Experiment No. 4
Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance: 14-08-23
Date of Submission: 22-08-23

Aim: Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: Able to perform various feature engineering tasks, apply Random Forest Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

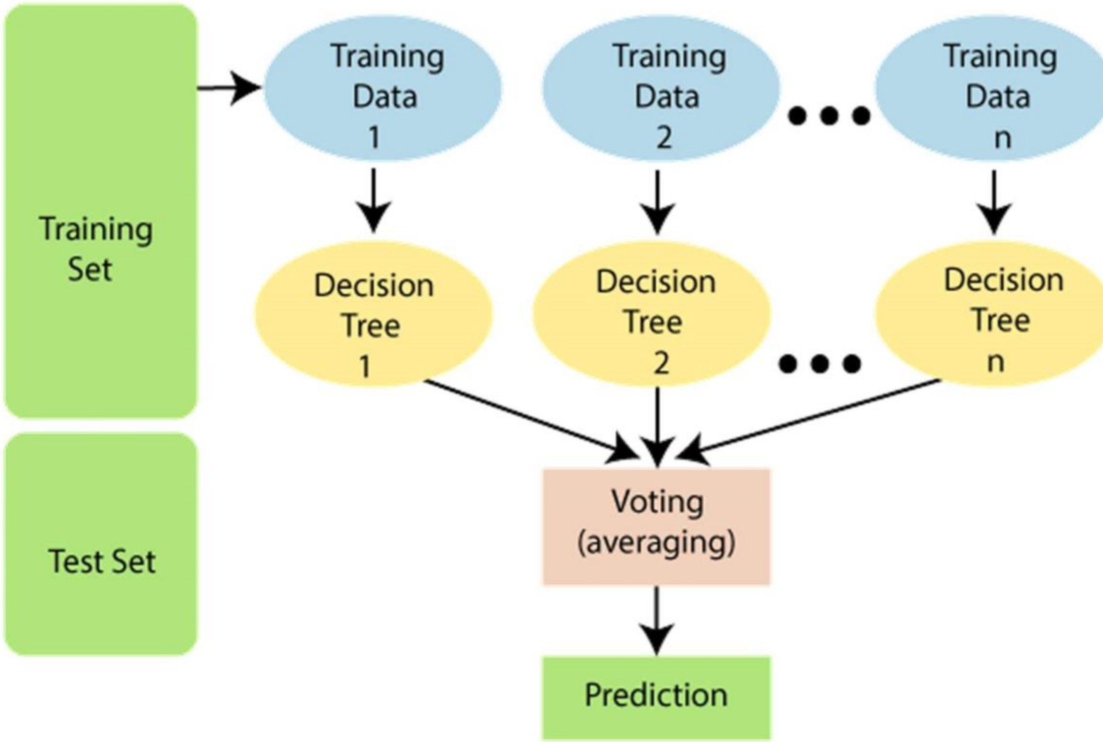
Theory:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Dataset:

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K. age:

continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Marriedspouse-absent, Married-AF-spouse.

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: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous. capital-

loss: continuous. hours-per-week:

continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, OutlyingUS(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, Holand-Netherlands.

Code:

Conclusion:

1. Findings from the Correlation Heat Map: The correlation heat map is a helpful tool for figuring out how various features in the dataset are related to one another. This is an important point to consider while, The "relationship" and "sex" features show an association in the correlation heatmap, as can be seen.

2. Performance Measurements
The confusion matrix describes true/false predictions, precision concentrates on accurate positive classifications, recall finds pertinent instances, and F1-Score balances precision and recall, which is essential for imbalanced classes. Accuracy measures overall accuracy. Precision concentrates on accurate positive classifications.

```
confusion matrix
 [[ 732  767]
 [ 191 4338]]
              precision    recall  f1-score   support

           0       0.79      0.49      0.60      1499
           1       0.85      0.96      0.90      4529

    accuracy                           0.84      6028
   macro avg       0.82      0.72      0.75      6028
weighted avg       0.84      0.84      0.83      6028
```

1. Comparison with Decision Tree Algorithm:

Result obtain using decision tree were:

```
confusion matrix
 [[ 764  735]
 [ 214 4315]]
              precision    recall  f1-score   support

           0       0.78      0.51      0.62      1499
           1       0.85      0.95      0.90      4529

    accuracy                           0.84      6028
   macro avg       0.82      0.73      0.76      6028
weighted avg       0.84      0.84      0.83      6028
```

Both the Decision Tree and Random Forest models achieved an accuracy of approximately 84%, indicating that they correctly predicted income levels for most individuals. However, they exhibited lower recall for the higher income class (1), indicating a tendency to miss some individuals with incomes over 50K, despite having good precision for the lower income class (0).

```
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()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relatio
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-
2	66	?	186061	Some-college	10	Widowed	?	Unm
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unm
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Ow

```
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()
df.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.w
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000

```
# Info for categorical features
df.describe(include=['O'])
```

	workclass	education	marital.status	occupation	relationship	race	sex	na
count	32561	32561	32561	32561	32561	32561	32561	
unique	9	16	7	15	6	5	2	
top	Private	HS-grad	Married-civspouse	Profspecialty	Husband	White	Male	
freq	22696	10501	14976	4140	13193	27816	21790	

```
duplicated_rows = df.duplicated()
any_duplicates = duplicated_rows.any()

print("Duplicated Rows:")
df[duplicated_rows] Duplicated Rows:
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	rel
8453	25	Private	308144	Bachelors	13	Never-married	Craft-repair	N
8645	90	Private	52386	Some-college	10	Never-married	Other-service	N
12202	21	Private	250051	Some-college	10	Never-married	Prof-specialty	
				Some-			Tech-	

14346	20	Private	107658	college	10	Never-married	support	N
15603	25	Private	195994	1st-4th	2	Never-married	Priv-house-serv	N
17344	21	Private	243368	Preschool	1	Never-married	Farming-fishing	N
19067	46	Private	173243	HS-grad	9	Married-civ-spouse	Craft-repair	
20388	30	Private	144593	HS-grad	9	Never-married	Other-service	N
20507	19	Private	97261	HS-grad	9	Never-married	Farming-fishing	N
22783	19	Private	138153	Some-college	10	Never-married	Adm-clerical	
22934	19	Private	146679	Some-college	10	Never-married	Exec-managerial	
23276	49	Private	31267	7th-8th	4	Married-civ-spouse	Craft-repair	
23660	25	Private	195994	1st-4th	2	Never-married	Priv-house-serv	N
23720	44	Private	367749	Bachelors	13	Never-married	Prof-specialty	N
23827	49	Self-emp-not-inc	43479	Some-college	10	Married-civ-spouse	Craft-repair	
26738	23	Private	240137	5th-6th	3	Never-married	Handlers-cleaners	N
27133	28	Private	274679	Masters	14	Never-married	Prof-specialty	N
							Machine-	

28796	27	Private	255582	HS-grad	9	Never-married	op-inspct	N
29051	42	Private	204235	Some-college	10	Married-civ-spouse	Prof-specialty	
29334	39	Private	30916	HS-grad	9	Married-civspouse	Craft-repair	
29604	38	Private	207202	HS-grad	9	Married-civspouse		
31060	46	Private	133616	Some-college	10	Divorced	Machineop-inspct	
							Admclerical	
32065	19	Private	251579	Some-college	10	Never-married	Otherservice	
32419	35	Private	379959	HS-grad	9	Divorced	Otherservice	N

```
df = df.drop_duplicates()
```

```
# Correction of target value using a map
income_map = {'<=50K': 1, '>50K': 0}
df['income'] = df['income'].map(income_map)
```

```
<ipython-input-47-00c8c2884cd1>:3: 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_
  df['income'] = df['income'].map(income_map)
```

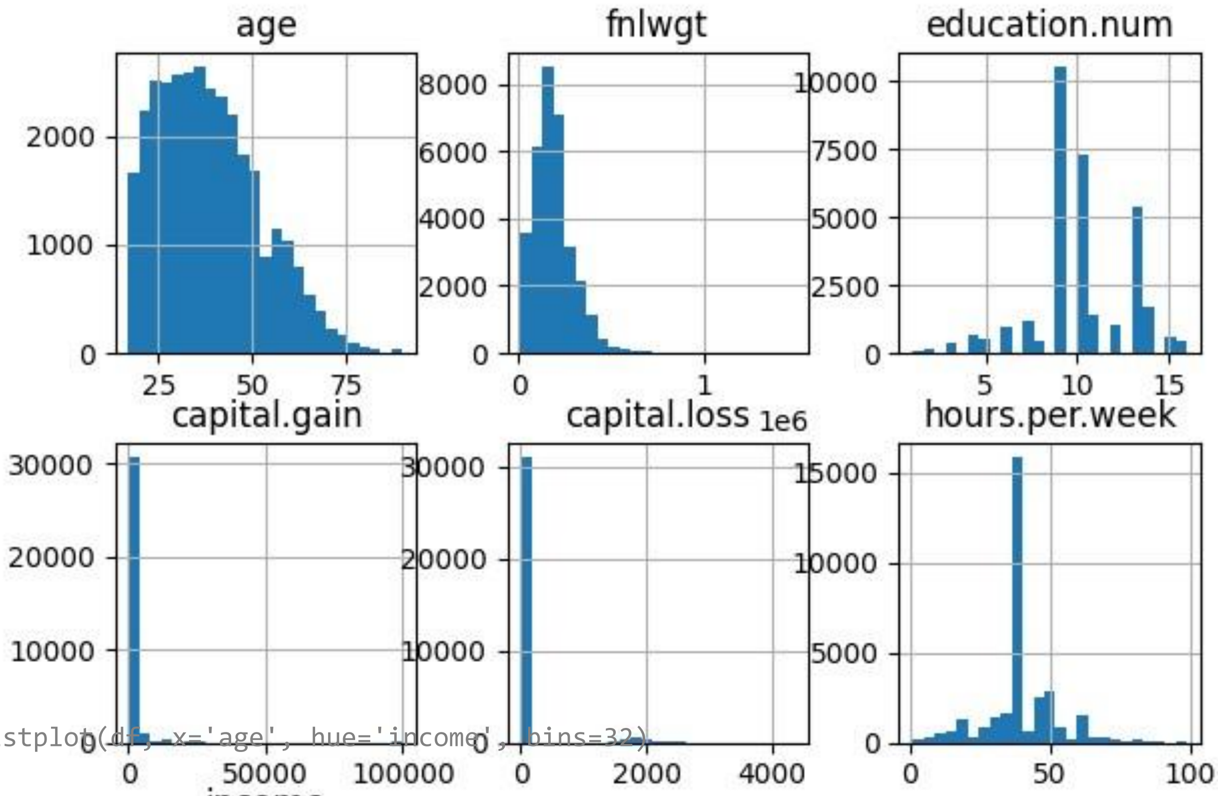
```
df['income'] = df['income'].astype('int')
```

```
<ipython-input-48-c88e10f6120e>:1: 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_
  df['income'] = df['income'].astype('int')
```

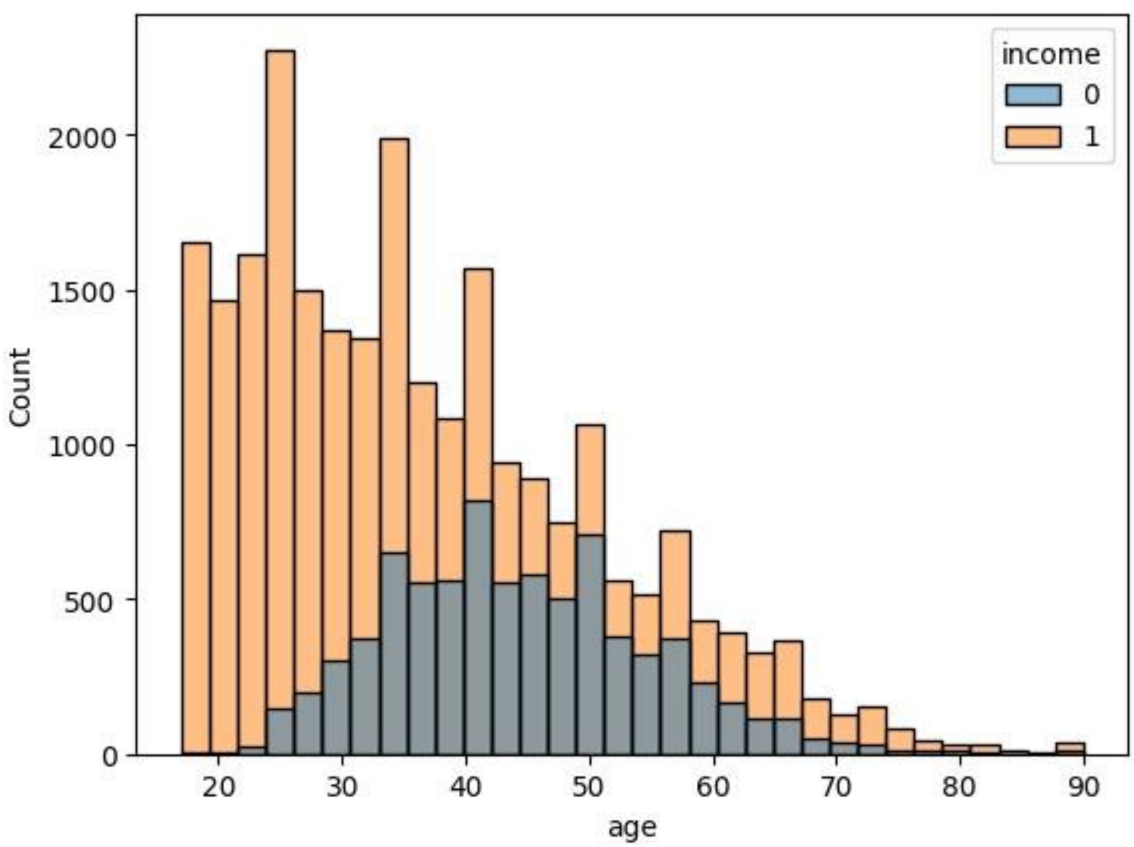
```
categorical = [col for col in df.columns if df[col].dtype == 'object' ]
numerical = [col for col in df.columns if df[col].dtype != 'object' ]
```

```
df[numerical].hist(bins=25, figsize=(7, 7))
plt.show()
```




```
sns.histplot(df, x='age', hue='income', bins=32)
```

```
<Axes: xlabel='age', ylabel='Count'>
```

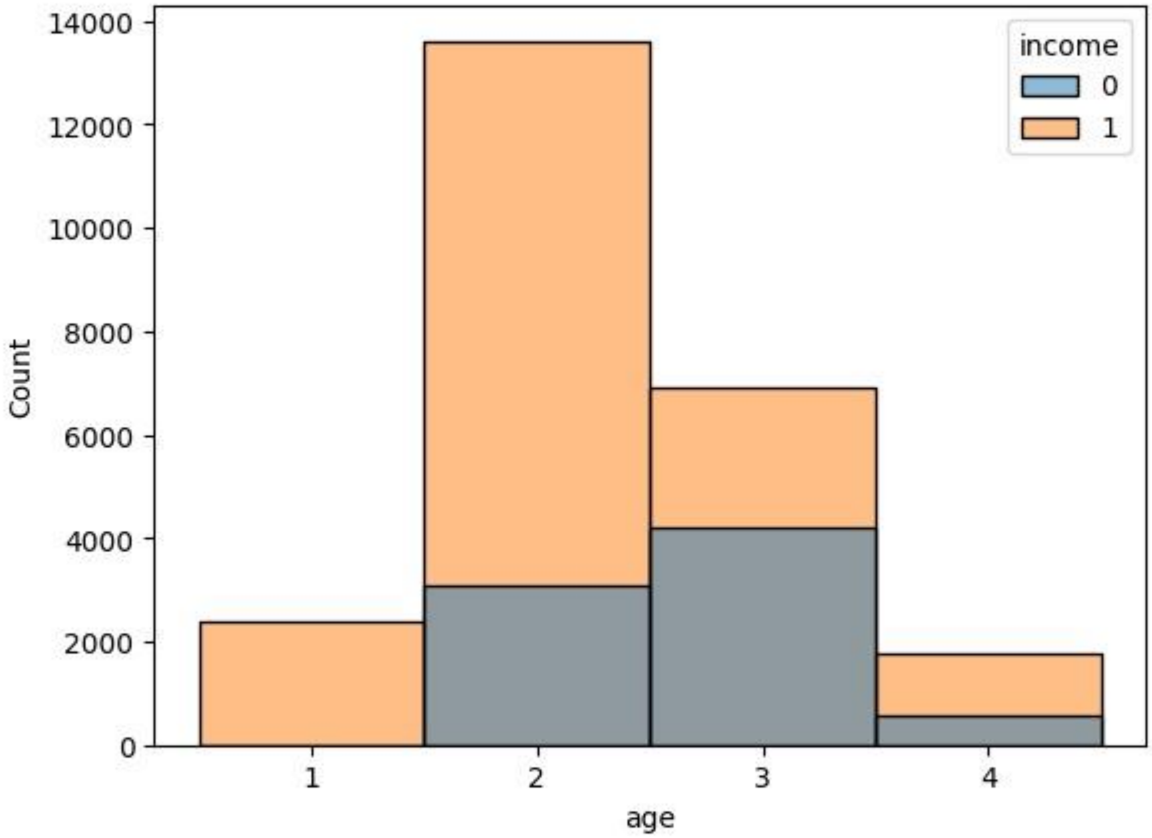



```
#describe ages in classes
def age_group(df):
    age_bins = [0, 20, 40, 60, float('inf')]
age_labels = ['1', '2', '3', '4']     df_age_range =
df.copy()     df_age_range['age'] =
```

```
pd.cut(df_age_range['age'], bins=age_bins,
labels=age_labels)     return df_age_range df =
age_group(df).copy()
```

```
sns.histplot(df, x='age', hue='income', bins= 32)
```

```
<Axes: xlabel='age', ylabel='Count'>
```



Missing Values

```
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     582
income               0
dtype: int64
```

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

y
()

race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
White	Female	0	4356	18	United-States	1
White	Female	0	3900	40	United-States	1
White	Female	0	3900	40	United-States	1
White	Female	0	3770	45	United-States	1
White	Male	0	3770	40	United-States	1

```
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
```

Data Preparation

```
from sklearn import preprocessing df_categorical =
df.select_dtypes(include=['object'])
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	native
1	Private	HS-grad	Widowed	Execmanagerial	Not-in-family	White	Female	Unit
3	Private	7th-8th	Divorced	Machineop-inspct	Unmarried	White	Female	Unit
4	Private	Somecollege	Separated	Profspecialty	Own-child	White	Female	Unit
5	Private	HS-grad	Divorced	Otherservice	Unmarried	White	Female	Unit
6	Private	10th	Separated	Admclerical	Unmarried	White	Male	Unit

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

	workclass	education	marital.status	occupation	relationship	race	sex	native.cou
1	2	11	6	3	1	4	0	

3 2 5 0 6 4 4 0

4 2 15 5 9 3 4 0

5 2 11 0 7 4 4 0 6 2 0 5 0

4 4 1

```
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	income	workc
1	4 132870		9	0	4356	18	1	

3 3 1403594 0 3900 40 1

4	3	26466310	0	3900	40	1	
5	2	216864	9	0	3770	45	1

```
df.info()
```

6	2	150601	6	0	3770	40	1

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30139 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
0     age             30139 non-null  category
1     fnlwgt          30139 non-null  int64
2     education.num   30139 non-null  int64
3     capital.gain    30139 non-null  int64
4     capital.loss    30139 non-null  int64
5     hours.per.week  30139 non-null  int64
6     income          30139 non-null  category
7     workclass       30139 non-null  int64
8     education       30139 non-null  int64
9     marital.status  30139 non-null  int64
10    occupation      30139 non-null  int64
11    relationship    30139 non-null  int64
12    race            30139 non-null  int64
13    sex             30139 non-null  int64    14
native.country  30139 non-null  int64   dtypes:
category(2), int64(13) memory usage: 3.3 MB
```

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	ed
1	4	132870	9	0	4356	18	2	

```
y.head()
```

```
1    1
3    1
4    1
5    1
6    1
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 Algo

```
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
 [[ 732  767]
[ 191 4338]]
              precision    recall  f1-score   support

           0       0.79      0.49      0.60      1499
1       0.85      0.96      0.90      4529

    accuracy                           0.84      6028
macro avg       0.82      0.72      0.75      6028
weighted avg       0.84      0.84      0.83      6028
```

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

```
accuracy score 0.8410749834107498
```

0s completed at 23:43 PM