IDS PROJECT REPORT

"PREDICTING WHETHER THE INCOME EXCEEDS A GIVEN INCOME BASED ON CENSUS DATA"

Team Members:

- 20UCC087 - Rahul Vijayvargiya
- 20UCS034 - Aryan Gupta
- 20UCC091 - Sahil Yadav
- 20UCC118 - Vivek Singh

GitHub REPOSITORY:

View Here – GitHub repository for this project

Course Instructors:

- Dr. Alok Dutta.
- Dr. Subrat Dash.
- Dr. Sakhti Balan.

Department of Computer Science Engineering The LNM Institute of Information Technology

Table of Contents

Problem Statement:

Performing Data Pre-processing and preliminary analysis to predict whether the income of an adult will exceed 50k per year or not by developing a supervised machine learning model.

We need to collect a dataset from the given website and perform the following steps:

1. Data pre-processing and its visualization.
2. Explain all the inferences we got from our data.
3. Explain what ML Classification Algorithms are being used and Why?
4. Implementing those algorithms.
5. Output the result of the testing set and its visualization.

All the tasks are performed with the help of pre-existing Python Libraries such as:

- scikit_learn
- matplotlib
- seaborn
- numpy
- pandas

Attributes involved:

Age.
Work class.
Final Weight.
Education.
Education Number of Years.
Marital-status.
Occupation.
Relationship.
Race.
Sex.
Capital-gain.
Capital-loss.
Hours-per-week.
Native-country.

(All these attributes can be made clear by their names easily, so not going in deep to define them all.)

Approach:

We will follow a straightforward procedure. First, we'll read the dataset in a format which we'll use throughout the project, then separate the sentiments and statements. Then we looked at things like the number of statements in a particular category, whether there are any missing values and the relationship between the length of the statement and the sentiment. Then, after text pre-processing, the dataset is split into training and testing datasets, and various ML classification algorithms are used to evaluate their efficiency.

Introduction to Dataset:

We will use Jupyter Notebook for this

Dataset consists of:

- Dataset characteristics: Multivariate
- Area: Social
- Number of Instances: 32561
- Number of Attributes: 14
- Attributes Characteristic: Categorial, Integer
- Associated Task: Classification, Regression
- Missing Values: Yes
- Date Donated: 1996-05-01

Data Set: Also included in Repository. (Linked above)

Implementation:

We will use Jupyter Notebook for this project.

1. Importing Libraries and Loading Dataset :

(i) Importing Libraries:

Basically, we need to first import the libraries here.

- **pandas**: for manipulation and analysis of data.
- **NumPy**: contains a large collection of high-level mathematical functions to operate on large, arrays and matrices with multiple dimensions.
- **matplotlib**: for embedding plots.
- **seaborn**: for drawing attractive and informative statistical graphics.
- **SK-Learn:** for ML Classification Algorithms used in the project.
- **Sys and Warnings**: to ignore warnings.

In [1]:

```
#Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,plot_confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import warnings
warnings.simplefilter(action='ignore')
```

(ii) Reading Data Set:

a) Importing

In [2]:
```python
# Data Importing
df = pd.read_csv("adult.csv")
print(df.shape)
df.head()
```

(32561, 15)

Out[2]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-States	<=50K
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-States	<=50K
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States	<=50K
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K

b) Overall Description of the dataset

In [4]:
```python
df.describe()
```

Out[4]:

	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

2. Data Exploration:

Distribution of Data based on different Categorical Attributes:

a) For Work Class:

	Total count	Percentage
Private	22696	0.697030
Govt-Employees	4351	0.133626
Self-Employed	3657	0.112312
null	1836	0.056386
Without-pay	14	0.000430
Never-worked	7	0.000215

b) For Education:

	Total count	Percentage
High-School	13556	0.416326
Some-college	7291	0.223918
Bachelors	5355	0.164461
Masters	1723	0.052916
Assoc-voc	1382	0.042443
Elementary-School	1147	0.035226
Assoc-acdm	1067	0.032769
Prof-school	576	0.017690
Doctorate	413	0.012684
Preschool	51	0.001566

c) For Occupation:

	Total count	Percentage
Prof-specialty	4140	0.127146
Craft-repair	4099	0.125887

Exec-managerial	4066	0.124873
Adm-clerical	3770	0.115783
Sales	3650	0.112097
Other-service	3295	0.101195
Machine-op-inspct	2002	0.061485
null	1843	0.056601
Transport-moving	1597	0.049046
Handlers-cleaners	1370	0.042075
Farming-fishing	994	0.030527
Tech-support	928	0.028500
Protective-serv	649	0.019932
Priv-house-serv	149	0.004576
Armed-Forces	9	0.000276

d) <u>For relationship:</u>

	Total count	Percentage
Husband	13193	0.405178
Not-in-family	8305	0.255060
Own-child	5068	0.155646
Unmarried	3446	0.105832
Wife	1568	0.048156
Other-relative	981	0.030128

e) <u>For race:</u>

	Total count	Percentage
White	27816	0.854274
Black	3124	0.095943

Asian-Pac-Islander	1039	0.031909
Amer-Indian-Eskimo	311	0.009551
Other	271	0.008323

f) For marital status:

	Total count	Percentage
Married	15417	0.473481
Never-married	10683	0.328092
Separated	5468	0.167931
Widowed	993	0.030497

g) For sex:

	Total count	Percentage
Male	21790	0.669205
Female	10771	0.330795

h) For Native. Country:

	Total count	Percentage
United-States	29170	0.895857
Mexico	643	0.019748
null	583	0.017905
Philippines	198	0.006081
Germany	137	0.004207
Canada	121	0.003716
Puerto-Rico	114	0.003501
El-Salvador	106	0.003255

India	100	0.003071
Cuba	95	0.002918
England	90	0.002764
Jamaica	81	0.002488
South	80	0.002457
China	75	0.002303
Italy	73	0.002242
Dominican-Republic	70	0.002150
Vietnam	67	0.002058
Guatemala	64	0.001966
Japan	62	0.001904
Poland	60	0.001843
Columbia	59	0.001812
Taiwan	51	0.001566
Haiti	44	0.001351
Iran	43	0.001321
Portugal	37	0.001136
Nicaragua	34	0.001044
Peru	31	0.000952
Greece	29	0.000891
France	29	0.000891
Ecuador	28	0.000860
Ireland	24	0.000737
Hong	20	0.000614
Cambodia	19	0.000584
Trinadad&Tobago	19	0.000584
Laos	18	0.000553

Thailand	18	0.000553
Yugoslavia	16	0.000491
Outlying-US(Guam-USVI-etc)	14	0.000430
Hungary	13	0.000399
Honduras	13	0.000399
Scotland	12	0.000369
Holland-Netherlands	1	0.000031

i) <u>For Income:</u>

	Total count	Percentage
<=50K	24720	0.75919
>50K	7841	0.24081

<u>Process Involved:</u>

(i) <u>Converting missing value into the null values:</u>

Just for the easiness, we had replaced " ? " with null and then we cross checked it.

```
In [7]:
# to replace ? with null
change_cols = ['workclass', 'occupation', 'native.country']
for col in change_cols:
    df.loc[df[col] == '?', col] = 'null'
```

```
In [8]:
# cross check
for col in change_cols:
    print(f"? in {col}: {df[(df[col] == '?')].any().sum()}")
```

```
? in workclass: 0
? in occupation: 0
? in native.country: 0
```

(ii) Merging and replacing the attributes in the list:

(a) For Education: Merged the High School grad, 1st to 12th into Schooling, Replaced the bachelors to undergraduates and masters to postgraduates.

```
In [9]:
# merging and replacing elements in the list

school = ['HS-grad', '12th', '11th', '10th', '9th','1st-4th','5th-6th','7th-8th', 'Preschool']
df['education'].replace(to_replace = school, value = 'Schooling', inplace = True)
df['education'].replace(to_replace = ['Bachelors'], value = "Undergraduates", inplace = True)
df['education'].replace(to_replace = ['Masters'], value = "Post-Graduates", inplace = True)
df['education'].value_counts()
```

```
Out[9]: Schooling         14754
        Some-college       7291
        Undergraduates     5355
        Post-Graduates     1723
        Assoc-voc          1382
        Assoc-acdm         1067
        Prof-school         576
        Doctorate           413
        Name: education, dtype: int64
```

(b) For Marital Status: Merged the married-spouse-absent, married-civ-spouse, married-AF-spouse into married, separated and divorced into separated and replaced never-married into single.

```
In [10]:
married= ['Married-spouse-absent','Married-civ-spouse','Married-AF-spouse']
separated = ['Separated','Divorced']

df['marital.status'].replace(to_replace = married ,value = 'Married',inplace = True)
df['marital.status'].replace(to_replace = separated,value = 'Separated',inplace = True)
df['marital.status'].replace(to_replace = ['Never-married'], value = "Single", inplace = True)

df['marital.status'].value_counts()
```

```
Out[10]: Married      15417
         Single       10683
         Separated     5468
         Widowed        993
         Name: marital.status, dtype: int64
```

(c) For Work Class: Merged self-emp-not-inc and self-emp-inc into Self-Employed, merged Local-gov, State-gov and federal-gov into govt-employees and replaced never worked to unemployed

In [11]:

```python
self_employed = ['Self-emp-not-inc','Self-emp-inc']
govt_employees = ['Local-gov','State-gov','Federal-gov']

df['workclass'].replace(to_replace = self_employed ,value = 'Self-Employed',inplace = True)
df['workclass'].replace(to_replace = govt_employees,value = 'Govt-Employees',inplace = True)
df['workclass'].replace(to_replace = ['Never-worked'], value = 'Unemployed', inplace = True)

df['workclass'].value_counts()
```

Out[11]:

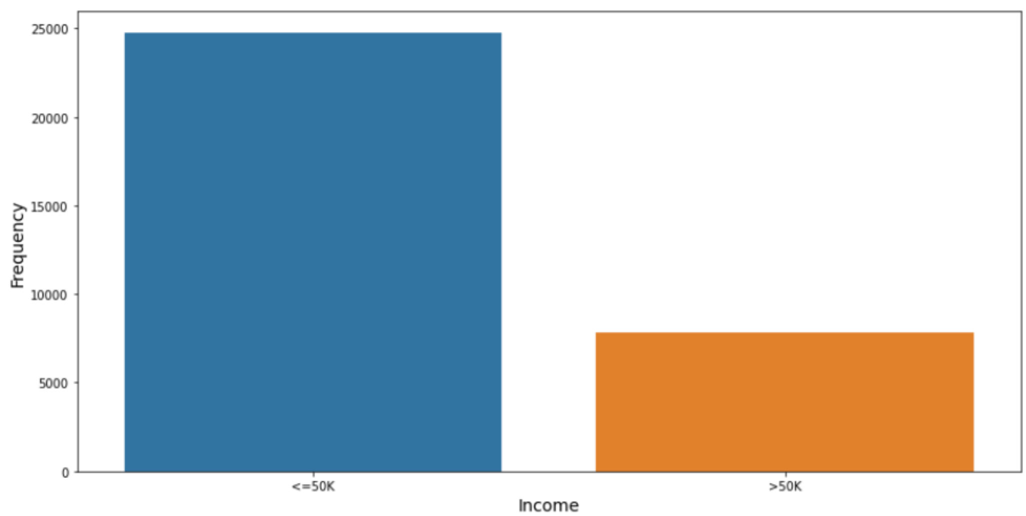
```
Private           22696
Govt-Employees     4351
Self-Employed      3657
null               1836
Without-pay          14
Unemployed            7
Name: workclass, dtype: int64
```

3. Data Visualization:

Here we would try to find out a relation between each column of the final numerical dataset and the target attribute income. This can be achieved by creating a Bar graph between income and each of the other columns.

(i) Checking the total number of people having income greater than 50K and less than or equal to 50K:

In [12]:
```
plt.figure(figsize=(14,7))
sns.countplot(x = 'income', data = df)
plt.xlabel('Income', fontsize = 14)
plt.ylabel('Frequency', fontsize = 14)
plt.show()
```



This means almost 75% of people having income less than or equal to 50K and almost 25% of people having income greater than 50K.

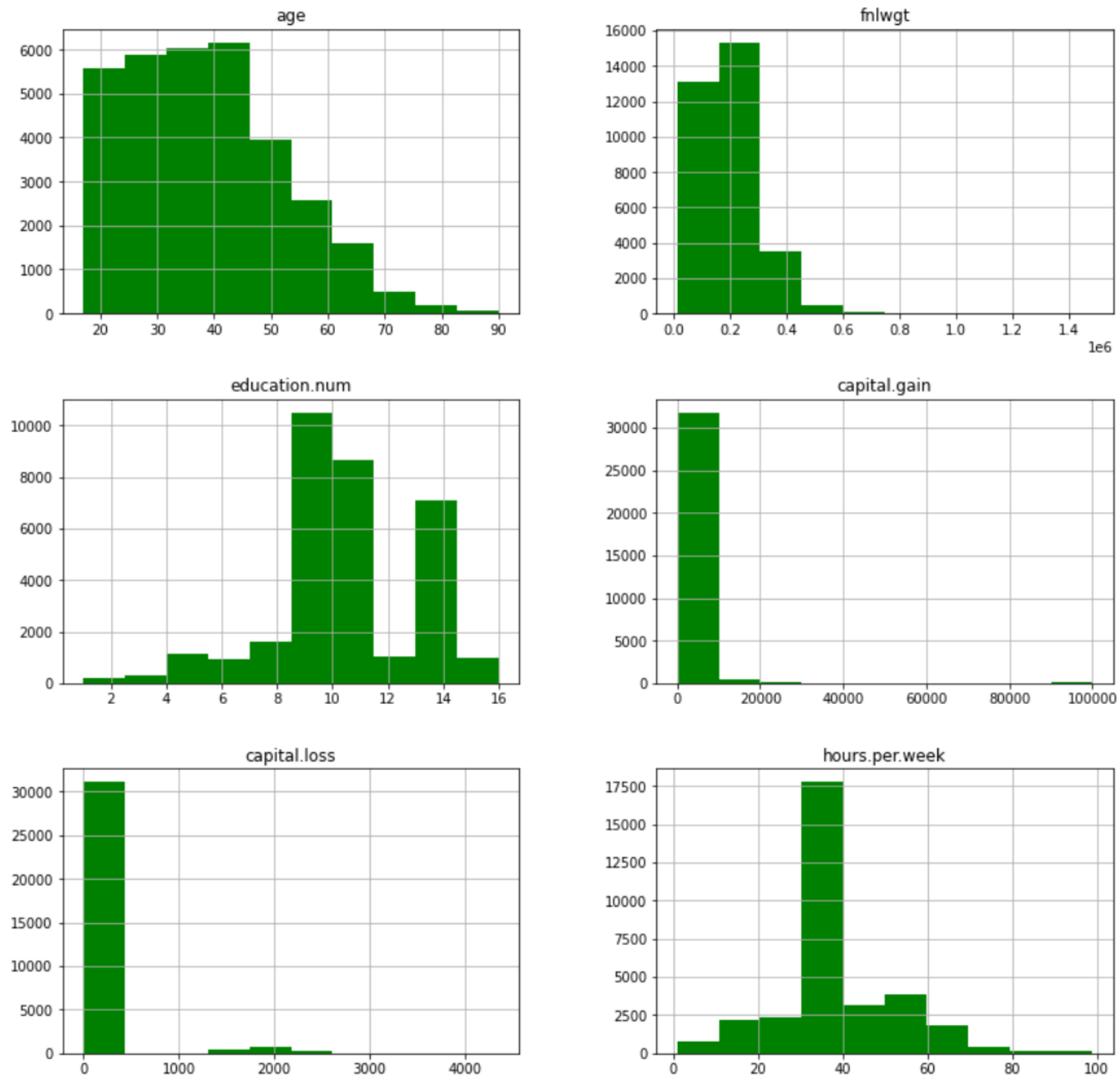
(ii) Exploring the distribution of population on the basis of numerical features:

a) Maximum people belong from the age group of 15-50

b) Generally, people work from 30-40 hours per week

c) Maximum number of people are qualified up to 8th Standard.

In [13]:
```
#distribution of population on the basis of numerical features
df[list(num_column.index)].hist(figsize = (14,14), color="green");
```



(iii) <u>Exploring the data based on Capital Gain and Capital Loss:</u>

a) 1519 people having capital loss above the median value which is almost 4.67%
b) 2712 people having capital gain above the median value which is almost 8.33%
c) Almost 92% of people having capital gain equals to zero.

Total Observations:

```
Number of observations having capital gain and capital loss zero: (28330, 15)
*************** workclass ***************
Private            19982
Govt-Employees      3714
Self-Employed       2960
null                1655
Without-pay           12
Unemployed             7
Name: workclass, dtype: int64


*************** education ***************
Schooling          13342
Some-college        6533
Undergraduates      4384
Post-Graduates      1300
Assoc-voc           1194
Assoc-acdm           930
Prof-school          363
Doctorate            284
Name: education, dtype: int64


*************** marital.status ***************
Married      12603
Single        9914
Separated     4934
Widowed        879
Name: marital.status, dtype: int64


*************** occupation ***************
Craft-repair         3593
Adm-clerical         3408
Prof-specialty       3290
Exec-managerial      3219
Sales                3138
Other-service        3122
Machine-op-inspct    1806
null                 1662
Transport-moving     1416
Handlers-cleaners    1274
Farming-fishing       890
Tech-support          795
Protective-serv       570
Priv-house-serv       139
Armed-Forces            8
Name: occupation, dtype: int64
```

```
Male      18551
Female     9779
Name: sex, dtype: int64


*************** native.country ***************
United-States                 25320
Mexico                          612
null                            493
Philippines                     174
Germany                         117
Puerto-Rico                     103
Canada                          103
El-Salvador                      95
Cuba                             85
India                            79
Jamaica                          78
England                          78
South                            68
Dominican-Republic               67
Italy                            65
China                            64
Guatemala                        60
Vietnam                          57
Columbia                         55
Poland                           53
Japan                            51
Taiwan                           44
Haiti                            42
Portugal                         35
Iran                             35
Nicaragua                        30
Peru                             29
France                           26
Ecuador                          25
Ireland                          21
Greece                           20
Hong                             19
Thailand                         18
Laos                             17
Trinadad&Tobago                  17
Yugoslavia                       15
Outlying-US(Guam-USVI-etc)       14
Cambodia                         14
Honduras                         12
Scotland                         11
Hungary                           9
Name: native.country, dtype: int64


*************** income ***************
<=50K    22939
>50K      5391
Name: income, dtype: int64
```

Details for the capital gain greater than zero:

In [17]:
```
df.loc[df['capital.gain'] > 0,:].describe()
```

Out[17]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	2712.000000	2.712000e+03	2712.000000	2712.000000	2712.0	2712.000000
mean	44.016224	1.880805e+05	11.066003	12938.541298	0.0	43.510324
std	13.268269	1.033775e+05	2.663273	22395.413530	0.0	12.207654
min	17.000000	1.930200e+04	1.000000	114.000000	0.0	1.000000
25%	35.000000	1.180670e+05	9.000000	3411.000000	0.0	40.000000
50%	43.000000	1.759390e+05	10.000000	7298.000000	0.0	40.000000
75%	52.000000	2.364735e+05	13.000000	14084.000000	0.0	50.000000
max	90.000000	1.033222e+06	16.000000	99999.000000	0.0	99.000000

The maximum capital gain is 99999 by 159 observations

In [19]:
```
print(f"Number of observations having capital gain of 99999:{df.loc[df['capital.gain'] == 99999,:].shape}")
print(f"Income counts: {df.loc[df['capital.gain'] == 99999,:]['income'].value_counts()}")
```

```
Number of observations having capital gain of 99999:(159, 15)
Income counts: >50K    159
Name: income, dtype: int64
```

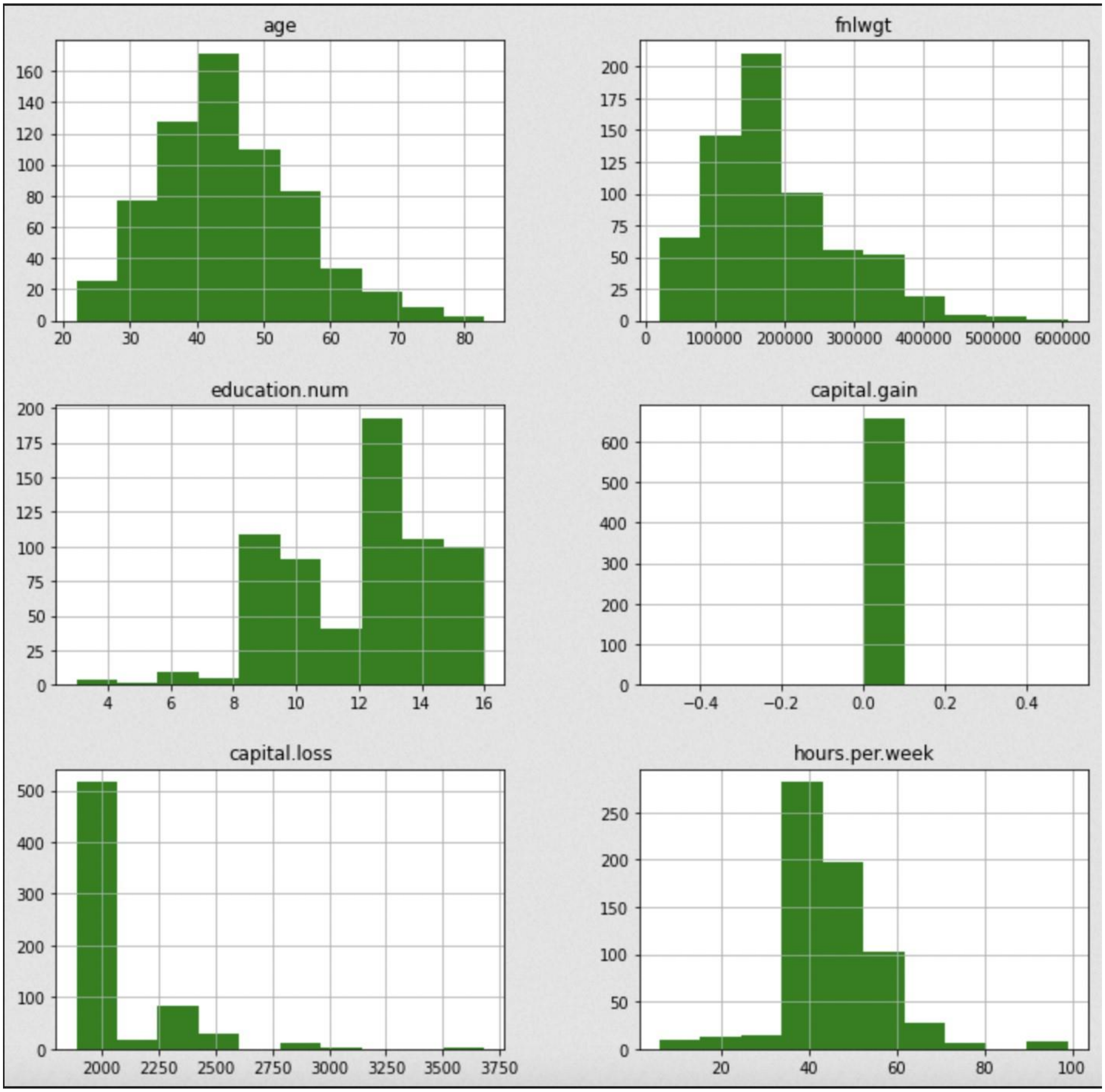
Details for the capital loss greater than zero:

The Maximum capital loss is 4356 by 3 observations

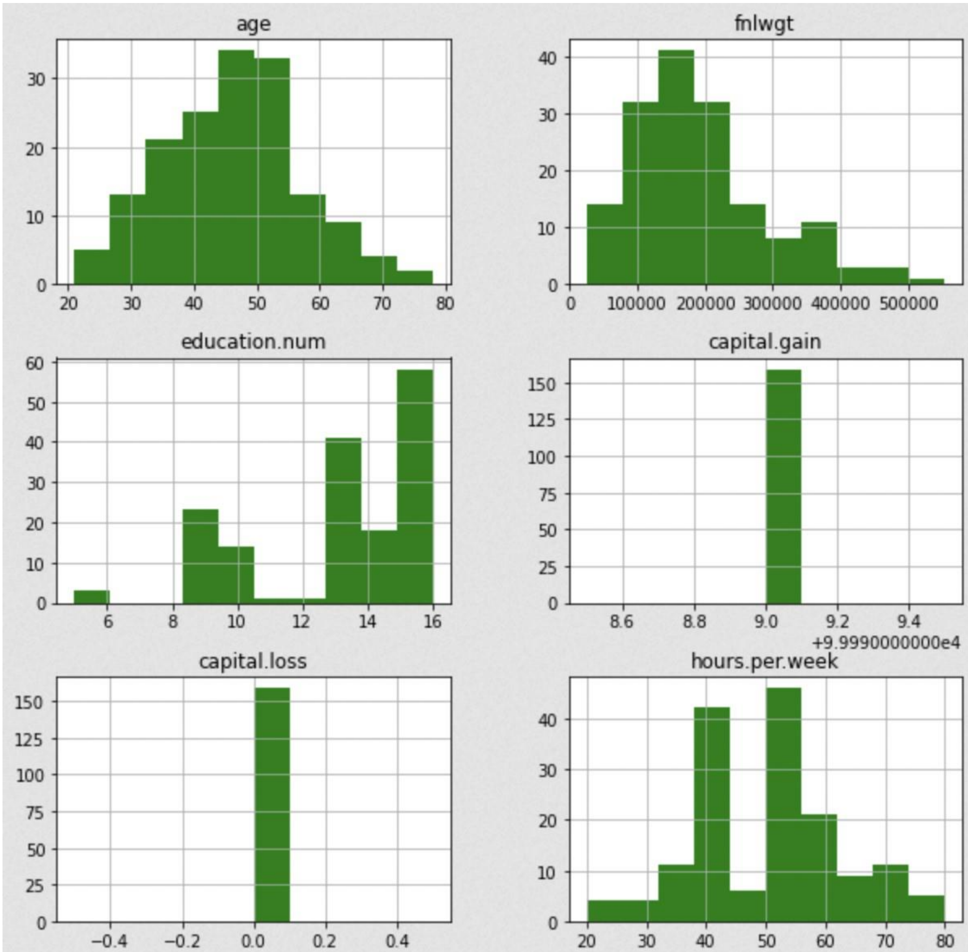
In [20]:
```
print(f"Number of observations having capital loss of 4356:{df.loc[df['capital.loss'] == 4356,:].shape}")
print(f"Income counts: {df.loc[df['capital.loss'] == 4356,:]['income'].value_counts()}")
```

```
Number of observations having capital loss of 4356:(3, 15)
Income counts: <=50K    3
Name: income, dtype: int64
```

Observation distribution among different fields when the capital loss is greater than or equal to the 1871 (mean) and their income is less than or equal to 50k



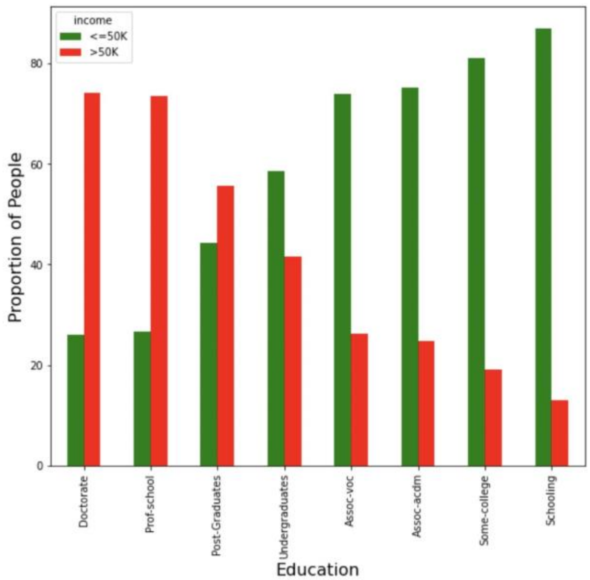
Observations distributed among different fields when the maximum capital gain is 99999

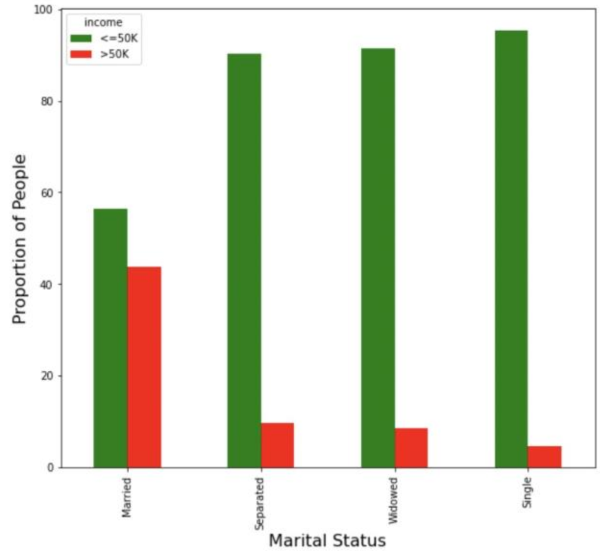


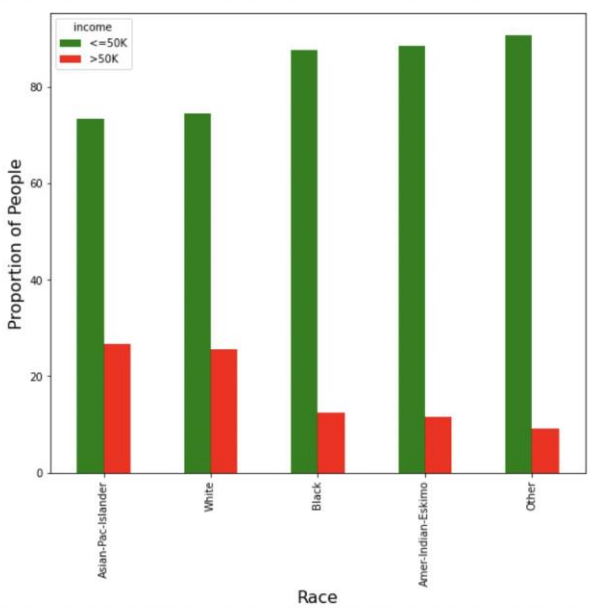
1. Maximum of observations have graduated from high schools

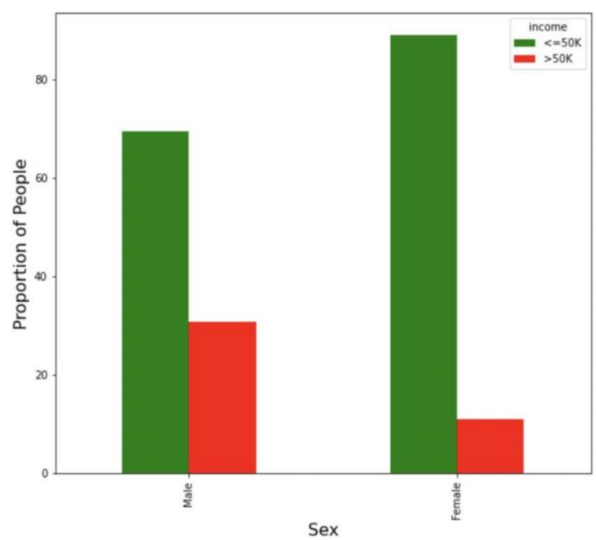
2. They generally worked for more than 50 hours per week

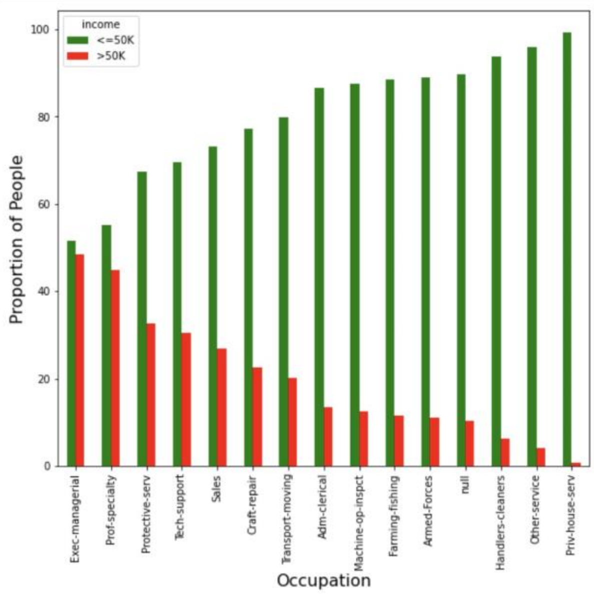
Given below are the tables comparing between observations having income greater than 50k and less than or equal to 50k based on the categorical attributes:

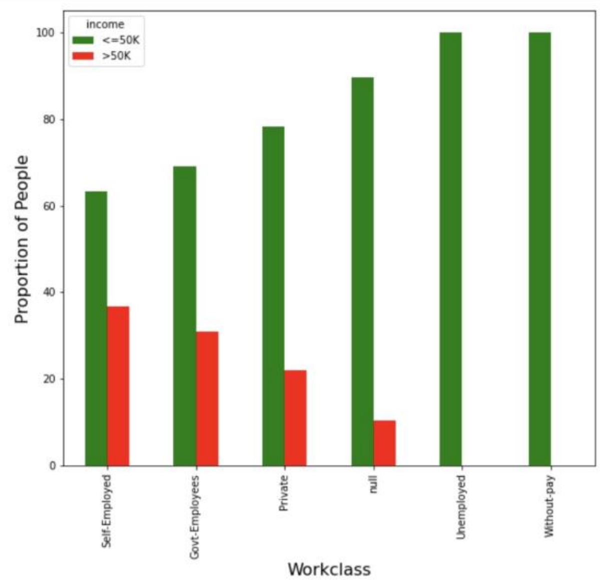

income
<=50K
>50K
0
20
40
60
80
Proportion of People
Doctorate
Prof-school
Post-Graduates
Undergraduates
Assoc-voc
Assoc-acdm
Some-college
Schooling
Education

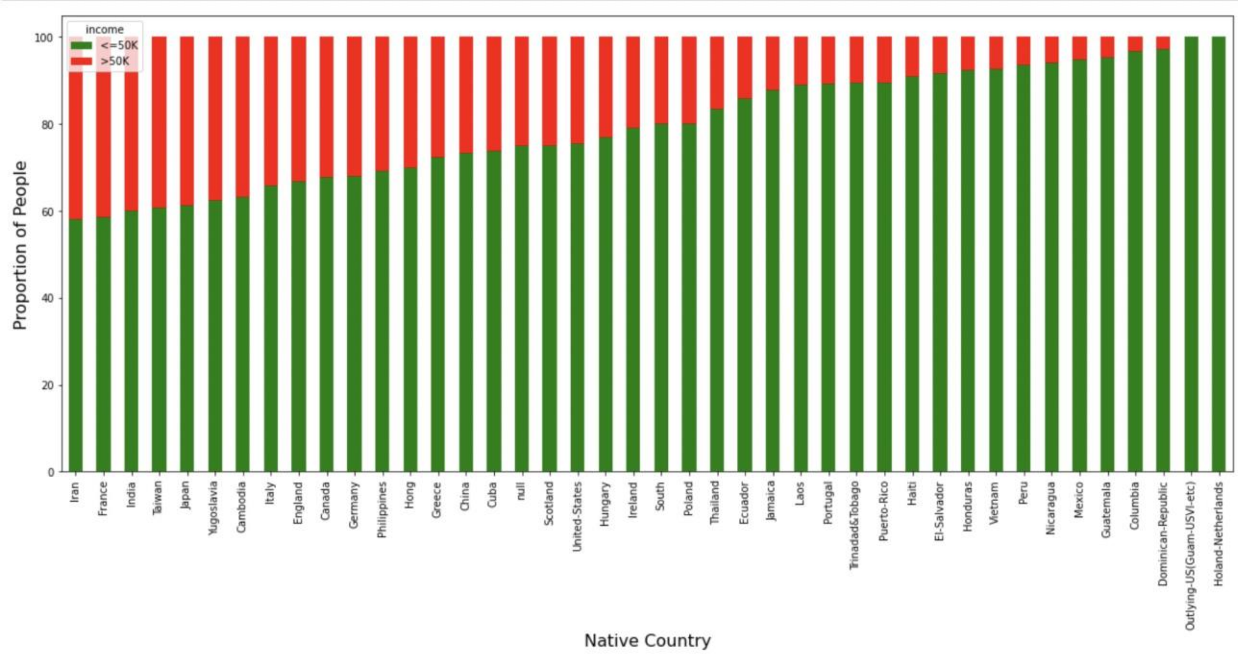

income
<=50K
>50K
0
20
40
60
80
100
Proportion of People
Married
Separated
Widowed
Single
Marital Status


income
<=50K
>50K
0
20
40
60
80
Proportion of People
Asian-Pac-Islander
White
Black
Amer-Indian-Eskimo
Other
Race


income
<=50K
>50K
0
20
40
60
80
Proportion of People
Male
Female
Sex


income
<=50K
>50K
Proportion of People
Occupation
Exec-managerial
Prof-specialty
Protective-serv
Tech-support
Sales
Craft-repair
Transport-moving
Adm-clerical
Machine-op-inspct
Farming-fishing
Armed-Forces
null
Handlers-cleaners
Other-service
Priv-house-serv


income
<=50K
>50K
Proportion of People
Workclass
Self-Employed
Govt-Employees
Private
null
Unemployed
Without-pay

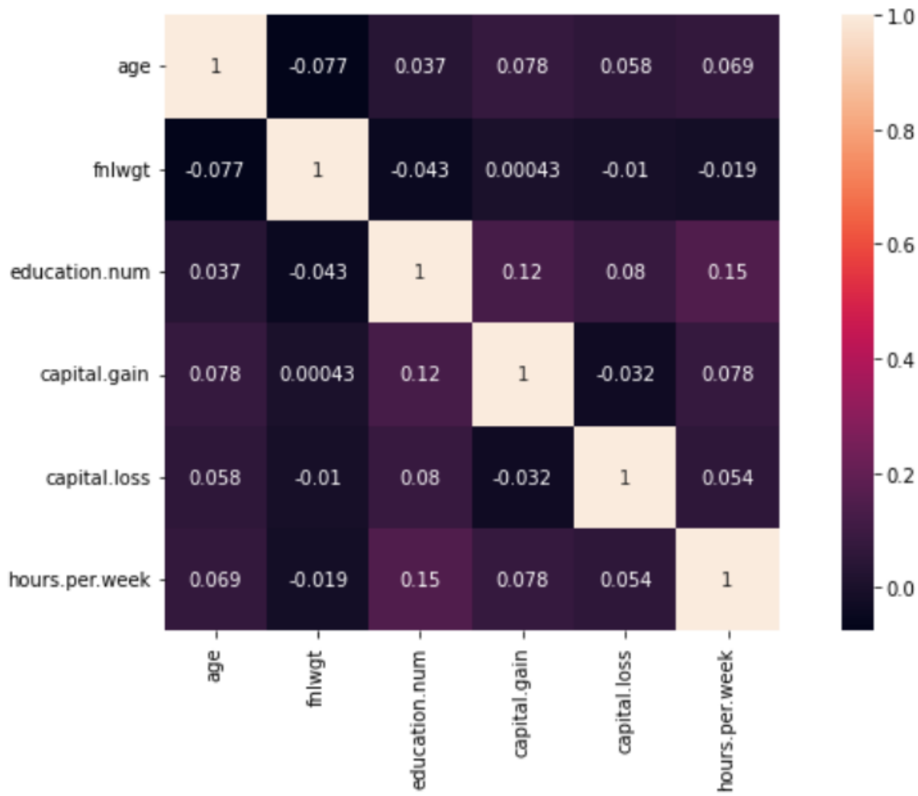

income
<=50K
>50K
Proportion of People
Native Country
Iran
France
India
Taiwan
Japan
Yugoslavia
Cambodia
Italy
England
Canada
Germany
Philippines
Hong
Greece
China
Cuba
null
Scotland
United-States
Hungary
Ireland
South
Poland
Thailand
Ecuador
Jamaica
Laos
Portugal
Trinadad&Tobago
Puerto-Rico
Haiti
El-Salvador
Honduras
Vietnam
Peru
Nicaragua
Mexico
Guatemala
Columbia
Dominican-Republic
Outlying-US(Guam-USVI-etc)
Holand-Netherlands

After displaying the comparison between the observations based on categorical attributes, we move towards the mathematical values of correlation between the attributes…

<u>The Correlation Matrix using seaborn.heatmap() function:</u>

In [30]:
```
fig = plt.figure(figsize = (12,6))

sns.heatmap(df[list(num_column.index)].corr(),annot = True,square = True);
```



Now, we move forward towards the data pre-processing.

4. Data Pre-processing:

1. We removed the numerical data present in the given data.

```
In [31]: df.drop(['age','fnlwgt','education.num','capital.gain','capital.loss','hours.per.week'], axis = 1, inplace = True)
```

```
In [32]: df.info()
```

```
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   workclass       32561 non-null  object
 1   education       32561 non-null  object
 2   marital.status  32561 non-null  object
 3   occupation      32561 non-null  object
 4   relationship    32561 non-null  object
 5   race            32561 non-null  object
 6   sex             32561 non-null  object
 7   native.country  32561 non-null  object
 8   income          32561 non-null  object
dtypes: object(9)
memory usage: 2.2+ MB
```

2. We then changed the income into 0 and 1, 0 represents income less than or equal to 50k and 1 represents greater than 50k

3. Then we changed the categorical data into natural numbers for the convenience.

In [45]:
```
df['relationship'] = df['relationship'].map({'Own-child': 0, 'Not-in-family': 1,
                                             'Wife': 2,
                                             'Husband': 3,
                                             'Unmarried': 4,
                                             'Other-relative': 5}).astype(int)
```

In [46]:
```
df
```

Out[46]:

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
0	4	1	3	13	1	3	1	United-States	0
1	1	1	3	7	1	3	1	United-States	0
2	4	5	3	13	4	0	1	United-States	0
3	1	1	1	5	4	3	1	United-States	0
4	1	5	1	4	0	3	1	United-States	0
...	...	...	...	...	...	...	...	...	...
32556	1	5	2	2	1	3	0	United-States	0
32557	1	2	0	6	2	3	1	United-States	0
32558	1	1	0	5	3	3	0	United-States	1
32559	1	1	3	14	4	3	1	United-States	0
32560	1	1	2	14	0	3	0	United-States	0

32561 rows × 9 columns

4. Then we removed the native country attribute.

In [47]:
```
df.drop(['native.country'], axis = 1, inplace = True)
```

In [48]:
```
df
```

Out[48]:

	workclass	education	marital.status	occupation	relationship	race	sex	income
0	4	1	3	13	1	3	1	0
1	1	1	3	7	1	3	1	0
2	4	5	3	13	4	0	1	0
3	1	1	1	5	4	3	1	0
4	1	5	1	4	0	3	1	0
...	...	...	...	...	...	...	...	...
32556	1	5	2	2	1	3	0	0
32557	1	2	0	6	2	3	1	0
32558	1	1	0	5	3	3	0	1
32559	1	1	3	14	4	3	1	0
32560	1	1	2	14	0	3	0	0

32561 rows × 8 columns

3. Then we split the data into 2 parts, X and Y. X contains categorical data and Y contains income in form of 0 and 1.

In [50]:
```
dfx = pd.DataFrame(X)
dfy = pd.DataFrame(Y)
```

4. We then split the X and Y into 2 parts each in the ratio 3:1 for training data and testing data.

In [51]:
```
# Splitting the data in the ratio 3:1 where 3 is for training data and 1 is for testing data
X_train, X_test, Y_train, Y_test = train_test_split(dfx, dfy, test_size = 0.25, random_state = 42)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(24420, 7)
(8141, 7)
(24420, 1)
(8141, 1)
```

5. ML CLASSIFICATION ALGORITHMS:

After we have a clean dataset, we can use our prediction algorithms to forecast the quality of our wine. Instead of just one classification algorithm, we use five in our project to predict the results.

We wanted to compare their accuracy results to see which algorithm worked best on our dataset, so we used all of these algorithms. We applied the algorithms given below:

○ Logistic regression

○Decision Tree Classifier

○ Random Forest Classifier

○ Naive Bayes Classifier

○ Support Vector Machine

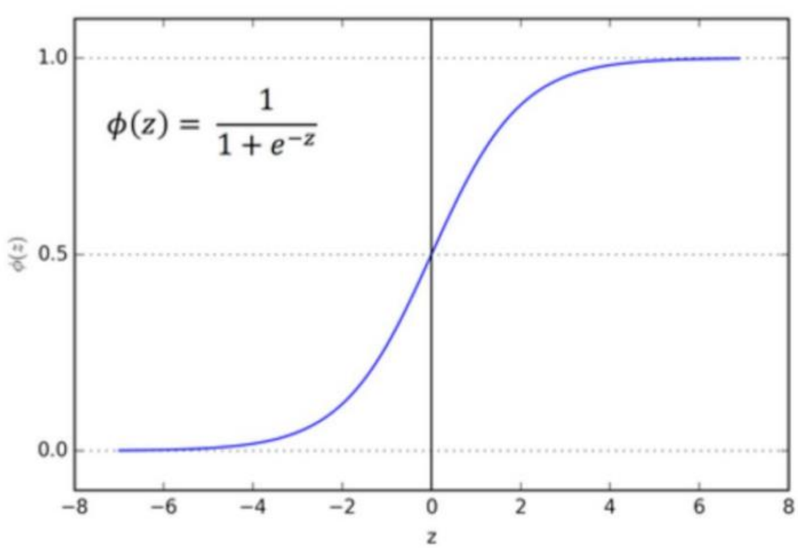
- For each case, we have visualized the Confusion Matrix along with it.
- We have also displayed the accuracy percentage for each case too.

(i) Logistic Regression:

IF we talk about most fundamental structure then, Logistic Regression is a model based on statistics. It is used when we have a categorical dependent variable(y), instead of continuous.

This model is used to calculate the probability of a certain class. It can also be used for multiclass attribute values as well.

It basically follows the linear regression model, but the continuous output value is passed through a function called as "Sigmoid Function" which is used to scale the value between 0 and 1. a threshold value selected. For our problem which is a 2-Class, if the sigmoid function gives a value greater than threshold, then class 1 is selected, otherwise class 0.



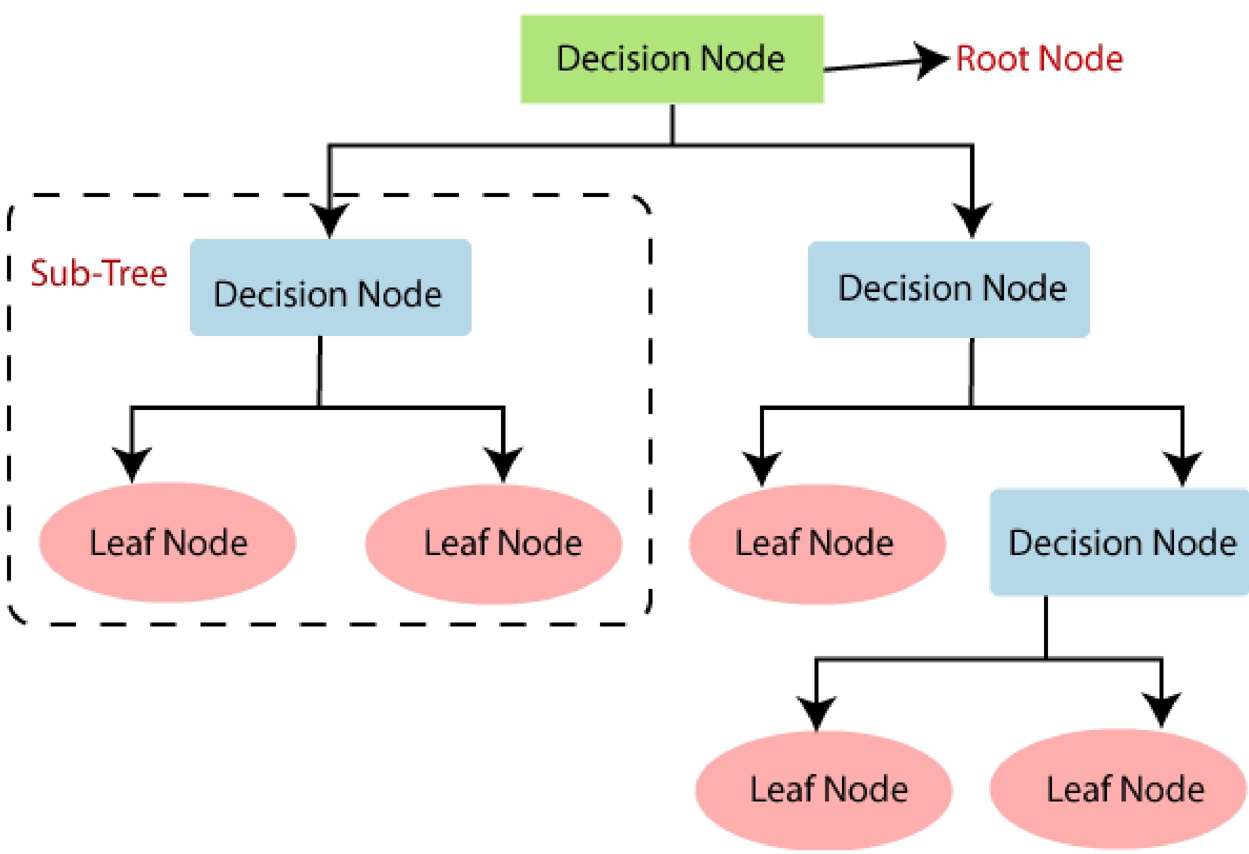
This sigmoid function helps to scale the values between 0-1.

(ii) Decision Tree Classifier

This model forms a decision tree based on the input dataset. It helps to predict the class of a new input record.

It is like a tree structure, with each node representing a condition on attribute values. Based on the condition output, we select our path (answer to conditions) and proceed to predict our class till we encounter a leaf node(output class variables).

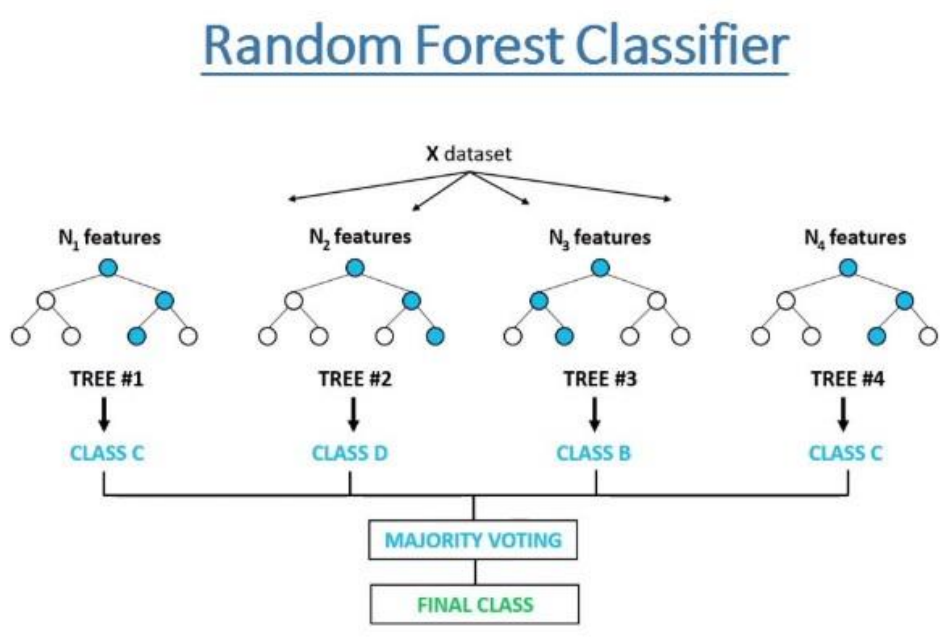
We have learnt that decision trees are made with the help of GINI Index, Entropy calculations etc which measure impurity, to try to determine what should be taken as the best split.



(iii) Random Forest Classifier

This classifier follows an ensemble learning. Instead of one, multiple decision trees work as an **“ensemble”.** It adds randomness to the ensemble by randomly creating a forest of decision trees.

Each decision tree produces an output and the final class for the input record is done by majority voting. They are found to be more accurate than single decision trees, but have an extended time for training.



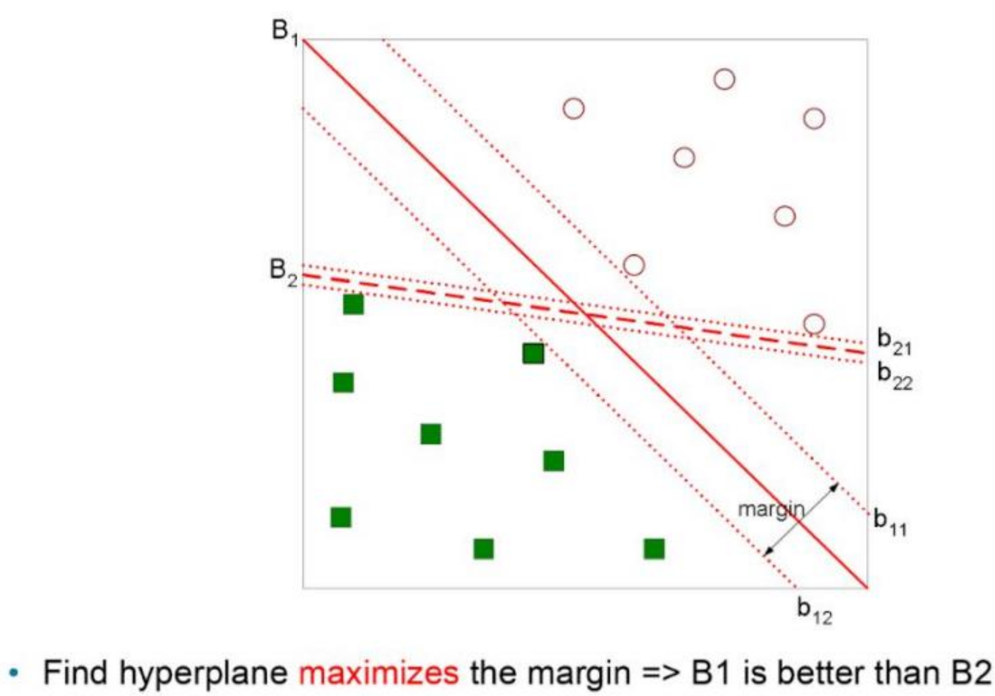
Structure of how random forest works

(iv) Support Vector Machine

Support Vector Machine uses a supervised learning algorithm. It is used to find a hyperplane that will separate the data classes.

Now, there may be many hyperplanes which can separate the data, but SVM tries to find the best fit line for this separation.

This classifier generally works well when there is a clear line of separation between the classes, and the dataset is not large enough.



Structure of SVM

(v) Naive Bayes Classifier

Naive Bayes classifier is based on the Bayes theorem which we study in Probability. In ML, it takes an assumption that there is independence among the attributes X(i) when prediction for the class.

- The formula for calculating probability using Bayes Theorem.

x represents features calculated individually.

$$\underbrace{P(c|x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x|c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Where,

- $P(c|x)$ is the posterior probability of *class* c given *predictor* (*features*).
- $P(c)$ is the probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

- If we take this assumption, then it becomes Naive Bayes Classifier.

- Assume independence among attributes X_i when class is given:
  - $P(X_1, X_2, \ldots, X_d | Y_j) = P(X_1| Y_j)\, P(X_2| Y_j) \ldots P(X_d| Y_j)$

6. IMPLEMENTATION OF THE ALGORITHMS AND THEIR CONFUSION MATRIX

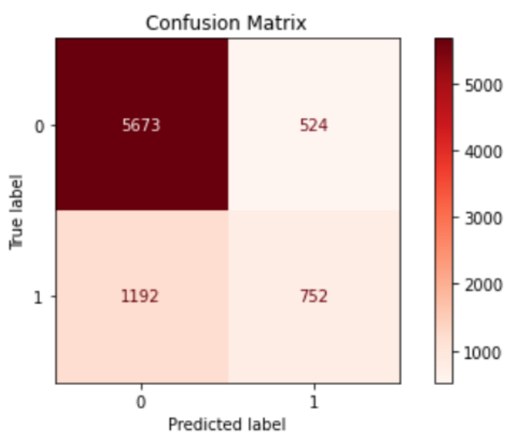
a) Logistic Regression:

In [52]:
```
# Using Logistic Regression
operation1 = LogisticRegression()
# Train our model with the training data
operation1.fit(X_train, Y_train)
Y_pred = operation1.predict(X_test)
```

In [53]:
```
score1 = accuracy_score(Y_test, Y_pred)
print("Prediction Accuracy = " + str(score1*100))
```

```
Prediction Accuracy = 78.92150841419972
```

In [54]:
```
class_names = [0,1]
fig, ax = plt.subplots(figsize=(8,4))
plot_confusion_matrix(operation1, X_test, Y_test,cmap=plt.cm.Reds,labels=class_names,ax=ax,values_format
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



Accuracy: 78.92%

Right Prediction: 5673 + 752 = 6425
Wrong Prediction: 1192 + 524 = 1944

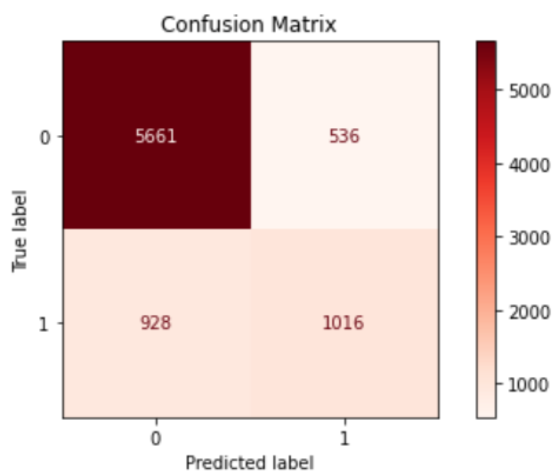
b) Decision Tree Classifier:

In [58]:
```
operation2 = DecisionTreeClassifier()
operation2.fit(X_train, Y_train)
Y_pred = operation2.predict(X_test)
```

In [59]:
```
score2 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score2*100))
```

```
Prediction Accuracy = 82.01695123449207
```

In [60]:
```
class_names = [0,1]
fig, ax = plt.subplots(figsize=(8,4))
plot_confusion_matrix(operation2, X_test, Y_test,cmap=plt.cm.Reds,labels=class_names,ax=ax,valu
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



Accuracy: 82%

Right Prediction: 5661 + 1015 = 6676
Wrong Prediction: 929 + 536 = 1492

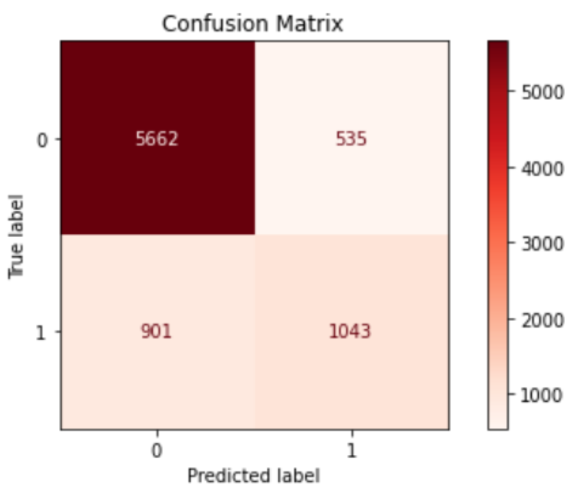
c) Random Forest Classifier:

In [64]:
```
operation4 = RandomForestClassifier()
operation4.fit(X_train, Y_train)
Y_pred = operation4.predict(X_test)
```

In [65]:
```
score4 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score4*100))
```

```
Prediction Accuracy = 82.36088932563567
```

In [66]:
```
class_names = [0,1]
fig, ax = plt.subplots(figsize=(8,4))
plot_confusion_matrix(operation4, X_test, Y_test,cmap=plt.cm.Reds,labels=class_names,ax=ax,v
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



Accuracy: 82.36%

Right Prediction: 5662 + 535 = 6705
Wrong Prediction: 901 + 535= 1436

d) Support Vector Machine:

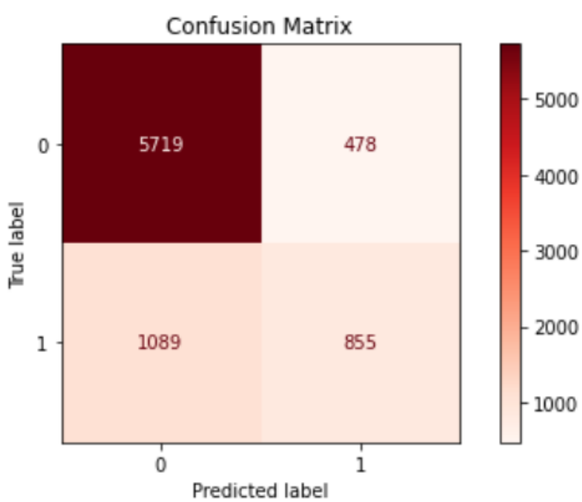
Support Vector Machine

In [67]:
```
operation5 = SVC()
operation5.fit(X_train, Y_train)
Y_pred = operation5.predict(X_test)
```

In [68]:
```
score5 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score5*100))
```

```
Prediction Accuracy = 80.75175039921385
```

In [69]:
```
class_names = [0,1]
fig, ax = plt.subplots(figsize=(8,4))
plot_confusion_matrix(operation5, X_test, Y_test,cmap=plt.cm.Reds,labels=class_names,ax=
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



Accuracy: 80.75%

Right Prediction: 5719 + 855 = 6574
Wrong Prediction: 1089 + 478 = 1567

e) Naive Bayes Classifier:

In [61]:

```
operation3 = GaussianNB()
operation3.fit(X_train, Y_train)
Y_pred = operation3.predict(X_test)
```

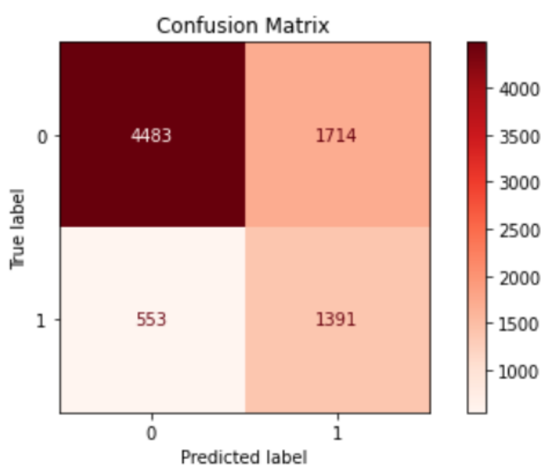
In [62]:

```
score3 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score3*100))
```

```
Prediction Accuracy = 72.153298120624
```

In [63]:

```
class_names = [0,1]
fig, ax = plt.subplots(figsize=(8,4))
plot_confusion_matrix(operation3, X_test, Y_test,cmap=plt.cm.Reds,labels=class_names,ax=ax,values_f
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



Accuracy: 72.15%

Right Prediction: 4483 + 1391 = 5874
Wrong Prediction: 553 + 1704 = 2257

7. CONCLUSION

To summarize our results from the implementations, we get the following table:

Algorithms	**Accuracy**
Logistic Regression	78.92%
Decision Tree	82%
Random Forest	82.36%
Support Vector Machine	80.75%
Naive Bayes	72.15%

From the results, it is clear that Random Forest Algorithm gives the highest accuracy of 82.36% on our dataset.

7. REFRENCES

- The class presentations of IDS
- Official documentations of Seaborn, Matplotlib, scikit-learn, Pandas, Numpy
- https://archive.ics.uci.edu/ml/datasets/Adult