Census Income Project

In [1]:
```
# Importing the necessary libraries for loading the data
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Creating the variable df and loading the dataset to the variable df
df = pd.read_csv('census.csv')
df
```

Out[2]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	Race	S
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fem
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Fem
...	...	...	...	...	...	...	...	...	...	
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Fem
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	M
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Fem
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	M
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Fem

32560 rows × 15 columns

In [3]:
```
# Checking the dataset for null values, if null value present we need to remove the null v
df.isnull().sum()
```

Out[3]:
```
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
```

From the above table we can say that there is no null values in the dataset

In [4]:
```
# cheking the datatype of df dataframe columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Age             32560 non-null  int64 
 1   Workclass       32560 non-null  object
 2   Fnlwgt          32560 non-null  int64 
 3   Education       32560 non-null  object
 4   Education_num   32560 non-null  int64 
 5   Marital_status  32560 non-null  object
 6   Occupation      32560 non-null  object
 7   Relationship    32560 non-null  object
 8   Race            32560 non-null  object
 9   Sex             32560 non-null  object
 10  Capital_gain    32560 non-null  int64 
 11  Capital_loss    32560 non-null  int64 
 12  Hours_per_week  32560 non-null  int64 
 13  Native_country  32560 non-null  object
 14  Income          32560 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

int and object data type present in the dataset df

In [6]:
```
# checking the shape of the dataframe df
df.shape
```

Out[6]:
```
(32560, 15)
```

df dataset as 32560 rows and 15 columns

In [7]:
```
# Checking the counts of target column
income = df['Income'].value_counts(normalize=True)
round(income * 100, 2).astype('str') + (' %')
```

Out[7]:
```
 <=50K    75.92 %
 >50K     24.08 %
Name: Income, dtype: object
```

The df dataset is unbalanced, as the dependent feature or the label column 'income' contains 75.92% values have income less than 50k and 24.08% values have income more than 50k.

In [8]:
```
# to know the statistical data we use describe function
df.describe()
```

Out[8]:

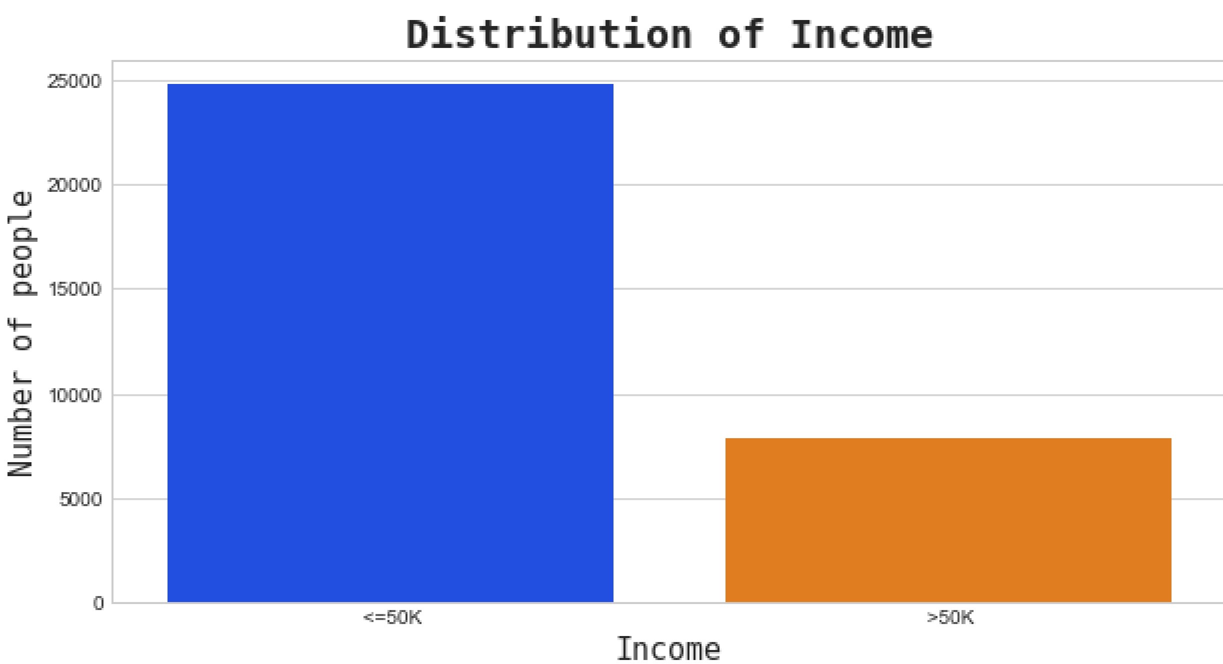
	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week
count	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581634	1.897818e+05	10.080590	1077.615172	87.306511	40.437469
std	13.640642	1.055498e+05	2.572709	7385.402999	402.966116	12.347618

	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

- Number of counts in each columns are same which means there is no null value in the data set
- by seeing the 75% percentail value and max value we can say that outliers are present in the dataset which need to be removed
- With the help of above table we can know the statistical information of each columns in the data set

In [9]:

```
# Creating a barplot for 'Income'
import matplotlib.pyplot as plt
import seaborn as sns
income = df['Income'].value_counts()

plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(10, 5))
sns.barplot(income.index, income.values, palette='bright')
plt.title('Distribution of Income', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Income', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=10)
plt.show()
```

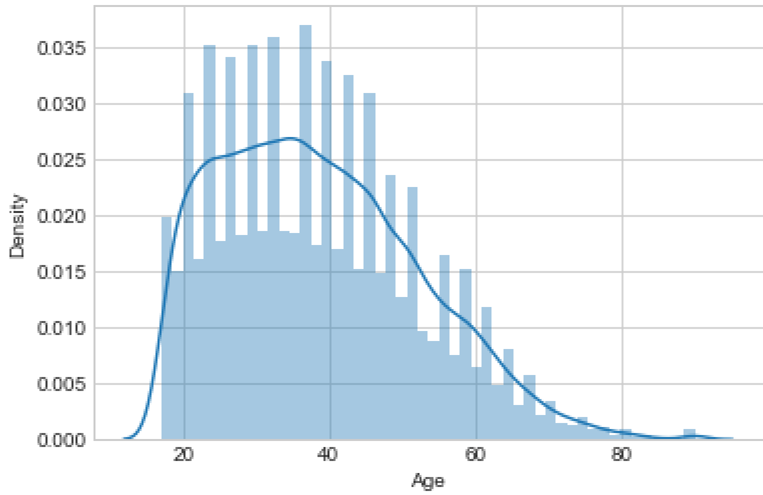


From graph we can say that the number of people whose income below 50k is more

In [10]:

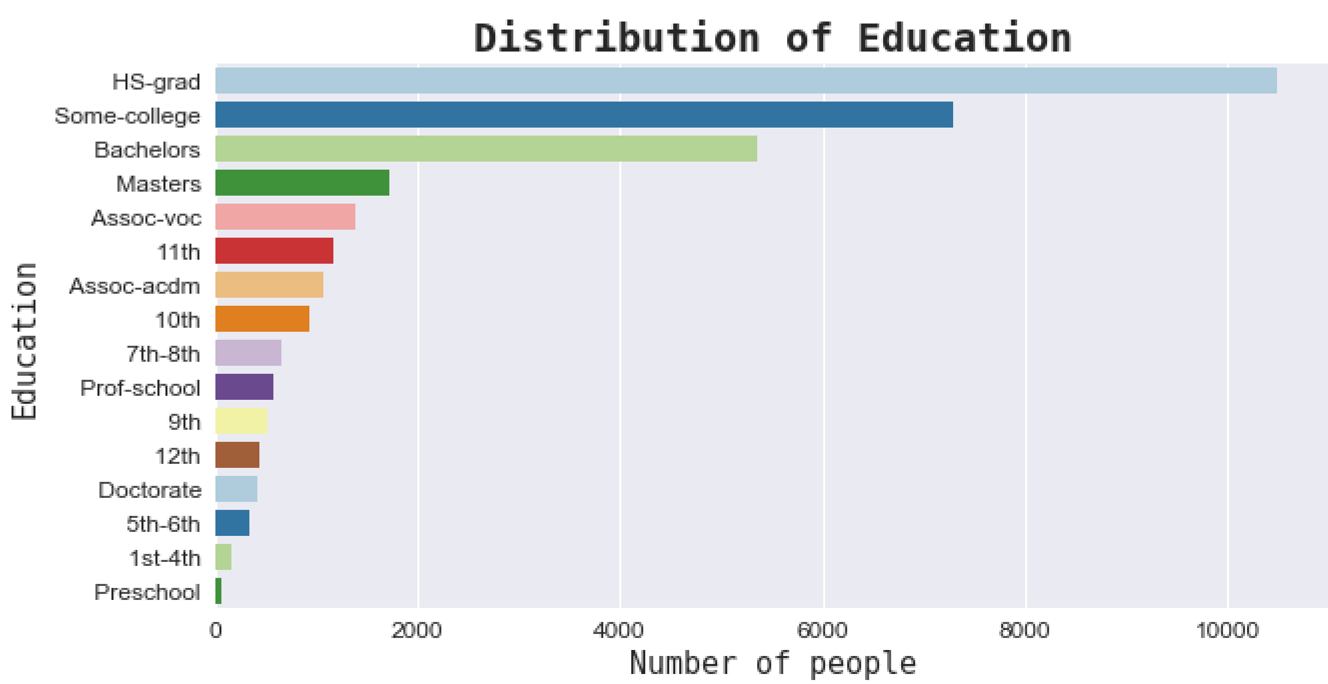
```
sns.distplot(df['Age'],kde=True)
```

Out[10]: <AxesSubplot:xlabel='Age', ylabel='Density'>



From the graph , the people between the age 20 to 40 is more

In [11]:
```
# Creating a barplot for 'Education'
edu = df['Education'].value_counts()

plt.style.use('seaborn')
plt.figure(figsize=(10, 5))
sns.barplot(edu.values, edu.index, palette='Paired')
plt.title('Distribution of Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Education', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.show()
```

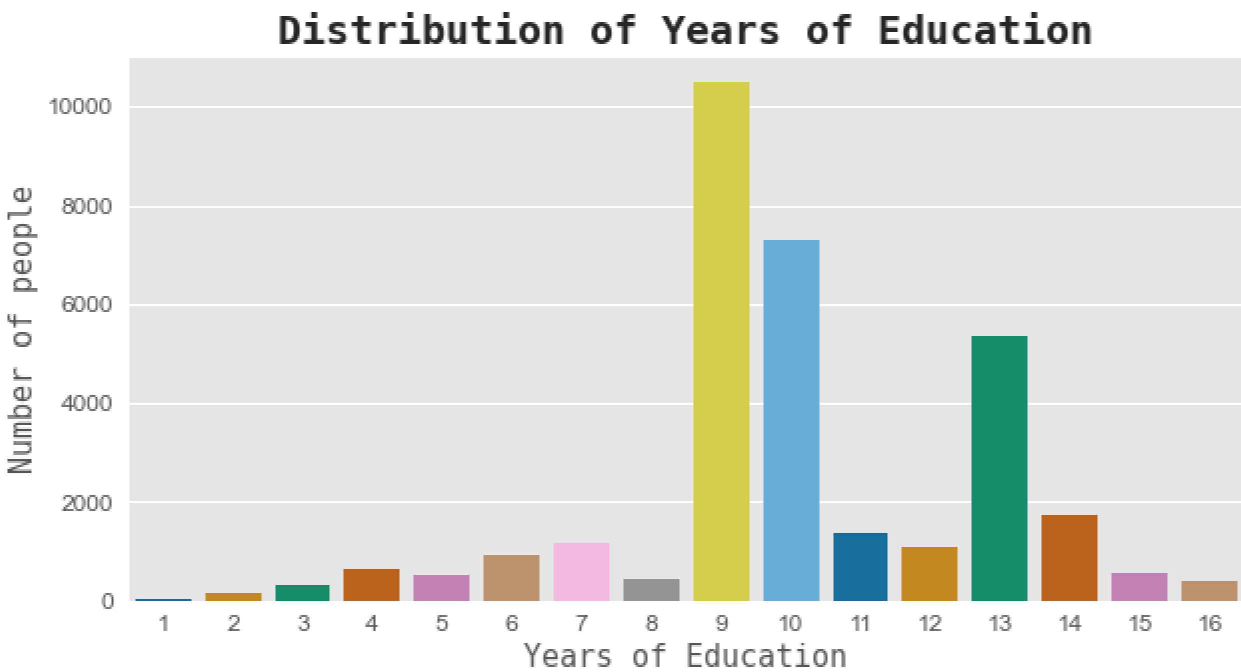


From graph we can say that , the maximum people are having HS grad education

very less number of people having the preschool education

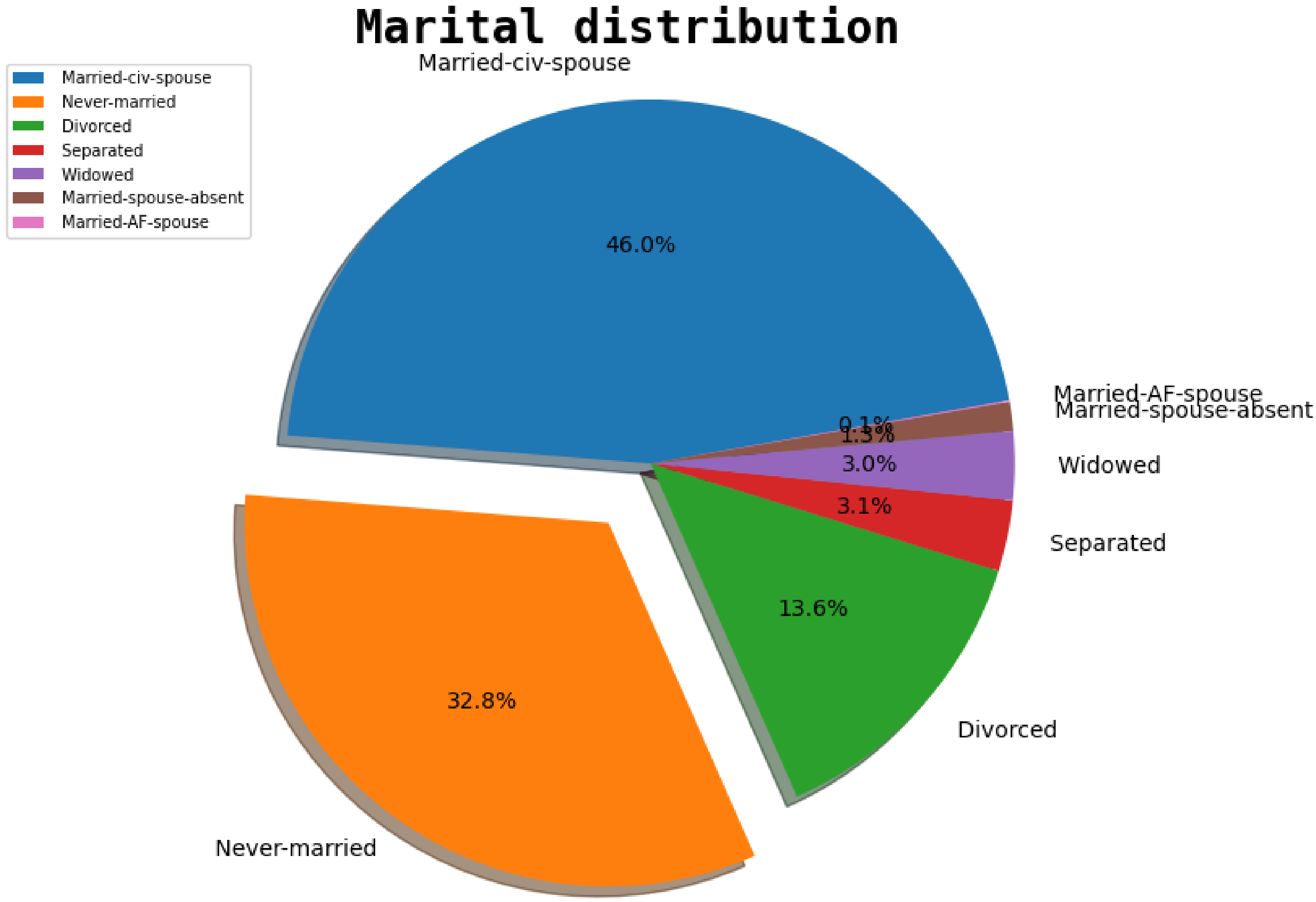
In [12]:
```
# Creating a barplot for 'Years of Education'
edu_num = df['Education_num'].value_counts()

plt.style.use('ggplot')
```

```
plt.figure(figsize=(10, 5))
sns.barplot(edu_num.index, edu_num.values, palette='colorblind')
plt.title('Distribution of Years of Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Years of Education', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.show()
```



In [13]:

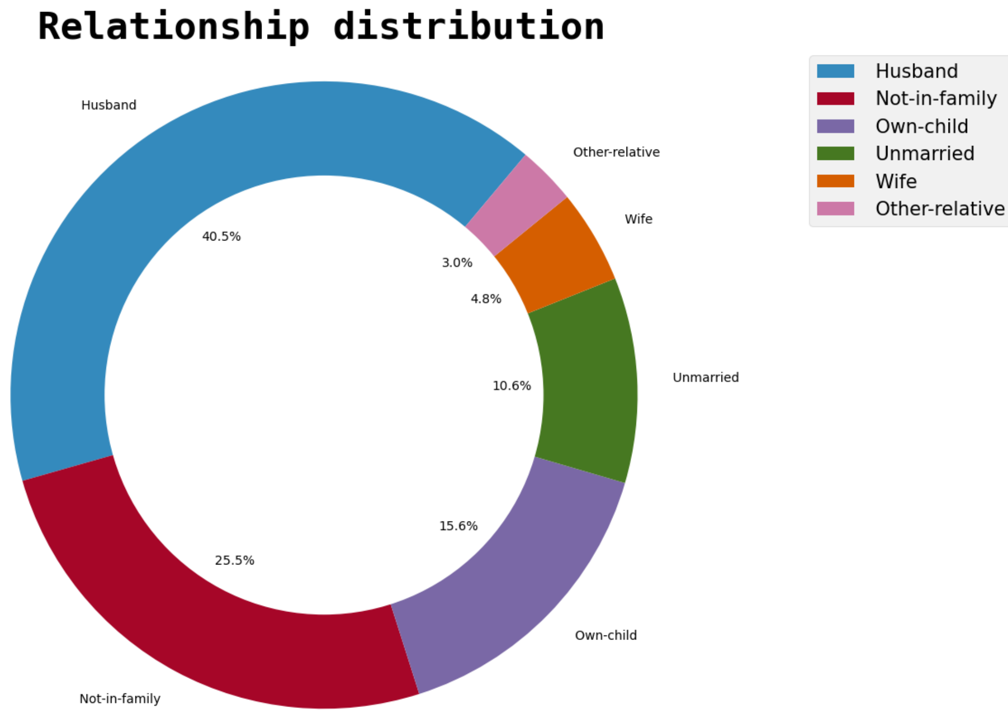
```
# Creating a pie chart for 'Marital status'
marital = df['Marital_status'].value_counts()

plt.style.use('default')
plt.figure(figsize=(10, 7))
plt.pie(marital.values, labels=marital.index, startangle=10, explode=(
    0, 0.20, 0, 0, 0, 0, 0), shadow=True, autopct='%1.1f%%')
plt.title('Marital distribution', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.legend()
plt.legend(prop={'size': 7})
plt.axis('equal')
plt.show()
```



From graph , the Married-civ-spouse is having maximum percentage 46% , never married 32.8% and respectively as showen in the graph

In [14]:

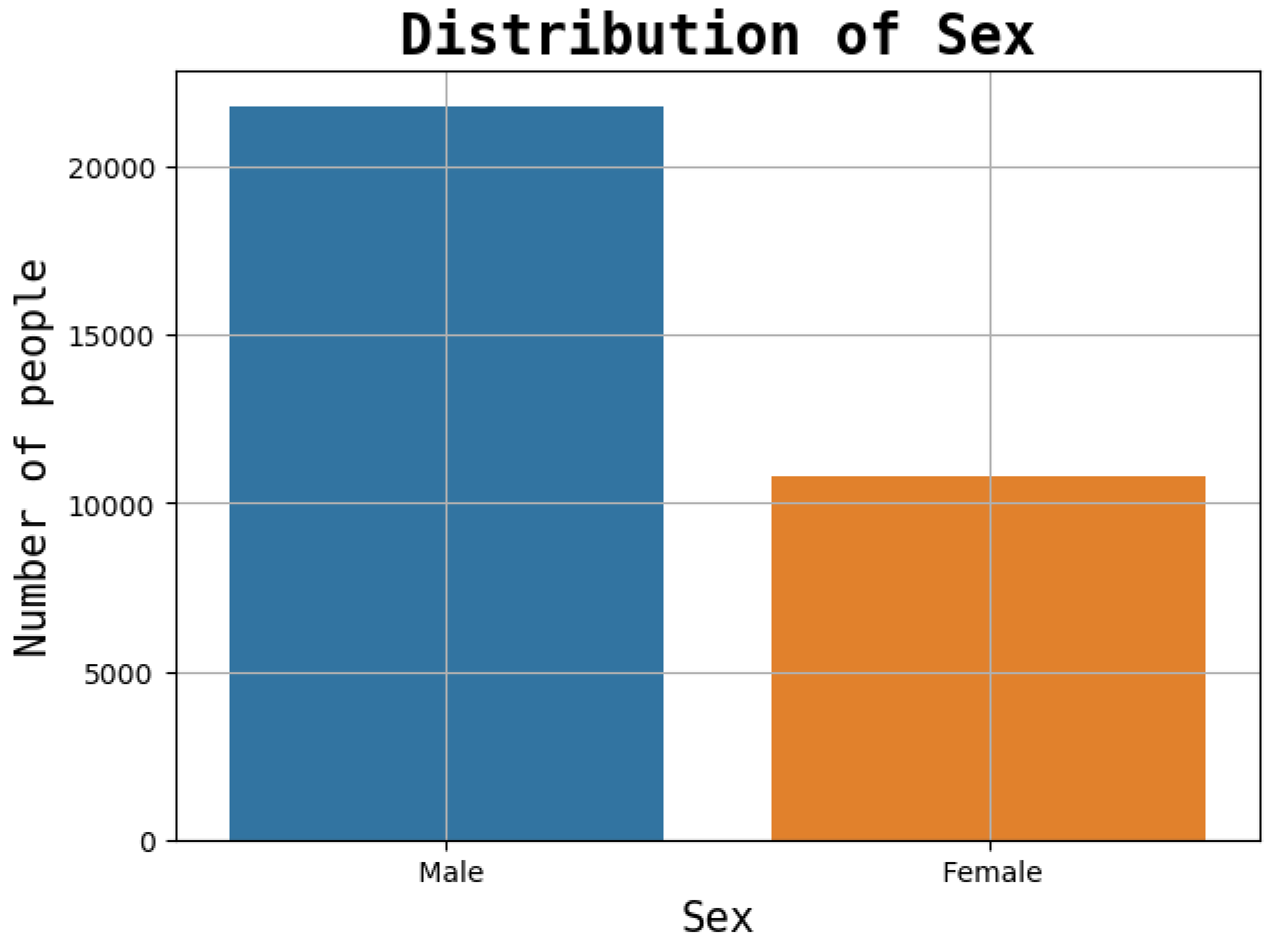
```
# Creating a donut chart for 'Age'
relation = df['Relationship'].value_counts()

plt.style.use('bmh')
plt.figure(figsize=(20, 10))
plt.pie(relation.values, labels=relation.index,
        startangle=50, autopct='%1.1f%%')
centre_circle = plt.Circle((0, 0), 0.7, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Relationship distribution', fontdict={
          'fontname': 'Monospace', 'fontsize': 30, 'fontweight': 'bold'})
plt.axis('equal')
plt.legend(prop={'size': 15})
plt.show()
```



From graph , the relation distribution where Husband is 40.5%, Not-in-family is 25.5% and as follows as shown in the graph

In [15]:

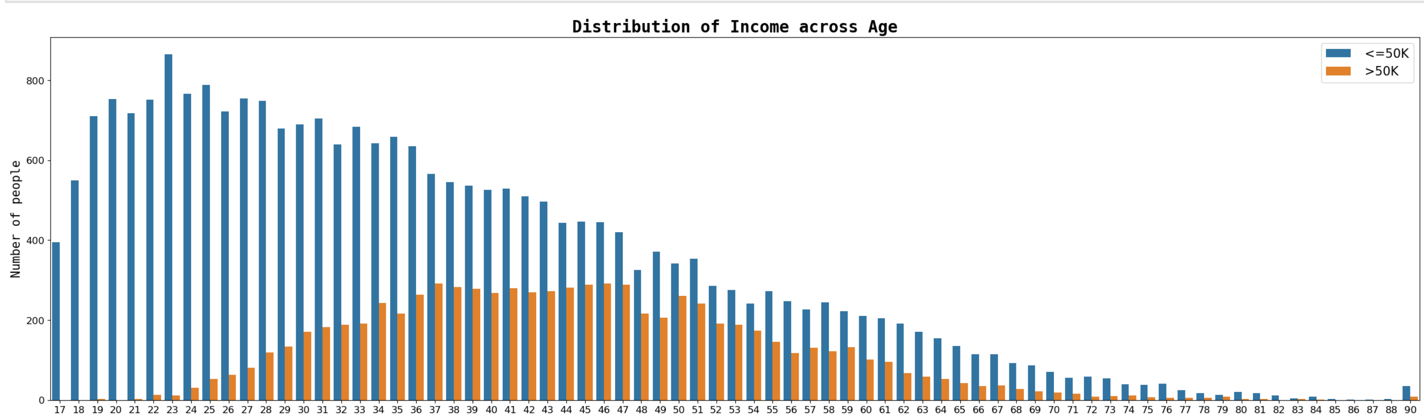
```
# Creating a barplot for 'Sex'
sex = df['Sex'].value_counts()

plt.style.use('default')
plt.figure(figsize=(7, 5))
sns.barplot(sex.index, sex.values)
plt.title('Distribution of Sex', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Sex', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=10)
plt.grid()
plt.show()
```



From graoh we can say , The male percentage is more when compare to female

In [16]:

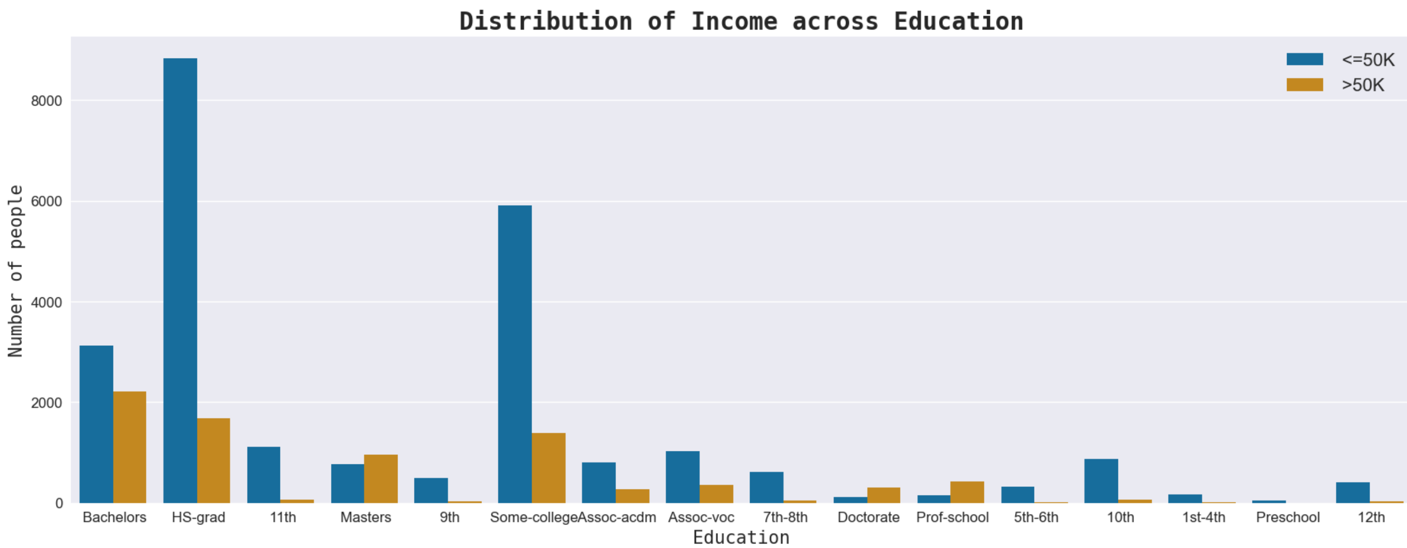
```
# Creating a countplot of income across age
plt.style.use('default')
plt.figure(figsize=(30, 8))
sns.countplot(df['Age'], hue=df['Income'])
plt.title('Distribution of Income across Age', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Age', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.legend(loc=1, prop={'size': 15})
plt.show()
```



Comparing the age with the income colume , from graph we can say that the maximum people are getting the income less then 50k at the age 24 on an average between age 36 to 47 the people are getting the income above 50k

In [17]:

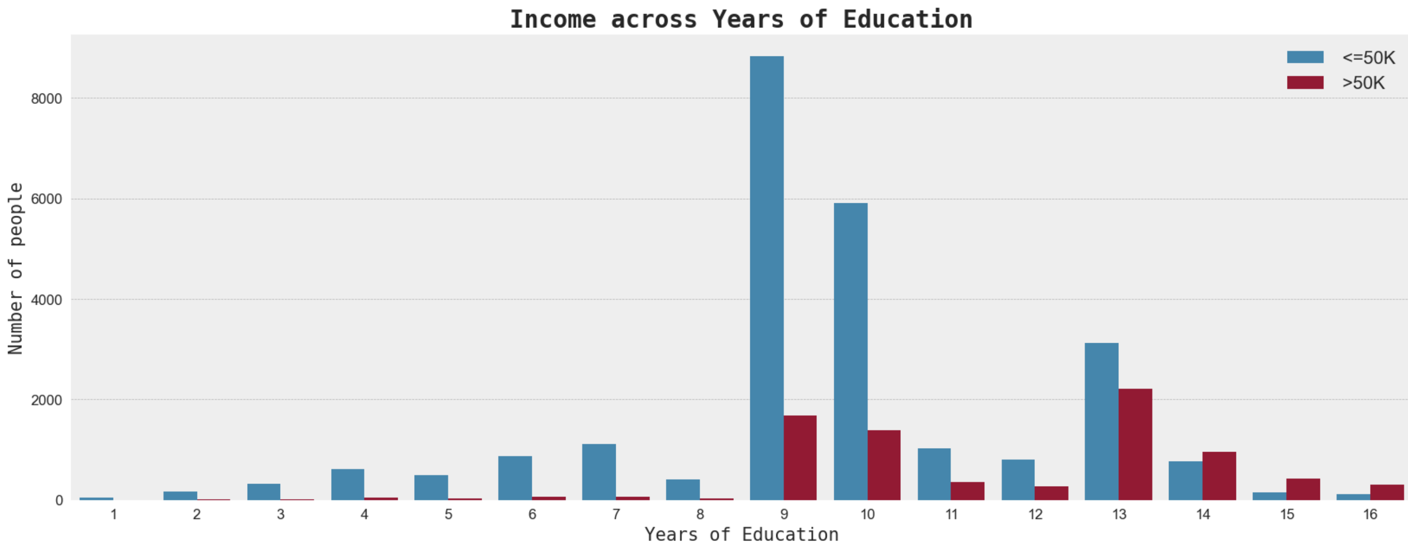
```
# Creating a countplot of income across education
plt.style.use('seaborn')
plt.figure(figsize=(20, 7))
sns.countplot(df['Education'],
              hue=df['Income'], palette='colorblind')
plt.title('Distribution of Income across Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Education', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.legend(loc=1, prop={'size': 15})
plt.show()
```



tha variation of the income column with the Education column is as shown in the above graph

In [18]:

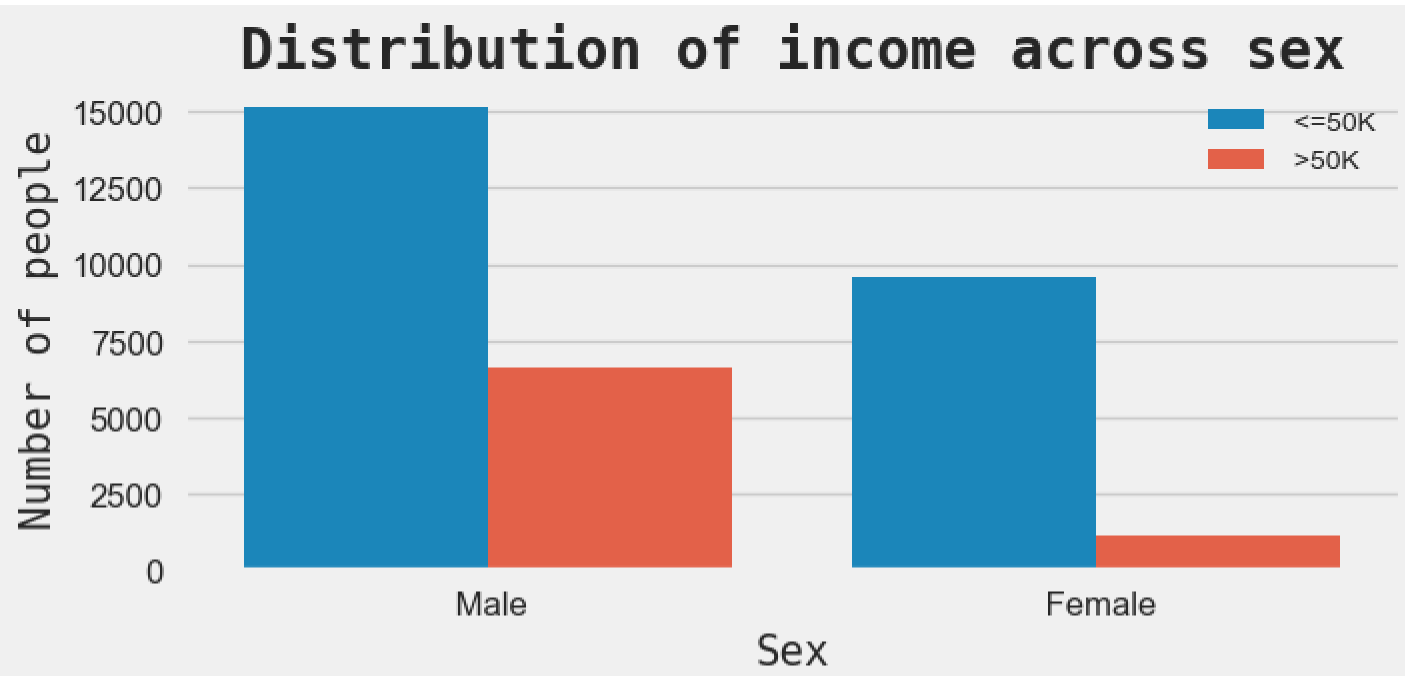
```
# Creating a countplot of income across years of education
plt.style.use('bmh')
plt.figure(figsize=(20, 7))
sns.countplot(df['Education_num'],
              hue=df['Income'])
plt.title('Income across Years of Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Years of Education', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.legend(loc=1, prop={'size': 15})
plt.savefig('bi2.png')
plt.show()
```



tha variation of the income column with the Education_num column is as shown in the above graph

In [19]:

```
# Creating a countplot of income across sex
plt.style.use('fivethirtyeight')
plt.figure(figsize=(7, 3))
sns.countplot(df['Sex'], hue=df['Income'])
plt.title('Distribution of income across sex', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Sex', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.legend(loc=1, prop={'size': 10})
plt.savefig('bi3.png')
plt.show()
```



In [20]:

```
# converting the categorical columns to numerical columns by labelEncoder
from sklearn.preprocessing import LabelEncoder
for col in df.columns:
    if df[col].dtype=='object':
        encode=LabelEncoder()
```

```
        df[col]=encode.fit_transform(df[col])
df
```

Out[20]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	Race	Sex
0	50	6	83311	9	13	2	4	0	4	1
1	38	4	215646	11	9	0	6	1	4	1
2	53	4	234721	1	7	2	6	0	2	1
3	28	4	338409	9	13	2	10	5	2	0
4	37	4	284582	12	14	2	4	5	4	0
...	...	...	...	...	...	...	...	...	...	...
32555	27	4	257302	7	12	2	13	5	4	0
32556	40	4	154374	11	9	2	7	0	4	1
32557	58	4	151910	11	9	6	1	4	4	0
32558	22	4	201490	11	9	4	1	3	4	1
32559	52	5	287927	11	9	2	4	5	4	0

32560 rows × 15 columns

In [21]:

```python
# Corelation can also be reprasented by heatmap
import matplotlib.pyplot as plt
plt.figure(figsize=(15,6))
sns.heatmap(df.corr(),annot=True,linewidth=0.1,linecolor='black',fmt='0.2f')
```

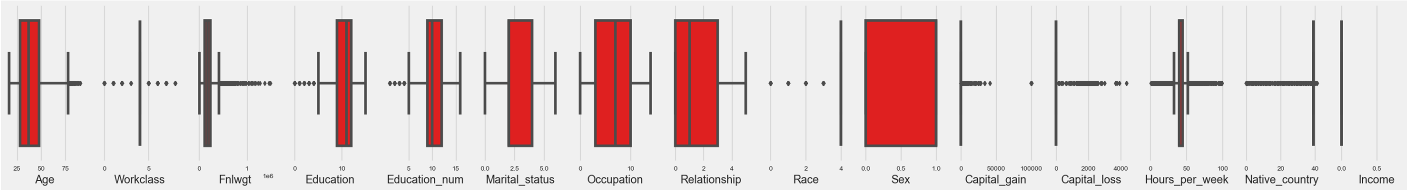
Out[21]: <AxesSubplot:>

From the correlation heatmap, we can say that the dependent feature 'Income' is highly correlated with Age, Education_num, Sex,Capital gain and number of Hours per week

In [22]:

```python
#For checking the outliers present in the dataset or not we use boxplot
columns_list=df.columns.values
ncol=100
nrows=50
plt.figure(figsize=(2*ncol,2*ncol))
```

```
for i in range (0,len(columns_list)):
    plt.subplot(nrows,ncol,i+1)
    sns.boxplot(df[columns_list[i]],color='red',orient='h')
    plt.tight_layout()
```



From the boxplot we can conclude that we have outliers present in the dataset

In [23]:

```
# checking how much number of outliers are present in the dataframe
from scipy.stats import zscore
z=np.abs(zscore(df))
print(df.shape)
print(z.shape)
threshold=3
print(np.where(z>3))
len(np.where(z>3)[0])
```

```
(32560, 15)
(32560, 15)
(array([    3,     9,    10, ..., 32532, 32550, 32552], dtype=int64), array([13, 12,  8,
...,  8,  8,  8], dtype=int64))
```

Out[23]:

```
5667
```

5667 outliers are present in the data set

In [24]:

```
# creating the new dataframe name df_new , which does't have any outliers
df_new=df[(z<3).all(axis=1)]
print(df_new.shape)
```

```
(27417, 15)
```

In [25]:

```
df_new.skew()
```

Out[25]:

```
Age                0.483478
Workclass         -0.738023
Fnlwgt             0.626221
Education         -0.957458
Education_num     -0.143960
Marital_status    -0.044317
Occupation         0.131148
Relationship       0.750207
Race              -2.592931
Sex               -0.684115
Capital_gain       4.934878
Capital_loss      29.325736
Hours_per_week    -0.358396
Native_country    -5.460675
Income             1.324919
dtype: float64
```

In [26]:

```
#For training the model we need to split the data frame values into Xtrain,Xtest,Ytrain,Yt
x= df.drop('Income', axis=1)
y= df['Income']
print(x)
print(y)
```

```
       Age  Workclass  Fnlwgt  Education  Education_num  Marital_status  \
0       50          6   83311          9             13               2   
1       38          4  215646         11              9               0   
2       53          4  234721          1              7               2   
3       28          4  338409          9             13               2   
4       37          4  284582         12             14               2   
...    ...        ...     ...        ...            ...             ...   
32555   27          4  257302          7             12               2   
32556   40          4  154374         11              9               2   
32557   58          4  151910         11              9               6   
32558   22          4  201490         11              9               4   
32559   52          5  287927         11              9               2   

       Occupation  Relationship  Race  Sex  Capital_gain  Capital_loss  \
0               4             0     4    1             0             0   
1               6             1     4    1             0             0   
2               6             0     2    1             0             0   
3              10             5     2    0             0             0   
4               4             5     4    0             0             0   
...           ...           ...   ...  ...           ...           ...   
32555          13             5     4    0             0             0   
32556           7             0     4    1             0             0   
32557           1             4     4    0             0             0   
32558           1             3     4    1             0             0   
32559           4             5     4    0         15024             0   

       Hours_per_week  Native_country  
0                  13              39  
1                  40              39  
2                  40              39  
3                  40               5  
4                  40              39  
...               ...             ...  
32555              38              39  
32556              40              39  
32557              40              39  
32558              20              39  
32559              40              39  

[32560 rows x 14 columns]
0        0
1        0
2        0
3        0
4        0
        ..
32555    0
32556    1
32557    0
32558    0
32559    1
Name: Income, Length: 32560, dtype: int32
```

In [27]:

```
pd.DataFrame(x).skew()
```

Out[27]:

```
Age               0.558738
Workclass        -0.752280
Fnlwgt            1.446972
Education        -0.934063
Education_num    -0.311630
Marital_status   -0.013448
Occupation        0.114540
Relationship      0.786784
Race             -2.435332
```

```
Sex               -0.719244
Capital_gain      11.953690
Capital_loss       4.594549
Hours_per_week     0.227636
Native_country    -3.658235
dtype: float64
```

In [28]:

```
# the columns like workclass,education,race,sex,capital_loss,native_country doesnot have n
x=x.drop(['Workclass', 'Education', 'Race', 'Sex',
            'Capital_loss', 'Native_country'], axis=1)
```

In [29]:

```
from sklearn.preprocessing import StandardScaler
for col in x.columns:
    scaler = StandardScaler()
    x[col] = scaler.fit_transform(x[col].values.reshape(-1, 1))
```

In [31]:

```
# importing all the algorithems for checking the R2_score and model perforfance
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
```

In [32]:

```
model=[RandomForestClassifier(),DecisionTreeClassifier(),KNeighborsClassifier(),GaussianNB
max_r2_score=0
for i_state in range(5,10):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i_state,test_size=0.33
    for a in model:
        a.fit(x_train,y_train)
        pred=a.predict(x_test)
        score=accuracy_score(y_test,pred)
        print('score for random_state',i_state,'is',score)
        if score>max_r2_score:
            max_r2_score=score
            Final_state=i_state
            Final_model= a
print('accuracy_score ',max_r2_score,'for random state ',Final_state, 'and model is ',Fina
```

```
score for random_state 5 is 0.8444858073522569
score for random_state 5 is 0.7982317356910191
score for random_state 5 is 0.8315495579339227
score for random_state 5 is 0.801768264308981
score for random_state 5 is 0.8233597021870638
score for random_state 5 is 0.8451372731503025
score for random_state 6 is 0.845788738948348
score for random_state 6 is 0.8013959981386691
score for random_state 6 is 0.8255932992089344
score for random_state 6 is 0.7952536063285249
score for random_state 6 is 0.8189855746859004
score for random_state 6 is 0.8434620753838995
score for random_state 7 is 0.8435551419264774
score for random_state 7 is 0.7963704048394602
score for random_state 7 is 0.8333178222429036
score for random_state 7 is 0.7903210795718939
score for random_state 7 is 0.8172173103769195
score for random_state 7 is 0.8472778036295951
score for random_state 8 is 0.8448580735225686
```

```
score for random_state 8 is 0.8010237319683574
score for random_state 8 is 0.8294090274546301
score for random_state 8 is 0.7907864122847836
score for random_state 8 is 0.8205677059097255
score for random_state 8 is 0.8451372731503025
score for random_state 9 is 0.8428106095858539
score for random_state 9 is 0.8024197301070265
score for random_state 9 is 0.8292228943694742
score for random_state 9 is 0.7977664029781294
score for random_state 9 is 0.8174965100046533
score for random_state 9 is 0.8461610051186599
accuracy_score  0.8472778036295951 for random state  7 and model is  SVC()
```

In [33]:

```python
# we are training the model with KNeighborsRegressor for randomstate 18 and checking the r
svc=SVC()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=7,test_size=0.33)
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
pred_y=svc.predict(x_test)
svcs=accuracy_score(y_test,pred_y)
print('accuracy_score =',svcs*100)
print('classification _report ',classification_report(y_test,pred_y))
print(confusion_matrix(y_test,pred_y))
```

```
accuracy_score = 84.72778036295952
classification _report                precision    recall  f1-score   support

           0       0.86      0.95      0.90      8154
           1       0.77      0.52      0.62      2591

    accuracy                           0.85     10745
   macro avg       0.82      0.74      0.76     10745
weighted avg       0.84      0.85      0.84     10745

[[7744  410]
 [1231 1360]]
```

In [34]:

```python
from sklearn.model_selection import GridSearchCV
# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(x_train,y_train)
print(grid.best_params_)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.810 total time=  17.4s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.818 total time=  21.3s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.813 total time=  17.4s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.811 total time=  17.4s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.812 total time=  17.4s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.833 total time=  11.6s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.850 total time=  11.6s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.839 total time=  11.4s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.840 total time=  11.6s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.842 total time=  11.5s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.799 total time=  13.1s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.796 total time=  13.1s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.799 total time=  13.2s
```

```
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.802 total time=  13.1s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.796 total time=  13.3s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.765 total time=  13.9s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.764 total time=  13.7s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.765 total time=  13.7s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.765 total time=  13.6s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.764 total time=  13.7s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.759 total time=  14.1s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.759 total time=  13.8s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.759 total time=  13.9s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.759 total time=  13.8s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.759 total time=  13.9s
[CV 1/5] END .........C=1, gamma=1, kernel=rbf;, score=0.835 total time=  23.2s
[CV 2/5] END .........C=1, gamma=1, kernel=rbf;, score=0.845 total time=  23.0s
[CV 3/5] END .........C=1, gamma=1, kernel=rbf;, score=0.833 total time=  28.2s
[CV 4/5] END .........C=1, gamma=1, kernel=rbf;, score=0.840 total time=  27.1s
[CV 5/5] END .........C=1, gamma=1, kernel=rbf;, score=0.841 total time=  23.5s
[CV 1/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.838 total time=  11.2s
[CV 2/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.857 total time=  11.5s
[CV 3/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.842 total time=  11.3s
[CV 4/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.849 total time=  11.3s
[CV 5/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.842 total time=  11.3s
[CV 1/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.828 total time=  11.3s
[CV 2/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.843 total time=  11.8s
[CV 3/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.831 total time=  11.6s
[CV 4/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.839 total time=  11.4s
[CV 5/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.836 total time=  11.6s
[CV 1/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.793 total time=  13.5s
[CV 2/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.793 total time=  13.3s
[CV 3/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.796 total time=  13.6s
[CV 4/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.797 total time=  13.4s
[CV 5/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.793 total time=  13.3s
[CV 1/5] END ....C=1, gamma=0.0001, kernel=rbf;, score=0.765 total time=  13.7s
[CV 2/5] END ....C=1, gamma=0.0001, kernel=rbf;, score=0.764 total time=  13.8s
[CV 3/5] END ....C=1, gamma=0.0001, kernel=rbf;, score=0.765 total time=  13.7s
[CV 4/5] END ....C=1, gamma=0.0001, kernel=rbf;, score=0.765 total time=  13.8s
[CV 5/5] END ....C=1, gamma=0.0001, kernel=rbf;, score=0.764 total time=  13.8s
[CV 1/5] END ........C=10, gamma=1, kernel=rbf;, score=0.820 total time=  36.7s
[CV 2/5] END ........C=10, gamma=1, kernel=rbf;, score=0.824 total time=  37.7s
[CV 3/5] END ........C=10, gamma=1, kernel=rbf;, score=0.820 total time=  38.1s
[CV 4/5] END ........C=10, gamma=1, kernel=rbf;, score=0.824 total time=  36.9s
[CV 5/5] END ........C=10, gamma=1, kernel=rbf;, score=0.821 total time=  37.7s
[CV 1/5] END ......C=10, gamma=0.1, kernel=rbf;, score=0.837 total time=  13.3s
[CV 2/5] END ......C=10, gamma=0.1, kernel=rbf;, score=0.856 total time=  13.6s
[CV 3/5] END ......C=10, gamma=0.1, kernel=rbf;, score=0.843 total time=  13.9s
[CV 4/5] END ......C=10, gamma=0.1, kernel=rbf;, score=0.847 total time=  13.7s
[CV 5/5] END ......C=10, gamma=0.1, kernel=rbf;, score=0.847 total time=  13.5s
[CV 1/5] END .....C=10, gamma=0.01, kernel=rbf;, score=0.834 total time=  11.3s
[CV 2/5] END .....C=10, gamma=0.01, kernel=rbf;, score=0.857 total time=  11.5s
[CV 3/5] END .....C=10, gamma=0.01, kernel=rbf;, score=0.838 total time=  11.5s
[CV 4/5] END .....C=10, gamma=0.01, kernel=rbf;, score=0.843 total time=  11.4s
[CV 5/5] END .....C=10, gamma=0.01, kernel=rbf;, score=0.841 total time=  11.3s
[CV 1/5] END ....C=10, gamma=0.001, kernel=rbf;, score=0.808 total time=  12.6s
[CV 2/5] END ....C=10, gamma=0.001, kernel=rbf;, score=0.806 total time=  14.0s
[CV 3/5] END ....C=10, gamma=0.001, kernel=rbf;, score=0.805 total time=  13.0s
[CV 4/5] END ....C=10, gamma=0.001, kernel=rbf;, score=0.812 total time=  12.9s
[CV 5/5] END ....C=10, gamma=0.001, kernel=rbf;, score=0.803 total time=  12.9s
[CV 1/5] END ...C=10, gamma=0.0001, kernel=rbf;, score=0.794 total time=  13.6s
[CV 2/5] END ...C=10, gamma=0.0001, kernel=rbf;, score=0.793 total time=  13.5s
[CV 3/5] END ...C=10, gamma=0.0001, kernel=rbf;, score=0.796 total time=  13.7s
[CV 4/5] END ...C=10, gamma=0.0001, kernel=rbf;, score=0.797 total time=  13.8s
[CV 5/5] END ...C=10, gamma=0.0001, kernel=rbf;, score=0.793 total time=  13.5s
[CV 1/5] END .......C=100, gamma=1, kernel=rbf;, score=0.798 total time=  58.4s
[CV 2/5] END .......C=100, gamma=1, kernel=rbf;, score=0.796 total time=  56.3s
```

```
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.796 total time=  59.2s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.812 total time= 1.0min
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.796 total time=  58.9s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.839 total time=  28.0s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.853 total time=  28.0s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.841 total time=  28.8s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.843 total time=  27.5s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.842 total time=  28.7s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.836 total time=  13.8s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.858 total time=  13.9s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.842 total time=  13.5s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.846 total time=  13.7s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.845 total time=  14.1s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.827 total time=  12.5s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.843 total time=  12.2s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.831 total time=  12.2s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.838 total time=  12.3s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.834 total time=  12.2s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.799 total time=  13.2s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.804 total time=  14.4s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.799 total time=  14.7s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.804 total time=  13.1s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.797 total time=  13.2s
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

In [35]:

```python
smv1=SVC(C=10,gamma=0.1,kernel='rbf')
smv1.fit(x_train,y_train)
pred_smv=smv1.predict(x_test)
score=accuracy_score(y_test,pred_smv)
print('accuracy_score= ',score*100)
cv_score=cross_val_score(smv1,x,y,cv=5)
cv_mean=cv_score.mean()
print('accuracy_score= ',score*100)
print('mean_cv value = ',cv_mean*100)
print(confusion_matrix(y_test,pred_y))
print(classification_report(y_test,pred_y))
```

```
accuracy_score=  84.69055374592834
accuracy_score=  84.69055374592834
mean_cv value =  84.60687960687962
[[7744  410]
 [1231 1360]]
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      8154
           1       0.77      0.52      0.62      2591

    accuracy                           0.85     10745
   macro avg       0.82      0.74      0.76     10745
weighted avg       0.84      0.85      0.84     10745
```

Conclusion:

In this project, we build various models like logistic regression, knn classifier, support vector classifier, decision tree classifier, random forest classifier.

A hyperparameter tuned support vector classifier gives the highest accuracy score of 84.69

In [ ]: