IDS PROJECT REPORT

Implementing ML Classification Algorithm for

Predicting Whether the Income exceeds a given Income based on census data

https://archive.ics.uci.eduml/datasets/adult

Team Members:

Swati Mor : 21ucc101

Arpit Gupta : 21ucs030

Arpit Jain : 21ucs031

Elishben Manojbhai Baraiya : 21ucs077

Course Instructors:

Dr. Aloke Datta

Dr. Subrat Das

Dr. Lal Upendra Pratap Singh

Problem Statement

We are given an Excel dataset that has 15 columns and 32561 rows.

We are trying to predict whether the income of an adult will exceed 50k per year or not by developing a supervised machine learning model.

To achieve it we need to follow the data science/ML life cycle.Starting with data collection/extraction(which we have already collected).After that we need to follow following steps:

1. Data pre-processing and its visualization and explain all the inferences we got from our data.
2. Decide what ML Classification Algorithms to use and why
3. Implementing those algorithms
4. Output the result of the testing set and its visualization

Python has many libraries which makes the above steps more easy and efficient.Below is the list of some of libraries we would be using:

- scikit_learn
- matplotlib
- seaborn
- numpy
- pandas

Implementation

1.Importing the libraries and loading the dataset

(i) Importing Libraries :

•**pandas**: for manipulation and analysis of data.(using Dataframes)

•**numpy**: to operate on large arrays and matrices with multiple dimensions.

•**matplotlib**:to visualize data and get inferences using graphs and other elements.

•**seaborn**: for drawing attractive and informative statistical graphics.

•**SK-Learn**: for ML Classification Algorithms.

•**Sys and Warnings**: to ignore warnings.

(ii) Loading the Dataset

```
In [13]: # Data Importing
df = pd.read_csv("adult.csv")
print(df.shape)
df.head()

(32561, 15)
```

Out[13]:

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

```
In [15]: df.describe()
```

Out[15]:

	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

Dataset Contains Attributes:

Age
Workclass
Final Weight
Education
Education Number of Years
Marital-status.
Occupation
Relationship
Race
Sex
Capital-gain

Capital-loss
Hours-per-week
Native-country

2. Data analysis and exploration

I. Converting missing values to null:

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

```
# 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 attributes in the list

- **Education**: Merged the High School grad, 1st to 12th into Schooling, Replaced the bachelors to undergraduates and masters to postgraduates.

```python
# 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()
```

```
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
```

- **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

```python
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()
```

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

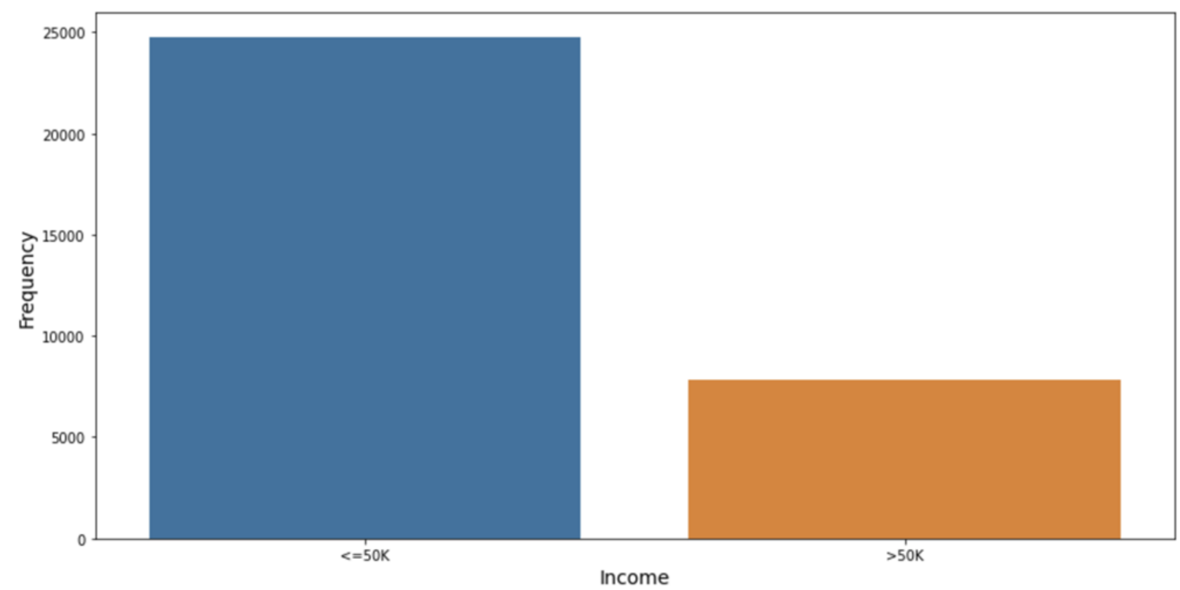
- **Work Class**: Merged self-emp-not-inc and self-emp-inc into Self-Employed, merged Local-gov, State-gov and federal-gov into govt-employees and replaced never worked to unemployee.

```
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()
```

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

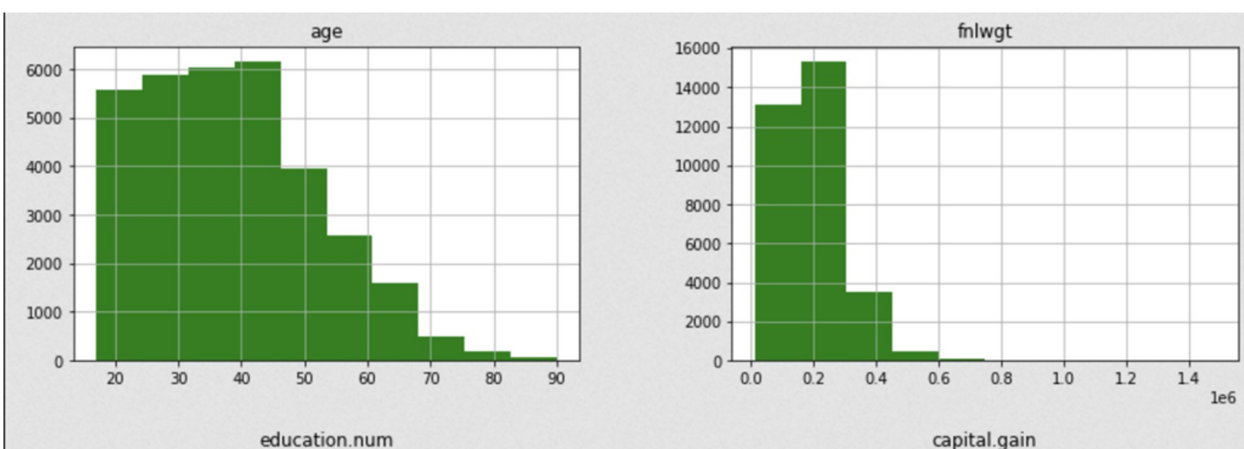
III. Visualization

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



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

- Distribution of population on the basis of numerical features:



inference: 1. Maximum individuals are in age between 15-45
2. Most people work 30-40 hours per week
3. Most individuals studied till 9th standard

- Exploring the data based on Capital Gain and Capital Loss:

```
*************** sex ***************
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
```

```
*************** 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


*************** relationship ***************
Husband           10739
Not-in-family      7427
Own-child          4810
Unmarried          3172
Wife               1272
Other-relative      910
Name: relationship, dtype: int64


*************** race ***************
White                 24061
Black                  2839
Asian-Pac-Islander      902
Amer-Indian-Eskimo      280
Other                   248
Name: race, dtype: int64
```

```
*************** 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
```

inference:

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.

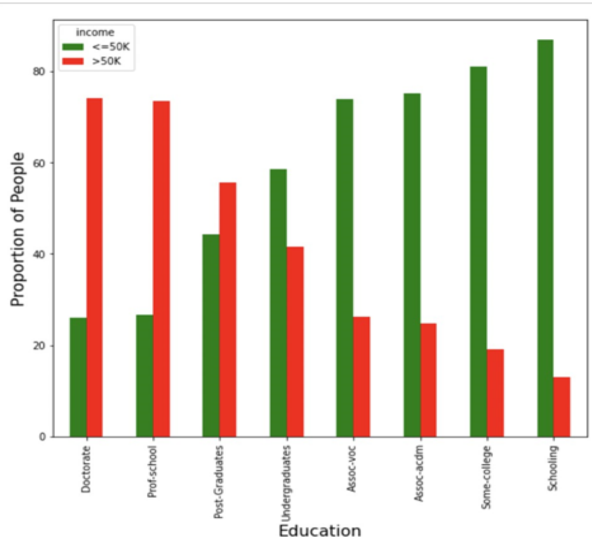
- The maximum capital gain is 99999 by 159 observations

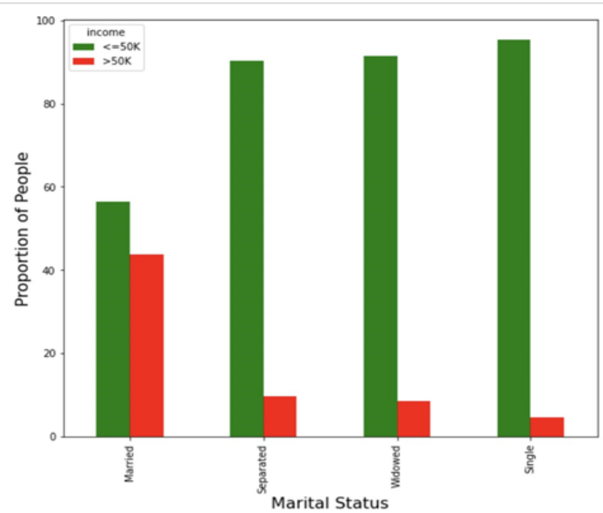
```
Number of observations having capital gain of 99999:(159, 15)
Income counts: >50K    159
Name: income, dtype: int64
```

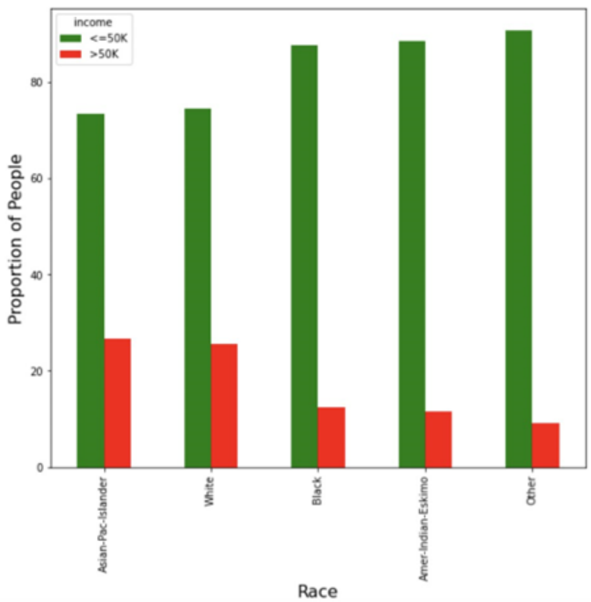
- The Maximum capital loss is 4356 by 3 observations

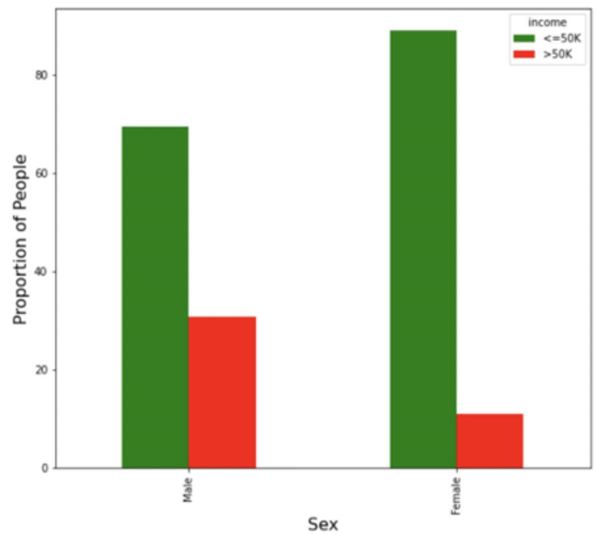
```
Number of observations having capital loss of 4356:(3, 15)
Income counts: <=50K    3
Name: income, dtype: int64
```

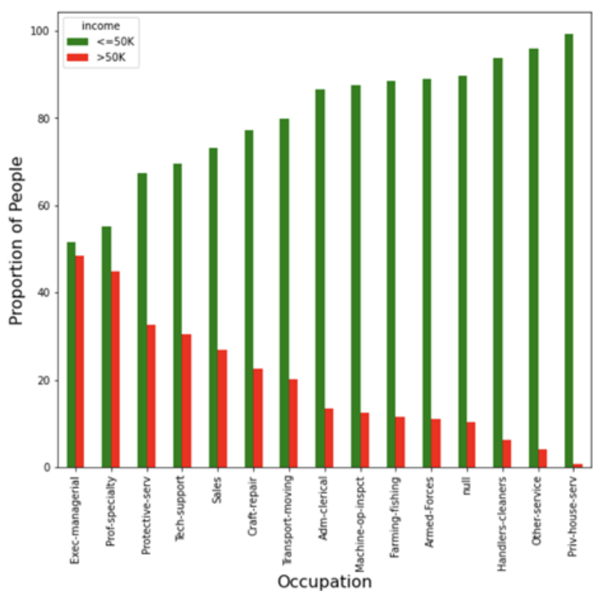
- comparing between observations having income greater than 50k and less than or equal to 50k on the basis of the categorical attributes:

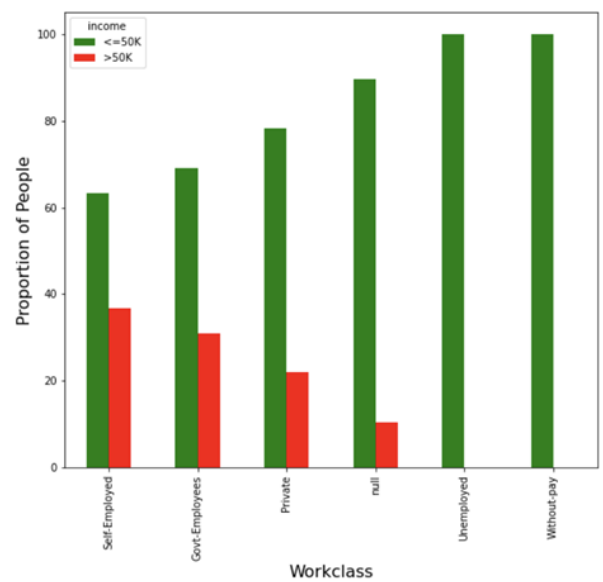


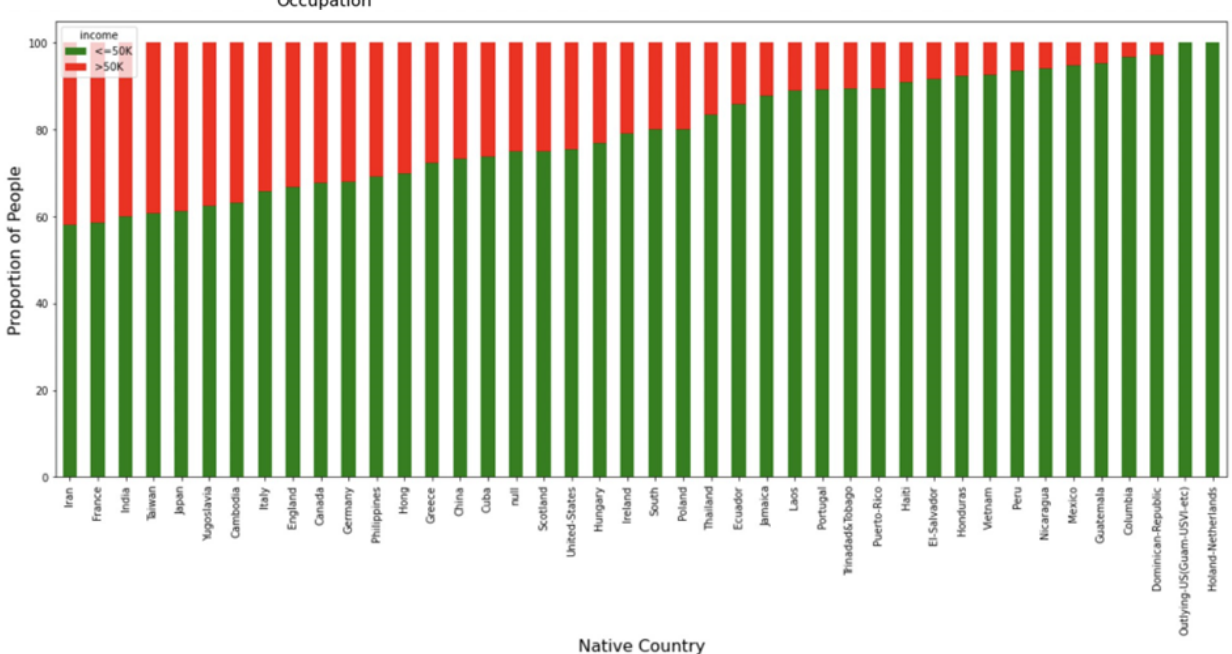



income
<=50K
>50K
Proportion of People
0
20
40
60
80
Asian-Pac-Islander
White
Black
Amer-Indian-Eskimo
Other
Race

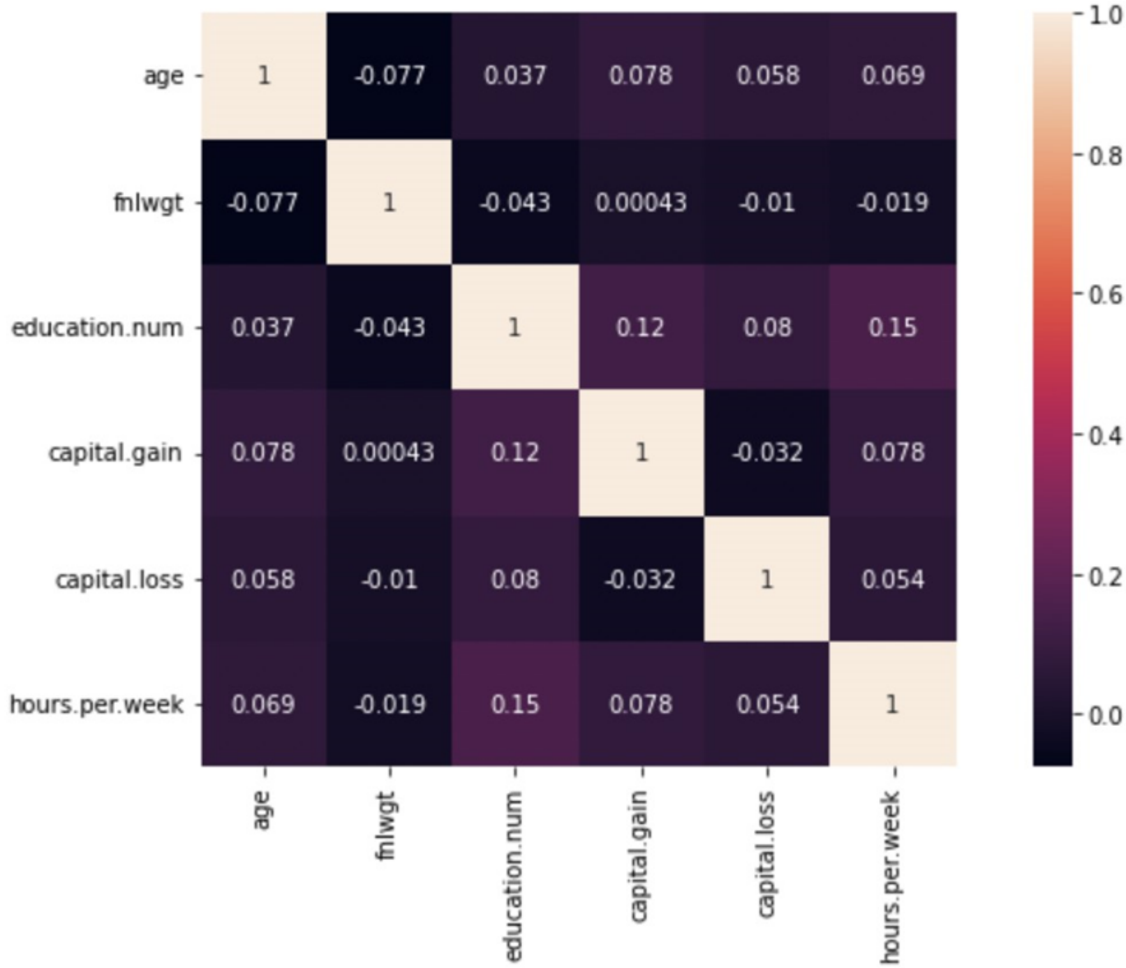

income
<=50K
>50K
Proportion of People
0
20
40
60
80
Male
Female
Sex


income
<=50K
>50K
Proportion of People
0
20
40
60
80
100
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
Occupation


income
<=50K
>50K
Proportion of People
0
20
40
60
80
100
Self-Employed
Govt-Employees
Private
null
Unemployed
Without-pay
Workclass


income
<=50K
>50K
Proportion of People
0
20
40
60
80
100
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)
Holland-Netherlands
Native Country

- Correlation between the attributes



3. Data preprocessing

1.removing the numerical data present in the given data

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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)
```

2.changing income into 0 and 1,
0 representing income less than or equal to 50k and
1 representing income greater than 50k

3.Converting Categorical data to numbers to handle them more effectively

	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

4.removing native country attribute as not needed apparently

	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

5. Now, splitting the data into input features and output label i.e X contains categorical data(predictors) and Y contains income (0/1)

6.Dividing X and Y into training and test sets

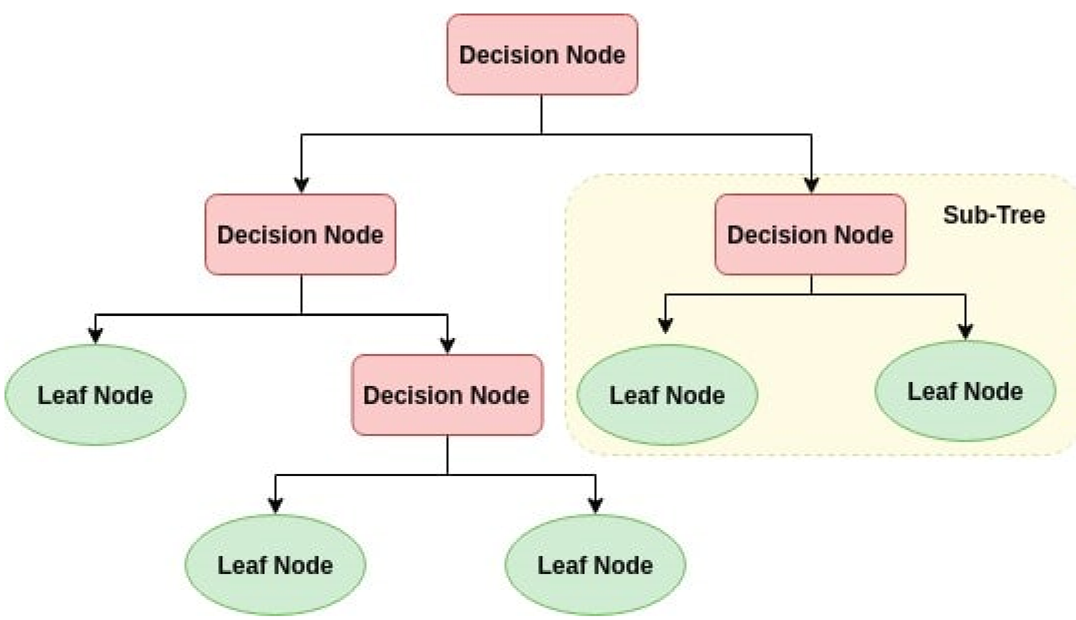
```
# Splitting the data in the ratio 3:1 where 3 is for training data and 1 is for testing data
X_train, X_test, Y_train, Y_test = train_test_split(dfx, dfy, test_size = 0.25, random_state = 42)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

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

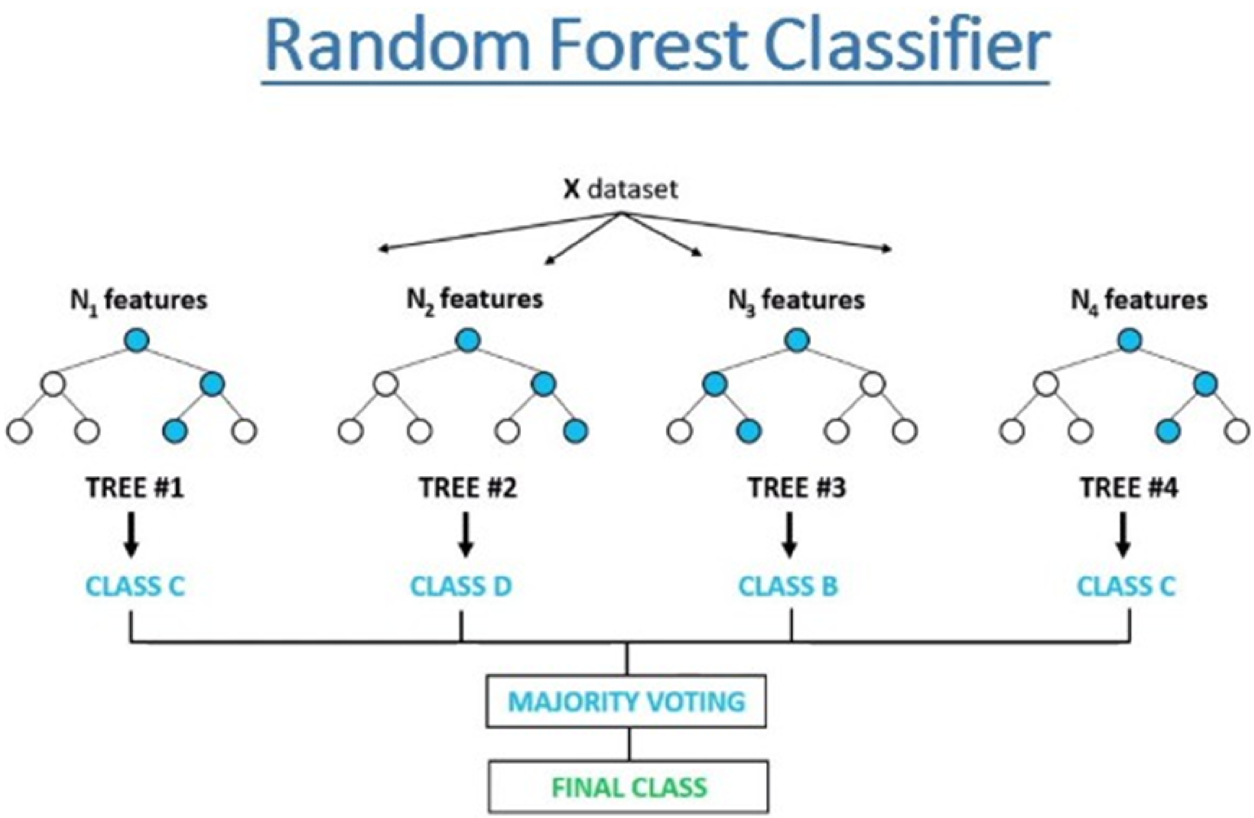
4. Applying ML Classification Algorithms

We would be applying 3 ML algorithms on the given dataset and look for the best one by comparing the accuracy obtained from each of them.

1. Decision Tree Classifier:
   In a decision tree, the algorithm begins at the root node and works its way up to predict the class of a given dataset. This algorithm checks the values of the root attribute with the values of the record (actual dataset) attribute and then follows the branch and jumps to the next node based on the comparison.
   The algorithm compares the attribute value with the other sub-nodes and moves on to the next node. It repeats the process until it reaches the tree's leaf node.

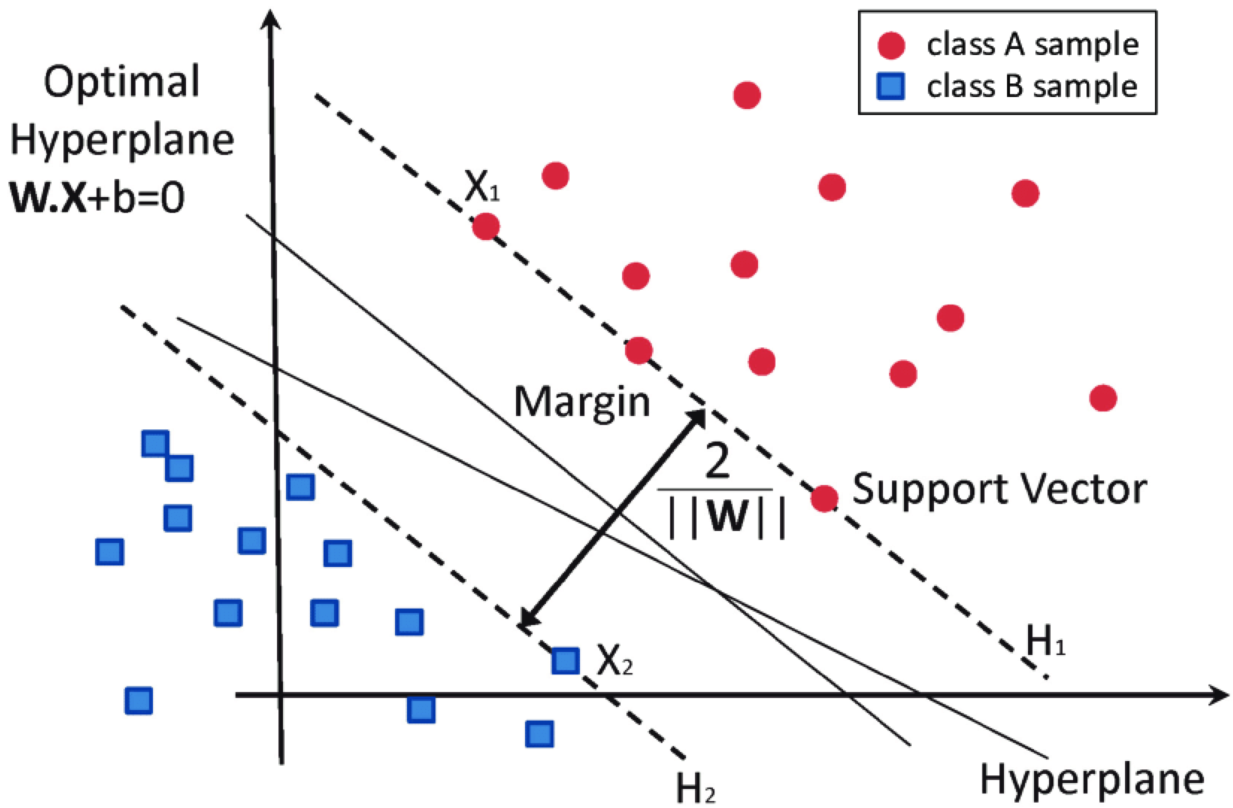


2. Random Forest Classifier:
   This process is based on learning together. Many decision trees are not a single tree but work as a whole. It increases the randomness of the integration by randomly generating a forest of decision trees. Each tree decides to produce the output, and the final ordering of the input data is determined by majority vote.
   They are more accurate than single decision trees but take longer to train.



3. Support Vector Machine

Support vector machines use supervised learning algorithms. It is used to find hyperplanes separating groups of data. Now there may be many planes separating the data, but SVM tries to find the best line for this separation. When there is a clear dividing line between groups and the data sets are not large enough.



5.Implementing these algorithms and displaying their confusion matrix

1.Decision Tree Classifier

Accuracy: 82%

Right Prediction: 5661 + 1015 = 6676

Wrong Prediction: 929 + 536 = 1492

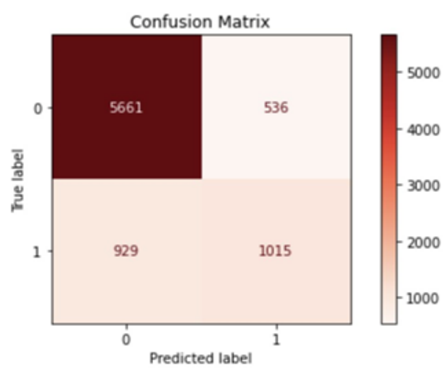
Decision Tree Classifier

```
operation2 = DecisionTreeClassifier()
operation2.fit(X_train, Y_train)
Y_pred = operation2.predict(X_test)
```

```
score2 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score2*100))
```

```
Prediction Accuracy = 82.00466773123695
```

```
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,values_format = '.0f')
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



2.Random Forest Classifier

Accuracy: 82.36%

Right Prediction: 5662 + 535 = 6705

Wrong Prediction: 901 + 535= 1436

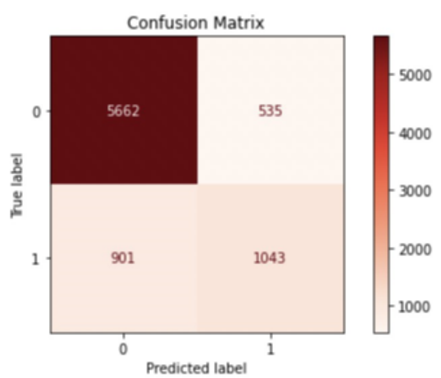
Random Forest ¶

```
operation4 = RandomForestClassifier()
operation4.fit(X_train, Y_train)
Y_pred = operation4.predict(X_test)
```

```
score4 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score4*100))
```

```
Prediction Accuracy = 82.36088932563567
```

```
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,values_format = '.0f')
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



3.Support Vector Machine

Accuracy: 80.75%

Right Prediction: 5719 + 855 = 6574

Wrong Prediction: 1089 + 478 = 1567

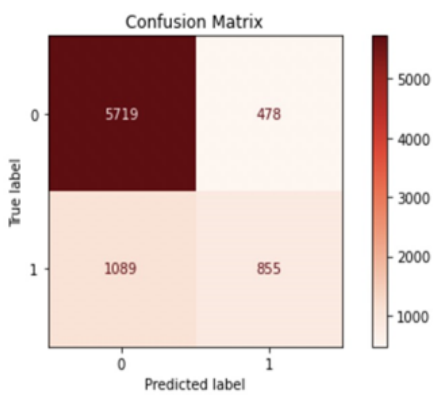
Support Vector Machine

```
operation5 = SVC()
operation5.fit(X_train, Y_train)
Y_pred = operation5.predict(X_test)
```

```
score5 = accuracy_score(Y_test, Y_pred)
print('Prediction Accuracy = ' + str(score5*100))
```

```
Prediction Accuracy = 80.75175039921385
```

```
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=ax,values_format = '.0f')
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```



6. Conclusion

Algorithm	Accuracy
Decision Tree	82%
Random Forest	82.36%
SVM	80.75%

Thus after performing data collection, data exploration , data preprocessing and finally training 3 different models for the given problem we conclude that Random Forest is giving the best accuracy and is the most suited algorithm here.

7.References

- Class notes and discussions
- Documentation of python libraries i.e Matplotlib,seaborn,scikit-learn,pandas and numpy

- https://archive.ics.uci.edu/ml/datasets/adult