Experiment No. 4
Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance: 14-08-2023
Date of Submission: 22-08-2023

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.

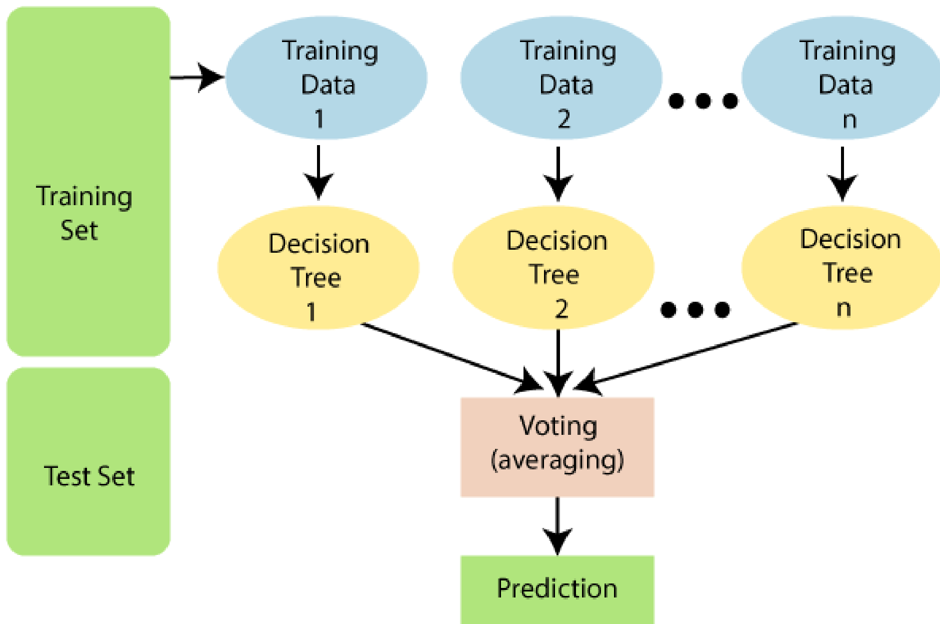
Theory:

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, Married-spouse-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, Outlying-US(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.

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

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

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

sns.set(style="whitegrid")
```

```
import warnings

warnings.filterwarnings('ignore')
```

```
data = 'income_evaluation.csv'

df = pd.read_csv(data)
```

```
# print the shape
print('The shape of the dataset : ', df.shape)
```

```
The shape of the dataset :  (26874, 15)
```

```
df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0

```
col_names = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation', 'relationship',
             'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'income']

df.columns = col_names

df.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'income'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26874 entries, 0 to 26873
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             26874 non-null  int64
 1   workclass       26874 non-null  object
 2   fnlwgt          26874 non-null  int64
```

```
 3   education       26874 non-null  object
 4   education_num   26874 non-null  int64
 5   marital_status  26874 non-null  object
 6   occupation      26874 non-null  object
 7   relationship    26874 non-null  object
 8   race            26874 non-null  object
 9   sex             26874 non-null  object
 10  capital_gain    26874 non-null  int64
 11  capital_loss    26874 non-null  int64
 12  hours_per_week  26874 non-null  int64
 13  native_country  26874 non-null  object
 14  income          26874 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.1+ MB
```

```
df.dtypes
```

```
age                int64
workclass         object
fnlwgt             int64
education         object
education_num      int64
marital_status    object
occupation        object
relationship      object
race              object
sex               object
capital_gain       int64
capital_loss       int64
hours_per_week     int64
native_country    object
income            object
dtype: object
```

```
df.describe()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
count	26874.000000	2.687400e+04	26874.000000	26874.000000	26874.000000	26874.000000
mean	38.614460	1.897317e+05	10.084692	1091.752995	86.985376	40.408648
std	13.672543	1.051297e+05	2.564925	7501.346726	402.254353	12.302287
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.179630e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.784250e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.368788e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

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

	count	mean	std	min	25%	50%	75%	max
age	26874.0	38.614460	13.672543	17.0	28.0	37.0	48.00	90.0
fnlwgt	26874.0	189731.746521	105129.680744	12285.0	117963.0	178425.0	236878.75	1484705.0
education_num	26874.0	10.084692	2.564925	1.0	9.0	10.0	12.00	16.0
capital_gain	26874.0	1091.752995	7501.346726	0.0	0.0	0.0	0.00	99999.0
capital_loss	26874.0	86.985376	402.254353	0.0	0.0	0.0	0.00	4356.0
hours_per_week	26874.0	40.408648	12.302287	1.0	40.0	40.0	45.00	99.0

```
df.describe(include='all')
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	ca
count	26874.000000	26874	2.687400e+04	26874	26874.000000	26874	26874	26874	26874	26874	26
unique	NaN	9	NaN	16	NaN	7	15	6	5	2	
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	White	Male	
freq	NaN	18698	NaN	8696	NaN	12304	3432	10824	22983	17977	

1
7
9

```
# check for missing values

df.isnull().sum()
```

```
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
```

```
#assert that there are no missing values in the dataframe

assert pd.notnull(df).all().all()
```

```
def initial_eda(df):
    if isinstance(df, pd.DataFrame):
        total_na = df.isna().sum().sum()
        print("Dimensions : %d rows, %d columns" % (df.shape[0], df.shape[1]))
        print("Total NA Values : %d " % (total_na))
        print("%38s %10s     %10s %10s" % ("Column Name", "Data Type", "#Distinct", "NA Values"))
        col_name = df.columns
        dtyp = df.dtypes
        uniq = df.nunique()
        na_val = df.isna().sum()
        for i in range(len(df.columns)):
            print("%38s %10s   %10s %10s" % (col_name[i], dtyp[i], uniq[i], na_val[i]))

    else:
        print("Expect a DataFrame but got a %15s" % (type(df)))
```

```
initial_eda(df)
```

```
Dimensions : 26874 rows, 15 columns
Total NA Values : 0
                           Column Name  Data Type      #Distinct  NA Values
                                   age      int64             72          0
                             workclass     object              9          0
                                fnlwgt      int64          18818          0
                             education     object             16          0
                         education_num      int64             16          0
                        marital_status     object              7          0
                            occupation     object             15          0
                          relationship     object              6          0
                                  race     object              5          0
                                   sex     object              2          0
                          capital_gain      int64            117          0
                          capital_loss      int64             90          0
                        hours_per_week      int64             94          0
                        native_country     object             42          0
                                income     object              2          0
```

```
categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :\n\n', categorical)
```

```
There are 9 categorical variables

The categorical variables are :

 ['workclass', 'education', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'native_country', 'income']
```

```
df[categorical].head()
```

	workclass	education	marital_status	occupation	relationship	race	sex	native_country	income
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	United-States	<=50K
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States	<=50K
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba	<=50K

```
for var in categorical:

    print(df[var].value_counts())
```

```
 Wife               1303
 Other-relative      795
Name: relationship, dtype: int64
 White                 22983
 Black                  2562
 Asian-Pac-Islander      842
 Amer-Indian-Eskimo      261
 Other                   226
Name: race, dtype: int64
 Male      17977
 Female     8897
Name: sex, dtype: int64
 United-States                 24087
 Mexico                          528
 ?                               482
 Philippines                     162
 Germany                         109
 Canada                          105
 Puerto-Rico                     103
 El-Salvador                      82
 England                          80
 Cuba                             80
 India                            74
 South                            69
 China                            65
 Jamaica                          64
 Italy                            56
 Dominican-Republic               55
 Guatemala                        54
 Vietnam                          53
 Japan                            53
 Poland                           53
 Columbia                         48
 Taiwan                           46
 Iran                             40
 Haiti                            38
 Portugal                         29
 Nicaragua                        29
 Peru                             24
 Greece                           24
 France                           23
 Ireland                          22
 Ecuador                          20
 Thailand                         16
 Cambodia                         16
 Hong                             12
 Yugoslavia                       12
 Trinadad&Tobago                  12
 Laos                             11
 Hungary                          10
 Outlying-US(Guam-USVI-etc)        9
 Honduras                          9
 Scotland                          9
 Holand-Netherlands                1
Name: native_country, dtype: int64
 <=50K     20438
 >50K       6436
Name: income, dtype: int64
```

```
for var in categorical:

     print(df[var].value_counts()/np.float(len(df)))
```

```
 Asian-Pac-Islander    0.031331
 Amer-Indian-Eskimo    0.009712
 Other                 0.008410
Name: race, dtype: float64
 Male      0.668937
 Female    0.331063
Name: sex, dtype: float64
 United-States                 0.896294
 Mexico                        0.019647
 ?                             0.017936
 Philippines                   0.006028
 Germany                       0.004056
 Canada                        0.003907
 Puerto-Rico                   0.003833
 El-Salvador                   0.003051
 England                       0.002977
 Cuba                          0.002977
 India                         0.002754
 South                         0.002568
 China                         0.002419
 Jamaica                       0.002381
 Italy                         0.002084
 Dominican-Republic            0.002047
 Guatemala                     0.002009
 Vietnam                       0.001972
 Japan                         0.001972
 Poland                        0.001972
 Columbia                      0.001786
 Taiwan                        0.001712
 Iran                          0.001488
 Haiti                         0.001414
 Portugal                      0.001079
 Nicaragua                     0.001079
 Peru                          0.000893
 Greece                        0.000893
 France                        0.000856
 Ireland                       0.000819
 Ecuador                       0.000744
 Thailand                      0.000595
 Cambodia                      0.000595
 Hong                          0.000447
 Yugoslavia                    0.000447
 Trinadad&Tobago               0.000447
 Laos                          0.000409
 Hungary                       0.000372
 Outlying-US(Guam-USVI-etc)    0.000335
 Honduras                      0.000335
 Scotland                      0.000335
 Holand-Netherlands            0.000037
Name: native_country, dtype: float64
 <=50K    0.760512
 >50K     0.239488
Name: income, dtype: float64
```

```
# check for missing values

df['income'].isnull().sum()
```

```
0
```

```
# view number of unique values

df['income'].nunique()
```

```
2
```

```
# view the unique values

df['income'].unique()
```

```
array([' <=50K', ' >50K'], dtype=object)
```

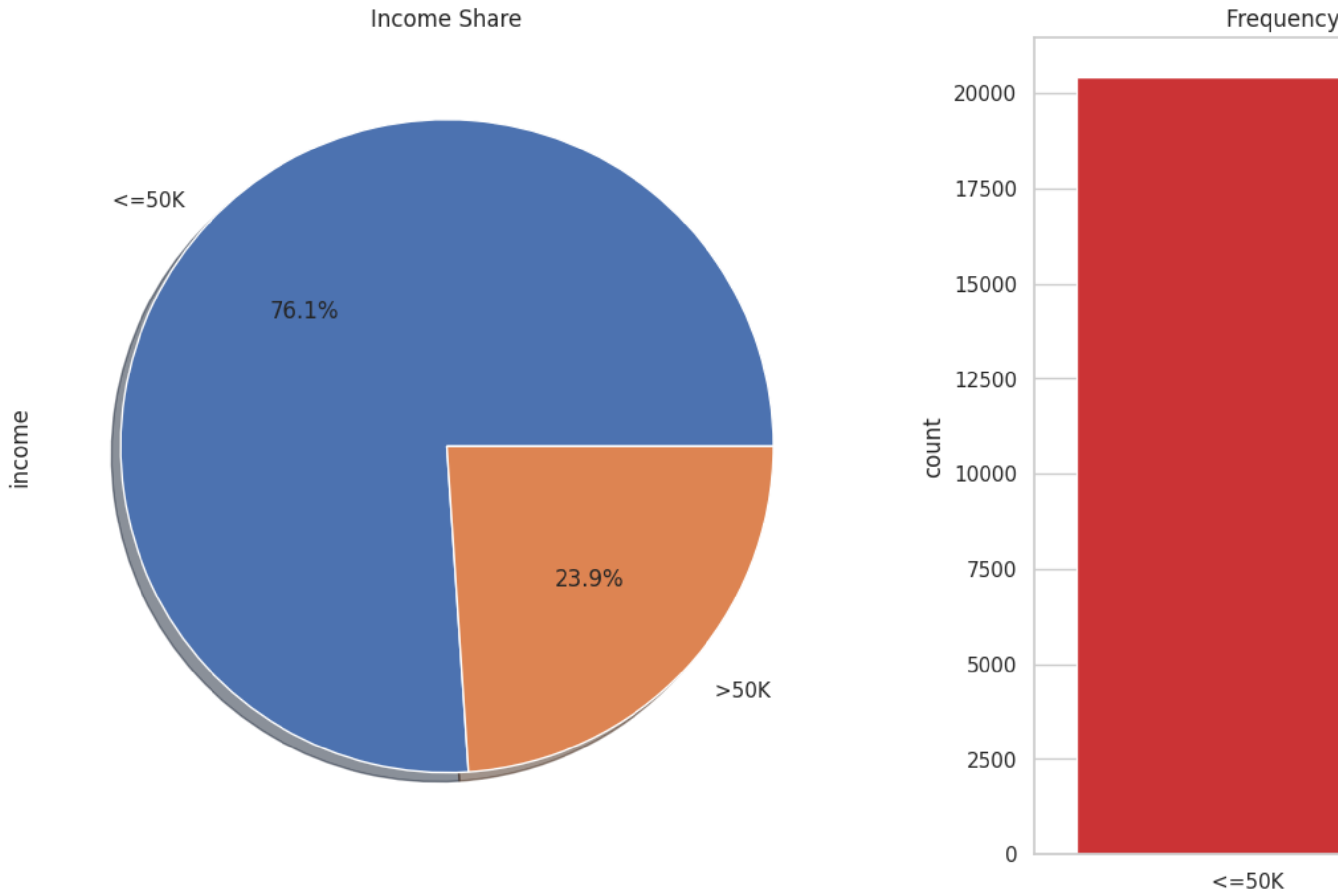
```
# view the frequency distribution of values

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

```
 <=50K    20438
 >50K      6436
Name: income, dtype: int64
```

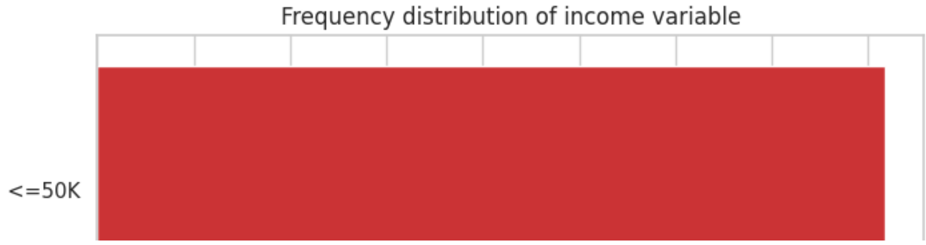
```
# view percentage of frequency distribution of values

df['income'].value_counts()/len(df)
```

```
<=50K    0.760512
>50K     0.239488
Name: income, dtype: float64
```

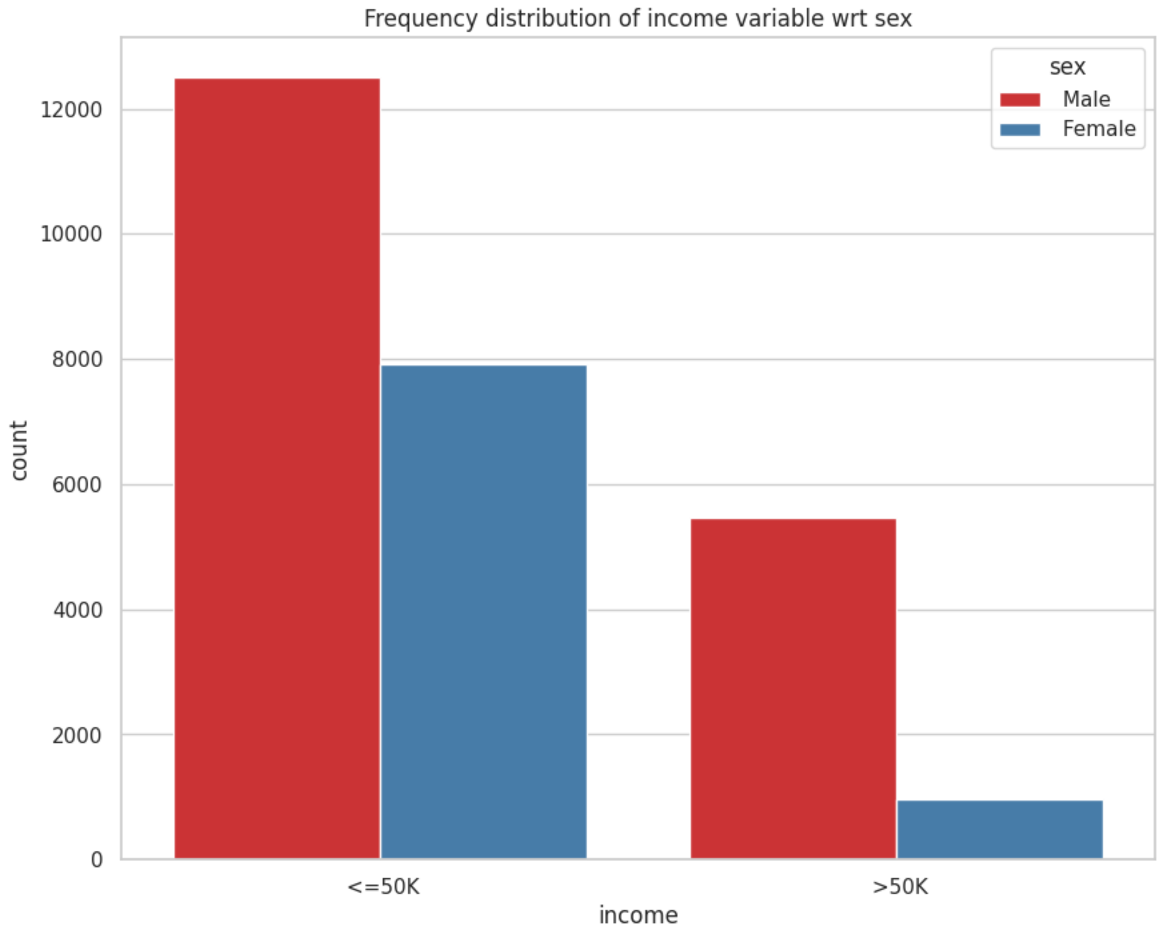
```
# visualize frequency distribution of income variable

f,ax=plt.subplots(1,2,figsize=(18,8))

ax[0] = df['income'].value_counts().plot.pie(explode=[0,0],autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('Income Share')


#f, ax = plt.subplots(figsize=(6, 8))
ax[1] = sns.countplot(x="income", data=df, palette="Set1")
ax[1].set_title("Frequency distribution of income variable")

plt.show()
```



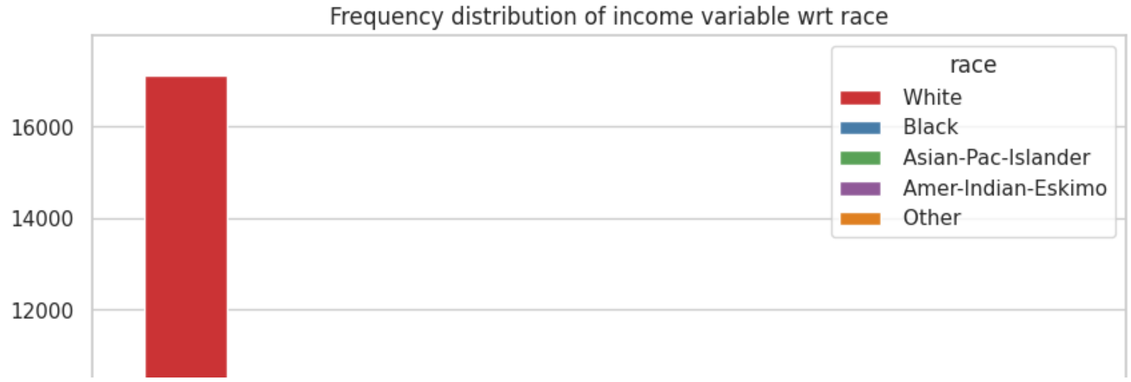

```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(y="income", data=df, palette="Set1")
ax.set_title("Frequency distribution of income variable")
plt.show()
```




```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.countplot(x="income", hue="sex", data=df, palette="Set1")
ax.set_title("Frequency distribution of income variable wrt sex")
plt.show()
```




```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.countplot(x="income", hue="race", data=df, palette="Set1")
ax.set_title("Frequency distribution of income variable wrt race")
plt.show()
```




```
# check number of unique labels

df.workclass.nunique()
```

```
9
```

```
# view the unique labels

df.workclass.unique()
```

```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

```
# view frequency distribution of values

df.workclass.value_counts()
```

```
 Private             18698
 Self-emp-not-inc     2117
 Local-gov            1753
 ?                    1502
 State-gov            1066
 Self-emp-inc          927
 Federal-gov           796
 Without-pay            10
 Never-worked            5
Name: workclass, dtype: int64
```

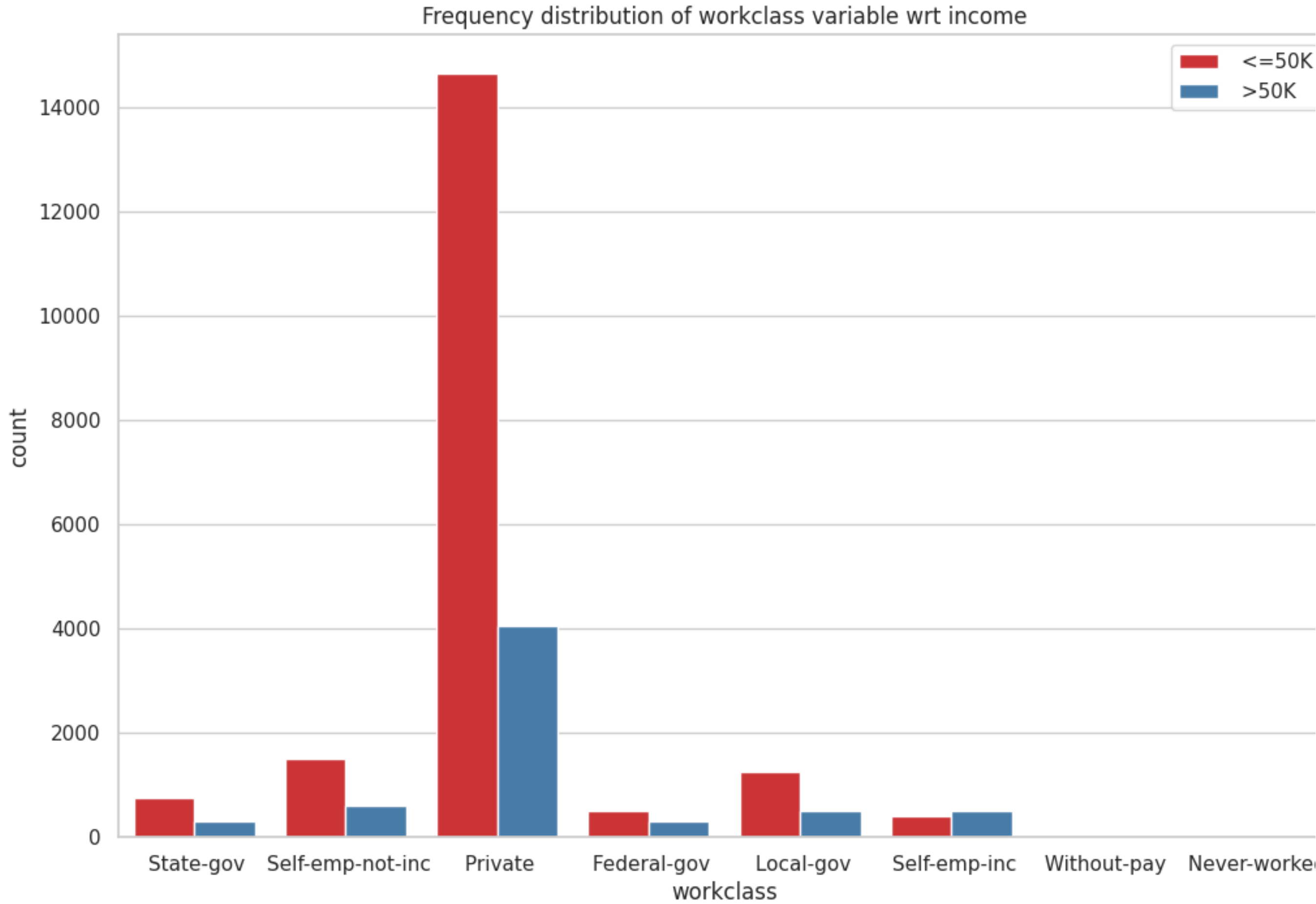
```
# replace '?' values in workclass variable with `NaN`

df['workclass'].replace(' ?', np.NaN, inplace=True)
```

```
# again check the frequency distribution of values in workclass variable

df.workclass.value_counts()
```

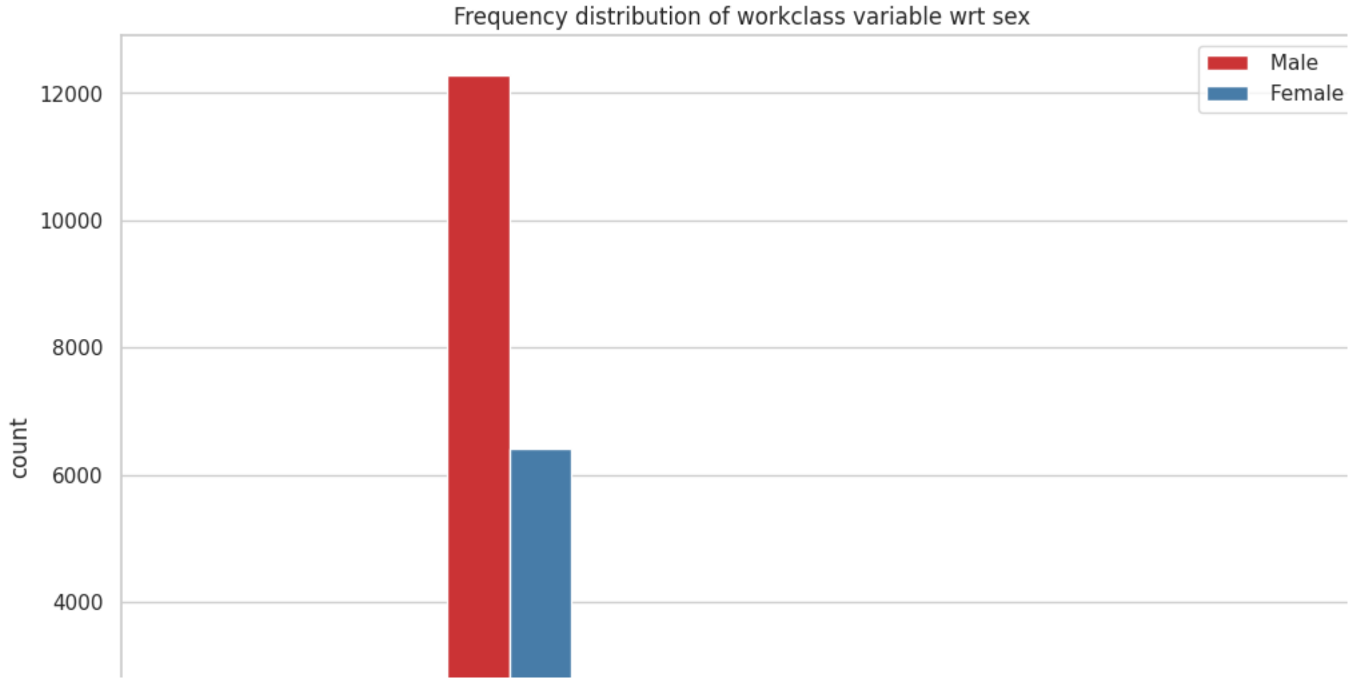
```
 Private             18698
 Self-emp-not-inc     2117
 Local-gov            1753
 State-gov            1066
 Self-emp-inc          927
 Federal-gov           796
 Without-pay            10
 Never-worked            5
Name: workclass, dtype: int64
```

```
f, ax = plt.subplots(figsize=(10, 6))
ax = df.workclass.value_counts().plot(kind="bar", color="green")
ax.set_title("Frequency distribution of workclass variable")
ax.set_xticklabels(df.workclass.value_counts().index, rotation=30)
plt.show()
```


```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.countplot(x="workclass", hue="income", data=df, palette="Set1")
ax.set_title("Frequency distribution of workclass variable wrt income")
ax.legend(loc='upper right')
plt.show()
```




```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.countplot(x="workclass", hue="sex", data=df, palette="Set1")
ax.set_title("Frequency distribution of workclass variable wrt sex")
ax.legend(loc='upper right')
plt.show()
```




```
# check number of unique labels

df.occupation.nunique()
```

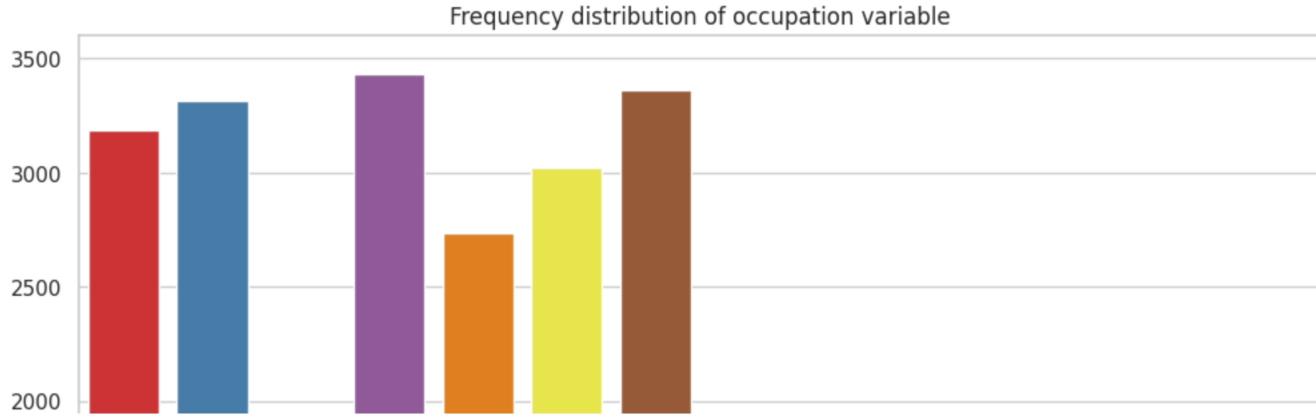
    15

```
# view unique labels

df.occupation.unique()
```

```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' ?', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

```
df['occupation'].replace(' ?', np.NaN, inplace=True)
```

```
# visualize frequency distribution of `occupation` variable

f, ax = plt.subplots(figsize=(12, 8))
ax = sns.countplot(x="occupation", data=df, palette="Set1")
ax.set_title("Frequency distribution of occupation variable")
ax.set_xticklabels(df.occupation.value_counts().index, rotation=30)
plt.show()
```




```
# replace '?' values in native_country variable with `NaN`

df['native_country'].replace(' ?', np.NaN, inplace=True)
```

```
# visualize frequency distribution of `native_country` variable

f, ax = plt.subplots(figsize=(16, 12))
ax = sns.countplot(x="native_country", data=df, palette="Set1")
ax.set_title("Frequency distribution of native_country variable")
ax.set_xticklabels(df.native_country.value_counts().index, rotation=90)
plt.show()
```


```
df[categorical].isnull().sum()
```

```
workclass         1502
education            0
marital_status       0
occupation        1507
relationship         0
race                 0
sex                  0
native_country     482
income               0
dtype: int64
```

```
# check for cardinality in categorical variables

for var in categorical:

    print(var, ' contains ', len(df[var].unique()), ' labels')
```

```
workclass  contains  9  labels
education  contains  16  labels
marital_status  contains  7  labels
occupation  contains  15  labels
relationship  contains  6  labels
race  contains  5  labels
sex  contains  2  labels
native_country  contains  42  labels
income  contains  2  labels
```

```
numerical = [var for var in df.columns if df[var].dtype!='O']

print('There are {} numerical variables\n'.format(len(numerical)))

print('The numerical variables are :\n\n', numerical)
```

```
There are 6 numerical variables

The numerical variables are :

 ['age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss', 'hours_per_week']
```

```
df[numerical].head()
```

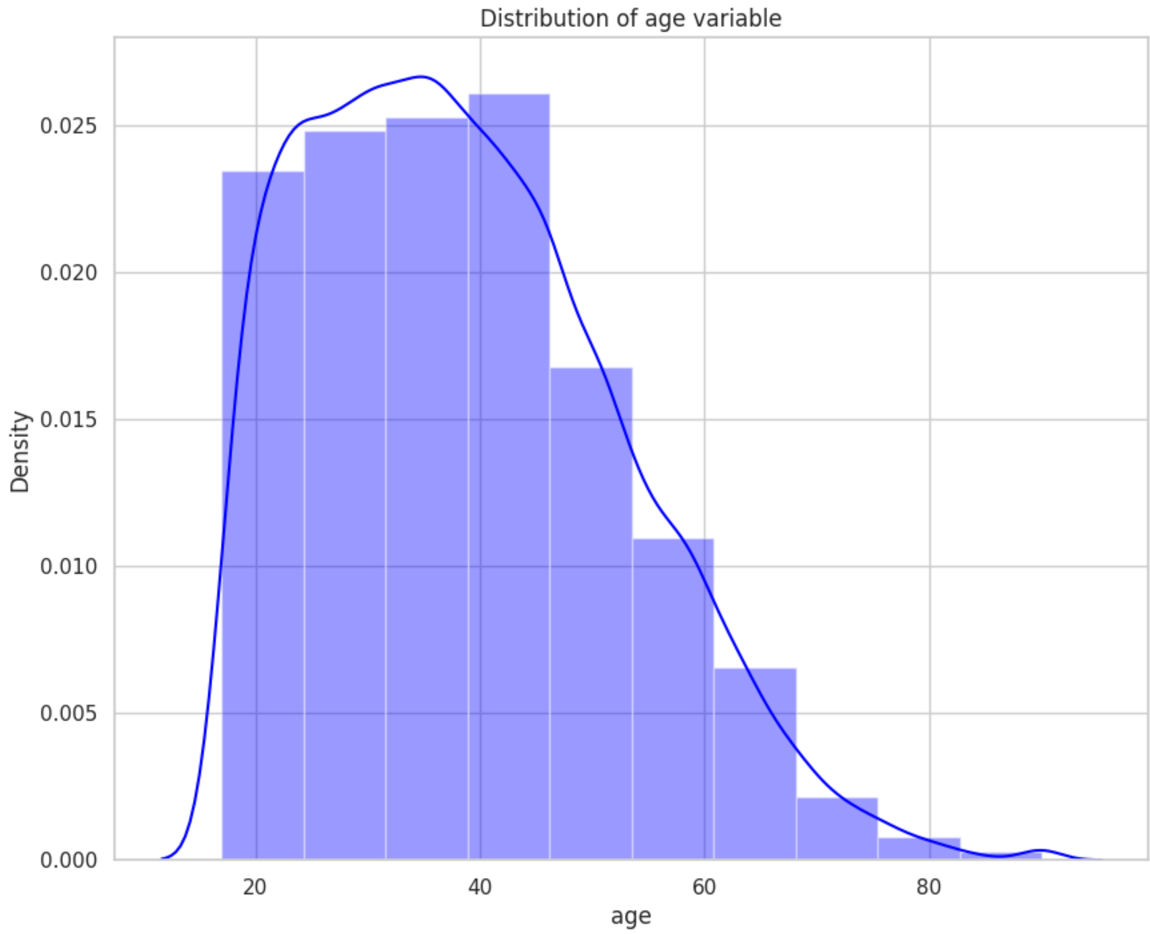
	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40

```
df[numerical].isnull().sum()
```

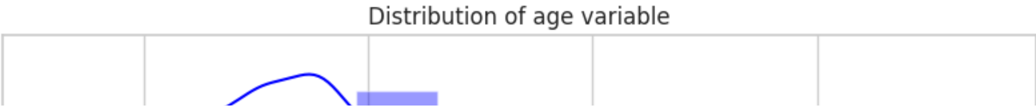
```
age               0
fnlwgt            0
education_num     0
capital_gain      0
capital_loss      0
hours_per_week    0
dtype: int64
```

```
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
ax = sns.distplot(x, bins=10, color='blue')
```

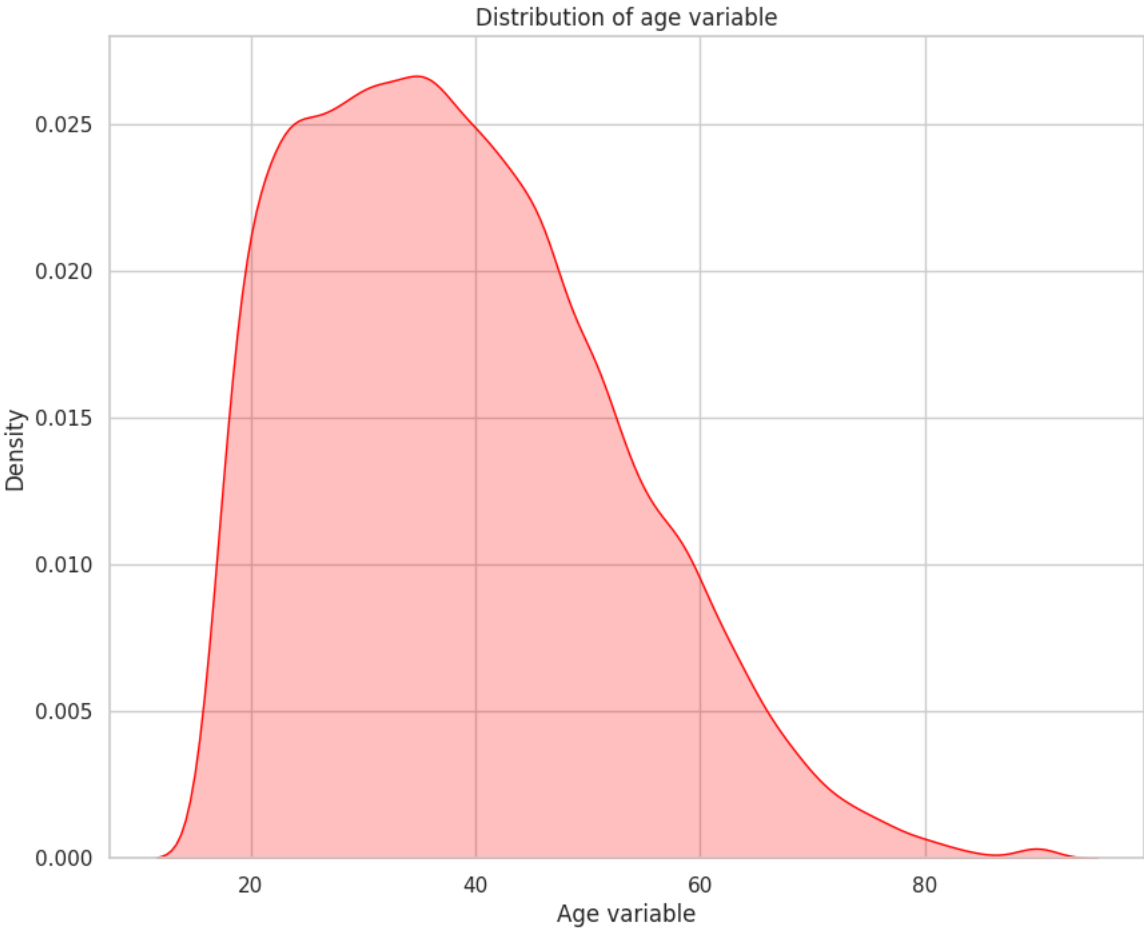
```
ax.set_title("Distribution of age variable")
plt.show()
```



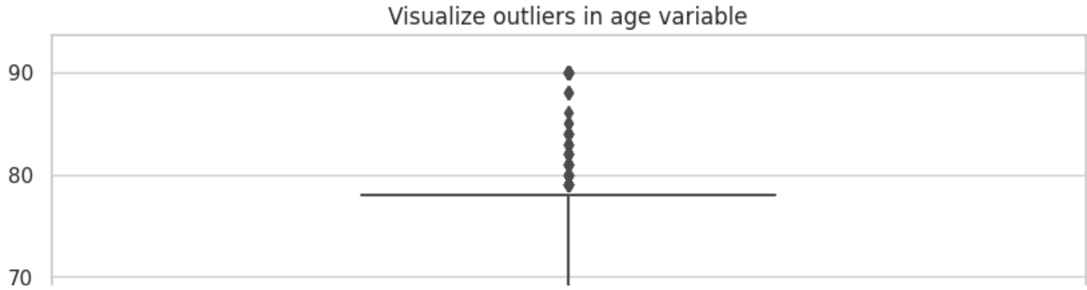

```
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
x = pd.Series(x, name="Age variable")
ax = sns.distplot(x, bins=10, color='blue')
ax.set_title("Distribution of age variable")
plt.show()
```



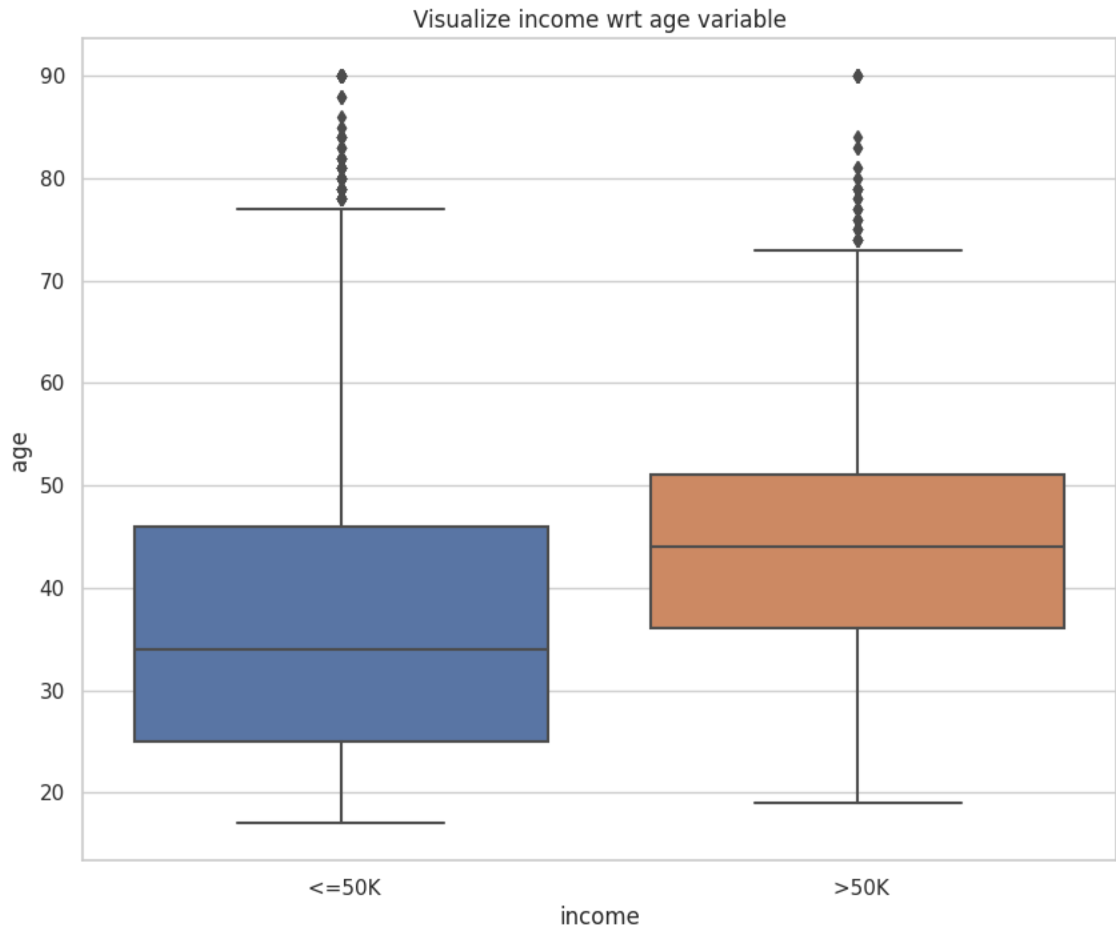

```
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
x = pd.Series(x, name="Age variable")
ax = sns.kdeplot(x, shade=True, color='red')
ax.set_title("Distribution of age variable")
plt.show()
```



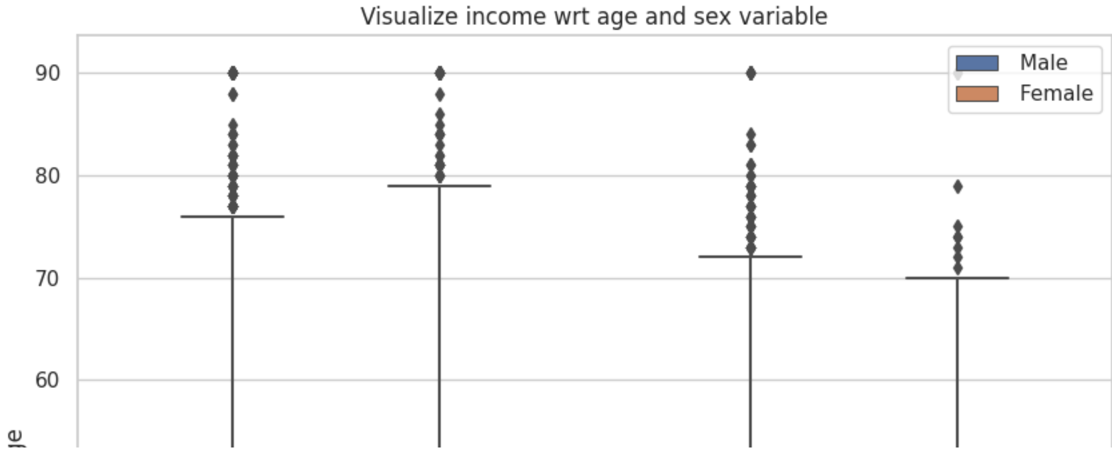

```
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
ax = sns.boxplot(x)
ax.set_title("Visualize outliers in age variable")
plt.show()
```



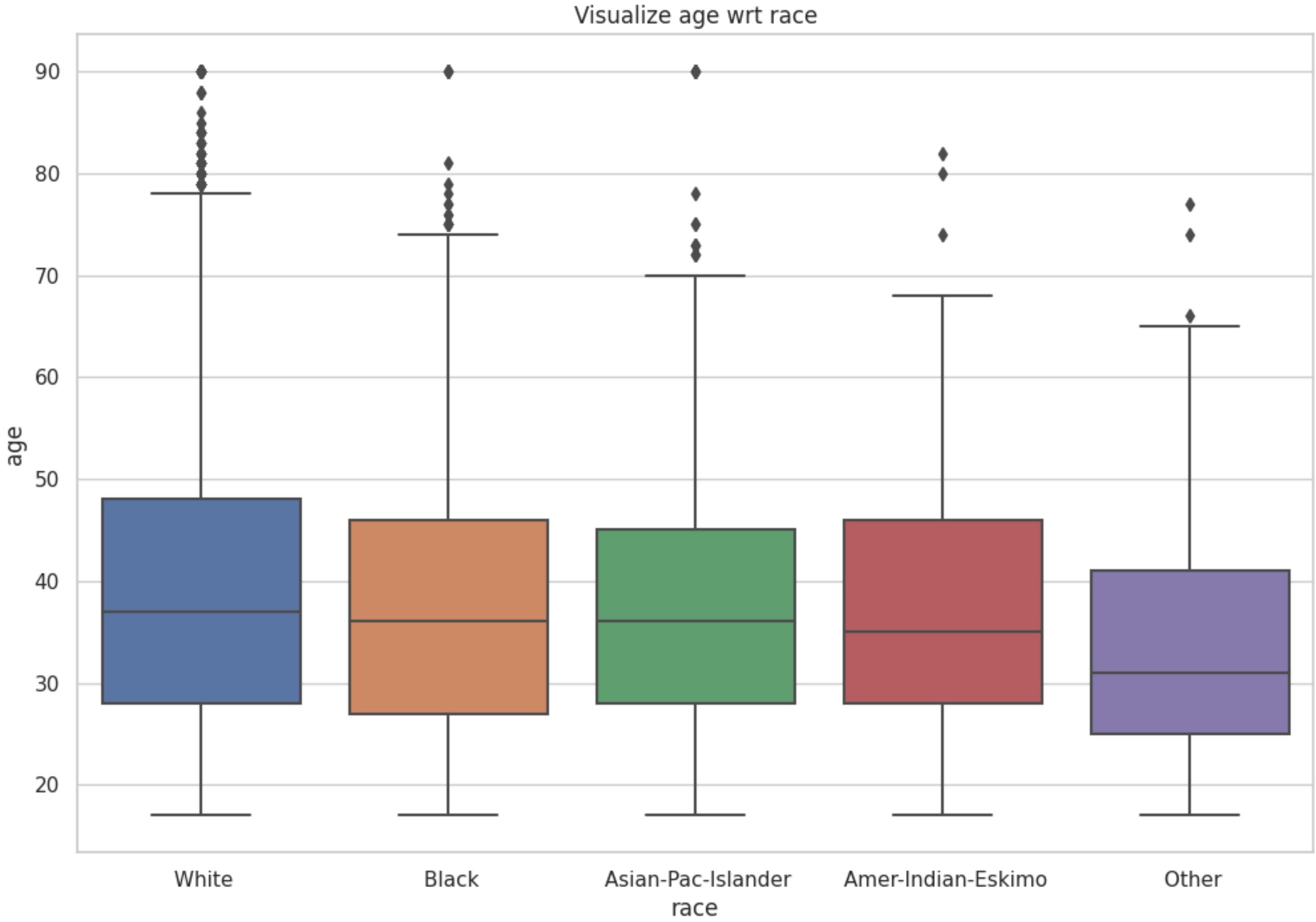

```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(x="income", y="age", data=df)
ax.set_title("Visualize income wrt age variable")
plt.show()
```




```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(x="income", y="age", hue="sex", data=df)
ax.set_title("Visualize income wrt age and sex variable")
ax.legend(loc='upper right')
plt.show()
```




```
plt.figure(figsize=(12,8))
sns.boxplot(x ='race', y="age", data = df)
plt.title("Visualize age wrt race")
plt.show()
```




```
# plot correlation heatmap to find out correlations

df.corr().style.format("{:.4}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
age	1.0	-0.07394	0.0367	0.07677	0.05622	0.06902
fnlwgt	-0.07394	1.0	-0.04467	0.002065	-0.01331	-0.01764
education_num	0.0367	-0.04467	1.0	0.1235	0.08034	0.148
capital_gain	0.07677	0.002065	0.1235	1.0	-0.03147	0.07628
capital_loss	0.05622	-0.01331	0.08034	-0.03147	1.0	0.05701
hours_per_week	0.06902	-0.01764	0.148	0.07628	0.05701	1.0

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

```
y = df['income']
```

```
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 = 0)
```

```
X_train.shape, X_test.shape
```

```
((18811, 14), (8063, 14))
```

```
categorical = [col for col in X_train.columns if X_train[col].dtypes == 'O']

categorical
```

```
['workclass',
 'education',
 'marital_status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native_country']
```

```
numerical = [col for col in X_train.columns if X_train[col].dtypes != 'O']

numerical
```

```
['age',
 'fnlwgt',
 'education_num',
 'capital_gain',
 'capital_loss',
 'hours_per_week']
```

```
X_train[categorical].isnull().mean()
```

```
workclass         0.056084
education         0.000000
marital_status    0.000000
occupation        0.056244
relationship      0.000000
race              0.000000
sex               0.000000
native_country    0.017064
dtype: float64
```

```
for col in categorical:
    if X_train[col].isnull().mean()>0:
        print(col, (X_train[col].isnull().mean()))
```

```
workclass 0.0560842060496518
occupation 0.05624368720429536
native_country 0.017064483546860878
```

```
# impute missing categorical variables with most frequent value

for df2 in [X_train, X_test]:
    df2['workclass'].fillna(X_train['workclass'].mode()[0], inplace=True)
    df2['occupation'].fillna(X_train['occupation'].mode()[0], inplace=True)
    df2['native_country'].fillna(X_train['native_country'].mode()[0], inplace=True)
```

```
# check missing values in categorical variables in X_train

X_train[categorical].isnull().sum()
```

```
workclass         0
education         0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
native_country    0
dtype: int64
```

```
X_test[categorical].isnull().sum()
```

```
workclass         0
education         0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
native_country    0
dtype: int64
```

```
X_train.isnull().sum()
```

```
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
dtype: int64
```

```
X_test.isnull().sum()
```

```
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
dtype: int64
```

```
X_test.isnull().sum()
```

```
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
dtype: int64
```

```
X_train[categorical].head()
```

	workclass	education	marital_status	occupation	relationship	race	sex	native_country
2248	Private	Some-college	Never-married	Handlers-cleaners	Own-child	White	Male	United-States
14568	Private	Bachelors	Divorced	Prof-specialty	Unmarried	White	Female	United-States
684	State-gov	11th	Never-married	Adm-clerical	Own-child	White	Female	United-States
10731	Local-gov	Assoc-voc	Married-civ-spouse	Protective-serv	Husband	White	Male	United-States
10013	Private	9th	Widowed	Prof-specialty	Unmarried	White	Female	United-States

```
# import category encoders
!pip install category_encoders
import category_encoders as ce
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.2-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.8/81.8 kB 2.6 MB/s eta 0:00:00
```

```
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.23.5)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.3)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.0)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.5.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_enco
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (202
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encode
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encode
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.2
```

```python
# encode categorical variables with one-hot encoding

encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'marital_status', 'occupation', 'relationship',
                                 'race', 'sex', 'native_country'])

X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

```python
cols = X_train.columns
```

```python
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

```python
X_train = pd.DataFrame(X_train, columns=[cols])
```

```python
X_test = pd.DataFrame(X_test, columns=[cols])
```

```python
# import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier



# instantiate the classifier

rfc = RandomForestClassifier(random_state=0)



# fit the model

rfc.fit(X_train, y_train)



# Predict the Test set results

y_pred = rfc.predict(X_test)



# Check accuracy score

from sklearn.metrics import accuracy_score

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with 10 decision-trees : 0.8541
```

```python
rfc_100 = RandomForestClassifier(n_estimators=100, random_state=0)



# fit the model to the training set

rfc_100.fit(X_train, y_train)
```

```
# Predict on the test set results

y_pred_100 = rfc_100.predict(X_test)


# Check accuracy score

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred_100)))
```

```
Model accuracy score with 100 decision-trees : 0.8541
```

```
clf = RandomForestClassifier(n_estimators=100, random_state=0)


# fit the model to the training set

clf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=0)
```

Conclusions:

Random Forest Classifier is a versatile and powerful machine learning algorithm commonly used for classification tasks. It is an ensemble learning method that builds multiple decision trees during the training process and combines their predictions to produce a more accurate and robust final prediction

Initially we considered the entire dataset and used the visualization technique.
Initially we create a Income share pie diagram and a bar graph of frequency.
Then we plot various distribution frequency graphs and then we draw down various conclusions
which will be useful for further consideration
We also consider age distribution.
Then after considering these attributes we then train the model accordingly and the we use Random Forest Classifier with a max depth of 10
This depth is an important hyperparameter that can significantly impact the model's performance and behavior. The depth of a decision tree refers to the number of levels or splits it can make from the root node to a leaf node.