# Predictive Analytics for Controlling Tax Evasion Using Deep Learning: The Case of Ethiopia Customs Commission Taxpayers.

**Tibeb Wondimu Lemma**

**A Thesis Submitted to the Department of Computer Science and Engineering**
**School of Electrical Engineering and Computing**

**Office of Graduate Studies**
**Adama Science and Technology University**

**September, 2024**
**Adama, Ethiopia**

# Predictive Analytics for Controlling Tax Evasion Using Deep Learning: The Case of Ethiopia Customs Commission Taxpayers.

**Tibeb Wondimu Lemma**

**Advisor: Dr. Teklu Urgesa**

**A Thesis Submitted to the Department of Computer Science and Engineering School of Electrical Engineering and Computing**

**Office of Graduate Studies**

**Adama Science and Technology University**

**September, 2024**

**Adama, Ethiopia**

# DECLARATION

I declare that this master thesis entitled **"Predictive Analytics for Controlling Tax Evasion using Deep Learning: The case of Ethiopia Customs Commission Taxpayers."** is my own work and has not been submitted to any university for similar purposes. The references used in this thesis are duly recognized by proper citations.

**TibebWondimu Lemma** _____ _____

Name of Student Signature Date

# RECOMMENDATION OF ADVISORS

I, the major advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled **"Predictive Analytics for Controlling Tax Evasion using Deep Learning: The Case of Ethiopia Customs Commission Taxpayers."** prepared under my guidance by **Tibeb Wondimu** submitted in partial fulfilment of the requirements for the degree of masters of science in computer science and Engineering. Therefore, I recommend the submission of the revised version of the thesis to the department following the applicable procedures.

**Dr.Teklu Urgesa** _____ _____

Major Advisor Signature Date

# APPROVAL PAGE

I, the advisor of the master thesis entitled **"Predictive Analytics for Controlling Tax Evasion using Deep Learning: The Case of Ethiopia Customs Commission Taxpayers."** and developed by Tibeb Wondimu, hereby certifying that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

| _____ | _____ | _____ |
|---|---|---|
| Major Advisor | Signature | Date |

We, the undersigned, members of the Board of Examiners of the thesis by **Tibeb Wondimu** have read and evaluated the thesis entitled **"Predictive Analytics for Controlling Tax Evasion using Deep Learning: The Case of Ethiopia Customs Commission Taxpayers."** and examined the candidate during the open defence. This is, therefore, to certify that the thesis is accepted for partial fulfilment of the requirement of the degree of Master of Science in Computer science and Engineering.

| _____ | _____ | _____ |
|---|---|---|
| Chairperson | Signature | Date |
| _____ | _____ | _____ |
| Internal Examiner | Signature | Date |
| _____ | _____ | _____ |
| External Examiner | Signature | Date |

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

| _____ | _____ | _____ |
|---|---|---|
| Department Head | Signature | Date |
| _____ | _____ | _____ |
| School Dean | Signature | Date |
| _____ | _____ | _____ |
| Office of Postgraduate Studies, Dean | Signature | Date |

# ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to the Almighty God, St. Mary Virgin, and St. Gabriel for granting me strength, peace of mind, and good health throughout my journey. I extend my deepest appreciation to my advisor, Dr. Tekilu Urgesa, for his unwavering support, invaluable feedback, and constructive guidance during the completion of this thesis. Special thanks are due to Ms. Nardos Mengesha and her colleague from the Ethiopia customs commission head office post-clearance audit unit for their insightful lectures and discussions on the manual system, as well as for sharing essential documents. I am also grateful to Mr. Getu Alemu, the General Manager of Modjo branch Customs Commission, for his recommendations to the head office manager, which facilitated my access to the necessary data and support for my research.

I would like to acknowledge Mr. Mulu Likesa, Mr. Demena Dirba, and Mr. Milkesa, unit leaders at my workplace which is Modjo branch customs commission, for granting me permission to conduct my study and collect data.

Finally, I wish to express my profound gratitude to my family, friends, and all those who have offered their advice and support throughout this thesis.

# LIST ACRONYMS

**AI**: Artificial intelligence

**ANN**: Artificial neural networks

**ARX**: Auto regression

**ATAF**: African Tax Administration Forum

**BRB-DL**: Belief rule based with deep learning

**BRBES**: Belief Rule Based Expert System ·

**CIF**: cost insurance freight

**CNN**: Convolutional neural networks

**COD**: Coefficient of determination

**DAE**: DE Noise AutoEncoders

**DBN**: Deep Belief Networks

**DL**: Deep Learning

**ECMS**: Ethiopia customs commission system

**EU**: European Union

**FFNN**: Feed forward neural networks

**GC**: Grigourian calander

**GP**: Gaussian process

**KNN**: K nearest neighbour

**LSTM:** Long short term memory

**LSTM**-VAE: Long short memory with variational AutoEncoder

**MAP**: Mean Average Precision

**MB**: Mean Bias

**MRR**: Mean reciprocal rank

**MSE**: Mean squared error

**NRMSE**: Normalized Root Mean Square Error

**PCA**: post clearance audit

**RBM**: Constrained Boltzmann Machine

**RFE**: Feature elimination

**RMSE**: Root mean squared error

**RNN**: Recurrent neural network

**SDAE**: Stacked De noising AutoEncoder

**TIN**: Taxpayer identification number

**UN**: United nation

**VAE**: Virational AutoEncoder

**WCO**: World customs organization

# LIST OF TABLES

# LIST OF FIGURES

# TABLE CONTENTS

# ABSTRACT

Tax evasion is an illegal evasion by a person or a company. This is the intentional misrepresentation of a tax return in order to reduce your liability in tax. Tax evasion represents one of the big dilemmas of governments and tax authorities around the world, but most especially in developing countries. Tax authorities control tax evasion through a system based on Tax audits. Tax audit selection is based on audit selection criteria, auditor experience, and a non-adaptive randomization selection manual system. The process therefore faces subjective judgments, limited data utilization, and susceptibility to corruption-all these ultimately cause tax evasion. Herein, we present predictive analytics for controlling tax evasion using deep learning. In this study, raw data is collected from three sources: Customs Commission (138,644 data), domestic (15,070 data) and post-clearance audit history (2,686 data) over 5 years in the Ethiopia Customs Commission taxpayers. Using domain knowledge we integrated and pre-processed data. The final pre-processed dataset is the 15070 dataset; then we split the dataset to 80% training dataset(12056), 10%test datasets(1507), and 10% validation datasets(1507).The data have been trained using deep neural networks, AutoEncoder, and feed-forward neural networks then we used two hyperparameter tuning techniques random search CV and Bayesian optimize; and transfer learning to accurate predictions. We have predicted five consecutive annual selections for a tax audit and three highest tax evasion conditions. We Model then evaluated its mean bias, mean squared error, root mean squared error, R-squared, explained variance score, and training and validation loss comparing two hyperparameter tuning technique. We concluded that the random search CV hyperparameter tuning technique better by reducing error. We also determined using random search CV the feed-forward neural network the best model, with a mean bias of -0.0009, mean squared error of 0.0000, and root mean squared error of 0.0038. The best performance yielded an R-squared of 99.98%, explained variance core of 99.98%, and accurate prediction of tax audit selection and three top tax evasion cases over 5 consecutive years. Predictive analysis of each variable is done sequentially but feature researcher uses any parallel processing system to minimize analysis time and integrated it with behavioural model.

**Keywords:** Predictive analytics, Tax evasion, Customs Commission, Deep learning, Random search CV, Bayesian optimizer and Transfer learning

# CHAPTER ONE
# INTRODUCTION

## 1.1 Background of the Study

Taxation can be thought of, in general terms, as a compulsory charge or other levy imposed on a taxpayer by a form of government organization. In that context, the funds collected are utilized for government spending and collective public expenditures, or to redistribute and correct market failures, including negative externalities. There are mostly two kinds of taxes: direct and indirect. The main difference between them is in their implementation. Direct taxes are labelled on individuals and corporate entities and cannot be transferred to others. It mainly comprises income tax, wealth tax, and gift tax, while indirect taxes are labelled on goods and services (Kumar, 2018). In this context, we focus on the direct taxation system specifically income import taxes. However, some taxpayers often try to avoid paying the correct taxes. Tax evasion is illegal evasion by an individual or company. This is a deliberate act of incorrectly showing and declaring tax returns aimed at reducing your tax liability. Tax evasion is a major problem for governments and tax authorities around the world, especially in developing countries (W. Fox et al, 2017).

Tax authorities control tax evasion through a system based on Tax audits. Tax audits are an important part of tax collection as the main process to correct errors, increase compliance, and ensure a fair taxation system (Alm J, 1993). That being said, two organizations control evasion by labelled domestic tax called Revenue Post Clearance Audit or international trade tax evasion through the Customs Commission known as the import-export post clearance audit. This thesis focus on Ethiopia customs commission import taxpayers. The post-clearance audit process usually starts with taxpayer audit selection through many risk-based audit selection criteria; after that, an experienced auditor reviews the selection criteria to identify which returns shall be audited([1], [2], [3], [4]) The selection process of the taxpayers to be audited by tax agencies was a manual system, hence requiring subjective judgments, limited data utilizations, and susceptibility to corruption causes of tax evasion.

1. WCO GUIDELINE VOL2, 2011,
2. GUIDELINES FOR POST-CLEARANCE AUDIT (PCA) VOLUME 1, 2018,
3. የድህረ ዕቃ አወጣጥ የመስክ አዲት የመምረጫ መስፈርት የአሰራር ሰነድ, 2018,
4. የጉምሩክ ቅ/ጽ/ቤቶች ስጋት ሥራ አመራር የአሠራር ማኑዋል ሁለተኛ ዕትም, 2017

To address tax evasion issues, predictive analytics are used to control tax evasion using deep learning. Predictive analytics has lately been extensively used in companies and other entities looking for ways of making better decisions and achieving better results by leveraging datasets, advances in computational power, and a new, emerging class of analysts capable of extracting knowledge from the ocean of data and information that is currently at our fingertips. Revenue and customs pursue the use of predictive analytics in pursuits because it recognizes what value can be derived from such an approach (Davenport & Harris, 2007; Miller et al., 2006; Davenport & Harris, 2010, Cleary, 2011). Predictive analytics aims to anticipate tax fraud so that tax auditors can take appropriate preventative measures (CPI, 2018). The use of predictive analytics in tax audits not only uses various types of technical data analysis to determine the classification of taxpayers based on specific criteria but also determines which tax cases or taxpayers are more suitable for tax policy purposes. It informs decision-making in the audit selection (Hashim Zade et al., 2016). Demonstrating predictive analytics that can revolutionize tax authorities and decision-making processes (Rustagi & Goel, 2022). Deep learning is a form of machine learning, the use of data to train a model to make predictions from new data. It has shown significant potential in approximating and reducing large, complex datasets into highly accurate predictive and transformational output, greatly facilitating human-centered systems (Chen & Lin, 2014; Hatcher & Yu, 2018). The potential of applying deep learning in auditing is already visible for various risk-based audit procedures (Sun and Vasarhelyi 2017). The recent studies demonstrated the capabilities of deep learning in different audit tasks: - anomaly detection (Schreyer et al. 2017; Schultz and Tropmann-Frick 2020), audit sampling (Schreyer, Sattarov, Gierbl, and Reimer 2020), and financial report analysis (Deußer et al. 2022; Hillebrand et al. 2022a, 2022b). Audit firms are gathering their first empirical experiences when utilizing deep learning in real-world audit settings (Brenner, Van Giffen, and Koehler 2020; Lars Föhr et al. 2023). Deep learning model architectures are also associated with high applicability for risk-based audit approaches (Gierbl et al. 2021; Sun and Vasarhelyi 2017; Lars Föhr et al. 2023). We focus on the application of deep learning in audit sampling or audit target audit selection using predictive analytics to control tax evasion.

With this goal in mind, we review different related work for controlling tax evasion: - Predictive analytics as a service on tax evasion uses feature engineering strategies (Kishore Babu & Vasavi, 2019). This focuses on property tax. The method used is recursive feature elimination (RFE) with a random forest algorithm for feature engineering, and ensemble models combining artificial neural networks (ANN), auto-regression (ARX), and Gaussian

process (GP) for predictive modelling. Finalize by proving feed-forward back propagation neural network better in terms of accuracy than regression. Neural network Normalized Root Mean Square Error (NRMSE) 0.0113 and Coefficient of determination (COD) 0.99. In this research only determine Normalized Root Mean Square Error (NRMSE) and Coefficient of determination (COD) to judge the performance of different models. Deep neural network and time series approach for finance systems: Predicting the movement of the Indian stock market (Srivastava et al., 2021) using Historical data of NIFTY 50 is collected and converted into a time series format and Compared support vector machine, random forest, gradient boosting, and deep neural networks. Deep Neural Network performed better over another model with an accuracy of 98.7% achieved with 99.5% precision and 93% recalls. In this model, the prediction performance varies due to stock sales and the model using fewer datasets. Predictive Analytics for Controlling Tax Evasion (Kumar, 2018)this research is focusing on indirect tax which is good and services tax. By formulating the problem of detecting circular trading fraud as finding clusters in a weighted directed graph, where the edge weights were defined using Benford's analysis. A cut-off equal to 0.5 is 86.38% and testing accuracy is 86.33%. The precision of the model is 85.5%. Recall of the model is 97.97%. This research has a time complexity issue. Audit lead selection and yield prediction from historical tax data using artificial neural networks (Chan et al., 2022)achieved a 40.1% precision and 58.7% recall (F1-score of 0.42) on classifying positive audit leads. This research used simple computational methods that under fit in terms of model complexity are usually based on empirical rules that have not been statistically validated and do not learn from historical data and audit outcomes. A deep learning-inspired belief rule-based expert system (Ul Islam et al., 2020)compares the long-short memory, Deep Neural Network, and Deep Learning (DL) method with the Belief Rule-Based Expert System (BRBES). BRB-DL outperforms other methods like Long-Short Term Memory, Deep Neural Network, Fuzzy Deep Neural Network, Adaptive Neuro Fuzzy Inference System, and traditional BRBES in terms of prediction accuracy, e.g. achieving a Mean Square Error of 4.12 compared to 18.66, 28.49, 17.05, 16.37, and 38.15 respectively for a combined cycle power plant dataset. In this research, only Mean Square Error is compared to decide which model is out preformed.

our thesis predictive analytics for controlling tax evasion using deep learning develops by automating risk-based data pre-processing to handle the black box nature of deep learning and compare from deep learning model deep neural network and AutoEncoder, and from the artificial neural network feed-forward neural network to fit Predictive analytics. Using predictive analytics powered by deep learning, tax authorities can identify patterns that

indicate tax evasion and take proactive steps to prevent it. So, using deep learning with predictive analytics to control tax evasion is a Promising solution that can analyse vast amounts of data to uncover hidden patterns and trends. By implementing these techniques, tax authorities are more likely to detect and prevent tax evasion, ultimately leading to higher incomes, a fair and equitable tax system enhanced economic development, and strengthened governance and transparency.

## 1.2 Statement of problem

Tax evasion generally refers to the illegal evasion of paying taxes in whole or in part, and this is usually done through deliberate failure to declare or false declarations to tax authorities. It includes understating income, profits, and gains earned and/or overstating deductions. Tax evasion poses a significant challenge for governments and tax authorities around the world (W. Fox et al, 2017). Globally, tax evasion remains a significant issue, with estimates suggesting that around $500 billion is lost annually due to tax evasion practices. The Global Tax Evasion Report 2024 by the EU Tax Observatory indicates that while there have been efforts to combat tax evasion, approximately 25% of global offshore financial wealth remains untaxed, highlighting the on-going challenges in addressing this issue (EU Tax Observatory, 2023). In Africa, tax evasion is a critical concern that affects revenue collection significantly. The African Tax Administration Forum (ATAF) estimates that the continent loses about $50 billion annually due to tax evasion. This loss severely hampers the ability of African governments to finance development projects and improve public services. The report emphasizes the need for stronger tax compliance measures and better enforcement to address these challenges (Social Europe, 2023). In Ethiopia faces substantial challenges with tax evasion, particularly within its revenue and customs authority. A study focusing on the Ethiopian Revenue and Customs Authority indicates that tax evasion is prevalent, with significant methods including non-declaration of income and underreporting of business expenses. The study found that tax evasion and avoidance are significant barriers to effective tax collection in Ethiopia, with estimates suggesting that tax evasion could account for up to 33.3% of potential tax revenue in some sectors (Worku Mekonnen, 2021).

Tax evasion happens differently, we focus on manual audit selection processes which face Subjective Selection Criteria: In manual audit selection, auditors often rely on their discretion and judgment to select audit cases. This subjectivity can lead to inconsistencies in how cases are chosen. Taxpayers may exploit this by presenting themselves as low-risk candidates, thus

avoiding audits. For instance, if an auditor is less familiar with certain industries or taxpayer behaviours, they may overlook potential red flags, allowing tax evasion to persist (Damissie Tolossa, 2023). Limited Data Utilization: Manual audit processes may not effectively utilize available data to assess taxpayer risk. Without comprehensive data analysis, auditors may miss critical indicators of non-compliance. Corruption and Collusion: In some cases, manual audit processes can be susceptible to corruption. If auditors are bribed or colluded with taxpayers, it undermines the integrity of the audit process. Taxpayers may feel emboldened to evade taxes if they believe they can avoid detection through corrupt practices (Damissie Tolossa, 2023). All those results in revenue loss: -Tax evasion significantly reduces government revenue, which can have far-reaching consequences for public services and infrastructure development. According to the OECD, tax evasion leads to substantial losses in tax revenues, limiting the government's ability to fund essential programs such as education, healthcare, and social welfare. This shortfall can adversely affect the well-being of citizens, leading to inadequate public services and diminished quality of life (OECD, 2021), Unfairness and Inequality: - Tax evasion creates an imbalanced tax system where some individuals and businesses evade their responsibilities while others comply. This unfairness exacerbates social inequality, as those who evade taxes contribute less to the communal pot that funds public services. Research from the Institute on Taxation and Economic Policy (ITEP) indicates that honest taxpayers may feel disillusioned and frustrated, as they bear a disproportionate burden of funding essential services compared to tax evaders (ITEP, 2020). This sense of injustice can erode public trust in the tax system and government institutions. Market Distortions Businesses that engage in tax evasion can gain an unfair competitive advantage over their law-abiding counterparts. This advantage distorts market dynamics, undermining fair competition and potentially leading to economic inefficiencies. According to a report by the World Bank, tax evasion can result in market imbalances, business closures, and reduced innovation, as compliant businesses struggle to compete (World Bank, 2019), Weakened Governance: - Tax evasion undermines good governance and the rule of law. It erodes trust in government institutions and can foster an environment conducive to corruption and illicit financial activities. A study by the United Nations highlights that addressing tax evasion is crucial for strengthening governance and promoting transparency in financial practices (UN, 2021). A robust tax system is essential for maintaining public confidence and ensuring that government functions effectively.

our thesis to overcome the tax evasion problem developed predictive analytics for controlling tax evasion by integrating company rule to manage the black box nature of deep learning;

using deep learning models to analyse complex and large data, and using predictive analytics to predict taxpayers to be audit for five consecutive years by using transfer learning. So using deep learning with predictive analytics to control tax evasion can analyse vast amounts of data to uncover hidden patterns and trends. After all, reducing the problem, the reality became increased government revenue, fair and equitable tax system, enhanced economic development, and strengthened governance and transparency.

## 1.3 Research question

The research questions are:

1. Which deep learning model can best suit predicting patterns and anomalies of tax evasion in a large dataset and compare with traditional predictive models for five consecutive years?
2. Which Hyperparameter tuning technique is better optimizing the models?

## 1.4 Objective

### 1.4.1 General objective

The general objective of the thesis is predictive analytics for controlling tax evasion using deep learning: The case of the Ethiopia Customs Commission taxpayers.

### 1.4.2 Specific objective

The following specific objective to achieve the above general objective: -

- To review post clearance audit selection criteria and customs commission risk analysis.
- To prepare the data according to post clearance audit selection criteria
- To pre-processed the data by handling missing value, outliers, and inconsistence of data.
- To design Predictive analytics models using different deep learning techniques to predict tax evasion and traditional predictive analytics models to predict for five consecutive years using transfer learning.
- To optimize the model using hyperparameter tuning techniques
- To evaluate the performance of the deep learning model in predicting tax evasion. This involves assessing the model's mean square error, root mean square error, r-squared, and mean bias.
- To Compare different model evaluation results

- To propose recommendations on how deep learning-based models can be used to boost the effectiveness of existing tax administration measures. This involves exploring how the model can be integrated into existing tax systems and identifying areas where further research is needed.

## 1.5 Significance of the study

Predictive analytics and deep learning can be extremely useful in controlling tax evasion. The significance of study are early detection of tax evasion, improved accuracy in tax assessments, reduce fraud and errors, fair and equitable tax system and faster processing. All these benefits contribute to more reliable tax collection and ultimately contribute to strengthening national economies.

## 1.6 Scope and limitation

### 1.6.1 Scope

In this research, predictive analytics for controlling tax evasion in the case of Ethiopia customs commission. Currently there is a lot of Customs Commission all over the world. There are also different audit selection criteria for different country. For this research work, Ethiopia customs commission is selected. We focus on direct tax evasion specifically due to import taxpayers. Tax evasion happens differently; we focus on manual audit selection processes that face Subjective Selection, limited utilization of data, and susceptibility to corruption ultimately causing tax evasion. To conduct the objectives, deep learning techniques are compared with the classical model of artificial neural networks. We selected deep neural network, AutoEncoder, and feed-forward neural networks then used for predictive analytics.

### 1.6.2 Limitation.

- The study focuses only on tax evasion due to import taxpayer audit selection.
- Due to time constraints, the findings of this study have not been experimented with using new data.

## 1.7 Thesis Organization

The organization of the study is the overall content of the study which was conducted to develop predictive analytics for controlling tax evasion: The case of Ethiopia customs commission taxpayers.

**Chapter One:** This chapter of the study includes an introduction to the background of the study and discusses a general overview of the title, and statement of the problem by answering what is problem of tax evasion is. When this problem begins to import taxpayers? Why tax evasion is the problem? What predictive analytic and deep learning solution to the problem? Finalize what reality becomes after the solution, the objective of the study, research question, significance of study, scope and limitation, and thesis organization.

**Chapter Two**: in this chapter literature review discussed theoretical reviews and related work of study. A theoretical review helps us to get a supportive idea related to predictive analytics, deep learning, tax evasion, and another used to model and optimizes deep learning. Related work is to analyse the existing system to gain a further understanding of the study area and to indicate what other researchers have already done to bear their limitations to the present study.

**Chapter Three:** In this chapter research methodology and material are discussed. These include an introduction, research objectives, data collections, data understanding, data preparations, data pre-processing, modelling, evaluation, a summary of methodology and material used based on customs commission risk analysis, time series analysis, predictive analytics, and deep learning.

**Chapter Four**: in this chapter the proposed solution and implementation. Discuss proposed model work and ethical consideration.

**Chapter Five:** In this chapter's results and discussion. Then obtained result from deep learning model and comparing it. Discuss the results of different models. Then perform hyperparameter tuning using random search CV and Bayesian optimizer.

**Chapter six:** In this chapter conclusion, research contribution, and recommendation. The conclusion of the study should be summarized and report clearly how all the research questions are answered, and the objective is achieved. The research contribution is precisely the contribution of this thesis. The recommendation part gives information for the user of this study and what the other researcher must include to enhance this study is mentioned.

# CHAPTER TWO
# LITERATURE REVIEWS

## 2.1 Theoretical reviews

**2.1.1 Predictive Analytics:** is a form of data analysis that makes use of statistical techniques and machine learning algorithms in parsing data, identifying predictive patterns, and forecasting future events or trends. The approach analyses past data for patterns and correlations upon which forecasts regarding future trends or occurrences are made (Linda Tucci, 2021).

**2.1.2 Why Predictive Analysis**

Organizations seek help from predictive analytics to solve problems by (Rustagi & Goel, 2022): -

• Identifying fraud: Combining several analytical methods can refine pattern detection and prevent criminal behaviour.

• Optimizing trading campaigns: Predictive analysis is used to examine customer responses, create new business models, and promote opportunities. Models may help businesses attract, retain, and grow their most profitable customers.

• Enhancement in operations. Most industries use predictive models to predict inventory and management of resources.

**2.1.3 Process of predictive analytics**

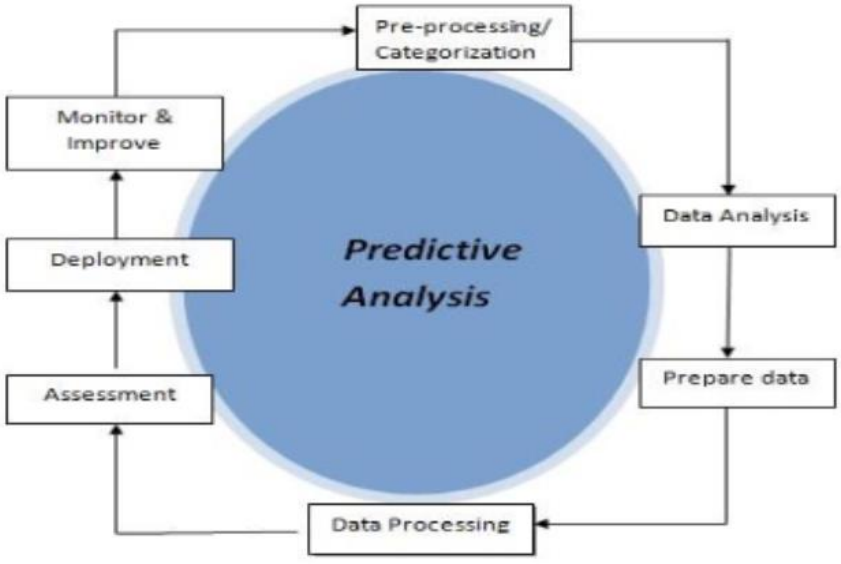


Figure 2-1 process of predictive analytics

Process of predictive analytics: -

**Pre-Processing (Figure 2-1):** Forecast by determining the previous and present performance and studying the historical data.

**Data Analysis (Figure 2-1):** By this, we determine what has happened and why. Prepare Data: Identify the data resources. Data could be in different formats and how we can use the data. Evaluate the quality of the data.

**Data Processing (Figure 2-1):** Based on a study of datasets, best-fit modelling techniques have been applied such as regression, pattern recognition, K nearest neighbour (KNN), support vector machines, etc.

**Assessment (Figure 2-1):** is process includes a graphical representation of data, reports for comparative study, and analysis.

**Deployment (Figure 2-1):** It helps in decision-making in day-to-day processes. It is a time-dependent and delayed process that comes across with a lot of challenges.

**Monitor and Improve (Figure 2-1):** Determines what is happening now and necessary measures to be taken for the enhancement

### 2.1.4 Deep learning

Deep learning is a subset of machine learning, a field dedicated to the research and development of learnable machines (to sometimes reach general artificial intelligence). In industry, deep learning is used to solve practical tasks in various fields such as computer vision (images), natural language processing (text), and automatic speech recognition (audio) (Andrew W. Trask, 2019). In other words, Deep Learning is a subset of methods in machine learning toolboxes that primarily use artificial neural networks, a class of algorithms loosely inspired by the human brain (Figure 2-2).

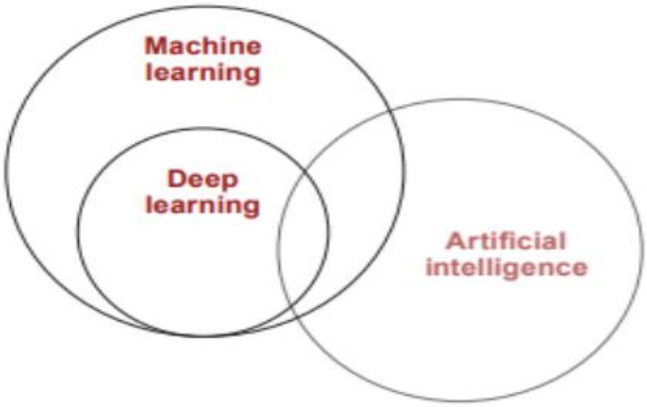


Figure 2-3 Deep learning relations with AI and MI

### 2.1.5 Deep Learning Methods

Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), DE Noise AutoEncoders (DAE), Deep Belief Networks (DBN), Long Short-Term Memory (LSTM) and deep neural networks (DNN) are the most popular and widely used deep learning techniques. A description of each method is provided along with notable applications (Mosavi et al., 2020).

#### 2.1.5.1 Convolutional Neural Network (CNN)

The CNN is one of the most famous architectures in DL technology. Generally, this technique is used for image processing applications. A CNN contains three kinds of layers: convolutional layer, pooling layer, and fully connected layer (Figure 2-4**)**. It has two phases in its training process: feed-forward phase and back-propagation phase. Famous architectures in CNNs are ZFNet, Google Net, VGGNet, Alex Net, and ResNet. CNNs are known primarily for their applications in image processing, although other areas of applications are represented in the literature as well (Mosavi et al., 2020).

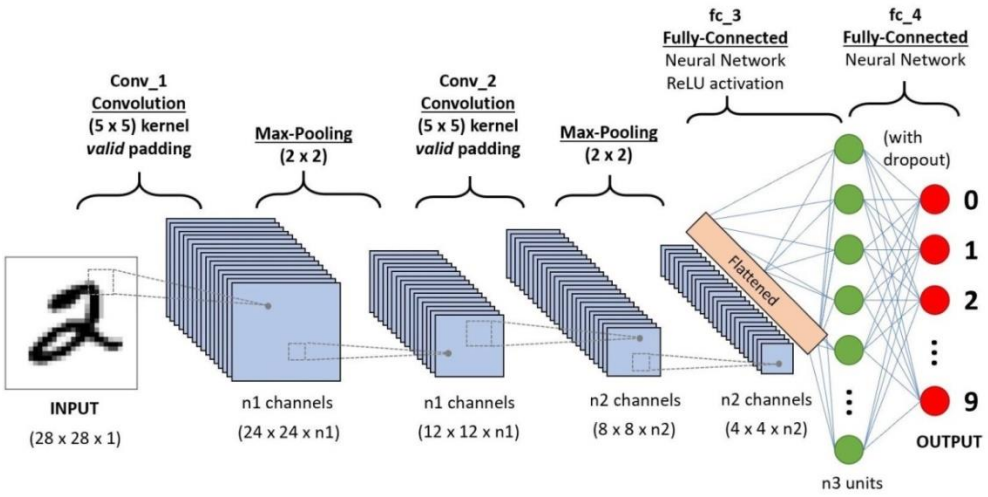


Figure 2-5 Convolutional Neural Networks (CNN) architecture

#### 2.1.5.2 Recurrent Neural Networks (RNN)

RNNs are designed to recognize sequences and patterns, like speech, handwriting, text, and applications alike. The cyclic connections are helpful in RNNs for the iterative computation-based architectures that are designed to process input data sequentially (Figure 2-6). RNNs usually stand for feed-forward neural networks that, over time, have been extended by having the edges flow to the next step in time instead of the next layer within the same time step. The RNNs keep every previous input data in the hidden units of state vectors and use them to compute the output (Mosavi et al., 2020).

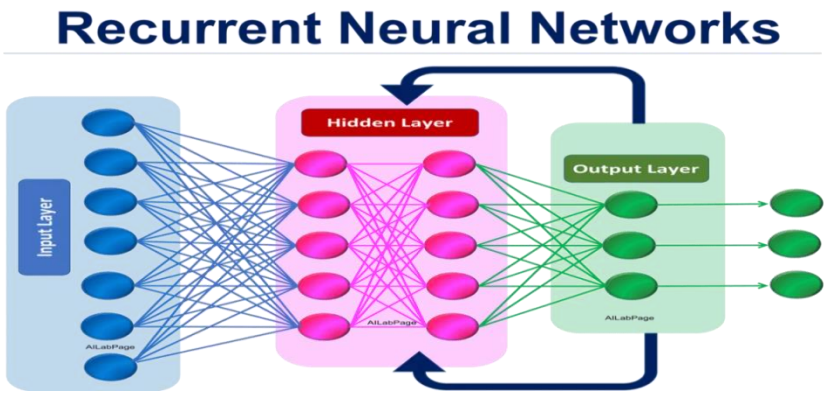


Figure 2-7 Recurrent Neural Networks (RNN) architecture

### 2.1.5.3 DE Noising AutoEncoder (DAE)

DAE was extended from AE as an asymmetric neural network for learning functions from noisy datasets. A DAE consists of three main layers, including an input layer, an encoding layer, and a decoding layer **(**Figure 2-8). DAEs can be aggregated to handle higher-level functions. Stacked DE Noising AutoEncoder (SDAE) is generated as an unsupervised algorithm by the DEA method and can be used for nonlinear dimensionality reduction. This method is a kind of feed forward neural network and uses a deep architecture with multiple hidden layers and a pre-training strategy (Mosavi et al., 2020).

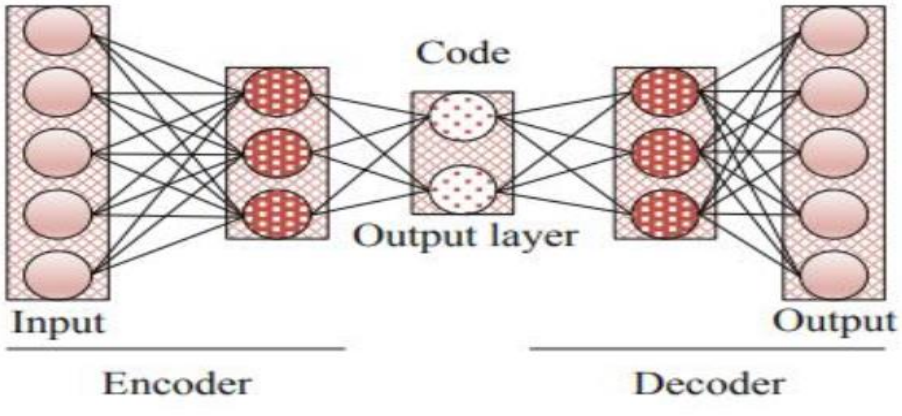


Figure 2-9 DE noising AutoEncoder (DAE) architecture

### 2.1.5.4 The Deep Belief Networks (DBNs)

DBNs is used to learn high-dimensional manifolds of data. This method involves multiple layers with connections between layers, except connections between units within each layer **(**Figure 2-10**)**. A DBN can be viewed as a hybrid multilayer neural network containing directed and undirected connections. The DBN contains a greedily trained Constrained

BoltzmannMachine(RBM)(Mosavi,et,al.,2020).

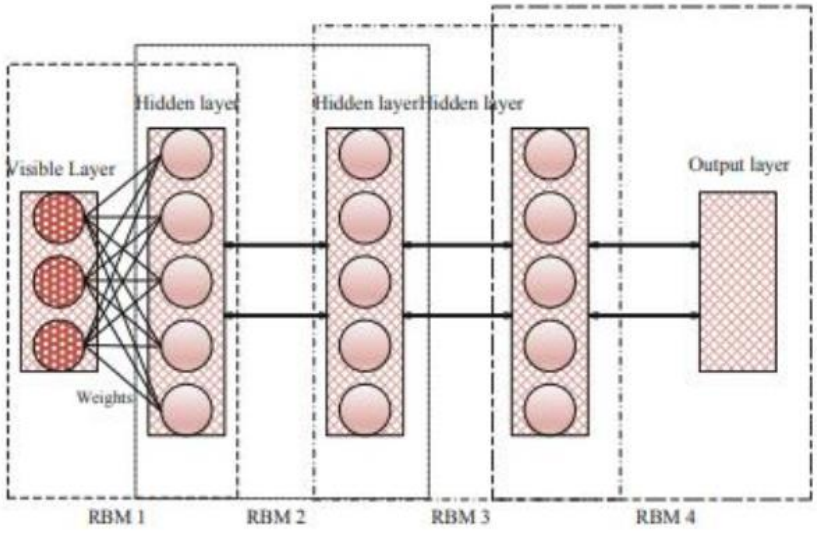


Figure 2-11 Deep Belief Networks (DBNs) architecture

### 2.1.5.5 Long Short-Term Memory (LSTM)

LSTM is an RNN technique that uses feedback connections for general-use destination computers. This method can be used for both sequence and pattern recognition and image processing applications. In general, an LSTM contains three central units, including input, output, and forget gates **(**Figure 2-12**)**. LSTMs control the decision when to enter a neuron and can remember what was computed at the previous time step. One of the main strengths of the LSTM method is that it makes all these decisions based on the current situation. Input it (Mosavi et al., 2020).

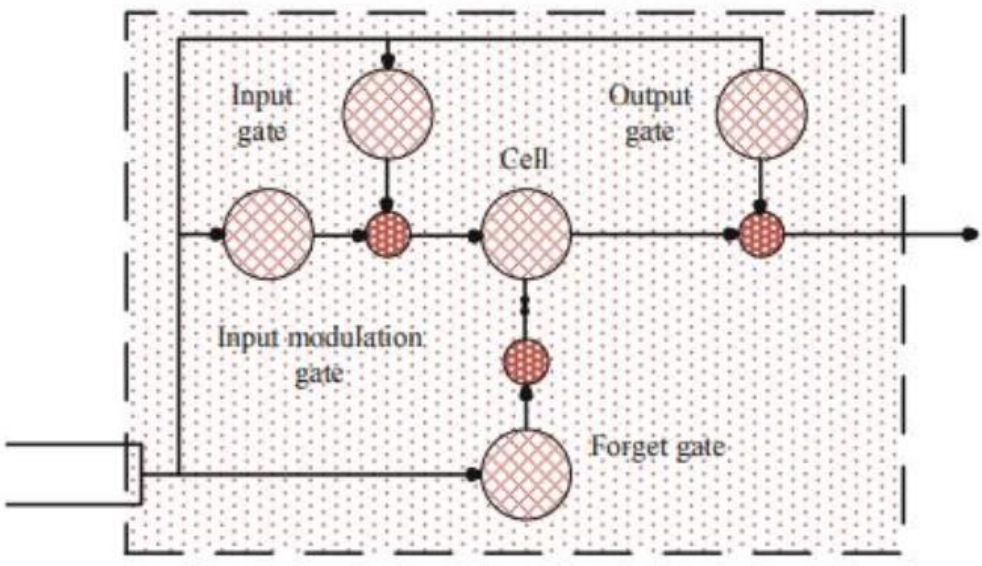


Figure 2-13 Long Short-Term Memory (LSTM) architecture

**2.1.5.6 Deep neural network (DNN):** is develops hierarchical artificial neural networks consisting of layers of neurons. a DNN trained with a large volume of data can represent more and more complex functions. Compared to a traditional neural network, a DNN has more consecutive hidden layers and more neurons within each layer (Figure 2-14). This structure allows the neural network to identify high-level and abstract data features from the raw data. Specifically, the more complex data features identified by a successive layer are built upon the other, simpler data features extracted by the predecessor layer. Such a data transition and transformation process through multiple layers of neurons makes a DNN a "thinking" machine (Najafabadi, et al., 2015).

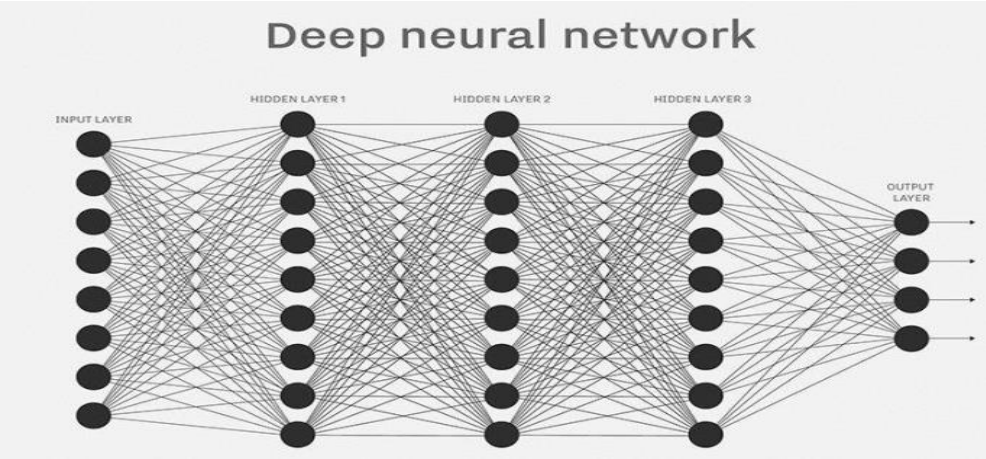


Figure 2-15 deep neural network architecture

**2.1.6 Tax evasion**: It is the illegal process of evading paying tax liability by misrepresenting one's or business entity's actual tax liability (W. Fox et al, 2017, Julia Kagan, 2022).

**2.1.7 Ethical consideration:** Ethical consideration means the basis of conduct or practice with a system of moral principles or values concerning the consequences of certain actions that affect human beings and society. Ethics in deep learning and artificial intelligence will highly contribute to the building up and deployment of such technologies responsibly. The following key ethical issues arise from the use of deep learning and AI:

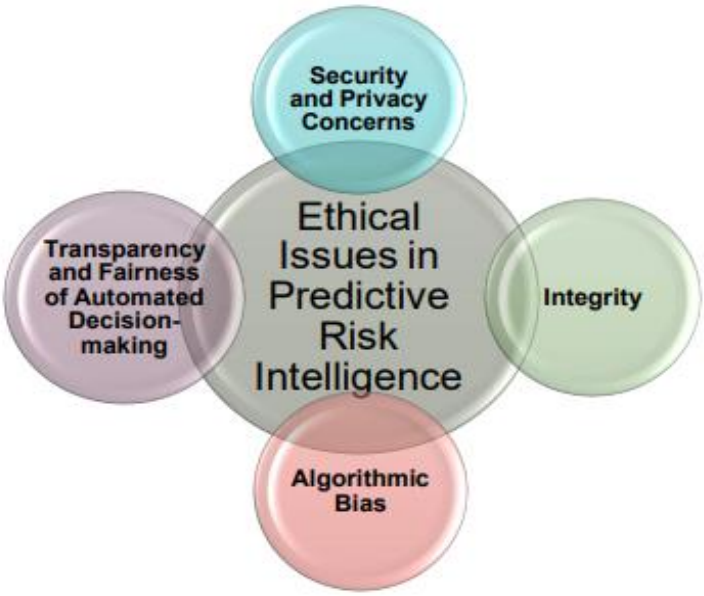


Figure 2-16 Ethical consideration for predictive analytics

**Security and Privacy Concerns (Figure 2-16):** Ideally, predictive analytics or predictive intelligence involves linking data from multiple sources with social data to identify trends and provide insights for decision-making. With many new sources of data becoming available, such as data from social media applications, aggregating these data sources to achieve predictive analytics raises privacy concerns and requires new ways to preserve privacy (Bates et al., 2014; Petersen, 2018).

**Integrity (Figure 2-16):** Another ethical issue with predictive risk intelligence relates to a lack of integrity when designing or using algorithms. For many companies, revealing certain information would have a knock-on effect on their business. Therefore, they may compromise the integrity of their processes to save their business (Hacker, 2018). There exists an associated danger of a business and its ethical responsibilities towards other stakeholders conflicting.

**Transparency and fairness of automated decision-making (Figure 2-16):** There is a question of algorithms' transparency and fairness with the intelligence being predictive risk intelligence. (Wachter, Mittelstadt, & Floridi, 2017b). According to Wachter et al., (2017), this concern arises because SIS use complex and opaque algorithmic mechanisms that can have many unintended and unexpected effects. When it comes to transparency and fairness in the automated decision-making process, such as predictive risk intelligence, users or clients only get a limited idea of why a decision has been made in a certain way, which does not mean the decision is justified, or legitimate (Wachter, Mittelstadt, & Floridi, 2017a).

**Algorithmic Bias (Figure 2-16):** There are also concerns about the reliability of using AI in making predictions. For example, AI can learn bias and prejudicial values when they are present within the dataset, leading to unfair or inaccurate predictions (Barocas & Selbst, 2016). A lack of reliability in the predictions that are made by the SIS can be introduced when certain data are either included in or excluded from the training dataset. Due to potential bias in developing algorithms, AI can learn pre-existing inequalities that are present in the training dataset, resulting in a bias towards historically disadvantaged populations (Barocas & Selbst, 2016).

**2.1.8 Hyperparameters** Some of the most important configuration settings are in machine learning, with hyperparameters controlling how a model learns or makes predictions. Proper tuning of these hyperparameters is critical in bringing significant improvements in model performance (Bartz, 2023).

**Key Hyperparameters: -**

**The Number of Neurons**: This is used in neural networks for the number of neurons within a layer. Having more neurons allows the model to learn more complicated patterns, but it might also raise the chances of overfitting.

**Learning Rate:** amplitude of the step taken at each iteration of optimization, this hyperparameter will set the size. If too high, the learning rate might get stuck into a suboptimal solution after a few iterations; if too low, the training might take too much time.

**Activation Function:** Type of activation function-the most common are ReLu, Sigmoid, and Tanh-affects how the output of a neuron is computed and can significantly influence the learning dynamics of the model.

**Epochs:** This is a hyperparameter that defines the number of times a feed forward network sees the entire training dataset. Having more epochs could improve learning; however, this may also cause overfitting when your model starts memorizing the training data.

**Batch Size**: The number of training examples used in one iteration. Smaller batch sizes can be more representative of the true gradient, but this causes longer training times. Optimizer: The optimization algorithm used, such as Stochastic Gradient Descent and Adam, defines the method of updating the model parameters based on the gradients. Consequently, different optimizers converge at different rates and may be more suitable for various problems than others.

#### 2.1.8.1 Hyperparameter Tuning

In a nutshell, hyperparameter tuning means enhancing these settings even further to better optimize model performance. Essentially, this involves experimenting with various combinations of hyperparameters and then testing the performance with a validation set (Bartz, 2023).

#### 2.1.8.2 Hyperparameter Tuning Techniques

**Grid Search**: A grid search entails an exhaustive search in the pre-defined subset of hyperparameters to identify their most optimal combination (Bartz, 2023).

**Random Search:** Unlike evaluating all combinations, this technique randomly samples the hyperparameter space, which in most cases is more efficient than grid search (Bartz, 2023).

**Bayesian Optimization:** This method is a way of learning a probabilistic model of the mapping between hyperparameter values and model performance. Then, using this model, it decides which hyperparameters are the most promising to evaluate next (Bartz, 2023).

**Evolutionary Algorithms**: are bio-evolution-inspired algorithms which employ mutation and selection processes in order to study the hyperparameter spaces (Bartz, 2023).

**Automated Hyperparameter Tuning**: Hyperband, neural architecture search, etc.- advanced techniques have helped in automating the tuning process, hence making it more efficient and effective.

## 2.2 Related work

In related research, the research era related to my research, various algorithms were used to predict tax evasion. Predicting tax fraud using a supervised machine learning approach (Murorunkwere et al., 2023). This paper compares the performance between Artificial Neural Networks, Logistic Regression, Random Forest, Decision Trees, GaussianNB, and eXtreme Gradient Boosting. The best accuracy of 92% was found in Artificial Neural Networks over the rest of the models. Predictive analytics as a service on tax evasion using feature engineering strategies (Kishore Babu & Vasavi, 2019). In this paper, the researcher used recursive feature elimination (RFE) with a random forest algorithm for feature engineering, and ensemble models combining artificial neural networks (ANN), auto-regression (ARX), and Gaussian process (GP) for predictive modelling. Found that feed-forward back propagation neural network proved to be better in terms of accuracy than regression. Neural network Normalized Root Mean Square Error (NRMSE) 0.0113 and Coefficient of determination (COD) 0.99. Deep Learning Applications In Audit Decision

Making(Sun et al., 2018). In this paper, the researcher used a deep learning textual analyser provided by IBM Watson's Alchemy Language API to extract sentiment features from earnings conference call transcripts. And compare three models logistic regression, random forest, and artificial neural network. Then concluded random forest was performed sentimental model with an accuracy of 83.57%. Data Mining Based Tax Audit Selection: A Case Study of a Pilot Project at the Minnesota Department of Revenue (Hsu et al., 2015). In this paper researcher used a data mining-based approach. Found that The ROI value for the manual audit selection process used at the time of the pilot project for APGEN Large is 1652 % and the same for the data mining-based approach is 2300 %. This represents a 39.2 % increase in efficiency. Bayesian Networks on Income Tax Audit Selection Case Study of Brazilian Tax Administration (Sólon da Silva et al., 2023.) . In this paper researcher used two types of Bayesian networks Naive Bayes and Tree-Augmented Naive Bayes. Naïve Bayes is already a good tool to select those taxpayers who can and cannot be dismissed from being audited performing 45%. Tree-Augmented Naïve Bayes had no major advantages and performed 34%. Deep neural network and time series approach for finance systems: Predicting the movement of the Indian stock market (Srivastava et al., 2021). In this paper, the researcher compares support vector machines, random forests, gradient boosting, and deep neural networks. Found that Deep Neural Network performed better over another model with an accuracy of 98.7% achieved with 99.5% precision and 93% recalls. A deep learning-inspired belief rule-based expert system (Ul Islam et al., 2020) . In this paper, the researcher compares the Long-Short Term Memory, Deep Neural Network, and Deep Learning (DL) method with the Belief Rule-Based Expert System (BRBES). Found that BRB-DL outperforms other methods like Long-Short Term Memory, Deep Neural Network, Fuzzy Deep Neural Network, Adaptive Neuro Fuzzy Inference System, and traditional BRBES in terms of prediction accuracy, e.g. achieving a Mean Square Error of 4.12 compared to 18.66, 28.49, 17.05, 16.37, and 38.15 respectively for a combined cycle power plant dataset. Deep Learning meets Risk-based Auditing: A holistic Framework for leveraging Foundation and Task-specific Models in Audit Procedures(Lars Föhr et al., 2020.). In this paper researcher proposes a framework based on the CRISP-DM (Cross-Industry Standard Process for Data Mining) process model and AI (Artificial intelligence) Canvas for the implementation and utilization of DL in risk-based auditing. Empower auditors to make use of DL models in identifying and evaluating the risks of material misstatement of the financial statements, which is part of the general trend to elevate the quality of audits. Various studies on Predictive analytics, Deep Learning, and Artificial Intelligence have

been carried out for prediction of fraudulent claims against tax frauds and evasion. Further research in these areas includes:

Table 2-1 comparison of related works

| No | Title | method | Limitation | Result |
|---|---|---|---|---|
| 1. | Predictive Analytics for Controlling Tax Evasion (Kumar, 2018) | Formulating the problem of detecting circular trading fraud as finding clusters in a weighted directed graph, where the edge weights were defined using Benford's analysis. | Used traditional machine learning algorithm. | cutoff equal to 0.5 is 86.38% and testing accuracy is 86.33%. Precision of the model is 85.5%. Recall of the model is 97.97% |
| 2. | Audit lead selection and yield prediction from historical tax data using artificial neural networks (Chan et al., 2022) | artificial neural networks | Not been statistically validated, and do not learn from historical data and audit outcomes. | achieved a 40.1% precision and 58.7% recall (F1-score of 0.42) on classifying positive audit leads |

| | | | | |
|---|---|---|---|---|
| 3. | Predictive Analytics and the Targeting of Audits (Hashimzade et al., 2015) | predictive analytics based on Tobit and logit regression models | Used traditional model | The reported coe¢ cients and standard errors are the maximum likelihood estimates from the data accumulated up to the final period of one simulation are 2077 observations and 1943 observations in Tobit and logit regression model respectively. |
| 4. | Tax crime prediction with machine learning: A case study in the municipality of São Paulo (Ippolito & Lozano, 2020) | Neural Networks, Naive Bayes, Decision Trees, Logistic Regression, Random Forests, and Ensemble Learning | Used traditional model | Random Forests best performed Recall 51.6% , Precision 87.0% , F-measure 64.8 %, Accuracy 66.2% , Specificity 88.3% |
| 5. | Non-Factoid Answer | Long Short-Term Memory | Use small datasets. | LSTM model |

| | | | | |
|---|---|---|---|---|
| | Selection in Indonesian Science Question Answering System using Long Short-Term Memory (LSTM) (Hanifah & Kusumaningrum, 2021) | (LSTM) | | was evaluated using Mean Reciprocal Rank (MRR) is 90.06% and Mean Average Precision (MAP) metrics is 78.69% |
| 6. | Gated Mixture Vibrational AutoEncoders for Value Added Tax audit case selection (Kleanthous & Chatzis, 2020) | Gated Mixture Vibrational AutoEncoder deep network | Not optimized the model and not compared with other model. | 76% accuracy |
| 7. | Applying robotic process automation (RPA) in auditing: A framework (Huang & Vasarhelyi, 2019) | robotic process automation | Do not specify which country to auditing. | 99.9% accuracy |

# CHAPTER THREE
# METHODOLOGY AND MATERIALS

## Introduction

In this chapter, we carried out Predictive analytics for controlling tax evasion using deep learning: in the case of Ethiopia Customs Commission taxpayers. These procedures include research objectives, data collection, data understanding, and data preparations; data pre-processing, modelling, evaluation, and material used. A general explanation of the research process is shown in the Figure

.

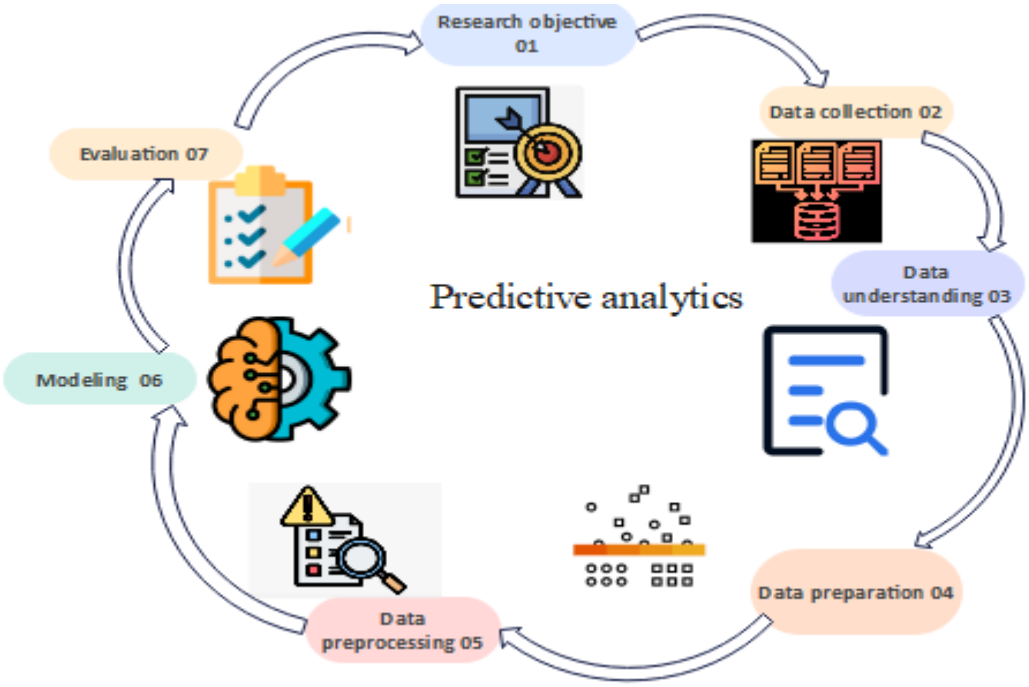


Figure 3-1 Predictive analytics using deep learning methods

### 3.1 Data collection procedure

- **The Purpose of Data Collection:** The main purpose of data collection is to fulfil the general objective of Predictive analytics for controlling tax evasion using deep learning: the case of Ethiopia Customs Commission taxpayer's thesis. Mainly focusing on controlling tax evasion in the audit selection process.
- **Plan the Data Collection Process:** planned three months to collect data. First, one month of understanding how to use deep learning and predictive analytics for controlling tax evasion and business. Then the data required for the model controlling

tax evasion. Second, for one week identified how and where to get the data set, thirdly designed the data collection tools, fourthly for one month and two weeks we collected the data required from identified sources.

**3.1.1 Identify Data Sources**: the relevant data sources are Ethiopia Customs Commission ECMS data of importers from Ethiopia Customs Commission, historical data of post clearance audit from Ethiopia Customs Commission post clearance audit, and domestic revenue data of importers from Ethiopia revenue minister to model deep learning and predictive analytics for controlling tax evasion.

**3.1.2 Design Data Collection Tools:** we created the necessary tools to collect the data. This involves soft tools notebook and questionnaires, and hard tools 32GB flash drive set.

**3.1.3 Data collection:** to execute a data collection plans; gathering data from the identified sources. we collected the five years importer data from 2018 to 2022 GC years which contain 138,644 row by 37 columns excel datasets of import taxpayers from the Ethiopia customs commission head office provided via Gmail waleligntsige25@gmail.com and flash drive, they are employer, domestic revenue five years data of importer from 2018 to 2022 GC years which is collected from Ethiopia revenue minister which contain 15,070 row by 2 columns excel data sets provided via Gmail bakinardyee@gmail.com employer of Ethiopia customs commission head office and historical data of post clearance audit five years data from 2018 to 2022 GC years which contain 2686 rows by 6 columns excel data sets provided via Gmail abrehetg@gmail.com Ethiopia customs commission employer, the books of risk analysis, manual audit section criteria and post clearance audit manual are provided via flash drivers. And also token one month 15 hrs. Lecture by senior expert on audit selection processes; how the current audit selection works based on manual rule-based using Excel sheet and each of 11 steps for audit selection criteria calculation from the data sets and the company rule. The collected data consisted of three data types categorical, numerical, and date time.

| | Year | TIN | PCA_Audit_Types | PCA_Fraud_Types | PCA_Top_Up 2 | Branches |
|---|---|---|---|---|---|---|
| **0** | 2018 | 0005332825 | Issue | Valuation | 1983935.01 | Mojo |
| **1** | 2018 | 0000055255 | Issue | Valuation | 73069.37 | Mojo |
| **2** | 2018 | 0000007020 | Issue | Tariff | 62587.69 | Mojo |
| **3** | 2018 | 0035725737 | Issue | Tariff | 37988.08 | Mojo |
| **4** | 2018 | 0000022530 | Issue | Tariff | 7788953.78 | Mojo |
| **...** | ... | ... | ... | ... | ... | ... |
| **2681** | 2022 | 0032265332 | Issue | Valuation | 57785.63308 | Mojo |
| **2682** | 2022 | 0032333260 | Issue | Valuation | 998.193493 | Mojo |
| **2683** | 2022 | 0032257353 | Desk | NaN | NaN | Mojo |
| **2684** | 2022 | 0036082832 | Desk | NaN | NaN | Mojo |
| **2685** | 2022 | 0036337580 | Desk | Valuation & Freight | 128295.54 | Mojo |

2686 rows × 6 columns

Figure 3-2 Historical audit datasets of Ethiopia customs commission taxpayers

| | Row Labels | Domestic risk |
|---|---|---|
| **0** | 00000000KR | YELLOW |
| **1** | 0000000LBS | YELLOW |
| **2** | 0000002006 | YELLOW |
| **3** | 0000002032 | YELLOW |
| **4** | 0000002223 | YELLOW |
| **...** | ... | ... |
| **15065** | 0270722333 | YELLOW |
| **15066** | 0270727383 | GREEN |
| **15067** | 0270738256 | YELLOW |
| **15068** | 0270768607 | RED |
| **15069** | 0270802238 | RED |

15070 rows × 2 columns

Figure 3-3 Domestic datasets of Ethiopia customs commission import taxpayers

| | Office Code | Office Name | Year | Dec. Type | Reg. Serial | Reg. No | Dec. Reference | Reg. Date | Asmt. Date | Section | ... | Agent Name | Agent Telephone | CIF ETB | Total Amount Paid | Assessment Notice Status | Total Tax Amount | Top Up | Total Amount TO Be Paid | Cleared Flag | Cleared Date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | MOJ00 | MOJO Dry Port (MOJ) | 2018 | IM | 4 | Nan | 9938/T93698 | 2018/06/30 | 2018/06/30 | ME | ... | Nelaku Bekele Celila | 0944040249 | 167680.0 | 0 | PAID | 199618.26 | 320572.01 | 117.38 | Cleared | 2022/12/07 |
| 1 | MOJ00 | MOJO Dry Port (MOJ) | 2018 | IM | 4 | Nan | TwFTw999349 | 2018/07/02 | 2018/07/02 | ME | ... | Csehaynesh Bekele W/Senbet | +294944242424 | 5333441.0 | 0 | PAID | 285922.88 | 44193.56 | 1110422.31 | Cleared | 2021/06/29 |
| 2 | MOJ00 | MOJO Dry Port (MOJ) | 2018 | IM | 4 | Nan | wIM99335 | 2018/07/02 | 2018/07/02 | MF | ... | Nesfin Ndmasu Yihayis | 0944622294 | 9598362.0 | 0 | PAID | 209083.41 | 1429942.23 | 0.00 | Cleared | 2021/04/23 |
| 3 | MOJ00 | MOJO Dry Port (MOJ) | 2018 | IM | 4 | Nan | wIM99339 | 2018/07/02 | 2018/07/02 | MF | ... | Nesfin Ndmasu Yihayis | 0944622294 | 9390380.0 | 0 | PAID | 196659.34 | 1672549.88 | 0.00 | Cleared | 2021/04/23 |
| 4 | MOJ00 | MOJO Dry Port (MOJ) | 2018 | IM | 4 | Nan | wIM99333 | 2018/07/02 | 2018/07/02 | MF | ... | Nesfin Ndmasu Yihayis | 0944622294 | 9598383.0 | 0 | PAID | 229475.52 | 2532612.80 | 0.00 | Cleared | 2021/04/23 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 138639 | MOJ00 | MOJO Dry Port (MOJ) | 2022 | IM | 5 | Nan | TwTL998 | 2022/02/04 | 2022/02/04 | IOG | ... | oetachew Nuleta Badasa | 0920000274 | 2612764.0 | 0 | PAID | 436098.48 | 96855.08 | 0.00 | Cleared | 2022/02/24 |
| 138640 | MOJ00 | MOJO Dry Port (MOJ) | 2022 | IM | 5 | Nan | wRIM9995 | 2022/02/08 | 2022/02/08 | IOG | ... | oedion Zuri EEZCoNZoN | 0944924299 | 4057544.0 | 0 | PAID | 904042.66 | 729969.17 | 0.00 | Cleared | 2022/02/11 |
| 138641 | MOJ00 | MOJO Dry Port (MOJ) | 2022 | IM | 6 | Nan | 999934T | 2022/01/17 | 2022/01/17 | MG | ... | Cewfik Idris Nhmed | +294 944200406 | 1457244.0 | 0 | PAID | 661105.83 | 990749.73 | 1020.07 | Cleared | 2022/02/24 |
| 138642 | MOJ00 | MOJO Dry Port (MOJ) | 2022 | IM | 6 | Nan | FRL497 | 2022/12/15 | 2022/12/15 | MG | ... | ERNINS o/NEEolZ BELNY | 0929924772 | 90054.0 | 0 | PAID | 121866.96 | 0.00 | 21450.91 | Cleared | 2022/12/24 |
| 138643 | MOJ00 | MOJO Dry Port (MOJ) | 2022 | IM | 6 | Nan | 6639-9943 | 2022/02/15 | 2022/02/15 | MG | ... | Nehari Kiros o/N | 0000000000 | 111559.0 | 0 | PAID | 0.00 | 371562.31 | 26573.36 | Cleared | 2022/03/25 |

138644 rows × 37 columns

Figure 3-4 ECMS datasets of Ethiopia customs commission taxpayers

### 3.1.4 Data description

Table 3-1 Data description ECMS datasets

| No | Attribute Name | Data type | Data description |
|---|---|---|---|
| 1 | TIN | object | The taxpayers' identification numbers |
| 2 | Importer | object | The importer is taxpayer |
| 3 | CIF ETB | float64 | Total amount cost, insurance and fright |
| 4 | Declarant Name | object | The name of person work for importer. |
| 5 | Declarant Code | int64 | It is the unique number for declarant |
| 6 | Agent Code | object | It is the unique number for agent |
| 7 | Agent Name | object | The name of person work for declarant |
| 8 | Total Amount Paid | int64 | The payment in destination office. |
| 9 | Total Tax Amount | float64 | The total all tax paid by importer. |
| 10 | Agent Telephone | object | The telephone number of agent. |
| 11 | Top Up | float64 | The amount paid by offence. |
| 12 | Total amount to be paid | float64 | The amount of tax not paid |
| 13 | Cleared Date | object | The date the document cleared. |
| 14 | Importer Telephone | object | The telephone number of importer. |
| 15 | Importer Address | object | The address of importer. |
| 16 | Cleared Flag | object | The document cleared flag. |
| 17 | Assessment Notice Status | object | The document status when assessment. |

| | | | |
|---|---|---|---|
| 18 | Released_By | object | The customs officer released document. |
| 19 | Country (Origin) | object | The country where goods made in. |
| 20 | Country (Consignment) | object | The country where goods consign. |
| 21 | Released | object | The good release from clearance unit. |
| 22 | Dec. Status | object | The document declaration status. |
| 23 | Invoice Amount | float64 | The amount the customer paid. |
| 24 | Currency Code Decs. | object | The code given to currency on document. |
| 25 | Currency Code | object | The code of currency. |
| 26 | Current Lane | object | The risk after payment of tax and duty. |
| 27 | Original Lane | object | The risk when document declare. |
| 28 | Section | object | The different units in customs office. |
| 29 | Asmt. Date | object | The assessment date of documents. |
| 30 | Reg. Date | object | Date of document registration to customs. |
| 31 | Dec. Reference | object | The declaration reference of document. |
| 32 | Reg. No | object | The document registration number. |
| 33 | Reg. Serial | int64 | The document registration serial. |
| 34 | Dec. Type | object | The document declaration type. |
| 35 | Year | int64 | The year of importer take service. |
| 36 | Office Name | object | The branch office goods is consigned. |
| 37 | Office Code | object | The code give to branch office. |

Table 3-2 Data description domestic datasets

| NO. | Attribute Name | Data types | Data description |
|---|---|---|---|
| 1 | TIN | object | Taxpayer identification numbers. |
| 2 | Domestic risk | object | The risk of importer domestic. |

Table 3-3 Data description historical audit datasets

| NO. | Attribute Name | Data types | Data description |
|---|---|---|---|
| 1 | Year | int64 | The year taxpayer audited. |
| 2 | TIN | object | Taxpayer identification numbers. |
| 3 | PCA_Audit_Types | object | The post clearance audit types. |
| 4 | PCA_Fraud_Types | object | The post clearance audit fraud types. |
| 5 | PCA_Top_Up 2 | object | The past clearance audit office amount |
| 6 | Branches | object | The customs branch audited. |

## 3.2 Data preparation

In the data preparation steps, we accomplished data conversion, data cleaning, data sampling, and feature extraction (dimensionality reduction).

### 3.2.1 Data conversion

1. Data replacement: for data confidentiality the TIN, declarant code, Reg. No, Dec. References, Importer, Importer Address, Importer Telephone, Declarant Name, Agent Code, Agent Name, and Agent Telephone columns are replaced by another equivalent value. For example, for TIN columns the current value is 0093273222 then it did amend by excel function = SUBSTITUTE (SUBSTITUTE (sheet1 TIN columns, "9", "5"), "2", "7") then new TIN be 005373777. This is done by a data-providing company and it works for the rest of the columns that need confidentiality.

2. Convert the Excel data to CSV data (comma-separated value)

3. Rename data sets: The Ethiopia Customs Commission ECMS data set is renamed as Data1, the Ethiopia Ministry of Revenue data is renamed as Data2, and the Ethiopia Customs Commission post clearance audit history data set is renamed as Data3.

### 3.2.2 Data Cleaning

A. **Handle missing values**: - Check missing values in each of the columns, and decide how to handle missing values; this is accomplished by imputing the missing value with appropriate values and removing the columns based on the necessity of the columns for feature selection.

In Data1 the column Currency Code Decs are not necessary for feature selection according to domain knowledge so remove the column. This is done by removing the missing value column the column 'Currency Code Decs.' is specified as an argument to the drop() method. The axis parameter is set to 'columns' to indicate that we want to drop the specified column 'Currency Code Decs.'. Then assign the Data1 data frame to Data1_rmvc.

In Data 2 the Domestic risks are necessary for feature selection according to domain knowledge so fill the non-value columns with Green. This is done by the fillna () method to replace any none or null values in the 'Domestic risks' column of the 'Data2' data Frame with the string 'Green'. Since the Green is the lowest risk value from the customs commissioner is zero so for string value making Green is equal to zero. Then assign the Data2 data frame to Data2_rmvr.

In Data3 the two columns have missing values. Those are PCA_Fraud_Types and PCA_Top_Up 2. From those PCA_Top_Up 2 is necessary for feature selection so that the missing value is handled by filling zero value to PCA_Top_Up and PCA_Fraud_Types columns is not necessary for feature selection based on domain knowledge so removing the column from the data frame by the missing value column 'PCA_Fraud_Types' is specified as an argument to the drop() method. The axis parameter is set to 'columns' to indicate that we want to drop the specified column 'PCA_Fraud_Types'. The data frame is assigned as Data3_re_drop.

B. **Handle outliers:** - The outliers due to data entry errors or measurement mistakes can be removed from the dataset.

In Data3 the outlier data due data entry mistake of column PCA_Top_Up 2 must contain the numeric data, either float or int data types according to domain knowledge but due to mistake data entry, object data types. So we identified the non-numeric data types in the PCA_Top_Up 2 column and replaced it with zero value.

C. **Handle inconsistency: -** Handle by formatting data in the correct form.

In Data1 all formatting is correct since the data collected from the ECMS system is consistent.

In Data2 the data fills the Excel sheet manual so it has data inconsistency in capitalization. To handle the data inconsistency first, change the string value columns into lowercase letters and then again capitalize the first letter Domestic risk column necessary for feature selection so make it constituency.

In Data3 the data fills the Excel sheet manual so it has data inconsistency in capitalization but the data needed for feature selection is consistent so leave these steps.

### 3.2.3. Data Sampling

The dataset is large; consider sampling a representative subset for analysis or model training to reduce computational requirements. In this we select dependent variable based on domain knowledge from Data1 columns selected 'Office Name', 'Year', 'Original Lane', 'Country (Consignment)', 'Country (Origin)', 'TIN', 'Declarant Code', 'CIF ETB', 'Total Tax Amount', 'Top Up', Data2 columns selected all columns, and Data3 columns selected Year, TIN', 'PCA_Top_Up 2'. Those columns are used to calculate the independent variable according to the domain knowledge of Customs Commission post-clearance audit selection criteria.

### 3.2.4 Feature extraction

After selecting the sample data based on domain knowledge audit selection criteria. Need to reduce the dimensionality of our data to save computation resources.

Dimensionality reduction: Reduce the number of features in the dataset while preserving important information. Data1 columns selected to be reduced ‘Office Code', 'Dec. Type', 'Reg. Serial', 'Reg. No', 'Dec. Reference', 'Reg. Date', 'Asmt. Date', 'Section', 'Current Lane', 'Dec. Status', 'Invoice Amount', 'Currency Code', 'Currency Code Decs.', 'Released', 'Released_By', 'Importer', 'Importer Address', 'Importer Telephone', 'Declarant Name', 'Agent Code', 'Agent Name', 'Agent Telephone', 'Assessment Notice Status', 'Total Amount TO Be Paid', 'Cleared Flag', 'Cleared Date' and Data3 columns selected to be reduced 'PCA_Audit_Types', 'Branches'. Those columns are not used for audit selection according to domain knowledge customs commission post clearance audit selection criteria.

## 3.3 Data pre-processing

After data preparation, data pre-processing by data integration, data conversion, normalization, and feature engineering using domain knowledge, the Ethiopia Customs

Commission audit selection criteria and data-driven decisions are then ready to train deep learning models.

### 3.3.1 Data integration

Data integration: focuses on combining data from Data1 and Data2 sources to create a unified and consistent view of the data. This involves merging data that contain the same variable belonging to Data1_reduce, Data2_rmvrc, and Data3_reduced sources. The objective of data integration is to provide a unified data source that can be easily accessed and analysed. Merge data frame Data1_reduce and Data3_reduced using the common columns TIN and Year. We assigned the merged dataset as Data13. Merge data frame Data13 and Data2_rmvrc using the common columns TIN. We assigned the merged dataset as Data132.

### 3.3.2 Feature Engineering

Feature engineering using statically analysis of company rules; which are accomplished using a risk analysis system and post-clearance audit selection criteria simplify analysis by automating traditional Excel analysis using pandas python. General feature engineering is done by calculating the consequence, likelihood, number of shipments, number of offenses, of 'Cost Insurance and Freight', 'Customer profile', 'Declarant profile', 'Country of origins', 'Country of consignments', 'PCA history', 'Clearance unit risk', 'Intelligent risk', and 'Sector risk' variables from data sets per each TIN numbers. The data sets contain three data types: numeric, categorical, and time data types: all data types are calculated according to company rules, and time series. The 'CIF', 'Customer profile', 'Declarant profile', 'Clearance unit risk', 'Intelligent risk', 'PCA history' and 'Sector risk' variables done per 5 years and the 'Country of origins' and 'Country of consignments ' variables done per 1 year and check either the country is preferential or non-preferential according to the agreement between country and Ethiopia if there is agreement it preferential unless no preferential[3].

To remove the following: - The data processing of each variable …. Data driven decisions…. …Taking many pages of your thesis. So removed 35 pages as comments from this sections

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00000000KR | 0 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 0 | 0 | 0.68 |
| 1 | 0000000LBS | 0 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 0 | 0 | 0.68 |
| 2 | 0000002006 | 0 | 1 | 1 | 2 | 2 | 0 | 1 | 2 | 0 | 0 | 0.64 |
| 3 | 0000002032 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 0 | 0 | 0.84 |
| 4 | 0000002223 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 0.68 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 15065 | 0270722333 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 1 | 0 | 0.68 |
| 15066 | 0270727383 | 0 | 1 | 1 | 2 | 2 | 1 | 0 | 2 | 0 | 0 | 0.52 |
| 15067 | 0270738256 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 0 | 0 | 0.84 |
| 15068 | 0270768607 | 2 | 2 | 2 | 2 | 2 | 0 | 2 | 2 | 1 | 2 | 1.80 |
| 15069 | 0270802238 | 0 | 1 | 2 | 0 | 0 | 2 | 2 | 2 | 1 | 0 | 0.92 |

15070 rows × 12 columns

Figure 3-5 Pre-processed datasets

# 3.5 Data Splitting

### 3.5.1 Split data into variable X and target Y

The system contains two types of variables: 11 independent variables 11 variables derived from the features in datasets called X and the dependent variable after predictive analysis is target Y. Independent variable X is from audit selection data frame columns are Risk of CIF, Customer Profile Risk, Declarant Profile Risks, country of origin risks, country of consignment risks, PCA history Risks, Declaration Risks, Domestic risk, Risk of Intelligence, Random Selected Risk Dependant variable Y from Audit selection data frame column PA.

### 3.5.2 Split data into training, test, validation

From Pre-processed data 80% training set, 10% test set, and 10% validation set. The data after pre-processing is a 15070 data set, from this 12056 training set, 1507 test set, and 1507 validation set.

# 3.6 Model selection criteria

Model selection in deep learning is the process of choosing the best-suited model for predictive analytics. We Review and compare different research papers to understand the complexity, interpretability, and performance model. Predictive analytics for controlling tax evasion (Kumar, 2018) in this research control tax evasion related to circular trading by using

machine learning algorithm and Benford's analysis. The machine learning is not handle large tax datasets and Benford's analysis is more traditional analysis compare to predictive analytics. Audit lead selection and yield prediction from historical tax data using artificial neural networks (Chan et al., 2022) this research use artificial neural networks but not been statistically validated, and do not learn from historical data and audit outcomes. Predictive analytics and the targeting of audits(Hashimzade et al., 2015) this research used Tobit and logit regression models which are not efficient to handle large complex tax datasets and the traditional. Gated Mixture Vibrational AutoEncoders for Value Added Tax audit case selection (Kleanthous & Chatzis, 2020)this research used Gated Mixture Vibrational AutoEncoder deep network but the model not optimized. To overcome above gaps, we develop predictive analytics for controlling tax evasion using deep learning.

## 3.7 System evaluation techniques

### 3.7.1 Measuring performance

Performance metric can be defined as a logical and mathematical construct designed to measure how close the predicted outcome is to the actual result. Performance metrics (Evaluation metrics or error metrics) are crucial components of regression analysis and deep learning-based prediction models (Plevris, V. et, al., 2022)

The Bias Error e = ei ∈ [i], with [i] = {1, 2… N}, can be expressed as the difference between the predicted values and the real (target) values, it can take positive, negative or zero values for each observation, and is stated as

Let p (N×1 vector) be the predicted values and r (N×1 vector) be the real values of a quantity calculated (or measured) and then predicted a number N of times.

$$ei = pi + ri$$ ……..Equation 1

In above (Equation 1)the variables represents:-

$ei$ - Bias Error

$pi$- predicted values

$ri$- real values

The performance metrics are following: -

### 3.7.1.1 Mean Bias (MB)

The mean bias is the average of the bias errors. It can also take both positive, negative, or zero values and it is given by (Plevris, V. et.al. 2022).

$$MB = \bar{e} = \frac{1}{N}\sum_{i=1}^{N} ei = \frac{1}{N}\sum_{i=1}^{N}(pi - ri) = \bar{p} - \bar{r} \quad \text{……Equation 2}$$

In above (Equation 2) $\bar{e}$ represents mean bias, whereas, $\bar{p}$ and $\bar{r}$ are the mean value of p and r, respectively and N represents number of times

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} pi$$

$$\bar{r} = \frac{1}{N}\sum_{i=1}^{N} ri$$

Positive MB: If the Mean Bias is positive, it indicates that the model tends to underestimate the actual values on average. This means the predictions are generally lower than the observed values.

Negative MB: If the Mean Bias is negative, it indicates overestimation by the model, with regard to the actual values; the forecasted values are relatively high compared to observed values.

MB Close to Zero: Closeness of the value of MB to zero indicates that, on average, the predictions of a model are unbiased, and there was a good balance between overestimations and underestimations.

### 3.7.1.2 Mean squared error (MSE)

The Mean Squared Error is one of the most in-demand metrics related to regression, based on the mean squared difference between predicted and actual values (Plevris, V. et.al. 2022). It takes positive or zero values and is given by

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Pi - ri)\,2 \quad \text{……Equation 3}$$

In above (Equation 3) the MSE is mean squared error, $pi$ is predicted value, $ri$ is real value and N is the numbers of times.

MSE a value close to zero suggests that the predictions are very close to the actual values, meaning the model is accurately capturing the underlying data patterns.

#### 3.7.1.3 Root Mean squared error (RMSE)

The Root Mean Squared Error (RMSE) is also a frequently used measure of the differences between values (sample or population values) predicted by a model, or an estimator and the values observed. It is the square root of MSE. Unlike MSE, RMSE provides an error measure in the same unit as the target variable (Plevris, V.et.al, 2022).

$RMSE = \sqrt{MSE}$

$RMSE = \sqrt{MSE} = \sqrt{(\frac{1}{N}\sum_{i=1}^{N}(pi - ri)2)}$……..Equation 4

In above (Equation 4) the MSE is mean squared error, RMSE is root mean squared error, $pi$ is predicted value, $ri$ is real value and N is the numbers of times.

RMSE, A near-zero RMSE indicates that the average prediction error is minimal, which is desirable.

#### 3.7.1.4 R-squared

R-squared is also called a coefficient of determination (Plevris, V. al. et. 2022).

The formula for calculating R-squared is as shown below:

$R2 = \frac{Explained\ variation}{Total\ variation} = \frac{\sum_{i=1}^{N}(p^i - r)}{\sum_{i=1}^{N}(pi - \bar{r})}$ ……Equation 5

In above (Equation 5) $R2$ is R-squared, $\bar{r}$ is mean of real value, $pi$ is predicted value, $r$ is real value and $p^i$ is powered predicted value.

The R-squared ranges between 0 to 1. If the model does not fit the algorithm, it will produce a negative result. If the value is higher or nearer to 1, then it acts as the better model

## 3.8 Hyperparameter tuning techniques

Hyperparameter tuning is a process of finding out the best combination of hyperparameters that works well for our model. A hyperparameter is any setting that we choose before training our models, such as the number of hidden units, learning rate, batch size, optimizer, epochs, and activation function.

To perform hyperparameter tuning, there are a lot of hyperparameter techniques like grid search, randomized search, Bayesian optimizer and so on. We selected RandomizedSearchCV from sci-kit-learn because randomized search tunes by random samples instead of evaluating all combinations, which can be more efficient than grid search

(Bartz, 2023). The randomized search work by param_distributions dictionary contains the hyperparameters we want to tune and their respective values to search over. We specify different options for each hyperparameter, and RandomizedSearchCV will randomly sample combinations from these options.

We set n_iter to the number of parameter settings that are sampled, CV to the number of cross-validation folds, and random_state for reproducibility.

Once we call the fit method on the RandomizedSearchCV object, it will train and evaluate our model with different hyperparameter combinations. After the search is complete, we can access the best hyperparameters using the best_params_ attribute.

In this way, we can automatically search for the best hyperparameters without the need for manual trial and error. This helps us find the optimal settings for our model, leading to improved performance.

We also used the Bayesian optimizer from review paper (Sólon da Silva et al., 2023.), which generate 50 trial object for each models , suggest best optimal condition of models.

So we compared the result from RandomizedSearchCV object with Bayesian optimizer.

# CHAPTER FOUR

# PROPOSED SOLUTION AND IMPLEMENTATION

## Introduction

This proposed solution and implementations was conducted to develop predictive analytics for controlling tax evasion using deep learning in the case of the Ethiopia customs commission taxpayers. We model with different algorithm predictive analytics with deep neural networks, AutoEncoder, and feed-forward neural networks using random search CV, Bayesian optimizer and transfer learning.

## 4.1. The models using random search CV

### 4.1.1 Deep neural network

The DNN model is defined using a sequential architecture. It consists of An input layer that accepts input data with a shape matching the number of features in X_train; three hidden layers 64, 32, and 16 units respectively with ReLu activation; dropout layers are added after each of the hidden layers to reduce overfitting by randomly setting a fraction (20%) and an output layer with a single unit and a linear activation, suitable for regression tasks. The model is compiled using the RMSprop optimizer and mean squared error (MSE) as the loss function.

```
# Compile the model
dnn_model.compile(optimizer='rmsprop', loss='mean_squared_error', metrics=['mse'])
```

We used An EarlyStopping callback to monitor the validation loss during training. If the validation loss does not improve for 10 consecutive epochs, training will stop. This helps prevent overfitting by stopping the training process at the right time.

```
# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
```

The model is trained on the training dataset (X_train, y_train) for up to 100 epochs, with a batch size of 128. The validation data (X_val, y_val) is used to evaluate the model's performance after each epoch. After training, predictions are made on the test set (X_test).

```
# Fit the model
dnn_model.fit(X_train, y_train, epochs=100, batch_size=128,
              validation_data=(X_val, y_val), callbacks=[early_stopping])

# Make predictions
dnn_predictions = dnn_model.predict(X_test)
```

Then evaluate the model using various evaluation metrics: mean squared error, mean bias, root mean squared error, r-squared, and explained variance score.

Then after create customer selection logic using DNN model. The logic loop iterates over a range of years (2024 to 2028) to select customers for audit based on the predictions made by the DNN model. The predictions are made for the training data and the results are sorted to select the top PA values in train data. It selects customers for this year excludes it from the next selection and does the same for five consecutive years. And identifying three variables based on the highest number of red risk levels. The model is selected by excluding previously selected for the next selection, and then the data size is minimized so that it affects the next selection prediction. To handle this model is further fine-tuned on data representing the last year by using transfer learning. .

This allows the model to adapt to potentially new patterns in the data. Finally, the model can predict five consecutive years to be audit select customers and the top 3 variables highest number of red risk levels.

**Pseudocode**

1. Import necessary libraries: NumPy, Pandas, Matplotlib, Scikit-learn (for datasets, models, and metrics) and TensorFlow (for building neural networks)
2. Define the Deep Neural Network (DNN) model:
   a. Initialize a Sequential model.
   b. Add an Input layer with shape matching the number of features in X_train.
   c. Add several dense layers with 'ReLu' activation functions and Dropout layers.
   d. Add an output dense layer with 'linear' activation for regression.
3. Compile the DNN model: Use RMSprop optimizer and set loss to 'mean_squared_error'.
4. Set up EarlyStopping callback to monitor validation loss.
5. Fit the DNN model on the training data: Include validation split to monitor performance during training.
6. Make predictions using the trained DNN model on the test set.

7. Evaluate model performance: Calculate Mean Squared Error (MSE), calculate Root Mean Squared Error (RMSE), calculate R-squared (R2), calculate Explained Variance Score (EVS), and calculate Mean Bias.
8. Print evaluation metrics for the model.
9. Customer selection logic:
   a. Determine number of customers to select based on from datasets.
   b. Initialise empty structures for selected customers and target values.
10. For each year in range 2024 to 2028(for five consecutive years):
   a. Predict target values using the model on training data.
   b. Sort datasets by predicted target values.
   c. Exclude previously selected target values.
   d. Select highest risk customers based on highest target values.
   e. Save selected customers to based on years.
11. Identify columns with highest risk values and print the top three columns.
12. Fine-tune the DNN model on last year's data using transfer learning:
   a. Generate random features and target data for last year.
   b. Fit the DNN model on this new data.
13. Make final predictions using the fine-tuned DNN model.
14. Update datasets with last year's predictions and sort by target values.
15. Filter out duplicates in datasets based on target values and TIN numbers.
16. Print selected customers for each year.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_65 (Dense) | (None, 64) | 704 |
| dropout_20 (Dropout) | (None, 64) | 0 |
| dense_66 (Dense) | (None, 32) | 2,080 |
| dropout_21 (Dropout) | (None, 32) | 0 |
| dense_67 (Dense) | (None, 16) | 528 |
| dropout_22 (Dropout) | (None, 16) | 0 |
| dense_68 (Dense) | (None, 1) | 17 |

Total params: 3,329 (13.00 KB)
Trainable params: 3,329 (13.00 KB)
Non-trainable params: 0 (0.00 B)

Figure 4 -1 Initial DNN model architecture summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_75 (Dense) | (None, 64) | 704 |
| dropout_28 (Dropout) | (None, 64) | 0 |
| dense_76 (Dense) | (None, 32) | 2,080 |
| dropout_29 (Dropout) | (None, 32) | 0 |
| dense_77 (Dense) | (None, 16) | 528 |
| dropout_30 (Dropout) | (None, 16) | 0 |
| dense_78 (Dense) | (None, 1) | 17 |

Total params: 6,660 (26.02 KB)
Trainable params: 3,329 (13.00 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 3,331 (13.02 KB)

Figure 4-2 Fine-tuned DNN model architecture summary

Layers details in DNN model architecture (Figure 4-1 and Figure 4-2):

1. Dense layer: are fully connected layers where each neuron is connected to every neuron in the previous layer

Output shape none indicates that the output shape is dynamic and depends on the batch size during training. Each neuron has weights for each input plus a bias term. Our input is the number of features (n).

The parameter of each layer is, Parameter = nxunits+units

The number of features in the first dense layer is the input shape of the model is 10

Units are 64 neurons

Parameter = 10x64+64=704

The number of features for the next dense layer is 64

Units are 32 neurons

Parameter = 32x64 +32=2080

The number of features for the third dense layer is 32

Units is 16

Parameter = 16 x32 +16=528

The number of features for the fourth dense layer is 16

Units are 1

Parameter = 1x16+1=17

Dropout layer: a regularization technique that randomly sets a fraction of input units to 0, which helps prevent overfitting.

First dropout layer, Output shape(none, rate 64), params 0, none indicates the output shape is dynamic, dropout 64% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

Next, the dropout layer, Output shape(none, rate 32), params 0, none indicates the output shape is dynamic, dropout 32% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

Third dropout layer, Output shape(none, rate 16), params 0, none indicates the output shape is dynamic, dropout 16% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

The initial DNN model's total number of parameters is 3329(13.00KB), which measures how much memory the model occupies when stored. It includes weights and biases across all layers of the model. Trainable parameters 3329(13.00KB) are parameters that will updated during the training process through backpropagation. Non-trainable parameters 0(0.00B) are parameters that will not updated or fixed during the training process.

The fine-tuned DNN model's total number of parameters is increased to 6660(26.02KB), this indicates that the model has grown in complexity or capacity during the fine-tuning process. Trainable parameters are 3329(13.00KB) is remain the same as the initial model, this means the original parameters are still being updated during training. Non-trainable parameters remain at zero, this reinforces that the entire model is trainable, which can be advantageous for learning complex patterns in the data. Optimizer parameters 3331(13.02KB) refers to the parameters used by the optimizer itself; this maintains additional parameters using rmsprop optimizer to help with the training process. This number is slightly higher than the model parameters; this means the optimizer is storing some additional information to facilitate the learning process.

To summarise the DNN model architecture, the total parameters of the model start with 3329 parameters, which is relatively small which means a lightweight architecture; all parameters are trainable, which helps for learning from the data; there are no non-trainable parameters, which means the model is full adaptability. After fine-tuning, the total parameter increases to 6660, which means the model has been modified to potentially capture more complex relationships in the data. The parameter size shows the model is efficient in terms of memory usage while being capable of learning from the data effectively.

### 4.1.2 AutoEncoder

The AutoEncoder model is defined using a sequential architecture with an input layer, an encoder part, and a decoder part. The encoder part consists of dense layers with ReLu

activation and L2 regularization to prevent overfitting. The number of units in each layer decreases, creating a bottleneck that forces the model to learn a compressed representation of the input (64, 64, 32, 8 units). The decoder part consists of a single dense layer with a linear activation function to reconstruct the input. The model is compiled using the Adam optimizer and mean squared error (MSE) as the loss function.

```
# Compile the model
autoencoder_model.compile(optimizer=Adam(learning_rate=0.0001), loss='mean_squared_error', metrics=['mse'])
```

An EarlyStopping callback is used to monitor the validation loss and stop training if the loss does not improve for a specified number of epochs 5. This helps prevent overfitting.

```
# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5)
```

The model is training data (X_train, y_train) for 100 epochs with a batch size of 64. The validation data (X_val, y_val) is used to monitor the model's performance during training and to trigger the EarlyStopping callback if necessary. After training, the model is used to make predictions on the test set (X_test).

```
# Fit the model
autoencoder_model.fit(X_train, y_train, epochs=100, batch_size=64,
                      callbacks=[early_stopping], validation_data=(X_val, y_val))
# Make predictions
autoencoder_predictions = autoencoder_model.predict(X_test)
```

Then evaluate the model using various evaluation metrics: mean squared error, mean bias, root mean squared error, r-squared, and explained variance score.

Then after creating customer selection logic using the AutoEncoder model. The logic loop iterates over a range of years (2024 to 2028) to select customers for audit based on the predictions made by the AutoEncoder model. The predictions are made for the training data and the results are sorted to select the top PA values in train data. It selects customers for this year excludes it from the next selection and does the same for five consecutive years. And identifying three variables based on the highest number of red risk levels. The model is selected by excluding previously selected for the next selection, and then the data size is minimized so that it affects the next selection prediction. To handle this model is further fine-tuned on data representing the last year by using transfer learning.

This allows the model to adapt to potentially new patterns in the data. Finally, the model can predict five consecutive years to be audit select customers and the top 3 variables highest number of red risk levels

**Pseudocode**

1. Import necessary libraries: NumPy, Pandas, Matplotlib, Scikit-learn (for datasets, models, and metrics) and TensorFlow (for building neural networks)

2. Define the AutoEncoder model:

   a. Initialize a Sequential model.

   b. Add an Input layer with shape matching the number of features in X_train.

   c. Add several dense layers with 'ReLu' activation functions and Dropout layers.

   d. Add an output dense layer with 'linear' activation for regression.

3. Compile the AutoEncoder model: Use Adam optimizer and set loss to 'mean_squared_error'.

4. Set up EarlyStopping callback to monitor validation loss.

5. Fit the AutoEncoder model on the training data: Include validation split to monitor performance during training.

6. Make predictions using the trained AutoEncoder model on the test set.

7. Evaluate model performance: Calculate Mean Squared Error (MSE), calculate Root Mean Squared Error (RMSE), calculate R-squared (R2), calculate Explained Variance Score (EVS), and calculate Mean Bias.

8. Print evaluation metrics for the model.

9. Customer selection logic:

   a. Determine number of customers to select based on from datasets.

   b. Initialise empty structures for selected customers and target values.

10. For each year in range 2024 to 2028(for five consecutive years):

   a. Predict target values using the model on training data.

   b. Sort datasets by predicted target values.

   c. Exclude previously selected target values.

   d. Select highest risk customers based on highest target values.

   e. Save selected customers to based on years.

11. Identify columns with highest risk values and print the top three columns.

12. Fine-tune the AutoEncoder model on last year's data using transfer learning:

   a. Generate random features and target data for last year.

   b. Fit the AutoEncoder model on this new data.

13. Make final predictions using the fine-tuned AutoEncoder model.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_12 (InputLayer) | (None, 10) | 0 |
| dense_41 (Dense) | (None, 64) | 704 |
| dropout_9 (Dropout) | (None, 64) | 0 |
| dense_42 (Dense) | (None, 64) | 4,160 |
| dropout_10 (Dropout) | (None, 64) | 0 |
| dense_43 (Dense) | (None, 32) | 2,080 |
| dense_44 (Dense) | (None, 8) | 264 |
| dense_45 (Dense) | (None, 1) | 9 |

Total params: 7,217 (28.19 KB)
Trainable params: 7,217 (28.19 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-3 Initial AutoEncoder model architecture summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_30 (InputLayer) | (None, 10) | 0 |
| dense_109 (Dense) | (None, 64) | 704 |
| dropout_37 (Dropout) | (None, 64) | 0 |
| dense_110 (Dense) | (None, 64) | 4,160 |
| dropout_38 (Dropout) | (None, 64) | 0 |
| dense_111 (Dense) | (None, 32) | 2,080 |
| dense_112 (Dense) | (None, 8) | 264 |
| dense_113 (Dense) | (None, 1) | 9 |

Total params: 21,653 (84.59 KB)
Trainable params: 7,217 (28.19 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 14,436 (56.39 KB)

Figure 4-4 Fine-tuned AutoEncoder model architecture summary

Layers details in AutoEncoder model architecture (Table 4-3 and Table 4-4):

1. Dense layer: are fully connected layers where each neuron is connected to every neuron in the previous layer. Output shape none indicates that the output shape is dynamic and depends on the batch size during training. Each neuron has weights for each input plus a bias term. Our input is the number of features (n).

The parameter of each layer is, Parameter = nxunits+units

The number of features in the first dense layer is the input shape of the model is 10

Units are 64 neurons

Parameter = 10x64+64=704

The number of features for the next dense layer is 64

Units are 64 neurons

Parameter = 64x64 +64=4160

The number of features for the third dense layer is 64

Units is 32

Parameter = 32x64 +32=2080

The number of features for the fourth dense layer is 32

Units are 8

Parameter = 8x32+8=264

The number of features for the fifth  dense layer is 8

Units are 1

Parameter = 1x8+1=9

2. Dropout layer: a regularization technique that randomly sets a fraction of input units to 0, which helps prevent overfitting.

First dropout layer, Output shape(none, rate 64), params 0,  none indicates the output shape is dynamic, dropout 64%  of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

Next, the dropout layer, Output shape(none, rate 64), params 0,  none indicates the output shape is dynamic, dropout 64%  of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

The initial AutoEncoder model's total number of parameters is 7217(28.19KB), which measures how much memory the model occupies when stored. It includes weights and biases across all layers of the model.  Trainable parameters 7217(28.19KB) are parameters that will updated during the training process through backpropagation. Non-trainable parameters 0(0.00B) are parameters that will not updated or fixed during the training process.

The fine-tuned AutoEncoder model's total number of parameters is increased to 21,653(84.59KB), this indicates that the model has grown in complexity or capacity during the fine-tuning process. Trainable parameters are 7217(28.19KB) is remain the same as the initial model; this means the original parameters are still being updated during training. Non-trainable parameters remain at zero, this reinforces that the entire model is trainable, which can be advantageous for learning complex patterns in the data. Optimizer parameters 14436(56.39KB) refers to the parameters used by the optimizer itself; this maintains additional parameters using ReLu optimizer to the training process. This number is double to the model parameters; this means the optimizer is storing large amount of additional information to facilitate the learning process.

To summarise the AutoEncoder model architecture, the total parameters of the model start with 7217 parameters, which is relatively large which means a complex architecture; all

parameters are trainable, which helps for learning from the data; there are no non-trainable parameters, which means the model is full adaptability. After fine-tuning, the total parameter increases to 21,653, which means the model has been modified to potentially capture more complex relationships in the data. The parameter size shows the model is efficient in terms of memory usage while being capable of learning from the data effectively.

### 4.1.3 Feed forward neural network

The FFNN model is defined using a sequential architecture, where: an input layer is created with a shape matching the number of features in X_train (12056, 10); two hidden layers with 64 and 32 units respectively, both using ReLu activation and an output layer with a single unit, suitable for regression tasks. The model is compiled with the Adam optimizer and mean squared error (MSE) as a loss function.

```
# Compile the model
ffnn_model.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=0.0001))
```

The model is trained on the training (X_train, y_train) for 50 epochs, with a batch size of 32. The validation data (X_val, y_val) is used to monitor the model's performance during training. An EarlyStopping callback is implemented to halt training if the validation loss does not improve for 5 epochs, which helps prevent overfitting. After training, the model is evaluated on the test (X_test), and predictions are made.

```
# Compile the model
ffnn_model.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=0.0001))

# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

# Fit the model on the first four years of data
ffnn_model.fit(X_train, y_train, epochs=50, batch_size=32, callbacks=[early_stopping], validation_data=(X_val, y_val))

# Evaluate the model on the test set
ffnn_predictions = ffnn_model.predict(X_test)
```

Then evaluate the model using various evaluation metrics: mean squared error, mean bias, root mean squared error, r-squared, and explained variance score.

Then after creating customer selection logic using the FFNN model. The logic loop iterates over a range of years (2024 to 2028) to select customers for audit based on the predictions made by the FFNN model. The predictions are made for the training data and the results are sorted to select the top PA values in train data. It selects customers for this year excludes it from the next selection and does the same for five consecutive years. And identifying three variables based on the highest number of red risk levels. The model is selected by excluding previously selected for the next selection, and then the data size is minimized so that it affects the next selection prediction. To handle this model is further fine-tuned on data representing the last year by using transfer learning.

This allows the model to adapt to potentially new patterns in the data. Finally, the model can predict five consecutive years to be audit select customers and the top three variables highest number of red risk levels.

**Pseudocode**

1. Import necessary libraries: NumPy, Pandas, Matplotlib, Scikit-learn (for datasets, models, and metrics) and TensorFlow (for building neural networks)
2. Define the Feedforward Neural Network (FFNN) model:
   a. Initialize a Sequential model.
   b. Add an Input layer with shape matching the number of features in X_train.
   c. Add several dense layers with 'ReLu' activation functions.
   d. Add an output dense layer with 'linear' activation for regression.
3. Compile the FFNN model: Use Adam optimizer and set loss to 'mean_squared_error'.
4. Set up EarlyStopping callback to monitor validation loss.
5. Fit the FFNN model on the training data: Include validation split to monitor performance during training.
6. Make predictions using the trained FFNN model on the test set.
7. Evaluate FFNN model performance: Calculate Mean Squared Error (MSE), calculate Root Mean Squared Error (RMSE), calculate R-squared (R2), calculate Explained Variance Score (EVS), and calculate Mean Bias.
8. Print evaluation metrics for the model.
9. Customer selection logic:
   a. Determine number of customers to select based on from datasets.
   b. Initialise empty structures for selected customers and target values.
10. For each year in range 2024 to 2028(for five consecutive years):
   a. Predict target values using the model on training data.
   b. Sort datasets by predicted target values.
   c. Exclude previously selected target values.
   d. Select highest risk customers based on highest target values.
   e. Save selected customers to based on years.
11. Identify columns with highest risk values and print the top three columns.
12. Fine-tune the FFNN model on last year's data using transfer learning:
   a. Generate random features and target data for last year.
   b. Fit the FFNN model on this new data.
13. Make final predictions using the fine-tuned FFNN model.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_90 (Dense) | (None, 64) | 704 |
| dense_91 (Dense) | (None, 32) | 2,080 |
| dense_92 (Dense) | (None, 1) | 33 |

Total params: 2,817 (11.00 KB)
Trainable params: 2,817 (11.00 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-5 Initial FFNN model architecture summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_106 (Dense) | (None, 64) | 704 |
| dense_107 (Dense) | (None, 32) | 2,080 |
| dense_108 (Dense) | (None, 1) | 33 |

Total params: 8,453 (33.02 KB)
Trainable params: 2,817 (11.00 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 5,636 (22.02 KB)

Figure 4-6 Fine-tuned FFNN model architecture summary

Layers details in FFNN model architecture (Figure 4-5 and Figure 4-6):

1. Dense layer: are fully connected layers where each neuron is connected to every neuron in the previous layer. Output shape none indicates that the output shape is dynamic and depends on the batch size during training. Each neuron has weights for each input plus a bias term. Our input is the number of features (n).

The parameter of each layer is, Parameter = nxunits+units

The number of features in the first dense layer is the input shape of the model is 10

Units are 64 neurons

Parameter = 10x64+64=704

The number of features for the next dense layer is 64

Units are 32 neurons

Parameter = 32x64 +32=2080

The number of features for the third dense layer is 32

Units is 1

Parameter = 1 x32 +1=33

The initial FFNN model's total number of parameters is 2817(11.00KB), which measures how much memory the model occupies when stored. It includes weights and biases across all

layers of the model.  Trainable parameters 2817(11.00KB) are parameters that will updated during the training process through backpropagation. Non-trainable parameters 0(0.00B) are parameters that will not updated or fixed during the training process.

The fine-tuned FFNN model's total number of parameters is increased to 8453(33.02KB), this indicates that the model has grown in complexity or capacity during the fine-tuning process. Trainable parameters are 2817(11.00KB) is remain the same as the initial model; this means the original parameters are still being updated during training. Non-trainable parameters remain at zero, this reinforces that the entire model is trainable, which can be advantageous for learning complex patterns in the data. Optimizer parameters 5636(22.02KB) refers to the parameters used by the optimizer itself; this maintains additional parameters using ReLu optimizer to help the training process. This number is almost double to the model parameters; this means the optimizer is storing a lot of additional information to facilitate the learning process.

To summarise the FFNN model architecture, the total parameters of the model start with 2817 parameters, which is relatively small which means a lightweight architecture; all parameters are trainable, which helps for learning from the data; there are no non-trainable parameters, which means the model is full adaptability. After fine-tuning, the total parameter increases to 8453, which means the model has been modified to potentially capture more complex relationships in the data. The parameter size shows the model is efficient in terms of memory usage while being capable of learning from the data effectively.

## 4.2. The model using Bayesian optimization

### 4.2.1 Deep neural network

The DNN model is defined using a sequential architecture. It consists of An input layer that accepts input data with a shape matching the number of features in X_train; three hidden layers 71, 19, and 25 units respectively with sigmoid activation; dropout layers are added after each of the hidden layers to reduce overfitting by randomly setting (0.12483) and an output layer with a single unit and a linear activation, suitable for regression tasks. The model is compiled using the Adam optimizer and mean squared error (MSE) as the loss function.

```
# Compile the model
dnn_model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mse'])
```

We used An EarlyStopping callback to monitor the validation loss during training. If the validation loss does not improve for 10 consecutive epochs, training will stop. This helps prevent overfitting by stopping the training process at the right time.

```
# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
```

The model is trained on the training dataset (X_train, y_train) for up to 150 epochs, with a batch size of 64. The validation data (X_val, y_val) is used to evaluate the model's performance after each epoch. After training, predictions are made on the test set (X_test).

Then evaluate the model using various evaluation metrics: mean squared error, mean bias, root mean squared error, r-squared, and explained variance score.

```
dnn_model.fit(X_train, y_train, epochs=150, batch_size=64,
              validation_data=(X_val, y_val), callbacks=[early_stopping])

# Make predictions
dnn_predictions = dnn_model.predict(X_test)
```

Then after create customer selection logic using DNN model. The logic loop iterates over a range of years (2024 to 2028) to select customers for audit based on the predictions made by the DNN model. The predictions are made for the training data and the results are sorted to select the top PA values in train data. It selects customers for this year excludes it from the next selection and does the same for five consecutive years. And identifying three variables based on the highest number of red risk levels. The model is selected by excluding previously selected for the next selection, and then the data size is minimized so that it affects the next selection prediction. To handle this model is further fine-tuned on data representing the last year by using transfer learning. .

This allows the model to adapt to potentially new patterns in the data. Finally, the model can predict five consecutive years to be audit select customers and the top 3 variables highest number of red risk levels.

**Pseudocode**

1. Import necessary libraries: NumPy, Pandas, Matplotlib, Scikit-learn (for datasets, models, and metrics) and TensorFlow (for building neural networks)
2. Define the Deep Neural Network (DNN) model:
   a. Initialize a Sequential model.
   b. Add an Input layer with shape matching the number of features in X_train.
   c. Add several dense layers with 'sigmoid' activation functions and dropout layers.
   d. Add an output dense layer with 'linear' activation for regression.

3. Compile the DNN model: Use Adam optimizer and set loss to 'mean_squared_error'.

4. Set up EarlyStopping callback to monitor validation loss.

5. Fit the DNN model on the training data: Include validation split to monitor performance during training.

6. Make predictions using the trained DNN model on the test set.

7. Evaluate model performance: Calculate Mean Squared Error (MSE), calculate Root Mean Squared Error (RMSE), calculate R-squared (R2), calculate Explained Variance Score (EVS), and calculate Mean Bias.

8. Print evaluation metrics for the model.

9. Customer selection logic:

   a. Determine number of customers to select based on from datasets.

   b. Initialise empty structures for selected customers and target values.

10. For each year in range 2024 to 2028(for five consecutive years):

   a. Predict target values using the model on training data.

   b. Sort datasets by predicted target values.

   c. Exclude previously selected target values.

   d. Select highest risk customers based on highest target values.

   e. Save selected customers to based on years.

11. Identify columns with highest risk values and print the top three columns.

12. Fine-tune the DNN model on last year's data using transfer learning:

   a. Generate random features and target data for last year.

   b. Fit the DNN model on this new data.

13. Make final predictions using the fine-tuned DNN model.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_65 (Dense) | (None, 71) | 781 |
| dropout_29 (Dropout) | (None, 71) | 0 |
| dense_66 (Dense) | (None, 19) | 1,368 |
| dropout_30 (Dropout) | (None, 19) | 0 |
| dense_67 (Dense) | (None, 25) | 500 |
| dropout_31 (Dropout) | (None, 25) | 0 |
| dense_68 (Dense) | (None, 1) | 26 |

Total params: 2,675 (10.45 KB)
Trainable params: 2,675 (10.45 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-7 Initial DNN model architecture summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_65 (Dense) | (None, 71) | 781 |
| dropout_29 (Dropout) | (None, 71) | 0 |
| dense_66 (Dense) | (None, 19) | 1,368 |
| dropout_30 (Dropout) | (None, 19) | 0 |
| dense_67 (Dense) | (None, 25) | 500 |
| dropout_31 (Dropout) | (None, 25) | 0 |
| dense_68 (Dense) | (None, 1) | 26 |

Total params: 8,027 (31.36 KB)
Trainable params: 2,675 (10.45 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 5,352 (20.91 KB)

Figure 4-8 Fine-tuned DNN model architecture summary

Layers details in DNN model architecture (Figure 4-7 and Figure 4-8):

Dense layer: fully connected layers where each neuron is connected to every neuron in the previous layer. Output shape none indicates that the output shape is dynamic and depends on the batch size during training. Each neuron has weights for each input plus a bias term. Our input is the number of features (n).

The parameter of each layer is, Parameter = nxunits+units

The number of features in the first dense layer is the input shape of the model is 10

Units are 71 neurons

Parameter = 10x71+71=781

The number of features for the next dense layer is 71

Units are 19 neurons

Parameter = 19x71 +19=1368

The number of features for the third dense layer is 19

Units is 25 neurons

Parameter = 25 x19 +25=500

The number of features for the fourth dense layer is 25

Units are 1 neuron

Parameter = 1x25+1=26

2. Dropout layer: a regularization technique that randomly sets a fraction of input units to 0, which helps prevent overfitting.

First dropout layer, Output shape(none, rate 71), params 0, none indicates the output shape is dynamic, dropout 71% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

Next, the dropout layer, Output shape(none, rate 19), params 0, none indicates the output shape is dynamic, dropout 19% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

Third dropout layer, Output shape(none, rate 25), params 0, none indicates the output shape is dynamic, dropout 25% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

The initial DNN model's total number of parameters is 2675(10.45KB), which measures how much memory the model occupies when stored. It includes weights and biases across all layers of the model. Trainable parameters 2675(10.45KB) are parameters that will updated during the training process through backpropagation. Non-trainable parameters 0(0.00B) are parameters that will not updated or fixed during the training process.

The fine-tuned DNN model's total number of parameters is increased to 8027(31.36KB), this indicates that the model has grown in complexity or capacity during the fine-tuning process. Trainable parameters are 2675(10.45KB) is remain the same as the initial model; this means the original parameters are still being updated during training. Non-trainable parameters remain at zero, this reinforces that the entire model is trainable, which can be advantageous for learning complex patterns in the data. Optimizer parameters 5352(20.91KB) refers to the parameters used by the optimizer itself; this maintains additional parameters using Adam optimizer to help with the training process. This number is double to the model parameters; this means the optimizer is storing more additional information to facilitate the learning process.

To summarise the DNN model architecture, the total parameters of the model start with 2675 parameters, which is relatively small which means a lightweight architecture; all parameters are trainable, which helps for learning from the data; there are no non-trainable parameters, which means the model is full adaptability. After fine-tuning, the total parameter increases to 8027, which means the model has been modified to potentially capture more complex relationships in the data. The parameter size shows the model is efficient in terms of memory usage while being capable of learning from the data effectively.

### 4.2.2 AutoEncoder

The AutoEncoder model is defined using a sequential architecture with an input layer, an encoder part, and a decoder part. The encoder part consists of dense layers with tanh

activation and L2 regularization to prevent overfitting. The number of units in each layer decreases, creating a bottleneck that forces the model to learn a compressed representation of the input (88, 40, 22, 15 units). The decoder part consists of a single dense layer with a linear activation function to reconstruct the input. The model is compiled using the Adam optimizer and mean squared error (MSE) as the loss function.

An EarlyStopping callback is used to monitor the validation loss and stop training if the loss does not improve for a specified number of epochs 5. This helps prevent overfitting.

```
# Compile the model
autoencoder_model.compile(optimizer=Adam(learning_rate=0.00022027), loss='mean_squared_error', metrics=['mse'])

# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5)
```

The model is training data (X_train, y_train) for 96 epochs with a batch size of 64. The validation data (X_val, y_val) is used to monitor the model's performance during training and to trigger the EarlyStopping callback if necessary. After training, the model is used to make predictions on the test set (X_test).

```
# Fit the model
autoencoder_model.fit(X_train, y_train, epochs=96, batch_size=64,
                      callbacks=[early_stopping], validation_data=(X_val, y_val))
# Ensure predictions are 1-dimensional
autoencoder_predictions = autoencoder_model.predict(X_train).flatten()
```

Then evaluate the model using various evaluation metrics: mean squared error, mean bias, root mean squared error, r-squared, and explained variance score.

Then after creating customer selection logic using the AutoEncoder model. The logic loop iterates over a range of years (2024 to 2028) to select customers for audit based on the predictions made by the AutoEncoder model. The predictions are made for the training data and the results are sorted to select the top PA values in train data. It selects customers for this year excludes it from the next selection and does the same for five consecutive years. And identifying three variables based on the highest number of red risk levels. The model is selected by excluding previously selected for the next selection, and then the data size is minimized so that it affects the next selection prediction. To handle this model is further fine-tuned on data representing the last year by using transfer learning.

This allows the model to adapt to potentially new patterns in the data. Finally, the model can predict five consecutive years to be audit select customers and the top 3 variables highest number of red risk levels

**Pseudocode**

1. Import necessary libraries: NumPy, Pandas, Matplotlib, Scikit-learn (for datasets, models, and metrics) and TensorFlow (for building AutoEncoder model)
2. Define the AutoEncoder model:
   a. Initialize a Sequential model.
   b. Add an Input layer with shape matching the number of features in X_train.
   c. Add several dense layers with 'tanh' activation functions, dropout layers and L2 regularization.
   d. Add an output dense layer with 'linear' activation for regression.
3. Compile the AutoEncoder model: Use Adam optimizer and set loss to 'mean_squared_error'.
4. Set up EarlyStopping callback to monitor validation loss.
5. Fit the AutoEncoder model on the training data: Include validation split to monitor performance during training.
6. Make predictions using the trained AutoEncoder model on the test set.
7. Evaluate AutoEncoder model performance: Calculate Mean Squared Error (MSE), calculate Root Mean Squared Error (RMSE), calculate R-squared (R2), calculate Explained Variance Score (EVS), and calculate Mean Bias.
8. Print evaluation metrics for the model.
9. Customer selection logic:
   a. Determine number of customers to select based on from datasets.
   b. Initialise empty structures for selected customers and target values.
10. For each year in range 2024 to 2028(for five consecutive years):
   a. Predict target values using the model on training data.
   b. Sort datasets by predicted target values.
   c. Exclude previously selected target values.
   d. Select highest risk customers based on highest target values.
   e. Save selected customers to based on years.
11. Identify columns with highest risk values and print the top three columns.
12. Fine-tune the AutoEncoder model on last year's data using transfer learning:
   a. Generate random features and target data for last year.
   b. Fit the AutoEncoder model on this new data.
13. Make final predictions using the fine-tuned AutoEncoder model.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_11 (InputLayer) | (None, 10) | 0 |
| dense_41 (Dense) | (None, 88) | 968 |
| dropout_18 (Dropout) | (None, 88) | 0 |
| dense_42 (Dense) | (None, 40) | 3,560 |
| dropout_19 (Dropout) | (None, 40) | 0 |
| dense_43 (Dense) | (None, 22) | 902 |
| dense_44 (Dense) | (None, 15) | 345 |
| dense_45 (Dense) | (None, 1) | 16 |

**Total params:** 5,791 (22.62 KB)
**Trainable params:** 5,791 (22.62 KB)
**Non-trainable params:** 0 (0.00 B)

Figure 4-9 Initial AutoEncoder model architecture summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_11 (InputLayer) | (None, 10) | 0 |
| dense_41 (Dense) | (None, 88) | 968 |
| dropout_18 (Dropout) | (None, 88) | 0 |
| dense_42 (Dense) | (None, 40) | 3,560 |
| dropout_19 (Dropout) | (None, 40) | 0 |
| dense_43 (Dense) | (None, 22) | 902 |
| dense_44 (Dense) | (None, 15) | 345 |
| dense_45 (Dense) | (None, 1) | 16 |

**Total params:** 17,375 (67.88 KB)
**Trainable params:** 5,791 (22.62 KB)
**Non-trainable params:** 0 (0.00 B)
**Optimizer params:** 11,584 (45.25 KB)

Figure 4-10 Fine-tuned AutoEncoder architecture

Layers details in AutoEncoder model architecture (Figure 4-9 and Figure 4-10):

**Dense layer** fully connected layers where each neuron is connected to every neuron in the previous layer. Output shape none indicates that the output shape is dynamic and depends on the batch size during training. Each neuron has weights for each input plus a bias term. Our input is the number of features (n).

The parameter of each layer is, Parameter = nxunits+units

The number of features in the first dense layer is the input shape of the model is 10

Units are 88 neurons

Parameter = 10x88+88=968

The number of features for the next dense layer is 88

Units are 40 neurons

Parameter = 40x88 + 40=3560

The number of features for the third dense layer is 40

Units is 22 neurons

Parameter = 22x40 +22=902

The number of features for the fourth dense layer is 22

Units are 15 neurons

Parameter = 15x22+15=345

The number of features for the fifth dense layer is 15

Units are 1 neuron

Parameter = 1x15+1=16

**Dropout layers** a regularization technique that randomly sets a fraction of input units to 0, which helps prevent overfitting. First dropout layer, Output shape(none, rate 88), params 0, none indicates the output shape is dynamic, dropout 88% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters. Next, the dropout layer, Output shape(none, rate 40), params 0, none indicates the output shape is dynamic, dropout 40% of the neurons will be dropped during training, params 0 is the 0 parameters, as dropout has no learnable parameters.

The initial AutoEncoder model's total number of parameters is 5791(22.62KB), which measures how much memory the model occupies when stored. It includes weights and biases across all layers of the model. Trainable parameters 5791(22.62KB) are parameters that will updated during the training process through backpropagation. Non-trainable parameters 0(0.00B) are parameters that will not updated or fixed during the training process.

The fine-tuned AutoEncoder model's total number of parameters is increased to 17,375(67.88KB), this indicates that the model has grown in complexity or capacity during the fine-tuning process. Trainable parameters are 5791(22.62KB) is remain the same as the initial model; this means the original parameters are still being updated during training. Non-trainable parameters remain at zero, this reinforces that the entire model is trainable, which can be advantageous for learning complex patterns in the data. Optimizer parameters 11,584(42.25KB) refers to the parameters used by the optimizer itself; this maintains additional parameters using Adam optimizer to the training process. This number is double to

the model parameters; this means the optimizer is storing large amount of additional information to facilitate the learning process.

To summarise the AutoEncoder model architecture, the total parameters of the model start with 5791 parameters, which is relatively large which means a complex architecture; all parameters are trainable, which helps for learning from the data; there are no non-trainable parameters, which means the model is full adaptability. After fine-tuning, the total parameter increases to 17,375 which means the model has been modified to potentially capture more complex relationships in the data. The parameter size shows the model is efficient in terms of memory usage while being capable of learning from the data effectively.

### 4.2.3 Feed forward neural network

The FFNN model is defined using a sequential architecture, where: an input layer is created with a shape matching the number of features in X_train (12056, 10); two hidden layers with 96 and 48 units respectively, both using tanh activation and an output layer with a single unit, suitable for regression tasks. The model is compiled with the sgd optimizer and mean squared error (MSE) as a loss function.

```
# Compile the model
ffnn_model.compile(loss='mean_squared_error', optimizer=SGD(learning_rate=0.00003511))
```

The model is trained on the training (X_train, y_train) for 150 epochs, with a batch size of 32. The validation data (X_val, y_val) is used to monitor the model's performance during training. An EarlyStopping callback is implemented to halt training if the validation loss does not improve for 5 epochs, which helps prevent overfitting. After training, the model is evaluated on the test (X_test), and predictions are made.

```
# Compile the model
ffnn_model.compile(loss='mean_squared_error', optimizer=SGD(learning_rate=0.00003511))

# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

# Fit the model on the first four years of data
ffnn_model.fit(X_train, y_train, epochs=150, batch_size=32, callbacks=[early_stopping], validation_data=(X_val, y_val))

# Evaluate the model on the test set
ffnn_predictions = ffnn_model.predict(X_test)

# Ensure predictions are 1-dimensional
ffnn_predictions = ffnn_predictions.flatten()
```

Then evaluate the model using various evaluation metrics: mean squared error, mean bias, root mean squared error, r-squared, and explained variance score.

Then after creating customer selection logic using the FFNN model. The logic loop iterates over a range of years (2024 to 2028) to select customers for audit based on the predictions made by the FFNN model. The predictions are made for the training data and the results are sorted to select the top PA values in train data. It selects customers for this year excludes it from the next selection and does the same for five consecutive years. And identifying three variables based on the highest number of red risk levels. The model is selected by excluding previously selected for the next selection, and then the data size is minimized so that it affects the next selection prediction. To handle this model is further fine-tuned on data representing the last year by using transfer learning.

This allows the model to adapt to potentially new patterns in the data. Finally, the model can predict five consecutive years to be audit select customers and the top three variables highest number of red risk levels.

**Pseudocode**

1. Import necessary libraries: NumPy, Pandas, Matplotlib, Scikit-learn (for datasets, models, and metrics) and TensorFlow (for building neural networks)
2. Define the Feedforward Neural Network (FFNN) model:
   a. Initialize a Sequential model.
   b. Add an Input layer with shape matching the number of features in X_train.
   c. Add several dense layers with 'tanh' activation functions and Dropout layers.
   d. Add an output dense layer with 'linear' activation for regression.
3. Compile the FFNN model: Use sgd optimizer and set loss to 'mean_squared_error'.
4. Set up EarlyStopping callback to monitor validation loss.
5. Fit the FFNN model on the training data: Include validation split to monitor performance during training.
6. Make predictions using the trained FFNN model on the test set.
7. Evaluate model performance: Calculate Mean Squared Error (MSE), calculate Root Mean Squared Error (RMSE), calculate R-squared (R2), calculate Explained Variance Score (EVS), and calculate Mean Bias.
8. Print evaluation metrics for the model.
9. Customer selection logic:
   a. Determine number of customers to select based on from datasets.
   b. Initialise empty structures for selected customers and target values.

10. For each year in range 2024 to 2028 (for five consecutive years):
    a. Predict target values using the model on training data.
    b. Sort datasets by predicted target values.
    c. Exclude previously selected target values.
    d. Select highest risk customers based on highest target values.
    e. Save selected customers to based on years.
11. Identify columns with highest risk values and print the top three columns.
12. Fine-tune the FFNN model on last year's data using transfer learning:
    a. Generate random features and target data for last year.
    b. Fit the FFNN model on this new data.
13. Make final predictions using the fine-tuned FFNN model.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_33 (Dense) | (None, 96) | 1,056 |
| dropout_15 (Dropout) | (None, 96) | 0 |
| dense_34 (Dense) | (None, 48) | 4,656 |
| dense_35 (Dense) | (None, 1) | 49 |

Total params: 5,761 (22.50 KB)
Trainable params: 5,761 (22.50 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-11 Initial FFNN model architecture summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_33 (Dense) | (None, 96) | 1,056 |
| dropout_15 (Dropout) | (None, 96) | 0 |
| dense_34 (Dense) | (None, 48) | 4,656 |
| dense_35 (Dense) | (None, 1) | 49 |

Total params: 5,763 (22.52 KB)
Trainable params: 5,761 (22.50 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)

Figure 4-12 Fine-tuned FFNN model architecture summary

Layers details in FFNN model architecture (Table 4-11 and Table 4-12):
Dense layer: Each neuron has weights for each input plus a bias term. Our input is the number of features (n).
The parameter of each layer is, Parameter = nxunits+units
The number of features in the first dense layer is the input shape of the model is 10

Units are 96 neurons

Parameter = 10x96+96=1056

The number of features for the next dense layer is 96

Units are 48 neurons

Parameter = 48x96 +48=4656

The number of features for the third dense layer is 48

Units is 1 neuron

Parameter = 1 x48 +1=49

The initial FFNN model's total number of parameters is 5761(22.50KB), which measures how much memory the model occupies when stored. It includes weights and biases across all layers of the model. Trainable parameters 5761(22.50KB) are parameters that will updated during the training process through backpropagation. Non-trainable parameters 0(0.00B) are parameters that will not updated or fixed during the training process.

The fine-tuned FFNN model's total number of parameters is slightly increased to 5763(22.50KB). Trainable parameters are 5761(22.50KB) is remain the same as the initial model; this means the original parameters are still being updated during training. Non-trainable parameters remain at zero, this reinforces that the entire model is trainable, which can be advantageous for learning complex patterns in the data. Optimizer parameters 2(12.00KB) refers to the parameters used by the optimizer itself; this maintains additional parameters using sgd optimizer to the training process. The number of optimizer parameters is minimum due to a simple optimizer configuration stochastic gradient descent optimizer with a fixed learning rate and no additional parameters.

To summarise the FFNN model architecture, the total parameters of the model start with 5761 parameters, which is relatively small which means a lightweight architecture; all parameters are trainable, which helps for learning from the data; there are no non-trainable parameters, which means the model is full adaptability. After fine-tuning, the total parameter to 5763, which means the model used simple optimizer sgd. The parameter size shows the model is efficient in terms of memory usage while being capable of learning from the data effectively.

## 4.3 Ethical consideration

The following ethics of the system are considered for developing predictive analytics for controlling tax evasion using deep learning: -

**Data security and privacy:** We collected data from three sources; all data contain confidential data; Tin, and personal information. To make data confidential, we replaced the equivalent value in Tin and dropped the other personal information like name, phone number, location and so on that was not necessary.

**Integrity:** We proposed to use deep learning, but deep learning algorithms are well known for their black-box nature. So, we used integrity risk analysis and the audit selection criteria of the post-clearance audit process from the customs commission for data pre-processing and feature selection.

**Transference and fairness of decision making:** Integrating risk analysis and audit selection criteria for data pre-processing and feature selection. We use the result from models.

**Algorithm Bias:** We considered the following to the reliability of using deep learning to make predictions: -

We did a proper data split; we analysed all training and test datasets for training and tested all algorithms.

We tested the model's well-known regression metrics MB, MSE, RMSE, and R-squared and explained the variance score to test the prediction error.

## 4.4 Tool used

### 4.4.1 Hardware Tools

There are requirements used for system development, running, and everything which software wants to integrate, such as a computer with the following specification:

Laptop Computer Processor: corei5, RAM: 8GB, HDD: 1TB and SSD: 256GB.

Flash drive: 32GB

Paper: 1000 sheets A4 size

Pen : 20 peace’s

### 4.4.2 Software tools

There are requirements used for implementing the code before installing for that the following software requirements are used to develop the proposed system such as Window 11 operating system, Google colab and keras library with the tensor flow as a back end

### 4.4.3 Programming Language

**Python:** we used python for creating a model: predictive analytics for controlling tax evasion using deep learning

# CHAPTER FIVE
# EXPERIMENTAL RESULT AND DISCUSSION

## Introduction

We discuss the result of different evaluation metrics mean bias, mean squared error, root mean squared error, and explained variance error and hyperparameter tuning. We also compare the model and suggest the best algorithm based on evaluation metrics for prediction.

## 5.1 The model using random research CV

### 5.1.1 Model architecture using random search CV

Table 5-1 Model architecture using random search CV

| **Model** | **Initial model** | **Fine-tuned model** |
|---|---|---|
| DNN | Total params 3329(13.00KB)<br>Trainable params 3329(13.00KB)<br>Non-trainable params 0(0.00B) | Total params 6660(26.02KB)<br>Trainable params 3329(13.00KB)<br>Non-trainable params 0(0.00B)<br>Optimizer params 3331(13.02) |
| AutoEncoder | Total params 7217(28.19KB)<br>Trainable params 7217(28.19KB)<br>Non-trainable params 0(0.00B) | Total params 21,653(84.59KB)<br>Trainable params 7217(28.19KB)<br>Non-trainable params 0(0.00B)<br>Optimizer params 14436(56.39KB) |
| FFNN | Total params 2817(11.00KB)<br>Trainable params 2817(11.00KB)<br>Non-trainable params 0(0.00B) | Total params 8453(33.02KB)<br>Trainable params 2817(11.00KB)<br>Non-trainable params 0(0.00B)<br>Optimizer params 5636(22.02KB) |

Total parameters comparison of models in above table (Table 5-1). The AutoEncoder model has highest total parameters in both initial and fine-tuned states, show its complexity. The AutoEncoder also shows increase, suggesting a more complex architecture. The DNN and FFNN have fewer parameters overall, with the FNN being the simplest among them. Trainable parameters comparison of models in above table (Table 5-1), all model trainable parameters remain unchanged, show that the models stable architectures. Non-trainable parameters comparison of models in above table (Table 5-1), for all models initially and after

fine tuning hove 0, this show that all parameters in the models are designed to be trainable, contributing directly to model learning. Optimizer parameters comparison of models in above table (Table 5-1), the AutoEncoder model has the highest optimizer parameters, reflecting a potentially more complex optimization process. The DNN follow to ensemble model. The FFNN have fewer optimizer parameters, suggesting simpler optimization strategies.

Overall comparison

1. Complexity: AutoEncoder>DNN>FFNN; the AutoEncoder model is the most complex, while the FNN is the simplest.
2. Parameter stability: All models maintain stable trainable parameter, due to there is no non trainable parameter initial model and after fine tuning.
3. Optimization approach: the AutoEncoder model shows that the most involved optimization process, followed by the DNN then the FNN.

Each model serves different purposes based on complexity, stability, and optimization strategies. The AutoEncoder models are best suited for complex tasks requiring high performance, while the FFNN be preferable for simpler tasks.

### 5.1.2 Training and Validations loss using random search CV

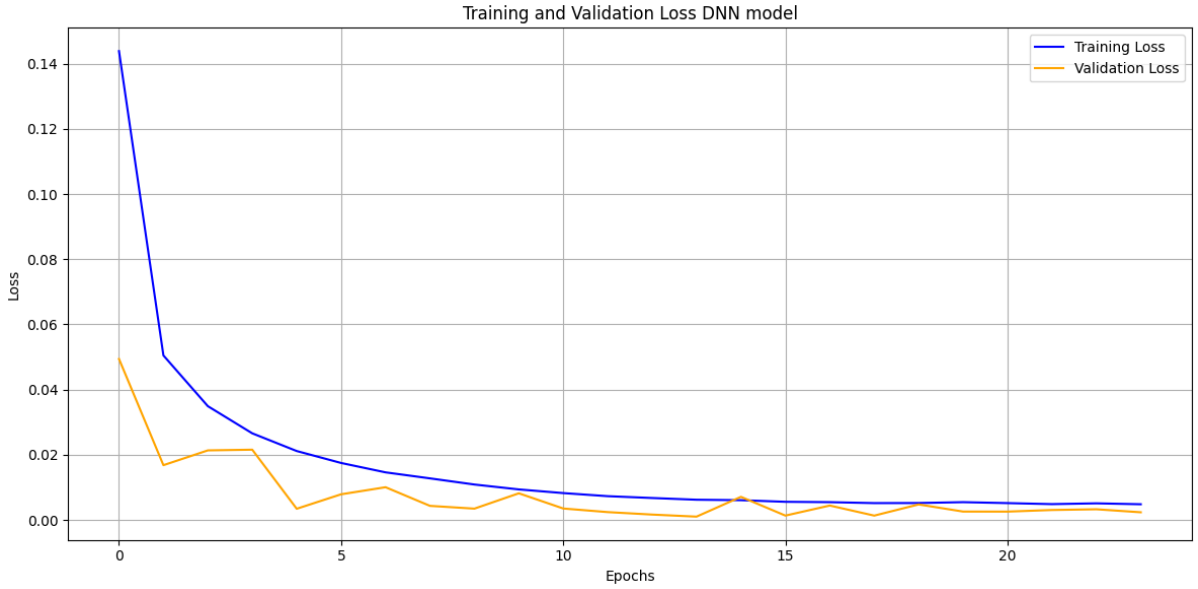


Figure 5-1 Training and validation loss using DNN model

The training and validation loss using DNN model in above figure (Figure 5-1), show that the both validation loss and training loss decreasing; the validation loss is less than the training loss this implies that the DNN model learning and generalizing well on unseen data.

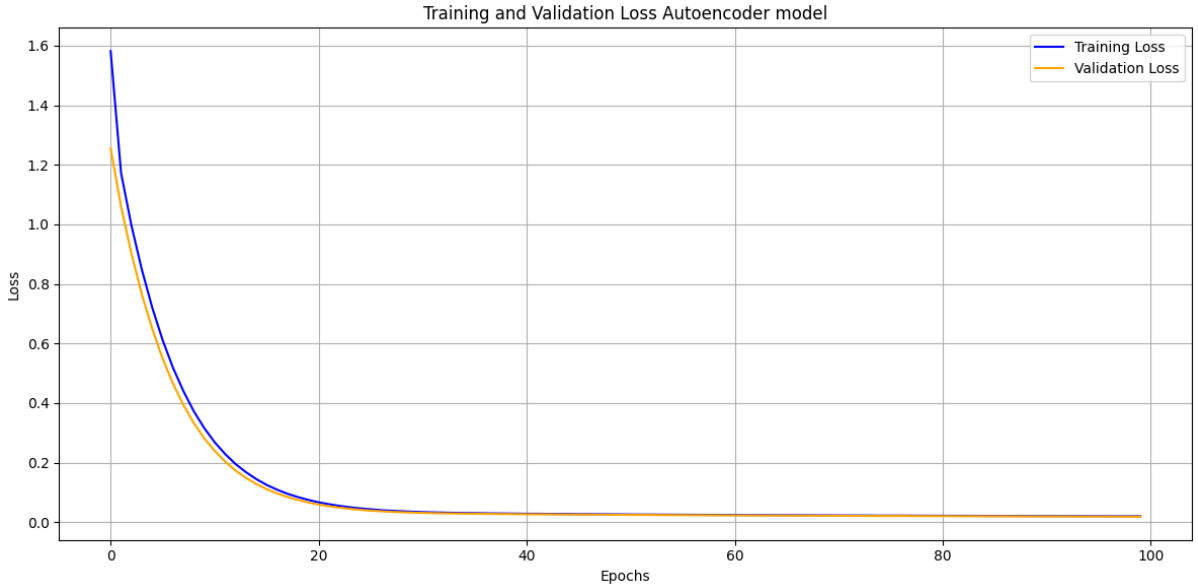


Figure 5-2 Training and validation loss for AutoEncoder model

The training and validation loss for AutoEncoder model in above figure (Figure 5-2), show that both losses is decreasing, and low, this show that the model effectively learning and generalization. The small difference in losses initial and the same through final suggest that the model is not overfitting.

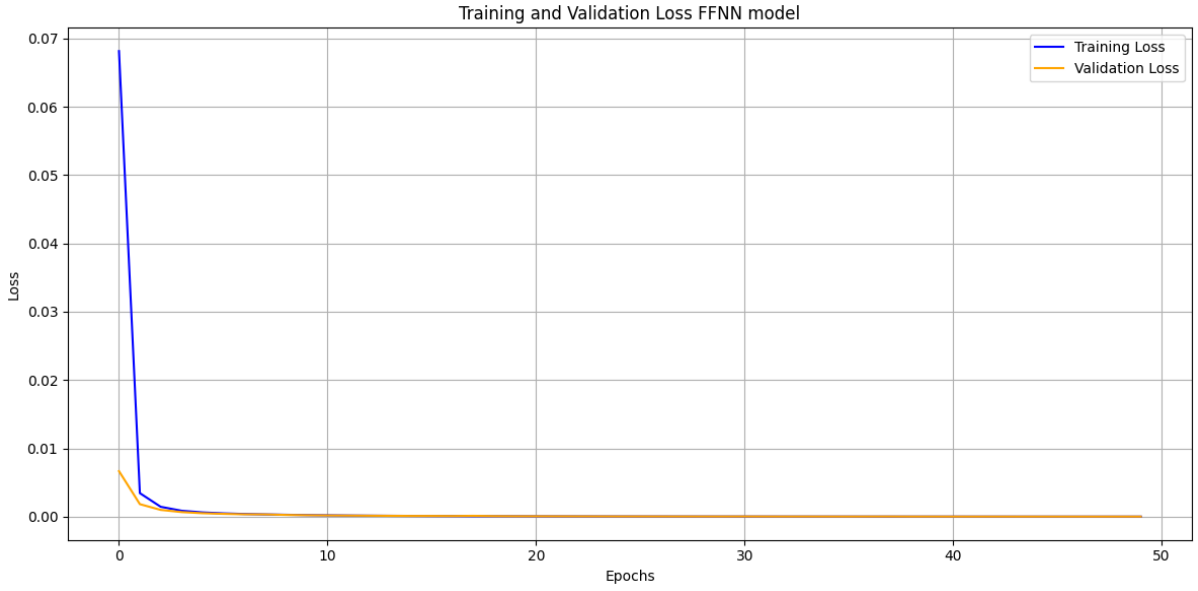


Figure 5-3 Training and validation loss for FFNN model

The training and validation loss for FFNN model in above figure (Figure 5-3), show there is no difference in training and validation loss, suggest that model is learn for both training and unseen data.

Table 5-2 Training and validation loss of models summary

| Model | Training loss | Validation loss |
|---|---|---|
| DNN | 0.0048 | 0.0023 |
| AutoEncoder | 0.0198 | 0.0186 |
| FFNN | 0.00 | 0.00 |

The training and validation loss of DNN models in above table (Table 5-2), shows a low training loss and an even lower validation, suggesting that it has learned the training data well while also generalizing effectively to unseen data. This balance indicates a well performing model.

The training and validation loss of AutoEncoder in above table (Table 5-2), shows slightly higher training and validation losses compared to DNN, but both values still low and close values. This shows that the models is performing well but might not be as effective as the DNN in generalizing.

The training and validation loss of FFNN model in above table (Table 5-2), shows the FFNN achieves a perfect fit on both training and validation datasets. While this show strong performance, it raises concerns about potential overfitting, as it suggests the model may have memorized the data rather than learned general patterns. So we needed to measure another performance metrics to check either the model have prediction error.

Overall summary of table (Table 5-2), the DNN shows the best balance between training and validation losses, indicating strong generalization. The AutoEncoder performs well but not as robustly as the DNN. In contrast, the FFNN, while achieving perfect losses, raise concerns about overfitting and the ability to generalize to new data. So that, the FFNN models needed further evaluation to checks its prediction error.

### 5.1.3 Model performance results

| | Model | Mean Bias | Mean Squared error | Root Mean Squared error | R-squared(%) | Explained Variance Score(%) |
|---|---|---|---|---|---|---|
| 0 | FFNN | -0.0009 | 0.0000 | 0.0038 | 99.9816 | 99.9828 |
| 1 | DNN | -0.0078 | 0.0023 | 0.0482 | 96.9793 | 97.0584 |
| 2 | Autoencoder | 0.0045 | 0.0007 | 0.0258 | 99.1330 | 99.1594 |

Figure 5-4 Model performance results using random search CV

### 5.1.4 Model performance visualization

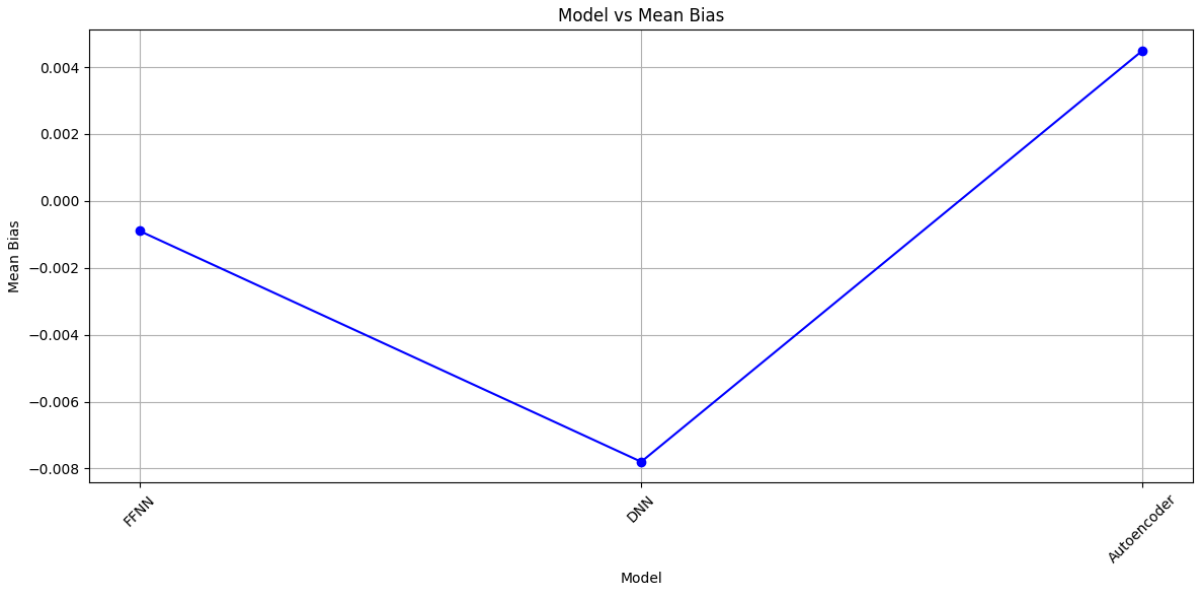


Figure 5-5 Model vs. Mean Bias using random search CV

The model performance visualization with mean bias in above figure (Figure 5-5), shows the FFNN and DNN mean bias values in negative directions, indicating slight tendency to underestimate the target values. The AutoEncoder exhibits mean bias values in positive directions, suggesting slight overestimation.

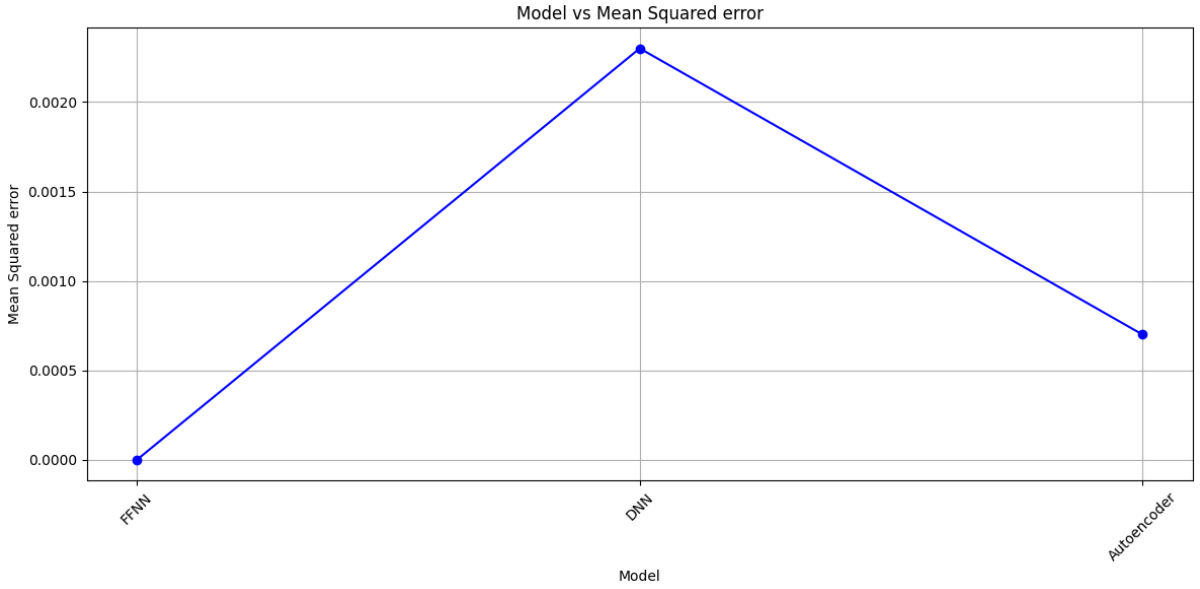


Figure 5-6 Model vs. Mean squared error using random search CV

The model performance visualization with mean squared error in above figure (Figure 5-6), shows the FFNN achieves the lowest mean squared error, and indicates the superior

performance. The AutoEncoder models also perform well with low mean squared error. The DNN has highest mean squared error, suggesting it has larger prediction errors compared to the other models.

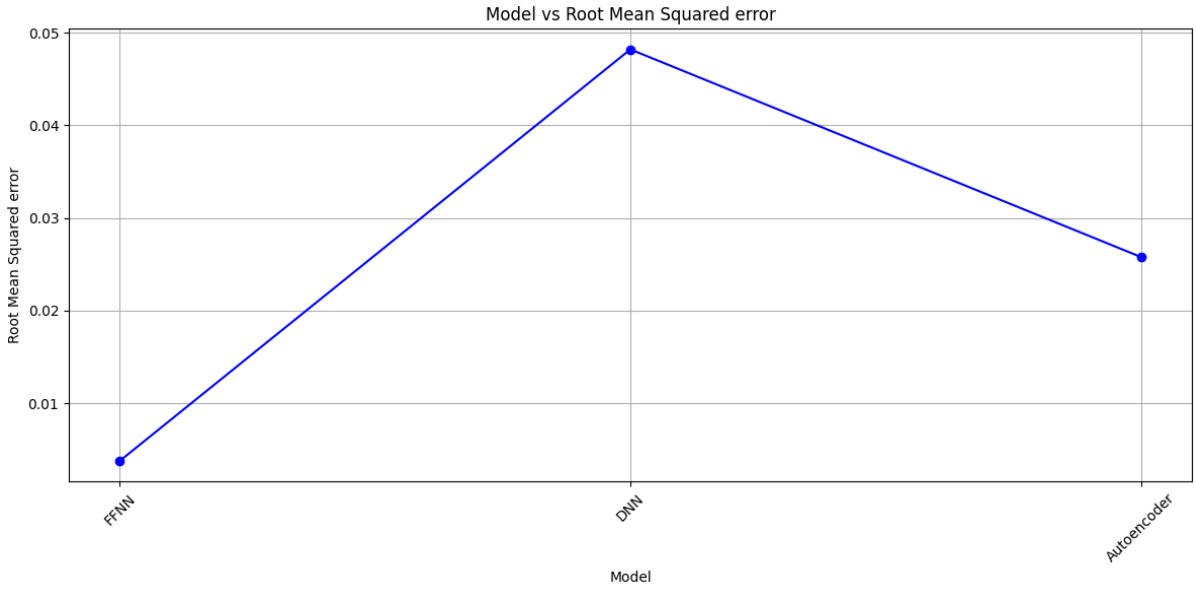


Figure 5-7 Model vs. Root Mean Squared error using random search CV

The model performance results visualization using root mean squared error in above figure (Figure 5-7), show consistent with mean square error, the FFNN has the lowest RMSE, indicating the best overall prediction accuracy. The AutoEncoder follow with moderate root mean squared error. The DNN has the highest root mean squared errors in its predictions, indicate that less prediction accuracy.

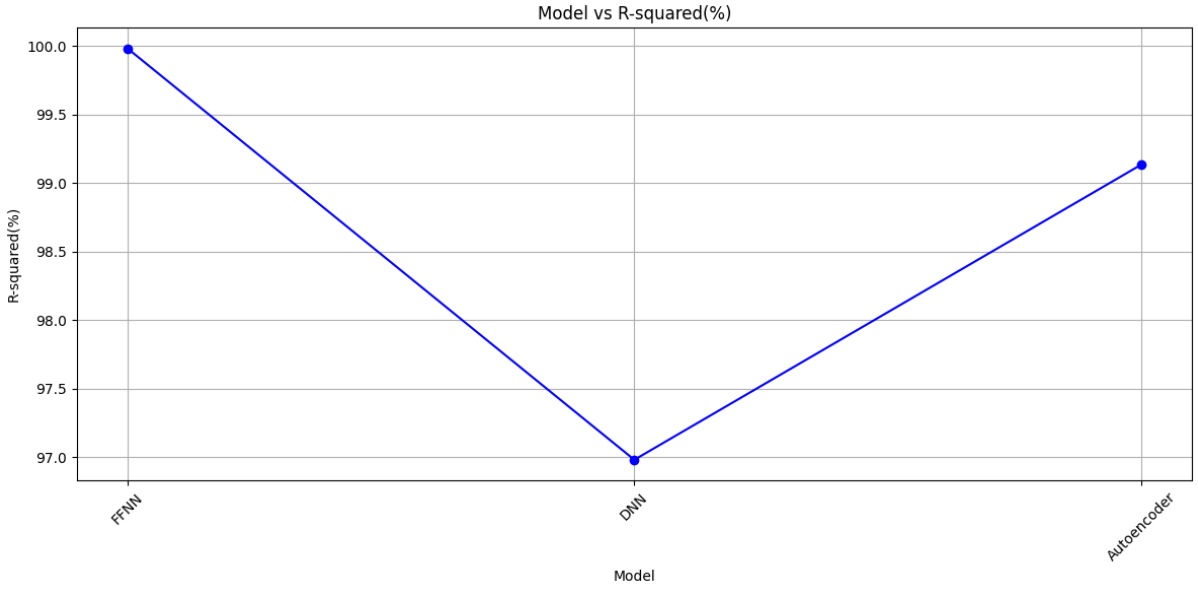


Figure 5-8 Model vs. R-squared using random search CV

The model performance results visualisation with r-squared value in above figure (Figure 5-8), shows the FFNN has the highest r-squared, that means it explains nearly all the variance in the target variable followed by AutoEncoder model. The DNN has a significantly lower r-squared value, suggesting it explains less variance compared to the other models.

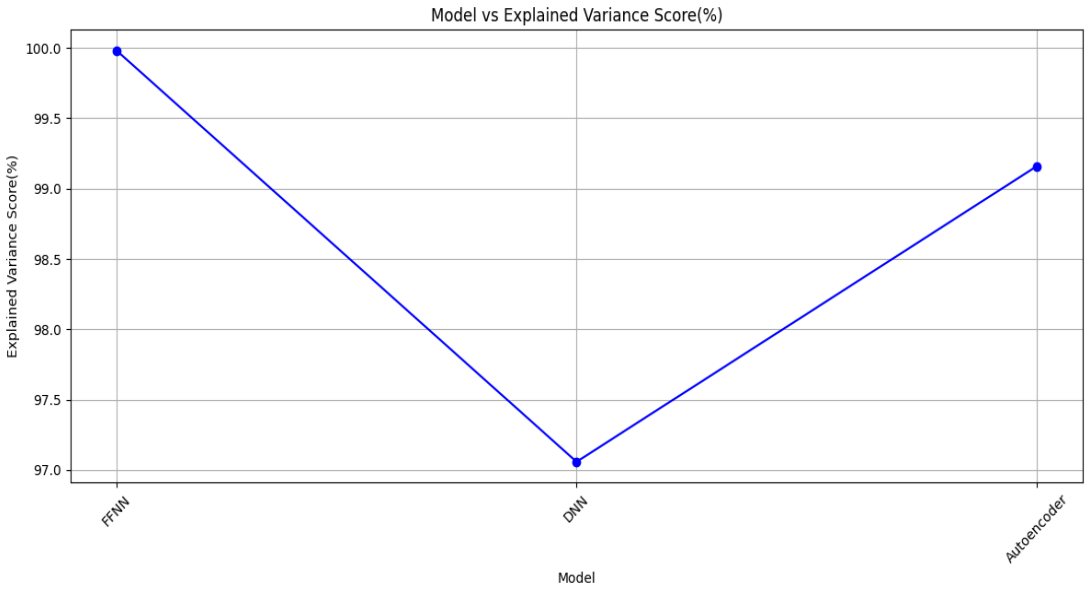


Figure 5-9 Model vs. explained variance score using random search CV

The model performance results visualization using explained variance score in above figure (Figure 5-9), show that the FFNN has the highest explained variance score, indicating it captures nearly all the variability in the data. The AutoEncoder also performs well, closely following the FFNN. The DNN has the lowest score, again reflecting its limitations in variability capture compared to the other models.

### 5.1.5 Model performance implication

**Feed-forward neural network (FFNN):-** the FFNN model exhibits very low error metrics (MSE and RMSE) in above table (Table 5-3), indicating high accuracy in predictions. The R-squared value is extremely close to 1, suggesting that the model explains almost all variability in the data. The negative mean bias indicates a slight tendency to underestimate the target variable, but it is negligible given the overall performance.

**Deep neural network (DNN):-** The DNN shows a higher MSE and RMSE in above table (Table 5-3) compared to the FNN, indicating less accuracy in its predictions. The mean bias is negative, suggesting a tendency to underestimate the target variable. Although the R-squared and explained variance scores are still reasonably good, they indicate that the DNN does not capture as much variability in the data as the FFNN.

**AutoEncoder:** - the AutoEncoder performs well, with low MSE and RMSE values. Its Rsquared and Explained variance score express that it catches a critical amount of irregula rity in the data. The mean bias is nearly zero, suggesting that the model does not steadily over state or undervalue the objective variable in above table (Table 5-3).

Overall summary, FFNN consistently outperforms other model across all metrics that make it the best model for these problems. The AutoEncoder shows strong performance next to FFNN. The DNN lags behind in most metrics, suggesting it may not be the best fit for this dataset or tasks.

We choice the FFNN as best model for prediction due to perfect fit model , simplicity , stability, adaptability and outperforming other models across all metrics,

### 5.1.6 Hyperparameters tuning results using Random search CV

The randomized search CV is technique used for hyperparameter tuning in machine learning models. It is useful with large number of hyperparameter. It works by defining the parameter distributions, this done by allowing the algorithm to sample values; specify the number of iterations, this done by adjusting how many time the algorithm will sample hyperparameter values and evaluate the model; perform the search , this done randomly samples hyperparameter values from the specified distributions; update the search, this done after each iteration the algorithm updates its knowledge about the hyperparameter based on the performance of sampled hyperparameters; and return the best hyperparameters , this done at the end of search process by specifying the best scoring hyperparameters

```
param_dist = {
    'unit3': [32, 64, 128],
    'unit2': [16, 32, 64],
    'unit1': [8, 16, 32],
    'dropout_rate': [0.2, 0.3, 0.4],
    'batch_size': [32, 64, 128],
    'epochs': [50, 100, 150],
    'optimizer': ['adam', 'sgd', 'rmsprop'],
    'activation': ['relu', 'sigmoid', 'tanh']
}

# Create an instance of the DNNRegressor class
regressor = DNNRegressor()

# Create a RandomizedSearchCV object with the model and parameter distributions
random_search = RandomizedSearchCV(estimator=regressor, param_distributions=param_dist, scoring='neg_mean_squared_error', cv=3, n_iter=10, random_state=4

# Fit the RandomizedSearchCV object to your data
random_search.fit(X_train, y_train)

# Print the best hyperparameters
print("Best Hyperparameters: ", random_search.best_params_)
print("Best MSE: ", -random_search.best_score_)
```

The following models parameter optimization using random search CV:-

Table 5-3 DNN Model using random search CV hyperparmeter tuning

| Hyperparameters | Hyperparmeter tuning | Random search CV results |
|---|---|---|
| Activation fuction | ReLu,sigmoid, tanh | ReLu |
| Units | Layer3: 32,64, 128<br>Layer2: 16,32, 64<br>Layer1: 8, 16, 32 | Layer3:64,<br>Layer2:32, layer1:16 |
| Epochs | 50, 100, 150 | 100 |
| Batch size | 32, 64, 128 | 128 |
| optimizer | adam, sgd, rmsprop | rmsprop |
| Dropout | 0.2, 0.3, 0.4 | 0.2 |

Table 5-4 AutoEncoder Model using random search CV hyperparmeter tuning

| **Hyperparameters** | **Hyperparmeter tuning** | **Random search CV results** |
|---|---|---|
| Activation fuction | ReLu,tanh,sigmoid | ReLu |
| Dropout | 0.1, 0.2, 0.3 | 0.1 |
| Learning rate | '0.001,0.0001,0.00001 | 0.0001 |

| | | |
|---|---|---|
| Units | Layer4:32, 64, 128,<br>Layer3: 16, 32, 64,<br>Layer2: 8, 16, 32,<br>Layer1: 4, 8, 16, | Laye4:64, Layer3:64, Layer2:32,<br>Layer: 8 |
| Epochs | 50, 100, 200 | 100 |
| Batch size | 32, 64, 128 | 64 |
| optimizer | Adam,sgd,RMSprop | Adam |

Table 5-5 FFNN model using random search CV hyperparmeter tuning

| **Hyperparameters** | **Hyperparmeter tuning** | **Random search CV results** |
|---|---|---|
| Activation fuction | ReLu,tanh, sigmoid | ReLu |
| Learning rate | 0.001,0.0001,0.00001 | 0.0001 |
| Units | Laye2: 16, 32, 64,<br>Laye1: 8, 16, 32 | Laye2: 64,<br>Layer1: 32 |
| Epochs | 50, 100, 200 | 50 |
| Batch size | 32, 64, 128 | 32 |
| optimizer | Adam,sgd, rmsprop | Adam |

### 5.1.7 The prediction of audit selection using Feed forward neural (FFNN)

The following audit selection prediction and the top three indicators tax evasion using Feed forward neural for five consecutive years: -

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **14606** | 0072306850 | 2 | 2 | 2 | 2 | 2 | 0 | 2 | 2 | 2 | 2 | 1.912279 |
| **4487** | 0005286323 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1.835797 |
| **9422** | 0036307353 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1.835797 |
| **7480** | 0025873332 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1.835797 |
| **778** | 0000055226 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1.835797 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **658** | 0000037252 | 0 | 2 | 2 | 2 | 2 | 0 | 2 | 2 | 0 | 2 | 1.115849 |
| **266** | 0000023520 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 1.115811 |
| **10126** | 0038673820 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 1.115811 |
| **11000** | 0053353235 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 1.115811 |
| **10555** | 0052303523 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 1.115811 |



2411 rows × 12 columns

Figure 5-10 Audit selection prediction using feed forward neural for 2024 year

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4494** | 0005288057 | 0 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 0 | 0 | 1.115619 |
| **11613** | 0055557885 | 1 | 1 | 1 | 2 | 2 | 0 | 1 | 2 | 0 | 0 | 1.115032 |
| **1069** | 0000232223 | 0 | 1 | 1 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 1.115032 |
| **14408** | 0068223006 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 1.115032 |
| **2193** | 0002332803 | 0 | 2 | 1 | 2 | 2 | 1 | 0 | 2 | 0 | 0 | 1.115032 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **972** | 0000078632 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 0 | 2 | 0.920292 |
| **204** | 0000022265 | 2 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 0.920292 |
| **10853** | 0053025303 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 0 | 0 | 0.920292 |
| **3062** | 0003033538 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 0 | 0 | 0.920292 |
| **846** | 0000060808 | 2 | 1 | 1 | 1 | 1 | 0 | 2 | 2 | 1 | 0 | 0.920292 |

2411 rows × 12 columns

Figure 5-11 Audit selection prediction using feed forward neural for 2025 year

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2693** | 0002653883 | 0 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0.920098 |
| **5995** | 0022383362 | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 2 | 1 | 0 | 0.920098 |
| **14412** | 0068232822 | 2 | 1 | 1 | 2 | 2 | 1 | 0 | 2 | 0 | 0 | 0.920098 |
| **14772** | 0073336332 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 2 | 0 | 0 | 0.920098 |
| **4878** | 0005707337 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0.920098 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3009** | 0003003825 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 1 | 2 | 0.797873 |
| **4013** | 0003723380 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 0 | 0 | 0.797873 |
| **12547** | 0058333555 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | 2 | 0 | 0.797873 |
| **1288** | 0000335207 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 0 | 0.797873 |
| **11911** | 0056526066 | 0 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 0.797873 |

2411 rows × 12 columns

Figure 5-12 Audit selection prediction using feed forward neural for 2026 year

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **9948** | 0038077357 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 0 | 0.797873 |
| **8723** | 0032623372 | 1 | 1 | 1 | 2 | 2 | 0 | 0 | 2 | 1 | 0 | 0.797722 |
| **2068** | 0002280753 | 1 | 1 | 1 | 0 | 0 | 1 | 2 | 2 | 1 | 0 | 0.796947 |
| **9866** | 0037635500 | 0 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 0 | 0.796344 |
| **12145** | 0057285726 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 | 0.796344 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2374** | 0002507257 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | 2 | 0 | 0.638453 |
| **2515** | 0002553850 | 0 | 1 | 1 | 2 | 2 | 0 | 1 | 2 | 0 | 0 | 0.638453 |
| **11600** | 0055553372 | 1 | 2 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 2 | 0.638453 |
| **3868** | 0003577726 | 1 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0.638453 |
| **14207** | 0066277523 | 1 | 1 | 1 | 2 | 2 | 0 | 0 | 2 | 1 | 0 | 0.638453 |

2411 rows × 12 columns

Figure 5-13 Audit selection prediction using feed forward neural for 2027 year

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **6701** | 0023523258 | 0 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 2 | 0 | 0.637828 |
| **13** | 0000002352 | 1 | 1 | 2 | 0 | 0 | 2 | 2 | 2 | 1 | 0 | 0.626302 |
| **9386** | 0036256355 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 1 | 0 | 0.625632 |
| **10471** | 0052057322 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0.624097 |
| **13864** | 0063605227 | 0 | 1 | 1 | 2 | 2 | 1 | 0 | 2 | 1 | 0 | 0.622684 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **841** | 0000060255 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 | 0.546189 |
| **13225** | 0062322570 | 0 | 1 | 1 | 0 | 0 | 1 | 2 | 2 | 0 | 0 | 0.546188 |
| **4926** | 0005758283 | 0 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 0 | 0.546184 |
| **1712** | 0000883325 | 0 | 1 | 1 | 2 | 2 | 0 | 1 | 2 | 1 | 0 | 0.546176 |
| **6082** | 0022553263 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0.546161 |

2411 rows × 12 columns

Figure 5-14 Audit selection prediction using feed forward neural for 2028 year

The top three indictor's tax evasion prediction for five consecutive years using feed-forward neural networks is the same results for each year are customer profile, declarant profile, and random selection.

## 5.2 The model using Bayesian optimizer

### 5.2.1 Model architecture using Bayesian optimizer

Table 5-6 Model architecture using Bayesian optimizer

| **Model** | **Initial model** | **Fine-tuned model** |
|---|---|---|
| DNN | Total params 2675(10.45KB)<br>Trainable params 2675(10.45KB)<br>Non-trainable params 0(0.00B) | Total params 8027(31.36KB)<br>Trainable params 2675(10.45KB)<br>Non-trainable params 0(0.00B)<br>Optimizer params 5352(20.91KB) |
| AutoEncoder | Total params 5791(22.62KB)<br>Trainable params 5791(22.62KB)<br>Non-trainable params 0(0.00B) | Total params 17,375(67.88KB)<br>Trainable params 5791(22.62KB)<br>Non-trainable params 0.00(0.00B) |

| | | |
|---|---|---|
| | | Optimizer params 11,584(42.25KB) |
| FFNN | Total params 5761(22.50KB)<br>Trainable params 5761(22.50KB)<br>Non-trainable params 0(0.00B) | Total params 5763(22.52KB)<br>Trainable params 5761(22.50KB)<br>Non-trainable params 0(0.00B)<br>Optimizer params 2(12.00B) |

The total parameters of model in above table (Table 5-6) , show that the DNN and AutoEncoder model increases in total parameters after fine tuning, showcasing their enhanced complexity and capability to learn from data. In contrast, the FFNN showed minimal change in total parameters, showing that the fine-tuning process did not alter its architecture. The trainable parameters of model in above table (Table 5-6), show that all models trainable parameters remained unchanged; this show that the learning capacity of each model was preserved and the fine tuning is focused on optimizing the existing parameter rather than altering the model architecture. The non –trainable parameters in either the initial or fine-tuned model is the zero. This suggests that all parameters were designed to be trainable, allowing for full adaptability to the training data. The optimizer parameters in above table (Table 5-6), shows that the DNN and AutoEncoder model, have more complex optimization process. The AutoEncoder has the highest optimizer parameters that contribute to improved performance. The FFNN has low number of optimizer parameters, which shows that it has simpler optimization approach. This may limit its ability to adapt to complex data patterns compared to the other two models.

Overall comparisons:

1. Complexity: AutoEncoder> DNN> FFNN; the AutoEncoder model is the most complex, while the FNN is the simplest.
2. Parameter stability: All models maintain stable trainable parameter, there is no non trainable parameter initial model and after fine tuning.
3. Optimization approach: the AutoEncoder show that the most involved optimization process, followed by the DNN, then the FFNN.

Each model serves different purposes based on complexity, stability, and optimization strategies. The AutoEncoder model is best suited for complex tasks requiring high performance, while the FFNN be preferable for simpler tasks.

### 5.2.2 Training and validation loss using Bayesian optimizer

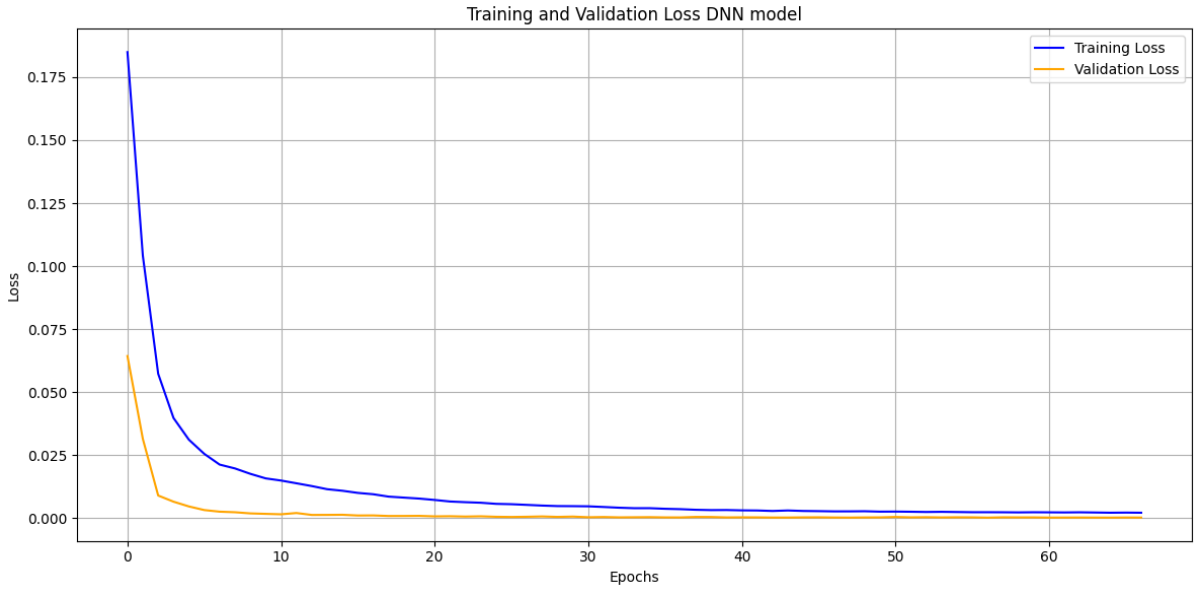


Figure 5-15 Training and validation loss DNN model using Bayesian optimizer

The training and validation loss DNN model using Bayesian optimizer in above figure (Figure 5-15), shows that the validation loss is lower than training loss, which means the DNN has learned effectively from the training data by maintaining strong generalization capabilities. And the model has risks overfitting, so we needed to evaluate different predictions error metrics.

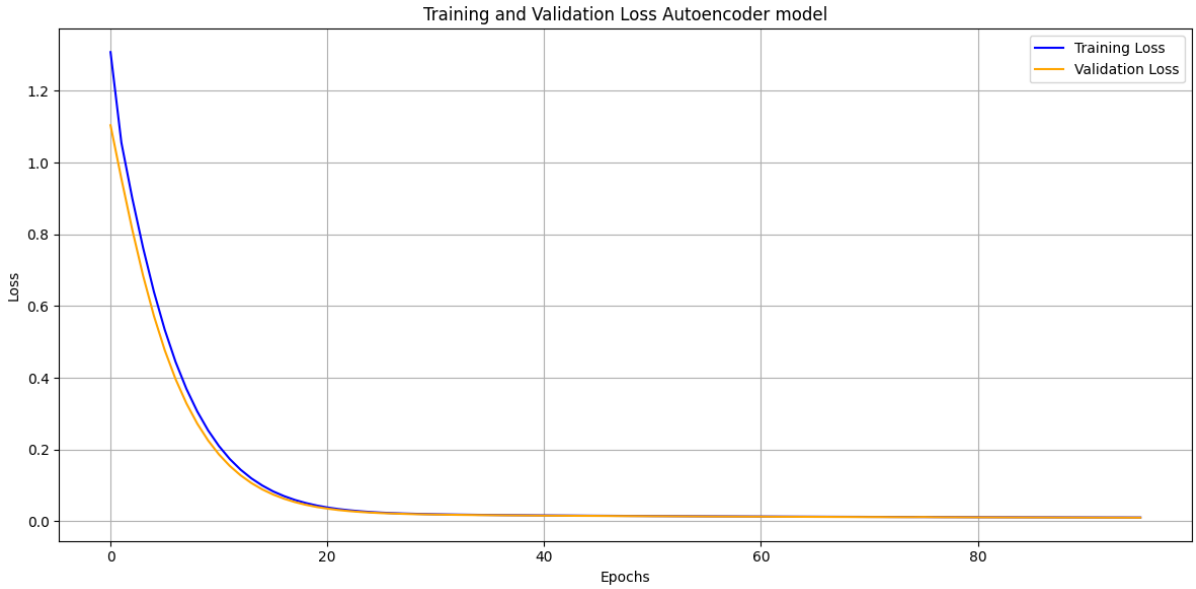


Figure 5-16 Training and validation loss AutoEncoder model using Bayesian optimizer

The training and validation loss AutoEncoder model using Bayesian optimizer in above figure (Figure 5-16), shows that both the losses are low, slightly training loss higher than

validations, this indicates the AutoEncoder has learned effectively from the training data while maintaining strong generalization capabilities..

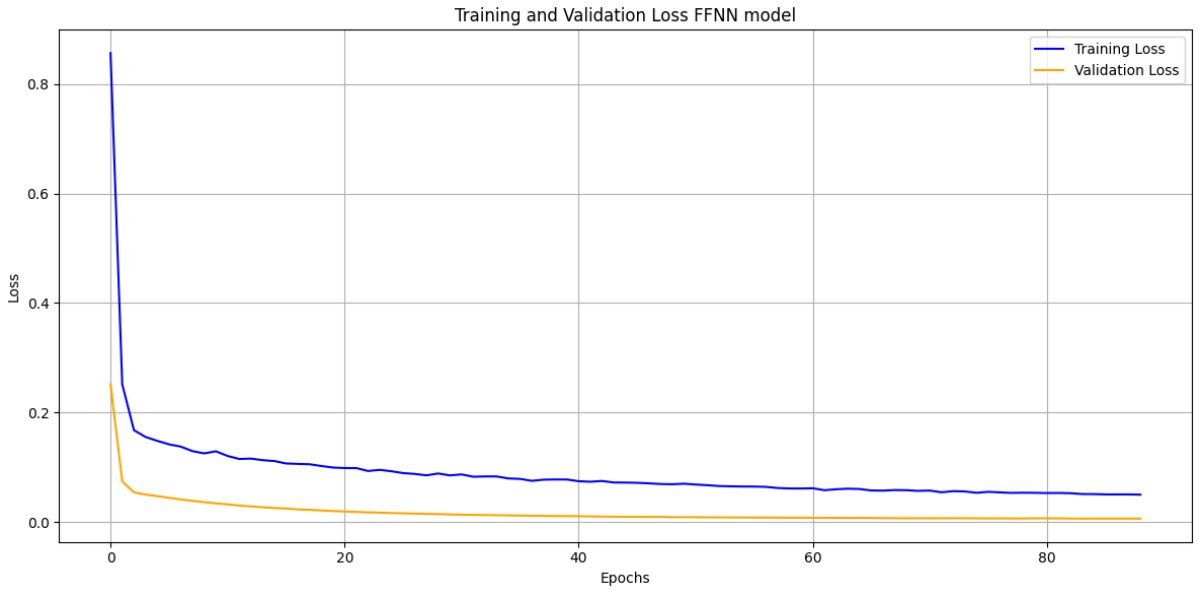


Figure 5-17 Training and validation loss FFNN model using Bayesian optimizer

The training and validation loss FFNN model using Bayesian optimizer in above figure (Figure 5-17), shows that, the validation loss less than training loss, this means the model is accurate predictions on new data. FFNN has learned effectively from the training data while maintaining strong generalization capabilities.

Table 5-7 Training and validation loss of models using Bayesian optimizer

| **Model** | **Training loss** | **Validation loss** |
|---|---|---|
| DNN | 0.0022 | 0.0003 |
| AutoEncoder | 0.0107 | 0.0103 |
| FFNN | 0.0500 | 0.0063 |

The training and validation loss of models using Bayesian optimizer in above table (Table 5-13), shows that the DNN models training data achieving a very low error rate, the validation loss is also significantly lower than the training loss, suggesting that the model excellent generalization to unseen data; the AutoEncoder training loss is higher than that of the DNN, show that while the AutoEncoder has learned from training data, it is not as effective as the DNN in minimizing reconstruction error, the validation loss is very close to the training loss, suggesting that the AutoEncoder generalizes well; the FFNN has the highest training loss among the three models, show that it has not learned as effectively from the training data compared to the DNN and AutoEncoder, the validation loss is significantly lower than the

training loss, suggesting that the FFNN may generalize well to unseen data however , the large gap between training and validation loss raises concerns about potential underfitting.

Overall summary of training and validation loss the DNN exhibits the best overall performance, followed by the AutoEncoder, then FFNN.

### 5.2.3 Model performance results using Bayesian optimizer

| | Model | Mean Bias | Mean Squared error | Root Mean Squared error | R-squared(%) | Explained Variance Score(%) |
|---|---|---|---|---|---|---|
| 0 | FFNN | -0.0524 | 0.0062 | 0.0788 | 91.9207 | 95.4894 |
| 1 | DNN | 0.0015 | 0.0002 | 0.0154 | 99.6901 | 99.6928 |
| 2 | Autoencoder | -0.0014 | 0.0003 | 0.0172 | 99.6159 | 99.6185 |

Figure 5-18 Model performance results using Bayesian Optimizer

### 5.2.4Model performance visualization using Bayesian optimizer

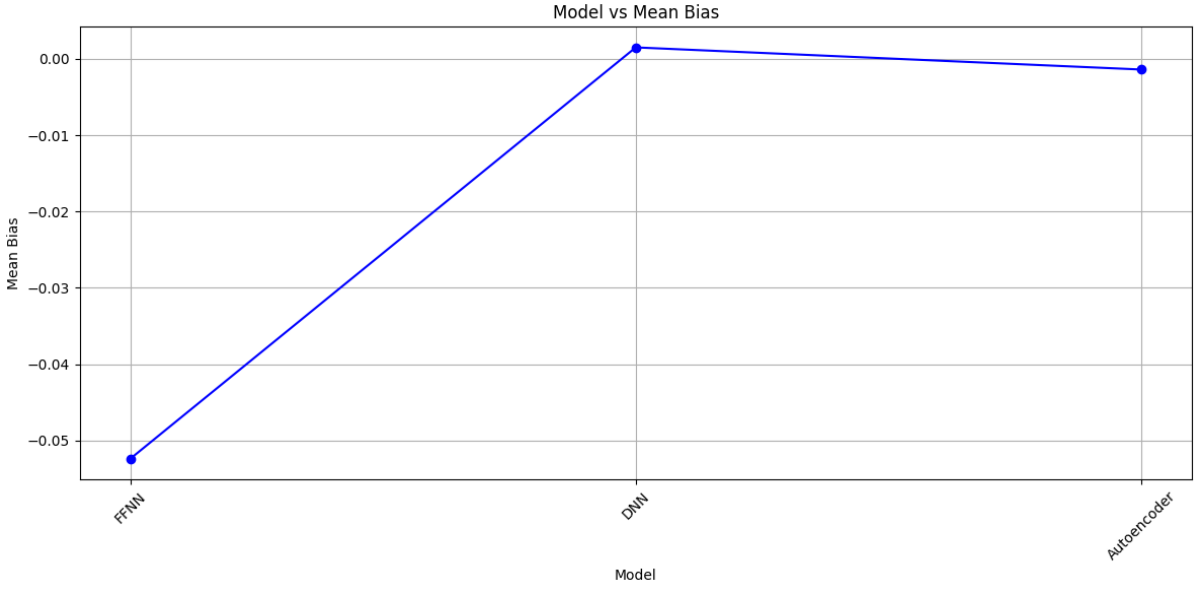


Figure 5-19 Model vs. Mean Bias using Bayesian optimizer

The model vs. mean bias using Bayesian Optimizer in above figure (Figure 5-19), show that the DNN model has the lowest mean bias, least tendency to overestimate the target variable. The AutoEncoder have slightly higher mean bias values than DNN, suggesting minimal bias in their predictions. The FFNN has the highest mean bias, with a negative value, implying that it tends to underestimate the target variable on average.

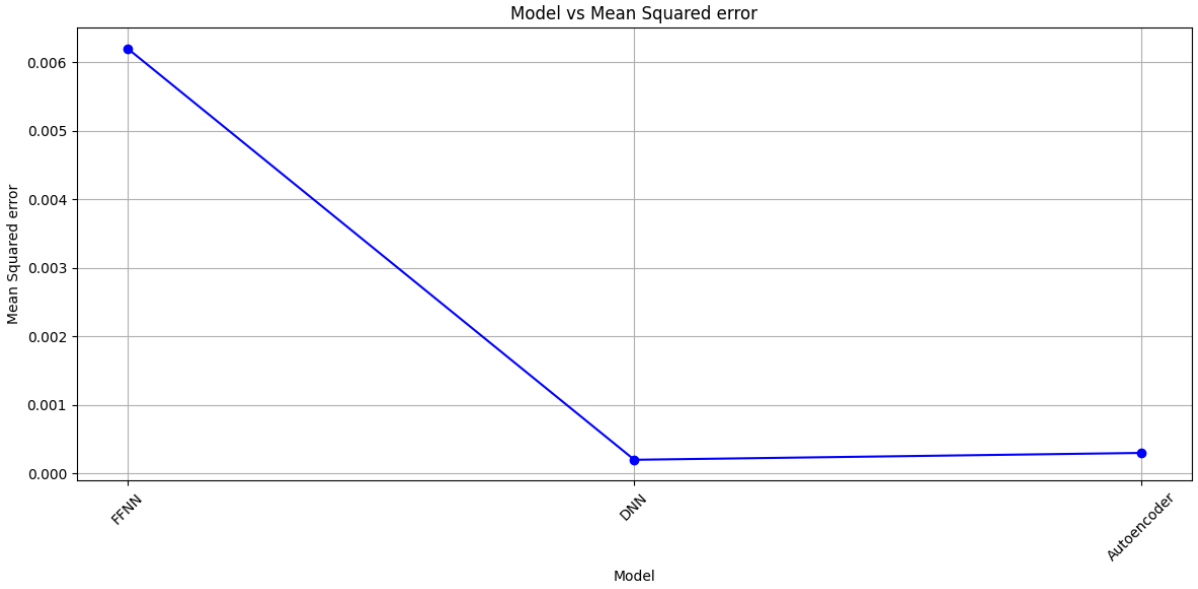


Figure 5-20 Model vs. Mean Squared error using Bayesian optimizer

The model vs. Mean Squared error using Bayesian optimizer in above figure (Figure 5-20); shows that the DNN model has the lowest MSE, that lowest prediction error; the AutoEncoder have slightly higher MSE but are still very low, shows high accuracy in their predictions; and the FFNN has the highest MSE among the models, show not good predictions as other models.

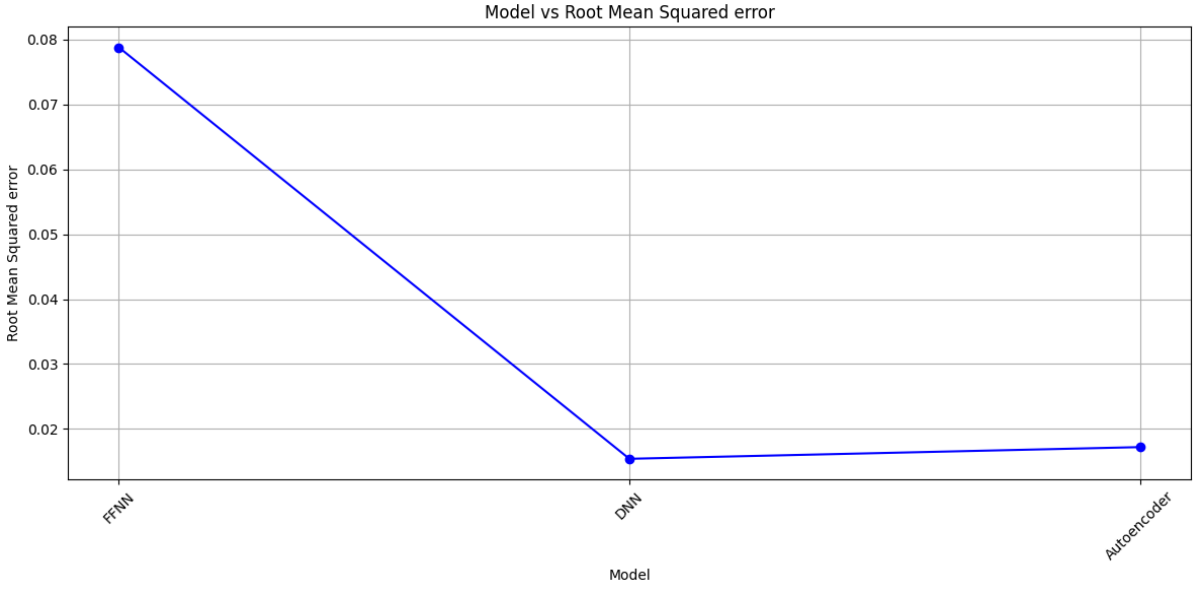


Figure 5-21 Model vs. Root Mean Squared error using Bayesian optimizer

The Model vs. Root Mean Squared error using Bayesian optimizer in above figure (Figure 5-21); show that the DNN model has the lowest RMSE, which means accurate prediction. The AutoEncoder have slightly higher RMSE values than DNN, suggesting high accuracy in their

predictions then to DNN. The FFNN has the highest RMSE among the models; it has not good prediction as other models.

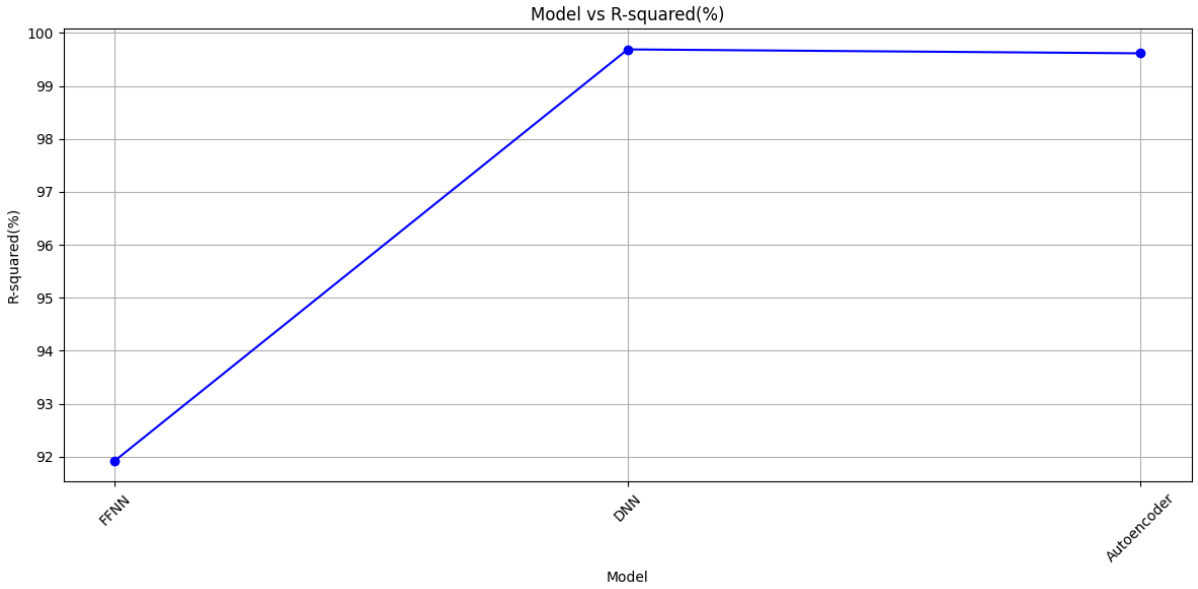


Figure 5-22 Model vs. R-squared using Bayesian optimizer

The Model vs. R-squared using Bayesian optimizer in figure (Figure 5-22); show that the DNN model has the highest R-squared value, show that it explains the highest percentage of the variance in the target variable. The AutoEncoder has slightly lower R-squared values but are still very high than DNN, suggesting that they explain a significant of the variance in the target variable. The FFNN has the lowest R-squared among the models, lowest percentages of the variance in the target variable.

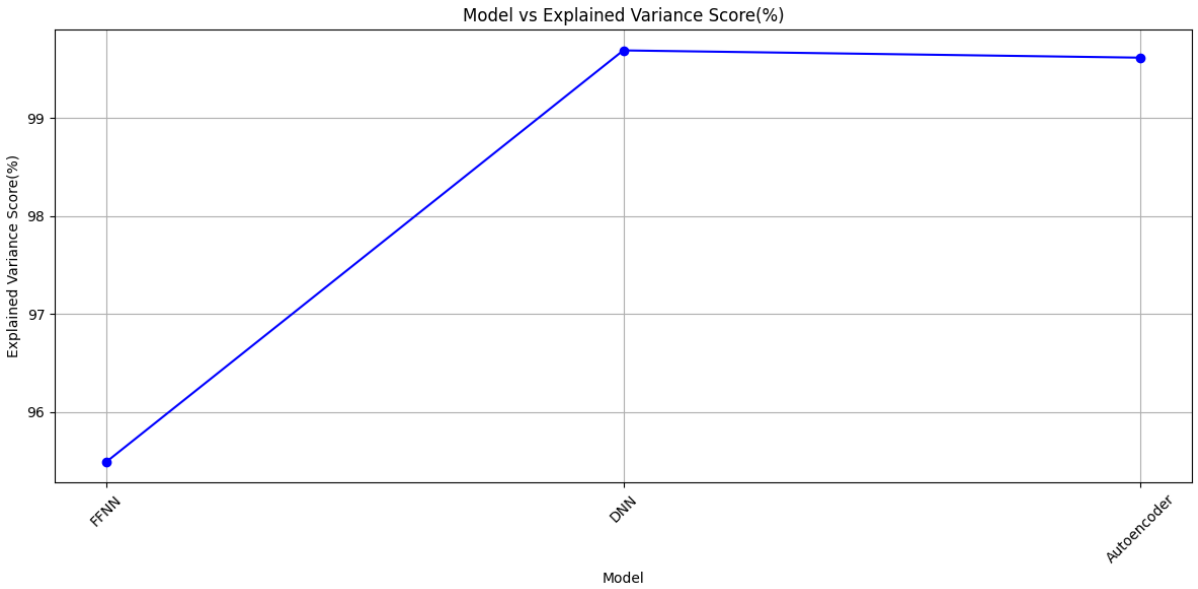


Figure 5-23 Model vs. Explained variance score using Bayesian optimizer

The Model vs. Explained variance score using Bayesian optimizer in the above figure (Figure 5-23); shows that the DNN model has the highest explained variance score that means it captures the highest percentage of the total variance in the target variance score. The AutoEncoder has slightly lower explained variance scores but are still very high than DNN, that means they capture a significant portion of the total variance in the target variable. The FFNN has the lowest explained variance score among the models, lowest percentage of the total variance in the target variance score than other models.

### 5.2.5 Model performance implication using Bayesian optimizer

**Feed-forward neural network (FFNN):-** the FFNN model exhibits higher error metrics (MSE and RMSE) in above table (Figure 5-18), indicating low accuracy in predictions. The R-squared value is close to 1, suggesting that the model explains variability in the data. The negative mean bias indicates a slight tendency to underestimate the target variable..

**Deep neural network (DNN):-** The DNN shows a lower MSE and RMSE in above table (Figure 5-18) compared to the FNN, indicating higher accuracy in its predictions. The mean bias is positive, suggesting a tendency to overestimate the target variable. Although the R-squared and explained variance scores are higher, they indicate that the DNN capture much variability in the data as than FFNN.

**AutoEncoder: -** the AutoEncoder performs well, with low MSE and RMSE values than both FFNN. Its Rsquared and Explained variance score express that it catches the data variability. The mean bias is negative, suggesting that the model does undervalue the objecti ve variable in above table (Figure 5-18).

We choice the DNN because outperform all prediction metrics and best to interpretability of the models.

### 5.2.6 Hyperparameter tuning using Bayesian optimizer

Bayesian optimization is useful in hyperparameter tuning. It works by surrogate model , this done by building objective function; Acquisition function, this done balances trying new hyperparameter combinations and refining known good hyperparameters; Iterative process, it process consists of sample hyperparameters, update surrogate model, optimize acquisition function and evaluate the objective function, repeat; and final convergence steps by specifying number of trials is completed.

```
def objective(trial):
    # Suggest hyperparameters
    units1 = trial.suggest_int('units1', 64, 256, step=32)
    units2 = trial.suggest_int('units2', 16, 128, step=16)
    dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5)
    learning_rate = trial.suggest_float('learning_rate', 1e-5, 1e-2, log=True)
    epochs = trial.suggest_int('epochs', 50, 200)
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64, 128])
    activation = trial.suggest_categorical('activation', ['relu', 'sigmoid', 'tanh'])
    optimizer = trial.suggest_categorical('optimizer', ['adam', 'sgd', 'rmsprop'])
```

```
    ffnn_model.compile(loss='mean_squared_error', optimizer=opt)

    # Early stopping callback
    early_stopping = EarlyStopping(monitor='val_loss', patience=5)

    # Fit the model
    ffnn_model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
                   callbacks=[early_stopping], validation_data=(X_val, y_val), verbose=0)

    # Evaluate the model on the validation set
    y_pred = ffnn_model.predict(X_val)
    mse = mean_squared_error(y_val, y_pred)

    return mse

# Create a study and optimize
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=50)

# Print the best hyperparameters and the best MSE
print("Best Hyperparameters: ", study.best_params)
print("Best MSE: ", study.best_value)
```

Table 5-8 DNN model hyperparmeter tuning using Bayesian optimizer

| Hyperparameters | Hyperparmeter tuning | Bayesian optimizer results |
|---|---|---|
| Activation fuction | trial.suggest_categorical('activation', ['ReLu', 'tanh', 'sigmoid']) | Sigmoid |
| Units | unit4 = trial.suggest_int('unit4',32, 128), unit3 = trial.suggest_int('unit3',16, 64), and unit2 = trial.suggest_int('unit2', 8, 32) | Unit4=71, unit3=19,unit2=25 |
| Epochs | trial.suggest_int('epochs', 50, 200) | 150 |
| Batch size | trial.suggest_categorical('batch_size', [16, 32, 64]) | 64 |
| optimizer | trial.suggest_categorical('optimizer', ['adam', 'sgd', 'rmsprop'] | Adam |
| Dropout | trial.suggest_float('dropout_rate', 0.1, 0.5) | 0.12483 |

Table 5-9 AutoEncoder model hyperparmeter tuning using Bayesian optimizer

| Hyperparameters | Hyperparmeter tuning | Bayesian optimizer results |
|---|---|---|
| Activation fuction | trial.suggest_categorical('activation', ['ReLu', 'tanh', 'sigmoid']) | tanh |
| Dropout | trial.suggest_float('dropout_rate', 0.1, 0.5) | 0.12146556 |
| Learning rate | trial.suggest_loguniform('lr', 1e-5, 1e-1) | 0.00022027 |

| | | |
|---|---|---|
| Units | units_1 = trial.suggest_int('units_1', 32, 128)<br>units_2 = trial.suggest_int('units_2', 16, 64)<br>units_3 = trial.suggest_int('units_3', 8, 32)<br>units_4 = trial.suggest_int('units_4', 4, 16) | Units_1=88,<br>Units_2=40,<br>Units_3=22,<br>Units_4=15 |
| Epochs | trial.suggest_int('epochs', 50, 200) | 96 |
| Batch size | trial.suggest_categorical('batch_size', [16, 32, 64]) | 64 |
| optimizer | trial.suggest_categorical('optimizer', ['Adam', 'sgd', 'RMSprop']) | Adam |

Table 5-10 FFNN model hyperparmeter tuning using Bayesian optimizer

| Hyperparameters | Hyperparmeter tuning | Bayesian optimizer results |
|---|---|---|
| Activation fuction | trial.suggest_categorical('activation', ['ReLu', 'sigmoid', 'tanh']) | tanh |
| Learning rate | trial.suggest_float('learning_rate', 1e-5, 1e-2, log=True) | 0.00003511 |
| Units | units1 = trial.suggest_int('units1', 64, 256, step=32), units2 = trial.suggest_int('units2', 16, 128, step=16) | Unit1=96, unit2=48 |
| Epochs | trial.suggest_int('epochs', 50, 200) | 150 |
| Batch size | trial.suggest_categorical('batch_size', | 32 |

| | | |
|---|---|---|
| | [16, 32, 64, 128]) | |
| Dropout | trial.suggest_float('dropout_rate', 0.1, 0.5) | 0.284558 |
| optimizer | trial.suggest_categorical('optimizer', ['adam', 'sgd', 'rmsprop']) | sgd |

**5.3 Comparison model architecture with random search CV and Bayesian optimizer**

The comparison model architecture with random search CV and Bayesian optimizer based on above table (Table 5-1 and Table5-6):-

Optimizer parameters: - the Bayesian optimization approach adds significant optimizer parameters compared to random search, especially in DNN and AutoEncoder models.

Model complexity: - the AutoEncoder model display substantial increases in total parameters with Random search, shows they may benefit from hyperparameter tuning than other models.

In general the hyperparameter tuning method affects both the model architecture and the parameter count. Bayesian optimization may lead to more complex models with potentially better performance, while Random search CV maintains simpler architectures with significant parameter increases.

**5.4 comparison model performance results using random search CV and Bayesian optimizer.**

Performance comparison: the random search CV shows that the FFNN and Ensemble perform well across all metrics, with showing the best overall performance, in contrast the Bayesian optimizer results drop in performance for the FFNN, while RF, AutoEncoder and DNN perform well, in MSE and RMSE; Bias and error metrics: the random search CV generally yields lower bias and error metrics across the board, while the Bayesian optimizer shows higher errors for the FFNN; and R-squared and explained variance in random search CV is higher show that better fit and predictive power compared to Bayesian optimizer results.

In general for our case the prediction power and minimal bias is crucial so that the random search CV is better.

# CHAPTER SIX
# CONCLUSION, CONTRIBUTION, AND FUTURE RECOMMENDATIONS

## 6.1Conclusion

We conclude the study by answering the research question. Research question 1. Which deep learning model can best suit predicting patterns and anomalies of tax evasion in a large dataset and compare with traditional predictive models for five consecutive years? We compare deep learning model DNN, AutoEncoder with artificial neural network feed-forward neural network. We found that the feed-forward neural networks performed best in all models. Research questions 4, Which Hyperparameter tuning technique is better optimizing the models? We compared two hyperparameter tuning techniques random search CV and Bayesian optimizer, the random search CV is best perform with minimal bias, mean squared error, root mean squared error and higher r-squared and explained variance score.

When we generalize the study, we used domain knowledge for data pre-processing; we model predictive analytics using deep neural networks, AutoEncoder, and feed-forward neural networks. We optimize the hyperparameter of the model using random search CV and Bayesian optimizer techniques. We found that the random search CV is better performed and feed forward neural networks best suit model. We saw the ethical consideration in data confidentiality, fair and equal decisions for risk analysis, and identifying the mean bias of algorithms ensuring that this research serves the customs commission.

## 6.2 Contribution of Study

1. Working the predictive analytics for controlling tax evasion using deep learning integrating company rules to handle the black box nature of deep learning algorithms
2. The study contributes by evaluating and comparing different deep learning models and selecting the best fit for predictive analysis.
3. The research done using local datasets from three source customs commission, domestic and PCA audit history.
4. The study contributes by evaluating and comparing two hyperparameter tuning technique, suggest the better optimizer.

5. Predict the next five years' audit selection and the top three variables that show tax evasion.

6. In predicting audit selection, the previous year's selected customer was excluded from the next year's audit selection, so that it reduced data because that it affect the performance of model prediction, to handle this we used transfer learning and more accurately predicted for five consecutive years.

## 6.3 Future Recommendations

Conduct different behavioural models since frequently the customer behaviour is changing and incorporating it’s to models.

Conduct predictive analytics for controlling tax evasion, data analysis of each variable is done separately or sequentially but feature researcher uses greedy algorithm or any parallel processing system to minimize analysis time and make a more efficient system.

Customise uses predictive analytics for controlling tax evasion related to the online market and works.

Continuously monitor and implementing a feedback loop to retrain and validate the models periodically can ensure their effectiveness and adaptability to evolving patterns and trends.

# REFERENCE


Baghdasaryan, V., Davtyan, H., Sarikyan, A., & Navasardyan, Z. (2022). Improving Tax Audit Efficiency Using Machine Learning: The Role of Taxpayer's Network Data in Fraud Detection. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2021.2012002

Chan, T., Tan, C. E., & Tagkopoulos, I. (2022). Audit lead selection and yield prediction from historical tax data using artificial neural networks. *PLoS ONE*, *17*(11 November). https://doi.org/10.1371/journal.pone.0278121

Andrew W. Trask: grokking Deep Learning book, 2019

Di Oliveira, V., Chaim, R. M., Weigang, L., Neto, S. A. P. B., & Filho, G. P. R. (2021). Towards a Smart Identification of Tax Default Risk with Machine Learning. *International Conference on Web Information Systems and Technologies, WEBIST - Proceedings*, *2021-October*, 422–429. https://doi.org/10.5220/0010712200003058

Hanifah, A. F., & Kusumaningrum, R. (2021). Non-Factoid Answer Selection in Indonesian Science Question Answering System using Long Short-Term Memory (LSTM). *Procedia Computer Science*, *179*, 736–746. https://doi.org/10.1016/j.procs.2021.01.062

Hashimzade, N., Myles, G. D., & Rablen, M. D. (2015). *Predictive Analytics and the Targeting of Audits*.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. 2015. Deep learning applications and challenges in big data analytics. Journal of Big Data, 2(1), 1-21.

Hsu, K.-W., Pathak, N., Srivastava, J., Tschida, G., & Bjorklund, E. (2015). *Data Mining Based Tax Audit Selection: A Case Study of a Pilot Project at the Minnesota Department of Revenue* (pp. 221–245). https://doi.org/10.1007/978-3-319-07812-0_12

Huang, F., & Vasarhelyi, M. A. (2019). Applying robotic process automation (RPA) in auditing: A framework. *International Journal of Accounting Information Systems*, *35*. https://doi.org/10.1016/j.accinf.2019.100433

Bartz, E., Bartz-Beielstein, T., Zaefferer, M., & Mersmann, O. (2023). *Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide*. Springer. https://link.springer.com/book/9789811951695

Mosavi, A., Ardabili, S., & Várkonyi-Kóczy, A. R. (2020). List of Deep Learning Models. In *Lecture Notes in Networks and Systems* (Vol. 101, pp. 202–214). Springer. https://doi.org/10.1007/978-3-030-36841-8_20

CPI (Centre for Public Impact). (2018). Artificial Intelligence in Taxation Case Study on the use of AI in Government. A BCG Foundation. Retrieved fromhttps://resources.centreforpublicimpact. org/production/2019/01/CPI-AI-Case-Study-Taxation.pdf.

Ippolito, A., & Lozano, A. C. G. (2020). Tax crime prediction with machine learning: A case study in the municipality of São Paulo. *ICEIS 2020 - Proceedings of the 22nd International Conference on Enterprise Information Systems*, *1*, 452–459. https://doi.org/10.5220/0009564704520459

በኢትዮጵያ ገቢዎችና ጉምሩክ ባለሥልጣን የሥጋት አመራር ዳይሬክቶሬት፣ የድህረ ዕቃ አወጣጥ የመስክ ኦዲት የመምረጫ መስፈርት የአሰራር ሰነድ፣ ግንቦት 2010ዓ.ም

የጉምሩክ ቅ/ጽ/ቤቶች ስጋት ሥራ አመራር የአሠራር ማኑዋል ሁለተኛ ዕትም፣ ሴነ 2009ዓ.ም

*GUIDELINES FOR POST-CLEARANCE AUDIT (PCA) VOLUME 1*. (2018).

WCO GUIDELINE VOL2,(2011).

Kishore Babu, S., & Vasavi, S. (2019). Predictive analytic as a service on tax evasion using feature engineering strategies. *Smart Innovation, Systems and Technologies*, *104*, 393–402. https://doi.org/10.1007/978-981-13-1921-1_39

Kleanthous, C., & Chatzis, S. (2020). Gated Mixture Variational AutoEncoders for Value Added Tax audit case selection. *Knowledge-Based Systems*, *188*. https://doi.org/10.1016/j.knosys.2019.105048

Kumar, S. (2018). *Predictive Analytics For Controlling Tax Evasion*.

Lars Föhr, T., Marten, K.-U., & Schreyer, M. (n.d.). *Deep Learning meets Risk-based Auditing: A holistic Framework for leveraging Foundation and Task-specific Models in Audit Procedures*.

Perbendaharaan, J., Negara Dan Kebijakan Publik, K., David Febriminanto, R., & Wasesa, M. (2022). Indonesian Treasury Review Machine Learning Analytics For Predicting Tax Revenue Potential. In *Machine Learning Analytics For Predicting Tax Revenue Potential Indonesian Treasury Review* (Vol. 7, Issue 3). Keuangan Negara dan Kebijakan Publik. www.pajak.com

Julia Kagan,"Tax evasion: meaning, definition and penalties", June, 18, 2022

Hatcher, W. G., & Yu, W. (2018). A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access*, *6*, 24411–24432. https://doi.org/10.1109/ACCESS.2018.2830661

Rustagi, M., & Goel, N. (2022). International Association of Research Scholars. *Organismo Internacional IARS' International Research Journal*, *12*(1).

Brenner, W., B. Van Giffen, and J. Koehler. 2020. Management of Artificial Intelligence: Feasibility, Desirability and Viability. In *Engineering the Transformation of the Enterprise: A Desgin Science Research Perspective*, edited by S. Aier, P. Rohner and J. Schelp, 15-36. Cham, Switzerland.

Deußer, T., S. M. Ali, L. Hillebrand, D. Nurchalifah, B. Jacob, C. Bauckhage, and R. Sifa. 2022. KPI-EDGAR: A Novel Dataset and Accompanying Metric for Relation Extraction from Financial Documents. *arXiv preprint arXiv:2210.09163*.

Deußer, T., M. Pielka, L. Pucknat, B. Jacob, T. Dilmaghani, M. Nourimand, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa. 2023. Contradiction Detection in Financial Reports. *Proceedings of the Nothern Lights Deep Learning Workshop* 4

Sólon da Silva, L., de Rigitano, H. C., Carvalho, R. N., & Carlos Souza, J. F. (n.d.). *Bayesian Networks on Income Tax Audit Selection-A Case Study of Brazilian Tax Administration*.

Srivastava, P. R., Zhang, Z., & Eachempati, P. (2021). Deep neural network and time series approach for finance systems: Predicting the movement of the Indian stock market. *Journal of Organizational and End User Computing*, *33*(5), 1–24. https://doi.org/10.4018/JOEUC.20210901.oa10

Sun, T., Vasarhelyi, M. A., & Kogan, A. (2018). *DEEP LEARNING APPLICATIONS IN AUDIT DECISION MAKING*.

Ul Islam, R., Hossain, M. S., & Andersson, K. (2020). A deep learning inspired belief rule-based expert system. *IEEE Access*, *8*, 190637–190651. https://doi.org/10.1109/ACCESS.2020.3031438

Davenport, T.H. & Harris, J. 2007. Competing on Analytics: The New Science of Winning. Harvard Business School Press.

Davenport, T.H. & Harris, J. 2010. Analytics at Work: Smarter Decisions, Better Results. Harvard Business Press.

Miller, G.J, Bräutigan, D. & Gerlach, S.V. 2006. Business Intelligence Competency Centre's: A Team Approach to Maximizing Competitive Advantage. Wiley.

Hashim Zade, N., Myles, G D., Rable, M D. (2016). Predictive Analytics and the Targeting of Audits. Journal of Economic Behavior and Organization 124, 130-145.

Bates, D. W., Saria, S., Ohno-Machado, L., Shah, A., & Escobar, G. (2014). Big Data In Health Care: Using Analytics to Identify and Manage High-Risk and High-Cost Patients. Health Affairs, 33(7), 1123–1131.

Petersen, C. (2018). Through Patients' Eyes: Regulation, Technology, Privacy, and the Future. Yearbook of Medical Informatics, 27(01), 010–015.

Hacker, P. (2018). Teaching Fairness to Artificial Intelligence: Existing and Novel Strategies Against Algorithmic Discrimination Under EU Law (SSRN Scholarly Paper No. ID 3164973).

Sokolova, M. and Lapalme, G. (2009). *A Systematic Analysis of Performance Measures for Classification Tasks*. Information Processing and Management, 45(4), 427-437.

Wachter, S., Mittelstadt, B., & Floridi, L. (2017a). Transparent, explainable, and accountable AI for robotics. Science Robotics, 2(6), eaan6080.

Wachter, S., Mittelstadt, B., & Floridi, L. (2017b). Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation. International Data Privacy Law, 7(2), 76–99.

Baru, Chaitanya. (2019). *2019 IEEE International Conference on Big Data : proceedings : Dec 9 - Dec 12, 2019, Los Angeles, CA, USA*. IEEE.

Goumagias, N. D., Hristu-Varsakelis, D., & Assael, Y. M. (2018). *Using deep Q-learning to understand the tax evasion behavior of risk-averse firms*.

Gierbl, A., M. Schreyer, D. Borth, and P. Leibfried. 2021. Deep Learning für die Wirtschaftsprüfung: Eine Darstellung von Theorie, Funktionsweise und Anwendungsmöglichkeiten. *Zeitschrift für internationale Rechnungslegung (IRZ)* 16 (7/8): 317-322.

Barocas, S., & Salbst, A.D. (2016), Big Data's Disparate Impact SSRN Electronic Journal

Schreyer, M., T. Sattarov, A. Gierbl, and B. Reimer. 2020. "Learning Sampling in Financial Statement Auditing using Vector Quantized AutoEncoder Neural Networks." Proceedings of the First ACM International Conference on AI in Finance.

Sun, T., and M. A. Vasarhelyi. 2017. Deep Learning and the Future of Auditing: How an Evolving Technology Could Transform Analysis and Improve Judgement. *The CPA Journal* 87 (6): 24-29.

Heaton, J. B., Polson, N. G., & Witte, J. H. (n.d.). *Deep Learning for Finance: Deep Portfolios*. https://ssrn.com/abstract=2838013

Chen, X. W., & Lin, X. (2014). Big data deep learning: Challenges and perspectives. In *IEEE Access* (Vol. 2, pp. 514–525). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2014.2325029

Murorunkwere, B. F., Haughton, D., Nzabanita, J., Kipkogei, F., & Kabano, I. (2023). Predicting tax fraud using supervised machine learning approach. *African Journal of Science, Technology, Innovation and Development*, *15*(6), 731–742. https://doi.org/10.1080/20421338.2023.2187930

Zuo, X., Yang, X., Dou, Z., & Wen, J. R. (2019). RUCIR at TREC 2019: Conversational Assistance Track. *28th Text REtrieval Conference, TREC 2019 - Proceedings*. https://doi.org/10.1145/1122445.1122456

W. Fox, L. Lunab and G. Schau, "Destination Taxation and Evasion: Evidence from U.S. InterState Commodity Flows," Journal of Accounting and Economics, vol. 57, no. 1, p. 43–57, 2017

Alm J. "Tax Compliance and Administration," in Handbook on Taxation; eds. Hildreth W. B., Richardson J. A., pp. 741–768. Marcel Dekker, Inc. 1993

Cleary, D. (2011). Predictive Analytics in the Public Sector: Using Data Mining to Assist Better Target Selection for Audit. In Electronic Journal of e-Government (Vol. 9). www.ejeg.com

EU Tax Observatory. (2023). *Global Tax Evasion Report 2024*. https://www.taxobservatory.eu/publication/global-tax-evasion-report-2024/

Social Europe. (2023). *Global tax evasion: the good and the bad news*. https://www.socialeurope.eu/global-tax-evasion-the-good-and-the-bad-news

Worku Mekonnen. (2021). *Determinants of Tax Evasion in Ethiopia*. St. Mary's University Institutional Repository. http://repository.smuc.edu.et/bitstream/123456789/1777/1/WORKU%20MEKONNEN.pdf

Damissie Tolossa. (2023). *The Effect of Tax Audit on Tax Compliance*. Jimma University.

Mihiret, A. (2011). *Tax Audit Practice in Ethiopia: A Case Study of Federal Government*. St. Mary's University. http://www.repository.smuc.edu.et/bitstream/123456789/3173/1/cd.pdf

World Bank. (n.d.). *Risk-Based Tax Audits*. https://openknowledge.worldbank.org/server/api/core/bitstreams/d698f835-f3bb-546e-8f51-4ee984342994/content

Mughal, M. (2012). *Tax evasion, psychological egoism, and revenue collection performance: Evidence from Amhara region, Ethiopia*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9944599/

OECD. (2021). *Tax Administration 2021: Comparative Information on OECD and Other Advanced and Emerging Economies*. OECD Publishing. https://www.oecd.org/tax/administration/tax-administration-2021-9789264639422-en.htm

Linda Tucci, TechTarget"predictive analytics: enterprise guide", December, 2021

Institute on Taxation and Economic Policy (ITEP). (2020). *Who Pays? A Distributional Analysis of the Tax Systems in All 50 States*.https://itep.org/wp-content/uploads/whopays-2020.pdf

World Bank. (2019). *The Global Financial Development Report 2019: Bank Regulation and Supervision*. https://www.worldbank.org/en/publication/gfdr

United Nations. (2021). *Financing for Sustainable Development Report 2021*.

Plevris, V., Solorzano, G., Bakas, N. P., & Ben Seghier, M. E. A. (2022). Investigation of performance metrics in regression analysis and machine learning-based prediction models. Retrieved from https://www.researchgate.net/publication/362719708

# APPENDIX

Appendix A: Model implementation

A.1 sample Library

```
import tensorflow as tf # data automation, model tracking, performance monitoring, and model retraining.
import pandas as pd # working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.
import numpy as np # to perform a wide variety of mathematical operations on arrays
import matplotlib.pyplot as plt # for data visualization
import tensorflow_datasets as tfds # provides a collection of ready-to-use datasets for use with TensorFlow
import seaborn as sns # for data visualization
from tensorflow.keras.utils import plot_model
import scipy.stats as stats # to calculate probability function mean, std from datasets.
from sklearn.model_selection import train_test_split # to split data in Time frame
from functools import reduce # to merge multiple dataframes as separate arguments.
from sklearn.metrics import mean_squared_error, r2_score, explained_variance_score # to evaluate parameter model
from tensorflow.keras.models import Sequential
```

A.2 Data loading

```
Data1 = pd.read_csv(r"ECMS import dataset 1.csv", encoding='latin-1') #loading data1 and NB: Latin-1 because the dataset contains characters,including the ASCII character set (the basic Latin alphabet, digits, punctuation marks, etc.) and mathematical symbols to support all this used latin-1
Data2 = pd.read_csv(r"Domestic dataset 2.csv") #loading data2
Data3 = pd.read_csv(r"PCA history dataset 3.csv") #loading data3
```

A.3 Pre-processed data

| | TIN | CIF Risk | Customer Profile Risk | Declarant Profile Risks | Country of origin Risk | Country of consignment Risk | Declaration Risks | Domestic risk | Risk of Intelligence | PCA history Risks1 | Random Selected Risk | PA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00000000KR | 0 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 0 | 0 | 0.68 |
| 1 | 0000000LBS | 0 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 0 | 0 | 0.68 |
| 2 | 0000002006 | 0 | 1 | 1 | 2 | 2 | 0 | 1 | 2 | 0 | 0 | 0.64 |
| 3 | 0000002032 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 0 | 0 | 0.84 |
| 4 | 0000002223 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 0.68 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 15065 | 0270722333 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 1 | 0 | 0.68 |
| 15066 | 0270727383 | 0 | 1 | 1 | 2 | 2 | 1 | 0 | 2 | 0 | 0 | 0.52 |
| 15067 | 0270738256 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 0 | 0 | 0.84 |
| 15068 | 0270768607 | 2 | 2 | 2 | 2 | 2 | 0 | 2 | 2 | 1 | 2 | 1.80 |
| 15069 | 0270802238 | 0 | 1 | 2 | 0 | 0 | 2 | 2 | 2 | 1 | 0 | 0.92 |

15070 rows × 12 columns

A.4 Data spilt

```
X_train shape: (12056, 10)
y_train shape: (12056,)
X_test shape: (1507, 10)
y_test shape: (1507,)
X_val shape: (1507, 10)
y_val shape: (1507,)
```

A.4 Model Architecture FFNN model architecture.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_106 (Dense) | (None, 64) | 704 |
| dense_107 (Dense) | (None, 32) | 2,080 |
| dense_108 (Dense) | (None, 1) | 33 |

```
Total params: 8,453 (33.02 KB)
Trainable params: 2,817 (11.00 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 5,636 (22.02 KB)
```