**Course: Artificial Intelligence and Data Science**

**Module Leader: Mr. Sahan Priyanayana**

**CM2604 Machine Learning Assignment Type: Individual**

**Student Name: W.A.C.I.V Bandara**

**IIT ID-20222453**

**RGU ID-2311689**

**Table of content**

**Corpus preparation**

The UCL repository is a website which contain huge selection of datasets for machine learning tasks. Adult data set was selected form UCL repository and this dataset contain 48842 data records and 15 attributes and indicate that given person Census Income is more than 50K or not.

Data cleaning was one of the pre-processing steps that was carried out. A few of the steps included removing outliers, rejecting duplicate values, and looking for and removing null values.

Random Forest and Naïve Bayes are the algorithms that used for classify weather the salary exceeds 50K or not. After cleaning and preprocessing phase 70% of data is used for training while remaining amount used for testing purposes.

**Preprocessing techniques**

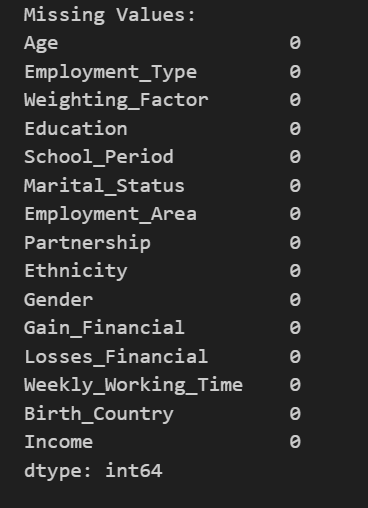
The preprocessing stage is crucial to the refinement of unprocessed data for machine learning models. To guarantee data quality and compatibility, they include activities like data cleansing, scaling, and encoding. By simplifying and emphasizing pertinent data, feature selection and dimensionality reduction expedite the training of models. Text preprocessing transforms textual data into numerical representations in order to make it ready for analysis. Class distribution biases are addressed via methods for handling unbalanced data, and efficient model evaluation is facilitated by data partitioning. These methods work together to provide the foundation for data preprocessing, which makes it possible to create reliable and accurate machine learning models.

**Data Cleaning visualization and preprocessing**

The process of finding and fixing mistakes, inconsistencies, and inaccuracies in a dataset such that it is reliable and of high quality for analysis or modelling is known as data cleaning. This includes dealing with duplicate or missing values, fixing mistakes in data entry, eliminating outliers, and standardizing formats, among other things. Through the resolution of these problems, data cleaning increases the dataset's integrity, lowers the possibility of inaccurate or biased outcomes, and boosts the efficiency of ensuing data analysis or machine learning operations.

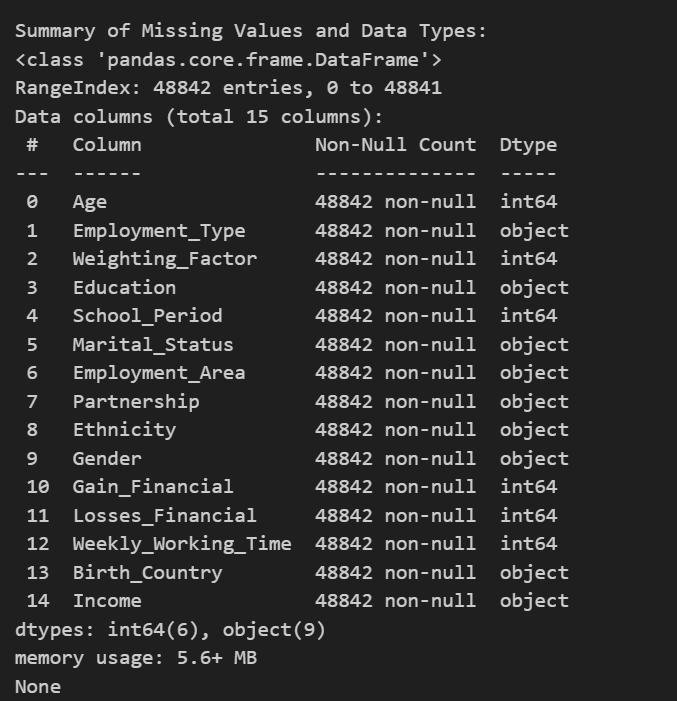
**Checking missing values in the dataset**



****

Printing missing values

****

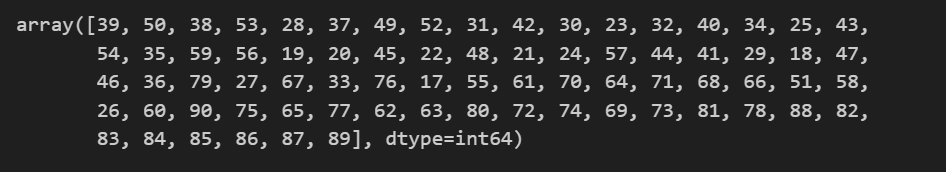
****

**Drop duplicate values**

****

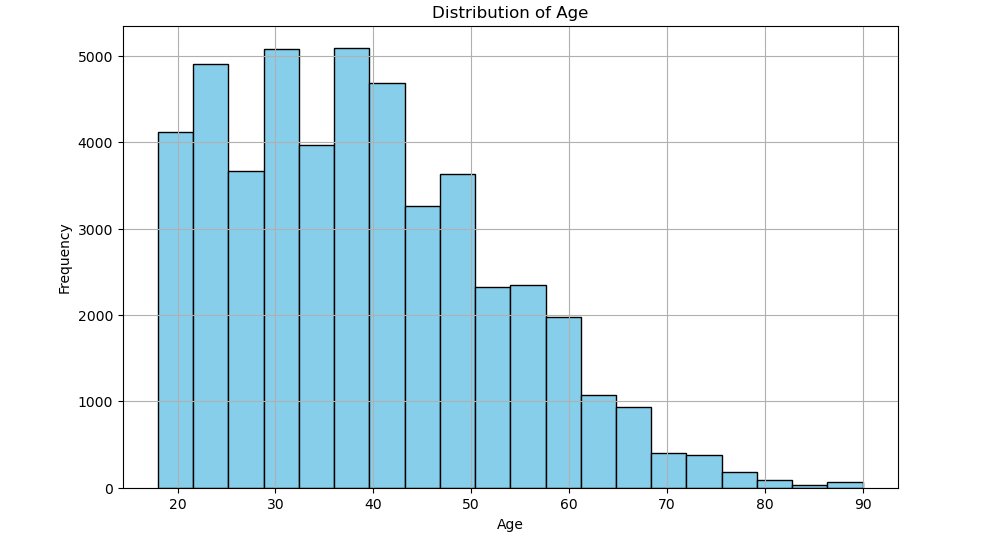
**Handling age Variable**





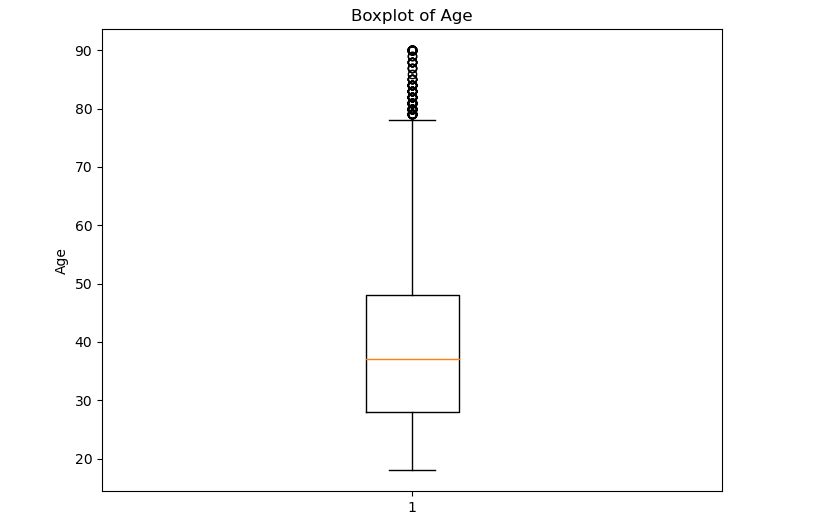
Visualizing age variable before handling outliers. Needed to check how the data is distributed with the age and good idea on data distribution is very important in handling outliers.





Made the boxplot using IQR method to handle outliers





Calculated the number of outliers to take the most appropriate decision to handle outliers



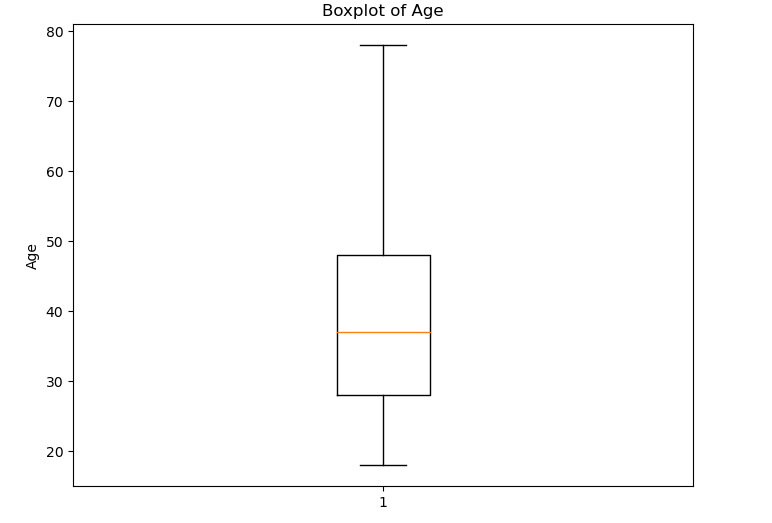
Number of outliers



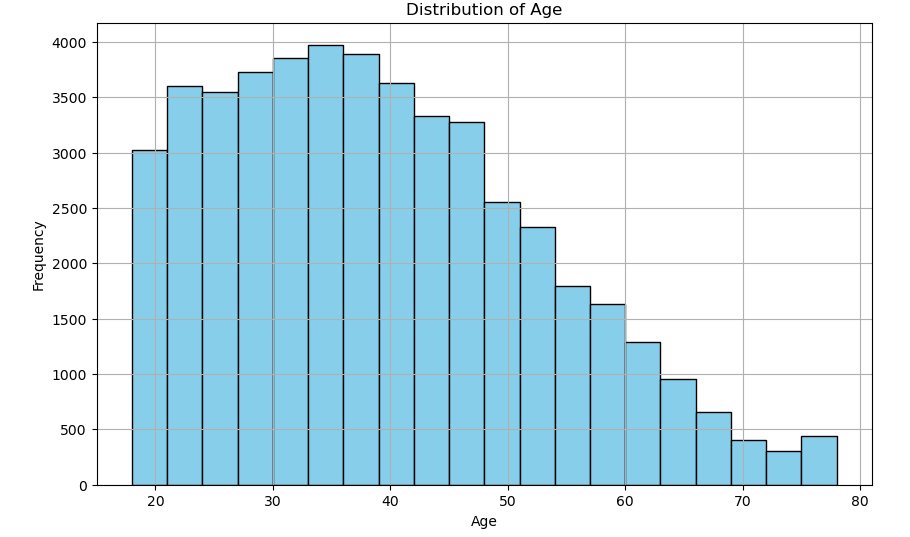
noticed that number of outliers are a little amount when comparing with total dataset and then decided to winsorize it to lower and upper bound of the boxplot



Visualization of boxplot after handling outliers



Distribution of age variable after handling the outliers

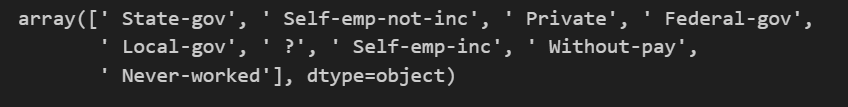


**Handling employment type Variable**

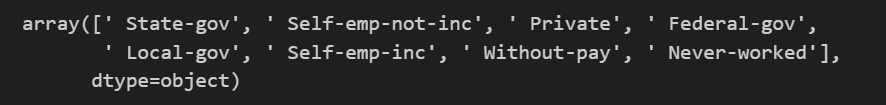
Found some data inconsistencies and removed those inconsistencies



Filtered and removed data inconsistencies in the specific variable.



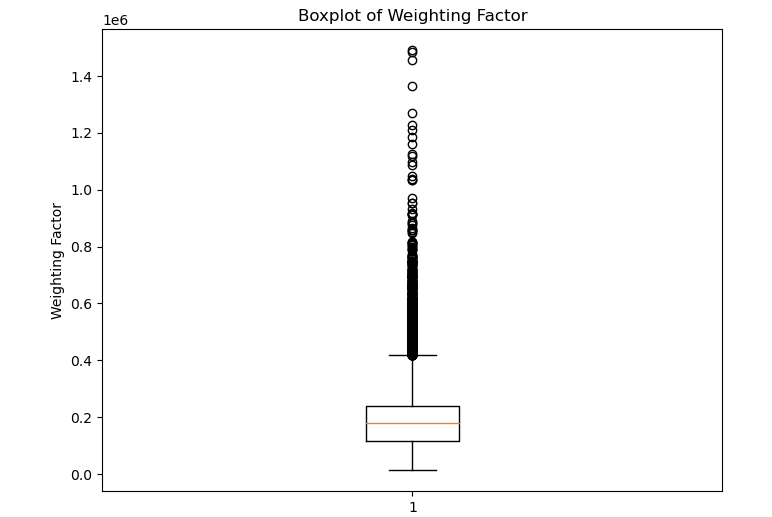
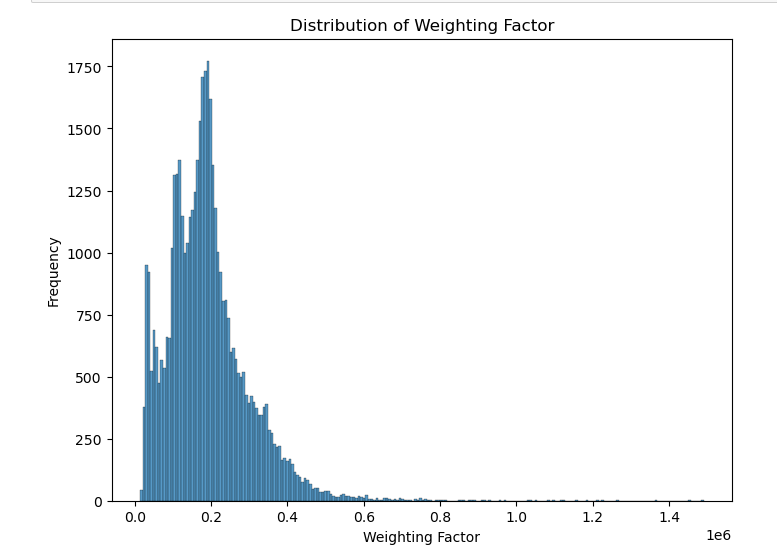
After handling inconsistencies



**Handling weighting factor Variable**

In this variable we cannot find proper description on meta data. But visualized the distribution of data and made the boxplot of outliers.

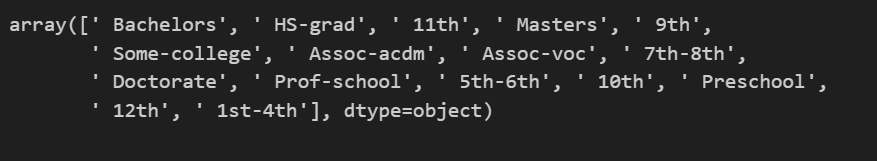




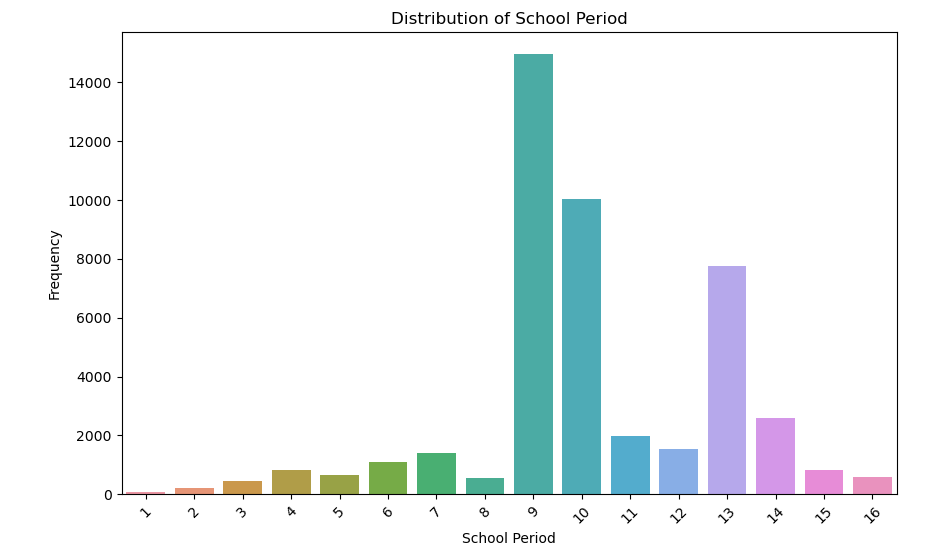
**Handling Education and School period variables**

Visualized and made a self-inspection on Education and School period.







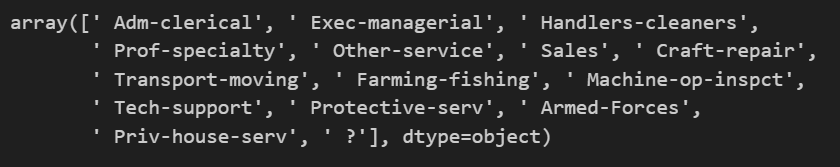


No issues in the distribution of this column and no data inconsistencies.

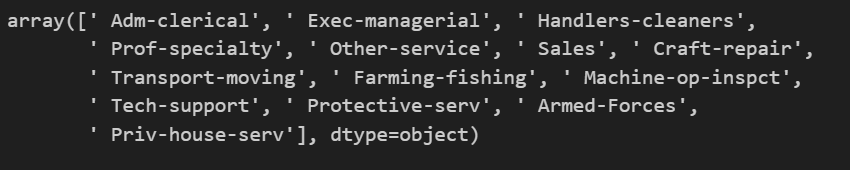
**Handling Employment area column**



Before handling data inconsistencies

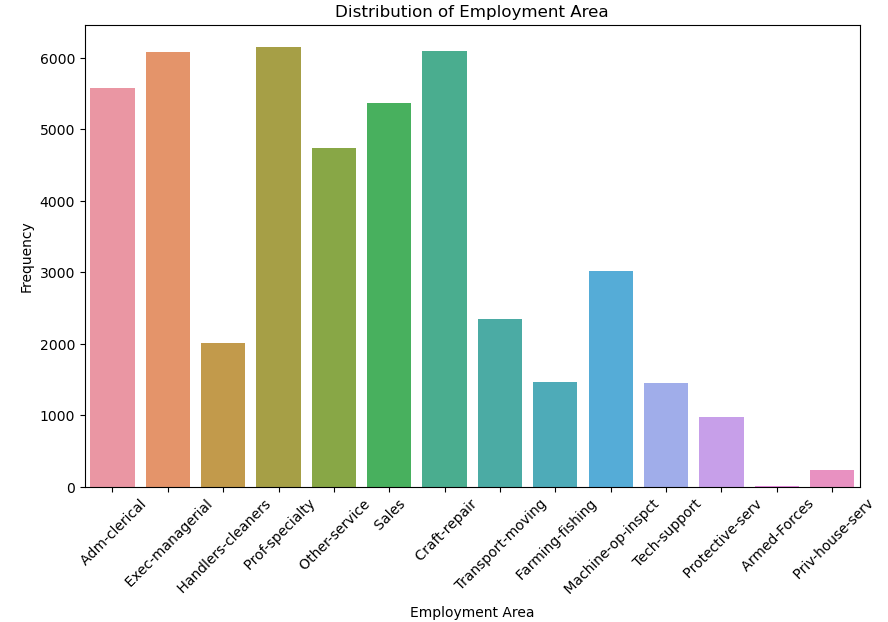


After handling data inconsistencies



Distribution of data in Employment area

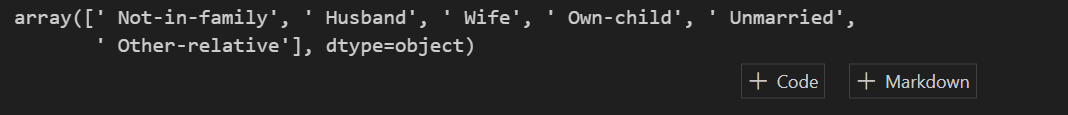


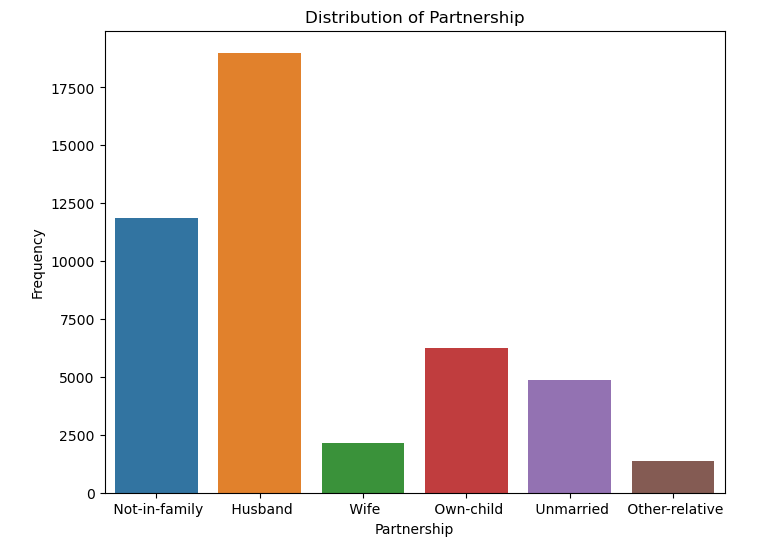


**Handling partnership variable**

There are no missing or data inconsistencies in this variable but visualized the distribution of data and unique values.



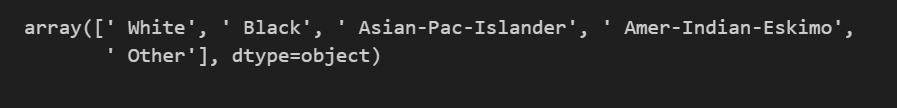




**Handling Ethnicity variable**

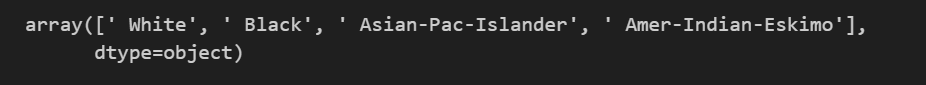


Before handling

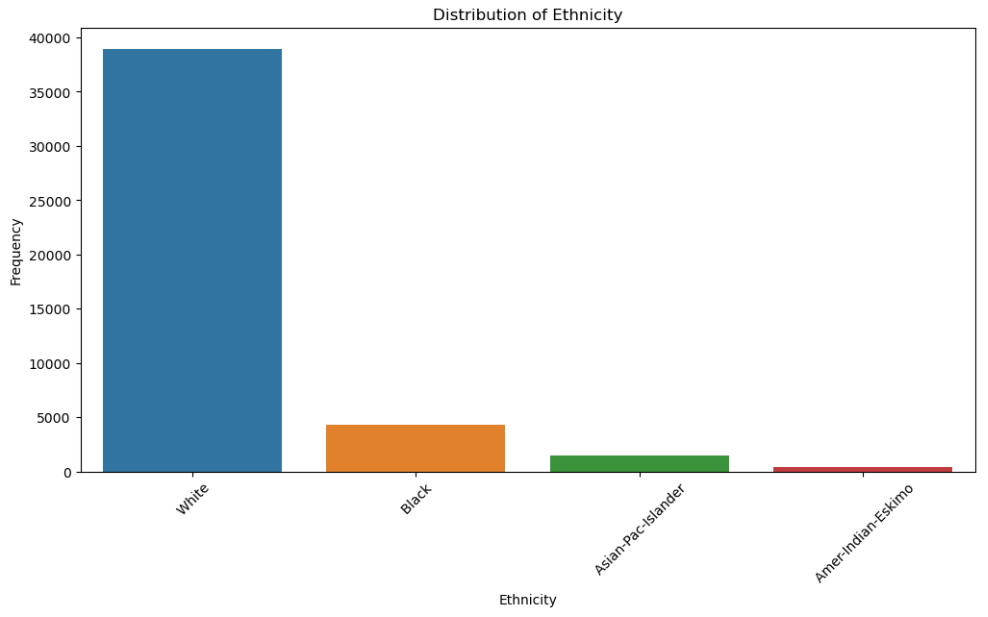


There is an unidentified value called “Other” and removed that variable.

After handling



Distribution of data

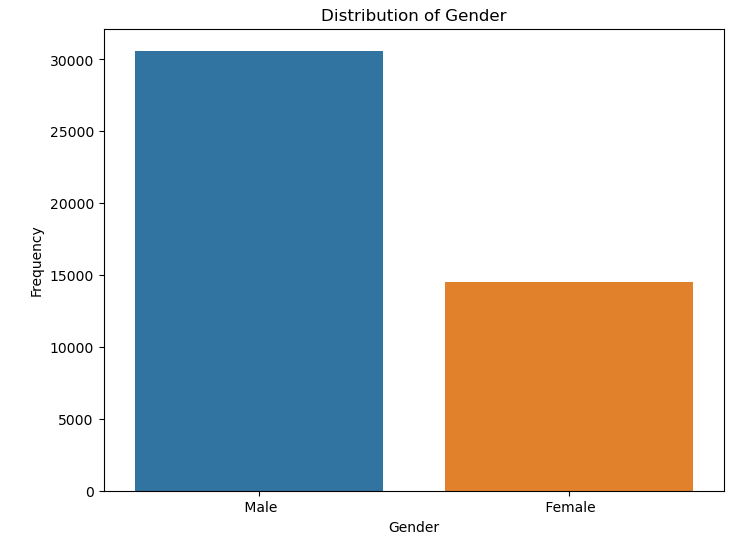


**Handling Gender variable**



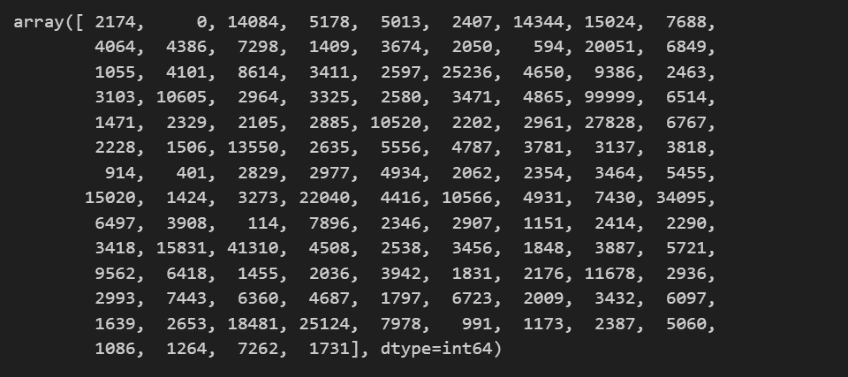


Distribution

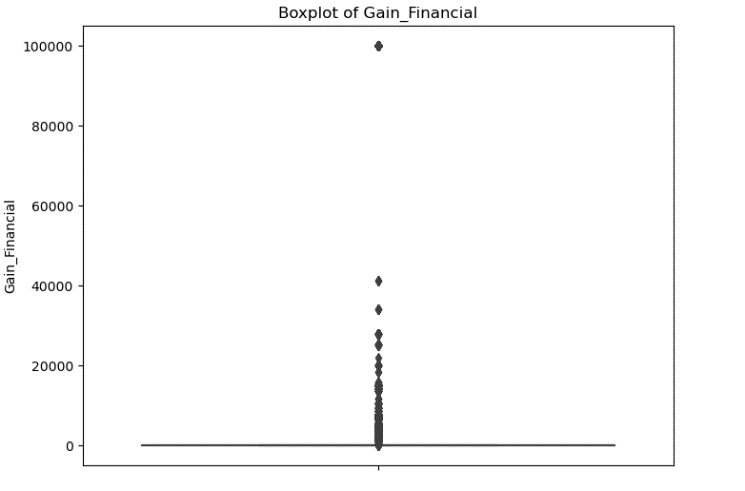


**Handling Gain financial variable**

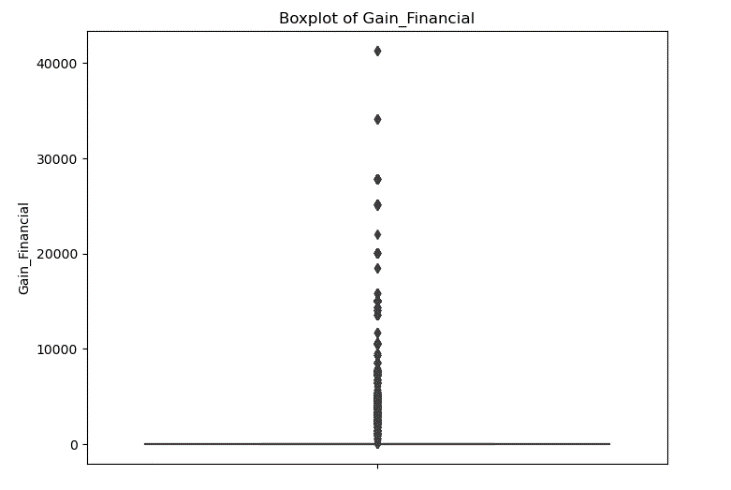
Distribution of the array



Boxplot before handling the outliers.



Boxplot after handling outliers



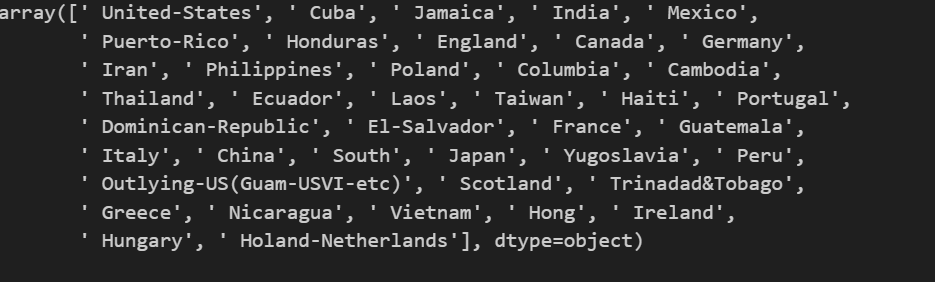
**Handling Birth Country variable**

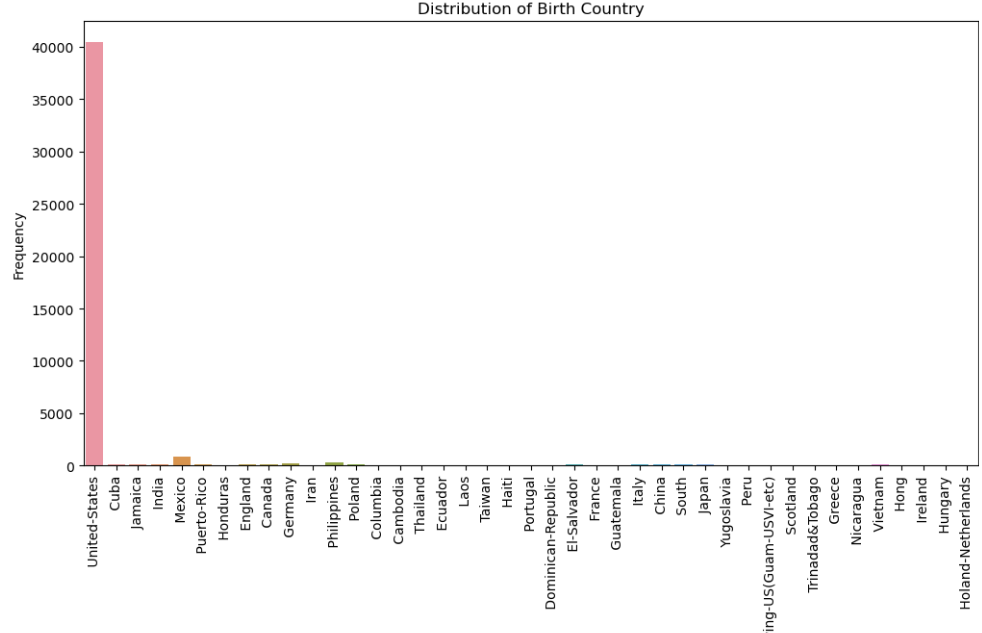


Before handling missing values



After handling missing values



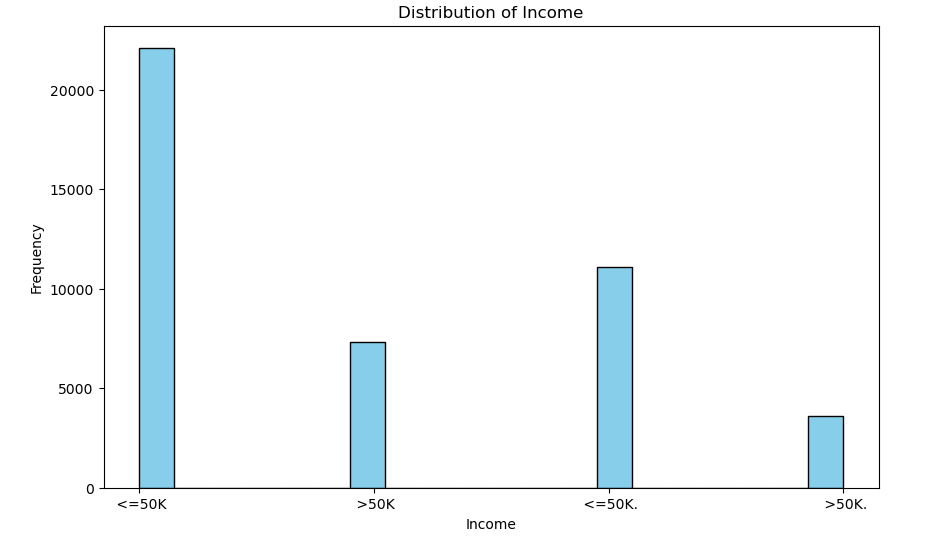


**Handling Income Column**

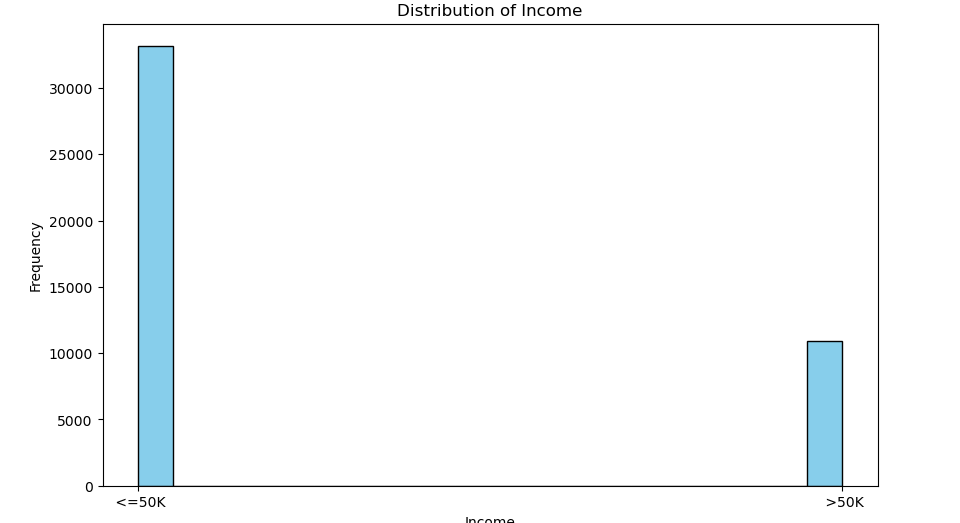




Before handling inconsistencies



After handling inconsistencies

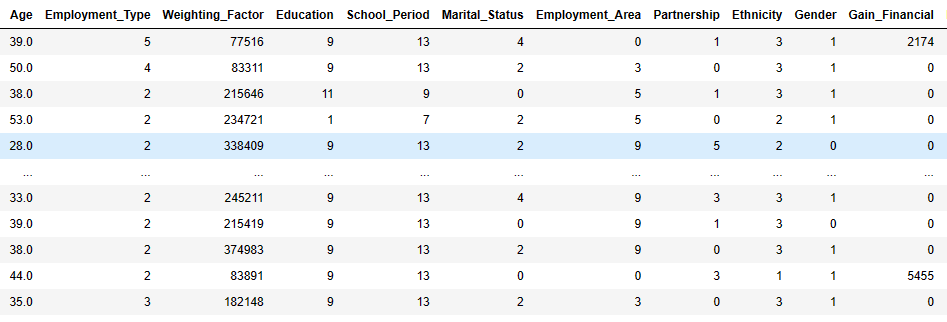


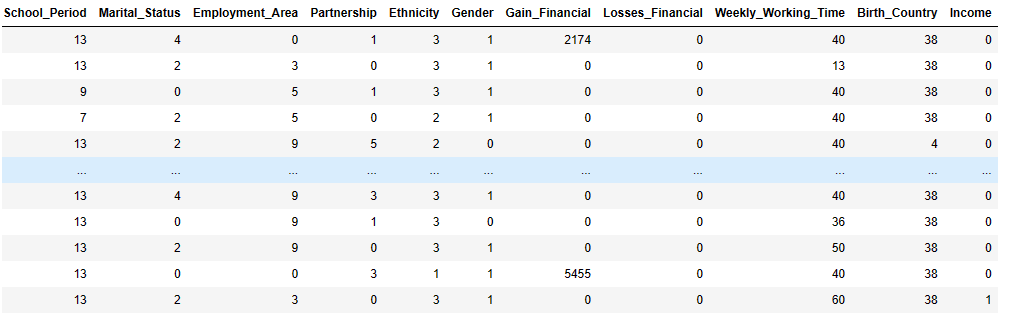
**Feature Encoding**

Used label encoder for encoding



After encoding

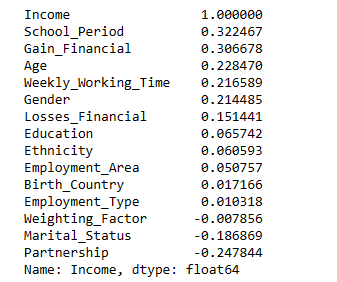




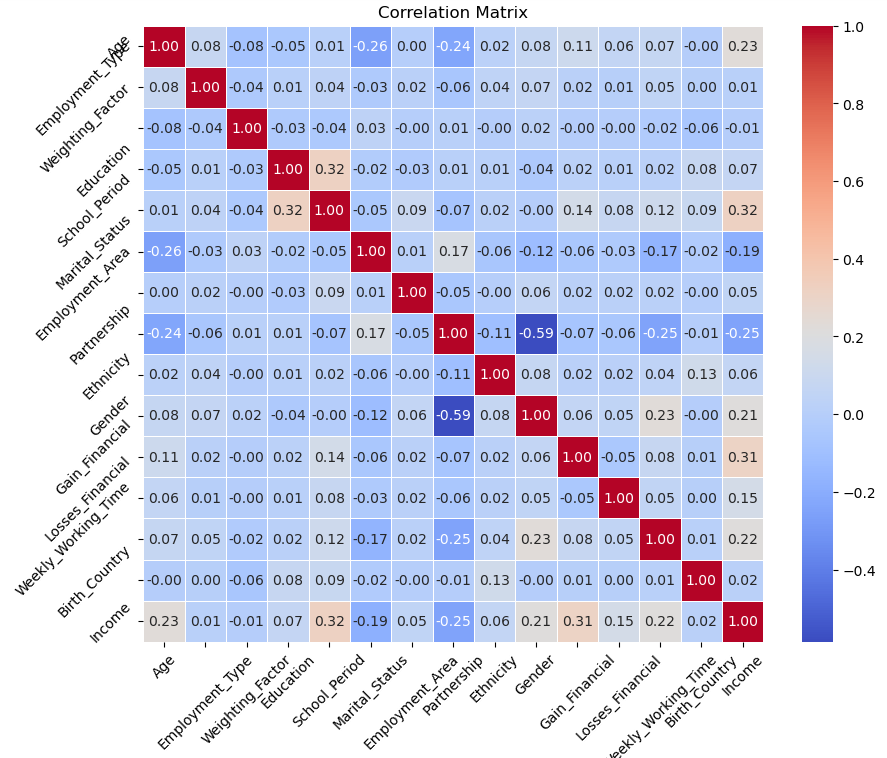
**Feature engineering**

For selecting best features for training the model used correlation matrix instead of PCA (Principal component analysis). Correlation matrix is the best way to find the most correlated features for the target variable when there are a smaller number of attributes in the dataset.









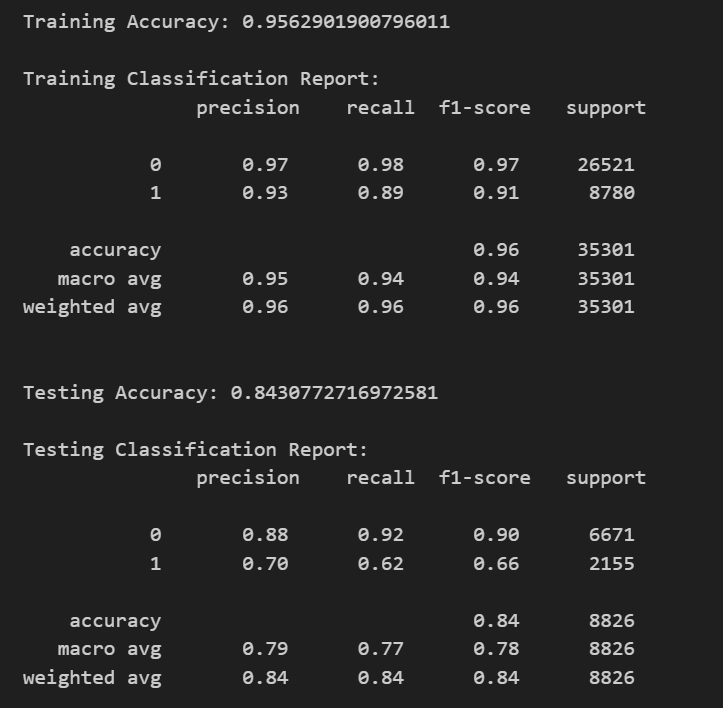
Selecting best features from the correlation matrix and dividing into training and test data



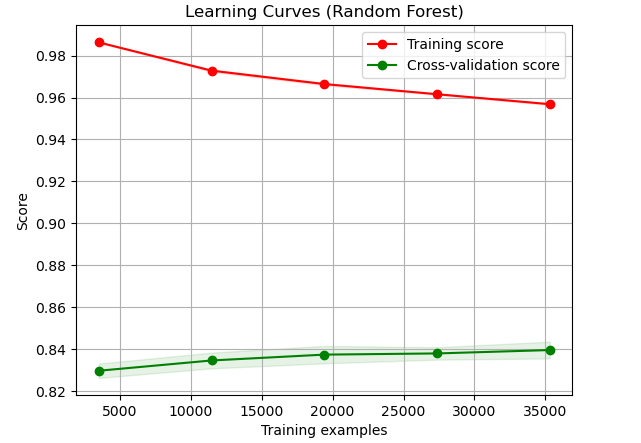
**Model Training using Random forest classifier**



Results of random forest algorithm

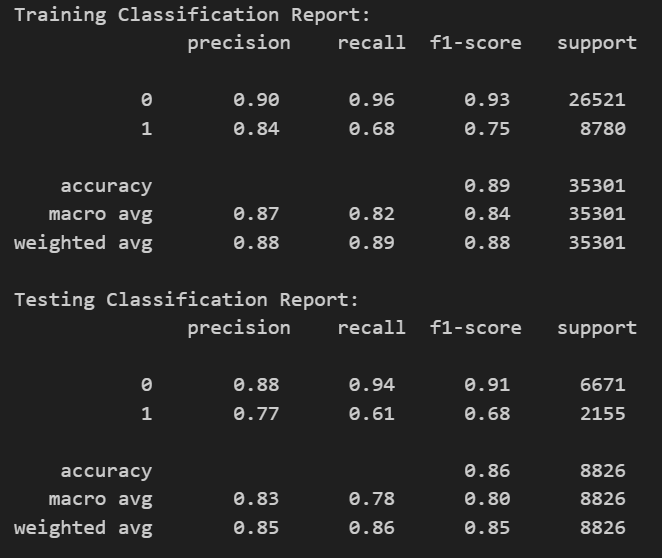


Classification reports are ok but there is a clear gap between training and testing data. For visualize that I used training and test curves



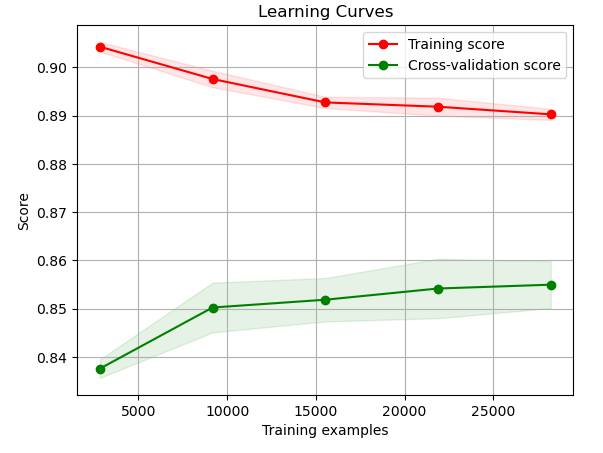
Then used Hyperparameter tunning for tunning process and I used GridSearchCv method for this





After evaluation learning curves





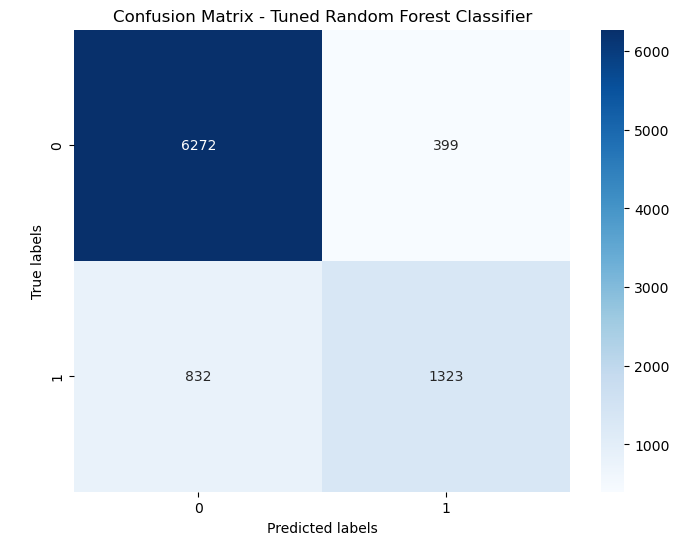
Applied cross validation to validate further



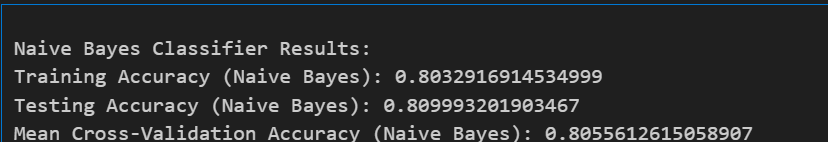
Result



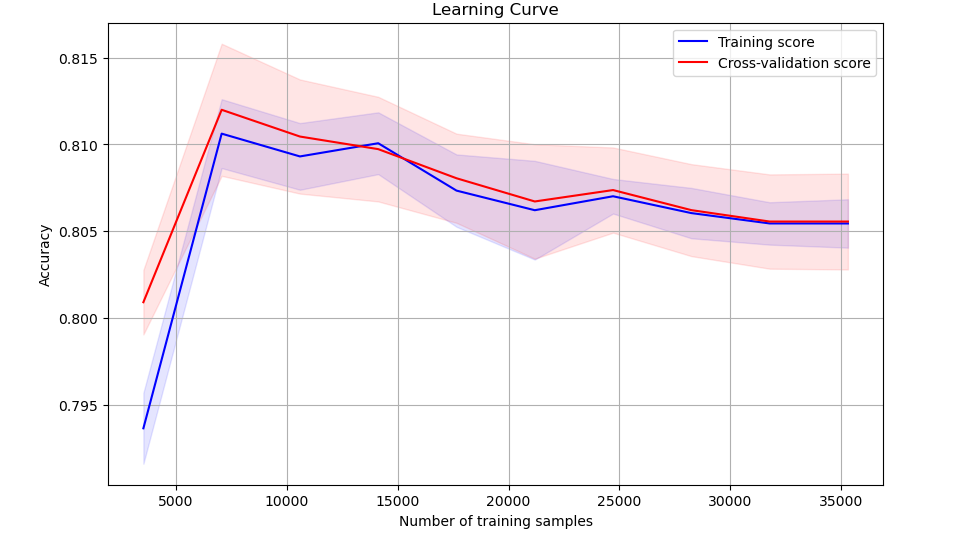
Confusion matrix



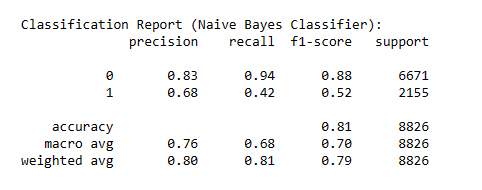
**Model Training using Naïve Bayes algorithm**



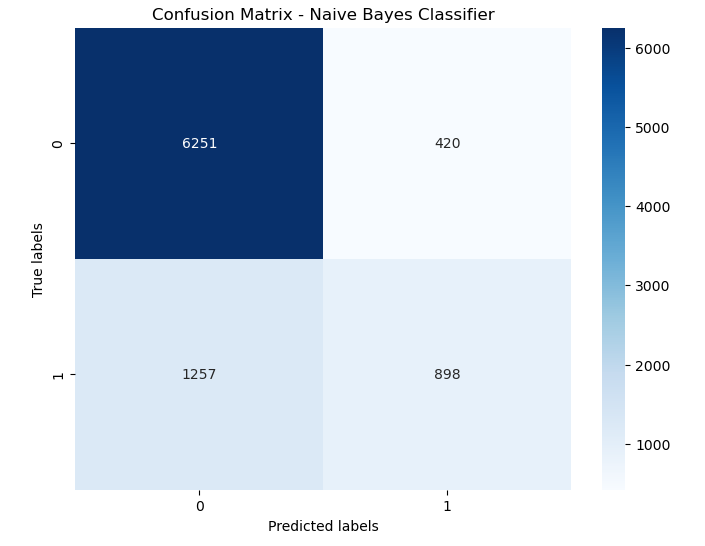
Training and testing curves



Evaluation results



Confusion matrix



**Solution methodology**

Random forest and Naïve Bayes algorithms are used for classify salary. Summary of two algorithms that used were given below.

**Random forest** - This is a popular and adaptable ensemble learning technique for tasks involving regression and classification. It's a decision tree extension in which several decision trees are trained using various training data subsets and then combined to provide predictions.

* Multiple subsets are created by bootstrapping, which involves sampling random subsets of the training data with replacement.
* There is a decision tree constructed for every data subset. Nevertheless, a random subset of features is chosen for splitting at each node of the tree rather than all of the features. The trees are better decorated because of this unpredictability.
* To arrive at the final forecast, predictions from individual trees are aggregated by voting (for classification) or averaging (for regression).

Random forests are renowned for their high accuracy, scalability to huge datasets, and resilience against overfitting. When working with multi-feature, high-dimensional data, they are especially useful.

**Naïve Bayes** – This algorithm relies on the premise of predictor independence and is a probabilistic classification method that applies the Bayes theorem.

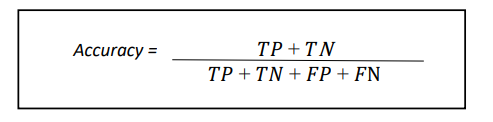
* Using the Bayes theorem, it determines the likelihood of each class given a collection of input features.
* The "naive" assumption of Naive Bayes simplifies probability calculations by assuming that the existence of a given characteristic in a class is independent of the existence of any other feature (i.e., features are analyzed independently).
* The class with the highest probability is then chosen to be the input data's predicted class.

In order to estimate the required parameters, naive Bayes classifiers require very minimal training data and are quick and simple to deploy. However, feature independence may not always hold true, which in some circumstances may result in less than ideal performance.

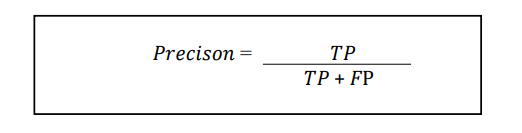
**Evaluation Criteria**

For income prediction use case used accuracy, precision, recall, and F1- scores as evaluation metrices and shown here, along with the explanations for their application.

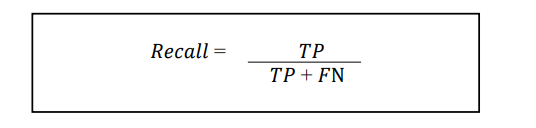
**Accuracy** - measure, which indicates the percentage of correctly predicted occurrences among all instances in a dataset, is used to evaluate the performance of a machine learning model.



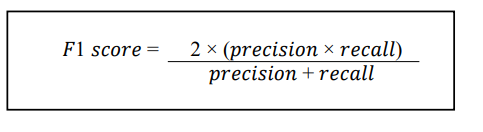
**Precision** - metric that evaluates the actual positive forecasts' accuracy



**Recall** - The proportion of cases that were accurately classified.



**F1 Score** - A model accuracy indicator that combines precision and recall into a single number.



**Evaluation results**

|  |  |  |
| --- | --- | --- |
|  | Random forest algorithm | Naïve Bayes algorithm |
| Train and test split | Train – 80%  Test – 20% | Train – 80%  Test – 20% |
| Test data accuracy | 86% | 80% |
| Train data accuracy | 89% | 80% |
| Precision | 0.88 | 0.83 |
| Recall | 0.94 | 0.94 |
| F1 Score | 0.91 | 0.88 |

**GitHub URL**

<https://github.com/Inupa6677/Machine-Learning-CW>