In [1]:

```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\shobhandeb\appdata\local\programs\py
thon\python310\lib\site-packages (6.5.0)
Requirement already satisfied: beautifulsoup4 in c:\users\shobhandeb\appdata\local\progra
ms\python\python310\lib\site-packages (from nbconvert) (4.9.1)
Requirement already satisfied: traitlets>=5.0 in c:\users\shobhandeb\appdata\local\progra
ms\python\python310\lib\site-packages (from nbconvert) (5.3.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\shobhandeb\appdata\local\p
rograms\python\python310\lib\site-packages (from nbconvert) (0.2.2)
Requirement already satisfied: jinja2>=3.0 in c:\users\shobhandeb\appdata\local\programs\
python\python310\lib\site-packages (from nbconvert) (3.0.3)
Requirement already satisfied: tinycss2 in c:\users\shobhandeb\appdata\local\programs\pyt
hon\python310\lib\site-packages (from nbconvert) (1.1.1)
Requirement already satisfied: packaging in c:\users\shobhandeb\appdata\local\programs\py
thon\python310\lib\site-packages (from nbconvert) (21.3)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\shobhandeb\appdata\local\pr
ograms\python\python310\lib\site-packages (from nbconvert) (0.4)
Requirement already satisfied: pygments>=2.4.1 in c:\users\shobhandeb\appdata\roaming\pyt
hon\python310\site-packages (from nbconvert) (2.12.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\shobhandeb\appdata\local\progr
ams\python\python310\lib\site-packages (from nbconvert) (2.1.1)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\shobhandeb\appdata\local\progr
ams\python\python310\lib\site-packages (from nbconvert) (0.6.4)
Requirement already satisfied: nbformat>=5.1 in c:\users\shobhandeb\appdata\local\program
s\python\python310\lib\site-packages (from nbconvert) (5.4.0)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\shobhandeb\appdata\local\pro
grams\python\python310\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\shobhandeb\appdata\local\pro
grams\python\python310\lib\site-packages (from nbconvert) (4.10.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\shobhandeb\appdata\local\
programs\python\python310\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: bleach in c:\users\shobhandeb\appdata\local\programs\pytho
n\python310\lib\site-packages (from nbconvert) (5.0.0)
Requirement already satisfied: defusedxml in c:\users\shobhandeb\appdata\local\programs\p
ython\python310\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\shobhandeb\appdata\local\programs
\python\python310\lib\site-packages (from jupyter-core>=4.7->nbconvert) (304)
Requirement already satisfied: nest-asyncio in c:\users\shobhandeb\appdata\local\programs
\python\python310\lib\site-packages (from nbclient>=0.5.0->nbconvert) (1.5.5)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\shobhandeb\appdata\local
\programs\python\python310\lib\site-packages (from nbclient>=0.5.0->nbconvert) (7.3.4)
Requirement already satisfied: jsonschema>=2.6 in c:\users\shobhandeb\appdata\local\progr
ams\python\python310\lib\site-packages (from nbformat>=5.1->nbconvert) (4.6.0)
Requirement already satisfied: fastjsonschema in c:\users\shobhandeb\appdata\local\progra
ms\python\python310\lib\site-packages (from nbformat>=5.1->nbconvert) (2.15.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\shobhandeb\appdata\local\program
s\python\python310\lib\site-packages (from beautifulsoup4->nbconvert) (2.0.1)
Requirement already satisfied: webencodings in c:\users\shobhandeb\appdata\local\programs
\python\python310\lib\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: six>=1.9.0 in c:\users\shobhandeb\appdata\local\programs\p
ython\python310\lib\site-packages (from bleach->nbconvert) (1.15.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shobhandeb\appdata\lo
cal\programs\python\python310\lib\site-packages (from packaging->nbconvert) (3.0.8)
Requirement already satisfied: attrs>=17.4.0 in c:\users\shobhandeb\appdata\local\program
s\python\python310\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (21
.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users\
shobhandeb\appdata\local\programs\python\python310\lib\site-packages (from jsonschema>=2.
6->nbformat>=5.1->nbconvert) (0.18.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\shobhandeb\appdata\loca
l\programs\python\python310\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.
0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=6.0 in c:\users\shobhandeb\appdata\local\programs
\python\python310\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconve
rt) (6.1)
Requirement already satisfied: pyzmq>=23.0 in c:\users\shobhandeb\appdata\local\programs\
```

```
python\python310\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconver
t) (23.2.0)
```

```
WARNING: There was an error checking the latest version of pip.
```

Description:

In this notebook, we are going to predict whether a person's income is above 50k or below 50k using various features like age, education, and occupation.

The dataset we are going to use is the Adult census income dataset from UCI which contains about 32561 rows and 15 features that can be downloaded here: https://archive.ics.uci.edu/ml/datasets/Census+Income

The dataset contains the labels which we have to predict and the labels are discrete and binary. So the problem we have is a Supervised Classification type.

Data Set Information:

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Prediction task is to determine whether a person makes over 50K a year.

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.

In [2]:

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

In [3]:

```
data = pd.read_csv('adult.csv',sep=',')
```

In [4]:

```
data.head()
```

Out[4]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	ca
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	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	

In [5]:

```
data.tail()
```

Out[5]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gai
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male	
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	

In [6]:

```
data.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
```

In [7]:

```python
data.describe(include='all')
```

Out[7]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	s
count	32561.000000	32561	3.256100e+04	32561	32561.000000	32561	32561	32561	32561	32!
unique	NaN	9	NaN	16	NaN	7	15	6	5	
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	White	M
freq	NaN	22696	NaN	10501	NaN	14976	4140	13193	27816	21
mean	38.581647	NaN	1.897784e+05	NaN	10.080679	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	
min	17.000000	NaN	1.228500e+04	NaN	1.000000	NaN	NaN	NaN	NaN	N
25%	28.000000	NaN	1.178270e+05	NaN	9.000000	NaN	NaN	NaN	NaN	N
50%	37.000000	NaN	1.783560e+05	NaN	10.000000	NaN	NaN	NaN	NaN	N
75%	48.000000	NaN	2.370510e+05	NaN	12.000000	NaN	NaN	NaN	NaN	N
max	90.000000	NaN	1.484705e+06	NaN	16.000000	NaN	NaN	NaN	NaN	N

In [8]:

```python
data.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
age	32561.0	38.581647	13.640433	17.0	28.0	37.0	48.0	90.0
fnlwgt	32561.0	189778.366512	105549.977697	12285.0	117827.0	178356.0	237051.0	1484705.0
education.num	32561.0	10.080679	2.572720	1.0	9.0	10.0	12.0	16.0
capital.gain	32561.0	1077.648844	7385.292085	0.0	0.0	0.0	0.0	99999.0
capital.loss	32561.0	87.303830	402.960219	0.0	0.0	0.0	0.0	4356.0
hours.per.week	32561.0	40.437456	12.347429	1.0	40.0	40.0	45.0	99.0

In [9]:

```python
data.isnull().sum()
```

Out[9]:

```
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
```

In [10]:

```
data.isna().sum()
```

Out[10]:

```
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
```

In [11]:

```
# Check for '?' in dataset
round((data.isin(['?']).sum() / data.shape[0])
      * 100, 2).astype(str) + ' %'
```

Out[11]:

```
age                0.0 %
workclass         5.64 %
fnlwgt             0.0 %
education          0.0 %
education.num      0.0 %
marital.status     0.0 %
occupation        5.66 %
relationship       0.0 %
race               0.0 %
sex                0.0 %
capital.gain       0.0 %
capital.loss       0.0 %
hours.per.week     0.0 %
native.country    1.79 %
income             0.0 %
dtype: object
```

In [12]:

```
# Checking the counts of label categories
income = data['income'].value_counts(normalize=True)
round(income * 100, 2).astype('str') + ' %'
```

Out[12]:

```
<=50K    75.92 %
>50K     24.08 %
Name: income, dtype: object
```

Observed:

The dataset doesn't have any null values, but it contains missing values in the form of '?' which needs to be preprocessed.

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

In [13]:

```
data.shape
```

```
data.shape
```

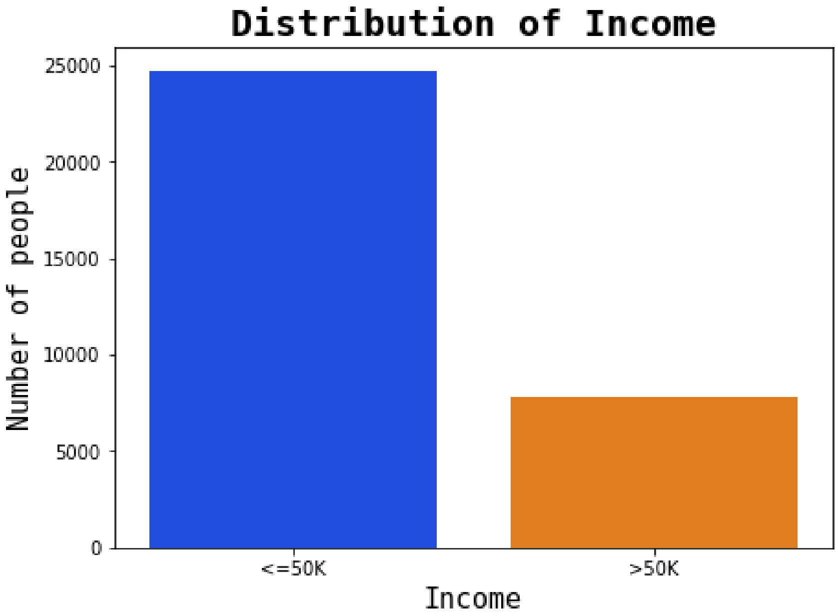
Out[13]:

```
(32561, 15)
```

In [14]:

```
# Creating a barplot for 'Income'
income = data['income'].value_counts()

# plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(7, 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()
```

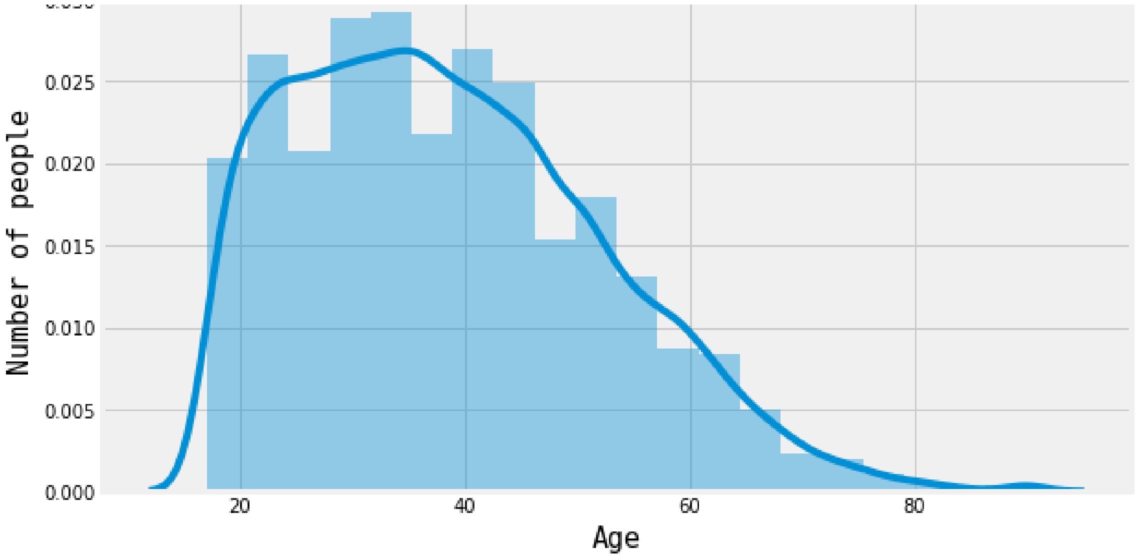
```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [15]:

```
# Creating a distribution plot for 'Age'
age = data['age'].value_counts()

plt.figure(figsize=(10, 5))
plt.style.use('fivethirtyeight')
sns.distplot(data['age'], bins=20)
plt.title('Distribution of 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=10)
plt.show()
```

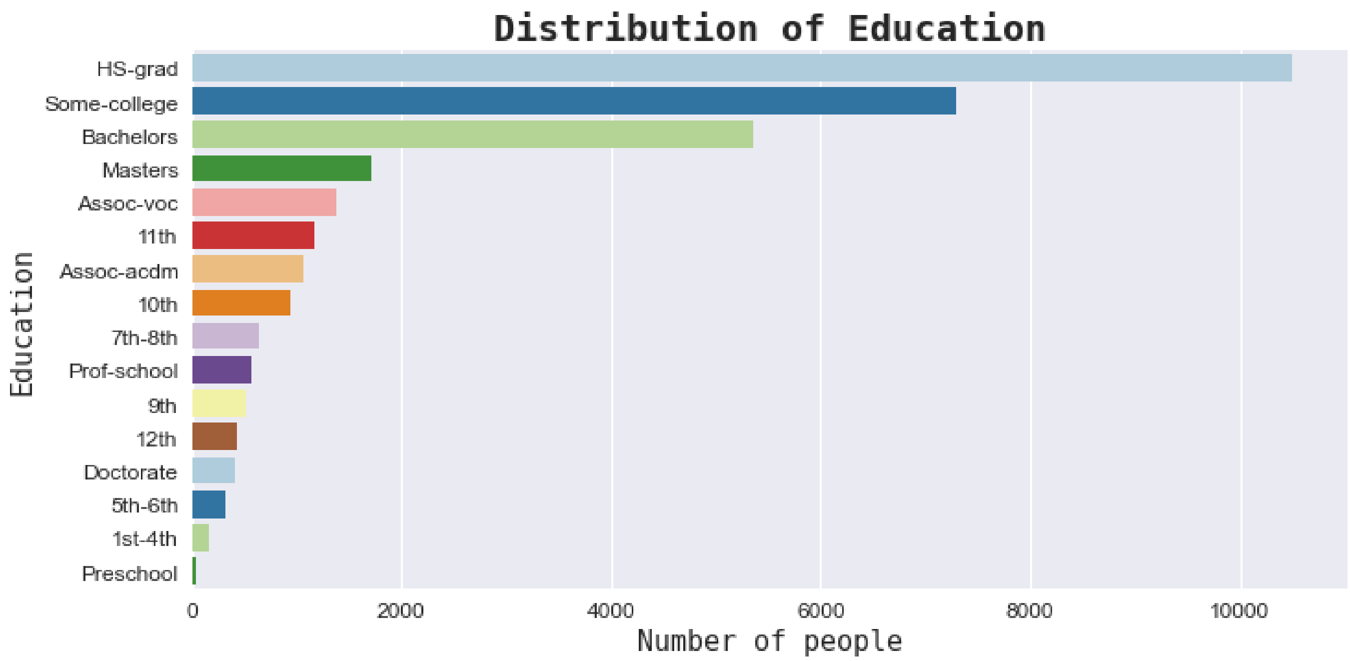
```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\dis
tributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [16]:

```
# Creating a barplot for 'Education'
education = data['education'].value_counts()

plt.style.use('seaborn')
plt.figure(figsize=(10, 5))
sns.barplot(education.values, education.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()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

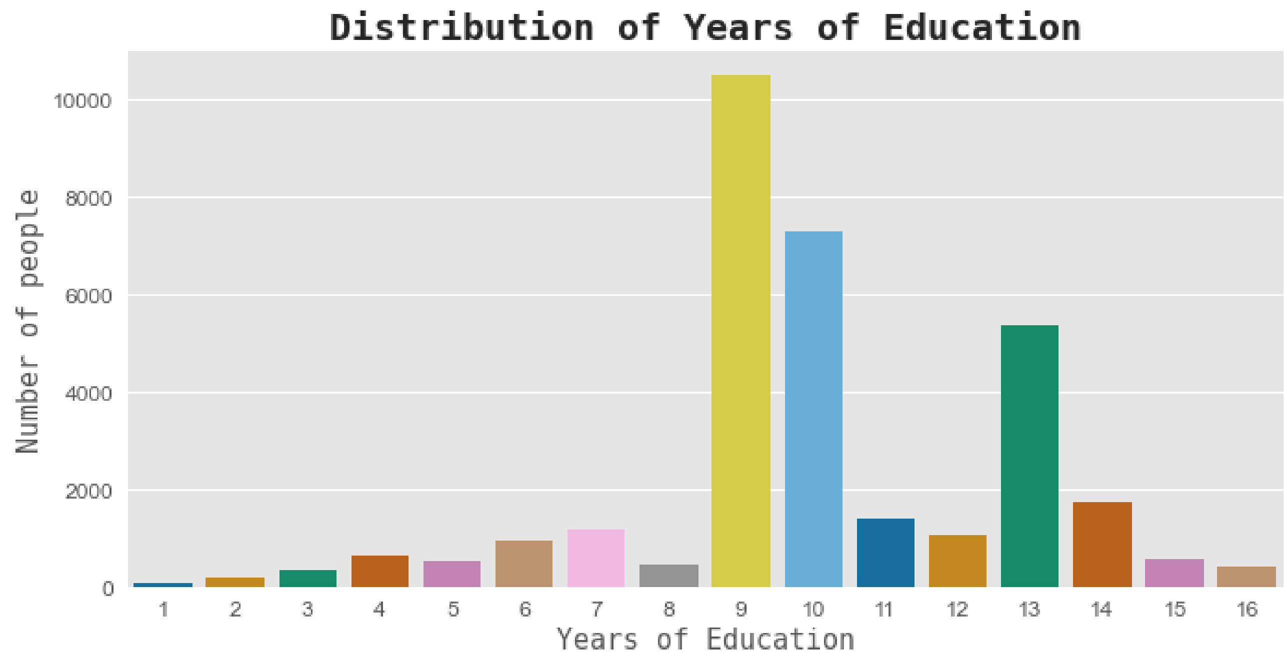


In [17]:

```
# Creating a barplot for 'Years of Education'
education_num = data['education.num'].value_counts()

plt.style.use('ggplot')
plt.figure(figsize=(10, 5))
sns.barplot(education_num.index, education_num.values, palette='colorblind')
```

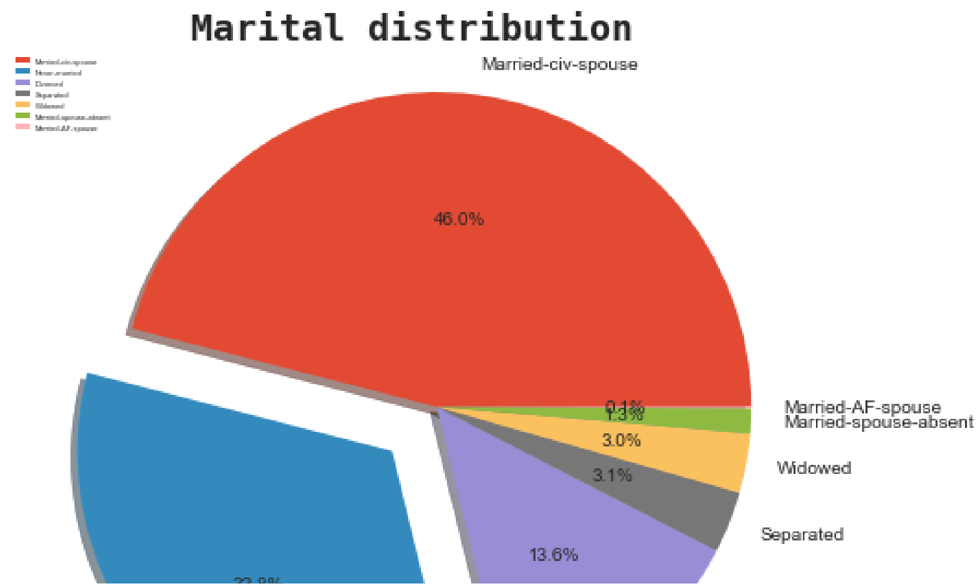
```python
plt.title('Distribution of Years of Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Years of Education', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.show()
```

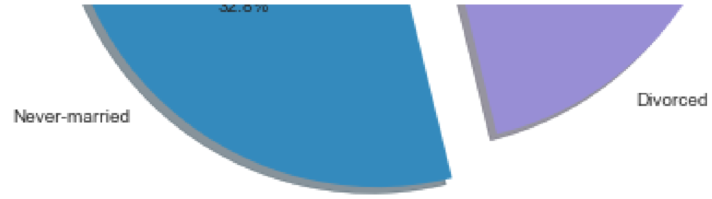
```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [18]:

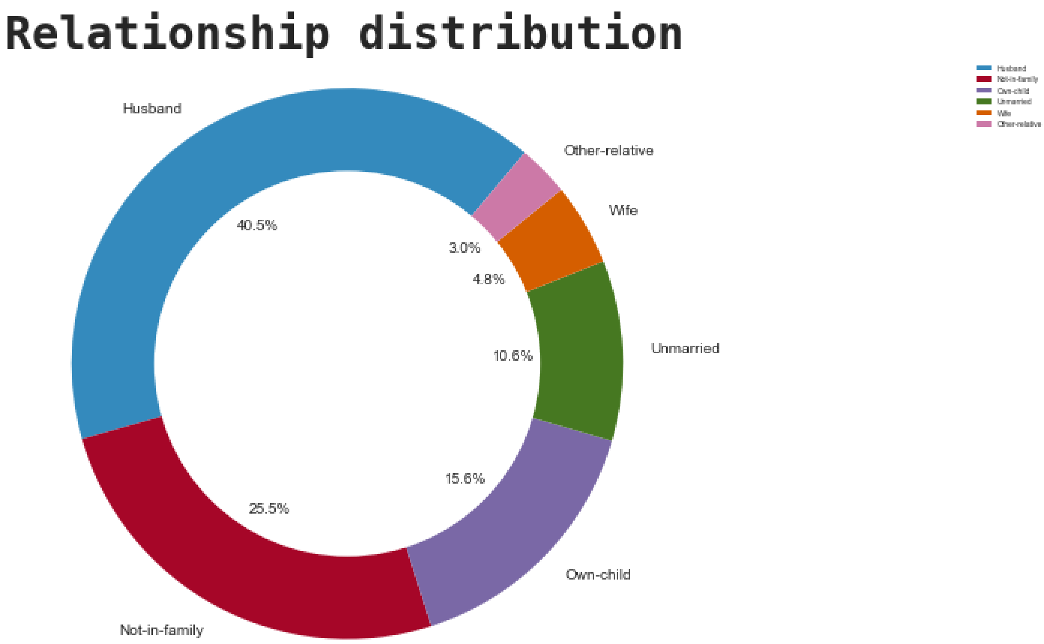
```python
# Creating a pie chart for 'Marital status'
marital = data['marital.status'].value_counts()

# plt.style.use('default')
plt.figure(figsize=(7, 7))
plt.pie(marital.values, labels=marital.index, 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': 4})
plt.axis('equal')
plt.show()
```





In [19]:

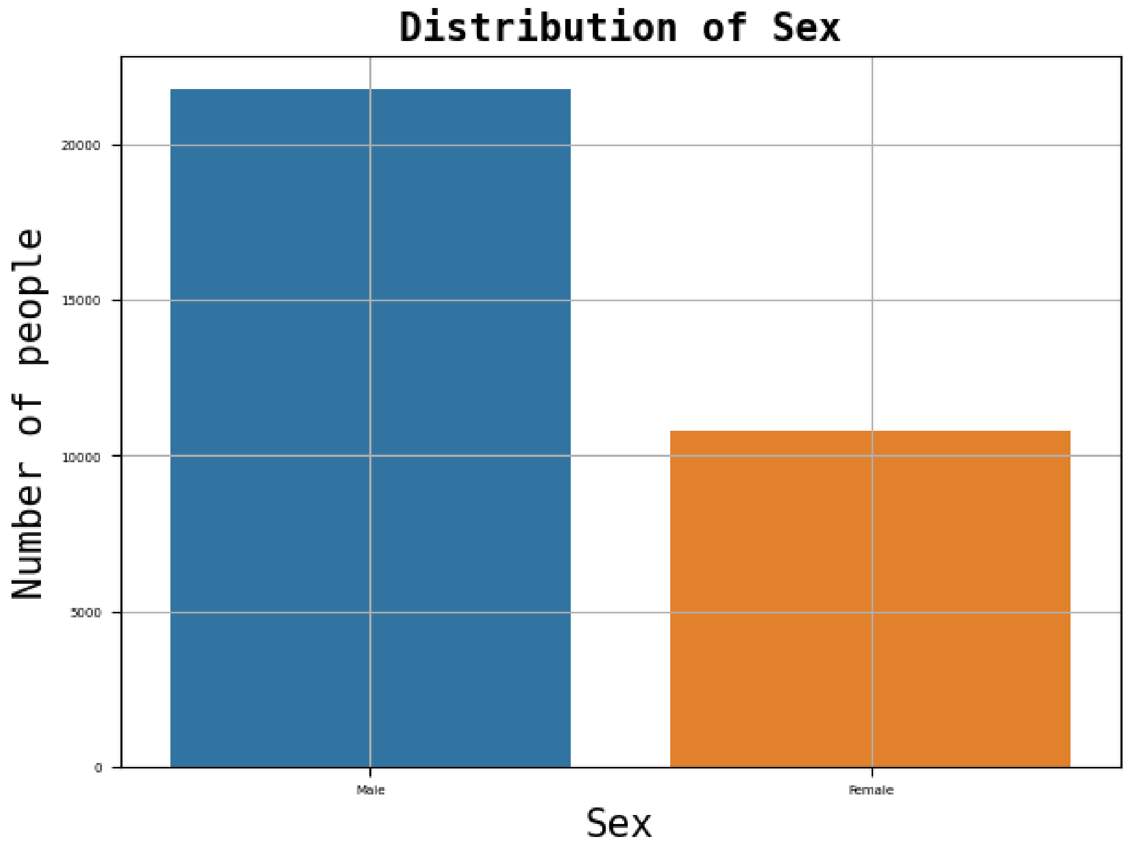
```
# Creating a donut chart for 'Age'
relation = data['relationship'].value_counts()

plt.style.use('bmh')
plt.figure(figsize=(15, 7))
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': 5})
plt.show()
```



In [20]:

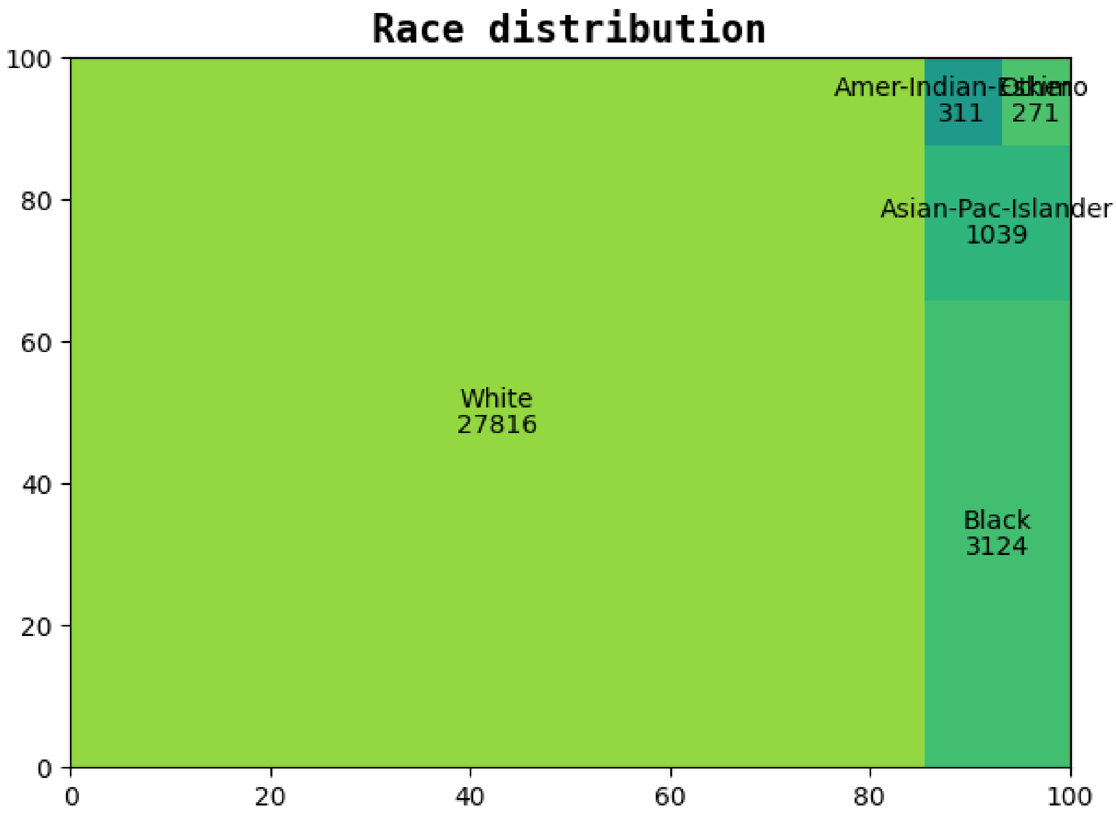
```
# Creating a barplot for 'Sex'
sex = data['sex'].value_counts()

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

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [21]:

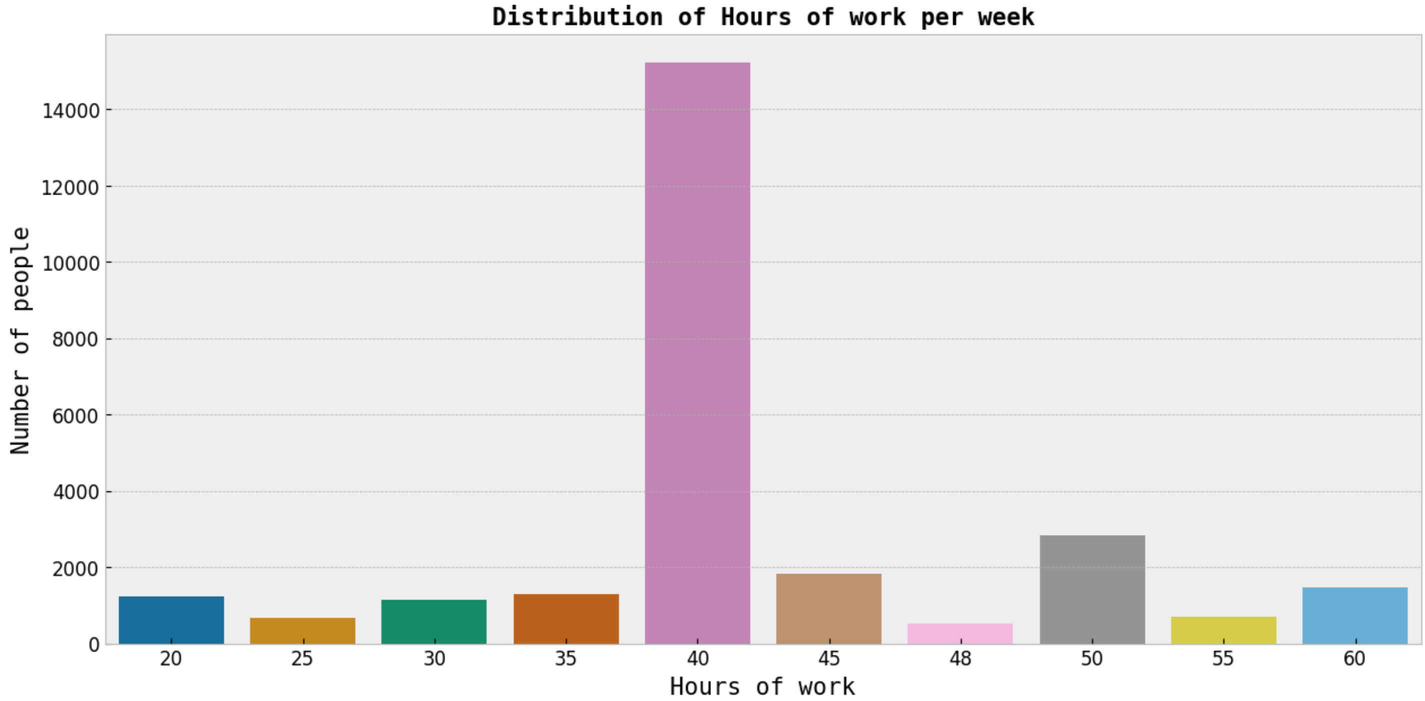
```python
# Creating a Treemap for 'Race'
import squarify
race = data['race'].value_counts()

plt.style.use('default')
plt.figure(figsize=(7, 5))
squarify.plot(sizes=race.values, label=race.index, value=race.values)
plt.title('Race distribution', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, 'fontweight': 'bold'})
plt.show()
```



In [22]:

```python
# Creating a barplot for 'Hours per week'
hours = data['hours.per.week'].value_counts().head(10)

plt.style.use('bmh')
plt.figure(figsize=(15, 7))
sns.barplot(hours.index, hours.values, palette='colorblind')
plt.title('Distribution of Hours of work per week', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, 'fontweight': 'bold'})
plt.xlabel('Hours of work', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Number of people', fontdict={
           'fontname': 'Monospace', 'fontsize': 15})
plt.tick_params(labelsize=12)
plt.show()
```

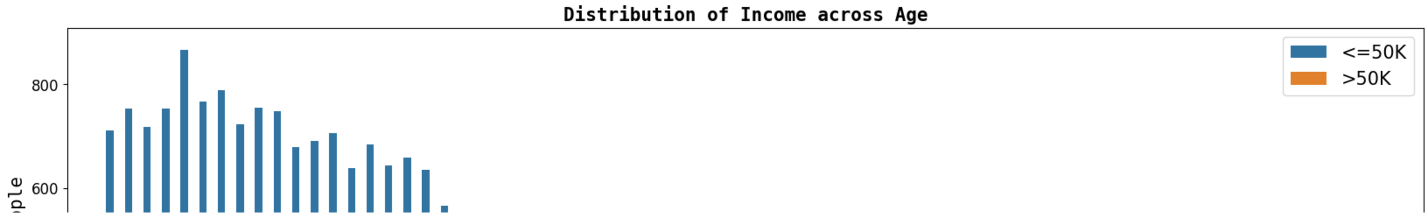
```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

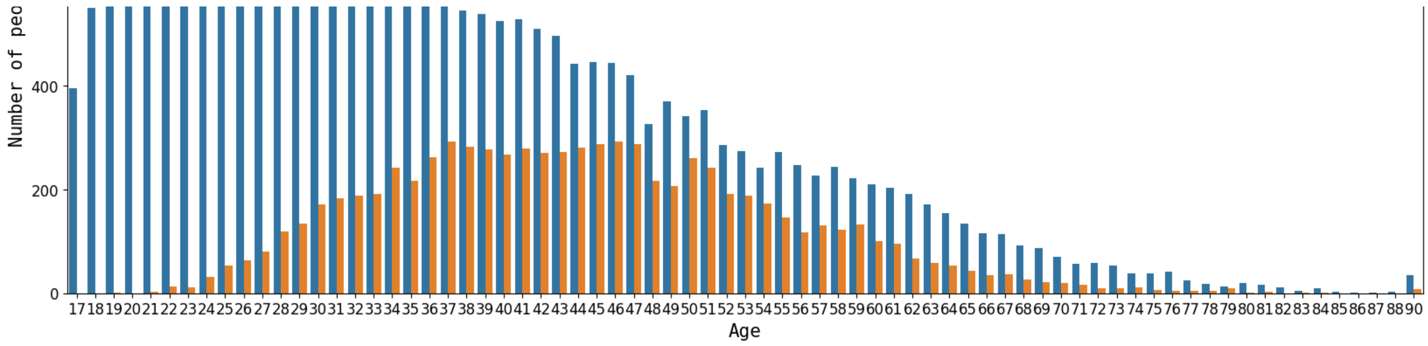


In [23]:

```python
# Creating a countplot of income across age
plt.style.use('default')
plt.figure(figsize=(20, 7))
sns.countplot(data['age'], hue=data['income'])
plt.title('Distribution of Income across Age', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, '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()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

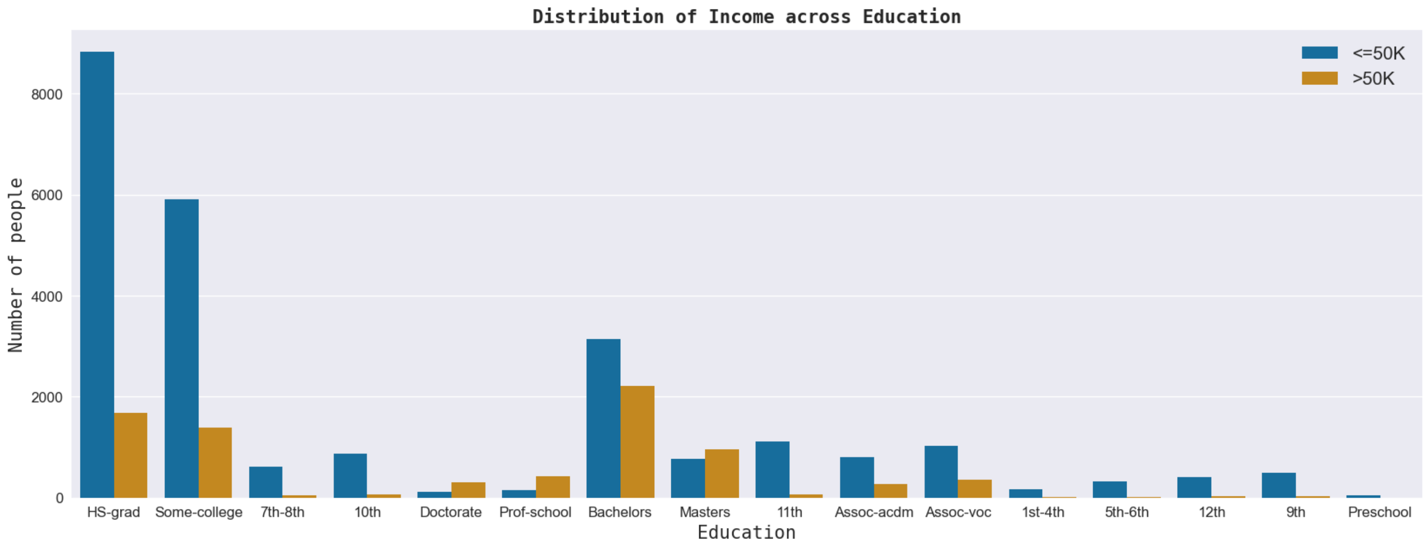




In [24]:

```
# Creating a countplot of income across education
plt.style.use('seaborn')
plt.figure(figsize=(20, 7))
sns.countplot(data['education'],
              hue=data['income'], palette='colorblind')
plt.title('Distribution of Income across Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, '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()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

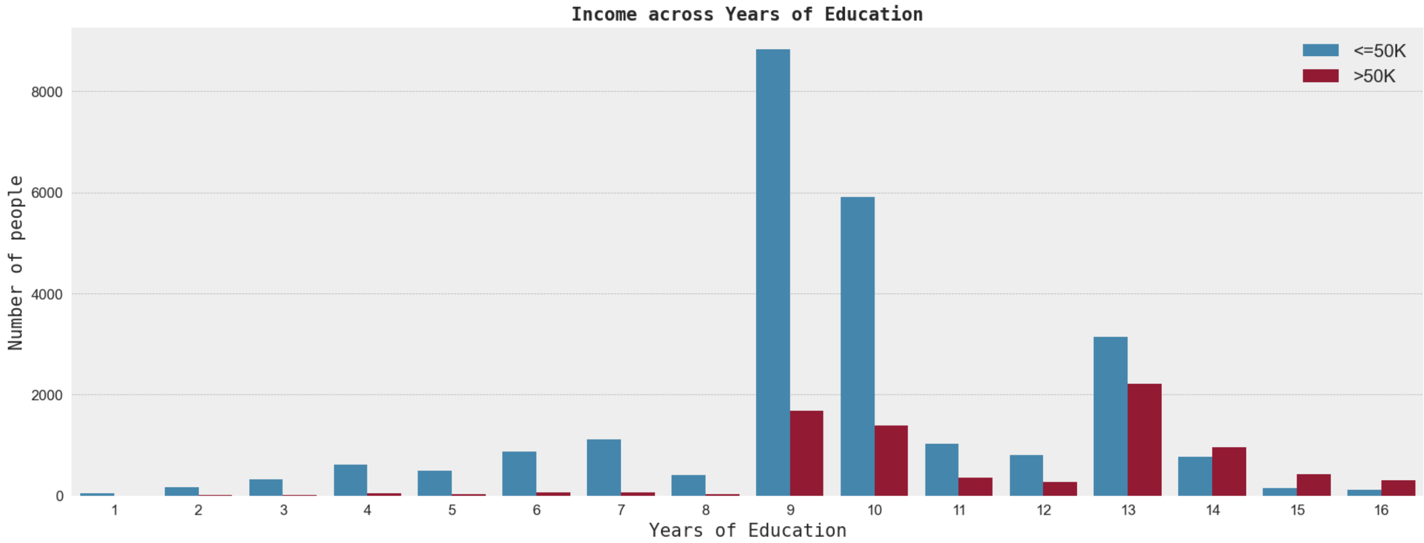


In [25]:

```
# Creating a countplot of income across years of education
plt.style.use('bmh')
plt.figure(figsize=(20, 7))
sns.countplot(data['education.num'],
              hue=data['income'])
plt.title('Income across Years of Education', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, '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()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
```

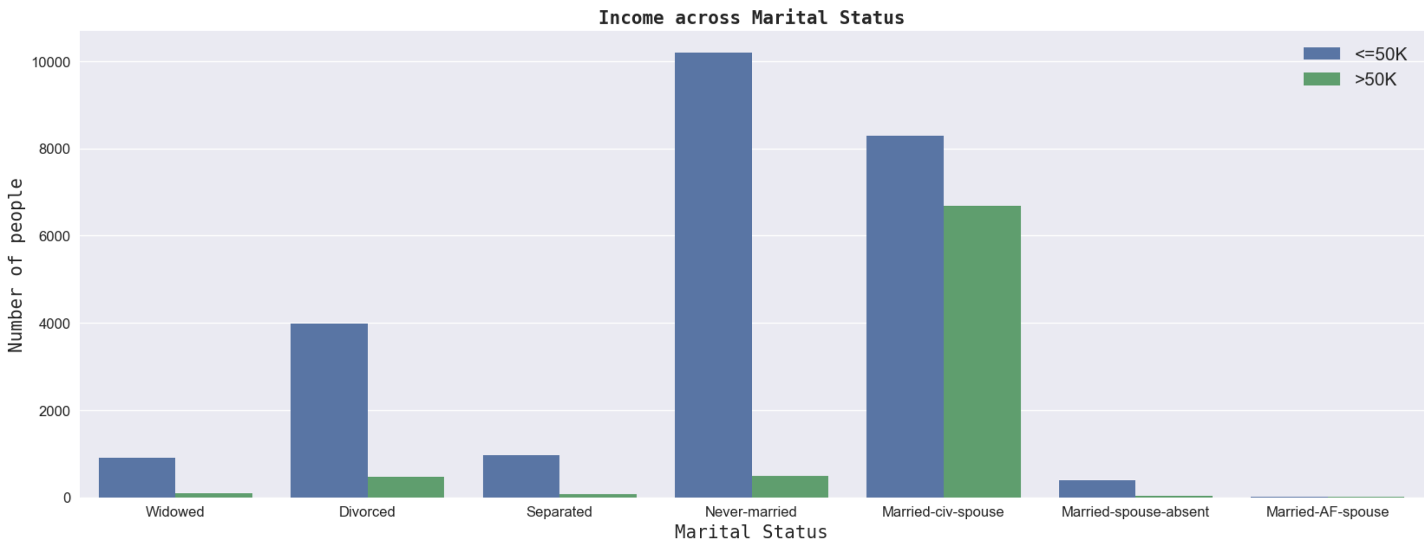
```
corators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [26]:

```
# Creating a countplot of income across Marital Status
plt.style.use('seaborn')
plt.figure(figsize=(20, 7))
sns.countplot(data['marital.status'], hue=data['income'])
plt.title('Income across Marital Status', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, 'fontweight': 'bold'})
plt.xlabel('Marital Status', 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()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

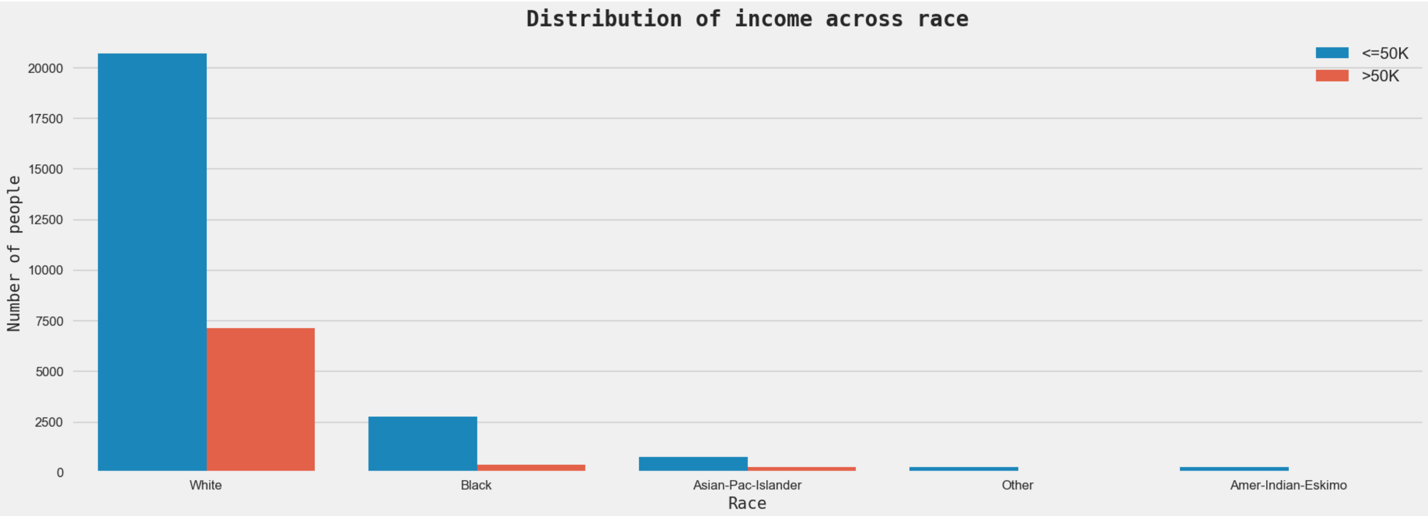


In [27]:

```
# Creating a countplot of income across race
plt.style.use('fivethirtyeight')
plt.figure(figsize=(20, 7))
sns.countplot(data['race'], hue=data['income'])
plt.title('Distribution of income across race', fontdict={
          'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
```

```
plt.xlabel('Race', 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()
```

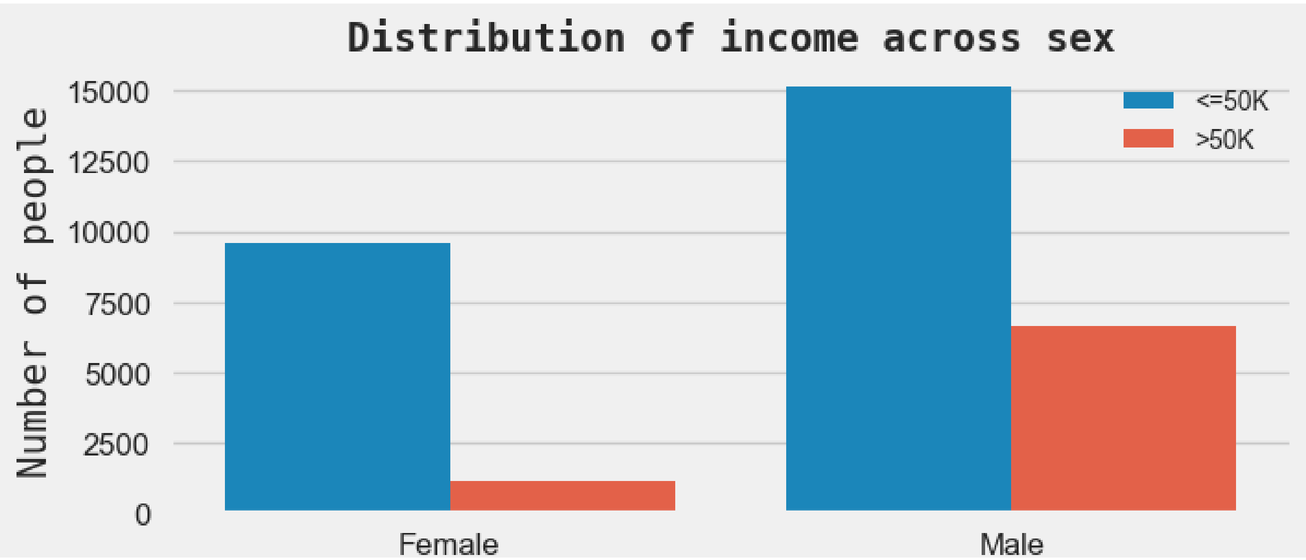
```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [28]:

```
# Creating a countplot of income across sex
plt.style.use('fivethirtyeight')
plt.figure(figsize=(7, 3))
sns.countplot(data['sex'], hue=data['income'])
plt.title('Distribution of income across sex', fontdict={
          'fontname': 'Monospace', 'fontsize': 15, '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()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_de
corators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Sex

In [29]:

```
data.corr()
```

Out[29]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
age	1.000000	-0.076646	0.036527	0.077674	0.057775	0.068756
fnlwgt	-0.076646	1.000000	-0.043195	0.000432	-0.010252	-0.018768
education.num	0.036527	-0.043195	1.000000	0.122630	0.079923	0.148123
capital.gain	0.077674	0.000432	0.122630	1.000000	-0.031615	0.078409
capital.loss	0.057775	-0.010252	0.079923	-0.031615	1.000000	0.054256
hours.per.week	0.068756	-0.018768	0.148123	0.078409	0.054256	1.000000

In [30]:

```
plt.figure(figsize=(20,15))
sns.heatmap(data.corr(),annot= True,linewidths=1, linecolor="white", cbar=True,
            cmap = "CMRmap",xticklabels="auto", yticklabels="auto")

plt.savefig('multi2.png')
```

Observations:

1.Most number of people are young, white, male, high school graduates with 9 to 10 years of education and work 40 hours per week.

1. **Heatmap shows that the dependent feature 'income' is highly correlated with age, numbers of years of education, capital gain and number of hours per week.**

In [31]:

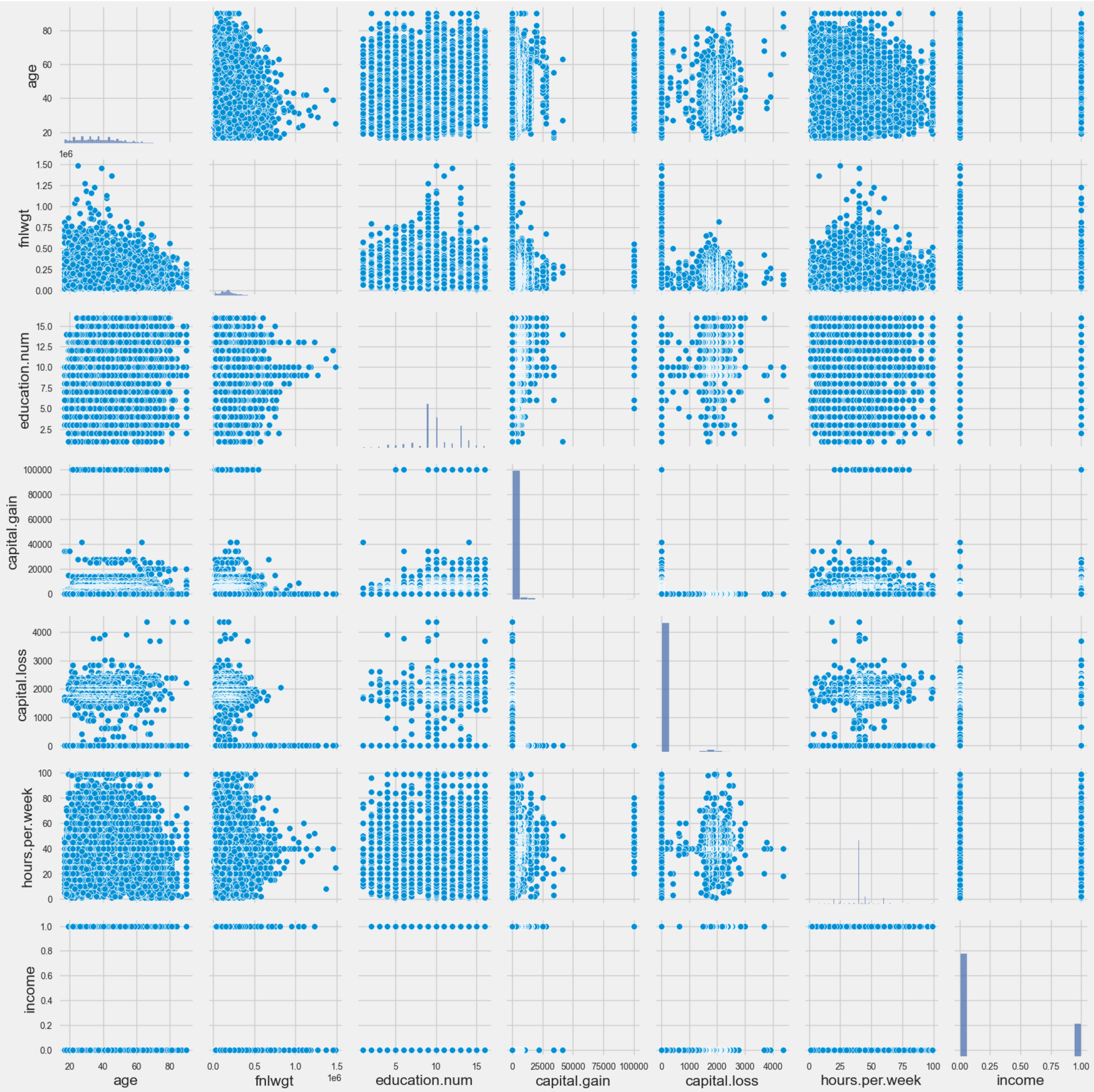
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [32]:

```
data['income'] = le.fit_transform(data['income'])
```

In [33]:

```
# Creating a pairplot of dataset
sns.pairplot(data)
plt.savefig('multi1.png')
plt.show()
```



In [34]:

```
plt.figure(figsize=(20,15))
```

```
sns.heatmap(data.corr(),annot= True,linewidths=1, linecolor="white", cbar=True,
            cmap = "CMRmap",xticklabels="auto", yticklabels="auto")

plt.savefig('multi2.png')
```

In [35]:

```
data = data.replace('?', np.nan)
```

In [36]:

```
# Checking null values
round((data.isnull().sum() / data.shape[0]) * 100, 2).astype(str) + ' %'
```

Out[36]:

```
age               0.0 %
workclass        5.64 %
fnlwgt            0.0 %
education         0.0 %
education.num     0.0 %
marital.status    0.0 %
occupation       5.66 %
relationship      0.0 %
race              0.0 %
sex               0.0 %
capital.gain      0.0 %
capital.loss      0.0 %
hours.per.week    0.0 %
native.country   1.79 %
income            0.0 %
dtype: object
```

In [37]:

```
columns_with_nan = ['workclass', 'occupation', 'native.country']
```

In [38]:

```
for col in columns_with_nan:
    data[col].fillna(data[col].mode()[0], inplace=True)
```

In [39]:

```
from sklearn.preprocessing import LabelEncoder
for col in data.columns:
    if data[col].dtypes == 'object':
        encoder = LabelEncoder()
        data[col] = encoder.fit_transform(data[col])
```

In [40]:

```
corr_data = data.corr('spearman').stack().reset_index(name='corr')
corr_data.loc[corr_data['corr'] == 1, 'corr'] = 0   # Remove diagonal
# Use abs so that we can visualize the impact of negative correaltion
corr_data['abs'] = corr_data['corr'].abs()
corr_data.sort_values('abs', ascending=False).head(n=5)
```

Out[40]:

	level_0	level_1	corr	abs
142	sex	relationship	-0.617570	0.617570
114	relationship	sex	-0.617570	0.617570
5	age	marital.status	-0.374850	0.374850
75	marital.status	age	-0.374850	0.374850
217	income	relationship	-0.329913	0.329913

In [41]:

```
import altair as alt
alt.Chart(corr_data).mark_circle().encode(
    x='level_0',
    y='level_1',
    size='abs',
    color=alt.Color('corr',
                    scale=alt.Scale(scheme='bluegreen',
                                    domain=(-1, 1)))).properties(
    height=250,
    width=500)
```

Out[41]:

In [42]:

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

In [43]:

```
# X = data.iloc[:,:-1].values
# Y = data.iloc[:,-1].values
```

In [44]:

```
X
```

Out[44]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	ca
0	90	3	77053	11	9	6	9	1	4	0	0	

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	ca
1	82	3	132870	11	9	6	3	1	4	0	0	
2	66	3	186061	15	10	6	9	4	2	0	0	
3	54	3	140359	5	4	0	6	4	4	0	0	
4	41	3	264663	15	10	5	9	3	4	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
32556	22	3	310152	15	10	4	10	1	4	1	0	
32557	27	3	257302	7	12	2	12	5	4	0	0	
32558	40	3	154374	11	9	2	6	0	4	1	0	
32559	58	3	151910	11	9	6	0	4	4	0	0	
32560	22	3	201490	11	9	4	0	3	4	1	0	

In [45]:

```
Y
```

Out[45]:

```
0        0
1        0
2        0
3        0
4        0
        ..
32556    0
32557    0
32558    1
32559    0
32560    0
Name: income, Length: 32561, dtype: int32
```

In [46]:

```
from sklearn.ensemble import ExtraTreesClassifier
selector = ExtraTreesClassifier(random_state=42)
```

In [47]:

```
selector.fit(X, Y)
```

Out[47]:

```
▾ ExtraTreesClassifier
ExtraTreesClassifier(random_state=42)
```

In [48]:

```
feature_imp = selector.feature_importances_
```

In [49]:

```
for index, val in enumerate(feature_imp):
    print(index, round((val * 100), 2))
```

```
0 15.59
1 4.13
2 16.71
3 3.87
4 8.66
5 8.04
6 7.27
7 8.62
8 1.47
9 2.84
10 8.83
```

```
11 2.81
12 9.64
13 1.53
```

In [50]:

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  int32
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  int32
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  int32
 6   occupation      32561 non-null  int32
 7   relationship    32561 non-null  int32
 8   race            32561 non-null  int32
 9   sex             32561 non-null  int32
 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  int32
dtypes: int32(8), int64(6)
memory usage: 2.5 MB
```

In [51]:

```
X.columns
```

Out[51]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education.num',
       'marital.status', 'occupation', 'relationship', 'race', 'sex',
       'capital.gain', 'capital.loss', 'hours.per.week', 'native.country'],
      dtype='object')
```

In [52]:

```
X = X.drop(['workclass', 'education', 'race', 'sex',
            'capital.loss', 'native.country'], axis=1)
```

In [53]:

```
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
= 42)
```

In [54]:

```
X_train
```

Out[54]:

	age	fnlwgt	education.num	marital.status	occupation	relationship	capital.gain	hours.per.week
19749	58	290661	9	2	2	0	0	40
1216	62	109463	10	5	11	4	0	33
27962	33	137088	13	2	6	0	0	40
23077	24	117767	12	4	11	3	0	20
10180	67	431426	9	2	0	5	0	2
...	...	...	...	...	...	...	...	...
29802	25	410240	9	4	2	3	0	40

	age	fnlwgt	education.num	marital.status	occupation	relationship	capital.gain	hours.per.week
5390	[illegible]	[illegible]	11	2	0	0	0	40
860	55	238192	9	2	12	0	0	40
15795	41	154076	10	2	0	0	0	50
23654	22	162667	9	4	5	3	0	50

In [55]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [56]:

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[56]:

```
▾ LogisticRegression
LogisticRegression()
```

In [57]:

```
y_pred_logreg = log_reg.predict(X_test)
```

In [58]:

```
accuracy_score(y_test, y_pred_logreg)
```

Out[58]:

```
0.7935305558399017
```

In [59]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_logreg))
```

```
              precision    recall  f1-score   support

           0       0.80      0.98      0.88      7429
           1       0.75      0.21      0.32      2340

    accuracy                           0.79      9769
   macro avg       0.77      0.59      0.60      9769
weighted avg       0.79      0.79      0.75      9769
```

In [60]:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
```

Out[60]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [61]:

```
y_pred_dt = dt.predict(X_test)
```

In [62]:

```
accuracy_score(y_test, y_pred_dt)
```

Out[62]:

```
0.8041764766096837
```

In [63]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      7429
           1       0.59      0.60      0.59      2340

    accuracy                           0.80      9769
   macro avg       0.73      0.73      0.73      9769
weighted avg       0.80      0.80      0.80      9769
```

In [64]:

```
from sklearn.svm import SVC
from sklearn.svm import LinearSVC,SVC
from sklearn.pipeline import make_pipeline
clf = SVC(C = 1.2, gamma =  0.9, kernel= 'rbf')
```

In [65]:

```
clf = clf.fit(X_train,y_train)
```

In [66]:

```
y_pred_svc = clf.predict(X_test)
```

In [67]:

```
#Finding best parameters for our SVC model
# from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score

# param = { 'C': [1],
#     'gamma': [1],
#     'kernel': ['rbf','linear','sigmoid']  }
# grid = GridSearchCV(SVC(), param, refit = True,cv=3, verbose = 3,n_jobs=-1)
```

In [68]:

```
# grid.fit(X_train, y_train)
```

In [69]:

```
accuracy_score(y_test, y_pred_svc)
```

Out[69]:

```
0.7606715119254785
```

In [70]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_svc))
```

```
              precision    recall  f1-score   support
```

```
           0       0.76      1.00      0.86      7429
           1       1.00      0.00      0.00      2340

    accuracy                           0.76      9769
   macro avg       0.88      0.50      0.43      9769
weighted avg       0.82      0.76      0.66      9769
```

In [71]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

In [72]:

```
log_reg_scaled = LogisticRegression()
log_reg_scaled.fit(X_train_scaled, y_train)
```

Out[72]:

```
▾ LogisticRegression
LogisticRegression()
```

In [73]:

```
y_pred_logreg_scaled = log_reg_scaled.predict(X_test_scaled)
```

In [74]:

```
accuracy_score(y_test, y_pred_logreg_scaled)
```

Out[74]:

```
0.8231139318251612
```

In [75]:

```
print('=========Classification Report of Logistic Regression with Scaled Data==========')
print(classification_report(y_test, y_pred_logreg_scaled))
```

```
=========Classification Report of Logistic Regression with Scaled Data==========
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      7429
           1       0.71      0.44      0.54      2340

    accuracy                           0.82      9769
   macro avg       0.78      0.69      0.72      9769
weighted avg       0.81      0.82      0.81      9769
```

In [76]:

```
dt_scaled = DecisionTreeClassifier()
dt_scaled.fit(X_train_scaled,y_train)
```

Out[76]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [77]:

```
y_pred_dt_scaled = dt_scaled.predict(X_test_scaled)
accuracy_score(y_test, y_pred_dt_scaled)
```

Out[77]:

0.8029481011362473

In [78]:

```
from sklearn.metrics import classification_report
print('=========Classification Report of Decision Tree Classifier with Scaled Data=======
===')
print(classification_report(y_test, y_pred_dt_scaled))
```

```
=========Classification Report of Decision Tree Classifier with Scaled Data==========
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      7429
           1       0.59      0.60      0.59      2340

    accuracy                           0.80      9769
   macro avg       0.73      0.73      0.73      9769
weighted avg       0.80      0.80      0.80      9769
```

In [90]:

```
from sklearn.svm import SVC
from sklearn.svm import LinearSVC,SVC
from sklearn.pipeline import make_pipeline
clf_scaled = SVC(C = 1.2, gamma =  0.9, kernel= 'rbf',probability=True)
```

In [91]:

```
clf_scaled = clf_scaled.fit(X_train_scaled,y_train)
```

In [92]:

```
y_pred_svc_scaled = clf_scaled.predict(X_test_scaled)
```

In [96]:

```
accuracy_score(y_test, y_pred_svc_scaled)
```

Out[96]:

0.8451223257242297

In [94]:

```
print('=========Classification Report of Support Vector Classifier with Scaled Data======
====')
print(classification_report(y_test, y_pred_svc_scaled))
```

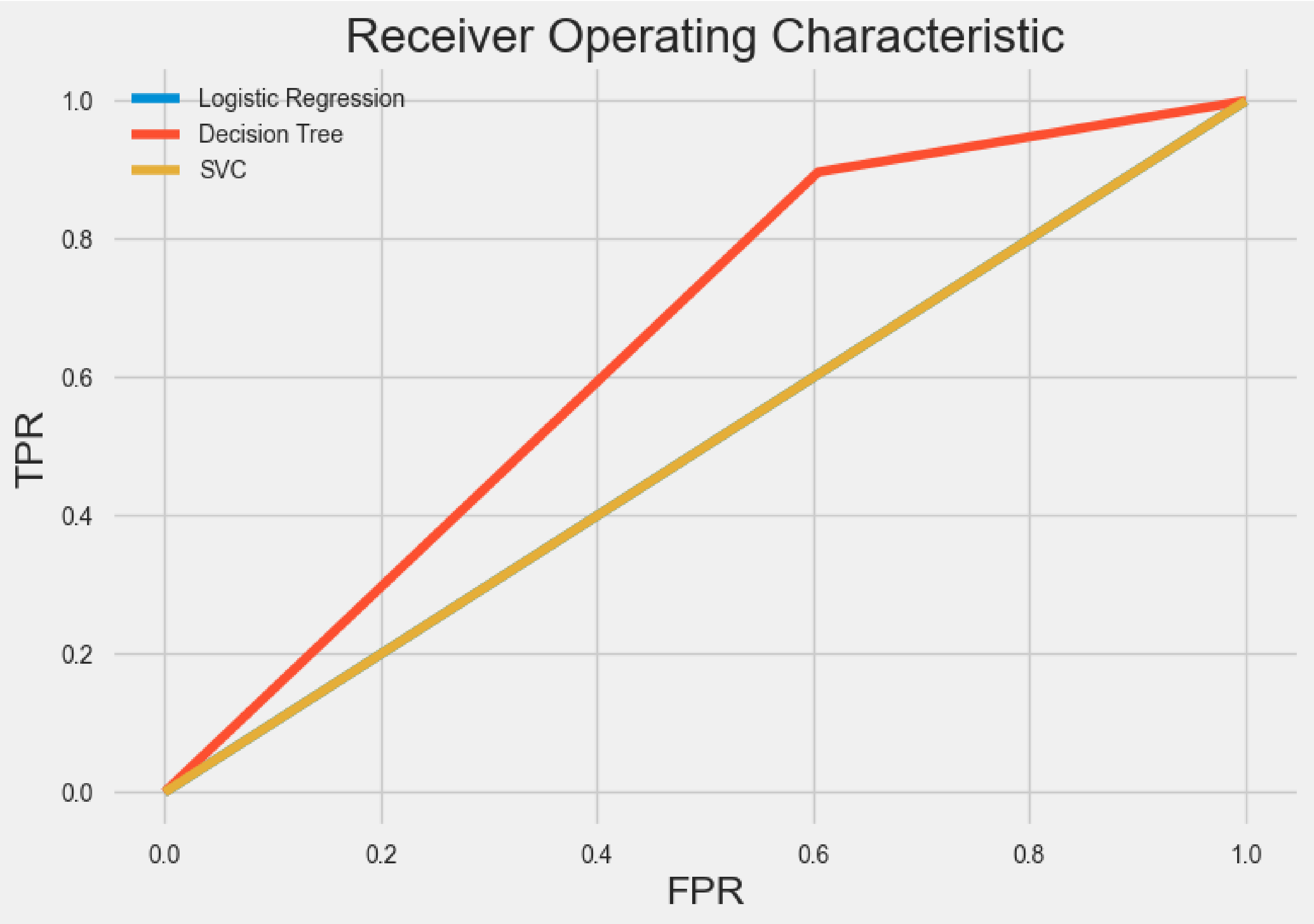
```
=========Classification Report of Support Vector Classifier with Scaled Data==========
              precision    recall  f1-score   support

           0       0.87      0.94      0.90      7429
           1       0.73      0.56      0.63      2340

    accuracy                           0.85      9769
   macro avg       0.80      0.75      0.77      9769
weighted avg       0.84      0.85      0.84      9769
```

In [100]:

```
#ROC Curve
from sklearn.metrics import roc_curve
y_pred_prob1 = log_reg_scaled.predict_proba(X_test)[:,1]
fpr1 , tpr1, thresholds1 = roc_curve(y_test, y_pred_prob1)

y_pred_prob2 = dt_scaled.predict_proba(X_test)[:,1]
fpr2 , tpr2, thresholds2 = roc_curve(y_test, y_pred_prob2)
```

```
y_pred_prob3 = clf_scaled.predict_proba(X_test)[:,1]
fpr3 , tpr3, thresholds3 = roc_curve(y_test, y_pred_prob3)


# plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr1, tpr1, label= "Logistic Regression")
plt.plot(fpr2, tpr2, label= "Decision Tree")
plt.plot(fpr3, tpr3, label= "SVC")

plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title('Receiver Operating Characteristic')
plt.show()
```

```
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\bas
e.py:443: UserWarning: X has feature names, but LogisticRegression was fitted without fea
ture names
  warnings.warn(
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\bas
e.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without
feature names
  warnings.warn(
C:\Users\Shobhandeb\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\bas
e.py:443: UserWarning: X has feature names, but SVC was fitted without feature names
  warnings.warn(
```



In [99]:

```
from sklearn.metrics import roc_auc_score

# auc scores
auc_score1 = roc_auc_score(y_test, y_pred_prob1)
auc_score2 = roc_auc_score(y_test, y_pred_prob2)
auc_score3 = roc_auc_score(y_test, y_pred_prob3)

print(auc_score1,auc_score2,auc_score3)
```

```
0.5 0.6462428655085809 0.5
```

That fact that auc score of Logistic Regression and SVC are same so they got overlapped

Summary:

Build various models like logistic regression, Decision Tree classifier, support vector classifier that gave accuracy: 79%, 80%, 76%, without hyperparameter tuning.

With scaling of the Train and Test data, the accuracy was improved for logistic regression, Decision Tree classifier, support vector classifier to 82%, 80%, 85% respectively.

In [ ]:

```

```