Project : Census Income

In this project, you are going to work on the **Census Income** dataset from the UCI Machine Learning Repository that contains the income information for over 48,000 individuals taken from the 1994 US census.

For more details about this dataset, you can refer to the following link: https://archive.ics.uci.edu/ml/datasets/census+income

Problem Statement:

In this project, initially you need to preprocess the data and then develop an understanding of the different features of the data by performing exploratory analysis and creating visualizations. Further, after having sufficient knowledge about the attributes, you will perform a predictive task of classification to predict whether an individual makes over 50,000 a year or less by using different machine learning algorithms.

Tasks To Be Performed:

1. Perform Exploratory Data Analysis to find key insights.
2. Use various machine learning algorithms to predict the response variable.

Exploratory Data Analysis:

In [66]:
```
#1. Perform Exploratory Data Analysis to find key insights
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('census-income (7).csv')
df.head()
```

Out[66]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [67]:
```
df.columns
```

Out[67]:
```
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 [68]:
```
df.rename(columns={df.columns[14]: 'Annual_Income'}, inplace=True)
```

In [69]:
```
df.head()
```

Out[69]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Annual_Income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [70]:
```
# Display the summary statistics of the numerical columns
df.describe()
```

Out[70]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

```
In [71]: # Display the frequency counts of the categorical columns
         df.value_counts()
```

```
Out[71]: age   workclass          fnlwgt   education      education-num   marital-status       occupation         relationship    race
sex      capital-gain    capital-loss    hours-per-week    native-country  Annual_Income
25    Private            195994   1st-4th        2               Never-married        Priv-house-serv    Not-in-family   White
Female  0                0               40                Guatemala       <=50K         3
23    Private            240137   5th-6th        3               Never-married        Handlers-cleaners  Not-in-family   White
Male    0                0               55                Mexico          <=50K         2
38    Private            207202   HS-grad        9               Married-civ-spouse   Machine-op-inspct  Husband         White
Male    0                0               48                United-States   >50K          2
30    Private            144593   HS-grad        9               Never-married        Other-service      Not-in-family   Black
Male    0                0               40                ?               <=50K         2
49    Self-emp-not-inc   43479    Some-college   10              Married-civ-spouse   Craft-repair       Husband         White
Male    0                0               40                United-States   <=50K         2

..
31    Private            128567   HS-grad        9               Married-civ-spouse   Craft-repair       Husband         White
Male    0                0               40                United-States   <=50K         1
                         128493   HS-grad        9               Divorced             Other-service      Not-in-family   White
Female  0                0               25                United-States   <=50K         1
                         128220   7th-8th        4               Widowed              Adm-clerical       Not-in-family   White
Female  0                0               35                United-States   <=50K         1
                         127610   Bachelors      13              Married-civ-spouse   Prof-specialty     Wife            White
Female  0                0               40                United-States   >50K          1
90    Self-emp-not-inc   282095   Some-college   10              Married-civ-spouse   Farming-fishing    Husband         White
Male    0                0               40                United-States   <=50K         1
Length: 32537, dtype: int64
```

In [72]:
```python
# Display the correlation matrix of the numerical columns
df.corr()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_29368\2274382617.py:2: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  df.corr()
```

Out[72]:

	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 [73]:
```
df.isnull().sum()
```

Out[73]:
```
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
Annual_Income     0
dtype: int64
```

machine learning:

In [74]:
```
df.head()
```

Out[74]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Annual_Income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [75]:
```
# Drop columns 1, 3, 5, 6, 7, 8, 9,13 and 14 by index
df.drop(df.columns[[1, 3, 5, 6, 7, 8, 9, 13,14]], axis=1, inplace=True)
```

In [76]:
```
df.head()
```

Out[76]:

	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

In [78]:
```python
# Print the column names of df
print(df.columns)
```

```
Index(['age', ' fnlwgt', ' education-num', ' capital-gain', ' capital-loss',
       ' hours-per-week'],
      dtype='object')
```

In [80]:
```python
# Define the response variable (y) and the features (X)
y = df[" hours-per-week"]
X = df.drop(" hours-per-week", axis=1)
```

In [81]:
```python
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Encode the categorical features as numbers
le = LabelEncoder()
X = X.apply(le.fit_transform)
```

In [82]:
```python
from sklearn.model_selection import train_test_split

# Split the data into train and test sets (80/20 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [83]:
```python
# Scale the numerical features to have zero mean and unit variance
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Linear Regression

In [86]:
```
from sklearn.linear_model import LinearRegression, LogisticRegression

# Fit and evaluate a linear regression model
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[86]:
```
▾ LinearRegression
LinearRegression()
```

In [87]:
```
y_pred_lr = lr.predict(X_test)
```

In [88]:
```
from sklearn.metrics import mean_squared_error, accuracy_score, confusion_matrix
mse_lr = mean_squared_error(y_test, y_pred_lr)
print("Linear regression MSE:", mse_lr)
```

```
Linear regression MSE: 148.74291085824908
```

Logistic Regression

In [89]:
```
# Fit and evaluate a logistic regression model
logr = LogisticRegression()
logr.fit(X_train, y_train)
```

```
C:\Users\user\anaconda3\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[89]:
```
▾ LogisticRegression
LogisticRegression()
```

In [90]:
```
y_pred_logr = logr.predict(X_test)
```

In [91]:
```
acc_logr = accuracy_score(y_test, y_pred_logr)
cm_logr = confusion_matrix(y_test, y_pred_logr)
print("Logistic regression accuracy:", acc_logr)
print("Logistic regression confusion matrix:", cm_logr)
```

```
Logistic regression accuracy: 0.46460924305235685
Logistic regression confusion matrix: [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Decision Tree Classifier

In [92]:
```
from sklearn.tree import DecisionTreeClassifier

# Fit and evaluate a decision tree model
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[92]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [93]:
```
y_pred_dt = dt.predict(X_test)
```

In [94]:
```
acc_dt = accuracy_score(y_test, y_pred_dt)
cm_dt = confusion_matrix(y_test, y_pred_dt)
print("Decision tree accuracy:", acc_dt)
print("Decision tree confusion matrix:", cm_dt)
```

```
Decision tree accuracy: 0.2528788576692768
Decision tree confusion matrix: [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Random Forest Classifier

In [95]:
```
from sklearn.ensemble import RandomForestClassifier

# Fit and evaluate a random forest model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[95]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [96]:
```
y_pred_rf = rf.predict(X_test)
```

In [97]:
```
acc_rf = accuracy_score(y_test, y_pred_rf)
cm_rf = confusion_matrix(y_test, y_pred_rf)
print("Random forest accuracy:", acc_rf)
print("Random forest confusion matrix:", cm_rf)
```

```
Random forest accuracy: 0.305542760632581
Random forest confusion matrix: [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```