Census-Income

February 20, 2019

1 Census-Income (KDD) Data Set

1.1 Data Set Information

This data set contains weighted census data extracted from the 1994 and 1995 Current Population Surveys conducted by the U.S. Census Bureau. The data contains 41 demographic and employment related variables.

One instance per line with comma delimited fields. There are 199523 instances in the data file and 99762 in the test file.

The data was split into train/test in approximately 2/3, 1/3 proportions using MineSet's MIndUtil mineset-to-mlc.

Source: https://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29

1.2 Attribute Information

More information detailing the meaning of the attributes can be found in the Census Bureau's documentation To make use of the data descriptions at this site, the following mappings to the Census Bureau's internal database column names will be needed:

age 'Age'
class of worker 'ClassOfWorker'
industry code 'IndustryCode'
occupation code 'OccupationCode'
education 'Education'
wage per hour 'WagePerHour'
enrolled in edu inst last wk 'EnrolledEducation'
marital status 'MaritalStatus'
major industry code 'MajorIndustryCode'
major occupation code 'MajorOccupationCode'
Race 'Race'
hispanic Origin 'HispanicOrigin'
sex 'Sex'
member of a labor union 'LabourUnion'
reason for unemployment 'ReasonUnemployed'
full or part time employment stat 'FullOrPartTime'
capital gains 'CapitalGains'
capital losses 'CapitalLosses'
divdends from stocks 'StockDividends' federal income tax liability
tax filer status 'TaxFilerStat'

region of previous residence ’PrevResidenceRegion’
state of previous residence ’PrevResidenceState’
detailed household and family stat ’HouseholdFamilyStatus’
detailed household summary in household ’HouseholdSummary’
instance weight ’InstanceWeight’
migration code-change in msa ’MigrationCodeChangeMSA’
migration code-change in reg ’MigrationCodeChangeReg’
migration code-move within reg ’MigrationCodeMoveWithinRegion’
live in this house 1 year ago ’LiveInHouse1Y’
migration prev res in sunbelt ’MigPrevResidenceSunbelt’
num persons worked for employer ’NumPersonsWorkedEmployer’
family members under 18 ’FamilyMembersU18’
country of birth father ’CountryBirthFather’
country of birth mother ’CountryBirthMother’
country of birth self ’CountryBirthSelf’
citizenship ’Citizenship’
own business or self employed ’OwnBusiness’
fill inc questionnaire for veteran’s admin ’QuestionnaireVeteran’
veterans benefits ’VeteranBenefits’
weeks worked in year ’WeeksWorkedInY’

Incomes ’Income’

Note that Incomes have been binned at the $50K level to present a binary classification problem, much like the original UCI/ADULT database. The goal field of this data, however, was drawn from the "total person income" field rather than the "adjusted gross income" and may, therefore, behave differently than the original ADULT goal field.

1.3 Problem Statement

The goal of this analysis is to try to find which factors can be used to predict an individual’s annual income (higher or lower than 50k USD), and then predict the income level based on these factors.

1.4 Summary

As part of the analysis, I will be going through the following steps:

1- Data Extraction:
Data is available in CSV files that can be downloaded at the source mentioned above.

2- Data Cleaning

3- Exploratory Data Analysis

4- Modeling:
a) Feature Selection: I will use Random Forest Classification for feature selection
b) Model Selection: I will use the selected features and apply Decision Trees and Logistic Regression. I will then proceed with the one with better rates.

5- Oversampling: Given that the data is skewed (only 8% are positive), I will also try oversampling using SMOTE to enhance the recall rates. I will then compare both results (with and without oversampling).

```
In [437]: import pandas as pd
          import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2 Read the data

```
In [438]: #Training and Test data is already split into two CSV's. Hence we will read them sepa

          columns = ['Age', 'ClassOfWorker', 'IndustryCode', 'OccupationCode','Education', 'Wag
                     'MaritalStatus', 'MajorIndustryCode', 'MajorOccupationCode', 'Race', 'Hispa
                     'ReasonUnemployed', 'FullOrPartTime', 'CapitalGains', 'CapitalLosses', 'Sto
                     'PrevResidenceRegion', 'PrevResidenceState', 'HouseholdFamilyStatus', 'Hous
                     'MigrationCodeChangeMSA','MigrationCodeChangeReg', 'MigrationCodeMoveWithin
                       'MigPrevResidenceSunbelt', 'NumPersonsWorkedEmployer', 'FamilyMembersU18
                     'CountryBirthMother', 'CountryBirthSelf', 'Citizenship', 'OwnBusiness', 'Qu
                     'VeteranBenefits','WeeksWorkedInY', 'Year','Income']

          df = pd.read_csv('Data/census-income.data', header=None)
          df.columns = columns
          df.drop(['InstanceWeight'], axis=1, inplace=True)

          df_test = pd.read_csv('Data/census-income.test', header=None)
          df_test.columns = columns
          df_test.drop(['InstanceWeight'], axis=1, inplace=True)

In [439]: df.shape

Out[439]: (199523, 41)

In [440]: df_test.shape

Out[440]: (99762, 41)

In [441]: df.head()

Out[441]:    Age                  ClassOfWorker  IndustryCode  OccupationCode  \
          0   73                Not in universe             0               0
          1   58   Self-employed-not incorporated             4              34
          2   18                Not in universe             0               0
          3    9                Not in universe             0               0
          4   10                Not in universe             0               0

                              Education  WagePerHour EnrolledEducation   MaritalStatus  \
          0         High school graduate            0   Not in universe         Widowed
          1   Some college but no degree            0   Not in universe        Divorced
          2                   10th grade            0       High school   Never married
          3                     Children            0   Not in universe   Never married
          4                     Children            0   Not in universe   Never married
```

```
               MajorIndustryCode                      MajorOccupationCode  \
0   Not in universe or children                           Not in universe
1                  Construction   Precision production craft & repair
2   Not in universe or children                           Not in universe
3   Not in universe or children                           Not in universe
4   Not in universe or children                           Not in universe

     ...     CountryBirthFather CountryBirthMother CountryBirthSelf  \
0    ...          United-States      United-States    United-States
1    ...          United-States      United-States    United-States
2    ...                Vietnam            Vietnam          Vietnam
3    ...          United-States      United-States    United-States
4    ...          United-States      United-States    United-States

                           Citizenship OwnBusiness QuestionnaireVeteran  \
0     Native- Born in the United States           0      Not in universe
1     Native- Born in the United States           0      Not in universe
2   Foreign born- Not a citizen of U S            0      Not in universe
3     Native- Born in the United States           0      Not in universe
4     Native- Born in the United States           0      Not in universe

   VeteranBenefits  WeeksWorkedInY  Year     Income
0                2               0    95   - 50000.
1                2              52    94   - 50000.
2                2               0    95   - 50000.
3                0               0    94   - 50000.
4                0               0    94   - 50000.

[5 rows x 41 columns]
```

In [442]:
```python
#Trim the strings in the data
for i in df.columns:
    if type(df[i][0]) == str:
        df[i] = df[i].apply(lambda x: str(x).strip())
        df_test[i] = df_test[i].apply(lambda x: str(x).strip())
```

In [443]:
```python
#Drop duplicates
df.drop_duplicates(inplace=True)
df_test.drop_duplicates(inplace=True)
```

3 Data Cleaning

In [444]:
```python
#Check missing data
for i in df.columns:
    print(df[i].unique())
```

```
[73 58 18  9 10 48 42 28 47 34  8 32 51 46 26 13 39 16 35 12 27 56 55  2  1
 37  4 63 25 81 11 30  7 66 84 52  5 36 72 61 41 90 49  6  0 33 57 50 24 17
```

```
 53 40 54 22 29 85 38 76 21 31 74 19 15  3 43 68 71 45 62 23 69 75 44 59 60
 64 65 70 67 78 20 14 83 86 89 77 79 82 80 87 88]
['Not in universe' 'Self-employed-not incorporated' 'Private'
 'Local government' 'Federal government' 'Self-employed-incorporated'
 'State government' 'Never worked' 'Without pay']
[ 0  4 40 34 43 37 24 39 12 35 45  3 19 29 32 48 33 23 44 36 31 30 41  5 11
  9 42  6 18 50  2  1 26 47 16 14 22 17  7  8 25 46 27 15 13 49 38 21 28 20
 51 10]
[ 0 34 10  3 40 26 37 31 12 36 41 22  2 35 25 23 42  8 19 29 27 16 33 13 18
  9 17 39 32 11 30 38 20  7 21 44 24 43 28  4  1  6 45 14  5 15 46]
['High school graduate' 'Some college but no degree' '10th grade'
 'Children' 'Bachelors degree(BA AB BS)'
 'Masters degree(MA MS MEng MEd MSW MBA)' 'Less than 1st grade'
 'Associates degree-academic program' '7th and 8th grade'
 '12th grade no diploma' 'Associates degree-occup /vocational'
 'Prof school degree (MD DDS DVM LLB JD)' '5th or 6th grade' '11th grade'
 'Doctorate degree(PhD EdD)' '9th grade' '1st 2nd 3rd or 4th grade']
[   0 1200  876 ..., 3156 2188 1092]
['Not in universe' 'High school' 'College or university']
['Widowed' 'Divorced' 'Never married' 'Married-civilian spouse present'
 'Separated' 'Married-spouse absent' 'Married-A F spouse present']
['Not in universe or children' 'Construction' 'Entertainment'
 'Finance insurance and real estate' 'Education'
 'Business and repair services' 'Manufacturing-nondurable goods'
 'Personal services except private HH' 'Manufacturing-durable goods'
 'Other professional services' 'Mining' 'Transportation' 'Wholesale trade'
 'Public administration' 'Retail trade' 'Social services'
 'Private household services' 'Utilities and sanitary services'
 'Communications' 'Hospital services' 'Medical except hospital'
 'Agriculture' 'Forestry and fisheries' 'Armed Forces']
['Not in universe' 'Precision production craft & repair'
 'Professional specialty' 'Executive admin and managerial'
 'Handlers equip cleaners etc' 'Adm support including clerical'
 'Machine operators assmblrs & inspctrs' 'Other service' 'Sales'
 'Private household services' 'Technicians and related support'
 'Transportation and material moving' 'Farming forestry and fishing'
 'Protective services' 'Armed Forces']
['White' 'Asian or Pacific Islander' 'Amer Indian Aleut or Eskimo' 'Black'
 'Other']
['All other' 'Do not know' 'Central or South American' 'Mexican (Mexicano)'
 'Mexican-American' 'Other Spanish' 'Puerto Rican' 'Cuban' 'Chicano' 'NA']
['Female' 'Male']
['Not in universe' 'No' 'Yes']
['Not in universe' 'Job loser - on layoff' 'Other job loser' 'New entrant'
 'Re-entrant' 'Job leaver']
['Not in labor force' 'Children or Armed Forces' 'Full-time schedules'
 'Unemployed full-time' 'Unemployed part- time'
 'PT for non-econ reasons usually FT' 'PT for econ reasons usually PT'
```

```
 'PT for econ reasons usually FT']
[    0  5178   991  2829  3464  5556  7298 15024  1831  3137 10605 20051
  2538  3908  2407  2050  3103  1086  7688  5013  4386  2414 99999 13550
  2174  4650  4064   914  2354  4787  2009  2597  1055  6097  2635  2105
  3325  6767  2228  2062  3942 27828  9562  2176  7262  2202  2290  1173
  8614  2329  2653  7430  3456  2580 10520  2907  3471  2885  9386  2993
  7896 14084  3818  1409   594  7978  1797  2964  4934  1848  4101  3418
  3432  2774  1424  6849  4687  6418  4508  3674  3411  2936  4416  2346
 10566  7443  5455  1151 25236  2463  1455  3781 14344  4865 11678  1471
  5060   114  4931  1506   401 25124 15020  2036  3273  6514  1111  2977
 41310 18481  6497  6723 15831  2098  1264 34095 22040  3887  2961  5721
  1090  6360  3800  2387  1731  6612  9472  4594  2601  1140  2227  8530]
[   0 1590 1977 1669 1719 2444 1421 1848 2205 2149 2001 1902 2090 1573 2415
 2377 1876 1602 1740 1974 2339 1887 1258 2597 2603 1408 1980 1721 1816 1340
 2788 2174 2042 1485 2489 2129 2457 2051 1762 2057 1672 2258  213 1651 2206
 3770 1628 1564 1668 1735 1579  625 4608 2559 2246 4356 1844 2002 2267 3175
 1380 2392 1092 1504 2238 2704 2467  810 1539 2824 1741 1870 1944 1825  419
 2547 1510  880 1617 1411 1648  323 2282 2352 3004 1755 1429  653 2163 2179
 1436 2722 3500 1640  974 1021 2754 1726 3900 2027  772 2231 1138 1594 2465
 2519 1956 1911 2472 2201 2080 3683  155]
[   0 6000  100 ...,  169 1055 7958]
['Nonfiler' 'Head of household' 'Joint both under 65' 'Single'
 'Joint both 65+' 'Joint one under 65 & one 65+']
['Not in universe' 'South' 'Northeast' 'Midwest' 'West' 'Abroad']
['Not in universe' 'Arkansas' 'Utah' 'Michigan' 'Minnesota' 'Alaska'
 'Kansas' 'Indiana' '?' 'Massachusetts' 'New Mexico' 'Nevada' 'Tennessee'
 'Colorado' 'Abroad' 'Kentucky' 'California' 'Arizona' 'North Carolina'
 'Connecticut' 'Florida' 'Vermont' 'Maryland' 'Oklahoma' 'Oregon' 'Ohio'
 'South Carolina' 'Texas' 'Montana' 'Wyoming' 'Georgia' 'Pennsylvania'
 'Iowa' 'New Hampshire' 'Missouri' 'Alabama' 'North Dakota' 'New Jersey'
 'Louisiana' 'West Virginia' 'Delaware' 'Illinois' 'Maine' 'Wisconsin'
 'New York' 'Idaho' 'District of Columbia' 'South Dakota' 'Nebraska'
 'Virginia' 'Mississippi']
['Other Rel 18+ ever marr not in subfamily' 'Householder'
 'Child 18+ never marr Not in a subfamily'
 'Child <18 never marr not in subfamily' 'Spouse of householder'
 'Secondary individual' 'Other Rel 18+ never marr not in subfamily'
 'Nonfamily householder' 'Grandchild <18 never marr not in subfamily'
 'Grandchild <18 never marr child of subfamily RP'
 'Child 18+ ever marr Not in a subfamily'
 'Child 18+ never marr RP of subfamily' 'Child 18+ spouse of subfamily RP'
 'Other Rel <18 never marr child of subfamily RP'
 'Child under 18 of RP of unrel subfamily'
 'Grandchild 18+ never marr not in subfamily'
 'Child 18+ ever marr RP of subfamily'
 'Other Rel 18+ ever marr RP of subfamily' 'RP of unrelated subfamily'
 'Other Rel 18+ spouse of subfamily RP'
 'Other Rel <18 never marr not in subfamily'
```

```
 'Other Rel <18 spouse of subfamily RP' 'In group quarters'
 'Grandchild 18+ spouse of subfamily RP'
 'Other Rel 18+ never marr RP of subfamily'
 'Child <18 never marr RP of subfamily'
 'Child <18 ever marr not in subfamily'
 'Other Rel <18 ever marr RP of subfamily'
 'Grandchild 18+ ever marr not in subfamily'
 'Child <18 spouse of subfamily RP' 'Spouse of RP of unrelated subfamily'
 'Other Rel <18 never married RP of subfamily'
 'Grandchild 18+ never marr RP of subfamily'
 'Grandchild 18+ ever marr RP of subfamily'
 'Child <18 ever marr RP of subfamily'
 'Other Rel <18 ever marr not in subfamily'
 'Grandchild <18 never marr RP of subfamily'
 'Grandchild <18 ever marr not in subfamily']
['Other relative of householder' 'Householder' 'Child 18 or older'
 'Child under 18 never married' 'Spouse of householder'
 'Nonrelative of householder' 'Group Quarters- Secondary individual'
 'Child under 18 ever married']
['?' 'MSA to MSA' 'Nonmover' 'NonMSA to nonMSA' 'Not in universe'
 'Not identifiable' 'Abroad to MSA' 'MSA to nonMSA' 'Abroad to nonMSA'
 'NonMSA to MSA']
['?' 'Same county' 'Nonmover' 'Different region'
 'Different county same state' 'Not in universe'
 'Different division same region' 'Abroad' 'Different state same division']
['?' 'Same county' 'Nonmover' 'Different state in South'
 'Different county same state' 'Not in universe'
 'Different state in Northeast' 'Abroad' 'Different state in Midwest'
 'Different state in West']
['Not in universe under 1 year old' 'No' 'Yes']
['?' 'Yes' 'Not in universe' 'No']
[0 1 6 4 5 3 2]
['Not in universe' 'Both parents present' 'Mother only present'
 'Neither parent present' 'Father only present']
['United-States' 'Vietnam' 'Philippines' '?' 'Columbia' 'Germany' 'Mexico'
 'Japan' 'Peru' 'Dominican-Republic' 'South Korea' 'Cuba' 'El-Salvador'
 'Canada' 'Scotland' 'Outlying-U S (Guam USVI etc)' 'Italy' 'Guatemala'
 'Ecuador' 'Puerto-Rico' 'Cambodia' 'China' 'Poland' 'Nicaragua' 'Taiwan'
 'England' 'Ireland' 'Hungary' 'Yugoslavia' 'Trinadad&Tobago' 'Jamaica'
 'Honduras' 'Portugal' 'Iran' 'France' 'India' 'Hong Kong' 'Haiti' 'Greece'
 'Holand-Netherlands' 'Thailand' 'Laos' 'Panama']
['United-States' 'Vietnam' '?' 'Columbia' 'Mexico' 'El-Salvador' 'Peru'
 'Puerto-Rico' 'Cuba' 'Philippines' 'Dominican-Republic' 'Germany'
 'England' 'Guatemala' 'Scotland' 'Portugal' 'Italy' 'Ecuador' 'Yugoslavia'
 'China' 'Poland' 'Hungary' 'Nicaragua' 'Taiwan' 'Ireland' 'Canada'
 'South Korea' 'Trinadad&Tobago' 'Jamaica' 'Honduras' 'Iran' 'France'
 'Cambodia' 'India' 'Hong Kong' 'Haiti' 'Japan' 'Greece'
 'Holand-Netherlands' 'Thailand' 'Panama' 'Laos'
```

```
 'Outlying-U S (Guam USVI etc)']
['United-States' 'Vietnam' '?' 'Columbia' 'Mexico' 'Peru' 'Cuba'
 'Philippines' 'Dominican-Republic' 'El-Salvador' 'Canada' 'Scotland'
 'Portugal' 'Guatemala' 'Ecuador' 'Germany' 'Outlying-U S (Guam USVI etc)'
 'Puerto-Rico' 'Italy' 'China' 'Poland' 'Nicaragua' 'Taiwan' 'England'
 'Ireland' 'South Korea' 'Trinadad&Tobago' 'Jamaica' 'Honduras' 'Iran'
 'Hungary' 'France' 'Cambodia' 'India' 'Hong Kong' 'Japan' 'Haiti'
 'Holand-Netherlands' 'Greece' 'Thailand' 'Panama' 'Yugoslavia' 'Laos']
['Native- Born in the United States' 'Foreign born- Not a citizen of U S'
 'Foreign born- U S citizen by naturalization'
 'Native- Born abroad of American Parent(s)'
 'Native- Born in Puerto Rico or U S Outlying']
[0 2 1]
['Not in universe' 'No' 'Yes']
[2 0 1]
[ 0 52 30 49 32 15 38 48  9 24 50 10 45 43  4 26 40 20  6 12 51  1  8 39 13
 16 34 14 36 44 22 41 46 28 23 35 25 17 11 37  5 42 29  2 21 19 47  3 27  7
 18 33 31]
[95 94]
['- 50000.' '50000+.']
```

In [445]:

```
#Some missing data is represented as '?', others are 'Not in universe'
df.replace("?", np.nan, inplace=True)
df.replace("Not in universe", np.nan, inplace=True)
df_test.replace("?", np.nan, inplace=True)
df_test.replace("Not in universe", np.nan, inplace=True)
```

In [446]:

```
df.isnull().sum()
```

Out[446]:

```
Age                         0
ClassOfWorker           54165
IndustryCode                0
OccupationCode              0
Education                   0
WagePerHour                 0
EnrolledEducation      142243
MaritalStatus               0
MajorIndustryCode           0
MajorOccupationCode     54548
Race                        0
HispanicOrigin              0
Sex                         0
LabourUnion            133840
ReasonUnemployed       146884
FullOrPartTime              0
CapitalGains                0
CapitalLosses               0
```

```
StockDividends                     0
TaxFilerStat                       0
PrevResidenceRegion           137492
PrevResidenceState            138190
HouseholdFamilyStatus              0
HouseholdSummary                   0
MigrationCodeChangeMSA         75288
MigrationCodeChangeReg         75288
MigrationCodeMoveWithinRegion  75288
LiveInHouse1Y                      0
MigPrevResidenceSunbelt       137492
NumPersonsWorkedEmployer           0
FamilyMembersU18              134959
CountryBirthFather              6383
CountryBirthMother              5810
CountryBirthSelf                3322
Citizenship                        0
OwnBusiness                        0
QuestionnaireVeteran          150931
VeteranBenefits                    0
WeeksWorkedInY                     0
Year                               0
Income                             0
dtype: int64
```

In [447]:
```
#Delete the columns with almost completely missing values
df.drop(['EnrolledEducation','LabourUnion', 'ReasonUnemployed', 'PrevResidenceRegion
         'MigPrevResidenceSunbelt','QuestionnaireVeteran','FamilyMembersU18','Migrat
         'MigrationCodeChangeReg','MigrationCodeMoveWithinRegion'], axis=1, inplace=

df_test.drop(['EnrolledEducation','LabourUnion', 'ReasonUnemployed', 'PrevResidenceR
         'MigPrevResidenceSunbelt','QuestionnaireVeteran','FamilyMembersU18','Migrat
         'MigrationCodeChangeReg','MigrationCodeMoveWithinRegion'], axis=1, inplace=
```

In [448]:
```
#Drop Rows with few missing column values
df.dropna(subset=['CountryBirthFather','CountryBirthMother','CountryBirthSelf'], inp
df_test.dropna(subset=['CountryBirthFather','CountryBirthMother','CountryBirthSelf']
```

In [449]:
```
df.isnull().sum()
```

Out[449]:
```
Age                      0
ClassOfWorker        49901
IndustryCode             0
OccupationCode           0
Education                0
WagePerHour              0
MaritalStatus            0
MajorIndustryCode        0
MajorOccupationCode  50258
```

```
Race                          0
HispanicOrigin                0
Sex                           0
FullOrPartTime                0
CapitalGains                  0
CapitalLosses                 0
StockDividends                0
TaxFilerStat                  0
HouseholdFamilyStatus         0
HouseholdSummary              0
LiveInHouse1Y                 0
NumPersonsWorkedEmployer      0
CountryBirthFather            0
CountryBirthMother            0
CountryBirthSelf              0
Citizenship                   0
OwnBusiness                   0
VeteranBenefits               0
WeeksWorkedInY                0
Year                          0
Income                        0
dtype: int64
```

In [450]:
```
#ClassOfWorker and MajorOccupationCode are missing for those who do not work. I will
df[['ClassOfWorker','MajorOccupationCode']] = df[['ClassOfWorker','MajorOccupationCo
df_test[['ClassOfWorker','MajorOccupationCode']] = df_test[['ClassOfWorker','MajorOcc
```

In [451]:
```
df['Income>50k'] = np.where(df['Income'] == '- 50000.', 0, 1)
df.drop('Income', axis=1, inplace=True)

df_test['Income>50k'] = np.where(df_test['Income'] == '- 50000.', 0, 1)
df_test.drop('Income', axis=1, inplace=True)
```

In [452]:
```
df.head(100)
```

Out[452]:

	Age	ClassOfWorker	IndustryCode	OccupationCode	\
0	73	NA	0	0	
1	58	Self-employed-not incorporated	4	34	
2	18	NA	0	0	
3	9	NA	0	0	
4	10	NA	0	0	
5	48	Private	40	10	
6	42	Private	34	3	
7	28	Private	4	40	
8	47	Local government	43	26	
9	34	Private	4	37	
10	8	NA	0	0	
12	51	Private	4	34	
13	46	Private	37	31	

14	26	Private	24	12
15	13	NA	0	0
16	47	Private	39	36
17	39	NA	0	0
18	16	NA	0	0
19	35	Private	12	41
20	12	NA	0	0
21	27	Self-employed-not incorporated	4	34
22	56	Private	35	22
23	46	Private	45	12
24	55	NA	0	0
25	2	NA	0	0
27	37	Private	3	34
28	4	NA	0	0
29	37	Private	4	2
30	63	Private	19	35
31	34	Federal government	29	25
..	...	...	...	...
74	51	Private	32	18
75	10	NA	0	0
76	48	State government	43	9
77	27	Private	33	16
78	24	NA	0	0
79	17	NA	0	0
80	58	Self-employed-not incorporated	35	17
81	2	NA	0	0
82	53	Private	4	34
83	33	Private	45	23
84	7	NA	0	0
85	40	Private	19	42
86	25	Private	33	29
88	54	NA	0	0
89	22	Private	34	26
90	29	NA	0	0
91	9	NA	0	0
93	40	Private	29	17
94	38	NA	0	0
95	7	NA	0	0
96	49	State government	29	26
98	7	NA	0	0
99	21	Private	33	3
100	52	NA	0	0
101	34	Local government	43	10
102	6	NA	0	0
103	31	Private	5	39
104	74	NA	0	0
105	0	NA	0	0
106	19	Private	39	32

	Education	WagePerHour \
0	High school graduate	0
1	Some college but no degree	0
2	10th grade	0
3	Children	0
4	Children	0
5	Some college but no degree	1200
6	Bachelors degree(BA AB BS)	0
7	High school graduate	0
8	Some college but no degree	876
9	Some college but no degree	0
10	Children	0
12	Some college but no degree	0
13	High school graduate	0
14	Bachelors degree(BA AB BS)	0
15	Children	0
16	Bachelors degree(BA AB BS)	0
17	10th grade	0
18	10th grade	0
19	High school graduate	0
20	Children	0
21	Some college but no degree	0
22	Some college but no degree	500
23	Masters degree(MA MS MEng MEd MSW MBA)	0
24	Some college but no degree	0
25	Children	0
27	Some college but no degree	0
28	Children	0
29	Bachelors degree(BA AB BS)	0
30	Less than 1st grade	0
31	Some college but no degree	0
..	...	...
74	Some college but no degree	0
75	Children	0
76	Some college but no degree	0
77	Some college but no degree	0
78	High school graduate	0
79	7th and 8th grade	0
80	Prof school degree (MD DDS DVM LLB JD)	0
81	Children	0
82	10th grade	0
83	Bachelors degree(BA AB BS)	0
84	Children	0
85	5th or 6th grade	0
86	High school graduate	0
88	High school graduate	0
89	High school graduate	0

```
90              Some college but no degree           0
91                                Children           0
93                    High school graduate           0
94                     Less than 1st grade           0
95                                Children           0
96                              11th grade           0
98                                Children           0
99              Some college but no degree           0
100                      7th and 8th grade           0
101             Bachelors degree(BA AB BS)           0
102                               Children           0
103                             11th grade           0
104                   High school graduate           0
105                               Children           0
106             Some college but no degree           0

                       MaritalStatus                MajorIndustryCode  \
0                            Widowed       Not in universe or children
1                           Divorced                      Construction
2                      Never married       Not in universe or children
3                      Never married       Not in universe or children
4                      Never married       Not in universe or children
5    Married-civilian spouse present                     Entertainment
6    Married-civilian spouse present   Finance insurance and real estate
7                      Never married                      Construction
8    Married-civilian spouse present                         Education
9    Married-civilian spouse present                      Construction
10                     Never married       Not in universe or children
12   Married-civilian spouse present                      Construction
13                          Divorced        Business and repair services
14                     Never married     Manufacturing-nondurable goods
15                     Never married       Not in universe or children
16                     Never married  Personal services except private HH
17   Married-civilian spouse present       Not in universe or children
18                     Never married       Not in universe or children
19   Married-civilian spouse present        Manufacturing-durable goods
20                     Never married       Not in universe or children
21   Married-civilian spouse present                      Construction
22   Married-civilian spouse present   Finance insurance and real estate
23   Married-civilian spouse present       Other professional services
24   Married-civilian spouse present       Not in universe or children
25                     Never married       Not in universe or children
27   Married-civilian spouse present                            Mining
28                     Never married       Not in universe or children
29                     Never married                      Construction
30   Married-civilian spouse present     Manufacturing-nondurable goods
31   Married-civilian spouse present                    Transportation
..                               ...                               ...
```

```
74   Married-civilian spouse present                      Wholesale trade
75                     Never married        Not in universe or children
76   Married-civilian spouse present                            Education
77                     Never married                         Retail trade
78   Married-civilian spouse present        Not in universe or children
79                     Never married        Not in universe or children
80   Married-civilian spouse present    Finance insurance and real estate
81                     Never married        Not in universe or children
82   Married-civilian spouse present                         Construction
83                     Never married        Other professional services
84                     Never married        Not in universe or children
85   Married-civilian spouse present     Manufacturing-nondurable goods
86                     Never married                         Retail trade
88   Married-civilian spouse present        Not in universe or children
89   Married-civilian spouse present    Finance insurance and real estate
90   Married-civilian spouse present        Not in universe or children
91                     Never married        Not in universe or children
93   Married-civilian spouse present                       Transportation
94                     Never married        Not in universe or children
95                     Never married        Not in universe or children
96   Married-civilian spouse present                       Transportation
98                     Never married        Not in universe or children
99                     Never married                         Retail trade
100  Married-civilian spouse present        Not in universe or children
101  Married-civilian spouse present                            Education
102                    Never married        Not in universe or children
103  Married-civilian spouse present        Manufacturing-durable goods
104  Married-civilian spouse present        Not in universe or children
105                    Never married        Not in universe or children
106                    Never married  Personal services except private HH

                     MajorOccupationCode                         Race  \
0                                     NA                        White
1     Precision production craft & repair                        White
2                                     NA   Asian or Pacific Islander
3                                     NA                        White
4                                     NA                        White
5                  Professional specialty  Amer Indian Aleut or Eskimo
6          Executive admin and managerial                        White
7            Handlers equip cleaners etc                        White
8          Adm support including clerical                        White
9    Machine operators assmblrs & inspctrs                        White
10                                    NA                        White
12    Precision production craft & repair                        White
13                          Other service                        White
14                 Professional specialty                        White
15                                    NA                        Black
16   Machine operators assmblrs & inspctrs                        White
```

```
17                                      NA                          White
18                                      NA                          White
19            Handlers equip cleaners etc                           White
20                                      NA                          Other
21     Precision production craft & repair                          White
22          Adm support including clerical                          White
23                  Professional specialty                          White
24                                      NA    Asian or Pacific Islander
25                                      NA                          White
27     Precision production craft & repair                          White
28                                      NA                          White
29          Executive admin and managerial                          White
30     Precision production craft & repair                          Other
31          Adm support including clerical                          White
..                                     ...                            ...
74                                   Sales                          White
75                                      NA                          White
76                  Professional specialty                          White
77                                   Sales                          White
78                                      NA                          Black
79                                      NA                          White
80                                   Sales                          White
81                                      NA                          Black
82     Precision production craft & repair                          White
83          Adm support including clerical                          Black
84                                      NA                          White
85            Handlers equip cleaners etc                           White
86                           Other service                          White
88                                      NA    Asian or Pacific Islander
89          Adm support including clerical                          White
90                                      NA                          Black
91                                      NA                          White
93                                   Sales                          White
94                                      NA                          White
95                                      NA                          White
96          Adm support including clerical                          White
98                                      NA                          White
99          Executive admin and managerial                          White
100                                     NA                          White
101                 Professional specialty                          White
102                                     NA                          White
103     Transportation and material moving                          White
104                                     NA                          White
105                                     NA                          White
106                          Other service                          White

       ...     NumPersonsWorkedEmployer CountryBirthFather CountryBirthMother  \
0      ...                            0      United-States      United-States
```

1	...	1	United-States	United-States
2	...	0	Vietnam	Vietnam
3	...	0	United-States	United-States
4	...	0	United-States	United-States
5	...	1	Philippines	United-States
6	...	6	United-States	United-States
7	...	4	United-States	United-States
8	...	5	United-States	United-States
9	...	6	United-States	United-States
10	...	0	United-States	United-States
12	...	3	United-States	United-States
13	...	6	Columbia	Columbia
14	...	6	United-States	United-States
15	...	0	United-States	United-States
16	...	6	Germany	United-States
17	...	0	Mexico	Mexico
18	...	0	United-States	El-Salvador
19	...	4	United-States	United-States
20	...	0	United-States	United-States
21	...	6	United-States	United-States
22	...	2	United-States	United-States
23	...	1	United-States	United-States
24	...	0	Japan	United-States
25	...	0	United-States	United-States
27	...	3	United-States	United-States
28	...	0	United-States	United-States
29	...	6	United-States	United-States
30	...	6	Mexico	Mexico
31	...	6	United-States	United-States
..	...	...	...	...
74	...	4	United-States	United-States
75	...	0	United-States	Mexico
76	...	6	United-States	United-States
77	...	6	United-States	United-States
78	...	6	United-States	United-States
79	...	0	United-States	United-States
80	...	1	United-States	United-States
81	...	0	United-States	United-States
82	...	1	United-States	United-States
83	...	6	United-States	United-States
84	...	0	Germany	United-States
85	...	3	Mexico	Mexico
86	...	3	United-States	United-States
88	...	0	Vietnam	Vietnam
89	...	3	United-States	United-States
90	...	6	United-States	United-States
91	...	0	United-States	United-States
93	...	3	United-States	United-States

```
94     ...                         0     United-States     United-States
95     ...                         0     United-States     United-States
96     ...                         6     United-States     United-States
98     ...                         0     United-States     United-States
99     ...                         6     United-States     United-States
100    ...                         0     United-States     United-States
101    ...                         6     United-States     United-States
102    ...                         0     United-States     United-States
103    ...                         2     United-States     United-States
104    ...                         0     United-States     United-States
105    ...                         0            Mexico            Mexico
106    ...                         3     United-States     United-States

    CountryBirthSelf                        Citizenship  OwnBusiness  \
0      United-States   Native- Born in the United States            0
1      United-States   Native- Born in the United States            0
2            Vietnam  Foreign born- Not a citizen of U S            0
3      United-States   Native- Born in the United States            0
4      United-States   Native- Born in the United States            0
5      United-States   Native- Born in the United States            2
6      United-States   Native- Born in the United States            0
7      United-States   Native- Born in the United States            0
8      United-States   Native- Born in the United States            0
9      United-States   Native- Born in the United States            0
10     United-States   Native- Born in the United States            0
12     United-States   Native- Born in the United States            0
13          Columbia  Foreign born- Not a citizen of U S            0
14     United-States   Native- Born in the United States            0
15     United-States   Native- Born in the United States            0
16     United-States   Native- Born in the United States            0
17            Mexico  Foreign born- Not a citizen of U S            0
18     United-States   Native- Born in the United States            0
19     United-States   Native- Born in the United States            0
20     United-States   Native- Born in the United States            0
21     United-States   Native- Born in the United States            1
22     United-States   Native- Born in the United States            2
23     United-States   Native- Born in the United States            0
24     United-States   Native- Born in the United States            0
25     United-States   Native- Born in the United States            0
27     United-States   Native- Born in the United States            0
28     United-States   Native- Born in the United States            0
29     United-States   Native- Born in the United States            0
30            Mexico  Foreign born- Not a citizen of U S            0
31     United-States   Native- Born in the United States            0
..               ...                                 ...          ...
74     United-States   Native- Born in the United States            0
75     United-States   Native- Born in the United States            0
76     United-States   Native- Born in the United States            2
```

77	United-States	Native- Born in the United States	0
78	United-States	Native- Born in the United States	0
79	United-States	Native- Born in the United States	0
80	United-States	Native- Born in the United States	0
81	United-States	Native- Born in the United States	0
82	United-States	Native- Born in the United States	0
83	United-States	Native- Born in the United States	0
84	United-States	Native- Born in the United States	0
85	Mexico	Foreign born- Not a citizen of U S	0
86	United-States	Native- Born in the United States	0
88	Vietnam	Foreign born- Not a citizen of U S	0
89	United-States	Native- Born in the United States	0
90	United-States	Native- Born in the United States	0
91	United-States	Native- Born in the United States	0
93	United-States	Native- Born in the United States	0
94	United-States	Native- Born in the United States	0
95	United-States	Native- Born in the United States	0
96	United-States	Native- Born in the United States	0
98	United-States	Native- Born in the United States	0
99	United-States	Native- Born in the United States	2
100	United-States	Native- Born in the United States	0
101	United-States	Native- Born in the United States	0
102	United-States	Native- Born in the United States	0
103	United-States	Native- Born in the United States	0
104	United-States	Native- Born in the United States	0
105	United-States	Native- Born in the United States	0
106	United-States	Native- Born in the United States	0

	VeteranBenefits	WeeksWorkedInY	Year	Income>50k
0	2	0	95	0
1	2	52	94	0
2	2	0	95	0
3	0	0	94	0
4	0	0	94	0
5	2	52	95	0
6	2	52	94	0
7	2	30	95	0
8	2	52	95	0
9	2	52	94	0
10	0	0	94	0
12	2	52	94	0
13	2	52	94	0
14	2	52	95	0
15	0	0	94	0
16	2	52	95	0
17	2	0	94	0
18	2	0	95	0
19	2	49	95	0

```
20                  0          0  94          0
21                  2         52  94          0
22                  2         32  95          0
23                  2         52  94          0
24                  2          0  94          0
25                  0          0  94          0
27                  2         52  95          0
28                  0          0  95          0
29                  2         52  94          0
30                  2         15  95          0
31                  2         52  95          0
..                ...        ...  ...        ...
74                  2         52  95          0
75                  0          0  95          0
76                  2         52  95          0
77                  2         52  95          0
78                  2         52  95          0
79                  2          0  94          0
80                  2         52  95          1
81                  0          0  94          0
82                  2         45  95          0
83                  2         52  95          0
84                  0          0  94          0
85                  2         43  94          0
86                  2          4  94          0
88                  2          0  95          1
89                  2         52  94          0
90                  2         52  95          0
91                  0          0  95          0
93                  2         52  94          0
94                  2          0  94          0
95                  0          0  94          0
96                  2         52  94          0
98                  0          0  95          0
99                  2         52  95          0
100                 2          0  95          0
101                 2         40  94          0
102                 0          0  94          0
103                 2         50  94          0
104                 2          0  94          0
105                 0          0  94          0
106                 2          4  95          0

[100 rows x 30 columns]
```

In [453]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 145005 entries, 0 to 199520
```

```
Data columns (total 30 columns):
Age                         145005 non-null int64
ClassOfWorker               145005 non-null object
IndustryCode                145005 non-null int64
OccupationCode              145005 non-null int64
Education                   145005 non-null object
WagePerHour                 145005 non-null int64
MaritalStatus               145005 non-null object
MajorIndustryCode           145005 non-null object
MajorOccupationCode         145005 non-null object
Race                        145005 non-null object
HispanicOrigin              145005 non-null object
Sex                         145005 non-null object
FullOrPartTime              145005 non-null object
CapitalGains                145005 non-null int64
CapitalLosses               145005 non-null int64
StockDividends              145005 non-null int64
TaxFilerStat                145005 non-null object
HouseholdFamilyStatus       145005 non-null object
HouseholdSummary            145005 non-null object
LiveInHouse1Y               145005 non-null object
NumPersonsWorkedEmployer    145005 non-null int64
CountryBirthFather          145005 non-null object
CountryBirthMother          145005 non-null object
CountryBirthSelf            145005 non-null object
Citizenship                 145005 non-null object
OwnBusiness                 145005 non-null int64
VeteranBenefits             145005 non-null int64
WeeksWorkedInY              145005 non-null int64
Year                        145005 non-null int64
Income>50k                  145005 non-null int32
dtypes: int32(1), int64(12), object(17)
memory usage: 33.7+ MB
```

In [454]: df.describe()

Out[454]:

```
                 Age   IndustryCode  OccupationCode    WagePerHour  \
count  145005.000000  145005.000000   145005.000000  145005.000000
mean       39.418468      20.221965       14.959374      73.836302
std        19.327772      18.201672       14.910480     313.499755
min         0.000000       0.000000        0.000000       0.000000
25%        25.000000       0.000000        0.000000       0.000000
50%        38.000000      24.000000       12.000000       0.000000
75%        52.000000      37.000000       29.000000       0.000000
max        90.000000      51.000000       46.000000    9999.000000

        CapitalGains  CapitalLosses  StockDividends  NumPersonsWorkedEmployer  \
```

```
count  145005.000000  145005.000000   145005.000000               145005.000000
mean      557.842136      48.684659      252.061688                    2.581587
std      5286.552736     308.852831     2260.621084                    2.402695
min         0.000000       0.000000        0.000000                    0.000000
25%         0.000000       0.000000        0.000000                    0.000000
50%         0.000000       0.000000        0.000000                    2.000000
75%         0.000000       0.000000        0.000000                    5.000000
max     99999.000000    4608.000000    99999.000000                    6.000000

          OwnBusiness  VeteranBenefits  WeeksWorkedInY           Year  \
count  145005.000000    145005.000000   145005.000000  145005.000000
mean        0.234282         1.826165       30.532313      94.488976
std         0.629074         0.551837       23.667317       0.499880
min         0.000000         0.000000        0.000000      94.000000
25%         0.000000         2.000000        0.000000      94.000000
50%         0.000000         2.000000       47.000000      94.000000
75%         0.000000         2.000000       52.000000      95.000000
max         2.000000         2.000000       52.000000      95.000000

            Income>50k
count  145005.000000
mean        0.080577
std         0.272185
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
```

Looks like there are some wrong values, such as WagePerHour (9999). I will look into those during EDA

4 Exploratory Data Analysis

In [455]:
```
df['Income>50k'].value_counts()
```

Out[455]:
```
0    133321
1     11684
Name: Income>50k, dtype: int64
```

In [456]:
```
df['Income>50k'].value_counts(normalize=True)
```

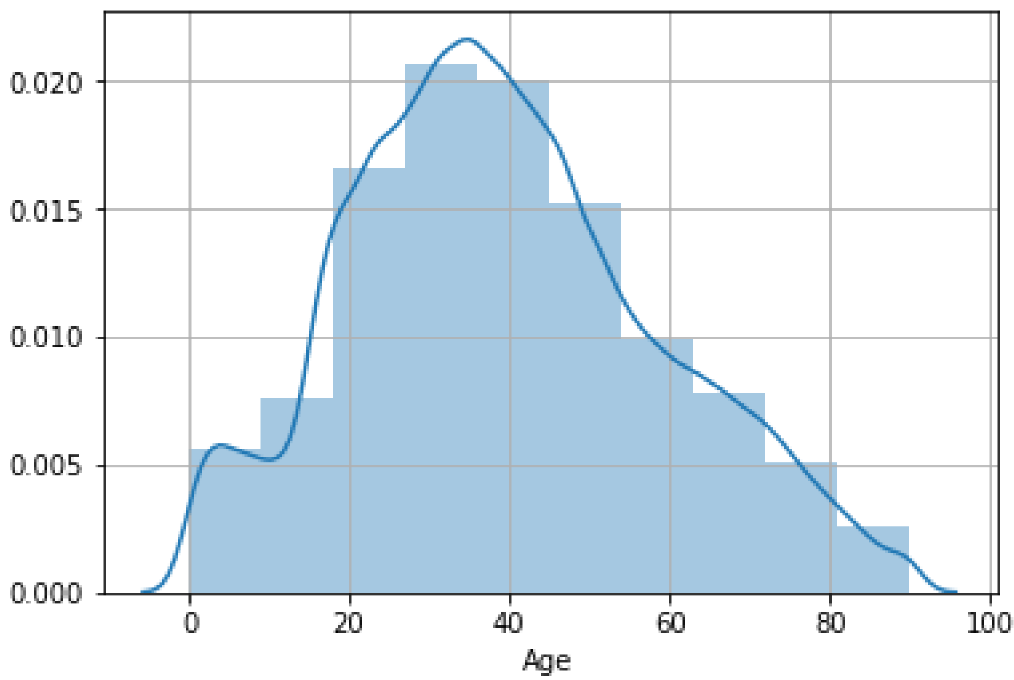
Out[456]:
```
0    0.919423
1    0.080577
Name: Income>50k, dtype: float64
```

Age

In [457]:
```
df['Age'].describe()
```

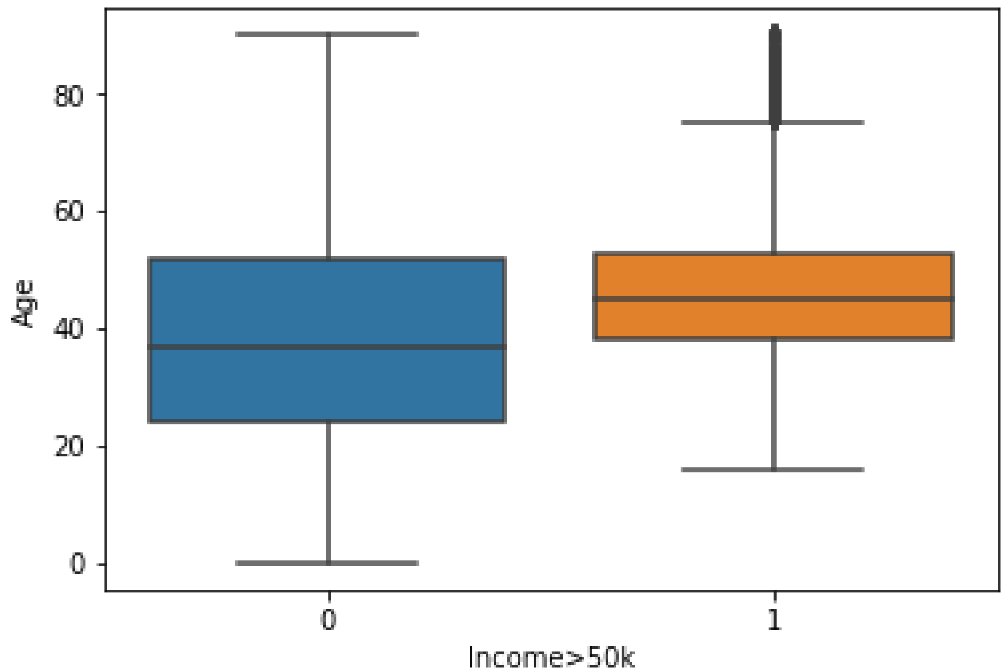
```
Out[457]: count    145005.000000
          mean         39.418468
          std          19.327772
          min           0.000000
          25%          25.000000
          50%          38.000000
          75%          52.000000
          max          90.000000
          Name: Age, dtype: float64
```

```
In [458]: sns.distplot(df['Age'], bins=10)
          plt.grid()
```

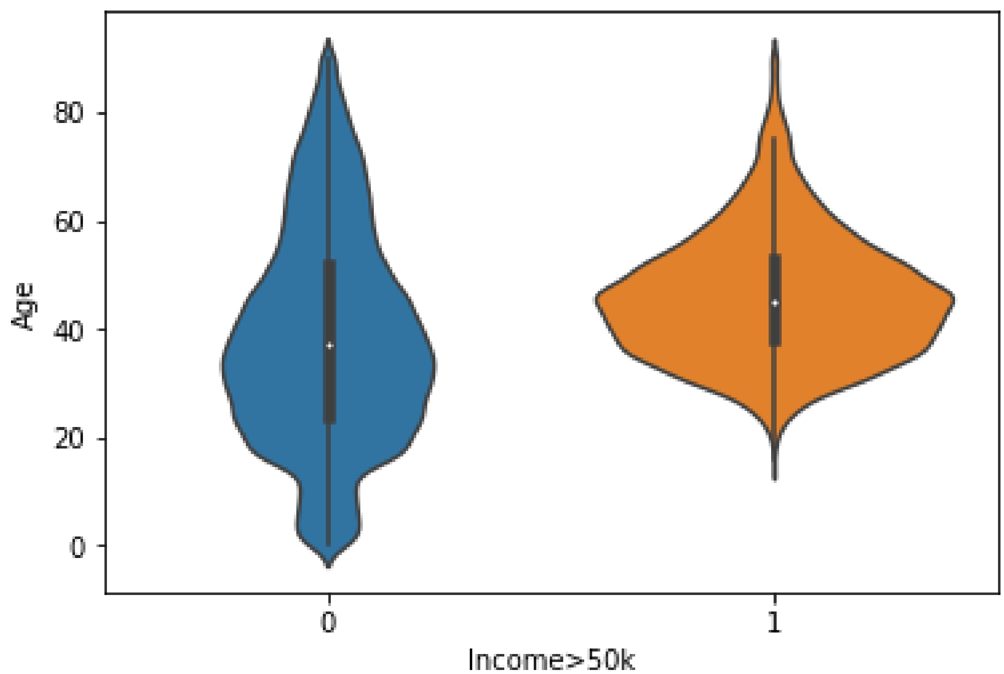



```
In [459]: sns.boxplot(x='Income>50k', y='Age',data=df)
```

```
Out[459]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc1a08160>
```

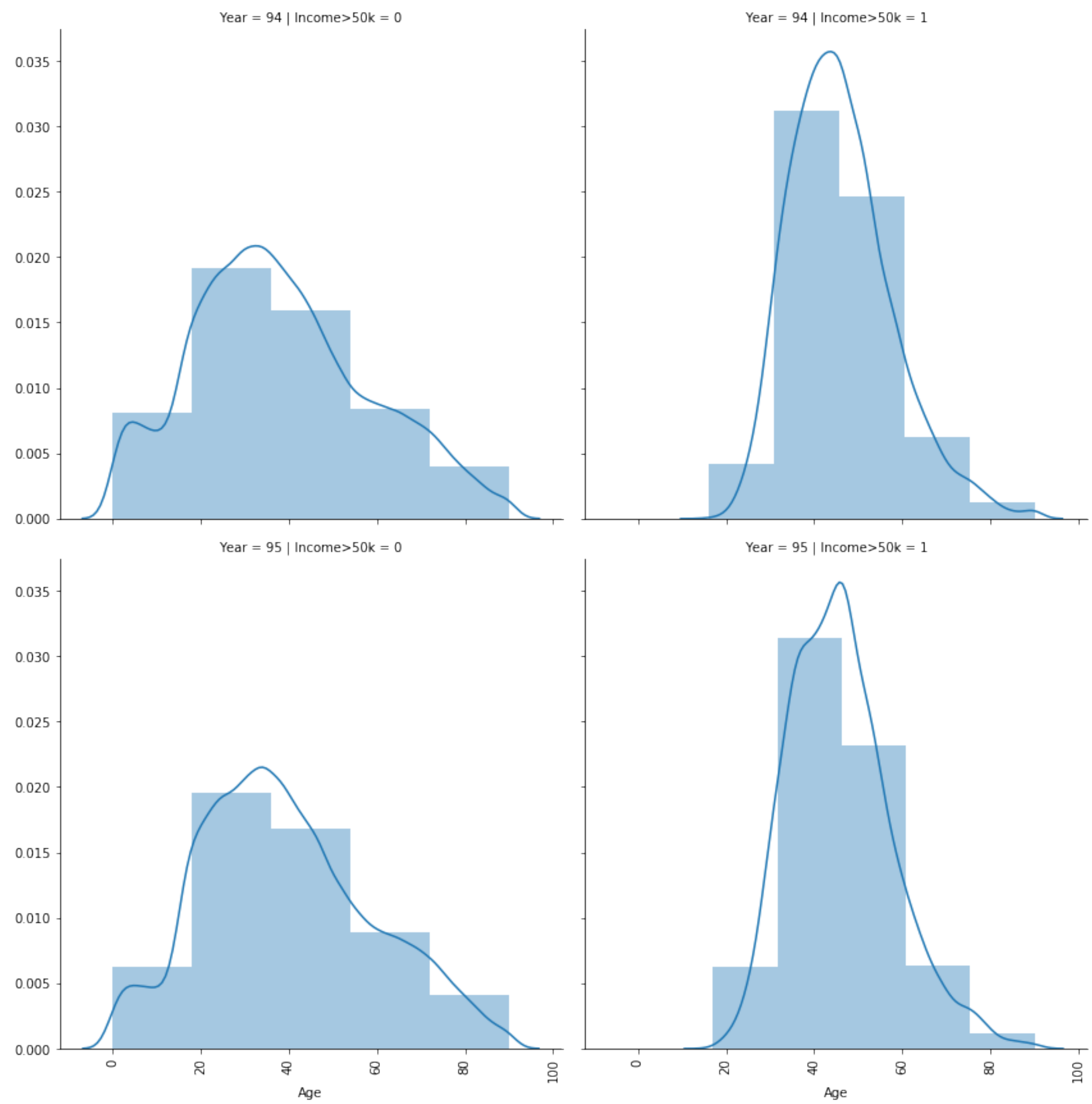



```
In [460]: sns.violinplot(x='Income>50k', y='Age',data=df)

Out[460]: <matplotlib.axes._subplots.AxesSubplot at 0x12da2f2cb00>
```



```
In [461]: plt.figure(figsize=(10,10))
          g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
          g.map(sns.distplot, 'Age', bins=5)
          for ax in g.axes.flat:
              labels = ax.get_xticklabels() # get x labels
              ax.set_xticklabels(labels, rotation=90) # set new labels
```

```
<Figure size 720x720 with 0 Axes>
```



ClassOfWorker

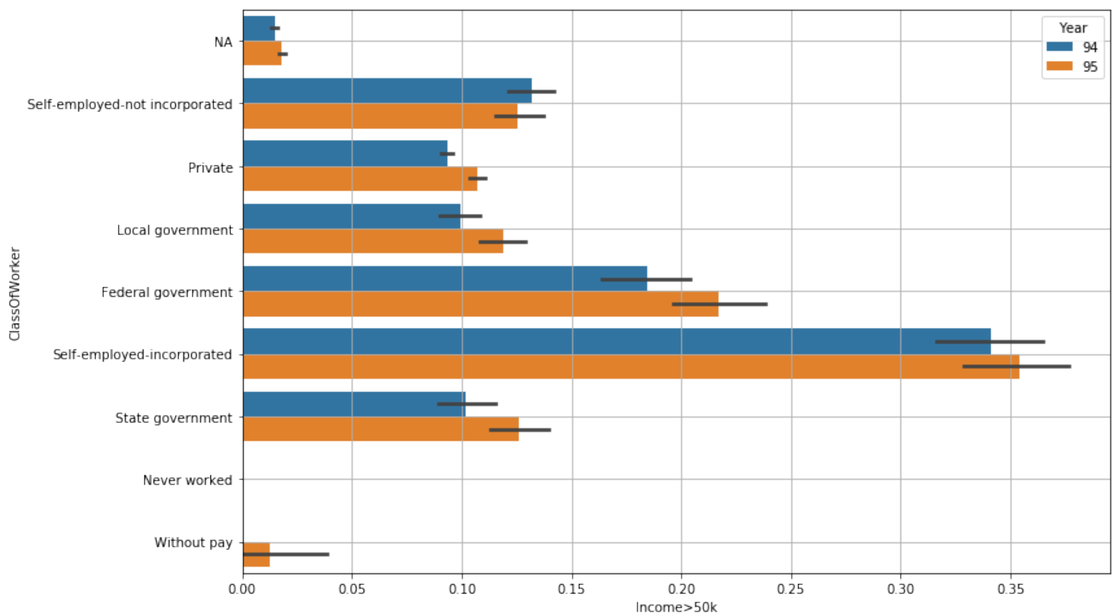
In [462]: df['ClassOfWorker'].describe()

```
Out[462]: count      145005
          unique          9
          top       Private
          freq        69208
          Name: ClassOfWorker, dtype: object
```

In [463]: df['ClassOfWorker'].value_counts()

```
Out[463]: Private                           69208
          NA                                49901
          Self-employed-not incorporated     7935
          Local government                   7512
          State government                   4090
          Self-employed-incorporated         3033
          Federal government                 2813
          Never worked                        357
          Without pay                         156
          Name: ClassOfWorker, dtype: int64
```

In [464]:
```
plt.figure(figsize=(12,8))
sns.barplot(y='ClassOfWorker', x='Income>50k', data=df, orient="h", hue='Year', dodge
plt.grid(True)
```



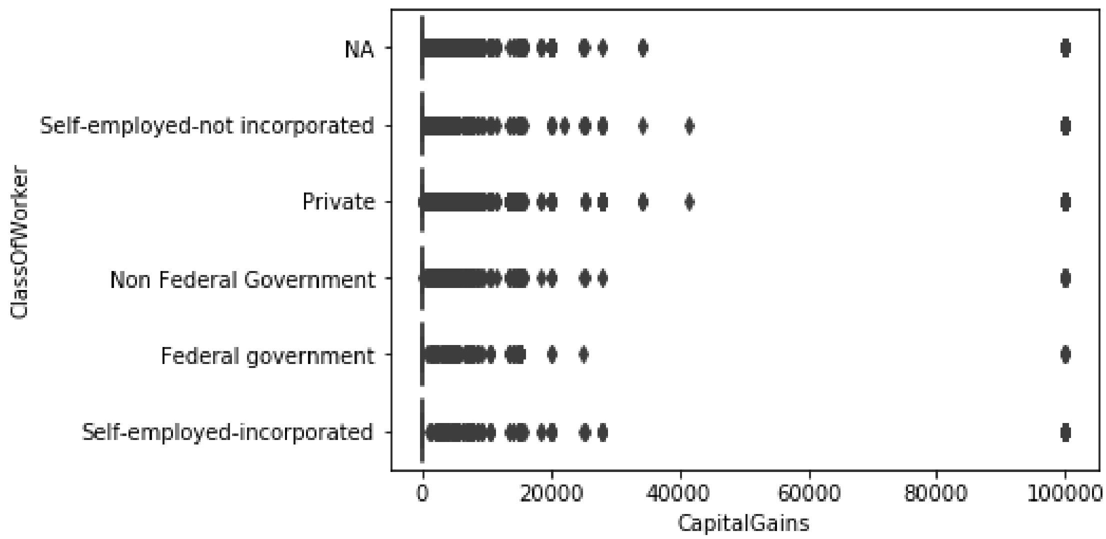
In [465]:
```
#Rename Never Worked and Without Pay to NA
#Join State and Local Govt
```

```
df['ClassOfWorker'] = np.where(df['ClassOfWorker'] == 'Never worked', 'NA', df['Class
df['ClassOfWorker'] = np.where(df['ClassOfWorker'] == 'Without pay', 'NA', df['Class
df['ClassOfWorker'] = np.where(df['ClassOfWorker'] == 'Local government', 'Non Federa
df['ClassOfWorker'] = np.where(df['ClassOfWorker'] == 'State government', 'Non Federa

df_test['ClassOfWorker'] = np.where(df_test['ClassOfWorker'] == 'Never worked', 'NA'
df_test['ClassOfWorker'] = np.where(df_test['ClassOfWorker'] == 'Without pay', 'NA',
df_test['ClassOfWorker'] = np.where(df_test['ClassOfWorker'] == 'Local government',
df_test['ClassOfWorker'] = np.where(df_test['ClassOfWorker'] == 'State government',

df['ClassOfWorker'].value_counts()
```

```
Out[465]: Private                            69208
          NA                                 50414
          Non Federal Government             11602
          Self-employed-not incorporated      7935
          Self-employed-incorporated          3033
          Federal government                  2813
          Name: ClassOfWorker, dtype: int64
```

```
In [466]: sns.boxplot(y='ClassOfWorker', x='CapitalGains', data=df, orient='h')
```

```
Out[466]: <matplotlib.axes._subplots.AxesSubplot at 0x12d9b2f4160>
```



Industry Code - Major Industry Code

```
In [467]: df['IndustryCode'].unique()
```

```
Out[467]: array([ 0,  4, 40, 34, 43, 37, 24, 39, 12, 35, 45,  3, 19, 29, 32, 48, 33,
                 23, 44, 36, 31, 30, 41,  5, 11,  9, 42,  6, 18, 50,  2,  1, 26, 47,
                 16, 14, 22, 17,  7,  8, 25, 46, 27, 15, 13, 49, 38, 21, 28, 51, 20,
                 10], dtype=int64)
```

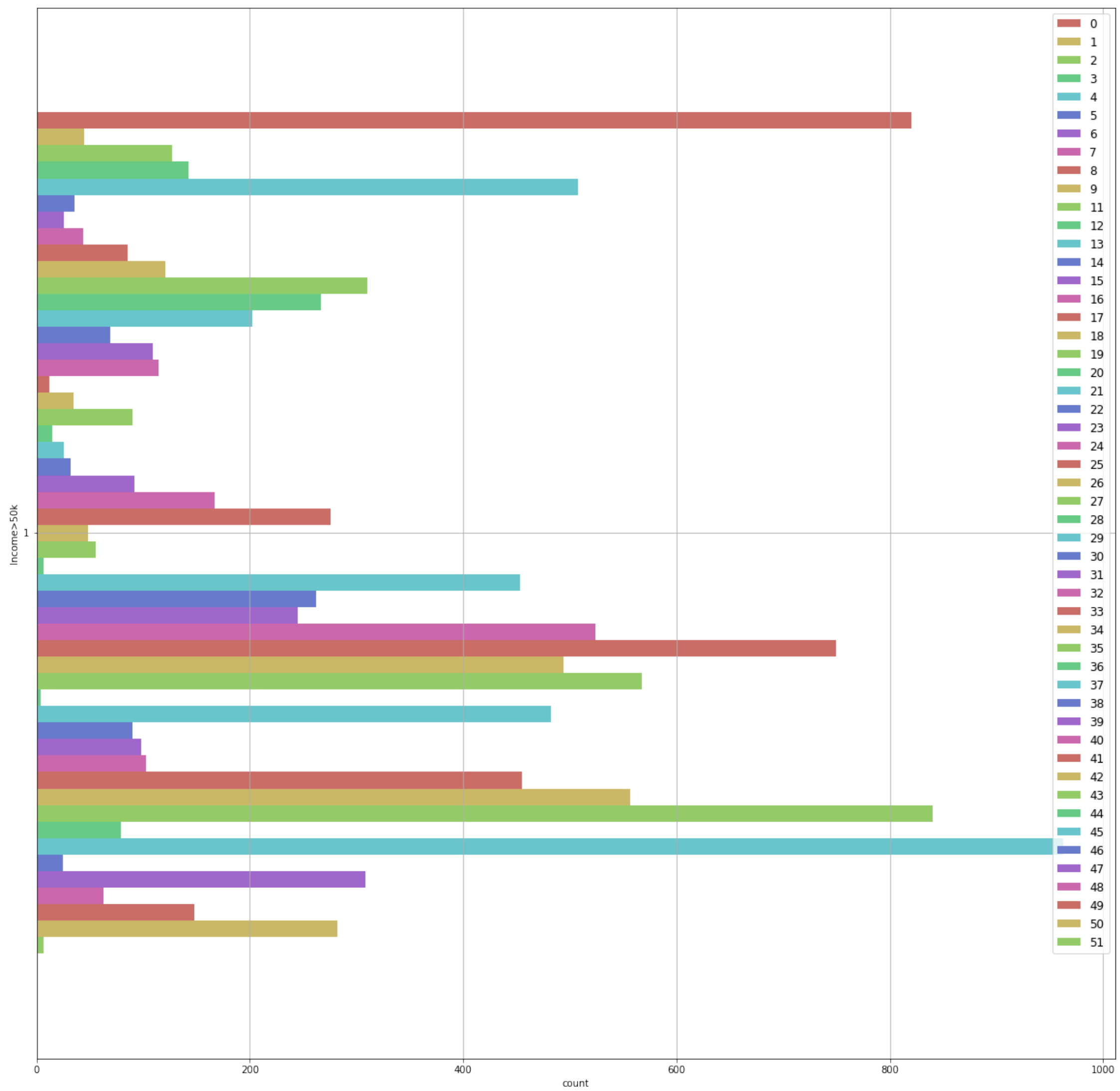
In [468]: df['IndustryCode'].value_counts()

```
Out[468]: 0      50258
          33     16241
          43      7922
          4       5771
          42      4493
          45      4226
          29      4041
          37      3817
          41      3808
          32      3458
          35      3254
          39      2797
          34      2629
          44      2457
          2       2072
          11      1695
          50      1656
          47      1609
          40      1589
          38      1544
          24      1444
          19      1319
          12      1302
          31      1143
          30      1141
          25      1023
          9        967
          22       917
          36       892
          13       874
          1        808
          48       626
          27       609
          49       587
          3        557
          5        548
          21       540
          6        539
          8        535
          16       516
          23       516
          18       461
          15       437
          7        412
          14       289
          46       183
```

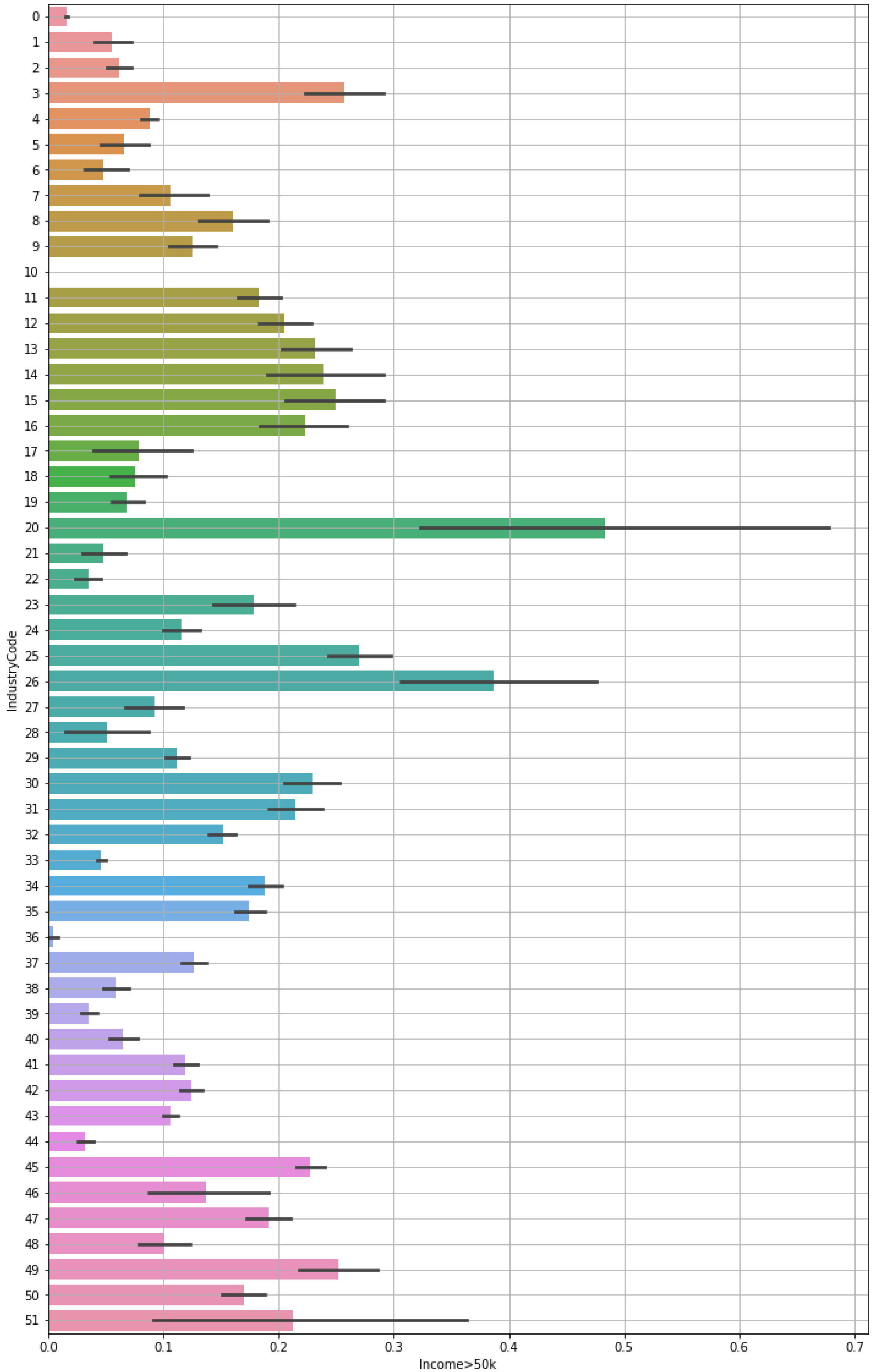
```
17      153
28      138
26      124
51       33
20       31
10        4
Name: IndustryCode, dtype: int64
```

In [469]:
```
plt.figure(figsize=(20,20))
sns.countplot(y='Income>50k', hue='IndustryCode', data=df[df['Income>50k']==1], palet
plt.grid()
plt.legend(fontsize='large')
```

Out[469]: <matplotlib.legend.Legend at 0x12d9b24d630>



In [470]:
```
plt.figure(figsize=(12,20))
sns.barplot(y='IndustryCode', x='Income>50k', data=df, orient="h", dodge=True)
plt.grid(True)
```

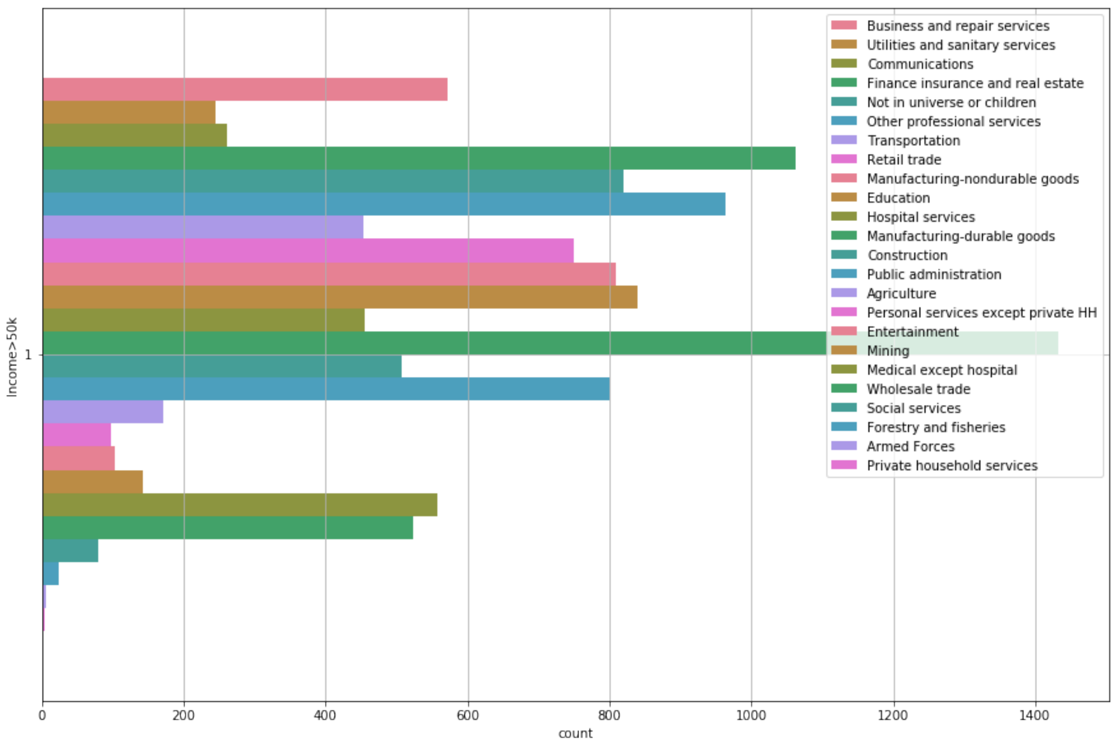

IndustryCode
Income>50k
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

In [471]: df['MajorIndustryCode'].value_counts()

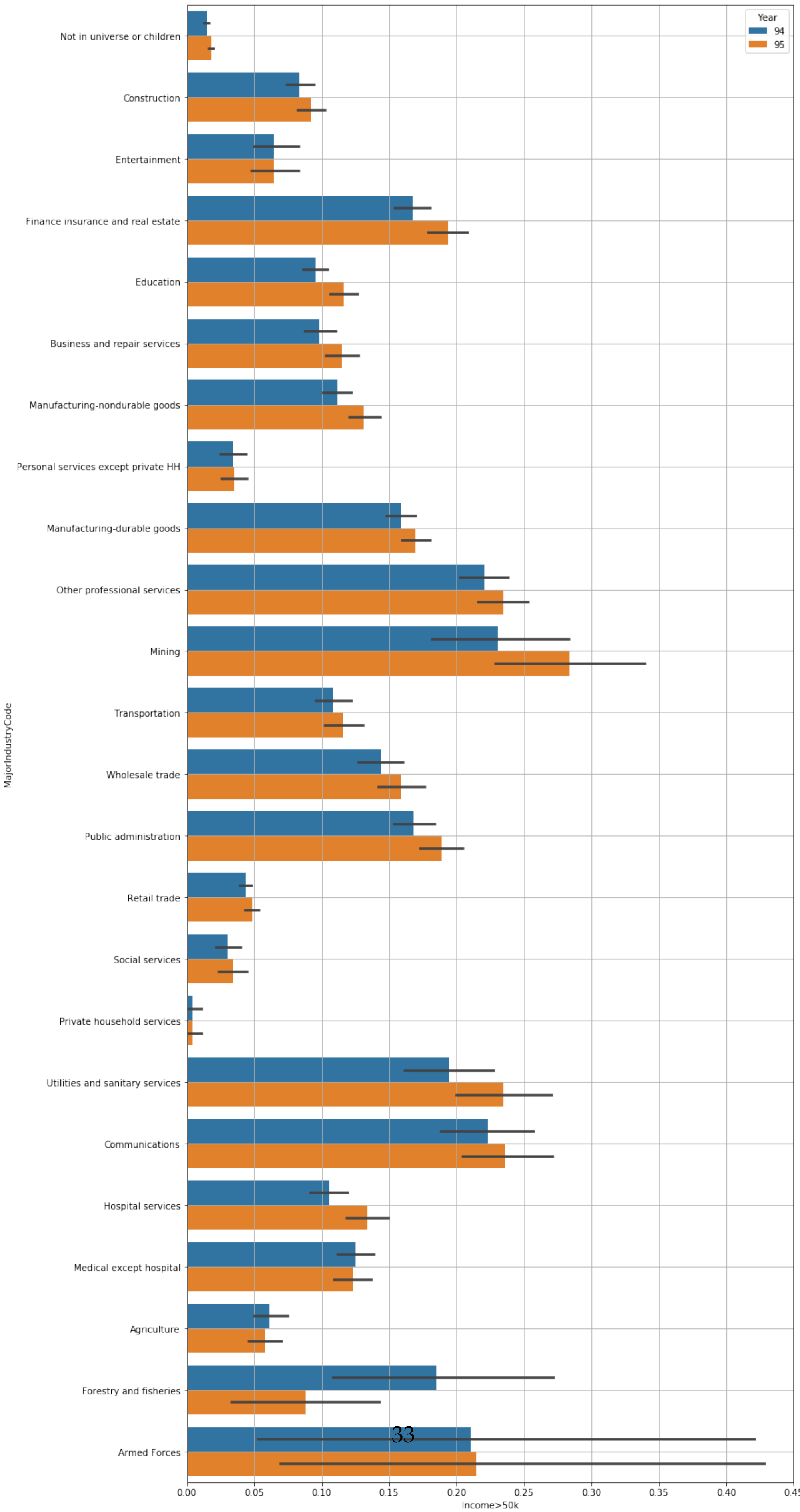
```
Out[471]: Not in universe or children            50258
          Retail trade                           16241
          Manufacturing-durable goods             8732
          Education                               7922
          Manufacturing-nondurable goods          6661
          Finance insurance and real estate       5883
          Construction                            5771
          Business and repair services            5361
          Medical except hospital                 4493
          Public administration                   4478
          Other professional services             4226
          Transportation                          4041
          Hospital services                       3808
          Wholesale trade                         3458
          Agriculture                             2880
          Personal services except private HH     2797
          Social services                         2457
          Entertainment                           1589
          Utilities and sanitary services         1143
          Communications                          1141
          Private household services               892
          Mining                                   557
          Forestry and fisheries                   183
          Armed Forces                              33
          Name: MajorIndustryCode, dtype: int64
```

In [472]:
```
plt.figure(figsize=(15,10))
sns.countplot(y='Income>50k', hue='MajorIndustryCode', data=df[df['Income>50k']==1],p
plt.grid()
plt.legend(fontsize='medium')
```

Out[472]: <matplotlib.legend.Legend at 0x12d9a1b8358>




```
In [473]: plt.figure(figsize=(12,30))
          sns.barplot(y='MajorIndustryCode', x='Income>50k', data=df, orient="h", hue='Year')
          plt.grid(True)
```


Year
94
95
Not in universe or children
Construction
Entertainment
Finance insurance and real estate
Education
Business and repair services
Manufacturing-nondurable goods
Personal services except private HH
Manufacturing-durable goods
Other professional services
Mining
Transportation
Wholesale trade
Public administration
Retail trade
Social services
Private household services
Utilities and sanitary services
Communications
Hospital services
Medical except hospital
Agriculture
Forestry and fisheries
Armed Forces
MajorIndustryCode
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
Income>50k

In [474]: df.groupby('MajorIndustryCode')['IndustryCode'].unique()

```
Out[474]: MajorIndustryCode
Agriculture                                                              [2
Armed Forces
Business and repair services                                           [37,
Communications
Construction
Education
Entertainment
Finance insurance and real estate                                      [34,
Forestry and fisheries
Hospital services
Manufacturing-durable goods         [12, 5, 11, 9, 6, 18, 16, 14, 17, 7, 8, 15, 1
Manufacturing-nondurable goods               [24, 19, 23, 26, 22, 25, 27, 21, 28,
Medical except hospital
Mining
Not in universe or children
Other professional services
Personal services except private HH
Private household services
Public administration                                              [48, 50, 47,
Retail trade
Social services
Transportation
Utilities and sanitary services
Wholesale trade
Name: IndustryCode, dtype: object
```

In [475]: df.groupby(['IndustryCode','MajorIndustryCode'])['Income>50k'].sum().sort_values()

```
Out[475]: IndustryCode  MajorIndustryCode
10            Manufacturing-durable goods          0
36            Private household services           4
28            Manufacturing-nondurable goods       7
51            Armed Forces                         7
17            Manufacturing-durable goods         12
20            Manufacturing-nondurable goods      15
46            Forestry and fisheries              25
6             Manufacturing-durable goods         26
21            Manufacturing-nondurable goods      26
22            Manufacturing-nondurable goods      32
18            Manufacturing-durable goods         35
5             Manufacturing-durable goods         36
7             Manufacturing-durable goods         44
1             Agriculture                         45
```

```
26          Manufacturing-nondurable goods         48
27          Manufacturing-nondurable goods         56
48          Public administration                  63
14          Manufacturing-durable goods            69
44          Social services                        79
8           Manufacturing-durable goods            86
19          Manufacturing-nondurable goods         90
38          Business and repair services           90
23          Manufacturing-nondurable goods         92
39          Personal services except private HH    98
40          Entertainment                         103
15          Manufacturing-durable goods           109
16          Manufacturing-durable goods           115
9           Manufacturing-durable goods           121
2           Agriculture                           127
3           Mining                                143
49          Public administration                 148
24          Manufacturing-nondurable goods        167
13          Manufacturing-durable goods           202
31          Utilities and sanitary services       245
30          Communications                        262
12          Manufacturing-durable goods           267
25          Manufacturing-nondurable goods        276
50          Public administration                 282
47          Public administration                 308
11          Manufacturing-durable goods           310
29          Transportation                        453
41          Hospital services                     455
37          Business and repair services          482
34          Finance insurance and real estate     494
4           Construction                          508
32          Wholesale trade                       524
42          Medical except hospital               557
35          Finance insurance and real estate     568
33          Retail trade                          750
0           Not in universe or children           820
43          Education                             840
45          Other professional services           963
Name: Income>50k, dtype: int32
```

In [476]:
```
#Industry Code is too scattered, and the values are too small for each Code. I will
df.drop('IndustryCode', axis=1, inplace=True)
df_test.drop('IndustryCode', axis=1, inplace=True)
```

OccupationCode and MajorOccupationCode

In [477]:
```
df['OccupationCode'].describe()
```

Out[477]:
```
count    145005.000000
mean         14.959374
```

```
std            14.910480
min             0.000000
25%             0.000000
50%            12.000000
75%            29.000000
max            46.000000
Name: OccupationCode, dtype: float64
```

In [478]:
```python
df['OccupationCode'].value_counts()
```

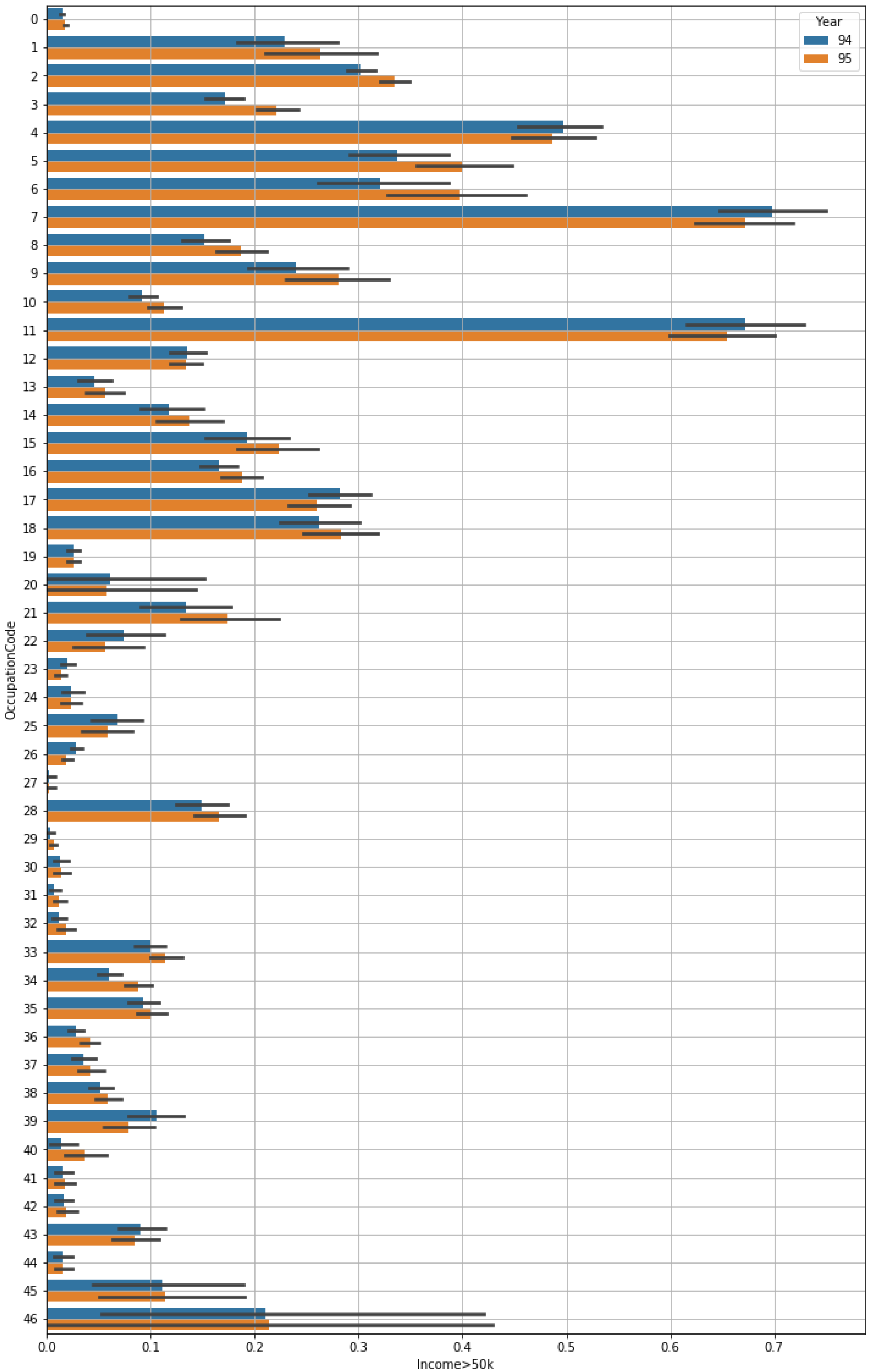
Out[478]:
```
0      50258
2       8368
26      7613
19      5130
29      4904
36      4011
34      3855
10      3531
23      3283
16      3272
33      3200
12      3171
35      3064
3       3044
38      2894
31      2601
32      2295
37      2169
8       2063
42      1873
30      1811
24      1784
17      1700
28      1608
44      1553
41      1525
43      1273
4       1269
13      1230
18      1035
39      1001
14       909
5        806
15       777
25       745
27       733
9        681
7        663
```

```
40      604
11      591
1       519
21      518
6       404
22      397
45      169
20       68
46       33
Name: OccupationCode, dtype: int64
```

```
In [479]: df['MajorOccupationCode'].unique()
```

```
Out[479]: array(['NA', 'Precision production craft & repair',
       'Professional specialty', 'Executive admin and managerial',
       'Handlers equip cleaners etc', 'Adm support including clerical',
       'Machine operators assmblrs & inspctrs', 'Other service', 'Sales',
       'Private household services', 'Technicians and related support',
       'Transportation and material moving',
       'Farming forestry and fishing', 'Protective services',
       'Armed Forces'], dtype=object)
```

```
In [480]: plt.figure(figsize=(12,20))
          sns.barplot(y='OccupationCode', x='Income>50k', data=df, orient="h", hue='Year', dodg
          plt.grid(True)
```

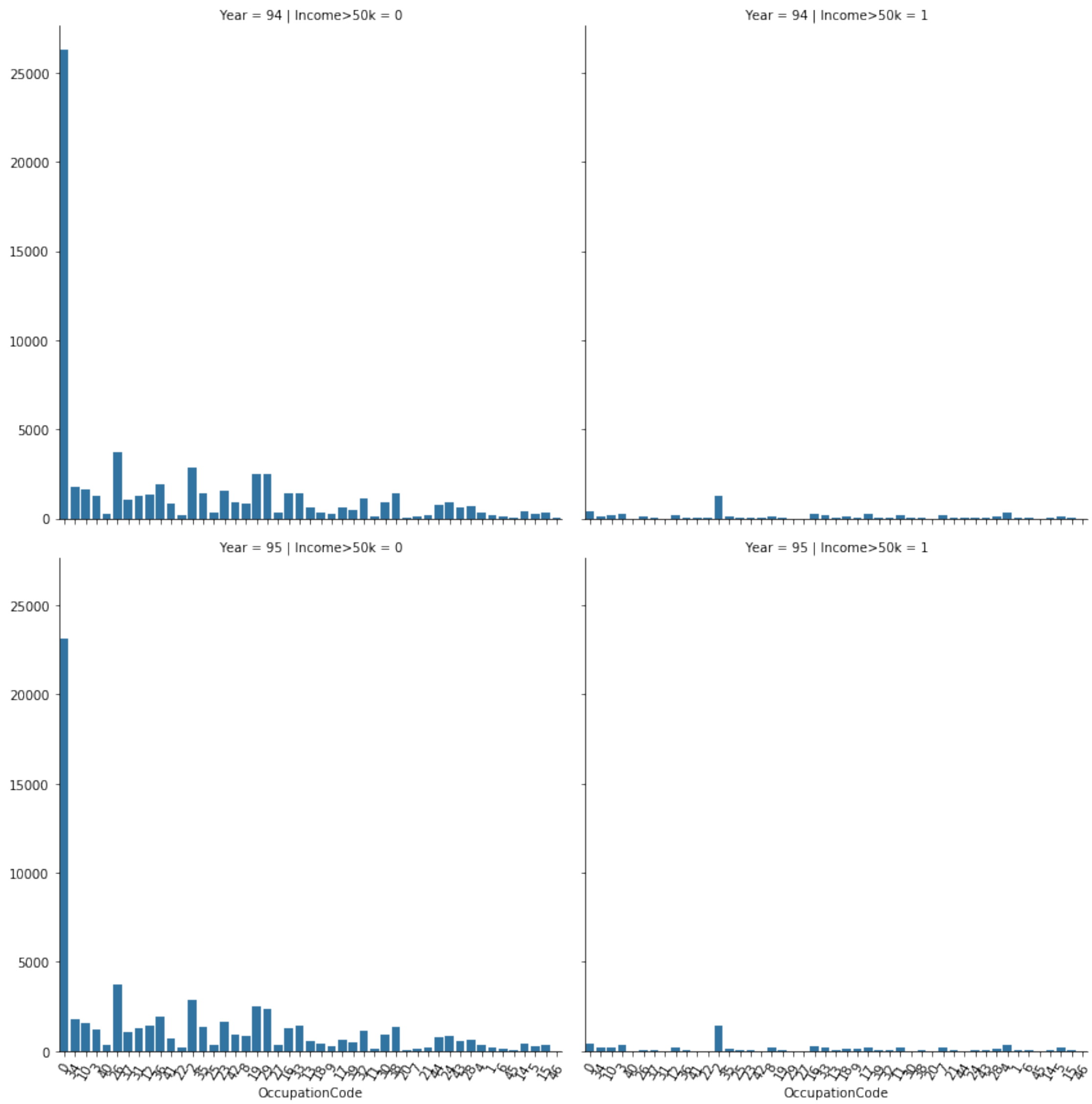


Year
94
95
OccupationCode
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Income>50k

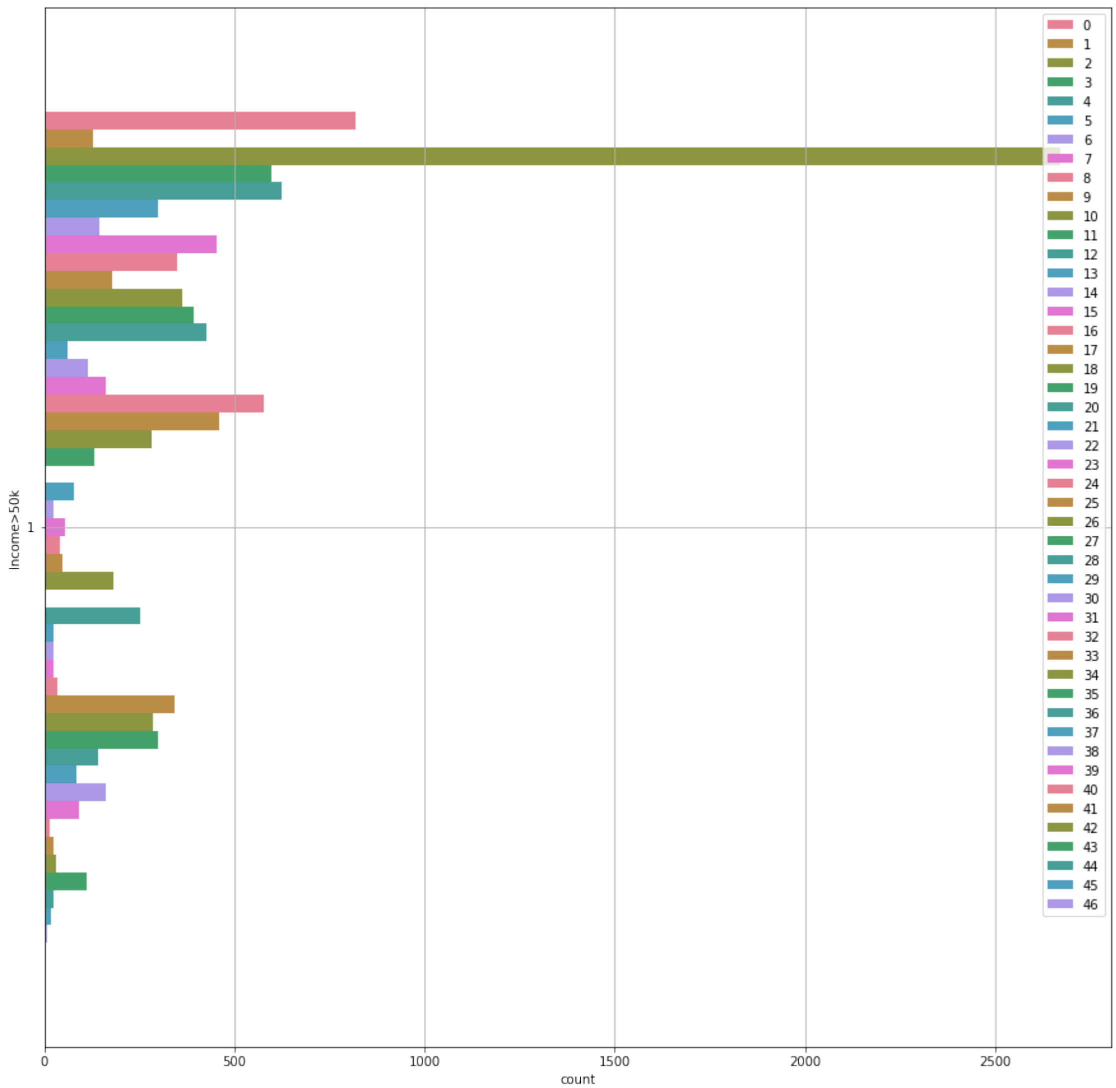
In [481]:
```
df.groupby('MajorOccupationCode')['OccupationCode'].unique()
```

Out[481]:
```
MajorOccupationCode
Adm support including clerical                          [26, 22, 25, 23, 21, 24]
Armed Forces                                                                [46]
Executive admin and managerial                                         [3, 2, 1]
Farming forestry and fishing                                        [44, 43, 45]
Handlers equip cleaners etc                                         [40, 41, 42]
Machine operators assmblrs & inspctrs                                   [37, 36]
NA                                                                           [0]
Other service                                                   [31, 29, 32, 30]
Precision production craft & repair                                 [34, 35, 33]
Private household services                                                  [27]
Professional specialty                                [10, 12, 8, 9, 11, 7, 4, 6, 5]
Protective services                                                         [28]
Sales                                                       [19, 16, 18, 17, 20]
Technicians and related support                                     [13, 14, 15]
Transportation and material moving                                      [39, 38]
Name: OccupationCode, dtype: object
```

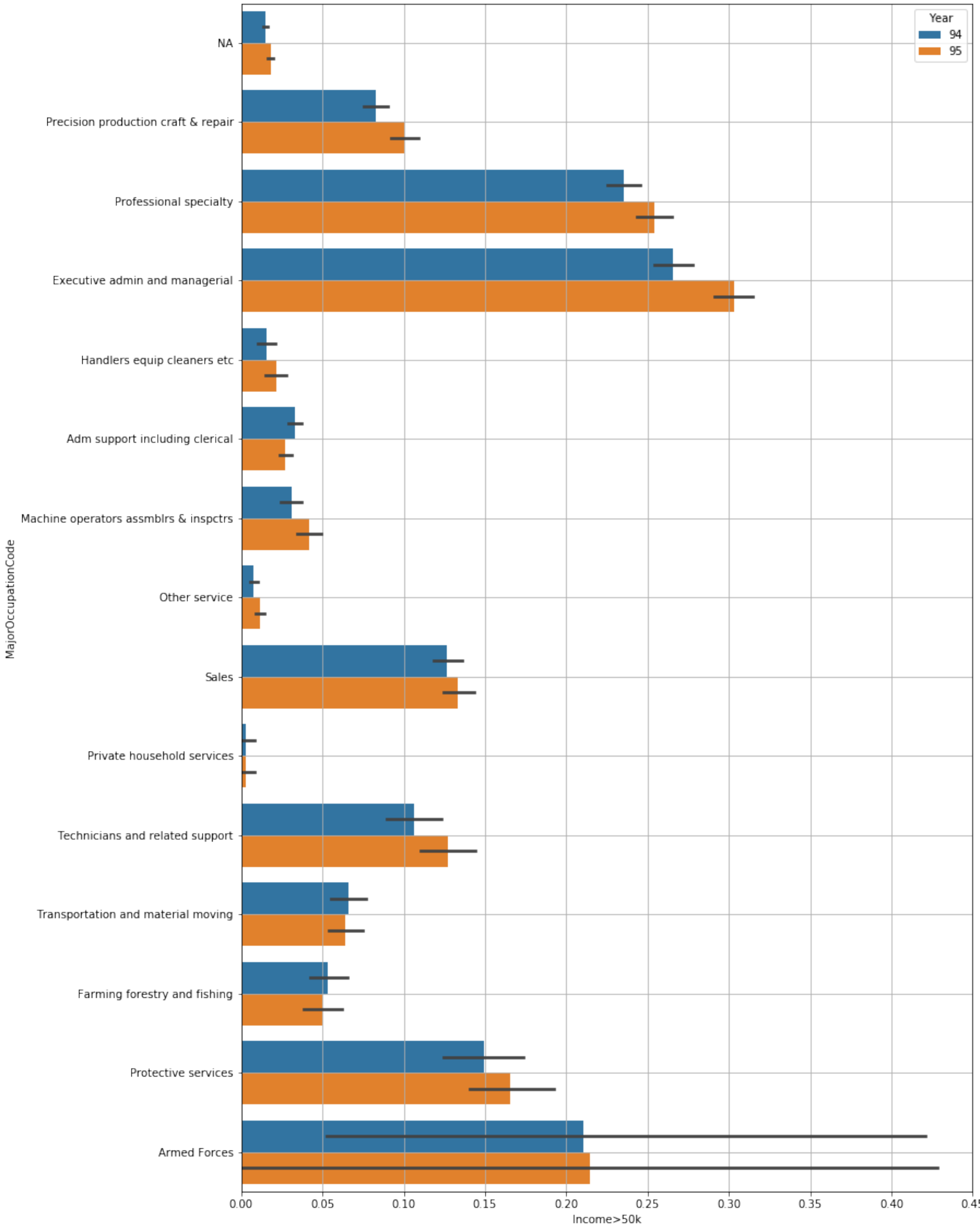
In [482]:
```
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'OccupationCode', order=df['OccupationCode'].unique())
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=60) # set new labels
```



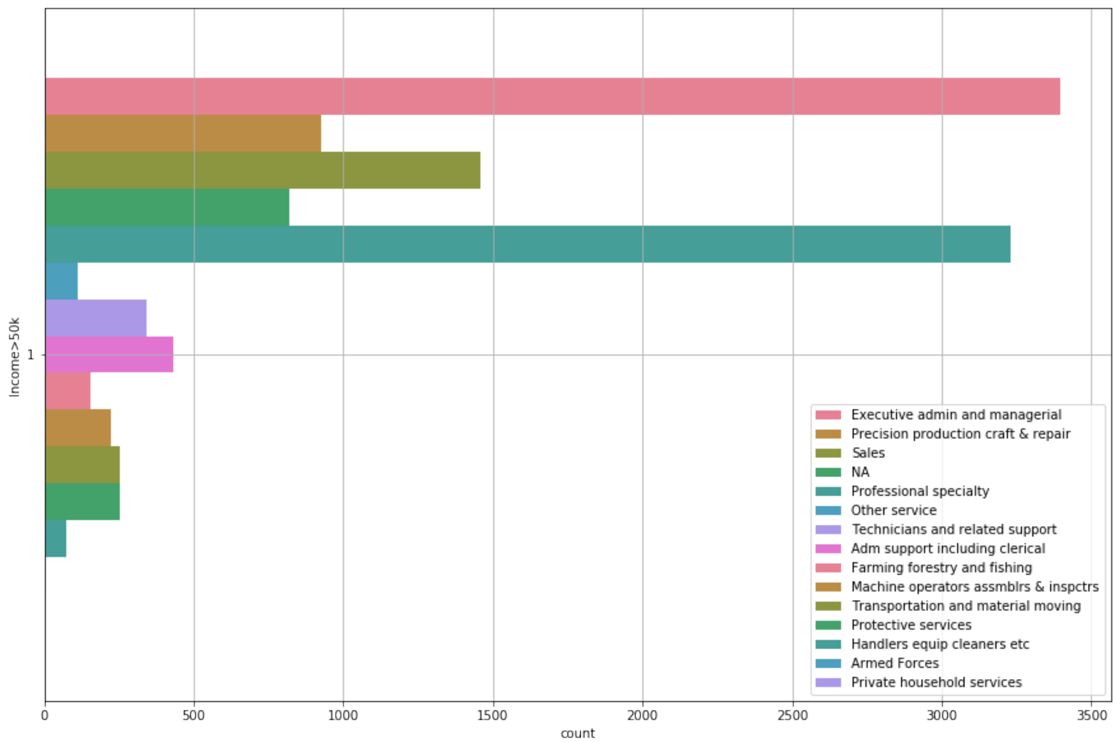

```
In [483]: plt.figure(figsize=(15,15))
          sns.countplot(y='Income>50k', hue='OccupationCode', data=df[df['Income>50k']==1],pale
          plt.grid()
          plt.legend(fontsize='medium')

Out[483]: <matplotlib.legend.Legend at 0x12d99f609e8>
```




```
In [484]: plt.figure(figsize=(12,20))
          sns.barplot(y='MajorOccupationCode', x='Income>50k', data=df, orient="h", hue='Year'
          plt.grid(True)
```




```
In [485]: plt.figure(figsize=(15,10))
          sns.countplot(y='Income>50k', hue='MajorOccupationCode', data=df[df['Income>50k']==1
          plt.grid()
          plt.legend(fontsize='medium')

Out[485]: <matplotlib.legend.Legend at 0x12c80222ef0>
```



In [486]:
```
#Occupation Code is too scattered, and the values are too small for each Code. I wil
df.drop('OccupationCode', axis=1, inplace=True)
df_test.drop('OccupationCode', axis=1, inplace=True)
```

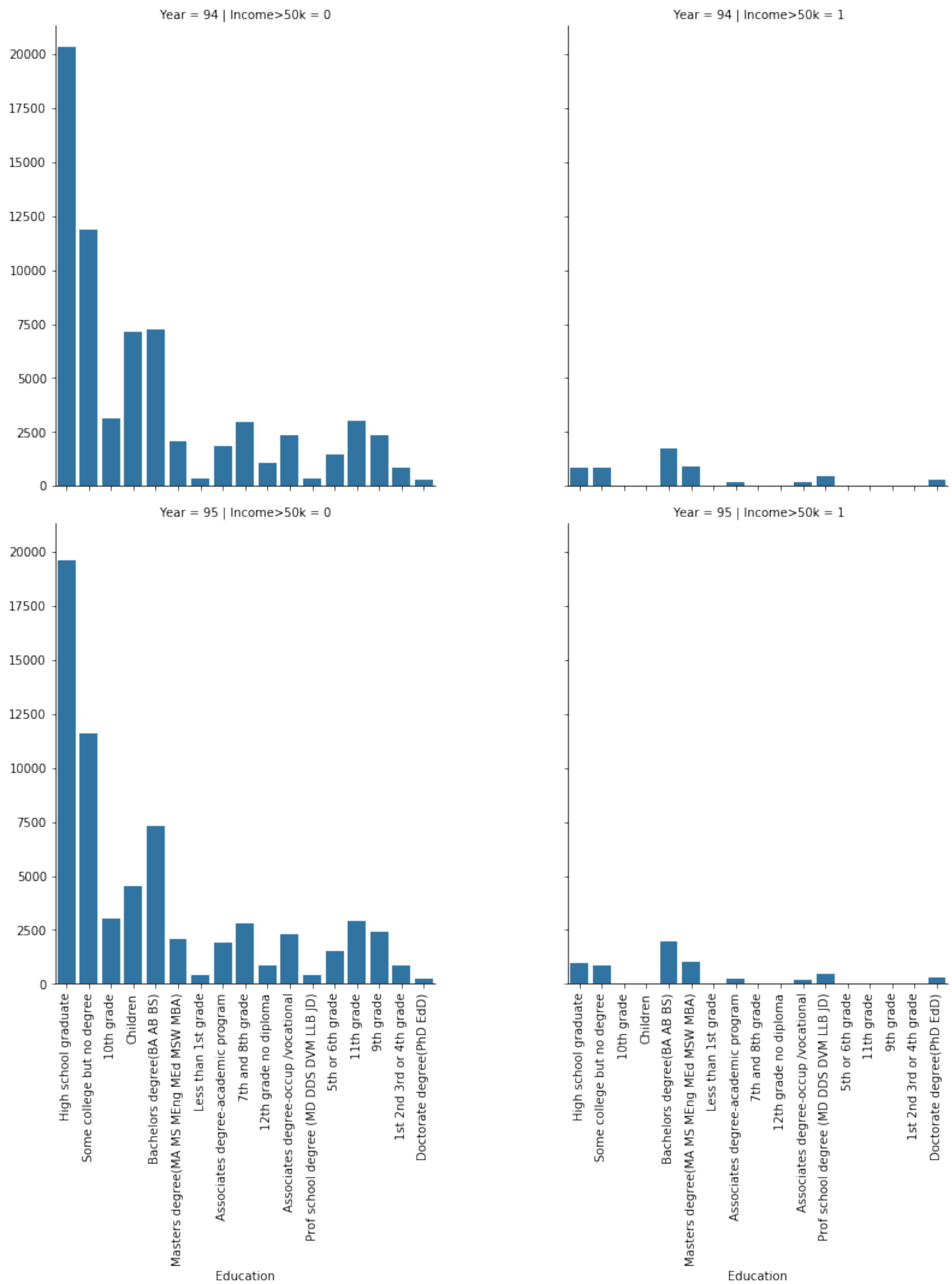
Education

In [487]:
```
df['Education'].value_counts()
```

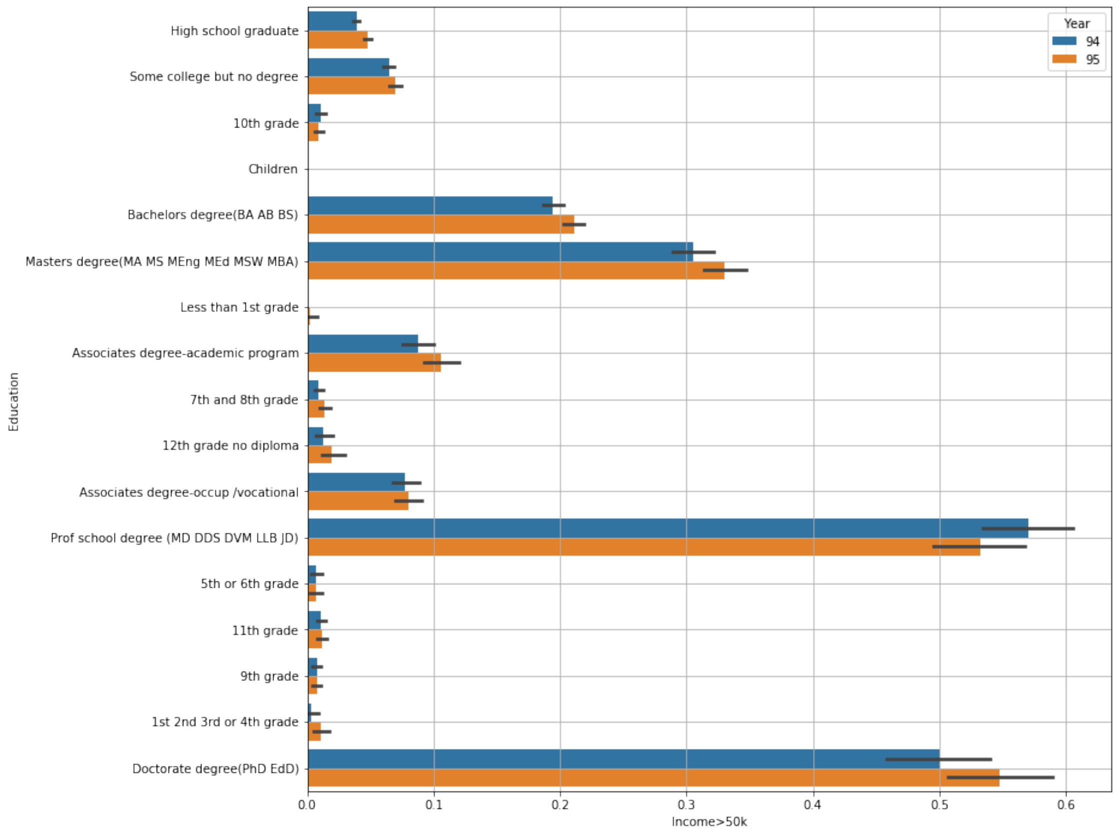
Out[487]:
```
High school graduate                      41733
Some college but no degree                25146
Bachelors degree(BA AB BS)                18278
Children                                  11679
10th grade                                 6199
Masters degree(MA MS MEng MEd MSW MBA)     6059
11th grade                                 6003
7th and 8th grade                          5852
Associates degree-occup /vocational        5034
9th grade                                  4787
Associates degree-academic program         4125
5th or 6th grade                           2999
12th grade no diploma                      1949
1st 2nd 3rd or 4th grade                   1683
Prof school degree (MD DDS DVM LLB JD)     1598
Doctorate degree(PhD EdD)                  1120
```

```
Less than 1st grade                         761
Name: Education, dtype: int64
```

In [488]:
```python
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'Education', order=df['Education'].unique() )
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```

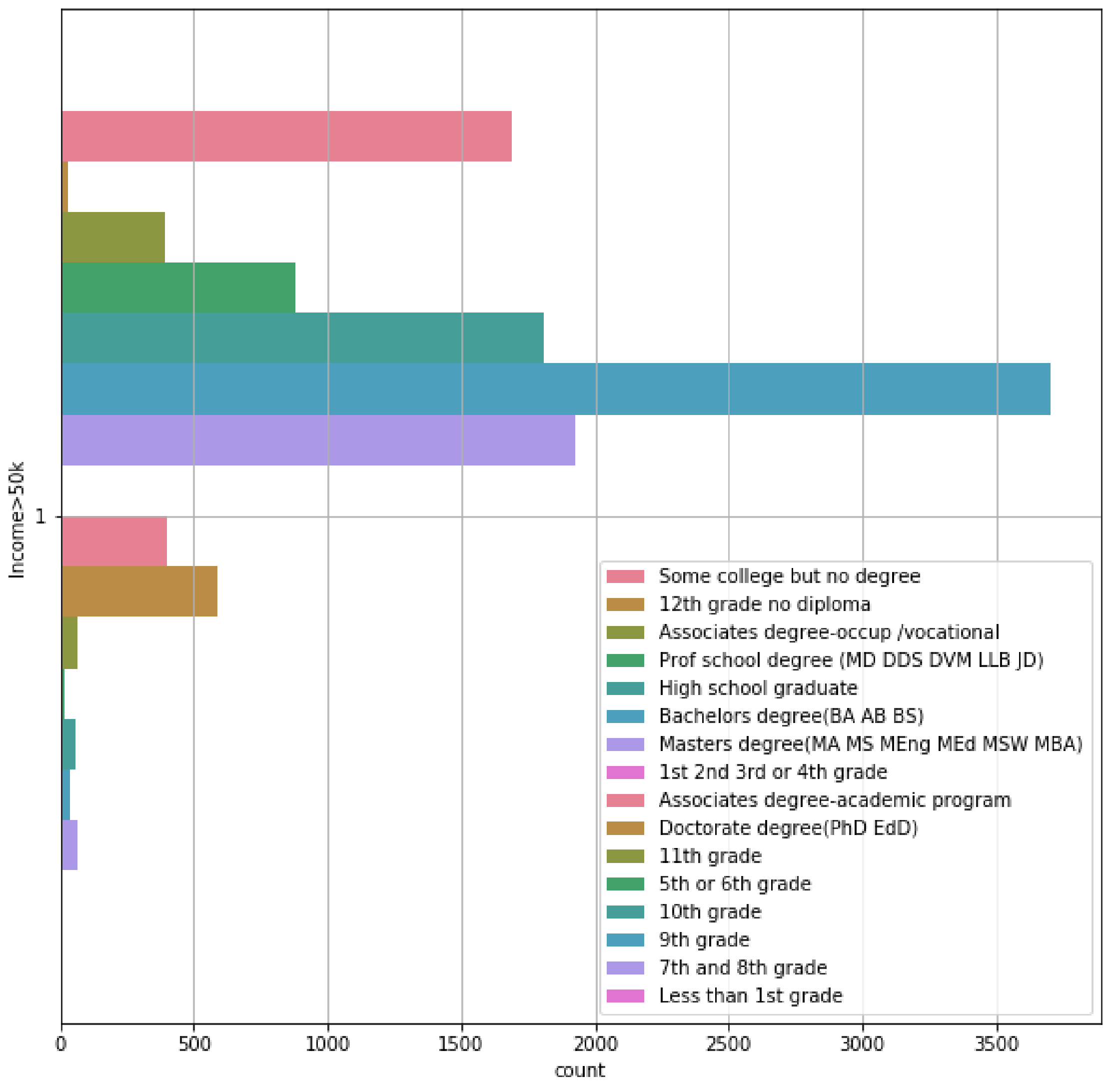



```
In [489]: plt.figure(figsize=(12,12))
          sns.barplot(y='Education', x='Income>50k', data=df, orient="h", hue='Year', dodge=Tr
          plt.grid(True)
```



In [490]:
```
plt.figure(figsize=(10,10))
sns.countplot(y='Income>50k', hue='Education', data=df[df['Income>50k']==1],palette=s
plt.grid()
plt.legend(fontsize='medium')
```

Out[490]: <matplotlib.legend.Legend at 0x12cd3999588>



In [491]:
```
df['Education'].unique()
```

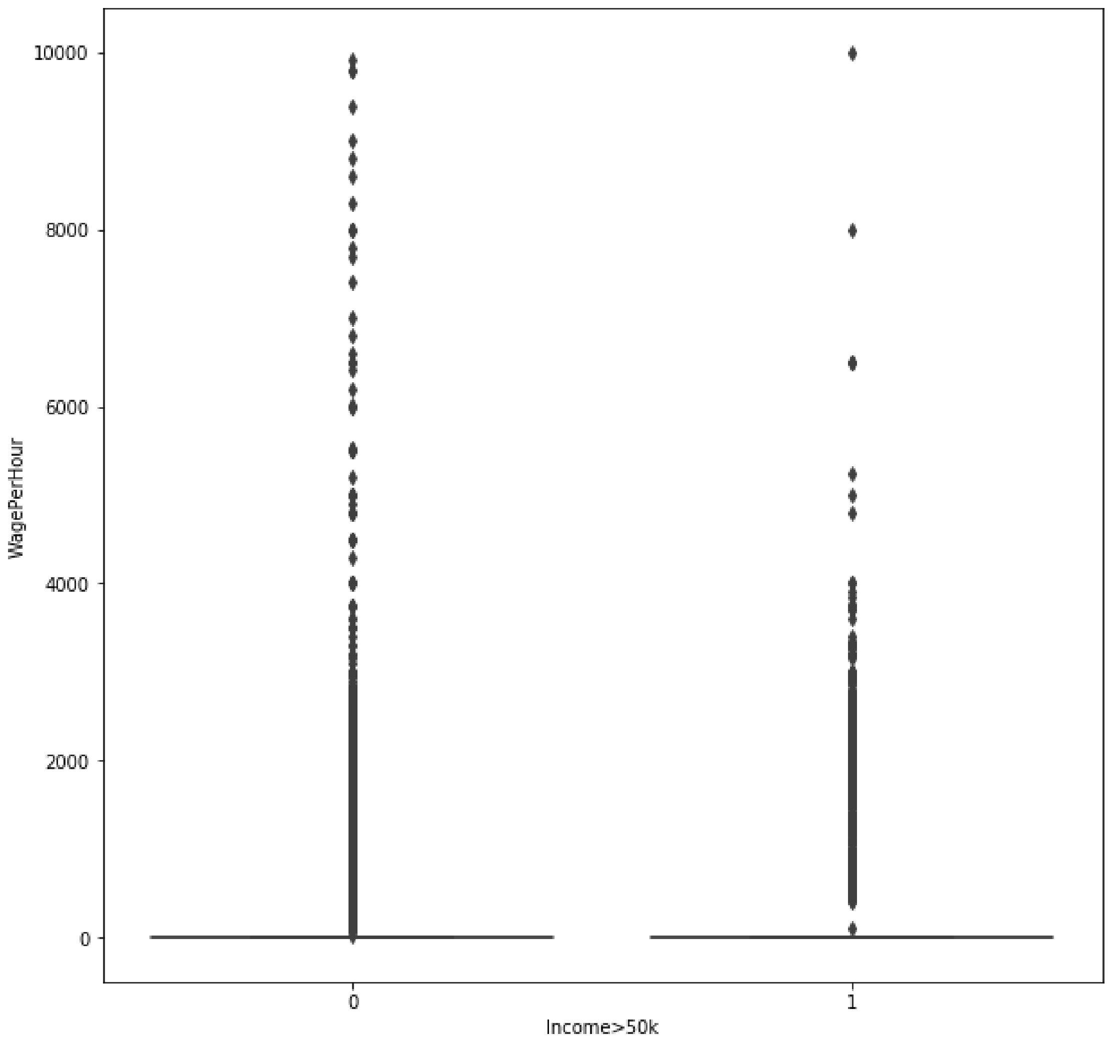
Out[491]:
```
array(['High school graduate', 'Some college but no degree', '10th grade',
       'Children', 'Bachelors degree(BA AB BS)',
       'Masters degree(MA MS MEng MEd MSW MBA)', 'Less than 1st grade',
       'Associates degree-academic program', '7th and 8th grade',
       '12th grade no diploma', 'Associates degree-occup /vocational',
       'Prof school degree (MD DDS DVM LLB JD)', '5th or 6th grade',
       '11th grade', '9th grade', '1st 2nd 3rd or 4th grade',
       'Doctorate degree(PhD EdD)'], dtype=object)
```

In [492]:
```
#I will group all Education levels other than 'High school graduate', 'Some college 
#Bachelors degree(BA AB BS)' and 'Masters degree(MA MS MEng MEd MSW MBA)' as 'Other'
#I will also rename 'Some college but no degree' as 'High school graduate'.
```

```
df['Education'] = df['Education'].apply(lambda x: 'Other' if x not in ['High school g
        'Bachelors degree(BA AB BS)','Masters degree(MA MS MEng MEd MSW MBA)',
        'Prof school degree (MD DDS DVM LLB JD)', 'Doctorate degree(PhD EdD)'] else x

df['Education'] = np.where(df['Education'] == 'Some college but no degree', 'High sch

df_test['Education'] = df_test['Education'].apply(lambda x: 'Other' if x not in ['Hig
        'Bachelors degree(BA AB BS)','Masters degree(MA MS MEng MEd MSW MBA)',
        'Prof school degree (MD DDS DVM LLB JD)', 'Doctorate degree(PhD EdD)'] else x

df_test['Education'] = np.where(df_test['Education'] == 'Some college but no degree'
```

WagePerHour

```
In [493]: df.columns

Out[493]: Index(['Age', 'ClassOfWorker', 'Education', 'WagePerHour', 'MaritalStatus',
                 'MajorIndustryCode', 'MajorOccupationCode', 'Race', 'HispanicOrigin',
                 'Sex', 'FullOrPartTime', 'CapitalGains', 'CapitalLosses',
                 'StockDividends', 'TaxFilerStat', 'HouseholdFamilyStatus',
                 'HouseholdSummary', 'LiveInHouse1Y', 'NumPersonsWorkedEmployer',
                 'CountryBirthFather', 'CountryBirthMother', 'CountryBirthSelf',
                 'Citizenship', 'OwnBusiness', 'VeteranBenefits', 'WeeksWorkedInY',
                 'Year', 'Income>50k'],
                dtype='object')

In [494]: plt.figure(figsize=(10,10))
          sns.boxplot(x='Income>50k', y='WagePerHour', data=df)

Out[494]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4eb24d68>
```



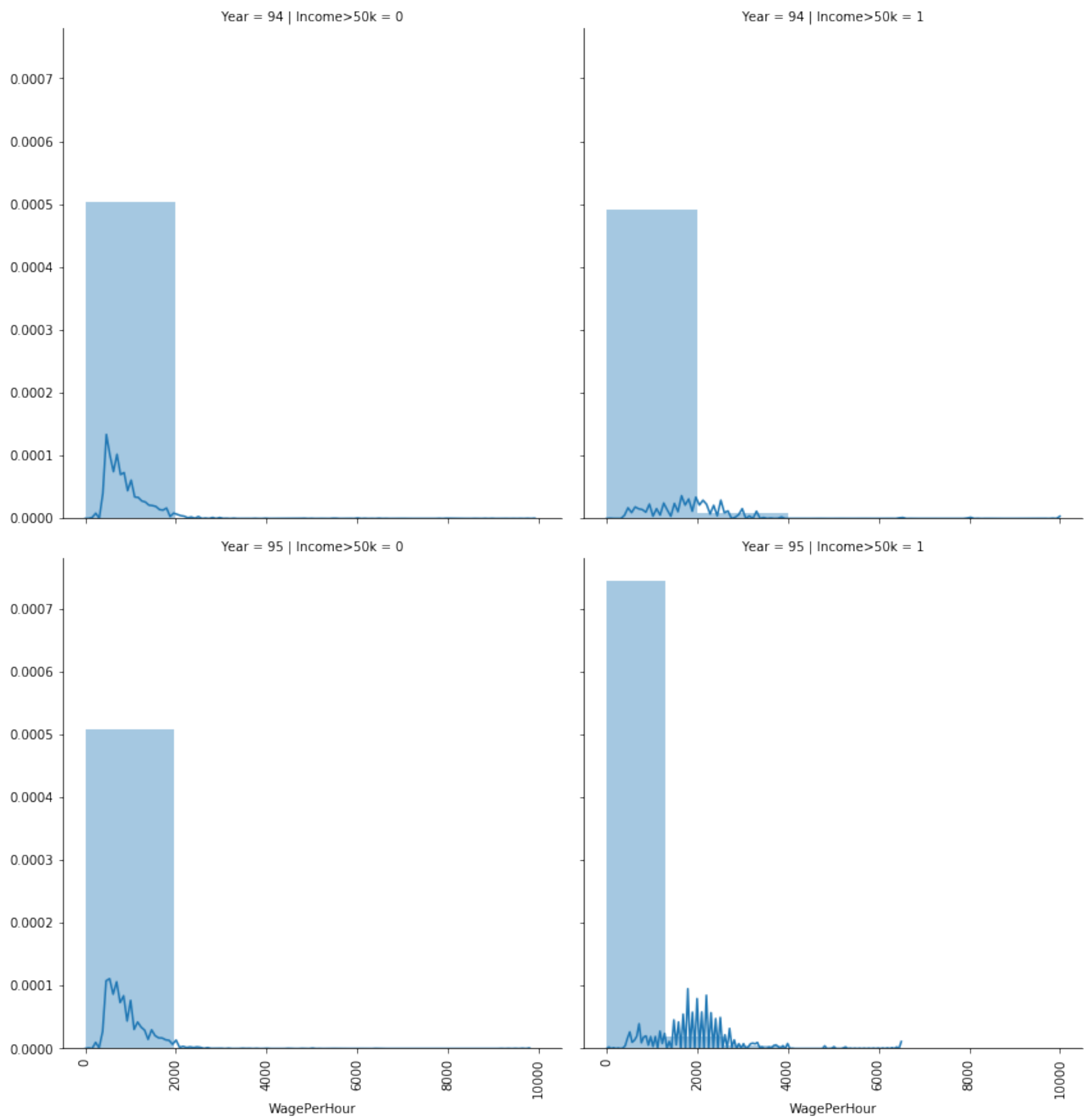

```
In [495]: df.groupby('WagePerHour')['Age'].count()

Out[495]: WagePerHour
          0        134017
          20            1
          70            1
          75            2
          100          11
          110           1
          125           1
          135           1
          143           1
          150           6
          170           1
          173           1
```

190	1
200	27
205	1
210	7
212	2
213	20
215	2
220	2
225	4
230	1
232	1
233	1
234	1
235	2
245	1
250	11
252	1
255	3
...	
4300	1
4500	4
4800	1
4807	5
4900	1
5000	5
5200	1
5250	1
5500	4
5525	1
6000	2
6009	1
6200	1
6410	1
6500	6
6600	1
6800	1
7000	1
7400	1
7700	1
7800	1
8000	4
8300	1
8600	1
8800	1
9000	1
9400	1
9800	2
9916	1

```
9999         1
Name: Age, Length: 1227, dtype: int64
```

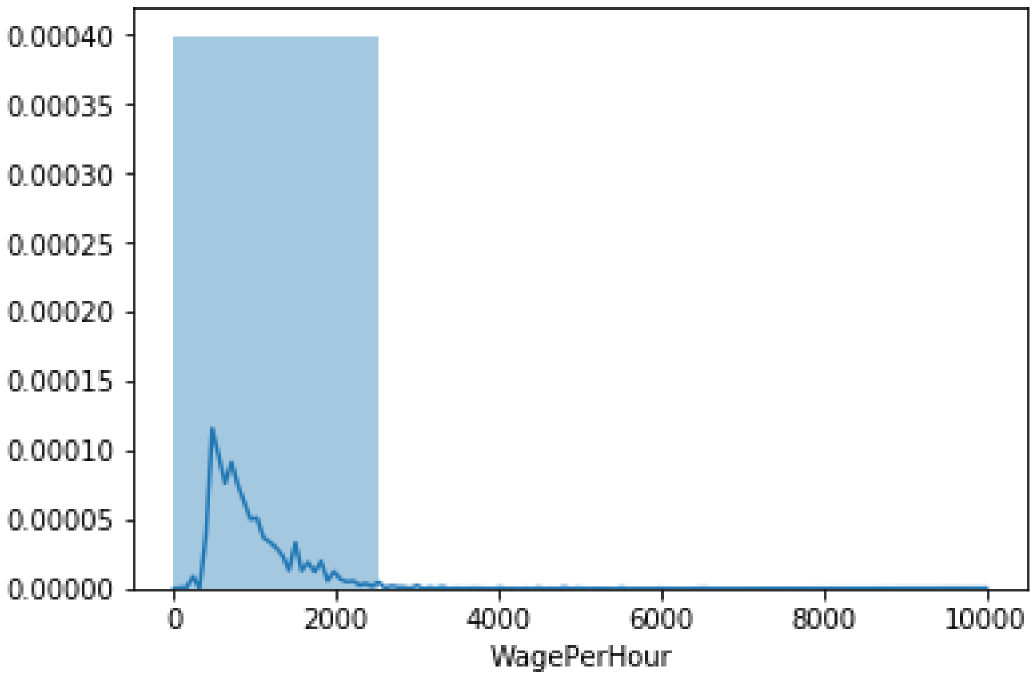
In [496]:
```
plt.figure(figsize=(10,10))

g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.distplot, 'WagePerHour', bins=5)
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```

```
<Figure size 720x720 with 0 Axes>
```



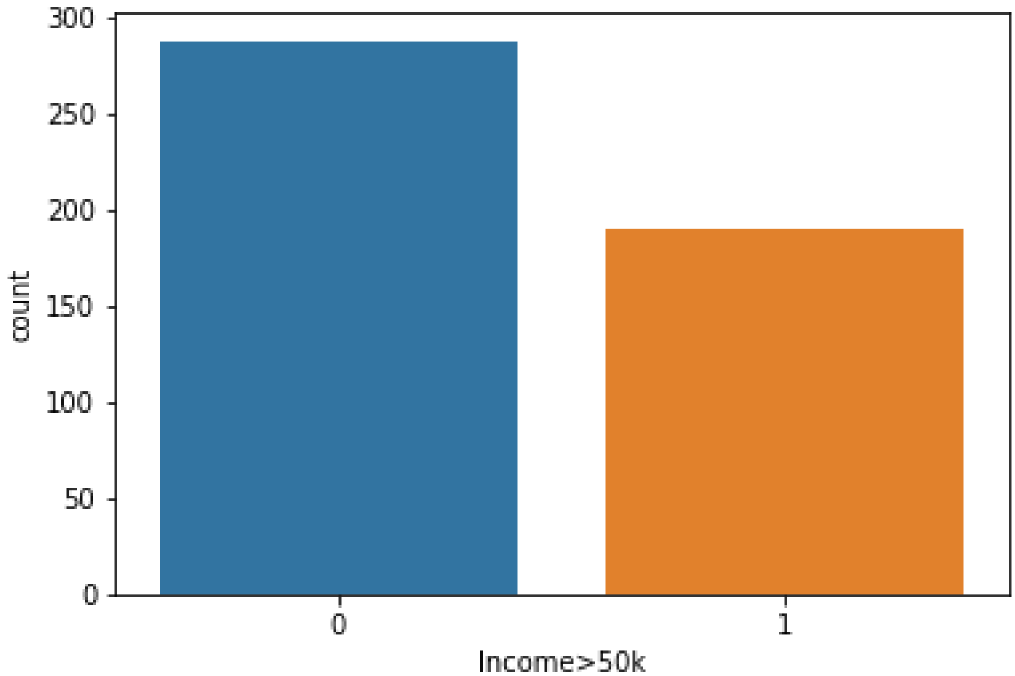
```
In [497]: sns.distplot(df['WagePerHour'], bins=4)
```

```
Out[497]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4be8aeb8>
```




```
In [498]: sns.countplot(x='Income>50k', data=df[df['WagePerHour']>2000])
```

```
Out[498]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4be50eb8>
```



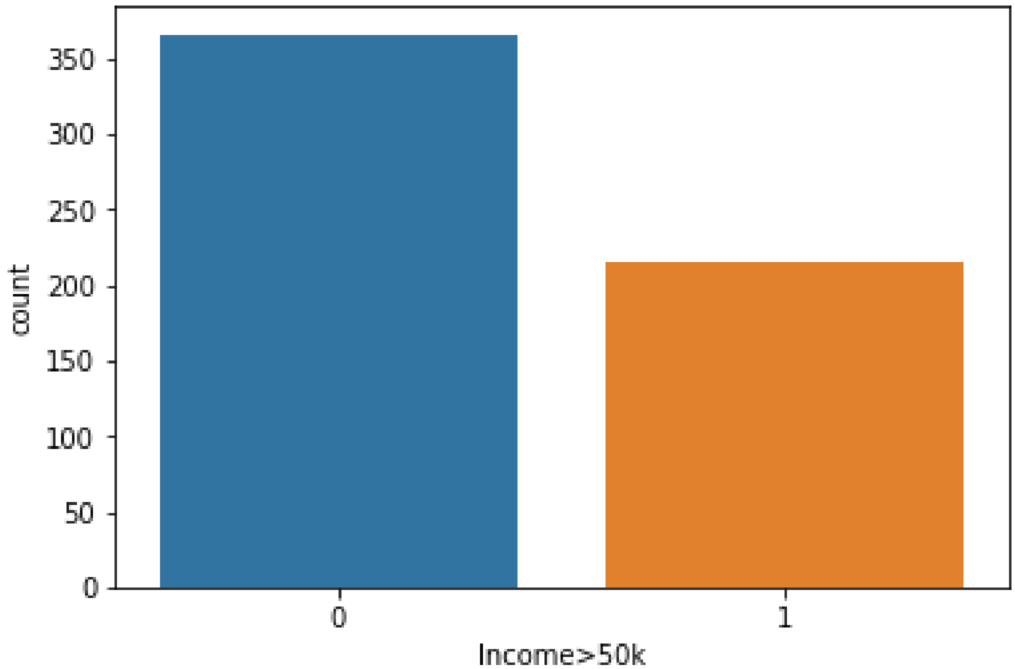
5 It looks like the WagePerHouse has some wrong inputs. I might delete the rows, but a big oercentage of them are positive

for Income>50K. I will cap the Wages at $2000 per hour.

In [499]:
```
#I will cap the WagePerHour to $2,000. Higher numbers do not seem to be correct.

df['WagePerHour'] = np.where(df['WagePerHour'] > 2000, 2000, df['WagePerHour'])
df_test['WagePerHour'] = np.where(df_test['WagePerHour'] > 2000, 2000, df_test['WageI
```

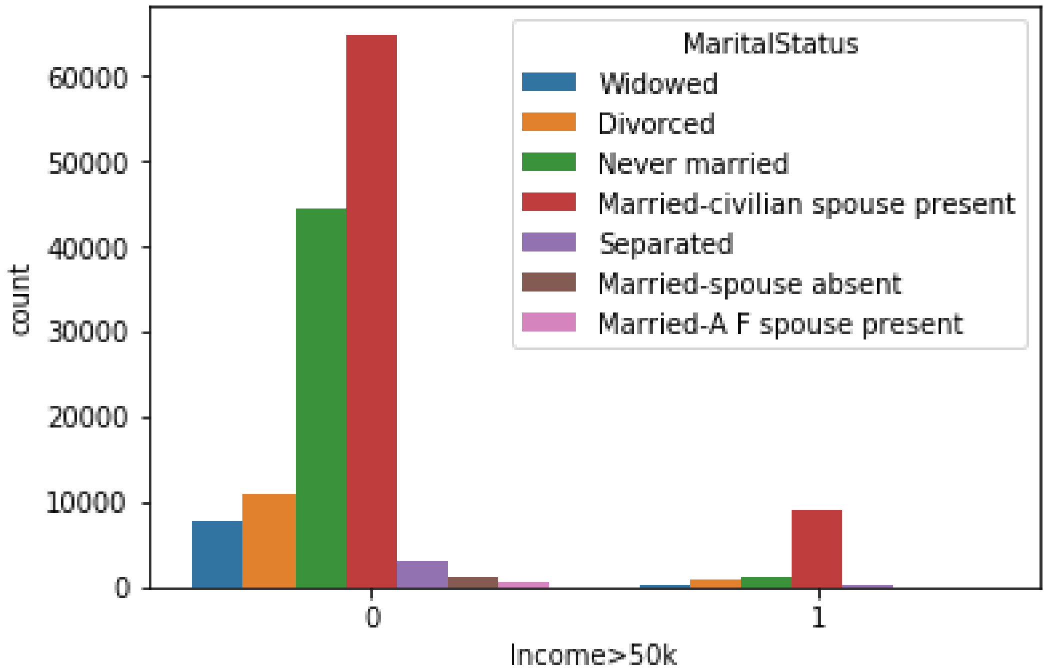
In [500]:
```
sns.countplot(x='Income>50k', data=df[df['WagePerHour']== 2000])
```

Out[500]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12d4be1cf28>
```

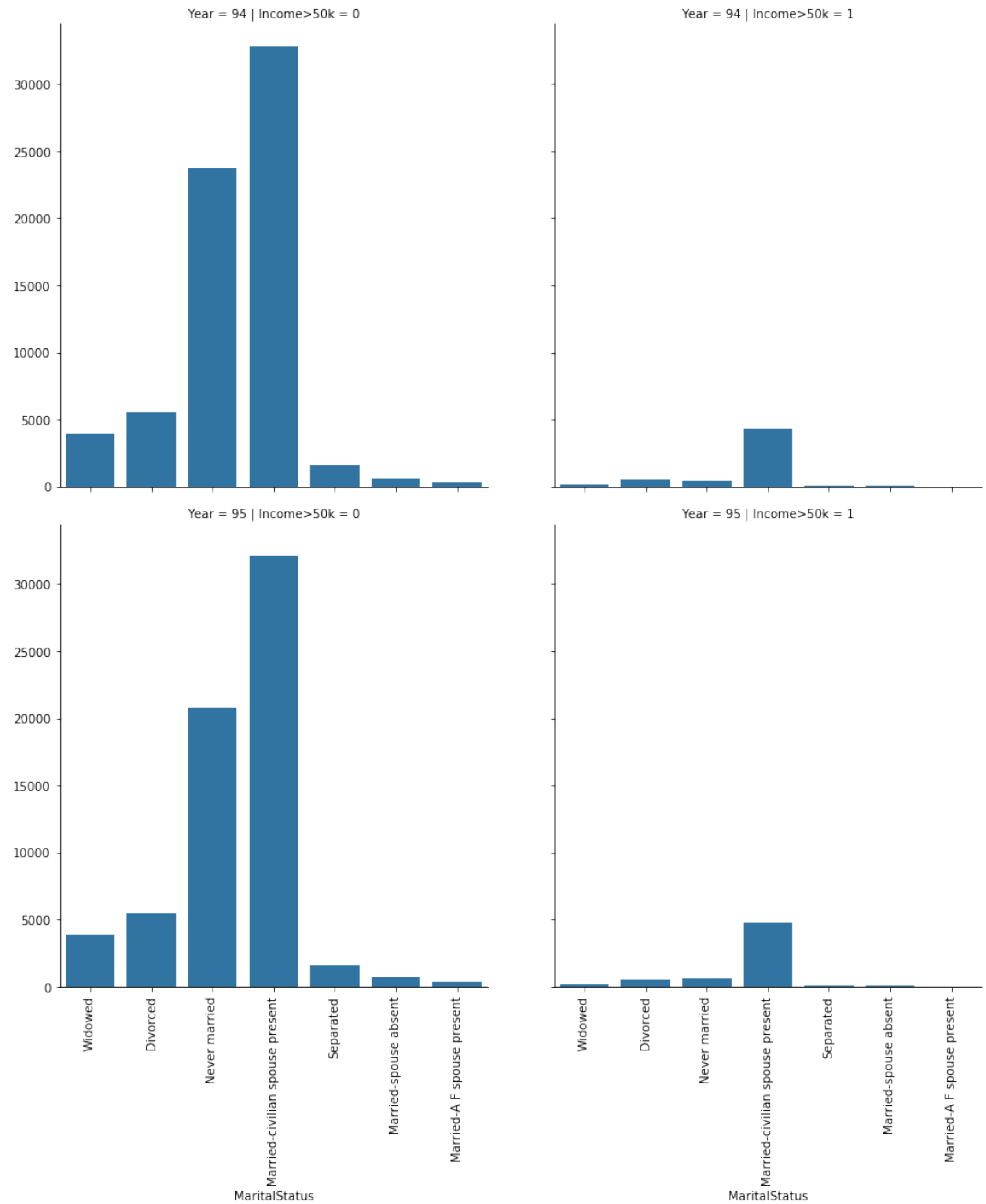


Marital Status

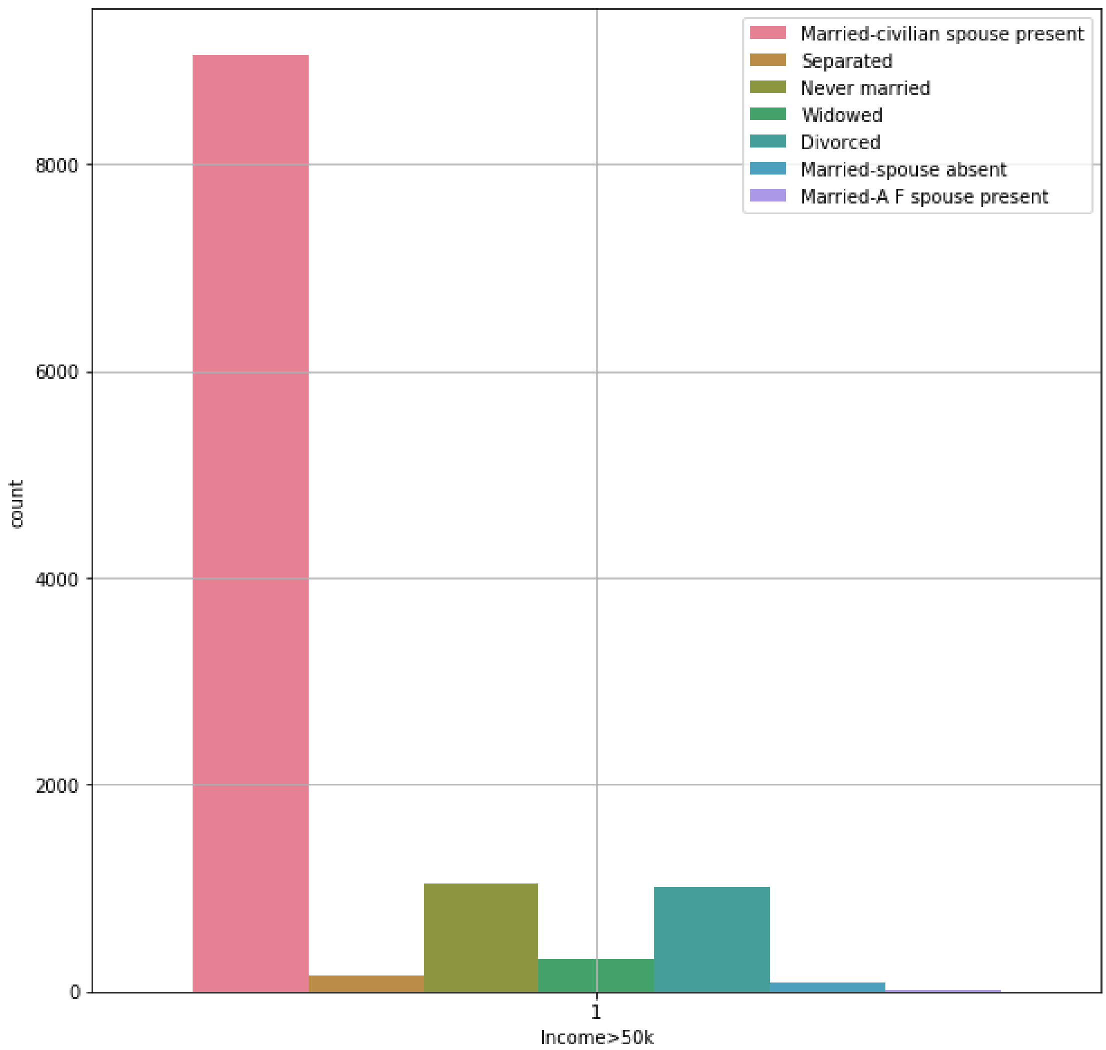
```
In [501]: df['MaritalStatus'].value_counts()

Out[501]: Married-civilian spouse present    73979
          Never married                      45594
          Divorced                           12046
          Widowed                             8032
          Separated                           3325
          Married-spouse absent               1396
          Married-A F spouse present           633
          Name: MaritalStatus, dtype: int64

In [502]: sns.countplot(x='Income>50k', hue='MaritalStatus', data=df)

Out[502]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4bdfb0f0>
```



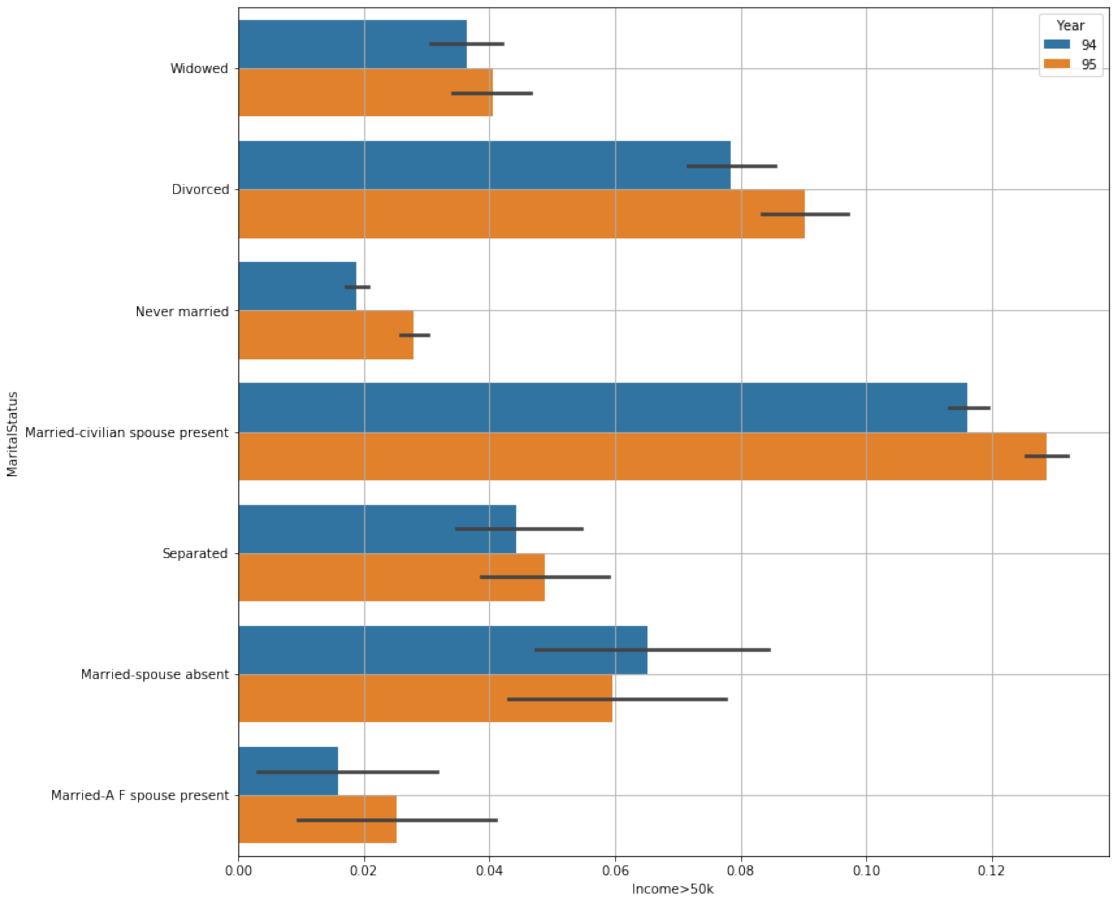

```
In [503]: g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
          g.map(sns.countplot, 'MaritalStatus', order=df['MaritalStatus'].unique())
          g.set_xticklabels(rotation=30)

          for ax in g.axes.flat:
              labels = ax.get_xticklabels() # get x labels
              ax.set_xticklabels(labels, rotation=90) # set new labels
```




```
In [504]: plt.figure(figsize=(10,10))
          sns.countplot(x='Income>50k', hue='MaritalStatus', data=df[df['Income>50k']==1],pale
          plt.grid()
          plt.legend(fontsize='medium')

Out[504]: <matplotlib.legend.Legend at 0x12d4eb55320>
```




```
In [505]: plt.figure(figsize=(12,12))
          sns.barplot(y='MaritalStatus', x='Income>50k', data=df, orient="h", hue='Year', dodge
          plt.grid(True)
```



In [506]:
```
df['MaritalStatus'].unique()
```

Out[506]:
```
array(['Widowed', 'Divorced', 'Never married',
       'Married-civilian spouse present', 'Separated',
       'Married-spouse absent', 'Married-A F spouse present'], dtype=object)
```

In [507]:
```
#I will join some values under two: Married and Divorced
df['MaritalStatus'] = df['MaritalStatus'].apply(lambda x: 'Divorced' if x in ['Divor
                  else 'Married' if x in ['Married-civilian spouse present', 'Married-s
                  else x)

df_test['MaritalStatus'] = df_test['MaritalStatus'].apply(lambda x: 'Divorced' if x i
                  else 'Married' if x in ['Married-civilian spouse present', 'Married-s
                  else x)
```

In [508]:
```
df['MaritalStatus'].value_counts()
```

Out[508]:
```
Married          76008
Never married    45594
Divorced         15371
```

```
Widowed           8032
Name: MaritalStatus, dtype: int64
```

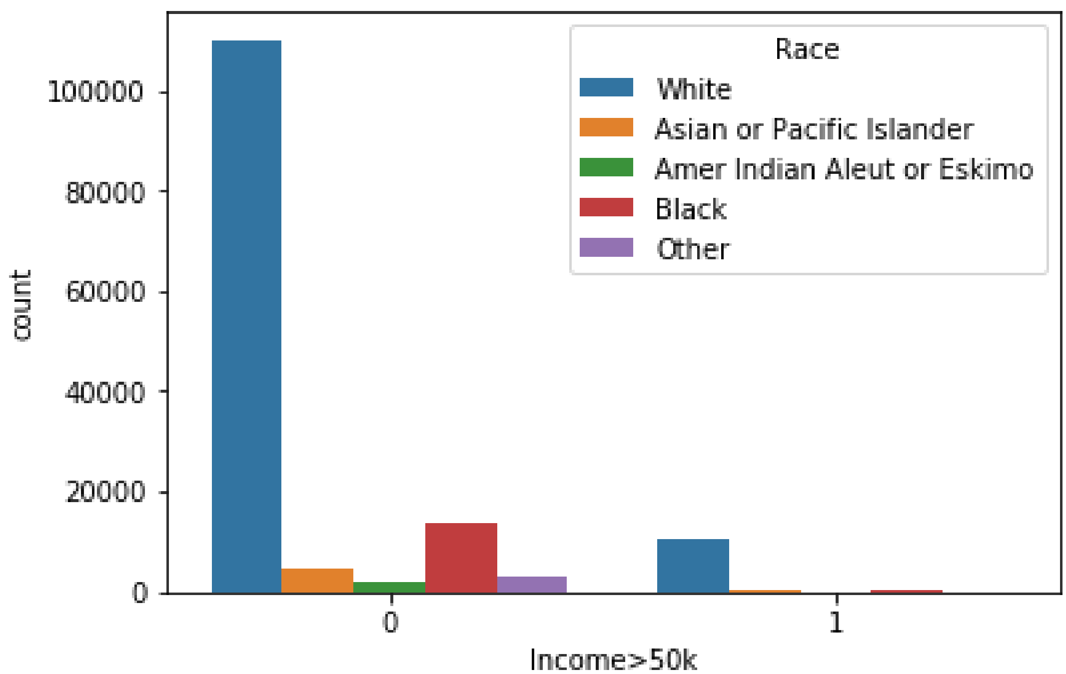
Race

In [509]:
```
df['Race'].value_counts()
```

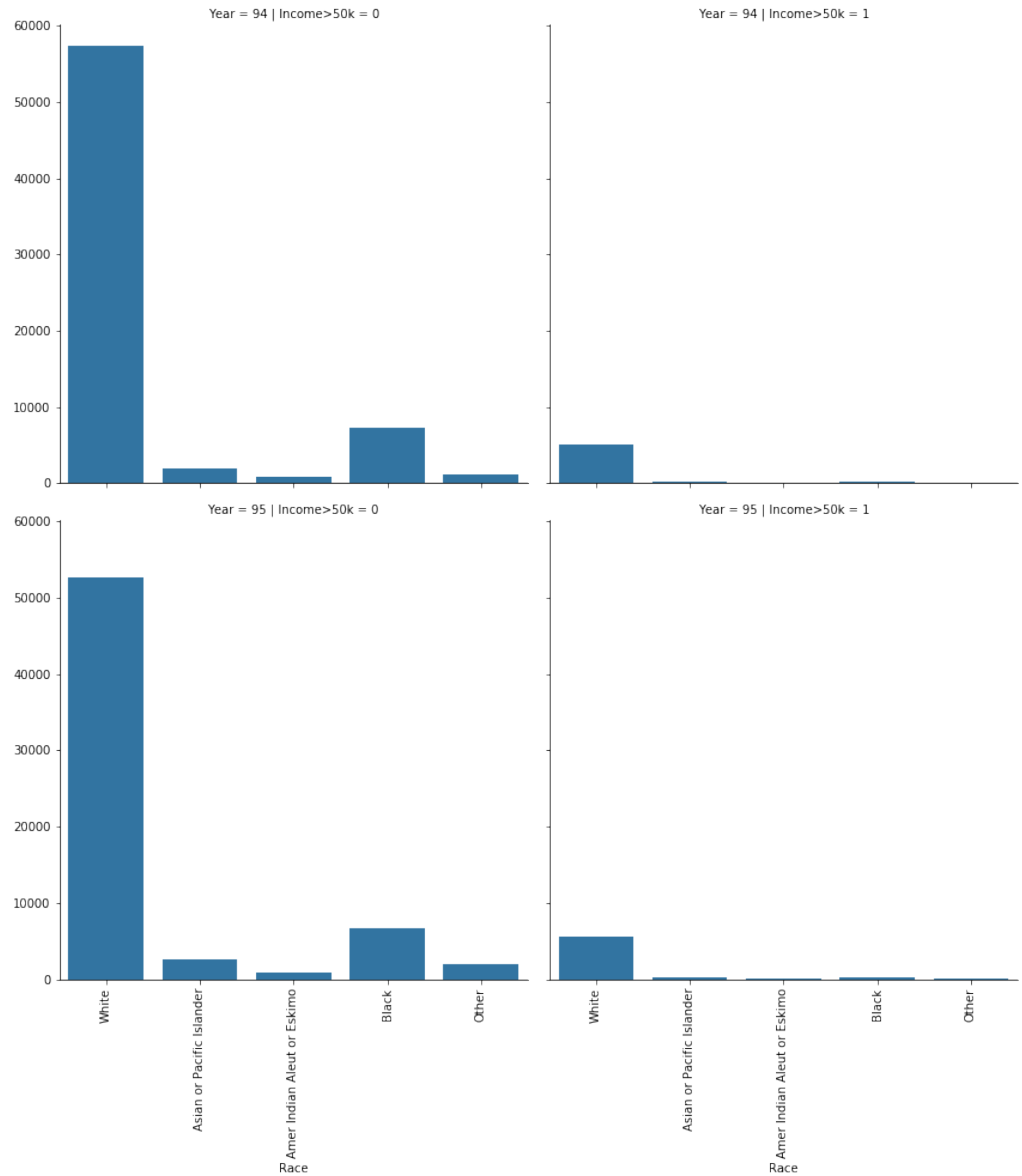
Out[509]:
```
White                          120679
Black                           14424
Asian or Pacific Islander        4896
Other                            3154
Amer Indian Aleut or Eskimo      1852
Name: Race, dtype: int64
```

In [510]:
```
sns.countplot(x='Income>50k', hue='Race', data=df)
```

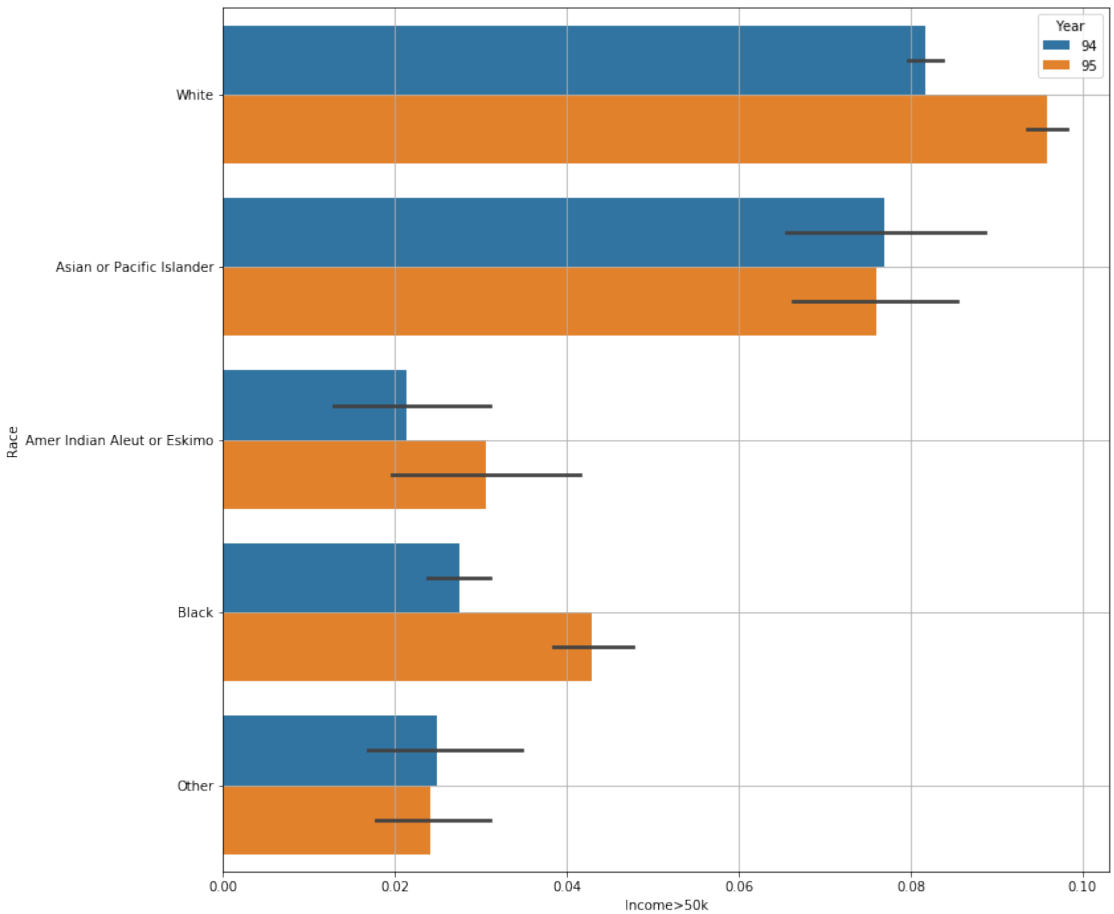
Out[510]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12d4ebc3c50>
```



In [511]:
```
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'Race', order=df['Race'].unique())
g.set_xticklabels(rotation=30)

for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```




```
In [512]: plt.figure(figsize=(12,12))
          sns.barplot(y='Race', x='Income>50k', data=df, orient="h", hue='Year', dodge=True)
          plt.grid(True)
```

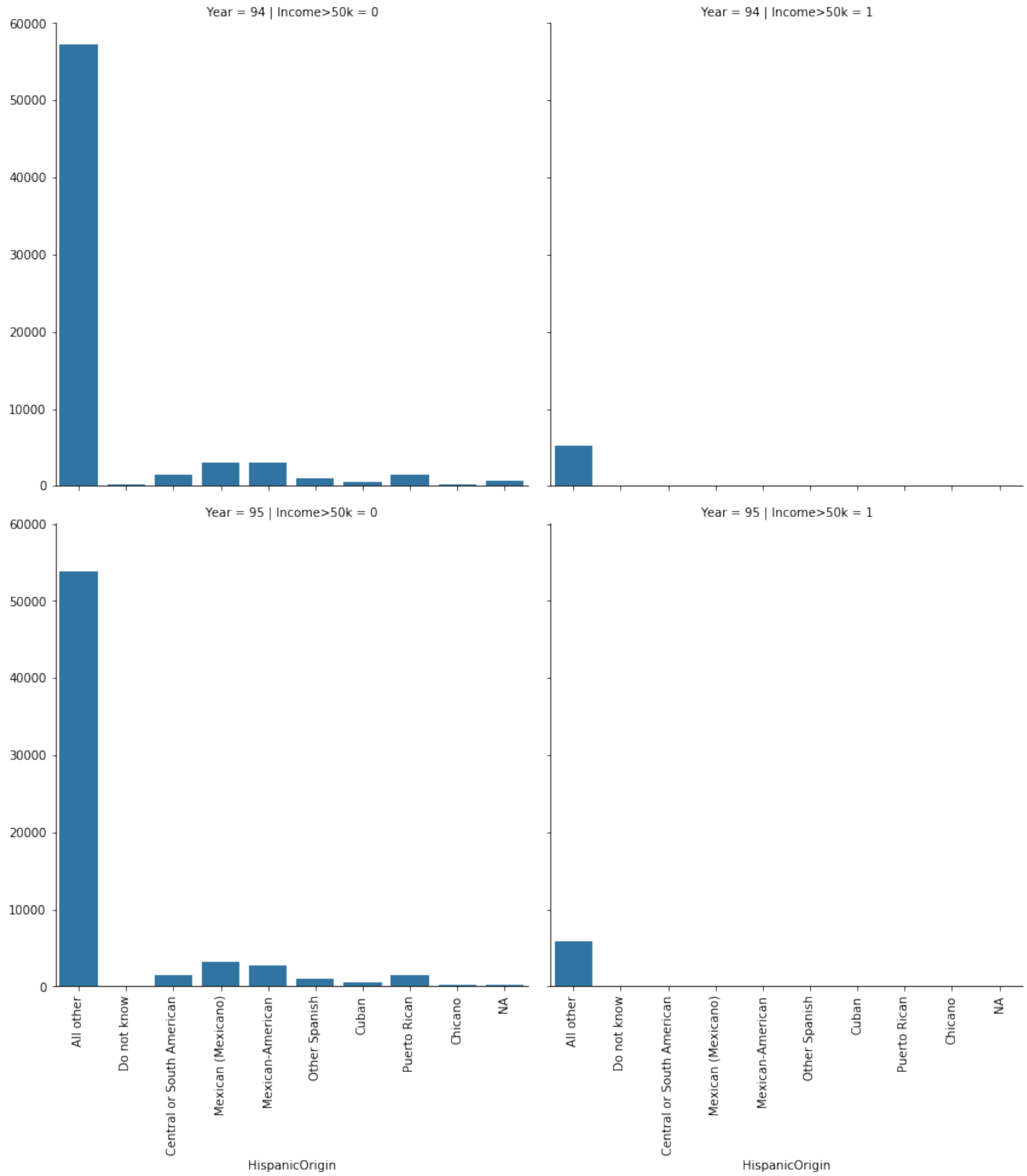



```
In [513]: #I will add 'Amer Indian Aleut or Eskimo' to 'Other'
          df['Race'] = df['Race'].apply(lambda x: 'Other' if x == 'Amer Indian Aleut or Eskimo
          df_test['Race'] = df_test['Race'].apply(lambda x: 'Other' if x == 'Amer Indian Aleut
```

HispanicOrigin

```
In [514]: df['HispanicOrigin'].value_counts(normalize=True)

Out[514]: All other                      0.842978
          Mexican (Mexicano)             0.043364
          Mexican-American               0.041185
          Central or South American      0.021323
          Puerto Rican                   0.020068
          Other Spanish                  0.014606
          Cuban                          0.006862
          NA                             0.005593
          Chicano                        0.002034
          Do not know                    0.001986
          Name: HispanicOrigin, dtype: float64
```

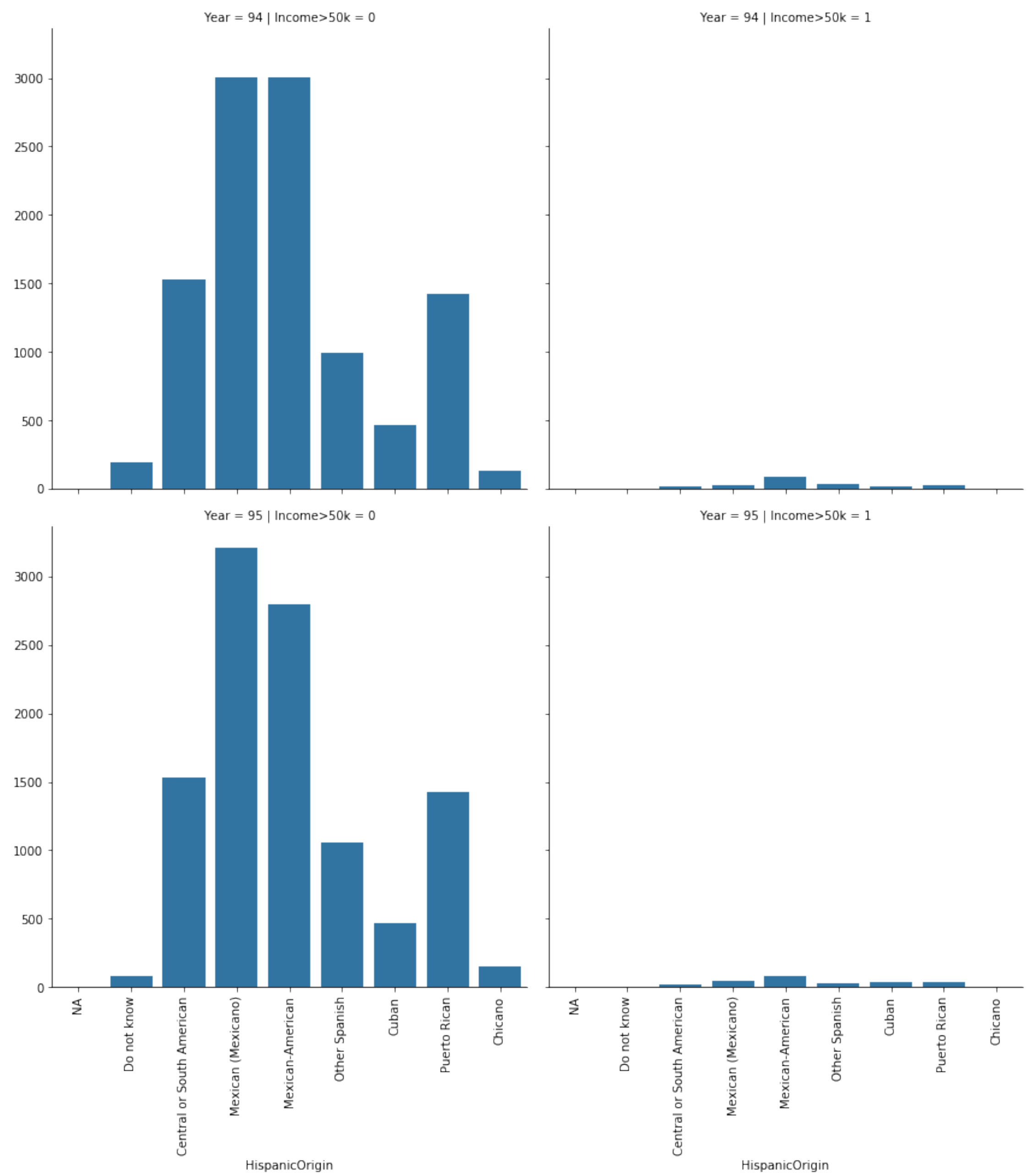
```
In [515]: g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
          g.map(sns.countplot, 'HispanicOrigin', order=df['HispanicOrigin'].unique())
          g.set_xticklabels(rotation=30)

          for ax in g.axes.flat:
              labels = ax.get_xticklabels() # get x labels
              ax.set_xticklabels(labels, rotation=90) # set new labels
```



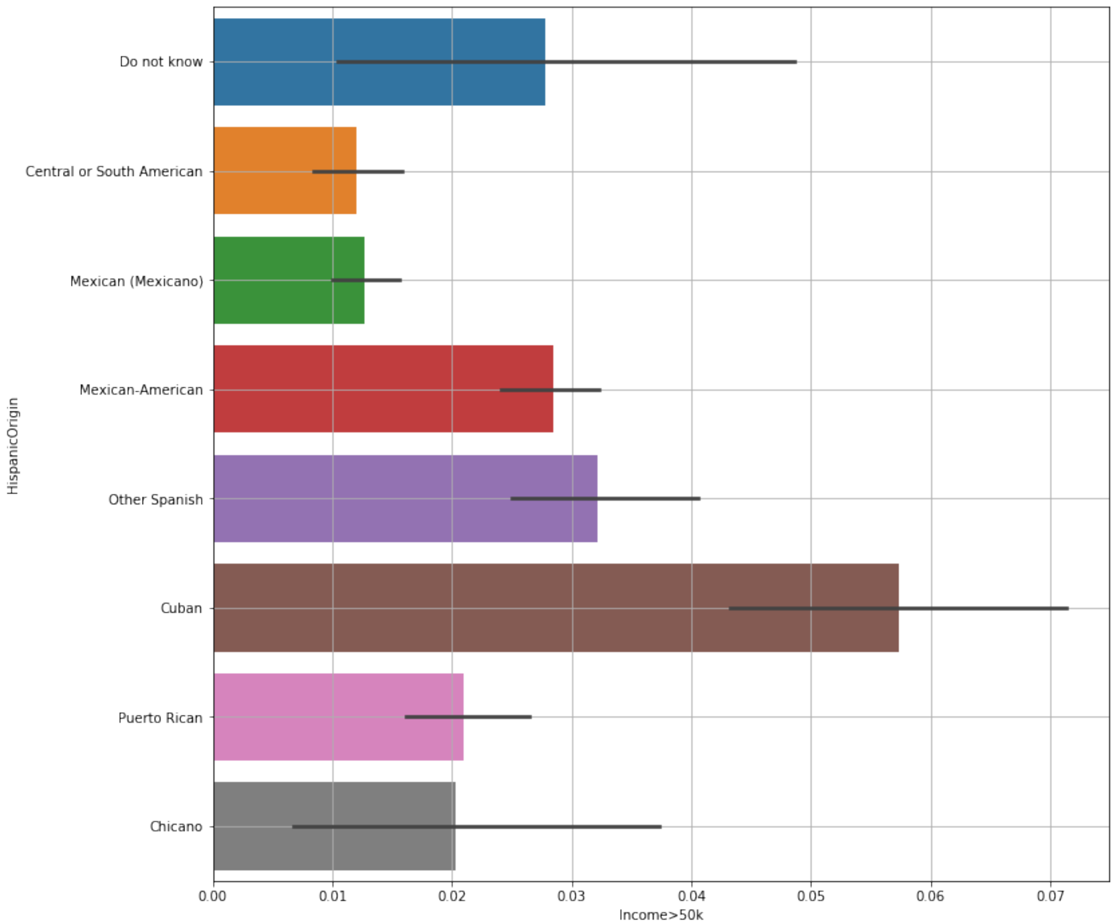

```
In [516]: df['HispanicOrigin'].replace('All other', 'NA', inplace=True)
```

```
df_test['HispanicOrigin'].replace('All other', 'NA', inplace=True)
```

In [517]:
```
g = sns.FacetGrid(data=df[df['HispanicOrigin'] !='NA'], col='Income>50k', row='Year'
g.map(sns.countplot, 'HispanicOrigin', order=df['HispanicOrigin'].unique())
g.set_xticklabels(rotation=30)

for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```



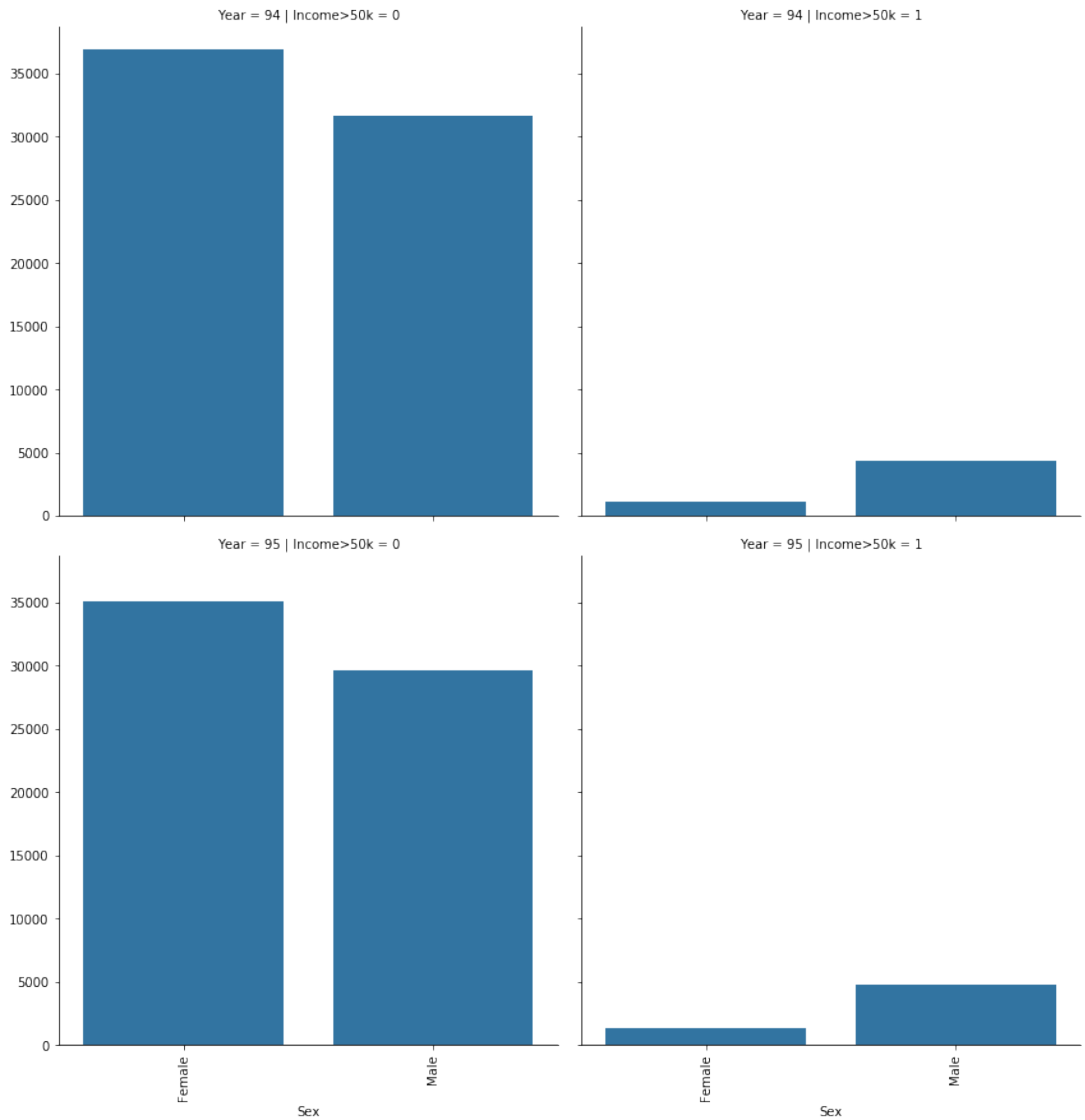
In [518]:
```
plt.figure(figsize=(12,12))
sns.barplot(y='HispanicOrigin', x='Income>50k', data=df[df['HispanicOrigin'] !='NA']
plt.grid(True)
```



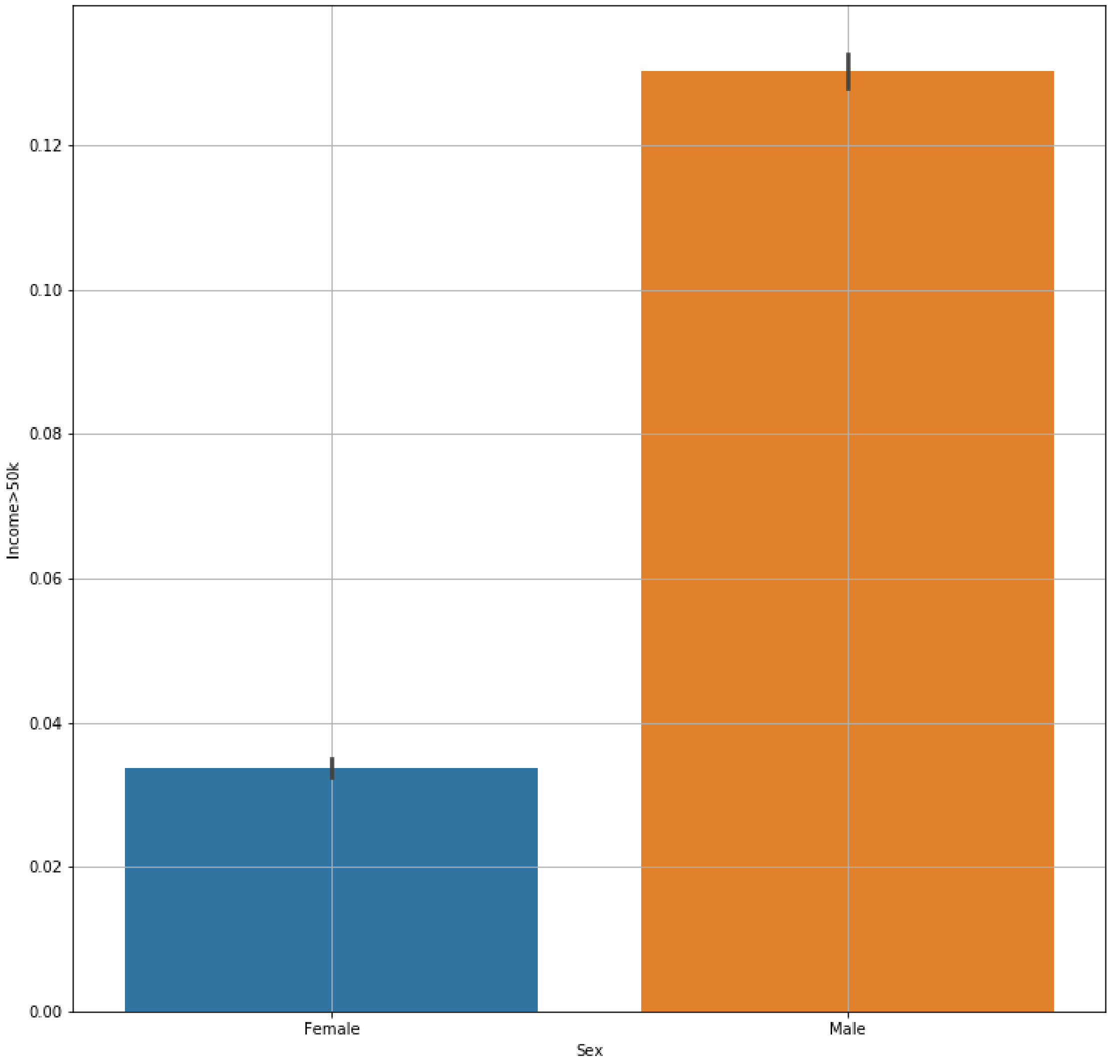
In [519]:
```
#I will drop this column. I will only be relying on Race
df.drop('HispanicOrigin', axis=1, inplace=True)
df_test.drop('HispanicOrigin', axis=1, inplace=True)
```

Sex

In [520]:
```
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'Sex', order=df['Sex'].unique())

for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```




```
In [521]: plt.figure(figsize=(12,12))
          sns.barplot(x='Sex', y='Income>50k', data=df,  dodge=True)
          plt.grid(True)
```




```
In [522]: df.rename(columns={'Sex':'Male'}, inplace=True)
          df['Male'] = np.where(df['Male']=='Male',1,0)
          df_test.rename(columns={'Sex':'Male'}, inplace=True)
          df_test['Male'] = np.where(df_test['Male']=='Male',1,0)
```

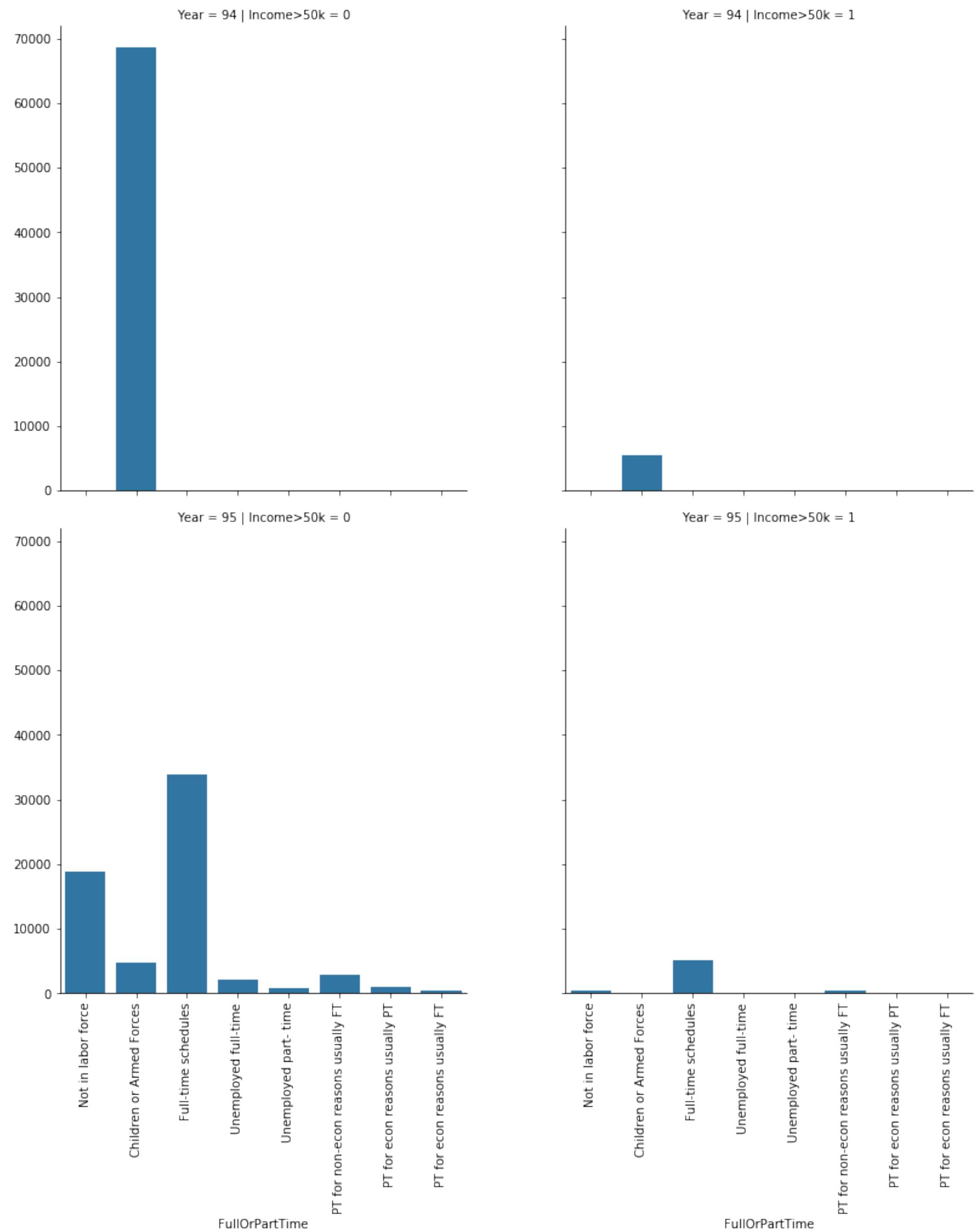
FullOrPartTime

```
In [523]: df['FullOrPartTime'].value_counts()
```

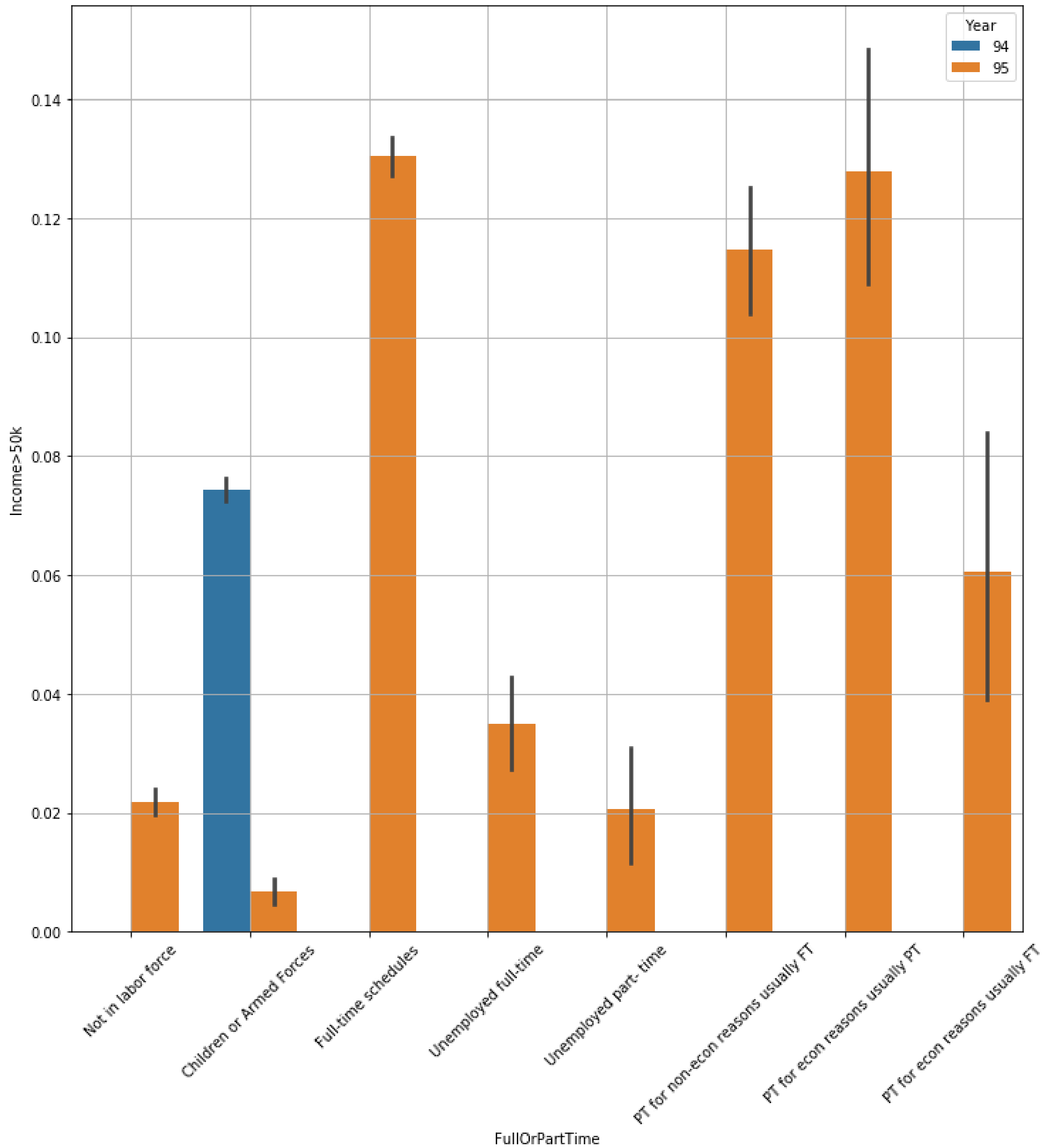
```
Out[523]: Children or Armed Forces              78981
          Full-time schedules                   38956
          Not in labor force                    19201
          PT for non-econ reasons usually FT     3210
          Unemployed full-time                   2200
          PT for econ reasons usually PT         1166
```

```
Unemployed part- time                     779
PT for econ reasons usually FT            512
Name: FullOrPartTime, dtype: int64
```

In [524]:
```
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'FullOrPartTime', order=df['FullOrPartTime'].unique() )
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```



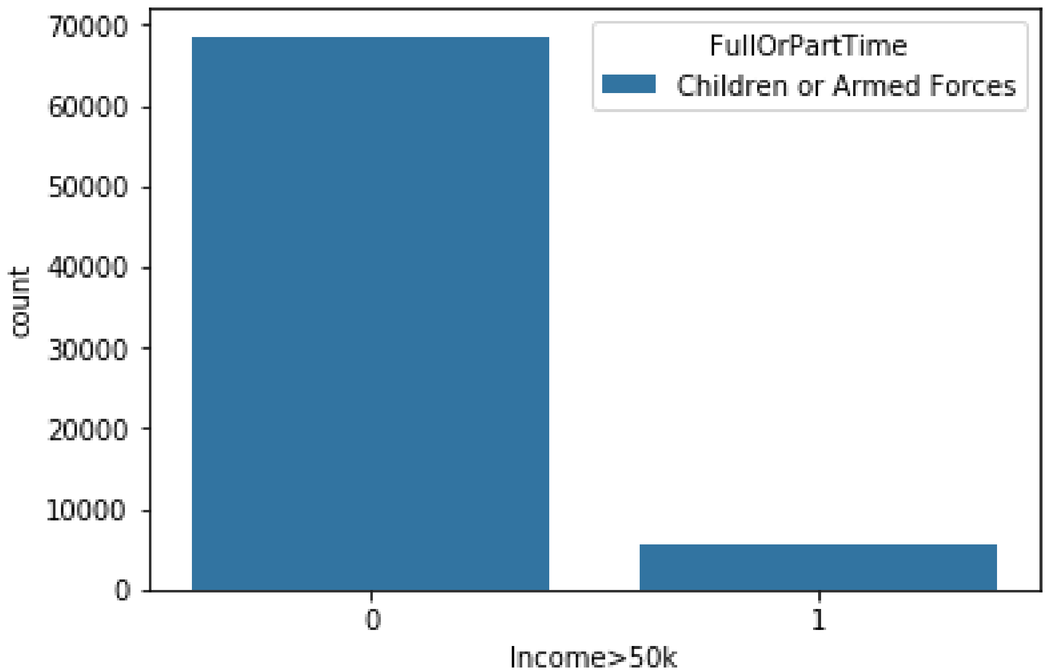

```
In [525]: plt.figure(figsize=(12,12))
          sns.barplot(x='FullOrPartTime', y='Income>50k', data=df,  hue='Year', dodge=True)
          plt.xticks(rotation=45)
          plt.grid(True)
```



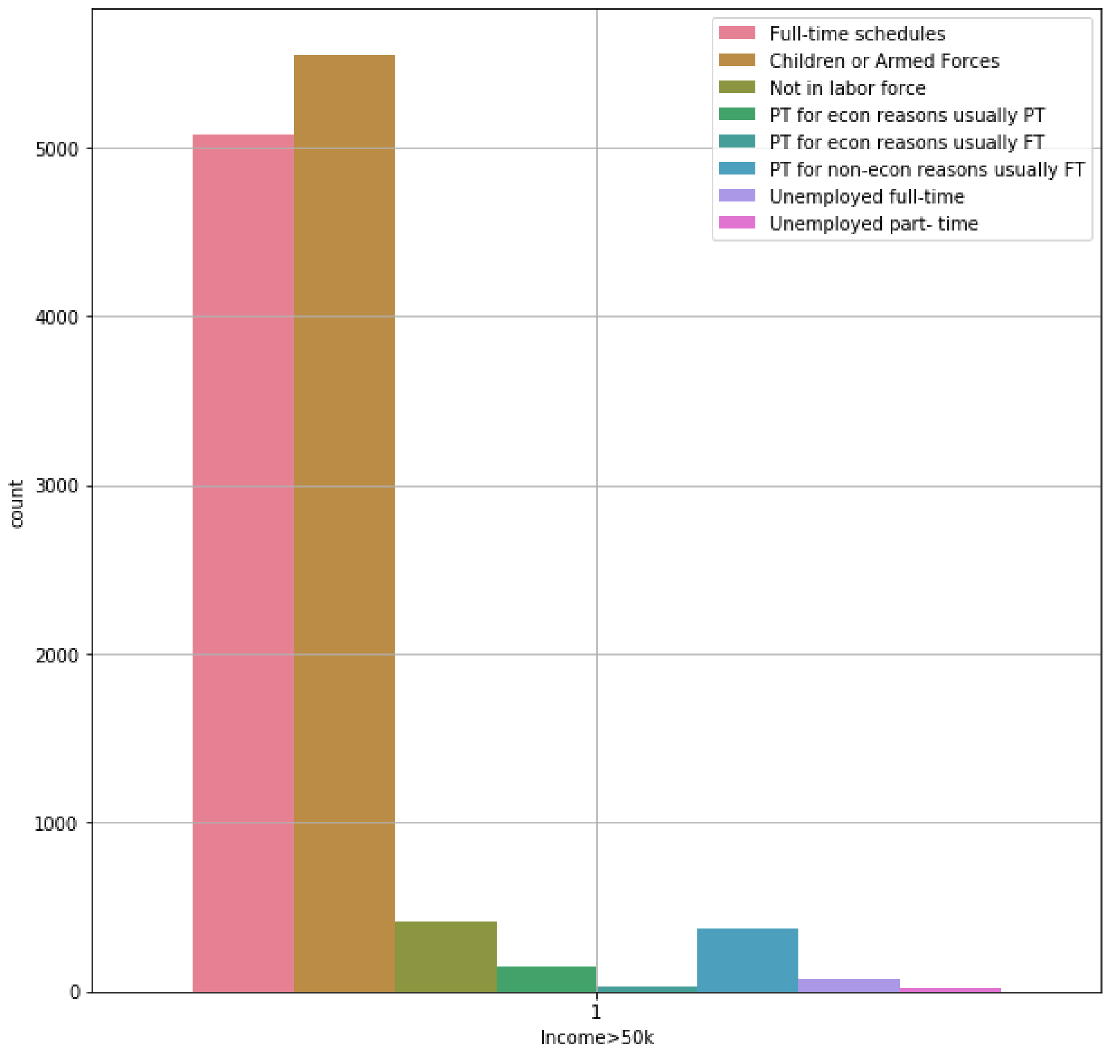
Looks like the data for this column is missing for 1994. It doesn't make sense that all those who answered the survey in 1994 are of the same type

```
In [526]: sns.countplot(x='Income>50k', hue='FullOrPartTime', data=df[df['Year']==94])
```

```
Out[526]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4fb44ac8>
```




```
In [527]: plt.figure(figsize=(10,10))
          sns.countplot(x='Income>50k', hue='FullOrPartTime', data=df[df['Income>50k']==1],pale
          plt.grid()
          plt.legend(fontsize='medium')

Out[527]: <matplotlib.legend.Legend at 0x12d4fb46da0>
```



Even though this columns seems to be correlated with Income levels, but I will have to drop it as I cannot impute the values of 1994

```
In [528]: df.drop('FullOrPartTime', axis=1, inplace=True)
          df_test.drop('FullOrPartTime', axis=1, inplace=True)
```

CapitalGains & CapitalLosses

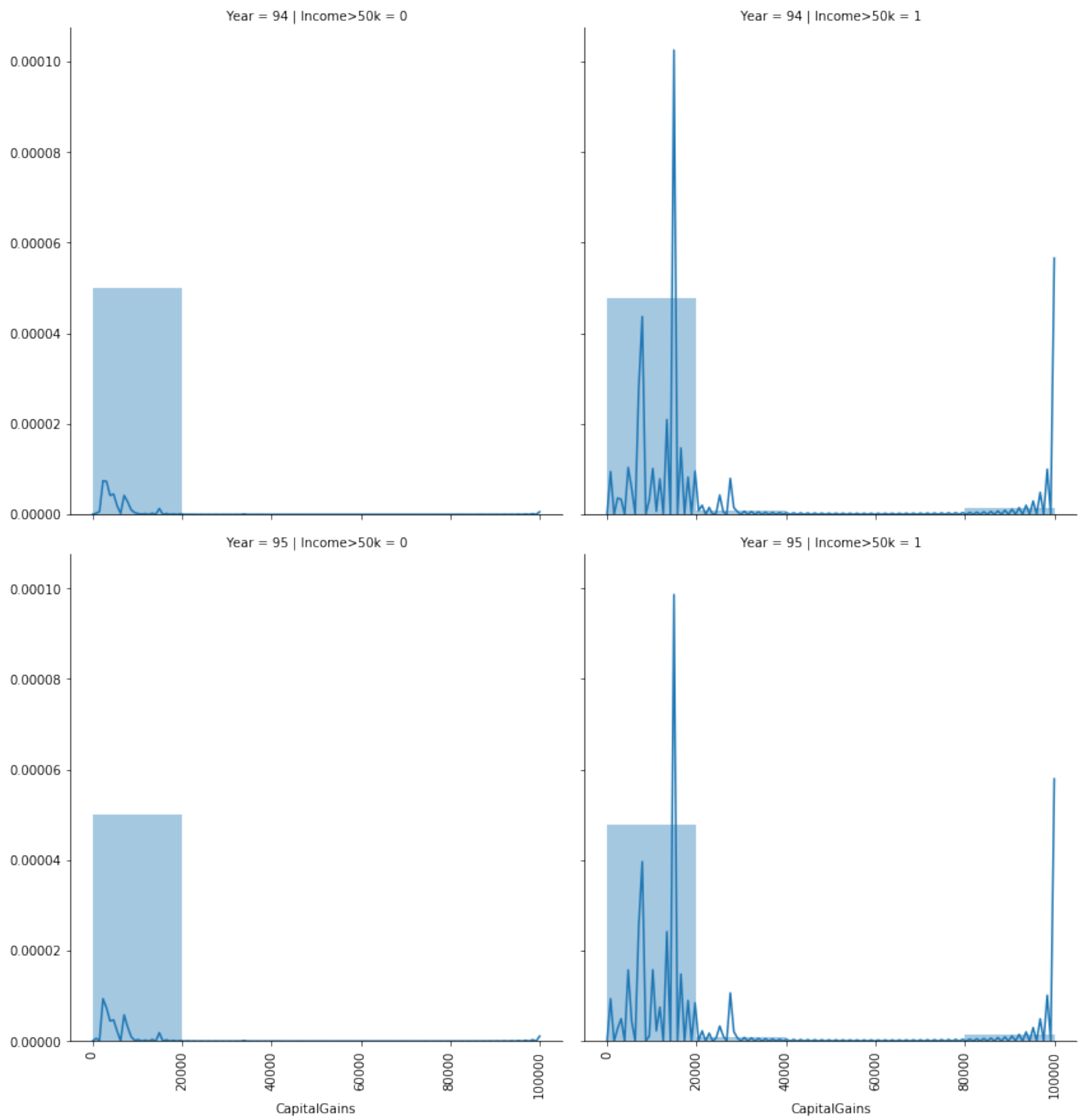
```
In [529]: df['CapitalGains'].describe()
```

```
Out[529]: count    145005.000000
          mean        557.842136
          std        5286.552736
          min           0.000000
          25%           0.000000
          50%           0.000000
```

```
75%              0.000000
max          99999.000000
Name: CapitalGains, dtype: float64
```

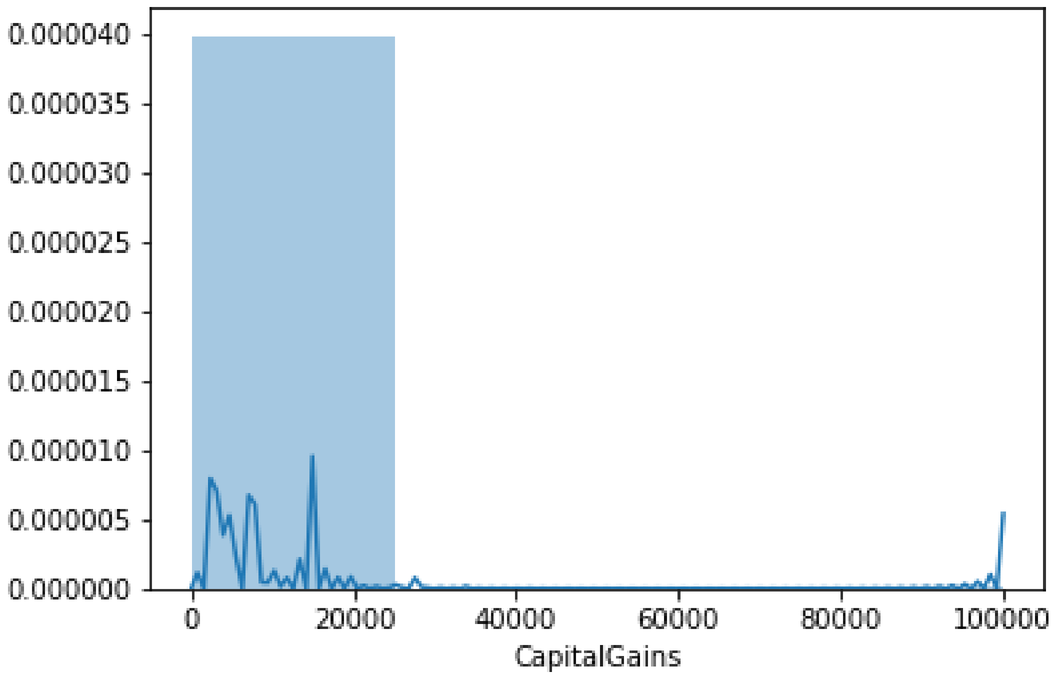
```
In [530]: plt.figure(figsize=(10,10))
          g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
          g.map(sns.distplot, 'CapitalGains', bins=5)
          for ax in g.axes.flat:
              labels = ax.get_xticklabels() # get x labels
              ax.set_xticklabels(labels, rotation=90) # set new labels
```

```
<Figure size 720x720 with 0 Axes>
```



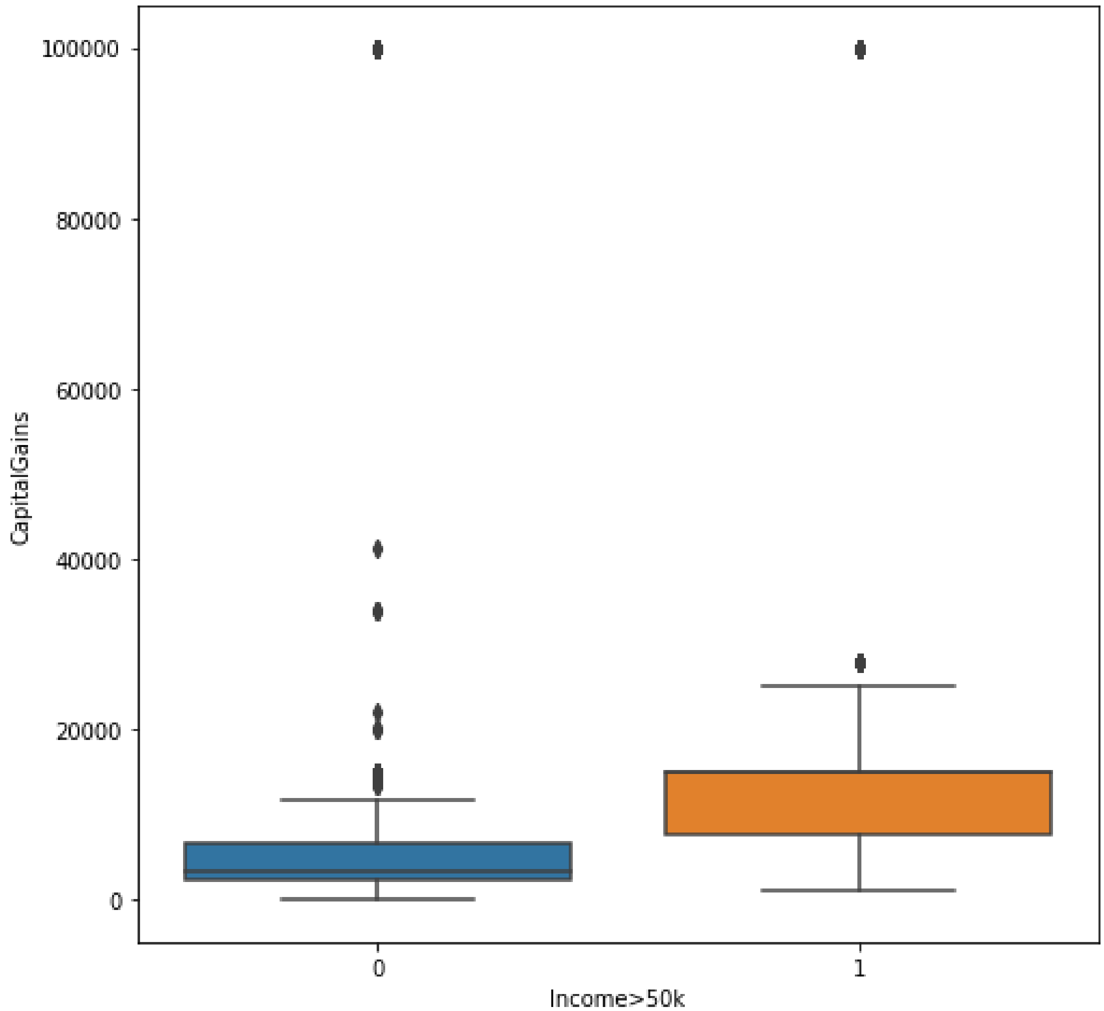
```
In [531]: sns.distplot(df['CapitalGains'], bins=4)
```

```
Out[531]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4f0c29b0>
```




```
In [532]: plt.figure(figsize=(8,8))
          sns.boxplot(y='CapitalGains', x='Income>50k', data=df[df['CapitalGains']>0])
```

```
Out[532]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4f0af390>
```



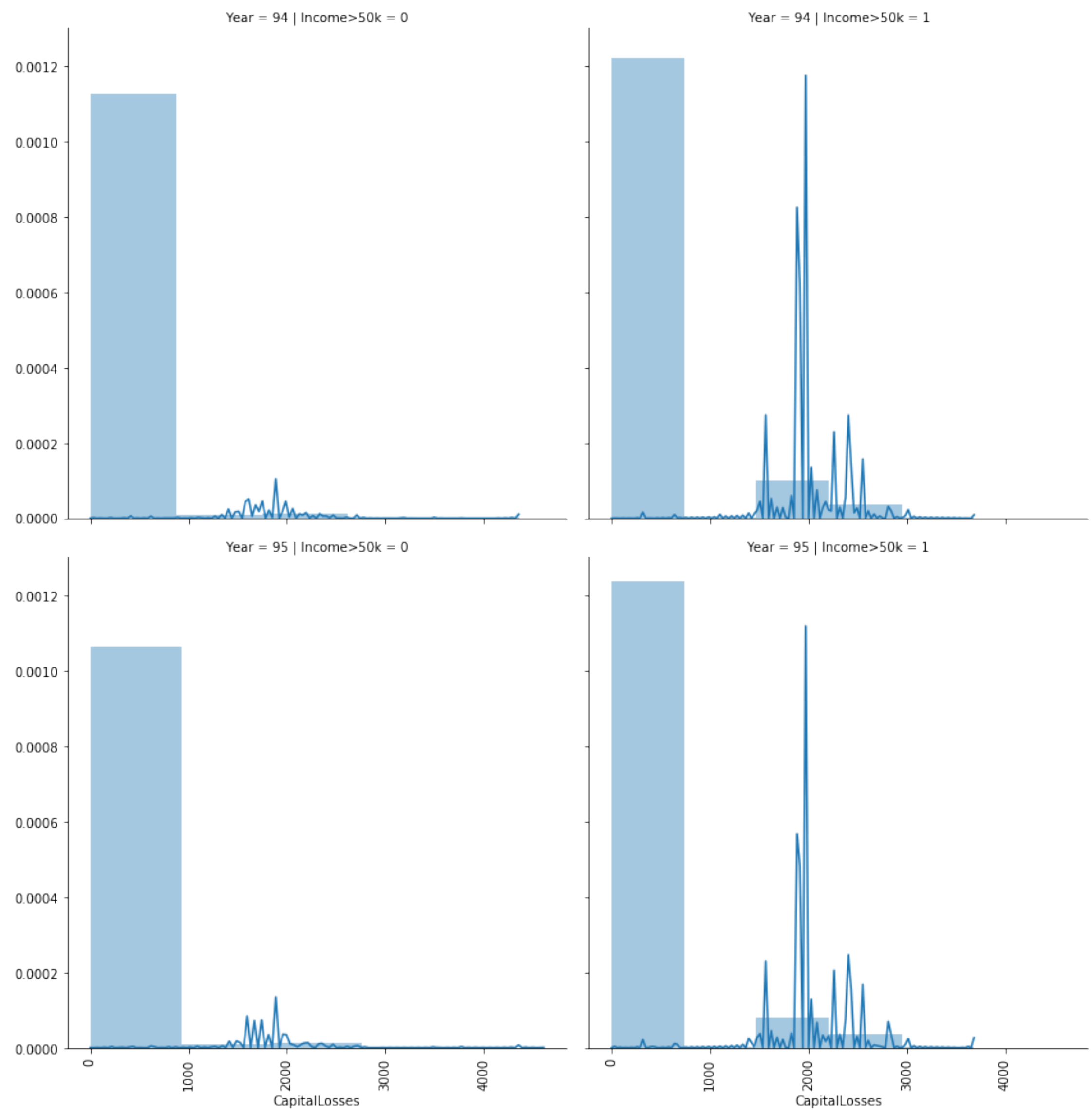
In [533]: df['CapitalLosses'].describe()

Out[533]:
```
count    145005.000000
mean         48.684659
std         308.852831
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max        4608.000000
Name: CapitalLosses, dtype: float64
```

In [534]:
```
plt.figure(figsize=(10,10))
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.distplot, 'CapitalLosses', bins=5)
for ax in g.axes.flat:
```

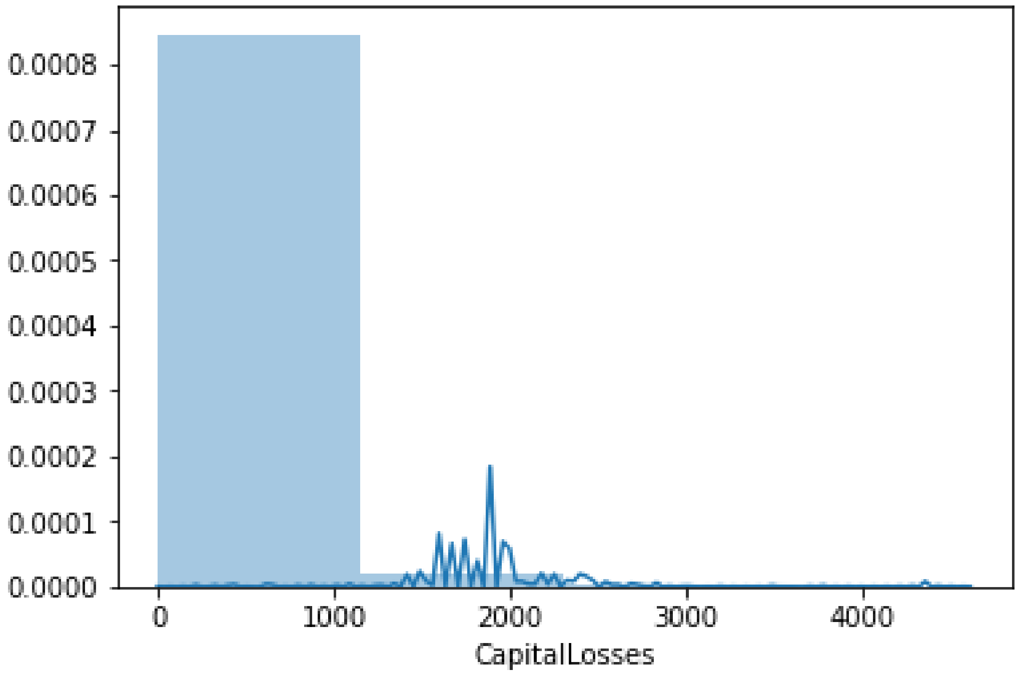
```
labels = ax.get_xticklabels() # get x labels
ax.set_xticklabels(labels, rotation=90) # set new labels
```

```
<Figure size 720x720 with 0 Axes>
```



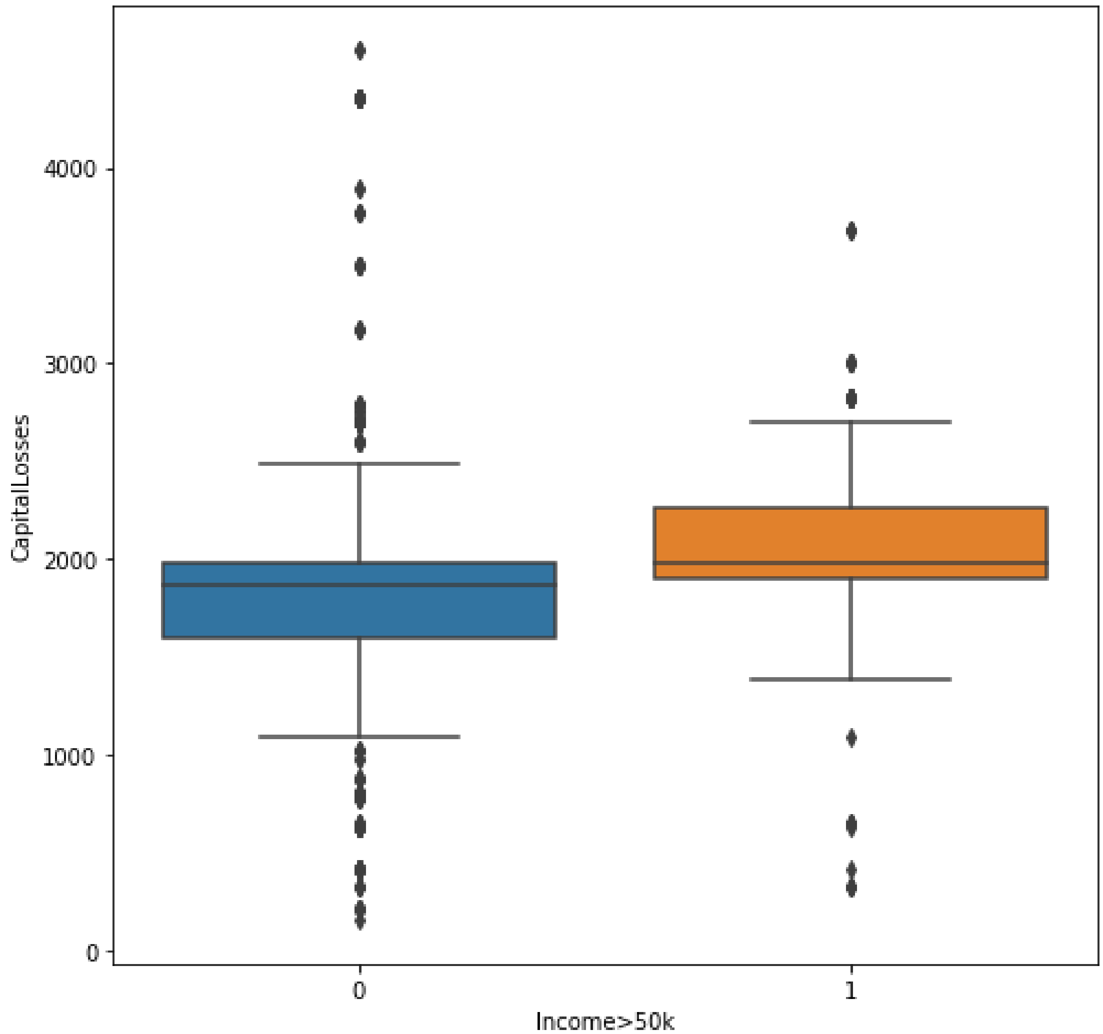

```
In [535]: sns.distplot(df['CapitalLosses'], bins=4)
```

```
Out[535]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4ebc07f0>
```

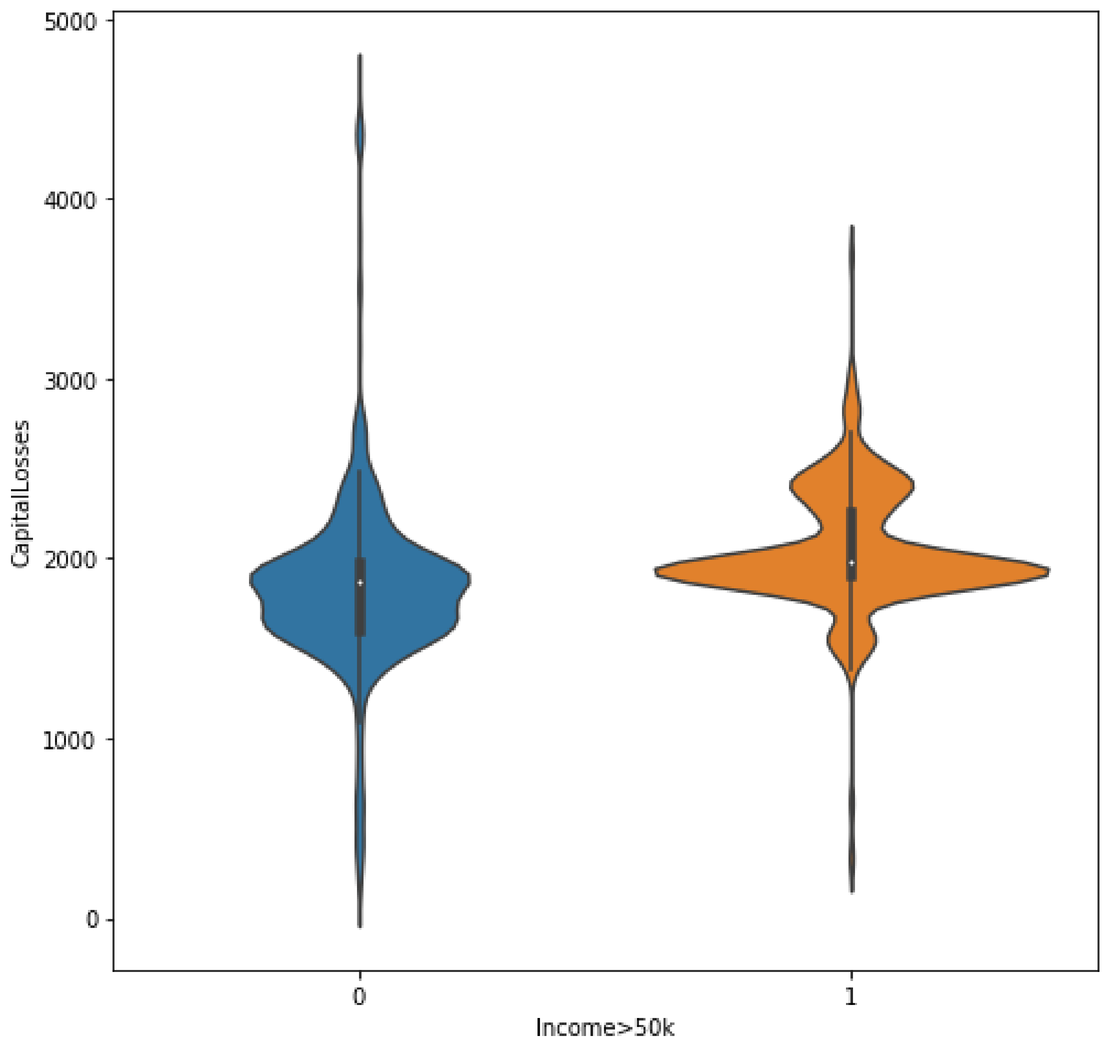


In [536]:
```
plt.figure(figsize=(8,8))
sns.boxplot(y='CapitalLosses', x='Income>50k', data=df[df['CapitalLosses']>0])
```

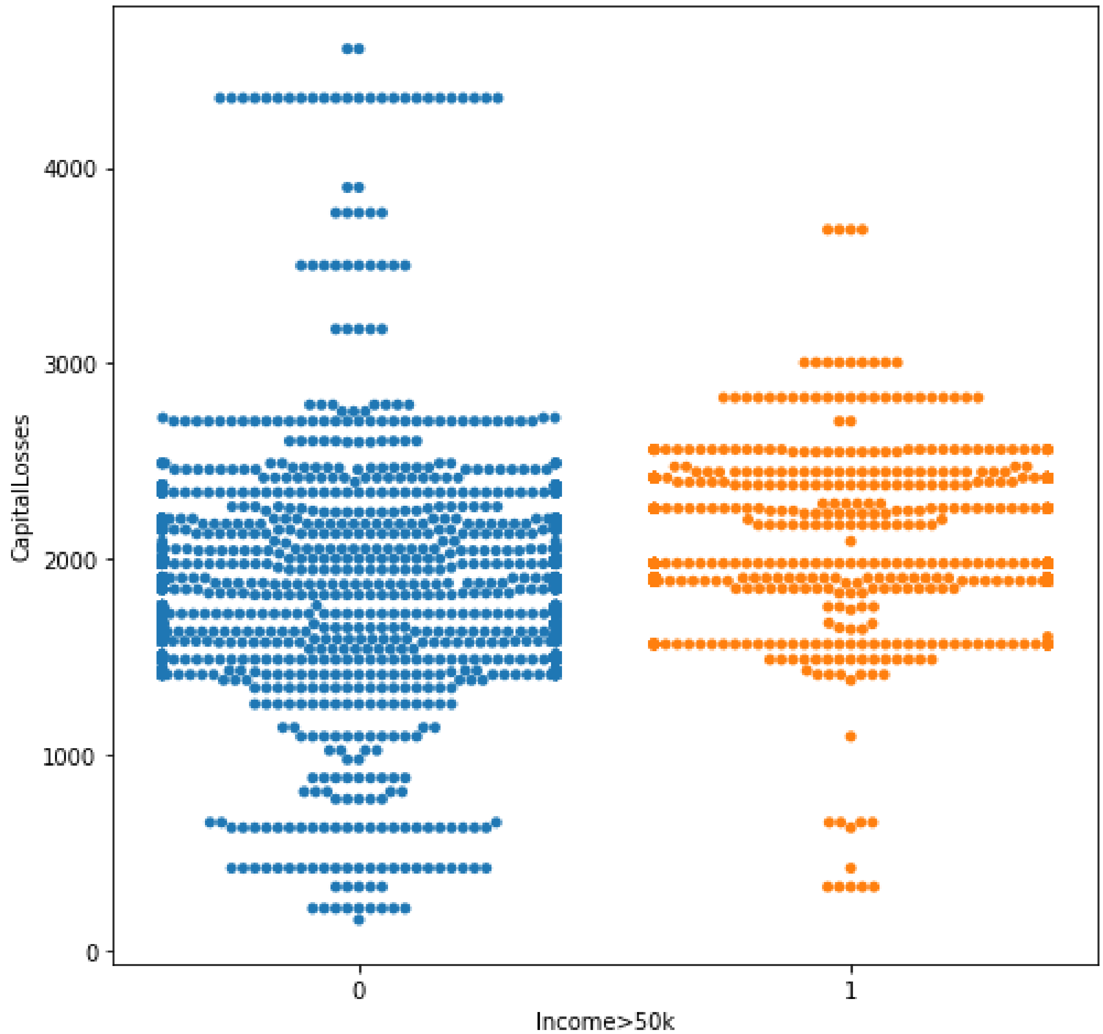
Out[536]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12d4ebcc860>
```




```
In [537]: plt.figure(figsize=(8,8))
          sns.violinplot(y='CapitalLosses', x='Income>50k', data=df[df['CapitalLosses']>0])

Out[537]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4ebc42e8>
```




```
In [538]: plt.figure(figsize=(8,8))
          sns.swarmplot(y='CapitalLosses', x='Income>50k', data=df[df['CapitalLosses']>0])

Out[538]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4eb964a8>
```



This is unexpected, since higher income should probably mean lower Capital Losses.
StockDividends

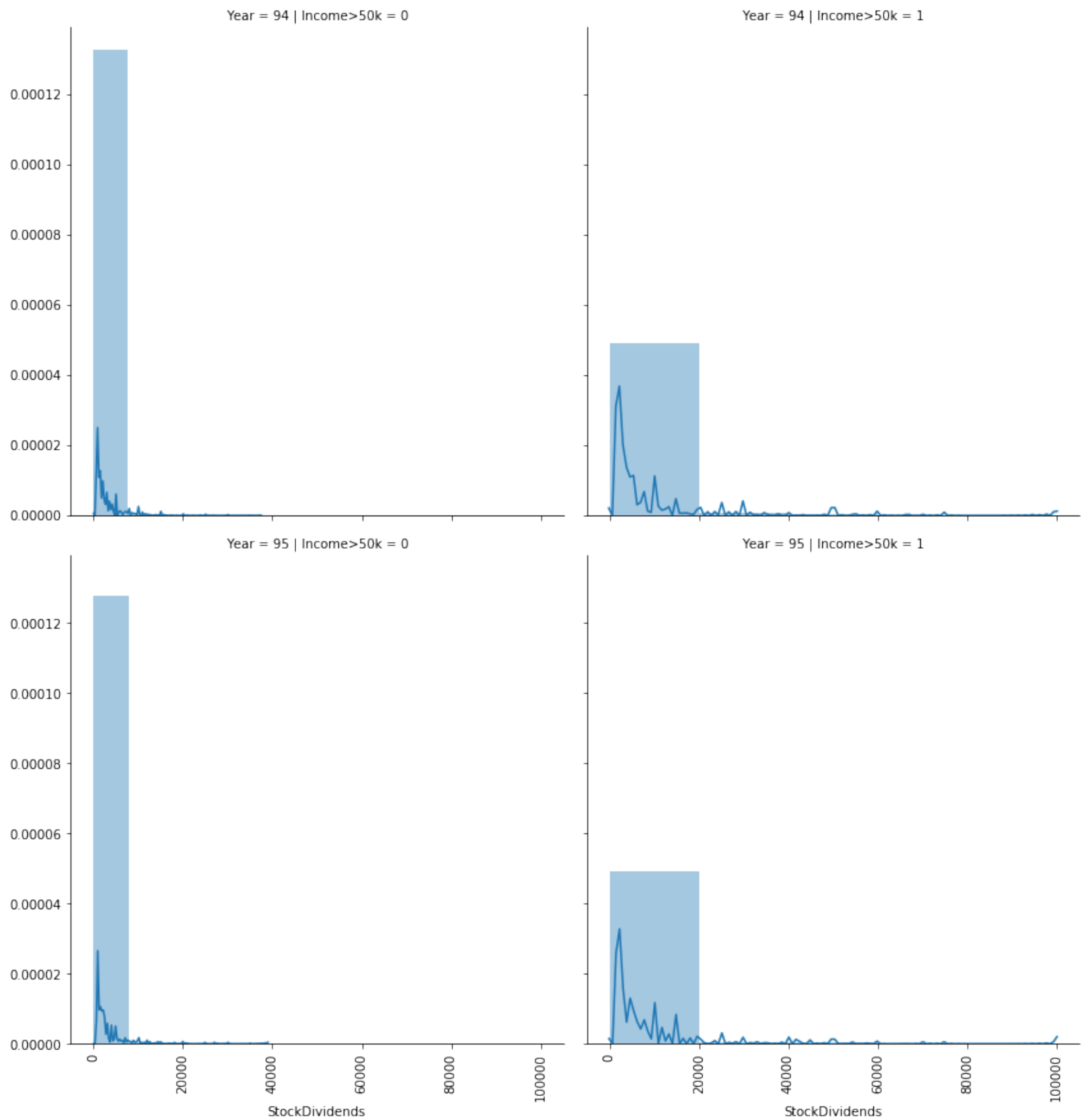
In [539]:
```
df['StockDividends'].describe()
```

Out[539]:
```
count    145005.000000
mean        252.061688
std        2260.621084
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max       99999.000000
Name: StockDividends, dtype: float64
```

In [540]:
```
plt.figure(figsize=(10,10))
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
```

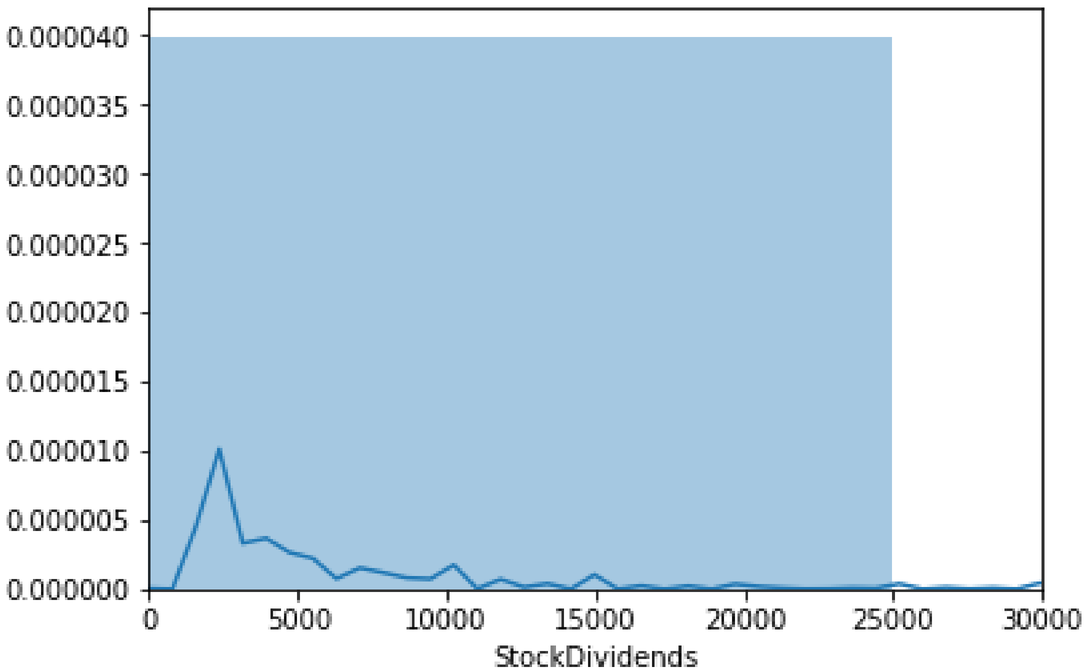
```
g.map(sns.distplot, 'StockDividends', bins=5)
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```

```
&lt;Figure size 720x720 with 0 Axes&gt;
```

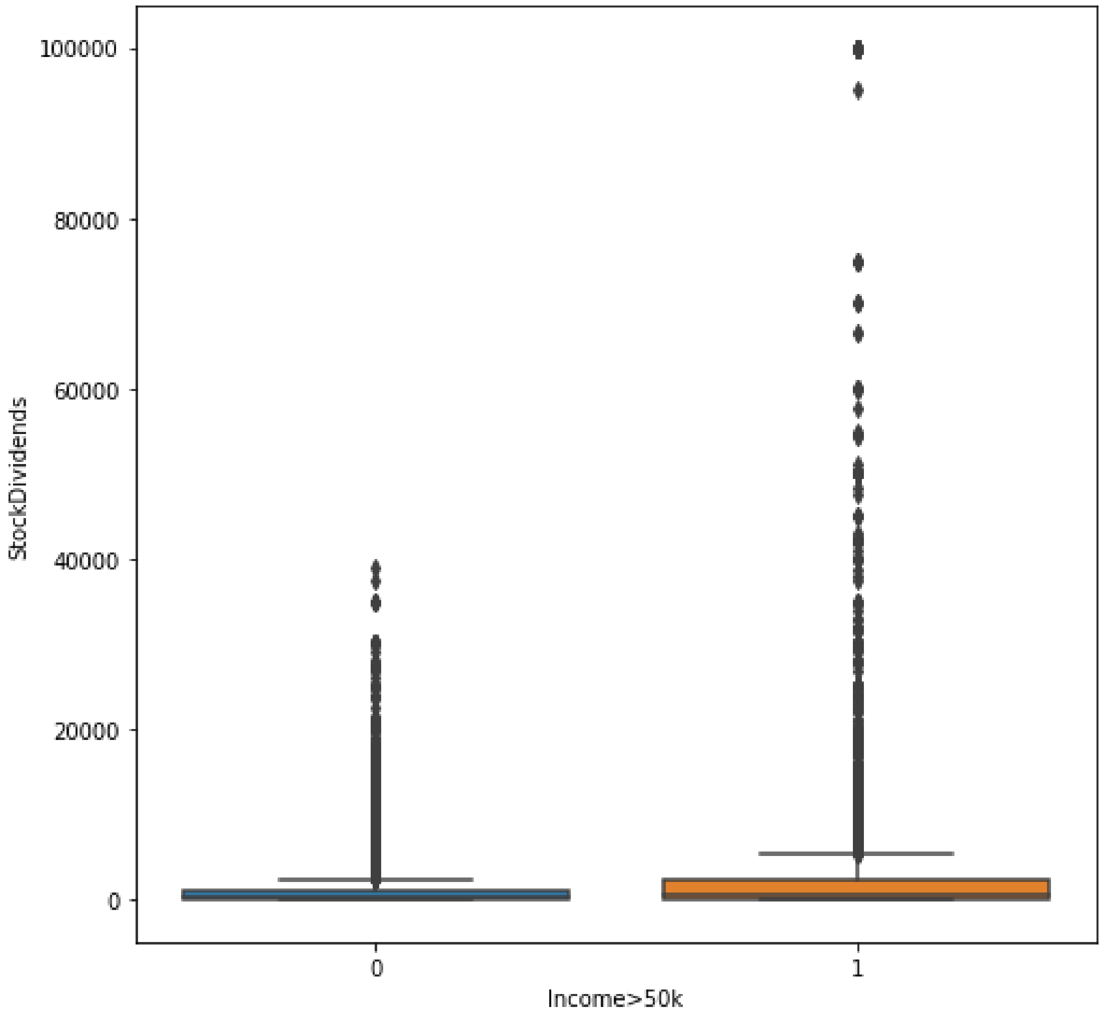



```
In [541]: sns.distplot(df['StockDividends'], bins=4)
          plt.xlim(0,30000)
```

```
Out[541]: (0, 30000)
```




```
In [542]: plt.figure(figsize=(8,8))
          sns.boxplot(y='StockDividends', x='Income>50k', data=df[df['StockDividends']>0])

Out[542]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4fb36cf8>
```

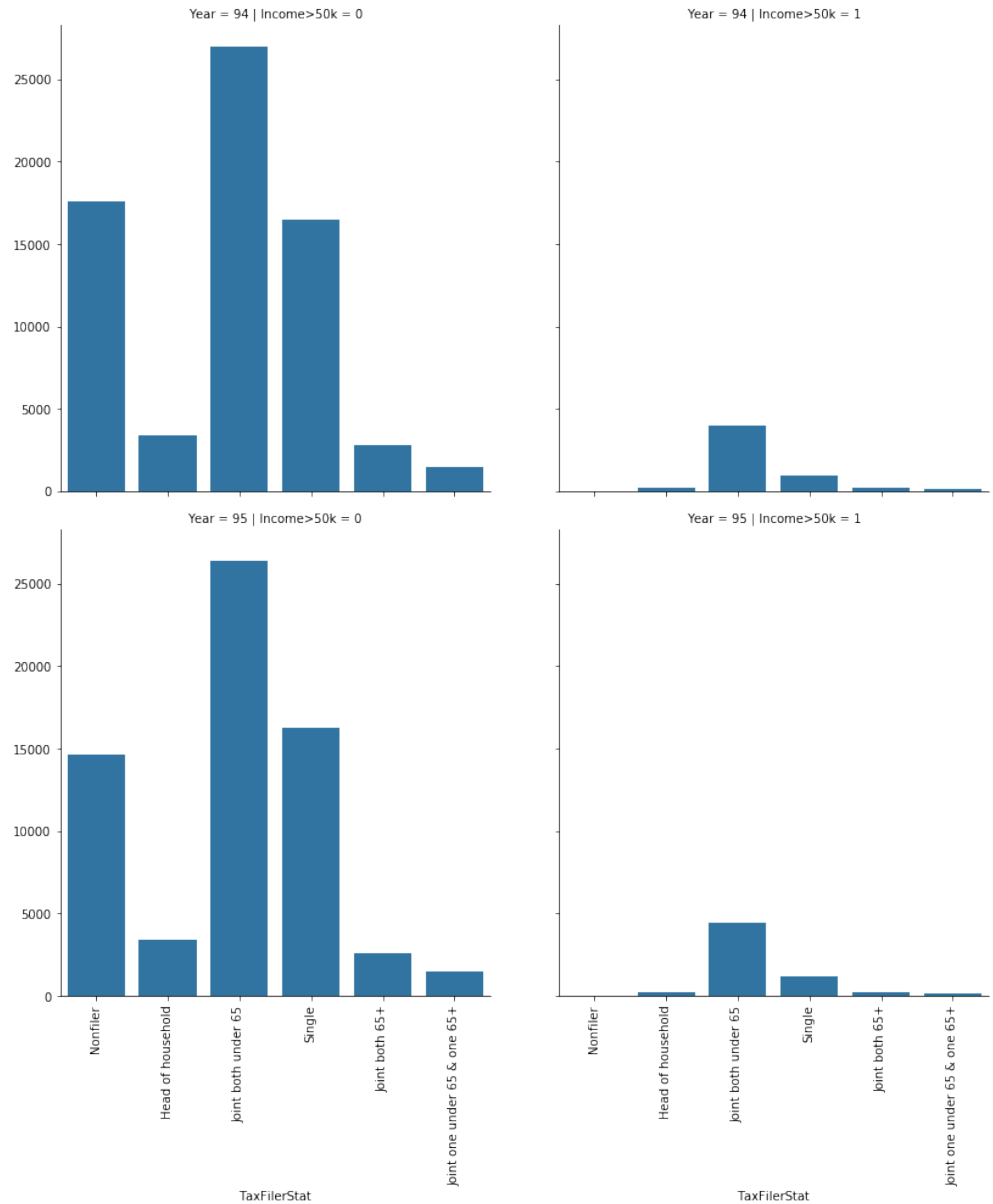


TaxFilerStat

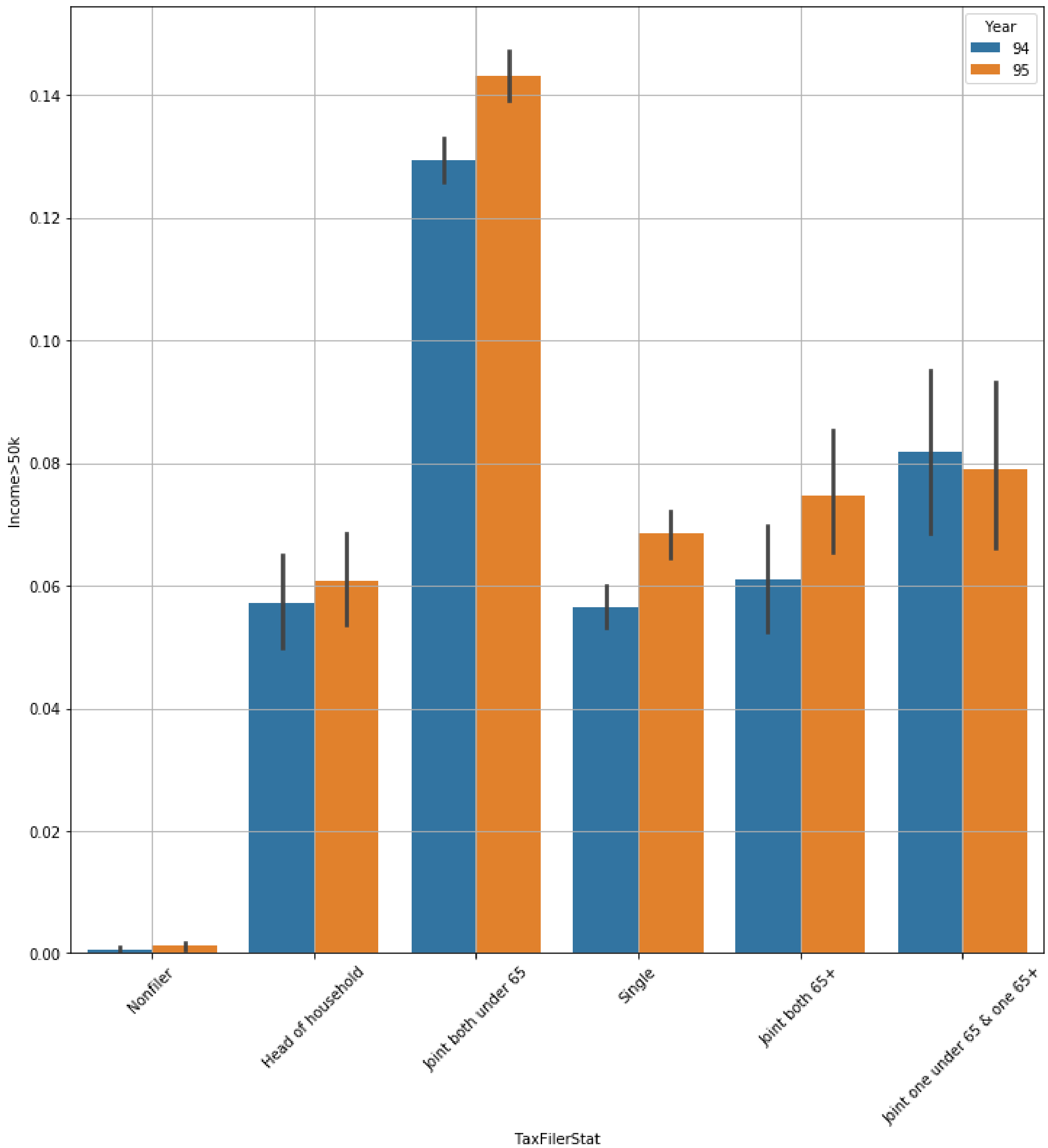
```
In [543]: df['TaxFilerStat'].describe()

Out[543]: count                    145005
          unique                        6
          top        Joint both under 65
          freq                      61723
          Name: TaxFilerStat, dtype: object

In [544]: df['TaxFilerStat'].value_counts()

Out[544]: Joint both under 65            61723
          Single                         34850
          Nonfiler                       32260
          Head of household               7195
          Joint both 65+                  5767
```

```
Joint one under 65 & one 65+      3210
Name: TaxFilerStat, dtype: int64
```

In [545]:
```python
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'TaxFilerStat', order=df['TaxFilerStat'].unique() )
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```

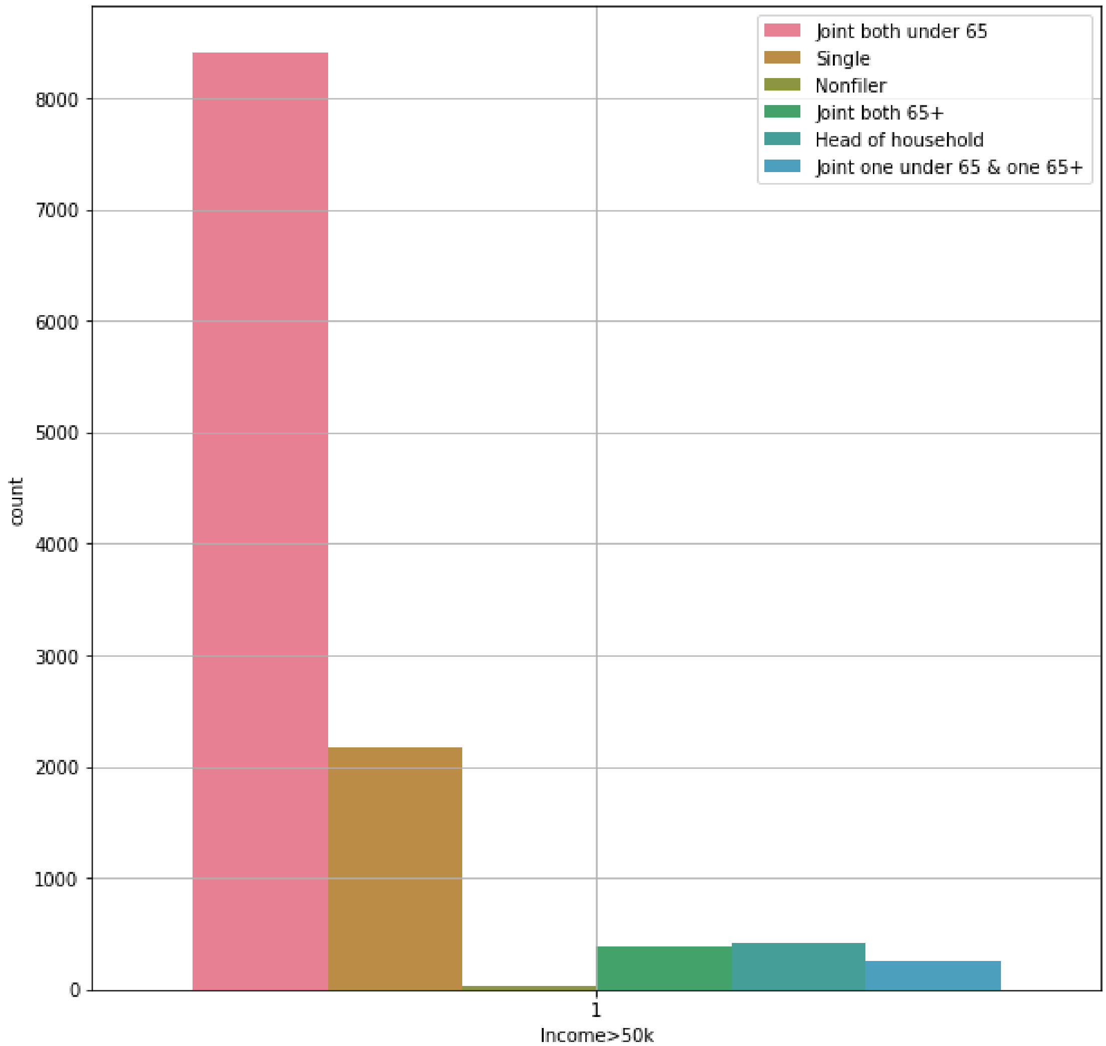


```
In [546]: plt.figure(figsize=(12,12))
          sns.barplot(x='TaxFilerStat', y='Income>50k', data=df,  hue='Year', dodge=True)
          plt.xticks(rotation=45)
          plt.grid(True)
```




```
In [547]: plt.figure(figsize=(10,10))
          sns.countplot(x='Income>50k', hue='TaxFilerStat', data=df[df['Income>50k']==1],palet
          plt.grid()
          plt.legend(fontsize='medium')
```

Out[547]: <matplotlib.legend.Legend at 0x12d4fb514e0>



In [548]:
```
#Group the values: ['Head of household', 'Joint both 65+', 'Joint one under 65 & one
df['TaxFilerStat'] = df['TaxFilerStat'].apply(lambda x: 'Other' if x in ['Head of hou
df_test['TaxFilerStat'] = df_test['TaxFilerStat'].apply(lambda x: 'Other' if x in ['H
```

HouseholdFamilyStatus

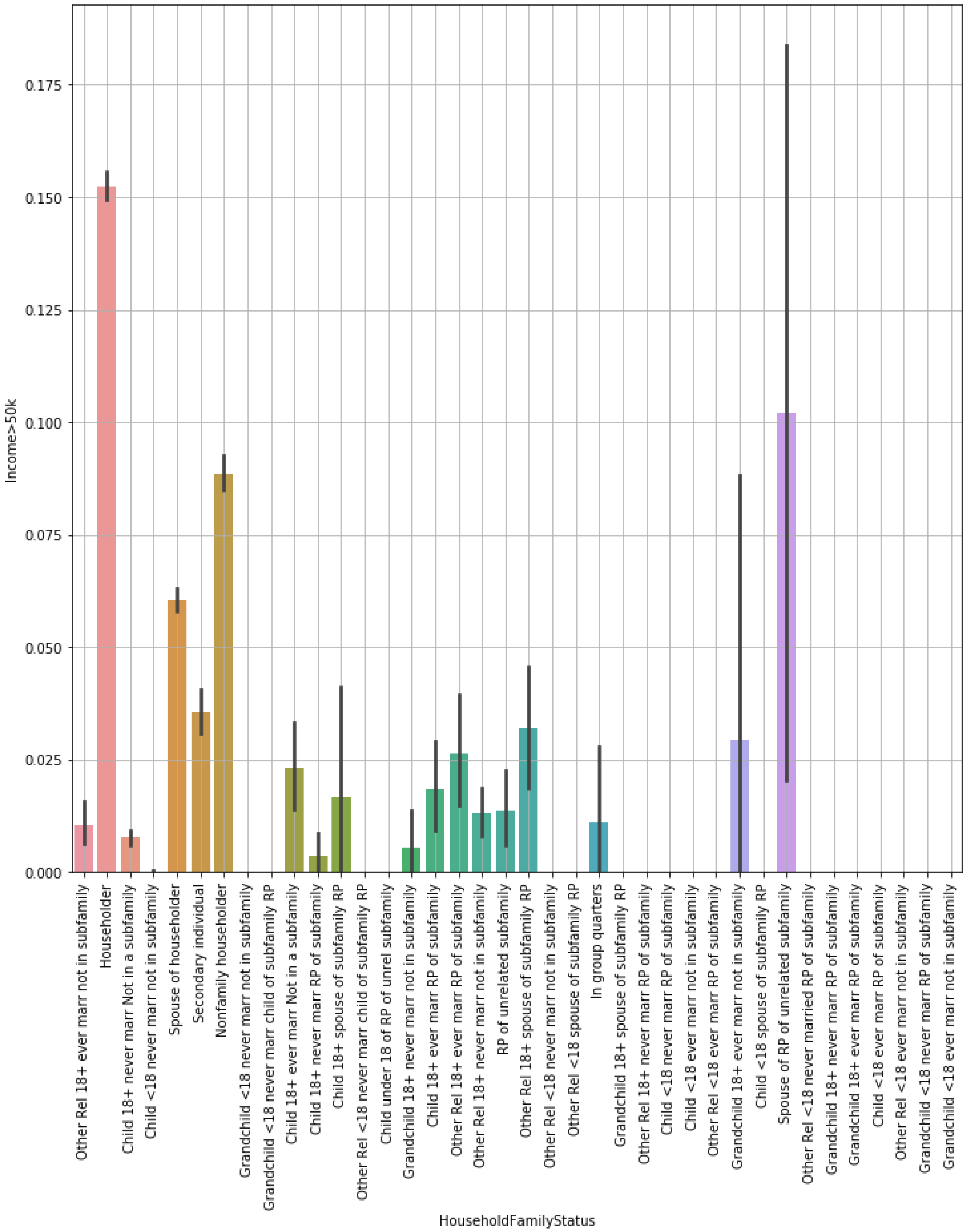
In [549]:
```
df['HouseholdFamilyStatus'].describe()
```

Out[549]:
```
count          145005
unique             38
top       Householder
freq            48707
Name: HouseholdFamilyStatus, dtype: object
```

In [550]: df['HouseholdFamilyStatus'].value_counts()

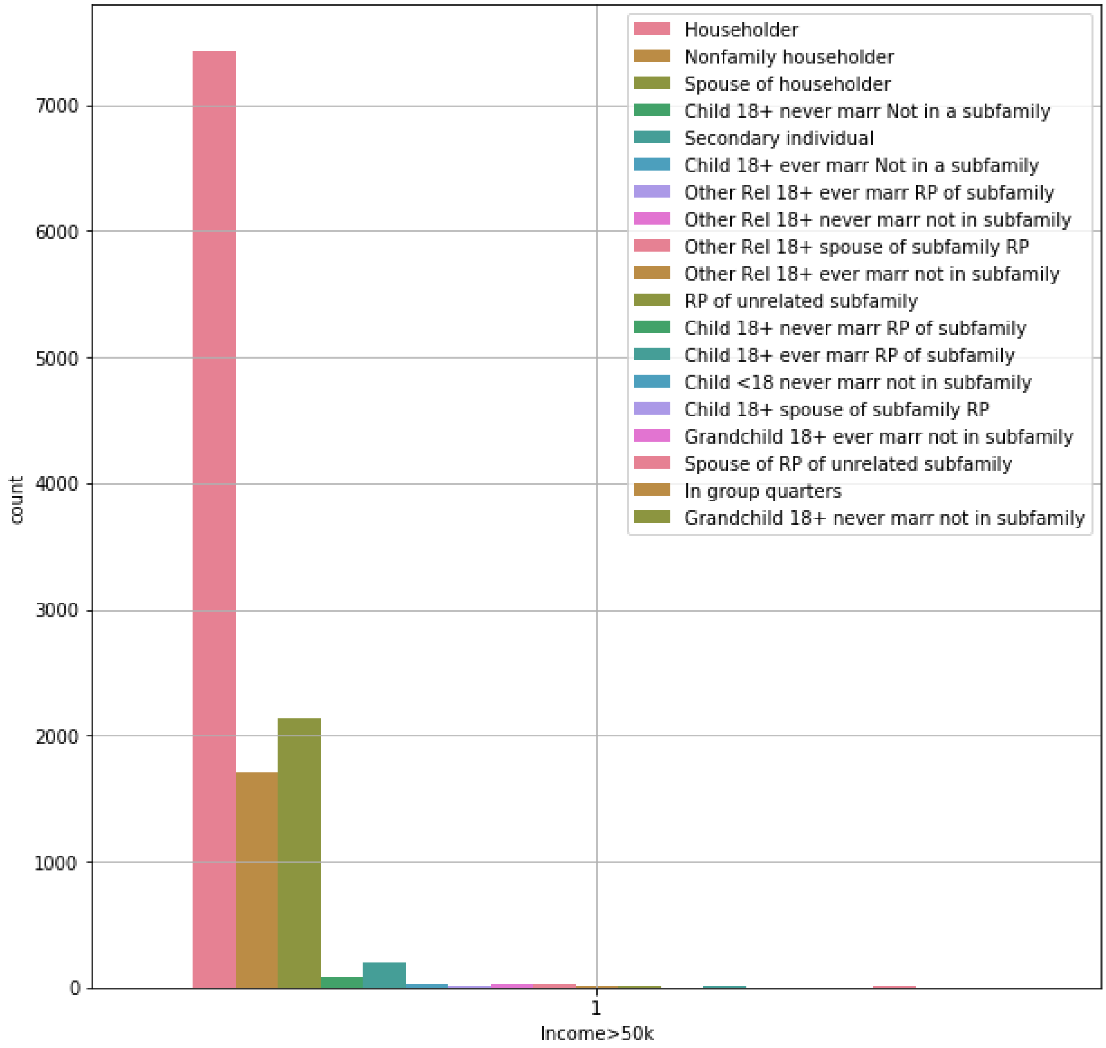
```
Out[550]: Householder                                       48707
          Spouse of householder                             35347
          Nonfamily householder                             19255
          Child <18 never marr not in subfamily             14015
          Child 18+ never marr Not in a subfamily           10969
          Secondary individual                               5711
          Other Rel 18+ ever marr not in subfamily           1709
          Other Rel 18+ never marr not in subfamily          1611
          Child 18+ ever marr Not in a subfamily              991
          Grandchild <18 never marr child of subfamily RP     799
          RP of unrelated subfamily                           663
          Child 18+ ever marr RP of subfamily                 652
          Other Rel 18+ ever marr RP of subfamily             609
          Other Rel 18+ spouse of subfamily RP                591
          Child 18+ never marr RP of subfamily                569
          Other Rel <18 never marr child of subfamily RP      515
          Grandchild <18 never marr not in subfamily          485
          Child under 18 of RP of unrel subfamily             412
          Other Rel <18 never marr not in subfamily           391
          Grandchild 18+ never marr not in subfamily          364
          In group quarters                                   179
          Child 18+ spouse of subfamily RP                    121
          Other Rel 18+ never marr RP of subfamily             92
          Child <18 never marr RP of subfamily                 76
          Spouse of RP of unrelated subfamily                  49
          Child <18 ever marr not in subfamily                 35
          Grandchild 18+ ever marr not in subfamily            34
          Grandchild 18+ spouse of subfamily RP                10
          Grandchild 18+ ever marr RP of subfamily              9
          Child <18 ever marr RP of subfamily                   9
          Grandchild 18+ never marr RP of subfamily             6
          Other Rel <18 ever marr RP of subfamily               6
          Other Rel <18 never married RP of subfamily           4
          Other Rel <18 spouse of subfamily RP                  3
          Child <18 spouse of subfamily RP                      2
          Grandchild <18 ever marr not in subfamily             2
          Grandchild <18 never marr RP of subfamily             2
          Other Rel <18 ever marr not in subfamily              1
          Name: HouseholdFamilyStatus, dtype: int64
```

In [551]:
```
plt.figure(figsize=(12,12))
sns.barplot(x='HouseholdFamilyStatus', y='Income>50k', data=df, dodge=True)
plt.xticks(rotation=90)
plt.grid(True)
```

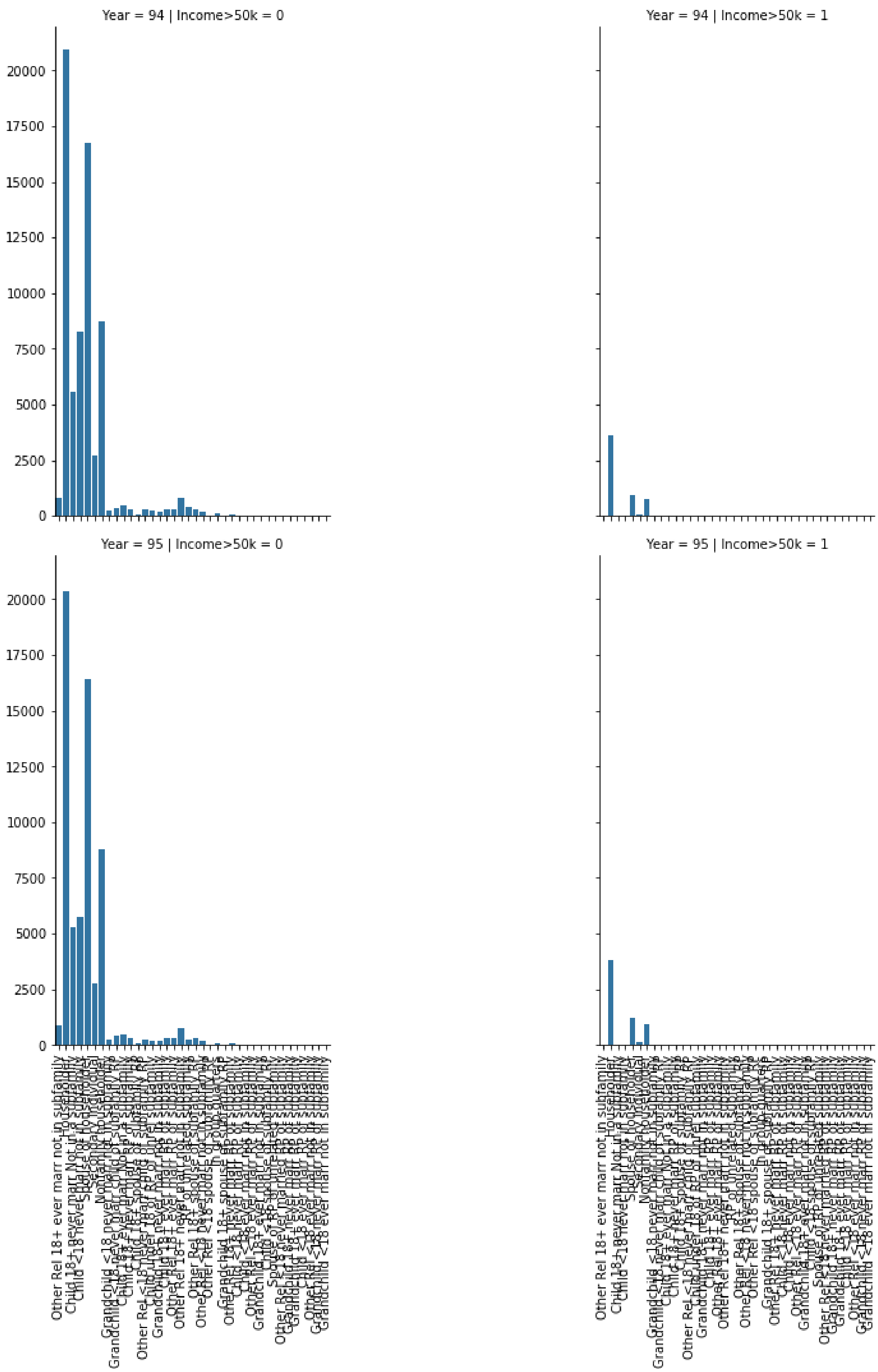



```
In [552]: plt.figure(figsize=(10,10))
          sns.countplot(x='Income>50k', hue='HouseholdFamilyStatus', data=df[df['Income>50k']==
          plt.grid()
          plt.legend(fontsize='medium')
```

Out[552]: <matplotlib.legend.Legend at 0x12d4fb71dd8>



In [553]:
```
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'HouseholdFamilyStatus', order=df['HouseholdFamilyStatus'].uniq
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```


Year = 94 | Income>50k = 0
Year = 94 | Income>50k = 1
Year = 95 | Income>50k = 0
Year = 95 | Income>50k = 1
0
2500
5000
7500
10000
12500
15000
17500
20000
HouseholdFamilyStatus
HouseholdFamilyStatus

In [554]:
```
#To simplify the model, I will replace all small values with Other
df['HouseholdFamilyStatus'] = df['HouseholdFamilyStatus'].apply(lambda x: 'Other' if
df_test['HouseholdFamilyStatus'] = df_test['HouseholdFamilyStatus'].apply(lambda x:
```

In [555]:
```
df['HouseholdFamilyStatus'].value_counts()
```

Out[555]:
```
Householder                 48707
Other                       41696
Spouse of householder       35347
Nonfamily householder       19255
Name: HouseholdFamilyStatus, dtype: int64
```

LiveInHouse1Y

In [556]:
```
df['LiveInHouse1Y'].describe()
```

Out[556]:
```
count                                145005
unique                                    3
top       Not in universe under 1 year old
freq                                  71217
Name: LiveInHouse1Y, dtype: object
```

In [557]:
```
df['LiveInHouse1Y'].value_counts()
```

Out[557]:
```
Not in universe under 1 year old    71217
Yes                                 58972
No                                  14816
Name: LiveInHouse1Y, dtype: int64
```

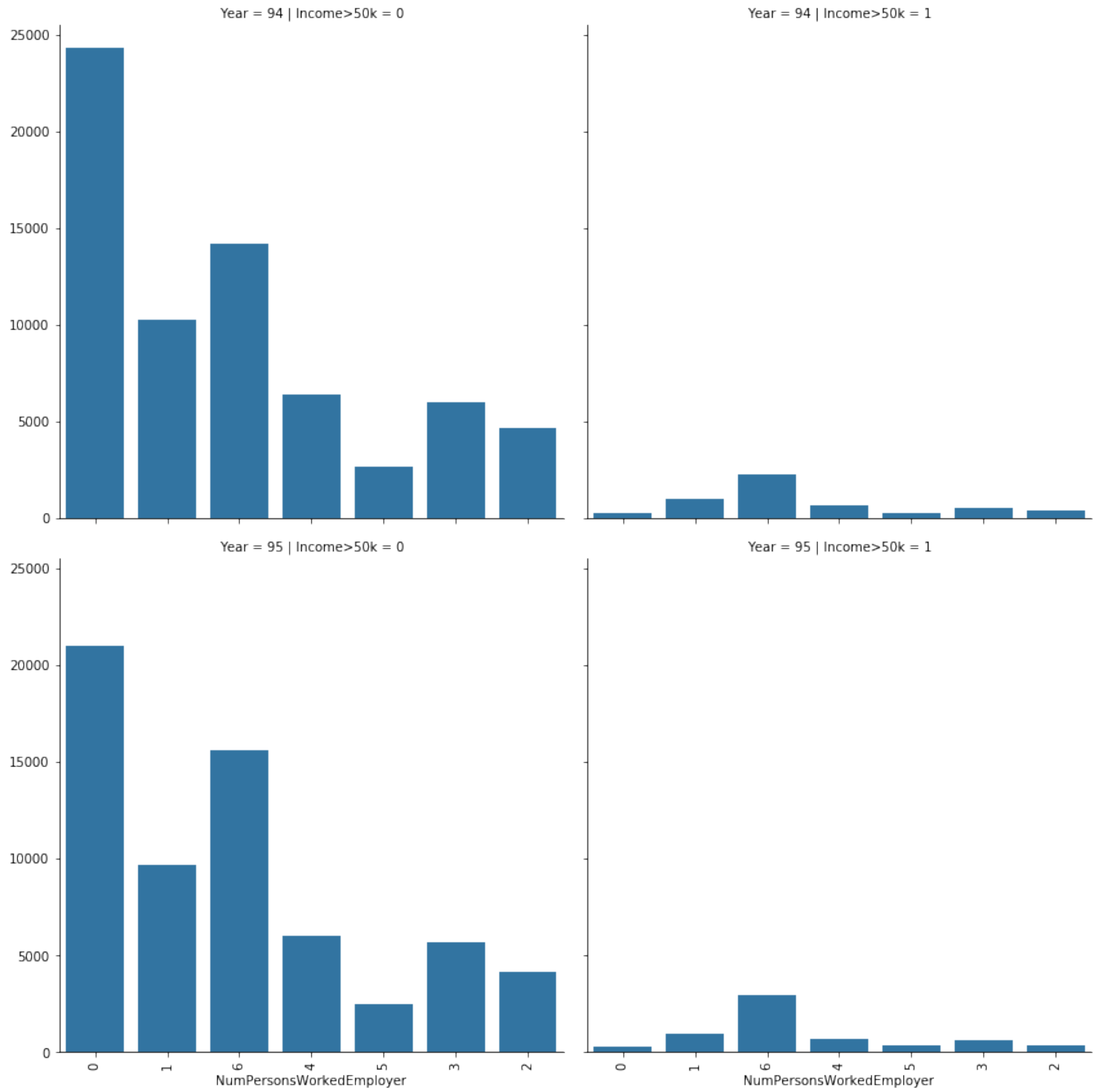
In [558]:
```
#Too much missing info, I will drop this column
df.drop('LiveInHouse1Y', axis=1, inplace=True)
df_test.drop('LiveInHouse1Y', axis=1, inplace=True)
```

NumPersonsWorkedEmployer

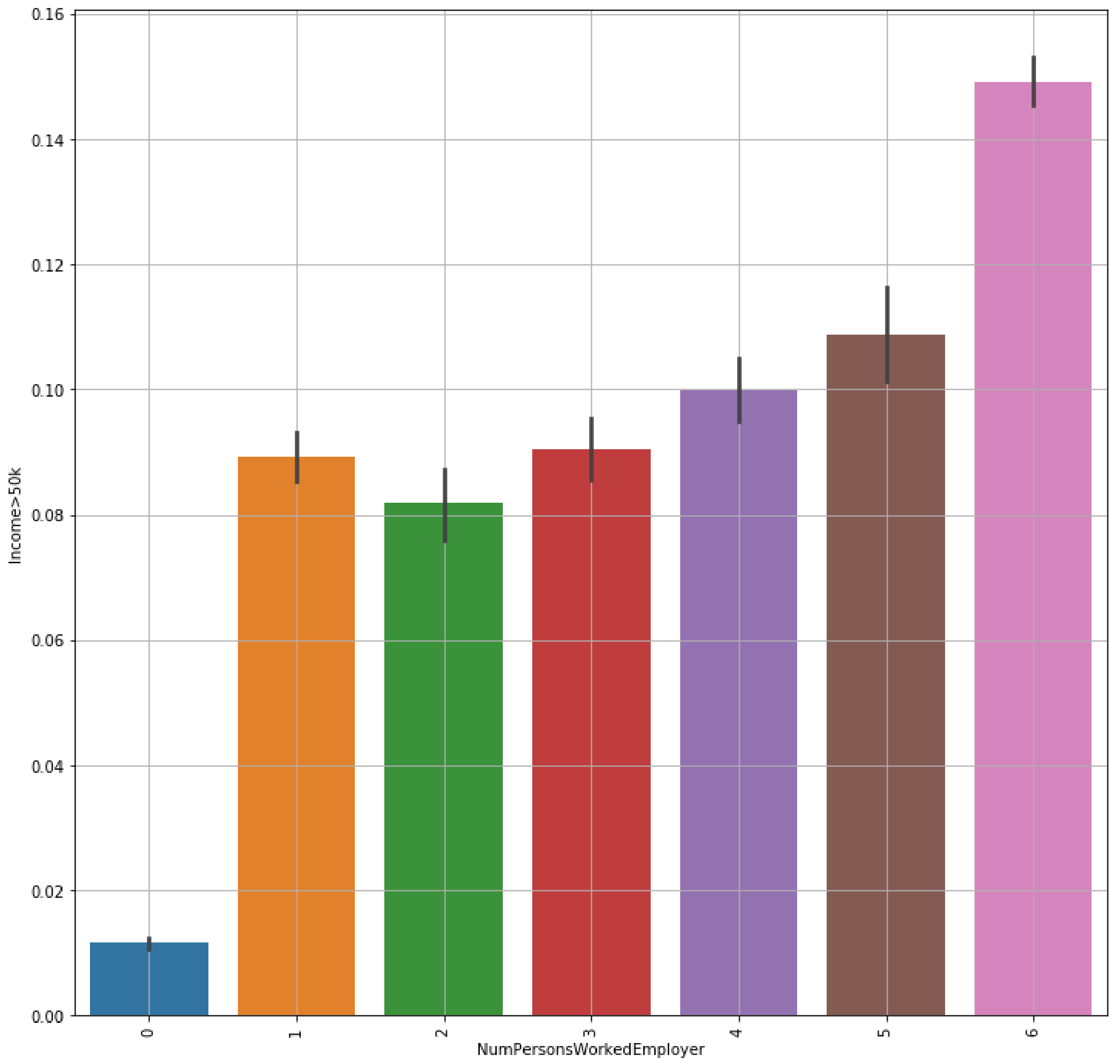
In [559]:
```
df['NumPersonsWorkedEmployer'].describe()
```

Out[559]:
```
count    145005.000000
mean          2.581587
std           2.402695
min           0.000000
25%           0.000000
50%           2.000000
75%           5.000000
max           6.000000
Name: NumPersonsWorkedEmployer, dtype: float64
```

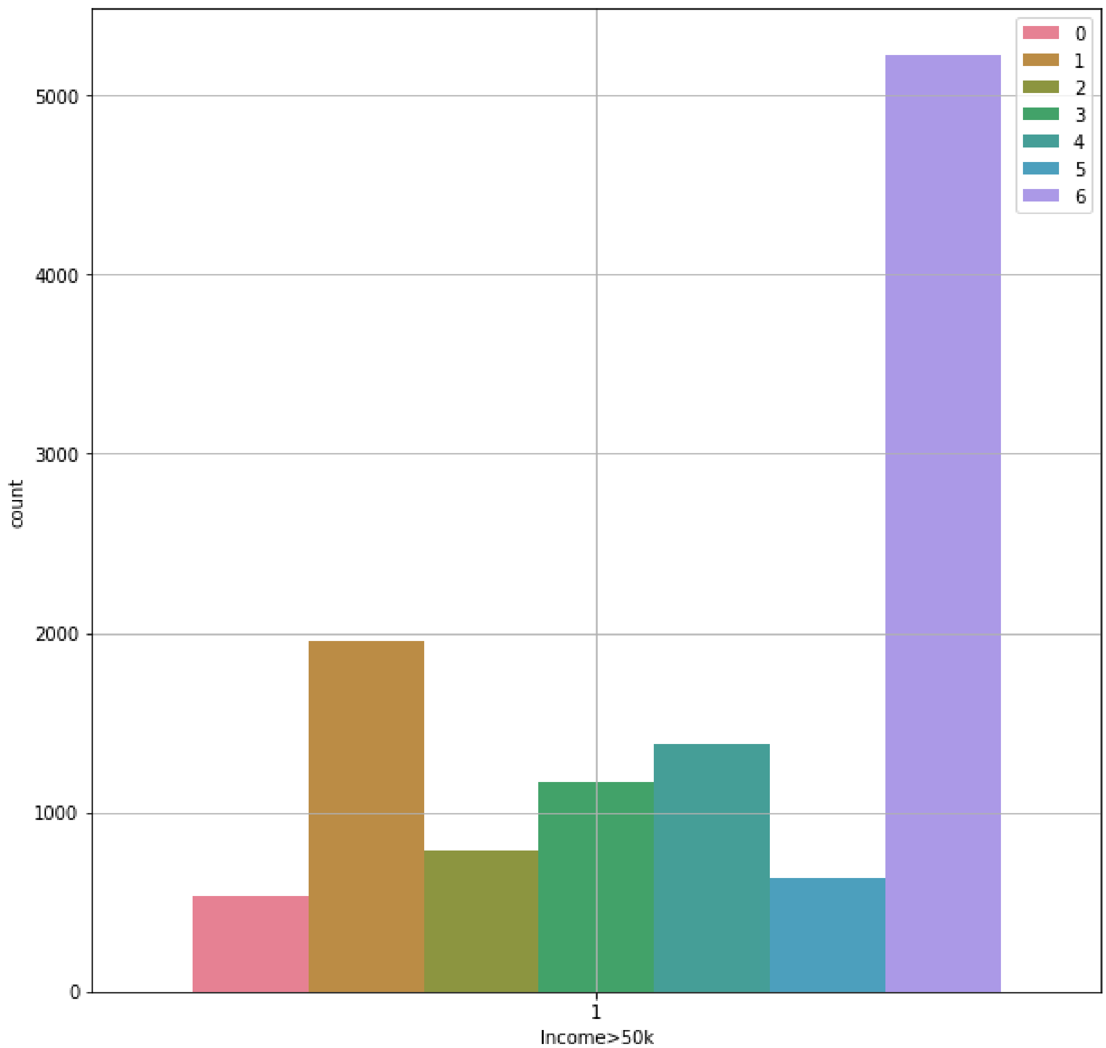
In [560]:
```python
g = sns.FacetGrid(data=df, col='Income>50k', row='Year', height=6)
g.map(sns.countplot, 'NumPersonsWorkedEmployer', order=df['NumPersonsWorkedEmployer']
for ax in g.axes.flat:
    labels = ax.get_xticklabels() # get x labels
    ax.set_xticklabels(labels, rotation=90) # set new labels
```



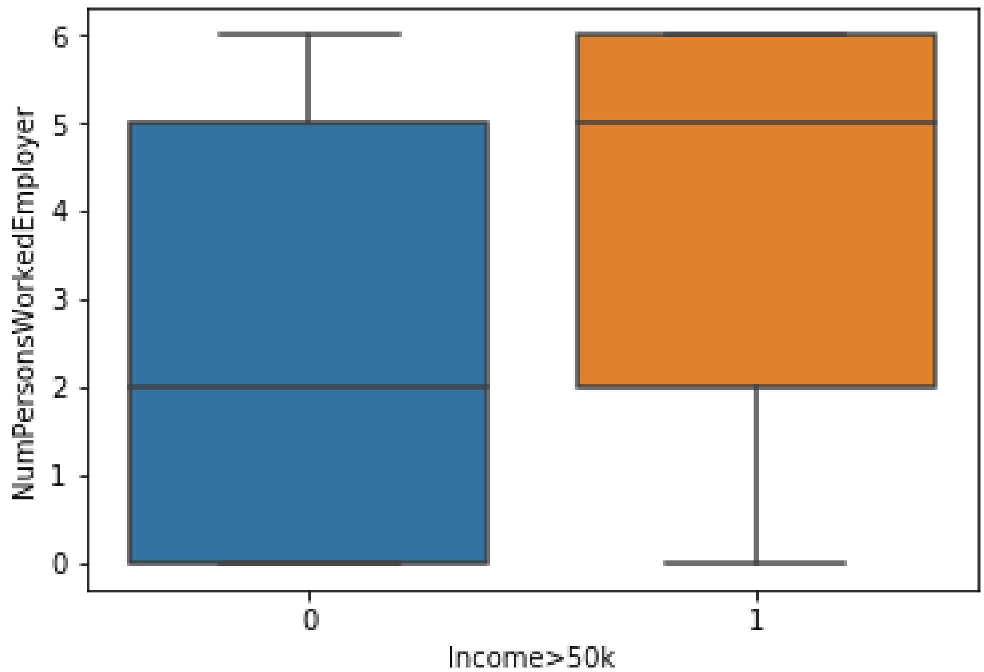
In [561]:
```python
plt.figure(figsize=(12,12))
sns.barplot(x='NumPersonsWorkedEmployer', y='Income>50k', data=df, dodge=True)
plt.xticks(rotation=90)
plt.grid(True)
```




```
In [562]: plt.figure(figsize=(10,10))
          sns.countplot(x='Income>50k', hue='NumPersonsWorkedEmployer', data=df[df['Income>50k
          plt.grid()
          plt.legend(fontsize='medium')

Out[562]: <matplotlib.legend.Legend at 0x12d9b33d358>
```

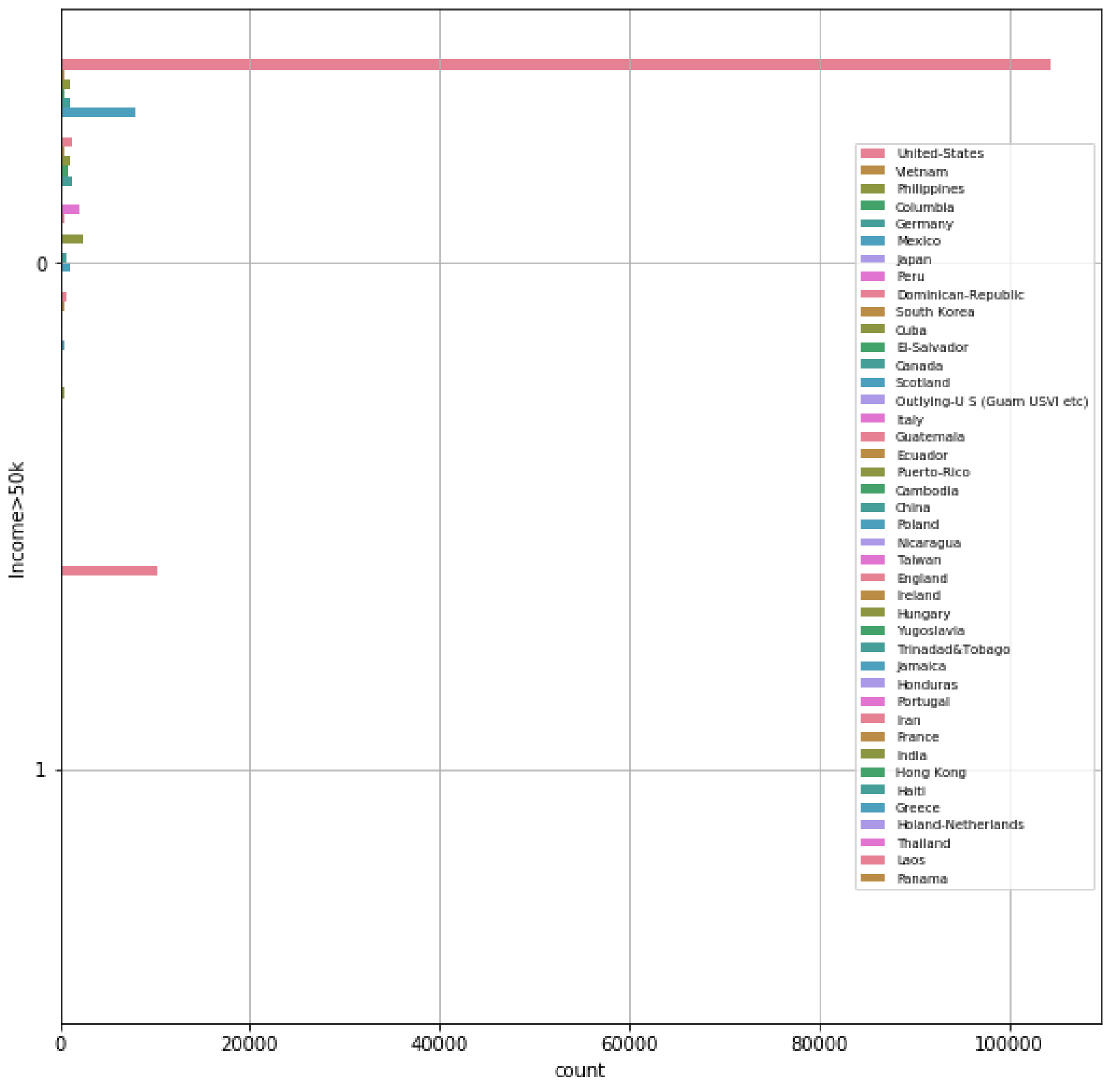



```
In [563]: sns.boxplot(x='Income>50k', y="NumPersonsWorkedEmployer", data=df)

Out[563]: <matplotlib.axes._subplots.AxesSubplot at 0x12d9b2e1438>
```



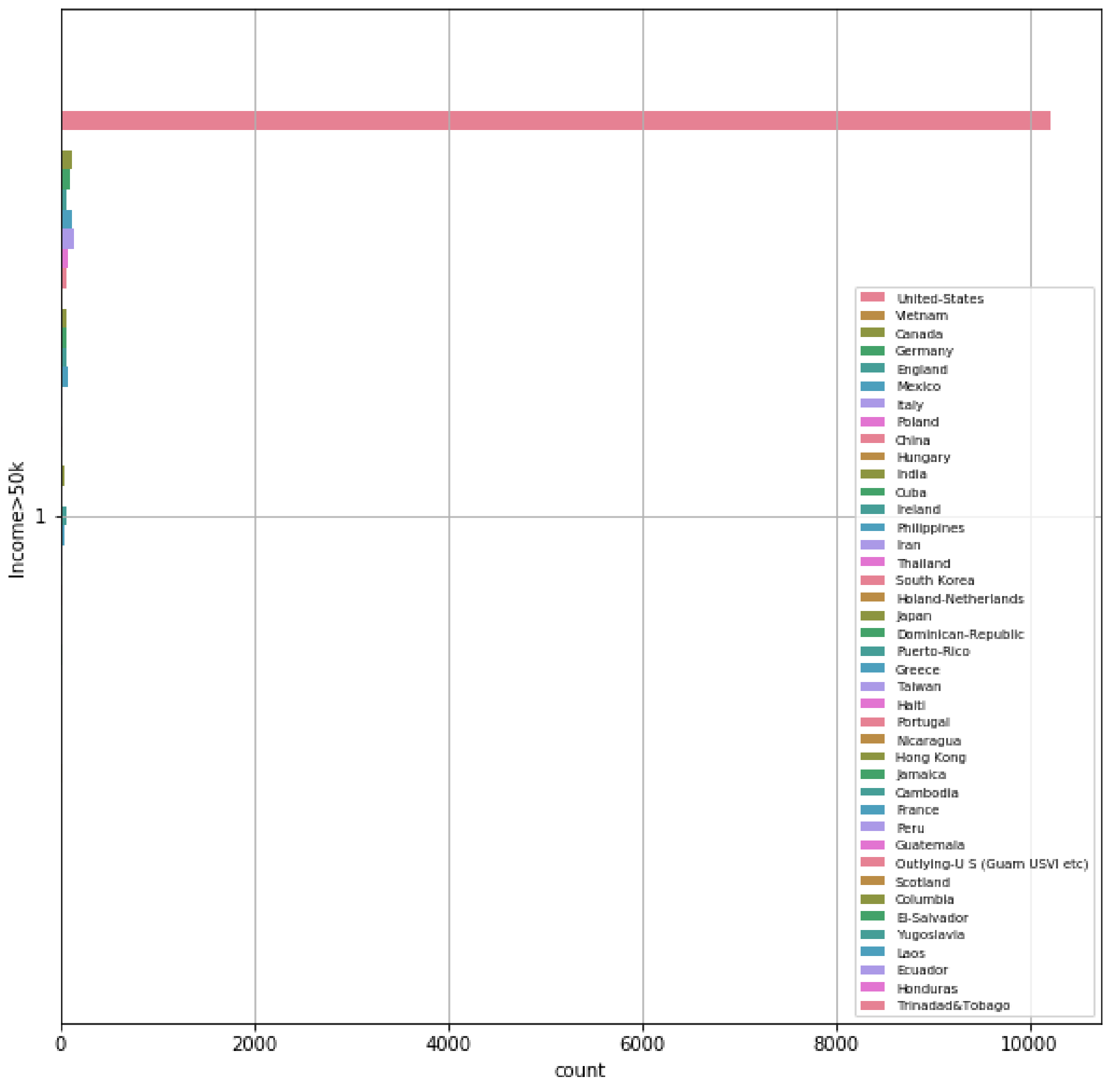
CountryBirthFather

```
In [564]: df['CountryBirthFather'].describe()

Out[564]: count              145005
          unique                 42
          top         United-States
          freq               114596
          Name: CountryBirthFather, dtype: object

In [565]: plt.figure(figsize=(10,10))
          sns.countplot(y='Income>50k', hue='CountryBirthFather', data=df, orient='H',palette=s
          plt.grid()
          plt.legend(fontsize='x-small')

Out[565]: <matplotlib.legend.Legend at 0x12d9b2a8ef0>
```

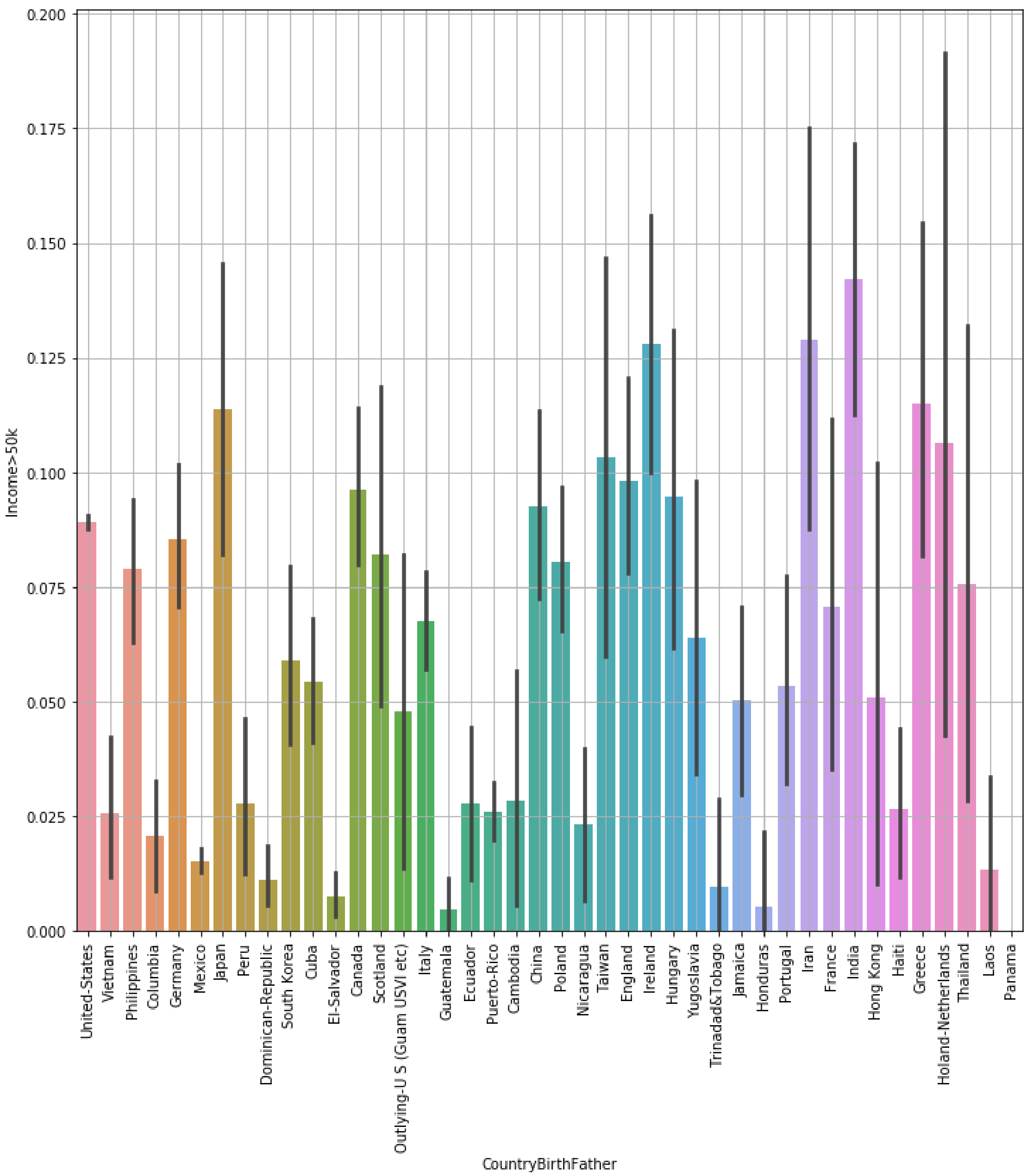


In [566]:
```
plt.figure(figsize=(10,10))
sns.countplot(y='Income>50k', hue='CountryBirthFather', data=df[df['Income>50k']==1]
plt.grid()
plt.legend(fontsize='x-small')
```

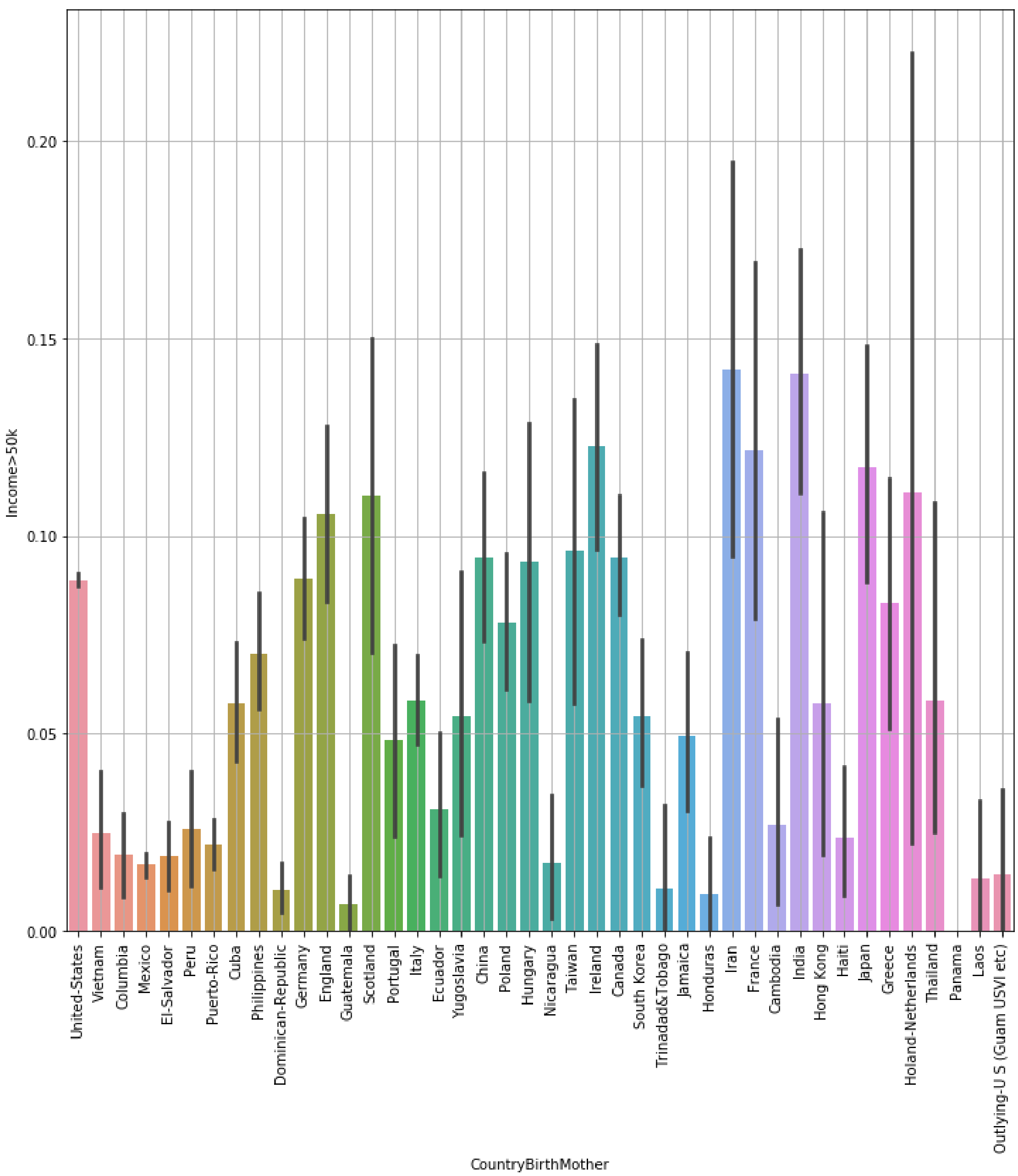
Out[566]: <matplotlib.legend.Legend at 0x12d9b36ce10>



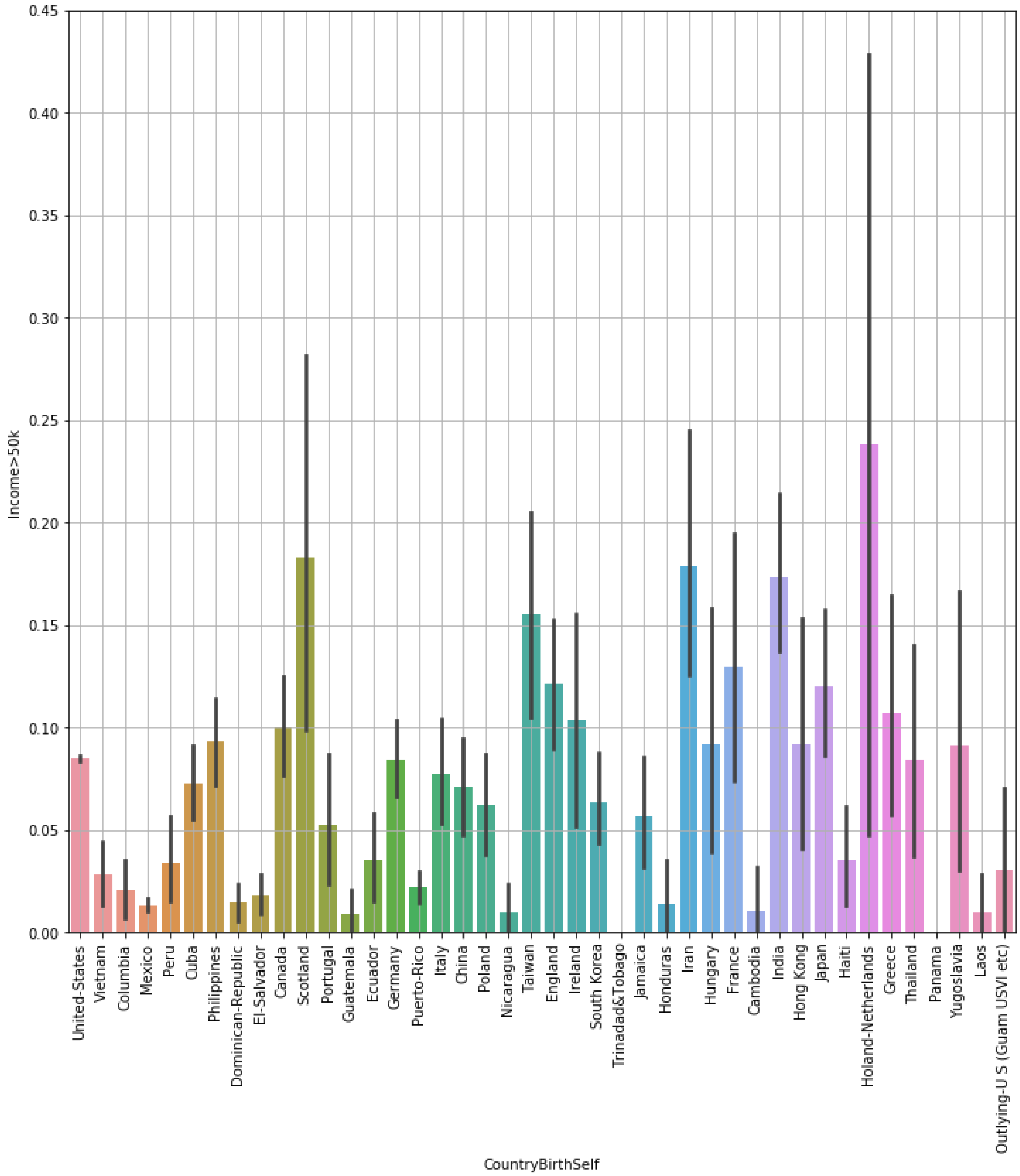
In [567]:
```
plt.figure(figsize=(12,12))
sns.barplot(x='CountryBirthFather', y='Income>50k', data=df, dodge=True)
plt.xticks(rotation=90)
plt.grid(True)
```




```
In [568]: plt.figure(figsize=(12,12))
          sns.barplot(x='CountryBirthMother', y='Income>50k', data=df, dodge=True)
          plt.xticks(rotation=90)
          plt.grid(True)
```



In [569]:
```
plt.figure(figsize=(12,12))
sns.barplot(x='CountryBirthSelf', y='Income>50k', data=df, dodge=True)
plt.xticks(rotation=90)
plt.grid(True)
```



In [570]:
```
#BirthCountry columns don't seem to be relevant. I will drop them for now.
df.drop(['CountryBirthFather','CountryBirthMother','CountryBirthSelf'], axis=1, inpla
df_test.drop(['CountryBirthFather','CountryBirthMother','CountryBirthSelf'], axis=1,
```

Citizenship

In [571]:
```
df['Citizenship'].describe()
```

Out[571]:
```
count                                    145005
unique                                        5
```

```
top       Native- Born in the United States
freq                                126782
Name: Citizenship, dtype: object
```

In [572]:
```
df['Citizenship'].value_counts()
```

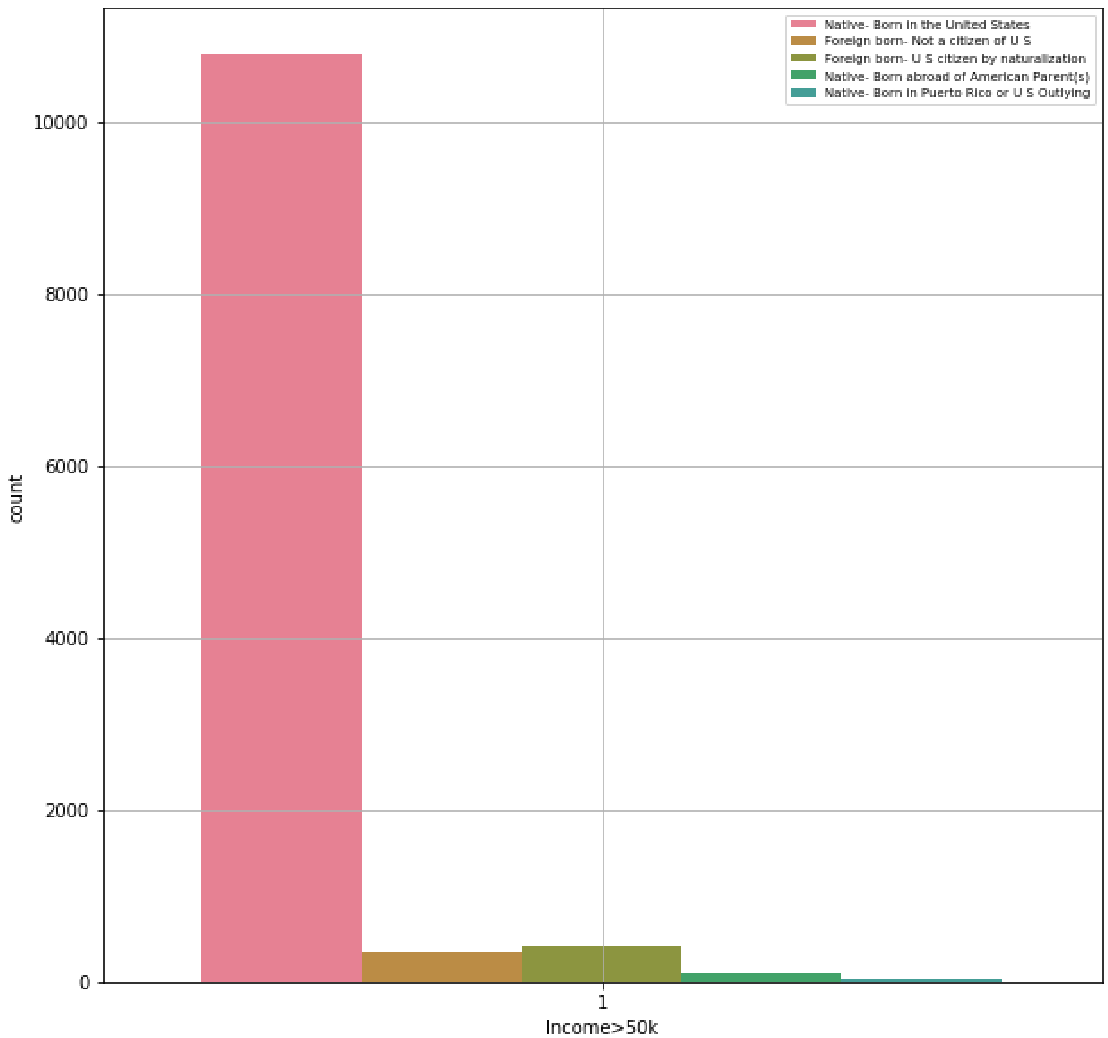
Out[572]:
```
Native- Born in the United States              126782
Foreign born- Not a citizen of U S              10829
Foreign born- U S citizen by naturalization      4614
Native- Born in Puerto Rico or U S Outlying      1469
Native- Born abroad of American Parent(s)        1311
Name: Citizenship, dtype: int64
```

In [573]:
```
df['Citizenship'].value_counts(normalize=True)
```

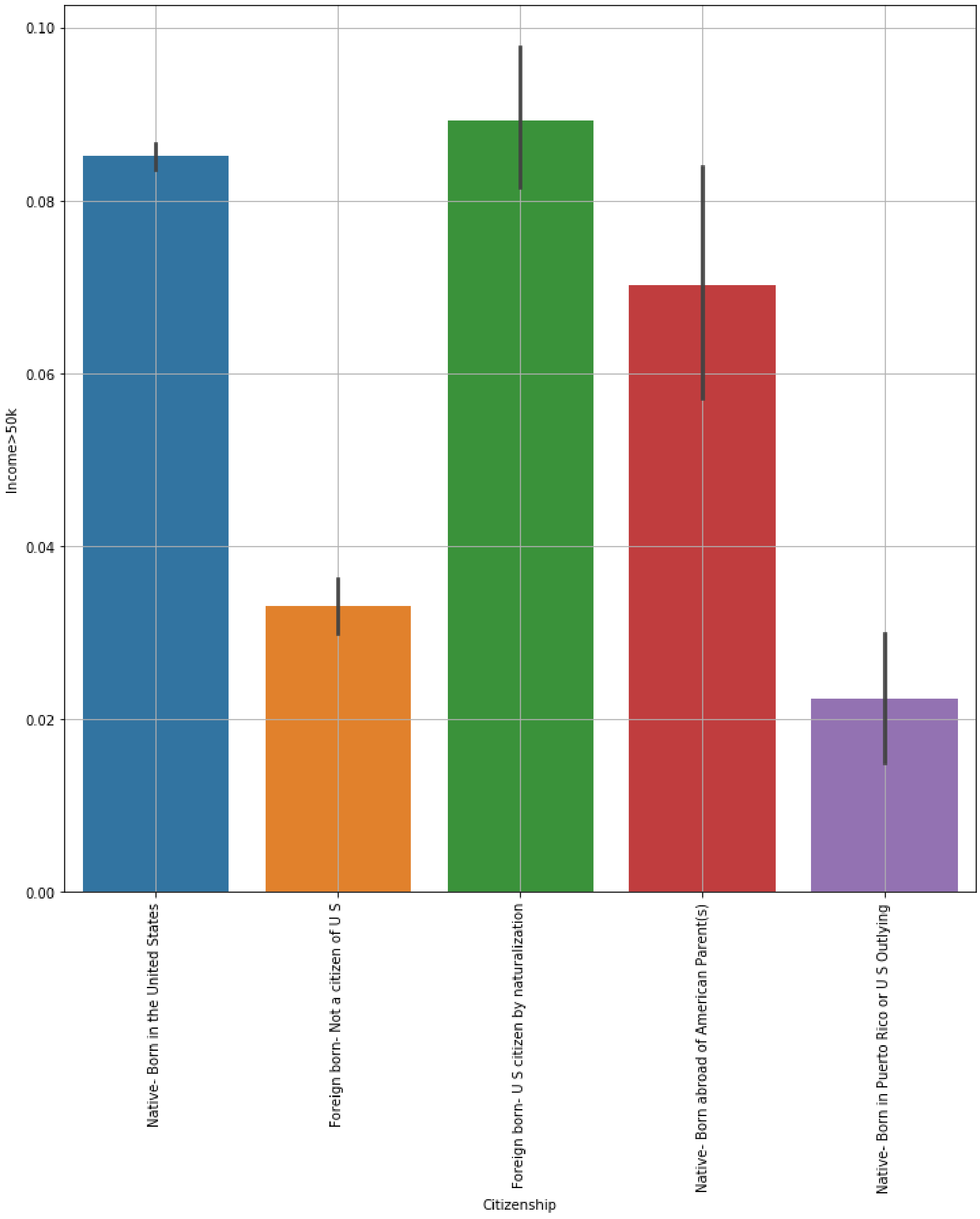
Out[573]:
```
Native- Born in the United States              0.874328
Foreign born- Not a citizen of U S             0.074680
Foreign born- U S citizen by naturalization    0.031820
Native- Born in Puerto Rico or U S Outlying    0.010131
Native- Born abroad of American Parent(s)      0.009041
Name: Citizenship, dtype: float64
```

In [574]:
```
plt.figure(figsize=(10,10))
sns.countplot(x='Income>50k', hue='Citizenship', data=df[df['Income>50k']==1],palett
plt.grid()
plt.legend(fontsize='x-small')
```

Out[574]:
```
<matplotlib.legend.Legend at 0x12da05a8cc0>
```



In [575]:
```
plt.figure(figsize=(12,12))
sns.barplot(x='Citizenship', y='Income>50k', data=df, dodge=True)
plt.xticks(rotation=90)
plt.grid(True)
```



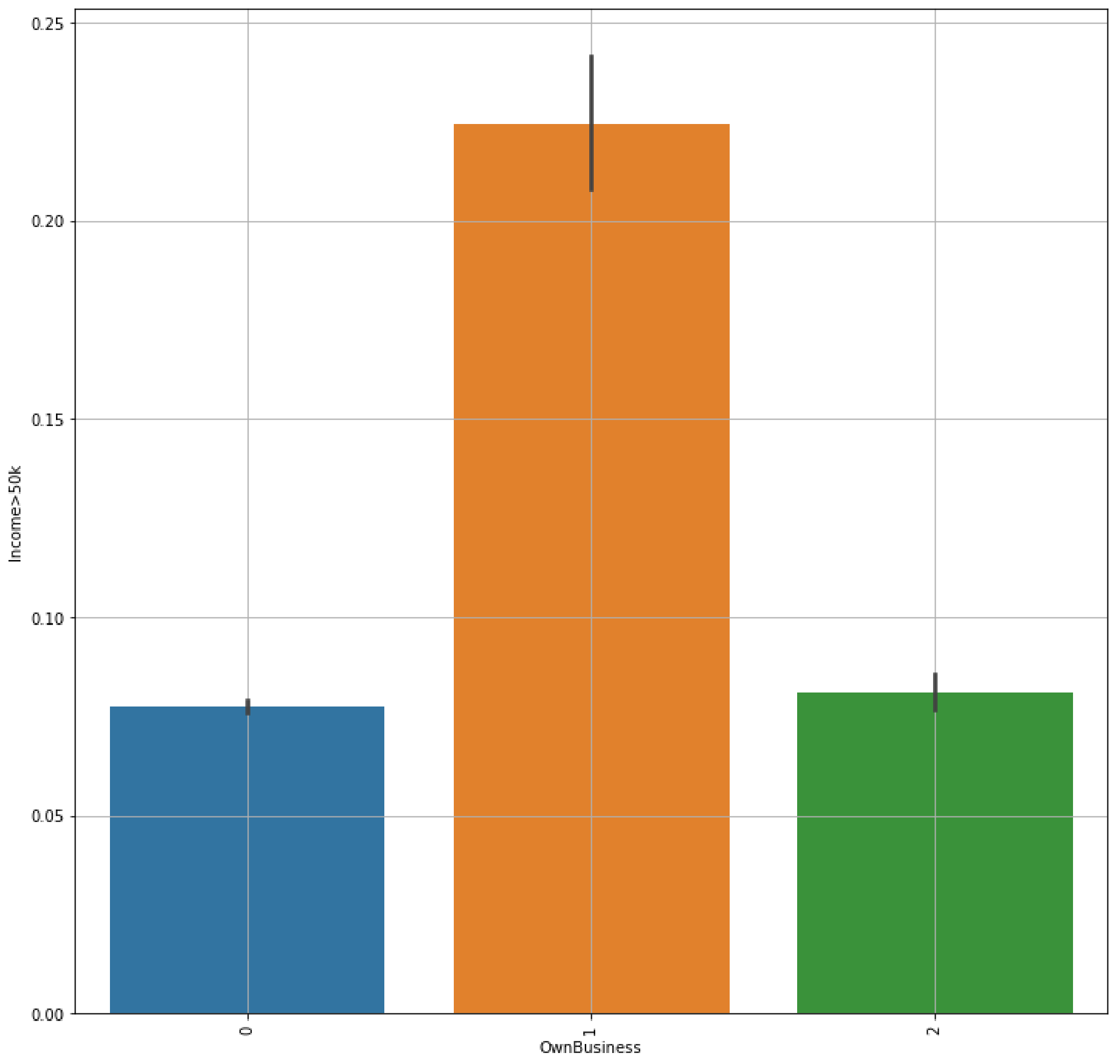

```
In [576]: #I will replace the values with 1 = US Citizen, and 0 = Non-US Citizen
          df['Citizenship'] = df['Citizenship'].apply(lambda x: 0 if x == 'Foreign born- Not a
          df_test['Citizenship'] = df_test['Citizenship'].apply(lambda x: 0 if x == 'Foreign bo
```

OwnBusiness

In [577]:
```
df['OwnBusiness'].value_counts()
```

Out[577]:
```
0    126718
2     15685
1      2602
Name: OwnBusiness, dtype: int64
```

In [578]:
```
plt.figure(figsize=(12,12))
sns.barplot(x='OwnBusiness', y='Income>50k', data=df, dodge=True)
plt.xticks(rotation=90)
plt.grid(True)
```



In [579]:
```
#Very clear correlation between Owning a Business and Income.
#I will rename '2' to 1.
df['OwnBusiness'] = np.where(df['OwnBusiness'] == 0, 0, 1)
```

```
df_test['OwnBusiness'] = np.where(df_test['OwnBusiness'] == 0, 0, 1)

df['OwnBusiness'].value_counts(normalize=True)
```

```
Out[579]: 0    0.873887
          1    0.126113
          Name: OwnBusiness, dtype: float64
```

VeteranBenefits'

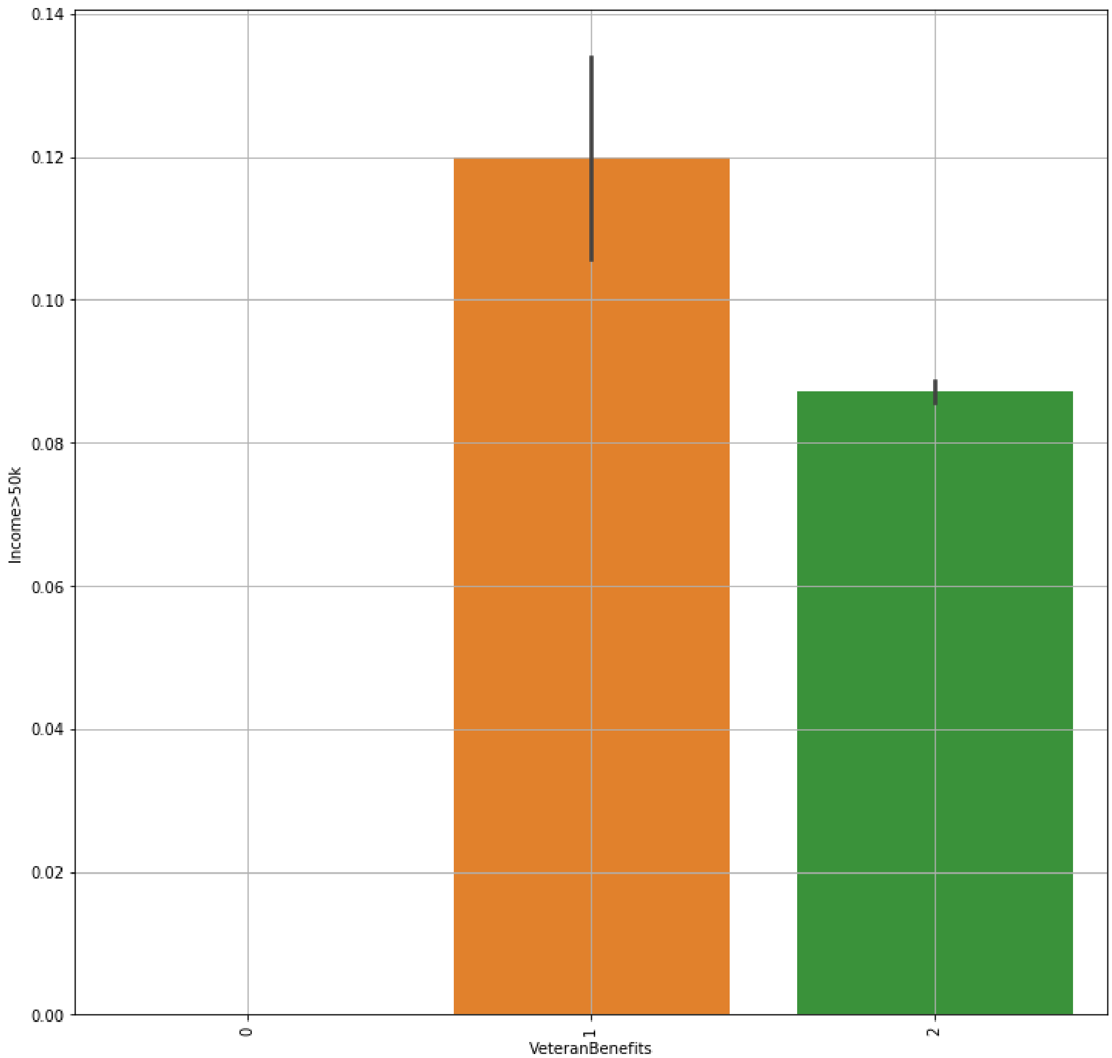
```
In [580]: df['VeteranBenefits'].describe()
```

```
Out[580]: count    145005.000000
          mean          1.826165
          std           0.551837
          min           0.000000
          25%           2.000000
          50%           2.000000
          75%           2.000000
          max           2.000000
          Name: VeteranBenefits, dtype: float64
```

```
In [581]: df['VeteranBenefits'].value_counts()
```

```
Out[581]: 2    131464
          0     11666
          1      1875
          Name: VeteranBenefits, dtype: int64
```

```
In [582]: plt.figure(figsize=(12,12))
          sns.barplot(x='VeteranBenefits', y='Income>50k', data=df, dodge=True)
          plt.xticks(rotation=90)
          plt.grid(True)
```




```
In [583]: #Couldn't find any explanations for this column, what it means, what are the digits.
          df.drop('VeteranBenefits', axis=1, inplace=True)
          df_test.drop('VeteranBenefits', axis=1, inplace=True)
```

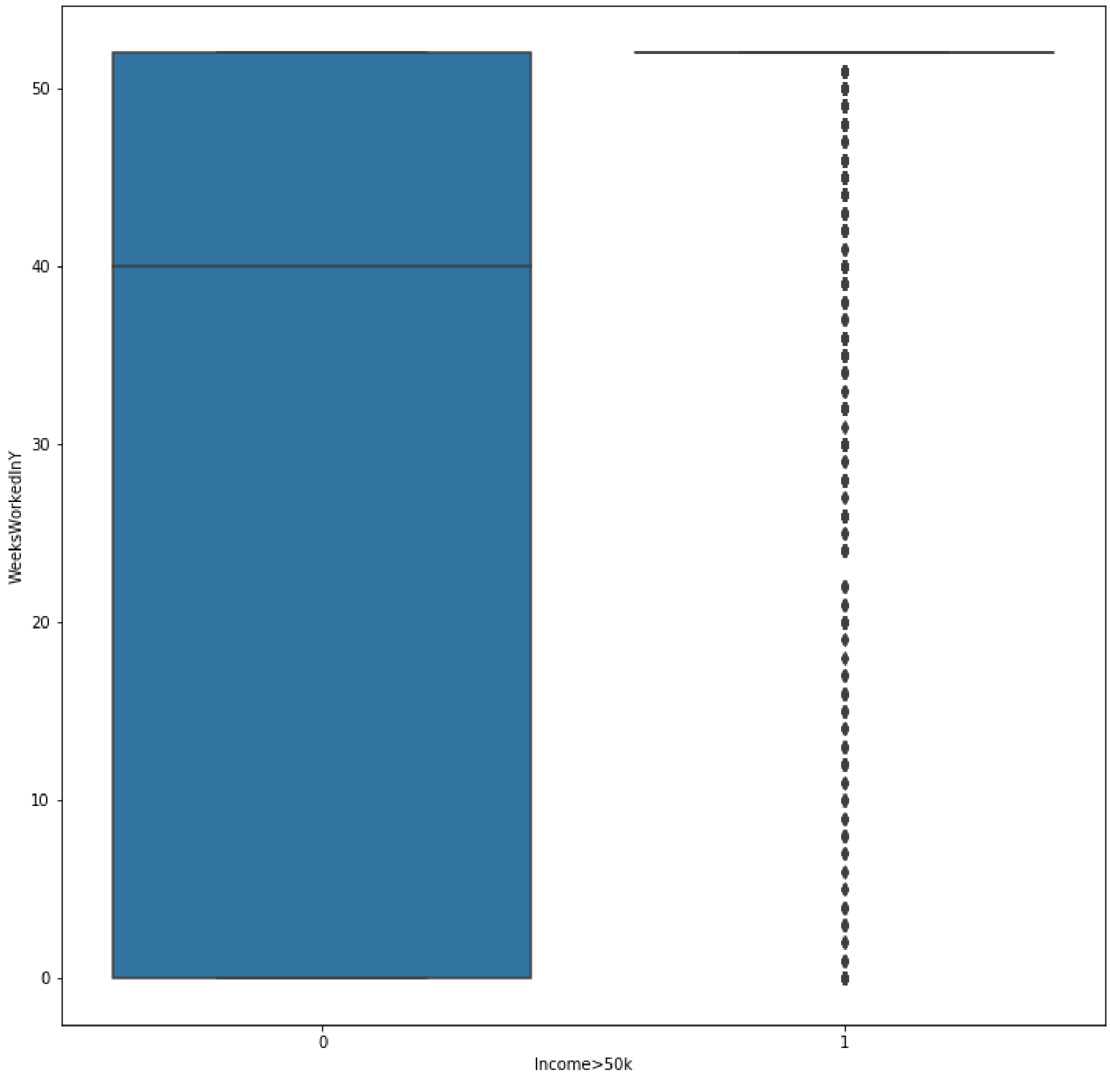
WeeksWorkedInY

```
In [584]: df['WeeksWorkedInY'].describe()

Out[584]: count    145005.000000
          mean         30.532313
          std          23.667317
          min           0.000000
          25%           0.000000
          50%          47.000000
          75%          52.000000
```

```
max           52.000000
Name: WeeksWorkedInY, dtype: float64
```

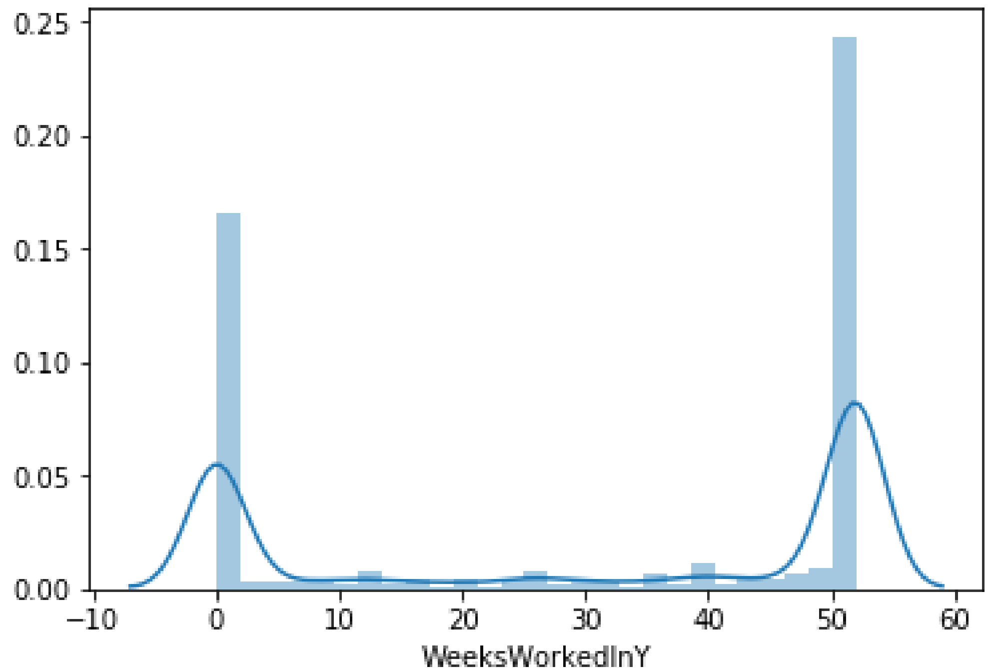
In [585]:
```
plt.figure(figsize=(12,12))
sns.boxplot(x='Income>50k', y='WeeksWorkedInY', data=df)
```

Out[585]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12cc6d242e8>
```

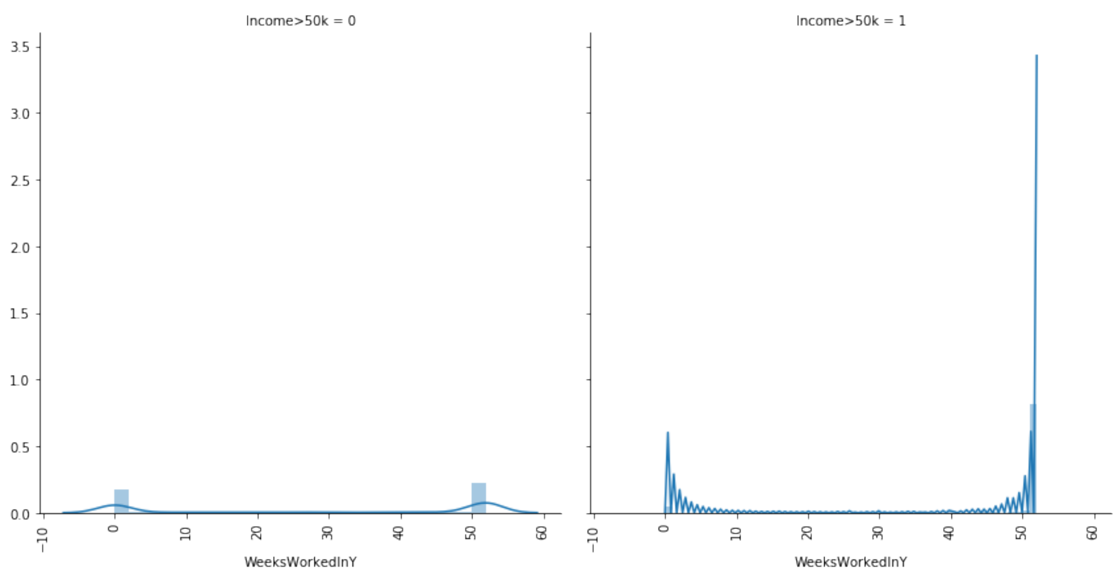


In [586]:
```
sns.distplot(df['WeeksWorkedInY'])
```

Out[586]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12caab58240>
```

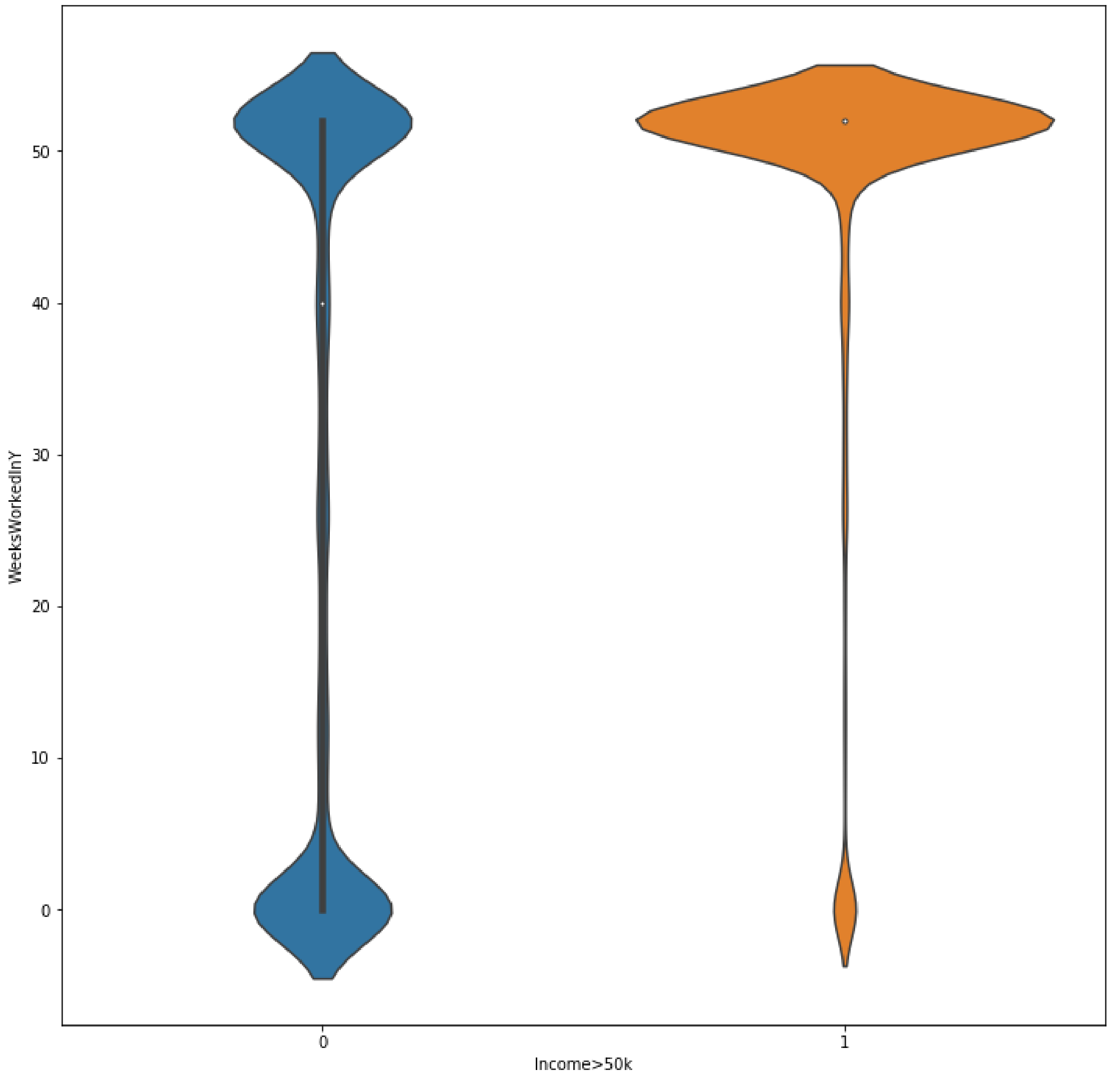



```
In [587]: g = sns.FacetGrid(data=df, col='Income>50k', height=6)
          g.map(sns.distplot, 'WeeksWorkedInY' )
          for ax in g.axes.flat:
              labels = ax.get_xticklabels() # get x labels
              ax.set_xticklabels(labels, rotation=90) # set new labels
```



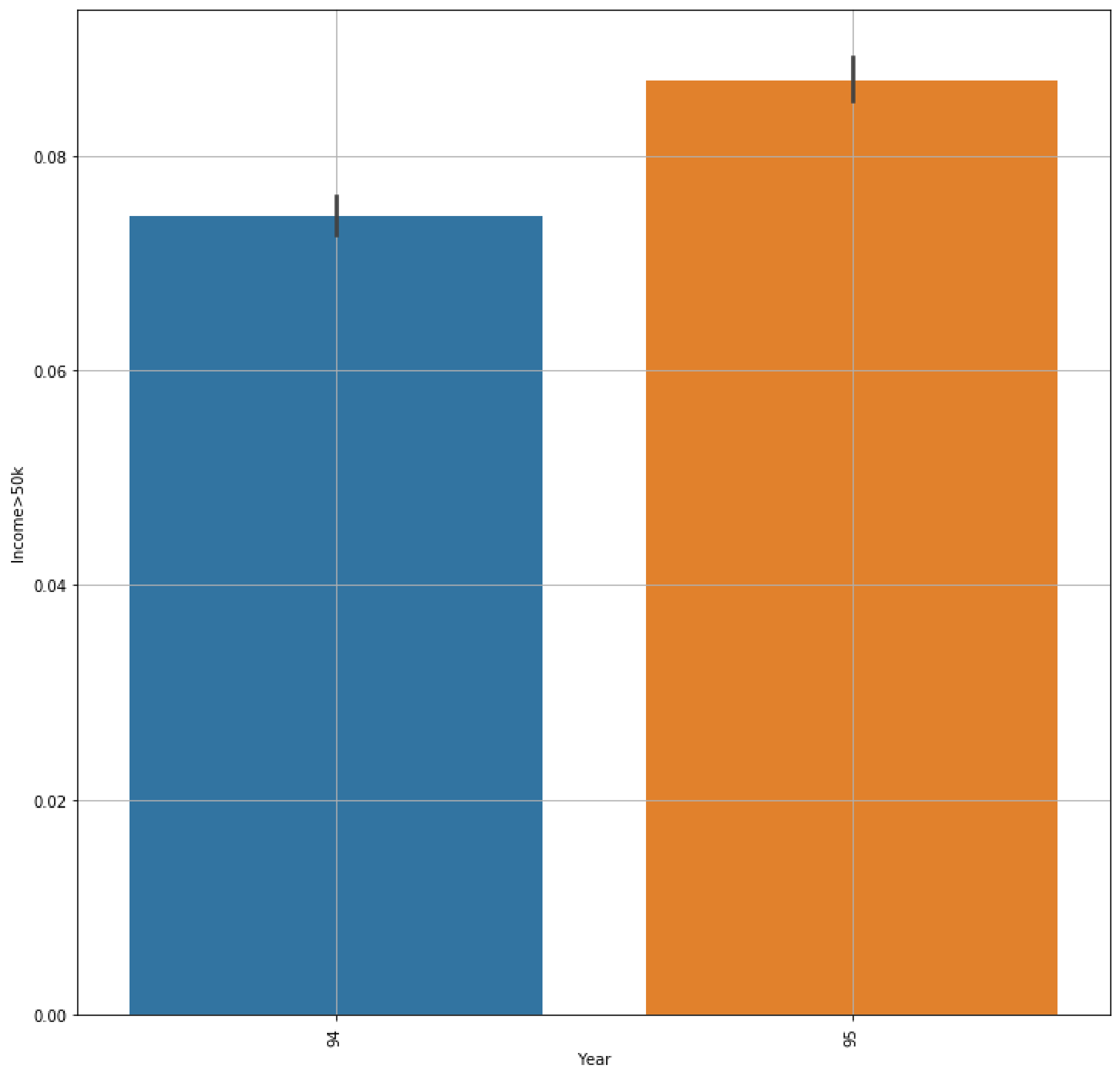
```
In [588]: plt.figure(figsize=(12,12))
          sns.violinplot(x='Income>50k', y='WeeksWorkedInY', data=df)
```

```
Out[588]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4fb583c8>
```

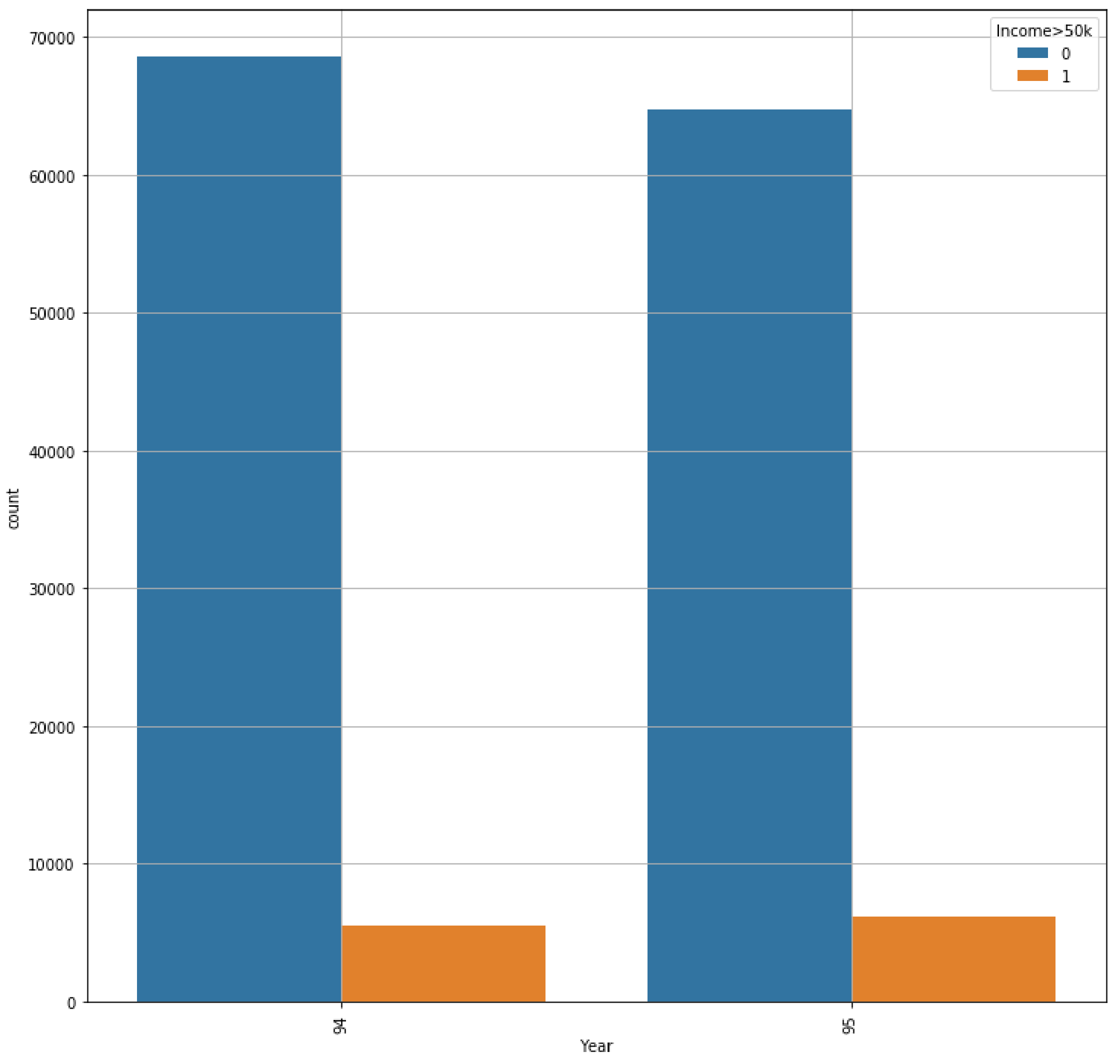


Year

```
In [589]: plt.figure(figsize=(12,12))
          sns.barplot(x='Year', y='Income>50k', data=df, dodge=True)
          plt.xticks(rotation=90)
          plt.grid(True)
```




```
In [590]: plt.figure(figsize=(12,12))
          sns.countplot(x='Year', data=df, hue='Income>50k',dodge=True)
          plt.xticks(rotation=90)
          plt.grid(True)
```




```
In [591]: #Year doesn't seem to be correlated with Income levels. Even the above charts were v
          df.drop('Year', axis=1, inplace=True)
          df_test.drop('Year', axis=1, inplace=True)

In [592]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 145005 entries, 0 to 199520
Data columns (total 20 columns):
Age                         145005 non-null int64
ClassOfWorker               145005 non-null object
Education                   145005 non-null object
WagePerHour                 145005 non-null int64
MaritalStatus               145005 non-null object
MajorIndustryCode           145005 non-null object
```

```
MajorOccupationCode         145005 non-null object
Race                        145005 non-null object
Male                        145005 non-null int32
CapitalGains                145005 non-null int64
CapitalLosses               145005 non-null int64
StockDividends              145005 non-null int64
TaxFilerStat                145005 non-null object
HouseholdFamilyStatus       145005 non-null object
HouseholdSummary            145005 non-null object
NumPersonsWorkedEmployer    145005 non-null int64
Citizenship                 145005 non-null int64
OwnBusiness                 145005 non-null int32
WeeksWorkedInY              145005 non-null int64
Income>50k                  145005 non-null int32
dtypes: int32(3), int64(8), object(9)
memory usage: 26.6+ MB
```

```
In [593]: df.describe()
```

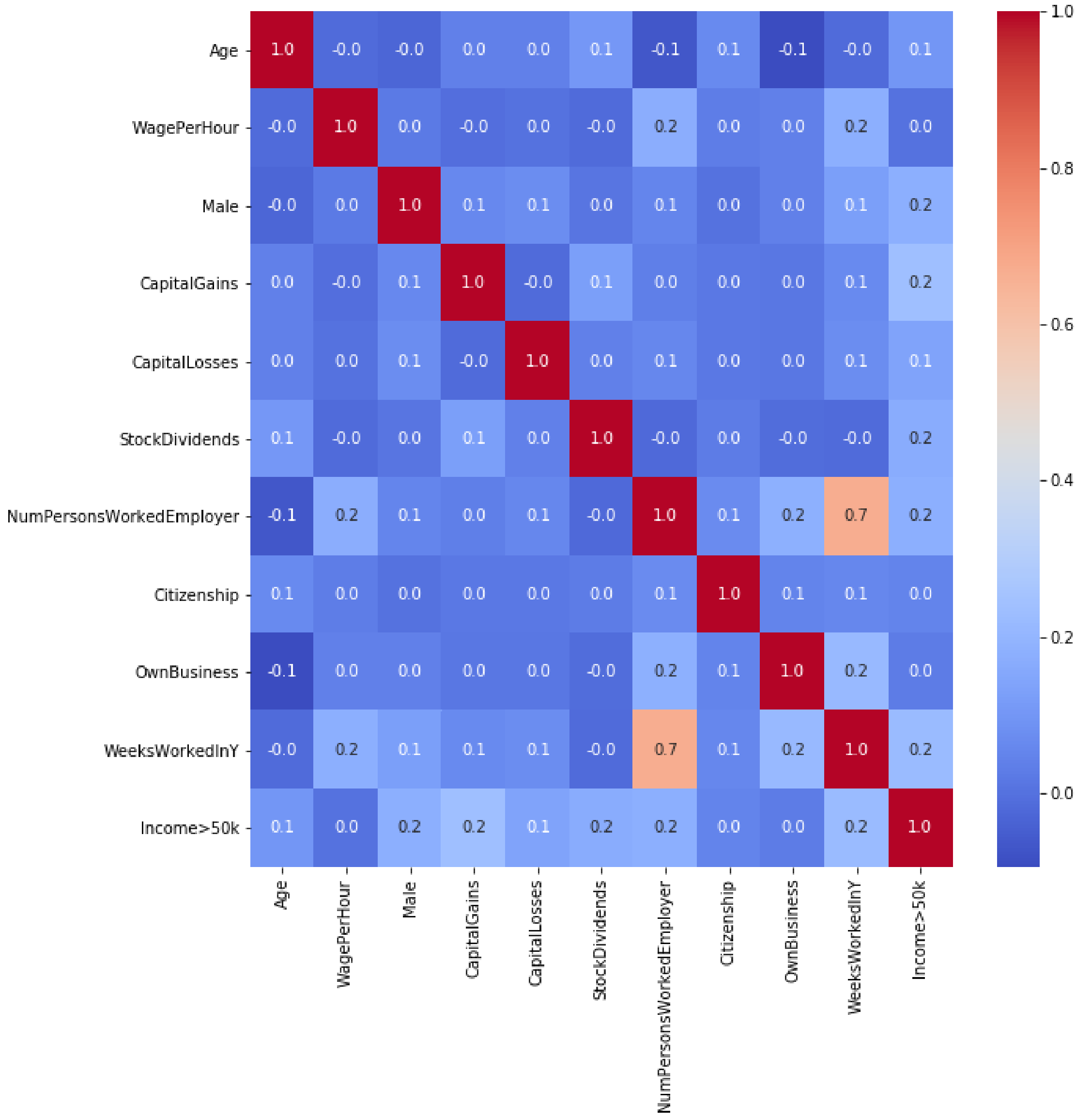
```
Out[593]:                 Age    WagePerHour           Male   CapitalGains  \
          count  145005.000000  145005.000000  145005.000000  145005.000000
          mean       39.418468      70.766222       0.485866     557.842136
          std        19.327772     277.214096       0.499802    5286.552736
          min         0.000000       0.000000       0.000000       0.000000
          25%        25.000000       0.000000       0.000000       0.000000
          50%        38.000000       0.000000       0.000000       0.000000
          75%        52.000000       0.000000       1.000000       0.000000
          max        90.000000    2000.000000       1.000000   99999.000000

                 CapitalLosses  StockDividends  NumPersonsWorkedEmployer    Citizenship  \
          count  145005.000000   145005.000000             145005.000000  145005.000000
          mean       48.684659      252.061688                  2.581587       0.925320
          std       308.852831     2260.621084                  2.402695       0.262876
          min         0.000000        0.000000                  0.000000       0.000000
          25%         0.000000        0.000000                  0.000000       1.000000
          50%         0.000000        0.000000                  2.000000       1.000000
          75%         0.000000        0.000000                  5.000000       1.000000
          max      4608.000000    99999.000000                  6.000000       1.000000

                   OwnBusiness  WeeksWorkedInY     Income>50k
          count  145005.000000   145005.000000  145005.000000
          mean        0.126113       30.532313       0.080577
          std         0.331978       23.667317       0.272185
          min         0.000000        0.000000       0.000000
          25%         0.000000        0.000000       0.000000
          50%         0.000000       47.000000       0.000000
          75%         0.000000       52.000000       0.000000
          max         1.000000       52.000000       1.000000
```

```
In [594]: df.head()

Out[594]:    Age                   ClassOfWorker             Education  WagePerHour  \
          0   73                              NA  High school graduate            0
          1   58  Self-employed-not incorporated  High school graduate            0
          2   18                              NA                 Other            0
          3    9                              NA                 Other            0
          4   10                              NA                 Other            0

             MaritalStatus            MajorIndustryCode  \
          0        Widowed  Not in universe or children
          1       Divorced                 Construction
          2  Never married  Not in universe or children
          3  Never married  Not in universe or children
          4  Never married  Not in universe or children

                           MajorOccupationCode                         Race  Male  \
          0                                 NA                        White     0
          1  Precision production craft & repair                        White     1
          2                                 NA  Asian or Pacific Islander     0
          3                                 NA                        White     0
          4                                 NA                        White     0

             CapitalGains  CapitalLosses  StockDividends TaxFilerStat  \
          0             0              0               0     Nonfiler
          1             0              0               0        Other
          2             0              0               0     Nonfiler
          3             0              0               0     Nonfiler
          4             0              0               0     Nonfiler

            HouseholdFamilyStatus                HouseholdSummary  \
          0                 Other   Other relative of householder
          1           Householder                     Householder
          2                 Other               Child 18 or older
          3                 Other  Child under 18 never married
          4                 Other  Child under 18 never married

             NumPersonsWorkedEmployer  Citizenship  OwnBusiness  WeeksWorkedInY  \
          0                         0            1            0               0
          1                         1            1            0              52
          2                         0            0            0               0
          3                         0            1            0               0
          4                         0            1            0               0

             Income>50k
          0           0
          1           0
          2           0
```

```
3            0
4            0
```

In [595]:
```
plt.subplots(figsize=(10, 10))
df_cor = df.corr()
sns.heatmap(df_cor, annot=True, fmt = ".1f", cmap = "coolwarm")
```

Out[595]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4fb709b0>



In [596]:
```
df_test.shape
```

Out[596]: (74777, 20)

In [597]:
```
df.shape
```

Out[597]: (145005, 20)

6 MODELING

6.1 Encoding

In [598]:
```
df_train_encoded=pd.get_dummies(df, columns=['ClassOfWorker', 'Education', 'MaritalSt
                                    'MajorOccupationCode', 'Race', 'TaxFilerStat'
                                   'HouseholdSummary'],
                          prefix=['ClassOfWorker', 'Education', 'MaritalStatus', 'Maj
                                    'MajorOccupationCode', 'Race', 'TaxFilerStat'
                                   'HouseholdSummary'])

df_test_encoded=pd.get_dummies(df_test, columns=['ClassOfWorker', 'Education', 'Mari
                                    'MajorOccupationCode', 'Race', 'TaxFilerStat'
                                   'HouseholdSummary'],
                          prefix=['ClassOfWorker', 'Education', 'MaritalStatus', 'Maj
                                    'MajorOccupationCode', 'Race', 'TaxFilerStat'
                                   'HouseholdSummary'])
```

In [599]:
```
df_train_encoded.shape
```

Out[599]:
```
(145005, 86)
```

In [600]:
```
X_train = df_train_encoded.loc[:,df_train_encoded.columns != 'Income>50k']
y_train = df_train_encoded['Income>50k']
X_test = df_test_encoded.loc[:,df_test_encoded.columns != 'Income>50k']
y_test = df_test_encoded['Income>50k']
```

6.2 Use RandomForest for Feature Selection

In [601]:
```
#First, scale the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = pd.DataFrame(scaler.transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test),  columns=X_test.columns)
```

```
C:\Users\ahmed\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:625: DataConversionWa
  return self.partial_fit(X, y)
C:\Users\ahmed\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DataConversionWarning: Data
  """
C:\Users\ahmed\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: DataConversionWarning: Data
```

In [602]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=300)
rfc.fit(X_train, y_train)
```

Out[602]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
```

```
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False)
```

```
In [603]: from sklearn.feature_selection import SelectFromModel
          sel = SelectFromModel(rfc, prefit=True)
          selected_feat= X_train.columns[(sel.get_support())]
```

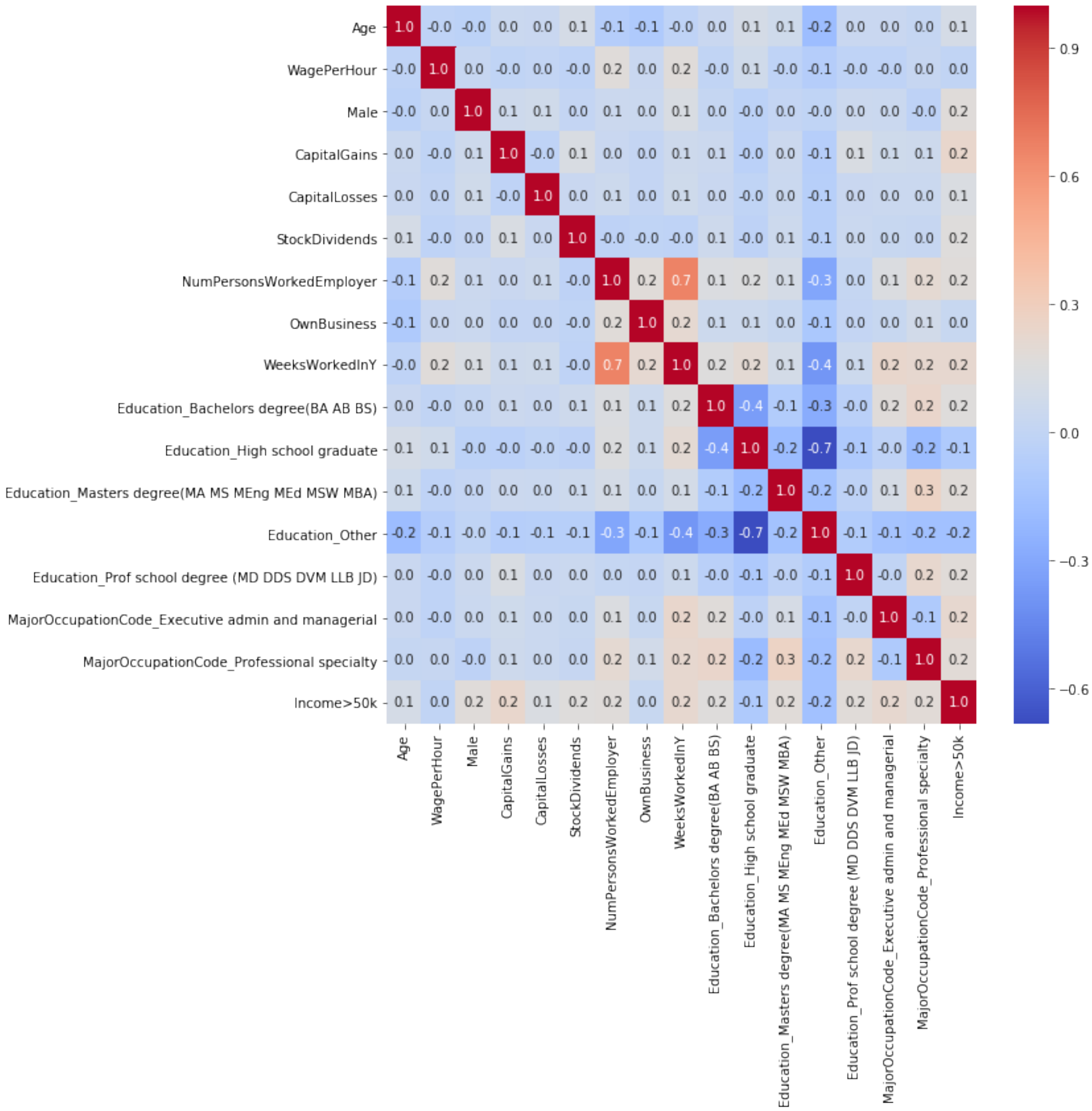
```
In [604]: print('Selected Features:\n',*selected_feat, sep='\n')
```

```
Selected Features:

Age
WagePerHour
Male
CapitalGains
CapitalLosses
StockDividends
NumPersonsWorkedEmployer
OwnBusiness
WeeksWorkedInY
Education_Bachelors degree(BA AB BS)
Education_High school graduate
Education_Masters degree(MA MS MEng MEd MSW MBA)
Education_Other
Education_Prof school degree (MD DDS DVM LLB JD)
MajorOccupationCode_Executive admin and managerial
MajorOccupationCode_Professional specialty
```

```
In [605]: X_train = X_train[selected_feat]
          X_test = X_test[selected_feat]
```

```
In [606]: df2 = pd.concat([X_train,y_train.reset_index()], axis=1)
          df2.drop('index', axis = 1 ,inplace=True)

          plt.subplots(figsize=(10, 10))
          df_cor = df2.corr()
          sns.heatmap(df_cor, annot=True, fmt = ".1f", cmap = "coolwarm")
```

```
Out[606]: <matplotlib.axes._subplots.AxesSubplot at 0x12d4eb89898>
```



6.3 Model Selection

```
In [607]: X_train.shape
```

```
Out[607]: (145005, 16)
```

```
In [608]: from sklearn.tree import DecisionTreeClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.svm import SVC
          from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold,
```

```
In [609]: kfold = StratifiedKFold(n_splits=10)
```

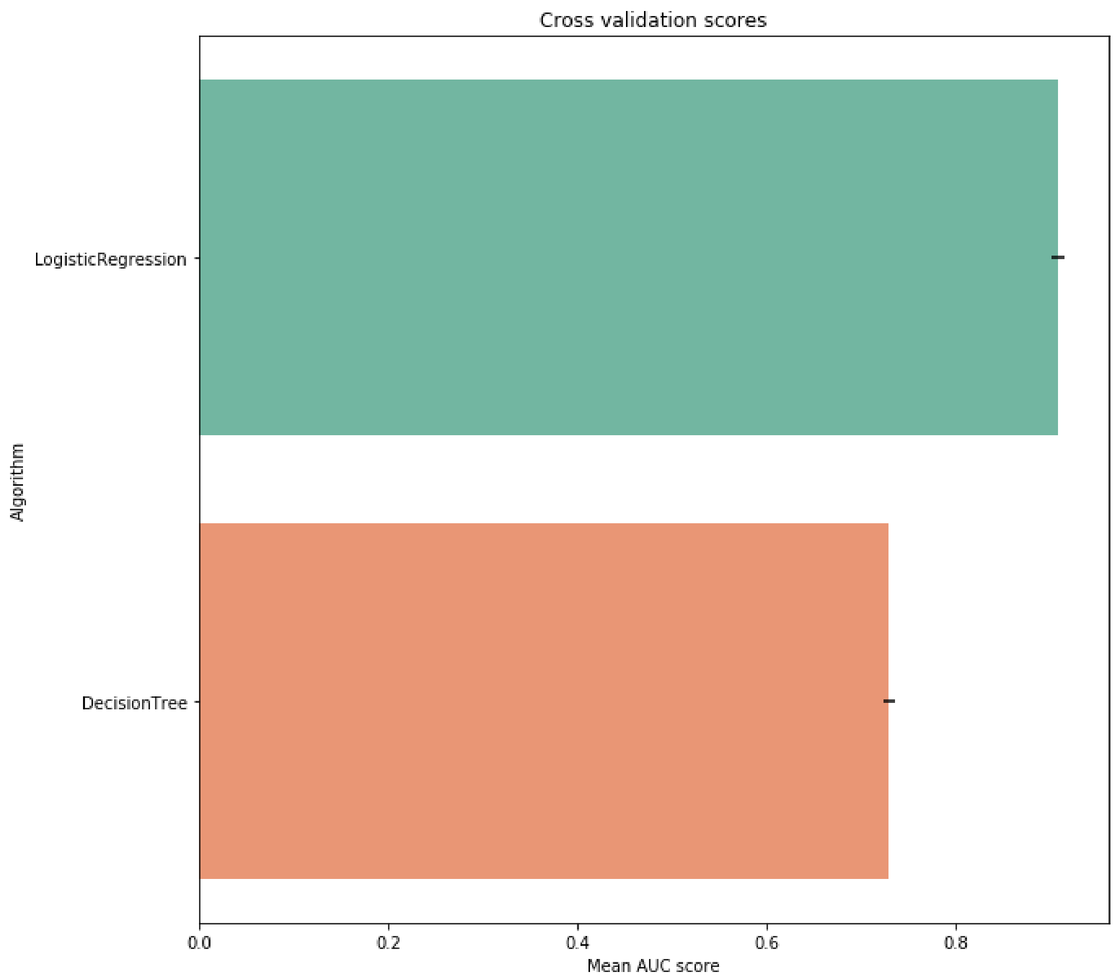
In [610]:
```
#Doing 10-fold cross validation, using Decision Tree and Logistic Regression

rs=42

classifiers = [] # list of classifiers tested
classifiers.append(LogisticRegression(random_state = rs))
classifiers.append(DecisionTreeClassifier(random_state = rs))

cv_results = []
for classifier in classifiers :
    cv_results.append(cross_val_score(classifier, X_train, y_train, scoring = 'roc_au

cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())

cv_res = pd.DataFrame({'CV_score':cv_means, 'CV_stddev':cv_std, 'Algorithm':['Logisti
```

In [611]:
```
cv_res
```

Out[611]:

	CV_score	CV_stddev	Algorithm
0	0.908088	0.006893	LogisticRegression
1	0.729347	0.005867	DecisionTree

In [612]:
```
plt.subplots(figsize=(10, 10))
g = sns.barplot('CV_score','Algorithm', data = cv_res, palette='Set2', orient = 'h',
g.set_xlabel('Mean AUC score')
g = g.set_title('Cross validation scores')
```

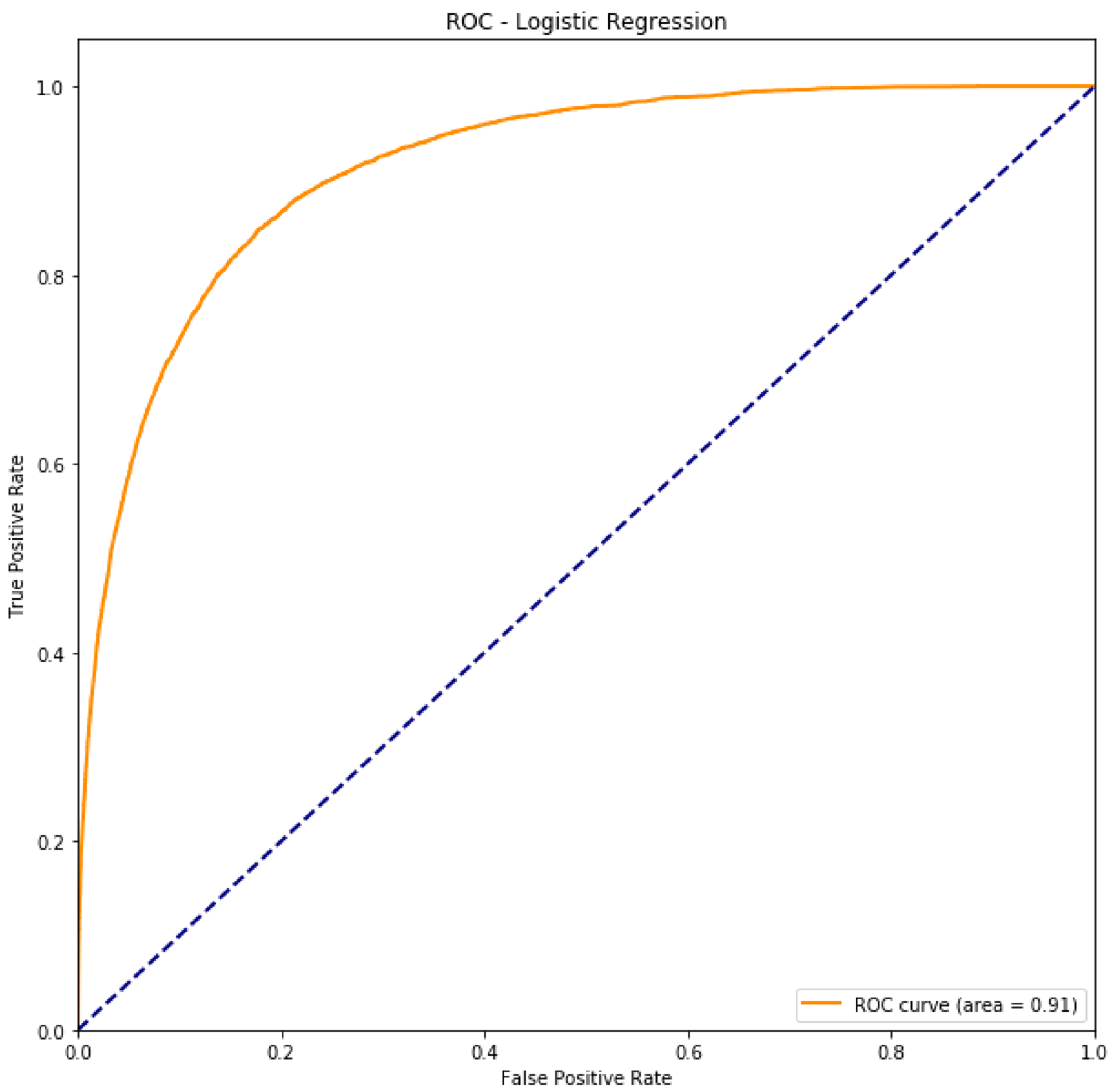


6.4 Logistic Regression

```
In [613]: from sklearn.metrics import roc_curve, auc

          LR = LogisticRegression(random_state=42)
          y_score = LR.fit(X_train, y_train).decision_function(X_test)
          fpr = dict()
          tpr = dict()
          roc_auc = dict()

          fpr[0], tpr[0], _ = roc_curve(y_test, y_score)
          roc_auc[0] = auc(fpr[0], tpr[0])

          # Compute micro-average ROC curve and ROC area
          fpr[1], tpr[1], _ = roc_curve(y_test.ravel(), y_score.ravel())
          roc_auc[1] = auc(fpr[1], tpr[1])
```

```
C:\Users\ahmed\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning
  FutureWarning)
```

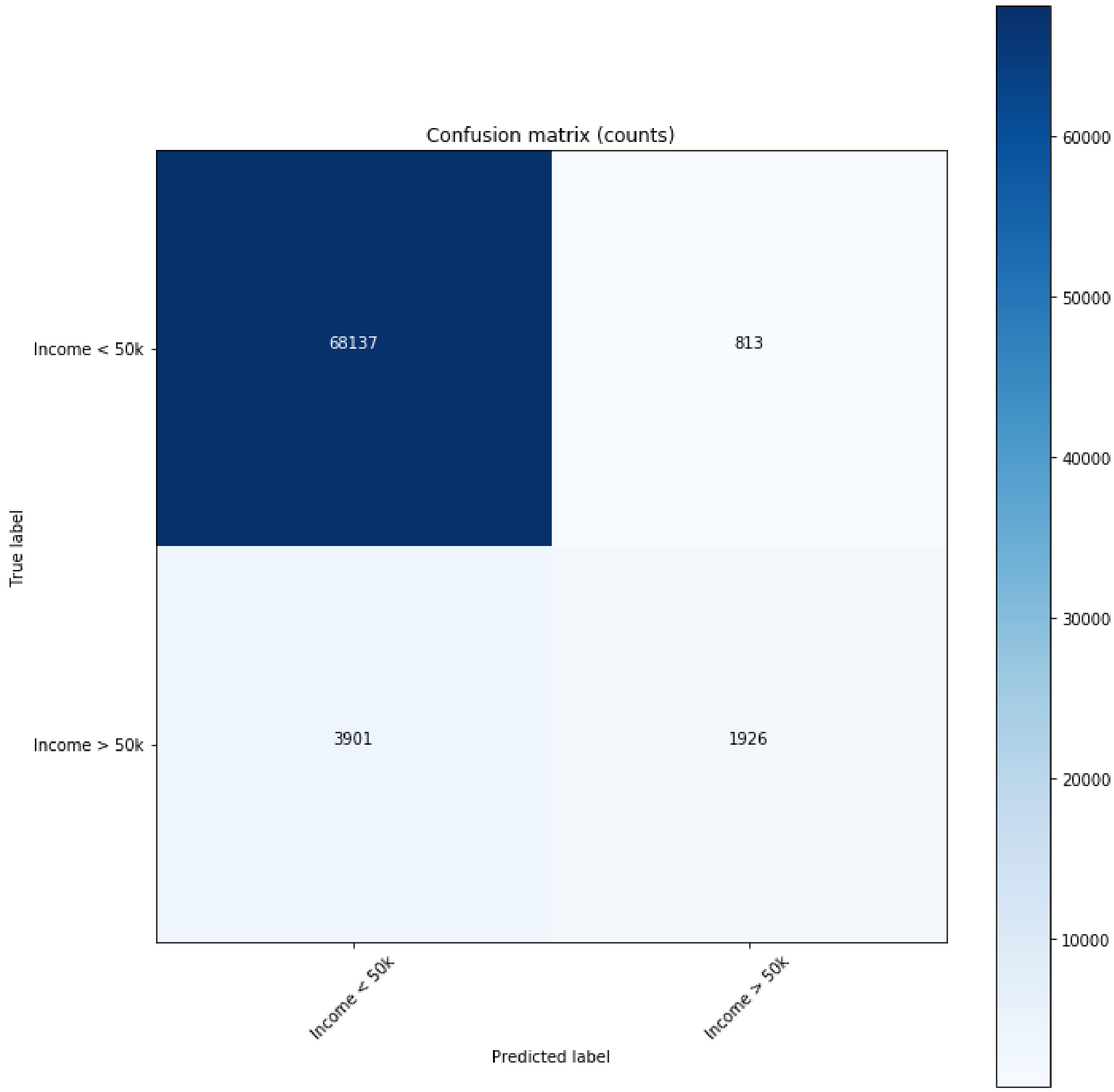
```
In [614]: plt.figure(figsize=(10,10))
          lw = 2
          plt.plot(fpr[0], tpr[0], color='darkorange',
                   lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[0])
          plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
          plt.xlim([0.0, 1.0])
          plt.ylim([0.0, 1.05])
          plt.xlabel('False Positive Rate')
          plt.ylabel('True Positive Rate')
          plt.title('ROC - Logistic Regression')
          plt.legend(loc="lower right")
          plt.show()
```



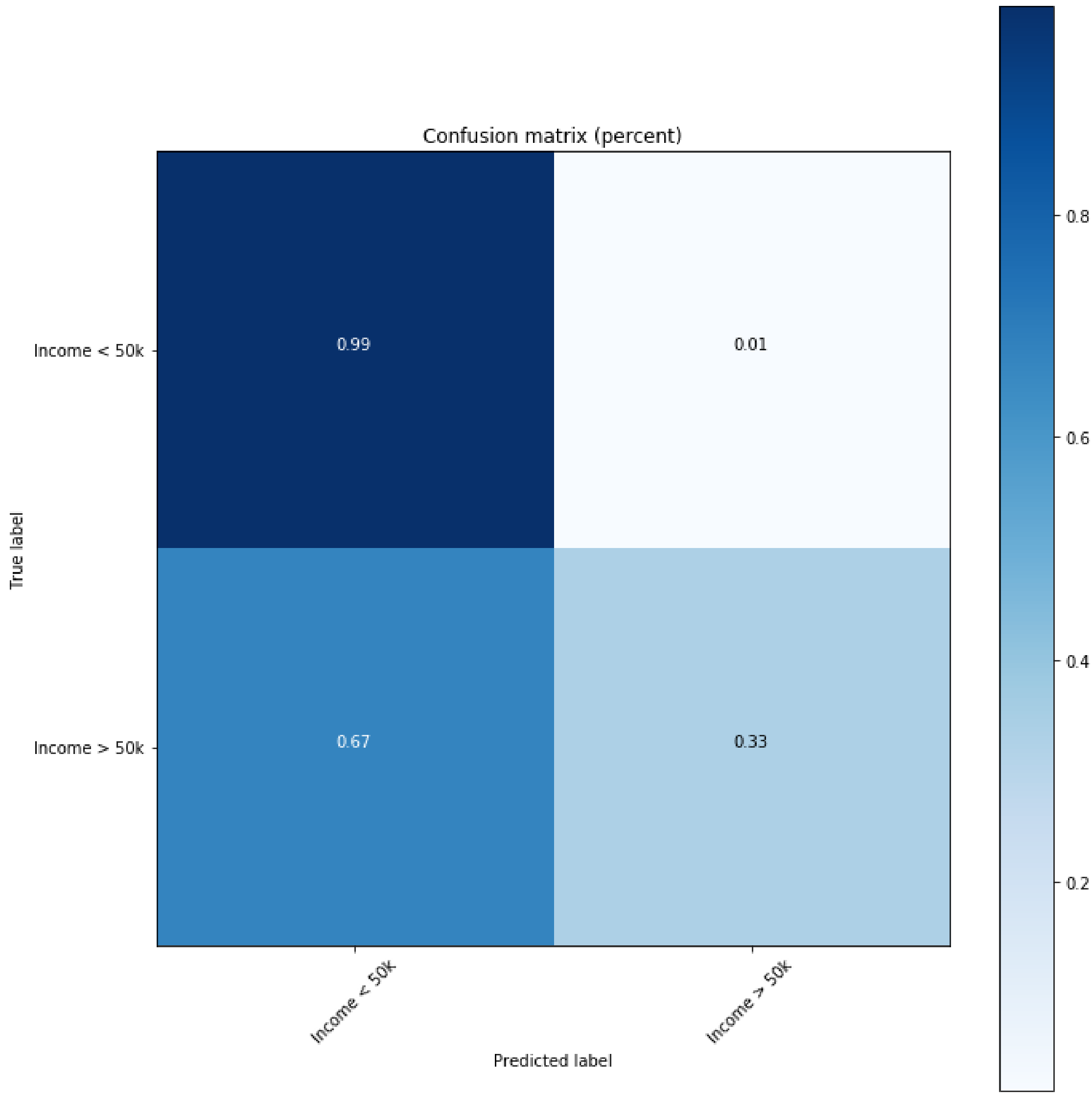
```
In [615]: import itertools
          from sklearn.metrics import confusion_matrix

          def plot_confusion_matrix(cm, classes,
                                    normalize=False,
                                    title='Confusion matrix',
                                    cmap=plt.cm.Blues):
              """
              This function prints and plots the confusion matrix.
              Normalization can be applied by setting `normalize=True`.
              """
              if normalize:
                  cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
                  print("Normalized confusion matrix")
              else:
                  print('Confusion matrix, without normalization')

              print(cm)

              plt.imshow(cm, interpolation='nearest', cmap=cmap)
              plt.title(title)
              plt.colorbar()
              tick_marks = np.arange(len(classes))
              plt.xticks(tick_marks, classes, rotation=45)
              plt.yticks(tick_marks, classes)

              fmt = '.2f' if normalize else 'd'
              thresh = cm.max() / 2.
              for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
                  plt.text(j, i, format(cm[i, j], fmt),
                           horizontalalignment="center",
                           color="white" if cm[i, j] > thresh else "black")

              plt.tight_layout()
              plt.ylabel('True label')
              plt.xlabel('Predicted label')

In [616]: # use trained model to make predictions on test set
          y_pred = LR.predict(X_test)

In [617]: # Compute confusion matrix
          cnf_matrix = confusion_matrix(y_test, y_pred)
          np.set_printoptions(precision=2)

          class_names = ['Income < 50k', 'Income > 50k']
```

```python
# Plot non-normalized confusion matrix
plt.figure(figsize=(10,10))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix (counts)')

# Plot normalized confusion matrix
plt.figure(figsize=(10,10))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Confusion matrix (percent)')
```

```
Confusion matrix, without normalization
[[68137   813]
 [ 3901  1926]]
Normalized confusion matrix
[[ 0.99  0.01]
 [ 0.67  0.33]]
```



Confusion matrix (counts)
Income < 50k
Income > 50k
68137
813
3901
1926
True label
Predicted label
Income < 50k
Income > 50k
60000
50000
40000
30000
20000
10000




```
In [618]: from sklearn.metrics import accuracy_score, precision_score, recall_score

          print("Accuracy: " , "%.2f" % (accuracy_score(y_test, y_pred)*100),'%')
          print("Precision: " , "%.2f" % (precision_score(y_test, y_pred)*100),'%')
          print("Recall: " , "%.2f" % (recall_score(y_test, y_pred)*100),'%')

Accuracy:  93.70 %
Precision:  70.32 %
Recall:  33.05 %
```

Accuracy is 93.70%, i.e. better than simply predicting that everybody make <50k. Precision 70.32% means that 70.32 of the one predicted >50k are actually >50k

Recall 33% means that out of the ones who actually make >50k, the model could only find 33% of them.
Given the skewness of the data, I will to use Oversampling, hoping to get better results.

6.5 Oversampling

Due to the unbalanced data I use SMOTE to oversample the training data.

```
In [619]: from imblearn.over_sampling import SMOTE

          print("Number transactions X_train dataset: ", X_train.shape)
          print("Number transactions y_train dataset: ", y_train.shape)
          print("Number transactions X_test dataset: ", X_test.shape)
          print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (145005, 16)
Number transactions y_train dataset:  (145005,)
Number transactions X_test dataset:  (74777, 16)
Number transactions y_test dataset:  (74777,)
```

```
In [620]: print("Before OverSampling, counts of label '1': {}".format(sum(y_train==1)))
          print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train==0)))

          sm = SMOTE(random_state=2)
          X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())

          print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
          print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

          print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
          print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
```

```
Before OverSampling, counts of label '1': 11684
Before OverSampling, counts of label '0': 133321

After OverSampling, the shape of train_X: (266642, 16)
After OverSampling, the shape of train_y: (266642,)

After OverSampling, counts of label '1': 133321
After OverSampling, counts of label '0': 133321
```

```
In [621]: X_train_res = pd.DataFrame(X_train_res, columns = X_train.columns)
```

```
In [622]: #Using Random Forest for feature selection
          rfc.fit(X_train_res, y_train_res)
          sel = SelectFromModel(rfc, prefit=True)
          selected_feat= X_train_res.columns[(sel.get_support())]
```

```
print('Selected Features:\n',*selected_feat, sep='\n')
X_train_res = X_train_res[selected_feat]
X_test = X_test[selected_feat]
```

```
Selected Features:

Age
Male
StockDividends
NumPersonsWorkedEmployer
WeeksWorkedInY
```

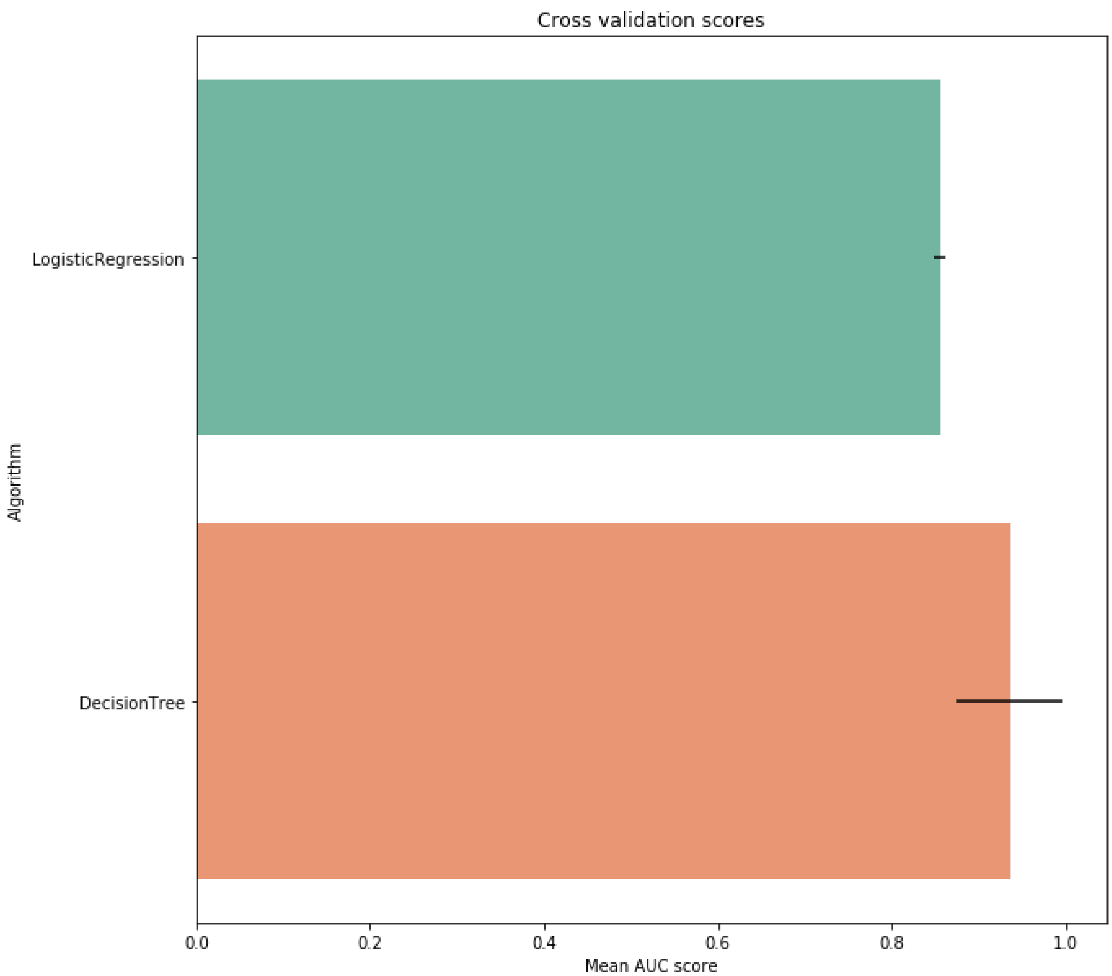
In [623]:
```
#Doing 10-fold cross validation, using Decision Tree and Logistic Regression

rs=42

classifiers = [] # list of classifiers tested
classifiers.append(LogisticRegression(random_state = rs))
classifiers.append(DecisionTreeClassifier(random_state = rs))

cv_results = []
for classifier in classifiers :
    cv_results.append(cross_val_score(classifier, X_train_res, y_train_res, scoring =

cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())

cv_res = pd.DataFrame({'CV_score':cv_means, 'CV_stddev':cv_std, 'Algorithm':['Logisti
```

In [624]:
```
cv_res
```

Out[624]:

	CV_score	CV_stddev	Algorithm
0	0.854965	0.006464	LogisticRegression
1	0.936062	0.060799	DecisionTree

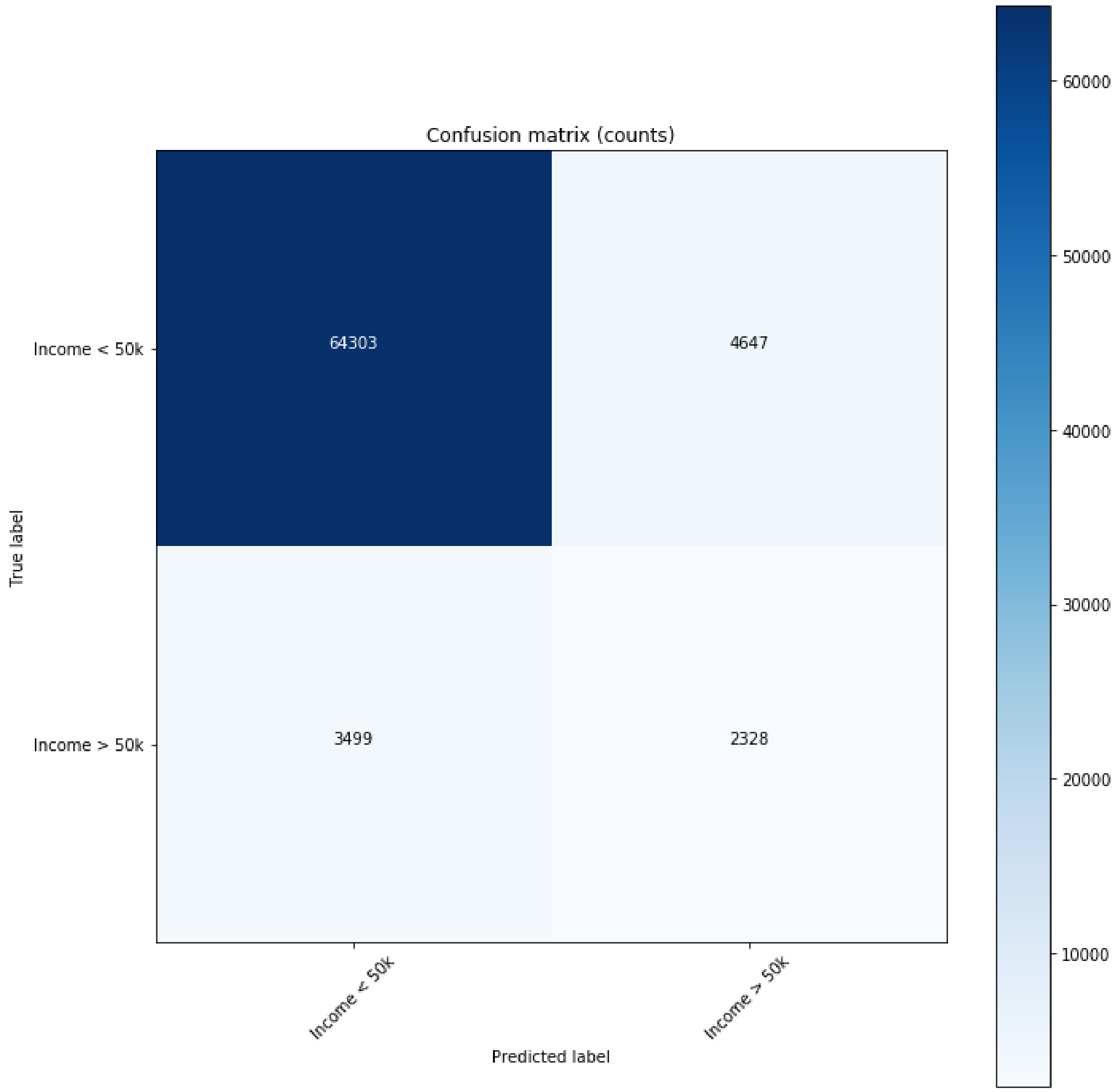
In [625]:
```
plt.subplots(figsize=(10, 10))
g = sns.barplot('CV_score','Algorithm', data = cv_res, palette='Set2', orient = 'h',*
g.set_xlabel('Mean AUC score')
g = g.set_title('Cross validation scores')
```

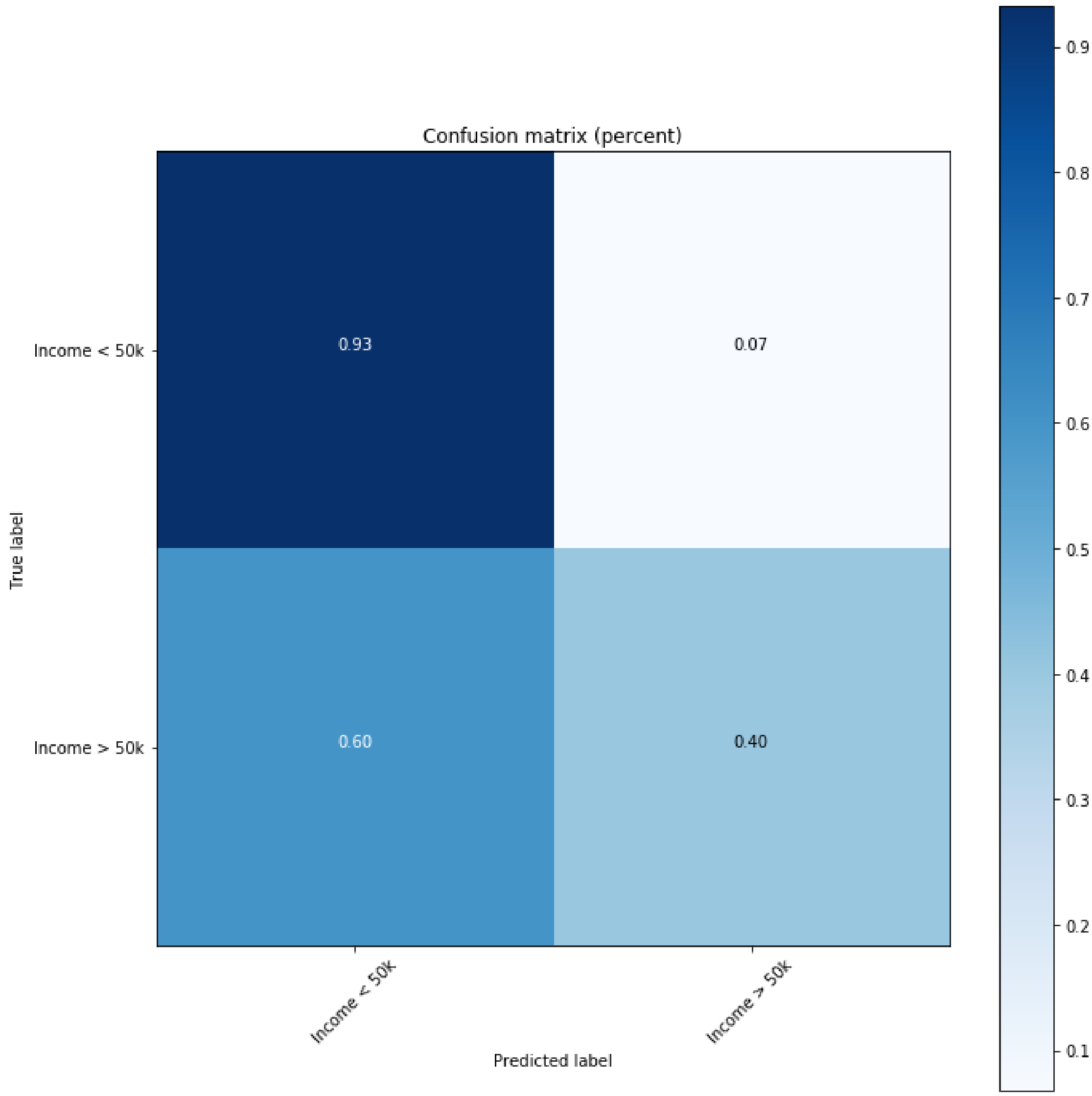



```
In [626]: # define LogisticRegression
          LR = DecisionTreeClassifier(random_state=42)

          # fit LR model to (oversampled) training data
          LR.fit(X_train_res, y_train_res)

Out[626]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort=False, random_state=42,
                      splitter='best')

In [627]: # use trained model to make predictions on test set
          y_pred = LR.predict(X_test)

In [628]: # Compute confusion matrix
          cnf_matrix = confusion_matrix(y_test, y_pred)
```

```python
np.set_printoptions(precision=2)

class_names = ['Income < 50k', 'Income > 50k']

# Plot non-normalized confusion matrix
plt.figure(figsize=(10,10))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix (counts)')

# Plot normalized confusion matrix
plt.figure(figsize=(10,10))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Confusion matrix (percent)')
```

```
Confusion matrix, without normalization
[[64303  4647]
 [ 3499  2328]]
Normalized confusion matrix
[[ 0.93  0.07]
 [ 0.6   0.4 ]]
```


Confusion matrix (counts)
Income < 50k
Income > 50k
True label
64303
4647
3499
2328
Income < 50k
Income > 50k
Predicted label
60000
50000
40000
30000
20000
10000




```
In [629]: print("Accuracy: " , "%.2f" % (accuracy_score(y_test, y_pred)*100),'%')
          print("Precision: " , "%.2f" % (precision_score(y_test, y_pred)*100),'%')
          print("Recall: " , "%.2f" % (recall_score(y_test, y_pred)*100),'%')
```

```
Accuracy:  89.11 %
Precision:  33.38 %
Recall:  39.95 %
```

Recall rate got much better after oversampling, but accuracy and precisoin went down.

6.6 Conclusion

Based on the above two scenarios (with and without oversampling), the user can select the one that matches his requirements. Since Recall and precision varied significantly, then it will depend on what is more important:
1- Making correct predictions => No oversampling
2- Finding as many >50k as possible => Oversampling
3- Correctly predicting >50k => No Oversampling