```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
/kaggle/input/train-and-test-datasets/train.csv
/kaggle/input/train-and-test-datasets/test.csv
```

```
traindata= pd.read_csv('../input/train-and-test-datasets/train.csv')
traindata.head()
traindata.shape
```

```
(43957, 15)
```

Check for null values

```
traindata.isnull().sum()
```

```
age                 0
workclass        2498
fnlwgt              0
education           0
educational-num     0
marital-status      0
occupation       2506
relationship        0
race                0
gender              0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country    763
income_>50K         0
dtype: int64
```

Fill na values when dtype=int

```
#traindata['workclass']= traindata['workclass'].fillna(traindata['workclass'].mean())
```

Remove null values

```
train1=traindata.dropna()
train1.isnull().sum()
train1.shape
```

```
(40727, 15)
```

```
train1.columns
train1.dtypes
#train1.describe()
```

```
age                int64
workclass         object
fnlwgt             int64
education         object
educational-num    int64
marital-status    object
occupation        object
relationship      object
race              object
gender            object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
income_>50K        int64
dtype: object
```

Selecting features of dtype='object'

```
objfeatures= list(train1)
objfeatures
```

```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'educational-num',
 'marital-status',
```

```
 'occupation',
 'relationship',
 'race',
 'gender',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country',
 'income_>50K']
```

Data preprocessing - to normalise the values of dtype= 'object'- using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
#import pickle
le=LabelEncoder()
for i in train1.columns:
    if train1[i].dtypes=='object':
        train1[i]=le.fit_transform(train1[i])
#output = open('Departure_encoder.pkl', 'wb')
#pickle.dump(le, output)
#output.close()
train1.head()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week
0	67	2	366425	10	16	0	3	1	4	1	99999	0	60
1	17	2	244602	2	8	4	7	3	4	1	0	0	15
2	31	2	174201	9	13	2	3	0	4	1	0	0	40
3	58	5	110199	5	4	2	13	0	4	1	0	0	40

ONE HOT ENCODING - not required in this classification problem - OneHotEncoding - 0 or 1 for the respective LabelEncoding values

```
from sklearn.preprocessing import OneHotEncoder
en=OneHotEncoder()
train2=pd.DataFrame(en.fit_transform(train1[['workclass']]).toarray())
train2.head()
a=train1.join(train2)
a.head()
f=a.drop(['workclass'],axis=1)
f.head()
```

	age	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	...	hours-per-week	native-country	income
0	67	366425	10	16	0	3	1	4	1	99999	...	60	38	
1	17	244602	2	8	4	7	3	4	1	0	...	15	38	
2	31	174201	9	13	2	3	0	4	1	0	...	40	38	
3	58	110199	5	4	2	13	0	4	1	0	...	40	38	
4	25	149248	15	10	4	7	1	2	1	0	...	40	38	

X= Independent variables or features or attributes, y = Dependent or target variables

```
X=train1.drop(['income_>50K'],axis=1)
y=train1[['income_>50K']]  #if two features train1[['age','income_>50K']]
print(X.head())
print(y.head())
```

```
   age  workclass  fnlwgt  education  educational-num  marital-status  \
0   67          2  366425         10               16               0
1   17          2  244602          2                8               4
2   31          2  174201          9               13               2
3   58          5  110199          5                4               2
4   25          5  149248         15               10               4

   occupation  relationship  race  gender  capital-gain  capital-loss  \
0           3             1     4       1         99999             0
1           7             3     4       1             0             0
2           3             0     4       1             0             0
```

```
3          13             0     4       1             0             0
4           7             1     2       1             0             0

   hours-per-week  native-country
0              60              38
1              15              38
2              40              38
3              40              38
4              40              38
   income_>50K
0            1
1            0
2            1
3            0
4            0
```

Spliting train and test data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=10)
print(X_train.shape,y_train.shape)
print(y_train.head())
```

```
(28508, 14) (28508, 1)
       income_>50K
14746            1
24388            0
6877             0
19775            1
38983            1
```

Importing Classifier now

1. **LogisticRegression Classifier: since target is 0/1**

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when
  y = column_or_1d(y, warn=True)
```

```
LogisticRegression()
```

1) a) LogisticRegression Model prediction

```
lr.predict(X_test)
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

1) b) Logistic Regression Model prediction score - Here, it is 79%

```
lr.score(X_test,y_test)
```

```
0.7934364514281038
```

2) Bernoulli Naive Bayes

```
from sklearn.naive_bayes import BernoulliNB
#model training
NB=BernoulliNB(alpha=0.3)
NB.fit(X_train,y_train)
#prediction
NB.predict(X_test)

#score
NB.score(X_test,y_test)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when
  y = column_or_1d(y, warn=True)
0.7312382355348228
```

3) Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
Rf=RandomForestClassifier()
Rf.fit(X_train,y_train)
Rf.score(X_test,y_test)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d arr
  This is separate from the ipykernel package so we can avoid doing imports until
0.8556346673213847
```

Importing the test.csv file for implementing the model for prediction

```
targettest= pd.read_csv('../input/train-and-test-datasets/test.csv')
targettest.head()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week
0	39	Self-emp-not-inc	327120	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40
1	32	Private	123253	Assoc-acdm	12	Married-civ-	Craft-repair	Husband	White	Male	0	0	42

Encoding the 'object' dtype as done earlier

```
from sklearn.preprocessing import LabelEncoder
#import pickle
le=LabelEncoder()
for i in targettest.columns:
    if targettest[i].dtypes==object:
        targettest[i]=le.fit_transform(targettest[i])
#output = open('Departure_encoder.pkl', 'wb')
#pickle.dump(le, output)
#output.close()
targettest.head()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week
0	39	4	327120	11	9	2	2	0	4	1	0	0	40
1	32	2	123253	7	12	2	2	0	4	1	0	0	42
2	47	2	232628	11	9	2	2	0	2	1	0	0	40

Using the LR model trained to predict the target

```
result=Rf.predict(targettest)
tar=pd.DataFrame(result)
tar.index.name='id'
tar.rename(columns={0: 'Outcome'}, inplace=True)
print(tar.head())
tar.to_csv('target.csv')
```

```
        Outcome
    id
    0         0
    1         0
    2         0
    3         0
    4         0
```

```
#for i in range(len(result)):
    #print(str(i)+','+str(result[i]))
```

Colab paid products - Cancel contracts here