**PREDICTION OF POTENTIAL FRAUD IN INCOME TAX USING RANDOM FOREST**

**Table of Contents**

[**OBJECTIVE**](#_htba1r8p7qdh) 2

[**METHODOLOGY**](#_dxbsnk28n2q3) 2

[**END-TO-END PROCESS WITH SOLUTION ARCHITECTURE**](#_wb4ylh9bfk8n) 2

[Importing libraries in Python](#_dktdskk31cg2) 2

[Data exploration](#_v1pkdflnq8tk) 4

[Data preprocessing](#_g0nf67ypy9r2) 4

[Exploratory data analysis (EDA)](#_el9im88t9nfs) 7

[Model development](#_1w8ma41xa7cy) 10

[**CHALLENGES FACED**](#_nytr2udy97v8) 12

[**COMPLEXITY LEVEL**](#_ks96fm2fdr7x) 13

# OBJECTIVE

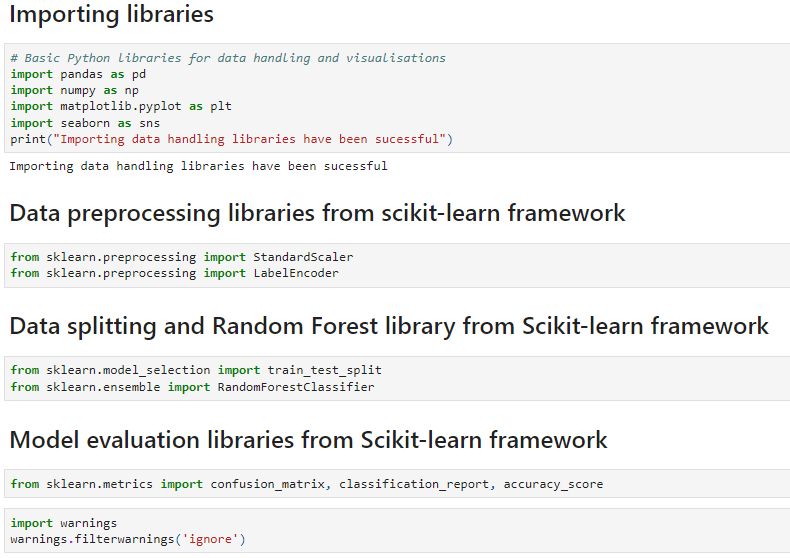
The aim of this project is to develop a Random Forest Classification model to identify individuals with ‘Risky’ taxable income.

# METHODOLOGY

The methodology is based on machine learning algorithms based on secondary quantitative fraud data. The target variable (Taxable.Income) has been categorised into two categories (Risky and Good), where taxable\_income <= 30000 has been considered as “Risky” and others are “Good”. Thus, due to the categorical nature of the target variable, the Classification algorithm has been preferred in this project. Random Forest Classification model has been opted for this study, which has been developed, trained and evaluated using Python programming language in Jupyter Notebook IDE.

# END-TO-END PROCESS WITH SOLUTION ARCHITECTURE

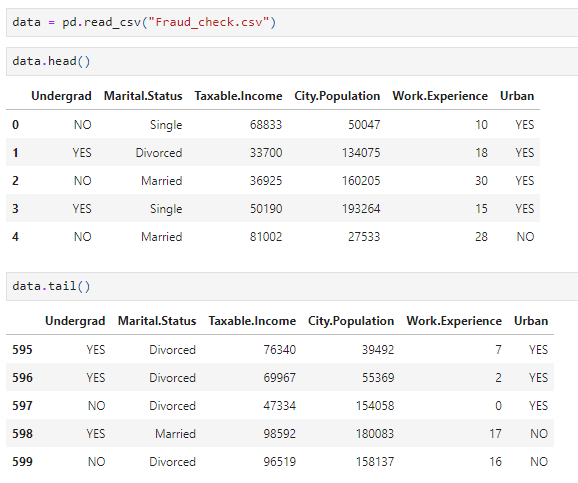
## Importing libraries in Python

****

**Figure 1: Importing Python libraries**

Pandas library has been imported into Python for data loading and manipulation, whereas Seaborn and Matplotlib libraries have been used for data visualisations. In fact, for performing mathematical operations and data aggregation, the Numpy library has been used in this project. For developing the random Forest Classification model, the ‘RandomForestClassifier ()’ module has been imported from the ‘sklearn.ensemble’ library. Additionally, for performing data transformation, the ‘LabelEncoder’ module has been utilised for label encoding of target variable. In fact, for scaling the numerical features, the StandardScalar () module has been imported from the ‘sklearn.preprocessing’ library (**Refer to Figure 1**). All the mentioned modules have been imported from sklearn.preprocessing library from ‘Scikit-learn’ framework. Additionally, essential modules for splitting the dataset (train\_test\_split) and evaluating the model using metrics such as the confusion matrix, accuracy score, and classification report have also been imported from the sklearn.metrics module.

## Data exploration

****

**Figure 2: Data loading**

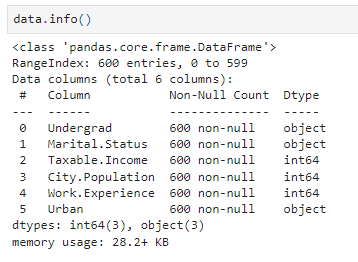
The dataset has been loaded into Python environment by using ‘read\_csv ()’ function from the Pandas library. After loading the dataset, head and tail of the dataset have been shown to check whether the dataset is appropriately loaded or not into the Python environment (**Refer to Figure 2**).

## Data preprocessing

****

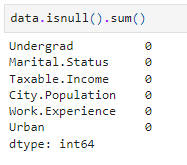
**Figure 3: Shape of the dataset**

The shape of the dataset is (600,6), which indicates that the dataset contains 600 observations and 6 variables (**Refer to Figure 3**).

****

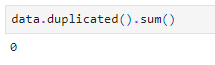
**Figure 4: Info of the dataset**

From Figure 4, it can be observed that the dataset contains 3 categorical variables (Undergrad, Marital. Status and Urban), and 3 numerical variables (Taxable.Income, City.Population and Work.Experience) (**Refer to Figure 4**).

****

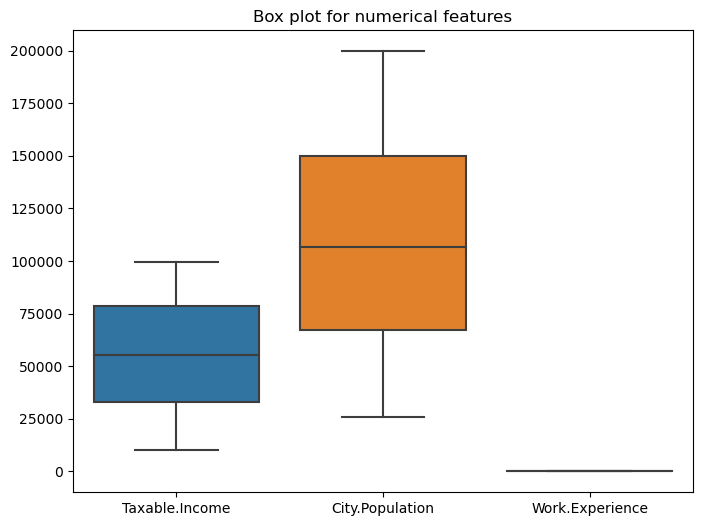
**Figure 5: Null values in the dataset**

Missing values in the dataset have been checked using the ‘isnull (). sum ()’ function, from which the observed null values in the dataset for all variables are 0 (**Refer to Figure 5**). This reflects that the dataset contains no data errors, reflecting appropriateness of the dataset for further analysis.

****

**Figure 6: Duplicate values in the dataset**

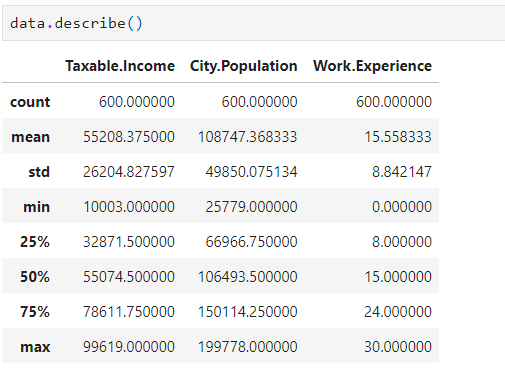
Duplicate values indicate data repetition, which can cause model overfitting, due to this, duplicate values have been checked using ‘duplicated (). sum ()’ from Pandas library (**Refer to Figure 6**). The observed duplicate values are 0, indicating the non-existence of duplicate values in the dataset.

****

**Figure 7: Box plots of the numerical variables to check outliers**

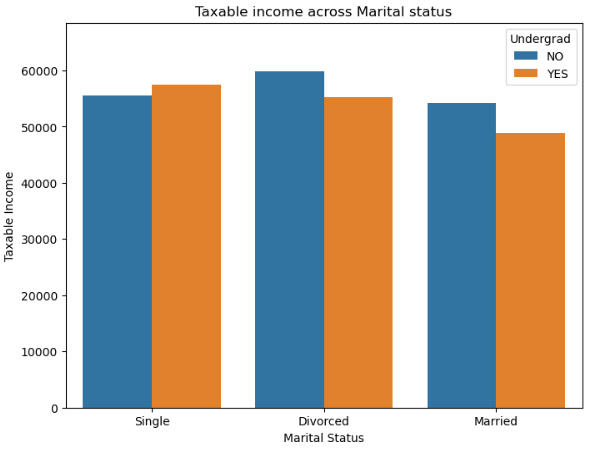
From the box plot, it can be observed that there are no outliers in the numerical variables. Additionally, the variables ‘Taxable.Income’ and ‘City.Population’ followed a widely spread distribution (**Refer to Figure 7**).

## Exploratory data analysis (EDA)

****

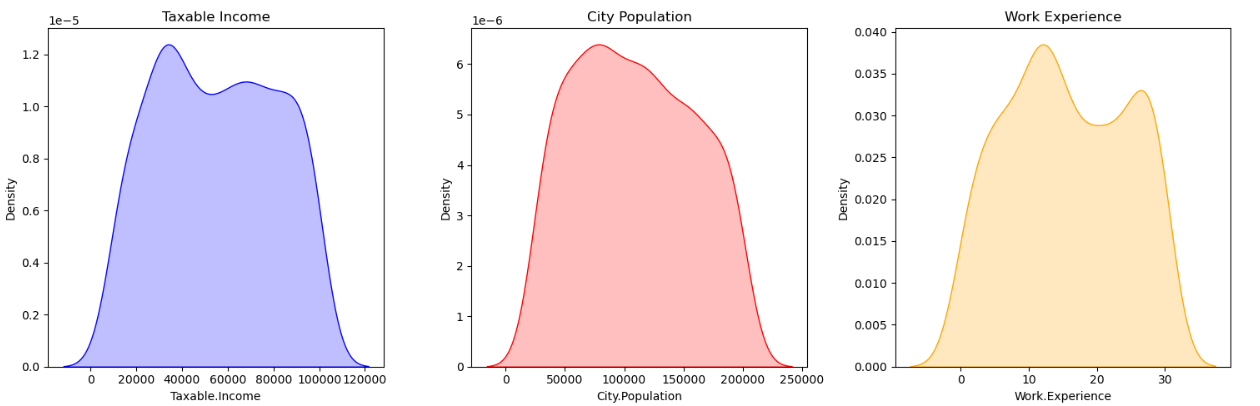
**Figure 8: Descriptive statistics**

**Figure 8** shows the summary statistics (central tendency and variability) of numerical variables (Taxable.Income, City.Population and Work.Experience), from which it can be observed that standard deviation of these variables is considerably high. This indicates a high level of variability.

****

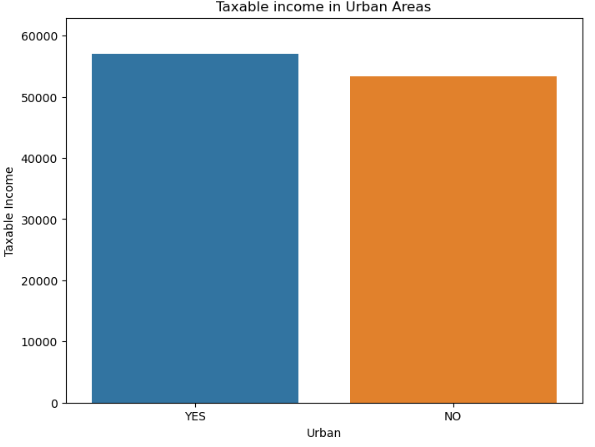
**Figure 9: Distribution of Taxable.Income across Marital Status**

From **Figure 9**, it can be observed that Taxable Income is slightly higher among the divorced population, compared to the single and married population. This reflects a low level of tax risk among the divorced population.

****

**Figure 10: KDE plot for numerical features (Taxable.Income, City.Population and Work Experience)**

Kernel density plots for Taxable.Income, City.Population and Work Experience have followed a bell-shaped curve, highlighting approximately normal distribution for these variables (**Refer to Figure 10**).

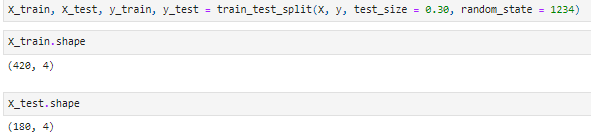
****

**Figure 11: Distribution of Taxable income across regions (Urban or not)**

Taxable.Income is slightly higher among the population from urban areas, reflecting a lower level of taxation risks across the urban population (**Refer to Figure 11**).

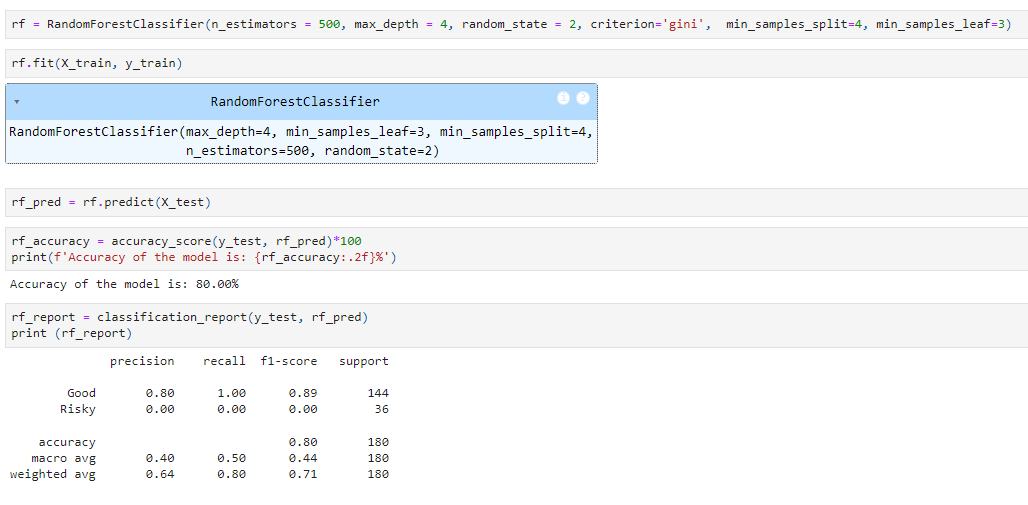
## Model development



****

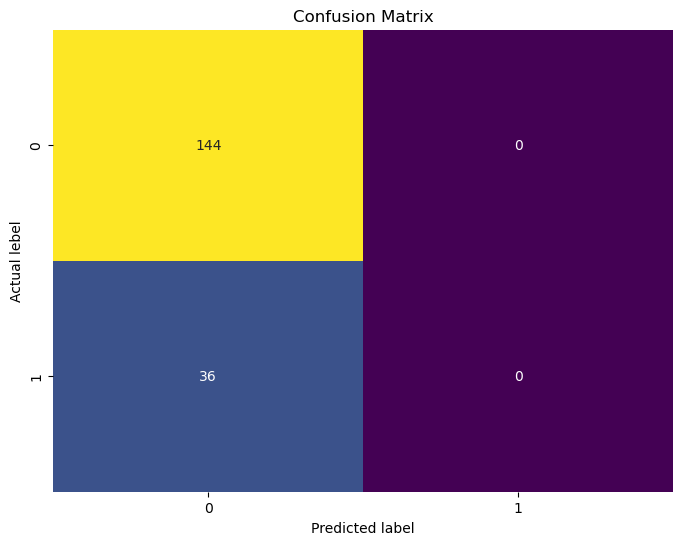
**Figure 12: target and features**

The features include categorical variables: Undergrad (0 = No, 1 = Yes), Urban (0 = No, 1 = Yes), and Marital.Status (0 = Single, 1 = Married, 2 = Divorced), along with numerical features like Work.Experience (years of work experience) (**Refer to Figure 12**). These factors are used to predict the target variable (Taxable.Income), likely a classification label such as Good or Risky.

****

**Figure 13: Model architecture and classification report**

**Figure 13** shows that the Random Forest model has 500 decision trees, each limited to a depth of 4, with a minimum of 4 samples required to split a node and 3 samples per leaf. It achieves 80% accuracy, but the model performs poorly in classifying the ‘Risky’ category, showing zero recall and precision, resulting in unbalanced performance.

****

**Figure: Confusion matrix**

The above figure shows the true positive, true negative, false positive and false negative cases, from which it can be observed that the true negative instances are 144, while the false negative instances are 36. Thus, the model shows a slight level of misclassification in the prediction of false negative cases.

# CHALLENGES FACED

The main challenge faced in this project is the treatment of the categorical variables in this project. The challenges have been tackled by selecting appropriate encoding methods (for example, LabelEncoder for the target variable). Additionally, challenges have been observed in tuning the Random Forest model for optimising model accuracy.

# COMPLEXITY LEVEL

The complexity of the project can be considered as intermediate due to the requirement of categorical encoding and selection of optimal values of the parameters of the Random Forest model for obtained optimal predictive accuracy.