Table of Contents

Introduction

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Predict whether income exceeds $50K/yr based on census data.

Attribute Information:

1. **age**: continuous.
2. **workclass**: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. **education**: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
4. **education-num**: continuous.
5. **marital-status**: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
6. **occupation**: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
7. **relationship**: Wife, Own-1. capital-loss: continuous.
8. hours-per-week: continuous.
9. native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
10. **fnlwgt**: continuous. The weights on the CPS files are controlled to independent estimates of the civilian noninstitutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau. We use 3 sets of controls. These are:
    A. A single cell estimate of the population 16+ for each state.
    B. Controls for Hispanic Origin by age and sex.
    C. Controls by Race, age and sex.

We use all three sets of controls in our weighting program and "rake" through them 6 times so that by the end we come back to all the controls we used.

The term estimate refers to population totals derived from CPS by creating "weighted tallies" of any specified socio-economic characteristics of the population. People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state.child, Husband, Not-in-family, Other-relative, Unmarried.

1. **race**: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
2. **sex**: Female, Male.
3. **capital-gain: continuous. 1

Importing Libraries

In [1]:
```
# Importing Necessary Libraries
import pandas as pd
import numpy as np
% matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns
```


Get the Data

In [2]:
```
adult_df = pd.read_csv(r'C:\Users\Sreekanth\Documents\Datascience\Adult-Census-Income-master\Adult-Census-Income-master\adult.csv')
print("Number of Observations in adult dataset:", adult_df.shape)

adult_df.head()
```

```
Number of Observations in adult dataset: (32561, 15)
```

Out[2]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900

In [3]:
```
adult_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age               32561 non-null int64
workclass         32561 non-null object
fnlwgt            32561 non-null int64
education         32561 non-null object
education.num     32561 non-null int64
marital.status    32561 non-null object
occupation        32561 non-null object
relationship      32561 non-null object
race              32561 non-null object
sex               32561 non-null object
capital.gain      32561 non-null int64
capital.loss      32561 non-null int64
hours.per.week    32561 non-null int64
native.country    32561 non-null object
income            32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [4]:
```
adult_df.describe()
```

Out[4]:

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

Data Exploration and Visualization

In [5]:
```
#Separate categorical and numberical columns
cat_col = adult_df.dtypes[adult_df.dtypes == 'object']
num_col = adult_df.dtypes[adult_df.dtypes != 'object']
```

In [7]:
```
for col in list(cat_col.index):
    print("--------------------{0}-------------------------".format(col.title()))
    total= adult_df[col].value_counts()
    percent = adult_df[col].value_counts() / adult_df.shape[0]
    df = pd.concat([total,percent],keys = ['total','percent'],axis = 1)
    print(df)
    print('\n')
```

```
--------------------Workclass-------------------------
                  total   percent
Private           22696  0.697030
Self-emp-not-inc   2541  0.078038
Local-gov          2093  0.064279
?                  1836  0.056386
State-gov          1298  0.039864
Self-emp-inc       1116  0.034274
Federal-gov         960  0.029483
Without-pay          14  0.000430
Never-worked          7  0.000215


--------------------Education-------------------------
              total   percent
HS-grad       10501  0.322502
Some-college   7291  0.223918
Bachelors      5355  0.164461
Masters        1723  0.052916
Assoc-voc      1382  0.042443
11th           1175  0.036086
Assoc-acdm     1067  0.032769
10th            933  0.028654
7th-8th         646  0.019840
Prof-school     576  0.017690
9th             514  0.015786
12th            433  0.013298
Doctorate       413  0.012684
5th-6th         333  0.010227
1st-4th         168  0.005160
Preschool        51  0.001566


--------------------Marital.Status-------------------------
                     total   percent
Married-civ-spouse     14976  0.459937
Never-married          10683  0.328092
Divorced                4443  0.136452
Separated               1025  0.031479
Widowed                  993  0.030497
Married-spouse-absent    418  0.012837
Married-AF-spouse         23  0.000706
```

```
--------------------Occupation-------------------------
                   total   percent
Prof-specialty      4140  0.127146
Craft-repair        4099  0.125887
Exec-managerial     4066  0.124873
Adm-clerical        3770  0.115783
Sales               3650  0.112097
Other-service       3295  0.101195
Machine-op-inspct   2002  0.061485
?                   1843  0.056601
Transport-moving    1597  0.049046
Handlers-cleaners   1370  0.042075
Farming-fishing      994  0.030527
Tech-support         928  0.028500
Protective-serv      649  0.019932
Priv-house-serv      149  0.004576
Armed-Forces           9  0.000276


--------------------Relationship-------------------------
                total   percent
Husband         13193  0.405178
Not-in-family    8305  0.255060
Own-child        5068  0.155646
Unmarried        3446  0.105832
Wife             1568  0.048156
Other-relative    981  0.030128


--------------------Race-------------------------
                    total   percent
White               27816  0.854274
Black                3124  0.095943
Asian-Pac-Islander   1039  0.031909
Amer-Indian-Eskimo    311  0.009551
Other                 271  0.008323


--------------------Sex-------------------------
        total   percent
Male    21790  0.669205
Female  10771  0.330795
```

```
--------------------Native.Country-------------------------
                            total   percent
United-States               29170  0.895857
Mexico                        643  0.019748
?                             583  0.017905
Philippines                   198  0.006081
Germany                       137  0.004207
Canada                        121  0.003716
Puerto-Rico                   114  0.003501
El-Salvador                   106  0.003255
India                         100  0.003071
Cuba                           95  0.002918
England                        90  0.002764
Jamaica                        81  0.002488
South                          80  0.002457
China                          75  0.002303
Italy                          73  0.002242
Dominican-Republic             70  0.002150
Vietnam                        67  0.002058
Guatemala                      64  0.001966
Japan                          62  0.001904
Poland                         60  0.001843
Columbia                       59  0.001812
Taiwan                         51  0.001566
Haiti                          44  0.001351
Iran                           43  0.001321
Portugal                       37  0.001136
Nicaragua                      34  0.001044
Peru                           31  0.000952
France                         29  0.000891
Greece                         29  0.000891
Ecuador                        28  0.000860
Ireland                        24  0.000737
Hong                           20  0.000614
Cambodia                       19  0.000584
Trinadad&Tobago                19  0.000584
Laos                           18  0.000553
Thailand                       18  0.000553
Yugoslavia                     16  0.000491
Outlying-US(Guam-USVI-etc)     14  0.000430
Hungary                        13  0.000399
```

```
Honduras                     13  0.000399
Scotland                     12  0.000369
Holand-Netherlands            1  0.000031


--------------------Income-------------------------
       total  percent
<=50K  24720  0.75919
>50K    7841  0.24081
```

Native.Country, Occupation, Workclass

- has unknown values represented by ?

Education

- 9th, 10th, 11th, 12th comes under HighSchool Grad but it has mentioned separately
- Create Elementary object for 1st-4th, 5th-6th, 7th-8th

Marital Status

- Married-civ-spouse,Married-spouse-absent,Married-AF-spouse comes under category Married
- Divorced, separated again comes under category separated.

Workclass

- Self-emp-not-inc, Self-emp-inc comes under category self employed
- Local-gov,State-gov,Federal-gov comes under category goverment emloyees

In [8]:
```
edit_cols = ['native.country','occupation','workclass']
# Replace ? with Unknown
for col in edit_cols:
    adult_df.loc[adult_df[col] == '?', col] = 'unknown'
```

In [11]:
```
# Check if ? is present
for col in edit_cols:
    print("? in {0}: {1}".format(col,adult_df[(adult_df[col] == '?')].any().sum() ))
```

```
? in native.country: 0
? in occupation: 0
? in workclass: 0
```

In [12]:
```
hs_grad = ['HS-grad','11th','10th','9th','12th']
elementary = ['1st-4th','5th-6th','7th-8th']

# replace elements in list.
adult_df['education'].replace(to_replace = hs_grad,value = 'HS-grad',inplace = True)
adult_df['education'].replace(to_replace = elementary,value = 'elementary_school',inplace = True)

adult_df['education'].value_counts()
```

Out[12]:
```
HS-grad              13556
Some-college          7291
Bachelors             5355
Masters               1723
Assoc-voc             1382
elementary_school     1147
Assoc-acdm            1067
Prof-school            576
Doctorate              413
Preschool               51
Name: education, dtype: int64
```

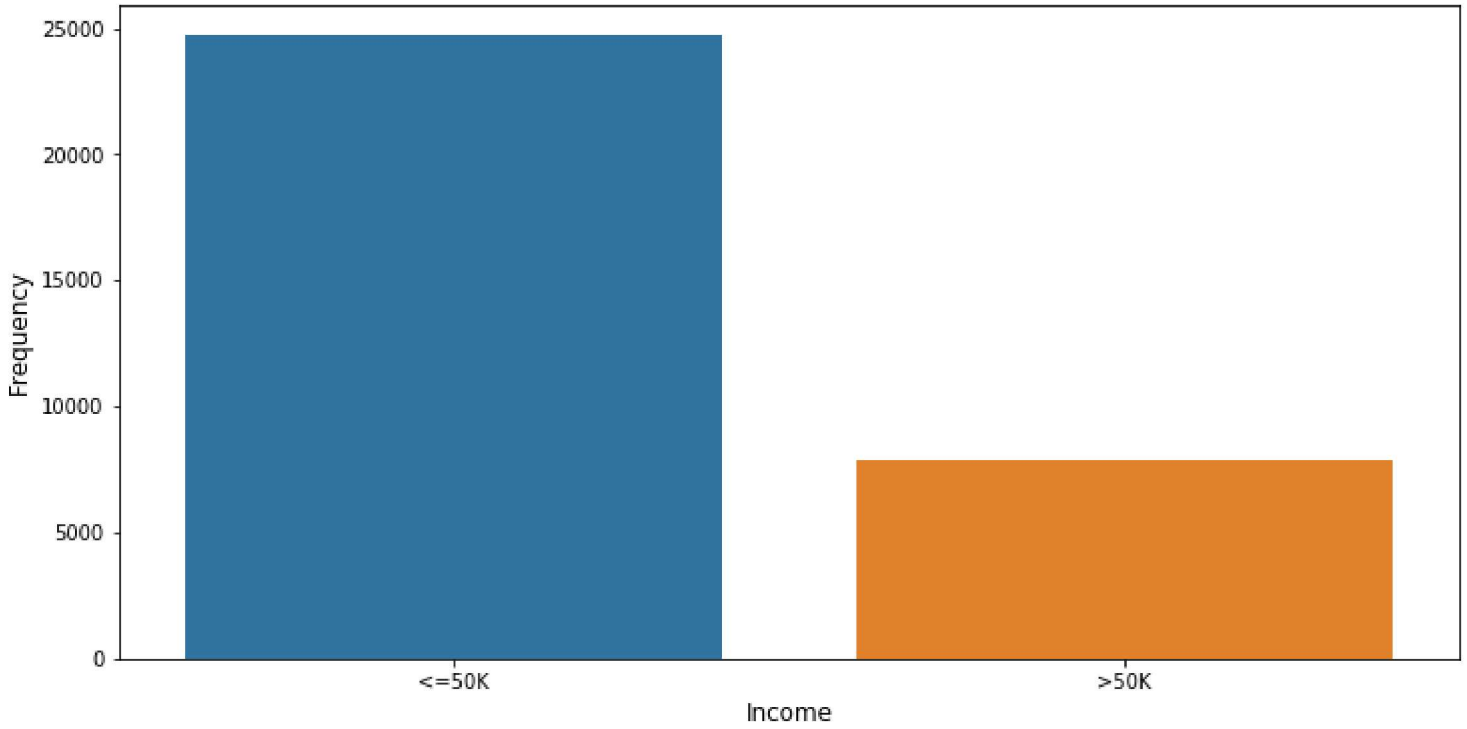
In [13]:
```
married= ['Married-spouse-absent','Married-civ-spouse','Married-AF-spouse']
separated = ['Separated','Divorced']

#replace elements in list.
adult_df['marital.status'].replace(to_replace = married ,value = 'Married',inplace = True)
adult_df['marital.status'].replace(to_replace = separated,value = 'Separated',inplace = True)

adult_df['marital.status'].value_counts()
```

Out[13]:
```
Married          15417
Never-married    10683
Separated         5468
Widowed            993
Name: marital.status, dtype: int64
```

In [15]:
```
self_employed = ['Self-emp-not-inc','Self-emp-inc']
govt_employees = ['Local-gov','State-gov','Federal-gov']

#replace elements in list.
adult_df['workclass'].replace(to_replace = self_employed ,value = 'Self_employed',inplace = True)
adult_df['workclass'].replace(to_replace = govt_employees,value = 'Govt_employees',inplace = True)

adult_df['workclass'].value_counts()
```

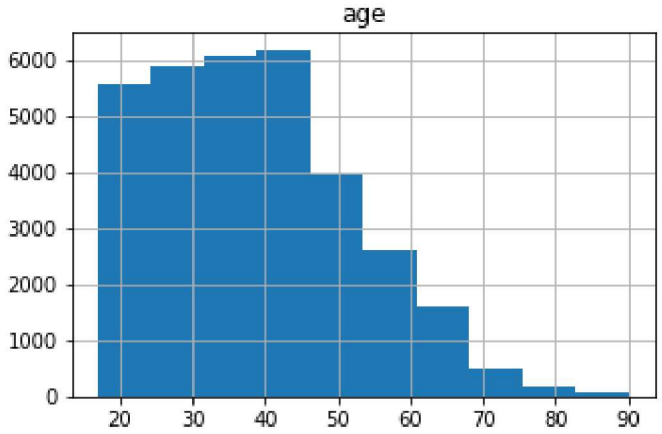
Out[15]:
```
Private           22696
Govt_employees     4351
Self_employed      3657
unknown            1836
Without-pay          14
Never-worked          7
Name: workclass, dtype: int64
```

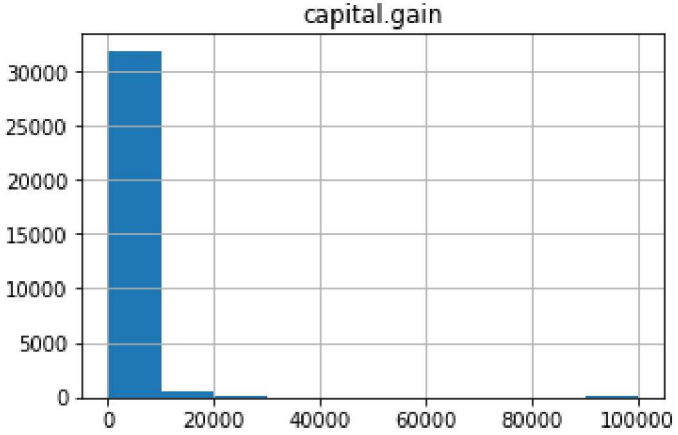
In [16]:
```
plt.figure(figsize =(12,6));
sns.countplot(x = 'income', data = adult_df);
plt.xlabel("Income",fontsize = 12);
plt.ylabel("Frequency",fontsize = 12);
```

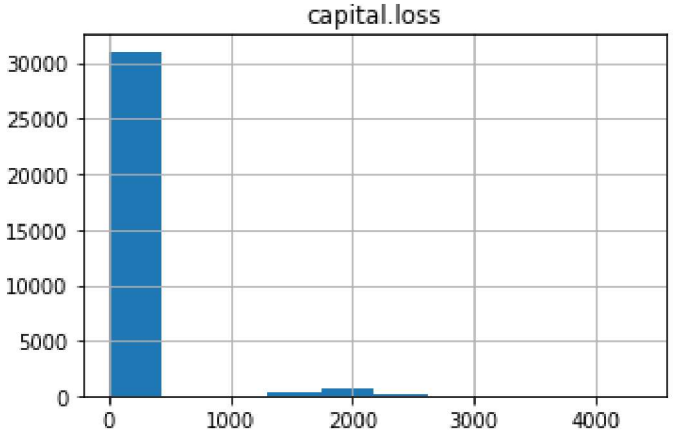


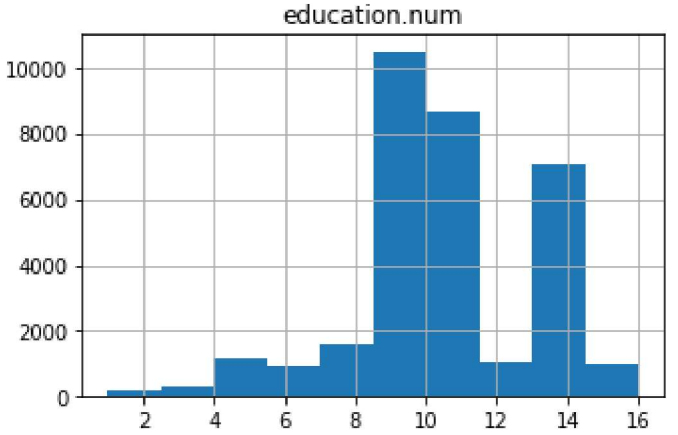
- Our dataset has 25000 people earning <=50K i.e. 75% and remainng 25% earns more than 50K.
- Let's explore distribution of numerical features.

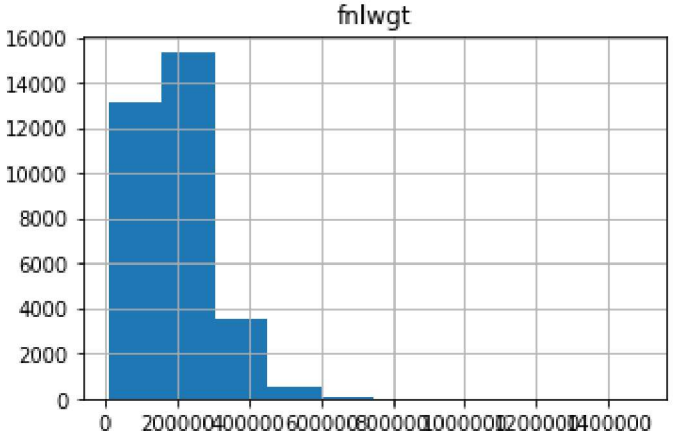
In [17]:
```
adult_df[list(num_col.index)].hist(figsize = (12,12));
```

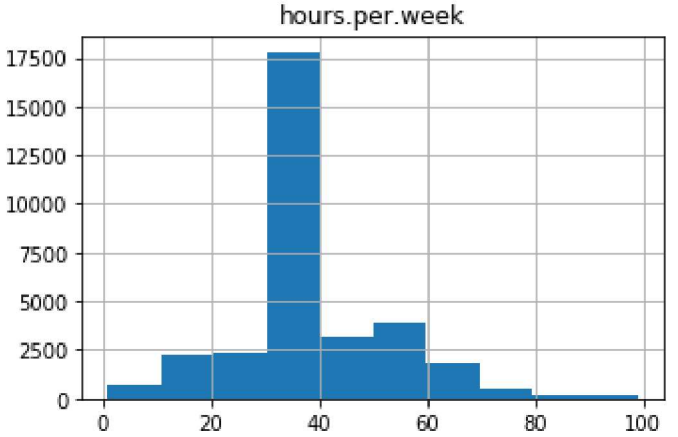

age
6000
5000
4000
3000
2000
1000
0
20
30
40
50
60
70
80
90


capital.gain
30000
25000
20000
15000
10000
5000
0
0
20000
40000
60000
80000
100000


capital.loss
30000
25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000


education.num
10000
8000
6000
4000
2000
0
2
4
6
8
10
12
14
16


fnlwgt
16000
14000
12000
10000
8000
6000
4000
2000
0
0
200000
400000
600000
800000
1000000
1200000
1400000


hours.per.week
17500
15000
12500
10000
7500
5000
2500
0
0
20
40
60
80
100

- age : In our dataset People from age group of 18 to 50 can be observed.
- hours per week : Generally people can be seen working for 30 hours to 40 hours per week.
- education : people above 8th grade are more in our dataset.
- Our dataset is highly skewed and we should consider scaling it for better performance of our models.

Let's explore more for people having capital loss and capital gain greater than zero.

In [20]:
```
capital_loss_df = adult_df[adult_df['capital.loss']>0]
capital_gain_df = adult_df[adult_df['capital.gain']>0]

print("Number of observations having capital loss above median value: {0}\nNumber of observations in capital
 gain dataset above median value: {1}".format(capital_loss_df.shape,capital_gain_df.shape))
print("Percentage of people having capital gain greater than median value: {0:.4f}%".format((adult_df.loc[adu
lt_df['capital.gain'] > 0,:].shape[0] / adult_df.shape[0])*100))
print("Percentage of people having capital loss greater than median value: {0:.4f}%".format((adult_df.loc[adu
lt_df['capital.loss'] > 0,:].shape[0] / adult_df.shape[0])*100))
```

```
Number of observations having capital loss above median value: (1519, 15)
Number of observations in capital gain dataset above median value: (2712, 15)
Percentage of people having capital gain greater than median value: 8.3290%
Percentage of people having capital loss greater than median value: 4.6651%
```

In [21]:
```
capital_gain_0= adult_df[adult_df['capital.loss'] > 0].loc[:,['capital.loss','capital.gain']].sample(10)

capital_loss_0 = adult_df[adult_df['capital.gain'] > 0].loc[:,['capital.loss','capital.gain']].sample(5)
print(capital_gain_0.head())
print(capital_loss_0.head())
```

```
      capital.loss  capital.gain
1014          1741             0
1303          1590             0
243           2080             0
1108          1719             0
604           1902             0
      capital.loss  capital.gain
1568             0         99999
4036             0          2174
1745             0         20051
2204             0         13550
3325             0          4386
```

- That means 92% of people are having capital gain equal to zero.

Possibilities for capital gain and capital loss

- Both capital gain and capital loss can be zero
- If capital.gain is zero there is possibility of capital loss being high or above zero.
- If capital loss is zero there is possibility of capital.gain being high or above zero.

In [24]:
```
# Exploring Case when capital.gain and capital loss both are zero. As this category of people are more in our
dataset.
print("Number of observations having capital gain and capital loss zero: {0}".format(adult_df[(adult_df['capi
tal.loss'] == 0) & (adult_df['capital.gain'] == 0)].shape))
for col in cat_col.index:
    print("================================{0}================================".format(col))
    print(adult_df[(adult_df['capital.loss'] == 0) & (adult_df['capital.gain'] == 0)][col].value_counts())
```

```
Number of observations having capital gain and capital loss zero: (28330, 15)
================================workclass=================================
Private           19982
Govt_employees     3714
Self_employed      2960
unknown            1655
Without-pay          12
Never-worked          7
Name: workclass, dtype: int64
================================education=================================
HS-grad              12246
Some-college          6533
Bachelors             4384
Masters               1300
Assoc-voc             1194
elementary_school     1049
Assoc-acdm             930
Prof-school            363
Doctorate              284
Preschool               47
Name: education, dtype: int64
==============================marital.status==============================
Married          12603
Never-married     9914
Separated         4934
Widowed            879
Name: marital.status, dtype: int64
================================occupation================================
Craft-repair         3593
Adm-clerical         3408
Prof-specialty       3290
Exec-managerial      3219
Sales                3138
Other-service        3122
Machine-op-inspct    1806
unknown              1662
Transport-moving     1416
Handlers-cleaners    1274
Farming-fishing       890
Tech-support          795
Protective-serv       570
Priv-house-serv       139
Armed-Forces            8
```

```
Name: occupation, dtype: int64
================================relationship================================
Husband           10739
Not-in-family      7427
Own-child          4810
Unmarried          3172
Wife               1272
Other-relative      910
Name: relationship, dtype: int64
================================race================================
White                24061
Black                 2839
Asian-Pac-Islander     902
Amer-Indian-Eskimo     280
Other                  248
Name: race, dtype: int64
================================sex================================
Male      18551
Female     9779
Name: sex, dtype: int64
================================native.country================================
United-States                25320
Mexico                         612
unknown                        493
Philippines                    174
Germany                        117
Puerto-Rico                    103
Canada                         103
El-Salvador                     95
Cuba                            85
India                           79
Jamaica                         78
England                         78
South                           68
Dominican-Republic              67
Italy                           65
China                           64
Guatemala                       60
Vietnam                         57
Columbia                        55
Poland                          53
Japan                           51
Taiwan                          44
```

```
Haiti                          42
Portugal                       35
Iran                           35
Nicaragua                      30
Peru                           29
France                         26
Ecuador                        25
Ireland                        21
Greece                         20
Hong                           19
Thailand                       18
Laos                           17
Trinadad&Tobago                17
Yugoslavia                     15
Outlying-US(Guam-USVI-etc)     14
Cambodia                       14
Honduras                       12
Scotland                       11
Hungary                         9
Name: native.country, dtype: int64
================================income================================
<=50K    22939
>50K      5391
Name: income, dtype: int64
```

- So 88% of people in our dataset has capital gain and loss = 0
- This concludes capital gain and capital loss are highly skewed features and even have outliers which need to be taken care of.
- Workclass contains unknown values.

Let's find more about capital gain by excluding zero value.

In [ ]:
```
adult_df.loc[adult_df['capital.gain'] > 0,:].describe()
```

- Maximum value of capital gain is 99999 which is far above 75% quartile range.This is definitely an outlier. Lets try and understand more about this maximum value of 99999.

In [25]:
```
print("Number of observations having capital gain of 99999:{0}".format(adult_df.loc[adult_df['capital.gain'] == 99999,:].shape))
print("Income counts: {0}".format(adult_df.loc[adult_df['capital.gain'] == 99999,:]['income'].value_counts()))
```

```
Number of observations having capital gain of 99999:(159, 15)
Income counts: >50K    159
Name: income, dtype: int64
```

- Ok,So their income is greater than 50K which is exactly what I expected.

In [26]:
```
adult_df.loc[adult_df['capital.loss'] > 0,:].describe()
```

Out[26]:

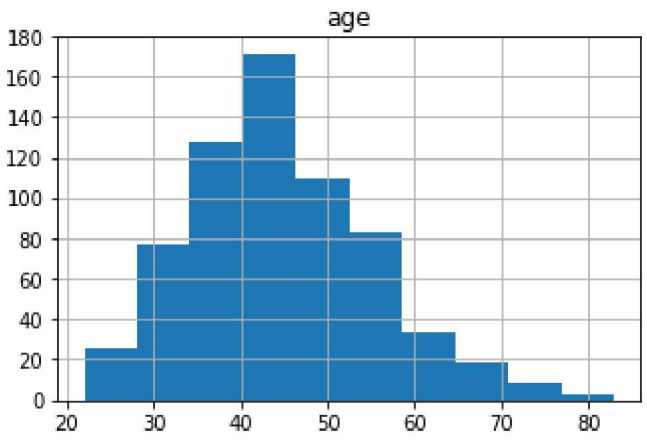
	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	1519.000000	1519.000000	1519.000000	1519.0	1519.000000	1519.000000
mean	41.697828	185492.941409	10.969059	0.0	1871.428571	43.252798
std	12.625609	97621.521906	2.688426	0.0	376.571535	12.247258
min	17.000000	20953.000000	1.000000	0.0	155.000000	1.000000
25%	32.000000	118854.000000	9.000000	0.0	1672.000000	40.000000
50%	41.000000	175109.000000	10.000000	0.0	1887.000000	40.000000
75%	50.000000	228991.500000	13.000000	0.0	1977.000000	50.000000
max	90.000000	816750.000000	16.000000	0.0	4356.000000	99.000000

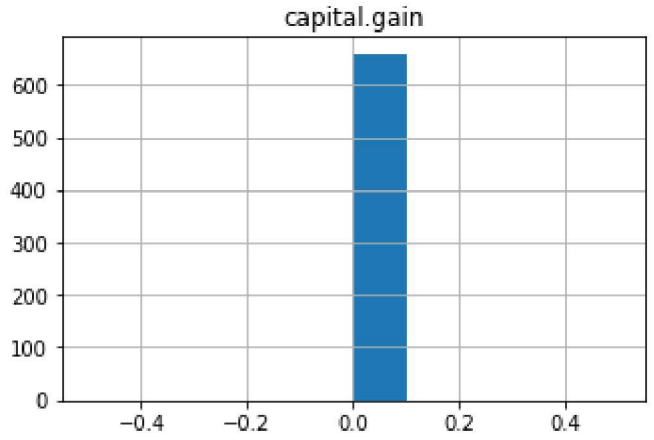
In [28]:
```
print("Number of observations having capital loss of 4356:{0}".format(adult_df.loc[adult_df['capital.loss'] ==
= 4356,:].shape))
print("\nIncome Distribution among people with capital loss above mean:")
print("Income counts:\n{0}".format(adult_df.loc[adult_df['capital.loss'] >= 1871,:]['income'].value_counts
()))
```

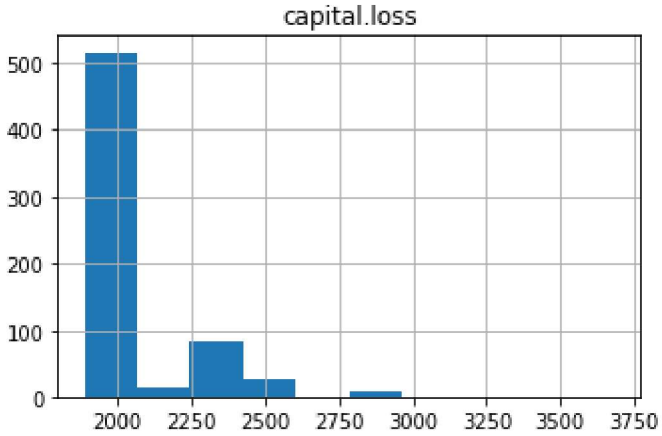
```
Number of observations having capital loss of 4356:(3, 15)

Income Distribution among people with capital loss above mean:
Income counts:
>50K     658
<=50K    277
Name: income, dtype: int64
```

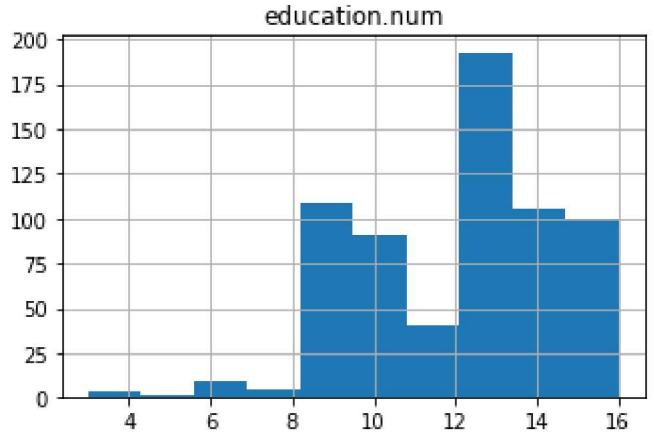
- People with Income above 50K are having capital loss above mean value.
- It shows there are some other parameters affecting their loss irrespective of high salary.

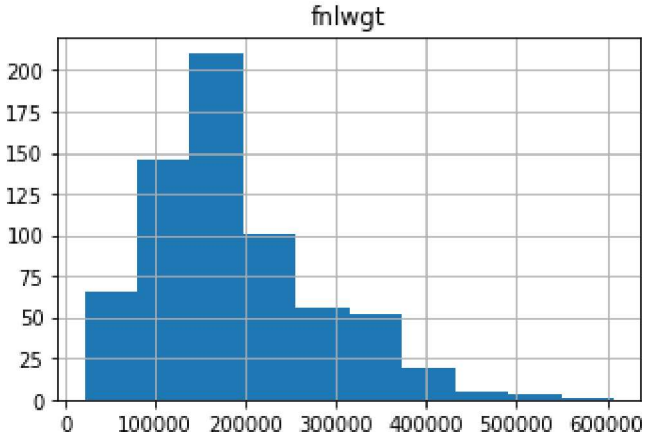
In [29]:
```
# let's understand characteristics of people having capital loss greater than mean value and having income gr
eater than 50K
adult_df[(adult_df['capital.loss'] >= 1871) & (adult_df['income'] == '>50K')].hist(figsize = (12,12));
```

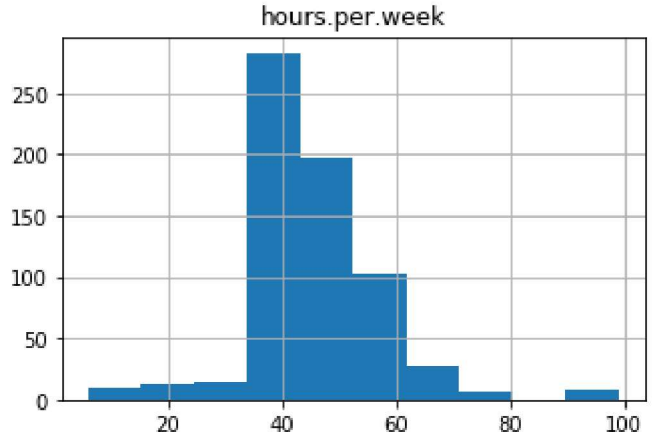

age
180
160
140
120
100
80
60
40
20
0
20
30
40
50
60
70
80


capital.gain
600
500
400
300
200
100
0
−0.4
−0.2
0.0
0.2
0.4


capital.loss
500
400
300
200
100
0
2000
2250
2500
2750
3000
3250
3500
3750

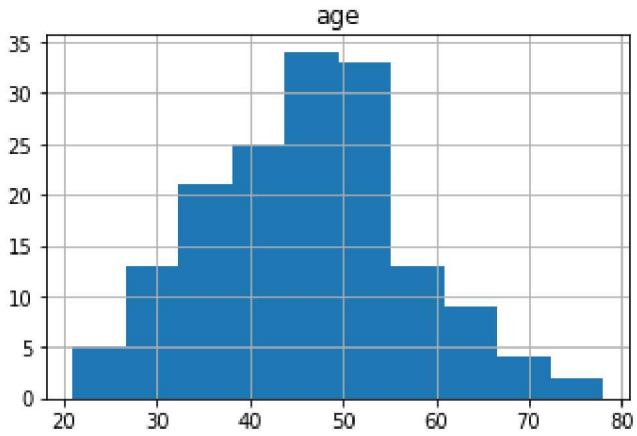

education.num
200
175
150
125
100
75
50
25
0
4
6
8
10
12
14
16

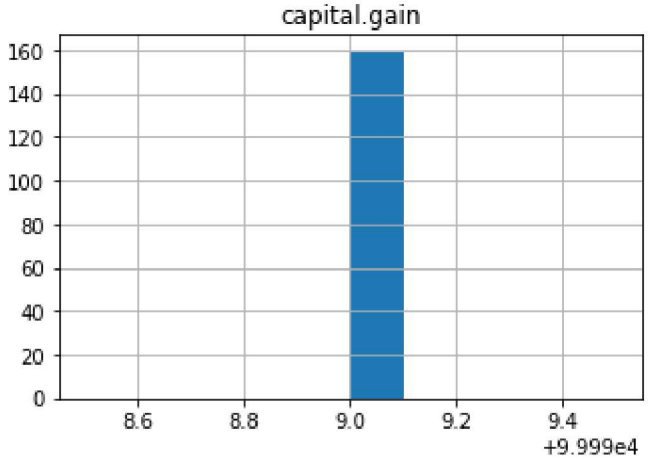

fnlwgt
200
175
150
125
100
75
50
25
0
0
100000
200000
300000
400000
500000
600000

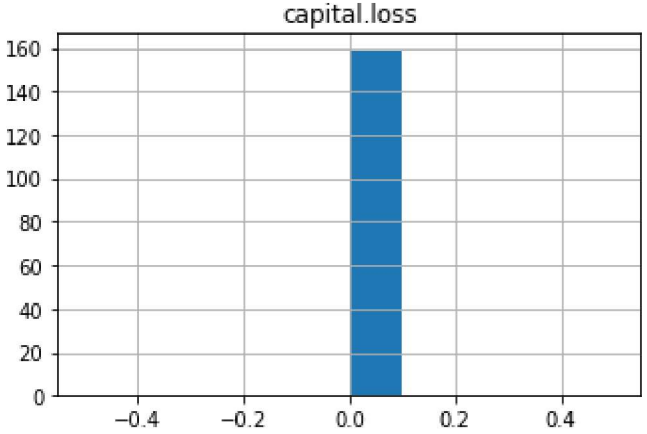

hours.per.week
250
200
150
100
50
0
20
40
60
80
100

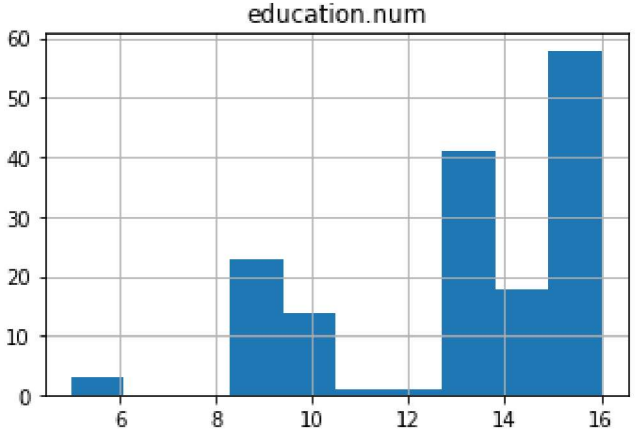
- Education and hours per week are significantaly high.
- There are some other factors affecting their loss other than the above.

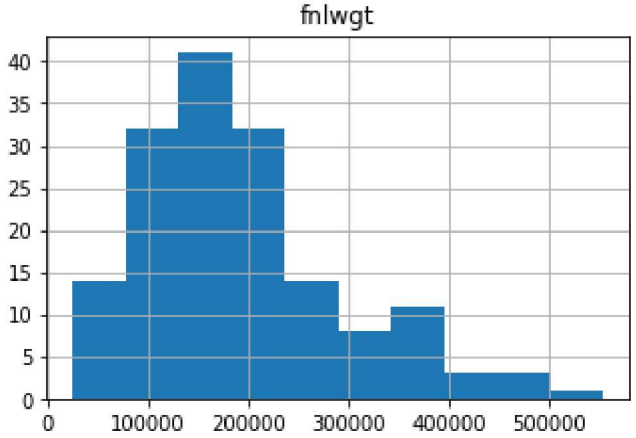
In [30]:
```
adult_df.loc[adult_df['capital.gain'] == 99999,:].hist(figsize = (12,12));
```

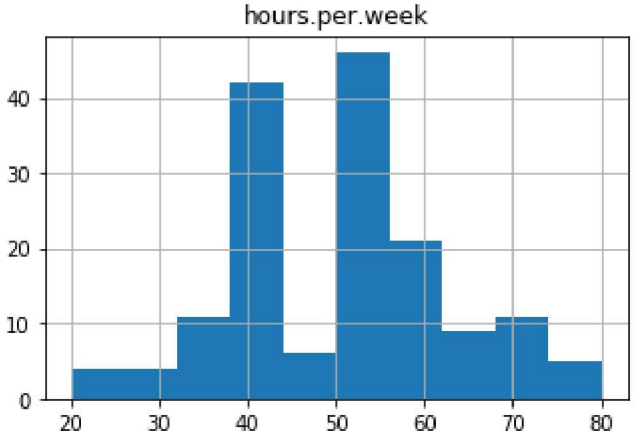

age
0
5
10
15
20
25
30
35
20
30
40
50
60
70
80


capital.gain
0
20
40
60
80
100
120
140
160
8.6
8.8
9.0
9.2
9.4
+9.999e4


capital.loss
0
20
40
60
80
100
120
140
160
−0.4
−0.2
0.0
0.2
0.4

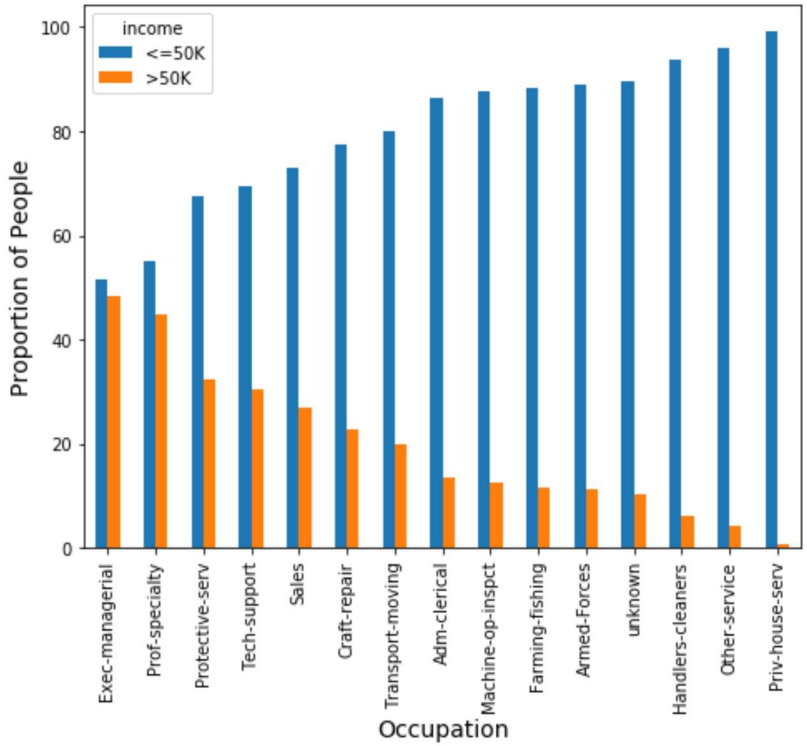

education.num
0
10
20
30
40
50
60
6
8
10
12
14
16

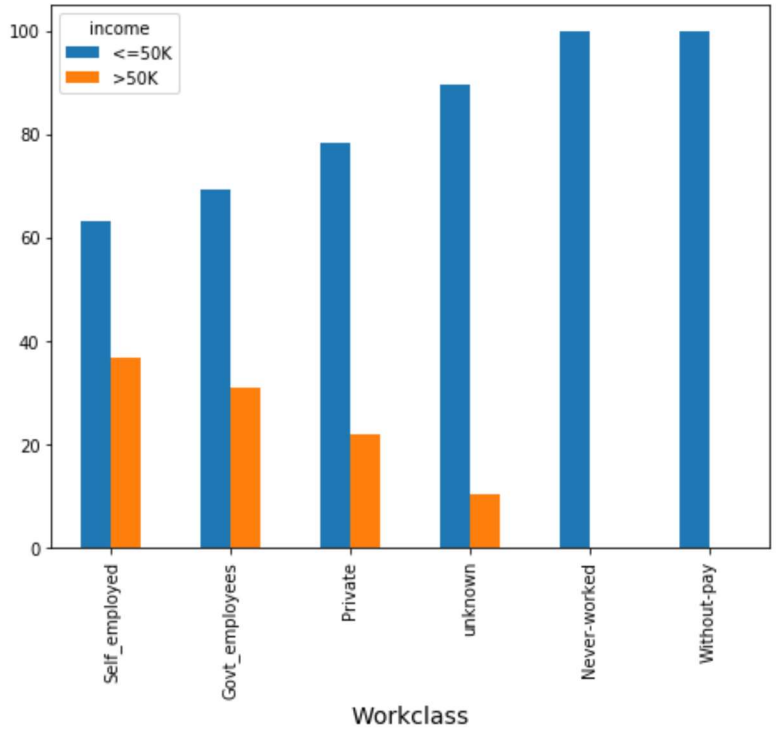

fnlwgt
0
5
10
15
20
25
30
35
40
0
100000
200000
300000
400000
500000


hours.per.week
0
10
20
30
40
20
30
40
50
60
70
80

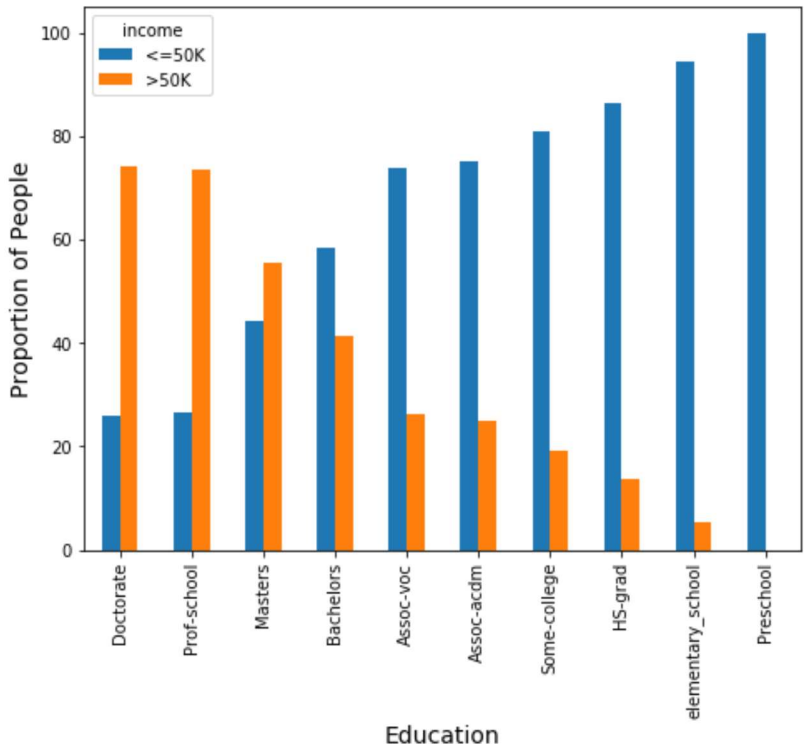
- Age group and hours per week are fairly distributed for capital gain of 99999.

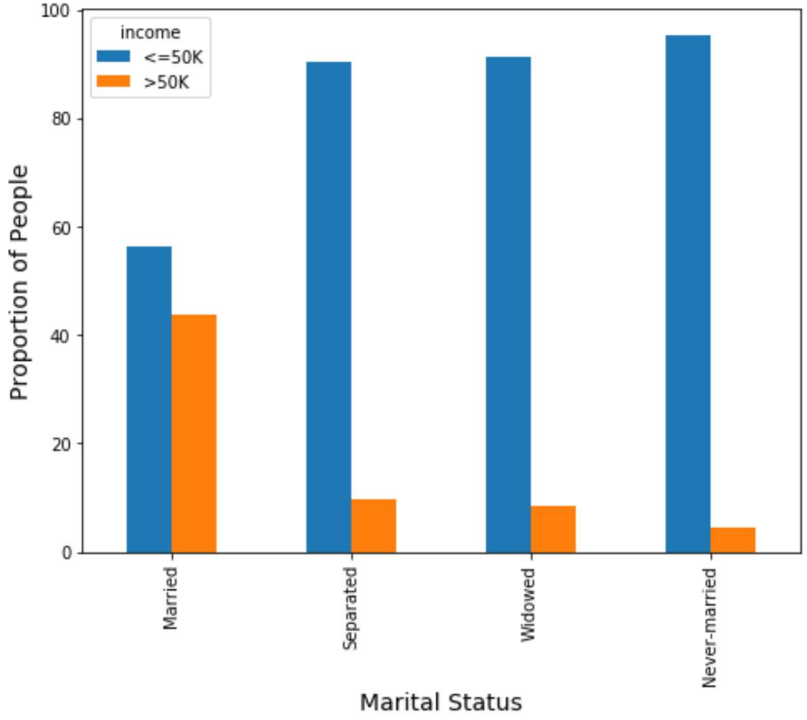
In [31]:
```
table_occu = pd.crosstab(adult_df['occupation'], adult_df['income'])
table_workclass = pd.crosstab(adult_df['workclass'], adult_df['income'])
table_edu = pd.crosstab(adult_df['education'], adult_df['income'])
table_marital = pd.crosstab(adult_df['marital.status'], adult_df['income'])
table_race = pd.crosstab(adult_df['race'], adult_df['income'])
table_sex = pd.crosstab(adult_df['sex'], adult_df['income'])
table_country = pd.crosstab(adult_df['native.country'], adult_df['income'])

fig = plt.figure(figsize = (17,6))

ax = fig.add_subplot(1,2,1)
(table_occu.div(table_occu.sum(axis= 1),axis = 0)*100).sort_values(by= '<=50K').plot(kind = 'bar',ax=ax);
plt.xlabel("Occupation",fontsize = 14);
plt.ylabel('Proportion of People',fontsize = 14);


ax = fig.add_subplot(1,2,2)
(table_workclass.div(table_workclass.sum(axis = 1),axis = 0)*100).sort_values(by = '<=50K').plot(kind = 'bar'
,ax=ax);
plt.xlabel("Workclass",fontsize = 14);
```


income
<=50K
>50K
Proportion of People
0
20
40
60
80
100
Exec-managerial
Prof-specialty
Protective-serv
Tech-support
Sales
Craft-repair
Transport-moving
Adm-clerical
Machine-op-inspct
Farming-fishing
Armed-Forces
unknown
Handlers-cleaners
Other-service
Priv-house-serv
Occupation

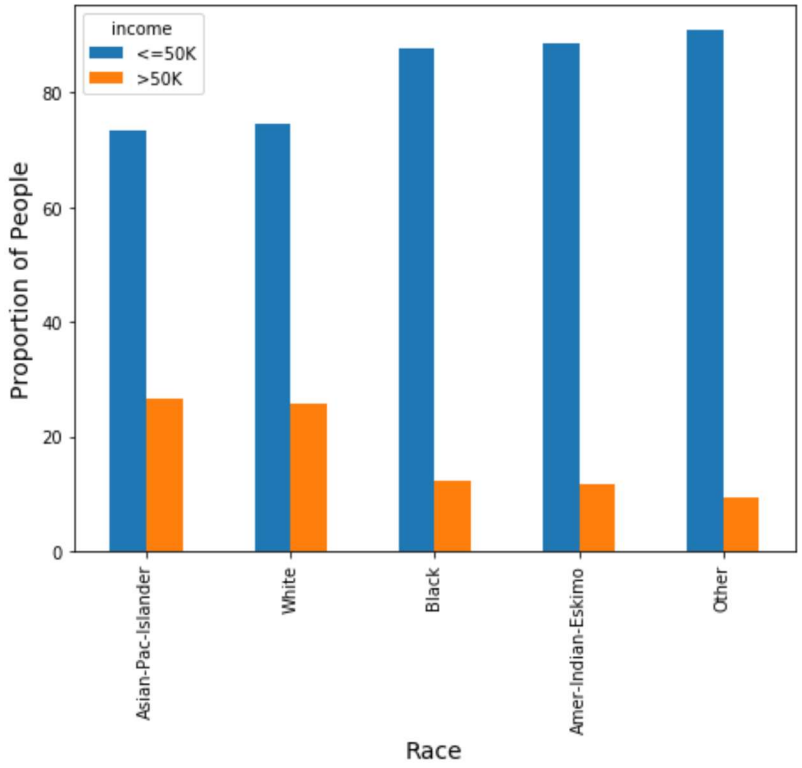

income
<=50K
>50K
0
20
40
60
80
100
Self_employed
Govt_employees
Private
unknown
Never-worked
Without-pay
Workclass

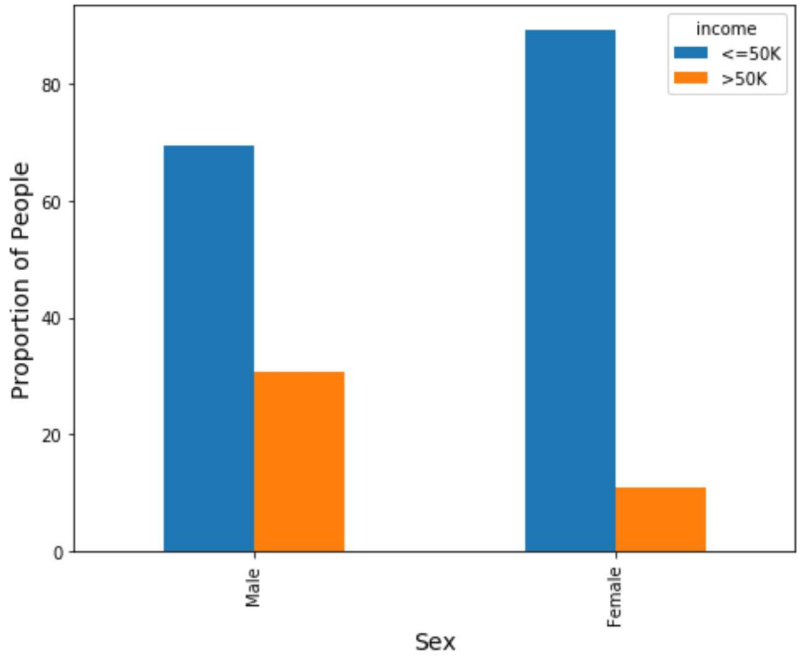
In [32]:
```
fig = plt.figure(figsize = (17,6))
ax = fig.add_subplot(1,2,1)
(table_edu.div(table_edu.sum(axis = 1),axis = 0)*100).sort_values(by = '<=50K').plot(kind = 'bar',ax =ax);
plt.xlabel('Education',fontsize = 14);
plt.ylabel('Proportion of People',fontsize = 14);


ax = fig.add_subplot(1,2,2)
(table_marital.div(table_marital.sum(axis = 1),axis = 0)*100).sort_values(by = '<=50K').plot(kind = 'bar',ax
= ax);
plt.xlabel('Marital Status',fontsize = 14);
plt.ylabel('Proportion of People',fontsize = 14);
```



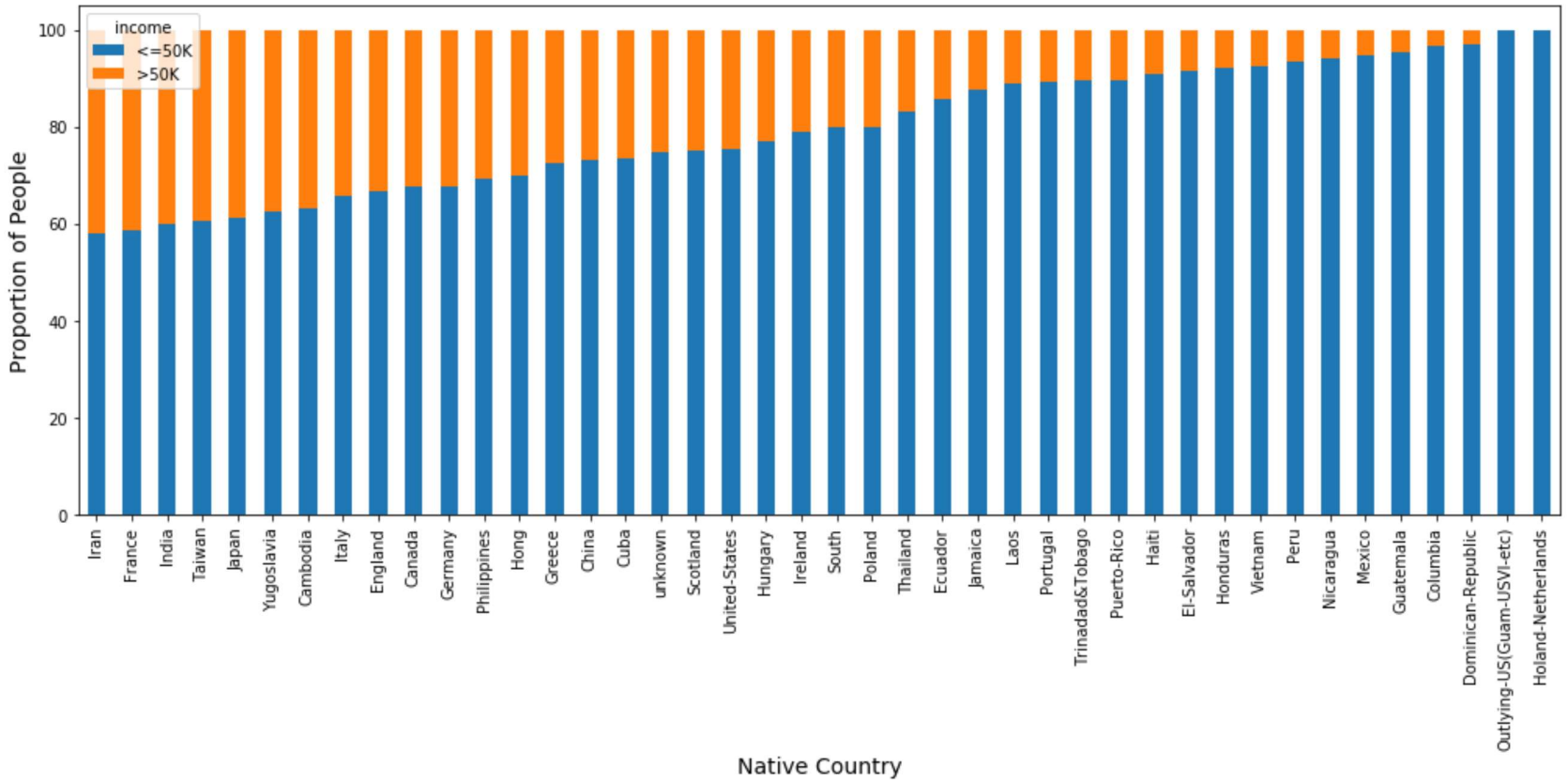


In [33]:
```
fig = plt.figure(figsize = (17,6))
ax = fig.add_subplot(1,2,1)
(table_race.div(table_race.sum(axis = 1),axis = 0)*100).sort_values(by = '<=50K').plot(kind = 'bar',ax =ax);
plt.xlabel('Race',fontsize = 14);
plt.ylabel('Proportion of People',fontsize = 14);

ax = fig.add_subplot(1,2,2)
(table_sex.div(table_sex.sum(axis = 1),axis = 0)*100).sort_values(by = '<=50K').plot(kind = 'bar',ax =ax);
plt.xlabel('Sex',fontsize = 14);
plt.ylabel('Proportion of People',fontsize = 14);
```





In [34]:
```
table_country = pd.crosstab(adult_df['native.country'], adult_df['income'])
(table_country.div(table_country.sum(axis = 1),axis = 0)*100).sort_values(by = '<=50K').plot(kind = 'bar',sta
cked = True,figsize = (17,6));
plt.xlabel('Native Country',fontsize = 14);
plt.ylabel('Proportion of People',fontsize = 14);
```

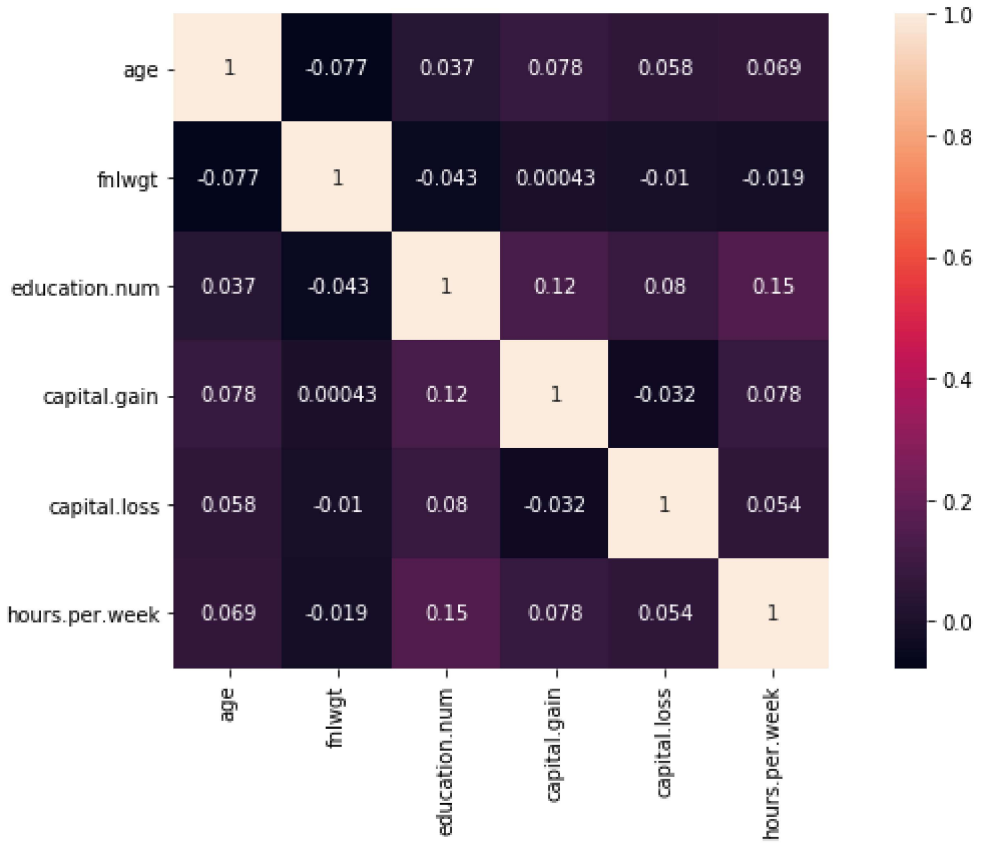


Summary

1. Sex:- Out of total male 30% of them earn salary more than 50K while less than 15% female earn more than 50K. 89% female earn less than 50K
2. Race:- White and asain-pac-Islander earn salary more than 50K
3. marital_status :- 41% of married people seem to earn salary greater than 50K.
4. People having degree doctorate,prof-school,masters are making salary more than 50K.
5. Out of all the workclass only 59% self employed people are making salary more than 50K.
6. If I check by occupation, Proportion of people making salary less than 50K is higher.

Let's find correlation between numerical fetatures

In [35]:
```
fig = plt.figure(figsize = (12,6))

sns.heatmap(adult_df[list(num_col.index)].corr(),annot = True,square = True);
```



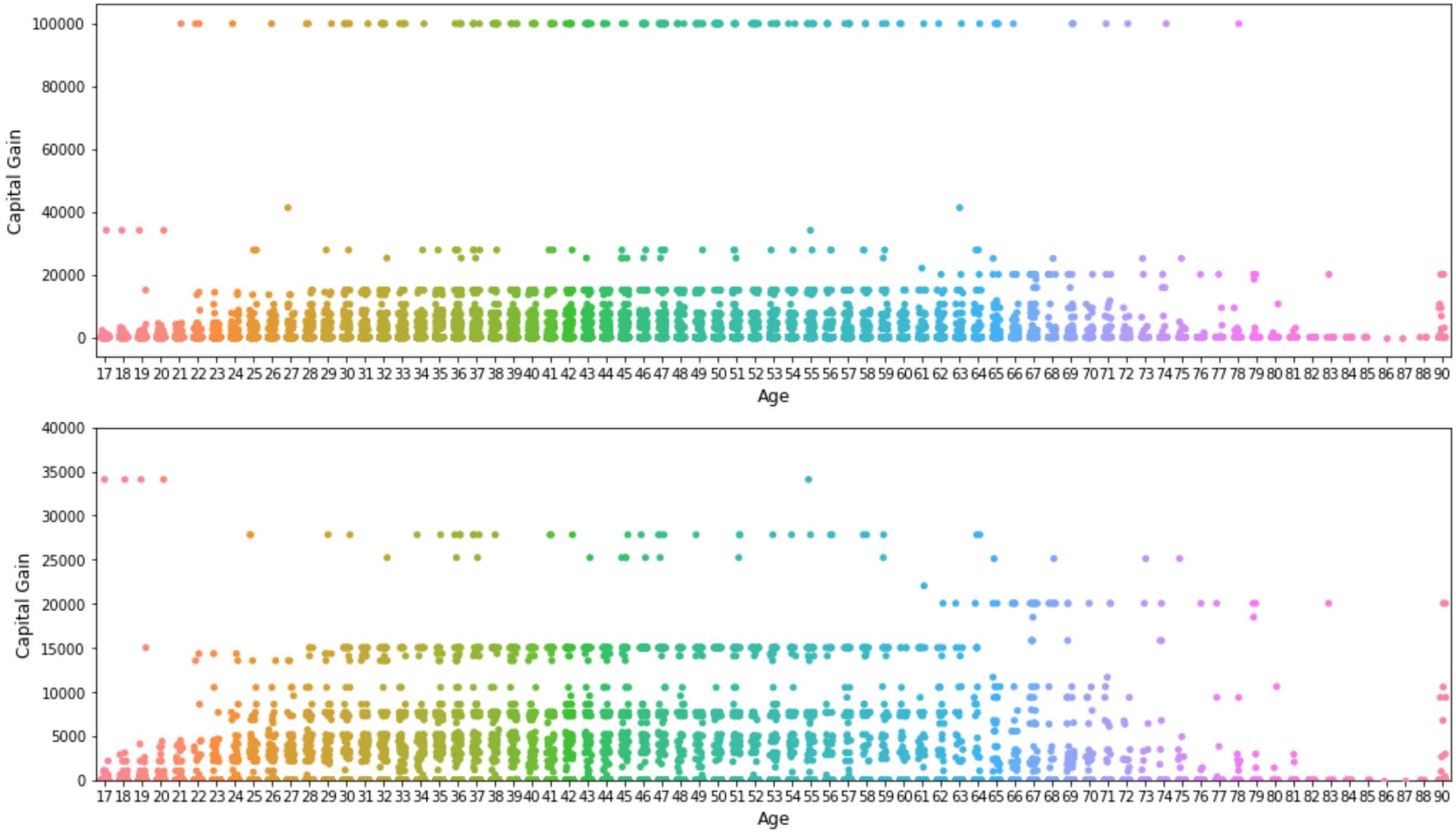
strong correlation.

- Age and Hours per week
- capital gain and hours per week

Moderate Correlation

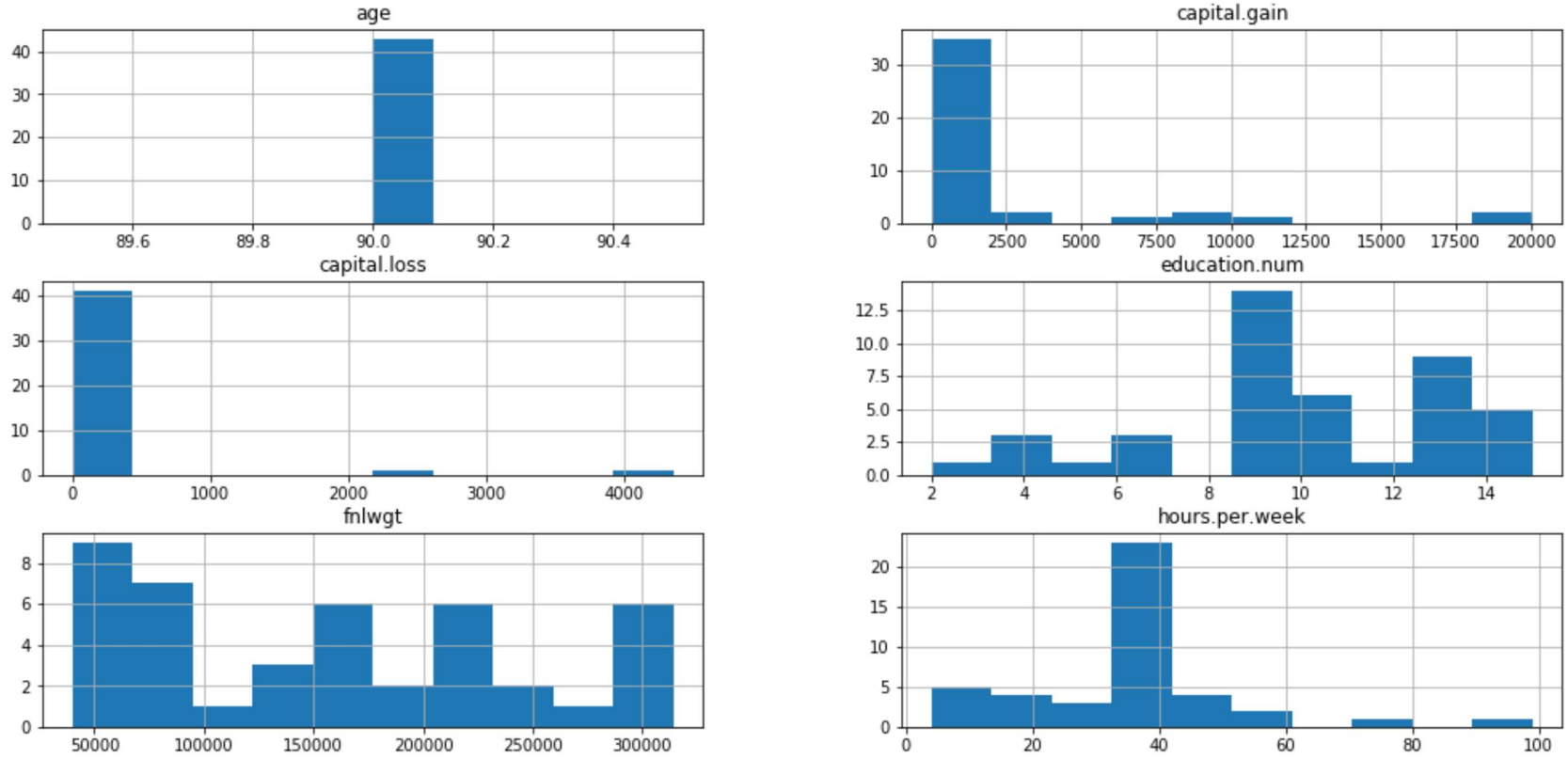
- capital loss and age

In [36]:
```
fig = plt.figure(figsize = (17,10))
ax = fig.add_subplot(2,1,1)
sns.stripplot('age', 'capital.gain', data = adult_df,
         jitter = 0.2,ax = ax);
plt.xlabel('Age',fontsize = 12);
plt.ylabel('Capital Gain',fontsize = 12);

ax = fig.add_subplot(2,1,2)
sns.stripplot('age', 'capital.gain', data = adult_df,
         jitter = 0.2);
plt.xlabel('Age',fontsize = 12);
plt.ylabel('Capital Gain',fontsize = 12);
plt.ylim(0,40000);
```



- Between age 28 and 64 capital gain is upto 15000 and after that it decreases and again increments at age 90
- Age 90 doesn't follow the pattern.
- Capital.gain of 99999 is clearly a outlier let's remove it.

In [37]:
```
adult_df[adult_df['age'] == 90].hist(figsize = (17,8));
```

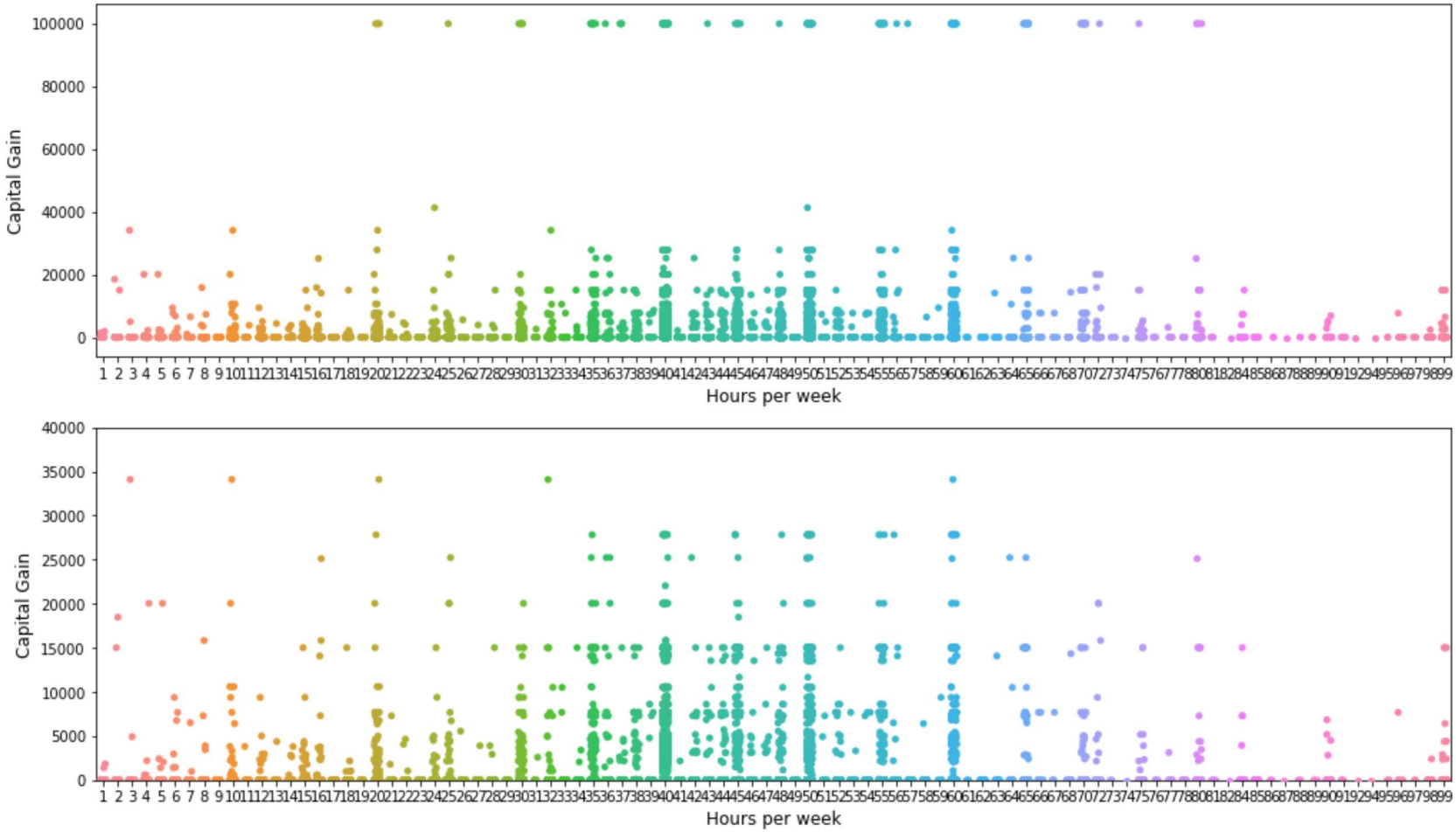


In [40]:
```
cols = ['workclass','occupation', 'income']
for col in cat_col.index:
    if col in cols:
        print("======================================{0}=======================".format(col))
        print(adult_df[adult_df['age'] == 90][col].value_counts())
    else:
        continue
```

```
======================================workclass=======================
Private           28
unknown            7
Govt_employees     5
Self_employed      3
Name: workclass, dtype: int64
======================================occupation=======================
Exec-managerial      8
unknown              7
Other-service        6
Prof-specialty       5
Adm-clerical         4
Sales                3
Machine-op-inspct    3
Craft-repair         3
Farming-fishing      1
Handlers-cleaners    1
Transport-moving     1
Protective-serv      1
Name: occupation, dtype: int64
======================================income=======================
<=50K    35
>50K      8
Name: income, dtype: int64
```

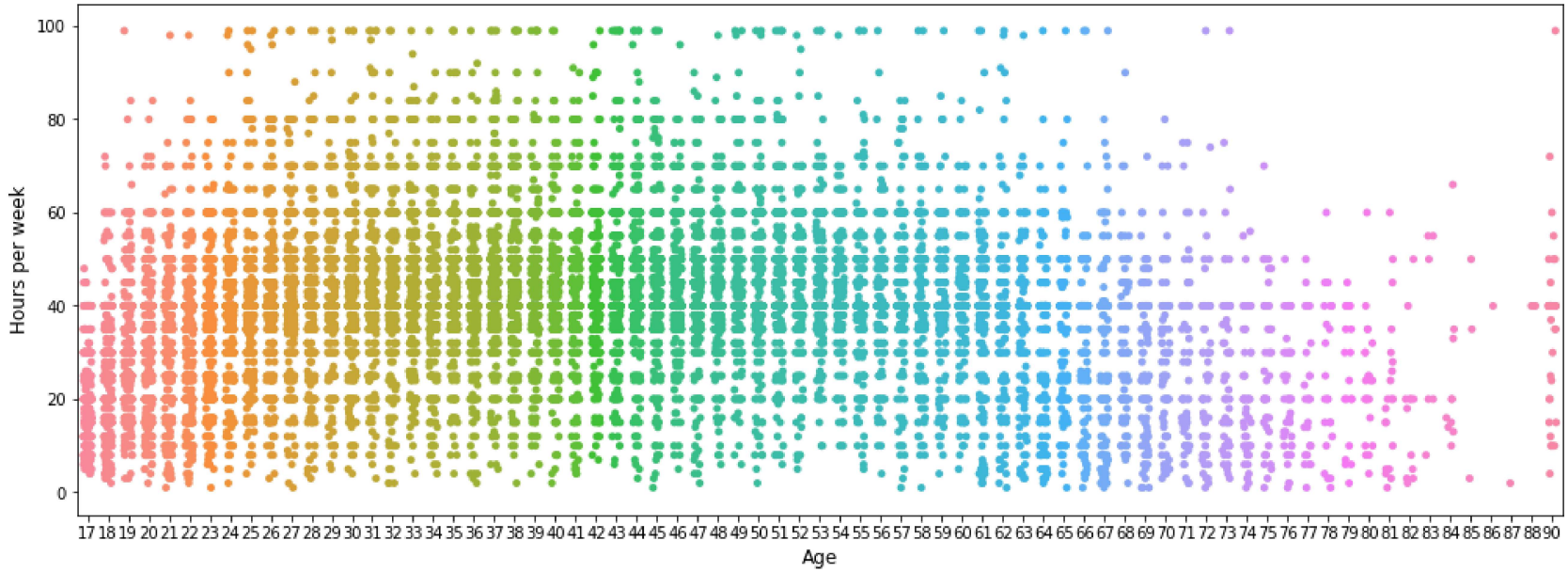
- At age 90 people can't work in goverment or private sectors
- Morever it shows the peak working hours as 40.

In [41]:
```
fig = plt.figure(figsize = (17,10))
ax = fig.add_subplot(2,1,1)
sns.stripplot('hours.per.week', 'capital.gain', data = adult_df,
         jitter = 0.2,ax = ax);
plt.xlabel('Hours per week',fontsize = 12);
plt.ylabel('Capital Gain',fontsize = 12);

ax = fig.add_subplot(2,1,2)
sns.stripplot('hours.per.week', 'capital.gain', data = adult_df,
         jitter = 0.2,ax = ax);
plt.xlabel('Hours per week',fontsize = 12);
plt.ylabel('Capital Gain',fontsize = 12);
plt.ylim(0,40000);
```



- Majority of people can be seen working for 40,50 and 60 hours per week and capital gain seems to be increasing.
- There are few people working for 99 hours per week but doesn't seem to make high capital gain. Conversely people working below 40 hours per week are making high capital gains.

In [42]:
```
fig = plt.figure(figsize = (17,6))

sns.stripplot('age','hours.per.week', data = adult_df,
         jitter = 0.2);
plt.xlabel('Age',fontsize = 12);
plt.ylabel('Hours per week',fontsize = 12);
```



Outliers Summary

- Capital gain of 99999 doesn't follow any pattern and from graph above it clearly distinguishes to be an outlier.
- Our dataset has people with age 90 and working for 40 hours per week in goverment or private sectors which is again unreasonable.
- Few people working for 99 hours per week.

Feature Removal

- Education num and education are giving similar information
- Relationship and marital status imply similar information.Hence keeping only one of the two.

Preparing Data for ML Algorithms

In [44]:
```
print("Number of columns before deleting: {0}".format(adult_df.shape[1]))

del_cols = ['relationship','education.num']
adult_df.drop(labels = del_cols,axis = 1,inplace = True)
print("Number of columns after deleting: {0}".format(adult_df.shape[1]))
```

```
Number of columns before deleting: 13

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-44-5e0a2a788791> in <module>()
      2
      3 del_cols = ['relationship','education.num']
----> 4 adult_df.drop(labels = del_cols,axis = 1,inplace = True)
      5 print("Number of columns after deleting: {0}".format(adult_df.shape[1]))

~\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inplac
e, errors)
   3695                                           index=index, columns=columns,
   3696                                           level=level, inplace=inplace,
-> 3697                                           errors=errors)
   3698
   3699     @rewrite_axis_style_signature('mapper', [('copy', True),

~\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inpla
ce, errors)
   3109         for axis, labels in axes.items():
   3110             if labels is not None:
-> 3111                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3112
   3113         if inplace:

~\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors)
   3141                 new_axis = axis.drop(labels, level=level, errors=errors)
   3142             else:
-> 3143                 new_axis = axis.drop(labels, errors=errors)
   3144             result = self.reindex(**{axis_name: new_axis})
   3145

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   4402             if errors != 'ignore':
   4403                 raise KeyError(
-> 4404                     '{} not found in axis'.format(labels[mask]))
   4405             indexer = indexer[~mask]
   4406         return self.delete(indexer)

KeyError: "['relationship' 'education.num'] not found in axis"
```

In [45]:
```
hrs_per_week = adult_df[adult_df['hours.per.week'] == 99]
print("Number of people working for 99 hours per week:", hrs_per_week.shape[0])
```

```
Number of people working for 99 hours per week: 85
```

- It won't be a good idea to delete all the outliers because if that's the case with test data too we might get false results. So I am keeping people working for 99 hourse per week.

In [46]:
```
# drop rows with age 90
print("Number of observation before removing:",adult_df.shape)
index_age = adult_df[adult_df['age'] == 90].index
adult_df.drop(labels = index_age,axis = 0,inplace =True)
print("Number of observation after removing:",adult_df.shape)
```

```
Number of observation before removing: (32561, 13)
Number of observation after removing: (32518, 13)
```

In [47]:
```
print("Number of observation before removing:",adult_df.shape)
index_gain = adult_df[adult_df['capital.gain'] == 99999].index
adult_df.drop(labels = index_gain,axis = 0,inplace =True)
print("Number of observation after removing:",adult_df.shape)
```

```
Number of observation before removing: (32518, 13)
Number of observation after removing: (32359, 13)
```

In [48]:
```
num_col_new = ['age','capital.gain', 'capital.loss',
       'hours.per.week','fnlwgt']
cat_col_new = ['workclass', 'education', 'marital.status', 'occupation',
               'race', 'sex', 'native.country', 'income']
```

In [49]:
```
from sklearn.pipeline import Pipeline
from sklearn.base import TransformerMixin
from sklearn.preprocessing import MinMaxScaler,StandardScaler
scaler = MinMaxScaler()
pd.DataFrame(scaler.fit_transform(adult_df[num_col_new]),columns = num_col_new).head(3)
```

Out[49]:

	age	capital.gain	capital.loss	hours.per.week	fnlwgt
0	0.915493	0.0	1.000000	0.173469	0.081896
1	0.690141	0.0	1.000000	0.397959	0.118021
2	0.521127	0.0	0.895317	0.397959	0.086982

In [50]:
```
class DataFrameSelector(TransformerMixin):
    def __init__(self,attribute_names):
        self.attribute_names = attribute_names

    def fit(self,X,y = None):
        return self

    def transform(self,X):
        return X[self.attribute_names]


class num_trans(TransformerMixin):
    def __init__(self):
        pass

    def fit(self,X,y=None):
        return self

    def transform(self,X):
        df = pd.DataFrame(X)
        df.columns = num_col_new
        return df



pipeline = Pipeline([('selector',DataFrameSelector(num_col_new)),
                     ('scaler',MinMaxScaler()),
                    ('transform',num_trans())])
```

In [51]:
```
num_df = pipeline.fit_transform(adult_df)
num_df.shape
```

Out[51]: (32359, 5)

In [52]:
```
# columns which I don't need after creating dummy variables dataframe
cols = ['workclass_Govt_employess','education_Some-college',
        'marital.status_Never-married','occupation_Other-service',
        'race_Black','sex_Male','income_>50K']
```

In [53]:
```
class dummies(TransformerMixin):
    def __init__(self,cols):
        self.cols = cols

    def fit(self,X,y = None):
        return self

    def transform(self,X):
        df = pd.get_dummies(X)
        df_new = df[df.columns.difference(cols)]
#difference returns the original columns, with the columns passed as argument removed.
        return df_new

pipeline_cat=Pipeline([('selector',DataFrameSelector(cat_col_new)),
                       ('dummies',dummies(cols))])
cat_df = pipeline_cat.fit_transform(adult_df)
cat_df.shape
```

Out[53]: (32359, 80)

difference (https://stackoverflow.com/questions/14940743/selecting-excluding-sets-of-columns-in-pandas)

In [54]:
```
cat_df['id'] = pd.Series(range(cat_df.shape[0]))
num_df['id'] = pd.Series(range(num_df.shape[0]))
```

In [56]:
```
final_df = pd.merge(cat_df,num_df,how = 'inner', on = 'id')
print("Number of observations in final dataset: {0}".format(final_df.shape))
```

```
Number of observations in final dataset: (32157, 86)
```

Train and Fine Tuning on the model

In [57]:
```
y = final_df['income_<=50K']
final_df.drop(labels = ['id','income_<=50K'],axis = 1,inplace = True)
X = final_df
```

In [58]:
```
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,BaggingClassifier,ExtraTreesClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curve, auc
from datetime import datetime
from sklearn.feature_selection import RFE
```

```
C:\Users\Sreekanth\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: nu
mpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed in a future N
umPy release.
  from numpy.core.umath_tests import inner1d
```

In [59]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =0.25,random_state = 42)
```

In [60]:
```
#Instantiate the classifiers
clf_logreg = LogisticRegression()
clf_tree = DecisionTreeClassifier()
clf_knn =  KNeighborsClassifier()
clf_svc = SVC()
clf_forest = RandomForestClassifier()
clf_ada = AdaBoostClassifier()
clf_bagging = BaggingClassifier()
clf_extratrees = ExtraTreesClassifier()
clf_gnb = GaussianNB()
```

In [61]:
```
classifiers = ['LogisticRegression', 'DecisionTree', 'KNN', 'SVC', 'RandomForest', 'Adaboost', 'Bagging', 'Ex
tratrees', 'Naive']
```

In [62]:
```
models = {clf_logreg:'LogisticRegression',
          clf_tree:'DecisionTree',
          clf_knn: 'KNN',
          clf_svc: 'SVC',
          clf_forest: 'RandomForest',
          clf_ada: 'Adaboost',
          clf_bagging: 'Bagging',
          clf_extratrees:'Extratrees' ,
          clf_gnb: 'Naive'}
```

In [63]:
```
# train function fits the model and returns accuracy score
def train(algo,name,X_train,y_train,X_test,y_test):
    algo.fit(X_train,y_train)
    y_pred = algo.predict(X_test)
    score = accuracy_score(y_test,y_pred)
    print("-----------------------------------------{0}-----------------------------------------------
".format(name))
    print("Accuracy Score for {0}: {1:.4f}%".format(name,score*100))
    return y_test, y_pred, score

# acc_res function calculates confusion matrix
def acc_res(y_test,y_pred):
    null_accuracy = y_test.value_counts()[0]/len(y_test)
    print("Null Accuracy: {0:.4f}%".format(null_accuracy*100))
    print("Confusion Matrix")
    matrix = confusion_matrix(y_test,y_pred)
    print(matrix)
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    TN = matrix[0,0]
    FP = matrix[0,1]
    FN = matrix[1,0]
    TP = matrix[1,1]
    accuracy_score=(TN+TP) / float(TP+TN+FP+FN)
    recall_score = (TP)/ float(TP+FN)
    specificity = TN / float(TN+FP)
    FPR = FP / float(FP+TN)
    precision_score = TP / float(TP+FP)
    print("Accuracy Score: {0:.4f}%".format(accuracy_score*100))
    print("Recall Score: {0:.4f}%".format(recall_score*100))
    print("Specificity Score: {0:.4f}%".format(specificity*100))
    print("False Positive Rate: {0:.4f}%".format(FPR*100))
    print("Precision Score: {0:.4f}%".format(precision_score*100))
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("Classification Report")
    print(classification_report(y_test,y_pred))

def main(models):
    accuracy_scores = []
    for algo,name in models.items():
        y_test_train,y_pred,acc_score = train(algo,name,X_train,y_train,X_test,y_test)
        acc_res(y_test_train,y_pred)
        accuracy_scores.append(acc_score)
```

```
    return accuracy_scores

accuracy_scores = main(models)
```

```
-------------------------------------------Naive-------------------------------------------------
Accuracy Score for Naive: 30.5100%
Null Accuracy: 23.4950%
Confusion Matrix
[[1850   39]
 [5548  603]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 30.5100%
Recall Score: 9.8033%
Specificity Score: 97.9354%
False Positive Rate: 2.0646%
Precision Score: 93.9252%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.25      0.98      0.40      1889
          1       0.94      0.10      0.18      6151

avg / total       0.78      0.31      0.23      8040

-------------------------------------------Extratrees--------------------------------------------
Accuracy Score for Extratrees: 81.3557%
Null Accuracy: 23.4950%
Confusion Matrix
[[1127  762]
 [ 737 5414]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 81.3557%
Recall Score: 88.0182%
Specificity Score: 59.6612%
False Positive Rate: 40.3388%
Precision Score: 87.6619%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.60      0.60      0.60      1889
          1       0.88      0.88      0.88      6151

avg / total       0.81      0.81      0.81      8040

-------------------------------------------Adaboost----------------------------------------------
```

```
Accuracy Score for Adaboost: 85.2363%
Null Accuracy: 23.4950%
Confusion Matrix
[[1070  819]
 [ 368 5783]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 85.2363%
Recall Score: 94.0172%
Specificity Score: 56.6437%
False Positive Rate: 43.3563%
Precision Score: 87.5947%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.74      0.57      0.64      1889
          1       0.88      0.94      0.91      6151

avg / total       0.84      0.85      0.84      8040

-------------------------------------------Bagging--------------------------------------------------
Accuracy Score for Bagging: 82.5746%
Null Accuracy: 23.4950%
Confusion Matrix
[[1150  739]
 [ 662 5489]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 82.5746%
Recall Score: 89.2375%
Specificity Score: 60.8788%
False Positive Rate: 39.1212%
Precision Score: 88.1342%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.63      0.61      0.62      1889
          1       0.88      0.89      0.89      6151

avg / total       0.82      0.83      0.82      8040

-----------------------------------------KNN--------------------------------------------------
Accuracy Score for KNN: 81.0821%
```

```
Null Accuracy: 23.4950%
Confusion Matrix
[[ 939  950]
 [ 571 5580]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 81.0821%
Recall Score: 90.7170%
Specificity Score: 49.7088%
False Positive Rate: 50.2912%
Precision Score: 85.4518%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.62      0.50      0.55      1889
          1       0.85      0.91      0.88      6151

avg / total       0.80      0.81      0.80      8040

-----------------------------------------RandomForest-----------------------------------------------------
Accuracy Score for RandomForest: 82.5124%
Null Accuracy: 23.4950%
Confusion Matrix
[[1130  759]
 [ 647 5504]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 82.5124%
Recall Score: 89.4814%
Specificity Score: 59.8200%
False Positive Rate: 40.1800%
Precision Score: 87.8812%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.64      0.60      0.62      1889
          1       0.88      0.89      0.89      6151

avg / total       0.82      0.83      0.82      8040

-----------------------------------------SVC-------------------------------------------------------
Accuracy Score for SVC: 83.4328%
Null Accuracy: 23.4950%
```

```
Confusion Matrix
[[ 912  977]
 [ 355 5796]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 83.4328%
Recall Score: 94.2286%
Specificity Score: 48.2795%
False Positive Rate: 51.7205%
Precision Score: 85.5751%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.72      0.48      0.58      1889
          1       0.86      0.94      0.90      6151

avg / total       0.82      0.83      0.82      8040

-------------------------------------------LogisticRegression-------------------------------------------------
----
Accuracy Score for LogisticRegression: 84.2040%
Null Accuracy: 23.4950%
Confusion Matrix
[[1042  847]
 [ 423 5728]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 84.2040%
Recall Score: 93.1231%
Specificity Score: 55.1615%
False Positive Rate: 44.8385%
Precision Score: 87.1179%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.71      0.55      0.62      1889
          1       0.87      0.93      0.90      6151

avg / total       0.83      0.84      0.83      8040

-------------------------------------------DecisionTree-------------------------------------------------
Accuracy Score for DecisionTree: 79.8259%
Null Accuracy: 23.4950%
```

```
Confusion Matrix
[[1089  800]
 [ 822 5329]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Accuracy Score: 79.8259%
Recall Score: 86.6363%
Specificity Score: 57.6496%
False Positive Rate: 42.3504%
Precision Score: 86.9473%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Classification Report
             precision    recall  f1-score   support

          0       0.57      0.58      0.57      1889
          1       0.87      0.87      0.87      6151

avg / total       0.80      0.80      0.80      8040
```

In [64]:
```
pd.DataFrame(accuracy_scores,columns = ['Accuracy Scores'],index = classifiers).sort_values(by = 'Accuracy Scores',ascending = False)
```

Out[64]:

	Accuracy Scores
KNN	0.852363
Extratrees	0.842040
Bagging	0.834328
SVC	0.825746
Adaboost	0.825124
DecisionTree	0.813557
RandomForest	0.810821
Naive	0.798259
LogisticRegression	0.305100

In [65]:
```
models_new = {clf_logreg:'LogisticRegression',
              clf_forest: 'RandomForest',
              clf_ada: 'Adaboost',
              clf_bagging: 'Bagging'}

classifiers_new = models_new.values()
```

K-fold cross validation divides data into k folds and uses k-1 folds for training and k folds for testing. Eventually all the data gets used for training and testing.

In [68]:
```
mean_score = []
for model,value in models_new.items():
    start_time = datetime.now()
    scores = cross_val_score(model,X_train,y_train,scoring = 'accuracy',cv = 10)
    elapsed_time = datetime.now() - start_time
    print("Time taken to complete training {0} model: (hh:mm:ss.ms) {1}".format(value,elapsed_time ))
    mean_score.append(scores.mean())
```

```
Time taken to complete training Adaboost model: (hh:mm:ss.ms) 0:00:20.167431
Time taken to complete training LogisticRegression model: (hh:mm:ss.ms) 0:00:02.883438
Time taken to complete training RandomForest model: (hh:mm:ss.ms) 0:00:06.080253
Time taken to complete training Bagging model: (hh:mm:ss.ms) 0:00:32.123589
```

In [69]:
```
pd.DataFrame(mean_score,index = classifiers_new,
             columns = ['Accuracy Score']).sort_values(by = 'Accuracy Score', ascending = False)
```

Out[69]:

	Accuracy Score
Adaboost	0.853505
LogisticRegression	0.845958
RandomForest	0.823029
Bagging	0.820623

In [70]:
```
param_grid_logreg = {'C':[0.0001,0.001,0.01,0.1,1,10,100], 'penalty':['l1','l2']}
param_grid_rf = {'n_estimators' : [50,60],
                 'max_depth': range(5,16,2)}
param_grid_ada = {'n_estimators':[50,60,70]}
param_grid_bagging = {'n_estimators':[50,60,70]}
```

In [71]:
```
def feature_selection(model,X_train,y_train):
    rfe_model = RFE(model)
    rfe_model = rfe_model.fit(X_train,y_train)
    X_train_cols = list(X_train.columns[rfe_model.support_])
    X_train_new = X_train[X_train_cols]
    return X_train_new

def grid(model,parameters,X_train_new):
    grid = GridSearchCV(estimator = model, param_grid = parameters, cv = 10, return_train_score = False,
                        scoring = 'accuracy')
    grid.fit(X_train_new,y_train)
    return grid.best_score_, grid.best_estimator_

def main(model,X_train,y_train,parameters):
    X_train_modi = feature_selection(model,X_train,y_train)
    grid_best_score,grid_best_params = grid(model,parameters,X_train_modi)
    return grid_best_score,grid_best_params,X_train_modi
```

In [72]:
```
grid_best_score_logreg, grid_best_params_logreg , X_train_modi_logreg = main(clf_logreg,X_train,y_train,param
_grid_logreg)
grid_best_score_rf, grid_best_params_rf , X_train_modi_rf = main(clf_forest,X_train,y_train,param_grid_rf)
grid_best_score_ab, grid_best_params_ab , X_train_modi_ab = main(clf_ada,X_train,y_train,param_grid_ada)
grid_best_score_bc, grid_best_params_bc , X_train_modi_bc = main(clf_ada,X_train,y_train,param_grid_bagging)
```

In [73]:
```
pd.DataFrame([grid_best_score_logreg, grid_best_score_rf,grid_best_score_ab,grid_best_score_bc],
             index = ['LogisticRegression','RandomForest','AdaBoost','Bagging'],
             columns = ['Accuracy Score']).sort_values(by = 'Accuracy Score',ascending  = False)
```

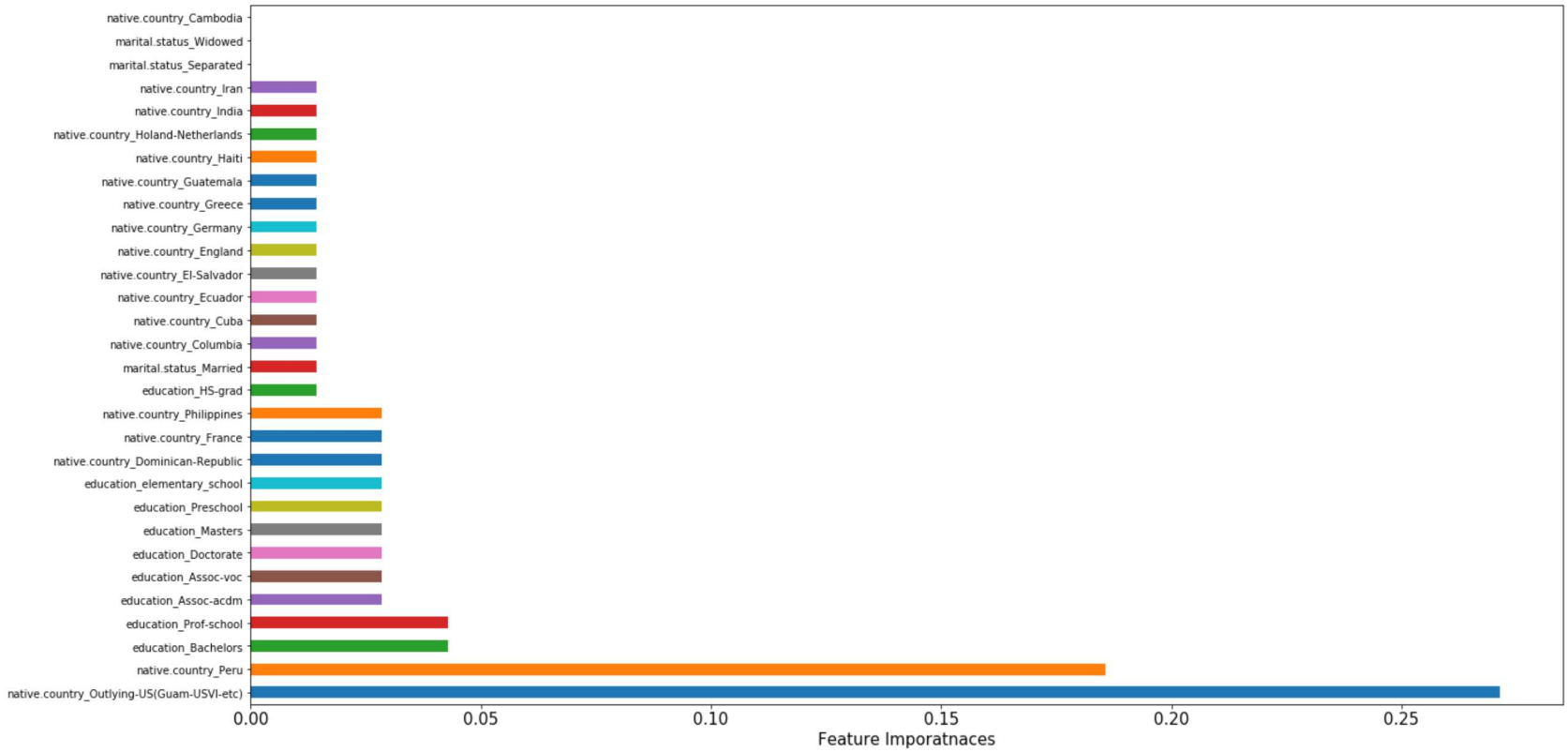
Out[73]:

	Accuracy Score
AdaBoost	0.854460
Bagging	0.854460
RandomForest	0.853216
LogisticRegression	0.845006

In [74]:
```
print("Number of estimators:",grid_best_params_ab.get_params()['n_estimators'])
```

```
Number of estimators: 70
```

In [78]:
```
ab_mod  = grid_best_params_ab
plt.figure(figsize = (22,12))
(pd.Series(ab_mod.feature_importances_, index=X_train.columns[0:42]).nlargest(30).plot(kind='barh'));
plt.xlabel('Feature Imporatnaces',fontsize = 15);
plt.xticks(fontsize = 15);
```



- Machine Learning Coursera (https://www.coursera.org/learn/machine-learning)
- Logistic Regression Tuning Parameters (http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Machine%20Learning%20Lecture%202.htm)
- Why Scaling (https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e)
- passing variables between function (https://stackoverflow.com/questions/16043797/python-passing-variables-between-functions)
- Feature Importances (https://stackoverflow.com/questions/44511636/matplotlib-plot-feature-importance-with-feature-names)

In [ ]:
```
```