```
# Import libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


# Adult dataset path
adult_dataset_path = "/content/adult_dataset.csv"

# Function for loading adult dataset
def load_adult_data(adult_path=adult_dataset_path):
    csv_path = os.path.join(adult_path)
    return pd.read_csv(csv_path)


# Calling load adult function and assigning to a new variable df
df = load_adult_data()
# load top 3 rows values from adult dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
print ("Rows     : " ,df.shape[0])
print ("Columns  : " ,df.shape[1])
print ("\nFeatures : \n" ,df.columns.tolist())
print ("\nMissing values :  ", df.isnull().sum().values.sum())
print ("\nUnique values :  \n",df.nunique())
```

```
Rows     :  32561
Columns  :  15

Features :
 ['age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.status', 'occupat

Missing values :   0

Unique values :
 age                73
workclass            9
fnlwgt           21648
education           16
education.num       16
marital.status       7
occupation          15
relationship         6
race                 5
sex                  2
capital.gain       119
capital.loss        92
hours.per.week      94
native.country      42
income               2
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
```

```
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
df.describe()
```

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

```
df_check_missing_workclass = (df['workclass']=='?').sum()
df_check_missing_workclass
```

```
1836
```

```
df_check_missing_occupation = (df['occupation']=='?').sum()
df_check_missing_occupation
```

```
1843
```

```
df_missing = (df=='?').sum()
df_missing
```

	0
age	0
workclass	1836
fnlwgt	0
education	0
education.num	0
marital.status	0
occupation	1843
relationship	0
race	0
sex	0
capital.gain	0
capital.loss	0
hours.per.week	0
native.country	583
income	0

dtype: int64

```
percent_missing = (df=='?').sum() * 100/len(df)
percent_missing
```

	0
age	0.000000
workclass	5.638647
fnlwgt	0.000000
education	0.000000
education.num	0.000000
marital.status	0.000000
occupation	5.660146
relationship	0.000000
race	0.000000
sex	0.000000
capital.gain	0.000000
capital.loss	0.000000
hours.per.week	0.000000
native.country	1.790486
income	0.000000

dtype: float64

```
df.apply(lambda x: x !='?',axis=1).sum()
```

	0
age	32561
workclass	30725
fnlwgt	32561
education	32561
education.num	32561
marital.status	32561
occupation	30718
relationship	32561
race	32561
sex	32561
capital.gain	32561
capital.loss	32561
hours.per.week	32561
native.country	31978
income	32561

dtype: int64

```
# dropping the rows having missing values in workclass
df = df[df['workclass'] !='?']
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relatio
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unm
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unm
6	38	Private	150601	10th	6	Separated	Adm-	Unm

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# select all categorical variables
df_categorical = df.select_dtypes(include=['object'])

# checking whether any other column contains '?' value
df_categorical.apply(lambda x: x=='?',axis=1).sum()
```

	0
workclass	0
education	0
marital.status	0
occupation	7
relationship	0
race	0
sex	0
native.country	556
income	0

dtype: int64

```
# dropping the "?"s from occupation and native.country
df = df[df['occupation'] !='?']
df = df[df['native.country'] !='?']


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             30162 non-null  int64 
 1   workclass       30162 non-null  object
 2   fnlwgt          30162 non-null  int64 
 3   education       30162 non-null  object
 4   education.num   30162 non-null  int64 
 5   marital.status  30162 non-null  object
 6   occupation      30162 non-null  object
 7   relationship    30162 non-null  object
 8   race            30162 non-null  object
 9   sex             30162 non-null  object
 10  capital.gain    30162 non-null  int64 
 11  capital.loss    30162 non-null  int64 
 12  hours.per.week  30162 non-null  int64 
 13  native.country  30162 non-null  object
 14  income          30162 non-null  object
```

```
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
from sklearn import preprocessing

# encode categorical variables using label Encoder

# select all categorical variables
df_categorical = df.select_dtypes(include=['object'])
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	native
1	Private	HS-grad	Widowed	Exec-managerial	Not-in-family	White	Female	Unit
3	Private	7th-8th	Divorced	Machine-op-inspct	Unmarried	White	Female	Unit
4	Private	Some-college	Separated	Prof-specialty	Own-child	White	Female	Unit
5	[illegible]	[illegible]	[illegible]	Other-	[illegible]	[illegible]	[illegible]	[illegible]

Next steps: Generate code with df_categorical | View recommended plots | New interactive sheet

```
# apply label encoder to df_categorical
le = preprocessing.LabelEncoder()
df_categorical = df_categorical.apply(le.fit_transform)
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	native.cou
1	2	11	6	3	1	4	0	
3	2	5	0	6	4	4	0	
4	2	15	5	9	3	4	0	
5	2	11	0	7	4	4	0	
6	2	0	5	0	4	4	1	

Next steps: Generate code with df_categorical | View recommended plots | New interactive sheet

```
# Next, Concatenate df_categorical dataframe with original df (dataframe)

# first, Drop earlier duplicate columns which had categorical values
df = df.drop(df_categorical.columns,axis=1)
df = pd.concat([df,df_categorical],axis=1)
df.head()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	ed
1	82	132870	9	0	4356	18	2	
3	54	140359	4	0	3900	40	2	
4	41	264663	10	0	3900	40	2	
5	34	216864	9	0	3770	45	2	
6	38	150601	6	0	3770	40	2	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# convert target variable income to categorical
df['income'] = df['income'].astype('category')


# Importing train_test_split
from sklearn.model_selection import train_test_split


# Putting independent variables/features to X
X = df.drop('income',axis=1)

# Putting response/dependent variable/feature to y
y = df['income']


X.head(3)
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	ed
1	82	132870	9	0	4356	18	2	
3	54	140359	4	0	3900	40	2	
4	41	264663	10	0	3900	40	2	

Next steps: Generate code with X | View recommended plots | New interactive sheet

```
# Splitting the data into train and test
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.30,random_state=99)
```

```
# Importing decision tree classifier from sklearn library
from sklearn.tree import DecisionTreeClassifier

# Fitting the decision tree with default hyperparameters, apart from
# max_depth which is 5 so that we can plot and read the tree.
dt_default = DecisionTreeClassifier(max_depth=5)
dt_default.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

```
# Let's check the evaluation metrics of our default model

# Importing classification report and confusion matrix from sklearn metrics
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

# making predictions
y_pred_default = dt_default.predict(X_test)

# Printing classifier report after prediction
print(classification_report(y_test,y_pred_default))
```

```
              precision    recall  f1-score   support

           0       0.86      0.95      0.91      6867
           1       0.78      0.52      0.63      2182

    accuracy                           0.85      9049
   macro avg       0.82      0.74      0.77      9049
weighted avg       0.84      0.85      0.84      9049
```

```
# Printing confusion matrix and accuracy
print(confusion_matrix(y_test,y_pred_default))
print(accuracy_score(y_test,y_pred_default))
```

```
[[6553  314]
 [1039 1143]]
0.8504807161012267
```

```
# Importing required packages for visualization
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydotplus,graphviz

# Putting features
features = list(df.columns[1:])
features
```

```
['fnlwgt',
 'education.num',
 'capital.gain',
 'capital.loss',
 'hours.per.week',
 'workclass',
 'education',
 'marital.status',
 'occupation',
 'relationship',
 'race',
```

```
dot_data = StringIO()
```