```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('adult.csv')
df.head(10)
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	rela
0	90	?	77053	HS-grad	9	Widowed	?	No
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	No
2	66	?	186061	Some-college	10	Widowed	?	U
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	U
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	U
6	38	Private	150601	10th	6	Separated	Adm-clerical	U
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Oth
8	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	No
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	U

```
print("total rows:", df.shape[0])
dataset_row=df.shape[0]
print("total columns:", df.shape[1])
print("\n features:\n", df.columns.tolist())
print("\nmissing values:", df.isnull().sum().values.sum())
print("\n unique values:\n", df.nunique())
```

```
total rows: 32561
total columns: 15

 features:
 ['age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'capit

missing values: 0

 unique values:
 age                 73
workclass             9
fnlwgt            21648
education            16
education.num        16
marital.status        7
occupation           15
relationship          6
race                  5
sex                   2
capital.gain        119
capital.loss         92
hours.per.week       94
native.country       42
income                2
dtype: int64
```

```
df.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
```

```
df.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

```
df_missing=(df=='?').sum()
print(df_missing)
```

```
age                 0
workclass        1836
fnlwgt              0
education           0
education.num       0
marital.status      0
occupation       1843
relationship        0
race                0
sex                 0
capital.gain        0
capital.loss        0
hours.per.week      0
native.country    583
income              0
dtype: int64
```

```
#droping row having missing values from dataset
df= df[df['workclass']!='?']
df= df[df['occupation']!='?']
df= df[df['native.country']!='?']
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	ca
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	

```
df_missing= (df=='?').sum()
print(df_missing)
```

```
age               0
workclass         0
fnlwgt            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
```

```
print("total rows after droping rows:", df.shape[0])
print("numbers of rows drop:",dataset_row- df.shape[0])
```

```
total rows after droping rows: 30162
numbers of rows drop: 2399
```

```
# Data preprocessing
from sklearn import preprocessing
```

```
df_categorical= df.select_dtypes(include=['object'])
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
1	Private	HS-grad	Widowed	Exec-managerial	Not-in-family	White	Female	United-States	<=50K
3	Private	7th-8th	Divorced	Machine-op-inspct	Unmarried	White	Female	United-States	<=50K
4	Private	Some-college	Separated	Prof-specialty	Own-child	White	Female	United-States	<=50K
5	Private	HS-grad	Divorced	Other-service	Unmarried	White	Female	United-States	<=50K
6	Private	10th	Separated	Adm-clerical	Unmarried	White	Male	United-States	<=50K

```
le=preprocessing.LabelEncoder()
df_categorical= df_categorical.apply(le.fit_transform)
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
1	2	11	6	3	1	4	0	38	0
3	2	5	0	6	4	4	0	38	0
4	2	15	5	9	3	4	0	38	0
5	2	11	0	7	4	4	0	38	0
6	2	0	5	0	4	4	1	38	0

```
df=df.drop(df_categorical.columns,axis=1)
df=pd.concat([df,df_categorical], axis=1)
df['income']=df['income'].astype('category')
df.head()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation	relatio
1	82	132870	9	0	4356	18	2	11	6	3	
3	54	140359	4	0	3900	40	2	5	0	6	
4	41	264663	10	0	3900	40	2	15	5	9	
5	34	216864	9	0	3770	45	2	11	0	7	
6	38	150601	6	0	3770	40	2	0	5	0	

```
df.info()
```

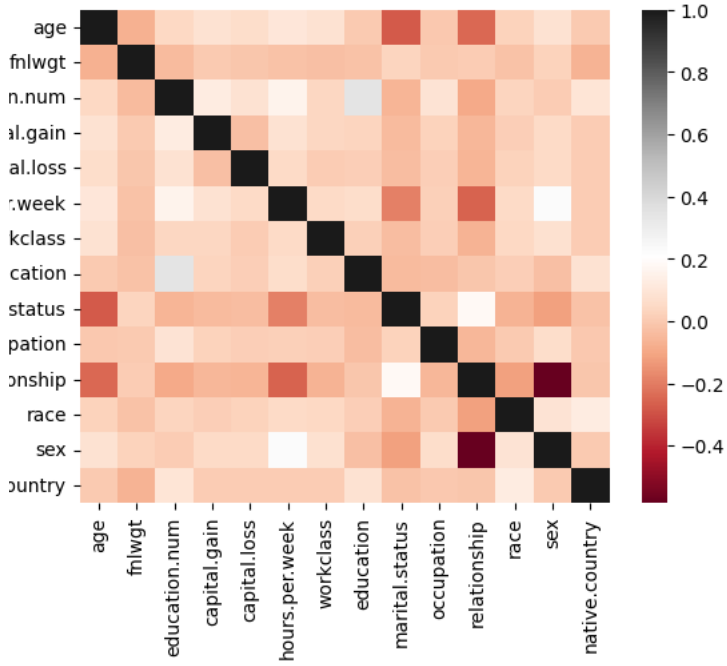
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   fnlwgt          30162 non-null  int64
 2   education.num   30162 non-null  int64
 3   capital.gain    30162 non-null  int64
 4   capital.loss    30162 non-null  int64
 5   hours.per.week  30162 non-null  int64
 6   workclass       30162 non-null  int64
```

```
 7   education       30162 non-null  int64
 8   marital.status  30162 non-null  int64
 9   occupation      30162 non-null  int64
 10  relationship    30162 non-null  int64
 11  race            30162 non-null  int64
 12  sex             30162 non-null  int64
 13  native.country  30162 non-null  int64
 14  income          30162 non-null  category
dtypes: category(1), int64(14)
memory usage: 3.5 MB
```

Visualization

```
sns.heatmap(df.corr(), cmap='RdGy')
```

```
input-20-33c73b4a87c1>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
tmap(df.corr(), cmap='RdGy')
```



Splitting dataset

```
from sklearn.model_selection import train_test_split
```

```
X=df.drop('income', axis=1)
X=X.drop('sex', axis=1)
y=df['income']
X.head()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation	relatio
1	82	132870	9	0	4356	18	2	11	6	3	
3	54	140359	4	0	3900	40	2	5	0	6	
4	41	264663	10	0	3900	40	2	15	5	9	
5	34	216864	9	0	3770	45	2	11	0	7	
6	38	150601	6	0	3770	40	2	0	5	0	

```
y.head()
```

```
1    0
3    0
4    0
5    0
6    0
Name: income, dtype: category
Categories (2, int64): [0, 1]
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20)
```

Applying RandomForest Algorithm

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

```
dt_default= RandomForestClassifier(max_depth=5)
dt_default.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=5)
```

```
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
```

```
y_pred_default=dt_default.predict(X_test)
print("confusion matrix\n", confusion_matrix(y_test,y_pred_default))
print(classification_report(y_test,y_pred_default))
```

```
confusion matrix
 [[4319  181]
 [ 731  802]]
              precision    recall  f1-score   support

           0       0.86      0.96      0.90      4500
           1       0.82      0.52      0.64      1533

    accuracy                           0.85      6033
   macro avg       0.84      0.74      0.77      6033
weighted avg       0.85      0.85      0.84      6033
```

```
print("accuracy score:", accuracy_score(y_test,y_pred_default))
```

```
accuracy score: 0.8488314271506713
```

✓ 0s completed at 3:47 PM