Assessing The Impact of Cybersecurity Attacks On Machine Learning Model

A PROJECT REPORT

Submitted by

Moukhik Misra (19BCE2190)

Mohit Suhasaria (19BCE2167)

CSE3501 INFORMATION SECURITY ANALYSIS AND AUDIT

Slot – L29+L30 (F2)

Computer Science and Engineering

DECEMBER 2021

1. Problem Statement

Recently machine learning models have been applied to predict and detect cybersecurity attacks in various scenarios. However, there is a possibility that the attacker may target the model itself. Cybersecurity in recent times relies heavily upon machine learning techniques to detect attacks however attacking the models themselves haven't been a topic of huge discussion. In this project we look at the effect of attacks on the machine learning model itself. We will be assessing the impact of both targeted and non-targeted attacks on a logistic regression model that predicts numbers based on given image and try to attack it by various methods to understand the impact that these sorts of attacks have on the accuracy of made machine learning model. Statistical analysis of the results will be conducted, and the effects of the attacks will be shown and explained.

1.1. Idea

Logistic regression, also known as logit model is a kind of statistical analysis which is frequently used in machine learning applications for predictive analysis modelling. In this approach the dependent variable is categorical and depending on the number of categorical variables it is further divided into binary logistic regression and multinomial logistic regression. It is used to understand the relationship between categorical variables and independent variables by computing probabilities through a logistic regression equation.

Say, if we use a linear regression model for classifying data then a threshold must be set to distinguish between classes. So if the predicted continuous value falls in a range outside the threshold of its true class then it would lead to inaccurate classification. This means a model suitable for classification is needed, such as logistic regression.

Sophisticated attacks on a system exploit various vulnerabilities to gain access to restricted information. It becomes essential for a cyber security administrator to establish associations between attacks and the vulnerabilities present. Attacks on a network involving data breach are rare events, so a more common event that can be considered in

this context is a warning which is in general a more common class of cyber security incident. Cyber security administrators often make use of statistic models to reveal vulnerabilities in the system that factor significantly in the occurrence of cyber security incidents. These models are also used to predict the probability with which said incident would occur based on given system vulnerabilities according to real world data.

Logistic regression models are used for classifying incidents based on system vulnerabilities and other independent variables that may be considered as affecting factors. This includes data preparation through imputation and aggregation. The vulnerability related data is converted to host based data covering all factors associated with the cyber environment. The logistic regression model evaluates the associations with the cyber environment and incidents. The model is validated after performing various statistical analyses and is fit to predict the occurrence of incidents on a system based on cyber environment and other contributing factors which includes the operating system, mode of management and host type.

Since a lot of threat detection systems and incident prediction systems employ machine learning models for multiclass classification such as multinomial logistic regression, it becomes necessary for cyber security analysts to ensure that these models themselves are accurate and secure. The different ways in which such a model can be manipulated to provide inaccurate predictions is an important topic of study in cyber security. It is in a way the security of cyber security models.

In our project we consider a multinomial logistic regression model fit to a handwritten digit recognition dataset that classifies input images into true labels. We designed the model to predict digits with high accuracy to obtain a properly functioning multinomial logistic regression model on which we can test methods to decrease its accuracy efficiently. Our idea was to observe the accuracy scores of the model without any attacks and compare it to three different types of attacks, namely, non-targeted attack, natural fooling targeted attack and non-natural fooling targeted attack. In doing this, we can determine the method an adversary might use against the threat detection model of a cyber

security system. It is the first step to design successful attacks on a machine learning model to optimise it and make it more secure against similar attacks against adversaries.

1.2. Scope

This project is based on designing ways to attack a machine learning model which in turn highlights a few attacks that real world adversaries might use to break into a system. Recognising possible future attacks allows administrators to put preventive measures in place for the same. Then it is possible to say that the scope of this project extends to the protection and security of machine learning models used in any field.

The shear volume of cyber attacks has risen to the point where it is practically impossible for human beings to detect and analyse even a significant fraction of them. There are billions of malware attacks in a single year which is too much for humans to process so machine learning models are designed to process this data. Machine learning models that are able to perform functions that they haven't been explicitly programmed to do, process millions of files to identify potential threats and hazardous data. This way they can detect potential threats and neutralise them automatically.

Microsoft's cyber security platform Windows Defender ATP (Advanced Threat Protection) uses multiple levels of machine learning algorithms for breach detection, preventive protection and automated response.

Alphabet's cybersecurity company Chronicle uses machine learning to analyse the large volumes of security telemetry data generated by companies. The name of the cybersecurity product is Backstory which employs machine learning algorithms for analysing large amounts of data like known bad domains, internal network activities, suspected malware and hazardous files which are condensed using their machine learning algorithms to generate more insightful data, easy to read for security analysts to ensure the health of the network.

Sqqrl is yet another cybersecurity platform designed using machine learning models to search through networks for detection and elimination of bad code that can go undetected by the existing security measures in place. It uses machine learning to convert data points into something that depicts its behavioural trends in a visual representation for a given computer network.

The scope of this project covers all applications of machine learning as it provides an insight on the attacker's perspective while designing an attack. However, machine learning is not the only technology being used in cyber security as artificial intelligence and deep learning techniques are also being used which are beyond the scope of this project.

As a matter of fact, many attackers are also using machine learning to carry out their malicious endeavours. This means that the need for finding weaknesses in machine learning models extends to both sides of the attack, that is, if one recognises the types of possible attacks on their machine learning model then it becomes easier to put detection controls in place and implement preventive measures, or if we look at this information from the other perspective, we can say that it becomes easier to recognise the vulnerabilities in the attacker's machine learning model and make it ineffective by attacking those vulnerabilities to protect the computer network.

1.3. Novelty

The rapid growth in technology and its scope has led to an exponentially growing network of computers and databases. These components are more connected now than they have ever been in the history of existence. It is a boon to human kind as vast networks imply better connectivity which translates to faster communication, smoother data transfer and in general all tasks get executed much faster. Everyone wants to complete their tasks faster and achieve goals as soon as possible so every domain is shifting towards technology and using these networks in one way or another.

Having a large network also means better access to all kinds of data for everyone and that includes people and organisations with malicious intent. These people may want to find vulnerabilities in a network and exploit them for personal gain. The personal gain from attacking a network might be from getting access to private data through a data breach, or shutting down a system and rendering it useless and stagnant for a period of time, or even theft of proprietary information.

These cybersecurity risks have been extensively studied and researched by cybersecurity analysts to design security protocols for the protection and healthy functioning of a network. In recent years, the discussion on the mentioned topic has largely shifted to the applications of machine learning in this area of network security.

Extensive applications of machine learning algorithms in cybersecurity led to a number of different ways in which it can be used in cybersecurity.

Regression uses the knowledge about existing data to get an idea or in a way predict the implications of future data. For example, if we consider the trend of property prices in a given market and we use the existing database to fit a regression model then that model can be used to detect anomalies in property prices indicating fraudulent transactions. Types of regression that are used include; linear regression, polynomial regression, ridge regression, decision trees, SVR or support vector regression and random forest models.

Classification algorithms use categorical variables as the dependents and other affecting factors as independent variables. Classification is the most commonly used technique in cybersecurity incidents as most real world cyber incidents can be categorised and hence generate categorical data. For example, email data can be classified as important or spam and maybe other categories depending on the complexity of the classification algorithm used. Classification models used include; logistic regression, k-nearest neighbours, support vector machine, kernel SVM, Naive Bayes, decision tree classification and random forest classification.

At this point it is very clear that machine learning has been accepted by the majority of security analysts and tech companies as the go to solution for their cybersecurity needs. We have seen a lot of discussion on this topic too, but the security of these models becomes more critical as their use has become so widespread.

If a large number of security tools make use of machine learning models and algorithms in one way or another then it becomes convenient for an attacker to directly design attacks on these models which will affect the security tools and make the networks vulnerable and attacks will go undetected. In this project we discuss the approach a potential threat may use to make a machine learning model inaccurate. So we could just say that the focus of discussion here is the security of machine learning algorithms and we demonstrate it by designing and assessing various attacks on the logistic regression model. There are many ways to attack a machine learning model but the main method that we are concerned with in our project is by distorting the test data. The novel idea is observing the impact of attacking the model by considering the digits that they are more likely to be mistaken as, making them their natural fooling targets.

Then the novelty of this project is the recognition of vulnerabilities in machine learning models and its impact on cybersecurity platforms that use said algorithms.

1.4. Comparative Statement

1. Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS

The above paper aims to survey the various interactions between resilient Cyber Physical Systems using Machine Learning and resilient Machine Learning when applied to the said systems. The paper puts forth a number or promising trends in research and future directions for the research into resilient ML in CPS. On reading this paper, readers can obtain a comprehensive understanding of recent advances in Machine Learning based security and securing ML for the said system as well as develop counter measures and their research trends and topics. The paper delves into the potentials and challenges in applying AI and ML in the field of cybersecurity. It has been ascertained from the paper that these two topics will play a hugely impactful role in the securing of CPS form attackers. However there remain challenges that need to be addressed to ensure the success of the above systems. The research into adversarial attacks on ML is a hot topic for the future. There are however possibilities of intermediate attacks by the time defence mechanisms have been securely deployed against adversarial attacks and this might prove to be useful to the attackers for potential attacks.

2. A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity

In order to better understand the proper evaluation parameters for the robustness and resistance of machine learning models, specifically for classification problems in cybersecurity, against threats from adversarial attacks, the authors of this paper propose an attack method based on the brute force approach. The attack is designed using the black-box method to look at attacks from the adversary’s perspective and recognise the shortcomings of the existing attack methods. They have used their attack to produce adversarial examples for various machine learning models commonly used in

cybersecurity systems. These include android malware detection using machine learning, intrusion detection and network intrusion detection using machine learning. Their results show that they have successfully designed a more efficient attack method than the currently accepted attack methods which may be used to test various machine learning based security systems.

3. Machine Learning Security: Threats, Countermeasures, and Evaluations

This paper is a survey on the robustness of machine learning models. The authors develop a systematic approach to understand and evaluate the security issues of machine learning models against adversarial attacks. They discuss the defence strategies associated with these attacks through training phase to the test phase. The reasons an adversarial attack on a model might take place are discussed and a machine learning model in the presence of an attack is presented for better analysis. The issues related to the security of machine learning models is categorised under five labels; backdoors in the training set, poisoning of training set, model theft, adversarial example attacks and recovery of sensitive training data. The attacks are reviewed in real-world conditions to understand the real-world impact.

4. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey

The authors of this paper have surveyed attacks on computer vision and deep learning models from adversaries in a systematic and comprehensive manner. The literature associated with design of adversarial attacks is discussed to analyse their functioning and suggest defence strategies against said attacks. Real world scenarios of these attacks are also reviewed from other literature about its realistic threat. It was observed that even though deep learning models are the most suitable approach for computer vision as they provide the most accurate outputs, these models are exposed to the threat of perturbed inputs leading to completely inaccurate outputs. This paper reviews the various adversarial attacks on these deep learning model applications in computer vision and artificial intelligence, and their respective defence mechanisms. Safety and

cybersecurity related applications are specially vulnerable to these adversarial attacks in real world situations so the authors demonstrate the effective attack vectors to these models in realistic scenarios.

5. A System-Driven Taxonomy of Attacks and Defenses in Adversarial Machine Learning

The authors' main goal in this paper was to develop an unambiguous methodology for a microscopic and systematic approach for the taxonomy to specify adversarial attacks in machine learning model based applications. This would allow others to replicate these experiments in order to escalate the rate at which machine learning models are being made more robust against adversarial attacks. The article provides taxonomies for; adversary's strategy, the ML architecture, the defence response, the dataset, and the adversary's knowledge, capability, and goal. They also establish relationships between these taxonomies and models by proposing an adversarial machine learning cycle. The comparison between various literature in this field is done and the gaps and missing components are revealed.

1.5. Dataset

The dataset used in this project is from the Modified National Institute of Standards and Technology (MNIST) database of handwritten digits. It is a modified and smaller version of a larger set available from the NIST database. It contains data of handwritten digits which have been preprocessed for computer vision and machine learning.

The digits are already size normalised and centred with respect to a fixed size. The original images in the NIST database were bilevel images, that is black and white with a normalised size of 20x20 pixel box with the aspect ratio preserved as the raw. The processed images contained grey levels resulting from the anti-aliasing technique of the normalisation algorithm. The centre of mass of the pixels for each image was computed to align it with the centre of the 28x28 pixel image to maintain consistent relative positioning of the hand written digit in each image.

The centre of the image may be aligned with the 28x28 box differently based on the classification algorithm used for the dataset. It generally needs to be centred using bounding box rather than centre of mass of pixels for particularly template based classification algorithms such as k-nearest neighbours (k-NN) and support vector machine (SVM). Since this project deals with the classification algorithm of logistic regression (LR) we may not need to process the dataset again to modify alignment from centre of mass of pixels to bounding box alignment.

The NIST database was divided into Special Database 3 which was collected from the employees at Census Bureau and Special Database 1 which was collected from high school students. Special Database 3 was originally used as the training set and Special Database 1 was used as the test set but it was quite apparent that SD-3 was much easier to recognise as it the data was cleaner than SD-1. So, it was suitable to make a new database by mixing the data from both databases SD-1 and SD-3 making the conclusions derived from the experiments and models to be independent from the choice of training set and test set. Hence, a modified version of a combination of both sample sets was used to form the MNIST dataset.

MNIST digit recogniser dataset contains a total of 70,000 samples where half are taken from Special Database 3 of NIST database and the other half from Special Database 1.

Each sample consists of the true label of the image which represents the actual digit that is stored in the image of the hand written sample and the remaining data is for each pixel in the 28x28 image. This results in 785 columns for each image where the first stores the true label and the remaining 784 hold values for each pixel for the given image.

1.6. Test Bed

The project is programmed in Python which is an interpreter based high level language, it is a general purpose programming language and has extensive open source libraries for programming in various fields. We are using Python 3.9 for our project. It is easily readable due to its indentations and the code is reusable because of its extensible nature from defining functions and object oriented approach.

We are using Jupyter Notebook as the platform for coding, debugging, testing and visualising various parts of the project. It is an open-source web based application containing chunks of live code, algorithms, visualisations and text for providing insightful description.

The code was executed on Windows 10 operating system as well as macOS 12.0.1 with minor modifications related to the libraries imported in the code. Since Jupyter notebook is a web-application, it is compatible with most operating systems like Windows, Ubuntu and macOS.

We use pandas 1.3.0 which is a software library for Python and it is used for manipulating data during data analysis. Specifically, it provides data structures and operations for manipulation and analysis of data frames.

NumPy 1.21.1 is another Python library which adds support for large and multidimensional matrices and arrays and enables us to perform a large number of high level mathematical operations on these arrays.

Seaborn 3.4.2, which is a Python library used for data visualisation is based on matplotlib. It is used for visualising data and plotting graphs. It is built on matplotlib and is highly compatible with the data structures provided by pandas.

For further visualisations Matplotlib 3.4.2 is use which is a plotting library for Python. It is compatible with NumPy functions and structures. It uses and object oriented approach for embedding plots into applications.

Additionally, Plotly is used as a Python library for online data analytics and visualisation. It is however compatible with other programming languages like R, Matlab, Perl etc. It provides statistical tools and online graphing features.

Scikit-learn 1.0.1, also known as sklearn is an open source machine learning library made for Python. It has a large number of algorithms for classification, regression and clustering which includes logistic regression, support vector machine, random forest, k-means and many more. This library has been used for making our Logistic Regression model.

Scikit-image 0.17.2, also known as skimage is an open source Python library for image processing. It includes algorithms for colour space manipulation, segmentation, geometric transformations, colour space manipulation, filtering, morphology and more.

1.7. Expected Result

In the first step the dataset needs to be loaded into memory properly and inspected. We expect to see the head of the dataset as the first column holding true values of the digits in the image for that row. We then expect to see the next 784 columns with intensity values for each pixel making the image for each handwritten digit.

Then the images need to be observed to verify the authenticity of the dataset loaded in the previous set. We expect to see handwritten digit images and the corresponding true digit values alongside them. This would let us know that valid true labels are present in the dataset for the images. We also expect to see handwritten digits which may not be distinctly recognisable by the human eye due to variations in writing styles of different sources for these samples.

Then the model needs to be designed and fit to the dataset. A logistic regression model with an accuracy greater than 90% should be made so we may expect the model to give highly accurate predictions for test inputs. So it would be ideal to fit the logistic regression model to the training set and obtain its accuracy scores and optimise the model to get a high accuracy.

Then the attack needs to be designed so it would seem right to have a non-targeted attack that adds perturbations to the test input images of handwritten digits. The images would be perturbed using various values of epsilon. A graph should be obtained to see the relation between the epsilon used for perturbations and the resultant accuracy score of the model. The accuracy should decrease as epsilon is increased because the test image is being modified to fool the model, hence leading to incorrect classifications. Looking at this graph we would be able to determine the accuracy of the model based on the epsilon value chosen to perturb the test images for fooling logistic regression. Based on this information, we can choose the right value for epsilon and create a suitable decrease in the model's accuracy with the non-targeted attack.

Now we would be able to attack the model with selected epsilon perturbations and obtain the fooling results for that epsilon. The resultant model would have a lower accuracy of prediction and we could then inspect the true labels against the predicted labels for the ideal model and the fooled predictions for the attacked model. This would give us the suitable fooling targets for each digit as the count of fooled predictions for each label would reveal the digit it is mistaken for by the model using the selected perturbation.

The weighted parameters for true digits would be closer to other specific digits, that is, each digit would be fooled as some particular digit for the maximum number of cases. We could then recognise these digits as natural fooling targets. Then these natural fooling targets would enable us to design a targeted attack on the model enabling us to attack input test data in such a way that its natural fooling target is the fooled prediction given by the model.

This approach, in turn would give us the ability to distinguish between natural fooling targets for each digit and its non-natural fooling targets and compare their accuracies.

2. ARCHITECTURE

This project has four main stages:

- Making a fairly accurate logistic regression model
- Attacking the model by adding perturbations to the input to recognise natural fooling targets
- Attacking the model for the observed natural fooling targets
- Attacking the model for the observed non-natural fooling targets.

We will see the design of the architecture in more detail in the sections below. The architecture can be viewed in two ways; we'll look at the overview of the project and then we'll look at the detailed working.

2.1. High Level Design

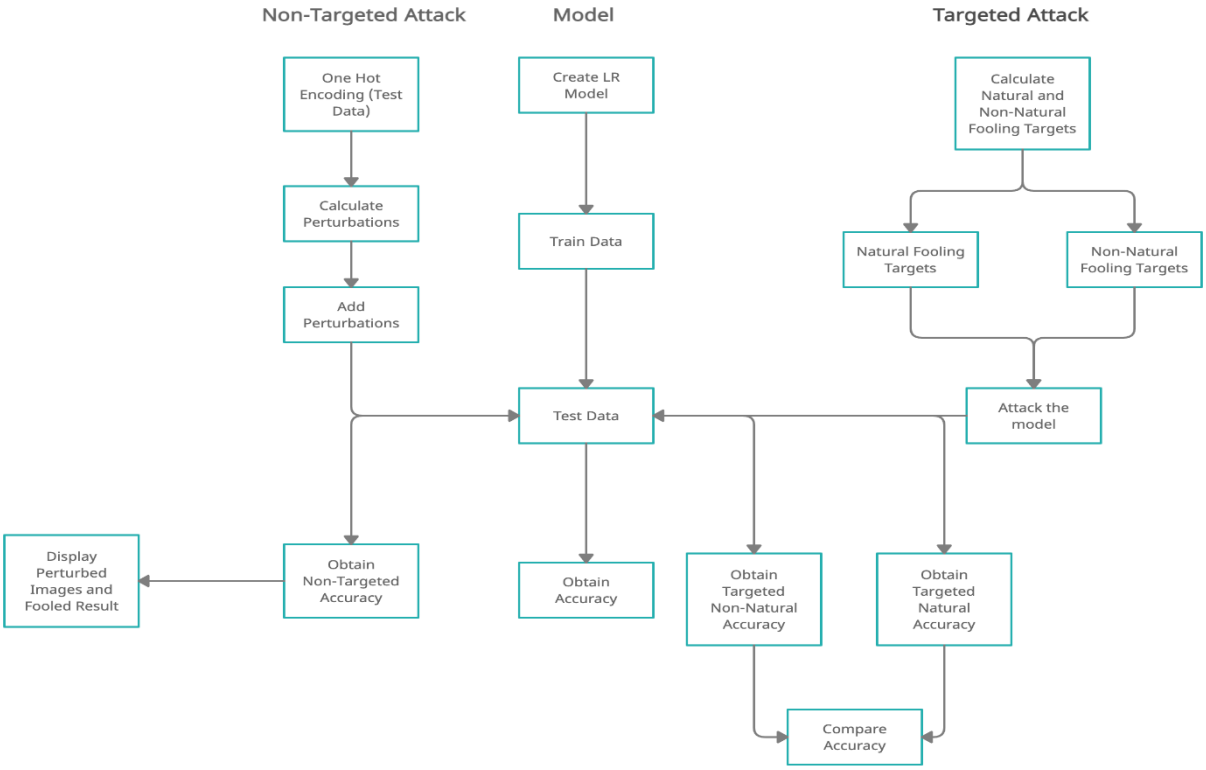

Figure 1: High Level Architecture Diagram

The Logistic Regression model is fit to the train data from MNIST digit recogniser dataset. Then we use the test data to obtain the accuracy of the model made. The parameters in the model are optimised to have a fairly good accuracy of classification so that the initial model that we are working with can be considered as a successful machine learning model giving us a proper base around which an attack can be designed. So the training data will be loaded into memory for us to perform various operations on it. The model is fit to the data and we use test data and obtain accuracy scores from it. One we are satisfied we the model we have made, we move on to designing the attack methods on this model. First we'll try to attack the model directly by changing the input images and observe the outputs and their trends through histograms and heatmaps.

The test data is one hot encoded so that we can approach it as a classification problem. We then calculate the perturbations to be added for each test image based on the epsilon for image distortion. We add perturbations with increasing epsilon values to test images until the accuracy of the model becomes zero and we can say with certainty that for that given epsilon, the model will predict the digit incorrectly. A graph needs to be plotted to observe the relationship between the perturbations added to the test images and the resultant accuracy of the Logistic Regression model. Based on the relationship between epsilon and accuracy, we can choose a specific value of epsilon to be used for the non-targeted attack. The perturbations are added to the test images and the predicted values from the LR model are observed.

We use the predictions for the non-targeted attack to observe the true labels, correct predictions and fooled predictions. Various visualisations may be made to observe any patterns that may be present in the predicted values. The histogram of predictions for each true label is plotted. This data will clearly show the number of times each digit has been predicted by the model and if there are digits that are predicted for a significantly more number of times or if a digit is predicted significantly less as compared to the other digits then that observation can be noted.

A heat-map of this data also needs to be plotted so that a relationship between true labels and fooled predictions over different classes of digits can be observed. This would reveal the tendency of each digit to appear similar as a particular digit to the model after a non-targeted attack by adding perturbations. This information can be used to determine the natural fooling target for each digit. Similarly, each digit in the dataset would have the tendency to appear the least number of times as the fooled prediction for a given digit and hence it can be considered as the non-natural fooling target for the given digit.

So we can determine the natural and non-natural fooling targets for each digit and proceed to designing targeted attacks on our Logistic Regression model by affecting the weights and biases of the test data appropriately to achieve desired fooled predictions while decreasing the accuracy of the LR model.

The test images are first changed to get the natural fooling targets as the prediction from our model and then we obtain the resultant accuracy of the model. In a similar manner, the test data is changed to get the non-natural fooling target of that digit as the predicted output from our classifier. We will obtain the accuracy of the model resulting from non-natural fooling target approach of attack as well. We can finally compare the accuracies obtained in all the cases and tabulate and plot them to note our observations.

2.2. Low Level Design

The low level architecture of the project is shown below.

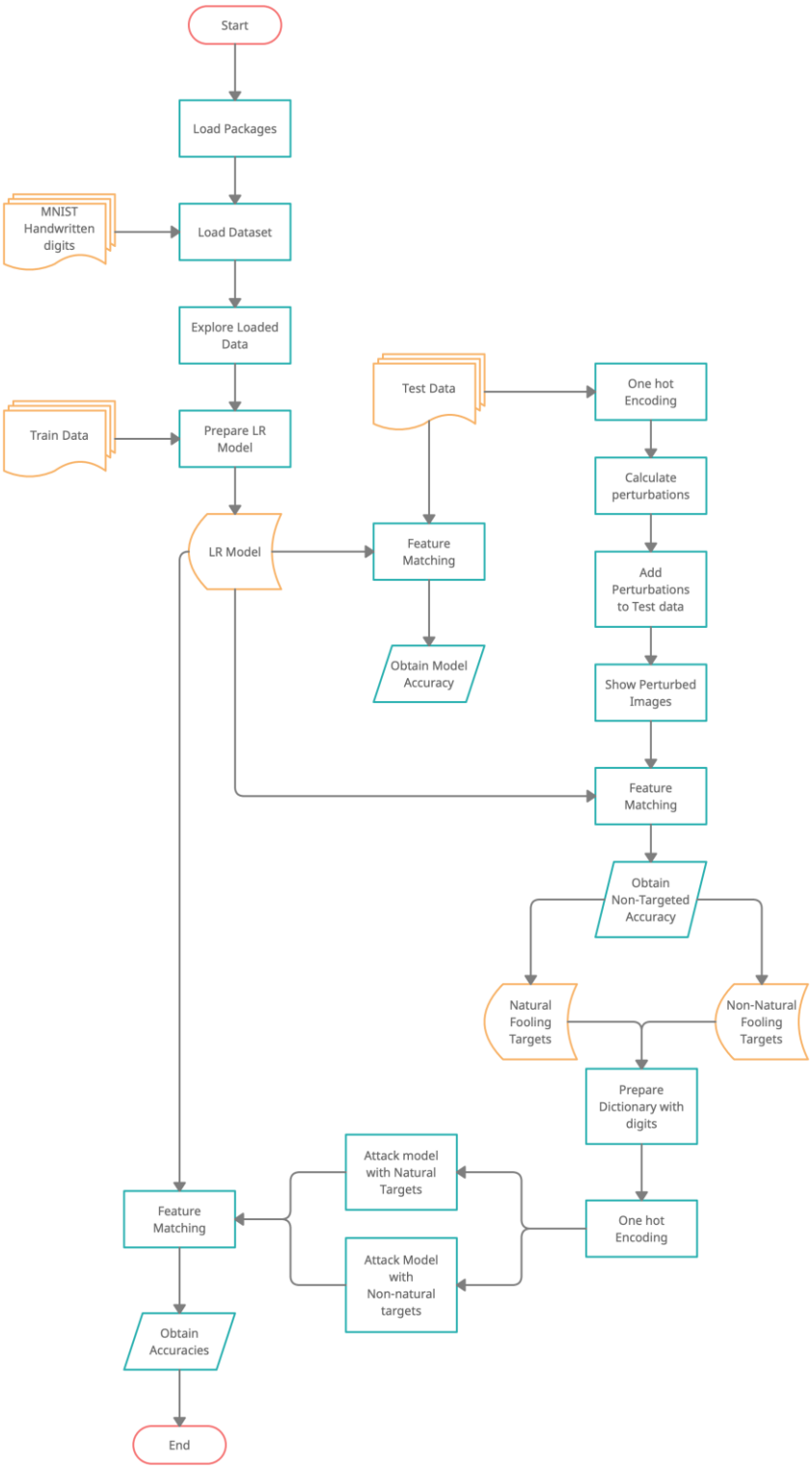

Figure 2: Low Level Architecture Diagram

To understand the low level architecture of this project in further detail, the low level design of each module is explained below.

a. Loading Packages and Exploring the Dataset

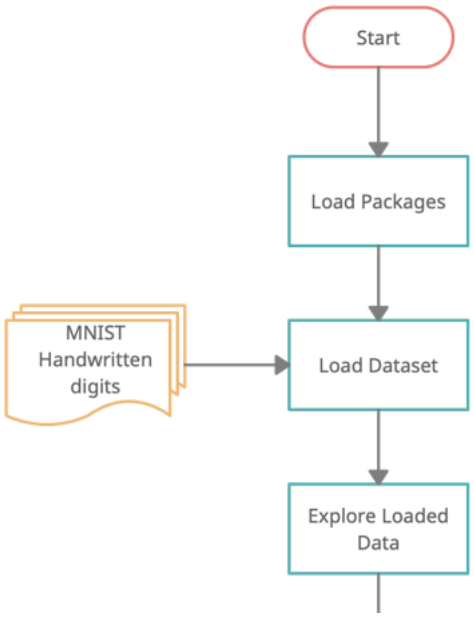

Figure 3:Loading Packages and Exploring the Dataset

We start by loading all the python libraries and packages required for this project. The libraries imported are;

- Scikit-learn or sklearn for the Logistic Regression model,
- Pandas for data structures,
- Numpy for mathematical operations,
- Seaborn for visualisations and graphs,
- Plotly for additional plots,
- Matplotlib for making graphs and visualisations,

- And Scikit-image or skimage for image operations.

Then we load the dataset into memory for storing it into arrays and data frames to apply mathematical operations on it. The data is also explored through some basic methods; first by observing the initial few rows of the dataset, then by using the pixel information to produce images for few of the rows in the beginning. The count of each true label in the dataset is also plotted as histogram to ensure the balance of the samples.

b. Preparing LR Model

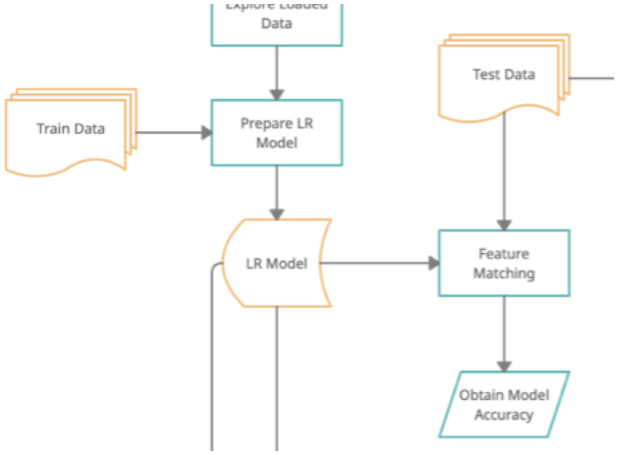

Figure 4: Preparing the Logistic Regression Model

The Logistic Regression model is imported from sklearn and the model is fit to the training data. The test data is used to generate predictions from the model and obtain the accuracy of the LR model thus created.

c. Non-Targeted Attack

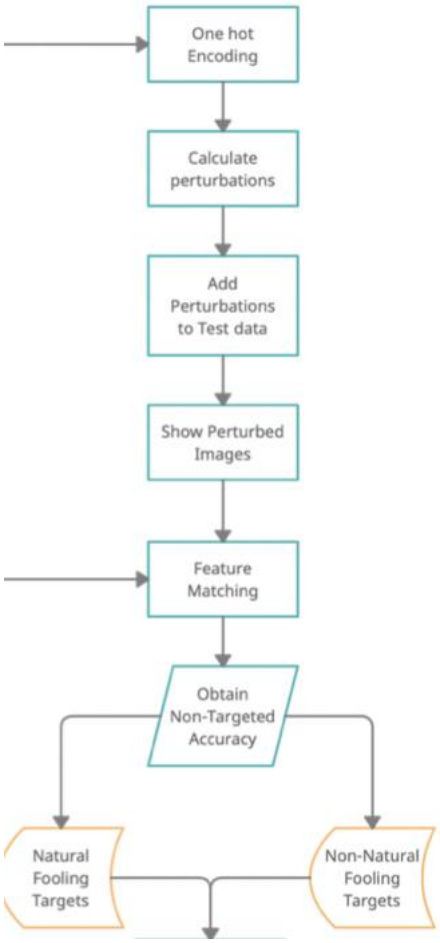

Figure 5: Non-targeted attack

For the non-targeted attack, the model fit to the training data is used by adding perturbations to the input test images. Various values of epsilon are used to calculate perturbation to be added to the test image and the images are observed. A relationship between the increasing value of epsilon and the accuracy of the LR model is plotted and a specific value of epsilon is selected. Then the model is attacked using these perturbed test images and the resultant accuracy of the model is obtained. These predictions are used to determine the natural and non-natural fooling targets for each digit.

d. Natural and Non-Natural Targeted Attack

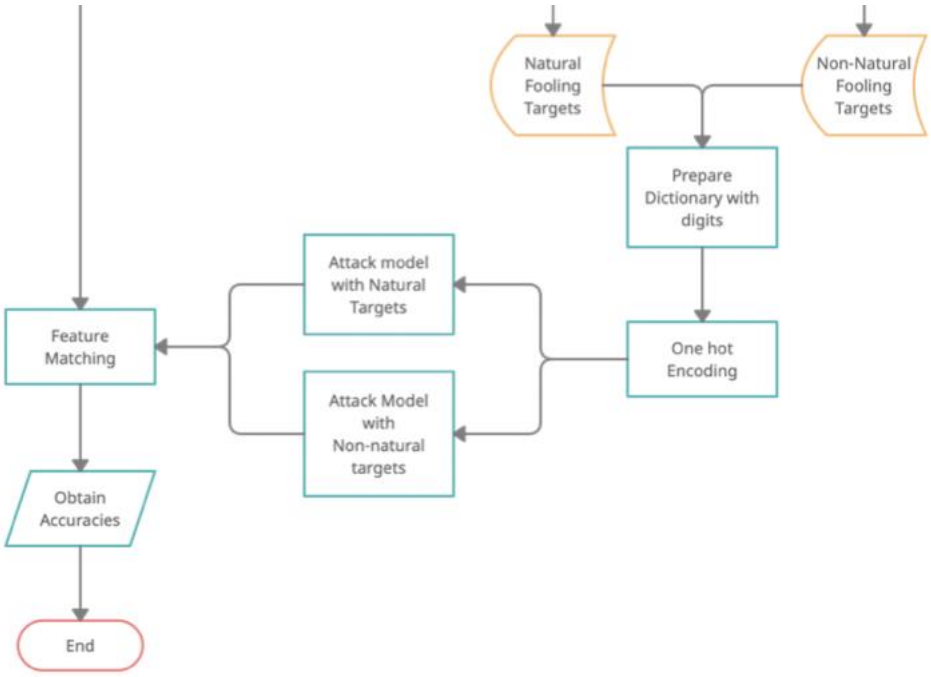

Figure 6: Natural and Non Natural Targeted Attacks

The natural fooling and non-natural fooling targets for each digit are stored in dictionaries and the targets are one-hot encoded. Then the model is attacked using these values to obtain the accuracies associated with the natural fooling target attack on the model and the non-natural fooling target attack on the LR model. These accuracies are compared and plotted on a graph to observe the trend between the increasing epsilon value for each type of attack.

3. Implementation

The implementation of our project was done using python and various libraries made for python and the whole project was run as a Jupyter Notebook file.

3.1. Algorithms and Procedure of Implementation

The Logistic Regression model from Scikit-learn is used with a few other functions from the same library as well, such as to obtain the accuracy metrics of the model and splitting the dataset into train and test data.

The Logistic Regression Model is trained using the MNIST digit recognizer datset and the trained model is then tested to obtain an initial reading of the accuracy of the model. The parameters in the model are optimised to have a fairly good accuracy of classification so that the initial model that we are working with can be considered as a successful machine learning model giving us a proper base around which an attack can be designed. The training data is trained by the Multinomial Logistic Regression model and it is tested with the test data to obtain the initial accuracy score to see that we are fooling an accurate ML model.

After the initial LR model is obtained we can proceed with the creation of our attack classes for various forms of attacks on the model namely, non-targeted attacks, targeted attacks with natural fooling targets and targeted attacks with non natural fooling targets.

The test data is one hot encoded to convert it into1s and 0s so that we can approach the given problem as a classification problem. We then calculate the perturbations to be added for each test image based on the epsilon for image distortion. We add perturbations with increasing epsilon values to test images until the accuracy of the model becomes zero and we can say with certainty that for that given epsilon, the model will predict the digit incorrectly. A graph needs to be plotted to observe the relationship between the perturbations added to the test images and the resultant accuracy of the Multinomial

Logistic Regression model.. The perturbations are added to the test images and the predicted values from the LR model are observed.

On completing the non-targeted attacks, we can observe the true labels, correct predictions and fooled predictions in the form of graphs and heatmaps. Various visualisations may be made to observe any patterns that may be present in the predicted values. The histogram of predictions for each true label is plotted. This data will clearly show the number of times each digit has been predicted by the model and if there are digits that are predicted for a significantly greater number of times or if a digit is predicted significantly less as compared to the other digits then that observation can be noted.

A most effective visualization would be a heat map which would give a complete overview of the true label and the fooling targets used for those true labels. From the heatmap we can obtain the values of the digits which are most used as fooling targets for a particular digit and these become the natural fooling targets and the opposite which are the least used fooling targets for a particular digit which becomes the non-natural fooling targets. So we can determine the natural and non-natural fooling targets for each digit and proceed to designing targeted attacks on our Logistic Regression model by affecting the weights and biases of the test data appropriately to achieve desired fooled predictions while decreasing the accuracy of the LR model.

The test images are first changed to get the natural fooling targets as the prediction from our model and then we obtain the resultant accuracy of the model. In a similar manner, the test data is changed to get the non-natural fooling target of that digit as the predicted output from our classifier. We will obtain the accuracy of the model resulting from non-natural fooling target approach of attack as well. We can finally compare the accuracies obtained in all the cases and tabulate and plot them to note our observations.

Other functions for plotting graphs and charts are used from appropriate python libraries. Array operations and data frame processes are imported from corresponding libraries for python as well.

The image operations such as importing image data from dataframes and displaying image data in visual format is being done using scikit-image library for python.

We can consider the sequence below as the basic algorithm followed for the project:

- Import Python Libraries (Sklearn, Numpy, Pandas, Seaborn, Matplotlib, Skimage, Plotly)
- Load Dataset into memory
- Store data in data frame and display first few rows
- Drop true label column
- Split remaining data into train and test 60-40 respectively
- Display first few rows as images with corresponding true labels
- Define a class with all the necessary functions
- Fit model to training data
- Use prepare function from the class to obtain initial model accuracy
- Explore weight vectors generated by the model as compared to number of classes
- Create one hot encoded targets for the test data
- Call attack to maximum epsilon function from the class with upper limit 30 for epsilon
- Obtain the accuracy scores of the model for all epsilon values
- Plot relationship between Accuracy and epsilon
- Select epsilon with intermediate impact on accuracy score of model
- Obtain prediction data for selected epsilon
- Display All predictions
- Display successfully fooled predictions
- Display sample test digit image with increasing perturbations and predicted value with true label

- Plot count of successfully fooled predictions
- Plot heatmap between true labels and fooled predictions
- Make dictionary of natural fooling targets
- Prepare dictionary of non-natural fooling targets
- Call attack to maximum epsilon using natural fooling targets
- Obtain accuracy of the model
- Call attack to maximum epsilon using non-natural fooling targets
- Obtain accuracy of the model
- Compare all the accuracies obtained

3.2. Mathematical Models

1. Logistic Regression

Logistic Regression is a supervised machine learning algorithm that is used for classification problems. It is used to predict the probability of a particular variable. There exists only two possible classes for the dependent variable as its nature is dichotomous. It is binary and its data contains either 1 which implies success or 0 implying failure. A logistic regression model predicts the probability that Y=1 as a function of X. It is a very commonly used machine learning algorithm and has its applications in many fields. The logistic regression model utilized by us in our project is a multinomial logistic regression model. A multinomial logistic regression model is one which is widely used for classification purposes. In this kind of classification, the dependent variable can have multiple unordered types or have quantifiers without significance.

Logistic Regression in its simplest terms is a predictive analysis algorithm that is based on probability. Logistic Regression is a more complex version of Linear regression in that it uses a more complex cost function namely the sigmoid function. The sigmoid function is used to map the predicted values to the probabilities that have been given. It basically maps values between 0 and 1. The sigmoid function is represented by f(x)= 1/(1+e^(-x)).

Logistic Regression models also have a more complex hypothesis representation as compared to the linear regression counterpart.

Multinomial Logistic Regression is a powerful analysis tool as it is free from the assumptions of things such as linearity or normality of the data. The solver used by the multinomial logistic regression function is th4 lbfgs solver which stands for Limited Memory Broyden Fletcher Goldfarb Shanno. This solver approximates the second derivative matrix updates with gradient evaluations. It is an extremely memory efficient solver method and is the default solver method for multinomial logistic regression. In our logistic regression model, the inputs and classes are considered to be independent of each other and they are assumed to be identically distributed. Multinomial logistic regression assumes the inputs and features to be independent and equally distributed in probability space. The target is to maximize the likelihood that the digits will match. A softmax activation function for multinomial probability prediction is used to obtain the output for the model.

2. Non-targeted attack

The objective of the non targeted attack is to obtain the maximum possible deviation from the original digit which is referred to as discrepancy. The need is to increase the chances that the model incorrectly predicts the digits in the dataset. This is basically the reverse of the method used normally where we would want to maximize the probability of the correct output. Hence we need to reverse the logistic regression model. To make sure that the digits are incorrectly recognized, we can add imperfections or convolve the images with perturbations to fool the machine learning model. We use a perturbation function to add perturbations to the image based on the specified value of epsilon that has been passed. The more the value of epsilon the more perturbed the resultant image will be resulting in more likelihood that it is incorrectly predicted by the LR Model. Furthermore, gradient ascent technique is also used in conjunction with the perturbations to obtain best fooling results. After adding these steps, we can run the model and test the accuracy of

the model with the values of epsilon to obtain the visualization of the decrease in accuracy from the non-targeted attacks

3. Targeted Attacks with Natural Fooling Targets

Once the non-targeted attacks have been completed, we can obtain the plots and charts and the heatmap of the true label compared to the fooling target used. From this we obtain for each digit, the most commonly used fooling target for that particular digit. Now these values can be considered to be the most effective fooling targets for those respective digits and we can now attack those particular digits to be fooled by their most effective fooling targets.

4. Targeted Attacks with Non- Natural Fooling Targets

Like the natural fooling targets procedure, once the non-targeted attacks have been completed, we can obtain the plots and charts and the heatmap of the true label compared to the fooling target used. From this we obtain for each digit, the least used fooling target for that digit which is the exact opposite of what is being done in the above method. Now these values can be the least effective fooling targets for those respective digits, and we can now attack those particular digits to be fooled by their least effective fooling targets and observe how it effects the accuracy of the model.

4. Results and Discussion

4.1. Implementation with Coding

1. Loading the Packages

Loading packages ¶

In [1]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import datasets as dt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

import pandas as pd
import numpy as np
import seaborn as sns


import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls

import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline

from skimage.io import imread, imshow

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
```

Figure 7: Code Snippet of Loading of Packages

2. Loading Data

Loading data

[2]:
```
df = pd.read_csv("digit-recognizer/train.csv")
df.head(10)
```

Figure 8: Code Snippet of Loading dataset

```
In [3]: y = df.label.values
        X = df.drop("label",axis=1).values
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)

In [4]: fig1, ax1 = plt.subplots(1,15, figsize=(15,10))
        for i in range(15):
            ax1[i].imshow(X_test[i].reshape((28,28)), cmap="gray_r")
            ax1[i].axis('off')
            ax1[i].set_title(y_test[i])
```

Figure 9: Code snippet for train test split and displaying sample elements

Loading the train data from the digit recognizer dataset and view the first 10 elements. Then splitting the train and test data set followed by the display of 15 digits in the dataset.

3. Attack Class

```
In [5]: class Attack:

            def __init__(self, model):
                self.fooling_targets = None
                self.model = model

            def prepare(self, X_train, y_train, X_test, y_test):
                self.images = X_test
                self.true_targets = y_test
                self.num_samples = X_test.shape[0]
                self.train(X_train, y_train)
                print("Model training finished.")
                self.test(X_test, y_test)
                print("Model testing finished. Initial accuracy score: " + str(self.initial_score))

            def set_fooling_targets(self, fooling_targets):
                self.fooling_targets = fooling_targets

            def train(self, X_train, y_train):
                self.model.fit(X_train, y_train)
                self.weights = self.model.coef_
                self.num_classes = self.weights.shape[0]

            def test(self, X_test, y_test):
                self.preds = self.model.predict(X_test)
                self.preds_proba = self.model.predict_proba(X_test)
                self.initial_score = accuracy_score(y_test, self.preds)

            def create_one_hot_targets(self, targets):
                self.one_hot_targets = np.zeros(self.preds_proba.shape)
                for n in range(targets.shape[0]):
                    self.one_hot_targets[n, targets[n]] = 1
```

Figure 10: Code snippet for Attack Class

```
    def attack(self, attackmethod, epsilon):
        perturbed_images, highest_epsilon = self.perturb_images(epsilon, attackmethod)
        perturbed_preds = self.model.predict(perturbed_images)
        score = accuracy_score(self.true_targets, perturbed_preds)
        return perturbed_images, perturbed_preds, score, highest_epsilon

    def perturb_images(self, epsilon, gradient_method):
        perturbed = np.zeros(self.images.shape)
        max_perturbations = []
        for n in range(self.images.shape[0]):
            perturbation = self.get_perturbation(epsilon, gradient_method, self.one_hot_targets[n], self.preds_proba[n])
            perturbed[n] = self.images[n] + perturbation
            max_perturbations.append(np.max(perturbation))
        highest_epsilon = np.max(np.array(max_perturbations))
        return perturbed, highest_epsilon

    def get_perturbation(self, epsilon, gradient_method, target, pred_proba):
        gradient = gradient_method(target, pred_proba, self.weights)
        inf_norm = np.max(gradient)
        perturbation = epsilon/inf_norm * gradient
        return perturbation

    def attack_to_max_epsilon(self, attackmethod, max_epsilon):
        self.max_epsilon = max_epsilon
        self.scores = []
        self.epsilons = []
        self.perturbed_images_per_epsilon = []
        self.perturbed_outputs_per_epsilon = []
        for epsilon in range(0, self.max_epsilon):
            perturbed_images, perturbed_preds, score, highest_epsilon = self.attack(attackmethod, epsilon)
            self.epsilons.append(highest_epsilon)
            self.scores.append(score)
            self.perturbed_images_per_epsilon.append(perturbed_images)
            self.perturbed_outputs_per_epsilon.append(perturbed_preds)
```

Figure 11: Code snippet for Attack Class Part 2

The above classes consist of the attack classes as well as the preparation, train and test function to obtain the initial accuracy of the multinomial logistic regression model. Another function provided is to perform one hot encoding to covert to classification problem. Another function is the attack function which uses the perturbed images to fool the target. More functions which perform and store the perturbed images are given and a final function which attacks the model from lowest value of epsilon (least perturbation) to max epsilon (most perturbation).

4. Weights and Gradient Methods

Weight and Gradient methods

In [6]:
```
def calc_output_weighted_weights(output, w):
    for c in range(len(output)):
        if c == 0:
            weighted_weights = output[c] * w[c]
        else:
            weighted_weights += output[c] * w[c]
    return weighted_weights

def targeted_gradient(foolingtarget, output, w):
    ww = calc_output_weighted_weights(output, w)
    for k in range(len(output)):
        if k == 0:
            gradient = foolingtarget[k] * (w[k]-ww)
        else:
            gradient += foolingtarget[k] * (w[k]-ww)
    return gradient

def non_targeted_gradient(target, output, w):
    ww = calc_output_weighted_weights(output, w)
    for k in range(len(target)):
        if k == 0:
            gradient = (1-target[k]) * (w[k]-ww)
        else:
            gradient += (1-target[k]) * (w[k]-ww)
    return gradient
```

Figure 12: Weights and Gradients Method

These are the gradient and weight calculation methods.

5. Training the model

```
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', fit_intercept=False)
```

```
attack = Attack(model)
attack.prepare(X_train, y_train, X_test, y_test)
```

Figure 13: Training Model

This trains the Logistic Regression model and we can display the accuracy.

6. Non Targeted Attack

```
attack.create_one_hot_targets(y_test)
attack.attack_to_max_epsilon(non_targeted_gradient, 30)
non_targeted_scores = attack.scores
```

```
sns.set()
plt.figure(figsize=(10,5))
plt.plot(attack.epsilons, attack.scores, 'g*')
plt.ylabel('accuracy_score')
plt.xlabel('epsilon')
plt.title('Accuracy score breakdown - non-targeted attack');
```

Figure 14: Non-targeted Attack

7. Heatmap to view the natural and non-natural fooling targets

```
attacktargets = example_results.loc[example_results.y_true != example_results.y_fooled].groupby(
    'y_true').y_fooled.value_counts()
counts = example_results.loc[example_results.y_true != example_results.y_fooled].groupby(
    'y_true').y_fooled.count()
attacktargets = attacktargets/counts * 100
attacktargets = attacktargets.unstack()
attacktargets = attacktargets.fillna(0.0)
attacktargets = attacktargets.apply(np.round).astype(np.int)
```

```
f, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(attacktargets, annot=True, ax=ax, cbar=False, square=True, cmap="Reds", fmt="g");
ax.set_title("How often was y_true predicted as some y_fooled digit in percent?");
```

Figure 15: Heatmap for natural and non-natural fooling targets

8. Obtaining the natural and non natural fooling targets

```
natural_targets_dict = {}
non_natural_targets_dict = {}
for ix, series in attacktargets.iterrows():
    natural_targets_dict[ix] = series.argmax()
    non_natural_targets_dict[ix] = series.drop(ix).argmin()
```

Figure 16: Getting natural and non-natural fooling targets

9. Natural and Non-natural Targeted Attacks

```
natural_foolingtargets = np.zeros((y_test.shape[0]))
non_natural_foolingtargets = np.zeros((y_test.shape[0]))

for n in range(len(natural_foolingtargets)):
    target = y_test[n]
    natural_foolingtargets[n] = natural_targets_dict[target]
    non_natural_foolingtargets[n] = non_natural_targets_dict[target]
```

```
attack.create_one_hot_targets(natural_foolingtargets.astype(np.int))
attack.attack_to_max_epsilon(targeted_gradient, 30)
natural_scores = attack.scores
attack.create_one_hot_targets(non_natural_foolingtargets.astype(np.int))
attack.attack_to_max_epsilon(targeted_gradient, 30)
non_natural_scores = attack.scores
```

Figure 17: Natural and Non-natural Targeted attacks

10. Comparison of accuracy of Natural and Non-natural Accuracy with increasing epsilon values

```
plt.figure(figsize=(10,5))
nf, = plt.plot(attack.epsilons, natural_scores, 'g*', label='natural fooling')
nnf, = plt.plot(attack.epsilons, non_natural_scores, 'b*', label='non-natural fooling')
plt.legend(handles=[nf, nnf])
plt.ylabel('accuracy_score')
plt.xlabel('epsilon')
plt.title('Accuracy score breakdown: natural vs non-natural targeted attack');
```

Figure 18: Comparison of accuracy between the two

4.2. Result

1. Loading data

Loading data

```
In [2]: df = pd.read_csv("digit-recognizer/train.csv")
        df.head(10)
```

Out[2]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pi
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
6	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
7	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
8	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
9	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	

10 rows × 785 columns

Figure 18: Loading the data

2. Sample digits in the dataset

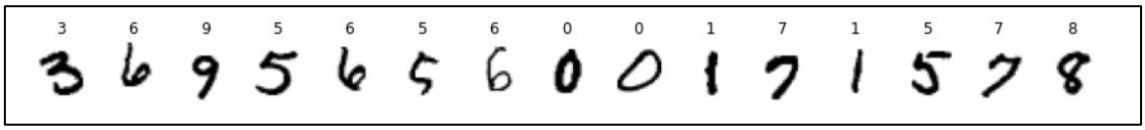

Figure 19: Sample digits

3. Initial accuracy of the multinomial LR model

Training the model

```
: model = LogisticRegression(multi_class='multinomial', solver='lbfgs', fit_intercept=False)

: attack = Attack(model)
  attack.prepare(X_train, y_train, X_test, y_test)

  Model training finished.
  Model testing finished. Initial accuracy score: 0.909345238095238
```

Figure 20: Initial accuracy of multinomial LR model

4. Non-Targeted Attack Accuracy

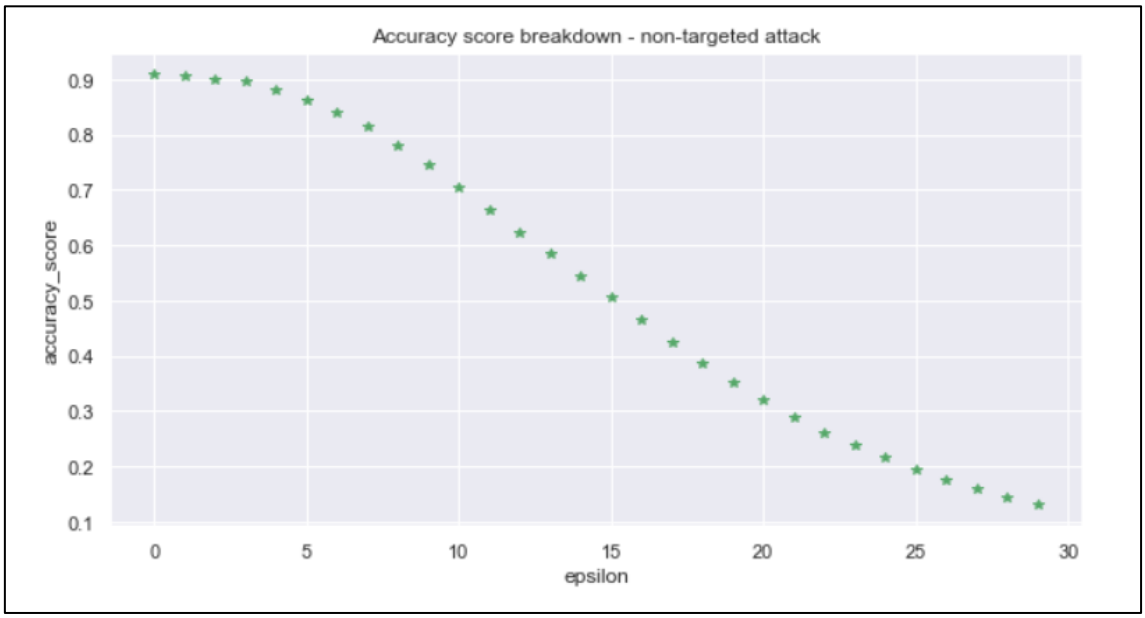

Figure 21: Non-targeted attack acuracy

5. Sample of results from the non- targeted attack

	y_true	y_fooled	y_predicted	id
0	3	8	3	0
1	6	6	6	1
2	9	9	9	2
3	5	8	5	3
4	6	6	6	4

	y_true	y_fooled	y_predicted	id
0	3	8	3	0
3	5	8	5	3
5	5	9	0	5
6	6	2	6	6
14	8	5	8	14

Figure 22: Sample of results from non-targeted attack

Table on list shows all the true labels, fooled labels and predicted labels and table on the right shows the labels only where the true label is not equal to the fooled label.

6. **Fooling** targets **used per epsilon for sample image**

Figure 23: Sample output per epsilon value

7. Sample result for perturbation for epsilon 16

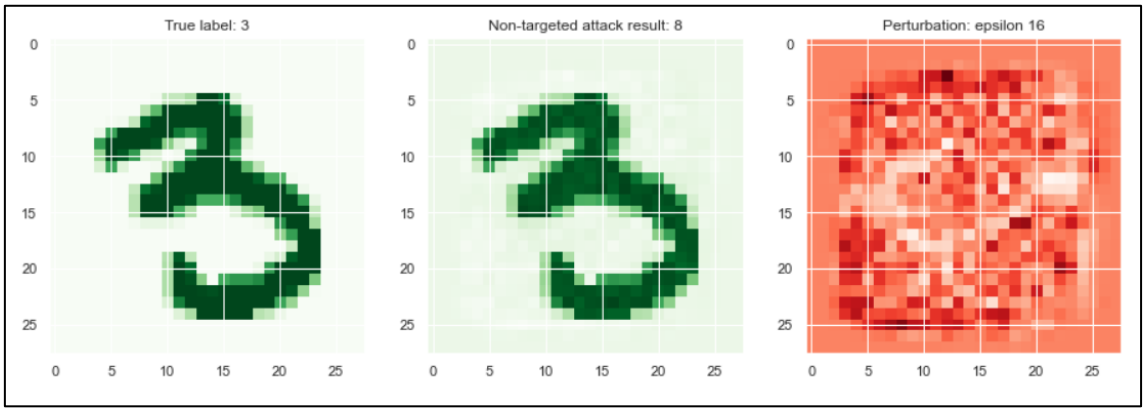

Figure 24: Sample results of perturbation

8. Histogram with counts of y_fooled, y_predicted and y_true

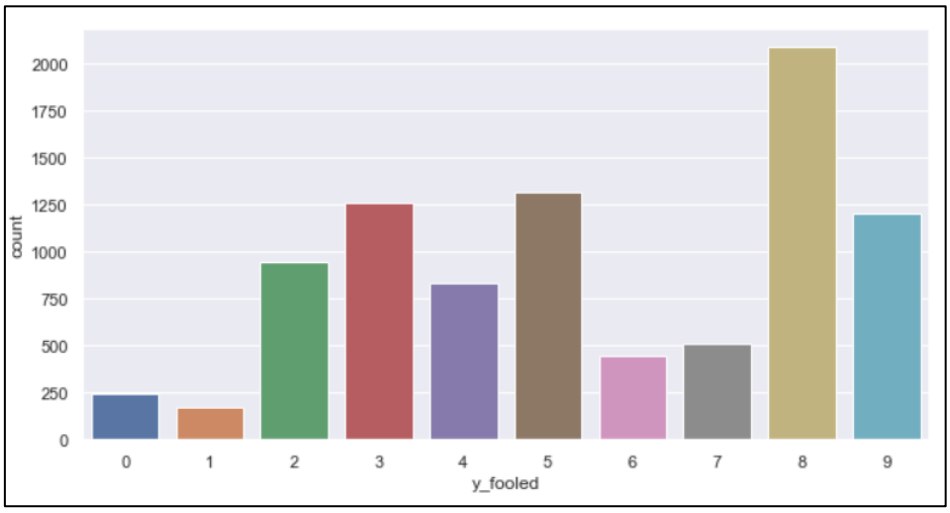

Figure 25: count of y_fooled

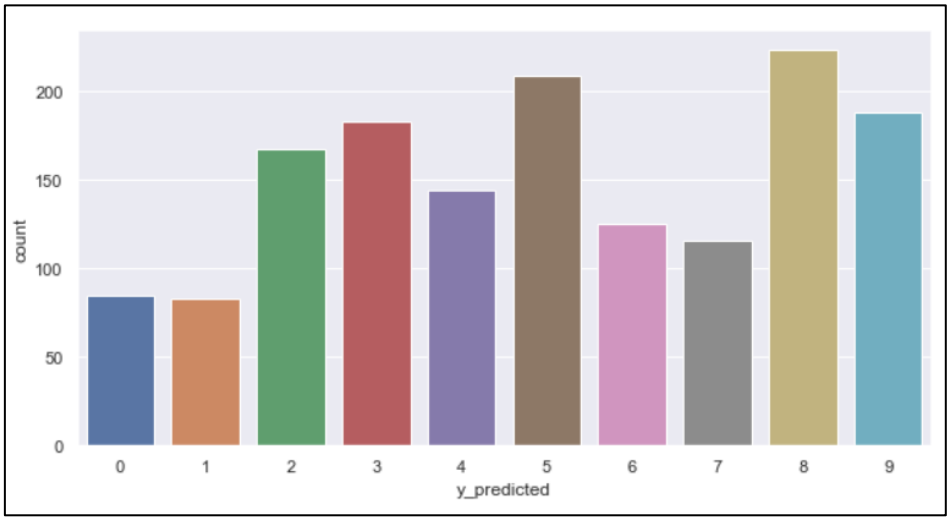

Figure 26: Count of y_predicted

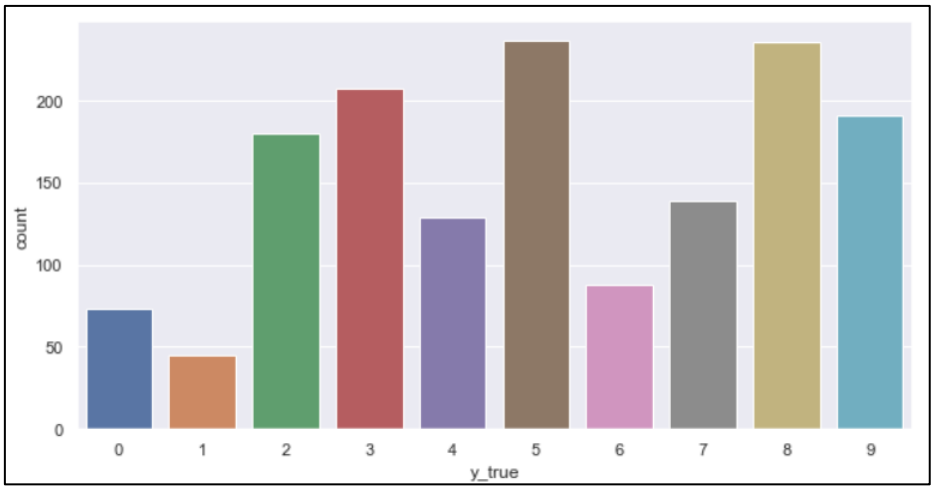

Figure 27: Count of y_true

9. Heatmap displaying the percentage of fooling target used for the given true digit

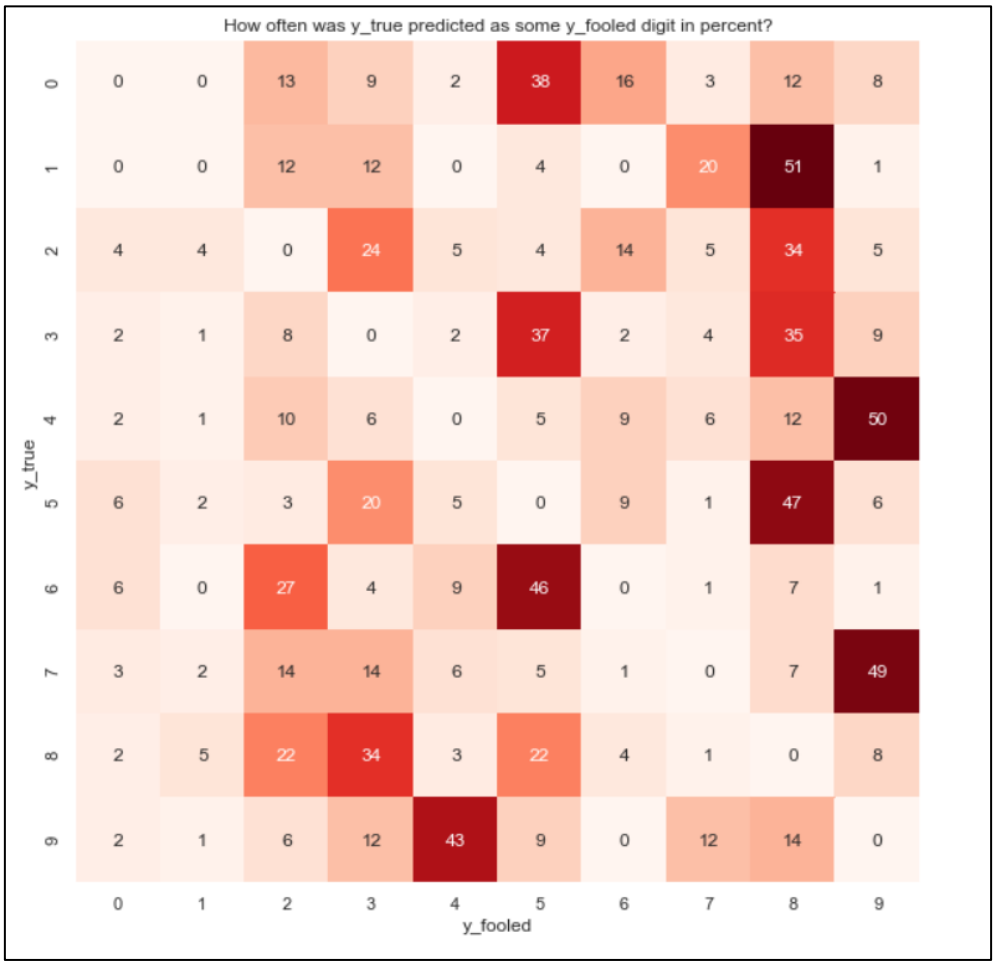

Figure 28: Heatmap of percentage of fooling target used for the given true digit

10. Natural and Non-natural Targeted Attack Accuracy Scores with increasing epsilon

```
natural_scores
```

```
[0.909345238095238,
 0.9044047619047619,
 0.8961904761904762,
 0.8870833333333333,
 0.8773809523809524,
 0.8629166666666667,
 0.8448809523809524,
 0.8233333333333334,
 0.7999404761904761,
 0.7747619047619048,
 0.7445833333333334,
 0.7110714285714286,
 0.6754166666666667,
 0.63875,
 0.5989285714285715,
 0.5591071428571428,
 0.5178571428571429,
 0.4764285714285714,
 0.43857142857142856,
 0.40279761904761907,
 0.37148809523809523,
 0.3410714285714286,
 0.3145833333333333,
 0.2920238095238095,
 0.2681547619047619,
 0.24720238095238095,
 0.23083333333333333,
 0.2125,
 0.19505952380952382,
 0.18154761904761904]
```

```
non_natural_scores
```

```
[0.909345238095238,
 0.9097619047619048,
 0.9094642857142857,
 0.9088690476190476,
 0.9066071428571428,
 0.9020833333333333,
 0.8966071428571428,
 0.8891666666666667,
 0.8794642857142857,
 0.861845238095238,
 0.8392261904761905,
 0.8151785714285714,
 0.7904761904761904,
 0.7633928571428571,
 0.735,
 0.7085714285714285,
 0.6801190476190476,
 0.6502976190476191,
 0.6202380952380953,
 0.5886309523809524,
 0.55875,
 0.5266666666666666,
 0.4936309523809524,
 0.4620238095238095,
 0.43273809523809526,
 0.4019642857142857,
 0.376547619047619,
 0.35160714285714284,
 0.3289285714285714,
 0.3086904761904762]
```

Figure 29: Natural and Non-natural Targeted Attack Accuracy Scores with increasing epsilon

11. Graphical Representation of comparison of accuracy of natural and non-natural targeted attacks with increasing epsilon

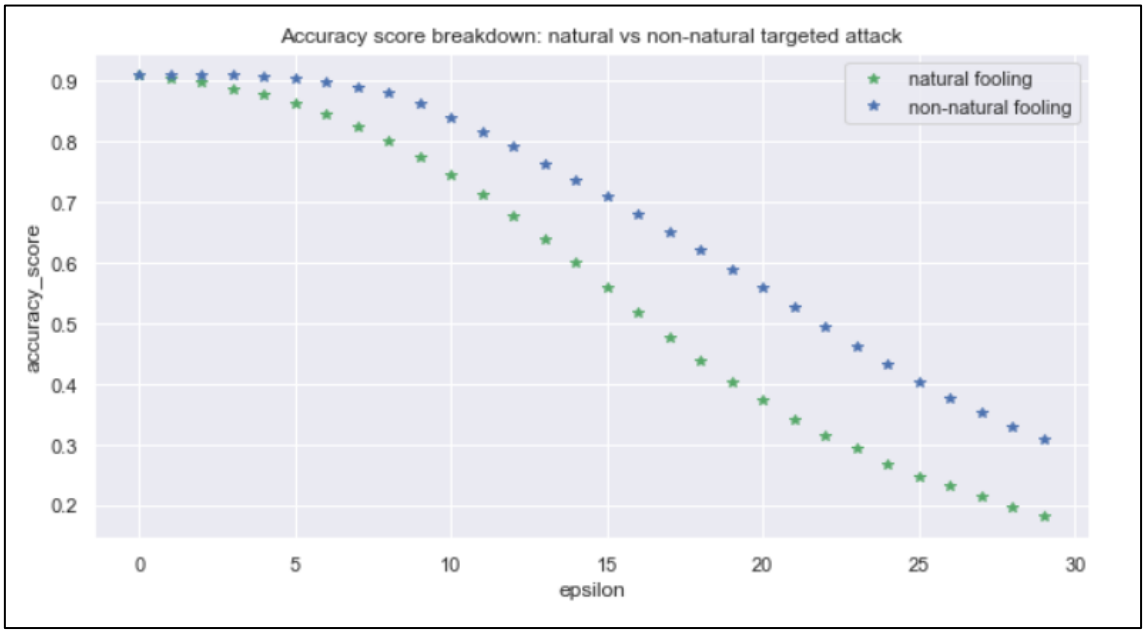

Figure 30: Graphical Representation of comparison of accuracy of natural and non-natural targeted attacks with increasing epsilon

4.3. Conclusion

We have successfully created three attacks on a multinomial regression model using digit recognizer database. These three attacks are namely non-targeted attacks, targeted attacks with natural fooling targets and targeted attacks with non-natural fooling targets. We have obtained, analysed and plotted the impact of these attacks on the given ML model and obtained the results in terms of impact on the accuracy of the model. We can clearly see that on increasing the number of perturbations in the image which is on increasing the epsilon value, we can fool the ML model much more frequently. From the non-targeted attacks, we got an overview of the most and least frequently used fooling targets for a particular digit. These fooling targets become the natural and non-natural fooling targets for the targeted attacks. The accuracies for these attacks were also compared and it can be established that the accuracy of the model decreases with a higher rate as we increase the epsilon value for natural fooling targets as compared to when the model is attacked for non-natural fooling. These findings show the realistic threat to classification models

which may be used in various cybersecurity applications and a real-world scenario where an efficient attack to fool the model for various types of targets was created.

5. References

[1] F. O. Olowononi, D. B. Rawat, and C. Liu, “Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS,” *IEEE Commun. Surv. Tutorials*, vol. 23, no. 1, pp. 524–552, 2021, doi: 10.1109/COMST.2020.3036778.

[2] S. Zhang, X. Xie, and Y. Xu, “A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity,” *IEEE Access*, vol. 8, pp. 128250–128263, 2020, doi: 10.1109/ACCESS.2020.3008433.

[3] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, “Machine Learning Security: Threats, Countermeasures, and Evaluations,” *IEEE Access*, vol. 8, pp. 74720–74742, 2020, doi: 10.1109/ACCESS.2020.2987435.

[4] N. Akhtar and A. Mian, “Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey,” *IEEE Access*, vol. 6, pp. 14410–14430, 2018, doi: 10.1109/ACCESS.2018.2807385.

[5] K. Sadeghi, A. Banerjee, and S. K. S. Gupta, “A System-driven taxonomy of attacks and defenses in adversarial machine learning,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 4, no. 4, pp. 450–467, 2020, doi: 10.1109/TETCI.2020.2968933.

UNICHECK

User name:
MOHIT SUHASARIA 19BCE2167

Check ID:
57823167

Check date:
09.12.2021 00:08:53 IST

Check type:
Doc vs Internet + Library

Report date:
09.12.2021 00:10:43 IST

User ID:
77900

File name: **ISAA Report.docx**

Page count: **44** Word count: **8214** Character count: **50472** File size: **1.83 MB** File ID: **68750897**

Text modifications detected (similarity score might be affected)

3.51% Matches

Highest match: **0.42%** with Internet source (**https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html?sourc..**

3.17% Internet sources	193	Page 46
0.67% Library sources	41	Page 47

0% Quotes

Exclusion of quotes is off

References	1	Page 47

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Suspicious formatting	12 Pages

Assessing The Impact of Cybersecurity Attacks On Machine Learning Model

A PROJECT REPORT

Submitted by

Moukhik Misra (19BCE2190)

Mohit Suhasaria (19BCE2167)

CSE3501 INFORMATION SECURITY ANALYSIS AND AUDIT

Slot – F2

Computer Science and Engineering

DECEMBER 2021

1. Problem Statement

Recently machine learning models have been applied to predict and detect cybersecurity attacks in various scenarios. However, there is a possibility that the attacker may target the model itself. Cybersecurity in recent times relies heavily upon machine learning techniques to detect attacks however attacking the models themselves haven't been a topic of huge discussion. In this project we look at the effect of attacks on the machine learning model itself. We will be assessing the impact of both targeted and non-targeted attacks on a logistic regression model that predicts numbers based on given image and try to attack it by various methods to understand the impact that these sorts of attacks have on the accuracy of made machine learning model. Statistical analysis of the results will be conducted, and the effects of the attacks will be shown and explained.

1.1. Idea

Logistic regression, also known as logit model is a kind of statistical analysis which is frequently used in machine learning applications for predictive analysis modelling. In this approach the dependent variable is categorical and depending on the number of categorical variables it is further divided into binary logistic regression and multinomial logistic regression. It is used to understand the relationship between categorical variables and independent variables by computing probabilities through a logistic regression equation.

Say, if we use a linear regression model for classifying data then a threshold must be set to distinguish between classes. So if the predicted continuous value falls in a range outside the threshold of its true class then it would lead to inaccurate classification. This means a model suitable for classification is needed, such as logistic regression.

Sophisticated attacks on a system exploit various vulnerabilities to gain access to restricted information. It becomes essential for a cyber security administrator to establish associations between attacks and the vulnerabilities present. Attacks on a network involving data breach are rare events, so a more common event that can be

considered in this context is a warning which is in general a more common class of cyber security incident. Cyber security administrators often make use of statistic models to reveal vulnerabilities in the system that factor significantly in the occurrence of cyber security incidents. These models are also used to predict the probability with which said incident would occur based on given system vulnerabilities according to real world data.

Logistic regression models are used for classifying incidents based on system vulnerabilities and other independent variables that may be considered as affecting factors. This includes data preparation through imputation and aggregation. The vulnerability related data is converted to host based data covering all factors associated with the cyber environment. The logistic regression model evaluates the associations with the cyber environment and incidents. The model is validated after performing various statistical analyses and is fit to predict the occurrence of incidents on a system based on cyber environment and other contributing factors which includes the operating system, mode of management and host type.

Since a lot of threat detection systems and incident prediction systems employ machine learning models for multiclass classification such as multinomial logistic regression, it becomes necessary for cyber security analysts to ensure that these models themselves are accurate and secure. The different ways in which such a model can be manipulated to provide inaccurate predictions is an important topic of study in cyber security. It is in a way the security of cyber security models.

In our project we consider a multinomial logistic regression model fit to a handwritten digit recognition dataset that classifies input images into true labels. We designed the model to predict digits with high accuracy to obtain a properly functioning multinomial logistic regression model on which we can test methods to decrease its accuracy efficiently. Our idea was to observe the accuracy scores of the model without any attacks and compare it to three different types of attacks, namely, non-targeted attack, natural fooling targeted attack and non-natural fooling targeted attack. In doing this, we can determine the method an adversary might use against the threat detection model of a cyber security system. It is the first step to design successful attacks on a machine

learning model to optimise it and make it more secure against similar attacks against adversaries.

1.2. Scope

This project is based on designing ways to attack a machine learning model which in turn highlights a few attacks that real world adversaries might use to break into a system. Recognising possible future attacks allows administrators to put preventive measures in place for the same. Then it is possible to say that the scope of this project extends to the protection and security of machine learning models used in any field.

The shear volume of cyber attacks has risen to the point where it is practically impossible for human beings to detect and analyse even a significant fraction of them. There are billions of malware attacks in a single year which is too much for humans to process so machine learning models are designed to process this data. Machine learning models that are able to perform functions that they haven't been explicitly programmed to do, process millions of files to identify potential threats and hazardous data. This way they can detect potential threats and neutralise them automatically.

Microsoft's cyber security platform Windows Defender ATP (Advanced Threat Protection) uses multiple levels of machine learning algorithms for breach detection, preventive protection and automated response.

Alphabet's cybersecurity company Chronicle uses machine learning to analyse the large volumes of security telemetry data generated by companies. The name of the cybersecurity product is Backstory which employs machine learning algorithms for analysing large amounts of data like known bad domains, internal network activities, suspected malware and hazardous files which are condensed using their machine learning algorithms to generate more insightful data, easy to read for security analysts to ensure the health of the network.

Sqqrl is yet another cybersecurity platform designed using machine learning models to search through networks for detection and elimination of bad code that can go undetected by the existing security measures in place. It uses machine learning to convert data points into something that depicts its behavioural trends in a visual representation for a given computer network.

The scope of this project covers all applications of machine learning as it provides an insight on the attacker's perspective while designing an attack. However, machine learning is not the only technology being used in cyber security as artificial intelligence and deep learning techniques are also being used which are beyond the scope of this project.

As a matter of fact, many attackers are also using machine learning to carry out their malicious endeavours. This means that the need for finding weaknesses in machine learning models extends to both sides of the attack, that is, if one recognises the types of possible attacks on their machine learning model then it becomes easier to put detection controls in place and implement preventive measures, or if we look at this information from the other perspective, we can say that it becomes easier to recognise the vulnerabilities in the attacker's machine learning model and make it ineffective by attacking those vulnerabilities to protect the computer network.

1.3. Novelty

The rapid growth in technology and its scope has led to an exponentially growing network of computers and databases. These components are more connected now than they have ever been in the history of existence. It is a boon to human kind as vast networks imply better connectivity which translates to faster communication, smoother data transfer and in general all tasks get executed much faster. Everyone wants to complete their tasks faster and achieve goals as soon as possible so every domain is shifting towards technology and using these networks in one way or another.

Having a large network also means better access to all kinds of data for everyone and that includes people and organisations with malicious intent. These people may want to find vulnerabilities in a network and exploit them for personal gain. The personal gain from attacking a network might be from getting access to private data through a data breach, or shutting down a system and rendering it useless and stagnant for a period of time, or even theft of proprietary information.

These cybersecurity risks have been extensively studied and researched by cybersecurity analysts to design security protocols for the protection and healthy functioning of a network. In recent years, the discussion on the mentioned topic has largely shifted to the applications of machine learning in this area of network security.

Extensive applications of machine learning algorithms in cybersecurity led to a number of different ways in which it can be used in cybersecurity.

Regression uses the knowledge about existing data to get an idea or in a way predict the implications of future data. For example, if we consider the trend of property prices in a given market and we use the existing database to fit a regression model then that model can be used to detect anomalies in property prices indicating fraudulent transactions. Types of regression that are used include; linear regression, polynomial regression, ridge regression, decision trees, SVR or support vector regression and random forest models.

Classification algorithms use categorical variables as the dependents and other affecting factors as independent variables. Classification is the most commonly used technique in cybersecurity incidents as most real world cyber incidents can be categorised and hence generate categorical data. For example, email data can be classified as important or spam and maybe other categories depending on the complexity of the classification algorithm used. Classification models used include; logistic regression, k-nearest neighbours, support vector machine, kernel SVM, Naive Bayes, decision tree classification and random forest classification.

At this point it is very clear that machine learning has been accepted by the majority of security analysts and tech companies as the go to solution for their cybersecurity needs. We have seen a lot of discussion on this topic too, but the security of these models becomes more critical as their use has become so widespread.

If a large number of security tools make use of machine learning models and algorithms in one way or another then it becomes convenient for an attacker to directly design attacks on these models which will affect the security tools and make the networks vulnerable and attacks will go undetected. In this project we discuss the approach a potential threat may use to make a machine learning model inaccurate. So we could just say that the focus of discussion here is the security of machine learning algorithms and we demonstrate it by designing and assessing various attacks on the logistic regression model. There are many ways to attack a machine learning model but the main method that we are concerned with in our project is by distorting the test data. The novel idea is observing the impact of attacking the model by considering the digits that they are more likely to be mistaken as, making them their natural fooling targets.

Then the novelty of this project is the recognition of vulnerabilities in machine learning models and its impact on cybersecurity platforms that use said algorithms.

1.4. Comparative Statement

1. Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS

The above paper aims to survey the various interactions between resilient Cyber Physical Systems using Machine Learning and resilient Machine Learning when applied to the said systems. The paper puts forth a number or promising trends in research and future directions for the research into resilient ML in CPS. On reading this paper, readers can obtain a comprehensive understanding of recent advances in Machine Learning based security and securing ML for the said system as well as develop counter measures and their research trends and topics. The paper delves into the potentials and challenges in applying AI and ML in the field of cybersecurity. It has been ascertained from the paper that these two topics will play a hugely impactful role in the securing of CPS form attackers. However there remain challenges that need to be addressed to ensure the success of the above systems. The research into adversarial attacks on ML is a hot topic for the future. There are however possibilities of intermediate attacks by the time defence mechanisms have been securely deployed against adversarial attacks and this might prove to be useful to the attackers for potential attacks.

2. A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity

In order to better understand the proper evaluation parameters for the robustness and resistance of machine learning models, specifically for classification problems in cybersecurity, against threats from adversarial attacks, the authors of this paper propose an attack method based on the brute force approach. The attack is designed using the black-box method to look at attacks from the adversary's perspective and recognise the shortcomings of the existing attack methods. They have used their attack to produce adversarial examples for various machine learning models commonly used in

cybersecurity systems. These include android malware detection using machine learning, intrusion detection and network intrusion detection using machine learning. Their results show that they have successfully designed a more efficient attack method than the currently accepted attack methods which may be used to test various machine learning based security systems.

3. Machine Learning Security: Threats, Countermeasures, and Evaluations

This paper is a survey on the robustness of machine learning models. The authors develop a systematic approach to understand and evaluate the security issues of machine learning models against adversarial attacks. They discuss the defence strategies associated with these attacks through training phase to the test phase. The reasons an adversarial attack on a model might take place are discussed and a machine learning model in the presence of an attack is presented for better analysis. The issues related to the security of machine learning models is categorised under five labels; backdoors in the training set, poisoning of training set, model theft, adversarial example attacks and recovery of sensitive training data. The attacks are reviewed in real-world conditions to understand the real-world impact.

4. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey

The authors of this paper have surveyed attacks on computer vision and deep learning models from adversaries in a systematic and comprehensive manner. The literature associated with design of adversarial attacks is discussed to analyse their functioning and suggest defence strategies against said attacks. Real world scenarios of these attacks are also reviewed from other literature about its realistic threat. It was observed that even though deep learning models are the most suitable approach for computer vision as they provide the most accurate outputs, these models are exposed to the threat of perturbed inputs leading to completely inaccurate outputs. This paper reviews the various adversarial attacks on these deep learning model applications in computer vision and artificial intelligence, and their respective defence mechanisms. Safety and

cybersecurity related applications are specially vulnerable to these adversarial attacks in real world situations so the authors demonstrate the effective attack vectors to these models in realistic scenarios.

5. A System-Driven Taxonomy of Attacks and Defenses in Adversarial Machine Learning

The authors' main goal in this paper was to develop an unambiguous methodology for a microscopic and systematic approach for the taxonomy to specify adversarial attacks in machine learning model based applications. This would allow others to replicate these experiments in order to escalate the rate at which machine learning models are being made more robust against adversarial attacks. The article provides taxonomies for; adversary's strategy, the ML architecture, the defence response, the dataset, and the adversary's knowledge, capability, and goal. They also establish relationships between these taxonomies and models by proposing an adversarial machine learning cycle. The comparison between various literature in this field is done and the gaps and missing components are revealed.

1.5. Dataset

The dataset used in this project is from the Modified National Institute of Standards and Technology (MNIST) database of handwritten digits. It is a modified and smaller version of a larger set available from the NIST database. It contains data of handwritten digits which have been preprocessed for computer vision and machine learning.

The digits are already size normalised and centred with respect to a fixed size. The original images in the NIST database were bilevel images, that is black and white with a normalised size of 20x20 pixel box with the aspect ratio preserved as the raw. The processed images contained grey levels resulting from the anti-aliasing technique of the normalisation algorithm. The centre of mass of the pixels for each image was computed to align it with the centre of the 28x28 pixel image to maintain consistent relative positioning of the hand written digit in each image.

The centre of the image may be aligned with the 28x28 box differently based on the classification algorithm used for the dataset. It generally needs to be centred using bounding box rather than centre of mass of pixels for particularly template based classification algorithms such as k-nearest neighbours (k-NN) and support vector machine (SVM). Since this project deals with the classification algorithm of logistic regression (LR) we may not need to process the dataset again to modify alignment from centre of mass of pixels to bounding box alignment.

The NIST database was divided into Special Database 3 which was collected from the employees at Census Bureau and Special Database 1 which was collected from high school students. Special Database 3 was originally used as the training set and Special Database 1 was used as the test set but it was quite apparent that SD-3 was much easier to recognise as it the data was cleaner than SD-1. So, it was suitable to make a new database by mixing the data from both databases SD-1 and SD-3 making the conclusions derived from the experiments and models to be independent from the choice of training set and test set. Hence, a modified version of a combination of both sample sets was used to form the MNIST dataset.

MNIST digit recogniser dataset contains a total of 70,000 samples where half are taken from Special Database 3 of NIST database and the other half from Special Database 1.

Each sample consists of the true label of the image which represents the actual digit that is stored in the image of the hand written sample and the remaining data is for each pixel in the 28x28 image. This results in 785 columns for each image where the first stores the true label and the remaining 784 hold values for each pixel for the given image.

1.6. Test Bed

The project is programmed in Python which is an interpreter based high level language, it is a general purpose programming language and has extensive open source libraries for programming in various fields. We are using Python 3.9 for our project. It is easily readable due to its indentations and the code is reusable because of its extensible nature from defining functions and object oriented approach.

We are using Jupyter Notebook as the platform for coding, debugging, testing and visualising various parts of the project. It is an open-source web based application containing chunks of live code, algorithms, visualisations and text for providing insightful description.

The code was executed on Windows 10 operating system as well as macOS 12.0.1 with minor modifications related to the libraries imported in the code. Since Jupyter notebook is a web-application, it is compatible with most operating systems like Windows, Ubuntu and macOS.

We use pandas 1.3.0 which is a software library for Python and it is used for manipulating data during data analysis. Specifically, it provides data structures and operations for manipulation and analysis of data frames.

NumPy 1.21.1 is another Python library which adds support for large and multidimensional matrices and arrays and enables us to perform a large number of high level mathematical operations on these arrays.

Seaborn 3.4.2, which is a Python library used for data visualisation is based on matplotlib. It is used for visualising data and plotting graphs. It is built on matplotlib and is highly compatible with the data structures provided by pandas.

For further visualisations Matplotlib 3.4.2 is use which is a plotting library for Python. It is compatible with NumPy functions and structures. It uses and object oriented approach for embedding plots into applications.

Additionally, Plotly is used as a Python library for online data analytics and visualisation. It is however compatible with other programming languages like R, Matlab, Perl etc. It provides statistical tools and online graphing features.

Scikit-learn 1.0.1, also known as sklearn is an open source machine learning library made for Python. It has a large number of algorithms for classification, regression and clustering which includes logistic regression, support vector machine, random forest, k-means and many more. This library has been used for making our Logistic Regression model.

Scikit-image 0.17.2, also known as skimage is an open source Python library for image processing. It includes algorithms for colour space manipulation, segmentation, geometric transformations, colour space manipulation, filtering, morphology and more.

1.7. Expected Result

In the first step the dataset needs to be loaded into memory properly and inspected. We expect to see the head of the dataset as the first column holding true values of the digits in the image for that row. We then expect to see the next 784 columns with intensity values for each pixel making the image for each handwritten digit.

Then the images need to be observed to verify the authenticity of the dataset loaded in the previous set. We expect to see handwritten digit images and the corresponding true digit values alongside them. This would let us know that valid true labels are present in the dataset for the images. We also expect to see handwritten digits which may not be distinctly recognisable by the human eye due to variations in writing styles of different sources for these samples.

Then the model needs to be designed and fit to the dataset. A logistic regression model with an accuracy greater than 90% should be made so we may expect the model to give highly accurate predictions for test inputs. So it would be ideal to fit the logistic regression model to the training set and obtain its accuracy scores and optimise the model to get a high accuracy.

Then the attack needs to be designed so it would seem right to have a non-targeted attack that adds perturbations to the test input images of handwritten digits. The images would be perturbed using various values of epsilon. A graph should be obtained to see the relation between the epsilon used for perturbations and the resultant accuracy score of the model. The accuracy should decrease as epsilon is increased because the test image is being modified to fool the model, hence leading to incorrect classifications. Looking at this graph we would be able to determine the accuracy of the model based on the epsilon value chosen to perturb the test images for fooling logistic regression.

Based on this information, we can choose the right value for epsilon and create a suitable decrease in the model's accuracy with the non-targeted attack.

Now we would be able to attack the model with selected epsilon perturbations and obtain the fooling results for that epsilon. The resultant model would have a lower accuracy of prediction and we could then inspect the true labels against the predicted labels for the ideal model and the fooled predictions for the attacked model. This would give us the suitable fooling targets for each digit as the count of fooled predictions for each label would reveal the digit it is mistaken for by the model using the selected perturbation.

The weighted parameters for true digits would be closer to other specific digits, that is, each digit would be fooled as some particular digit for the maximum number of cases. We could then recognise these digits as natural fooling targets. Then these natural fooling targets would enable us to design a targeted attack on the model enabling us to attack input test data in such a way that its natural fooling target is the fooled prediction given by the model.

This approach, in turn would give us the ability to distinguish between natural fooling targets for each digit and its non-natural fooling targets and compare their accuracies.

2. ARCHITECTURE

This project has four main stages:

- Making a fairly accurate logistic regression model
- Attacking the model by adding perturbations to the input to recognise natural fooling targets
- Attacking the model for the observed natural fooling targets
- Attacking the model for the observed non-natural fooling targets.

We will see the design of the architecture in more detail in the sections below. The architecture can be viewed in two ways; we'll look at the overview of the project and then we'll look at the detailed working.

2.1. High Level Design

Figure 1: High Level Architecture Diagram

The Logistic Regression model is fit to the train data from MNIST digit recogniser dataset. Then we use the test data to obtain the accuracy of the model made. The parameters in the model are optimised to have a fairly good accuracy of classification so that the initial model that we are working with can be considered as a successful machine learning model giving us a proper base around which an attack can be designed. So the training data will be loaded into memory for us to perform various operations on it. The model is fit to the data and we use test data and obtain accuracy scores from it. One we are satisfied we the model we have made, we move on to designing the attack methods on this model. First we'll try to attack the model directly by changing the input images and observe the outputs and their trends through histograms and heatmaps.

The test data is one hot encoded so that we can approach it as a classification problem. We then calculate the perturbations to be added for each test image based on the epsilon for image distortion. We add perturbations with increasing epsilon values to test images until the accuracy of the model becomes zero and we can say with certainty that for that given epsilon, the model will predict the digit incorrectly. A graph needs to be plotted to observe the relationship between the perturbations added to the test images and the resultant accuracy of the Logistic Regression model. Based on the relationship between epsilon and accuracy, we can choose a specific value of epsilon to be used for the non-targeted attack. The perturbations are added to the test images and the predicted values from the LR model are observed.

We use the predictions for the non-targeted attack to observe the true labels, correct predictions and fooled predictions. Various visualisations may be made to observe any patterns that may be present in the predicted values. The histogram of predictions for each true label is plotted. This data will clearly show the number of times each digit has been predicted by the model and if there are digits that are predicted for a significantly more number of times or if a digit is predicted significantly less as compared to the other digits then that observation can be noted.

A heat-map of this data also needs to be plotted so that a relationship between true labels and fooled predictions over different classes of digits can be observed. This would reveal the tendency of each digit to appear similar as a particular digit to the model after a non-targeted attack by adding perturbations. This information can be used to determine the natural fooling target for each digit. Similarly, each digit in the dataset would have the tendency to appear the least number of times as the fooled prediction for a given digit and hence it can be considered as the non-natural fooling target for the given digit.

So we can determine the natural and non-natural fooling targets for each digit and proceed to designing targeted attacks on our Logistic Regression model by affecting the weights and biases of the test data appropriately to achieve desired fooled predictions while decreasing the accuracy of the LR model.

The test images are first changed to get the natural fooling targets as the prediction from our model and then we obtain the resultant accuracy of the model. In a similar manner, the test data is changed to get the non-natural fooling target of that digit as the predicted output from our classifier. We will obtain the accuracy of the model resulting from non-natural fooling target approach of attack as well. We can finally compare the accuracies obtained in all the cases and tabulate and plot them to note our observations.

2.2. Low Level Design

The low level architecture of the project is shown below.

Figure 2: Low Level Architecture Diagram

To understand the low level architecture of this project in further detail, the low level design of each module is explained below.

a. Loading Packages and Exploring the Dataset

Figure 3:Loading Packages and Exploring the Dataset

We start by loading all the python libraries and packages required for this project. The libraries imported are;

- Scikit-learn or sklearn for the Logistic Regression model,
- Pandas for data structures,
- Numpy for mathematical operations,
- Seaborn for visualisations and graphs,
- Plotly for additional plots,
- Matplotlib for making graphs and visualisations,
- And Scikit-image or skimage for image operations.

Then we load the dataset into memory for storing it into arrays and data frames to apply mathematical operations on it. The data is also explored through some basic methods; first by observing the initial few rows of the dataset, then by using the pixel information to produce images for few of the rows in the beginning. The count of each true label in the dataset is also plotted as histogram to ensure the balance of the samples.

b. Preparing LR Model

Figure 4: Preparing the Logistic Regression Model

The Logistic Regression model is imported from sklearn and the model is fit to the training data. The test data is used to generate predictions from the model and obtain the accuracy of the LR model thus created.

c. Non-Targeted Attack

Figure 5: Non-targeted attack

For the non-targeted attack, the model fit to the training data is used by adding perturbations to the input test images. Various values of epsilon are used to calculate perturbation to be added to the test image and the images are observed. A relationship between the increasing value of epsilon and the accuracy of the LR model is plotted and a specific value of epsilon is selected. Then the model is attacked using these perturbed test images and the resultant accuracy of the model is obtained. These predictions are used to determine the natural and non-natural fooling targets for each digit.

d. Natural and Non-Natural Targeted Attack

Figure 6: Natural and Non Natural Targeted Attacks

The natural fooling and non-natural fooling targets for each digit are stored in dictionaries and the targets are one-hot encoded. Then the model is attacked using these values to obtain the accuracies associated with the natural fooling target attack on the model and the non-natural fooling target attack on the LR model. These accuracies are compared and plotted on a graph to observe the trend between the increasing epsilon value for each type of attack.

3. Implementation

The implementation of our project was done using python and various libraries made for python and the whole project was run as a Jupyter Notebook file.

3.1. Algorithms and Procedure of Implementation

The Logistic Regression model from Scikit-learn is used with a few other functions from the same library as well, such as to obtain the accuracy metrics of the model and splitting the dataset into train and test data.

The Logistic Regression Model is trained using the MNIST digit recognizer datset and the trained model is then tested to obtain an initial reading of the accuracy of the model. The parameters in the model are optimised to have a fairly good accuracy of classification so that the initial model that we are working with can be considered as a successful machine learning model giving us a proper base around which an attack can be designed. The training data is trained by the Multinomial Logistic Regression model and it is tested with the test data to obtain the initial accuracy score to see that we are fooling an accurate ML model.

After the initial LR model is obtained we can proceed with the creation of our attack classes for various forms of attacks on the model namely, non-targeted attacks, targeted attacks with natural fooling targets and targeted attacks with non natural fooling targets.

The test data is one hot encoded to convert it into1s and 0s so that we can approach the given problem as a classification problem. We then calculate the perturbations to be added for each test image based on the epsilon for image distortion. We add perturbations with increasing epsilon values to test images until the accuracy of the model becomes zero and we can say with certainty that for that given epsilon, the model will predict the digit incorrectly. A graph needs to be plotted to observe the relationship between the perturbations added to the test images and the resultant accuracy of the

Multinomial Logistic Regression model.. The perturbations are added to the test images and the predicted values from the LR model are observed.

On completing the non-targeted attacks, we can observe the true labels, correct predictions and fooled predictions in the form of graphs and heatmaps. Various visualisations may be made to observe any patterns that may be present in the predicted values. The histogram of predictions for each true label is plotted. This data will clearly show the number of times each digit has been predicted by the model and if there are digits that are predicted for a significantly greater number of times or if a digit is predicted significantly less as compared to the other digits then that observation can be noted.

A most effective visualization would be a heat map which would give a complete overview of the true label and the fooling targets used for those true labels. From the heatmap we can obtain the values of the digits which are most used as fooling targets for a particular digit and these become the natural fooling targets and the opposite which are the least used fooling targets for a particular digit which becomes the non-natural fooling targets. So we can determine the natural and non-natural fooling targets for each digit and proceed to designing targeted attacks on our Logistic Regression model by affecting the weights and biases of the test data appropriately to achieve desired fooled predictions while decreasing the accuracy of the LR model.

The test images are first changed to get the natural fooling targets as the prediction from our model and then we obtain the resultant accuracy of the model. In a similar manner, the test data is changed to get the non-natural fooling target of that digit as the predicted output from our classifier. We will obtain the accuracy of the model resulting from non-natural fooling target approach of attack as well. We can finally compare the accuracies obtained in all the cases and tabulate and plot them to note our observations.

Other functions for plotting graphs and charts are used from appropriate python libraries. Array operations and data frame processes are imported from corresponding libraries for python as well.

The image operations such as importing image data from dataframes and displaying image data in visual format is being done using scikit-image library for python.

We can consider the sequence below as the basic algorithm followed for the project:

- Import Python Libraries (Sklearn, Numpy, Pandas, Seaborn, Matplotlib, Skimage, Plotly)
- Load Dataset into memory
- Store data in data frame and display first few rows
- Drop true label column
- Split remaining data into train and test 60-40 respectively
- Display first few rows as images with corresponding true labels
- Define a class with all the necessary functions
- Fit model to training data
- Use prepare function from the class to obtain initial model accuracy
- Explore weight vectors generated by the model as compared to number of classes
- Create one hot encoded targets for the test data
- Call attack to maximum epsilon function from the class with upper limit 30 for epsilon
- Obtain the accuracy scores of the model for all epsilon values
- Plot relationship between Accuracy and epsilon
- Select epsilon with intermediate impact on accuracy score of model
- Obtain prediction data for selected epsilon
- Display All predictions
- Display successfully fooled predictions
- Display sample test digit image with increasing perturbations and predicted value with true label

- Plot count of successfully fooled predictions
- Plot heatmap between true labels and fooled predictions
- Make dictionary of natural fooling targets
- Prepare dictionary of non-natural fooling targets
- Call attack to maximum epsilon using natural fooling targets
- Obtain accuracy of the model
- Call attack to maximum epsilon using non-natural fooling targets
- Obtain accuracy of the model
- Compare all the accuracies obtained

3.2. Mathematical Models

1. Logistic Regression

Logistic Regression is a supervised machine learning algorithm that is used for classification problems. It is used to predict the probability of a particular variable. There exists only two possible classes for the dependent variable as its nature is dichotomous. It is binary and its data contains either 1 which implies success or 0 implying failure. A logistic regression model predicts the probability that Y=1 as a function of X. It is a very commonly used machine learning algorithm and has its applications in many fields. The logistic regression model utilized by us in our project is a multinomial logistic regression model. A multinomial logistic regression model is one which is widely used for classification purposes. In this kind of classification, the dependent variable can have multiple unordered types or have quantifiers without significance.

Logistic Regression in its simplest terms is a predictive analysis algorithm that is based on probability. Logistic Regression is a more complex version of Linear regression in that it uses a more complex cost function namely the sigmoid function. The sigmoid function is used to map the predicted values to the probabilities that have been given. It

basically maps values between 0 and 1. The sigmoid function is represented by f(x)= 1/ (1+e^(-x)).
Logistic Regression models also have a more complex hypothesis representation as compared to the linear regression counterpart.

Multinomial Logistic Regression is a powerful analysis tool as it is free from the assumptions of things such as linearity or normality of the data. The solver used by the multinomial logistic regression function is th4 lbfgs solver which stands for Limited Memory Broyden Fletcher Goldfarb Shanno. This solver approximates the second derivative matrix updates with gradient evaluations. It is an extremely memory efficient solver method and is the default solver method for multinomial logistic regression. In our logistic regression model, the inputs and classes are considered to be independent of each other and they are assumed to be identically distributed. Multinomial logistic regression assumes the inputs and features to be independent and equally distributed in probability space. The target is to maximize the likelihood that the digits will match. A softmax activation function for multinomial probability prediction is used to obtain the output for the model.

2. Non-targeted attack

The objective of the non targeted attack is to obtain the maximum possible deviation from the original digit which is referred to as discrepancy. The need is to increase the chances that the model incorrectly predicts the digits in the dataset. This is basically the reverse of the method used normally where we would want to maximize the probability of the correct output. Hence we need to reverse the logistic regression model. To make sure that the digits are incorrectly recognized, we can add imperfections or convolve the images with perturbations to fool the machine learning model. We use a perturbation function to add perturbations to the image based on the specified value of epsilon that has been passed. The more the value of epsilon the more perturbed the resultant image will be resulting in more likelihood that it is incorrectly predicted by the LR Model. Furthermore, gradient ascent technique is also used in conjunction with the

perturbations to obtain best fooling results. After adding these steps, we can run the model and test the accuracy of the model with the values of epsilon to obtain the visualization of the decrease in accuracy from the non-targeted attacks

3. Targeted Attacks with Natural Fooling Targets

Once the non-targeted attacks have been completed, we can obtain the plots and charts and the heatmap of the true label compared to the fooling target used. From this we obtain for each digit, the most commonly used fooling target for that particular digit. Now these values can be considered to be the most effective fooling targets for those respective digits and we can now attack those particular digits to be fooled by their most effective fooling targets.

4. Targeted Attacks with Non- Natural Fooling Targets

Like the natural fooling targets procedure, once the non-targeted attacks have been completed, we can obtain the plots and charts and the heatmap of the true label compared to the fooling target used. From this we obtain for each digit, the least used fooling target for that digit which is the exact opposite of what is being done in the above method. Now these values can be the least effective fooling targets for those respective digits, and we can now attack those particular digits to be fooled by their least effective fooling targets and observe how it effects the accuracy of the model.

4. Results and Discussion

4.1. Implementation with Coding

1. Loading the Packages

Figure 7: Code Snippet of Loading of Packages

2. Loading Data

Figure 8: Code Snippet of Loading dataset

Figure 9: Code snippet for train test split and displaying sample elements

Loading the train data from the digit recognizer dataset and view the first 10 elements. Then splitting the train and test data set followed by the display of 15 digits in the dataset.

3. Attack Class

Figure 10: Code snippet for Attack Class

Figure 11: Code snippet for Attack Class Part 2

The above classes consist of the attack classes as well as the preparation, train and test function to obtain the initial accuracy of the multinomial logistic regression model. Another function provided is to perform one hot encoding to covert to classification problem. Another function is the attack function which uses the perturbed images to fool the target. More functions which perform and store the perturbed images are given and a final function which attacks the model from lowest value of epsilon (least perturbation) to max epsilon (most perturbation).

4. Weights and Gradient Methods

Figure 12: Weights and Gradients Method

These are the gradient and weight calculation methods.

5. Training the model

Figure 13: Training Model

This trains the Logistic Regression model and we can display the accuracy.

6. Non Targeted Attack

```
attack.create_one_hot_targets(y_test)
attack.attack_to_max_epsilon(non_targeted_gradient, 30)
non_targeted_scores = attack.scores

sns.set()
plt.figure(figsize=(10,5))
plt.plot(attack.epsilons, attack.scores, 'g*')
plt.ylabel('accuracy score')
plt.xlabel('epsilon')
plt.title('Accuracy score breakdown - non-targeted attack');
```

Figure 14: Non-targeted Attack

7. Heatmap to view the natural and non-natural fooling targets

```
attacktargets = example_results.loc[example_results.y_true != example_results.y_fooled].groupby(
    'y_true').y_fooled.value_counts()
counts = example_results.loc[example_results.y_true != example_results.y_fooled].groupby(
    'y_true').y_fooled.count()
attacktargets = attacktargets/counts * 100
attacktargets = attacktargets.unstack()
attacktargets = attacktargets.fillna(0.0)
attacktargets = attacktargets.apply(np.round).astype(np.int)

f, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(attacktargets, annot=True, ax=ax, cbar=False, square=True, cmap="Reds", fmt='g');
ax.set_title("How often was y_true predicted as some y_fooled digit in percent?");
```

Figure 15: Heatmap for natural and non-natural fooling targets

8. Obtaining the natural and non natural fooling targets

```
natural_targets_dict = {}
non_natural_targets_dict = {}
for ix, series in attacktargets.iterrows():
    natural_targets_dict[ix] = series.argmax()
    non_natural_targets_dict[ix] = series.drop(ix).argmin()
```

Figure 16: Getting natural and non-natural fooling targets

9. Natural and Non-natural Targeted Attacks

Figure 17: Natural and Non-natural Targeted attacks

10. Comparison of accuracy of Natural and Non-natural Accuracy with increasing epsilon values

Figure 18: Comparison of accuracy between the two

4.2. Result

1. Loading data

Figure 18: Loading the data

2. Sample digits in the dataset

Figure 19: Sample digits

3. Initial accuracy of the multinomial LR model

Figure 20: Initial accuracy of multinomial LR model

4. Non-Targeted Attack Accuracy

Figure 21: Non-targeted attack acuracy

5. Sample of results from the non- targeted attack

	y_true	y_fooled	y_predicted	id
0	3	8	3	0
1	6	6	6	1
2	9	9	9	2
3	5	8	5	3
4	6	6	6	4

	y_true	y_fooled	y_predicted	id
0	3	8	3	0
3	5	8	5	3
5	5	9	0	5
6	6	2	6	8
14	8	5	8	14

Figure 22: Sample of results from non-targeted attack

Table on list shows all the true labels, fooled labels and predicted labels and table on the right shows the labels only where the true label is not equal to the fooled label.

6. Fooling targets **used per epsilon for sample image**

Figure 23: Sample output per epsilon value

7. Sample result for perturbation for epsilon 16

Figure 24: Sample results of perturbation

8. Histogram with counts of y_fooled, y_predicted and y_true

Figure 25: count of y_fooled

Figure 26: Count of y_predicted

Figure 27: Count of y_true

9. Heatmap displaying the percentage of fooling target used for the given true digit

Figure 28: Heatmap of percentage of fooling target used for the given true digit

10. Natural and Non-natural Targeted Attack Accuracy Scores with increasing epsilon

Figure 29: Natural and Non-natural Targeted Attack Accuracy Scores with increasing epsilon

11. Graphical Representation of comparison of accuracy of natural and non-natural targeted attacks with increasing epsilon

Figure 30: Graphical Representation of comparison of accuracy of natural and non-natural targeted attacks with increasing epsilon

4.3. Conclusion

We have successfully created three attacks on a multinomial regression model using digit recognizer database. These three attacks are namely non-targeted attacks, targeted attacks with natural fooling targets and targeted attacks with non-natural fooling targets. We have obtained, analysed and plotted the impact of these attacks on the given ML model and obtained the results in terms of impact on the accuracy of the model. We can clearly see that on increasing the number of perturbations in the image which is on increasing the epsilon value, we can fool the ML model much more frequently. From the non-targeted attacks, we got an overview of the most and least frequently used fooling targets for a particular digit. These fooling targets become the natural and non-natural fooling targets for the targeted attacks. The accuracies for these attacks were also compared and it can be established that the accuracy of the model decreases with a higher rate as we increase the epsilon value for natural fooling targets as compared to when the model is attacked for non-natural fooling. These findings show the realistic

threat to classification models which may be used in various cybersecurity applications and a real-world scenario where an efficient attack to fool the model for various types of targets was created.

5. References

[1] F. O. Olowononi, D. B. Rawat, and C. Liu, "Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 1, pp. 524–552, 2021, doi: 10.1109/COMST.2020.3036778.

[2] S. Zhang, X. Xie, and Y. Xu, "A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity," *IEEE Access*, vol. 8, pp. 128250–128263, 2020, doi: 10.1109/ACCESS.2020.3008433.

[3] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, "Machine Learning Security: Threats, Countermeasures, and Evaluations," *IEEE Access*, vol. 8, pp. 74720–74742, 2020, doi: 10.1109/ACCESS.2020.2987435.

[4] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018, doi: 10.1109/ACCESS.2018.2807385.

[5] K. Sadeghi, A. Banerjee, and S. K. S. Gupta, "A System-driven taxonomy of attacks and defenses in adversarial machine learning," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 4, no. 4, pp. 450–467, 2020, doi: 10.1109/TETCI.2020.2968933.

Matches

Internet sources	193

1	https://www.hindawi.com/journals/wcmc/2021/4907754	25 Sources	0.35%
2	https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html?source=post_page----------	6 Sources	0.42%
3	https://ijcit.com/index.php/ijcit/article/view/79	12 Sources	0.26%
4	https://www.mdpi.com/1099-4300/23/10/1304/htm	3 Sources	0.32%
5	http://www.cister.isep.pt/docs/cloud_versus_edge_deployment_strategies_of_real_time_face_recognition_inference/170	5 Sources	0.16%
6	https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11746/2583970/Recent-advances-in-adversarial-mac	7 Sources	0.37%
7	https://ojs.unud.ac.id/index.php/lontar/article/view/68199	4 Sources	0.16%
8	https://www.imageconindia.com/courses/micro-degree-on-machine-learning	16 Sources	0.19%
9	https://tches.iacr.org/index.php/TCHES/article/download/8971/8549/6303	14 Sources	0.17%
10	https://www.frontiersin.org/articles/10.3389/fpsyg.2021.596038/full		0.15%
11	https://link.springer.com/content/pdf/10.1007/978-1-4842-4470-8.pdf	6 Sources	0.14%
12	http://etd.aau.edu.et/bitstream/handle/123456789/15804/65.%20TAGEL%20ALEMU%20TAFESE%20THESIS.pdf?isAllowed	4 Sources	0.14%
13	https://ebin.pub/machine-learning-guide-for-oil-and-gas-using-python-a-step-by-step-breakdown-with-data-algorithms-	2 Sources	0.14%
16	https://doaj.org/article/6e9f91d032304ef79c9d1bb73bf5027d	4 Sources	0.12%
17	https://iamr0h1t.medium.com/logistic-regression-e0e63f7097d7		0.12%
18	https://tomblackson.com/PHI_319_420/lecture16.html		0.12%
19	https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest		0.11%
20	https://worldwidescience.org/topicpages/c/cancer+screening+systematic.html	17 Sources	0.11%
21	http://repository.sustech.edu/bitstream/handle/123456789/24239/Research.pdf?isAllowed=y&sequence=2		0.11%
22	https://www.shatam.com/Predict-The-Mode-Of-Delivery.pdf		0.11%

27	http://hdl.handle.net/10500/27685		0.1%
28	https://apps.dtic.mil/sti/pdfs/ADA570534.pdf		0.1%
29	https://www.ncbi.nlm.nih.gov/pubmed/32971987		0.1%
30	https://www.cbsnews.com/pictures/notable-deaths-in-2019/102	3 Sources	0.1%
31	https://qualitywriters.us/hrm-s-role-in-the-performance-management-process		0.1%
32	https://www.edureka.co/blog/decision-tree-algorithm		0.1%
33	https://www.ons.gov.uk/economy/inflationandpriceindices/articles/researchindicesusingwebscrapedpricedata/august2017upd...		0.1%
34	https://dokumen.pub/data-science-and-security-proceedings-of-idscs-2021-lecture-notes-in-networks-and-systems-290-1st-ed-2...		0.1%
35	http://ateec.org/seetreview/wp-content/uploads/carbon-footprint-lab.doc		0.1%
36	http://www.ecoi.net/file_upload/227_1167833280_g0612253.pdf	2 Sources	0.1%
37	https://scholarcommons.usf.edu/etd/4895	5 Sources	0.1%
38	https://www.science.gov/topicpages/a/article+reviews+important.html	29 Sources	0.1%
39	http://www.wireless-earth.net/paper/habilFinal_opt.pdf	15 Sources	0.1%

Library sources 41

14	**Student submission**	File ID: **57390250**	Institution: **VIT University**		0.12%
15	**Student submission**	File ID: **57042932**	Institution: **VIT University**	2 Sources	0.12%
23	**Student submission**	File ID: **68745429**	Institution: **VIT University**	30 Sources	0.11%
24	**Student submission**	File ID: **57401544**	Institution: **VIT University**	3 Sources	0.1%
25	**Student submission**	File ID: **18352699**	Institution: **VIT University**		0.1%
26	**Student submission**	File ID: **41871302**	Institution: **VIT University**	3 Sources	0.1%
40	**Student submission**	File ID: **10668191**	Institution: **VIT University**		0.1%

Quotes

References **1**

1 [1] F. O. Olowononi, D. B. Rawat, and C. Liu, “Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS,” IEEE Commun. Surv. Tutorials, vol. 23, no. 1, pp. 524–552, 2021, doi: 10.1109/COMST.2020.3036778. [2] S. Zhang, X. Xie, and Y. Xu, “A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity,” IEEE Access, vol. 8, pp. 128250–128263, 2020, doi: 10.1109/ACCESS.2020.3008433. [3] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, “Machine Learning Security: Threats, Countermeasures, and Evaluations,” IEEE Access, vol. 8, pp. 74720–74742, 2020, doi: 10.1109/ACCESS.2020.2987435. [4] N. Akhtar and A. Mian, “Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey,” IEEE Access, vol. 6, pp. 14410–14430, 2018, doi: 10.1109/ACCESS.2018.2807385. [5] K. Sadeghi, A. Banerjee, and S. K. S. Gupta, “A System-driven taxonomy of attacks and defenses in adversarial machine learning,” IEEE Trans. Emerg. Top. Comput. Intell., vol. 4, no. 4, pp. 450–467, 2020, doi: 10.1109/TETCI.2020.2968933.