A

REPORT

ON

"*Detecting Credit Card Fraud Using Machine Learning*"

In partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

In

Information Technology

(Session 2018-19)

Submitted to:

Dr. Sunil Kumar Jangir

Head of Department

Submitted by:

Asmita Goswami(15EJCIT014)

Lokesh Soni (15EJCIT043)

Harsh Banshiwal(15EJCIT030)

Anshul Jain(15EJCIT009)

DEPARTMENT OF INFORMATION TECHNOLOGY

JAIPUR ENGINEERING COLLEGE & RESEARCH CENTRE

Rajasthan Technical University, Kota

(Session 2018-2019)

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the work, which is being presented in the project entitled "Detecting Credit Card Fraud Using Machine Learning" submitted by ***Mr. Lokesh Soni, Ms. Asmita Goswami, Mr. Anshul Jain and Mr. Harsh Banshiwal,*** *the students of fourth year (VII Semester) B.Tech in Information Technology in partial fulfillment for the award of degree of Bachelor of Technology is a record of student's work carried out and found satisfactory for submission.*

Dr. Sunil Kumar Jangir
HOD, IT

CANDIDATE'S DECLARATION

We hereby declare that the work, which is being presented in the Project Stage I entitled ***"Detecting Credit Card Fraud Using Machine Learning"*** in partial fulfillment for the award of Degree of "Bachelor of Technology" in Information Technology, and submitted to the **Department of Information Technology,** Jaipur Engineering College and Research Centre, Affiliated to Rajasthan Technical University is a record of our own work carried out under the Guidance of **Dr. Sunil Kumar Jangir ,** Head of Department of Information Technology.

ASMITA GOSWAMI
(15EJCIT014)

LOKESH SONI
(15EJCIT043)

ANSHUL JAIN
(15EJCIT009)

HARSH BANSHIWAL
(15EJCIT030)

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to our Project Guide **Dr. Sunil Kumar Jangir**, HOD of Department of Information Technology, Jaipur Engineering College and Research Centre, Jaipur for guiding us from the inception till the completion of the project and for consistent encouragement and support for shaping our project in the presentable form. We sincerely acknowledge them for giving their valuable guidance, support for literature survey, critical reviews and comments for our Project.

We would like to first of all express our thanks to **Mr. Arpit Agrawal**, Director of JECRC, for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to **Dr. V. K. Chandna**, Principal of JECRC, for his kind cooperation and extendible support towards the completion of our project.

We also like to express our thanks to all supporting IT faculty members who have been a constant source of encouragement for successful completion of the project.

Also our warm thanks to **Jaipur Engineering College and Research Centre**, who provided us this opportunity to carry out this prestigious Project and enhance our learning in various technical fields.

ASMITA GOSWAMI
(15EJCIT014)

LOKESH SONI
(15EJCIT043)

ANSHUL JAIN
(15EJCIT009)

HARSH BANSHIWAL
(15EJCIT030)

ABSTRACT

The Objective behind this is to demonstrate the technical feasibility of a deep learning approach to design of an effective fraud detection system is necessary in order to reduce the losses incurred by the customers and financial companies. Research has been done on many models and methods to prevent and detect frauds.

For years, fraudsters would simply take numbers from credit or debit card and print them onto blank plastic cards to use at brick-and-mortar stores. But in 2015, Visa and Mastercard mandated that banks and merchants introduce EMV—chip card technology, which made it possible for merchants to start requesting a PIN for each transaction. Nevertheless, experts predict online credit card fraud to soar to a whopping $32 billion in 2020.

Putting it into perspective, this amount is superior to the profits posted recently by some worldwide household, blue chip companies in 2017, such as Coca-Cola ($2 billions), Warren Buffet's Berkshire Hathaway ($24 billions) and JP Morgan Chase ($23.5 billions).In addition to the implementation of chip card technology, companies have been investing massive amounts in other technologies for detecting fraudulent transactions.

Financial fraud still amounts for considerable amounts of money. Hackers and crooks around the world are always looking into new ways of committing financial fraud at each minute. Relying exclusively on rule-based, conventionally programmed systems for detecting financial fraud would not provide the appropriate time-to-market. This is where Machine Learning shines as a unique solution for this type of problem.

The main challenge when it comes to modelling fraud detection as a classification problem comes from the fact that in real world data, the majority of transactions is not fraudulent. Investment in technology for fraud detection has increased over the years so this shouldn't be a surprise, but this brings us a problem: imbalanced data.

CHAPTER INDEX

FIGURE INDEX

Table Index

CHAPTER 1
INTRODUCTION

1.1 Domain Background

Ever since starting my journey into data science, I have been thinking about ways to use data science for good while generating value at the same time. Thus, when I came across this data set on Kaggle dealing with credit card fraud detection, I was immediately hooked. The data set has 31 features, 28 of which have been anonymized and are labeled V1 through V28. The remaining three features are the time and the amount of the transaction as well as whether that transaction was fraudulent or not. Before it was uploaded to Kaggle, the anonymized variables had been modified in the form of a PCA (Principal Component Analysis). Furthermore, there were no missing values in the data set. Equipped with this basic description of the data, let's jump into some exploratory data analysis.

1.2 Problem Statement

This project is to demonstrate the technical feasibility of a deep learning approach to design of an effective fraud detection system is necessary in order to reduce the losses incurred by the customers and financial companies. Research has been done on many models and methods to prevent and detect frauds.

1.3 Datasets and Inputs

The data set contains 284,807 transactions. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts to $25,691.16.

1.4 Solution Statement

The solution will be in Machine Learning, problems like fraud detection are usually framed as classification problems —predicting a discrete class label output given a data observation. Examples of classification problems that can be thought of are Spam Detectors**,** Recommender Systems and Loan Default Prediction**.**

Talking about the credit card payment fraud detection, the classification problem involves creating models that have enough intelligence in order to properly classify transactions as either legit or fraudulent, based on transaction details such as amount, merchant, location, time and others.

Financial fraud still amounts for considerable amounts of money. Hackers and crooks around the world are always looking into new ways of committing financial fraud at each minute. Relying exclusively on rule-based, conventionally programmed systems for detecting financial fraud would not provide the appropriate time-to-market. This is where Machine Learning shines as a unique solution for this type of problem.

The main challenge when it comes to modelling fraud detection as a classification problem comes from the fact that in real world data, the majority of transactions is not fraudulent. Investment in technology for fraud detection has increased over the years so this shouldn't be a surprise, but this brings us a problem: imbalanced data.

1.5 Benchmark Model

For this problem, the benchmark model will be XGBoost Classifier. I will try to beat its performance.

1.6 Evaluation Metrics

Prediction results will be evaluated based on F1 Score with 'weighted' average.

1.7 Project Design

Since nearly all predictors have been anonymized, I decided to focus on the non-anonymized predictors time and amount of the transaction during my EDA. The data set contains 284,807 transactions. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts to $25,691.16. However, as you might be guessing right now based on the mean and maximum, the distribution of the monetary value of all transactions is heavily right-skewed. The vast majority of transactions are relatively small and only a tiny fraction of transactions comes even close to the maximum.

The time is recorded in the number of seconds since the first transaction in the data set. Therefore, we can conclude that this data set includes all transactions recorded over the course of two days. As opposed to the distribution of the monetary value of the transactions, it is bimodal. This indicates that approximately 28 hours after the first transaction there was a significant drop in the volume of transactions. While the time of the first transaction is not provided, it would be reasonable to assume that the drop in volume occurred during the night.

What about the class distributions? How many transactions are fraudulent and how many are not? Well, as can be expected, most transactions are non-fraudulent. In fact, 99.83% of the transactions in this data set were not fraudulent while only 0.17% were fraudulent. The following visualization underlines this significant contrast.

Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regards to our class variable.

As you can see, some of our predictors do seem to be correlated with the class variable. Nonetheless, there seem to be relatively little significant correlations for such a big number of variables. This can probably be attributed to two factors:

1. The data was prepared using a PCA, therefore our predictors are principal components.
2. The huge class imbalance might distort the importance of certain correlations with regards to our class variable.

CHAPTER 2
TECHNOLOGY USED

2.1 What is Machine Learning?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

It is also closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

2.2 Machine Learning Algorithms

Machine learning algorithms are often categorized as supervised or unsupervised.

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

- **Unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.
- **Active learning algorithm**: The computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labeling.
- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

2.3 Machine learning applications

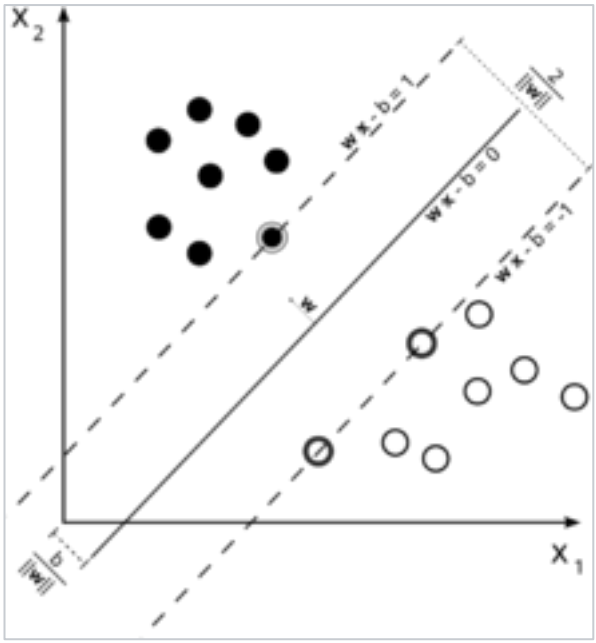

Figure 2.1: A support vector machine is a classifier that divides its input space into two regions, separated by a linear boundary. Here, it has learned to distinguish black and white circles.

Another categorization of machine learning tasks arises when one considers the desired *output* of a machine-learned system:

- In **classification**, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

- In **regression**, also a supervised problem, the outputs are continuous rather than discrete.

- In **clustering**, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

- Density estimation finds the distribution of inputs in some space.

- Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

Among other categories of machine learning problems, learning to learn learns its own inductive bias based on previous experience. Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous self-exploration and social interaction with human teachers and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

2.4 History of Machine Learning

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. As a scientific endeavour, machine learning grew out of the quest for artificial intelligence. Already in the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what were then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation. By 1980, expert systems had come to dominate AI, and statistics was out of favor. Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval. Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.

Machine learning, reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory. It also benefited from the increasing availability of digitized information, and the ability to distribute it via the Internet.

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on *known* properties learned from the training data, data mining focuses on the discovery of (previously) *unknown* properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to *reproduce known* knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously *unknown* knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples). The difference between the two fields arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.

2.4.1 Relation to statistics

Machine learning and statistics are closely related fields. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long prehistory in statistics. He also suggested the term data science as a placeholder to call the overall field.

Leo Breiman distinguished two statistical modelling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine learning algorithms like Random forest.

Some statisticians have adopted methods from machine learning, leading to a combined field that they call *statistical learning*.

2.5 Theory

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has underfit the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer.

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

2.6 Approach

2.6.1 Decision tree learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

2.6.2 Association rule learning

Association rule learning is a method for discovering interesting relations between variables in large databases.

2.6.3 Artificial neural networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

2.6.4 Deep learning

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

2.6.5 Inductive logic programming

Inductive logic programming (ILP) is an approach to rule learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

2.6.6 Support vector machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

2.6.7 Clustering

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to some predesignated criterion or

criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated for example by *internal compactness* (similarity between members of the same cluster) and *separation* between different clusters. Other methods are based on *estimated density* and *graph connectivity*. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

2.6.8 Bayesian networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

2.6.9 Representation learning

Several learning algorithms, mostly unsupervised learning algorithms, aim at discovering better representations of the inputs provided during training. Classical examples include principal components analysis and cluster analysis. Representation learning algorithms often attempt to preserve the information in their input but transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions, allowing reconstruction of the inputs coming from the unknown data generating distribution, while not being necessarily faithful for configurations that are implausible under that distribution.

Manifold learning algorithms attempt to do so under the constraint that the learned representation is low-dimensional. Sparse coding algorithms attempt to do so under the constraint that the learned representation is sparse (has many zeros). Multilinear subspace learning algorithms aim to learn low-dimensional representations directly from tensor representations for multidimensional data, without reshaping them into (high-dimensional) vectors. Deep learning algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or generating) lower-level features. It has been argued that an intelligent machine is one that learns a

representation that disentangles the underlying factors of variation that explain the observed data.

2.6.10 Similarity and metric learning

In this problem, the learning machine is given pairs of examples that are considered similar and pairs of less similar objects. It then needs to learn a similarity function (or a distance metric function) that can predict if new objects are similar. It is sometimes used in Recommendation systems.

2.6.11 Sparse dictionary learning

In this method, a datum is represented as a linear combination of basis functions, and the coefficients are assumed to be sparse. Let x be a d-dimensional datum, D be a d by n matrix, where each column of *Represents* a basis function. r is the coefficient to represent x using D. Mathematically, sparse dictionary learning means solving {\displaystyle x\approx Dr} where r is sparse. Generally speaking, n is assumed to be larger than $d$to allow the freedom for a sparse representation.

Learning a dictionary along with sparse representations is strongly NP-hard and also difficult to solve approximately. A popular heuristic method for sparse dictionary learning is K-SVD.

Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine which classes a previously unseen datum belongs to. Suppose a dictionary for each class has already been built. Then a new datum is associated with the class such that it's best sparsely represented by the corresponding dictionary. Sparse dictionary learning has also been applied in image denoising. The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.

2.6.12 Genetic algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

2.6.13 Rule-based machine learning

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply, knowledge. The defining characteristic of a rule-based machine learner is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learners that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

2.6.14 Learning classifier systems

Learning classifier systems (LCS) are a family of rule-based machine learning algorithms that combine a discovery component (e.g. typically a genetic algorithm) with a learning component (performing either supervised learning, reinforcement learning, or unsupervised learning). They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a piecewise manner in order to make predictions.

2.7 Applications

Some of Machine Learning applications include:-

- Agriculture
- Automated theorem proving
- Adaptive websites
- Affective computing
- Bioinformatics
- Brain–machine interfaces
- Cheminformatics
- Classifying DNA sequences
- Computational anatomy
- Computer Networks
- Telecommunication
- Computer vision, including object recognition
- Detecting credit-card fraud
- General game playing
- Information retrieval
- Internet fraud detection
- Computational linguistics
- Marketing
- Machine learning control

- Machine perception

In 2006, the online movie company Netflix held the first "Netflix Prize" competition to find a program to better predict user preferences and improve the accuracy on its existing Cinematic movie recommendation algorithm by at least 10%. A joint team made up of researchers from AT&T Labs-Research in collaboration with the teams Big Chaos and Pragmatic Theory built an ensemble model to win the Grand Prize in 2009 for $1 million. Shortly after the prize was awarded, Netflix realized that viewers' ratings were not the best indicators of their viewing patterns ("everything is a recommendation") and they changed their recommendation engine accordingly.

In 2010 The Wall Street Journal wrote about the firm Rebellion Research and their use of Machine Learning to predict the financial crisis.

In 2012, co-founder of Sun Microsystems Vinod Khosla predicted that 80% of medical doctors jobs would be lost in the next two decades to automated machine learning medical diagnostic software.

In 2014, it has been reported that a machine learning algorithm has been applied in Art History to study fine art paintings, and that it may have revealed previously unrecognized influences between artists.

2.8 Limitations

Although machine learning has been transformative in some fields, effective machine learning is difficult because finding patterns is hard and often not enough training data are available; as a result, many machine-learning programs often fail to deliver the expected value. Reasons for this are numerous: lack of (suitable) data, lack of access to the data, data bias, privacy problems, badly chosen tasks and algorithms, wrong tools and people, lack of resources, and evaluation problems.

Machine learning approaches in particular can suffer from different data biases. A machine learning system trained on your current customers only may not be able to predict the needs of new customer groups that are not represented in the training data. When trained on man-made data, machine learning is likely to pick up the same constitutional and unconscious biases already present in society. Language models learned from data have been shown to contain human-like biases. Machine learning systems used for criminal risk assessment have

been found to be biased against black people. In 2015, Google photos would often tag black people as gorillas, and in 2018 this still was not well resolved, but Google reportedly was still using the workaround to remove all gorilla from the training data, and thus was not able to recognize real gorillas at all. Similar issues with recognizing non-white people have been found in many other systems. In 2016, Microsoft tested a chatbot that learned from Twitter, and it quickly picked up racist and sexist language. Because of such challenges, the effective use of machine learning may take longer to be adopted in other domains. In 2018, a self-driving car from Uber failed to detect a pedestrian, who got killed in the accident. Attempts to use machine learning in healthcare with the IBM Watson system failed to deliver even after years of time and billions of investment.

2.9 Model assessments

Classification machine learning models can be validated by accuracy estimation techniques like the Holdout method, which splits the data in a training and test set (conventionally 2/3 training set and 1/3 test set designation) and evaluates the performance of the training model on the test set. In comparison, the N-fold-cross-validation method randomly splits the data in k subsets where the k-1 instances of the data are used to train the model while the kth instance is used to test the predictive ability of the training model. In addition to the holdout and cross-validation methods, bootstrap, which samples n instances with replacement from the dataset, can be used to assess model accuracy.

2.10 Software

Software suites containing a variety of machine learning algorithms include the following:

2.10.1 Free and open-source software

- CNTK
- Deeplearning4j
- ELKI
- H2O
- Mahout
- Mallet
- MLPACK
- MXNet
- OpenNN
- Orange
- scikit-learn
- Shogun
- Spark MLlib
- TensorFlow

- Torch / PyTorch
- Weka / MOA
- Yooreeka

2.10.2 Proprietary software with free and open-source editions

- KNIME
- RapidMiner

2.10.3 Proprietary software

- Amazon Machine Learning
- Angoss KnowledgeSTUDIO
- Ayasdi
- IBM Data Science Experience
- Google Prediction API
- IBM SPSS Modeler
- KXEN Modeler
- LIONsolver
- Mathematica
- MATLAB
- Microsoft Azure Machine Learning
- Neural Designer
- NeuroSolutions
- Oracle Data Mining
- Oracle AI Platform Cloud Service

CHAPTER 3
PROJECT OVERVIEW

3.1 Project Overview

For years, fraudsters would simply take numbers from credit or debit card and print them onto blank plastic cards to use at brick-and-mortar stores. But in 2015, Visa and Mastercard mandated that banks and merchants introduce EMV—chip card technology, which made it possible for merchants to start requesting a PIN for each transaction. Nevertheless, experts predict online credit card fraud to soar to a whopping $32 billion in 2020.

Putting it into perspective, this amount is superior to the profits posted recently by some worldwide household, blue chip companies in 2017, such as Coca-Cola ($2 billions), Warren Buffet's Berkshire Hathaway ($24 billions) and JP Morgan Chase ($23.5 billions).In addition to the implementation of chip card technology, companies have been investing massive amounts in other technologies for detecting fraudulent transactions.

The Objective behind this is to demonstrate the technical feasibility of a deep learning approach to design of an effective fraud detection system is necessary in order to reduce the losses incurred by the customers and financial companies. Research has been done on many models and methods to prevent and detect frauds.

3.2 Deep Learning

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts.

Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains (especially human brain), which make them incompatible with neuroscience evidences.

Parts of deep learning is as follows :-

3.2.1 Neural Network

Artificial neural networks (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve their ability) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express with a traditional computer algorithm using rule-based programming.

An ANN is based on a collection of connected units called artificial neurons, (analogous to biological neurons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream.

Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as backpropagation, or passing information in the reverse direction and adjusting the network to reflect that information.

Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

As of 2017, neural networks typically have a few thousand to a few million units and millions of connections. Despite this number being several order of magnitude less than the number of neurons on a human brain, these networks can perform many tasks at a level beyond that of humans (e.g., recognizing faces, playing "Go").

3.2.2 Deep Neural Network

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculate the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks. The goal is that eventually, the network will be trained to decompose an image into features, identify trends that exist across all samples and classify new images by their similarities without requiring human input.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network.

Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets.

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights

and inputs are multiplied and return an output between 0 and 1. If the network didn't accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

Recurrent neural networks (RNNs), in which data can flow in any direction, are used for applications such as language modeling. Long short-term memory is particularly effective for this use.

Convolutional deep neural networks (CNNs) are used in computer vision. CNNs also have been applied to acoustic modeling for automatic speech recognition (ASR).

3.2.3 Convolutional neural networks

Convolutional neural networks (CNNs) represent an interesting method for adaptive image processing, and form a link between general feed-forward neural networks and adaptive filters. Two dimensional CNNs are formed by one or more layers of two dimensional filters, with possible non-linear activation functions and/or down-sampling. CNNs possess key properties of translation invariance and spatially local connections (receptive fields). I present a description of the convolutional network architecture, and an application to practical image processing on a mobile robot.

A CNN is used to detect and characterize cracks on an autonomous sewer inspection robot. The filter sizes used in all cases were 4x4, with non-linear activations between each layer. The number of feature maps used in the three hidden layers was, from input to output, 4, 4, 4. The network was trained using a dataset of 48x48 sub-regions drawn from 30 still image 320x240 pixel frames sampled from a pre-recorded sewer pipe inspection video. 15 frames were used for training and 15 for validation of network performance.

Although development of a CNN system for civil use is on-going, the results support the notion that data-based adaptive image processing methods such as CNNs are useful for image processing, or other applications where the input arrays are large, and spatially / temporally distributed. Further refinements of the CNN architecture, such as the implementation of separable filters, or extensions to three dimensional (ie. video) processing, are suggested.

3.2.4 Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take *actions* in an *environment* so as to maximize some notion of cumulative *reward*. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, reinforcement learning is called *approximate dynamic programming,* or *neuro-dynamic programming.* The problems of interest in reinforcement learning have also been studied in the theory of optimal control, which is concerned mostly with the existence and characterization of optimal solutions, and algorithms for their exact computation, and less with learning or approximation, particularly in the absence of a mathematical model of the environment. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

In machine learning, the environment is typically formulated as a Markov Decision Process (MDP), as many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

Reinforcement learning differs from standard supervised learning in that correct input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected. Instead the focus is on performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The exploration vs. exploitation trade-off has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

3.2.5 Deep Q-Learning

The DeepMind system used a deep convolutional neural network, with layers of tiled convolutional filters to mimic the effects of receptive fields. Reinforcement learning is

unstable or divergent when a nonlinear function approximator such as a neural network is used to represent Q. This instability comes from the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and the data distribution, and the correlations between Q and the target values.

The technique used *experience replay,* a biologically inspired mechanism that uses a random sample of prior actions instead of the most recent action to proceed.[2] This removes correlations in the observation sequence and smooths changes in the data distribution. Iterative update adjusts Q towards target values that are only periodically updated, further reducing correlations with the target.

3.3 Libraries

Below are some libraries in python used in this project :-

3.3.1 About Tensorflow

TensorFlow™ is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

3.3.2 Keras

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer.

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine-learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0.

Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine.

It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU).

3.3.3 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The core functionality of NumPy is its "ndarray", for *n*-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be views into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK). NumPy has built-in support for memory-mapped ndarrays.

3.3.4 Pandas

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Library features include the following :

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

The library is highly optimized for performance, with critical code paths written in Cython or C.

3.3.5 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There

is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

As of 23 June 2017, matplotlib 2.0.x supports Python versions 2.7 through 3.6. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of matplotlib to support Python 2.6.

Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

3.4 IPython (Jupyter notebook)

IPython (Interactive Python) is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history. IPython provides the following features:

- Interactive shells (terminal and Qt-based).
- A browser-based notebook interface with support for code, text, mathematical expressions, inline plots and other media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into one's own projects.
- Tools for parallel computing.

CHAPTER 4
REQUIREMENT ANALYSIS

4.1 SYSTEM REQUIREMENT

The process of deciding on the requirement of a software system, which determines the responsibilities of a system, is called requirement analysis. Requirement analysis is a software engineering task that bridges the gap between system level requirements engineering and software design. Requirement engineering activities result in the specification of software's operational characteristics, indicate the software's interface with other system elements and establish constraints that the software must meet. The following section presents the detailed requirement analysis of our project.

4.1.1 HARDWARE REQUIREMENT:

- Minimum two computers with two hard drives
  - Processor of Pentium or above.
  - Minimum of 1 GB RAM.
  - Minimum of 50 GB Hard disk.
  - Monitor.
  - Mouse.
  - Key Board.
- Working network (LAN)

4.1.2 SOFTWARE REQUIREMENT:

- Jupyter Notebook
- Operating system (Windows,Mac OS,Linux etc.)
- Keras
- Scikit Learn
- Pandas
- Numpy
- Matplotlib
- TensorFlow

CHAPTER 5
PROBLEMS AND SOLUTIONS OF THE PROJECT

5.1 About the problem

Ever since starting my journey into data science, I have been thinking about ways to use data science for good while generating value at the same time. Thus, when I came across this data set on Kaggle dealing with credit card fraud detection, I was immediately hooked. The data set has 31 features, 28 of which have been anonymized and are labeled V1 through V28. The remaining three features are the time and the amount of the transaction as well as whether that transaction was fraudulent or not. Before it was uploaded to Kaggle, the anonymized variables had been modified in the form of a PCA (Principal Component Analysis). Furthermore, there were no missing values in the data set. Equipped with this basic description of the data, let's jump into some exploratory data analysis.

5.2 Datasets

The data set contains 284,807 transactions. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts to $25,691.16.

5.3 Problem Statement

This project is to demonstrate the technical feasibility of a deep learning approach to design of an effective fraud detection system is necessary in order to reduce the losses incurred by the customers and financial companies. Research has been done on many models and methods to prevent and detect frauds.

5.4 Matrices

The evaluation matrices used for this hackerearth competition is based on F1 Score with 'weighted' average.

In statistical analysis of binary classification, the **F_1 score** (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision *p* and the recall *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results returned by the classifier, and *r* is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as

positive). The F_1 score is the harmonic average of the precision and recall, where an F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. The formula for the F1 score is:

$$F1 = 2 * (precision * recall) / (precision + recall)$$

In the multi-class and multi-label case, this is the average of the F1 score of each class with weighting depending on the **average** parameter.

5.5 Analysis

5.5.1 Data Exploration and Visualisation

Since nearly all predictors have been anonymized, I decided to focus on the non-anonymized predictors time and amount of the transaction during my EDA. The data set contains 284,807 transactions. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts to $25,691.16. However, as you might be guessing right now based on the mean and maximum, the distribution of the monetary value of all transactions is heavily right-skewed. The vast majority of transactions are relatively small and only a tiny fraction of transactions comes even close to the maximum.

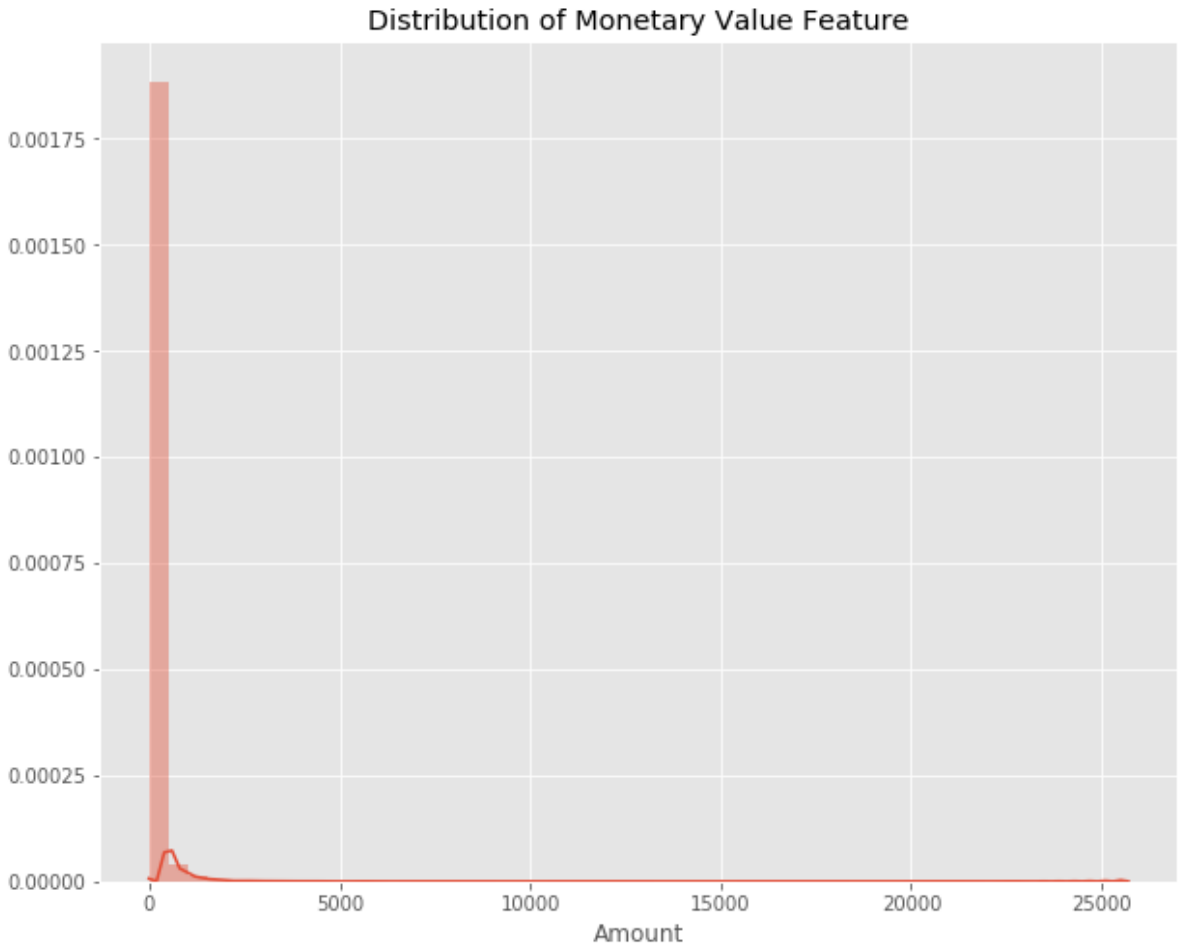

Fig 1.1 : Distribution of Monetary Value Feature

The time is recorded in the number of seconds since the first transaction in the data set. Therefore, we can conclude that this data set includes all transactions recorded over the course of two days. As opposed to the distribution of the monetary value of the transactions, it is bimodal. This indicates that approximately 28 hours after the first transaction there was a significant drop in the volume of transactions. While the time of the first transaction is not provided, it would be reasonable to assume that the drop in volume occurred during the night.

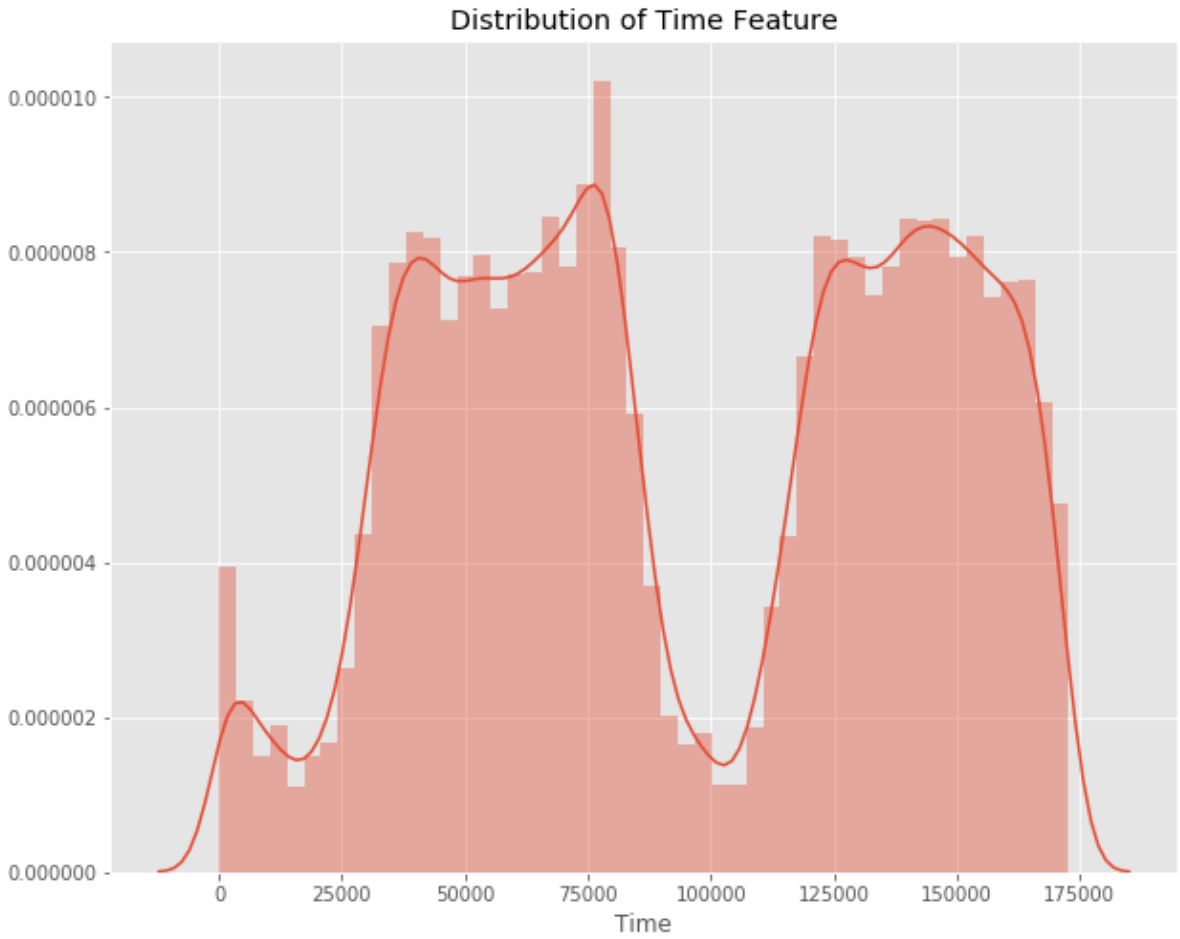

Fig 1.2 : Distribution of Time Feature

What about the class distributions? How many transactions are fraudulent and how many are not? Well, as can be expected, most transactions are non-fraudulent. In fact, 99.83% of the transactions in this data set were not fraudulent while only 0.17% were fraudulent. The following visualization underlines this significant contrast.

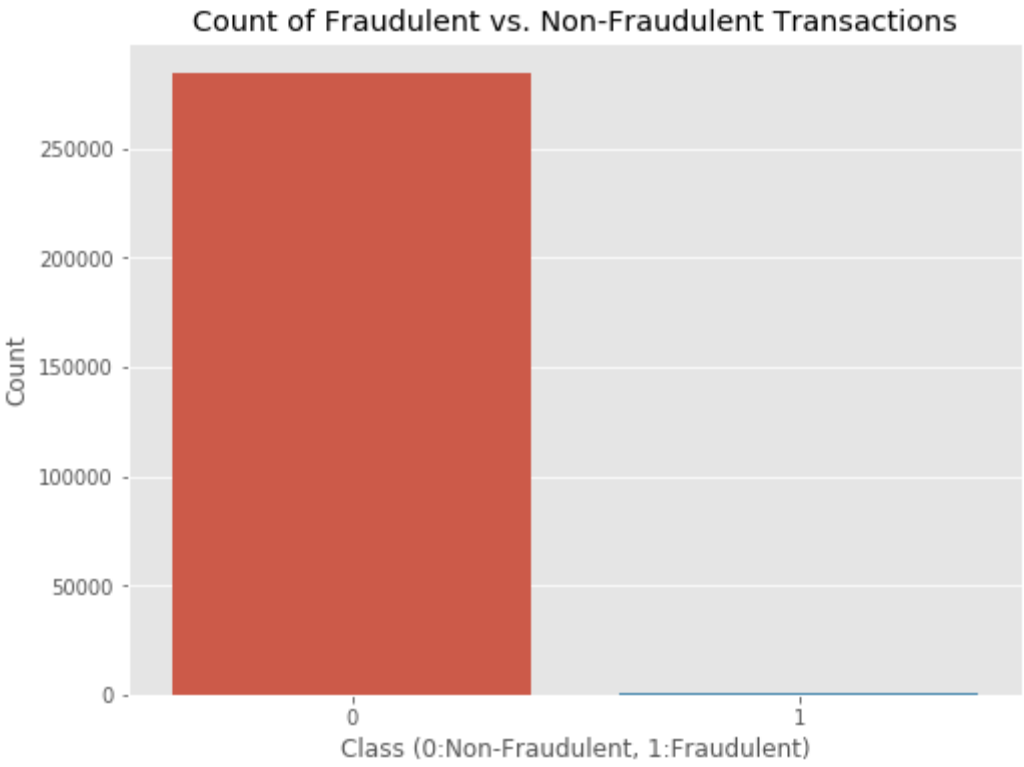

Fig 1.3 : Count of Fraudulent Vs. Non-Fraudulent Transactions

Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regards to our class variable.

Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regards to our class variable. One of the most visually appealing ways to determine that is by using a heatmap.

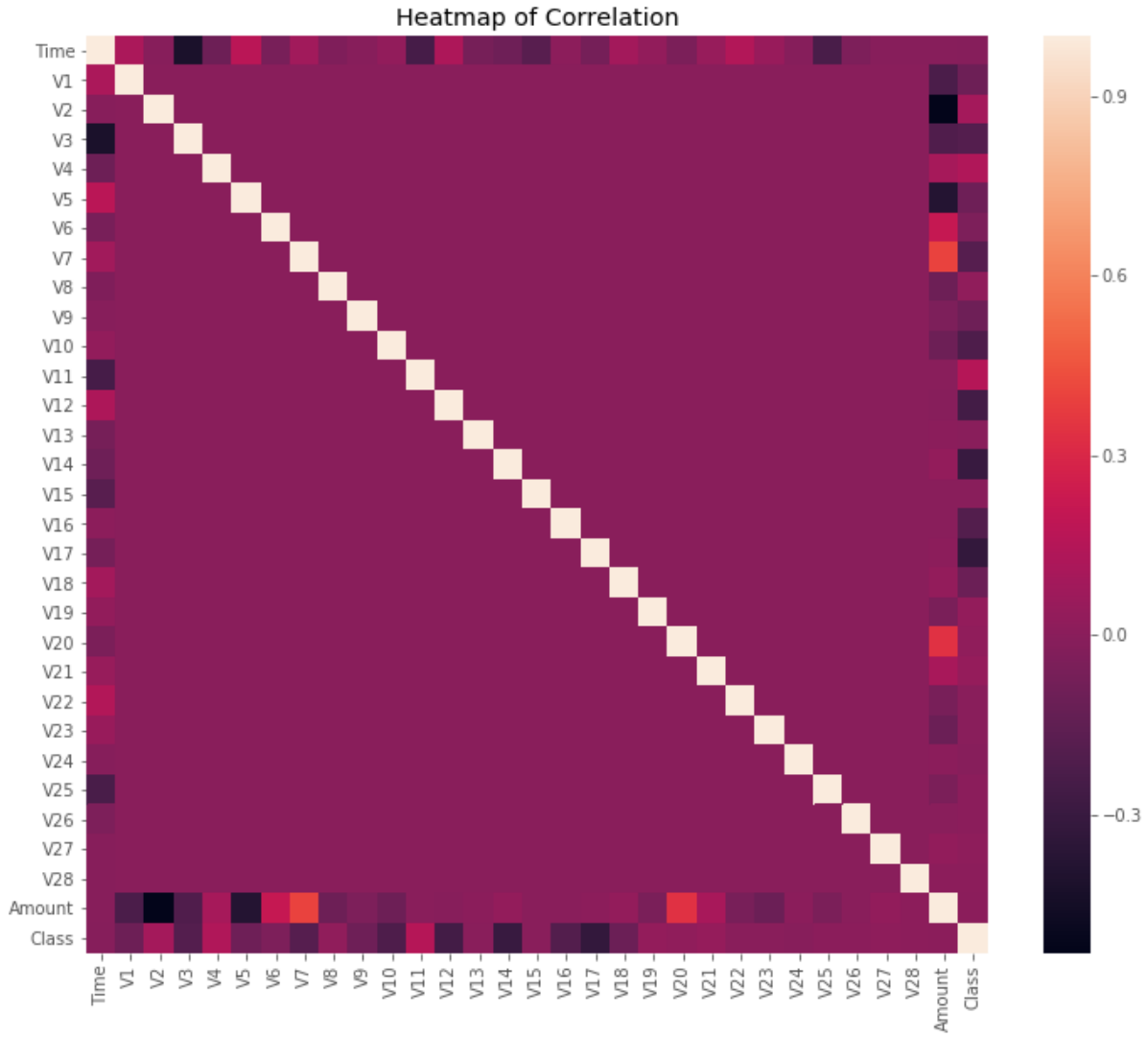

Fig 1.4 : Heatmap of Correlation

As you can see, some of our predictors do seem to be correlated with the class variable. Nonetheless, there seem to be relatively little significant correlations for such a big number of variables. This can probably be attributed to two factors:

3. The data was prepared using a PCA, therefore our predictors are principal components.
4. The huge class imbalance might distort the importance of certain correlations with regards to our class variable.

5.5.2 Data Preparation

Before continuing with our analysis, it is important not to forget that while the anonymized features have been scaled and seem to be centered around zero, our time and amount features have not. Not scaling them as well would result in certain machine learning algorithms that give weights to features (logistic regression) or rely on a distance measure (KNN) performing much worse. To avoid this issue, I standardized both the time and amount column. Luckily, there are no missing values and we, therefore, do not need to worry about missing value imputation.

5.5.3 Creating a Training Set for a Heavily Imbalanced Data Set

Now comes the challenging part: Creating a training data set that will allow our algorithms to pick up the specific characteristics that make a transaction more or less likely to be fraudulent. Using the original data set would not prove to be a good idea for a very simple reason: Since over 99% of our transactions are non-fraudulent, an algorithm that always predicts that the transaction is non-fraudulent would achieve an accuracy higher than 99%. Nevertheless, that is the opposite of what we want. We do not want a 99% accuracy that is achieved by never labeling a transaction as fraudulent, we want to detect fraudulent transactions and label them as such.

There are two key points to focus on to help us solve this. First, we are going to utilize random under-sampling to create a training dataset with a balanced class distribution that will force the algorithms to detect fraudulent transactions as such to achieve high performance. Speaking of performance, we are not going to rely on accuracy. Instead, we are going to make use of the Receiver Operating Characteristics-Area Under the Curve or ROC-AUC performance measure (I have linked further reading below this article). Essentially, the ROC-AUC outputs a value between zero and one, whereby one is a perfect score and zero the worst. If an algorithm has a ROC-AUC score of above 0.5, it is achieving a higher performance than random guessing.

To create our balanced training data set, I took all of the fraudulent transactions in our data set and counted them. Then, I randomly selected the same number of non-fraudulent transactions and concatenated the two. After shuffling this newly created data set, I decided to output the class distributions once more to visualize the difference.

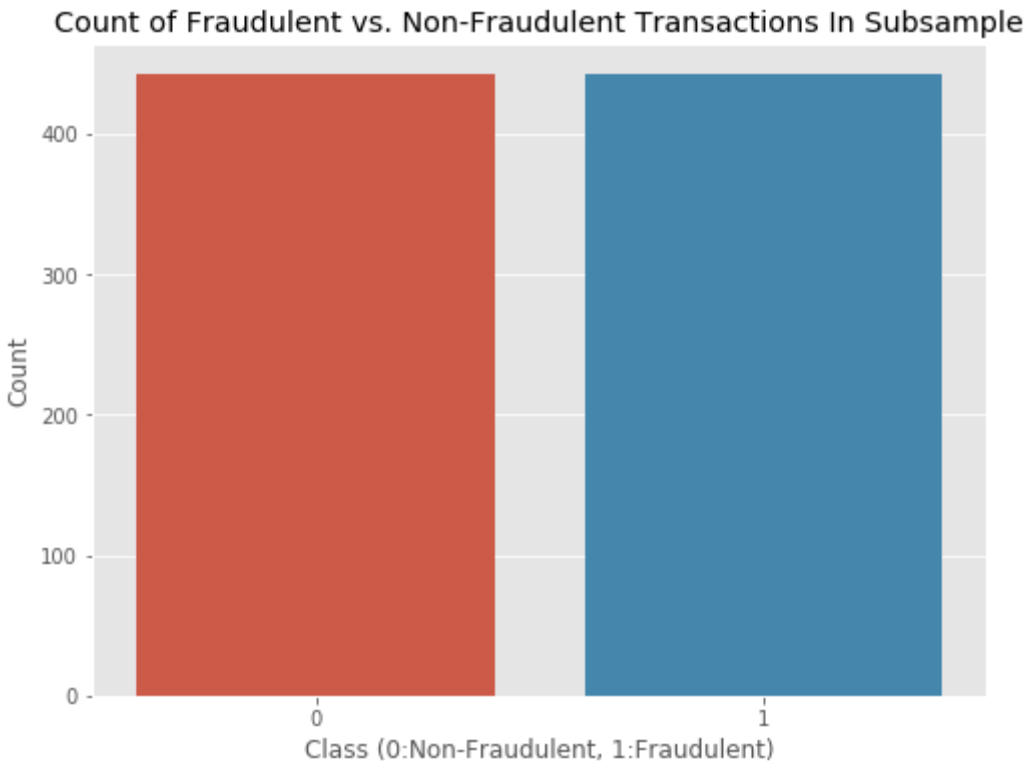

Fig 5.5 : Count of Fraudulent Vs. Non-Fraudulent Transactions In Subsample

5.5.4 Outlier Detection & Removal

Outlier detection is a complex topic. The trade-off between reducing the number of transactions and thus volume of information available to my algorithms and having extreme outliers skew the results of your predictions is not easily solvable and highly depends on your data and goals. In my case, I decided to focus exclusively on features with a correlation of 0.5 or higher with the class variable for outlier removal. Before getting into the actual outlier removal, let's take a look at visualizations of those features:

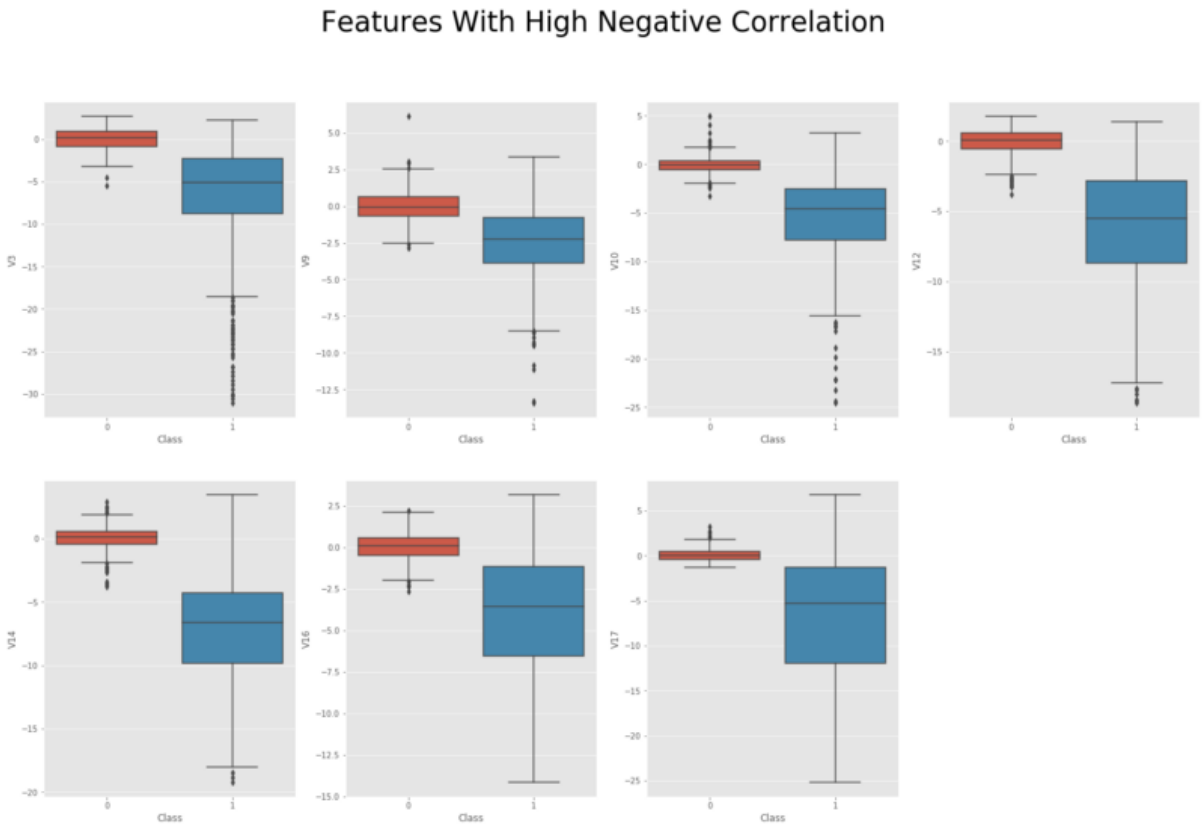

Fig 5.6 : Features with High Negative Correlation

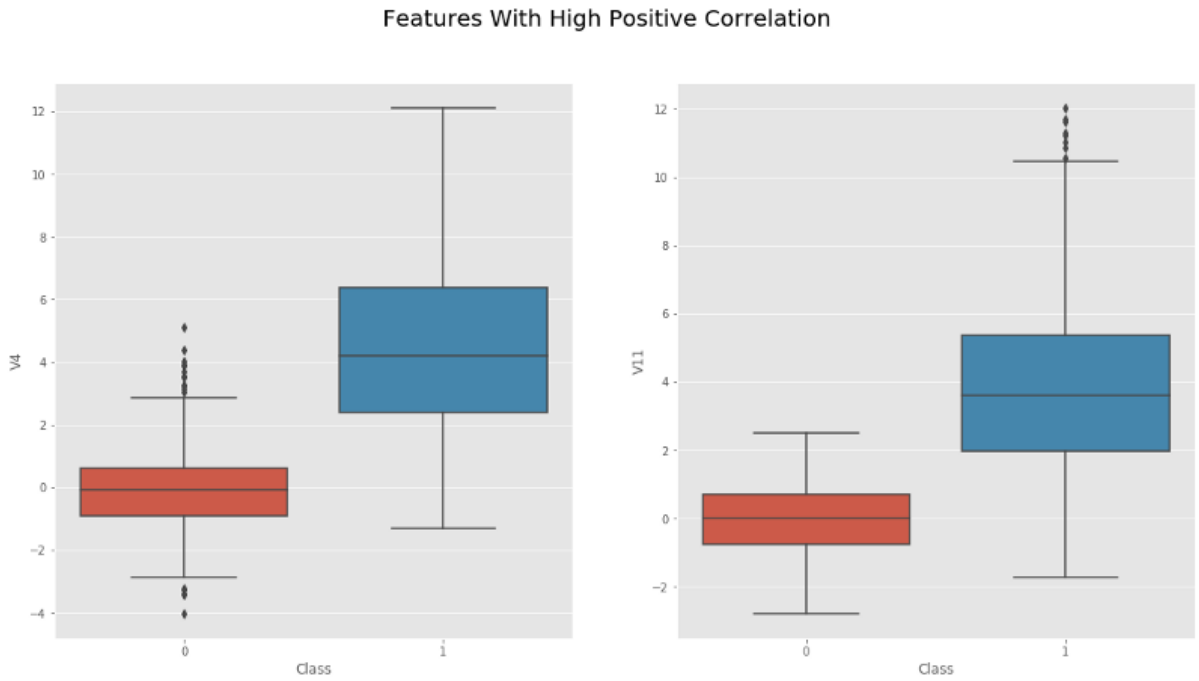

Fig 5.7 : Features with High Positive Correlation

Box plots provide us with a good intuition of whether we need to worry about outliers as all transactions outside of 1.5 times the IQR (Inter-Quartile Range) are usually considered to be outliers. However, removing all transactions outside of 1.5 times the IQR would dramatically decrease our training data size, which is not very large, to begin with. Thus, I decided to only focus on extreme outliers outside of 2.5 times the IQR.

5.5.5 Dimensionality Reduction With t-SNE for Visualization

Visualizing our classes would prove to be quite interesting and show us if they are clearly separable. However, it is not possible to produce a 30-dimensional plot using all of our predictors. Instead, using a dimensionality reduction technique such as t-SNE, we are able to project these higher dimensional distributions into lower-dimensional visualizations. For this project, I decided to use t-SNE, an algorithm that I had not been working with before. If you would like to know more about how this algorithm works.

Projecting our data set into a two-dimensional space, we are able to produce a scatter plot showing the clusters of fraudulent and non-fraudulent transactions:

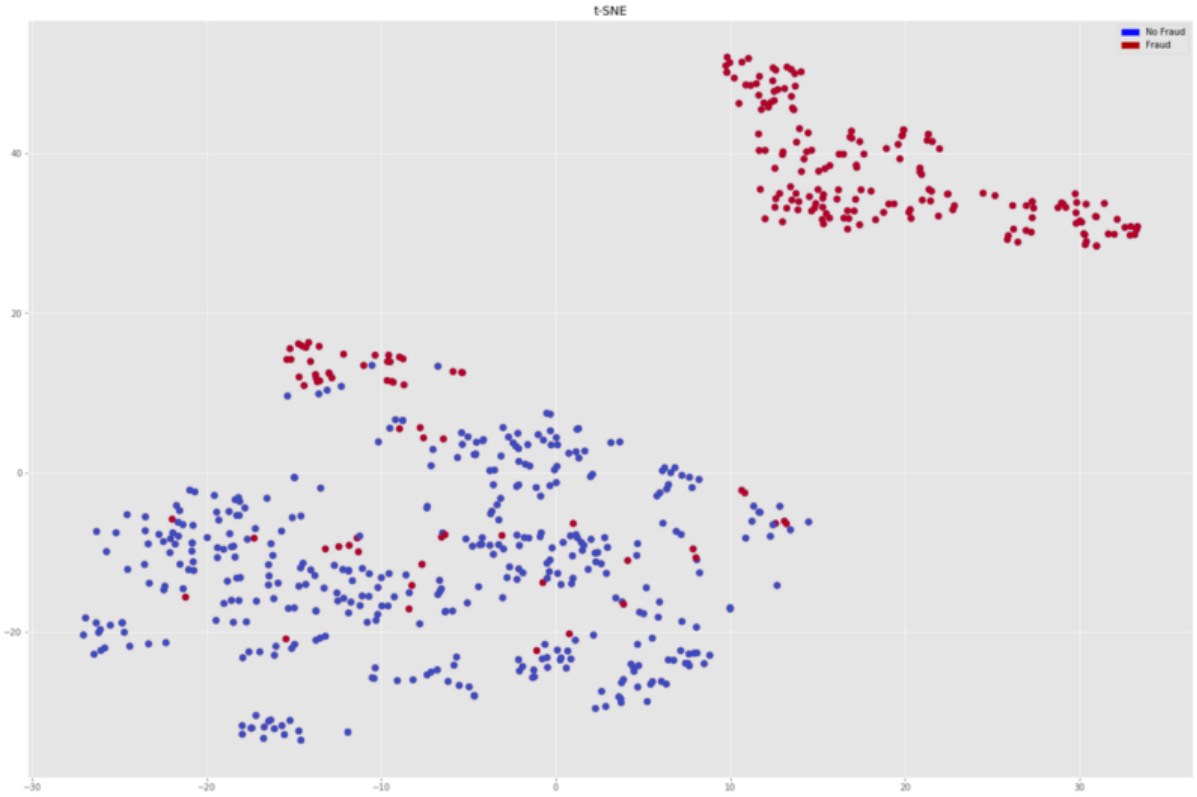

Figure 5.8 :Scatter Plot

5.5.6 Classifications Algorithms

Onto the part you've probably been waiting for all this time: training machine learning algorithms. To be able to test the performance of our algorithms, I first performed an 80/20 train-test split, splitting our balanced data set into two pieces. To avoid overfitting, I used the very common resampling technique of k-fold cross-validation. This simply means that you separate your training data into k parts (folds) and then fit your model on k-1 folds before making predictions for the kth hold-out fold. You then repeat this process for every single fold and average the resulting predictions.

To get a better feeling of which algorithm would perform best on our data, let's quickly spot-check some of the most popular classification algorithms:

- Logistic Regression
- Linear Discriminant Analysis
- K Nearest Neighbors (KNN)
- Classification Trees
- Support Vector Classifier
- Random Forest Classifier
- XGBoost Classifier

5.5.6.1 Logistic Regression

In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model; it is a form of binomial regression. Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the

independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name.

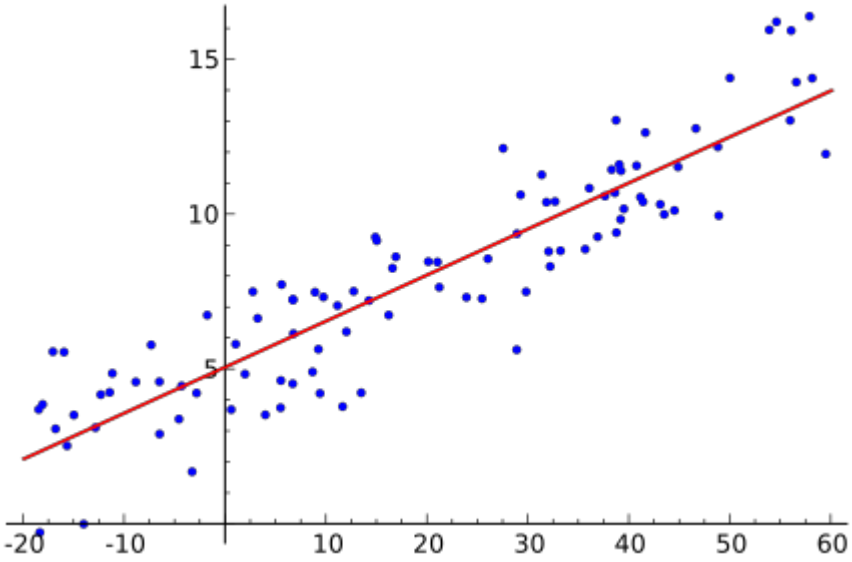

Figure 5.9: Regression Analysis

5.5.6.2 Decision Trees

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values(or expected utility) of competing alternatives are calculated.

5.5.6.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or

mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

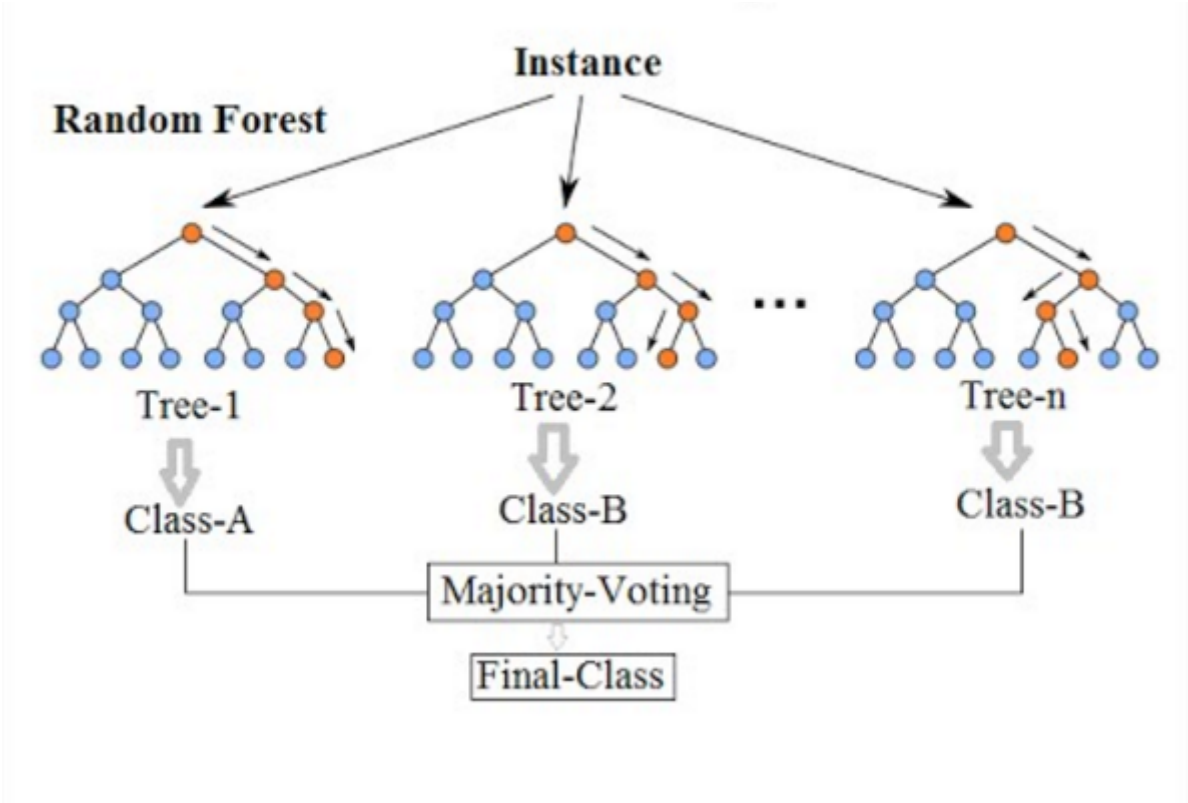

Figure 5.10 : Random Forest Simplified

5.5.6.4 K Nearest neighbors

In pattern recognition, the *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.

5.6 Result

The results of this spot-checking can be visualized as follows:

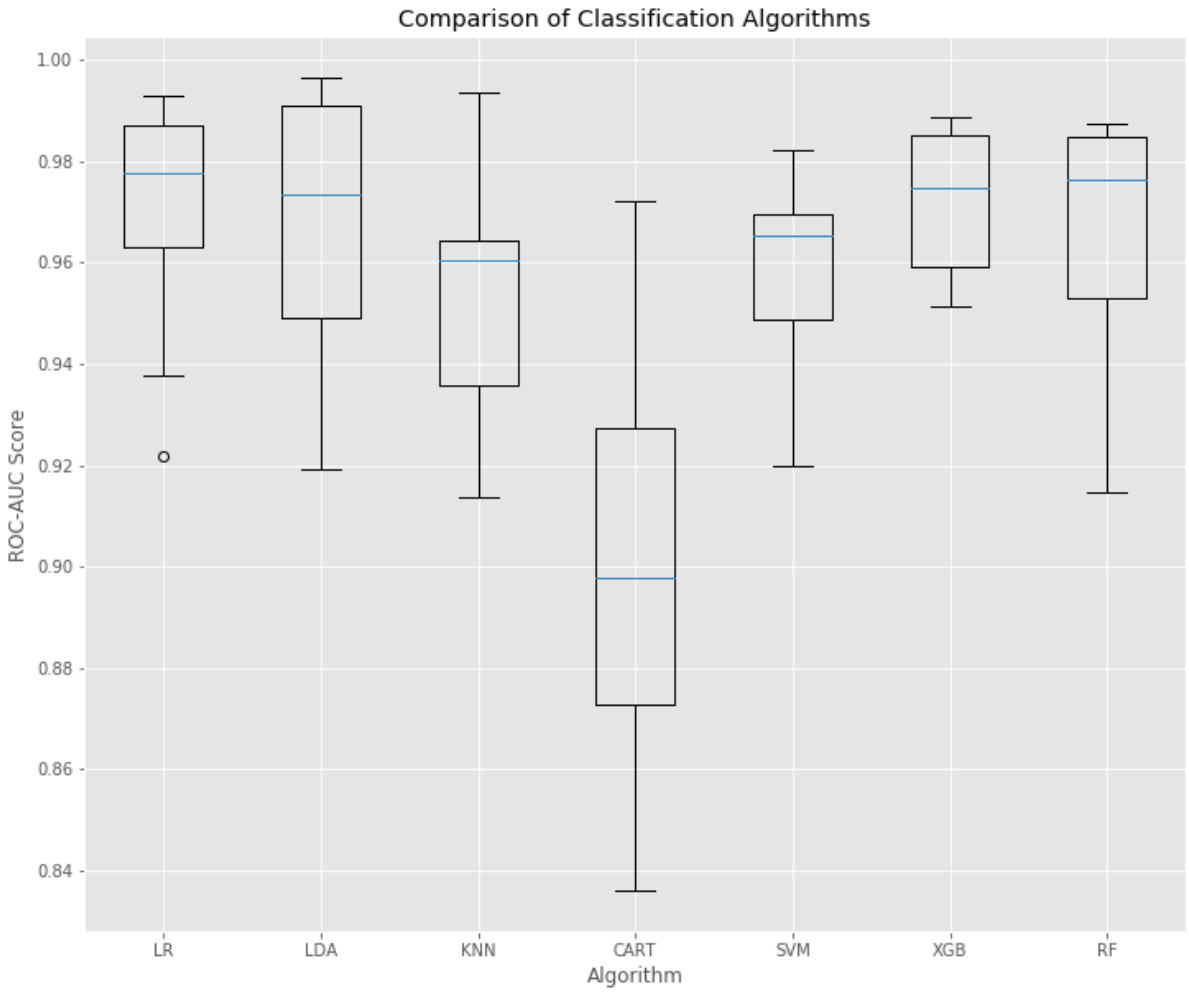

Figure 5.11 : Comparison of Classification Algorithms

5.7 Justification

As we can see, there are a few algorithms that quite significantly outperformed the others. Now, what algorithm do we choose? As mentioned above, this project had not only the focus of achieving the highest accuracy but also to create business value. Therefore, choosing Random Forest over XGBoost might be a reasonable approach in order to achieve a higher degree of comprehensiveness while only slightly decreasing performance. To further illustrate what I mean by this, here is a visualization of our Random Forest model that could easily be used to explain very simply why a certain decision was made:

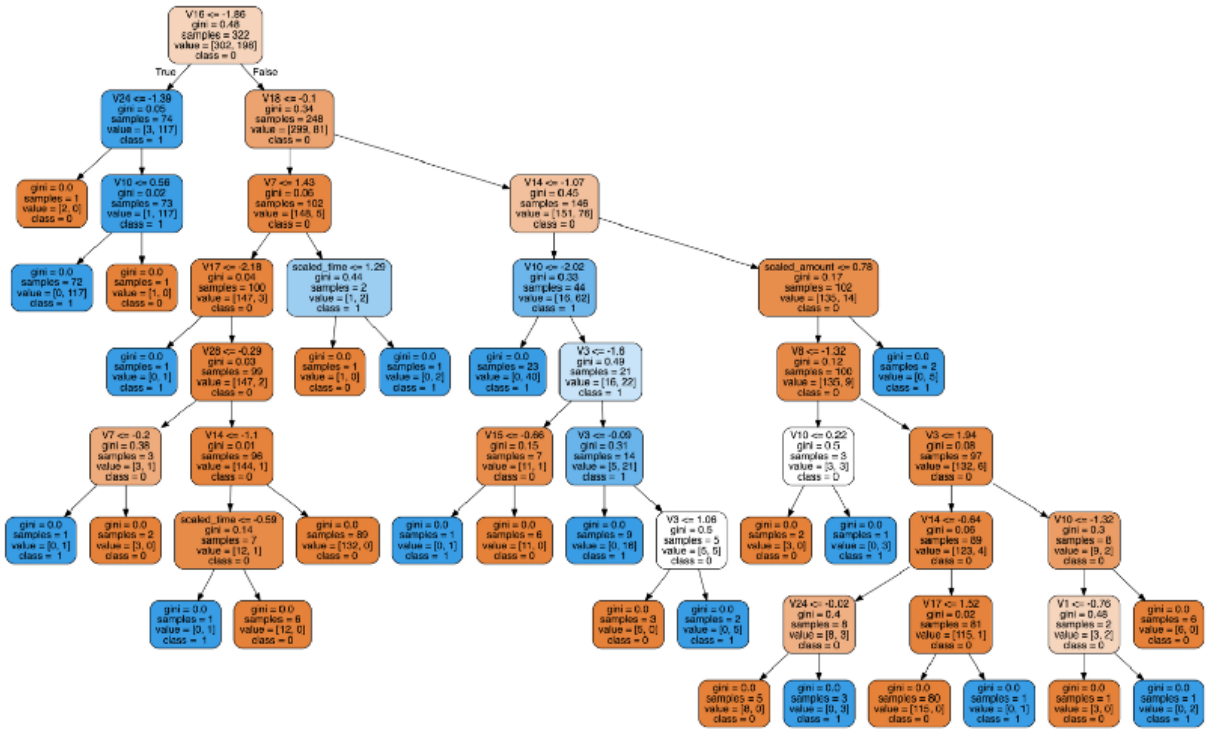

Figure 5.12 : Random Forest Model Visualisation

CONCLUSION

Fraud detection is a complex issue that requires a substantial amount of planning before throwing machine learning algorithms at it. Nonetheless, it is also an application of data science and machine learning for the good, which makes sure that the customer's money is safe and not easily tampered with.

FUTURE SCOPE

Future work will include a comprehensive tuning of the Random Forest algorithm I talked about earlier. Having a data set with non-anonymized features would make this particularly interesting as outputting the feature importance would enable one to see what specific factors are most important for detecting fraudulent transactions.

As always, if you have any questions or found mistakes, please do not hesitate to reach out to me. A link to the notebook with my code is provided at the beginning of this article.

REFERENCES

1. Books

Deep Learning Book by Aaron Courville, Ian Goodfellow, and Yoshua Bengio

2. Other Sources

[1]Nick Becker—The Right Way to Oversample in Predictive Modelling

[2]Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. *Calibrating Probability with Undersampling for Unbalanced Classification*. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

[3] L.J.P. van der Maaten and G.E. Hinton, Visualizing High-Dimensional Data Using t-SNE (2014), Journal of Machine Learning Research

[4] Machine Learning Group—ULB, Credit Card Fraud Detection (2018), Kaggle

[5] Nathalie Japkowicz, Learning from Imbalanced Data Sets: A Comparison of Various Strategies (2000), AAAI Technical Report WS-00–05

MAPPING OF COURSE OUTCOMES (CO) WITH PROGRAM OUTCOMES (PO)

Program Outcomes

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems in IT.

2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences in IT.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations using IT.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions using IT.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations in IT.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice using IT.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development in IT.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice using IT.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in IT.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project Management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage IT projects and in multidisciplinary environments.

12. **Lifelong Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological changes needed in IT.

7ITTR-Practical Training -Course Outcomes (CO)

On completion of the course,

CO-1: Graduate will be able to identify and analyze complex engineering problems through research methodology in Information Technology.

CO-2: Graduate will be able to apply fundamental engineering knowledge to create and interpret data for socio-economic solutions using modern IT tools.

CO-3: Graduate will be able to conduct investigations of complex problems using research-based knowledge to improve thinking, problem solving, and decision making.

CO-4: Graduate will be able to develop communication skills, technical report writing, and professional ethics for life-long learning.

Mapping of COs with POs:

Sem.	Subject	Code	L/T/P	CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
VII	Practical Training	7ITTR	P	**CO-1:** Graduate will be able to identify and analyze complex engineering problems through research methodology in Information Technology.	H	H	M	H	H	M	L	L	M	L	L	M
			P	**CO-2:** Graduate will be able to apply fundamental engineering knowledge to create and interpret data for socio-economic solutions using modern IT tools.	H	M	H	H	M	L	L	M	H	L	H	H
			P	**CO-3:** Graduate will be able to conduct investigations of complex problems using research-based knowledge to improve thinking, problem solving, and decision making.	H	H	M	H	H	L	M	M	H	M	M	H
			P	**CO-4:** Graduate will be able to develop communication skills, technical report writing, and professional ethics for life-long learning.	H	H	L	H	H	H	H	H	H	H	M	H

Detailed description of the mapping:

1. **Engineering Knowledge:** Applied the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems in IT.

2. **Problem analysis:** Identified, research literature, and analyze complex deep learning problems reaching substantiated conclusions using Convolutional Neural Network, Transfer Learning, and Reinforcement Learning in IT.

3. **Design/development of solutions:** Designed solutions for complex Convolutional Neural Network and design Algorithm that meet the specified needs with appropriate consideration for the Research literature using IT.

4. **Conduct investigations of complex problems:** Used research-based knowledge and research methods including design of architecture, analysis and interpretation of data, and synthesis of the information to provide valid conclusions using IT.

5. **Modern tool usage:** Created using Jupyter Notebook and applied appropriate techniques, Knowledge of Python, Keras and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations in IT.

6. **The engineer and society:** Applied reasoning informed by the contextual knowledge to assess societal, health issues and the consequent responsibilities relevant to the professional engineering practice using IT.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development in IT.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice using IT.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in IT.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project Management and finance:** Demonstrated knowledge and understanding of the Deep Learning and applied these to manage projects and in multidisciplinary environments.

12. **Lifelong Learning:** Recognized the need for, and have the preparation and ability to engage in independent and life-long learning in the Machine Learning in IT.

ASMITA GOSWAMI
(15EJCIT014)

LOKESH SONI
(15EJCIT043)

ANSHUL JAIN
(15EJCIT009)

HARSH BANSHIWAL
(15EJCIT030)