MACHINE LEARNING

(Finding Donors)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science and Engineering

By

K.Lokesh

221710313026

Under the Guidance of

Mr. _______

Assistant Professor

GITAM

(DEEMED TO BE UNIVERSITY)

(Estd. u/s 3 of the UGC Act, 1956)

VISAKHAPATNAM HYDERABAD BENGALURU

Department Of Computer Science and Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
June 2020

DECLARATION

I submit this industrial training work entitled “FINDING DONARS” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “Bachelor of Technology” in “Computer Science and Engineering”. I declare that it was carried out independently by me under the guidance of Mr. ______, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD k.Lokesh
Date: 221710313026

CERTIFICATE

This is to certify that the Industrial Training Report entitled “FINDING DONARS” is being submitted by K.Lokesh (221710313026) in partial fulfillment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2020-21

It is faithful record work carried out by her at the Computer Science and Engineering Department, GITAM University Hyderabad Campus under my guidance and supervision.

Mr.______________

Dr. S.Phani Kumar,
Professor and HOD,
CSE Dept.

certificate

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected Dr. N. Siva Prasad, Pro Vice Chancellor, GITAM Hyderabad and Dr. CH. Sanjay, Principal, GITAM Hyderabad.

I would like to thank respected Prof. S. Phani Kumar, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present the internship report. It helped me alot to realize what we study for.

I would like to thank the respected faculties ________ who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

K.Lokesh
221710313026

ABSTRACT

-> Machine learning algorithms are used to predict the values from the dataset by splitting the dataset into train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on donors dataset.

-> This sort of task can arise in a non-profit setting, where organizations survive on donations. Understanding an individual’s income can help a non-profit better understand how large of a donation to request, or whether or not they should reach out to begin with. As from our research we have found out that the individuals who are most likely to donate money to a charity are the ones that make more than $50,000.

Table of Contents:

CHAPTER 1

INFORMATION ABOUT MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

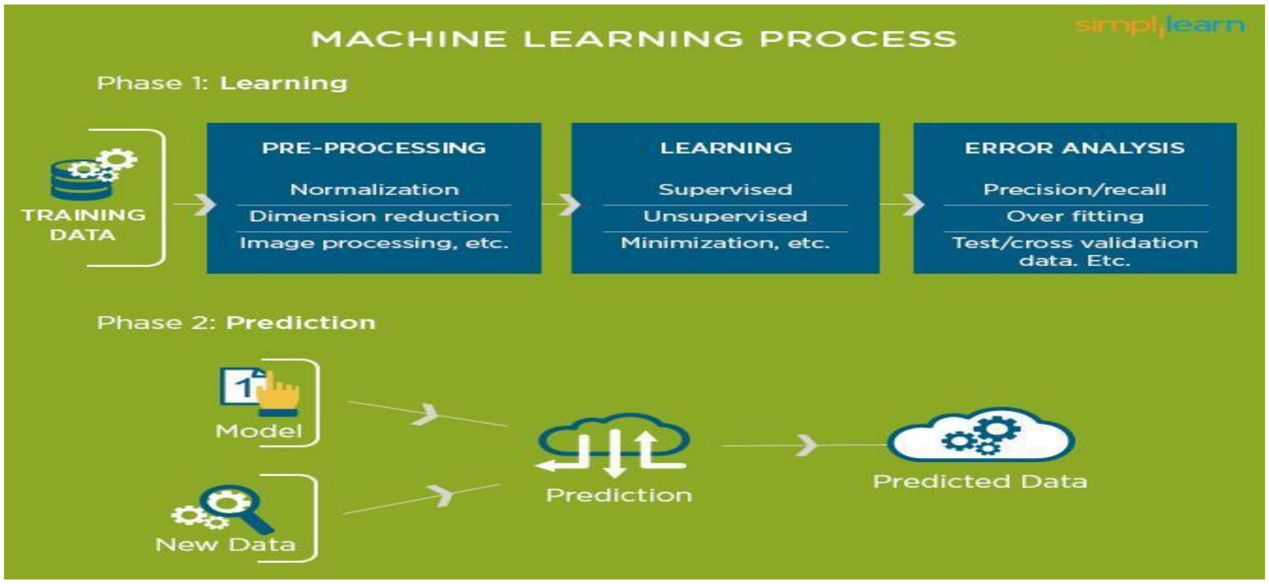


Fig 1.1 : The Process Flow

1.1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data Traditionally, data analysis was always being

characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.
By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

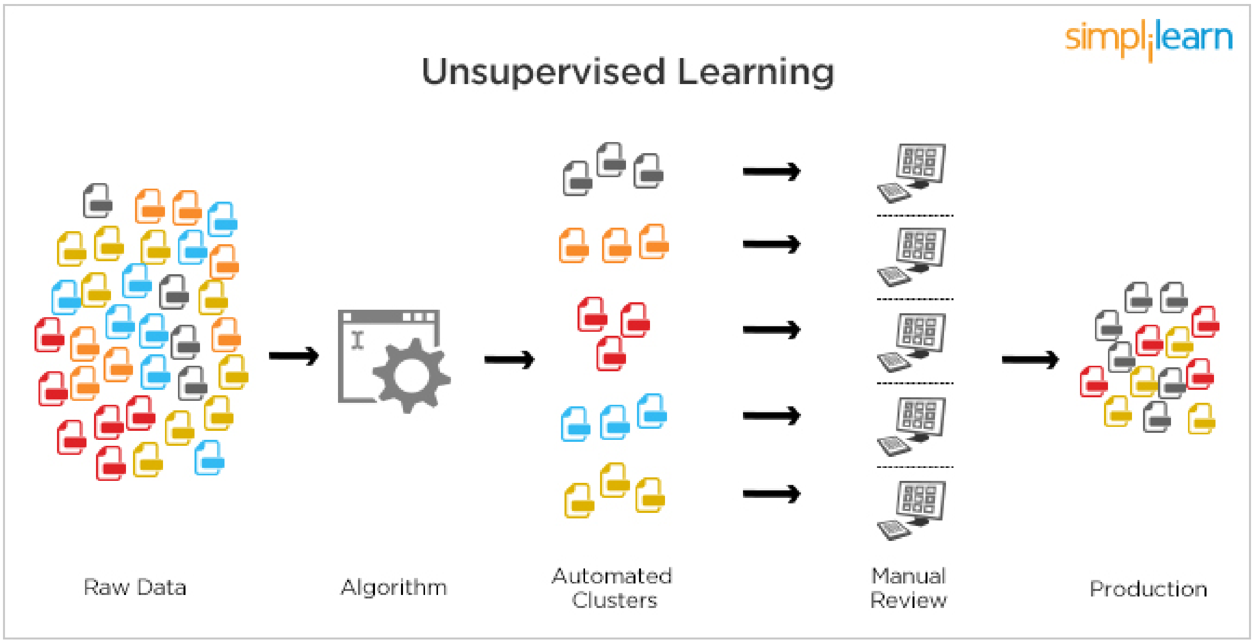


Fig 1.1.2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

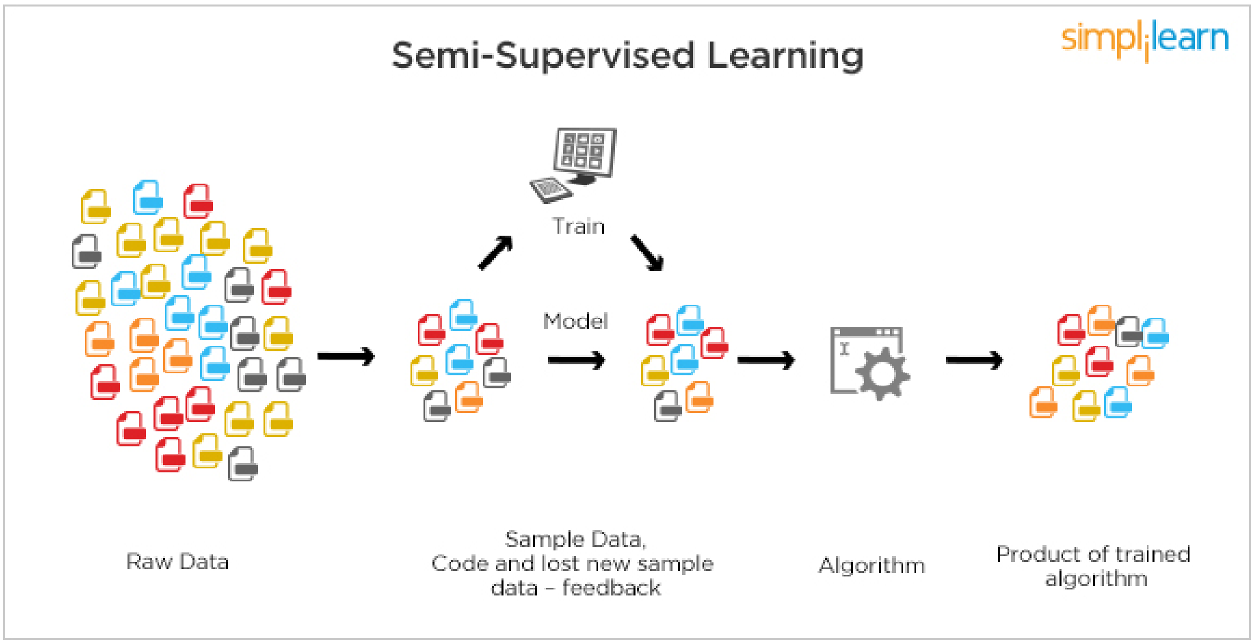


Fig 1.1.3 : Semi Supervised Learning

1.1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2
INFORMATION ABOUT PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

● Python is a high-level, interpreted, interactive and object-oriented scripting language.

● Python is a general purpose programming language that is often applied in scripting roles

● Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

● Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.

● Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

HISTORY OF PYTHON:

● Python was developed by GUIDO VAN ROSSUM in early 1990's

● Its latest version is 3.7 , it is generally called as python

2.2.2 FEATURES OF PYTHON:

● Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.

● Easy-to-read: Python code is more clearly defined and visible to the eyes.

● Easy-to-maintain: Python's source code is fairly easy-to-maintaining.

● A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

● Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

● Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

● Databases: Python provides interfaces to all major commercial databases.

● GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.2.3 HOW TO SETUP PYTHON:

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

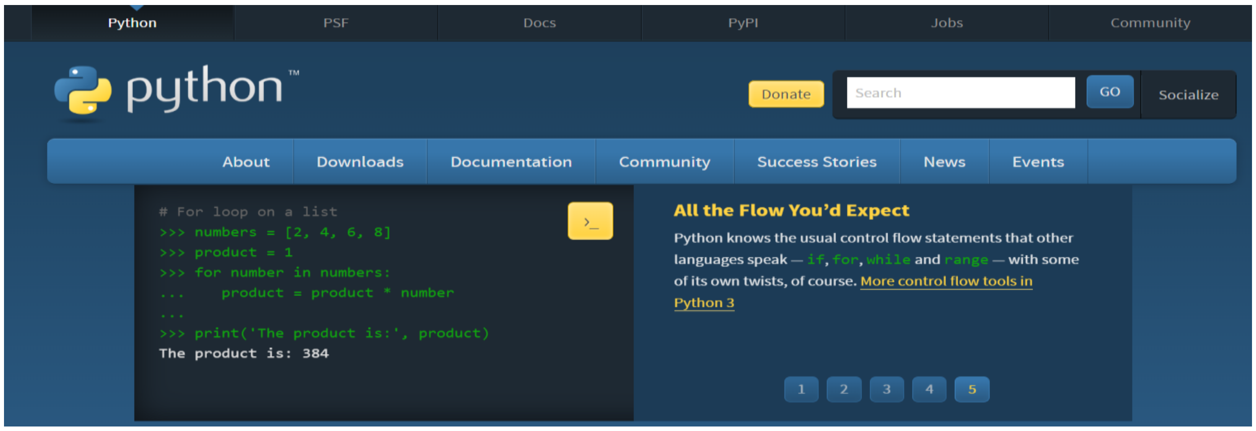
Installation(using python IDLE):

● Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.

● Download python from www.python.org

● When the download is completed, double click the file and follow the instructions to install it.

● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

All the Flow You'd Expect

Python knows the usual control flow statements that other languages speak — if, for, while and range — with some of its own twists, of course. More control flow tools in Python 3

Fig 2.1 : Python download

Installation(using Anaconda):

● Python programs are also executed using Anaconda.

● Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.

● Conda is a package manager quickly installs and manages packages.

● In WINDOWS:

● In windows

- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser).

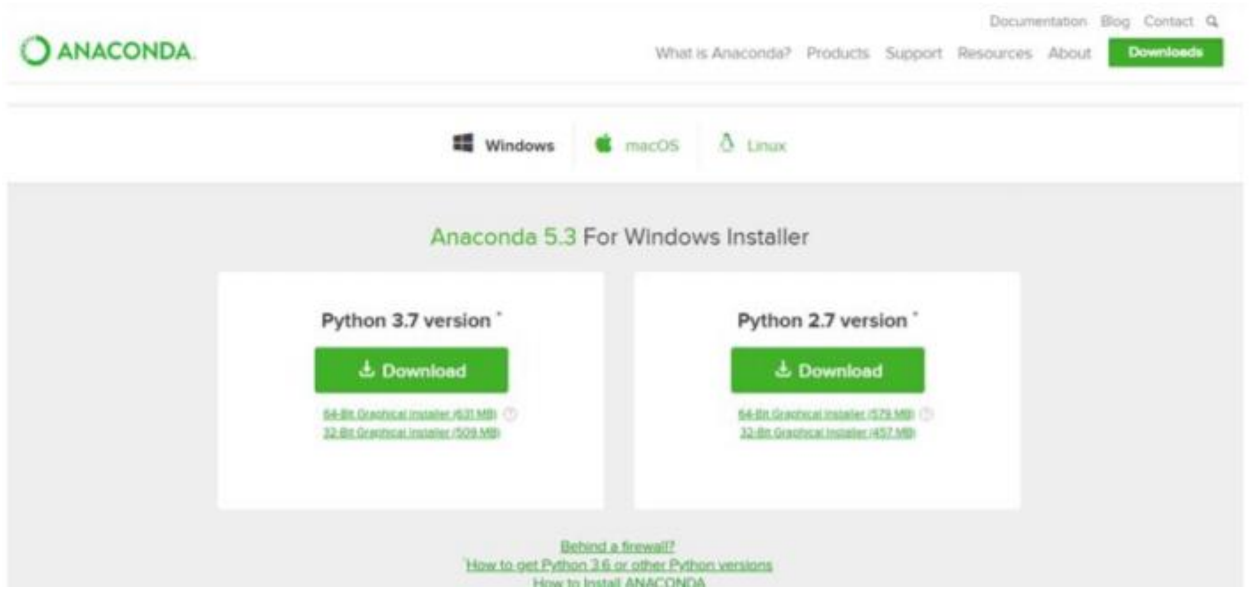
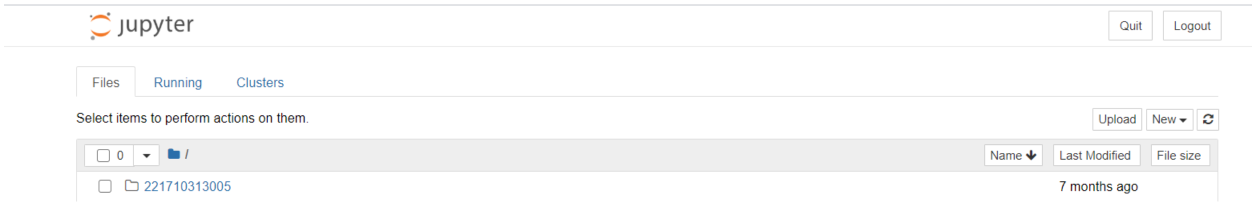

ANACONDA
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

Fig 2.2.2 : Anaconda download



Select items to perform actions on them.

221710313005 7 months ago

Fig 2.2.3 : Jupyter notebook

2.2.4 PYTHON VARIABLE TYPES:

● Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

● Variables are nothing but reserved memory locations to store values.

● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

o Numbers

o Strings

o Lists

o Tuples

o Dictionary

Python Numbers:

● Number data types store numeric values. Number objects are created when you assign a value to them.

● Python supports four different numerical types – int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

Python Strings:

● Strings in Python are identified as a contiguous set of characters represented in the quotation marks.

● Python allows for either pairs of single or double quotes.

● Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

● The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

Python Lists:

● Lists are the most versatile of Python's compound data types
. ● A list contains items separated by commas and enclosed within square brackets.

● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples:

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.2.5 PYTHON FUNCTION:

Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.2.6 PYTHON USING OOP's CONCEPTS:

Class:

● Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

● Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

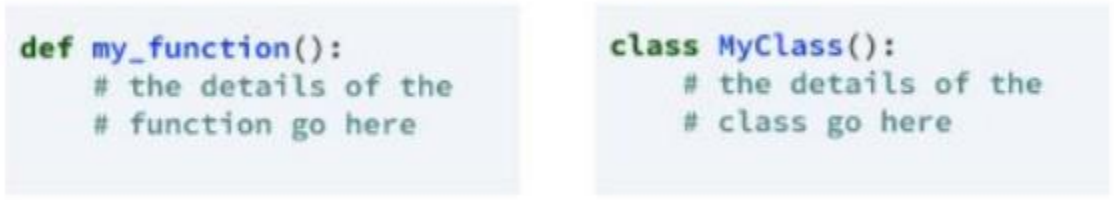
● Data member: A class variable or instance variable that holds data associated with a class and its objects.

● Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

● Defining a Class:

o We define a class in a very similar way how we define a function.

o Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is indented like a functions body is.



```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Fig 2.2.4 : Defining a Class

__init__ method in Class:

● The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 3
CASE STUDY

3.1 PROBLEM STATEMENT:

To find the donor for charity using Machine Learning algorithm.

3.2 Features

- ***age: Age***
- ***work class: Working Class (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked)***

- ***education level: Level of Education (Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-Adm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool)***
- ***education-num: Number of educational years completed***
- ***marital-status: Marital status (Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse)***
- ***occupation: Work Occupation (Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces)***
- ***relationship: Relationship Status (Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried)***
- ***race: Race (White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black)***
- ***sex: Sex (Female, Male)***
- ***capital-gain: Monetary Capital Gains***
- ***capital-loss: Monetary Capital Losses***
- ***hours-per-week: Average Hours Per Week Worked***
- ***native-country: Native Country (United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc.), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holland-Netherlands)***

3.3 Target Variable

income: Income Class (<=50K, >50K)

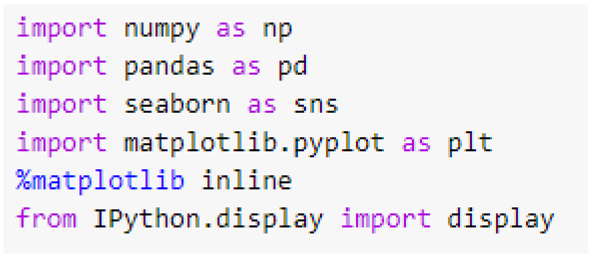
CHAPTER 4

PROJECT NAME(INFORMATION ABOUT THE PROJECT)

4.1 PROJECT REQUIREMENTS

4.1.1 Packages Used

The packages used are:



```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from IPython.display import display
```

Fig 4.1.1 importing packages

4.1.2 Algorithms Used

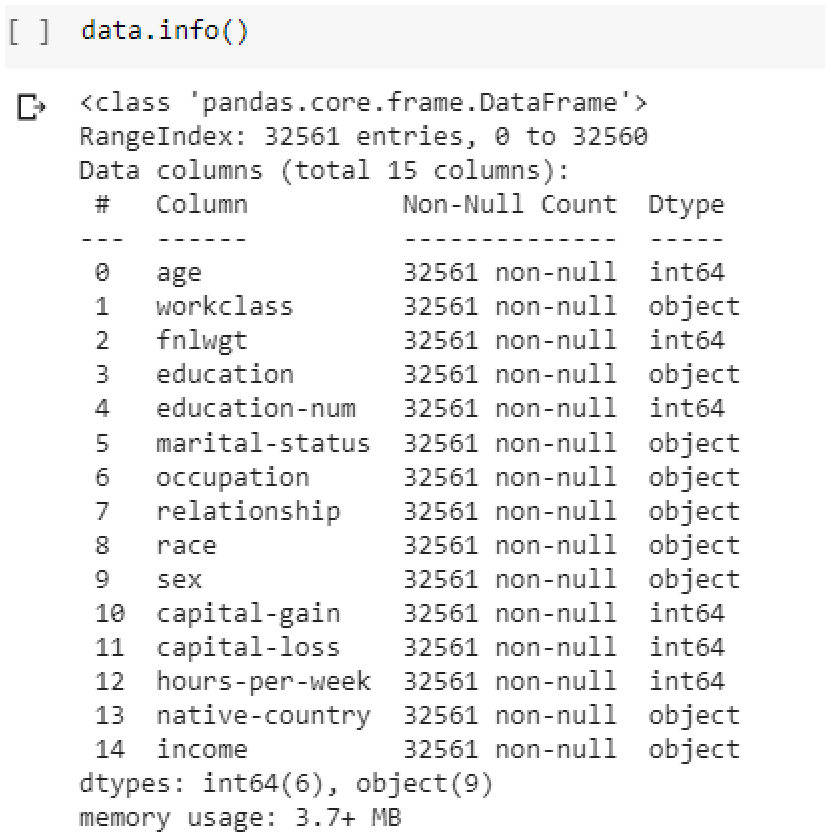
- Naive Bayes Classifier
- Logistic Regression
- RandomForest Classifier

4.2 PROBLEM STATEMENT

The problem we are going to solve here is the classification of a finding donors taking into consideration the features of the dataset donors _data.

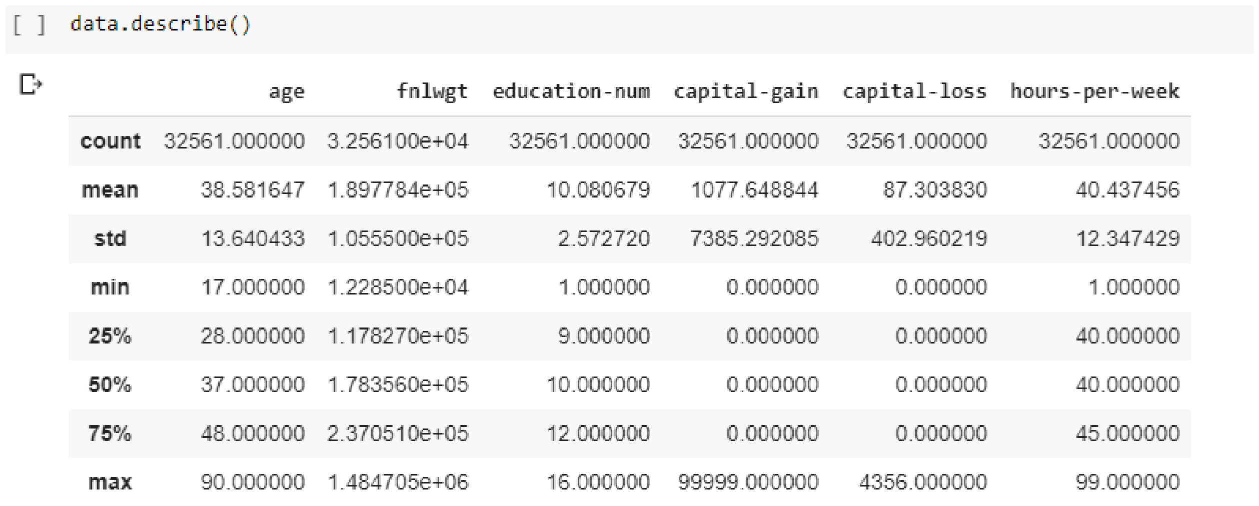
4.3 DATASET DESCRIPTION

∉ **I'll just take a glimpse of the data using the describe() and info() methods before I actually start processing and visualizing it.**



```
[ ]  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```



```
[ ]  data.describe()
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Fig 4.3 dataset description

4.4 OBJECTIVE OF CASE STUDY

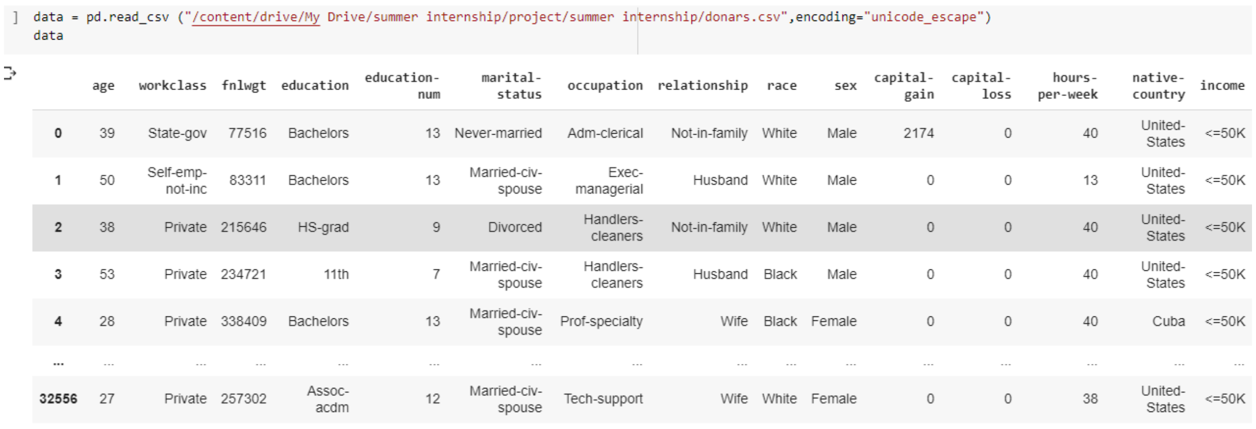
To get a better understanding of whether to finding donars by considering the features of the data and provide the client with desired results**.**

CHAPTER 5

DATA PREPROCESSING

5.1 READING THE DATASET

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a Data Frame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the data frame.

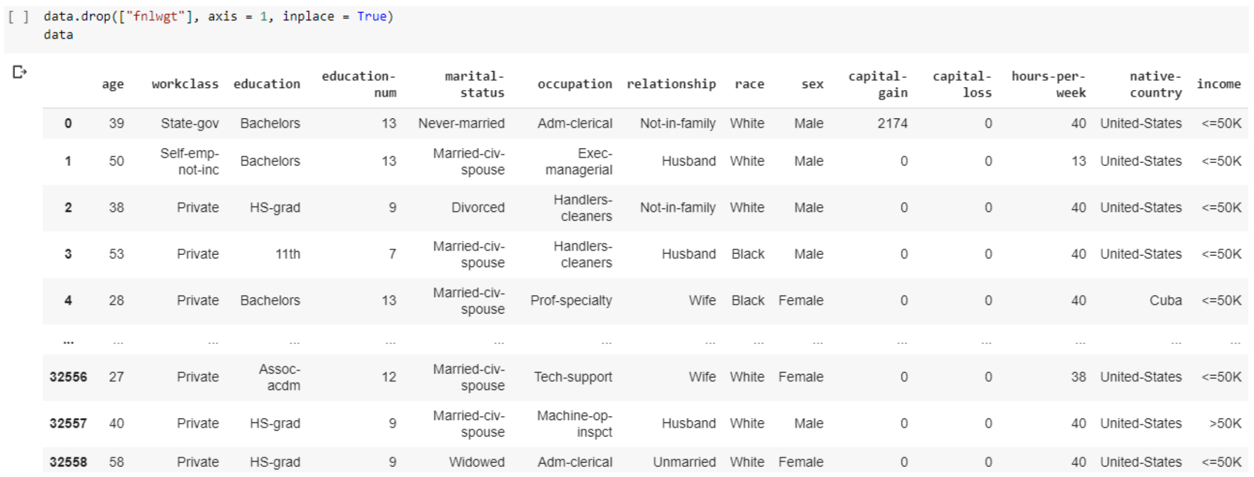


```
data = pd.read_csv ("/content/drive/My Drive/summer internship/project/summer internship/donars.csv",encoding="unicode_escape")
data
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K

Fig 5.1 reading data

- **The data we investigate here consists of small changes to the original dataset, such as removing the 'fnlwgt' feature and records with missing or ill-formatted entries.**



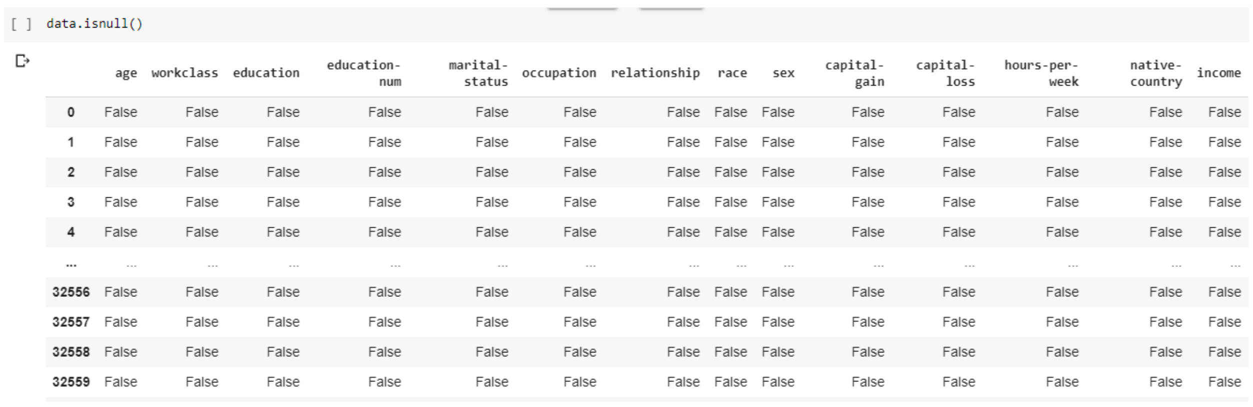
```
[ ] data.drop(["fnlwgt"], axis = 1, inplace = True)
    data
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32557	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
32558	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K

Fig 5.1.2 'fnlwgt'

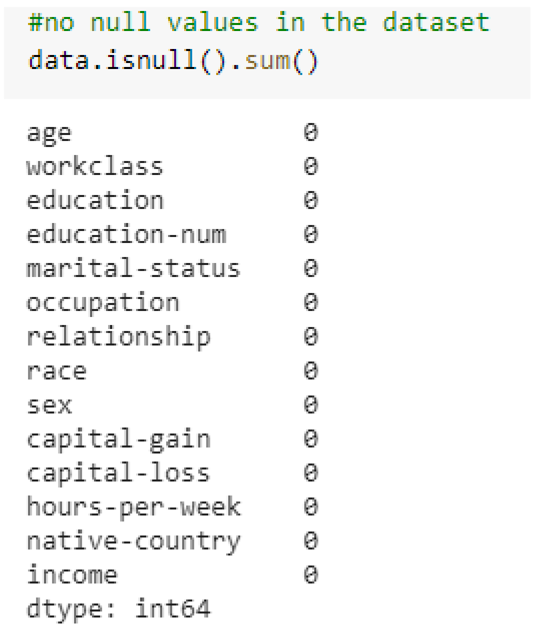
5.2 HANDLING MISSING VALUES AND DUPLICATE VALUES

We can find the number of missing values and duplicated values in each column using isnull() and duplicated() functions respectively. For our dataset no missing values or duplicated values were found.



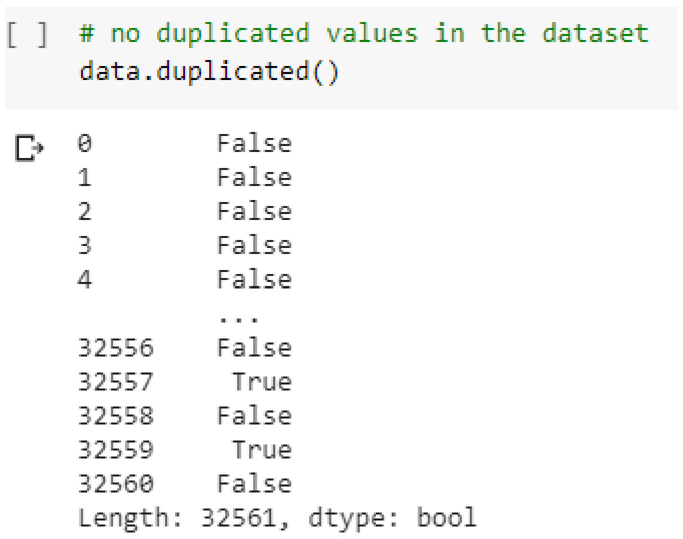
```
[ ] data.isnull()
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	False	False	False	False	False	False	False	False	False	False	False	False	False	False
32557	False	False	False	False	False	False	False	False	False	False	False	False	False	False
32558	False	False	False	False	False	False	False	False	False	False	False	False	False	False
32559	False	False	False	False	False	False	False	False	False	False	False	False	False	False



```
#no null values in the dataset
data.isnull().sum()
```

```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

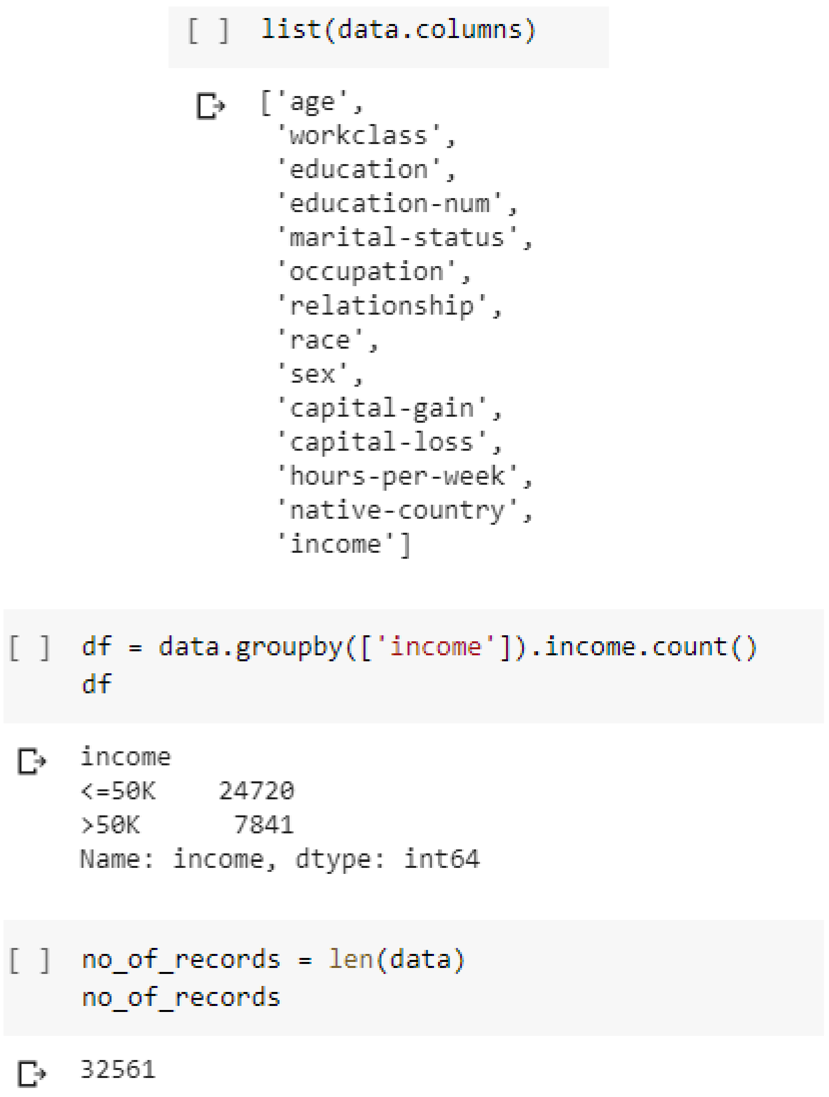


```
[ ] # no duplicated values in the dataset
    data.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
32556    False
32557     True
32558    False
32559     True
32560    False
Length: 32561, dtype: bool
```

Fig 5.2 handling missing values and duplicates

5.3 ENCODING CATEGORICAL DATA



```
[ ]  list(data.columns)
```

```
['age',
 'workclass',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country',
 'income']
```

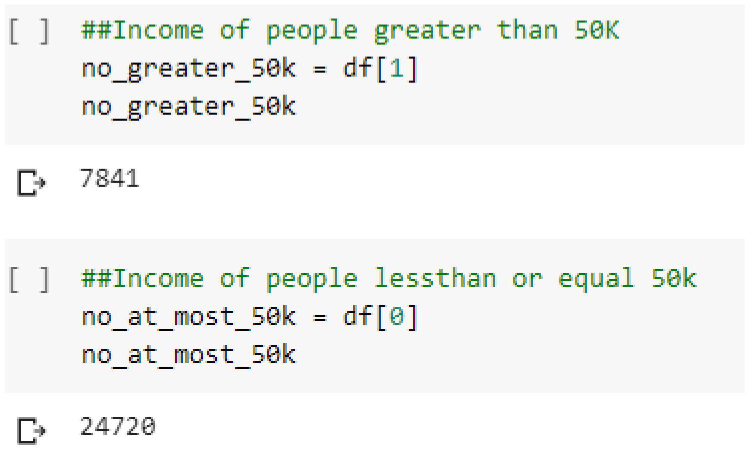
```
[ ]  df = data.groupby(['income']).income.count()
     df
```

```
income
<=50K    24720
>50K      7841
Name: income, dtype: int64
```

```
[ ]  no_of_records = len(data)
     no_of_records
```

```
32561
```

Fig 5.3 encoding categorical data



```
[ ]  ##Income of people greater than 50K
     no_greater_50k = df[1]
     no_greater_50k
```

```
7841
```

```
[ ]  ##Income of people lessthan or equal 50k
     no_at_most_50k = df[0]
     no_at_most_50k
```

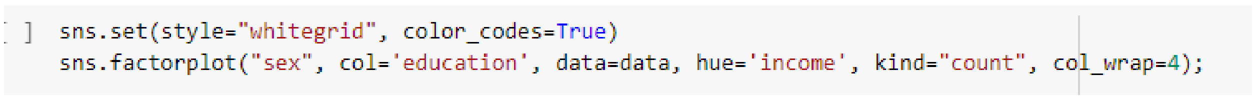
```
24720
```

Fig 5.4 showing > and < income of people

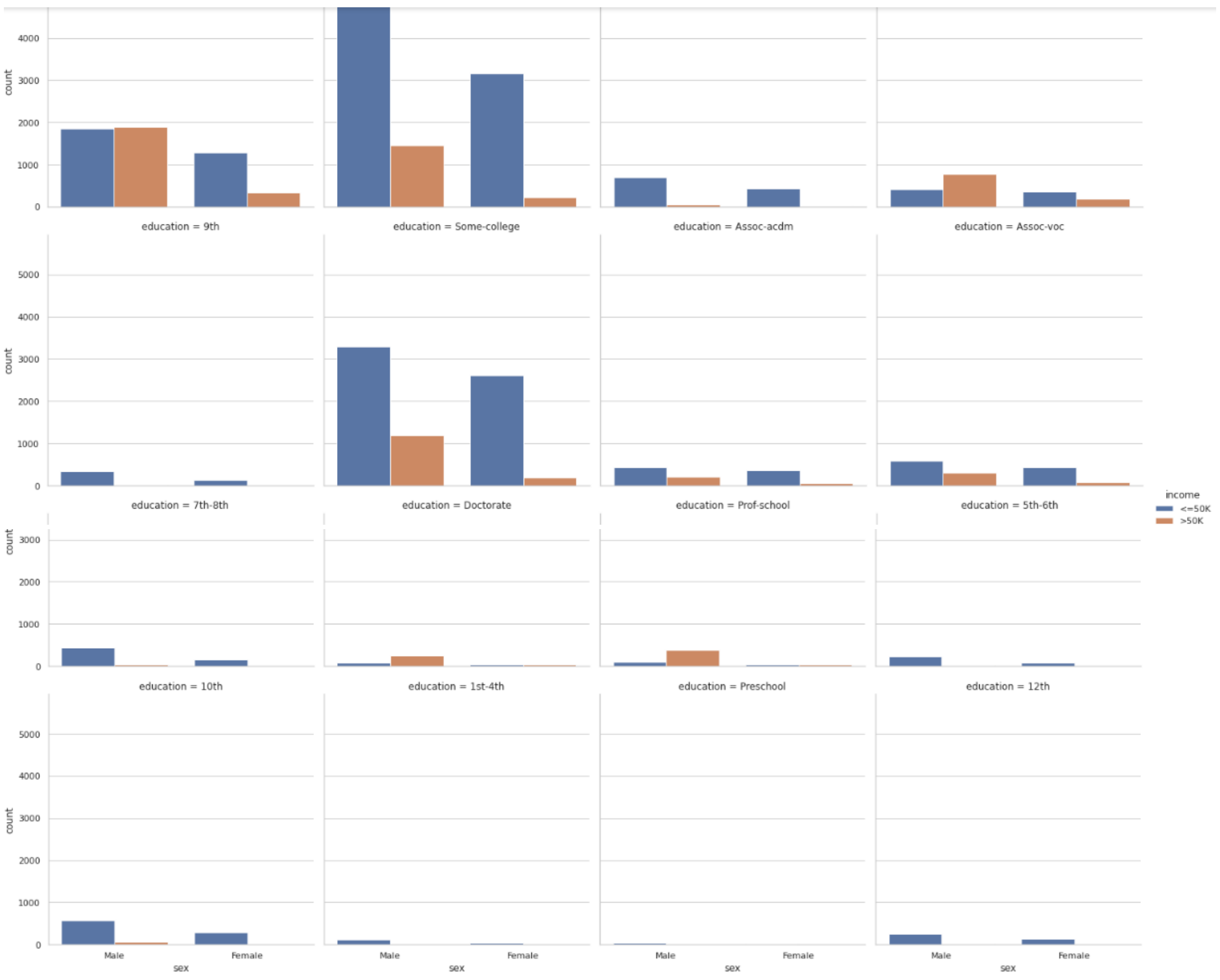
CHAPTER 6

FEATURE SELECTION

6.1 To perform log transformations to normalize its distribution.

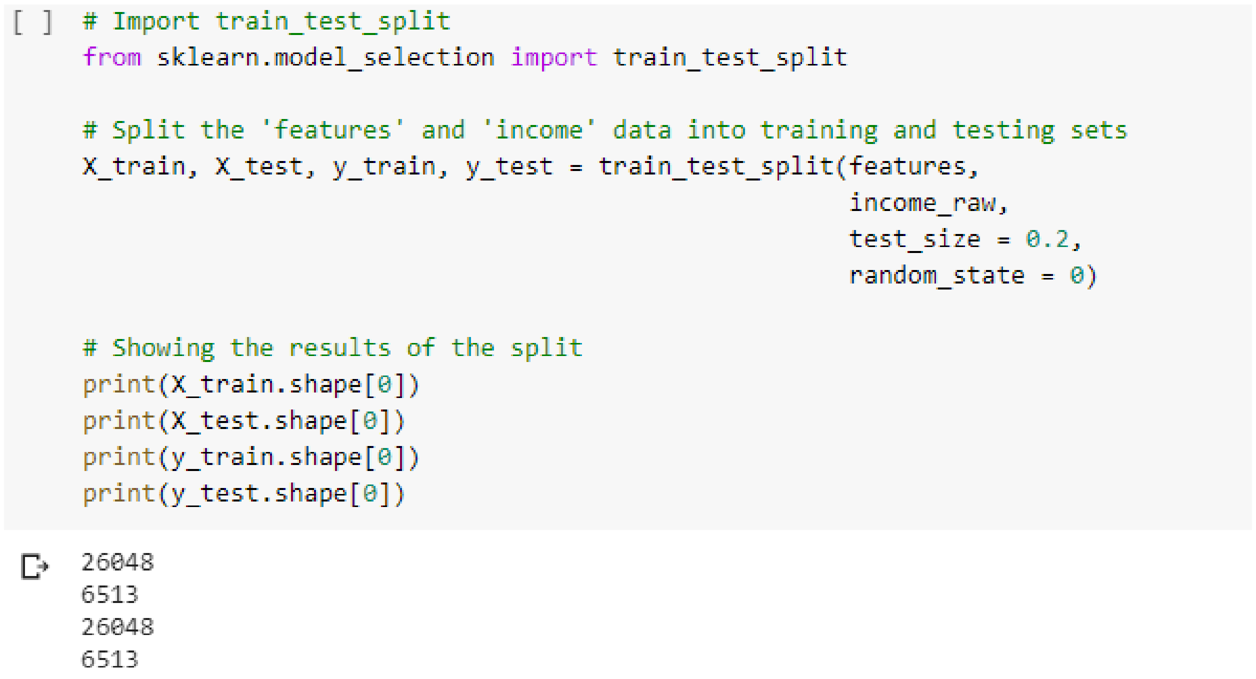


```
[ ]  sns.set(style="whitegrid", color_codes=True)
     sns.factorplot("sex", col='education', data=data, hue='income', kind="count", col_wrap=4);
```



Taking a look at the above graphs , I can see that each feature has a different range of distribution. Thus, using scaling before our predictions should be of great use. Also, the categorical features do stand out.

6.2 TRAIN-TEST-SPLIT



```
[ ]  # Import train_test_split
     from sklearn.model_selection import train_test_split

     # Split the 'features' and 'income' data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(features,
                                                         income_raw,
                                                         test_size = 0.2,
                                                         random_state = 0)

     # Showing the results of the split
     print(X_train.shape[0])
     print(X_test.shape[0])
     print(y_train.shape[0])
     print(y_test.shape[0])
```

```
26048
6513
26048
6513
```

Fig 6.2.1. Splitting the data

CHAPTER 7

MODEL BUILDING AND EVALUATION

Approach I : Naive Bayes Classifier

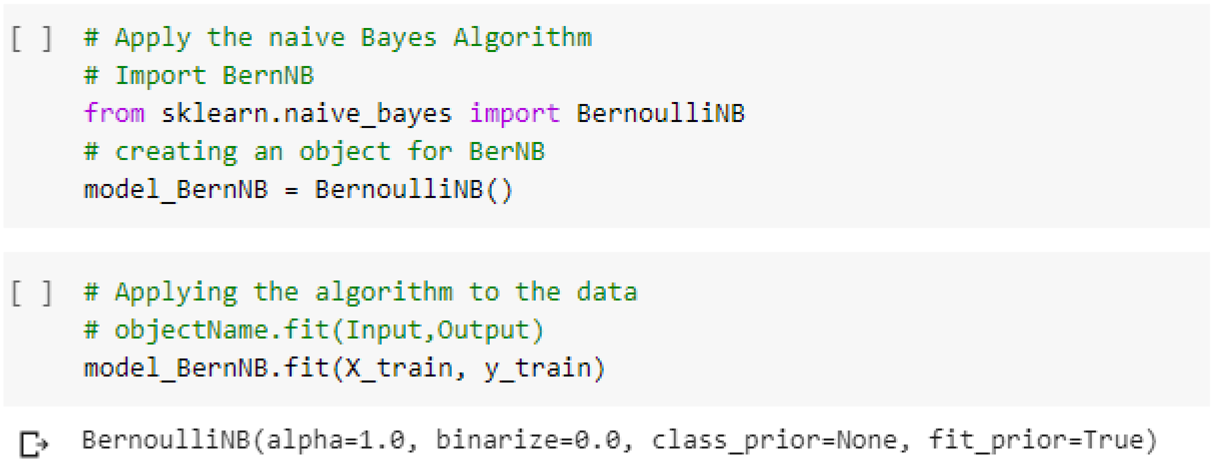
7.1.1 Brief about the algorithms used

In statistics, Naïve Bayes classifiers are a family of simple "probabilistic classifier" based on applying Bayes’ theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernal density estimation and achieve higher accuracy levels.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. maximum-liklihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a Baeysian method.

7.1.2. Apply naive Bayes algorithm to the data by importing BernoulliNB from sklearn.naive_bayes, then create an object and use the fit method on training data.



```
[ ] # Apply the naive Bayes Algorithm
    # Import BernNB
    from sklearn.naive_bayes import BernoulliNB
    # creating an object for BerNB
    model_BernNB = BernoulliNB()
```

```
[ ] # Applying the algorithm to the data
    # objectName.fit(Input,Output)
    model_BernNB.fit(X_train, y_train)
```

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Fig 7.1.2. applying naive bayes algorithm

7.1.3 Predicting on Train Data

We can predict the training data using predict function on X_train

```
[ ]  y_train_pred = model_BernNB.predict(X_train)
```

```
[ ]  # compare the actual values(y_test) with predicted values(y_test_pred)
     from sklearn.metrics import confusion_matrix,classification_report
     confusion_matrix(y_train,y_train_pred)

     array([[14826,  4976],
            [ 1297,  4949]])
```

Fig 7.1.3.a predicting on train data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heatmap

```
[ ]  from sklearn.metrics import confusion_matrix, accuracy_score
     conf = confusion_matrix(y_train, y_train_pred)
     conf
     sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})

     <matplotlib.axes._subplots.AxesSubplot at 0x7f826e862c88>
```

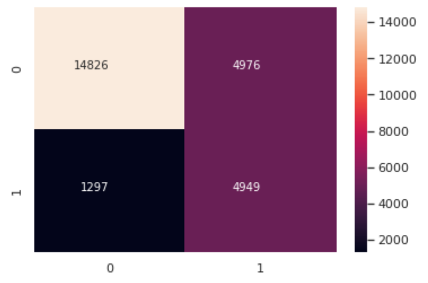
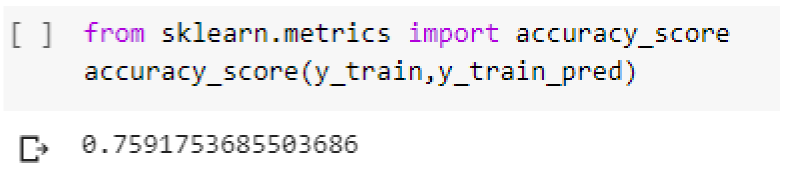


Fig 7.1.3.b confusion matrix visualization of train data

Print the classification report to know the accuracy score of training data

```
[ ]  print(classification_report(y_train,y_train_pred))

                  precision    recall  f1-score   support

            <=50K       0.92      0.75      0.83     19802
             >50K       0.50      0.79      0.61      6246

         accuracy                           0.76     26048
        macro avg       0.71      0.77      0.72     26048
     weighted avg       0.82      0.76      0.77     26048
```

Fig 7.1.3.c classification report for training data



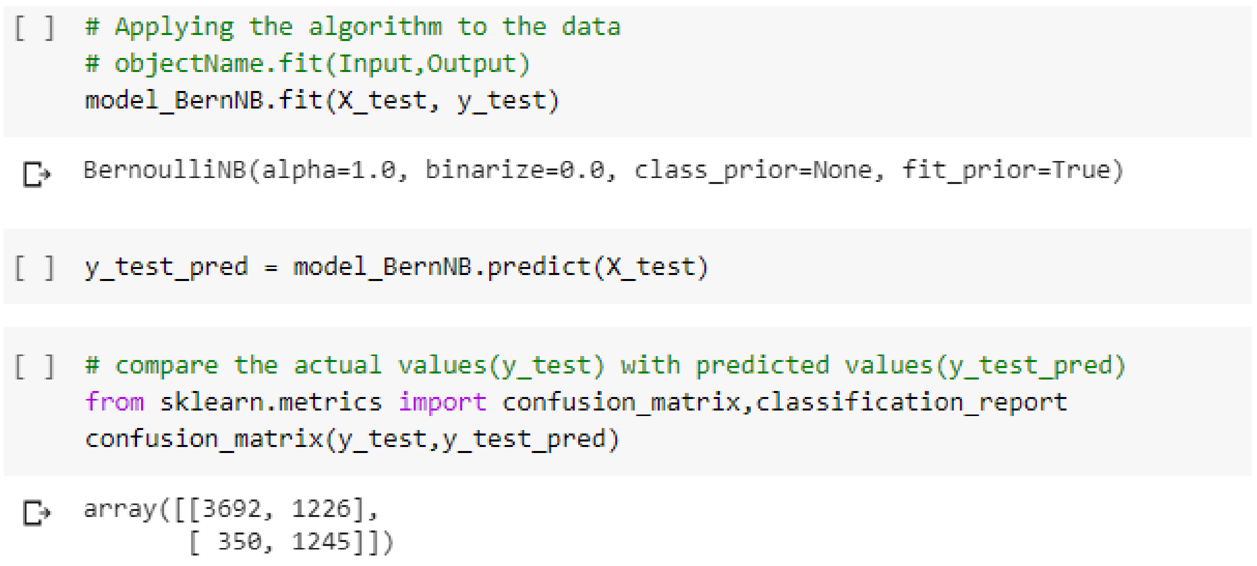
```
[ ]  from sklearn.metrics import accuracy_score
     accuracy_score(y_train,y_train_pred)

     0.7591753685503686
```

We got an accuracy score of about 71% to the training data which is considered to be a good score.

Now let’s check the same for test data

7.1.4 Predicting on Test Data

We can predict the testing data using predict function on X_test

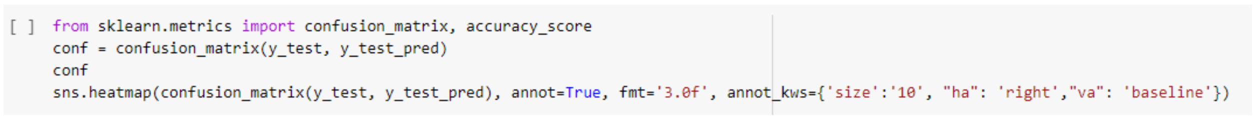


```
[ ]  # Applying the algorithm to the data
     # objectName.fit(Input,Output)
     model_BernNB.fit(X_test, y_test)

     BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)

[ ]  y_test_pred = model_BernNB.predict(X_test)

[ ]  # compare the actual values(y_test) with predicted values(y_test_pred)
     from sklearn.metrics import confusion_matrix,classification_report
     confusion_matrix(y_test,y_test_pred)

     array([[3692, 1226],
            [ 350, 1245]])
```

Fig 7.1.4.a predicting on test data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heatmap



```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    conf = confusion_matrix(y_test, y_test_pred)
    conf
    sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f826bc1add8>
```

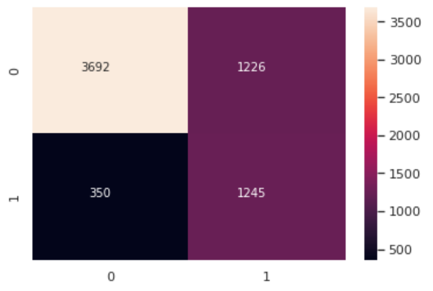
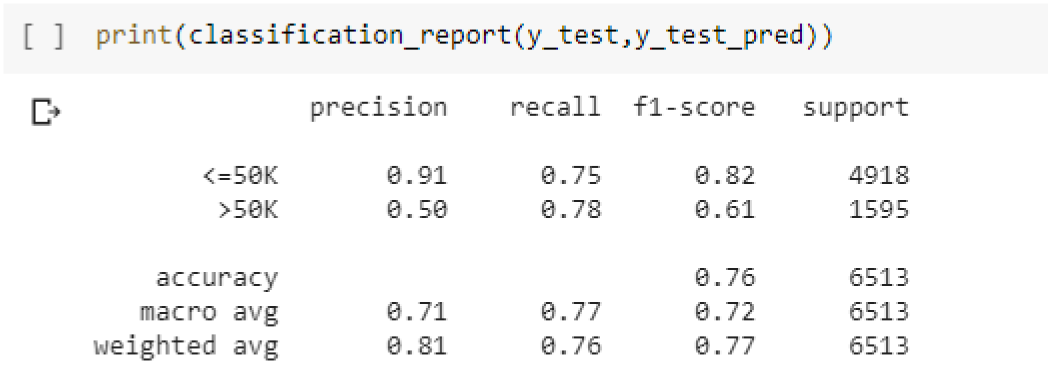


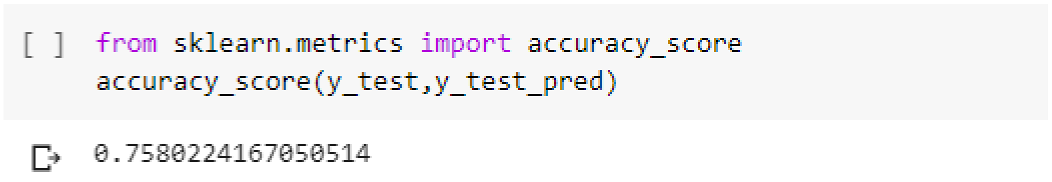
Fig 7.1.4.b confusion matrix visualization of test data

Print the classification report to know the accuracy score of training data



```
[ ] print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

        <=50K       0.91      0.75      0.82      4918
         >50K       0.50      0.78      0.61      1595

     accuracy                           0.76      6513
    macro avg       0.71      0.77      0.72      6513
 weighted avg       0.81      0.76      0.77      6513
```



```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_test,y_test_pred)
```

```
0.7580224167050514
```

Fig 7.1.4.c classification report for testing data(countvectorizer)

We got an accuracy score of around 71%, therefore we can say that it is a best fit and the model predicted very well.

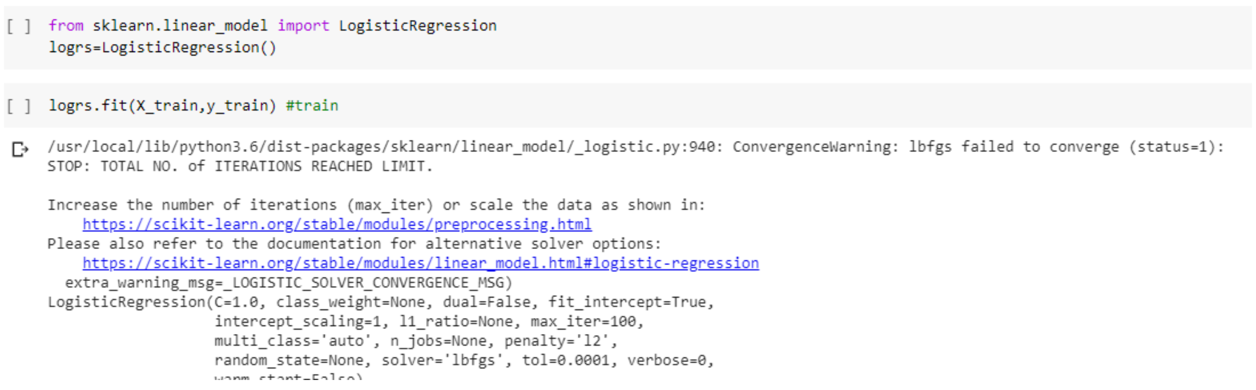
Approach II : Logistic Regression

7.2.1 Brief about the algorithms used

In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

7.2.2 Train the model

Import Logistic Regression from sklearn.linear_model and create an object. Fit the model on training data



```
[ ] from sklearn.linear_model import LogisticRegression
    logrs=LogisticRegression()
```

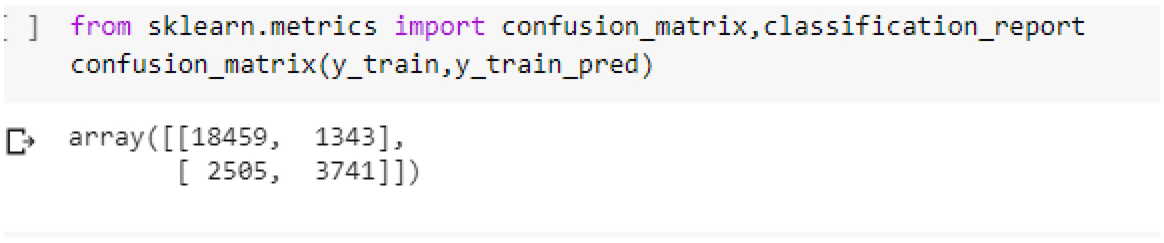
```
[ ] logrs.fit(X_train,y_train) #train
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig 7.2.2.importing logistic regression

7.2.3 Predicting on Train data

We can predict the training data using predict function on X_train



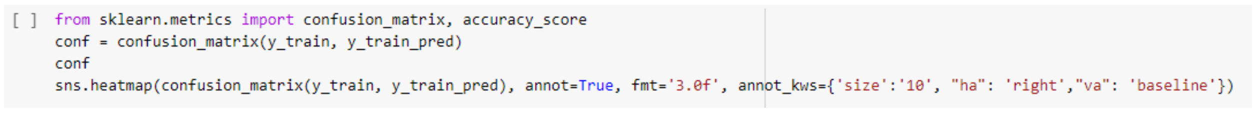
```
[ ] from sklearn.metrics import confusion_matrix,classification_report
    confusion_matrix(y_train,y_train_pred)
```

```
array([[18459,  1343],
       [ 2505,  3741]])
```

Fig 7.2.3.a predicting on train data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heat map



```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    conf = confusion_matrix(y_train, y_train_pred)
    conf
    sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82675dd0b8>
```

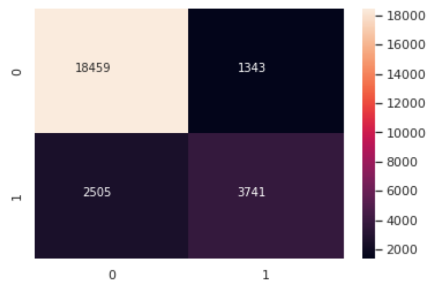
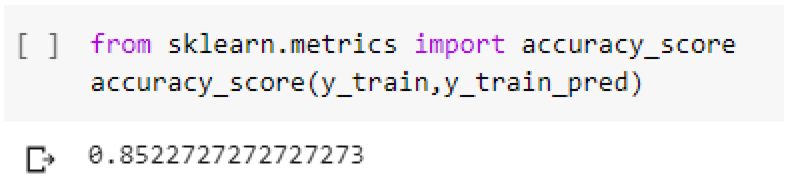


Fig 7.2.3.b visualizing confusion matrix of train data

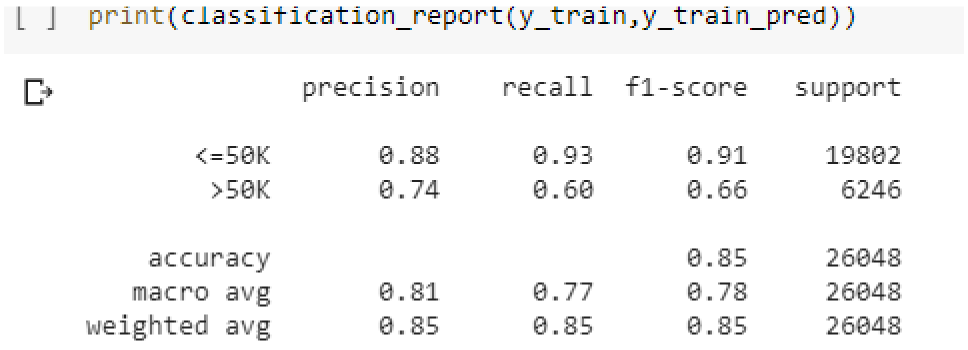
Find the accuracy score



```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_train,y_train_pred)

0.8522727272727273
```

Fig 7.2.3.c accuracy of train data

Print the classification report and check the accuracy of the training data



```
[ ] print(classification_report(y_train,y_train_pred))

              precision    recall  f1-score   support

       <=50K       0.88      0.93      0.91     19802
        >50K       0.74      0.60      0.66      6246

    accuracy                           0.85     26048
   macro avg       0.81      0.77      0.78     26048
weighted avg       0.85      0.85      0.85     26048
```

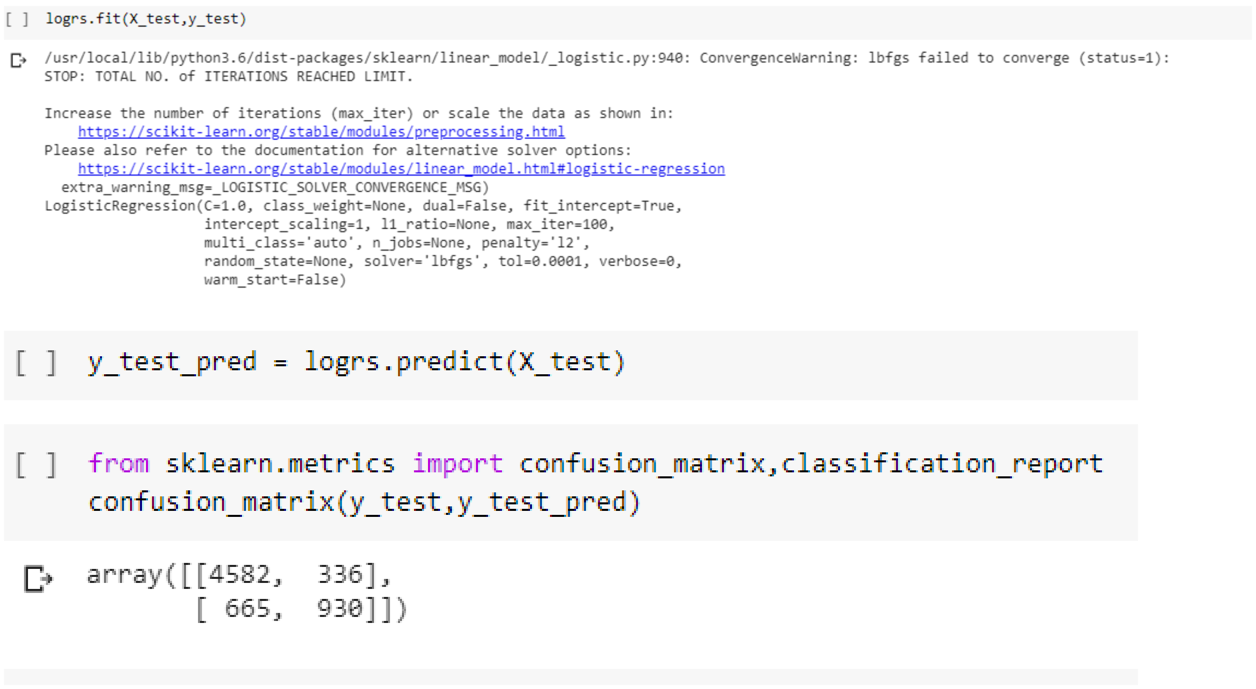
Fig 7.2.3.d classification report of train data

We got an accuracy score of around 81% for the training data which is considered to be a good score.

Now let’s check the same for test data.

7.2.4 Predicting on test data

We can predict the testing data using predict function on X_test



```
[ ]  logrs.fit(X_test,y_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
[ ]  y_test_pred = logrs.predict(X_test)
```

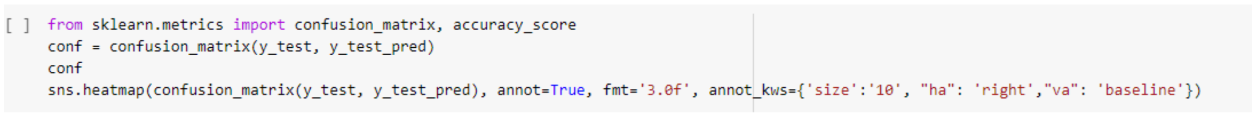
```
[ ]  from sklearn.metrics import confusion_matrix,classification_report
     confusion_matrix(y_test,y_test_pred)
```

```
array([[4582,  336],
       [ 665,  930]])
```

Fig 7.2.4.a predicting on test data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heat map



```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    conf = confusion_matrix(y_test, y_test_pred)
    conf
    sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f826e6fc198>
```

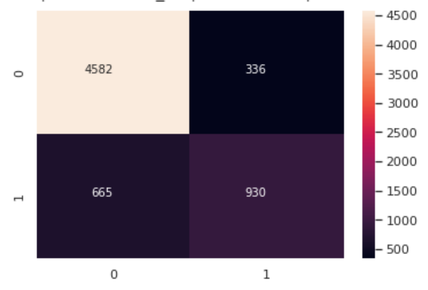
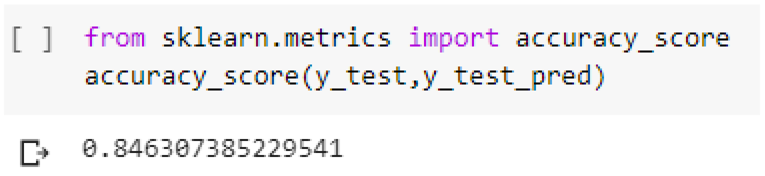


Fig 7.2.4.b visualizing confusion matrix of test data

Find the accuracy score

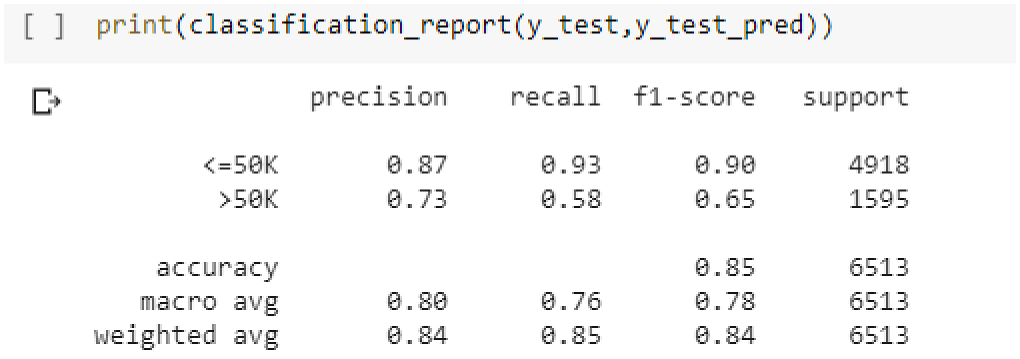


```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_test,y_test_pred)
```

```
0.846307385229541
```

Fig 7.2.4.c accuracy score of test data

Print the classification report and check the accuracy of the training data



```
[ ] print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

       <=50K       0.87      0.93      0.90      4918
        >50K       0.73      0.58      0.65      1595

    accuracy                           0.85      6513
   macro avg       0.80      0.76      0.78      6513
weighted avg       0.84      0.85      0.84      6513
```

Fig 7.2.4.d classification report of testing data

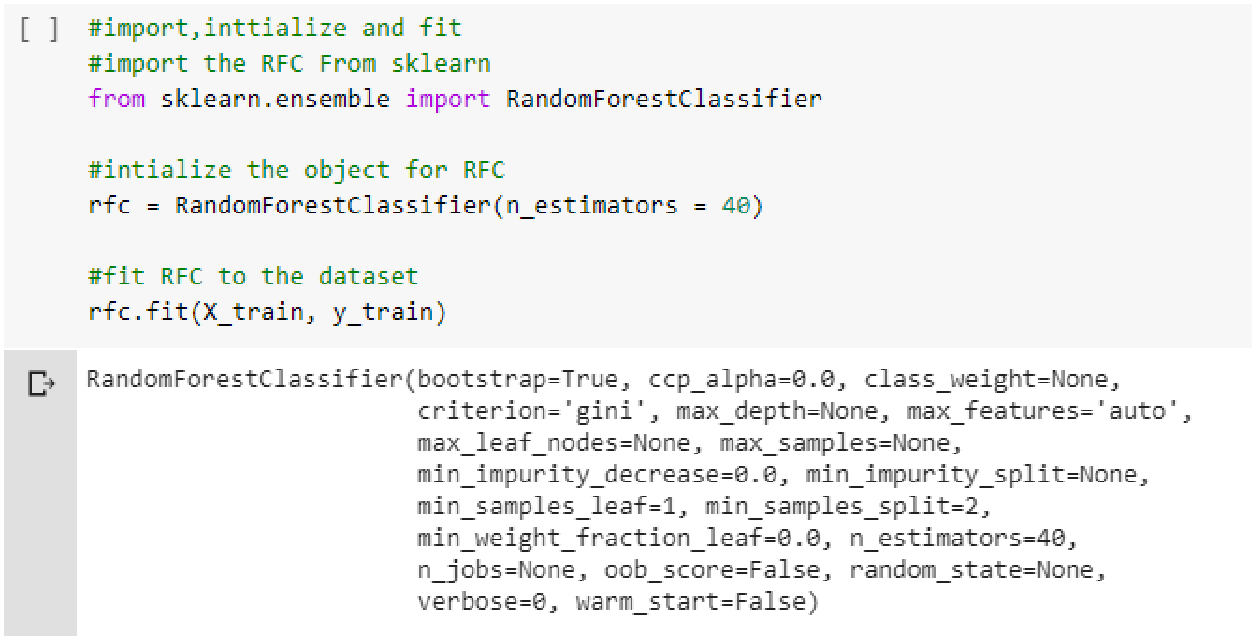
We got an accuracy score of around 80%, therefore we can say that it is a best fit and the model predicted very well.

Approach 3 : Random forest

7.3.1 Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).

Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:
A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.
The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:
There needs to be some actual signal in our features so that models built using those features do better than random guessing.
The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

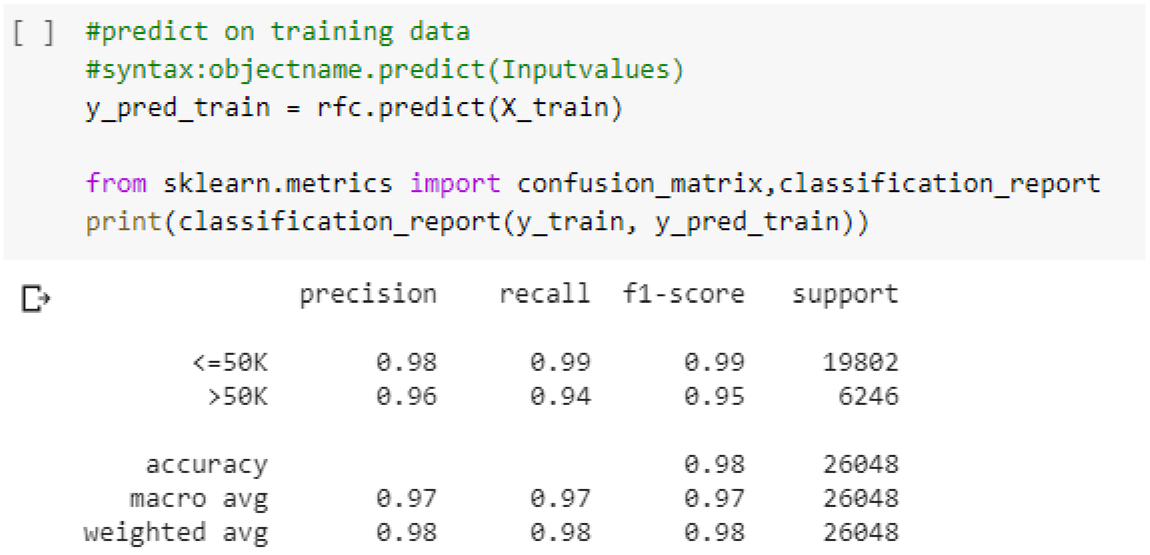


```
#import,inttialize and fit
#import the RFC From sklearn
from sklearn.ensemble import RandomForestClassifier

#intialize the object for RFC
rfc = RandomForestClassifier(n_estimators = 40)

#fit RFC to the dataset
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=40,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Fig 7.3.2.importing random forest classifier

7.3.3 Predicting on Train data

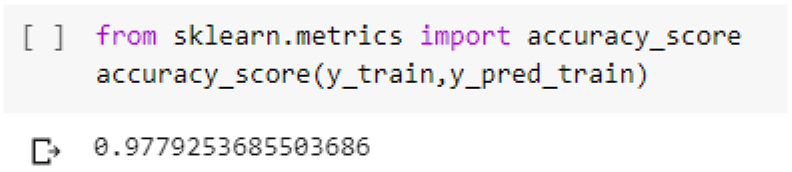


```
[ ]  #predict on training data
     #syntax:objectname.predict(Inputvalues)
     y_pred_train = rfc.predict(X_train)

     from sklearn.metrics import confusion_matrix,classification_report
     print(classification_report(y_train, y_pred_train))
```

```
              precision    recall  f1-score   support

        <=50K       0.98      0.99      0.99     19802
         >50K       0.96      0.94      0.95      6246

     accuracy                           0.98     26048
    macro avg       0.97      0.97      0.97     26048
 weighted avg       0.98      0.98      0.98     26048
```

Fig 7.3.3.a predicting on train data

Find the accuracy score

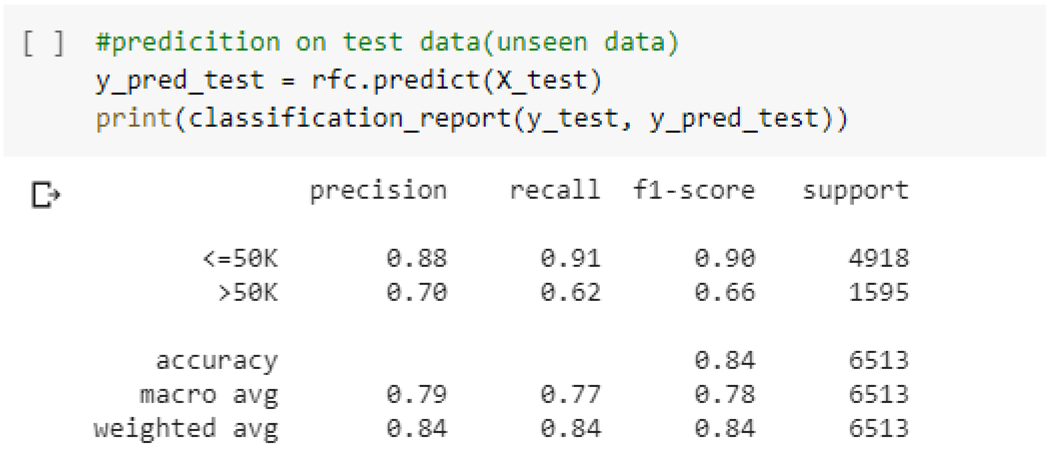


```
[ ]  from sklearn.metrics import accuracy_score
     accuracy_score(y_train,y_pred_train)
```

```
0.9779253685503686
```

Fig 7.3.3.b accuracy of train data

7.3.4 Predicting on test data

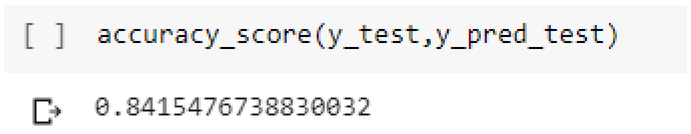


```
[ ] #predicition on test data(unseen data)
    y_pred_test = rfc.predict(X_test)
    print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

        <=50K       0.88      0.91      0.90      4918
         >50K       0.70      0.62      0.66      1595

     accuracy                           0.84      6513
    macro avg       0.79      0.77      0.78      6513
 weighted avg       0.84      0.84      0.84      6513
```

Fig 7.3.4.a predicting on test data

Find the accuracy score



```
[ ] accuracy_score(y_test,y_pred_test)
```

```
0.8415476738830032
```

Fig 7.3.4.c accuracy score of test data

8. CONCLUSION:

Comparing the three models we can clearly say that random forest classifier performed well with an accuracy of around 79-97% compared to others algorithms

9. REFERENCES:

1. https://en.wikipedia.org/wiki/Machine_learning
2. https://sajalsharma.com/portfolio/finding_donors
3. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
4. https://en.wikipedia.org/wiki/Logistic_regression
5. https://en.wikipedia.org/wiki/Random_forest