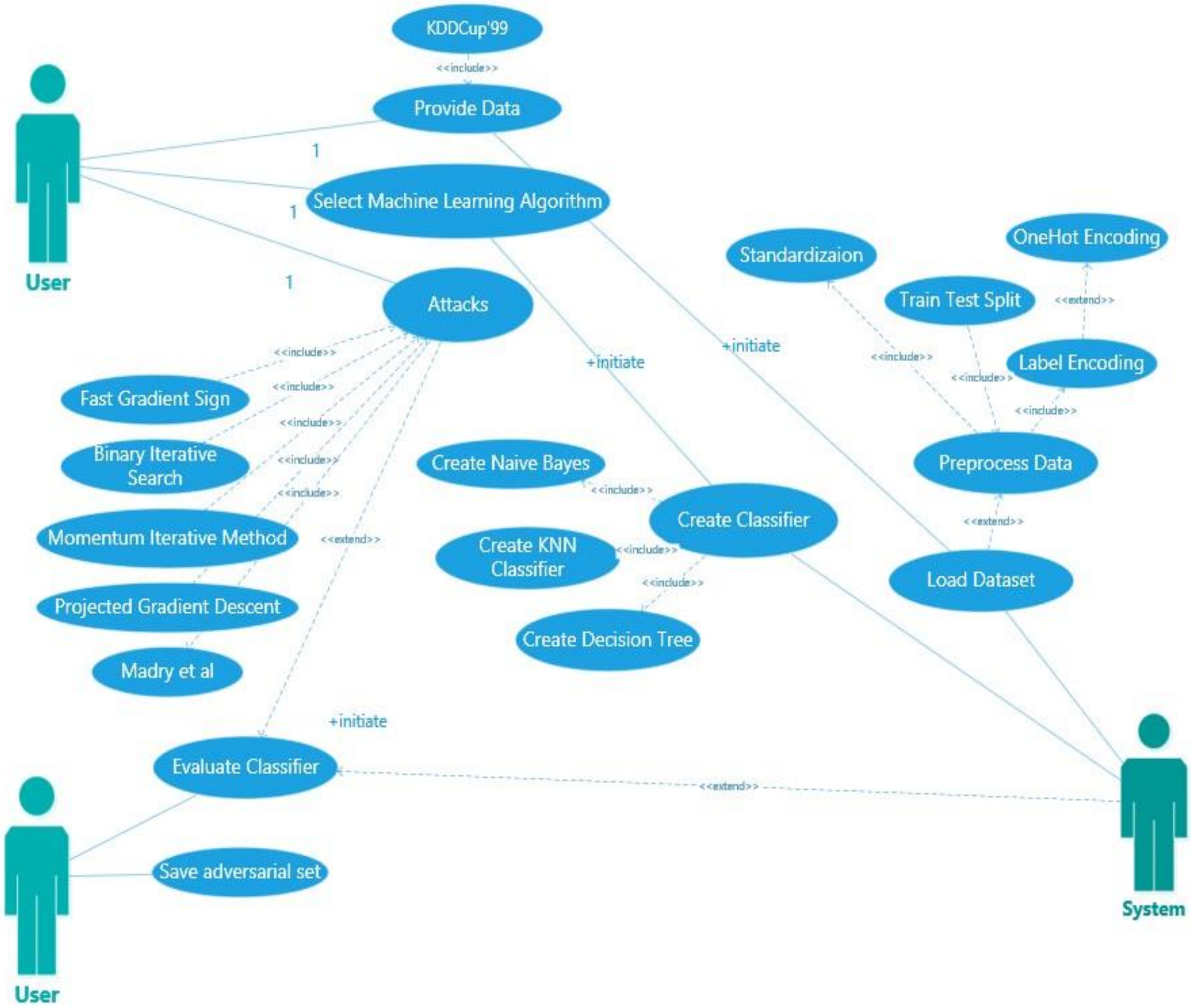


- **Working on GUI**

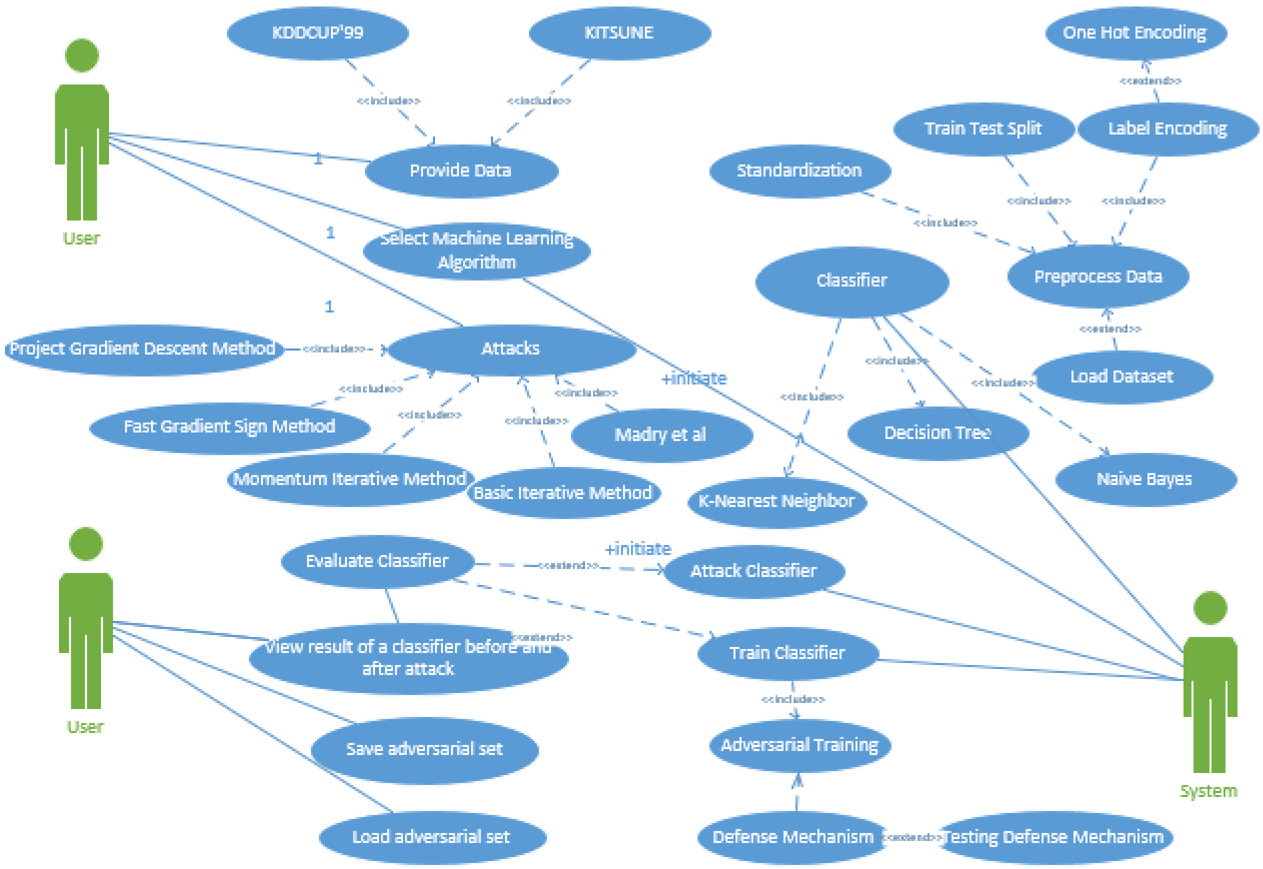


28.7 Use-case Diagrams

28.7.1FYP 1:

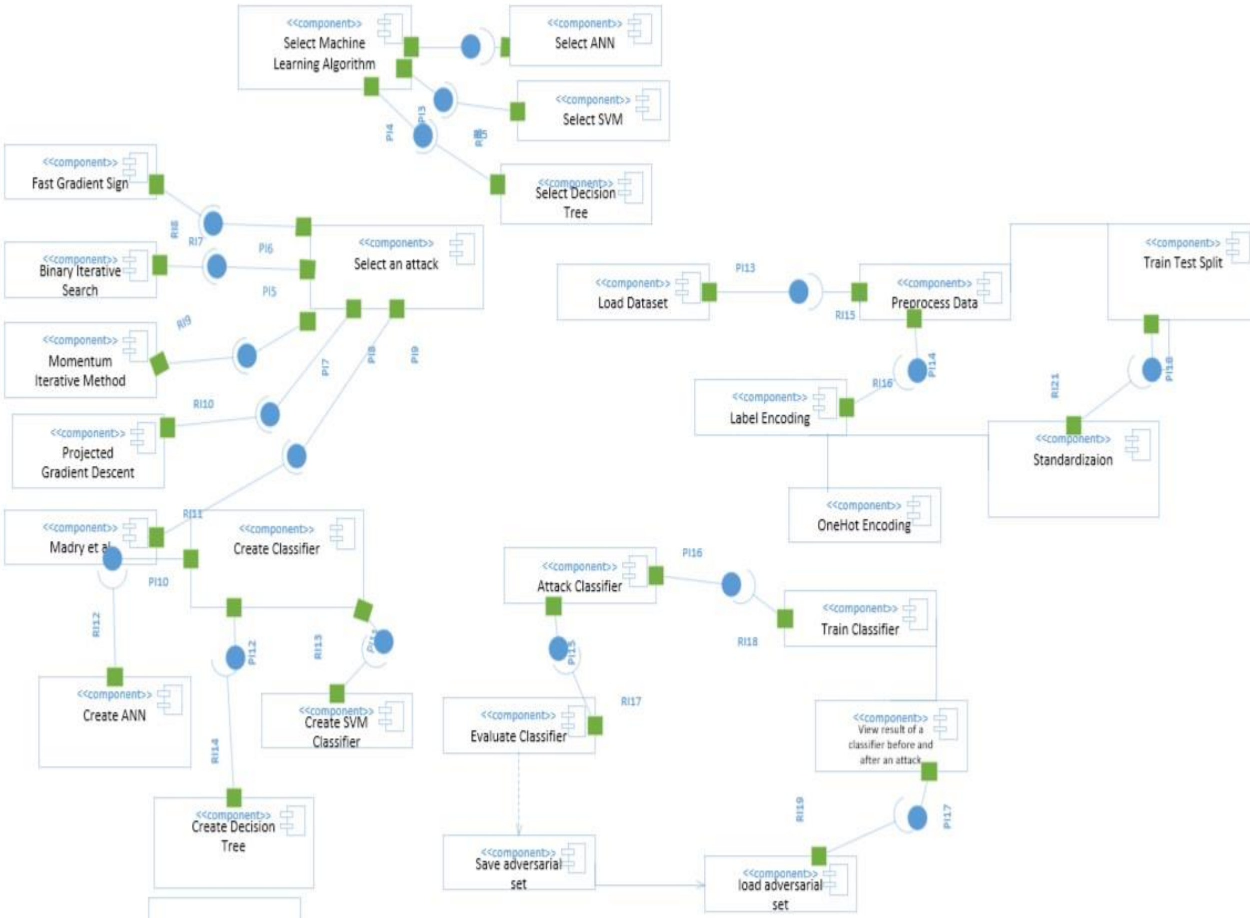

KDDCup'99
<<include>>
Provide Data
User
1
Select Machine Learning Algorithm
1
1
Attacks
<<include>>
Fast Gradient Sign
Binary Iterative Search
Momentum Iterative Method
Projected Gradient Descent
Madry et al
<<extend>>
+initiate
+initiate
+initiate
Create Naive Bayes
Create KNN Classifier
Create Decision Tree
Create Classifier
Standardizaion
Train Test Split
OneHot Encoding
<<extend>>
Label Encoding
Preprocess Data
Load Dataset
Evaluate Classifier
Save adversarial set
User
System

28.7.2FYP 2:



User
KDDCUP'99
KITSUNE
Provide Data
Select Machine Learning Algorithm
Attacks
Project Gradient Descent Method
Fast Gradient Sign Method
Momentum Iterative Method
Basic Iterative Method
Madry et al
One Hot Encoding
Train Test Split
Label Encoding
Standardization
Preprocess Data
Load Dataset
Classifier
Decision Tree
K-Nearest Neighbor
Naive Bayes
+initiate
Evaluate Classifier
Attack Classifier
View result of a classifier before and after attack
Train Classifier
Save adversarial set
Load adversarial set
Adversarial Training
Defense Mechanism
Testing Defense Mechanism
System

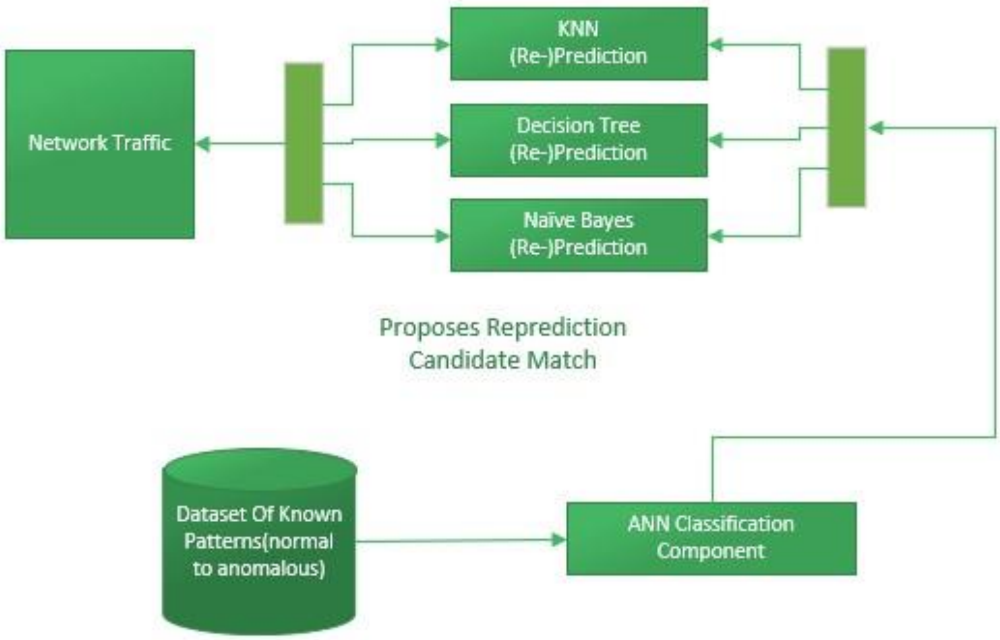
28.8 Component Diagram



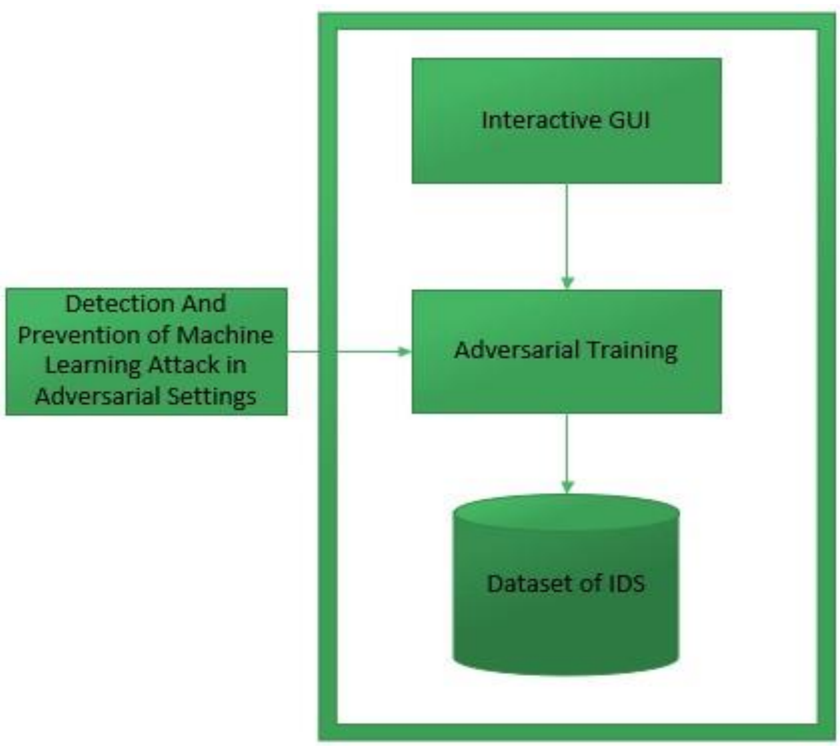
28.9 Deployment Diagram

28.10 System Block diagram

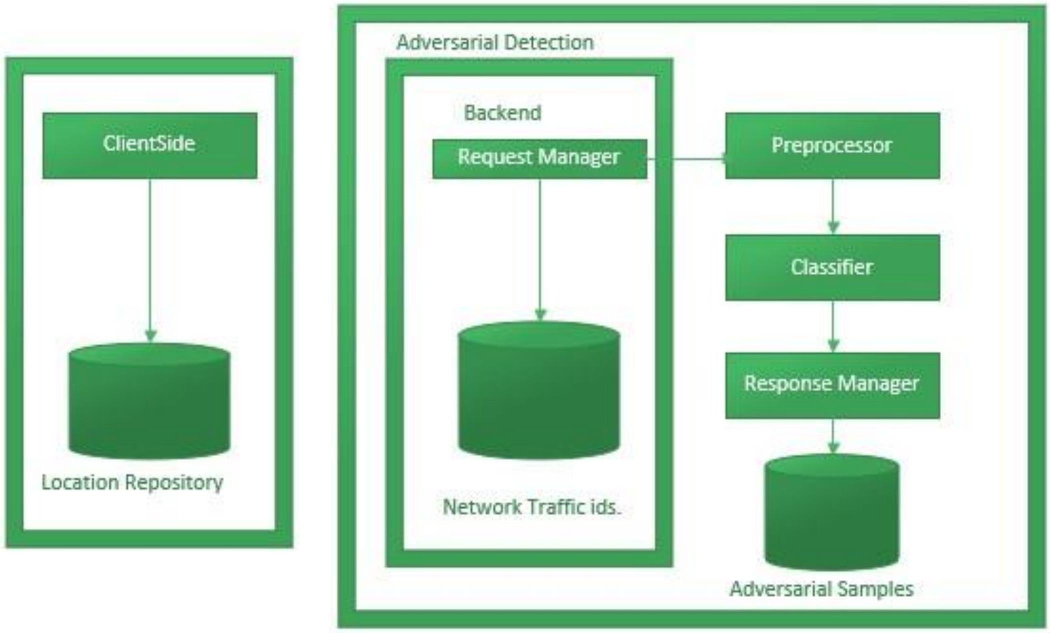
28.10.1 Context Diagram:



28.10.2 N-tier Architecture Diagram:



28.10.3 Design Architecture Diagram:



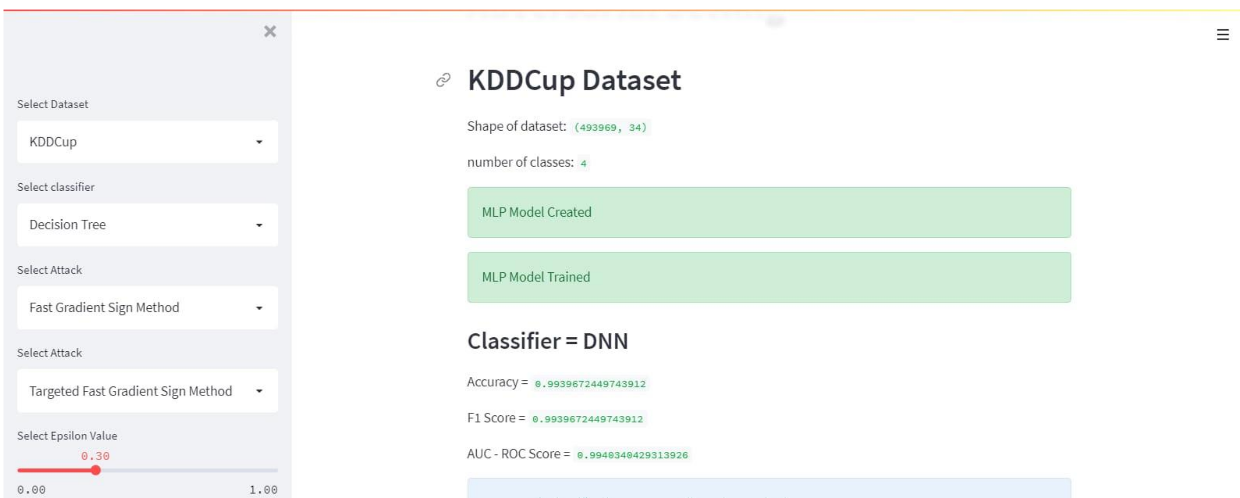
User Manual

29 Accessing through streamlit:

1. To access open anaconda prompt and type "streamlit run *name of directory where program is stored*"

2. Then wait for it to open on your browser

3. Once the program is running on browser we can see multiple option in the form of drop down. User can select options according to their needs and the result would then be displayed dynamically on the right side of the screen with proper headings



Iteration Plan

S.No.	Features	FYP-I Iterations				FYP-II Iterations		
		Monthly Iteration-I	Monthly Iteration-II	Monthly Iteration-III	Monthly Iteration-IV	Monthly Iteration-V	Monthly Iteration-VI	Monthly Iteration-VII
F1	Selecting Appropriate Benchmark	Requirements						
		Design						
		Implementation						
		Testing						
F2	Preprocessing	Requirements						
		Design						
		Implementation						
		Testing(50%)	Testing(100%)					
F3	Create Tensorflow Based Model		Requirements					
			Design					
				Implementation				
F4	Implementing classifiers	Requirements		Testing				
		Design						
		Implementation		…				
		Testing(50%)	Testing(100%)					
F5	Trainng our model		Requirements(100%)					
			Design(50%)					
				Design(100%)				
				Implementation				
F6	Implementing FGSM using Cleverhans Attack Module	Requirements		Testing				
		Design						
		Implementation						
		Testing						
F7	Creating a dataset with adversarial example		Requirements(100%)					
			Design(50%)					
				Design(100%)				
				Implementation				
F8	Analysis using AUC-ROC		Requirements(100%)	Testing				
			Design(100%)					
			Implementation					
			Testing					
F9	Implementing multiple attack from Cleverhans Attack Module		Requirements(100%)					
			Design(100%)					
				Implementation				
F10	Checking accuracy of our model with different attacks			Testing				
						Requirements(100%)		
						Design(100%)		
							Implementation	
F11	Using different ML models to check accuracy of different attack						Testing	
						Requirements(100%)		
						Design(100%)		
							Implementation	
F12	Implementing Defense Method (Adversarial Training)						Testing	
						Requirements(100%)		
						Design(100%)		
							Implementation	Implementation
F13	Testing Defense Method						Testing	Testing
						Requirements(100%)		
						Design(100%)		
							Implementation	
							Testing	
F14	Using multiple benchmarks for our model						Requirements(100%)	
							Design(100%)	
								Implementation
								Testing
F15	Preprocessing different benchmarks						Requirements(100%)	
							Design(100%)	
								Implementation
								Testing
F16	Testing results given by different benchmarks						Requirements(100%)	
							Design(100%)	
								Implementation
								Testing
F17	Working on GUI						Requirements(100%)	
							Design(100%)	
								Implementation
								Testing

Gantt Chart

Detection & Prevention Of Machine Learning Attack in Adversarial Setting Name

Learn about the Pro vers

Project Start Date	10/1/2021 (Friday)	Display Week	1		
Project Lead	Anzeela, Rumsha				
Week 1	Week 2	Week 3	Week 4		
27 Sep 2021	4 Oct 2021	11 Oct 2021	18 Oct 2021		

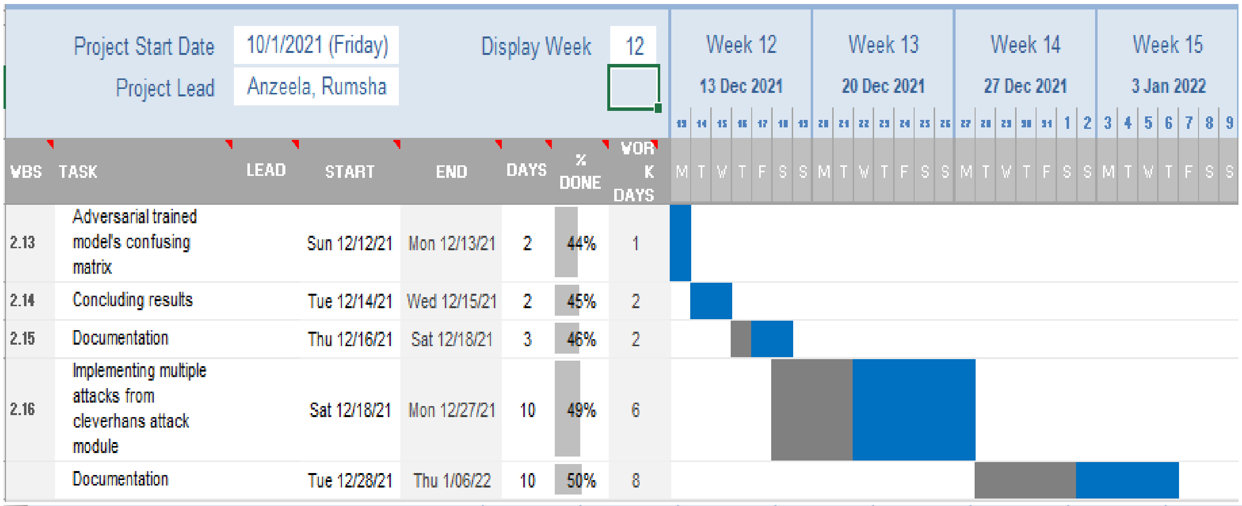
WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Executive Summary**						
1.1	Research for Ideas		Fri 10/01/21	Tue 10/05/21	5	10%	3
1.2	Approach advisors		Wed 10/06/21	Thu 10/07/21	2	20%	2
1.3	Confirm Advisor		Fri 10/08/21	Fri 10/08/21	1	30%	1
1.4	Detailed Discussion		Mon 10/11/21	Wed 10/13/21	3	60%	3
1.4.1	Scope and Project Description		Fri 10/15/21	Fri 10/15/21	1	100%	1
1.5	Fill & Submit Advisor Form		Sat 10/16/21	Sun 10/17/21	2	70%	0
1.6	Work on Project Proposal		Tue 10/19/21	Fri 10/22/21	4	95%	4
1.7	Submit Project Propsal		Sat 10/23/21	Sun 10/24/21	2	100%	0

Project Start Date	10/1/2021 (Friday)	Display Week	5		
Project Lead	Anzeela, Rumsha				
Week 5	Week 6	Week 7	Week 8		
25 Oct 2021	1 Nov 2021	8 Nov 2021	15 Nov 2021		

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
2	**FYP 1**						
2.1	Forming Project infrastructure		Mon 10/25/21	Thu 10/28/21	4	5%	4
2.2	Selecting appropriate benchmarks		Fri 10/29/21	Fri 10/29/21	1	7%	1
2.3	Preprocessing		Sat 10/30/21	Thu 11/04/21	6	15%	4
2.3.1	Data cleaning		Fri 11/05/21	Sat 11/06/21	2	40%	1
2.3.2	Removing Outlier		Sun 11/07/21	Mon 11/08/21	2	40%	1
2.3.3	Label Encoding		Tue 11/09/21	Wed 11/10/21	2	20%	2
2.4	Create a tensorflow based model		Thu 11/11/21	Sat 11/13/21	3	20%	2
2.5	Implementing Classifier		Sun 11/14/21	Wed 11/17/21	4	30%	3
2.5.1	Training Testing and Validation		Thu 11/18/21	Fri 11/19/21	2	25%	2
2.5.2	OneHot Encoding		Sat 11/20/21	Sun 11/21/21	2	30%	0

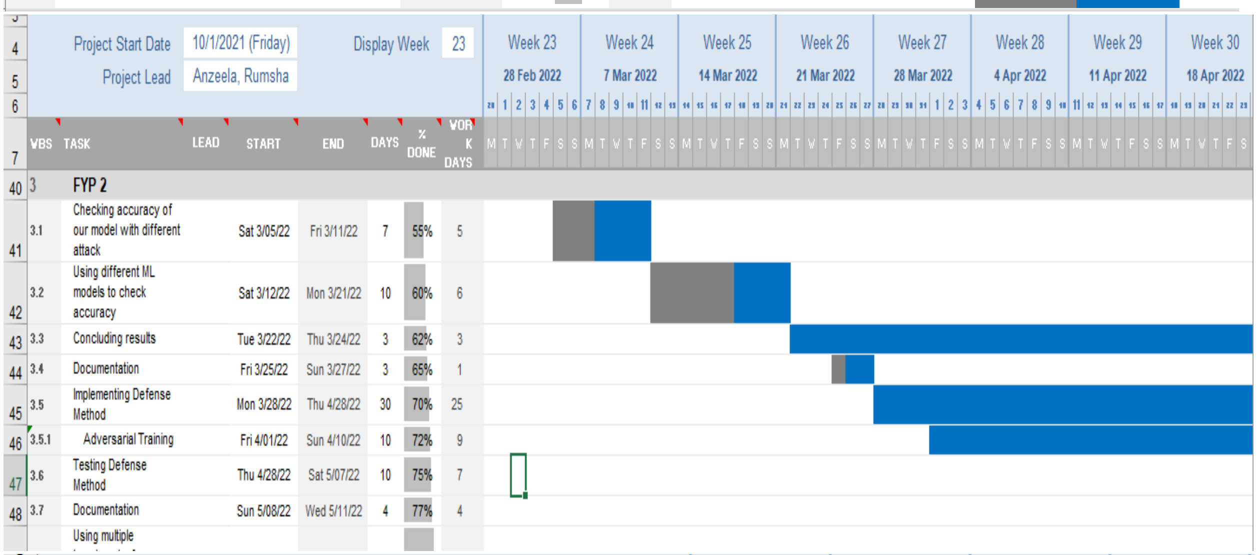
Project Start Date	10/1/2021 (Friday)	Display Week	9		
Project Lead	Anzeela, Rumsha				
Week 9	Week 10	Week 11	Week 12		
22 Nov 2021	29 Nov 2021	6 Dec 2021	13 Dec 2021		

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
2.6	Documentation		Mon 11/22/21	Wed 11/24/21	3	32%	3
2.7	Research for Adversarial Attacking Strategies		Thu 11/25/21	Fri 11/26/21	2	35%	2
2.8	Implementing FGSM attack using Cleverhans attack module		Sat 11/27/21	Tue 11/30/21	4	37%	2
2.9	Training of attack detection model with non manipulated dataset		Thu 12/09/21	Mon 12/13/21	5	40%	3
2.10	Original Model's Confusion Matrix		Tue 12/14/21	Wed 12/15/21	2	41%	2
2.11	Attack Model's Confusion Matrix		Thu 12/16/21	Fri 12/17/21	2	42%	2
2.12	Adversarial Training		Sat 12/18/21	Sun 12/19/21	2	43%	0



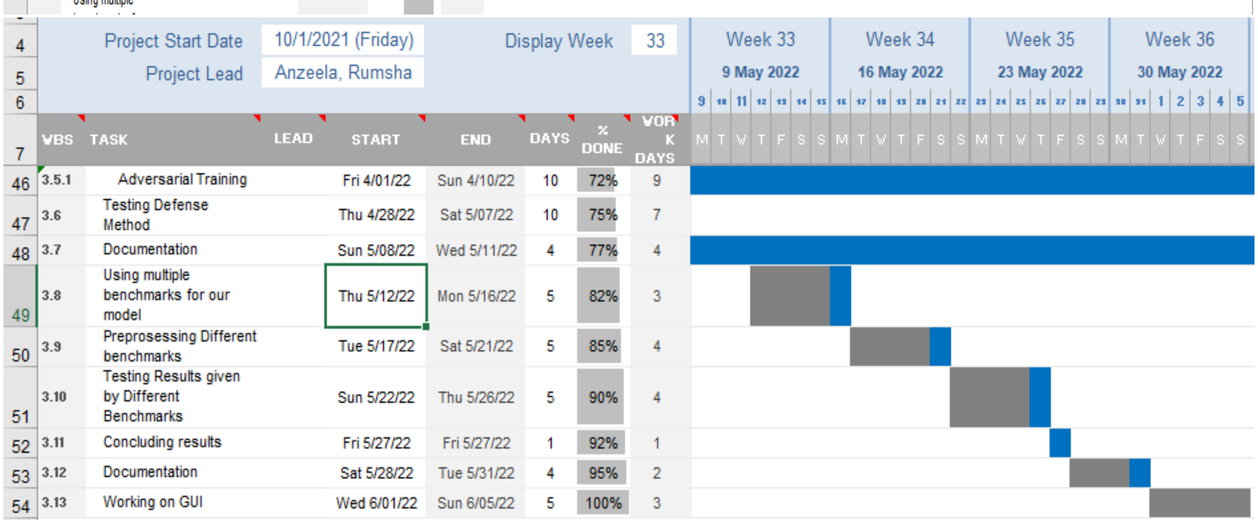
Project Start Date: 10/1/2021 (Friday) | Display Week: 12
Project Lead: Anzeela, Rumsha

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
2.13	Adversarial trained model's confusing matrix		Sun 12/12/21	Mon 12/13/21	2	44%	1
2.14	Concluding results		Tue 12/14/21	Wed 12/15/21	2	45%	2
2.15	Documentation		Thu 12/16/21	Sat 12/18/21	3	46%	2
2.16	Implementing multiple attacks from cleverhans attack module		Sat 12/18/21	Mon 12/27/21	10	49%	6
	Documentation		Tue 12/28/21	Thu 1/06/22	10	50%	8



Project Start Date: 10/1/2021 (Friday) | Display Week: 23
Project Lead: Anzeela, Rumsha

	WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
40	3	FYP 2						
41	3.1	Checking accuracy of our model with different attack		Sat 3/05/22	Fri 3/11/22	7	55%	5
42	3.2	Using different ML models to check accuracy		Sat 3/12/22	Mon 3/21/22	10	60%	6
43	3.3	Concluding results		Tue 3/22/22	Thu 3/24/22	3	62%	3
44	3.4	Documentation		Fri 3/25/22	Sun 3/27/22	3	65%	1
45	3.5	Implementing Defense Method		Mon 3/28/22	Thu 4/28/22	30	70%	25
46	3.5.1	Adversarial Training		Fri 4/01/22	Sun 4/10/22	10	72%	9
47	3.6	Testing Defense Method		Thu 4/28/22	Sat 5/07/22	10	75%	7
48	3.7	Documentation		Sun 5/08/22	Wed 5/11/22	4	77%	4
		Using multiple						



Project Start Date: 10/1/2021 (Friday) | Display Week: 33
Project Lead: Anzeela, Rumsha

	WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
46	3.5.1	Adversarial Training		Fri 4/01/22	Sun 4/10/22	10	72%	9
47	3.6	Testing Defense Method		Thu 4/28/22	Sat 5/07/22	10	75%	7
48	3.7	Documentation		Sun 5/08/22	Wed 5/11/22	4	77%	4
49	3.8	Using multiple benchmarks for our model		Thu 5/12/22	Mon 5/16/22	5	82%	3
50	3.9	Preprosessing Different benchmarks		Tue 5/17/22	Sat 5/21/22	5	85%	4
51	3.10	Testing Results given by Different Benchmarks		Sun 5/22/22	Thu 5/26/22	5	90%	4
52	3.11	Concluding results		Fri 5/27/22	Fri 5/27/22	1	92%	1
53	3.12	Documentation		Sat 5/28/22	Tue 5/31/22	4	95%	2
54	3.13	Working on GUI		Wed 6/01/22	Sun 6/05/22	5	100%	3