```
'''
*    Author: Olufemi Onimole
*    Date: 2019
*    Code version: 0.1

Dataset Source:
Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]

@misc{Dua:2019 ,
author = "Dua, Dheeru and Graff, Casey",
year = "2017",
title = "{UCI} Machine Learning Repository",
url = "http://archive.ics.uci.edu/ml",
institution = "University of California, Irvine, School of Information and Computer Sciences"
'''
```

```
!pip install --upgrade tensorflow
from __future__ import absolute_import, division, print_function, unicode_literals
import functools

import pandas as pd
import numpy as np
import tensorflow as tf
```

csv file path

```
train_file_path = "/content/drive/My Drive/Colab Notebooks/Projects/Census1994/adult.csv"
test_file_path = "/content/drive/My Drive/Colab Notebooks/Projects/Census1994/adult_test.csv"
```

read csv as dataframe

```
df = pd.read_csv(train_file_path)
df_test = pd.read_csv(test_file_path)
```

examine dataframe head

```
df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	ra
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Wh
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Wh
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Wh
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Bl

examine dataframe datatypes

```
df.dtypes
```

```
age               int64
workclass        object
fnlwgt            int64
education        object
education-num     int64
marital-status   object
occupation       object
relationship     object
race             object
sex              object
capital-gain      int64
capital-loss      int64
hours-per-week    int64
native-country   object
income           object
dtype: object
```

remove unneeded features

```
df.pop('fnlwgt')
df_test.pop('fnlwgt')
```

```
0        226802
1         89814
2        336951
3        160323
4        103497
          ...
16276    215419
16277    321403
16278    374983
16279     83891
16280    182148
Name: fnlwgt, Length: 16281, dtype: int64
```

examine dataframe

```
df.head()
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	s
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	M
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M

convert categories to numerical values

```
columns = ["workclass","education","marital-status","occupation","relationship","race","sex",
]

for column in columns:
  df[column] = pd.Categorical(df[column])
  df[column] = df[column].cat.codes
  df_test[column] = pd.Categorical(df_test[column])
  df_test[column] = df_test[column].cat.codes

df.head()
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
0	39	7	9	13	4	1	1	4	1
1	50	6	9	13	2	4	0	4	1
2	38	4	11	9	0	6	1	4	1
3	53	4	1	7	2	6	0	2	1
4	28	4	9	13	2	10	5	2	0

shuffle data

```
df = df.sample(frac=1).reset_index(drop=True)
df.head()
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
0	24	4	11	9	2	3	0	4	1
1	36	4	15	10	2	3	0	4	1
2	33	4	11	9	0	12	1	4	1
3	23	4	11	9	4	7	3	4	1
4	21	4	11	9	2	1	0	4	1

balance data

```
target_counts = df['income'].value_counts()
print(target_counts)
df = df.groupby('income').head(target_counts.min())
balanced_target_counts = df['income'].value_counts()
print(balanced_target_counts)
```

```
0    24720
1     7841
Name: income, dtype: int64
1    7841
0    7841
Name: income, dtype: int64
```

split training and validation

```
dflen = len(df.index)
```

```
split_amount = int(dflen * .8)
df_train = df[:split_amount]
df_val = df[split_amount:]
```

separate targets from data

```
target_train = df_train.pop('income')
target_val = df_val.pop('income')
target_test = df_test.pop('income')
```

convert dataframe to dataset

```
train_dataset = tf.data.Dataset.from_tensor_slices((df_train.values, target_train.values))
print(train_dataset)
val_dataset = tf.data.Dataset.from_tensor_slices((df_val.values, target_val.values))
print(val_dataset)
test_dataset = tf.data.Dataset.from_tensor_slices((df_test.values, target_test.values))
print(test_dataset)
```

```
<TensorSliceDataset shapes: ((13,), ()), types: (tf.int64, tf.int8)>
<TensorSliceDataset shapes: ((13,), ()), types: (tf.int64, tf.int8)>
<TensorSliceDataset shapes: ((13,), ()), types: (tf.int64, tf.int8)>
```

examine dataset

```
for feat, targ in train_dataset.take(5):
  print ('Features: {}, Target: {}'.format(feat, targ))
```

```
Features: [24  4 11  9  2  3  0  4  1  0  0 45 39], Target: 0
Features: [36  4 15 10  2  3  0  4  1  0  0 40  0], Target: 1
Features: [33  4 11  9  0 12  1  4  1  0  0 40 39], Target: 0
Features: [23  4 11  9  4  7  3  4  1  0  0 40 39], Target: 0
Features: [21  4 11  9  2  1  0  4  1  0  0 40 39], Target: 0
```

shuffle and batch the dataset

```
train_dataset = train_dataset.shuffle(len(df)).batch(16)
val_dataset = val_dataset.batch(16)
test_dataset = test_dataset.batch(16)
```

create and train a model

```
def get_compiled_model():
  model = tf.keras.Sequential([
                              tf.keras.layers.Dense(50, activation='relu'),
                              tf.keras.layers.Dense(50, activation='relu'),
                              tf.keras.layers.Dense(100, activation='relu'),
```

```
                                    tf.keras.layers.Dense(100, activation='relu'),
                                    tf.keras.layers.Dense(200, activation='relu'),
                                    tf.keras.layers.Dense(200, activation='relu'),
                                    tf.keras.layers.Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model


model = get_compiled_model()
model.fit(train_dataset, validation_data=val_dataset, epochs=30)
```

```
Epoch 1/30
785/785 [==============================] - 7s 9ms/step - loss: 1.4850 - accuracy: 0.7267
Epoch 2/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4833 - accuracy: 0.7685
Epoch 3/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4380 - accuracy: 0.7845
Epoch 4/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4224 - accuracy: 0.7914
Epoch 5/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4097 - accuracy: 0.8001
Epoch 6/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4542 - accuracy: 0.7950
Epoch 7/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4651 - accuracy: 0.7971
Epoch 8/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4025 - accuracy: 0.7997
Epoch 9/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4038 - accuracy: 0.8033
Epoch 10/30
785/785 [==============================] - 4s 4ms/step - loss: 0.4369 - accuracy: 0.7828
Epoch 11/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4020 - accuracy: 0.8045
Epoch 12/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4261 - accuracy: 0.7985
Epoch 13/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4184 - accuracy: 0.7974
Epoch 14/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4148 - accuracy: 0.7953
Epoch 15/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4138 - accuracy: 0.7973
Epoch 16/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4129 - accuracy: 0.8014
Epoch 17/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4100 - accuracy: 0.8011
Epoch 18/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4136 - accuracy: 0.8003
Epoch 19/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4076 - accuracy: 0.8033
Epoch 20/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4107 - accuracy: 0.7998
Epoch 21/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4097 - accuracy: 0.8004
Epoch 22/30
785/785 [==============================] - 3s 4ms/step - loss: 0.4090 - accuracy: 0.8024
Epoch 23/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3864 - accuracy: 0.8084
Epoch 24/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3835 - accuracy: 0.8160
Epoch 25/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3814 - accuracy: 0.8120
Epoch 26/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3783 - accuracy: 0.8165
Epoch 27/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3750 - accuracy: 0.8192
Epoch 28/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3817 - accuracy: 0.8154
Epoch 29/30
```

```
785/785 [==============================] - 3s 4ms/step - loss: 0.3752 - accuracy: 0.8173
Epoch 30/30
785/785 [==============================] - 3s 4ms/step - loss: 0.3763 - accuracy: 0.8180
<tensorflow.python.keras.callbacks.History at 0x7f00c9e962b0>
```

```
model.evaluate(test_dataset, verbose=2)
```

```
1018/1018 - 1s - loss: 0.3621 - accuracy: 0.8221
[0.3620566149518509, 0.82212394]
```