

*Go, change the world*



**RV Educational Institutions<sup>®</sup>**  
**RV College of Engineering<sup>®</sup>**

Autonomous  
Institution Affiliated  
to Visvesvaraya  
Technological  
University, Belagavi

Approved by AICTE,  
New Delhi, Accredited  
By NAAC, Bengaluru  
And NBA, New Delhi

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **CREDIT CARD FRAUD DETECTION AI-ML LAB PROJECT REPORT**

**Submitted by,**

**SHREESH MANSOTRA**

**1RV18CS159**

**SHUBH SHUKLA**

**1RV18CS163**

**Under the guidance of**  
Prof. Lalith V P  
Assistant professor  
Dept of CSE  
RV College of Engineering

**In partial fulfilment for the award of degree  
of  
Bachelor of Engineering  
in  
Computer Science and Engineering  
2020-2021**

# RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



### CERTIFICATE

Certified that the AI-ML Lab project work titled '*Credit Card Fraud Detection*' is carried out by **Shreesh Mansotra(1RV18CS159)** and **Shubh Shukla(1RV18CS163)** who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the project report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of AI-ML Lab project work prescribed by the institution for the said degree.

Signature of Guide

Signature of Head of the Department

Signature of Principal

Dr. Ramakanth Kumar P

Dr.K.N.Subramanya

External Viva

Name of Examiners

Signature with Date

1

2

**RV COLLEGE OF ENGINEERING<sup>®</sup>, BENGALURU-59**  
(Autonomous Institution Affiliated to VTU, Belagavi)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**DECLARATION**

**We, Shreesh and Shubh** students of sixth semester B.E., department of CSE, RV College of Engineering, Bengaluru, hereby declare that the AI-ML Lab project titled '*Credit Card Fraud Detection*' has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** during the year 2020-21.

Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

**Name**

**Signature**

1. SHREESH MANSOTRA (1RV18CS159)
2. SHUBH SHUKLA (1RV18CS163)

## ACKNOWLEDGEMENT

We express gratitude to our AI-ML Project lab faculty **Prof. Lalith V P, Assistant professor** Department of Computer Science and Engineering for her valuable comments and suggestions.

Our sincere thanks to **Dr. Ramakant Kumar P.**, Professor and Head, Department of Computer Science and Engineering, RVCE for his support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for his appreciation towards this project work.

We thank all the **teaching staff and technical staff** of Computer Science and Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our **family** members and **friends** who provided all the backup support throughout the project work.

## **ABSTRACT**

The rapid growth of the E-Commerce industry has led to an exponential increase in the use of credit cards for online purchases and consequently there has been a surge in the fraud related to it. In today's world banks have a very ineffective way of detecting the fraudulent transactions in the credit card system. Machine learning/Deep Learning techniques have emerged as a revolutionary technology which would help in detecting credit card fraudulent transactions in the coming years. For predicting these transactions banks make use of various machine learning methodologies, past data is collected and new features have been introduced, thereby enhancing the predictive power.

The credit card fraud detection system is to detect fraud accurately and timely. Even before the fraud is committed. The goal is to detect least and accurate false fraud detection using deep learning techniques. Each transaction has a set of unique features, such as the value of the transaction, the time at which it occurred, issuer, recipient, id and other sensitive data. The problem is structured as a classification problem, where each transaction would be classified as "Normal" or "Fraudulent". An exploratory analysis will be conducted to have a better understanding of the data by optimising the parameters of the Auto-encoder in such a way that the reconstruction error is minimised. We'll then be using a classification algorithm(Random Forest) based on those features and thus evaluate the results.

The dataset obtained is highly unbalanced and the fraudulent cases account for only 0.172% of all the transactions. A comparative analysis is done between Autoencoders and Random Forest Algorithm and the results obtained by Random Forest(99.89%) outperformed Autoencoder(86.63%). Although, we couldn't reach our goal of 100% accuracy in fraud detection yet we ended up creating a system that can, with enough time and data, get very close to that goal. As with any such project, there is a wide room for improvement here by adding more algorithms and also by improving the dataset.

# **Table of Contents**

## **Abstract**

**vii**

## **List of tables**

table 5.1	Mean accuracy & standard deviation
table 5.2	Confusion Matrix of the autoencoder
table 5.3	Confusion Matrix of the Random Forest

## **List of figure**

fig 3.1	Basic Autoencoder Architecture
fig 3.2	Autoencoder 4 hidden Layer and output
fig 3.2.1	Loss function of the autoencoder.
fig 4.3.1	Workflow of entire system
fig 5.1	Precision-Recall curves
fig 5.1	Confusion Matrix(s)

## **1. Introduction**

**0-1**

1.1	Project Domain and Problem addressing
1.2	Issues and Challenges
1.3	Need for AI-based solutions
1.4	Problem Statement
1.5	Objectives

1.6	Summary	
<b>2.</b>	<b>Literature Study</b>	<b>2-4</b>
2.1	Introduction	
2.2	Related work	
2.3	Summary	
<b>3.</b>	<b>Design Details</b>	<b>5-11</b>
3.1	Architecture	
3.2	Methodology	
3.3	Data set details	
3.4	ML/DL techniques used	
3.5	Hardware and Software requirements	
<b>4.</b>	<b>Implementation details of the Project</b>	<b>12-15</b>
4.1	Language/Tools/APIs used	
4.2	Workflow diagrams	
4.3	Data pre-processing	
4.4	Validation methodology	
<b>5.</b>	<b>Results and Analysis</b>	<b>16-18</b>
5.1	Model Evaluation and Validation	
5.2	Justification	
<b>6.</b>	<b>Conclusion and Future Enhancement</b>	<b>19-20</b>
6.1	Novelty in the proposed solution	
6.2	Limitations of the Project	

- 6.3 Future Enhancements
- 6.4 Summary

<b>References</b>	<b>21</b>
-------------------	-----------

## **Appendices**

### **Appendix A: Screenshots**



# **Chapter-1**

## **Introduction**

### **1.1 Project Domain and Problem addressing**

Tens of billions of dollars are lost every year due to credit card fraud. In an increasingly digital world, with our financial data and consumer records becoming only more accessible, it is absolutely imperative that we find reliable ways to develop means of defence and prevention. Transaction data is extremely multifaceted, which means deep learning may be a very effective solution for fraud detection due to learning models' ability to see through subtle intricacies in vast amounts of data.

### **1.2 Issues and Challenges**

A lot of research has been done to find a solution to this problem using ANN. In order to reproduce this kind of problem, we found a useful dataset available on Kaggle. The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced and the positive class (frauds) account for 0.172% of all transactions.

### **1.3 Need for AI-based solutions**

Credit card fraud costs consumers and the financial company billions of dollars annually, and fraudsters continuously try to find new rules and tactics to commit illegal actions. Thus, fraud detection systems have become essential for banks and financial institutions, to minimize their losses. Thus, developing an automatic system for detecting fraudulent cases would be beneficial for users without any delay.

## **1.4 Problem statement**

The aim of the fraud detection system is to detect fraud accurately and before fraud is committed. The goal is to detect least and accurate false fraud detection. The most commonly used techniques for fraud detection methods are Naive Bayes (NB), Support Vector Machines (SVM) and K-Nearest Neighbour algorithms (KNN). Each transaction has a set of unique features, such as the value of the transaction, the time at which it occurred, issuer and recipient, an id and other sensitive data. The problem will be structured as a classification problem, where each transaction could be classified as "Normal" or "Fraud".

## **1.5 Objectives**

- To conduct an exploratory analysis to have a better understanding of the data;
- To try to optimise the parameters of the Autoencoder in such a way that the reconstruction error is minimised.
- To classify the actions based on those features.
- Evaluation of results.

## **1.6 Summary**

The main aim of this project is to ease the process of fraudulent cases of credit card detection, with the best accuracy. This project will undoubtedly help the financial world by detecting the transactions accurately and in a very short span of time.

# Chapter -2

## Literature Study

### 2.1 Introduction

Fraudulent transactions act as the illegal activities which are meant to generate personal or financial profit. It is an intentional act that is criminal with an intention to make financial profit.

There are a number of literature or research papers available on this domain in the public platform. All these papers have helped us in our research.

In [2], the major work was done on analysing and preprocessing data sets as well as the deployment of different irregularity detection algorithms, for example, Local Outlier Factor and Isolation Forest algorithm on the PCA changed Credit Card Transaction information. However, the precision of the algorithms increases when the size of the dataset is increased. Hence, if sufficient data is not available, the accuracy decreases.

In [3], an extension of the best-of- both-worlds principle introduced by Micenková et al. The extension consists of definition of various outlier scores considering different levels of granularity and their reconciliation into the supervised approach. The Model proposes the implementation of a hybrid approach that makes use of unsupervised outlier scores to extend the feature set of a fraud detection classifier. The results are not convincing in terms of the global and local approaches.

In [5], a Comparative Analysis of the technology naïve bayes, k-nearest neighbour and logistic regression was done. The analysis states the effect of hybrid sampling on the performance of binary classification of imbalanced data. Future areas of research could be in examining meta classifiers and meta learning approaches in handling highly imbalanced credit card fraud data. In [6], again a comparative analysis with the existing models indicated that the time attention based recurrent layer network(RNN) model increased the accuracy. The proposed model is tested against a limited dataset. Moreover, the data collected is from a single site which can be further expanded to take data from multiple sites.

## 2.2 Related work

In [7], Auto-encoders are used for extracting essential features from the input data, followed by a classification algorithm.( MultiLayer Perceptron, K-Nearest Neighbour and Logistic Regression). The methodology manages to maintain a decent balance between precision and recall in detecting the fraudulent cases. It also outperforms the systems based on either different classifiers or variants of auto-encoder. The only limitation with the paper is that the model is trained using batch data and in future it can be extended to train on a stream of data.

In [8] Deep Convolutional neural networks (DCNN) + Classifiers were used. Their methods of analysis didn't contain a benchmark of other CNN architectures however DCNN architecture yielded optimal results, outperforming SVM, RF and GBC algorithms with an accuracy of 82%. The Optimized Light Gradient Boost Classifier technology used in the paper [9] indicated that the proposed approach outperformed the other machine learning algorithms and achieved the highest performance in terms of Accuracy, AUC, Precision and F1-score. In the paper [10] , HMM-based features extractor along with random forest classifier. The predictions made by the proposed model are different from that of the already existing LSTM model.

In [11], since the dataset was highly imbalanced, SMOTE technique was used. The algorithms used in the experiment were Logistic Regression,Random Forest, Naive Bayes and Multilayer Perceptron. According to given results, it is shown that classic algorithms, like RF(Best accuracy) can give similar results as a simple neural network with classical algorithms show that oversampling the data can improve fraud detection rate. The only limitation it had was that the predictions made by the proposed model are different from that of already existing models.

In [12],following algorithms are used to build the model and train the model for detection of frauds in credit card transactions:Random Forest Classifier, Decision Tree and Support Vector Machine. The scores of each model were 99.70%, 99.80%, 99.70% for the decision tree, support vector machine and random forest classifier algorithms respectively. [13], proposed a FDS using Random Forest for detection of fraudulent cases. The performance evaluation of the proposed system and other two classifiers i.e Decision Tree and Naive Bayes ,concluded that the proposed system using Random Forest technique performed much better than Decision Tree and Naïve Bayes Technique.

[14] had comparisons made for different machine learning algorithms such as Logistic Regression, Decision Trees, Random Forest, to determine which algorithm suits best. The accuracy for logistic regression, Decision tree and random forest classifier are 90.00, 94.30, and 95.50 respectively. By looking at all the three techniques ,the random forest classifier is better than the logistic regression and decision tree.

## **2.3 Summary**

The main problem we observed in all the papers is about the accuracy which can be a result of unbalanced data and thus not able to produce up to the mark results.

# Chapter 3

## Design Details

### 3.1 Architecture

An autoencoder consists of three components:

- **Encoder:** An encoder is a feedforward, fully connected neural network that compresses the input into a latent space representation and encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.
- **Code:** This part of the network contains the reduced representation of the input that is fed into the decoder.
- **Decoder:** Decoder is also a feedforward network like the encoder and has a similar structure to the encoder. This network is responsible for reconstructing the input back to the original dimensions from the code.

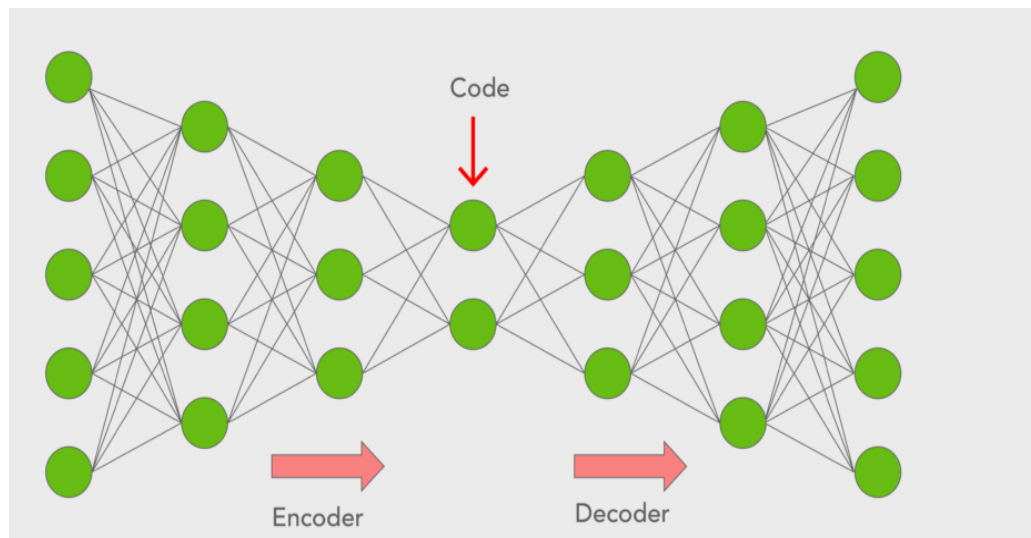


Figure 3.1 - Basic Autoencoder Architecture

First, the input goes through the encoder where it is compressed and stored in the layer called Code, then the decoder decompresses the original input from the code. The main objective of the autoencoder is to get an output identical to the input.

Note that the decoder architecture is the mirror image of the encoder. This is not a requirement but it's typically the case. The only requirement is the dimensionality of the input and output must be the same.

### **The architecture of our model is as follows :**

- First the data has been loaded, and splitted into training and testing (20% of the dataset) sets.
- The training has been split to use the 10% of it for validation. The autoencoder has been developed using keras Model class. It requires a bit of effort to design a network that learns without overfitting. You need to experiment a bit with the number of layers and neurons if you want the loss function to converge.
- From Figure 5 you can give a look at the structure of the network. It has four hidden layers
- . During the encoding phase, we map the input to 28 and 56 neurons,
- Then during the decoding phase, we try to reconstruct the original input with the decode phase (i.e. two hidden layers of 56 and 28 neurons respectively and 23 neurons on the output).

### **Algorithms and Techniques**

Each input element is treated as a large number of highly interconnected processing elements (neurons) working together to solve specific problems. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. The network learns the weights of each connection in such a way that the output error is minimised. A neural network usually is made of an input layer, a variable number of hidden layers and the output layer. You can represent a neural network as a function of functions. Each layer represents a particular function, whose input is given by the previous layer and the output is delivered to the next layer. The information flows only from the input to the output.

Autoencoders are a particular class of neural networks that take an input, "compress" that input down to core features and try to reconstruct the original input from this squeezed representation. The input is usually mapped into a smaller or larger space from the first half of the network. This first phase is called encoding and produces a modified representation of the original input. The

second part of the network "decodes" this representation, and tries to reconstruct the original input in the output layer. In order to set up an auto encoder, we need to select a certain number of layers and activation functions, the learning rate at which the network operates and the encoding dimension of each layer. This model is used for applications in which we want to recognise anomalies or "distortions" on data. We can use the auto encoder as follows: we train it to learn how to recognise normal transactions. During the first phase, the network encodes the original input (i.e. a transaction) into a larger space; then in the decoding phase, tries to recognise the representation and to reconstruct the original input.

Decision tree learning uses a decision tree to go from observations about an item (represented as branches) to conclusions about the item's target value (leaves). Trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time trained on different parts of the same training set, with the goal of reducing the variance. In this way random decision forests correct for decision trees' habit of overfitting to their training set. You can set up a random forest classifier adjusting the number of estimators, max depth, the minimum number of samples required to split an internal node or to be at a leaf node and a specific class weight.

This kind of model fits perfectly with classification problems as the one we are trying to solve. Thus it will be used to learn how to classify transactions as normal or fraud.

## **3.2 Methodology**

The classifiers were trained on the preprocessed training data. This was done in a Jupyter notebook (titled Auto encoders Credit Card Frauds Detection) using Python 3. The following libraries were used:

- keras: to implement the autoencoder;
- sklearn: to implement the random forest and metrics;



- imblearn: to implement SMOTE oversampling;
- matplotlib: to plot all the curves;
- pandas and numpy: in order to easily process the data.

First the data has been loaded, and splitted into training and testing (20% of the dataset) sets. The training has been split to use the 10% of it for validation. The autoencoder has been developed using keras Model class. It requires a bit of effort to design a network that learns without overfitting. You need to experiment a bit with the number of layers and neurons if you

Layer (type)	Output Shape	Param #
input_13 (InputLayer)	(None, 23)	0
dense_55 (Dense)	(None, 28)	672
dense_56 (Dense)	(None, 56)	1624
dense_57 (Dense)	(None, 56)	3192
dense_58 (Dense)	(None, 28)	1596
dense_59 (Dense)	(None, 23)	667
Total params: 7,751		
Trainable params: 7,751		
Non-trainable params: 0		

*Figure 3.2: The autoencoder, made of 4 hidden layers and the output.*

want the loss function to converge. From Figure 5 you can give a look at the structure of the network. It has four hidden layers. During the encoding phase, we map the input to 28 and 56 neurons, then we try to reconstruct the original input with the decode phase (i.e. two hidden layers of 56 and 28 neurons respectively and 23 neurons on the output).

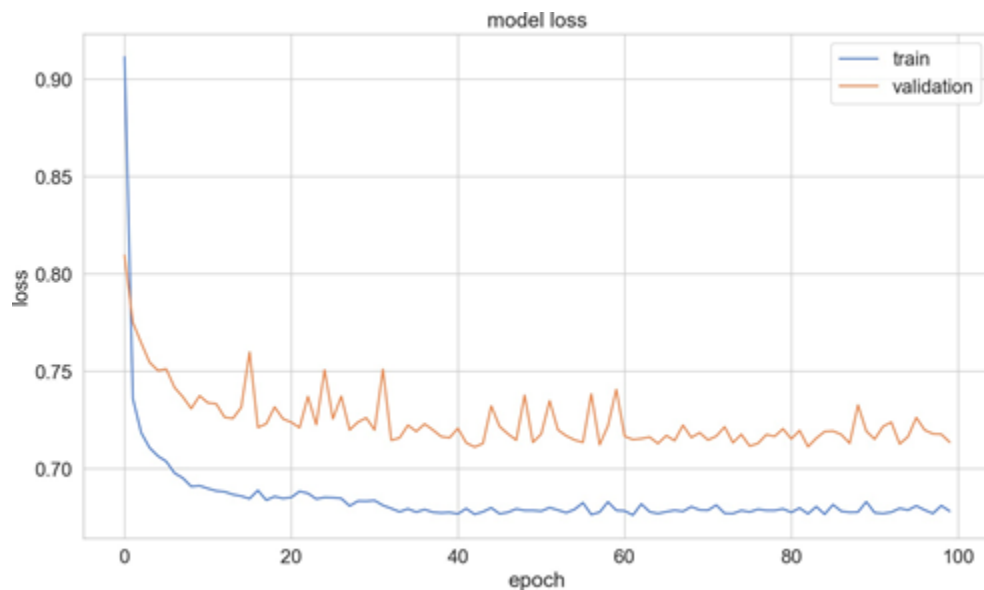
The model has been trained on training and validation sets for 100 epochs setting the batch size to 128 and learning rate to 0.001. The loss function has been plotted using math- plotlib and the system has been evaluated on the testing set using the metrics above. At the end a threshold has been fixed to 2.2 according to the results obtained during the evaluation. At the end we plot the confusion matrix.

As we mentioned above, we want to compare the auto-encoders with Random Forests. In order to increase the performance of this model, we also apply some data augmentation. We have divided the data into training, validation and testing sets as above. Thus we augmented the

training set using the SMOTE function from imblearn. The random forest has been trained and implemented using the standard sklearn library. Still all the evaluation metrics have been used as previously explained. Notice that data augmentation highly increases the performances. In fact, I experimented also with some other techniques like under sampling, but they did not provide the same results.

## Refinement

The implementation required a lot of refinement. I started with a simple net, trying to compress the original input and to extract the most relevant information. I set up an autoencoder with the encoding layer with 14 and 7 neurons. Then I set the batch size from 32 (by default) to 128. Initially I used the entire set of features except "Time". When I removed less important features, the accuracy of the network increased.



*Figure 3.2.1: Loss function of the autoencoder.*

### **3.3 Data set details**

The data set includes credit card transactions made by European cardholders over a period of two days in September 2013. Out of a total of 2,84,807 transactions, 492 were fraudulent. This data set is highly unbalanced, with the positive class (frauds) accounting for 0.172% of the total transactions. The data set has also been modified with Principal Component Analysis (PCA) to maintain confidentiality. Apart from 'time' and 'amount', all the other features (V1, V2, V3, up to V28) are the principal components obtained using PCA. The feature 'time' contains the seconds elapsed between the first transaction in the data set and the subsequent transactions. The feature 'amount' is the transaction amount. The feature 'class' represents class labelling, and it takes the value 1 in cases of fraud and 0 in others.

### **3.4 DL techniques used**

This project made use of Autoencoders. Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Specifically, we'll design a neural network architecture such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input.

And for the comparison purpose we used Random Forest classifier. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

### **3.5 Hardware and Software requirements**

#### **3.5.1 Hardware Requirements:**

- Google Collab with Tensorflow + GPU enabled for training
- As per Google Colab:

- GPU: 1xTesla K80, compute 3.7, having 2496 CUDA cores, 12GB
- GDDR5 VRAM
- CPU: 1x single core hyper threaded Xeon Processors @2.3Ghz i.e (1 core, 2 threads)
- RAM: ~12.6 GB Available
- Disk: ~33 GB Available

### **3.5.2 Software requirements**

Some of the software requirements are as follows:

- Jupyter Notebook
- Python
- scikit-image
- tensorflow utilities

# Chapter 4

## Implementation details of the Project

### 4.1 Language/Tools/APIs used

Language used in this project is the Python programming language.

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language.

Additionally, Python is appealing to many developers as it's easy to learn. Python code is understandable by humans, which makes it easier to build models for machine learning.

Many programmers say that Python is more intuitive than other programming languages. Others point out the many frameworks, libraries, and extensions that simplify the implementation of different functionalities. It's generally accepted that Python is suitable for collaborative implementation when multiple developers are involved. Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes

#### **Tools:**

- imblearn:

To implement SMOTE oversampling, imblearn techniques are the methods by which we can generate a data set that has an equal ratio of classes. The predictive model built on this type of data set would be able to generalize well.

- scikit-image

With machine learning growing at supersonic speed, many Python developers were creating python libraries for machine learning, especially for scientific and analytical computing.

- tensorflow

TensorFlow exposes very stable Python and C++ APIs. It can expose backward compatible APIs for other languages too, but they might be unstable. TensorFlow has a flexible architecture with which it can run on a variety of computational platforms CPUs, GPUs, and TPUs. TPU stands for Tensor processing unit, a hardware chip built around TensorFlow for machine learning and artificial intelligence

- keras

Keras works with neural-network building blocks like layers, objectives, activation functions, and optimizers. Keras also have a bunch of features to work on images and text images that comes handy when writing Deep Neural Network code.

## 4.2 Workflow diagrams

### Workflow of the Model

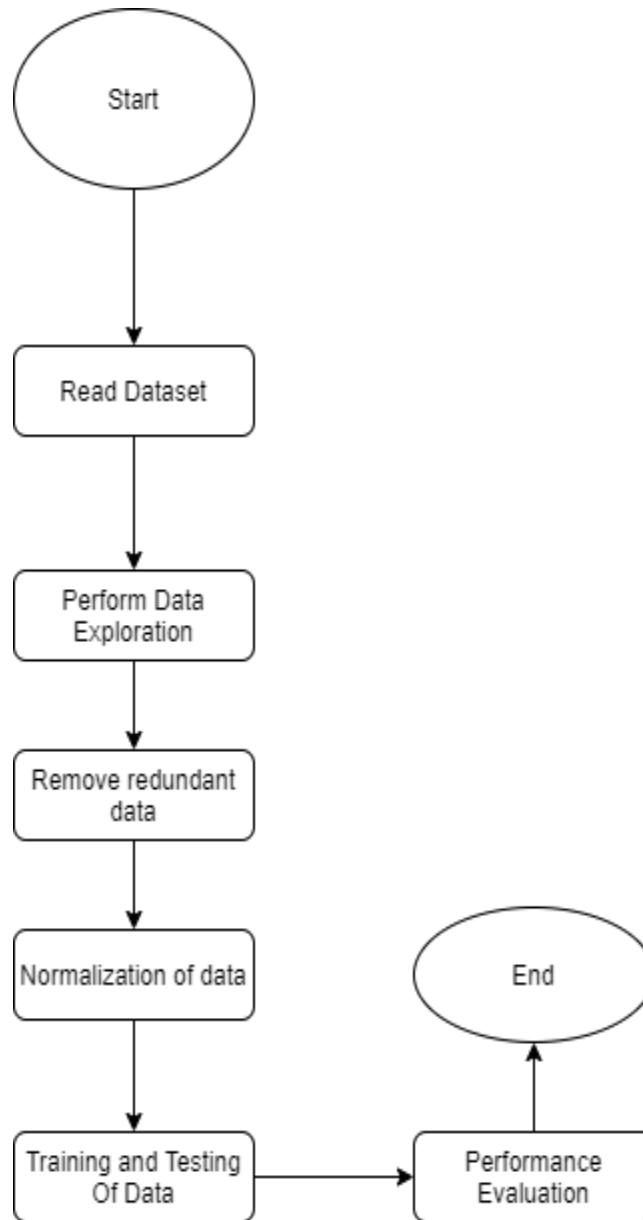


Fig 4.3.1 workflow of Model

## 4.3 Data preprocessing

The first thing to notice is that "Time" does not represent the hour at which each transaction

occurred. This feature only enacts the sequence by which the transaction appeared. This feature is not as representative, thus it will be removed. Most of the data result from the product of a PCA analysis. Due to confidentiality issues, the authors cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA. Notice that "Amount" is the only feature that has not been modified. Hence we normalise this feature in order to scale w.r.t. the others. Now we can judge the importance of each feature from Figure 3. From this visual exploration, we can remove the following features: "V13", "V15", "V22", "V23", "V26" and "Amount".

#### **4.4 Validation methodology**

Precision and Recall are a common metric for this kind of problems; decent values of both help to obtain a precise system that recognizes most of the bad transactions. Let us define TP, FP, TN and FN the number of true-positives, false-positives, true-negatives and false-negatives respectively.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

To solve the problem in the right way, we prefer a high value of recall (meaning that a high number of frauds are detected). We also take into account the ROC curve and confusion matrix, which will help us to evaluate the performances.

The aim of a fraud detection system is to correctly identify bad transactions with a low number of false negatives. We can even accept some normal transaction that is recognised as fraud, but we need to correctly classify bad transactions. Precision and recall will help us to evaluate the behaviour of the system. Particularly the ROC curve will help us to fix the correct values of precision-recall and confusion matrix will be a useful way to measure the results precisely.

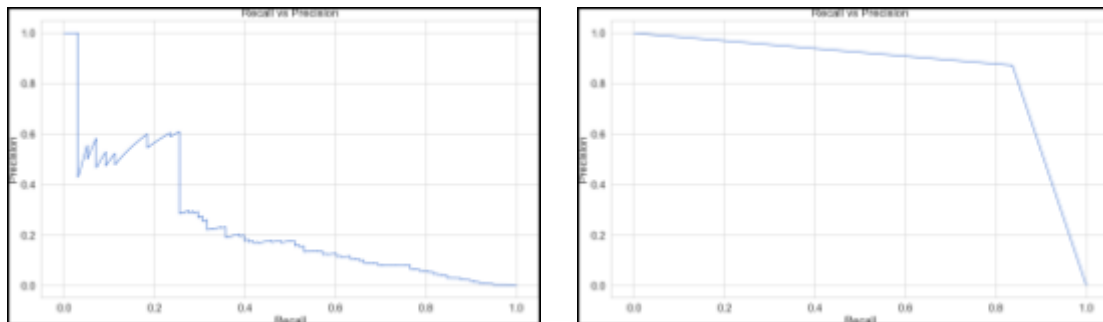


## Chapter 5

### Results and Analysis

#### 5.1 Model Evaluation and Validation

The results achieved with both models are quite interesting. The autoencoder is less precise, but correctly recognizes a good percentage of frauds. The random forest model on the other hand achieves higher results with a better precision. Both models have been trained and validated on a subset of the dataset, while the 20% of the dataset has been reserved for testing. The evaluation of the model has been done on this subset in order to guarantee the model's robustness. Figure 5.1 shows the precision-recall curves for both models.



*Figure 5.1: Precision-Recall curves of the autoencoder (left) and random forest (right).*

The confusion matrix can help us to have a better idea of the results. They both recognise most of the frauds, but the auto-encoder produces a higher number of false positives.

In order to validate the two models and validate their robustness, we can perform a k-fold cross validation. As you can see from Table 3, the random forest perfectly adapts to new inputs, while the auto-encoder has a strong standard deviation, meaning that the model

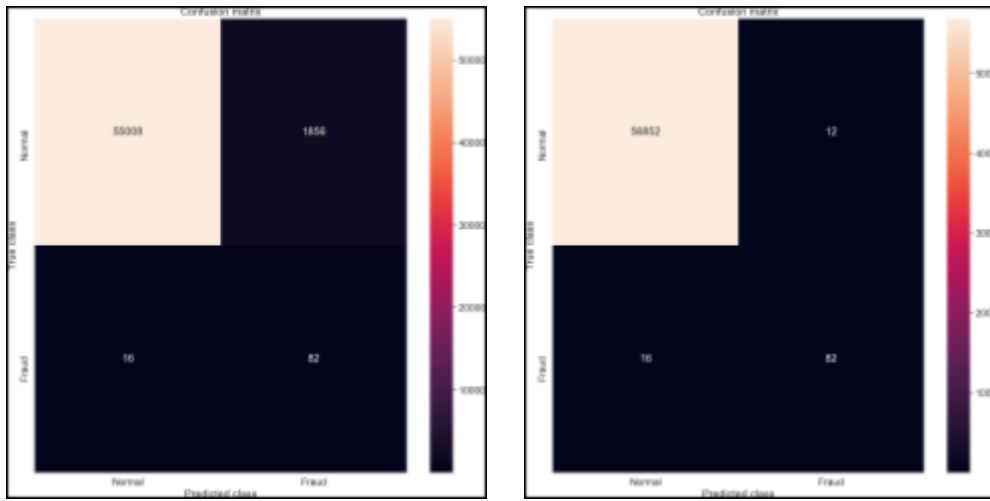


Figure 5.1: Confusion Matrix of auto-encoder(left) and random forest(right)

	Autoencoder	Ran. Forest
Mean Acc.	86.631%	99.888%
Std	(+/- 6.87%)	(+/- 0.00%)

Table 5.1: Mean accuracy and standard deviation after a 10-fold cross validation

## 5.2 Justification

To compare the results with the benchmark values we need to normalise the confusion matrices with percentages.

	Normal	Fraud
Normal	96.620%	3.260%
Fraud	0.028%	0.144%

Table 5.2: Confusion Matrix of the autoencoder [6].

The autoencoder was completely outperformed by both models. The random forest instead, is more precise than model [6]. Both [5] and [6] have a higher number of real frauds that have been detected.

	Normal	Fraud
Normal	99.859%	0.021%
Fraud	0.028%	0.144%

Table 5.3: Confusion Matrix of the random forest [6].

## **Chapter 6**

### **Conclusion and Future enhancement**

#### **6.1 Novelty in proposed solution**

From the results above we can conclude that both models learned how to recognise frauds. The graphs also help us to have an idea of the balance of the true-positives/true-negatives rate of autoencoder and random forest. Fraud detection is an important and sensitive task. It requires techniques that automatically recognise anomalies with low error rates. In fact, those systems need to correctly identify all the bad transactions, possibly with few false positives. The approach that we proposed in this project correctly solves the problem, even though they should still be improved. As expected the autoencoders are not the best model to solve this kind of problem. The network obviously learns, but is not as accurate as other models. Random Forests instead, is a better choice for anomaly detection problems but it is important to balance the model to obtain decent results, thus it requires a bit of pre- processing steps.

#### **6.2 Limitation**

Fraud detection is an important and sensitive task. It requires techniques that automatically recognise anomalies with low error rates. In fact, those systems need to correctly identify all the bad transactions, possibly with few false positives. The approach that we proposed in this project correctly solves the problem, even though they should still be improved. As expected the autoencoders are not the best model to solve this kind of problem. The network obviously learns, but is not as accurate as other models. Random Forests instead, are a better choice for anomaly detection problems but it is important to balance the model to obtain decent results, thus it requires a bit of pre-processing steps.

#### **6.3 Future enhancement**

The aim of this project was to compare a "standard" model as random forests with an alternative

approach (autoencoders). We did not expect the autoencoder to outperform the most established techniques. Further improvements can be obtained using alternative techniques. Random Forest model can be even improved with some other experimentation. However, we do expect to see some improvement with a combined approach for balancing the dataset using SMOTE+ENN.

## **6.4 Summary**

A lot of research has been done to find a solution to this problem using Autoencoders and Random Forest algorithms. However, Random Forest gives better results as compared to other techniques, but it is highly imperative that the model is balanced.

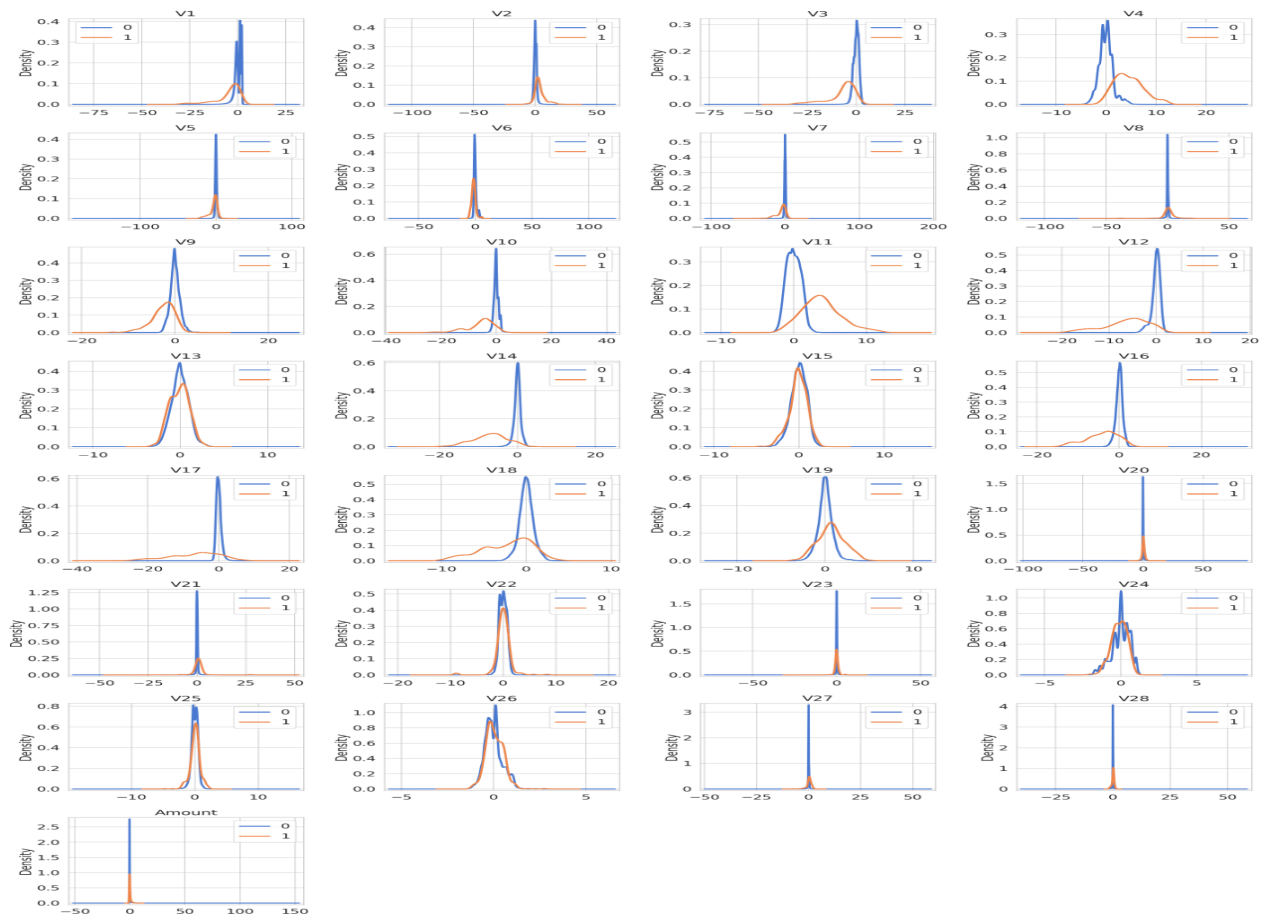
## References

- [1] Varmedja, Dejan & Karanovic, Mirjana & Sladojevic, Srdjan & Arsenovic, Marko & Anderla, Andras. (2019). *Credit Card Fraud Detection - Machine Learning methods*. 1-5.
- [2] Chouiekh, Alae & Haj, EL. (2018). *ConvNets for Fraud Detection analysis*. *Procedia Computer Science*. 127. .
- [3] More, Rashmi & Awati, Chetan & Shirgave, Suresh & Deshmukh, Rashmi & Patil, Sonam. (2021). *Credit Card Fraud Detection Using Supervised Learning Approach*. *International Journal of Scientific & Technology Research*.
- [4] Bhavya, L. & Reddy, V. & Mohan, U. & Karishma, S.. (2020). *Credit Card Fraud Detection using Classification, Unsupervised, Neural Networks Models*. *International Journal of Engineering Research and*. V9.
- [5] Mr.Manohar.s, Arvind Bedi, Shashank kumar3, Shounak kr Singh. *Fraud Detection in Credit Card using Machine Learning Techniques*. *International Research Journal of Engineering and Technology (IRJET)*
- [6] Tuyls, Karl & Maes, Sam & Vanschoenwinkel, B.. (2021). *Machine Learning Techniques for Fraud Detection*.
- [7] Dornadula, Vaishnavi & Geetha, S. (2019). *Credit Card Fraud Detection using Machine Learning Algorithms*. *Procedia Computer Science*. 165.
- [8] Varmedja, Dejan & Karanovic, Mirjana & Sladojevic, Srdjan & Arsenovic, Marko & Anderla, Andras. (2019). *Credit Card Fraud Detection - Machine Learning methods*. 1-5.
- [9] S P, Maniraj & Saini, Aditya & Ahmed, Shadab & Sarkar, Swarna. (2019). *Credit Card Fraud Detection using Machine Learning and Data Science*. *International Journal of Engineering Research and*. 08. .
- [10] Sahin, Yusuf & Duman, Ekrem. (2011). *Detecting credit card fraud by ANN and logistic regression*. *INISTA 2011 - 2011 International Symposium on INnovations in Intelligent SysTems and Applications*. .

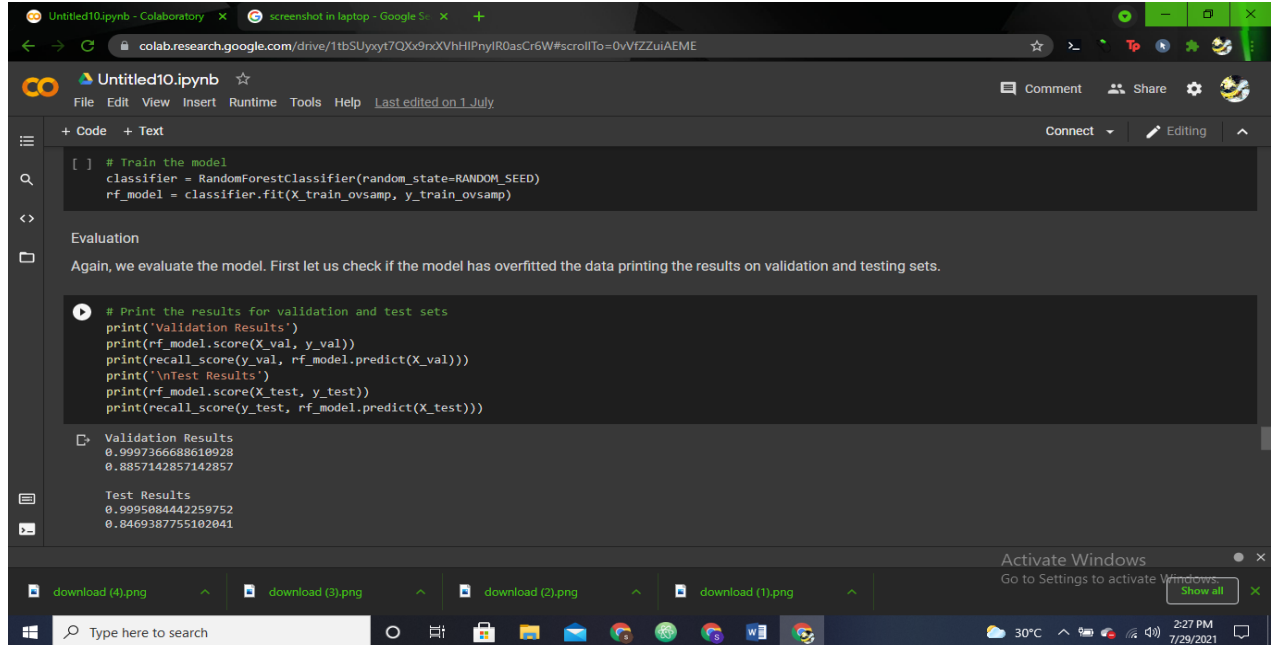
# Appendix 1

## Screenshots

### Data Analysis



## Random Forest Classifier Result



The screenshot shows a Jupyter Notebook titled 'Untitled10.ipynb' in a Google Colaboratory environment. The notebook contains two code cells. The first cell trains a Random Forest Classifier. The second cell evaluates the model, printing validation and test results. The output shows high performance with scores near 1.0.

```
[ ] # Train the model
classifier = RandomForestClassifier(random_state=RANDOM_SEED)
rf_model = classifier.fit(X_train_ovsamp, y_train_ovsamp)
```

Evaluation

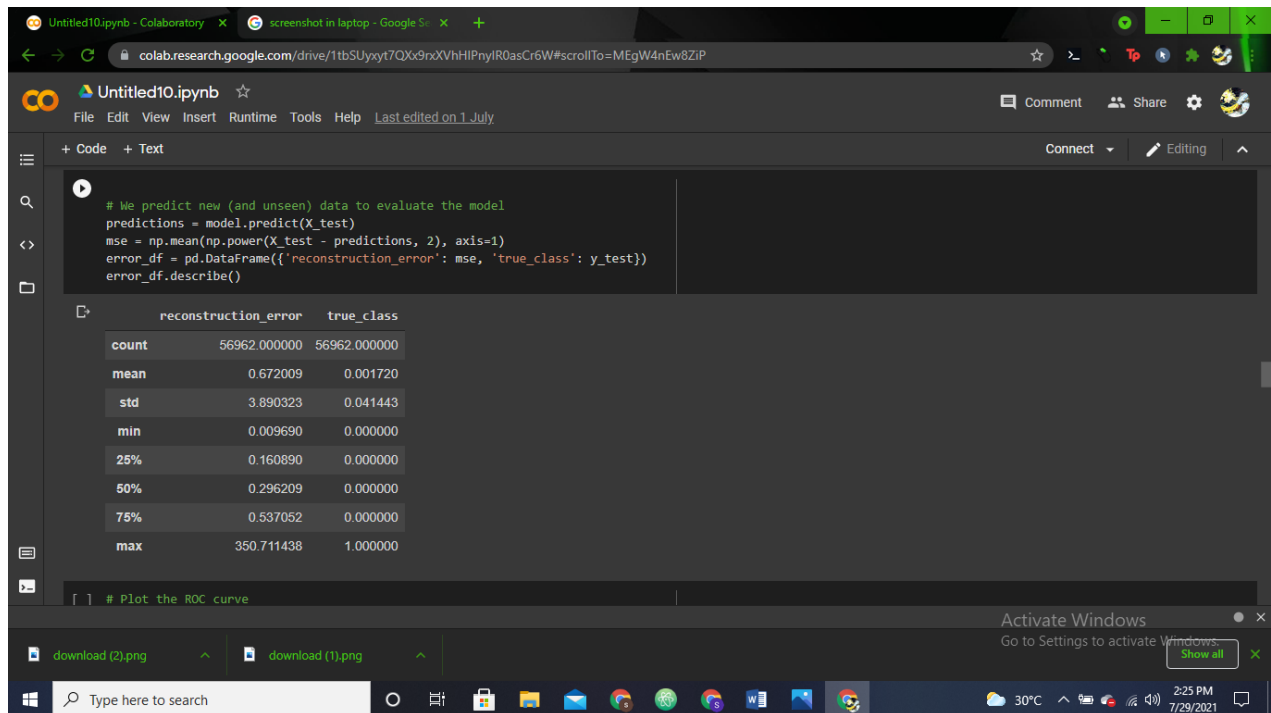
Again, we evaluate the model. First let us check if the model has overfitted the data printing the results on validation and testing sets.

```
[ ] # Print the results for validation and test sets
print('Validation Results')
print(rf_model.score(X_val, y_val))
print(recall_score(y_val, rf_model.predict(X_val)))
print('\nTest Results')
print(rf_model.score(X_test, y_test))
print(recall_score(y_test, rf_model.predict(X_test)))
```

Validation Results  
0.999736688610928  
0.8857142857142857

Test Results  
0.9995884442259752  
0.8469387755102041

## Reconstruction Error in Autoencoder Model



The screenshot shows a Jupyter Notebook titled 'Untitled10.ipynb' in a Google Colaboratory environment. The notebook contains two code cells. The first cell calculates the Mean Squared Error (MSE) for the reconstruction of test data. The second cell prints a summary of the reconstruction error and true class values.

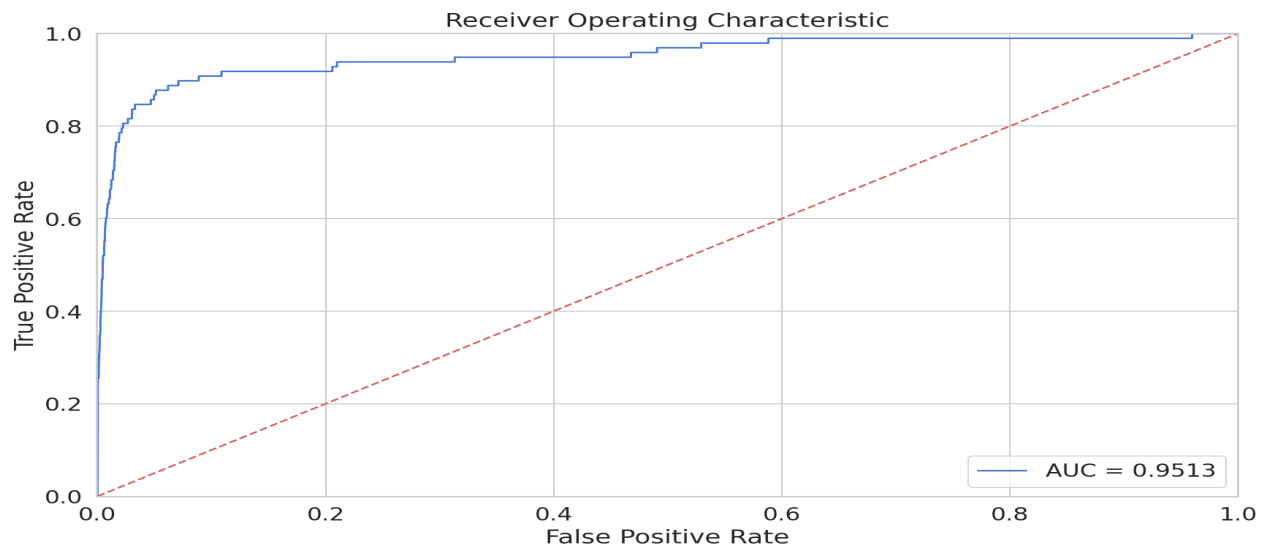
```
[ ] # We predict new (and unseen) data to evaluate the model
predictions = model.predict(X_test)
mse = np.mean(np.power(X_test - predictions, 2), axis=1)
error_df = pd.DataFrame({'reconstruction_error': mse, 'true_class': y_test})
error_df.describe()
```

	reconstruction_error	true_class
count	56962.000000	56962.000000
mean	0.672009	0.001720
std	3.890323	0.041443
min	0.009690	0.000000
25%	0.160890	0.000000
50%	0.296209	0.000000
75%	0.537052	0.000000
max	350.711438	1.000000

```
[ ] # Plot the ROC curve
```



## AUC characteristic for Autoencoder



## Precision vs Different thresholds for Autoencoder

