UNCOVERING INCOME TAX FRAUD: A LOGISTIC REGRESSION APPROACH FOR DETECTION AND PREVENTION

A Project Report

Submitted by,

SHIVAM NARAYAN - 20191ISE0160

Under the guidance of,

Dr. P Sudha Assistant Professor (SG)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY
IN
INFORMATION SCIENCE AND ENGINEERING

At

SCHOOL OF COMPUTER SCIENCE ENGINEERING & INFORMATION SCIENCE
PRESIDENCY UNIVERSITY
BENGALURU

MAY 2023

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project report **"UNCOVERING INCOME TAX FRAUD: A LOGISTIC REGRESSION APPROACH FOR DETECTION AND PREVENTION"** being submitted by "SHIVAM NARAYAN" bearing roll number "20191ISE0160" in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Information Science and Engineering is a Bonafide work carried out under my supervision.

Dr. C. KALAIARASAN
Associate Dean
School of CSE&IS
Presidency University

Dr. MD. SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

Dr. P SUDHA
Assistant Professor (SG)
Dept of CSE&IS
School of CS&ISE
Presidency University

Dr. SHANMUGARATHINAM G
HOD SCSE
School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **UNCOVERING INCOME TAX FRAUD: A LOGISTIC REGRESSION APPROACH FOR DETECTION AND PREVENTION** in partial fulfilment for the award of Degree of **Bachelor of Technology** in Information Science and Engineering, is a record of our own investigations carried under the guidance of **Dr. P Sudha, Assistant Professor (SG) , Dept of CSE, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Name	**Roll Number**	**Signature**
SHIVAM NARAYAN	20191ISE0160	

ABSTRACT

The compulsory tax levied by the government on individuals and businesses based on their income is known as income tax. Tax fraud involves the intentional manipulation of information on a tax return to reduce tax liability. Our project focuses on developing a machine learning model to identify income tax fraud by analyzing taxpayers' financial data. Six machine learning algorithms namely: Logistic Regression, Decision Tree, Random Forest, Naive Bayes, k-Nearest Neighbors and Feed forward Neural Network were compared, and logistic regression was found to be the most effective in detecting tax fraud. Compared to existing methods, the proposed model captures both linear and non-linear relationships among variables, making it more accurate in detecting complex patterns. The model was developed by training it on a OpenML dataset and evaluate on a test dataset. The research aim is to develop a model that can accurately detect tax fraud, and the objectives include comparing the effectiveness of various machine learning algorithms, identifying significant factors contributing to tax fraud, and providing insights for policymakers. The proposed model has significant potential in detecting tax fraud, which can reduce revenue losses and promote fairness in the tax system while remaining an affordable solution. Furthermore, the best performing model is deployed into Android Studio to develop a prediction app using TensorFlow, enhancing its practical usability and accessibility for users.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science & Engineering, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Dean **Dr. C. Kalaiarasan,** Professor **Dr. T K Thivakaran,** University Project-II In-charge, School of Computer Science & Engineering, Presidency University for rendering timely help for the successful completion of this project.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Co-Ordinators **Mr. Mrutyunjaya MS, Mr. Sanjeev P Kaulgud, Mr. Rama Krishna K and Dr. Madhusudhan MV**.

We are greatly indebted to our guide **Dr. P Sudha, Assistant Professor (SG)** School of Computer Science & Engineering, Presidency University for her inspirational guidance, valuable suggestions and providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Shivam Narayan

List of Tables

List of Figures

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

1.1 Problem Statement:

There is a grocery shop owner in Aundh Pune (decent area to live) who is filing an average income tax of 10k every year. This corresponds to a profitable income of 3L per annum. Considering a good profit margin of 20%, his total sales should be around 15L per financial year. Through our product that uses AI and ML, this will verify this data, by backtracking many related transactions.

1. As per the address and Aadhar information, he has two kids studying in school having average yearly expenses of 1.5L each (totals to 3L).
2. Property tax to the Govt reveals, he is living in an area that costs around 40K (taxes and maintenance).
3. Owns a car of 60k annual maintenance.
4. Pays 2L salary to his shop employees.
5. Minimum living cost is 4L for a family of four.
6. Average customers per shop in the area is 5K, per person average grocery expenditure is 2k (monthly). This does not include online shopping.
7. Pan information indicates 1L jewelry shopping this financial year.
8. Online shopping delivered at his address 50K. As per this data, his total income should be around 50L whereas he has declared only 3L (a vast difference).

The calculations indicate he is actually hiding his total income and stealing taxes. With an efficient system in place, these thefts can be identified.

1.2 Objective of the Project:

- Our project's goal is to create a predictive model utilizing logistic regression to enhance the detection of income tax fraud. This will be achieved by incorporating behavioral and demographic variables that could potentially contribute to tax fraud.
- The project seeks to address the research gap regarding the underutilization of behavioral and demographic factors in detecting tax fraud, employing the logistic regression framework.
- Through the accurate identification of fraudulent tax returns, we aim to support a just and ethical tax system that can bolster government revenue and reinforce the overall tax compliance framework.

1.3 Project Introduction:

Income tax is crucial for the functioning of our society, as it provides countries with the necessary revenue to make vital investments in infrastructure, health, and education. However, despite its importance, many people are averse to paying taxes, and make the government lose millions of dollars every year. There are various strategies to evade taxes, such as underreporting income, which reduces the tax liability. Criminals who commit fraud are becoming increasingly sophisticated in their methods, making it difficult to identify them. In many cases, they try to blend in with their environment, much like military units that use camouflage or chameleons that use their coloring to hide from predators. These tactics are not random, but rather carefully planned and executed. As a result, new techniques are needed to detect and address patterns that appear to be normal but are actually part of fraudulent activities. Tax authorities are given the task of finding these fraudsters and usually rely on experts' intuition. Random auditing is a way of discouraging tax frauds. Unfortunately, this approach is not cost-effective, and auditing some types of taxes can take up to six months or even a year, which puts a significant burden on the already overloaded tax auditors. Traditional methods, such as manual audits or statistical analysis, are time-consuming, expensive, and often ineffective. Therefore, the use of artificial intelligence or machine learning techniques has gained popularity in recent years, as they can analyze large datasets and detect patterns that humans may miss.

Machine learning (ML) is a field within artificial intelligence (AI) that employs statistical models and algorithms to enable computer systems to learn from data and enhance their performance on specific tasks. In essence, machine learning algorithms empower computers to learn and make decisions based on identified patterns and trends within extensive datasets. There are three primary categories of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning involves providing labeled data to the machine, which it can then utilize to make predictions or classifications. On the other hand, unsupervised learning entails feeding unlabeled data to the machine, enabling it to independently identify patterns and relationships. Reinforcement learning is a machine learning approach where the machine learns by taking actions in an environment and receiving feedback in the form of rewards or penalties.

Machine learning has exhibited its potency across various domains, including healthcare, finance, marketing, and cybersecurity. Its capacity to learn from data and adapt to new circumstances without explicit programming has established it as an invaluable tool in contemporary data analysis.

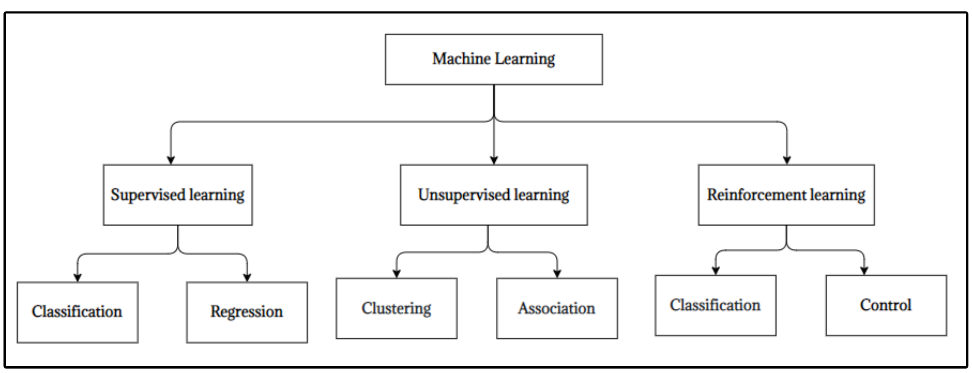

Fig-1 Classification of machine learning

Android Studio is an IDE (Integrated Development Environment) designed specifically for developing Android apps. The platform offers a wide range of tools and features that help developers with the design, coding, debugging, and deployment of their applications. Android

Studio is built on top of IntelliJ IDEA and comes bundled with the Android SDK, which contains the necessary APIs, libraries, and emulators to develop and test Android apps. This development platform supports different programming languages, such as Java, Kotlin, and C++. It also includes advanced features like code completion, debugging, profiling, and testing. Using Android Studio, developers can create aesthetically pleasing user interfaces, integrate with various backend services, and utilize machine learning frameworks such as TensorFlow to develop data-driven applications. It has revolutionized Android app development and made it easier than ever for developers to create high-quality, robust apps for the Android platform. Millions of Android devices worldwide are compatible with apps built with Android Studio.

This project report presents a methodology aimed at detecting and preventing income tax fraud through the utilization of diverse machine learning algorithms. Our study encompasses the implementation of decision tree, random forest, naive Bayes, k-nearest neighbors, feed-forward neural network, and logistic regression models, all of which contribute to the accurate identification of suspicious activities associated with income tax fraud. Our approach centers around the analysis of the OpenML Income Dataset, with particular emphasis on integrating behavioral and demographic factors into the logistic regression model. To assess the effectiveness of our approach, we conducted rigorous training and testing procedures using the dataset, resulting in noteworthy enhancements in fraud detection and prevention. Additionally, we successfully created an Android application by deploying the most effective model within the Android Studio IDE. Our innovative solution offers a distinct strategy to combat the enduring issue of income tax fraud that persists in numerous countries.

CHAPTER-2

LITERATURE SURVEY

2.1 Related Works:

The Article uses neural networks for fraud detection, which is a popular and effective technique in machine learning. It provides a case study of income tax fraud detection and claims to achieve a high level of accuracy. However, the problem statement provides limited data, and the proposed method relies on accessing sensitive personal data, raising concerns about data privacy. [1]. Neural networks represent a formidable machine learning technique capable of discerning intricate patterns within vast and intricate datasets, resulting in accuracy rates exceeding 95%. Nevertheless, detecting instances of tax evasion poses a challenge due to the absence of transparency, the potential for erroneous findings, and inadequate information availability. [2]. Data mining methods offer the possibility of automating the analysis of extensive datasets to detect taxpayers at high risk, thus minimizing the need for manual examination. This approach proves to be efficient and time-saving compared to traditional analysis, while also providing enhanced precision and flexibility. These techniques can be successfully applied to diverse data sources, including financial transactions, tax returns, and other pertinent information, and can easily accommodate large datasets. However, it is crucial to acknowledge that the accuracy of data mining techniques in identifying tax fraud is influenced by factors such as data quality, algorithm choice, model interpretation, and concerns surrounding privacy.[3]. The Improved Particle Swarm Optimization Algorithm has been improved to better detect tax evasion, with an accuracy rate of 95%. It is time and cost-effective but must be validated on a larger dataset to ensure it is robust and accurate. [4].

The proposed method uses a clustering technique to identify groups of taxpayers who have similar income profiles and then identifies those who are reporting significantly lower incomes. It relies heavily on a specific set of features and requires manual investigation to confirm whether tax fraud has occurred. [5]. Milos Savić developed a novel method for detecting tax evasion risks called Hybrid Unsupervised Outlier Detection. This approach combines the strengths of both unsupervised and supervised techniques to enhance the accuracy of the detection process. Although the method can identify tax evasion in a particular case involving a grocery shop owner, its applicability is limited, and it fails to address the ethical and legal issues associated with detecting tax evasion. Therefore, it is essential to test its reliability and effectiveness by applying it to different datasets and scenarios [6]. The article by González and Velásquez titled "Characterization and Detection of Taxpayers Engaging in Tax Evasion through False Invoices: A Data Mining Approach" concentrates on the identification and profiling of individuals who employ false invoices to evade taxes within the context of Colombia. The authors employed various data mining techniques, including decision trees, neural networks, and logistic regression, to unveil patterns within tax data that can aid in recognizing fraudulent activities. The findings of this study hold significant relevance for policymakers and tax authorities aiming to enhance tax compliance and minimize instances of tax evasion. [7]. This paper uses a neural network to detect credit card fraud. Neural networks are difficult to understand and require a lot of information to train, making them less effective in smaller datasets. Additionally, they are expensive to train and deploy and do not address issues related to data privacy and security. Appropriate measures need to be taken to ensure data is protected and used ethically [8].

AI and ML algorithms have proven to be valuable in identifying fraudulent tax returns during income tax audits. The application of these algorithms in Taiwan has showcased successful outcomes for both profit-seeking enterprise income tax and individual income tax, as illustrated in this study. This research offers significant insights into the key factors that contribute to tax fraud, thereby assisting in the formulation of efficient tax policies and regulations. It is crucial to acknowledge that the findings are exclusive to Taiwan's tax system and may not be directly applicable to India's tax system. Moreover, additional research is required to explore strategies for handling missing or unreliable data within the system. [9]. The book covers various techniques for fraud detection, including descriptive, predictive, and social network analysis. It provides practical examples and case studies of fraud detection in various industries and emphasizes the use of data mining tools for fraud detection. It provides a general approach to fraud detection, with limited focus on tax fraud, lack of emphasis on regulatory compliance, and dependence on data availability. The effectiveness of the techniques may depend on the availability and quality of data, which may be a limitation in the context of the given problem statement [10]. The research article provides a comparative analysis of supervised and unsupervised neural networks. It uses a large sample size of 1,700 Korean firms over a 10-year period and uses various financial ratios and non-financial factors as input variables. The study found that both supervised and unsupervised neural networks can effectively predict bankruptcy, which highlights the usefulness of machine learning techniques in financial analysis. Legal and ethical considerations should be considered when using such a system [11].

Detecting financial fraud presents a significant challenge due to the recurring use of illicit methods. In order to address this issue, a group of researchers conducted an analysis of 32 documents between 2015 and 2020 that discussed the advancements in neural network algorithms for fraud detection. The primary focus of their study was on deep neural network algorithms (DNN), convolutional neural networks (CNN), neural networks with SMOTE, and other complementary methodologies within artificial neural networks (ANN). Their experiments aimed to detect instances of credit card fraud and enhance online transaction security. The comparative analysis demonstrated that the convolutional ANN employing functional sequencing, the ANN integrated with Gradient Boosting Decision Tree (XGBoost), and the ANN utilizing automatic ontology learning fulfilled the prerequisites in terms of theoretical foundation, mathematical development, experimental investigation, and result accuracy. Nevertheless, it is crucial for future research to account for the temporal, financial, and data characterization aspects involved in neural network training, as these factors have a substantial impact on the effectiveness of the algorithms. [12]. Neural networks are an affordable and straightforward way to simplify analysis by avoiding the need to consider many statistical assumptions, such as matrix homogeneity, normality, and data processing. These models can automatically adjust connection weights and are fault-tolerant. They can also include all accessible variables in model estimation and enable quick revisions. A study found that the Multilayer Perceptron is effective for identifying fraudulent taxpayers and determining the likelihood of tax evasion, with an efficacy of 84.3%. The ROC curve-based sensitivity analysis demonstrated the model's excellent ability to distinguish between fraudulent and non-fraudulent taxpayers. The Multilayer Perceptron network appears to be a highly effective way to classify taxpayers, and this study's results offer opportunities for improving tax fraud detection by predicting fraud tendencies through sensitivity analysis. It would be interesting to explore the use of this concept in other taxes in the future [13].

CHAPTER-3

PROPOSED METHOD

3.1 Drawbacks of the Existing Works:

- The existing work primarily focuses on general tax fraud detection or prediction models that are applied to a wide range of scenarios. The problem statement provided, on the other hand, describes a specific case with unique factors, such as the number of children, property tax, car maintenance, employee salaries, minimum living costs, average customer behavior, jewelry shopping, and online shopping. These factors need to be considered to accurately assess the discrepancy between the declared income and the expected income.
- To address this drawback, a more tailored and customized approach is required. The existing work can serve as a foundation or reference, but additional research and analysis specific to the given scenario would be necessary to develop an efficient system for identifying tax evasion in the case of the grocery shop owner. This would involve considering the specific features and variables relevant to the case and incorporating them into the detection model or algorithm.
- In our project, the dataset used for analysis was the OpenML income dataset, which contains similar variables to those encountered in the case. However, to effectively tackle the grocery shop owner's situation, it would be necessary to perform additional research and analysis specific to this scenario.

3.2 Proposed System:

A. Logistic Regression (LR)

It is a popular approach in machine learning that is used to solve binary classification problems. To determine the likelihood of a specific outcome, the logistic function is utilized to express the relationship between the input features and the output variable.

The logistic function can be expressed as follows:

$$P(y=1|x) = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+\cdots\ldots+b_nx_n)}}$$

Where:

- $x_1, x_2, \ldots, x_n$ are the input features.
- $b_0, b_1, b_2, \ldots, b_n$ are the co-efficient of the input features.
- The natural logarithm base e is used in the expression.

An optimization procedure, such as gradient descent or Newton-Raphson, is applied to calculate the coefficients. Once the coefficients are determined, the logistic function can be used to predict the probability of the outcome for new observations.

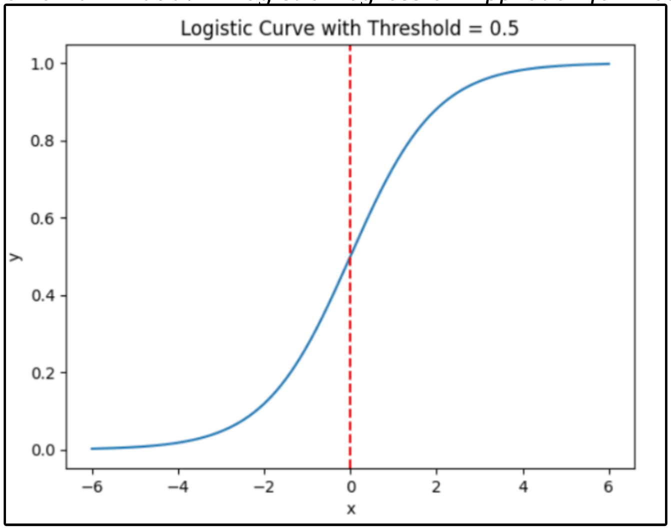

Fig-2 Logistic curve

The threshold for identifying observations as belonging to one of the binary outcomes can be set to 0.5 (i.e., if P(y=1|x) is larger than 0.5, the observation belongs to the positive outcome; otherwise, it belongs to the negative outcome).

B. Decision Tree (DT)

The DT algorithm is a popular technique used in machine learning for classification and regression tasks. It represents each internal node as a feature, each branch as a decision based on that feature, and each leaf node as a class or value. The approach recursively divides the data into subsets based on the most informative features until it meets a stopping requirement, such as maximum depth or minimum number of samples per leaf.

To mathematically represent the decision tree, we can use the equation:

$$f(x) = \sum \quad yi \cdot I(xi \in Ri)$$

Where:

- $f(x)$ denotes the predicted output for a new input x .

- y_i represents the output value for the i^{th} leaf node.
- R_i is the region of the i^{th} leaf node defined by the decision tree.
- $I(x_i \in R_i)$ is an indicator function that returns 1 if the input x_i belongs to the region R_i, and 0 otherwise.

In other words, this equation calculates the predicted output by adding the output values of the leaf nodes whose regions the new input belongs to. The decision tree algorithm learns to partition the input space into regions where the output values are similar and assigns a unique output value to each region.

C. *Random Forest (RT)*

A type of machine learning algorithm is capable of performing both classification and regression tasks. This algorithm belongs to the ensemble technique family, which combines multiple models to improve forecasting accuracy.

The algorithm works by creating a collection of decision trees based on random subsets of the training data and features. Each tree in the forest is trained on a distinct subset of the data, and the final forecast is made by aggregating the predictions of all the trees through a majority voting process.

The RF algorithm is described in detail below:

1. Randomly select 'n' samples from the training data set.
2. For each sample, randomly select k features from the total m features.
3. Build a decision tree using the selected samples and features.
4. Repeat steps 1-3 for T times to create T decision trees.
5. To make a prediction for a new data point, pass it through all T trees and calculate the average prediction from the majority votes.

The RF algorithm can be represented mathematically as follows:

a) Suppose we have a training dataset consisting of N observations with p input features and a corresponding set of target values $Y = \{y_1, y_2, \ldots., y_n\}$.
b) For each tree $t = 1, 2, \ldots., T$ in the forest, the algorithm selects a random subset of the training data D_t of size n (where $n < N$), and a random subset of the features F_t of size k (where $k < p$).
c) The algorithm then builds a decision tree using D_t and F_t are assigns a weight w_t to the tree based on its performance on the out-of-bag samples (samples not used in building the tree).
d) To make a prediction for a new data point x, the algorithm passes it through all T trees and calculates the average prediction as:

$$Y(x) = \left(\frac{1}{T}\right) \cdot \sum_{t=1}^{T} w_t \cdot h_t(x)$$

Where:

$h_t(x)$ is tree $t's$ prediction for input x and w_t is the weight given to tree t.

The Random Forest algorithm is good at dealing with high-dimensional datasets and is resistant to noise and outliers. It can also provide valuable information, which is useful for feature selection and model interpretation.

D. *Naive Bayes (NB)*

The Bayes' Theorem, a statistical principle that estimates the likelihood of a hypothesis based on knowledge of circumstances that might be relevant to the hypothesis, is the foundation of the Naive Bayes algorithm. The straightforward but efficient Naive Bayes method is frequently employed for classification jobs.

The basic principle of the Naive Bayes algorithm is to calculate the probability of each class based on a set of input features. The term "naive" refers to the assumption that each input feature is independent of all the other features. Despite this assumption, the algorithm often proves effective in real-world applications.

The following formula represents Naive Bayes:

$$P(y|x_1,\ x_2,\ \ldots.,\ x_n) = \frac{P(y) \cdot P(x_1|y) \cdot P(x_2|y) \cdot \ldots. \cdot P(x_n|y)}{P(x_1,\ x_2,\ \ldots.,\ xn)}$$

In this formula:

- $P(y|x_1, x_2, \ldots., x_n)$ denotes the posterior probability of class y given the input features $x_1, x_2, \ldots., x_n$.
- $P(y)$ represents the prior probability of class y .
- $P(x_i|y)$ denotes the probability of feature x_i given class y .
- $P(x_1, x_2, \ldots., x_n)$ represents the probability of the input features.

This is how the algorithm operates:

1. Calculate the prior probability $P(y)$ for each class given a collection of training data with labelled classes.
2. For each feature x_i calculate the conditional probability$P(x_i|y)$ for each class y .
3. For a new input instance with features $x_1, x_2, \ldots., x_n$ calculate the posterior probability $P(x_1, x_2, \ldots., x_n)$ for each class y using the above equation.
4. Assign the input instance to the class with the highest posterior probability.

Note that to ensure that the probabilities add up to 1, the denominator $P(x_1, x_2, \ldots., x_n)$ in the equation acts as a normalization constant. It can be determined by adding the numerator across all potential classes:

$$P(x_1,\ x_2,\ \ldots.,\ x_n) = \sum P(y) \cdot P(x_1|y) \cdot P(x_2|y) \cdot \ldots. \cdot P(x_n|y)\ for\ all\ classes\ y.$$

E. *k-Nearest Neighbors (K-NN)*

The k-NN algorithm is a supervised learning method that can be used for both classification and regression tasks. The main objective of this algorithm is to predict the label or value of a test data point by identifying the k data points in the training set that are closest to it.

To determine the distance between two data points, various metrics like the Manhattan distance, cosine similarity, or Euclidean distance can be employed. Based on the values or labels of the k-nearest neighbors, the algorithm can then predict the label or value of the test data point.

The equation for the k-NN algorithm can be expressed as follows:

For classification:

- Let D represent the training dataset.
- Let x be the test data point.

- The number of neighbors to take consideration is k .
- The distance metric between test point x & any point y in the dataset D is $dist(x, y)$.
- Let neighbors (x) be the set of k nearest neighbors to x in D .
- Let class (y) be the class label of y .

Then, the predicted value for x is:

predicted_class(x) = argmax(class(y)) for y in neighbors(x)

In this equation, **argmax** returns the class label that occurs most frequently among the k nearest neighbors.

For regression:

- Let D represent the training dataset.
- Let x be the test data point.
- The number of neighbors to take consideration is k .
- The distance metric between test point x & any point y in the dataset D is $dist(x, y)$.
- Let neighbors (x) be the set of k nearest neighbors to x in D .
- Let class (y) be the class label of y .

Then, the predicted value for x is:

predicted_value(x) = mean(value(y)) for y in neighbors(x)

In this equation, **mean** returns the average value of the k nearest neighbors.

F. *Feed Forward Neural Network (FFNN)*

A Feed Forward Neural Network (FFNN) is an artificial neural network designed to transmit information through multiple hidden layers in a unidirectional manner, moving from the input layer to the output layer. Unlike feedback loops, which are absent in the FFNN, the input is propagated across the network until it reaches the output layer.

The FFNN consists of neurons or perceptrons as its fundamental units. These units linearly transform input signals, apply an activation function, and pass the output to the subsequent layer. The layers of neurons in the FFNN are interconnected with the preceding and succeeding layers.

The initial layer of the FFNN is referred to as the input layer, which receives input data and forwards it to the top-level hidden layer. The hidden layer receives inputs from the previous layer, performs a linear transformation, applies an activation function, and transmits the output to the next layer. Finally, the output layer of the network generates the final output of the FFNN.

The equation for the output of a single neuron In an FFNN is as follows:

$$y = f\left(\sum_{i=1}^{n} w_i\, x_i + b\right)$$

Where:

- x_i is the input to the neuron from the previous layer or the input layer.
- w_i is the weight of the connection between the input x_i and the neuron.
- b is the bias term, which is added to shift the output of the neuron.
- $\sum_{i=1}^{n} w_i\, x_i + b$ represents a weighted sum of the inputs and bias.
- f is the activation function, which introduces non-linearity into the output of the neuron.

In a feedforward neural network (FFNN), an activation function f is used to introduce non-linearity. The most commonly used activation functions are the sigmoid, ReLU (Rectified Linear Unit), and softmax functions. The selection of the activation function depends on the problem and the desired output.

The output of the FFNN can be computed by combining the equations for each neuron in the network. During the training phase, the weights and biases of the neurons are learned by applying an optimization algorithm such as backpropagation.

3.3 Advantages of Proposed Method:

- The logistic regression approach has several benefits over traditional methods of income tax fraud detection. First, it is more accurate and efficient than manual methods of detecting fraud, which can be time-consuming and prone to errors.
- Second, it can be used to identify new types of fraud that may not have been detected using traditional methods. Finally, it can be used to prevent fraud before it occurs by identifying high-risk taxpayers and conducting audits or investigations.
- Third, logistic regression provides a probabilistic framework for fraud detection. It not only predicts the likelihood of an individual engaging in fraudulent activities but also assigns a probability score to each prediction.
- This probabilistic nature allows tax authorities to prioritize their efforts and allocate

resources effectively by focusing on individuals with higher fraud probabilities.

- By targeting high-risk taxpayers, logistic regression can help detect and prevent fraud more efficiently, leading to improved resource allocation and increased overall effectiveness of fraud detection measures.
- Fourth, logistic regression models can be easily interpreted and understood by stakeholders, including tax authorities, auditors, and taxpayers themselves. The coefficients associated with each predictor variable in the model provide valuable insights into the relative importance and direction of influence on fraud detection.
- This interpretability facilitates transparent decision-making processes and fosters trust in the fraud detection system. Tax authorities can explain the reasoning behind their actions and provide evidence based on the model's outputs, leading to increased accountability and fairness in fraud detection procedures.

Chapter-4

REQUIREMENT ANALYSIS

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

4.1 Functional Requirements & Non-functional Requirement:

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed, and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1) Authentication of user whenever he/she logs into the system
2) System shutdown in case of a cyber-attack
3) A verification email is sent to user whenever he/she register for the first time on some software system.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements. They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Examples of non-functional requirements:

1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
2) The processing of each request should be done within 10 seconds
3) The site should load in 3 seconds whenever of simultaneous users are > 10000

4.2 HARDWARE CONFIGURATION:

- Processor - I3/Intel Processor
- RAM - 8GB (min)
- Hard Disk - 500 GB

4.3 SOFTWARE CONFIGURATION:

- Operating System : Windows 10
- Software for model building : Jupyter Notebook (anaconda3)
- IDE used for app development : Android Studio
- Libraries Used : TensorFlow Lite

Chapter-5

METHODOLOGY

5.1 Project Flow:

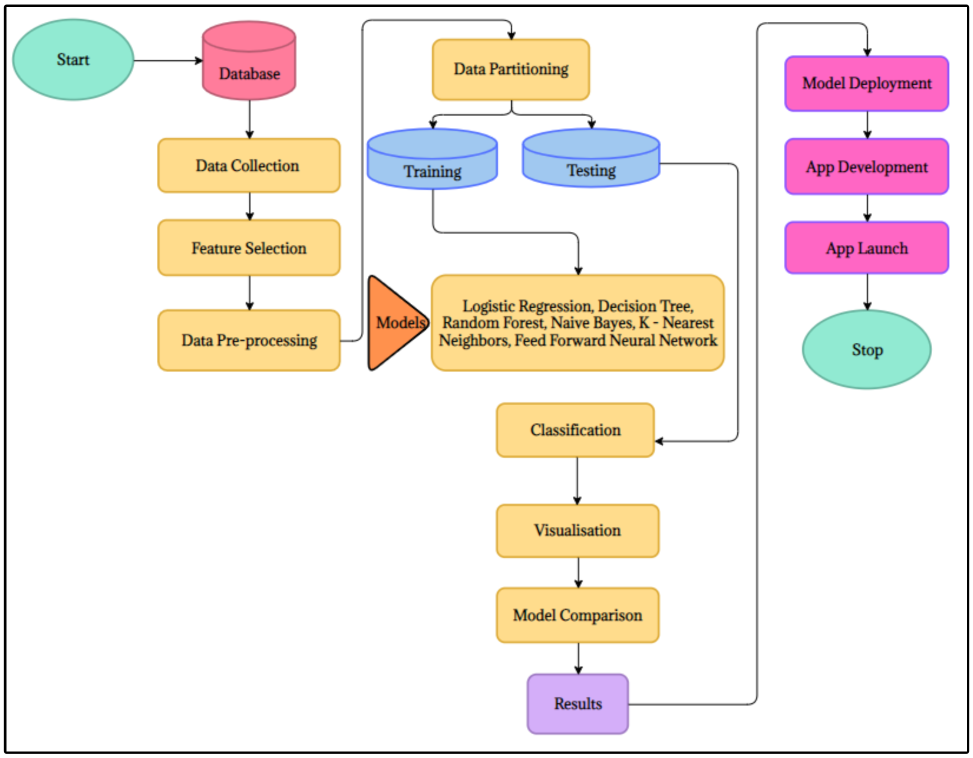

Fig-3 Architecture of the model

1. **Data Collection**: The data used to detect income tax fraud was obtained from the OpenML repository. The dataset included variables such as age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country, and income.
2. **Feature Selection**: The most important variables for detecting income tax fraud were selected using techniques like correlation analysis, feature importance, or domain knowledge. In this study, the selected features were age, workclass, race, sex, native-country, income (Demographic factors) and hours-per-week, education, marital-status, occupation, relationship (Behavioral factors).
3. **Data Pre-processing**: The data was pre-processed by removing missing values, outliers, and irrelevant variables. Additionally, feature scaling, normalization, and encoding categorical variables were performed.
4. **Data Partitioning**: The data was split into training and testing sets to train the models and evaluate their performance.
5. **Model Training**: Six different models, including Logistic Regression, Decision Tree,

Random Forest, Naive Bayes, k-Nearest Neighbors, and Feed Forward Neural Network, were trained on the selected features using the training data. These models learned to predict the likelihood of income tax fraud based on the input variables.

6. **Model Evaluation**: The performance of the trained models on the testing data was evaluated using metrics like accuracy, precision, recall, and F1 score to measure how well the model was performing in detecting income tax fraud.
7. **Visualization**: A visualization was created to compare the performance of different algorithms. Bar plots were used to compare the accuracy scores of different models, ROC curves to see which model performs better with an area under the curve (AUC), and precision-recall curves that compare precision and recall.
8. **Model Comparison**: The performance of each algorithm was compared to determine which algorithm performs the best.
9. **Model Deployment**: The best model is implemented in Android Studio using the TensorFlow library.
10. **Application Development**: The app's user interface is created through App view. Our implementation consists of three XML layout files that define the user interface. The app controller serves as the intermediate component between the Model and View elements, managing data flow and event handling.
11. **App Launch**: Upon running the application, the source code and resources are compiled into an APK. After installation, Android initiates the app by creating an instance of the main activity, which is then displayed on the screen.

5.2 Source Code:

5.2.1 Prediction Model Code:

Data Collection:

```
# Loading the OpenML Income Dataset

import pandas as pd

income_data = pd.read_csv("census-income.csv")

# Displaying first 5 rows

income_data.head()

# Structure of the dataset

income_data.info()

# No of rows and columns

income_data.shape

# Performing exploratory data analysis (EDA)

import matplotlib.pyplot as plt
```

```
import seaborn as sns

sns.pairplot(income_data[['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week', 'income']], hue='income')

plt.show()
```

Feature Selection:

```
relevant_features = ['age', 'workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'hours-per-week', 'native-country', 'income']

income_data = income_data[relevant_features]
```

Data Partitioning:

```
# Splitting the dataset into 80% as training and 20% as testing.

from sklearn.model_selection import train_test_split

X = income_data.drop(['income'], axis=1)

y = income_data['income']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Data Pre-processing:

```
# Scaling the dataset to ensure that each feature has the same scale

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer



# defining column transformer to encode categorical variables

categorical_transformer = OneHotEncoder(handle_unknown='ignore')



# defining the columns that are categorical variables

categorical_cols = [col for col in X_train.columns if X_train[col].dtype == 'object']



# defining the column transformer with both the scaler and the encoder

preprocessor = ColumnTransformer(
```

```
    transformers=[

        ('num', StandardScaler(), [col for col in X_train.columns if col not in
categorical_cols]),

        ('cat', categorical_transformer, categorical_cols)

    ])



# fit and transform the training and testing data

X_train = preprocessor.fit_transform(X_train)

X_test = preprocessor.transform(X_test)
```

Model Training:

```
#Train various machine learning models

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural_network import MLPClassifier

import warnings

warnings.filterwarnings('ignore')



models = [("Logistic Regression", LogisticRegression(random_state=42, max_iter=1000,
solver='saga')),

("Decision Tree", DecisionTreeClassifier(random_state=42)),

("Random Forest", RandomForestClassifier(random_state=42)),

("Feed Forward Neural Network", MLPClassifier(random_state=42, max_iter=1000)),

("k-Nearest Neighbors", KNeighborsClassifier()),

("Naive Bayes", GaussianNB())]
```

```
#converting X_train and X_test from a sparse matrix to a dense matrix using numpy array before passing it to the models.

X_train = X_train.toarray()

X_test = X_test.toarray()



for name, model in models:

    with warnings.catch_warnings():

        warnings.simplefilter("ignore")

        model.fit(X_train, y_train)
```

Model Evaluation:

```
# Evaluate th performance of each model

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, precision_recall_curve, auc

results = {}

for name, model in models:

    y_pred = model.predict(X_test)

    results[name] = {

        'Accuracy': accuracy_score(y_test, y_pred),

        'Precision': precision_score(y_test, y_pred, pos_label=' >50K'),

        'Recall': recall_score(y_test, y_pred, pos_label=' >50K'),

        'F1-score': f1_score(y_test, y_pred, pos_label=' >50K')

    }

results_df = pd.DataFrame(results).T

print(results_df)
```

Visualization:

```
#visualisations comparation of the performance of different algorithms

# Bar Plot
```

```
import matplotlib.pyplot as plt


plt.bar(results_df.index, results_df['Accuracy'])

plt.xlabel('Algorithm')

plt.ylabel('Accuracy')

plt.title('Accuracy Comparison')

plt.xticks(rotation=45)

plt.axhline(y=results_df.loc['Logistic Regression', 'Accuracy'], color='r', linestyle='-
-', label='LR')

plt.legend()

plt.show()

#ROC curve

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import roc_curve


# convert categorical labels to binary labels

le = LabelEncoder()

y_test_binary = le.fit_transform(y_test)


plt.plot([0, 1], [0, 1], 'k--')

for name, model in models:

    fpr, tpr, thresholds = roc_curve(y_test_binary, model.predict_proba(X_test)[:, 1])

    plt.plot(fpr, tpr, label=name)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve Comparison')

plt.legend()
```

```
plt.show()

#Precision-Recall Curve

for name, model in models:

  precision, recall, thresholds = precision_recall_curve(y_test_binary,
model.predict_proba(X_test)[:,1])

  plt.plot(recall, precision, label=name)

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve Comparison')

plt.legend()

plt.show()

#Performing cross-validation

from sklearn.model_selection import cross_val_score

import numpy as np

for name, model in models:

    cv_scores = cross_val_score(model, X_train, y_train, cv=10)

    print("Model: ", name)

    print(f"Mean cross-validation score: {np.mean(cv_scores)}")

#Using hyperparameter tuning to optimize the performance of the best-performing model

from sklearn.model_selection import GridSearchCV



lr_params = {"C": np.logspace(-3, 3, 7)}

lr_grid_search = GridSearchCV(LogisticRegression(random_state=42, max_iter=1000,
solver='saga'), lr_params, cv=10)

lr_grid_search.fit(X_train, y_train)

best_lr_model = lr_grid_search.best_estimator_



print(f"Best model: {best_lr_model}")
```

```
#Evaluate the performance of the best-performing model

y_pred = best_lr_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred, pos_label=' >50K')

recall = recall_score(y_test, y_pred, pos_label=' >50K')

f1 = f1_score(y_test, y_pred, pos_label=' >50K')

print(f"Best model performance: {best_lr_model}")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1 Score: {f1}")

print("")

# Determine the threshold for identifying suspicious tax returns

best_lr_model.fit(X_train, y_train)

y_proba = best_lr_model.predict_proba(X_test)[:,1]

precision, recall, thresholds = precision_recall_curve(y_test, y_proba, pos_label='
>50K')

fpr, tpr, thresholds_roc = roc_curve(y_test, y_proba, pos_label=' >50K')

auc_score = auc(fpr, tpr)



# Plot the Precision-Recall curve and the ROC curve

import matplotlib.pyplot as plt

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(15,5))



# Plot the Precision-Recall curve

ax1.plot(recall, precision, color='orange', label=f"Area under PR curve:
{auc_score:.2f}")

ax1.set_xlabel('Recall')
```

```
ax1.set_ylabel('Precision')

ax1.set_title('Precision-Recall Curve')

ax1.legend()


# Plot the ROC curve

ax2.plot(fpr, tpr, color='green', label=f"Area under ROC curve: {auc_score:.2f}")

ax2.set_xlabel('False Positive Rate')

ax2.set_ylabel('True Positive Rate')

ax2.set_title('ROC Curve')

ax2.legend()

# Set the threshold for identifying suspicious tax returns

threshold = 0.35

print(f"Threshold for identifying suspicious tax returns: {threshold:.2f}")


# Identify suspicious tax returns

best_lr_model.fit(X_train, y_train)

y_proba = best_lr_model.predict_proba(X_test)[:,1]

y_pred = (y_proba >= threshold).astype(int) #creates a binary classification based on
the threshold

fraudulent_cases = X_test[y_pred==1, :10] # select only the columns used for prediction

fraudulent_cases_df = pd.DataFrame(fraudulent_cases, columns=X.columns[:10])

print(f"Number of suspicious tax returns identified: {len(fraudulent_cases)}")
```

Model Comparison:

```
#Compare the performance of each model

names = []

accuracies = []

precisions = []
```

```
recalls = []

f1s = []


for name, model in models:

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    precision = precision_score(y_test, y_pred, pos_label=' >50K')

    recall = recall_score(y_test, y_pred, pos_label=' >50K')

    f1 = f1_score(y_test, y_pred, pos_label=' >50K')

    names.append(name)

    accuracies.append(accuracy)

    precisions.append(precision)

    recalls.append(recall)

    f1s.append(f1)

    results_df = pd.DataFrame({"Model": names, "Accuracy": accuracies, "Precision": 
precisions, "Recall": recalls, "F1 Score": f1s})

print(results_df)
```

Model Deployment:

```
# Import the required libraries

import tensorflow as tf

from tensorflow import keras

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

import numpy as np

import os

#Define input data to test the TensorFlow Lite model.
```

```
input_data = np.array([[0.123, -0.456, -1.234, 0.789, -2.345]])


#Assuming Declaredincome and Actualincome are categorical variables with two possible
values.

Declaredincome = '<=50K' # example value

Actualincome = '>50K' # example value

if Declaredincome == '<=50K':

    input_data = np.hstack((input_data, [[1, 0]]))

else:

    input_data = np.hstack((input_data, [[0, 1]]))

if Actualincome == '<=50K':

    input_data = np.hstack((input_data, [[1, 0]]))

else:

    input_data = np.hstack((input_data, [[0, 1]]))


#Convert input data to FLOAT32

input_data = input_data.astype(np.float32)


#Reshape input data to the correct shape

input_shape = (1, 5)

input_data = np.reshape(input_data[:, :5], input_shape)


#Convert scikit-learn model to TensorFlow model

input_shape = (5,)

model_input = keras.layers.Input(shape=input_shape, name="input")

model_output = keras.layers.Dense(1, activation="sigmoid", name="output")(model_input)

tf_model = keras.Model(inputs=model_input, outputs=model_output)
```

```
#Set the weights of the TensorFlow model
tf_model.layers[1].set_weights([best_lr_model.coef_[:, :5].T, best_lr_model.intercept_])

#Convert TensorFlow model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(tf_model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

#Save the TFLite model to a file on desktop
file_path = os.path.join(os.path.expanduser('~'), 'Desktop', 'TFmodel.tflite')
tf.io.write_file(file_path, tflite_model)

#Save the TFLite model to a file
file_path = "best_lr_model.tflite"
with open(file_path, "wb") as f:
    f.write(tflite_model)

#create an interpreter for the TensorFlow Lite model
interpreter = tf.lite.Interpreter(model_path=file_path)
interpreter.allocate_tensors()

#Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

#Set input data
```

```
interpreter.set_tensor(input_details[0]['index'], input_data)


#Run inference

interpreter.invoke()


#Get output and reshape to the correct shape

output_data = interpreter.get_tensor(output_details[0]['index'])

output_data = np.reshape(output_data, (1, 1))


#Testing the tensor flow model

if output_data[0][0] > 0.5:

    result = '1'

else:

    result = '0'

print(result)
```

5.2.2 Mobile Application Code:

Main xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/firstbackground"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/name"
```

```
        android:textSize="20sp"
        android:textStyle="normal"
        android:textColor="@color/purple_700"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.467"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.098" />

    <EditText
        android:id="@+id/ed1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:autofillHints=""
        android:ems="10"
        android:hint="@string/hint_name"
        android:inputType="textPersonName"
        android:textColor="@color/white"
        android:textColorHint="#FFFFFF"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        tools:ignore="TextContrastCheck" />

    <EditText
        android:id="@+id/ed2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:autofillHints=""
        android:ems="10"
        android:hint="@string/hint_password"
        android:inputType="textPassword"
        android:textColor="@color/white"
        android:textColorHint="#FFFFFF"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="@+id/ed1"
        app:layout_constraintTop_toBottomOf="@+id/ed1"
        tools:ignore="TextContrastCheck" />

    <Button
        android:id="@+id/btn"
        android:onClick="Login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text="@string/hint_btn1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ed2" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Main java file:

```java
package com.example.itfd;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText ed1, ed2;
    String username,password;
    SharedPreferences sharedPreferences;
    SharedPreferences.Editor editor;
    private static final String TAG = "MainActivity";
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ed1 = findViewById(R.id.ed1);
        ed2 = findViewById(R.id.ed2);
        sharedPreferences = getSharedPreferences("mypref",MODE_PRIVATE);
        editor = sharedPreferences.edit();
    }

    public void Login(View view) {
        try {
            username = ed1.getText().toString();
            password = ed2.getText().toString();
            if(username.equals("rafeeda") && password.equals("rafeeda")){
                editor.putString("name",username);
                editor.putString("password",password);
                editor.commit();
                Intent intent = new Intent(this,PredictionActivity.class);
                startActivity(intent);
            }
            else{
                Toast.makeText(this,"Login Unsuccessful",Toast.LENGTH_LONG).show();
            }

        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
            Toast.makeText(this, "Error: " + e.getMessage(), Toast.LENGTH_SHORT).show();
        }

    }
}
```

Prediction xml file:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:orientation="vertical"
android:background="@drawable/secondbackground"
tools:context=".PredictionActivity">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text"
    android:textSize="15sp"
    android:textStyle="italic"
    android:textColor="@color/purple_700"
    android:gravity="center"
    android:textAlignment="center" />

<EditText
    android:id="@+id/ed_1"
    android:layout_marginTop="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:autofillHints=""
    android:gravity="center"
    android:hint="@string/edit_1"
    android:inputType="textPersonName"
    android:minHeight="15dp"
    android:textColor="@color/black"
    android:textColorHint="#747679"
    android:textSize="15sp" />

<AutoCompleteTextView
    android:id="@+id/auto_1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:hint="@string/auto_1"
    android:inputType="textPersonName"
    android:minHeight="15dp"
    android:textColor="@color/black"
    android:textColorHint="#747679"
    android:textSize="15sp" />

<AutoCompleteTextView
    android:id="@+id/auto_2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:hint="@string/auto_2"
    android:inputType="textPersonName"
    android:minHeight="15dp"
    android:textColor="@color/black"
    android:textColorHint="#747679"
    android:textSize="15sp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text1"
    android:textSize="12sp"
    android:textStyle="italic"
```

```
    android:textColor="@color/black" />
<Spinner
    android:id="@+id/spinner1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:contentDescription="@string/spinner_1"
    android:gravity="center"
    android:minHeight="15dp"
    android:inputType="textPersonName"
    android:textColor="@color/black"
    android:textSize="15sp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text2"
    android:textSize="12sp"
    android:textStyle="italic"
    android:textColor="@color/black" />
<Spinner
    android:id="@+id/spinner2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:contentDescription="@string/spinner_2"
    android:gravity="center"
    android:minHeight="15dp"
    android:inputType="textPersonName"
    android:textColor="@color/black"
    android:textSize="15sp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text3"
    android:textSize="12sp"
    android:textStyle="italic"
    android:textColor="@color/black" />
<Spinner
    android:id="@+id/spinner3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:contentDescription="@string/spinner_3"
    android:gravity="center"
    android:minHeight="15dp"
    android:inputType="textPersonName"
    android:textColor="@color/black"
    android:textSize="15sp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text4"
    android:textSize="12sp"
    android:textStyle="italic"
    android:textColor="@color/black" />
<Spinner
    android:id="@+id/spinner4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:contentDescription="@string/spinner_4"
    android:gravity="center"
    android:minHeight="15dp"
    android:inputType="textPersonName"
    android:textColor="@color/black"
    android:textSize="15sp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text5"
    android:textSize="12sp"
    android:textStyle="italic"
    android:textColor="@color/black" />

<RadioGroup
    android:id="@+id/rg"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <RadioButton
        android:id="@+id/rb1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="15dp"
        android:text="@string/male" />

    <RadioButton
        android:id="@+id/rb2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="15dp"
        android:text="@string/female" />
</RadioGroup>

<EditText
    android:id="@+id/ed_2"
    android:layout_marginTop="5dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:autofillHints=""
    android:gravity="center"
    android:hint="@string/edit_2"
    android:inputType="textPersonName"
    android:minHeight="15dp"
    android:textColor="@color/black"
    android:textColorHint="#747679"
    android:textSize="15sp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/text6"
    android:textSize="12sp"
    android:textStyle="italic"
    android:textColor="@color/black" />
<Spinner
    android:id="@+id/spinner5"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:contentDescription="@string/spinner_5"
        android:gravity="center"
        android:inputType="textPersonName"
        android:minHeight="15dp"
        android:textColor="@color/black"
        android:textSize="15sp"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/text7"
        android:textSize="12sp"
        android:textStyle="italic"
        android:textColor="@color/black" />
    <Spinner
        android:id="@+id/spinner6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:contentDescription="@string/spinner_6"
        android:gravity="center"
        android:minHeight="15dp"
        android:inputType="textPersonName"
        android:textColor="@color/black"
        android:textSize="15sp" />

    <Button
        android:id="@+id/btn1"
        android:layout_marginTop="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:minHeight="20dp"
        android:onClick="Predict"
        android:text="@string/hint_btn2" />
    <Button
        android:id="@+id/btn2"
        android:layout_marginTop="5dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:minHeight="20dp"
        android:onClick="Back"
        android:text="@string/hint_btn3" />
</LinearLayout>
```

Prediction java file:

```
package com.example.itfd;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
```

```
import android.widget.AutoCompleteTextView;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.Toast;
public class PredictionActivity extends AppCompatActivity {
    private EditText age, hourPerWeek;
    private AutoCompleteTextView education, workClass;
    private Spinner occupation, maritalStatus, race, nativeCountry, declaredIncome,
actualIncome;
    RadioGroup radioGroup;
    RadioButton male, female;
    String gender;
    private static final String TAG = "PredictionActivity";
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prediction);
        // Initialize the spinners and the button
        age = findViewById(R.id.ed_1);
        hourPerWeek = findViewById(R.id.ed_2);
        education = findViewById(R.id.auto_1);
        workClass = findViewById(R.id.auto_2);
        occupation = findViewById(R.id.spinner1);
        maritalStatus = findViewById(R.id.spinner2);
        race = findViewById(R.id.spinner3);
        nativeCountry = findViewById(R.id.spinner4);
        declaredIncome = findViewById(R.id.spinner5);
        actualIncome = findViewById(R.id.spinner6);
        radioGroup = findViewById(R.id.rg);
        male = findViewById(R.id.rb1);
        female = findViewById(R.id.rb2);

        // Set the options for autoCompleteTextView and Spinners
        ArrayAdapter<CharSequence> educationAdapter =
ArrayAdapter.createFromResource(this, R.array.Education,
android.R.layout.simple_spinner_item);

educationAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        education.setAdapter(educationAdapter);
        ArrayAdapter<CharSequence> workclassAdapter =
ArrayAdapter.createFromResource(this, R.array.WorkClass,
android.R.layout.simple_spinner_item);

workclassAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        workClass.setAdapter(workclassAdapter);
        ArrayAdapter<CharSequence> occupationAdapter =
ArrayAdapter.createFromResource(this, R.array.Occupation,
android.R.layout.simple_spinner_item);

occupationAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
;
        occupation.setAdapter(occupationAdapter);
        ArrayAdapter<CharSequence> maritalstatusAdapter =
ArrayAdapter.createFromResource(this, R.array.MaritalStatus,
android.R.layout.simple_spinner_item);
```

```
maritalstatusAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
        maritalStatus.setAdapter(maritalstatusAdapter);
        ArrayAdapter<CharSequence> raceAdapter = ArrayAdapter.createFromResource(this,
R.array.Race, android.R.layout.simple_spinner_item);

raceAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        race.setAdapter(raceAdapter);
        ArrayAdapter<CharSequence> nativecountryAdapter =
ArrayAdapter.createFromResource(this, R.array.NativeCountry,
android.R.layout.simple_spinner_item);

nativecountryAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
        nativeCountry.setAdapter(nativecountryAdapter);
        ArrayAdapter<CharSequence> declaredincomeAdapter =
ArrayAdapter.createFromResource(this, R.array.DeclaredIncome,
android.R.layout.simple_spinner_item);

declaredincomeAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        declaredIncome.setAdapter(declaredincomeAdapter);
        ArrayAdapter<CharSequence> actualincomeAdapter =
ArrayAdapter.createFromResource(this, R.array.ActualIncome,
android.R.layout.simple_spinner_item);

actualincomeAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_ite
m);
        actualIncome.setAdapter(actualincomeAdapter);
    }
    public void Predict(View view) {
        // Get the values selected in the spinners
        int Age = Integer.parseInt(age.getText().toString());
        String Workclass = workClass.getText().toString();
        String Education = education.getText().toString();
        String Maritalstatus = maritalStatus.getSelectedItem().toString();
        String Occupation = occupation.getSelectedItem().toString();
        String RACE = race.getSelectedItem().toString();
        String Nativecountry = nativeCountry.getSelectedItem().toString();
        int Hoursperweek = Integer.parseInt(hourPerWeek.getText().toString());
        String Declaredincome = declaredIncome.getSelectedItem().toString();
        String Actualincome = actualIncome.getSelectedItem().toString();

        if(male.isChecked()){
            gender = male.getText().toString();
        }
        else if(female.isChecked()){
            gender = female.getText().toString();
        }

        // Preprocess the input values
        float[] input = preprocessInput(Age, Workclass, Education, Maritalstatus,
Occupation, RACE, gender, Hoursperweek,   Nativecountry, Declaredincome, Actualincome);

        try {
            float[] output = new Classifier(getAssets(), "best_lr_model.tflite", new
int[]{1, 11}, new int[]{1, 1}).classify(input);
            // Get the predicted income class
            int predictedClass;
```

```
            if (Declaredincome.equals("<=50K") && Actualincome.equals(">50K")) {
                predictedClass = output[0] > 0.5 ? 1 : 0;
            } else {
                predictedClass = output[0] <= 0.5 ? 1 : 0;
            }

            // Pass the predicted class value to the DisplayActivity
            Intent intent = new Intent(this, DisplayActivity.class);
            intent.putExtra("predictedClass", predictedClass);
            startActivity(intent);
        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
            Toast.makeText(this, "Error: " + e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }

    private float[] preprocessInput(int Age, String Workclass, String Education, String
Martialstatus, String Occupation, String RACE, String gender, int Hoursperweek, String
Nativecountry, String Declaredincome, String Actualincome) {
        // Perform any necessary preprocessing on the input values
        // Here, we will convert the categorical variables to one-hot encoded vectors


        float[] input = new float[29];
        input[0] = Age;
        input[1] = Workclass.equals("Private") ? 1 : 0;
        input[2] = Workclass.equals("Self-emp-not-inc") ? 1 : 0;
        input[3] = Workclass.equals("Local-gov") ? 1 : 0;
        input[4] = Workclass.equals("Federal-gov") ? 1 : 0;
        input[5] = Education.equals("Bachelors") ? 1 : 0;
        input[6] = Education.equals("Masters") ? 1 : 0;
        input[7] = Education.equals("Doctorate") ? 1 : 0;
        input[8] = Education.equals("HS-grad") ? 1 : 0;
        input[9] = Martialstatus.equals("Never-married") ? 1 : 0;
        input[10] = Martialstatus.equals("Married-civ-spouse") ? 1 : 0;
        input[11] = Martialstatus.equals("Divorced") ? 1 : 0;
        input[12] = Martialstatus.equals("Separated") ? 1 : 0;
        input[13] = Occupation.equals("Exec-managerial") ? 1 : 0;
        input[14] = Occupation.equals("Tech-support") ? 1 : 0;
        input[15] = Occupation.equals("Sales") ? 1 : 0;
        input[16] = Occupation.equals("Other-service") ? 1 : 0;
        input[17] = RACE.equals("Black") ? 1 : 0;
        input[18] = RACE.equals("White") ? 1 : 0;
        input[19] = RACE.equals("Asian-Pac-Islander") ? 1 : 0;
        input[20] = RACE.equals("Amer-Indian-Eskimo") ? 1 : 0;
        input[21] = gender.equals("Male") ? 1 : 0;
        input[22] = Hoursperweek;
        input[23] = Nativecountry.equals("United-States") ? 1 : 0;
        input[24] = Nativecountry.equals("Japan") ? 1 : 0;
        input[25] = Nativecountry.equals("India") ? 1 : 0;
        input[26] = Nativecountry.equals("Germany") ? 1 : 0;
        input[27] = Declaredincome.equals("<=50K") ? 1 : 0;
        input[28] = Actualincome.equals(">50K") ? 1 : 0;
        return input;
    }

    public void Back(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
```

```
    }
}
```

Classifier java file:

```
package com.example.itfd;

import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.util.Log;

import org.tensorflow.lite.Interpreter;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;

@SuppressWarnings("ALL")
public class Classifier {
    private final Interpreter interpreter; //object of the interpreter class
    private final int[] inputShape; //shape of the input tensors
    private final int[] outputShape; //shape of the output tensors



    // Constructor for the Classifier class that loads the ML model and sets its input
and output shapes
    public Classifier(AssetManager assetManager, String modelPath, int[] inputShape,
int[] outputShape) {
        Interpreter.Options options = new Interpreter.Options();
        options.setNumThreads(4);
        interpreter = new Interpreter(loadModelFile(assetManager, modelPath), options);
        this.inputShape = inputShape;
        this.outputShape = outputShape;
        interpreter.resizeInput(0, inputShape); //resize the input tensor to match the
input shape
    }

    // Method for running the loaded ML model on an input and returning the output
    public float[] classify(float[] input) {
        float[][] inputArr = new float[1][inputShape[0]];
        inputArr[0] = input;
        float[][] output = new float[2][outputShape[0]];
        interpreter.run(inputArr, output); // runs the loaded ML model on the input data
        return output[0];
    }
    // Method for loading the model file from assets and returning it to  loadModelFile
    private MappedByteBuffer loadModelFile(AssetManager assetManager, String modelPath)
{
        try {
            AssetFileDescriptor fileDescriptor = assetManager.openFd(modelPath); // Open
an AssetFileDescriptor for the model file
            FileInputStream inputStream = new
FileInputStream(fileDescriptor.getFileDescriptor());
            FileChannel fileChannel = inputStream.getChannel();
            long startOffset = fileDescriptor.getStartOffset();
            long declaredLength = fileDescriptor.getDeclaredLength();
            return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
```

```
declaredLength); // Map the file channel to object and return it
        } catch (IOException e) {
            Log.e("Classifier", "Error loading model file: " + e.getMessage());
            return null;
        }
    }
}
```

Display xml file:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_gravity="center"
    android:background="@drawable/secondbackground"
    tools:context=".DisplayActivity">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center"
        android:layout_marginRight="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="150dp"
        android:text="@string/textview"
        android:textColor="@color/white"
        android:textSize="30sp"
        android:textStyle="italic"
        tools:ignore="TextContrastCheck" />
    <Button
        android:id="@+id/btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"
        android:minHeight="30dp"
        android:onClick="Back"
        android:text="@string/hint_btn3"
        tools:ignore="TouchTargetSizeCheck" />

</LinearLayout>
```

Display java file:

```java
package com.example.itfd;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
```

```
public class DisplayActivity extends AppCompatActivity {
    @SuppressLint({"MissingInflatedId", "SetTextI18n"})
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display);
        // Get the predictedClass value from the intent
        int predictedClass = getIntent().getIntExtra("predictedClass",-1);

        // Set the text of a TextView based on the predictedClass value
        TextView predictedClassTextView = findViewById(R.id.textview);
        if (predictedClass == 1) {
            predictedClassTextView.setText("Suspicious of committing Income tax fraud");
        } else {
            predictedClassTextView.setText("Not suspicious of committing Income tax
fraud");
        }
    }

    public void Back(View view) {
        Intent intent = new Intent(this, PredictionActivity.class);
        startActivity(intent);
    }
}
```

String xml file:

```
<resources>
    <string name="app_name">ITFD</string>
    <string name="name">INCOME TAX FRAUD DETECTION</string>
    <string name="hint_name">UserName</string>
    <string name="hint_password">Password</string>
    <string name="hint_btn1">SUBMIT</string>

    <string name="text">SELECT THE INPUTS</string>
    <string name="text1">Select your occupation:</string>
    <string name="text2">Choose your MaritalStatus:</string>
    <string name="text3">Select your Race:</string>
    <string name="text4">Select your NativeCountry:</string>
    <string name="text5">Choose your gender:</string>
    <string name="text6">Select your Declared Income range:</string>
    <string name="text7">Select your Actual Income range:</string>

    <string name="edit_1">Enter your age</string>
    <string name="edit_2">Enter your HourPerWeek</string>
    <string name="auto_1">Start typing your Education</string>
    <string name="auto_2">Start typing your WorkClass</string>
    <string name="spinner_1">Occupation</string>
    <string name="spinner_2">MaritalStatus</string>
    <string name="spinner_3">Race</string>
    <string name="spinner_4">NativeCountry</string>
    <string name="spinner_5">DeclaredIncome</string>
    <string name="spinner_6">ActualIncome</string>
    <string name="hint_btn2">PREDICT</string>
    <string name="textview">TextView</string>
    <string name="male">male</string>
    <string name="female">female</string>
```

```xml
    <string name="hint_btn3">BACK</string>

    <string-array name="Education">
        <item>Bachelors</item>
        <item>Masters</item>
        <item>Doctorate</item>
        <item>HS-grad</item>
    </string-array>
    <string-array name="WorkClass">
        <item>Private</item>
        <item>Self-emp-not-inc</item>
        <item>Local-gov</item>
        <item>Federal-gov</item>
    </string-array>
    <string-array name="Occupation">
        <item>Exec-managerial</item>
        <item>Tech-support</item>
        <item>Sales</item>
        <item>Other-service</item>
    </string-array>
    <string-array name="MaritalStatus">
        <item>Never-married</item>
        <item>Married-civ-spouse</item>
        <item>Divorced</item>
        <item>Separated</item>
    </string-array>
    <string-array name="Race">
        <item>Black</item>
        <item>White</item>
        <item>Asian-Pac-Islander</item>
        <item>Amer-Indian-Eskimo</item>
    </string-array>
    <string-array name="NativeCountry">
        <item>United-States</item>
        <item>Japan</item>
        <item>India</item>
        <item>Germany</item>
    </string-array>
    <string-array name="DeclaredIncome">
        <item>&gt;50K</item>
        <item>&lt;=50K</item>
    </string-array>
    <string-array name="ActualIncome">
        <item>&gt;50K</item>
        <item>&lt;=50K</item>
    </string-array>

</resources>
```

Chapter-6

TIMELINE FOR EXECUTION OF PROJECT

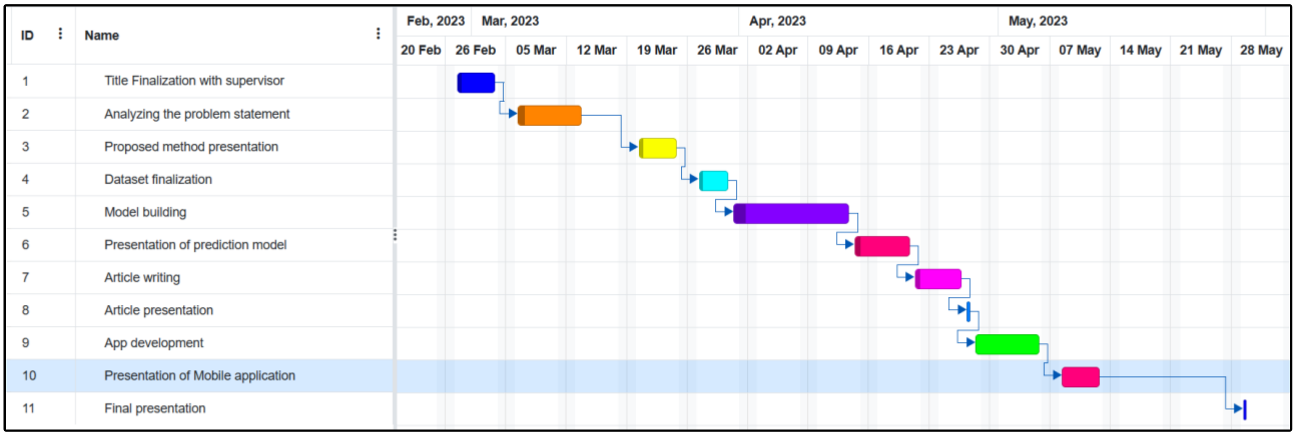

Fig-4 Timeline

Chapter-7

OUTCOMES

- A Machine Learning model that predicts whether a person is likely to hide their income and evade taxes based on a variety of factors such as their expenses, assets, income sources, and many others
- The model detects income tax fraud with better accuracy than the existing model, reducing financial losses associated with fraudulent activities.
- The logistic regression model with optimized hyperparameters was found to be the best-performing model.
- The created application compares the declared income and the actual income, then uses the deployed model to make predictions and shows its results

Chapter-8

RESULTS AND DISCUSSIONS

This section provides a detailed explanation of the results obtained from the computations for detecting income tax fraud. The data used in this project was obtained from OpenML, an open platform for sharing datasets. The dataset contained 48842 entries with 15 columns. Exploratory data analysis was performed to gain insights into the dataset, such as checking the correlation between variables and visualizing the distributions of variables. To achieve this, a scatterplot matrix with 6 rows and 6 columns was generated, displaying the relationship between two variables in each scatterplot. The plots on the diagonal displayed the distribution of each variable, while the plots above the diagonal were mirrored from those below the diagonal. The hue parameter was used to color the data points according to the income level, making it easier to identify any patterns or differences between the two classes. The resulting diagram is shown below.

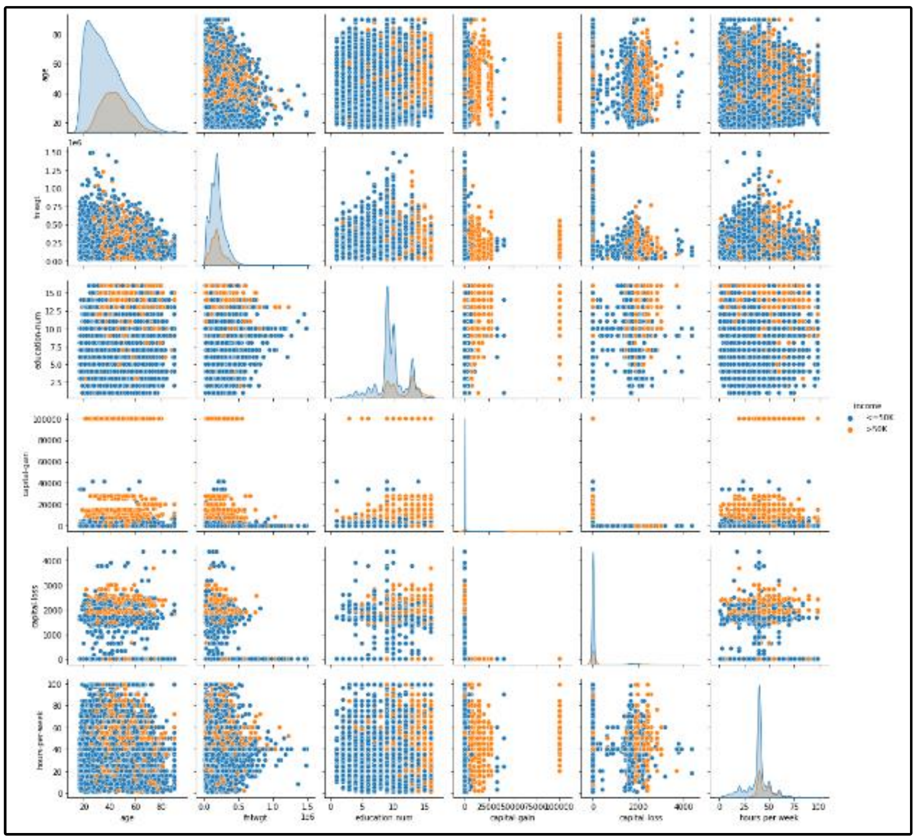

Fig-5 exploratory data analysis

After pre-processing the dataset to remove duplicates, handle missing values, encode categorical variables, and scale continuous variables, six machine learning models were trained on the training data. The performance of the models was evaluated using relevant assessment measures such as accuracy, precision, recall, and F1-score. To compare the performance of the algorithms, the following visualizations were created:

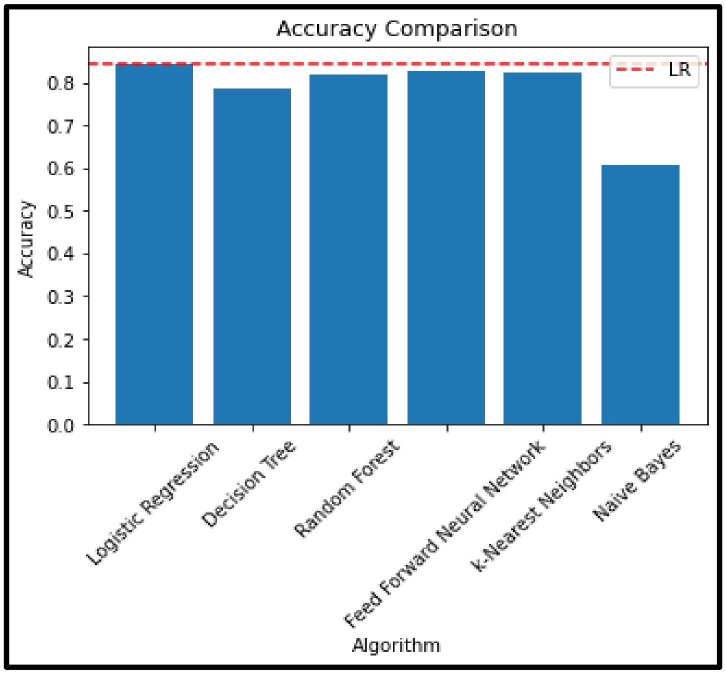

Fig-6 Accuracy bar plot

Bar Plot: A comparison of the accuracy of each model was made. The accuracy of the logistic regression model was represented by a red horizontal line, which served as a reference point for comparison.

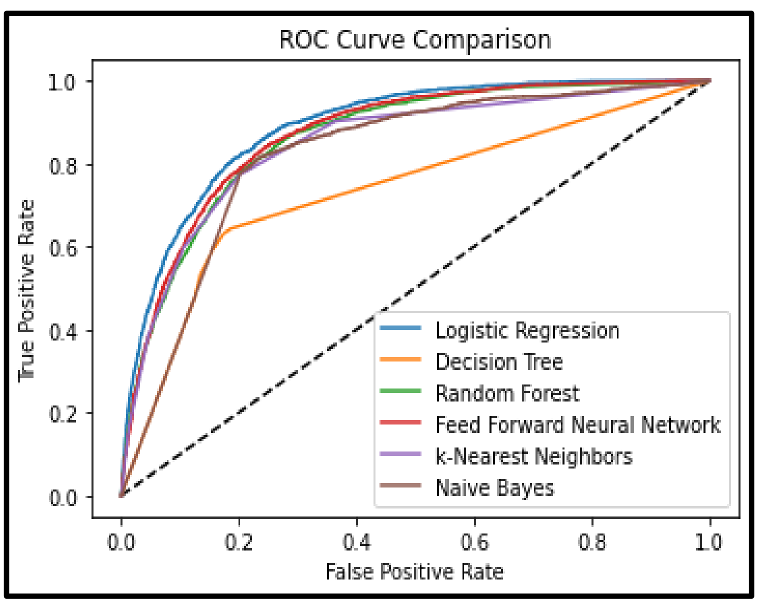

Fig-7 ROC curve

A plot of the **Receiver Operating Characteristic (ROC) Curve** was made for each algorithm using different colors for each model. This displays the tradeoff between sensitivity and specificity for each method, allowing them to be compared.

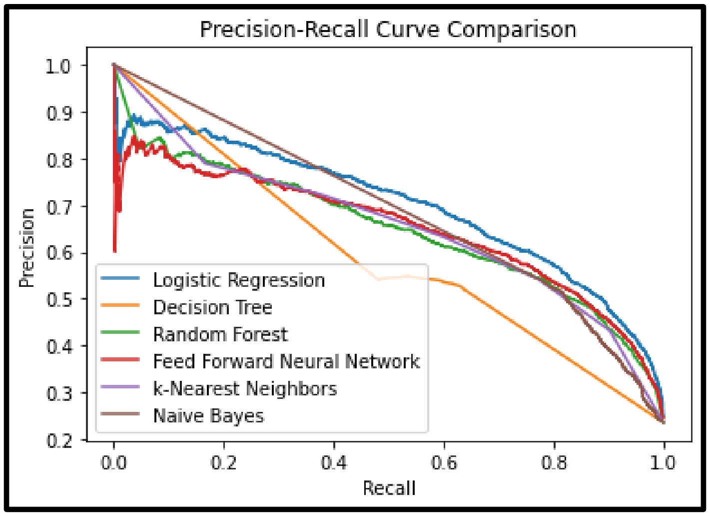

Fig-8 Precision-Recall Curve

Precision-Recall Curve: A plot of the precision-recall curves was made for each algorithm. This graph depicts the link between precision and recall for each method.

After comparing the performance of various algorithms, cross-validation was performed to reduce the impact of chance and sampling bias that may occur in a single train-test split. The output displayed the name of each model and its mean cross-validation score over 10 folds, which provides a measure of the generalization performance of each model. This indicates how well each model is likely to perform on new and unseen data.

TABLE I. CROSS VALIDATION TABLE

Models	Mean cross –validation score
Logistic Regression	0.8346172945234638
Decision Tree	0.7825865720373295
Random Forest	0.817162880366683
Naïve Bayes	0.605046495559648
K-Nearest Neighbors	0.8182374679046269
Feed Forword Neural Network	0.81933381089868617

Next, we optimize the performance of the best-performing model with hyperparameters using the GridSearchCV method from scikit-learn. The hyperparameters being tuned are the regularization strength parameter "C" of the logistic regression algorithm. After the hyperparameters are tuned, the best estimator model is selected based on the average score obtained across all folds of the cross-validation. The logistic regression model's prediction probabilities were used to identify suspicious fraudsters based on a threshold value of 0.35, and 2769 records were recognized as suspicious of committing income tax fraud.

Finally, the performance of each model is compared using evaluation metrics.

TABLE II. MODEL COMPARISON TABLE

Models	Evaluation metrics			
	Accuracy	Precision	Recall	FI Score
Logistic Regression	**0.832973**	**0.703883**	**0.569869**	**0.629826**
Decision Tree	**0.787184**	**0.547077**	**0.535371**	**0.541161**
Random Forest	**0.820248**	**0.629990**	**0.565066**	**0.595764**
Naïve Bayes	**0.608660**	**0.366626**	**0.920087**	**0.524325**
K-Nearest Neighbours	**0.824445**	**0.632917**	**0.597817**	**0.614866**
Feed Forword Neural Network	**0.825366**	**0.627178**	**0.628821**	**0.627998**

After completing the analysis, the best performing model, which is logistic regression model was converted into a TensorFlow lite format and deployed into an android studio to build a prediction app using the same inputs as the dataset used in ML models building.

In this study, an application that can run on a mobile phone is designed. Android and iOS are the two most widely used mobile operating system platforms (OS). The Android platform powers the majority of low-cost mobile smartphones suited for rural use. As a result, we picked the Android operating system as our framework for developing an income tax fraud detection application. Fig. 8 shows the Login Screen, which is the first interface of our app. The users must provide their name and password to enter the second interface.

Fig-9 Login Screen

The input screen is shown in Fig. 9, and it requests the user to enter 11 inputs, including their age, education, work class, occupation, marital status, race, gender, native country, hour per week, declared and actual income by using the edit text, auto complete text view, radio button and spinner controls. The app controller will then analyze the inputs and provide a predicted result.

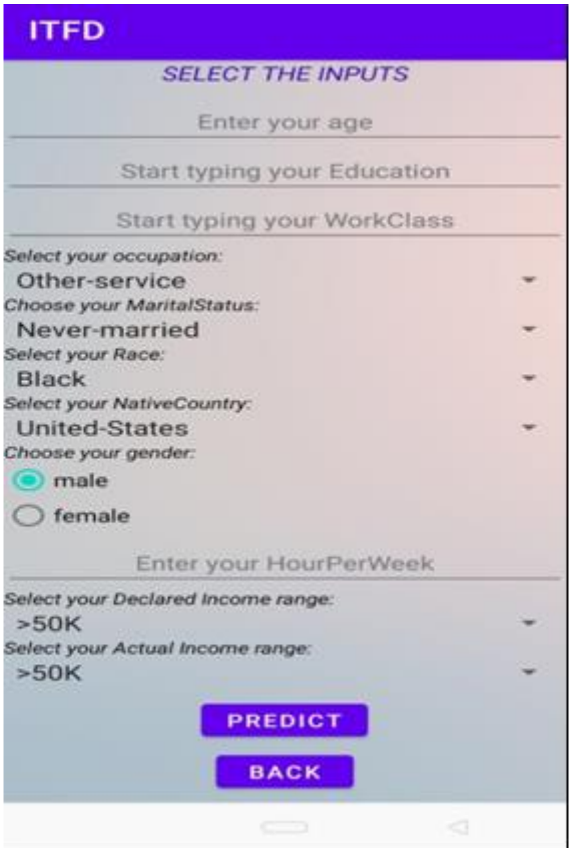

Fig-10 Input Screen before selecting the inputs.

Fig-11 Input Screen after selecting the inputs.

The third interface displayed in Fig-11 retrieves the predicted outcome from the inputs interface and displays it in text view.

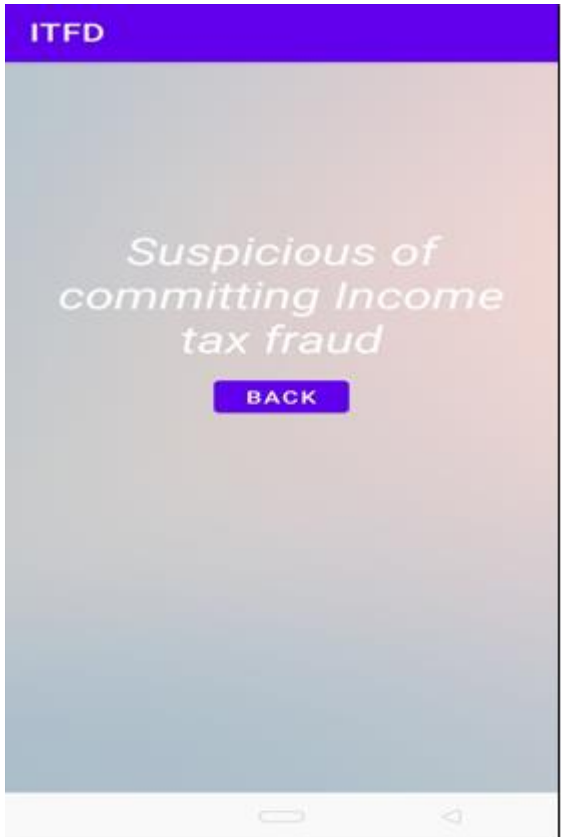

Fig-12 Display Screen

Chapter-9

CONCLUSION

The best-performing model among those considered was the logistic regression model with hyperparameters optimized using grid search. This model achieved an accuracy of 0.8429, precision of 0.7038, recall of 0.5698, and F1-score of 0.5698. The ROC and Precision-Recall curves demonstrate that the logistic regression model has the best trade-off between true positive rate and false positive rate, as well as precision and recall, compared to the other models. The results of cross-validation indicate that the performance of the models is consistent across folds, with the logistic regression model achieving the highest mean cross-validation score. In conclusion, our project provides a framework for building and comparing various machine learning models for detecting income tax fraud. The logistic regression model with optimized hyperparameters was found to be the best-performing model. The created application compares the declared income and the actual income, then uses the deployed model to make predictions and shows its results. As a result, we have developed an accurate and efficient way to find a person suspicious of committing income tax fraud, which can aid in financial planning, market research, and other areas.

. REFERENCES

[1]. Murorunkwere, B.F.; Tuyishimire, O.; Haughton, D.; Nzabanita, J. Fraud Detection Using Neural Networks: A Case Study of Income Tax. Future Internet 2022, 14, 168.

[2]. Pérez López, C.; Delgado Rodríguez, M.J.; de Lucas Santos, S. Tax Fraud Detection through Neural Networks: An Application Using a Sample of Personal Income Taxpayers. Future Internet 2019, 11, 86.

[3]. M. S. Rad and A. Shahbahrami, Detecting high risk taxpayers using data mining techniques, 2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS), Tehran, Iran, 2016, pp. 1-5.

[4]. Mojahedi, Houri & Babazadeh sangar, Amin & Masdari, Mohammad. (2022). Towards Tax Evasion Detection Using Improved Particle Swarm Optimization Algorithm. Mathematical Problems in Engineering. 2022. 1-17.

[5]. de Roux, D.; Perez, B.; Moreno, A.; Villamil, M.D.P.; Figueroa, C. Tax fraud detection for under-reporting declarations using an unsupervised machine learning approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 215–222

[6]. Miloš Savić, Jasna Atanasijević, Dušan Jakovetić, Nataša Krejić, Tax evasion risk management using a Hybrid Unsupervised Outlier Detection method, Expert Systems with Applications, Volume 193, 2022, 116409, ISSN 0957-4174.

[7]. González, P.C.; Velásquez, J.D. Characterization and detection of taxpayers with false invoices using data mining techniques. Expert Syst. Appl. 2013, 40, 1427–1436.

[8]. *Ghosh*, S.; Douglas, L.R. Credit card fraud detection with a neural-network. In Proceedings of the Twenty-Seventh Hawaii International Conference, Wailea, HI, USA, 4–7 January 1994.

[9]. Chi-Hung Lin, I-Chun Lin, Ching-Huei Wu, Ya-Ching Yang & Jinsheng Roan (2012) The application of decision tree and artificial neural network to income tax audit: the examples of profit-seeking enterprise income tax and individual income tax in Taiwan, Journal of the Chinese Institute of Engineers, 35:4, 401-41.

[10]. B. Baesens, V. Vlasselaer, W. Verbeke. “Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques”. Wiley, 2015.

[11]. Kidong Lee; David Booth; Pervaiz Alam (2005). *A comparison of supervised*

and unsupervised neural networks in predicting bankruptcy of Korean firms. , 29(1), 1–16.

[12]. Clavería Navarrete, A. y Carrasco Gallego, A. (2021). Neural network algorithms for fraud detection: a comparison of the complementary techniques in the last five years. Journal of Management Information and Decision Sciences, 24 (special 1), 1-16.

[13]. López, César & Delgado Rodríguez, María Jesús & Santos, Sonia. (2019). Tax Fraud Detection through Neural Networks: An Application Using a Sample of Personal Income Taxpayers.

Uncovering Income Tax Fraud: A Logistic Regression Approach for Detection and Prevention

ORIGINALITY REPORT

21%	12%	7%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Milwaukee School of Engineering Student Paper	4%
2	vbook.pub Internet Source	3%
3	Submitted to University of Derby Student Paper	3%
4	Hansi Gunasinghe, James McKelvie, Abigail Koay, Michael Mayo. "Comparison Of Pretrained Feature Extractors For Glaucoma Detection", 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), 2021 Publication	2%
5	Submitted to Middle East College of Information Technology Student Paper	2%
6	www.researchgate.net Internet Source	<1%

7	Submitted to Aston University Student Paper	<1 %
8	www.abacademies.org Internet Source	<1 %
9	Submitted to University of Sunderland Student Paper	<1 %
10	usermanual.wiki Internet Source	<1 %
11	Submitted to University of Salford Student Paper	<1 %
12	www.mdpi.com Internet Source	<1 %
13	yourstory.com Internet Source	<1 %
14	Submitted to University of Sydney Student Paper	<1 %
15	Submitted to Bilkent University Student Paper	<1 %
16	Zulfikar Zulfikar, Zulhelmi Zulhelmi, Teuku Yuliar Arif, Afdhal Afdhal, Putra Nasri Syawaldi. "Android Application: Skin Abnormality Analysis based on Edge Detection Technique", 2018 International Conference on Electrical Engineering and Informatics (ICELTICs)(44501), 2018	<1 %

Publication

17	core.ac.uk Internet Source	<1 %
18	G. Lohmann. "Co-occurrence-based analysis and synthesis of textures", Proceedings of 12th International Conference on Pattern Recognition ICPR-94, 1994 Publication	<1 %
19	Submitted to Chester College of Higher Education Student Paper	<1 %
20	Submitted to Universidad Politécnica de Madrid Student Paper	<1 %
21	Submitted to University of Edinburgh Student Paper	<1 %
22	tel.archives-ouvertes.fr Internet Source	<1 %
23	www.researchsquare.com Internet Source	<1 %
24	github.com Internet Source	<1 %
25	Simegnew Alaba. "Image Classification using Different Machine Learning Techniques", Institute of Electrical and Electronics Engineers (IEEE), 2023	<1 %

Publication

26	repositorio.ulima.edu.pe Internet Source	<1 %
27	web.archive.org Internet Source	<1 %
28	M. Heiser, C. Scheidl, J. Eisl, B. Spangl, J. Hübl. "Process type identification in torrential catchments in the eastern Alps", Geomorphology, 2015 Publication	<1 %
29	acikbilim.yok.gov.tr Internet Source	<1 %
30	link.springer.com Internet Source	<1 %
31	www.edureka.co Internet Source	<1 %
32	www.uv.es Internet Source	<1 %
33	"Advances in Knowledge Discovery and Data Mining", Springer Nature, 2009 Publication	<1 %
34	Submitted to Malaviya National Institute of Technology Student Paper	<1 %

35	Zou, S.x.. "A Novel Method for Prediction of Protein Domain Using Distance-Based Maximal Entropy", Journal of Bionic Engineering, 200809 Publication	<1 %
36	mtg.upf.edu Internet Source	<1 %
37	service.mof.gov.tw Internet Source	<1 %
38	vdocuments.mx Internet Source	<1 %
39	www.ijnrd.org Internet Source	<1 %
40	James Pearson, Clara Pennock, Tom Robinson. "Auto-detection of strong gravitational lenses using convolutional neural networks", Emergent Scientist, 2018 Publication	<1 %
41	www.ncbi.nlm.nih.gov Internet Source	<1 %

Exclude quotes Off
Exclude bibliography On
Exclude matches Off

PRESIDENCY UNIVERSITY

Presidency University Act, 2013 of the Karnataka Act No. 41 of 2013 | Established under Section 2(f) of UGC Act, 1956
Approved by AICTE, New Delhi | Approved By BCI
Bengaluru

PRESIDENCY GROUP OVER 45 YEARS OF ACADEMIC WISDOM

GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS

Certificate of Presentation

This is to certify that Mr./Ms. Shivam Narayan Under the Supervision of Dr./Mr./Ms. P Sudha from from PRESIDENCY UNIVERSITY, BANGALORE has successfully PRESENTED the paper at the National Conference on Recent Advancements and Challenges in Information Technology [NCRACIT-23] bearing the paper title **Uncovering Income Tax Fraud: A Logistic Regression Approach For Detection And Prevention** and paper ID **244** held during 28th April 2023 - 29th April 2023.

Dr. Gopal K Shyam	Dr. Manujakshi B C	Dr. C Kalaiarasan	Dr. Md. Sameeruddin Khan
Conference Chair, Prof. and Head, Dept. Of CSE	Conference Chair, Associate Prof., Dept. Of CSE	General Co-Chair, Associate Dean – SOCS&IS	General Chair, Dean – SOCS&IS