# **Extra Credit: Putting it All Together: Data Cleaning to Model Evaluation**

Choose any dataset that you will like to work with and is suitable for classification.  : US\_Census\_Dataset

point in the dataset must have a class label.  : Whether Income column is less than or equal to 50K or more than the 50K

What is the number of rows & columns in this dataset?

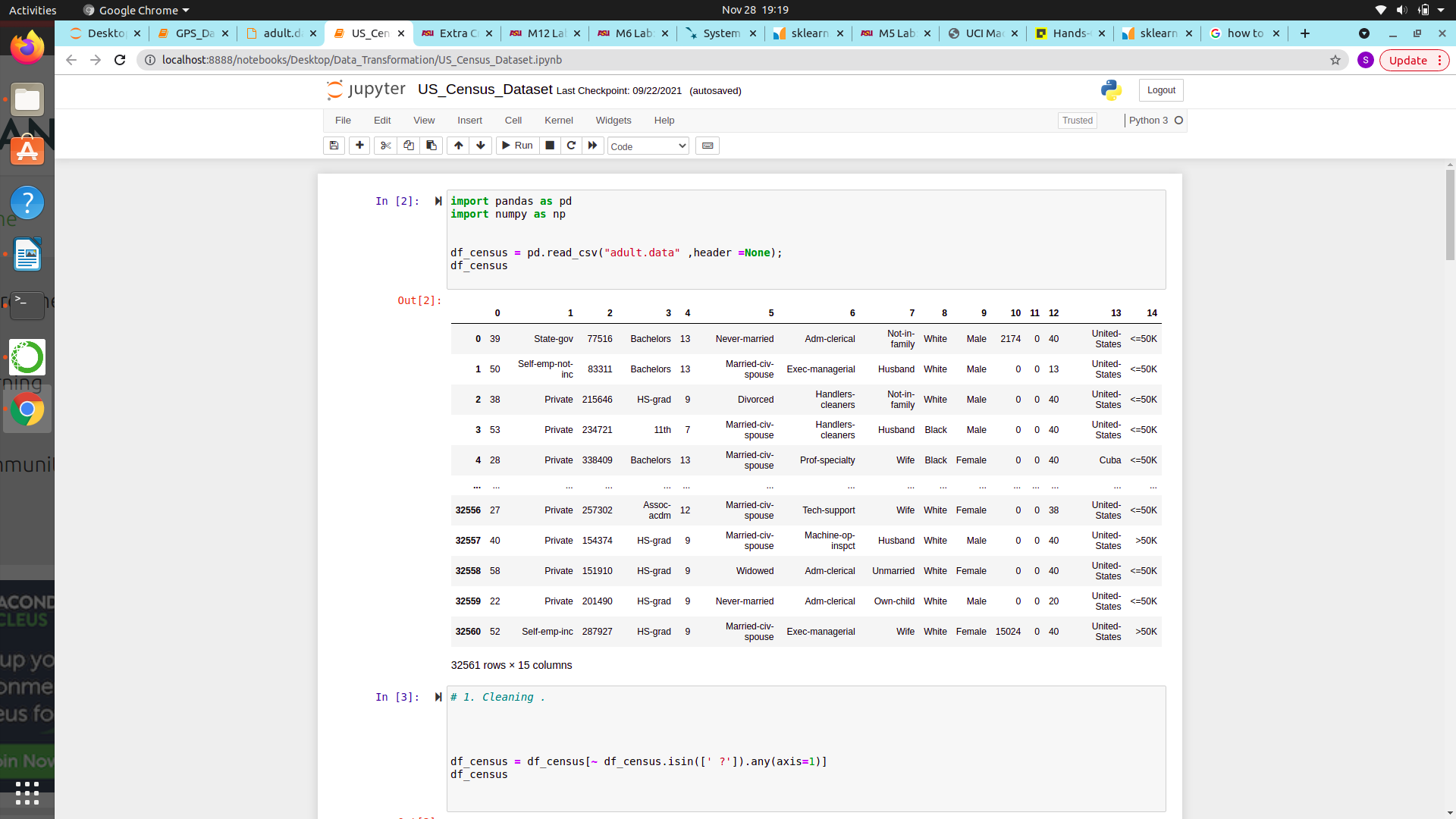
32561 rows × 15 columns

Write a script that implements the following steps:

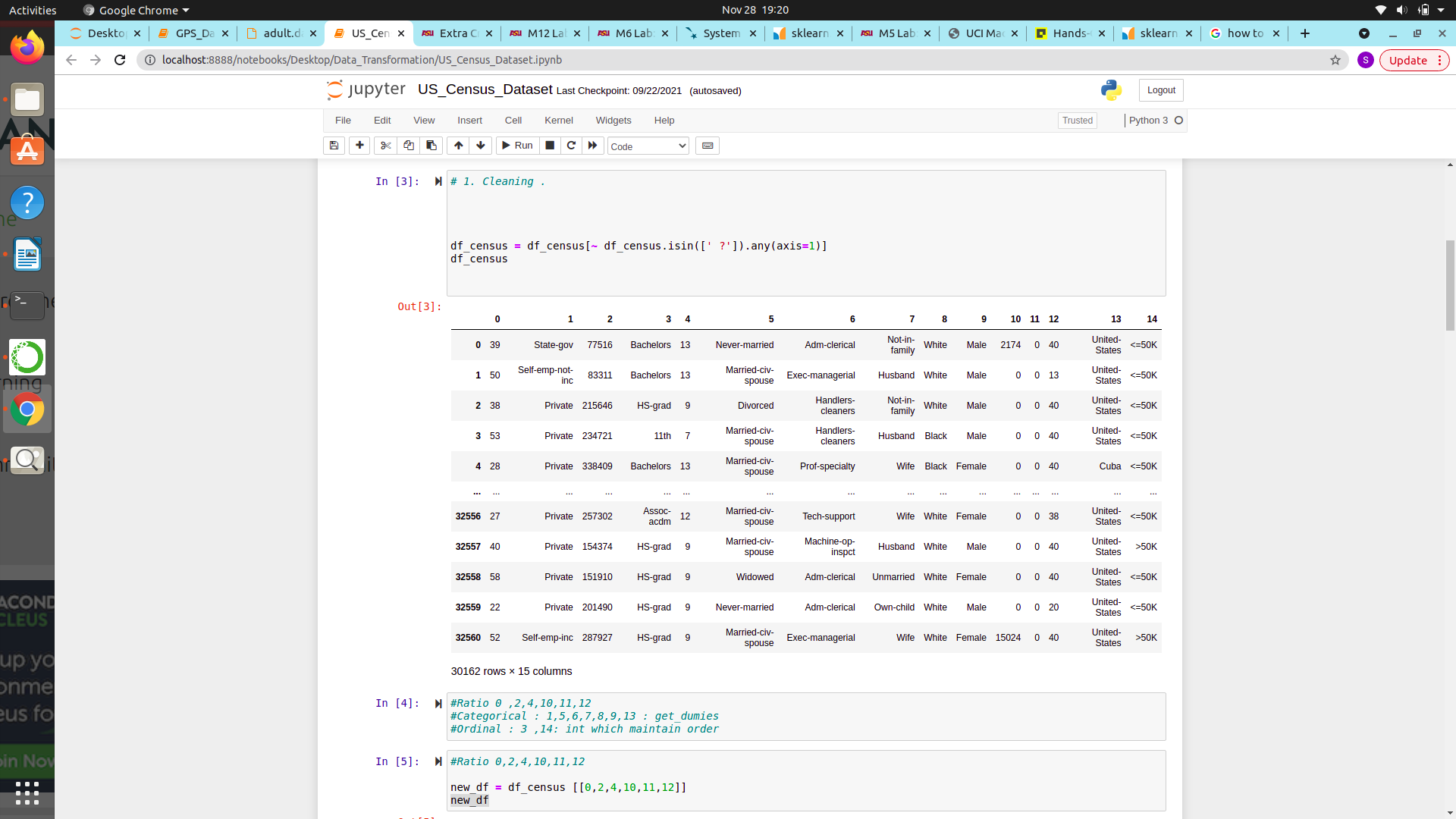
1. Clean the dataset by removing any rows/columns with missing values. Include an explanation for each removed row/column and the number of missing values in it.

import pandas as pd  
import numpy as np

df\_census = pd.read\_csv("adult.data" ,header =None);  
df\_census

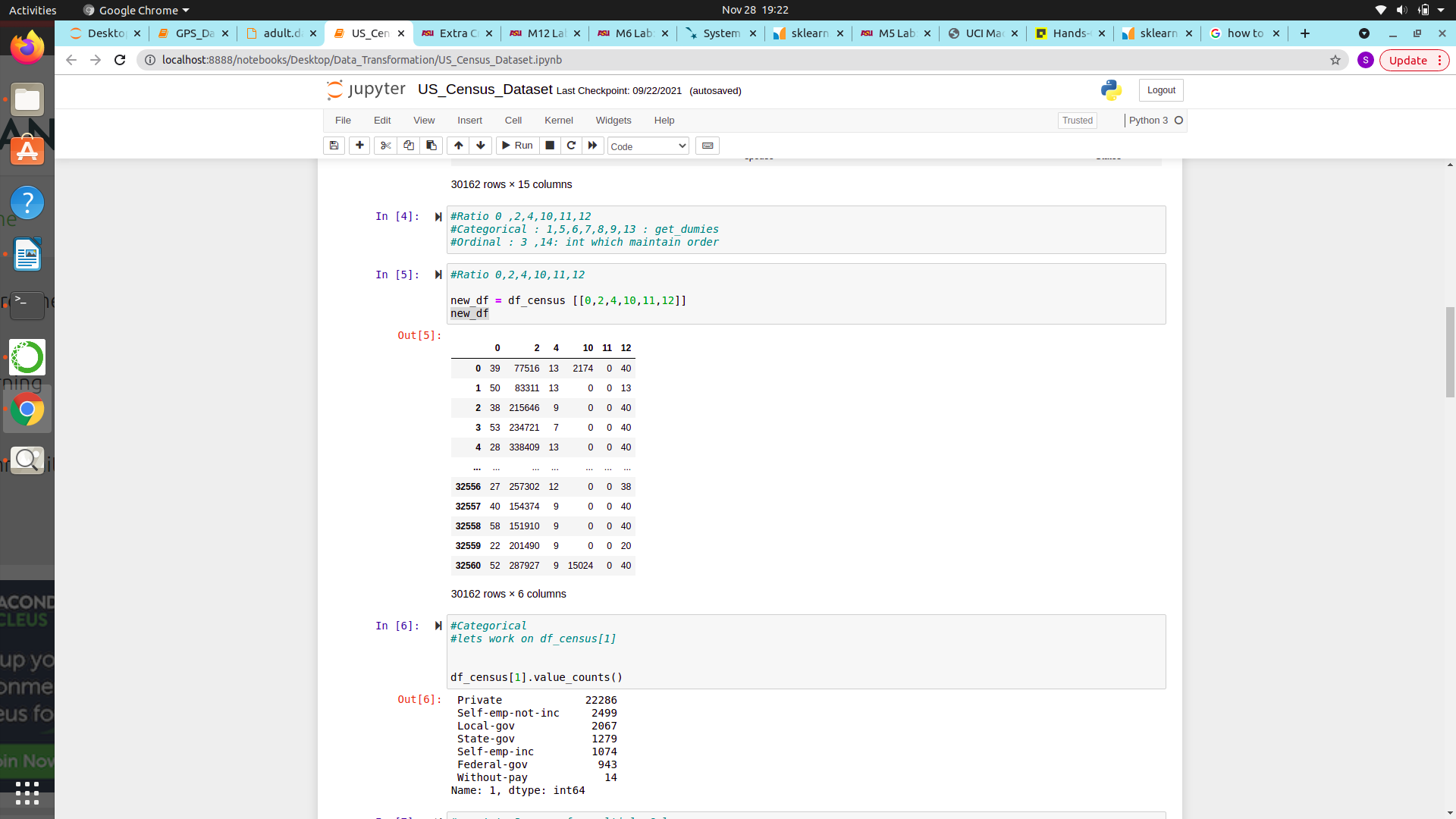


# 1. Cleaning .  
  
  
  
  
df\_census = df\_census[~ df\_census.isin([' ?']).any(axis=1)]  
df\_census

#Ratio 0 ,2,4,10,11,12

#Categorical : 1,5,6,7,8,9,13 : get\_dumies  
#Ordinal : 3 ,14: int which maintain order

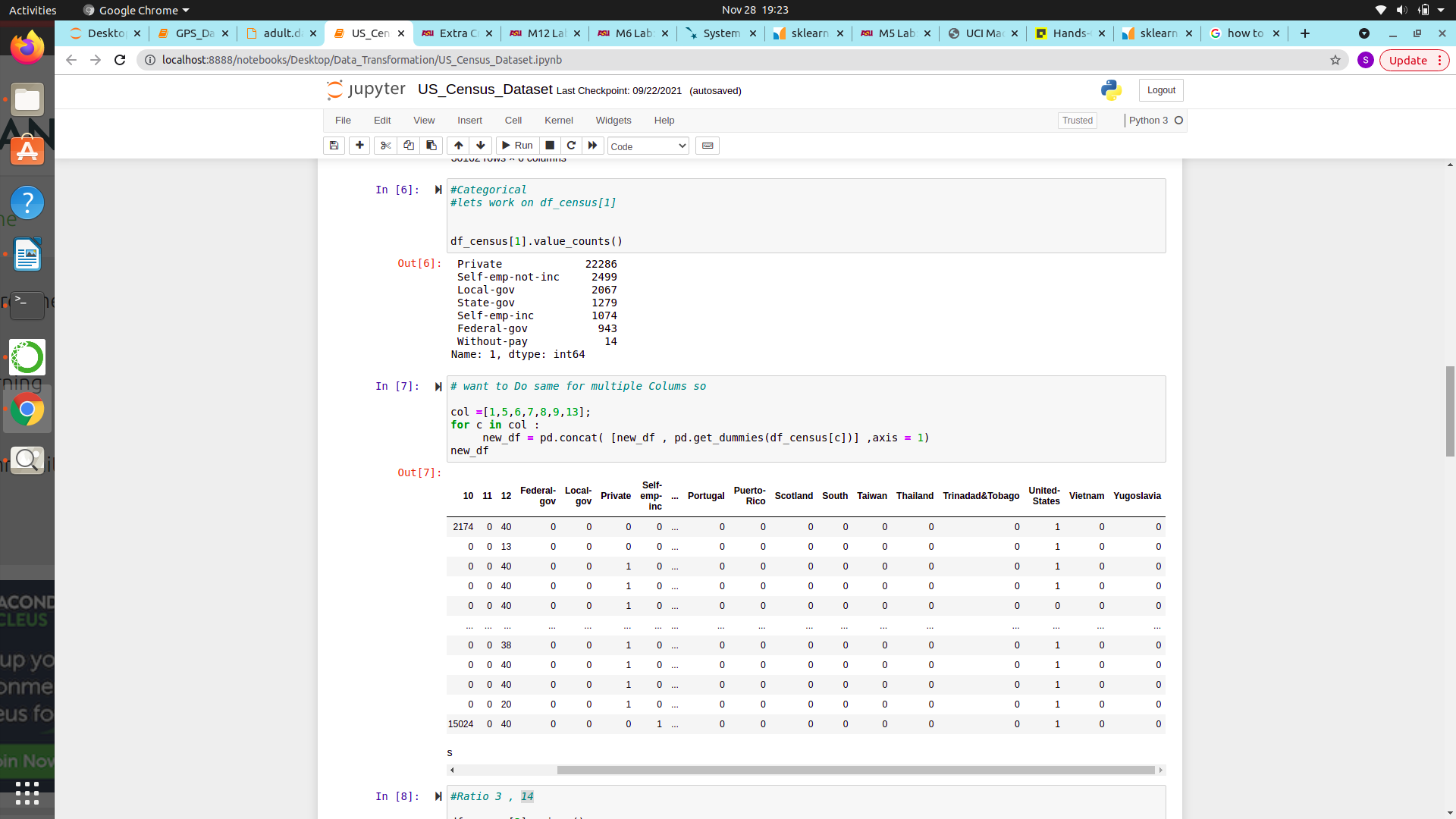
#Ratio 0,2,4,10,11,12  
  
new\_df = df\_census [[0,2,4,10,11,12]]  
new\_df



#Categorical   
#lets work on df\_census[1]  
  
  
df\_census[1].value\_counts()

Private 22286  
 Self-emp-not-inc 2499  
 Local-gov 2067  
 State-gov 1279  
 Self-emp-inc 1074  
 Federal-gov 943  
 Without-pay 14  
Name: 1, dtype: int64

# want to Do same for multiple Colums so   
  
col =[1,5,6,7,8,9,13];  
for c in col :  
 new\_df = pd.concat( [new\_df , pd.get\_dummies(df\_census[c])] ,axis = 1)  
new\_df



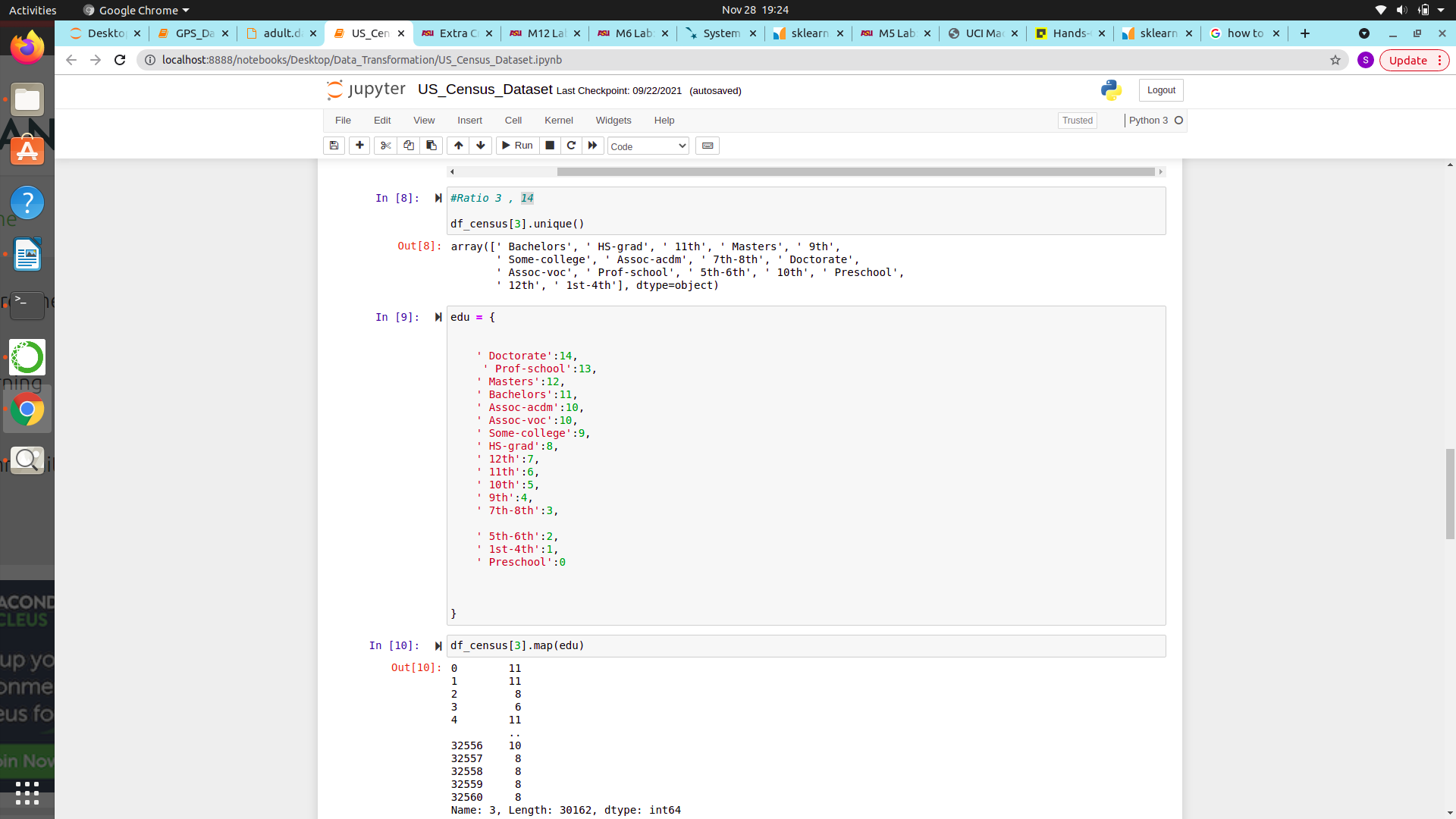
#Ratio 3 , 14  
  
df\_census[3].unique()

array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',  
 ' Some-college', ' Assoc-acdm', ' 7th-8th', ' Doctorate',  
 ' Assoc-voc', ' Prof-school', ' 5th-6th', ' 10th', ' Preschool',  
 ' 12th', ' 1st-4th'], dtype=object)

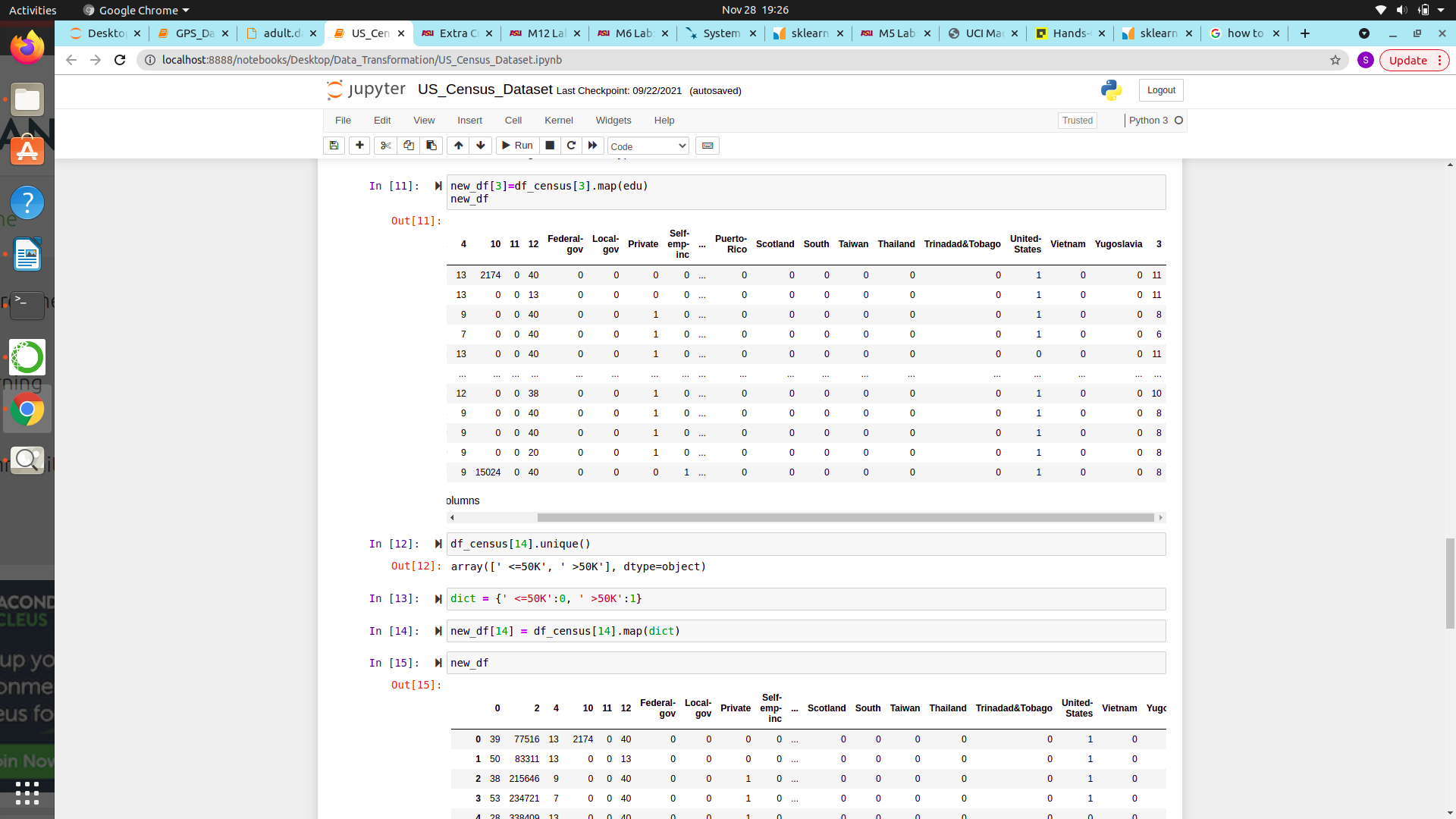
edu = {  
   
   
 ' Doctorate':14,  
 ' Prof-school':13,  
 ' Masters':12,  
 ' Bachelors':11,  
 ' Assoc-acdm':10,  
 ' Assoc-voc':10,  
 ' Some-college':9,  
 ' HS-grad':8,  
 ' 12th':7,  
 ' 11th':6,  
 ' 10th':5,  
 ' 9th':4,  
 ' 7th-8th':3,   
   
 ' 5th-6th':2,  
 ' 1st-4th':1,  
 ' Preschool':0  
   
   
   
}

df\_census[3].map(edu)

0 11  
1 11  
2 8  
3 6  
4 11  
 ..  
32556 10  
32557 8  
32558 8  
32559 8  
32560 8  
Name: 3, Length: 30162, dtype: int64



new\_df[3]=df\_census[3].map(edu)  
new\_df



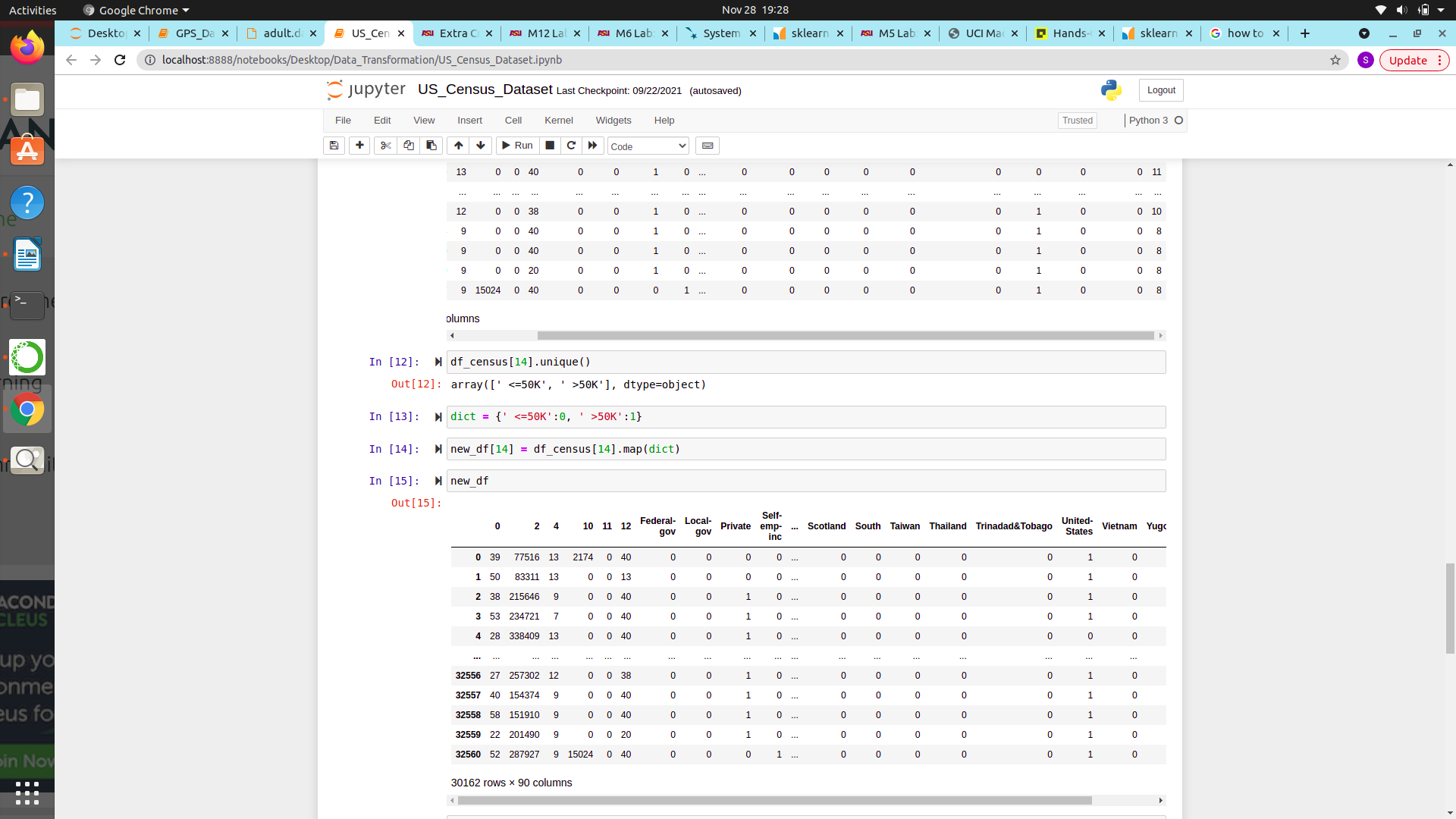
df\_census[14].unique()

array([' <=50K', ' >50K'], dtype=object)

dict = {' <=50K':0, ' >50K':1}

new\_df[14] = df\_census[14].map(dict)

new\_df



new\_df[14].value\_counts()

0 22654  
1 7508  
Name: 14, dtype: int64

X =new\_df.iloc[:,:-1].to\_numpy()  
y = new\_df.iloc[:,-1].to\_numpy()

X.shape

(30162, 89)

y.shape

(30162,)

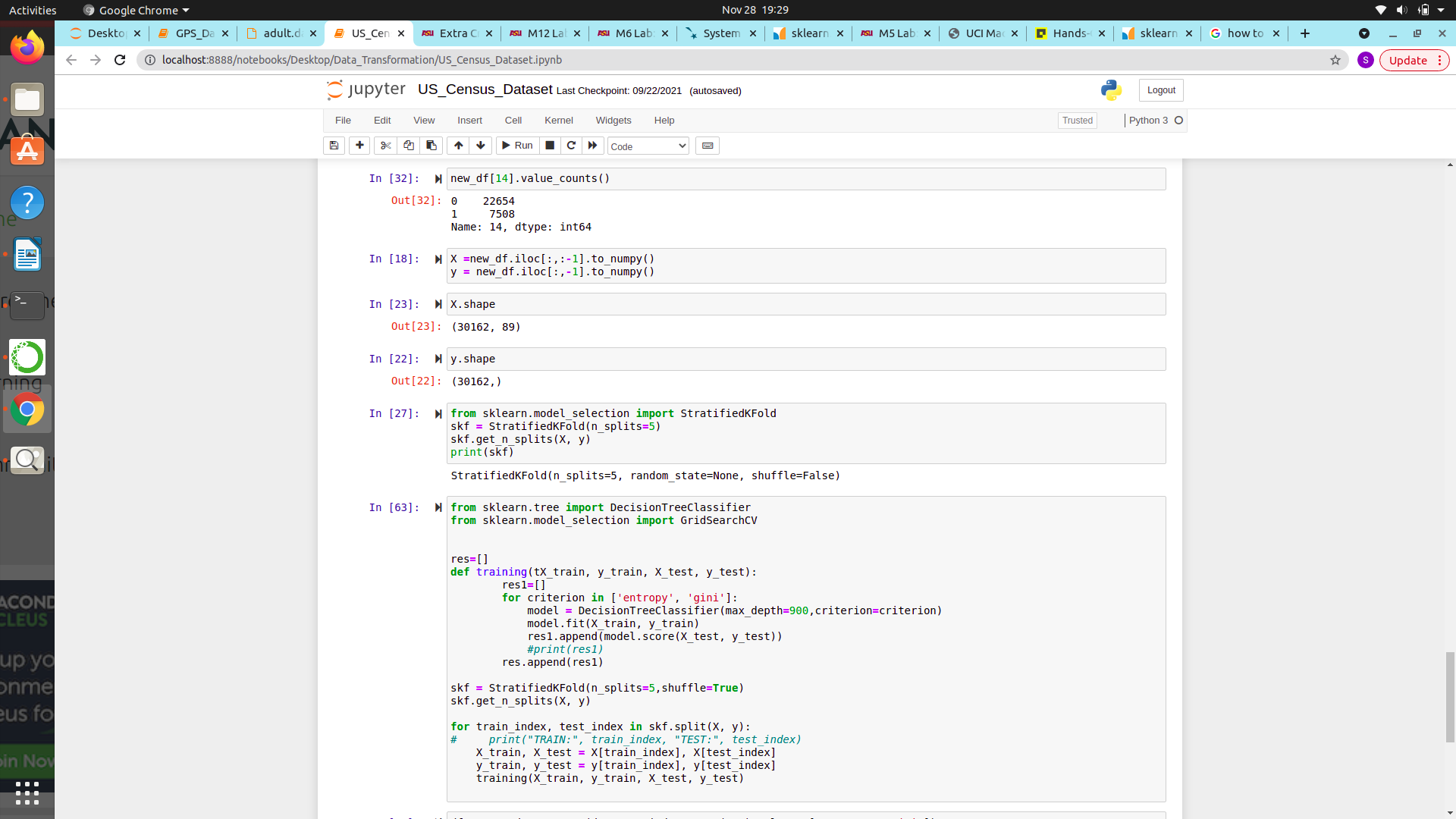
Randomly split the data into K equal folds. Set K= 5. For example, if the dataset contains 10,000 rows, randomly split it into 5 parts, each containing 2,000 rows. Use the [Startified K Fold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html) function for generating the random splits.

1. Create a for loop that passes over the 5 folds, each time it 4 folds for training a decision tree classifier and the remaining fold for testing and computing the classification accuracy. Notice that each iteration will use a different fold for testing.
   1. With each train-test 4-1 split, create a parameter grid that experiments with 'gini' & 'entropy' impurity measures.
   2. Make sure that the maximum tree depth is set to a value high enough for your dataset. You will not really fin-tune this parameter. Just set to a some high value. You can set it equal to 10 times the number of attributes (columns) in your dataset.
   3. Notice that each split-impurity measure will generate one accuracy value. That is, the total number of generated accuracies are 5 \* 2 = 10

from sklearn.model\_selection import StratifiedKFold  
skf = StratifiedKFold(n\_splits=5)  
skf.get\_n\_splits(X, y)  
print(skf)

StratifiedKFold(n\_splits=5, random\_state=None, shuffle=False)

from sklearn.tree import DecisionTreeClassifier  
from sklearn.model\_selection import GridSearchCV  
  
  
res=[]  
def training(tX\_train, y\_train, X\_test, y\_test):  
 res1=[]  
 for criterion in ['entropy', 'gini']:  
 model = DecisionTreeClassifier(max\_depth=900,criterion=criterion)  
 model.fit(X\_train, y\_train)   
 res1.append(model.score(X\_test, y\_test))  
 #print(res1)  
 res.append(res1)  
   
skf = StratifiedKFold(n\_splits=5,shuffle=True)  
skf.get\_n\_splits(X, y)  
  
for train\_index, test\_index in skf.split(X, y):  
# print("TRAIN:", train\_index, "TEST:", test\_index)  
 X\_train, X\_test = X[train\_index], X[test\_index]  
 y\_train, y\_test = y[train\_index], y[test\_index]  
 training(X\_train, y\_train, X\_test, y\_test)



1. Compute the overall accuracy for Gini by averaging over the 5 runs over the 5 folds that used Gini. Likewise compute the overall accuracy for Entropy.

df\_res = pd.DataFrame(data=res,index=range(1,6),columns=['entropy', 'gini'])  
df\_res

entropy gini  
1 0.813194 0.809879  
2 0.811205 0.813360  
3 0.815981 0.808687  
4 0.809516 0.809682  
5 0.804377 0.801227

Which parameter gives the best results?

Both Entropy and Gini shows approximately same results.

df\_res.mean(axis=0)

entropy 0.809661

gini 0.806147

dtype: float64

