The Impact of Adversarial Machine Learning Attacks on the Performance of Customer Churn Prediction Models

Ezra Abah, 40482302

School of Computing

Submitted in partial fulfilment of the requirements of
Edinburgh Napier University
for the award of
MSc. Computing with Professional Placement

· 18 August 2021 ·

MSc dissertation check list

Student Name: Abah Ezra	Matric: 40482302

Please insert this form, loose-leaf, into each copy of your dissertation submitted for marking.

Milestones	**Date of completion**	**Target deadline**
Proposal	Week 3	Week 3
Initial report	Week 7	Week 7
Full draft of the dissertation	Deadline	2 weeks before final deadline

Learning outcome	**The markers will assess**	**Chapter[1]**	**Hours spent**
Learning outcome 1 Conduct a literature search using an appropriate range of information sources and produce a critical review of the findings.	* Range of materials; list of references * The literature review/exposition/background information chapter	2	200 hours
Learning outcome 2 Demonstrate professional competence by sound project management and (a) by applying appropriate theoretical and practical computing concepts and techniques to a non-trivial problem, or (b) by undertaking an approved project of equivalent standard.	* Evidence of project management (Gantt chart, diary, etc.) * Depending on the topic: chapters on design, implementation, methods, experiments, results, etc.	Chapter: 3,4 Appendix: 8.2, 8.3	200 hours
Learning outcome 3 Show a capacity for self-appraisal by analysing the strengths and weakness of the project outcomes with reference to the initial objectives, and to the work of others.	* Chapter on evaluation (assessing your outcomes against the project aims and objectives) * Discussion of your project's output compared to the work of others.	Chapter: 5,6	100 hours
Learning outcome 4 Provide evidence of the meeting learning outcomes 1-3 in the form of a dissertation which complies with the requirements of the School of Computing both in style and content.	* Is the dissertation well-written (academic writing style, grammatical), spell-checked, free of typos, neatly formatted. * Does the dissertation contain all relevant chapters, appendices, title and contents pages, etc. * Style and content of the dissertation.		80 hours
Learning outcome 5 Defend the work orally at a viva voce examination.	* Performance * Confirm authorship		1 hour

Have you previously uploaded your dissertation to Turnitin? Yes

Has your supervisor seen a full draft of the dissertation before submission? Yes

Has your supervisor said that you are ready to submit the dissertation? Yes

[1] Please note the page numbers where evidence of meeting the learning outcome can be found in your dissertation.

Declarations

Authorship Declaration

I, *Abah Ezra*, confirm that this dissertation and the work presented in it and my own achievement.

1. Where I have consulted the published work of others this is always clearly attributed;
2. Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;
3. I have acknowledged all main sources of help;
4. If my research follows on from previous work or is part of any larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;
5. I have read and understood the penalties associated with academic misconduct.
6. I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the school's ethical guidelines.

Sign or type name: Abah Ezra
Date: 18th August 2021

Matriculation No: 40482302

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name under one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

Abah Ezra 18th August 2021

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Acknowledgements

I would like to appreciate Andrew Partridge for his patience, guidance and advice through out the duration of this dissertation. I also appreciate my lecturers for imparting me with the capacity required to carry out this project successfully.
I give special thanks to my parents for their financial and moral support throughout my studies. Not to forget, my understanding family and friends. Finally, I would like to thank authors of all works from which I have drawn knowledge and inspiration to create this piece.

Abstract

In recent times, industries around the globe have continued to experience fiercer competition. This is because markets have become increasingly saturated and competitors continue to improve on the services they provide and implement more aggressive marketing strategies. For this reason, understanding and preventing customer churn is a critical and challenging issue for businesses. In the quest to find a reliable strategy to solve this challenge, researchers have explored ways in which highly predictive classification models can be trained using customer data and state-of-the art machine learning algorithms. The classifiers help them ascertain customers who are likely to churn after which targeted retention campaigns can be launched to keep them.

Although machine learning seems to present a solution, growing concerns about the vulnerability of machine learning model continue to be on the rise. Incentivised hackers are able to attack classification models and cause them to misclassify by sending *adversaries*. It is important to know how robust models are in the presence of adversaries before they are deployed in potentially adversarial environments. This study was aimed at investigating the performance of customer churn prediction models in the presence of these adversaries.

This investigation was carried out by training three models including a logistic regression model, a support vector machine classification model and an artificial neural network model. After training the models, their performance was evaluated using clean data as well as using adversaries generated using fast gradient sign method (FGSM) at different perturbation rates, deep fool attack at different overshoot values and jacobian salient map atttack (JSMA) at different perturbation rates.

An analysis of results obtained by evaluating the performance of the models showed that in the absence of adversaries, logistic regression outperforms other models followed by artificial neural network and finally support vector classifier. However, in the presence of adversarial attacks, support vector classifier performed the best thus proving its robustness to adversaries.

This was followed by artificial neural network and logistic regression models consecutively, both of which performed poorly and showed vulnerability to adversarial attacks. On that basis, it was recommended that support vector machine algorithm is used when deploying customer churn prediction classifiers in potentially adversarial environments.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In an era when markets are becoming increasingly saturated and the competition between businesses continue to intensify, customer churn presents a real problem (Amin et al., 2019; Huang, Kechadi, & Buckley, 2012; Vafeiadis, Diamantaras, Sarigiannidis, & Chatzisavvas, 2015; Vafeiadis et al., 2015). It has become apparent to business owners that for their companies to compete favourably, there is a need to implement carefully thought-out customer retention strategies that seek to initiate, maintain, and strengthen long-lasting relationships with customers (Torkzadeh, Chang, & Hansen, 2006).This is especially applicable in highly competitive, subscription-based service industries such as the telecommunication industry (Huang et al., 2012).

1.1 Background

Customer retention is very important and remains the topmost priority of companies and while there are indeed many reasons for this, the following reasons are worth noting:

1. Cost of getting new customers

The cost of getting new customers significantly exceeds what is needed to retain an existing one (Farquad, Ravi, & Raju, 2014; Lu, Lin, Lu, & Zhang, 2014; Vafeiadis et al., 2015). In 2000, a report published by management consultancy firm, Bain and Company estimated that the cost of acquiring a new customer is 7 times that of retaining an existing customer (Baveja, n.d.). The monetary impact of customer churn is even more pronounced when the following statistic is considered: according to the 2020 CallMiner Churn Index report, US companies lost $136.8 billion per year due to consumers making switches that could have been prevented (CallMiner, 2020). This

cost is a sum of capital expended on marketing campaigns, salaries paid to sales and product development staff amongst other costs (Amin et al., 2017).

2. Social network influence

Long term customers tend to refer more people to the company; conversely, churned customers also tend to influence other customers within their social network to leave. This observation was formally made in analyses carried out by (Nitzan & Libai, 2011) which found that exposure of an existing customer to a defecting social network "neighbour" increases their risk of also defecting by up to 80%. Furthermore, this value was found to increase significantly with increasing social network size and/or decrease with increasing customer loyalty.

3. Long term customers are more beneficial

Companies benefit more from long term customers in terms of increased sales and reduced costs. Long term customers, who have developed an increased trust in a company, tend to spend more by acquiring and utilising more goods and services from that company (Burez & Van den Poel, 2009; De Caigny, Coussement, & De Bock, 2018; Thoumy & Abdallah, 2017). Conversely, newer customers with less developed trust tend to spend less as they are generally sceptical about the value of goods or services offered by the company.

It is also important to recognise that long term customers tend to have their behaviours, needs and preferences better accommodated due to the company's ability to analyse and understand these traits over longer time frames than newer customers. Furthermore, a significant amount of time, energy, and resources needs to be expended by the company when initially gathering the data of newer customers to be able to tailor future experiences appropriately (De Caigny et al., 2018; Huang et al., 2012).

4. Competition has less effect on loyal customers

Lastly, as customers stay committed to a particularly company for longer, marketing strategies of competing businesses tend to have less effect on them (Burez & Van den Poel, 2009).

1.1.1 Strategies for customer retention

Of no doubt, long term customers are the backbone of favourably competing businesses and consequently, customer retention remains a priority for thriv-

ing businesses. This is why a lot of research is done to improve retention and reduce attrition.

In research, customer churn has been tackled from two approaches. In the first approach, researchers seek to understand customer churn from a managerial perspective. They carefully and systematically investigate drivers of customer churn using statistical analytical methods and make recommendations from insights. In the second research standpoint, researchers seek to create powerfully predictive customer churn models using machine learning algorithms such as logistic regression (Amin et al., 2019; Huang et al., 2012), support vector machines (Brandusoiu & Toderean, 2013; Farquad et al., 2014), K- nearest neighbours and artificial neural networks (Sharma & Kumar Panigrahi, 2011), among others (Lu et al., 2014; Wai-Ho Au, Chan, & Xin Yao, 2003).

In the real world, both approaches are essential. This is because as churn prediction models help businesses to identify early churn signals and recognise customers who are likely to leave, an understanding of churn drivers will enable them to understand what actions to proactively take.

1.1.2 Adversarial attacks

As businesses continue to rely on machine learning models for automated churn prediction, it should be taken into account that there have been emerging concerns about the vulnerability and robustness of machine learning systems (Grosse, Papernot, Manoharan, Backes, & McDaniel, 2016). Incentivised attackers may seek to manipulate the performance of a model, causing it to malfunction by providing deceptive input and this is known as an adversarial attack (Fei, Xia, Yu, & Xiao, 2020; Rigaki, 2017; Zhang, Chan, Biggio, Yeung, & Roli, 2016). Adversarial machine learning as a research field, studies how machine learning systems act in the presence of adversarial examples that work against it. Experts in industry as well as in academia (Deldjoo, Di Noia, & Merra, 2020; Grosse et al., 2016; Liu, Chen, Liu, & Song, 2017; Yuan, He, Zhu, & Li, 2018) have stressed the need for these vulnerabilities to be considered. Although adversarial attacks have been studied in the contexts of object identification systems, portfolio management systems, intrusion detection systems among others, no attempt have been made to study how churn prediction models are impacted by these attacks and this is vital because an attack on a churn predictor can cause great damage to a company. This is the gap that this research work intends to fill.

1.2 Aim and Objectives

The aim of this study is to investigate the behaviour of customer churn prediction models in the presence of adversarial examples.

1.2.1 Objectives

The objective of this study is to:

- Conduct a systemic review of research literature relevant to customer churn prediction and adversarial attacks.
- Present research questions based on gaps found in literature search.
- Design and implement adversarial machine learning experiments on the basis of research questions.
- Critically analyse and discuss results obtained from implemented adversarial experiments.
- Draw conclusions from analysis made with reference to research questions and recommend future research based on limitations.

1.3 Thesis Structure

This document comprises of six chapters, each focusing on a specific stage of the dissertation. The first chapter being this chapter introduces the subject matter and clearly outlines the objectives of this study. Subsequent chapters are organised as followed:

- **Chapter 2. Literature review** - This chapter critically reviews recent research in customer churn prediction and adversarial machine learning.
- **Chapter 3. Methodology** - This chapter gives a detailed explanation of approaches taken and why they were taken.
- **Chapter 4. Implementation** - This chapter presents how adversarial experimentation was implemented
- **Chapter 5. Evaluation** - This chapter presents the analysis and discussion of results obtained from experiment.
- **Chapter 6. Conclusion** - This chapter summarises the findings from the research conducted and presents suggestions for further study.

Chapter 2

Literature Review

2.1 Customer Churn Prediction

The word churn is said to be derived from two words "change" and turn (Lazarov, München, Capota, & München, 2007). In simpler terms, can be defined as the discontinuation of a contract (Amin et al., 2017; Shaaban, Helmy, Khedr, & Nasr, 2012). As shown in Figure 3.1, customer churn prediction can appear in three main forms:

- **Deliberate churn:** The churner decides to switch to another provider; this could be as a result of their dissatisfaction with the quality of goods or services provided (Vafeiadis et al., 2015), unaffordable pricing, absence of reward for customer loyalty, bad customer support amongst other causes.
- **Incidental churn:** The churner decides to terminate the contract without the aim of moving to another company; this could occur as a result of changes in the customer's circumstances such as change in location, financial problems, illness, etc (Kaur, 2017; Lazarov et al., 2007; Shaaban et al., 2012).
- **Non voluntary churn:** The contract is terminated by the company and not the churner (Shaaban et al., 2012).

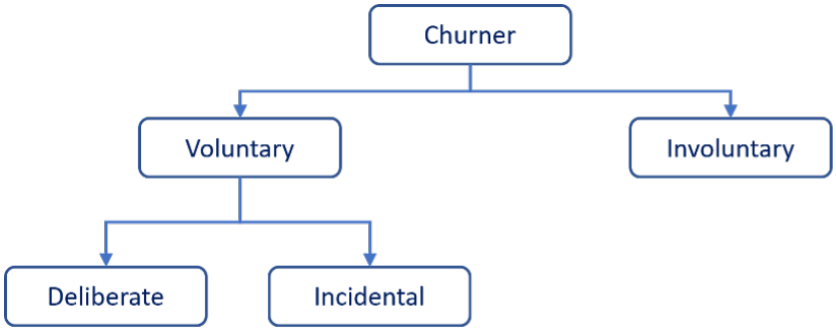

Figure 2.1: Types of customer churn

Deliberate and incidental churn are difficult to predict. Incidental churn often makes up a little percentage of voluntary churners of which the company has little or no ability to reverse their decision even when predicted. On the other hand, proactive measures can be taken when deliberate churners are predicted on time (Lazarov et al., 2007). This makes deliberate churn the class of interest.

2.1.1 Customer Churn Predictive Modelling

As previously mentioned in Chapter 1, long term loyal customers are very valuable to businesses. Today's companies can collect a significant amount of information about their customers including biographical information, purchase habits, service usage histories, etc. The ability to leverage this data to develop models that can help identify customers who show a likelihood and inclination to leave presents a significant business advantage to such companies (Mozer, Wolniewicz, Grimes, Johnson, & Kaushansky, 2000; Verbeke, Dejaeger, Martens, Hur, & Baesens, 2012) as they can effectively conduct targeted retention campaigns such as promotions, surveys, etc.

Developing customer churn prediction(CCP) models involves using Machine Learning (ML) algorithms to train models on pre-processed customer churn datasets. The modelling of customer churn predictors is of great interest to the ML research community, and various industries have implemented the technology, such as telecommunications (Brandusoiu & Toderean, 2013; Huang et al., 2012; Lu et al., 2014; Vafeiadis et al., 2015), banking (Lazarov et al., 2007), retail (De Bock & Van den Poel, 2012), pharmaceuticals, among others. An industry of particular interest to many researchers is telecommunications owing to fierce competition experienced as competitors continue to improve on the services they provide and implement more aggressive marketing strategies (Brandusoiu & Toderean, 2013; Huang et al., 2012; Lu et

al., 2014; Vafeiadis et al., 2015).

Before performing any modelling, it's essential to select suitable datasets and pre-process them appropriately (Verbeke et al., 2012). This section explores how different studies in the relevant literature conduct these tasks.

2.1.2 Customer Churn Datasets

Data retrieval is a vital stage of the data science process. CCP models typically use customer churn datasets obtained from Customer Relationship Management (CRM) databases; such databases will include data (i.e., personal information, lead sources, purchase histories, engagement levels, among others) collected and stored using CRM systems (Mozer et al., 2000; Shaaban et al., 2012).

Data protection legislation, such as the General Data Protection Regulation (GDPR), aims to protect customer privacy by regulating how businesses collect, store, and analyse their data. To ensure that such legislation is adhered to, most research papers (as presented in Table 2.1) train their models on privately collected data sets; however, this results in a lack of broad bench marking and cross-comparisons between results of different studies (Amin et al., 2017). Because of such limitations, it has been difficult for the research community to arrive at any generally accepted consensus with regards to the performance of classification modelling techniques.

A poignant example of this instance is that of studies by (Mozer et al., 2000) and (Hwang, 2004) which both trained models Customer Churn Prediction using neural networks and logistic regression. While the former suggests that neural networks have better accuracy than logistic regression, the latter study makes the exact opposite inference. This phenomenon is caused because both studies used different data sets and is not an isolated occurrence.

This problem is solved by publicly available data sets. Made accessible to independent researchers and research teams, they can foster cross comparability of results, techniques, and methodologies. For data sets from sources that comply with GDPR legislation to be made public, they must meet two conditions for anonymisation as provided by the legislation. According to WP 216, Article 29, the process in use must be irreversible and make it impossible to identify data subjects from the resulting data sets produced (GDPR, 2016).

Customer Churn Prediction models have utilised publicly available data sets such as the IBM Cognos telecommunications customer churn data set, Bank Churn Project's synthesised data set and the Brazilian E-Commerce Public Data set by Olist. Table 2.1 shows a range of studies conducted along with the datasets used and their availability/privacy characteristics.

Study	**Dataset**	**Data Privacy**	**Events**	**Features**
(Verbeke et al., 2012)	Telecom dataset	Private	388,874	727
(Ballings & Van den Poel, 2012)	Newspaper company	Private	129,892	1733
(Chen, Shu, & Sun, 2012)	Food, adventure and telecom	Public	633 to 8,842	20 to 36
(Coussement & De Bock, 2013)	Online gambling operator	Public	3,729	60
(Hou & Tang, 2010)	Bank	Private	19,774	22
(Moeyersoms & Martens, 2015)	Energy	Private	1,000,000	10
(Coussement, Benoit, & Van den Poel, 2010)	Telecom dataset	Private	30,104	956
(Latha, Baburao, & Kavitha, 2019)	IBM telecom dataset	Public	7,043	33
(Vafeiadis et al., 2015)	UCI ML repository, Churn dataset	Public	5,000	19

Table 2.1: Relevant studies, datasets used, nature of privacy, rows and features.

2.1.3 Customer churn data pre-processing

After a data set is obtained, it must be transformed from its raw form to a form that is suitable and optimal for the modelling process. This transformation is essential as the original data set may contain null values (De Caigny et al., 2018; Verbeke et al., 2012), unnecessary features (Vafeiadis et al., 2015), categorical data, and might be imbalanced (Burez & Van den Poel, 2009). Irrespective of the data set used, the data cannot be sent directly through the model due to the presence of some or all of these issues hence they must be appropriately handled. The processes for handling these issues are presented below:

- **Null Values:** Although there are no fixed rules for handling null values, (Verbeke et al., 2012) proposes that this is done based on the percentage of null values: if 5% or more than 5% of an attribute is missing, null entries are handled by imputation. If the number however falls below 5%, then the instances can be removed from the data set. This process is replicated in other studies such as (Coussement & De Bock, 2013) and (De Caigny et al., 2018). Rules of thumb are important as they give us an empirical basis through which tasks can be carried out, however, every case is different and may need to be handled at the discretion of the machine learning engineer.

- **Feature selection:** This is often an issue in customer churn data sets as the number of features may be large,out of which some may be irrelevant features. To find the most important features, feature selection techniques such as chi-square, information gain and correlation coefficient are used. Studies such as (Amin et al., 2017; Lu et al., 2014; Shaaban et al., 2012) show that significant improvement can be seen in results when irrelevant features are removed from the dataset.

- **Categorical variables:** Categorical variables may also present a problem during modelling since they may contain several categories. To handle this Tan et al., (2006) used coarse classification to reduce the number of categories down to four. This method was later replicated by (Ballings & Van den Poel, 2012) and (Verbeke et al., 2012) among others. After coarse classification has been carried out, categorical variables are then finally converted into binary values using dummy encoding.

- **Class Imbalance:** when the number of occurrences of a "majority" class far exceeds that of a "minority" class in a dataset, there exists

a class imbalance problem (Amin et al., 2016). ML models trained with imbalanced data sets do not classify instances of the minority class correctly and perform poorly; this is important as the minority class within a dataset is often the class of interest. Many authors (Amin et al., 2016; Burez & Van den Poel, 2009; Coussement et al., 2010) have also found that misleading results are more frequent when the dataset is unbalanced. Customer churn seldom occurs in service-based industries; therefore, class imbalances are common in datasets used for customer churn prediction. Most researchers (Deldjoo et al., 2020; Lu et al., 2014; Shaaban et al., 2012; Vafeiadis et al., 2015) handle class imbalances by employing random over/under-sampling. A detailed study by (Amin et al., 2016) presents analyses of six different sampling techniques: mega-trend diffusion function (MTDF), synthetic minority, oversampling, adaptive synthetic sampling approach, couples top-N reverse k-nearest neighbour, majority weighted minority oversampling technique, and immune centroids oversampling. Results from the study showed that MTDF performed better than other techniques overall.

Failure to handle these issues,if present, can result in errors and can also result in low model performances(De Caigny et al., 2018).

2.1.4 Algorithms

The research community has used a diverse range of machine learning techniques, as shown in Table 2.2, to create their Customer Churn Prediction (CCP) models. These techniques include Artificial Neural Networks (ANNs), Logistic regression (LR), Support Vector Machine (SVM) classifiers, k-Nearest Neighbours (KNNs), hybrid models and ensemble methods, among others. Background explanations of the techniques used in this study are presented in Chapter 3.

Logistic Regression

Logistic regression (LR) is a popular machine learning technique widely used in Customer Churn Prediction modelling. Many authors (Amin et al., 2016; Farquad et al., 2014; Hou & Tang, 2010; ?, ?; Verbeke et al., 2012) ascribe this popularity to its good predictive performance and comprehensibility. However, while LR handles linear relations between variables better than most classification techniques, a shortcoming of LR is that it does not recognise and consequently take into account the effects of interaction between variables(De Caigny et al., 2018).

Study	**Algorithm**	**Evaluation**
(De Bock & Van den Poel, 2012)	Bagging, LR	Accuracy, AUC
(Verbeke et al., 2012)	Decision Tree (DT), LR, bagging, boosting, SVM, ANN, k-Nearest Neighbours (KNNs)	Accuracy, AUC, TDL
(Ballings & Van den Poel, 2012)	LR, DT, bagging	AUC
(Chen et al., 2012)	SVM, ANN, LR, DT, Random Forest (RF)	Sensitivity, specificity, AUC, TDL
(Coussement & De Bock, 2013)	DT, SVM, LR	TDL, lift index
(De Caigny et al., 2018)	Logit-leaf Model (LLM), LR, DT	AUC, TDL
(Moeyersoms & Martens, 2015)	LR, SVM, ANN, DT	True positive rate, precision, AUC
(Coussement et al., 2010)	LR, Naïve Bayes, ANN, SVM, RF	AUC, TDL

Table 2.2: Studies, algorithms used and evaluation metrics.

In 2014, (Lu et al., 2014) compared the performance of CCP models based on an LR model against a boosted model. The study showed that although simplistic, LR models may perform as well as boosted models. Contrary to this study, however, the performance of LR across other reviewed literature shows that it consistently performs only moderately well and does not generally outperform more complex techniques such as SVC and Artificial neural networks(Coussement & De Bock, 2013).

Support Vector Machine

Like LR, Support Vector Machine (SVM) is a popular supervised ML technique and can be used for regression and classification problems (Chen et al., 2012). SVM-based churn classifiers work by creating one or multiple hyperplanes in a high-dimensional space to optimally differentiate between churned customers and non-churners (Huang et al., 2012).

As shown in Table 2.2, SVM techniques have been tested in many customer churn prediction studies and tend to outperform other traditional ML models. For example, a comparison of seven different algorithms by (Huang et al., 2012) found that SVM revealed the highest rate of true churners.

Likewise, a different study by (Brandusoiu & Toderean, 2013) showed four different SVM models with varying kernel functions performed well with an accuracy of over 80% compared to other models. Furthermore, the same study also showed an SVM with polynomial kernel functions performing even better with an accuracy of 88.56%. (Brandusoiu & Toderean, 2013) argue that the powerful prediction performance of SVM-based models is due to the algorithm's ability to handle collinearities. However, while SVM could be considered the current state-of-the-art CCP modelling technique, a significant drawback is its incomprehensibility which generates a black box illusion as reported by (Farquad et al., 2014).

Artificial Neural Networks

The use of Artificial Neural Networks (ANNs) in CCP modelling is another popular approach. Inspired by the human brain, ANNs mathematically mimic how biological neural networks function. ANNs, according to Mozer et al., (2000), generally perform better than traditional ML models, including SVM, in CCP applications. Several other studies, such as (Huang et al., 2012; Lazarov et al., 2007; Vafeiadis et al., 2015; Wai-Ho Au et al., 2003), also report similar findings derived from the results they obtained. However, notwithstanding these positive characteristics, it has been reported that ANNs are more computationally expensive than other techniques, thus

making them a little less attractive to machine learning engineers with constrained computational power.

Other machine learning based models

Other noteworthy techniques shown in Table 2.2, such as k-Nearest Neighbours and Decision Trees (DTs), have also been used by the research community to create Customer Churn Prediction models. Moreover, attempts at further increasing the predictive power of more mature classes of customer churn predictors have involved the use of state-of-the-art ensemble methods and hybrid models.

Ensemble methods combine several classifiers into an aggregated model and have generally demonstrated superior performance in a magnitude of CCP case studies. Unfortunately, due to their increased complexities, the results obtained from such methods are often difficult to interpret (Shaaban et al., 2012).

(De Bock & Van den Poel, 2012) presented a solution to this issue by incorporating Generalised Additive Models (GAM) concepts into ensemble classifiers. Good predictive performance, relative to training the classifiers individually using LR, was observed by the researchers using this method. Crucially, De Bock & Van den Poel's addresses the issue of reduced interpretability and understandability of the factors affecting customer churns, caused by ensemble methods, using generalised feature importance scores.

Hybrid models combine different algorithms by supplementing their strengths and weaknesses. Several research efforts have explored and adopted this class of model. For example, (De Caigny et al., 2018) proposes a Logit Leaf Model (LLM) as a new hybrid classification algorithm; this LLM incorporates the predictive strengths of DT and LR algorithms while dampening the individual weaknesses in both. Results obtained by De Caigny et al. showed that their LLM model performed better than individualised models based exclusively on DT or LR models alone.

2.2 Adversarial Machine Learning

First discovered by Szegedy et al. in their 2013 research on the properties of neural networks, adversarial machine learning is a field of machine learning that uses deceptive inputs to elicit incorrect responses from a model. The researchers had found that they could induce the misclassification of images crafted with carefully inserted perturbations that are not easily noticeable.

Since the publication of their discovery, many additional studies (Deldjoo et al., 2020; Liu et al., 2017; Nicolae et al., 2019) have successfully applied similar methodologies and observed similar behaviours in other areas of the machine and deep learning field. Early research into adversarial machine learning focused on its application in image recognition; Kurakin et al., (2017)(Kurakin, Goodfellow, & Bengio, 2017), which showed what amount of deceptive input threatens the integrity of an image recognition system, is a prime example of such research.

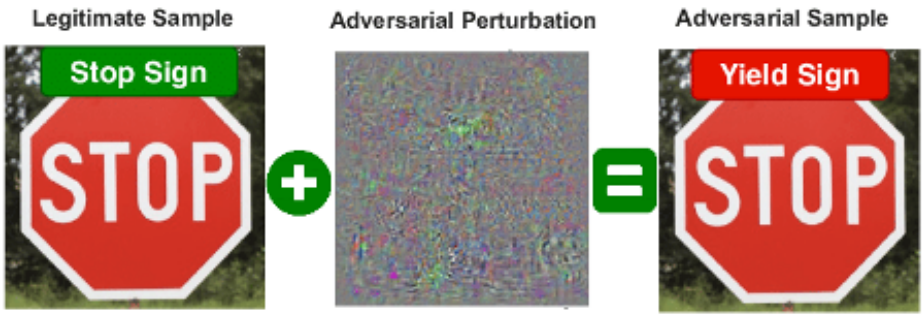

Figure 2.2: Example showing how adversarial examples can impact self-driving cars.

In that study, Kurakin et al. successfully applied their technique to the training datasets of traffic sign recognition systems used by autonomous vehicles highlighting the potential accidents that it could cause. Additional studies that applied adversarial machine learning to speech recognition, portfolio management and other AI areas also observed similar impactful malfunctions.

With Customer Churn Prediction, significant amounts of research attention have focused on performance improvements and a better understanding of the drivers of customer churn. The adoption of different machine learning algorithms, ensemble methods and hybrid combinations resulting from this attention has attained incredible results over the years.

Although these studies have been successful, no investigations into the robustness of CCP models to adversarial attacks or ways of mitigating such

attacks have occurred. Incentivised attackers may seek to manipulate the performance of a CCP model, causing it to malfunction.

This attack could be very damaging to a business because malfunctioning classifiers will cause it to spend resources on retention campaigns to incentivise the wrong target of customers. Recall that companies that understand the importance of customer retention spend as much as 75% of their marking budget on it, according to YSF Magazine. More importantly, many customers not targeted correctly will leave as the business experiences the consequences of customer churn described in Section 1.

The following subsections critically examine the literature on adversarial attacks, their creation and mitigation strategies against them.

2.2.1 Classification of Adversarial Attacks

As illustrated in Figure 2.3, adversarial attacks can occur during the training or production stages of the machine learning pipeline. Attacks are classified based on the stage of the pipeline at which they take place:

- A *poisoning attack* refers to an attack that takes place during the training stage.
- An *evasion attack* refers to an attack that takes place during the production stage.

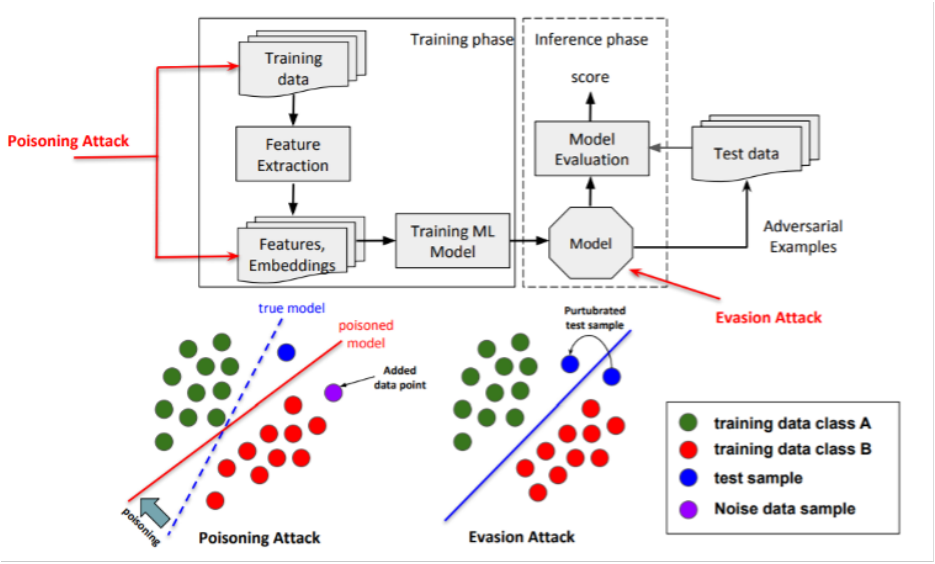

Figure 2.3: Illustration of poisoning and evasion attacks.

Poisoning attacks

As illustrated in Figure 2.3, poisoning attacks involve injecting malicious data into the dataset used for training, causing the model produced to malfunction. Different studies, including (Grosse et al., 2016; Papadopoulos et al., 2021; Yuan et al., 2018) have applied this type of attack after Biggio et al. first proposed it in 2013. It was also proposed that the attack is carried out based on the properties of the algorithm's optimal result in other to achieve maximum degradation of the model's performance.

Evasion attacks

Unlike poisoning attacks, evasion attacks involve injecting false data during the inference stage and therefore do not infer with the training stage (Kurakin et al., 2017). Evasion attacks are often referred to as decision time attacks since they always attempt to evade the decisions made by the classifier when tested.

Szegedy et al.'s pioneering study in 2013 first made this observation when they discovered that properly trained models misclassify adversarial examples with high confidence after perturbations, unseen to the human eye, are added to them; Figure 3.8 provides an illustrative example of this description. Since their discovery, researchers have implemented evasion attacks in spam (Zhang et al., 2016) and network intrusion detection (Papadopoulos et al., 2021) applications, among others.

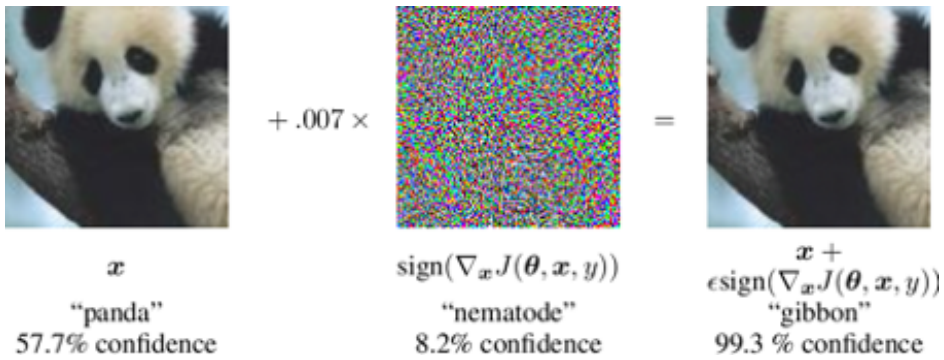

Figure 2.4: Illustrating the Effect of Evasion Attacks.

2.2.2 Crafting Adversarial Examples

In traditional machine learning, adversarial attacks are generally more impactful and effective when malicious injections are made in the training stage as opposed to the testing stage. The perturbations are added to either the dependent variables (features) of the data or the labels. The latter is referred to as label flipping and researchers(Papadopoulos et al., 2021; Taheri et al., 2020) agree that it is the most effective for achieving misclassification. The label flipping threat model influences the model performance by providing misclassified labels during the training process(Taheri et al., 2020). This is due to the fact that incorrect labels in training data leads to incorrect training of model. The flipping of labels can either be done randomly or targeted. Random label flipping involves selecting an arbitrary fraction of the data and flipping them. In contrast, targeted flipping other wise involves flipping only labels of data from a particular significant class in an attempt to increase the misclassification of that class(Kantartopoulos et al., 2020).

A theoretical example of such an attack could be label flipping a dataset used by a credit card fraud detector to cause misclassifications of fraud incidents. Such an attack could use random label flipping where the labels of fraudulent and non-fraudulent data are flipped. Alternatively, the attack could use targeted label flipping where the labels are flipped to cause maximum damage to the model performance with regards to its ability to predict the target class of which in this case is the *fraudulent* case.

Research carried out by (Kantartopoulos, Pitropakis, Mylonas, & Kylilis, 2020; Fei et al., 2020; Taheri et al., 2020) have shown that SVM-based ML algorithms can often withstand adversarial attacks are therefore the most appropriate choice for deployments in adversarial environments.

Artificial Neural Networks can also be affected by label flipping; however, label flipping isn't very effective in tampering with the performance of neural network models; this is because training occurs in batches, and crucially, such an attack demands that flipped data be present in each batch used during training (Papadopoulos et al., 2021).

Szegedy et al. proposed a more efficient approach using the Fast Gradient Sign Method (FGSM) in 2013. This technique can quickly generate new adversarial data as the manipulation is automated. The goal of FGSM is to increase the cost of the neural networks model such that the cost exceeds the loss gradient with the input data to produce new data.

Studies such as Kurakin et al. (2017), Yang et al. (2018), and Zhang et al. (2016) show how successful FGSM is in significantly degrading the performance of classifiers. Notwithstanding, in 2017, Rigaki posited that since FGSM-based attacks perturb the entire dataset, then such attacks could

conceivably be detected by application firewalls with little effort.

Furthermore, Rigaki also proposed using Jacobian Salient Map Attacks (JSMA), which only require portions of the dataset to be perturbed, unlike FGSM. First introduced in 2015 by Papernot et al. (2015), Wiyatno & Xu (2018) extended it when exploring random and targeted forms of the attack.

Later, Jeong et al. investigated both FGSM and JSMA techniques in 2019; a CNN model, which could classify intrusion detection events with an accuracy of 99.5%, was subjected in their study to both attacks. While the FGSM attack reduced the model's accuracy to 50.85%, the study demonstrated that the more efficient JSMA attack had also reduced the accuracy by the same amount.

However, many researchers may argue that the effectiveness of JSMA is not enough to compensate for its computational power requirement (Papadopoulos et al., 2021; Papernot et al., 2015; Rigaki, 2017).

2.2.3 Defending models from adversarial attacks

Mitigations can be taken against adversarial attacks and can either be reactive or proactive. Such measures are especially crucial if the models are for use in adversarial environments.

With reactive mitigations, adversarial behaviour is detectable only after the system has been trained and deployed; one may reconstruct or sanitise the entire dataset using statistical analysis or multiple classification systems to implement such mitigations (Kantartopoulos et al., 2020). In contrast, Yuan et al. (2018) propose training models proactively using adversarial examples to prepare them for such attacks.

2.3 Conclusion

In conclusion, customer churn prediction models are very important for favourably competing businesses as it helps them identify churning customers and target them with retention campaigns. For the model to be trained, data is first retrieved from CRM database and pre-processed adequately. Various machine learning techniques have been used in the past including logistic regression, artificial neural networks, decision trees, ensemble techniques and hybrid methods amongst others. The present performance of models in predicting churn has significantly improved over the years however, some gaps still exist. Some notable gaps includes: The use of more complicated techniques, improving comprehensibility of complex, highly predictive models, extending class imbalance investigations to include multi-class systems, ex-

ploring how different feature extraction techniques affect churn prediction and investigating the robustness of CCP models to adversarial examples. As this study chooses to focus on the last-mentioned gap, the gap leads to the following research questions which form the basis for this dissertation:

- How robust are customer churn prediction classifiers to adversarial attacks?
- Which adversarial attack has the highest impact on the models?

Chapter 3

Methodology

The main objective of this study is to analyse the performance of a customer churn prediction model in the presence of adversaries. To that end, the following experiments were carried out.

Experiment 1: Evaluation of customer churn prediction models using clean dataset.

Experiment 2: Evaluation of customer churn prediction model performance in the presence of Fast Gradient Sign Method (FGSM) adversarial samples at different perturbation degrees.

Experiment 3: Evaluation of customer churn prediction model performance in the presence of Deep fool adversarial samples at different overshoot values.

Experiment 4: Evaluation of customer churn prediction model performance in the presence of Jacobian Saliency Map Attack (JSMA) adversarial samples at different perturbation rates.

This chapter discusses the working principles of the modelling and attack methods used for this research.

3.1 Model Methodology

Three machine learning algorithms were selected for modelling. They include linear regression, support vector machines and artificial neural networks. This section provides detailed insight into how they are trained.

Logistic regression: Invented by Joseph Bekson in 1944, logistic regression is one of the most widely used machine learning classification algorithm. Logistic regression models classify by estimating the probability that an outcome belongs to each class. It is mathematically expressed by the logistic function below:

$$\text{prob}\,(y=1) = \frac{e^{\beta_1+\sum_{k=1}^{k}\beta_k x_k}}{1+e^{\beta_1+\sum_{k=1}^{k}\beta_k x_k}} \tag{3.1}$$

Where:

- y is the probability that an outcome is a class (e.g., $y = 1$ if it is that class, $y = 0$ if it is not);
- x_1, x_2, ..., x_k, are all feature inputs;
- β_1, β_2, ..., β_k are the coefficients of regression which are calculated by the maximum likelihood method on the basis of training data provided.

The model assigns a probability value between 0 and 1 to each possible class, with the sum of all probabilities being equal to one. An understanding of the logistic function helps explain logistic regression. The function is a Sigmoid function and unlike normal linear functions, takes in predictor inputs and restricts output to a value between zero and one.

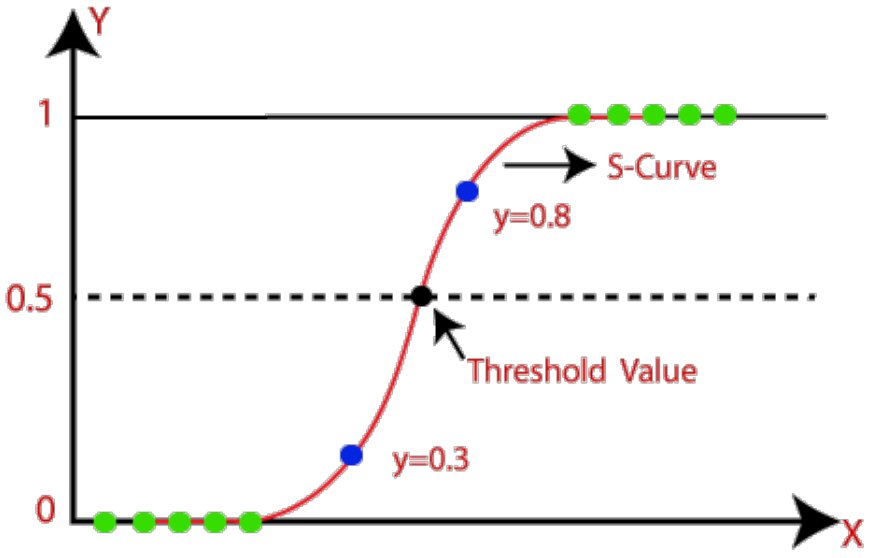

Figure 3.1: Logistic Regression

Support vector machines: The support vector classifier is trained by searching for the optimal hyper-plane which separates the data into classes. However, if the input feature vector is not linearly separable, SVM using a kernel trick maps the data into a higher dimensional feature space after which it finds a hyperplane which separates the data into the desired classes. The model can be expressed mathematically as:

$$f(x) = sgn\left(\sum_{i}^{M} y_i \alpha_i \varphi(x_i, x) + \partial\right) \tag{3.2}$$

Where:

- M represents the number of training set samples;
- x_i is a support vector with $\alpha_i > 0$;
- ϕ is a kernel function which could be polynomial or radial basis;
- x is an unknown sample feature vector;
- ∂ is a threshold.

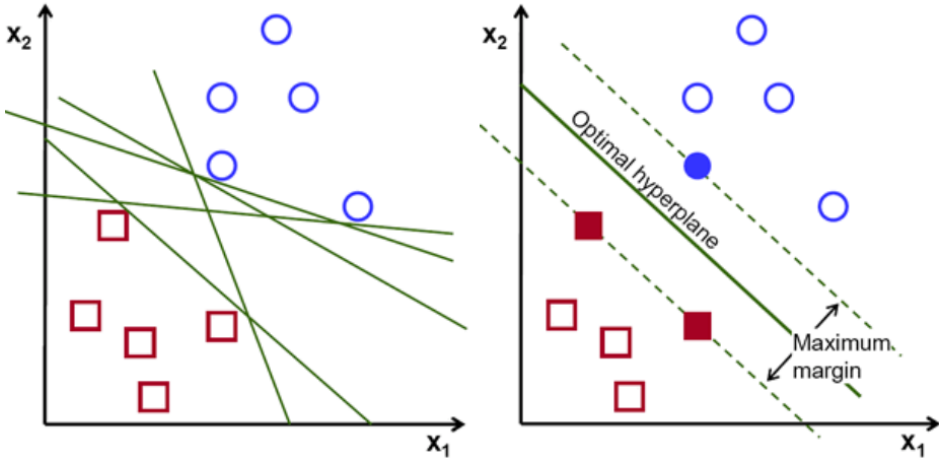

Figure 3.2: SVC selection of Optimal hyperplane

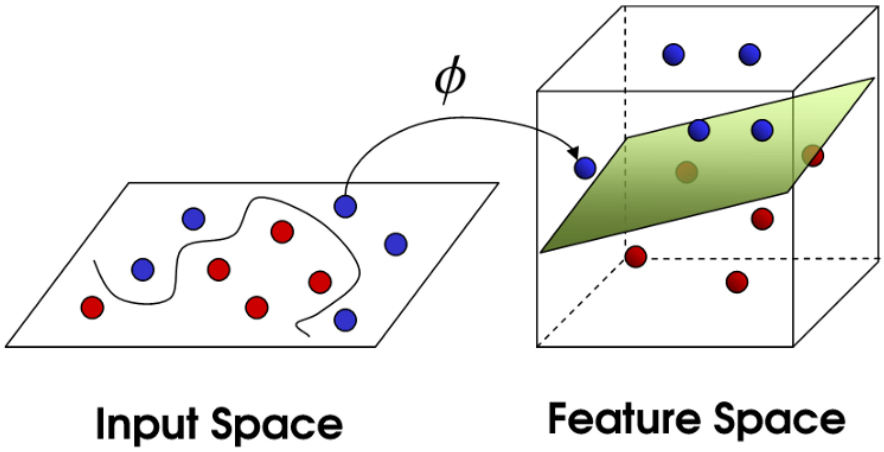

Figure 3.3: Support vector classifier transformation into higher dimensional space

Artificial neural networks: A neural network, also known as an artificial neural network (ANN) is a subset of machine learning inspired by the human brain, which mimics and simulate the neural activities of biological neurons. ANNs consist of layers of artificial neurons. The layers include an input layer, hidden layer(s), and an output layer. During the learning process of Neural networks, they systematically alter the synaptic weights connecting neurons. Neurons in network layers apply activation functions such as the Sigmoid function and this generates an output that serve as input to neurons of next layer. The output obtained by neurons are computed using the equation written as follows:

$$\text{Output} = f\left(\sum_{i=1}^{l} w_{jl} f\left(\sum_{i}^{D} w_{li} x_i\right)\right) \tag{3.3}$$

Where:

1. D, L and J are number of input, hidden and output layer, respectively;
2. f is an activation function;

The neural network architecture used for this study is as proposed by as used by (Warzynski & Kolaczek, 2018). Specifications include:

1. 3 hidden layers with each layer consisting of 100 neurons.

2. Rectified non-linearity as the activation function for every neuron.
3. SoftMax activation function was used for final layer to restrict output value.
4. Standard gradient descent is used for neural network training with batches of size 1000 that are split into training and validation sets, using 100 training epochs per iteration.

3.2 Model Evaluation

To comprehensively evaluate our models and attack algorithms, the following test scores are used: accuracy, precision, recall, F-1 score, and area under-cover (AUC). These evaluation parameters are calculated using values from the confusion matrix, seen in Figure 3.4. Confusion Matrices help to summarise the prediction results such that the number of correct and incorrect predictions are summarised with count values and broken down by each class This way, it is easy to visually examine how the model makes predictions. TP, FP, TN and FN represent True positive, false positive, true negative and false negative cases respectively.

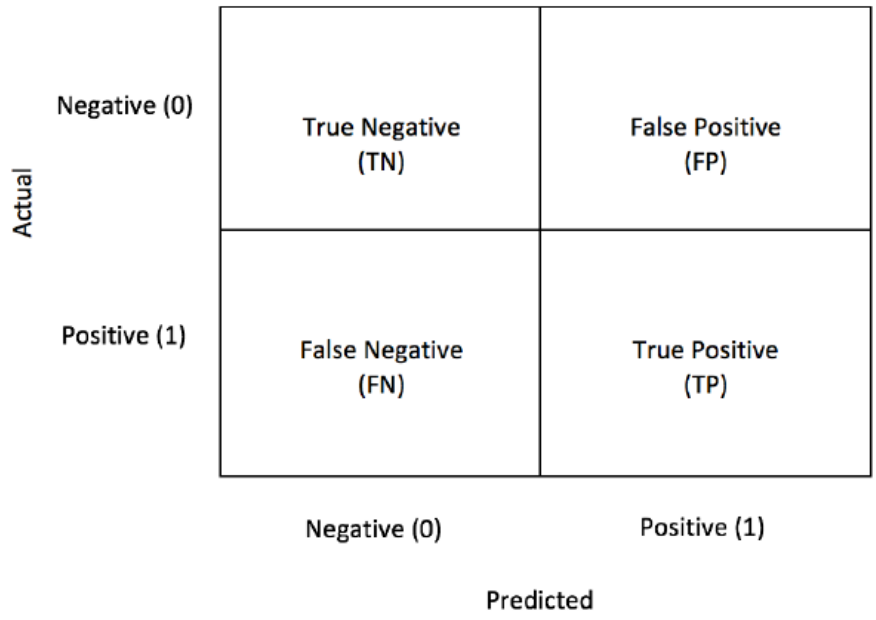

Figure 3.4: Confusion Matrix

Accuracy

Accuracy can be defined as the proportion of correct predictions to the total number of predictions made by a classifier:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.4}$$

The accuracy is used in this study as the main measure of overall performance

Precision

Precision refers to the proportion of correctly predicted positive cases:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.5}$$

Recall

Recall shows what proportion of churners that were correctly identified:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.6}$$

F-1 score

Classifiers cannot be efficiently described using precision or recall alone because a classifier can performance well in one of those indices not perform well on the other. As a result, with F1-score, precision and recall are both combined into a single test metric, and this is more efficiently used to evaluate the performance of the classifier. F-measure is the harmonic mean of precision and recall:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} \cdot \text{Recall}} \tag{3.7}$$

Area under ROC curve

A receiver operating characteristic (ROC) curve is a graph which shows how classification models perform at all classification probability thresholds (probability above which a sample is classified as positive). It is a plot of true positive rate (recall) and false positive rate (also referred to as specificity). ROC curves are plots of TPR vs. FPR at varying probability thresholds of classification. When the threshold is lowered, more samples are classified

positive, consequently increasing the number of False and True Positives. The figure depicts a typical ROC curve.

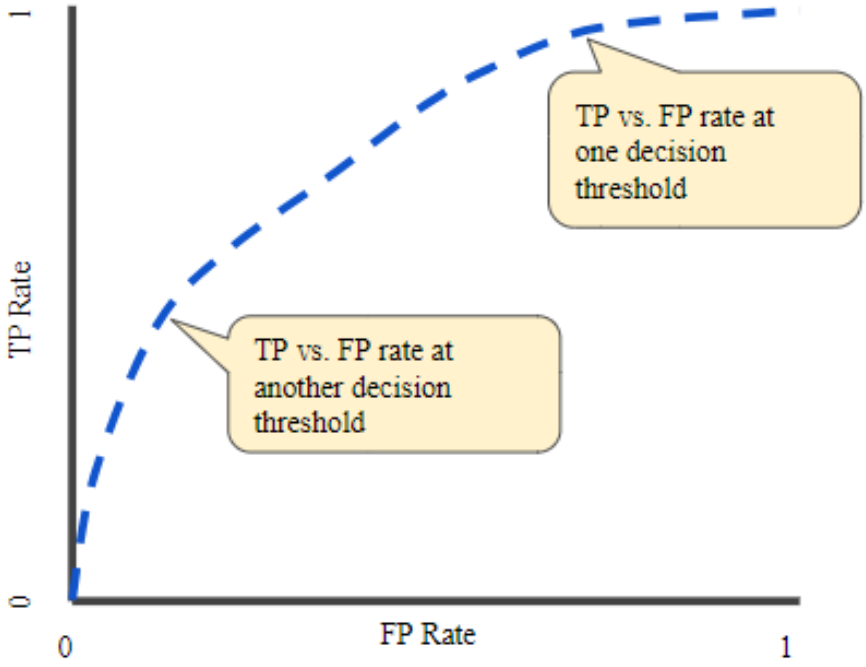

Figure 3.5: Typical ROC Curve

3.3 Attack Methodology

In this study, the impact of four adversarial attack methods is compared. This section provides a theoretical background of how the attacks algorithms. Fast Gradient Sign Method is the first attack to be explored in this section, followed by the deep fool algorithm which ... Carlini algorithm is then explored after which the final algorithm, JSMA is discussed.

Fast Gradient Sign method (Goodfellow, Shlens, & Szegedy, 2015)

FGSM is the fastest technique for crafting adversaries. It was created based on the hypothesis that neural networks are unable to resist linear perturbation due to their highly linear nature. This technique utilises the gradients of the model to create adversarial samples; it estimates the gradients of the linearly approximated cost function, such as mean squared error or cross-entropy to the input features and then uses the sign of that gradient to create newly perturbed adversarial samples. When the cost function

is linearised around θ which is the point intended for misclassification, the optimal max-norm perturbation is obtained. The equation below is used to estimate the max-norm perturbation, given that:

$$\eta =\in sign\ (\nabla_x\ J(\theta, x, y)) \tag{3.8}$$

Where:

1. $\in$ is the degree of perturbation;
2. θ represent the model parameters;
3. x is the model feature input;
4. y is the targets associated with feature input;
5. $J(\theta, x, y)$ is the cost function used to train the neural network.

In summary, the following steps are taken for FGSM:

1. Forward-propagation of features through our model
2. Cost function calculation
3. Gradient back-propagation of features
4. Nudge feature values in the direction to which the loss value is maximised.

The code below is a function `create_adversarial_function()` which can be used to implement the algorithm.

```
def create_adversarial_pattern(input_features, input_label):
    with tf.GradientTape() as tape:
        tape.watch(input_features)

        # forward propagate image and retrieve the loss
        prediction = pretrained_model(input_features)
        loss = loss_object(input_label, prediction)

        # obtain the gradient of the loss with repsect
        # to the input features
        gradient = tape.gradient(loss, input_features)

        # Get the sign of the gradients (directions)
        signed_grad = tf.sign(gradient)

        return signed_grad
```

Figure 3.6: Creating the adversarial pattern in Python.

Deep Fool (Moosavi-Dezfooli, Fawzi, & Frossard, 2016)

The deep fool algorithm presents a simple and efficient technique to fool models. DeepFool algorithm's main task is to generate adversarial features with the lowest possible of perturbation which is usually imperceptible. The figure illustrates how an adversarial example attack a linear model $f(x) = w^t x + b$. With this proposed algorithm, assuming the hyperplane, F is a binary classification model which separates two classes, input x_0 is projected onto the hyperplane. The input is then pushed a minuscule distance (η) beyond this boundary to misclassify it. This distance is known as the overshoot distance.

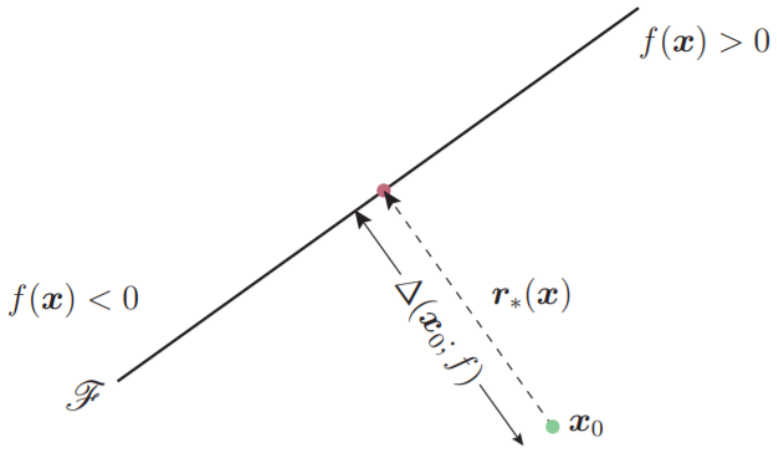

Figure 3.7: Illustration of Deep Fool Attack

The robustness of that model f at an input point x_0 is the perpendicular distance between that point x_0 and the hyperplane as illustrated in the figure above. This minimal perturbation is required to change the decision of the classifier and can be calculated using:

$$r_* (x_0) = -\frac{f(x_0)}{||w||_2^2} w \tag{3.9}$$

If the classification function, f is not linear, the robustness, $\Delta(x_0; f)$, is estimated by first linearising the function. The figure below summarises how the algorithm works.

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(x_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

Figure 3.8: DeepFool algorithm for binary classifiers.

Jacobian Salient Map attack (Papernot et al., 2015)

Given that $f : R^n \rightarrow R^m$ is a function in which R^n takes x as an input such that $x \in R^n$ and generates $f(x) \in R^m$ as an output vector. Then, the Jacobian of f is a matrix which assumes the shape $(n \cdot x)$ an is explicitly expressed as J in:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Figure 3.9: Jacobian of a matrix

Jacobian Salient Map attack (JSMA) was first introduced by (Papernot et al., 2015). The method proposes greedy algorithm which uses the Jacobian matrix seen below to determine the selection of features to be perturbed:

$$s_t = \frac{\partial t}{\partial t};\ s_o = \sum_{j=t} \frac{\partial t}{\partial t}\ ; s\left(x_i\right) = s_t \left| s_o \right| \ .\ (\ s_t < 0)\ .\ \ .\ (\ s_o > 0) \tag{3.10}$$

Where:

- s_t is the Jacobian of the target class t;
- s_0 is the summation of all Jacobian values of the non-target class;

Nudging the values of select features using this method will have a significant impact on classifier, increasing its tendency to classifier to misclassify labels as the target class.

Chapter 4

Implementation

4.1 Software and hardware

This project was simulated using the python language. Python language was used for many reasons. They include:

1. It is free and open source, so it is readily available for anyone
2. It has a syntax that is easy to understand
3. It is independent of platform and hence runs on most operating systems such as Linux, windows and macOS
4. It has extensively selected libraries that simplify data wrangling, visualization, machine learning, etc.

In this research, the following libraries were used to achieve the tasks below:

1. NumPy, Pandas - used for data wrangling
2. scikit-learn - used for machine learning modelling and evaluation
3. TensorFlow, Keras - used to create deep learning models
4. matplotlib, seaborn - used for visualisation
5. Adversarial Robustness Toolbox (ART) - used to craft adversarial examples

The python web integrated development environment (IDE) used for modelling is the Google Colaboratory (Colab) web IDE. Google Colab is a cloud Software as a Service (SaaS) platform which is excellent for machine learning

Specificaton	Configuration
GPU	1xTesla K80 , compute 3.7, having 2496 CUDA cores , 12GB GDDR5 VRAM
CPU	1xsingle core hyper threaded Intel Xeon Processors at 2.3Ghz i.e(1 core, 2 threads)
RAM	13 GB Available
Disk	100 GB Available
Cache size	56320 KB

Table 4.1: Specification and configuration of virtual machine used

and deep learning, requires no prior set ups and with different tiers, contains pre-installed machine learning libraries. With regard to cost, Google Colab provides a free tier with a powerful graphics processing unit (GPU) to help speed up computation. The virtual machine system configuration provided by Google Colab is summarised below.

4.2 Experiment Flow

In the first stage of this research, the data IBM Telco churn dataset described in Section 4.3 was pre-processed using steps shown in Section 4.4 below. The pre-processed data was randomly split into a 4:1 ratio for model training and testing respectively. After splitting was completed, the next part of the project involved training models and testing them firstly with clean data as seen in Figure 4.2 and then testing them with adversarial examples generated using the different attack methods as illustrated in Figure 4.3.

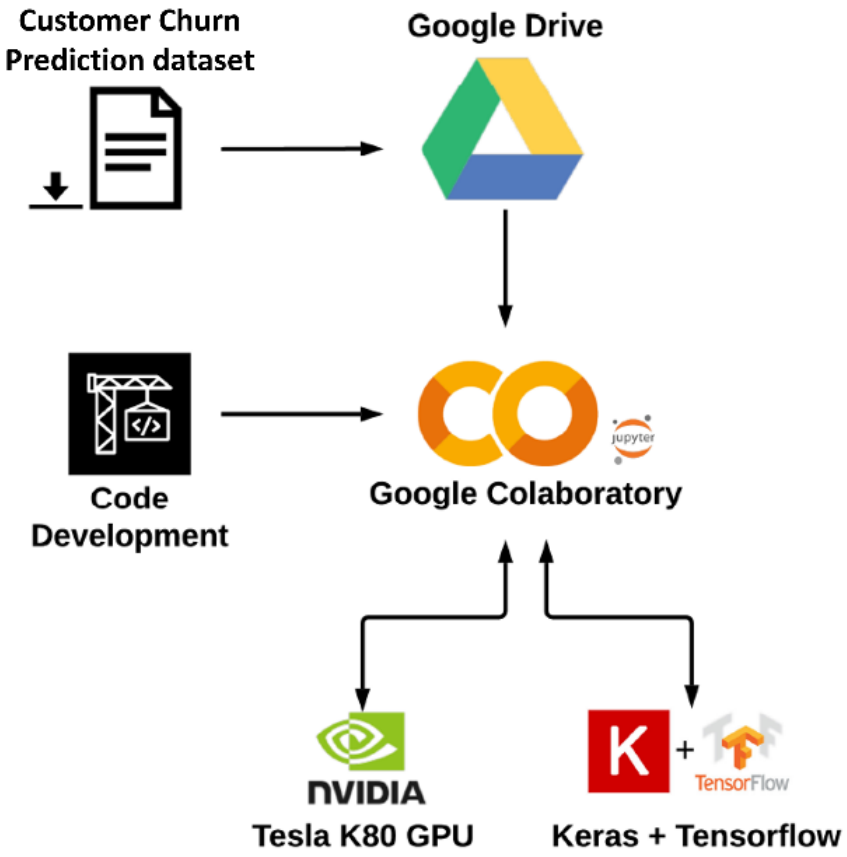

Figure 4.1: Experiment test bed

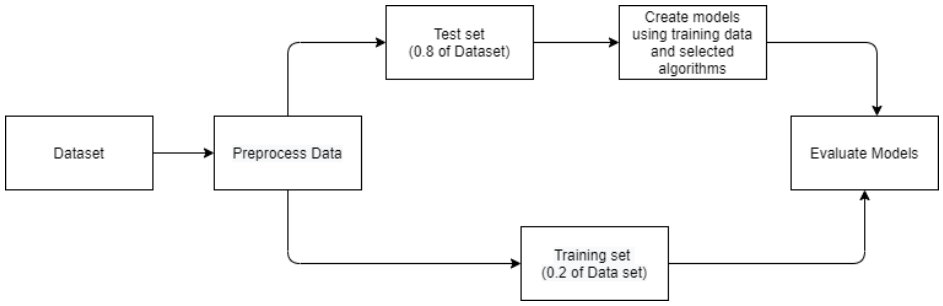

Figure 4.2: Testing of data with clean samples

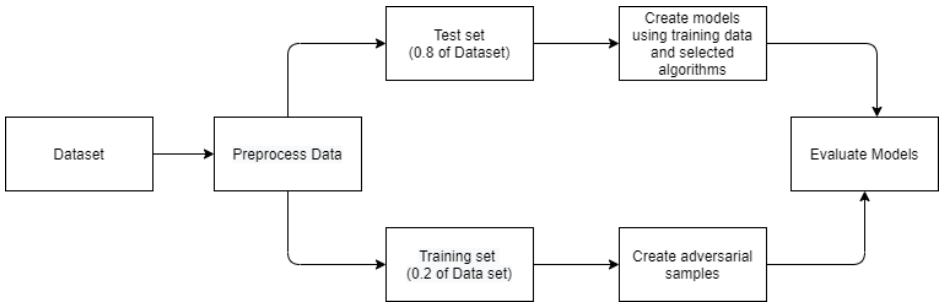

Figure 4.3: Testing of model with adversarial sample

4.3 Datasets

The dataset used for this study is the IBM Watson Telco churn dataset as used by (Ebrah & Elnasir, 2019) and (V J, 2019). Released in 2015, the dataset contains data of 7043 fictional customers of a California based company which provides telecommunication and internet services. The dataset contains 7043 instances and 21 variables with each instance representing a unique customer. The dataset consists of 20 features, of which 3 are numerical, 5 are binary and 12 are categorical with three or more categories. It also contains a target variable *churn* denotes whether the customer is a churned customer or not of which 73.46% are not churners and 26.53% are churners. Figure 4.4 contains more details of the data set used.

Category	Variable	Description
Classification labels	*Churn*	Whether the customer churned or not (Yes or No)
Customer services booked	*PhoneService*	Whether the customer has multiple lines (Yes, No, No phone service)
	MultipleLines	Customer's internet service provider (DSL, Fiber optic, No)
	InternetService	Customer's internet service provider (DSL, Fiber optic, No)
	OnlineSecurity	Whether the customer has online security (Yes, No, No internet service)
	OnlineBackup	Whether the customer has online backup (Yes, No, No internet service)
	DeviceProtection	Whether the customer has device protection (Yes, No, No internet service)
	TechSupport	Whether the customer has tech support (Yes, No, No internet service)
	StreamingTV	Whether the customer has streaming TV (Yes, No, No internet service)
	StreamingMovies	Whether the customer has streaming movies (Yes, No, No internet service)
Customer account information	*Tenure*	Number of months the customer has stayed with the company
	Contract	The contract term of the customer (Month-to-month, One year, Two year)
	PaperlessBilling	Whether the customer has paperless billing (Yes, No)
	PaymentMethod	The customer's payment method (Electronic check, mailed check, Bank transfer (automatic), Credit card (automatic))
	MonthlyCharges	The amount charged to the customer monthly
	TotalCharges	The total amount charged to the customer
Customers demographic info	*customerID*	Customer ID
	Gender	Whether the customer is a male or a female
	SeniorCitizen	Whether the customer is a senior citizen or not (Yes, No)
	Partner	Whether the customer has a partner or not (Yes, No)
	Dependents	Whether the customer has dependents or not (Yes, No)

Figure 4.4: IBM Telco Churn Dataset: categories, variables and descriptions

4.4 Pre-processing

It is essential to prepare data before it is used to train a machine learning model. The first step taken was to convert all categorical variables to numerical forms. This was achieved by binary encoding and One-Hot encoding. This was followed by normalisation as features of the datasets have different ranges. Finally, Synthetic Minority Oversampling Technique (SMOTE) is applied to balance out classes.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

data_scaled = scaler.fit_transform(data)
```

Figure 4.5: Creating the adversarial pattern in Python.

Categorical variable handling Machine learning models such as artificial neural networks, logistic regression and support vector classifiers are algebraic thus, they only accept numerical inputs. For modelling to be made possible, it is required that categories are first transformed into numbers. Binary features containing were converted into numerical data and this was achieved using a simple value replacement technique. The value replacement strategy provides the liberty to assign numbers to each category. For instance, *yes* and *no* categories were replaced with *1* and *0* respectively. The `replace()` function in the pandas library was used to achieve this.

Categorical features with more than two variables were transformed to numerical data by using the One-Hot encoding strategy. This strategy converts all categorical values into new variables and assign 1 or 0 to the columns. The `get_dummies()` method in the pandas library was used to achieve this

Normalisation After categorical variables were converted into numerical variables, normalisation was carried out with the goal of transforming the numerical values of all columns in the dataset to fit into a common scale, without tampering with the differences in the distribution of values. This was necessary because features of the numeric datasets had different ranges. For instance. The gender feature now had a value between 0 and 1 while features such as ... had . Normalisation was used to transform all variables to the same range. Scikit-learn's MinMax scalar function was used to achieve this as seen in code below.

Data balancing: As mentioned in Section 4.3, non-churners make up 73.46% of the data while only 26.53% are churners. This leads to a class imbalance problem, and it may lead to the difficulty of the classification algorithm to appropriately learn about the minority class of which in this case are the churners. This consequently results in poor classification power of the customer churn prediction model. To combat this, sampling strategies are used to balance the class distribution such as over sampling and under sampling methods. In this research, Synthetic Minority Oversampling Technique

```
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)

X_res, y_res = sm.fit_resample(X, y)
```

Figure 4.6: Creating the adversarial pattern in Python.

(SMOTE) was used. This technique generates synthetic data instances of the *churn* minority class so as to balance the class distribution. The `SMOTE()` function from the imblearn library was used to implement this as seen in code below:

4.5 Model Development

Model development is a vital stage in this study. As previously stated, the following algorithms were selected for model development. They include:

1. linear regression represented as LR;
2. support vector machines classifier represented as SVC;
3. artificial neural networks, represented as ANN.

Logistic regression and SVM were implemented using Scikit-learn library while the artificial neural network model was implemented using Keras. The general working principle of these algorithm can be found in Chapter 3. Presented below are the hyperparameters of all classifiers used in this experiment.

Logistic Regression To fit the Logistic regression classifier, a conditional probability threshold of 0.5 is used. This is the default present value used for classification.

Support Vector Machines Parameter selection for support vector machines depend on the utilized kernel functions. For this experiment, RBF has been selected for use (Vafeiadis et al., 2015) . The RBF kernel function can be represented as:

$$k\left(x,y\right)=exp\left(\frac{-|\left|x-y\right|\right|^{2}}{2\sigma^{2}}\right) \tag{4.1}$$

Classifier	**Parameters**
LR	Conditional probability = 0.5
SVC	Kernel=RBF, C=40, σ=0.01
ANN	Layer 1 = 100 neurons, Layer 2= 100 neurons, Layer 3= 100 neurons, Activation = Relu, Loss = Categorical crossentropy, Optimizer = Adam

Table 4.2: Parameters for fitting classifiers

Where σ is the variance and the constant of the equation ($\sigma = 0.01$ was selected). SVM parameter were selected as used in (Vafeiadis et al., 2015) in which they were obtained by grid search hyperparameter tuning.

Artificial Neural Network A basic back propagation algorithm was selected to train the model. Parameters used to fit this model are as used in (Pacheco & Sun, 2021). It includes two layers each consisting of 256 and 128 neurons consecutively. More information about this model are as found in the table below.

4.6 Generating adversarial examples

The main aim of this study is to develop models and evaluate their performance in the presence of adversarial examples. As previously stated, the three selected adversarial attack techniques to be used in this study include Fast Gradient Sign Method, Deep Fool and Jacobian Based Saliency Map Attack. These are the most widely studied attacks in model evasion scenario and are all available in the Adversarial Robustness Toolbox (ART) Library provided by IBM which helps simplify the implementation of these attacks. Other alternate library that can be used include the Cleverhans, adverTorch and Foolbox libraries, however, ART was chosen for its ease of use and availability of support. The general working principle of these adversarial methods can be found in Chapter 3.
Presented below are the parameters of all adversarial methods used in this experiment.

Fast Gradient Sign Method Fast Gradient Sign Method has only one parameter ϵ to control which is the degree of perturbation. FGSM adversaries were created using varying values of ϵ as seen in table 4.3 These adversaries,

Attack	**Parameters**
FGSM	epsilon=0, 0.1, 0.20, 0.3 ,0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
Deepfool	epsilon=0,0.000001, 0.0000020, 0.000003, 0.000004, 0.000005, 0.000006, 0.000007, 0.000008, 0.000009, 0.00001
JSMA	gamma=1%, theta=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Table 4.3: Attacks and Attack Parameters

which are modifications of the clean test set were all tested on all three trained customer churn classifiers trained.

Deep Fool For deep fool technique, the deep fool algorithm automates the process of searching for the minimum perturbation, however, an overshoot of this minimum perturbation can be controlled; this is known as the overshoot parameter ϵ. Deep fool adversaries were created using different values of ϵ as seen in table 4.3 These adversaries, which are modifications of the clean test set were all tested on all three trained customer churn classifiers trained.

Jacobian Saliency Map Attack As discussed in Section 3.3, with Jacobian Saliency Map Attack cases, it is possible to select a fraction of the test set to perturbed. This fraction is represented as γ and is a value between 1 and 10. In this research, a fixed fraction of $\gamma = 0.5$ was used. Perturbation rates of JSMA attacks, θ represent how much each modified feature is perturbed. JSMA adversaries were created using different values of θ as seen in table 4.3.
All results obtained after the implementation of these experiments are presented and discussed in the next chapter.

Chapter 5

Evaluation

5.1 Results

5.1.1 Baseline model with no adversaries

Logistic Regression (LR), Support Vector Classifier (SVC) and Artificial neural network (ANN) Customer churn classifiers were trained and tested with the IBM Telco churn dataset as reported in Chapter 4. The results obtained from evaluating these classifiers using unperturbed test data are presented in table below and are as graphically illustrated in bar plot shown in figure.

Model	Accuracy (%)	Precision (%)	F-1 Score (%)	Recall (%)
LR	85	84	85	86
SVC	77	73	79	85
ANN	80	76	82	88

Table 5.1: Evaluation of classifiers with unperturbed test data

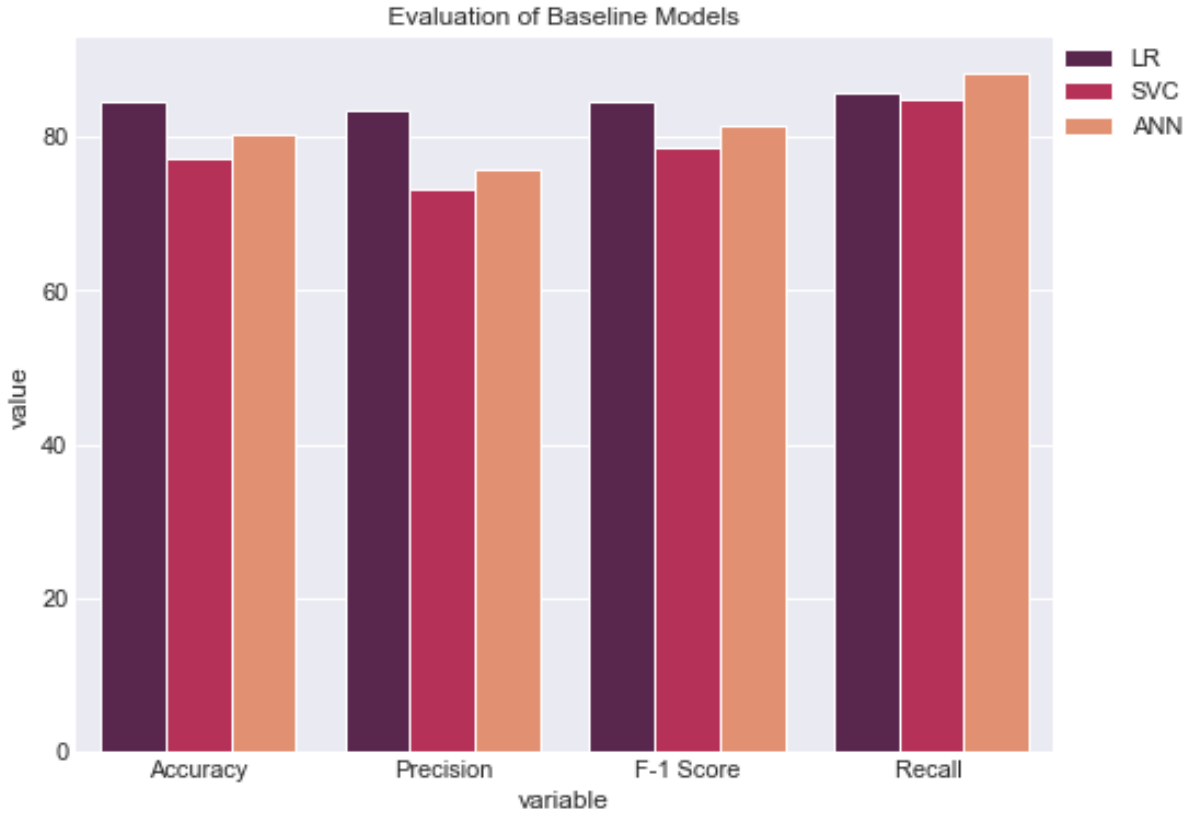

Figure 5.1: Model evaluation using unperturbed test data

The 2x2 confusion matrix obtained from the evaluating the performance of the LR, SVC and ANN classifiers are as seen in figure 5.2,5.3 and 5.4 respectively. The matrix provides a comparison of target and predicted values. The ROC curves generated for each classifier are as presented in figure 5.5, 5.6, 5.7 respectively. The area under the curves (AUC) as seen n the graph as 94, 84 and 89 for LR, SVC and ANN classifiers respectively.

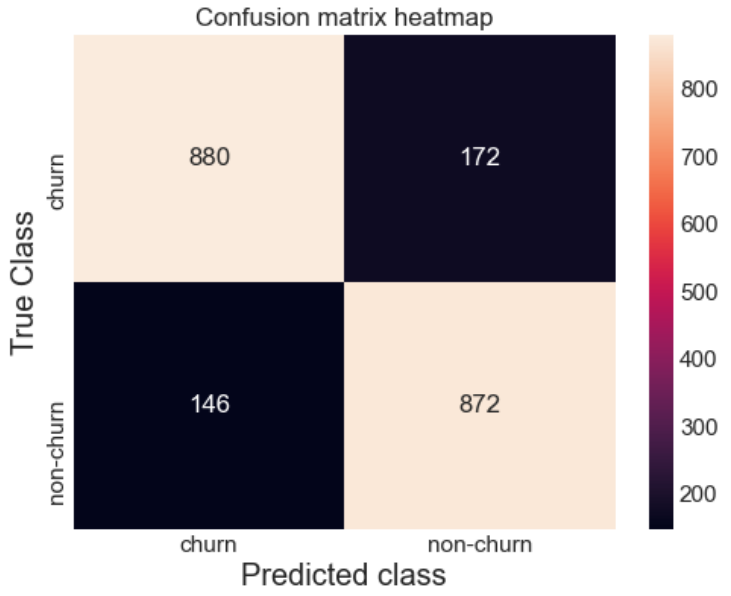

Figure 5.2: Logistic Regression (LR) Confusion matrix heat map

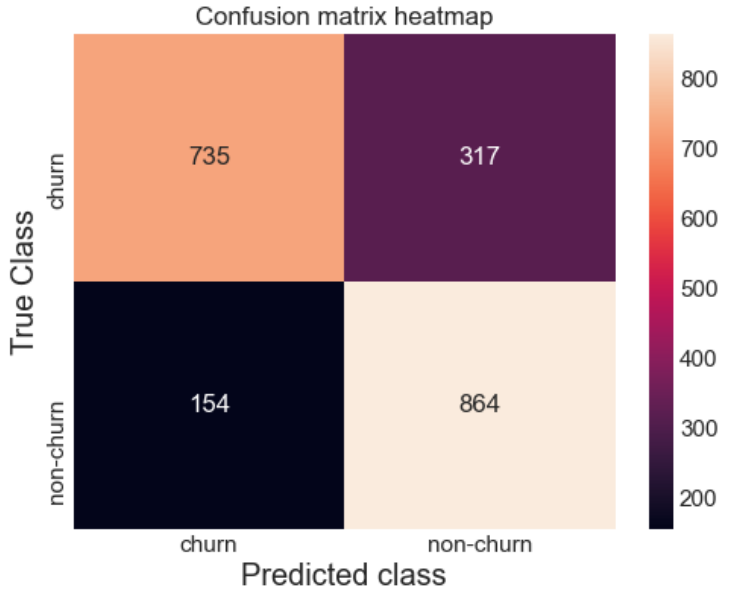

Figure 5.3: Support Vector Classifier (SVC) Heat Map

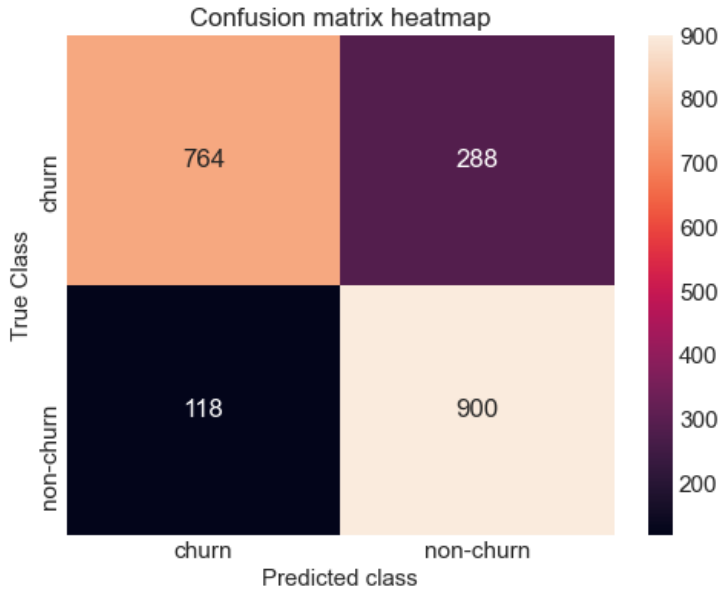

Figure 5.4: Artificial Neural Network (ANN) Heat map

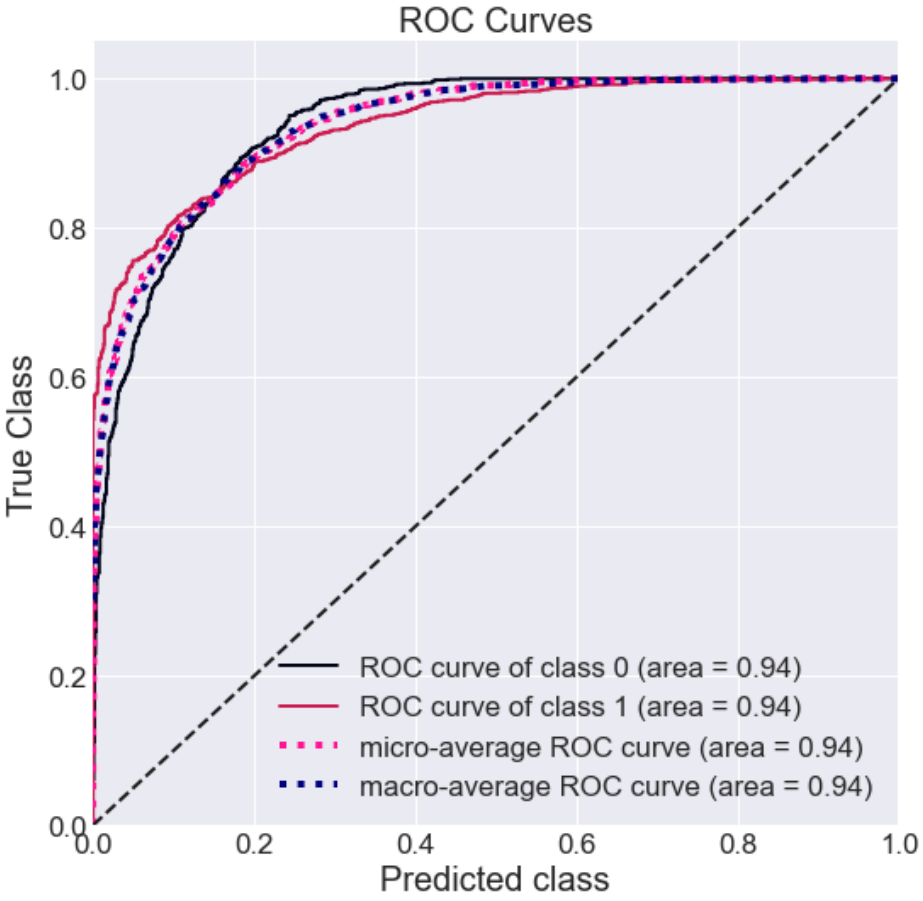

Figure 5.5: Linear Regression (LR) Receiver Operating Curve ROC

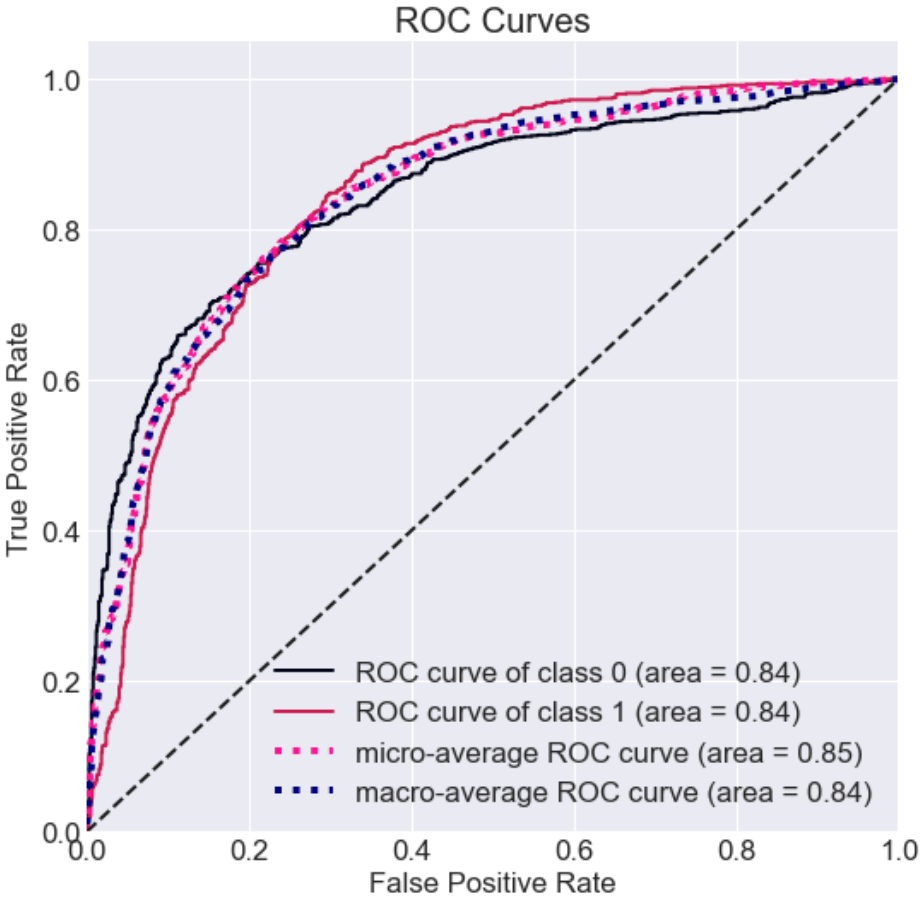

Figure 5.6: Support Vector Classifier (SVC) Receiver Operating Curve

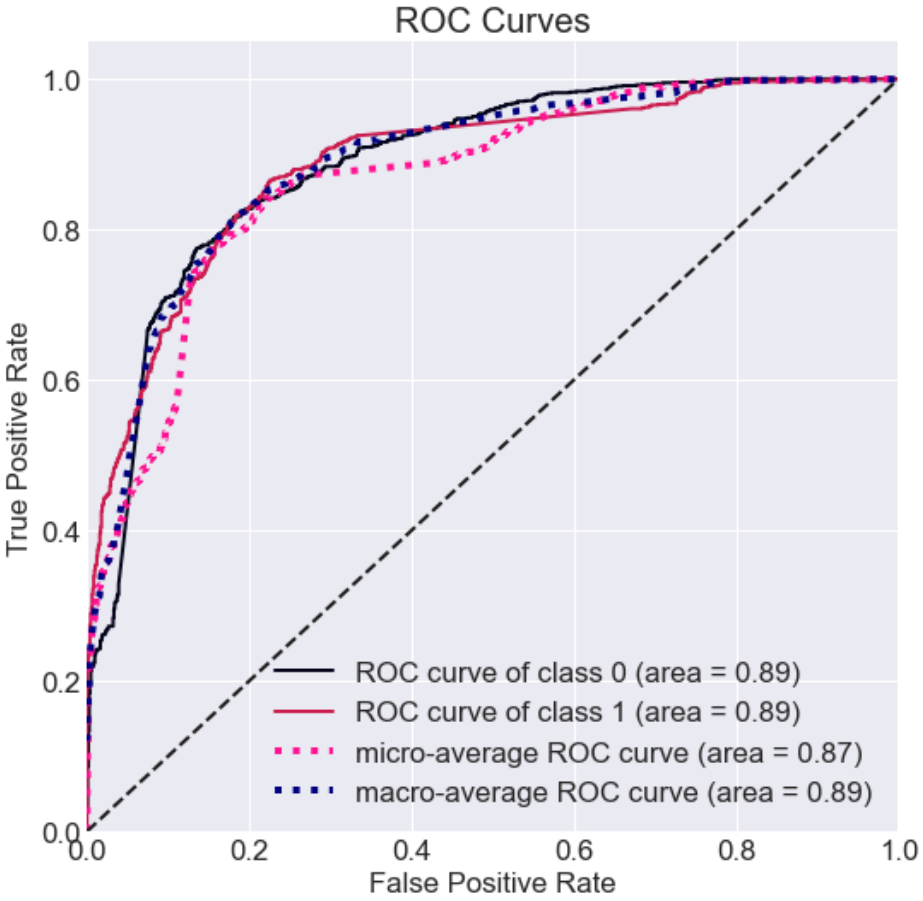

Figure 5.7: Artificial Neural network (ANN) Receiver Operating Curve

5.1.2 Model Performance in the presence of FSGM adversaries

This section presents results obtained by evaluating the performance of the customer churn prediction classifiers using perturbed adversarial test set generated using FGSM at degrees of perturbation (epsilon) between 0 and 1.0.

ϵ	Accuracy			Precision			F-1 Score			Recall		
	LR	SVM	ANN	LR	SVM	ANN	LR	SVM	ANN	LR	SVM	ANN
0	84.64	77.25	80.39	83.52	73.16	75.76	84.58	78.58	81.60	85.66	84.87	88.41
0.1	42.22	77.10	69.08	42.76	72.97	65.96	46.80	78.47	70.94	51.67	84.87	76.72
0.2	18.41	77.05	51.45	19.19	72.91	50.59	19.84	78.44	52.39	20.53	84.87	54.32
0.3	15.46	76.91	35.41	14.40	72.80	34.62	14.47	78.29	34.94	14.54	84.68	35.27
0.4	15.36	76.62	23.24	14.23	72.63	21.98	14.29	77.98	21.99	14.34	84.18	22.00
0.5	15.36	76.47	16.09	14.23	72.52	14.86	14.29	77.83	14.89	14.34	83.99	14.93
0.6	15.36	76.28	15.41	14.23	72.39	14.24	14.29	77.63	14.29	14.34	83.69	14.34
0.7	15.36	76.09	15.36	14.23	72.18	14.23	14.29	77.47	14.29	14.34	83.60	14.34
0.8	15.36	75.89	15.36	14.23	72.01	14.23	14.29	77.29	14.29	14.34	83.40	14.34
0.9	15.36	75.70	15.36	14.23	71.88	14.23	14.29	77.08	14.29	14.34	83.10	14.34
1	15.36	75.51	15.36	14.23	71.71	14.23	14.29	76.90	14.29	14.34	82.91	14.34

Table 5.2: Model performance at different perturbation rates.

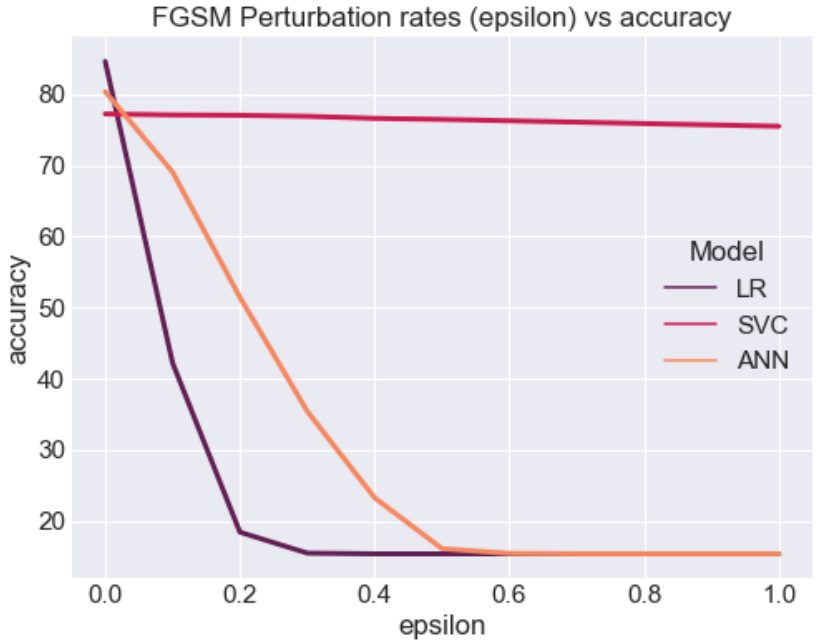

Figure 5.8: Plot of FGSM Perturbation rates against accuracy

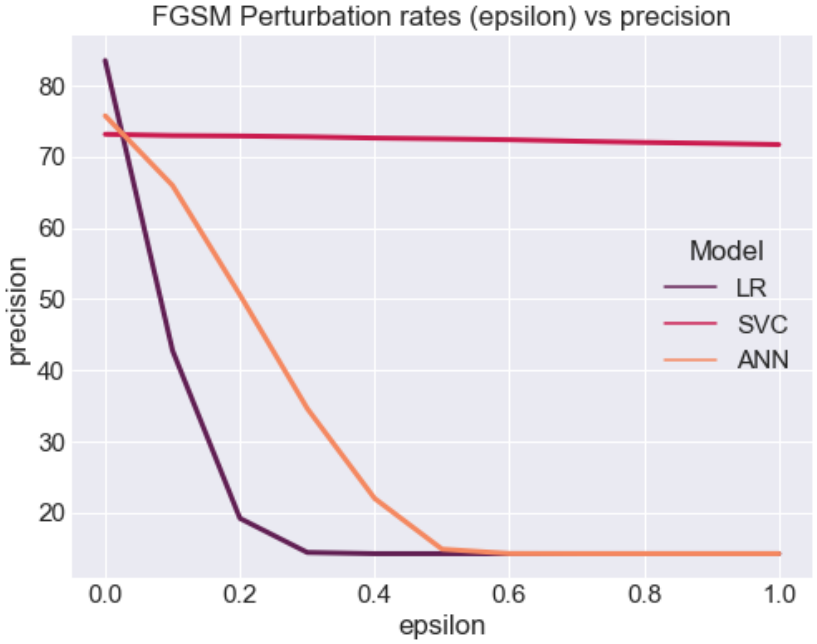

Figure 5.9: Plot of FGSM Perturbation rates against precision

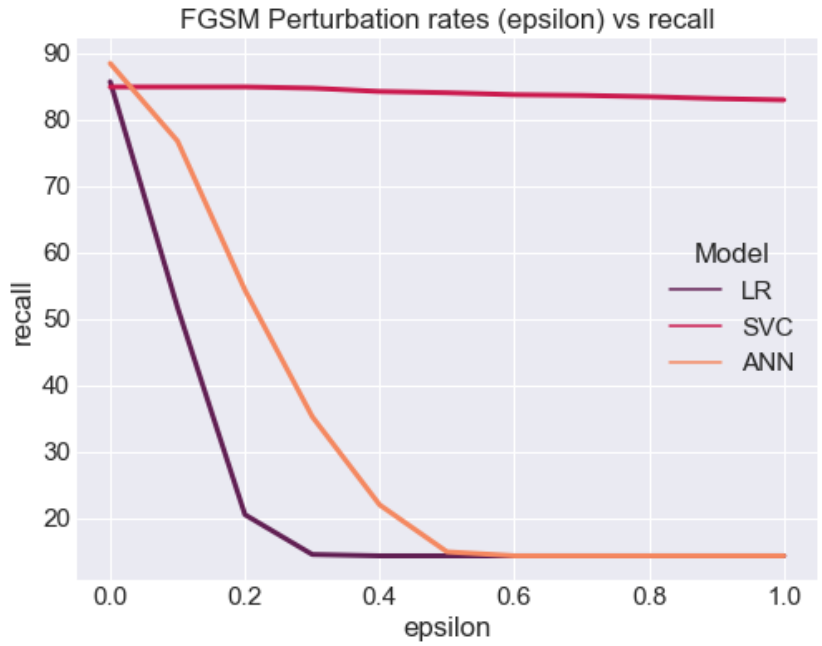

Figure 5.10: Plot of FGSM Perturbation rates against recall

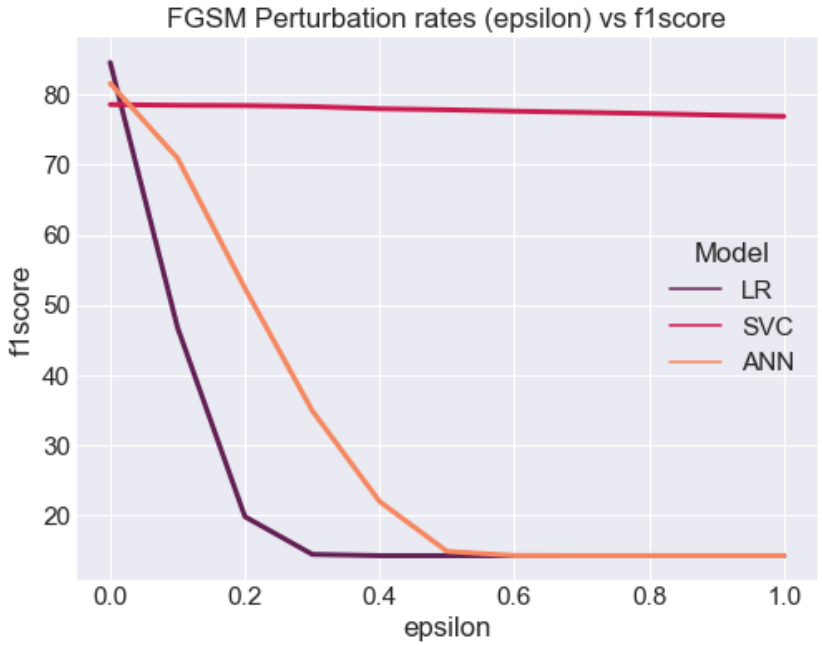

Figure 5.11: Plot of FGSM Perturbation rates against F-1 score

5.1.3 Model Performance in the presence of Deepfool adversaries

This section presents results obtained by evaluating the performance of the customer churn prediction classifiers using perturbed adversarial test set generated using Deep Fool attack at overshoot values (ϵ between 0 and 1×10^{-5}).

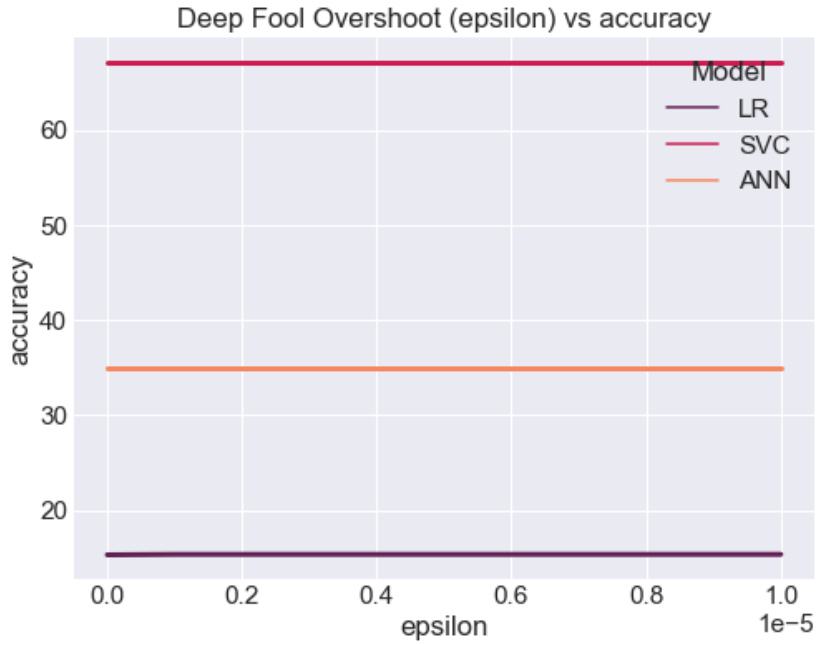

Figure 5.12: Plot of Deep Fool Overshoot rates against accuracy

ϵ	Accuracy			Precision			F-1 Score			Recall		
	LR	SVC	ANN	LR	SVC	ANN	LR	SVC	ANN	LR	SVC	ANN
0	15.31	67.10	34.88	14.22	67.21	35.68	14.28	65.90	37.88	14.34	64.64	40.37
1.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
2.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
3.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
4.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
5.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
6.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
7.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
8.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
9.00E-06	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37
1.00E-05	15.36	67.10	34.88	14.23	67.21	35.68	14.29	65.90	37.88	14.34	64.64	40.37

Table 5.3: Model performance at different overshoot levels.

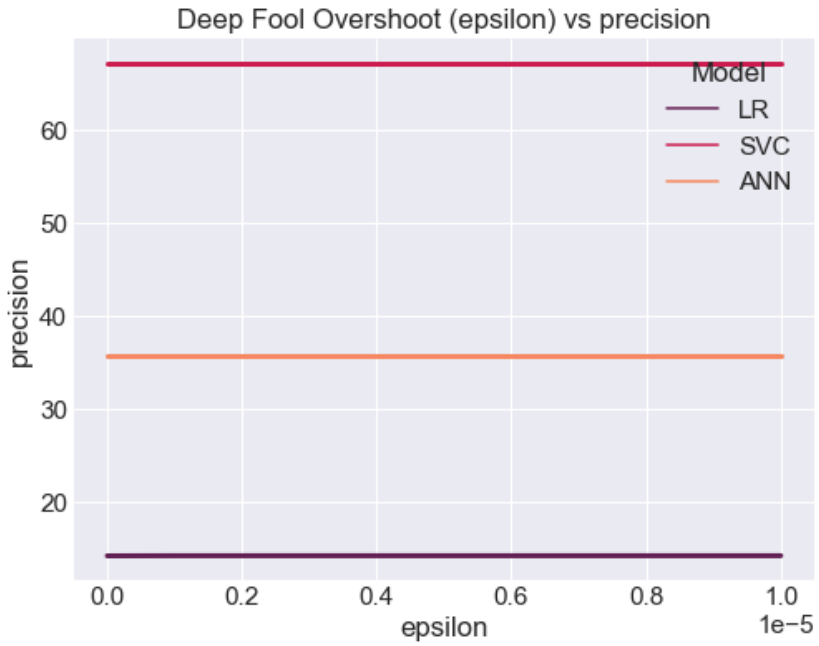

Figure 5.13: Plot of Deep Fool Overshoot rates against precision

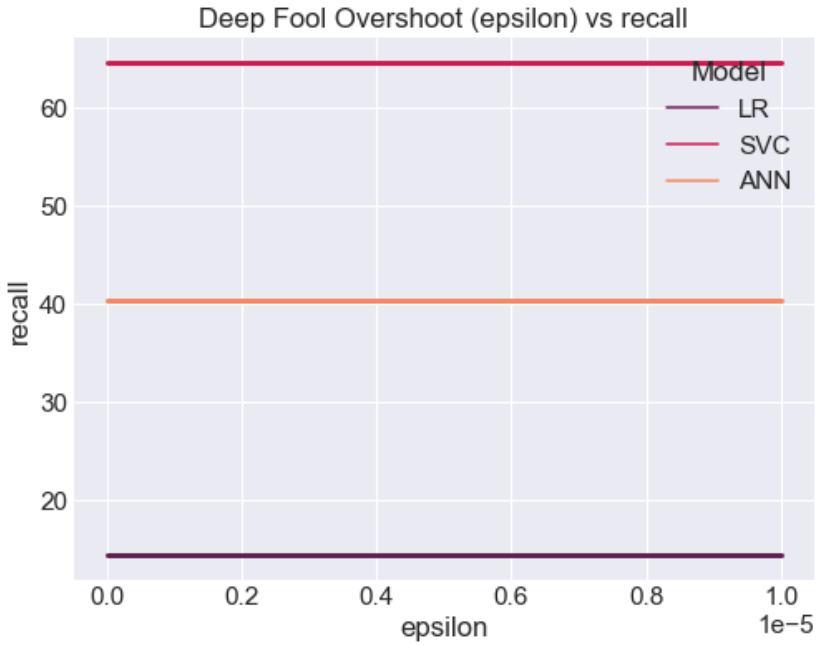

Figure 5.14: Plot of Deep Fool Overshoot rates against recall

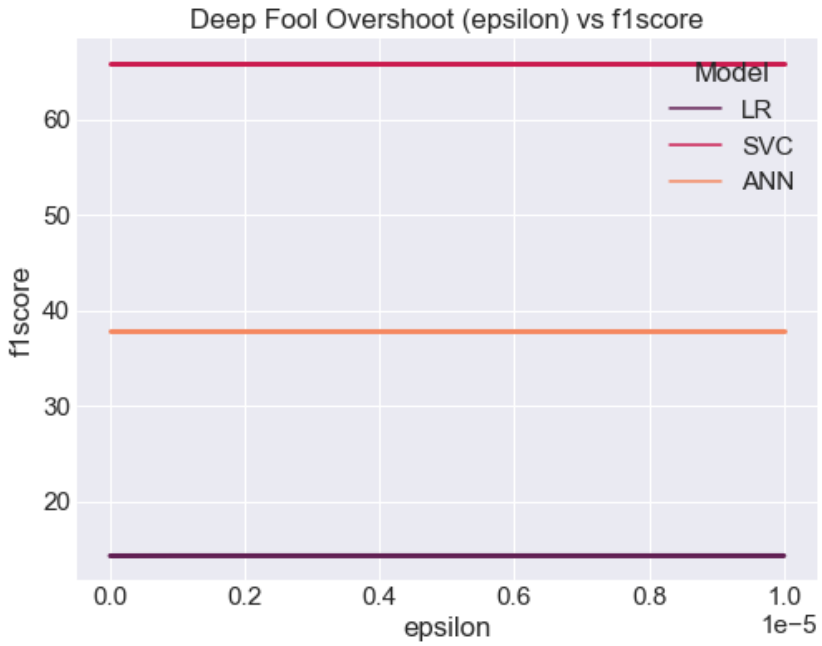

Figure 5.15: Plot of Deep Fool Overshoot rates against F-1 score

5.1.4 Model Performance in the presence of Jacobian adversaries

This section presents results obtained by evaluating the performance of the customer churn prediction classifiers using perturbed adversarial test set generated using JSMA attack at perturbation rates (θ between 0 and 1).

θ	Accuracy			Precision			F-1 Score			Recall		
	LR	SVM	ANN	LR	SVM	ANN	LR	SVM	ANN	LR	SVM	ANN
0.1	84.64	77.25	80.63	86.61	73.16	86.86	83.89	78.58	79.28	81.34	84.87	72.91
0.2	82.32	77.25	80.10	86.22	73.16	85.56	80.92	78.58	78.89	76.23	84.87	73.19
0.3	75.31	77.25	79.81	80.29	73.16	84.23	72.45	78.58	78.87	66.01	84.87	74.14
0.4	71.06	77.25	79.23	74.56	73.16	82.88	67.99	78.58	78.48	62.48	84.87	74.52
0.5	68.60	77.25	78.41	70.35	73.16	80.96	66.18	78.58	77.97	62.48	84.87	75.19
0.6	64.59	77.25	77.73	65.37	73.16	79.34	62.31	78.58	77.61	59.53	84.87	75.95
0.7	60.72	77.25	76.33	60.76	73.16	76.76	58.75	78.58	76.69	56.88	84.87	76.62
0.8	56.96	77.25	75.80	56.55	73.16	75.53	55.16	78.58	76.49	53.83	84.87	77.47
0.9	53.24	77.29	74.35	52.56	73.22	73.28	51.50	78.62	75.54	50.49	84.87	77.95

Table 5.4: Model performance at different FGSM perturbation rates.

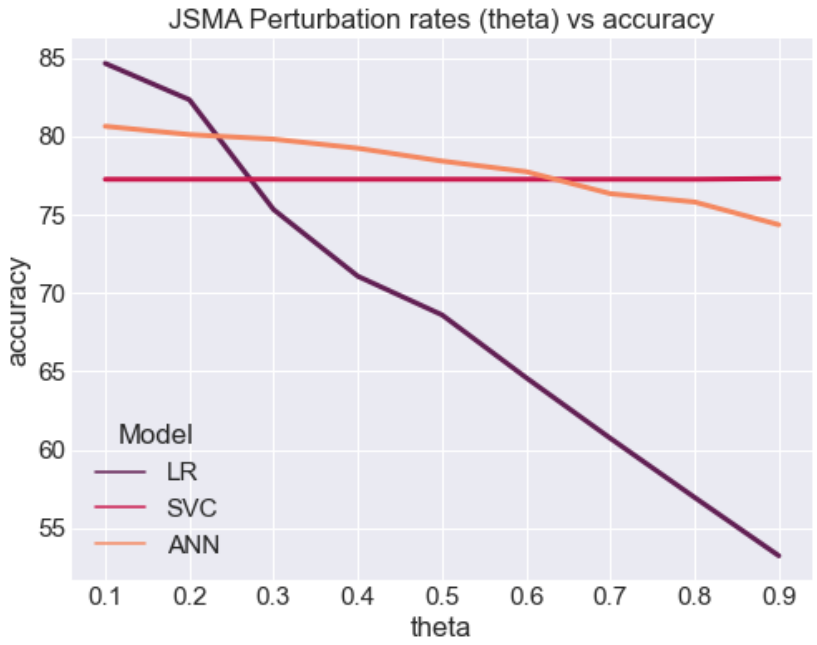

Figure 5.16: Plot of JSMA perturbation rates against accuracy

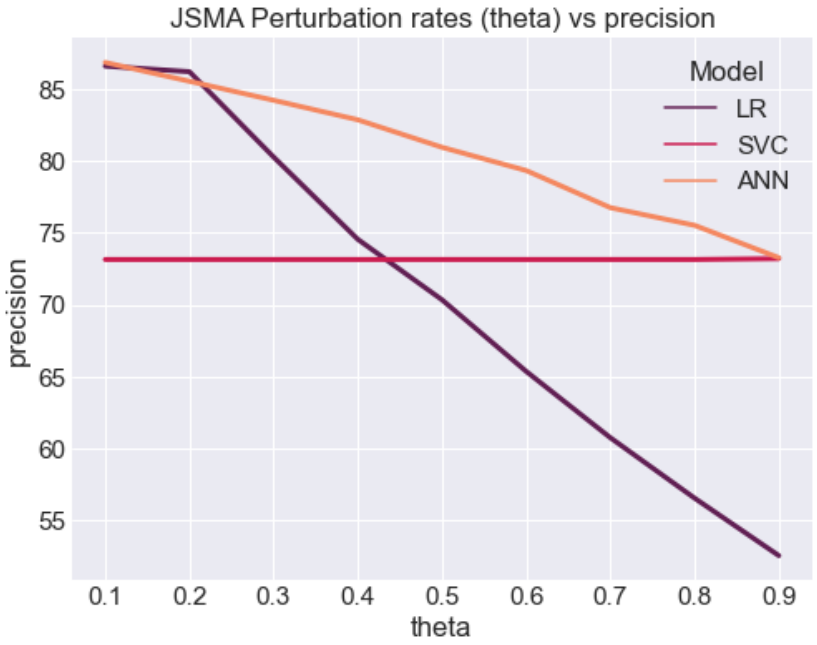

Figure 5.17: Plot of JSMA perturbation against precision

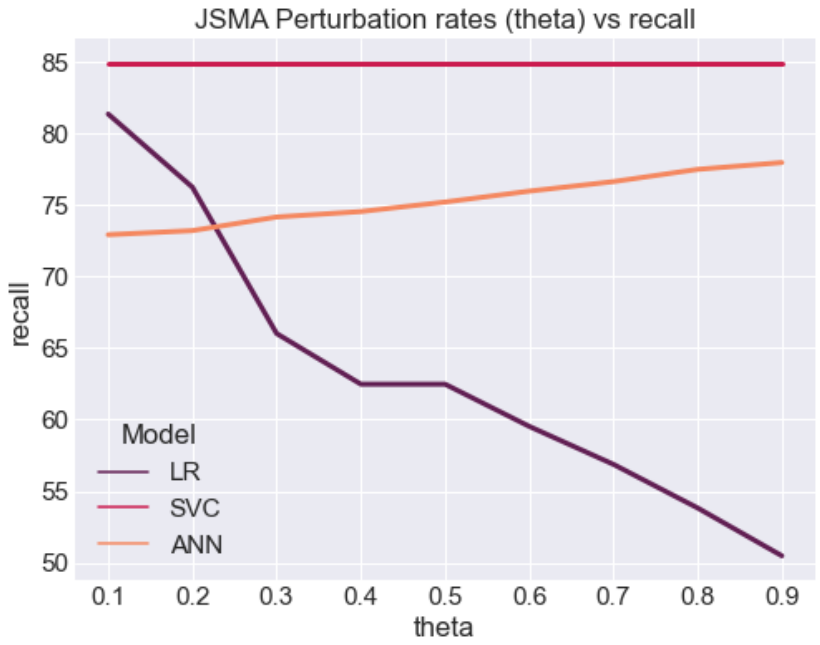

Figure 5.18: Plot of JSMA perturbation rates against recall

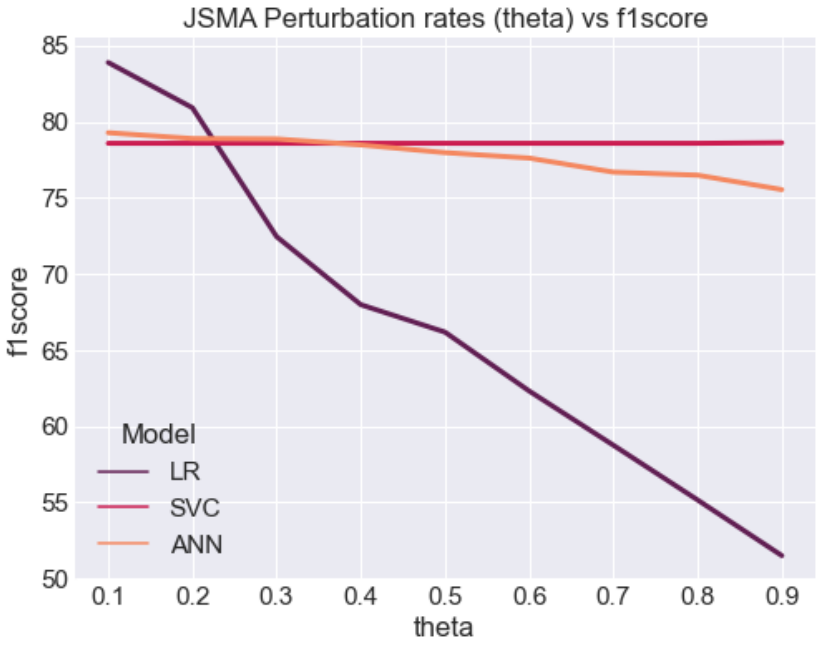

Figure 5.19: Plot of JSMA perturbation rates against F-1 score

5.2 Discussion

5.2.1 Baseline model with no adversaries

An observation of the results from evaluating the classifiers using unperturbed test set in Table 5.1 shows that although simple, LR classifier can be seen to outperform more complex algorithms, SVC, and ANN. With an accuracy of 85%, precision of 84%, a F-1 score of 85% and recall of 86%, the LR classifier performs best for predicting customer churn based on these metrics. LR also outperforms other models based on AUC metric with an AUC of 94% as seen in figure. 5.5 and identifies 880 churner, a value more than any other classifier as seen in the confusion matrix heat maps. This could be attributed to the fact that logistic regression performs better that other models in highly co linear data sets. The artificial neural network classifier comes next, outperforming and SVC on all evaluation metrics With an accuracy of 80%, precision of 76%, a F-1 score of 82% and recall of 88%, . On all metrics, support vector classifier performs the least with an accuracy, precision, F-1 score, recall and AUC of 77%, 73%, 79%, 85 % and 76% respectively.

5.2.2 Model Performance in the presence of FSGM adversaries

In the presence of FSGM adversaries, evaluation results of all three classifiers show that the LR is the most vulnerable to FGSM attacks. The accuracy is observed to drop rapidly by 70% as the degree of perturbation is increased to 0.4. Similar drops in performance is seen with precision, f1-score and recall metrics signifying high vulnerability and low robustness of the model to FGSM attacks. The next most vulnerable classifier is the ANN classifier. Although more robust than the logistic regression classifier, the performance of the ANN classifier is seen to diminish in the presence of FGSM adversaries as the degree of perturbation increases. The accuracy of the ANN classifier is observed to drop by about 70% at a perturbation degree of 0.6. What this implies is that the ANN classifier is as vulnerable as LR models to FSGM attacks in perturbation rates of 0.6 and above. Similar fall in ANN classifier performance is seen with precision, f1-score and recall metrics. The support vector classifier outperforms other classifiers in the presence of adversaries as negligible change is seen. This suggests that it is robust. There is but very negligible change in accuracy, precision, F-1 score and recall as the degree of perturbation is increased.

5.2.3 Model Performance in the presence of Deepfool adversaries at different overshoot values

On the basis of the working principle of the deep fool attack, it is seen to have an instant effect on the performance of all the classifiers after which an increase in the overshoot value has little to no effect on the performance of classifiers. A look at figure 5.12, 5.13, 5.14 and 5.15 shows that deep fool attack has a significant but invariable impact on the performance of a model. The most vulnerable classifier to the deep fool attack is the logistic regression classifier with a 70% drop in accuracy, precision, F-1 score and recall. The next most vulnerable to the deep fool attack is the support vector machine classifier with a 39% drop in accuracy, 34% drop in precision, 38% drop in F-1 score and 42% drop in recall. SVC experienced the least degradation in performance as a 10% drop in accuracy, 11% drop in precision, 9% drop in F-1 score and 12% drop in recall was experienced. The implication of this result is that SVC models are more suitable in deep fool adversarial prone environments.

5.2.4 Model Performance in the presence of JSMA adversaries

The effect of JSMA adversaries can be observed from Table 5.4 to have the least effect on models. This may be attributed to the fact that only 5% of the data was perturbed. Like with FGSM, model performance was seen to degrade with increasing perturbations however, SVC model performance was only affected negligibly. This also goes to show that SVC models are robust in adversarial environments.

Chapter 6

Conclusions

In conclusion, customer churn prediction is very important to companies because it helps them identify customers with the likelihood to churn after which they can be targeted with retention campaigns. In this study, customer churn prediction classifiers were created using classification algorithms such as logistic regression, support vector machines and artificial neural networks. Unlike prior studies done on customer churn prediction, this study took it a step further by critically examining how FGSM, deep fool and adversarial attacks impact the behaviour and performance of the classifiers.

6.1 Review of Research Questions in Relation to Results

After an extensive review of relevant literature in the following research questions were posed:

1. How robust are customer churn prediction classifiers to adversarial attacks?

2. Which adversarial attack has the highest impact on the models?

Answer to Q1. Judging from the results obtained from evaluating the performance of all the classifiers as well as the interpretation and discussion presented with regards to the results, it was deduced that logistic regression is the least robust to all attack forms. Artificial neural network, although more robust, was also identified to be vulnerable to adversarial attack. Support vector classifier on the other hand exhibited the least vulnerability to all three adversarial attacks and this is consistent with results obtained form (Yuan et al., 2018; Zhang et al., 2016; Rigaki, 2017).

Answer to Q2. With regards to the attack with the most impact on customer churn prediction classifiers, it was found that for the LR classifier, FSGM and deepfool were found to have the highest impact being able to reduce the model's performance by 70%. JSMA had the least impact on performance, and this could be attributed to the fact that only 1% of the data was perturbed. For support vector classifiers, it was found that the highest degradation on its performance was achieved in the presence of deep fool attack, reducing the accuracy by 10%. JSMA and FGSM had negligible effect on the SVC classifier. For the ANN classifier, FGSM caused the most degradation, reducing its accuracy by as much as 70%.

When tested on clean data, SVC did not perform as well as other models however it exhibits robustness for adversaries and for this reason is the recommended choice when deploying customer churn prediction classifiers in potentially adversarial environments.

6.2 Limitations and Future Work

This research work presented a first attempt to examine the impact of adversaries on customer churn prediction classifiers however, while this study considers three classification algorithms, LR, SVC and ANN, a future world would be to consider other classification algorithms such as decision trees, K-Nearest neighbour, etc. It will also be needful to examine how ensembling and model hybridization may minimise or enhance model vulnerability to adversarial attacks.

Another extension of this research would be to examine how other forms of attack not explored in this study affect classifiers. This could include poisoning attacks such as random and target flipping as well as other evasion attacks.

The data used for this experiment is unique to the telecommunication industry and the company from which it was collated. As a result, all results obtained from this study may be unique to the dataset used and hence may not generally apply to other case studies. This limitation implies that building customer churn prediction classifiers using datasets from other companies and industries will help provide a more generalised understanding of how the classifiers are impacted by the attacks.

Finally, this study provides us with an opportunity to understand how the classifier are impacted by the attacks but not how they can be protected from them. In future, it will be good to experimentally examine possible defences and establish whether they can be used or not.

6.3 Personal Relection

When the project began, I initially wanted to work on the impact of adversarial machine learning attacks on insider threat detection models however, plans were changed as the available data sets were very large and difficult to manoeuvre. My interest in business inspired me to transfer this concept to the business world and explore the impact of adversaries on customer churn prediction models.

The journey has been rewarding as it provided me with a great opportunity to learn and stretch the boundaries of my existing knowledge. I improved in theoretical as well as practical skills such as the use of python and python libraries which and my ability to critically analyse literature. The 15000-word dissertation also helped to shape my academic writing skills. I improved soft skills such as my ability plan and strategies using tools like Gantt charts and project diaries. I also learned to solve problems as I severally encountered problems I had to solve such as code errors. I improved my critical thinking skills as well and improved my ability to communicate due to the fact that the project required constant communication and feedback with the supervisor. my ability to learn also improved greatly because I had to get accustomed to many new concepts.

Chapter 7

References

References

Amin, A., Al-Obeidat, F., Shah, B., Adnan, A., Loo, J., & Anwar, S. (2019, January). Customer churn prediction in telecommunication industry using data certainty. *Journal of Business Research*, *94*, 290–301. Retrieved 2021-06-17, from https://linkinghub.elsevier.com/retrieve/pii/S0148296318301231 doi: 10.1016/j.jbusres.2018.03.003

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Alawfi, K., Hussain, A., & Huang, K. (2017, May). Customer churn prediction in the telecommunication sector using a rough set approach. *Neurocomputing*, *237*, 242–254. Retrieved 2021-06-17, from https://www.sciencedirect.com/science/article/pii/S0925231216314849 doi: 10.1016/j.neucom.2016.12.009

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., ... Hussain, A. (2016). Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study. *IEEE Access*, *4*, 7940–7957. (Conference Name: IEEE Access) doi: 10.1109/ACCESS.2016.2619719

Ballings, M., & Van den Poel, D. (2012, December). Customer event history for churn prediction: How long is long enough? *Expert Systems with Applications*, *39*(18), 13517–13522. Retrieved 2021-07-08, from https://linkinghub.elsevier.com/retrieve/pii/S0957417412008615 doi: 10.1016/j.eswa.2012.07.006

Baveja, S. S. (n.d.). The Value of Online Customer Loyalty and How You Can Capture It. , 10.

Brandusoiu, I., & Toderean, G. (2013). *Churn Prediction in the tecommunications Sector Using Support Vector Machines.*

Burez, J., & Van den Poel, D. (2009, April). Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, *36*(3), 4626–4636. Retrieved 2021-06-17, from https://linkinghub.elsevier.com/retrieve/pii/S0957417408002121 doi: 10.1016/j.eswa.2008.05.027

CallMiner. (2020). *2020 us callminer churn index infographic.* Retrieved from https://callminer.com/blog/2020-us-callminer-churn-index

Chen, Z.-Y., Shu, P., & Sun, M. (2012, December). A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data. *European Journal of Operational Research*, *223*, 461–472. doi: 10.1016/j.ejor.2012.06.040

Coussement, K., Benoit, D. F., & Van den Poel, D. (2010, March). Improved marketing decision making in a customer churn prediction context using generalized additive models. *Expert Systems with Applications*, *37*(3), 2132–2143. Retrieved 2021-07-08, from https://www.sciencedirect.com/science/article/pii/S0957417409007325 doi: 10.1016/j.eswa.2009.07.029

Coussement, K., & De Bock, K. W. (2013, September). Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning. *Journal of Business Research*, *66*(9), 1629–1636. Retrieved 2021-07-08, from https://linkinghub.elsevier.com/retrieve/pii/S0148296312003530 doi: 10.1016/j.jbusres.2012.12.008

De Bock, K. W., & Van den Poel, D. (2012, June). Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models. *Expert Systems with Applications*, *39*(8), 6816–6826. Retrieved 2021-07-06, from https://www.sciencedirect.com/science/article/pii/S0957417412000164 doi: 10.1016/j.eswa.2012.01.014

De Caigny, A., Coussement, K., & De Bock, K. W. (2018, September). A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *European Journal of Operational Research*, *269*(2), 760–772. Retrieved 2021-06-17, from https://linkinghub.elsevier.com/retrieve/pii/S0377221718301243 doi: 10.1016/j.ejor.2018.02.009

Deldjoo, Y., Di Noia, T., & Merra, F. A. (2020, November). A survey on Adversarial Recommender Systems: from Attack/Defense strategies to Generative Adversarial Networks. *arXiv:2005.10322 [cs]*. Retrieved 2021-07-06, from http://arxiv.org/abs/2005.10322 (arXiv:

2005.10322)

Ebrah, K., & Elnasir, S. (2019, November). Churn Prediction Using Machine Learning and Recommendations Plans for Telecoms. *Journal of Computer and Communications*, *7*(11), 33–53. Retrieved 2021-07-26, from http://www.scirp.org/Journal/Paperabs.aspx?paperid=96177 (Number: 11 Publisher: Scientific Research Publishing) doi: 10.4236/jcc.2019.711003

Farquad, M. A. H., Ravi, V., & Raju, S. B. (2014, June). Churn prediction using comprehensible support vector machine: An analytical CRM application. *Applied Soft Computing*, *19*, 31–40. Retrieved 2021-07-05, from https://www.sciencedirect.com/science/article/pii/S1568494614000507 doi: 10.1016/j.asoc.2014.01.031

Fei, J., Xia, Z., Yu, P., & Xiao, F. (2020, December). Adversarial attacks on fingerprint liveness detection. *EURASIP Journal on Image and Video Processing*, *2020*(1), 1. Retrieved 2021-08-04, from https://jivp-eurasipjournals.springeropen.com/articles/10.1186/s13640-020-0490-z doi: 10.1186/s13640-020-0490-z

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). *Explaining and harnessing adversarial examples.*

Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2016, June). Adversarial Perturbations Against Deep Neural Networks for Malware Classification. *arXiv:1606.04435 [cs]*. Retrieved 2021-06-01, from http://arxiv.org/abs/1606.04435 (arXiv: 1606.04435)

Hou, L., & Tang, X. (2010). Customer Churn Identifying Model Based on Dual Customer Value Gap. , 11.

Huang, B., Kechadi, M. T., & Buckley, B. (2012, January). Customer churn prediction in telecommunications. *Expert Systems with Applications*, *39*(1), 1414–1425. Retrieved 2021-06-17, from https://linkinghub.elsevier.com/retrieve/pii/S0957417411011353 doi: 10.1016/j.eswa.2011.08.024

Kantartopoulos, P., Pitropakis, N., Mylonas, A., & Kylilis, N. (2020, December). Exploring Adversarial Attacks and Defences for Fake Twitter Account Detection. *Technologies*, *8*(4), 64. Retrieved 2021-07-12, from https://www.mdpi.com/2227-7080/8/4/64 (Number: 4 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/technologies8040064

Kaur, S. (2017). Literature Review of Data Mining Techniques in Customer Churn Prediction for Telecommunications Industry. , *1*(2), 13.

Kurakin, A., Goodfellow, I., & Bengio, S. (2017, February). Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*. Re-

trieved 2021-07-12, from http://arxiv.org/abs/1607.02533 (arXiv: 1607.02533)

Latha, K. J., Baburao, M., & Kavitha, C. (2019, May). A Comparative study on Logit leaf model (LLM) and Support leaf model (SLM) for predicting the customer churn. *International Journal of Computer Sciences and Engineering*, *7*(5), 1628–1632. Retrieved 2021-07-12, from http://www.ijcseonline.org/full_paper_view.php?paper_id=4462 doi: 10.26438/ijcse/v7i5.16281632

Lazarov, V., München, T. U., Capota, M., & München, T. U. (2007). Churn Prediction. *Business Analytics Course. TUM Computer Science*.

Liu, Y., Chen, X., Liu, C., & Song, D. (2017, February). Delving into Transferable Adversarial Examples and Black-box Attacks. *arXiv:1611.02770 [cs]*. Retrieved 2021-05-25, from http://arxiv.org/abs/1611.02770 (arXiv: 1611.02770)

Lu, N., Lin, H., Lu, J., & Zhang, G. (2014, May). A Customer Churn Prediction Model in Telecom Industry Using Boosting. *IEEE Transactions on Industrial Informatics*, *10*(2), 1659–1665. (Conference Name: IEEE Transactions on Industrial Informatics) doi: 10.1109/TII.2012.2224355

Moeyersoms, J., & Martens, D. (2015, April). Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector. *Decision Support Systems*, *72*, 72–81. Retrieved 2021-07-08, from https://www.sciencedirect.com/science/article/pii/S0167923615000275 doi: 10.1016/j.dss.2015.02.007

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). *Deepfool: a simple and accurate method to fool deep neural networks.*

Mozer, M., Wolniewicz, R., Grimes, D., Johnson, E., & Kaushansky, H. (2000, May). Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. *IEEE Transactions on Neural Networks*, *11*(3), 690–696. (Conference Name: IEEE Transactions on Neural Networks) doi: 10.1109/72.846740

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., ... Edwards, B. (2019, November). Adversarial Robustness Toolbox v1.0.0. *arXiv:1807.01069 [cs, stat]*. Retrieved 2021-06-10, from http://arxiv.org/abs/1807.01069 (arXiv: 1807.01069)

Nitzan, I., & Libai, B. (2011, May). Social Effects on Customer Retention. *Journal of Marketing*, *75*. doi: 10.2307/41406857

Pacheco, Y., & Sun, W. (2021). Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets:. In *Proceedings of the 7th International Conference on Information Systems Security and Privacy* (pp. 160–171). Online Streaming, — Se-

lect a Country —: SCITEPRESS - Science and Technology Publications. Retrieved 2021-08-04, from https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010253501600171 doi: 10.5220/0010253501600171

Papadopoulos, P., Thornewill von Essen, O., Pitropakis, N., Chrysoulas, C., Mylonas, A., & Buchanan, W. J. (2021, June). Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT. *Journal of Cybersecurity and Privacy*, *1*(2), 252–273. Retrieved 2021-06-06, from https://www.mdpi.com/2624-800X/1/2/14 (Number: 2 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/jcp1020014

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2015, November). The Limitations of Deep Learning in Adversarial Settings. *arXiv:1511.07528 [cs, stat]*. Retrieved 2021-07-12, from http://arxiv.org/abs/1511.07528 (arXiv: 1511.07528)

Rigaki, M. (2017). *Adversarial Deep Learning Against Intrusion Detection Classifiers*. Retrieved 2021-07-12, from http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-64577

Shaaban, E., Helmy, Y., Khedr, A., & Nasr, M. (2012, January). A Proposed Churn Prediction Model. *International Journal of Engineering Research and Applications (IJERA*, *2*, 693–697.

Sharma, A., & Kumar Panigrahi, P. (2011, August). A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services. *International Journal of Computer Applications*, *27*(11), 26–31. Retrieved 2021-07-03, from http://www.ijcaonline.org/volume27/number11/pxc3874605.pdf doi: 10.5120/3344-4605

Taheri, R., Javidan, R., Shojafar, M., Pooranian, Z., Miri, A., & Conti, M. (2020, September). On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications*, *32*(18), 14781–14800. Retrieved 2021-06-08, from https://doi.org/10.1007/s00521-020-04831-9 doi: 10.1007/s00521-020-04831-9

Thoumy, M., & Abdallah, E. (2017, September). Switching costs impact on customer retention in telecommunication: An exploratory study in Lebanon. *Competition and Regulation in Network Industries*, *18*(3-4), 198–216. Retrieved 2021-07-06, from https://doi.org/10.1177/1783591718782307 (Publisher: SAGE Publications Ltd STM) doi: 10.1177/1783591718782307

Torkzadeh, G., Chang, J. C.-J., & Hansen, G. W. (2006, November). Identifying issues in customer relationship management at Merck-Medco. *Decision Support Systems*, *42*(2), 1116–1130. Retrieved 2021-06-17, from https://www.sciencedirect.com/science/article/pii/

S0167923605001491 doi: 10.1016/j.dss.2005.10.003

Vafeiadis, T., Diamantaras, K., Sarigiannidis, G., & Chatzisavvas, K. (2015, June). A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, *55*, 1–9. Retrieved 2021-06-17, from https://linkinghub.elsevier.com/retrieve/pii/S1569190X15000386 doi: 10.1016/j.simpat.2015.03.003

Verbeke, W., Dejaeger, K., Martens, D., Hur, J., & Baesens, B. (2012, April). New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European Journal of Operational Research*, *218*, 211–229. doi: 10.1016/j.ejor.2011.09.031

V J, A. (2019, June). An Effective Classifier for Predicting Churn in Telecommunication. , *Vol. 11*, 2019.

Wai-Ho Au, Chan, K., & Xin Yao. (2003, December). A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Transactions on Evolutionary Computation*, *7*(6), 532–545. Retrieved 2021-07-05, from http://ieeexplore.ieee.org/document/1255389/ doi: 10.1109/TEVC.2003.819264

Warzynski, A., & Kolaczek, G. (2018). Intrusion detection systems vulnerability on adversarial examples. *2018 Innovations in Intelligent Systems and Applications (INISTA)*. doi: 10.1109/INISTA.2018.8466271

Yuan, X., He, P., Zhu, Q., & Li, X. (2018, July). Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv:1712.07107 [cs, stat]*. Retrieved 2021-05-25, from http://arxiv.org/abs/1712.07107 (arXiv: 1712.07107)

Zhang, F., Chan, P. P. K., Biggio, B., Yeung, D. S., & Roli, F. (2016, March). Adversarial Feature Selection against Evasion Attacks. *IEEE Transactions on Cybernetics*, *46*(3), 766–777. Retrieved 2021-07-12, from http://arxiv.org/abs/2005.12154 (arXiv: 2005.12154) doi: 10.1109/TCYB.2015.2415032

Chapter 8

Appendices

8.1 Proposal

EDINBURGH NAPIER UNIVERSITY SCHOOL OF COMPUTING

MSc RESEARCH PROPOSAL

The process of completing and reviewing the contents of this form is intended ensure that the proposed project is viable. It is also intended to increase the chances of a good pass. Much of the material produced while completing this form may be reused in the dissertation itself.

1. **Student details**

First name	Ezra
Last (family) name	Abah
Edinburgh Napier matriculation number	40482302

2. **Details of your programme of study**

MSc Programme title	MSc. Computing with professional placement
Year that you started your diploma modules	2020

3. **Project outline details**

Please suggest a title for your proposed project. If you have worked with a supervisor on this proposal, please provide the name. You are strongly advised to work with a member of staff when putting your proposal together.

Title of the proposed project	Impact of Adversarial machine learning attacks on Customer churn Prediction Models
Is your project appropriate to your programme of study?	Yes
Name of supervisor	

4. **Brief description of the research area - background**

Please do not describe your project in this section. Instead, provide background information in the box below on the broad research area in which your project sits. You should write in narrative (not bullet points). The academic/theoretical basis of your description of the research area should be evident through the use of citations and references. Your description should be between half and one page in length.

Adversarial machine learning refers to a machine learning technique in which models are made to malfunction (misclassify or misestimate) by supplying deceptive input. It was first discovered in 2013 by Szegedy et al. In their research on the intriguing properties of neural networks, it was found that neural networks can be made to misclassify an image by applying a certain perturbation that is not easily noticeable. Since then, many studies have been carried out in the different machine learning and deep learning areas in which this can be applied.
In the early stages of this field, researchers focused on it's application in image recognition. One of these researches was an interesting study carried out by Kurakin et al. that showed how much adversarial input can threaten the integrity of a system as well as the impact of the threat it poses. In that study, it was proved that by manipulating training examples in a traffic sign recognition system,

it was possible to deceive autonomous vehicles and as a result, cause accidents. Impactful malfunctions were also seen to occur when adversarial machine learning was implemented in speech recognition, portfolio management and other AI areas.
The first time this was implemented in cybersecurity was by Grosse et al. In the research, it was observed that adversarial examples can be used to attack a neural network malware detection system. It also interestingly showed how using this method, hackers can conceal attempts to attacks. In another research by A. Warzyński and G. Kołaczek, a similar problem was found as adversarial examples were found to conceal real attacks from being detected by intrusion detection systems. Although adversarial examples have been largely implemented in different fields, they have never been implemented in customer churn prediction models, consequently, the impacts have never been studied. This is the gap that this research work intends to fill.

5. **Project outline for the work that you propose to complete**

Please complete the project outline in the box below. You should use the emboldened text as a framework. Your project outline should be between half and one page in length.

The idea for this research arose from:
I first heard about adversarial ML attacks during a conversation with Dr. Naghmeh after which I decided to educate myself further on the concept. On doing some research, I found the research area to be really interesting and the needful as adversarial ML attacks pose a serious threat to the integrity of smart systems. I explored the areas in which it had been implemented and researched spanning from object and speech recognition, portfolio management, cybersecurity. In the customer churn however, I found that no work has been done to examine the impact of adversarial examples on insider threat detection systems. This research idea arose from my knowledge of this gap and my desire to fill it.

The aims of the project are as follows:
- Generate adversarial examples from the customer churn dataset
- Using evaluation metrics, compare the performance of clean and attacked models

The main research questions that this work will address include:
1. How does the performance of insider threat detection systems change when adversarial examples are used for training?
2. Which attack has the most impact on customer churn prediction model?

The software development/design work/other deliverable of the project will be:
The Project will deliver
- An adversarial experiment

The project deliverable will be evaluated as follows:
The project deliverable will involve evaluation of the created models using model evaluation metrics such as accuracy, precision, recall and F-1 score.

The project will involve the following research/field work/experimentation/evaluation:
The project will involve research into machine learning, adversarial machine learning and cybersecurity especially as it pertains to insider threat detection. Models will be created experimentally and evaluated using evaluation metrics.

This work will require the use of specialist software:

This will require open source tools for data pre-processing, visualisation and modelling. Preferably python and python libraries

This work will require the use of specialist hardware:
It does not require specialist hardware

The project is being undertaken in collaboration with:
N/A

6. **References**

Please supply details of all the material that you have referenced in sections 4 and 5 above. You should include at least three references, and these should be to high quality sources such as refereed journal and conference papers, standards or white papers. Please ensure that you use a standardised referencing style for the presentation of your references, e.g. APA, as on myNapier > Your Studies > Improve your Academic & Study Skills > Referencing and Academic Integrity > Referencing Guidelines > SoC Referencing Guidelines.

A. Warzyński and G. Kołaczek, "Intrusion detection systems vulnerability on adversarial examples," 2018 Innovations in Intelligent Systems and Applications (INISTA), Thessaloniki, Greece, 2018, pp. 1-4, doi: 10.1109/INISTA.2018.8466271.

Carlini N., Mishra P., Vaidya T., Zhang Y., Sherr M., Shields C., Wagner D. and Zhou W., "Hidden voice commands." in USENIX Security Symposium, 2016

Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2016). Adversarial perturbations against deep neural networks for malware classification. arXiv preprint arXiv:1606.04435.

Kurakin, A., Goodfellow I. and Bengio, S. "Adversarial examples in the physical world", arXiv preprint arXiv:1607.02533, (2016).

Szegedy, Ch., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., Fergus, R. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013).

7. **Ethics**

If your research involves other people, privacy or controversial research there may be ethical issues to consider (please see the information on the module website). If the answer below is YES then you need to complete a research Ethics and Governance Approval form, available on the website: http://www.ethics.napier.ac.uk .

Does this project have any ethical or governance issues related to working with, studying or observing other people? (YES/NO)	No

8. **Confidentiality**

If your research is being done in conjunction with an outside firm or organisation, there may be issues of confidentiality or intellectual property.

Does this project have any issues of confidentiality or intellectual property? (YES/NO)	No

10. **Submitting your proposal**

1. Please save this file using your surname, e.g. macdonald_proposal.docx, and e-mail it to your supervisor, who will discuss it with you and suggest possible improvements.
2. When your supervisor is content with your proposal, submit it to the Research Proposal Upload link on Moodle, and email your internal examiner to notify them that you have submitted. They will leave feedback for you on Moodle.
3. Discuss your feedback from the internal examiner with your supervisor and if necessary make final changes to your proposal.
4. When you produce your dissertation, add your finalised proposal as an appendix.

8.2 Gantt Chart

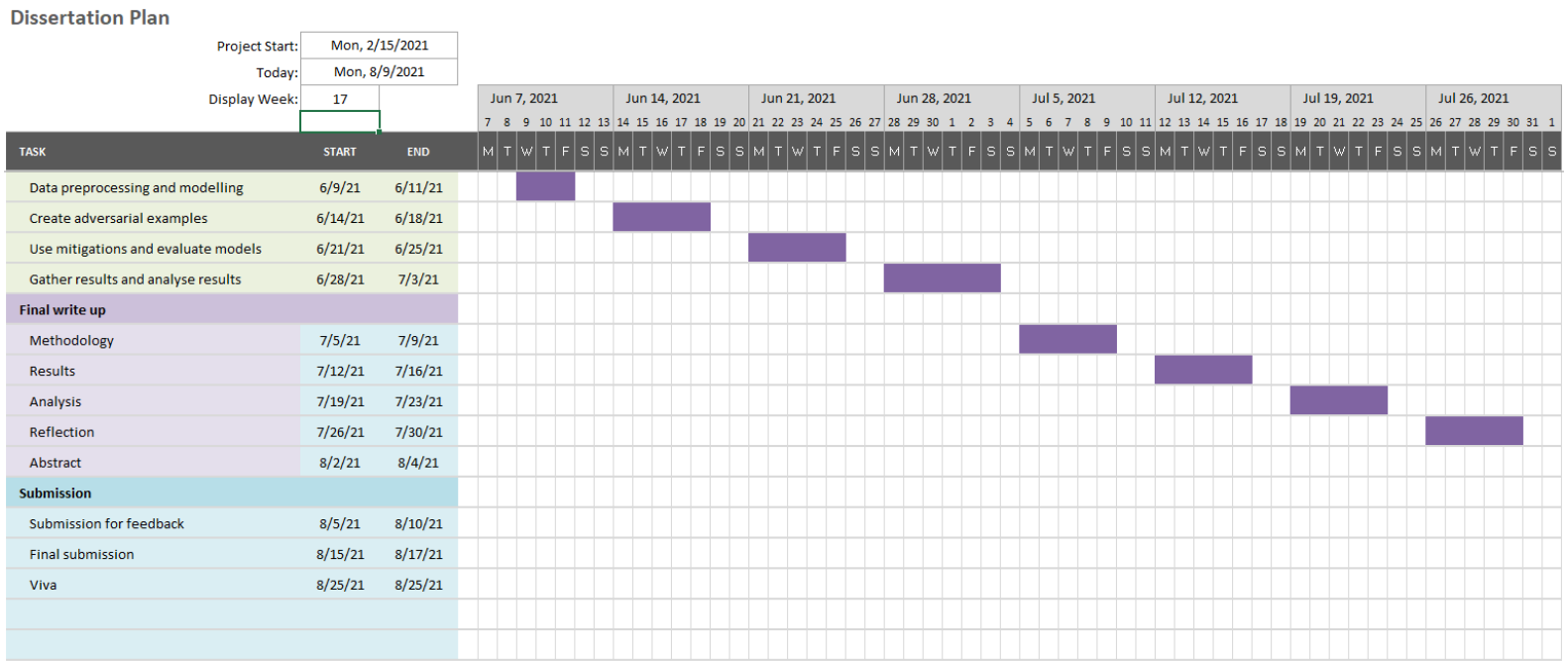

Figure 8.1: Gantt Chart part 1

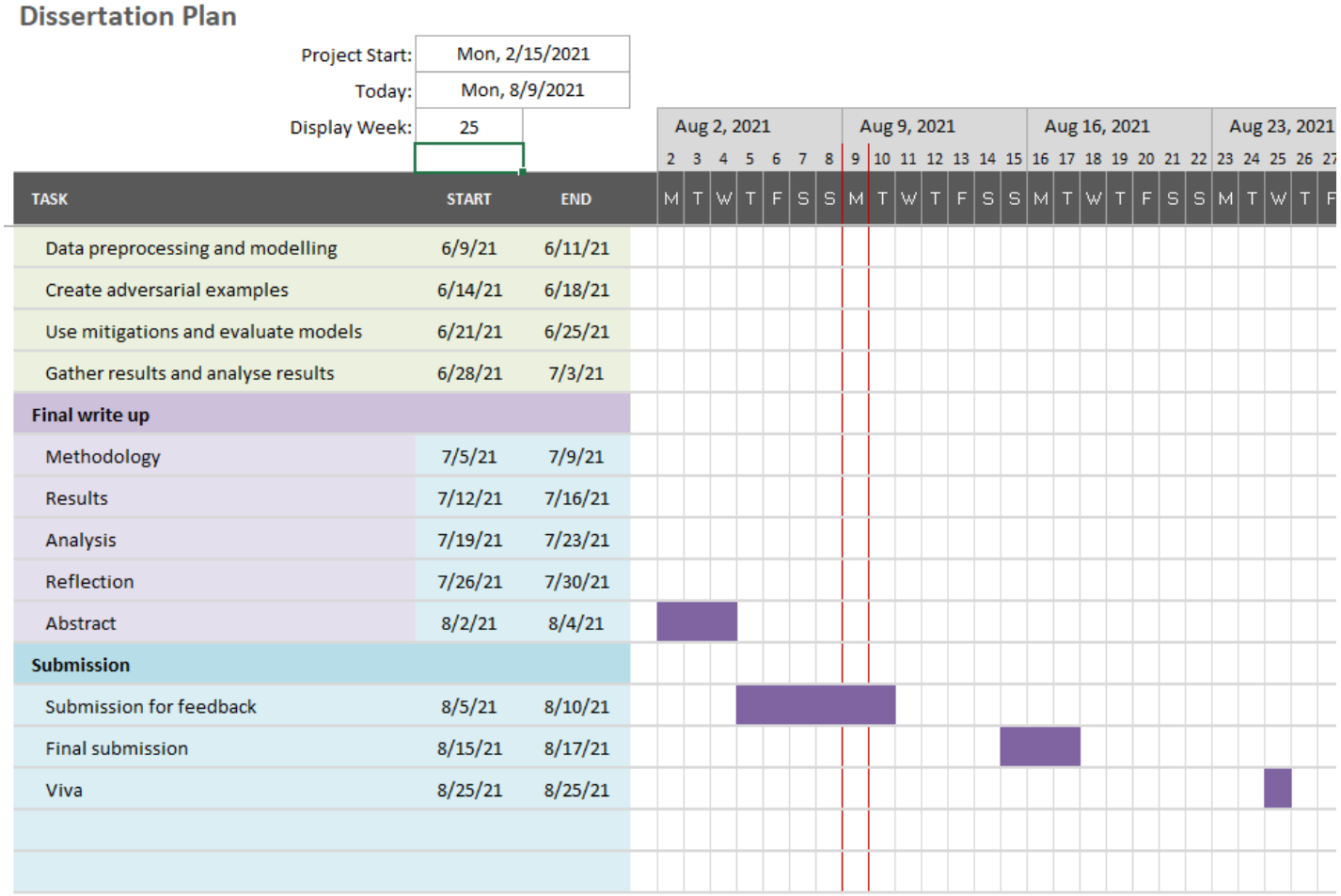

Figure 8.2: Gantt Chart part 2

8.3 Project Diary

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 24/June/2021 **Last diary date: -**

Objectives:

Get customer churn data set Clean and pre-process data Write introduction

Progress:

Data set gotten Data cleaned Error to fix with respect to normalisation

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 02/July/2021 **Last diary date: 24/June/2021**

Objectives:

Fix Normalization bug Complete modelling Update literature review on pre-processing, modelling and attacks Search for library to use for generating adversarial examples later on

Progress:

Fixed normalisation bug Completed model development Updated literature review Found cleverhans library for adversaries

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 08/July/2021 **Last diary date: 02/July/2021**

Objectives:

Fix adversary error or use another library
Update gantt chart and literature review
Update adversarial experiment python code

Progress:

Literature review almost completed
Completed Gantt chart and sent for review
Found IBM's adversarial robustness tool box
Error with generating adversaries using KNN and random forest

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 16/July/2021 **Last diary date: 08/July/2021**

Objectives:

Structure pages for methodology and implementation
Debug adversarial attack code
Set up and move code to google colab for speed
Start evaluation

Progress:

Methodology structure completed
Adversarial attack code debugged
Google colab set up and working
Evaluation code completed. Remaining to write loops to evaluate all models

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 23/July/2021 **Last diary date: 16/July/2021**

Objectives:

Write loops too evaluate all models and store results in dataframe Write methodology and implementation

Progress:

Evaluation loop completed Methodology draft completed (to be revised) Implementation draft completed (to be revised)

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 30/July/2021 **Last diary date: 23/July/2021**

Objectives:

Plot results Review methodology and implementation Send progress to supervisor for review

Progress:

Plots completed Review methodology and implementation Need to start result analysis and conclusion Present work sent to supervisor for revew

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 6/Aug/2021 **Last diary date: 30/July/2021**

Objectives:

Collate graphs into word and analyse them Present findings Draw conclusions Start gathering appendices materials into document (project diary, code, proposal

Progress:

All completed Problem with formatting encountered to be solved next week

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Ezra Abah **Supervisor: Andrew Partridge**

Date: 13/Aug/2021 **Last diary date: 6/Aug/2021**

Objectives:

Start using overleaf Transfer all documents into overleaf Review analysis and conclusion Update supervisor and ask for review Write recommendations for future works

Progress:

Work completes Work sent for final check

Supervisor's Comments:

8.4 Source Code

[ ]:
```
#Import needed libraries
import numpy as np
import pandas as pd
np.random.seed(5) # for reproducibility
from pandas import read_csv
from numpy import array
from numpy import argmax
from collections import Counter

#Libraries for preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

#Libraries for feature selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif, chi2
from sklearn.feature_selection import mutual_info_classif

#Libraries for modelling
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import tensorflow as tf
import keras
from keras.models import Sequential
from keras import layers
from keras.layers import Dense, Conv2D, Flatten
from keras.layers import Dense, Dropout, Activation
from art.estimators.classification import SklearnClassifier,KerasClassifier

# Libraries for metrics
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, accuracy_score, f1_score,␣
 ↪recall_score

# Libraries for attacks
from art.attacks.evasion import SaliencyMapMethod
from art.attacks.evasion import FastGradientMethod
from art.attacks.evasion import DeepFool
from art.estimators.classification import SklearnClassifier

# Libraries for visualisation
### pip install scikit-plot   ---> remove hash and run to install if uninstalled
import scikitplot as skplt
```

```
from matplotlib import pyplot
import seaborn as sns
import matplotlib.pyplot as plt
%pylab inline



import warnings
warnings.filterwarnings("ignore")

tf.compat.v1.disable_eager_execution()
tf.get_logger().setLevel('ERROR')
```

0.1 1.0 Dataset loading and exploration

[ ]:
```
data=pd.read_csv('Customer_Churn.csv')
print("Dataset: {}".format(data.shape))
print("Columns: {}".format(data.columns))
data.head(5)
```

[ ]:
```
# Check missing values
data.isnull().sum()
```

[ ]:
```
data.drop(['customerID'], axis=1, inplace=True)
```

[ ]:
```
columns_to_convert = ['Partner',
                      'Dependents',
                      'PhoneService',
                      'PaperlessBilling',
                      'Churn']

for item in columns_to_convert:
    data[item].replace(to_replace='Yes', value=1, inplace=True)
    data[item].replace(to_replace='No',  value=0, inplace=True)
data.head()

data['gender'].replace(to_replace='Female', value=1, inplace=True)
data['gender'].replace(to_replace='Male', value=1, inplace=True)
```

[ ]:
```
data['TotalCharges'] = data['TotalCharges'].replace(r'\s+', np.nan, regex=True)
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'])

df = pd.get_dummies(data)
df.fillna(value=0, inplace=True)
df.head()
```

[ ]:
```
target=df['Churn']
features= df.drop(columns=['Churn'])
```

[ ]:
```
from sklearn.preprocessing import MinMaxScaler
# define min max scaler
scaler = MinMaxScaler()

# transform data
df_scaled = scaler.fit_transform(df)
df_scaled.max()
```

[ ]:
```
from imblearn.over_sampling import SMOTE
```

[ ]:
```
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(features, target)
print('Resampled dataset shape {}'.format(Counter(y_res)))
```

[ ]:
```
def one_hot_encode(data):
    label_encoder = LabelEncoder()
    target_int = label_encoder.fit_transform(data)
    onehot_encoder = OneHotEncoder(sparse=False)
    target_int = target_int.reshape(len(target_int), 1)
    data_onehot = onehot_encoder.fit_transform(target_int)
    return data_onehot
```

[ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X_res, y_res,test_size=0.2,␣
 ↪random_state=9)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

[ ]:
```
y_train_onehot= one_hot_encode(y_train)
y_test_onehot= one_hot_encode(y_test)
```

1 Modelling

[ ]:
```
target_names= ['non-churn', 'churn' ]
```

1.0.1 LOGISTIC REGRESSION

[ ]:
```
### Logistic regression Model
lr_model_base=LogisticRegression(max_iter=10000).fit(X_train, y_train)
ypred = lr_model_base.predict(X_test)
yprob=lr_model_base.predict_proba(X_test)
print(classification_report(y_test, ypred, target_names=target_names))
```

[ ]:
```
sns.color_palette("rocket", as_cmap=True)
sns.set(font_scale=1.4)
plt.style.use('seaborn-darkgrid')
```

[ ]:
```
cf_matrix=confusion_matrix(y_test, ypred)
print(cf_matrix)

plt.figure(figsize = (8, 6))
sns.heatmap(cf_matrix, annot=True, cmap='rocket', fmt='g',␣
 ↪xticklabels=['churn','non-churn'], yticklabels=['churn' ,'non-churn'])
plt.title("Confusion matrix heatmap")
plt.xlabel("Predicted class", fontsize=20)
plt.ylabel("True Class", fontsize=20)
```

[ ]:
```
pylab.rcParams['figure.figsize'] = (8.5, 8.5)
skplt.metrics.plot_roc(y_test,yprob,cmap="rocket")
plt.xlabel("Predicted class", fontsize=20)
plt.ylabel("True Class", fontsize=20)
plt.show()
```

1.0.2 SUPPORT VECTOR CLASSIFIER

[ ]:
```
svc_model_base=SVC(probability=True,kernel='rbf',gamma=0.001, C=1).fit(X_train,␣
 ↪y_train)
ypred = svc_model_base.predict(X_test)
yprob=svc_model_base.predict_proba(X_test)
print(classification_report(y_test, ypred, target_names=target_names))
```

[ ]:
```
cf_matrix=confusion_matrix(y_test, ypred)
plt.figure(figsize = (8, 6))
sns.heatmap(cf_matrix, annot=True, cmap='rocket', fmt='g',␣
 ↪xticklabels=['churn','non-churn'], yticklabels=['churn' ,'non-churn'])
plt.title("Confusion matrix heatmap")
plt.xlabel("Predicted class", fontsize=20)
plt.ylabel("True Class", fontsize=20)
```

[ ]:
```
skplt.metrics.plot_roc(y_test,yprob, cmap="rocket")
plt.show()
```

1.0.3 ANN

[ ]:
```
PYTHONHASHSEED=0
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import RandomFourierFeatures
import random as python_random
```

```
python_random.seed(123)

tf.random.set_seed(1234)

ann_model_base = keras.Sequential(
    [
        keras.layers.Dense(100, input_shape=(X_train.shape[1],),␣
 ↪activation="relu"),
        keras.layers.Dense(100, activation="relu"),
        keras.layers.Dense(100, activation="relu",),
        keras.layers.Dense(2, activation="softmax"),
    ]
)

ann_model_base.compile(optimizer ='adam',
             loss='binary_crossentropy',
             metrics=['accuracy'])

ann_model_base.fit(X_train,y_train_onehot, epochs=30, verbose=False)
```

[ ]:
```
yprob=ann_model_base.predict(X_test)
ypred = np.argmax(yprob, axis=1)
y_test = np.argmax(y_test_onehot, axis=1)

print(classification_report(y_test, ypred))
```

[ ]:
```
cf_matrix=confusion_matrix(y_test, ypred)
plt.figure(figsize = (8, 6))
sns.heatmap(cf_matrix, annot=True, cmap='rocket', fmt='g',␣
 ↪xticklabels=['churn','non-churn'], yticklabels=['churn','non-churn'])
plt.title("Confusion matrix heatmap")
plt.xlabel("Predicted class", fontsize=15)
plt.ylabel("True Class", fontsize=15)
```

[ ]:
```
skplt.metrics.plot_roc(y_test,yprob,cmap="rocket")
plt.show()
```

2 ATTACKS

2.1 FSGM

[ ]:
```
results=[]
```

[ ]:
```
def fgsm_data(X_test,eps):
    lr_model=SklearnClassifier(model=lr_model_base)
```

```
    attack = FastGradientMethod(estimator=lr_model, eps=eps)
    new_x=attack.generate(x=X_test)
    return new_x
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = model.predict(X_test)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred,pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred,pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred,pos_label=1 )*100)
    return {'attack':"FSGM",'Model':"LR", 'epsilon' : i  ,'accuracy': accuracy,␣
 ↪'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
epsilon=[0,0.1, 0.20, 0.3 ,0.4,0.5,0.6,0.7,0.8,0.9,1]
for i in epsilon:
    new_x=fgsm_data(X_test=X_test,eps=i)
    lr_eval_data=evaluate(i,lr_model_base, y_test , new_x)
    results.append(lr_eval_data)
fgsm_res=pd.DataFrame(results)
fgsm_res
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = model.predict(X_test)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred,pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred,pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred,pos_label=1 )*100)
    return {'attack':"FSGM",'Model':"SVC", 'epsilon' : i  ,'accuracy': accuracy,␣
 ↪'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
for i in epsilon:
    new_x= fgsm_data(X_test=X_test,eps=i)
    svc_eval_data=evaluate(i,svc_model_base, y_test , new_x)
    results.append(svc_eval_data)
fgsm_res=pd.DataFrame(results)
fgsm_res
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = np.argmax(model.predict(X_test), axis=1)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred, pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred, pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred, pos_label=1 )*100)
    return {'attack':"FSGM",'Model':"ANN", 'epsilon' : i  ,'accuracy': accuracy,␣
 ↪'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
for i in epsilon:
    new_x= fgsm_data(X_test=X_test,eps=i)
    ann_eval_data=evaluate(i,ann_model_base, y_test, new_x)
    results.append(ann_eval_data)
fgsm_res=pd.DataFrame(results)
fgsm_res
```

2.2 Deep fool

[ ]:
```
results=[]
```

[ ]:
```
def deepfool_data(X_test,eps):
    lr_model=SklearnClassifier(model=lr_model_base)
    attack = DeepFool(classifier=lr_model, epsilon=eps)
    new_x=attack.generate(x=X_test)
    return new_x
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = model.predict(X_test)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred,pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred,pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred,pos_label=1 )*100)
    return {'attack':"deepfool",'Model':"LR", 'epsilon' : i  ,'accuracy':␣
 ↪accuracy, 'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
epsilon=[0,0.000001, 0.0000020, 0.000003 ,0.000004,0.000005,0.000006,0.000007,0.
 ↪000008,0.000009,0.00001]
for i in epsilon:
    new_x=deepfool_data(X_test=X_test,eps=i)
    lr_eval_data=evaluate(i,lr_model_base, y_test , new_x)
    results.append(lr_eval_data)
deepfool_res=pd.DataFrame(results)
deepfool_res
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = model.predict(X_test)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred,pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred,pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred,pos_label=1 )*100)
    return {'attack':"deepfool",'Model':"SVC", 'epsilon' : i  ,'accuracy':␣
 ↪accuracy, 'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
for i in epsilon:
    new_x= deepfool_data(X_test=X_test,eps=i)
```

```
    svc_eval_data=evaluate(i,svc_model_base, y_test , new_x)
    results.append(svc_eval_data)
deepfool_res=pd.DataFrame(results)
deepfool_res
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = np.argmax(model.predict(X_test), axis=1)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred, pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred, pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred, pos_label=1 )*100)
    return {'attack':"deepfool",'Model':"ANN", 'epsilon' : i  ,'accuracy':␣
 ↪accuracy, 'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
for i in epsilon:
    new_x= deepfool_data(X_test=X_test,eps=i)
    ann_eval_data=evaluate(i,ann_model_base, y_test, new_x)
    results.append(ann_eval_data)
deepfool_res=pd.DataFrame(results)
deepfool_res
```

2.3 JSMA

[ ]:
```
results=[]
```

[ ]:
```
def jsma_data(X_test,theta):
    lr_model=SklearnClassifier(model=lr_model_base)
    attack = SaliencyMapMethod(classifier=lr_model, theta=theta, gamma = 0.01)
    new_x=attack.generate(x=X_test)
    return new_x
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = model.predict(X_test)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred,pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred,pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred,pos_label=1 )*100)
    return {'attack':"JSMA",'Model':"LR", 'theta' : i  ,'accuracy': accuracy,␣
 ↪'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
theta=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in theta:
    new_x=jsma_data(X_test=X_test,theta=i)
    lr_eval_data=evaluate(i,lr_model_base, y_test , new_x)
    results.append(lr_eval_data)
jsma_res=pd.DataFrame(results)
```

```
jsma_res
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = model.predict(X_test)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred,pos_label=1 )*100)
    f1score= (f1_score(y_test, y_pred,pos_label=1 )*100)
    recall= (recall_score(y_test, y_pred,pos_label=1 )*100)
    return {'attack':"JSMA",'Model':"SVC", 'theta' : i  ,'accuracy': accuracy,␣
 ↪'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
theta=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in theta:
    new_x=jsma_data(X_test=X_test,theta=i)
    svc_eval_data=evaluate(i,svc_model_base, y_test , new_x)
    results.append(svc_eval_data)
jsma_res=pd.DataFrame(results)
jsma_res
```

[ ]:
```
def evaluate(i, model, y_test,X_test):
    y_pred = np.argmax(model.predict(X_test), axis=1)
    accuracy= (accuracy_score(y_test, y_pred )*100)
    precision= (precision_score(y_test, y_pred, pos_label=0 )*100)
    f1score= (f1_score(y_test, y_pred, pos_label=0 )*100)
    recall= (recall_score(y_test, y_pred, pos_label=0 )*100)
    return {'attack':"JSMA",'Model':"ANN", 'theta' : i  ,'accuracy': accuracy,␣
 ↪'precision':precision, 'f1score': f1score, 'recall': recall}
```

[ ]:
```
for i in theta:
    new_x= jsma_data(X_test=X_test,theta=i)
    ann_eval_data=evaluate(i,ann_model_base, y_test, new_x)
    results.append(ann_eval_data)
jsma_res=pd.DataFrame(results)
jsma_res
```

[ ]:
```
fgsm_res.head()
```

[ ]:
```
deepfool_res.head()
```

[ ]:
```
jsma_res.head()
```

[ ]:
```

```

[ ]:
```
metric=['accuracy','precision','f1score', 'recall']
for i in metric:
    plt.figure(figsize = (8, 6))
    sns.color_palette("Spectral", as_cmap=True)
```

```
    sns.axes_style("whitegrid")
    sns.lineplot(data = fgsm_res,
                 x = 'epsilon',
                 y = i,
                 hue = 'Model',
                 palette = "rocket",
                  linewidth = 3)
    plt.title("FGSM Perturbation rates (epsilon) vs " + i)
```

[ ]:

[ ]:
```
metric=['accuracy','precision','f1score', 'recall']
for i in metric:
    plt.figure(figsize = (8, 6))
    sns.color_palette("Spectral", as_cmap=True)
    sns.lineplot(data = jsma_res,
                 x = 'theta',
                 y = i,
                 hue = 'Model',
                 palette = "rocket",
                  linewidth = 3)
    plt.title("JSMA Perturbation rates (theta) vs " + i)
```

[ ]:
```
metric=['accuracy','precision','f1score', 'recall']
for i in metric:
    plt.figure(figsize = (8, 6))
    sns.color_palette("Spectral", as_cmap=True)
    sns.lineplot(data = deepfool_res,
                 x = 'epsilon',
                 y = i,
                 hue = 'Model',
                 palette = "rocket",
                  linewidth = 3)
    plt.title("Deep Fool Overshoot (epsilon) vs " + i)
```

[ ]:
```
#save as data
deepfool_res.to_csv('deepfool_results', index=False)
fgsm_res.to_csv('fgsm_results', index=False)
jsma_res.to_csv('jsma_results', index=False)
```

[ ]: