What are the most important variables that determine a person’s income?

Fabrizio Stella

Abstract

The purpose of this work is to understand **what are the individual characteristics of people who better than others explain the differences in wages among people**.

The **dataset used** in this study **come from** the “**UCI Machine Learning Repository**” and is based on the 1994 census. It’s composed by variables like age, education, occupation, sex, race, birth country, salary and others.

To study this relation a **decision tree classification model** has been created and performed by using **Scikit-learn** library.

The results seem to suggest that a **relation between individual characteristics of people and salary exists**, and the model built seems to predict correctly more than 80% of the data entered. The variables with the most informative capacity are: **Relationship, Education, Occupation and Age**. Other variables like race, sex, native country are not significant.

Motivation

What is driving my interest in the past months are the variables – personal or not – that can explain the future income of people. Many of them, of course, are not under our control: age or sex could explain income differences across people, for example, but aren’t under our control. Others, instead, at least partially, are under our control: Education for example.

Being able to determine the contribution of these variables to the future income of people is very important as we can better drive the behavior of the youngest to the right direction, telling them how much important is, if it is, education for their future or, at worst, we can, in any case, extract relevant policy implication.

Dataset(s)

The dataset used in this data science project is called “Adult” and come from the well-known data science repository “UCI Machine Learning Repository”. It was created in 1996 in the USA and it’s based on the 1994 Census Database.

It’s a collection of individual characteristics of people, including age, education, occupation, race, salary. The number of instances at the beginning of the analysis was 32560. The number of categorical variables is 15: among them, salary is our Target Variable.

The underlying idea behind the dataset is that a relation between Salary and all the other variables can be established.

Data Preparation and Cleaning

As said before, the dataset I used for my analysis is “Adult”, from the “UCI Machine Learning Repository. At the beginning of the analysis, the dataset was composed by 15 attributes (including the target one) and 32560 instances.

In the figure 1 is shown the list of the attributes in the initial dataset.

Figure 1

Variable Name	Kind	Description
age	continuous	17<age<90
workclass	categorical	8 different unique instances
fnlwgt	continuous	
education	categorical	16 different unique instances
education-num	continuous	16 different numbers
marital-status	categorical	7 different unique instances
occupation	categorical	14 different unique instances
relationship	categorical	6 different unique instances
race	categorical	5 different unique instances
sex	categorical	2 different unique instances
capital-gain	continuous	
capital-loss	continuous	
hours-per-week	continuous	1<hours per week<99
native-country	categorical	40 different unique instances
Salary	**categorical**	**2 different unique instances > 50K$ < 50K$**

Target Variable

Main actions taken to clean the dataset:

1. Drop not relevant columns: **the columns “fnlwgt”, education-num, capital-gain and capital-loss have been removed** from the dataset. The variable “fnlwgt” is considered not relevant for the analysis purposes. In capital-gain and capital-loss, instead, too many missing values made impossible their use for further analysis.

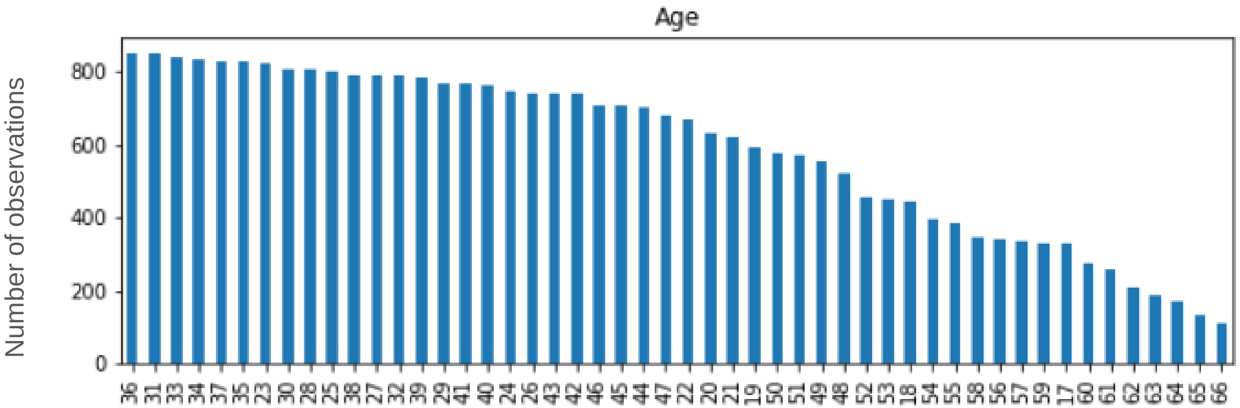
2. All rows in which were at least 1 **NaN values have been removed** from the dataset. This implies a reduction of 8% of the number of instances (from 32560 to 30117).

3. Attribute manipulation: for not removed attributes a further cleaning activity has been performed**: continuous variable have been transformed into categorical ones** and the **number of distinct observations of categorical variables has been reduced**. In the next slides some example of this activity will be provided.

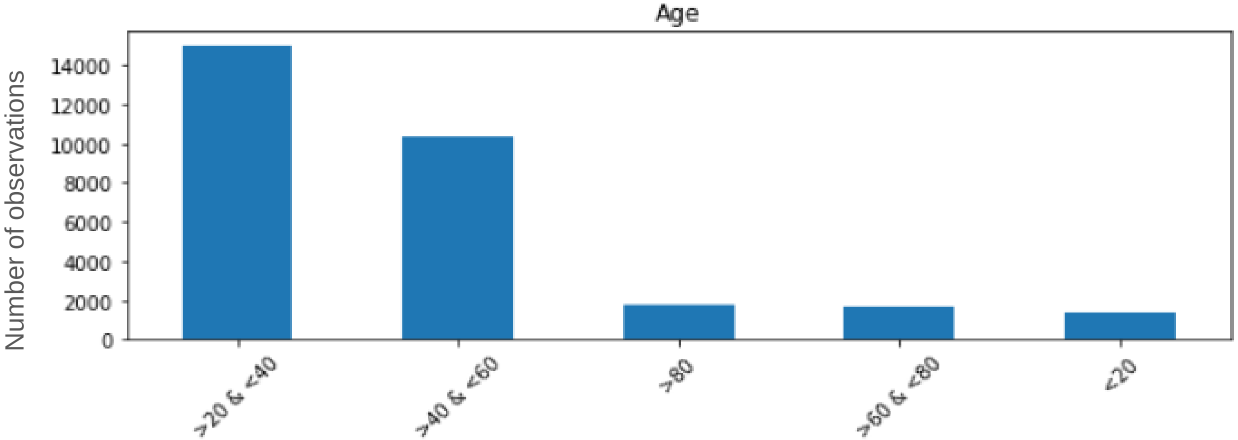
Data Preparation and Cleaning. Focus on attributes manipulation:

Reduction of the distinct values assumed by the ‘Age’ Attribute.

Before Changes



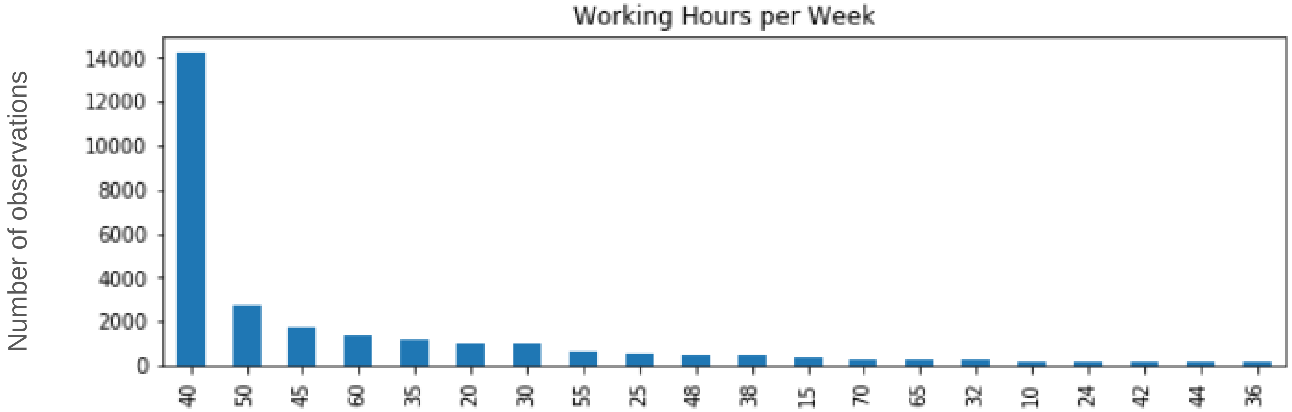
After Changes



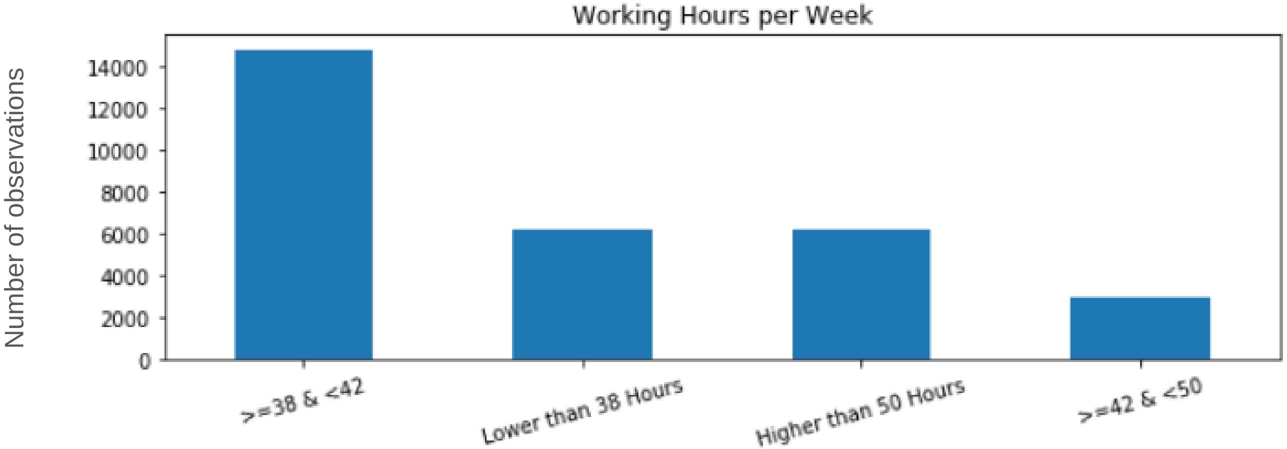
Data Preparation and Cleaning: Focus on attributes manipulation:

Reduction of the distinct values assumed by the ‘Woking Hours per Week’ Attribute.

Before Changes



After Changes



Research Question(s)

In this project I’m trying to understand what are the main individual characteristics associated to high salaries. Variables like age, education, race, sex, could somehow affect a person's salary. I would like to understand if this is true and, if it is, which of these variables are the most important.

Through a classification task I’ll verify whether the behavior of some of the variables mentioned above could be associated with the behavior of the remuneration that a person receive from the market.

In other, simpler word, what are the most important individual characteristics that explain the differences between individuals in the probability of getting a better wage? Is it true that educated people receive higher salaries? And, is it true that women, on average, receive lower salaries than men?

Methods

For this analysis I used **Scikit learn** to perform a **decision tree data classification**, as I know ex-ante the values of the target variable and many attributes are categorical (including the Target one).

To perform this decision tree the following actions have been taken:

1. I divided the **dataset into two:**

   - **Training set DB** — **67% in the instances (# 20178)**
   - **Test set DB** — **33% in the instances (# 9939)**

2. I applied the decision tree data classification algorithm to the **Training set** with the following parameters:

   ✓ **No maximum depth** has been imposed

   ✓ **Maximum leaf node** == 20

Findings: Results from the Training Phase (1)

After having applied the previously traced model during the training phase, we can determine how much good our attribute are and, in particular, wich are the most informative among them, i.e. the attributes that better than other help us in understand the behavior of our target variable: salary.

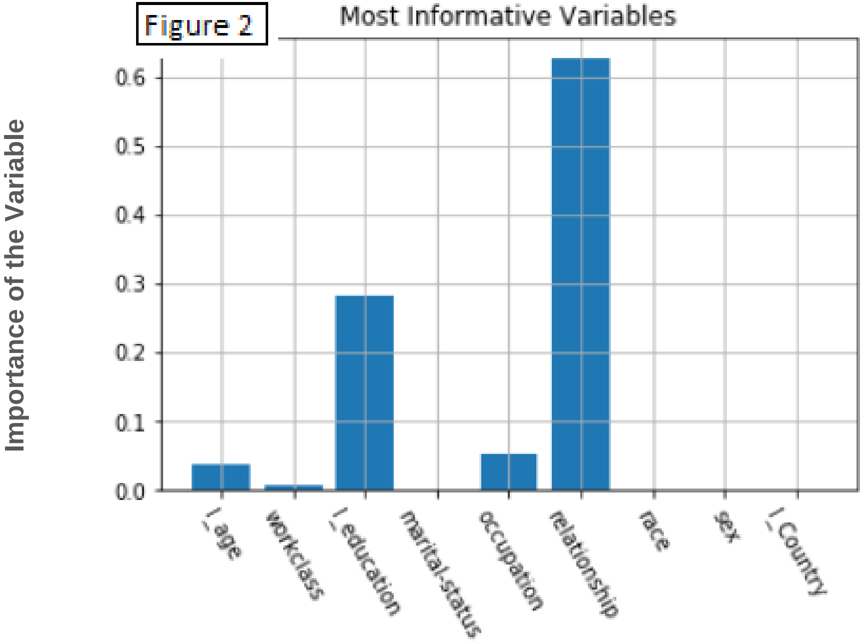


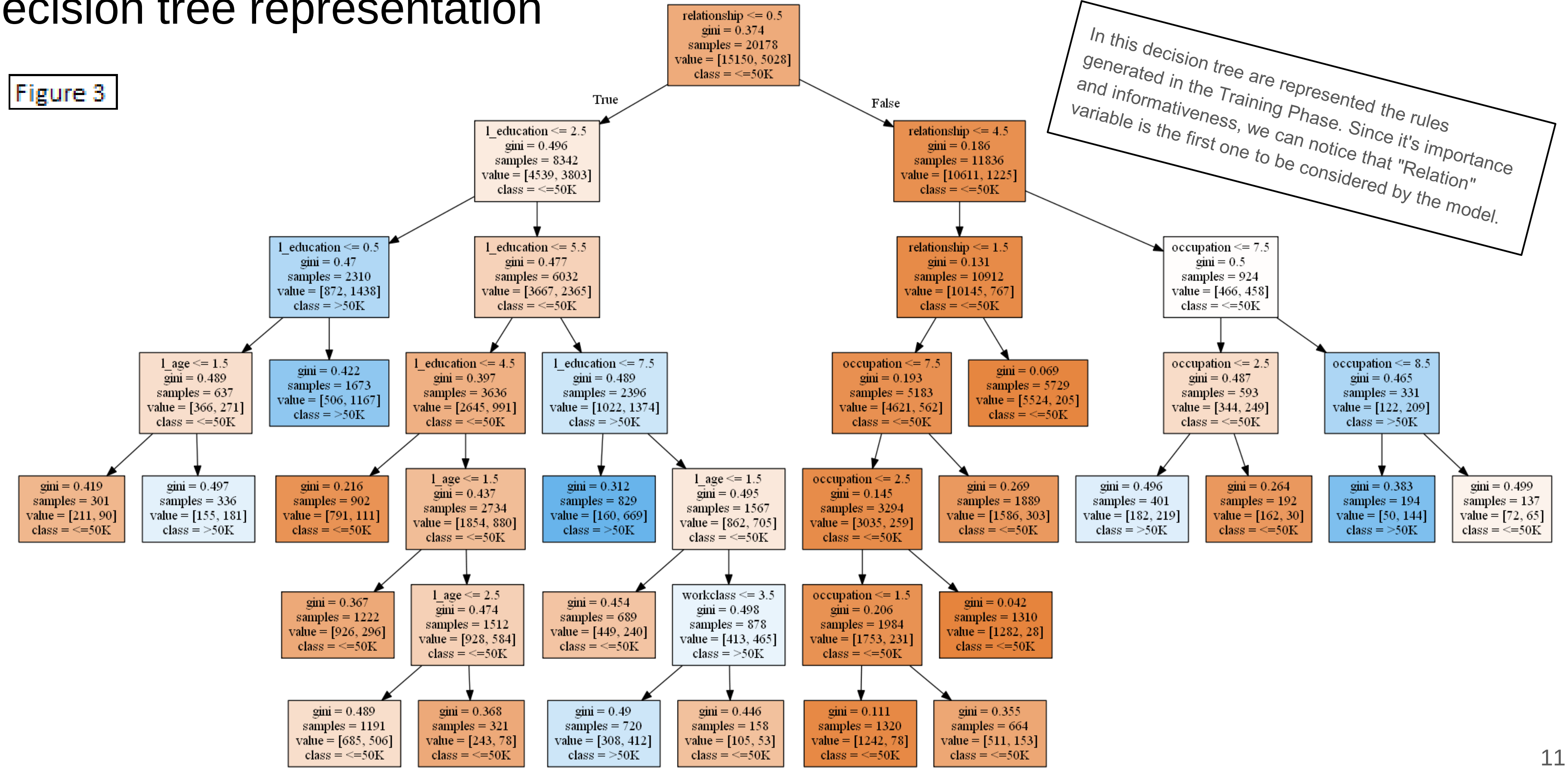
Figure 2

- **In the Figure 2** are displayed the **variables** that compose our **dataset**. The information we can get is the level **of importance** of each of them in terms of **informativeness**.
- ✓ Surprisingly, “**relationship**” is the most informative variable. It includes observations like “Unmarried”, “Husband”, ”Wife”, “Own-Child”, “not in Family”, “Other Relative”.
- ✓ The other most informative variables are **Education, Occupation, age, workclass**.
- ✓ No importance is associated to the other variables: Marital Status, Race, Sex, Country. They seems no to be able to explain income variation across people.

Findings: Results from the Training Phase (2):

Decision tree representation

Figure 3



In this decision tree are represented the rules generated in the Training Phase. Since it's importance and informativeness, we can notice that "Relation" variable is the first one to be considered by the model.

Findings: Results from the Test Phase

- ✓ The last point remained is: how good the built model is. Is it able to predict the target variable?
- ✓ To answer this question we can use the test dataset previously generated and try to understand how many mistakes we get by the application of the training phase model.
- ✓ One measure we can use the accuracy of our model is the '**Accuracy Score**':
  - It is equal to 1 if the model is perfect;
  - It is equal to 0 if the model is totally mistaken and no variable is classified correctly;
- ✓ An accuracy score of .75 could be considered good.

Figure 4 **Measure of accuracy**

```
In [775]: accuracy_score(y_true = y_test,
                         y_pred = predictions)

Out[775]: 0.820303853506389
```

In the Figure 4 is displayed the accuracy score: it is .82. It's very good: It's means that more than 82% of the instances that make up the Test set are classified correctly.

Limitations

This project work and its results have **some relevant limits:**

1) It's based on a census done in a town in the USA: it implies that probably it could not be applicable to other different places or Countries.
2) I'm not creating a model of causation: We can say that, based on this model, higher education is associated with a higher probability to get a salary >50K; but this is not a causation: I cannot say that higher education cause higher salaries. I think that is true, of course; but this is not the instrument that prove this statement. For example, richest families can finance better education to their children. In this case would not be education to cause good salaries but initial conditions like parent's money.
3) Since the target variable is categorical, the dataset is divided into two, based on the values of the target variable which are either >50k$ or <50k$. This, near the threshold value, generates identification problems for individuals: a person with an income of 49.5k$ should not be too much different from another with an income of 51.5k$. But the model, as it is built, divides them totally. Unfortunately, not having the exact data of income, using a classification model is the only method to derive some information.

Conclusions

- ✓ Using Scikit-learn, I created a decision tree classification model that **predict** the **relation between individual characteristics and Salary**.
- ✓ **The level of accuracy of this model is roughly 82%**: it means that, knowing ex-ante some individual characteristics, we can predict whether a person will have an income higher than 50K$ or not.
- ✓ Among the others, the most important variables that explain differences in salaries are **Relationship, Education, Occupation and Age**. Remain variables like race, sex, native country seem to be no useful in this.

Acknowledgements

Data used come from the repository called UCI Machine Learning repository, suggested by the professor Leo during the week 9 of the course of Python for Data Science.

All the analysis done in this final project come from my effort.

References

https://www.python.org/doc/

https://pandas.pydata.org/docs/

https://stackoverflow.com/

https://medium.com/

https://scikit-learn.org/

https://numpy.org/

https://www.graphviz.org/

Learning Python, 5th Edition, Mark Lutz

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
#pd.set_option('display.max_rows', None)
pd.options.mode.chained_assignment = None
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn import preprocessing
from matplotlib import pyplot
from numpy import loadtxt
from xgboost import XGBClassifier
from xgboost import plot_importance
import os
import graphviz
import pydotplus
from sklearn import tree
```

In [2]:

```
os.environ['PATH'] += os.pathsep + r'C:\Program Files\graphviz-2.38\release\bin'
```

In [3]:

```
#conda install -c anaconda py-xgboost
```

In [4]:

```
data = pd.read_csv('./AdultDataset/adult.data')
```

In [5]:

```
data.columns
```

Out[5]:

```
Index(['39', ' State-gov', ' 77516', ' Bachelors', ' 13', ' Never-married',
       ' Adm-clerical', ' Not-in-family', ' White', ' Male', ' 2174', ' 0',
       ' 40', ' United-States', ' <=50K'],
      dtype='object')
```

Description of the main variables in the dataset

1) age: continuous. 2) workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. 3) fnlwgt: continuous. 4) 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. 5) education-num: continuous. 6) marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. 7) 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. 8) relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. 9) race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. 10) sex: Female, Male. 11) capital-gain: continuous. 12) capital-loss: continuous. 13) hours-per-week: continuous. 14) 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.

Below, I insert the columns names

In [6]:

```
column_names= ['age',
```

```
column_names= [ 'age',
                'workclass',
                'fnlwgt',
                'education',
                'education-num',
                'marital-status',
                'occupation',
                'relationship',
                'race',
                'sex',
                'capital-gain',
                'capital-loss',
                'hours-per-week',
                'native-country',
                'Salary'
              ]
df1 = pd.DataFrame(data.values, columns = column_names )
```

In [7]:

```
df1.head()
```

Out[7]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States

I remove education num, as It's gives me not additional information compared to Education column

In [8]:

```
del(df1['education-num'])
```

In [9]:

```
df1.columns
```

Out[9]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country', 'Salary'],
      dtype='object')
```

In [10]:

```
df1.shape
```

Out[10]:

```
(32560, 14)
```

In [11]:

```
df1.isnull().any()
```

Out[11]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

In [12]:

```
toBeStrip=['workclass','education','marital-status','occupation',
           'relationship','race','sex','native-country','Salary']
for elem in toBeStrip:
    df1[elem]=df1[elem].str.strip()
```

Studying Single attributes

In [13]:

```
df1.columns
```

Out[13]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country', 'Salary'],
      dtype='object')
```

Age

In [14]:

```
print("The Minimum Age is: ", df1['age'].min(),";",
      "The maximum age is: ", df1['age'].max(),";",
      "The average age is: ", df1['age'].mean(),";",
      "The mode age is: ", df1['age'].mode())
```

```
The Minimum Age is:  17 ; The maximum age is:  90 ; The average age is:  38.581633906633904 ; The
mode age is:  0    36
dtype: object
```

In [15]:

```
df1[df1['age']=='?'].any()
```

Out[15]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
```

```
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

In [16]:

```
from collections import Counter
l=[]
for elem in df1['age']:
    l.append(elem)
#Counter(l)
```

- I can create 4-5 variable for the attribute Age: <20; >20<40; >40<60; >60<80; >80

workclass

In [17]:

```
df1['workclass'].value_counts()
```

Out[17]:

```
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
?                    1836
State-gov            1297
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

In [18]:

```
df1[df1['workclass']=='?'].any()
```

Out[18]:

```
age               True
workclass         True
fnlwgt            True
education         True
marital-status    True
occupation        True
relationship      True
race              True
sex               True
capital-gain      True
capital-loss      True
hours-per-week    True
native-country    True
Salary            True
dtype: bool
```

- What can I do with "Without pay" and "Never Worked?" Moreover I have to remove rows with "?"

Education

In [19]:

```
df1['education'].value_counts()
```

Out[19]:

```
HS-grad         10501
Some-college     7291
Bachelors        5354
Masters          1723
Assoc-voc        1382
11th             1175
Assoc-acdm       1067
10th              933
7th-8th           646
Prof-school       576
9th               514
12th              433
Doctorate         413
5th-6th           333
1st-4th           168
Preschool          51
Name: education, dtype: int64
```

- It has to be reduced the number of levels of instruction classes

In [20]:

```
df1[df1['education']=='?'].any()
```

Out[20]:

```
age                 False
workclass           False
fnlwgt              False
education           False
marital-status      False
occupation          False
relationship        False
race                False
sex                 False
capital-gain        False
capital-loss        False
hours-per-week      False
native-country      False
Salary              False
dtype: bool
```

Marital Status

In [21]:

```
df1['marital-status'].value_counts()
```

Out[21]:

```
Married-civ-spouse       14976
Never-married            10682
Divorced                  4443
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: marital-status, dtype: int64
```

- Should I eliminate "married spouse absent" and "married af spouse?"

In [22]:

```
df1[df1['marital-status']=='?'].any()
```

Out[22]:

Out[22]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

Occupation

In [23]:

```
df1['occupation'].value_counts()
```

Out[23]:

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3769
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: occupation, dtype: int64
```

In [24]:

```
df1['occupation'].value_counts().shape[0]
```

Out[24]:

```
15
```

- I can remove "Armed-Forces" and "?"

In [25]:

```
df1[df1['occupation']=='?'].any()
```

Out[25]:

```
age               True
workclass         True
fnlwgt            True
education         True
marital-status    True
occupation        True
relationship      True
race              True
sex               True
capital-gain      True
capital-loss      True
hours-per-week    True
```

```
native-country    True
Salary            True
dtype: bool
```

relationship

In [26]:

```python
df1['relationship'].value_counts()
```

Out[26]:

```
Husband           13193
Not-in-family      8304
Own-child          5068
Unmarried          3446
Wife               1568
Other-relative      981
Name: relationship, dtype: int64
```

In [27]:

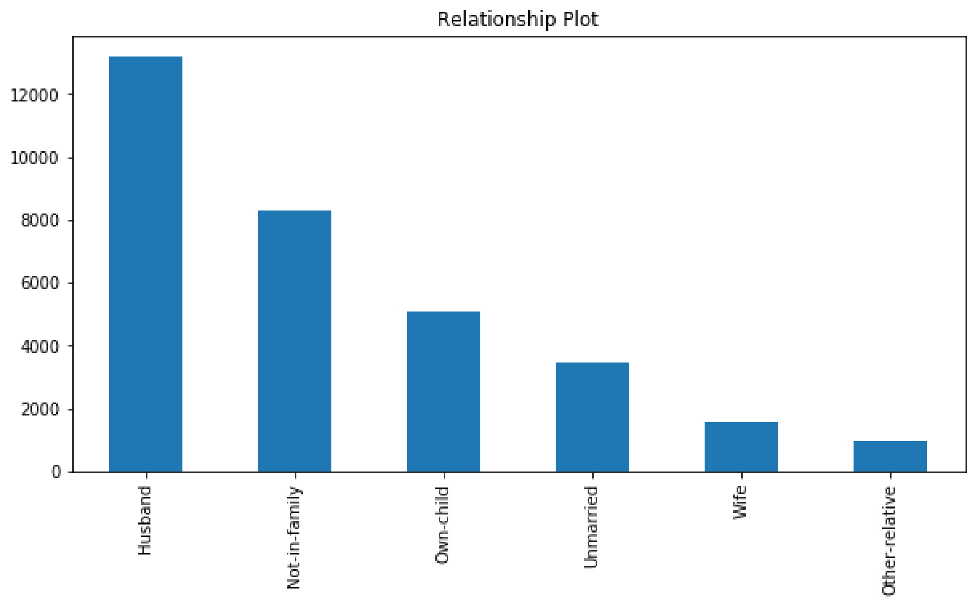
```python
df1[df1['relationship']=='?'].any()
```

Out[27]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

In [28]:

```python
RelationshipPlot=df1['relationship'].sort_values().value_counts().plot.bar(figsize=(10,5),x='Value'
, title="Relationship Plot")
```



race

In [29]:

```
df1['race'].value_counts()
```

Out[29]:

```
White                 27815
Black                  3124
Asian-Pac-Islander     1039
Amer-Indian-Eskimo      311
Other                   271
Name: race, dtype: int64
```

In [30]:

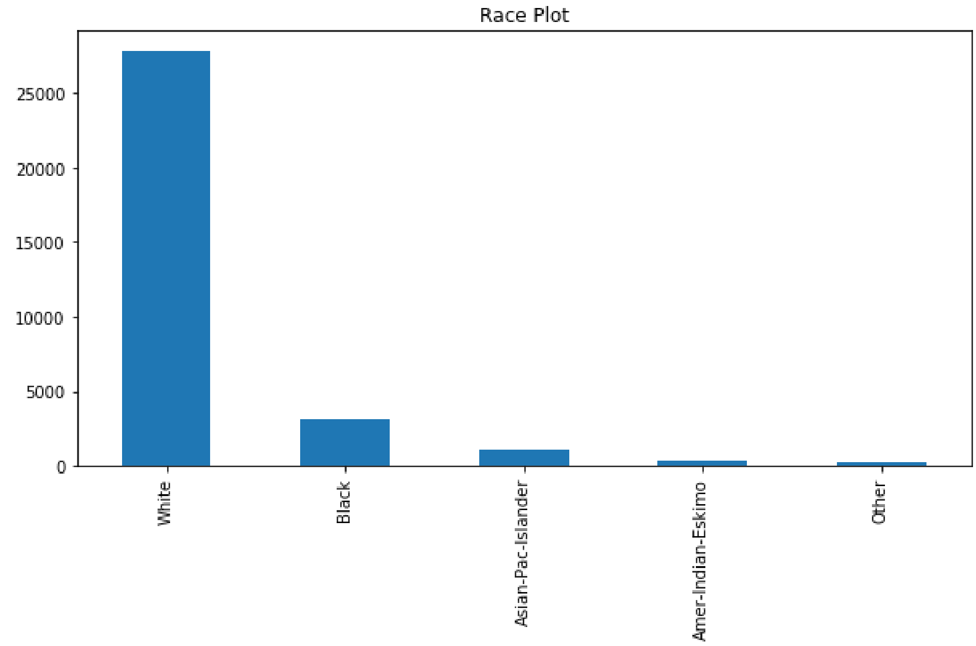
```
df1[df1['race']=='?'].any()
```

Out[30]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

In [31]:

```
RacePlot=df1['race'].sort_values().value_counts().plot.bar(figsize=(10,5),x='Value', title="Race Pl
ot")
```



Sex

In [32]:

```
df1['sex'].value_counts()
```

Out[32]:

```
Male      21789
Female    10771
Name: sex, dtype: int64
```

In [33]:

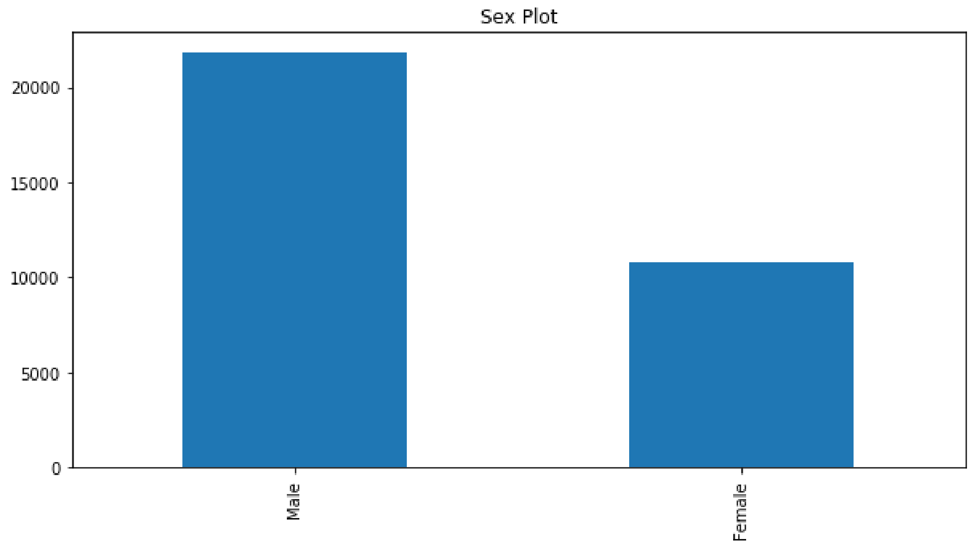
```
df1[df1['sex']=='?'].any()
```

Out[33]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

In [34]:

```
SexPlot=df1['sex'].sort_values().value_counts().plot.bar(figsize=(10,5),x='Value', title="Sex Plot"
)
```



capital-gain & Capital loss

In [35]:

```
df1['capital-gain'].value_counts()[:5]
```

Out[35]:

```
0        29849
15024      347
7688       284
7298       246
99999      159
Name: capital-gain, dtype: int64
```

In [36]:

```
df1['capital-loss'].value_counts()[:5]
```

Out[36]:

```
0       31041
1902      202
1977      168
1887      159
1485       51
Name: capital-loss, dtype: int64
```

In [37]:

```
df1[df1['capital-loss']==0].shape[0]
```

Out[37]:

```
31041
```

In [38]:

```
df1[df1['capital-gain']==0].shape[0]
```

Out[38]:

```
29849
```

- Probably Both to be removed

hours-per-week

In [39]:

```
df1['hours-per-week'].value_counts()[:5]
```

Out[39]:

```
40    15216
50     2819
45     1824
60     1475
35     1297
Name: hours-per-week, dtype: int64
```

In [40]:

```
df1['hours-per-week'].min()
```

Out[40]:

```
1
```

In [41]:

```
df1['hours-per-week'].max()
```

Out[41]:

```
99
```

In [42]:

```
df1['hours-per-week'].mean()
```

```
df1['hours-per-week'].mean()
```

Out[42]:

```
40.437469287469284
```

In [43]:

```
df1['hours-per-week'].mode()
```

Out[43]:

```
0    40
dtype: object
```

- This variable has to be arranged

In [44]:

```
df1[df1['hours-per-week']=='?'].any()
```

Out[44]:

```
age                 False
workclass           False
fnlwgt              False
education           False
marital-status      False
occupation          False
relationship        False
race                False
sex                 False
capital-gain        False
capital-loss        False
hours-per-week      False
native-country      False
Salary              False
dtype: bool
```

Native-Country

In [45]:

```
df1['native-country'].value_counts()[:10]
```

Out[45]:

```
United-States    29169
Mexico             643
?                  583
Philippines        198
Germany            137
Canada             121
Puerto-Rico        114
El-Salvador        106
India              100
Cuba                95
Name: native-country, dtype: int64
```

- Devo creare variabile others

In [46]:

```
df1[df1['native-country']=='?'].any()
```

Out[46]:

```
age                 True
```

```
workclass         True
fnlwgt            True
education         True
marital-status    True
occupation        True
relationship      True
race              True
sex               True
capital-gain      True
capital-loss      True
hours-per-week    True
native-country    True
Salary            True
dtype: bool
```

Salary

In [47]:

```
df1['Salary'].value_counts()
```

Out[47]:

```
<=50K    24719
>50K      7841
Name: Salary, dtype: int64
```

In [48]:

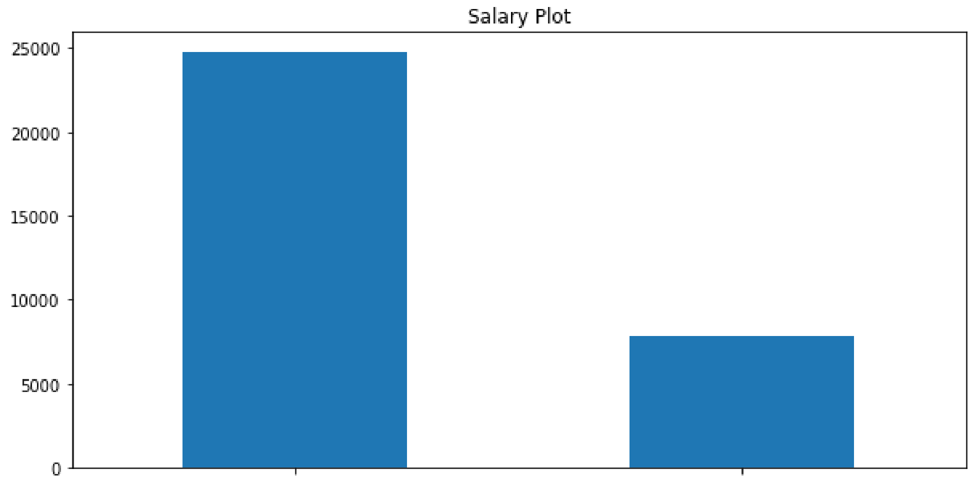
```
df1[df1['Salary']=='?'].any()
```

Out[48]:

```
age               False
workclass         False
fnlwgt            False
education         False
marital-status    False
occupation        False
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
Salary            False
dtype: bool
```

In [49]:

```
SalaryPlot=df1['Salary'].sort_values().value_counts().plot.bar(figsize=(10,5),x='Value', title="Salary Plot")
```



--

--

To do

1. Convert Unknown to "?" and eliminate it

In [50]:

```
beforeRows = df1.shape[0]
print(beforeRows)
```

```
32560
```

I eliminate all the rows in which there is at least a missing value in a certain attribute

Thanks to the prior analysis i know that missing values are in theese three instances: 1) workclass; 2) occupation; 3) native-country

First Filter

In [51]:

```
df1=df1[df1['workclass']!="?"]
```

In [52]:

```
df1.shape
```

Out[52]:

```
(30724, 14)
```

Second Filter

In [53]:

```
df1[df1['occupation']=='?'].any()
```

Out[53]:

```
age                True
workclass          True
fnlwgt             True
education          True
marital-status     True
occupation         True
relationship       True
race               True
sex                True
capital-gain      False
capital-loss      False
hours-per-week     True
native-country     True
```

```
Salary            True
dtype: bool
```

In [54]:

```
df1=df1[df1['occupation']!="?"]
```

In [55]:

```
df1.shape
```

Out[55]:

```
(30717, 14)
```

Third Filter

In [56]:

```
df1=df1[df1['native-country']!="?"]
```

In [57]:

```
df1.shape
```

Out[57]:

```
(30161, 14)
```

Here the remaining columns

In [58]:

```
afterRows = df1.shape[0]
print(afterRows)
```

```
30161
```

How many rows dropped due to cleaning?

In [59]:

```
beforeRows - afterRows
```

Out[59]:

```
2399
```

In [60]:

```
df1[:1]
```

Out[60]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Salary
0	50	Self-emp-not-inc	83311	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K

Generic code to make discrete all the cotinuous variables

Age

In [61]:

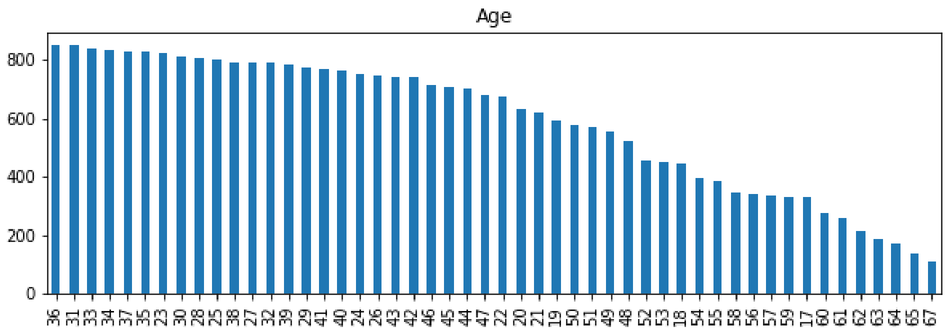
```
l_age=[]
for elem in df1['age']:
    if elem <20:
        l_age.append('<20')
    elif elem >20 and elem <40:
        l_age.append('>20 & <40')
    elif elem >40 and elem <60:
        l_age.append('>40 & <60')
    elif elem >60 and elem <80:
        l_age.append('>60 & <80')
    else:
        l_age.append('>80')
```

In [62]:

```
df1['l_age']=l_age
```

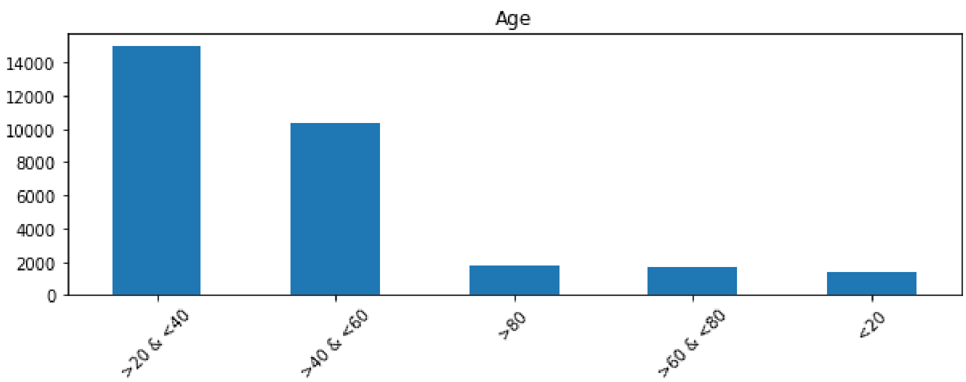
In [63]:

```
AgePlot=df1['age'].sort_values().value_counts()[:50].plot.bar(figsize=(10,3),x='Value', title="Age"
)
```



In [64]:

```
AgePlot=df1['l_age'].sort_values().value_counts().plot.bar(figsize=(10,3),x='Value', title="Age")
plt.xticks(rotation=45)
plt.show()
```



Workclass

In [65]:

```
RemoveLowCountInstancesCond1=df1['workclass']!="Never-worked"
RemoveLowCountInstancesCond2=df1['workclass']!="Without-pay"
```

In [66]:

```
df1=df1[RemoveLowCountInstancesCond1 & RemoveLowCountInstancesCond2]
```

In [67]:

```
df1.shape
```

Out[67]:

```
(30147, 15)
```

Education

In [68]:

```
df1['education'].value_counts()
```

Out[68]:

```
HS-grad         9831
Some-college    6675
Bachelors       5043
Masters         1627
Assoc-voc       1307
11th            1048
Assoc-acdm      1007
10th             820
7th-8th          556
Prof-school      542
9th              455
12th             377
Doctorate        375
5th-6th          288
1st-4th          151
Preschool         45
Name: education, dtype: int64
```

In [69]:

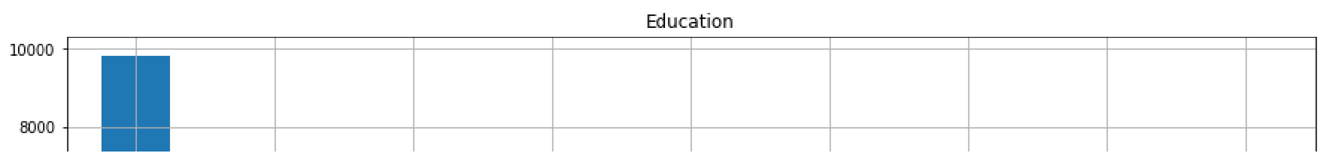
```
l_education=[]
for elem in df1['education']:
    if elem =="Preschool" or elem =="1st-4th"or elem =="5th-6th":
        l_education.append("Elementary school")
    elif elem =="7th-8th" or  elem =="9th" or elem =="10th" or elem =="11th" or elem =="12th":
        l_education.append("HS-Grad_NotCompleted")
    elif elem =="Assoc-voc" or elem =="Assoc-acdm":
        l_education.append("Assoc-acdm_or_Voc")
    else:
        l_education.append(elem)
```

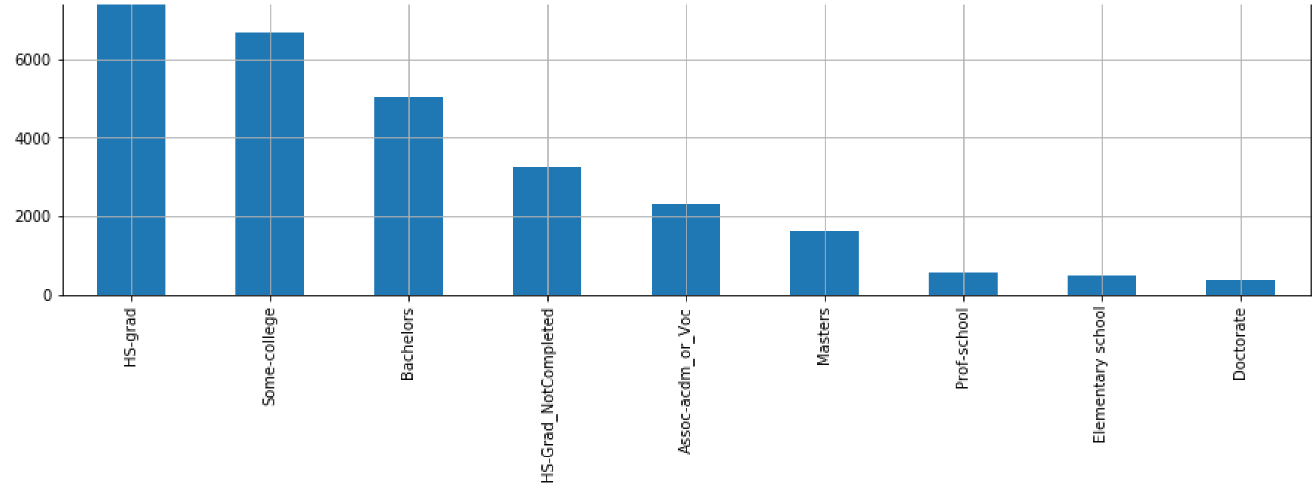
In [70]:

```
df1['l_education']=l_education
```

In [71]:

```
EducationPlot=df1['l_education'].sort_values().value_counts().plot.bar(figsize=(15,5),x='Value', title="Education", grid=True)
```





Marital Status

In [72]:

```
df1['marital-status'].value_counts()
```

Out[72]:

```
Married-civ-spouse       14057
Never-married             9721
Divorced                  4214
Separated                  939
Widowed                    826
Married-spouse-absent      369
Married-AF-spouse           21
Name: marital-status, dtype: int64
```

In [73]:

```
MaritalStatusToBeRemoved=df1['marital-status']!="Married-AF-spouse"
```

In [74]:

```
df1=df1[MaritalStatusToBeRemoved]
```

In [75]:

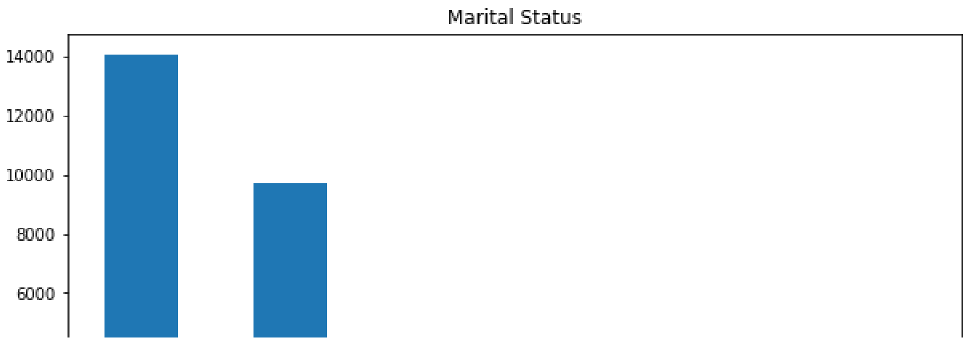
```
df1.shape
```

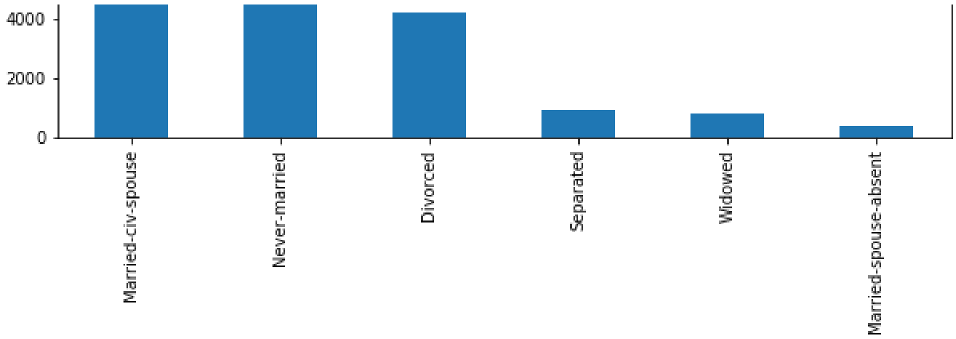
Out[75]:

```
(30126, 16)
```

In [76]:

```
HoursPerWeekPlot=df1['marital-
status'].sort_values().value_counts().plot.bar(figsize=(10,5),x='Value', title="Marital Status")
```





Occupation

In [77]:

```
df1['occupation'].value_counts()
```

Out[77]:

```
Prof-specialty       4035
Craft-repair         4026
Exec-managerial      3991
Adm-clerical         3712
Sales                3582
Other-service        3207
Machine-op-inspct    1965
Transport-moving     1570
Handlers-cleaners    1349
Farming-fishing       982
Tech-support          912
Protective-serv       643
Priv-house-serv       143
Armed-Forces            9
Name: occupation, dtype: int64
```

I decided to remove Armed Forces.

In [78]:

```
df1=df1[df1['occupation']!='Armed-Forces']
```

In [79]:

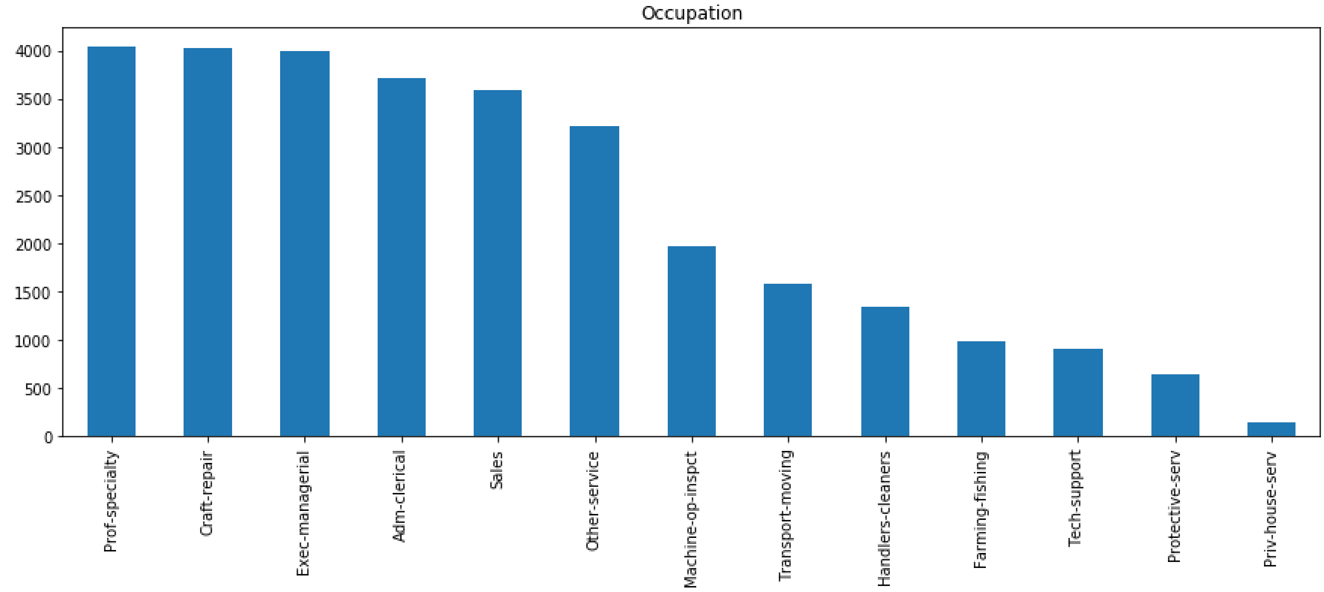
```
df1['occupation'].value_counts()
```

Out[79]:

```
Prof-specialty       4035
Craft-repair         4026
Exec-managerial      3991
Adm-clerical         3712
Sales                3582
Other-service        3207
Machine-op-inspct    1965
Transport-moving     1570
Handlers-cleaners    1349
Farming-fishing       982
Tech-support          912
Protective-serv       643
Priv-house-serv       143
Name: occupation, dtype: int64
```

In [80]:

```
OccupationPlot=df1['occupation'].sort_values().value_counts().plot.bar(figsize=(15,5),x='Value', ti
tle="Occupation")
```



Hours per Week

In [81]:

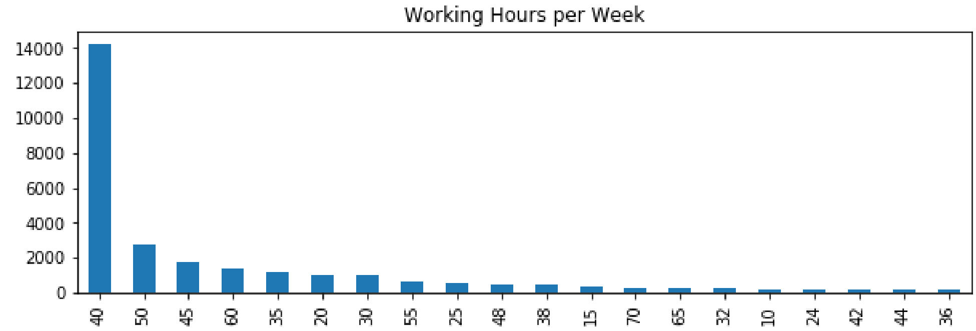
```
df1['hours-per-week'].sort_values().value_counts()[:20]
```

Out[81]:

```
40    14234
50     2713
45     1753
60     1403
35     1184
20     1052
30      988
55      670
25      570
48      493
38      455
15      346
70      276
65      240
32      238
10      220
24      217
42      212
44      208
36      202
Name: hours-per-week, dtype: int64
```

In [82]:

```
HoursPerWeekPlot=df1['hours-per-week'].sort_values().value_counts()[:20].plot.bar(figsize=(10,3),x
='Value', title="Working Hours per Week")
```



In [83]:

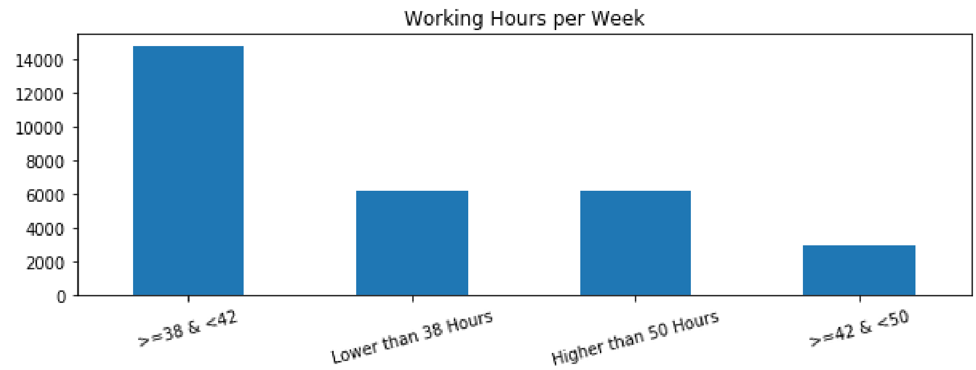
```
l_HoursPerWeek=[]
for elem in df1['hours-per-week']:
    if elem <38:
        l_HoursPerWeek.append("Lower than 38 Hours")
    elif elem >=38 and elem <42:
        l_HoursPerWeek.append(">=38 & <42")

    elif elem >=42 and elem <50:
        l_HoursPerWeek.append(">=42 & <50")
    else:
        l_HoursPerWeek.append("Higher than 50 Hours")
```

In [84]:

```
df1['l_HoursPerWeek']=l_HoursPerWeek
```

In [85]:

```
HoursPerWeekPlotDiscrete=df1['l_HoursPerWeek'].sort_values().value_counts().plot.bar(figsize=(10,3)
,x='Value', title="Working Hours per Week")
plt.xticks(rotation=15)
plt.show()
```



Native Country

In [86]:

```
df1['native-country'].value_counts()[:10]
```

Out[86]:

```
United-States    27460
Mexico             610
Philippines        187
Germany            128
Puerto-Rico        109
Canada             107
India              100
El-Salvador        100
Cuba                92
England             86
Name: native-country, dtype: int64
```

I decided to group in Others all the countries whose observations are lower than 100

In [87]:

```
l_Country=[]
for elem in df1['native-country']:
    if elem =='United-States':
        l_Country.append(elem)
    else:
```

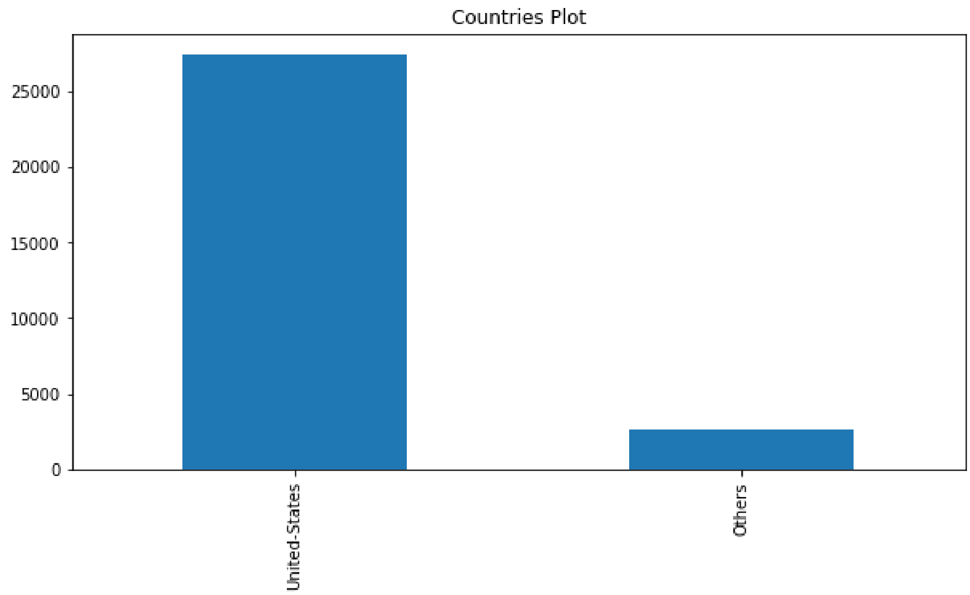
```
        l_Country.append('Others')
```

In [88]:

```
df1['l_Country']=l_Country
```

In [89]:

```
Country_Plot=df1['l_Country'].sort_values().value_counts().plot.bar(figsize=(10,5),x='Value', title
="Countries Plot")
```



In [90]:

```
df1.head()
```

Out[90]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Salary	l_a
0	50	Self-emp-not-inc	83311	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K	>4
1	38	Private	215646	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K	>2
2	53	Private	234721	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K	>4
3	28	Private	338409	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K	>2
4	37	Private	284582	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K	>2

In [91]:

```
CleanData=['l_age','workclass','l_education','marital-status','occupation','relationship','race','
sex','l_Country','Salary']
```

```
'sex', 'l_Country', 'Salary']
```

In [92]:

```
df2=df1[CleanData]
```

In [93]:

```
df2[:2]
```

Out[93]:

	l_age	workclass	l_education	marital-status	occupation	relationship	race	sex	l_Country	Salary
0	>40 & <60	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
1	>20 & <40	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K

Latest Check on the content of the attributes

In [94]:

```
df2['l_age'].value_counts()
```

Out[94]:

```
>20 & <40    14969
>40 & <60    10315
>80           1760
>60 & <80     1708
<20           1365
Name: l_age, dtype: int64
```

In [95]:

```
df2['workclass'].value_counts()
```

Out[95]:

```
Private             22271
Self-emp-not-inc     2497
Local-gov            2067
State-gov            1277
Self-emp-inc         1074
Federal-gov           931
Name: workclass, dtype: int64
```

In [96]:

```
df2['l_education'].value_counts()
```

Out[96]:

```
HS-grad                 9815
Some-college            6670
Bachelors               5038
HS-Grad_NotCompleted    3255
Assoc-acdm_or_Voc       2312
Masters                 1626
Prof-school              542
Elementary school        484
Doctorate                375
Name: l_education, dtype: int64
```

In [97]:

```
df2['marital-status'].value_counts()
```

Out[97]:

```
Married-civ-spouse       14054
Never-married             9715
Divorced                  4214
Separated                  939
Widowed                    826
Married-spouse-absent      369
Name: marital-status, dtype: int64
```

In [98]:

```
df2['occupation'].value_counts()
```

Out[98]:

```
Prof-specialty       4035
Craft-repair         4026
Exec-managerial      3991
Adm-clerical         3712
Sales                3582
Other-service        3207
Machine-op-inspct    1965
Transport-moving     1570
Handlers-cleaners    1349
Farming-fishing       982
Tech-support          912
Protective-serv       643
Priv-house-serv       143
Name: occupation, dtype: int64
```

In [99]:

```
df2['relationship'].value_counts()
```

Out[99]:

```
Husband           12448
Not-in-family      7721
Own-child          4459
Unmarried          3210
Wife               1393
Other-relative      886
Name: relationship, dtype: int64
```

In [100]:

```
df2['race'].value_counts()
```

Out[100]:

```
White                 25893
Black                  2814
Asian-Pac-Islander      894
Amer-Indian-Eskimo      285
Other                   231
Name: race, dtype: int64
```

In [101]:

```
df2['sex'].value_counts()
```

Out[101]:

```
Male      20352
Female     9765
Name: sex, dtype: int64
```

In [102]:

```
df2['l_Country'].value_counts()
```

Out[102]:

```
United-States    27460
Others            2657
Name: l_Country, dtype: int64
```

In [103]:

```
df2['Salary'].value_counts()
```

Out[103]:

```
<=50K    22620
>50K      7497
Name: Salary, dtype: int64
```

In [ ]:

```

```

It seems to be good. Now we can start with the classification activities

In [104]:

```
df2[-2:]
```

Out[104]:

	l_age	workclass	l_education	marital-status	occupation	relationship	race	sex	l_Country	Salary
32558	>20 & <40	Private	HS-grad	Never-married	Adm-clerical	Own-child	White	Male	United-States	<=50K
32559	>40 & <60	Self-emp-inc	HS-grad	Married-civ-spouse	Exec-managerial	Wife	White	Female	United-States	>50K

Now, I need to encode correctly my dataset

In [105]:

```
le = preprocessing.LabelEncoder()
df3 = df2.apply(le.fit_transform)
```

In [106]:

```
df3[:20]
```

Out[106]:

	l_age	workclass	l_education	marital-status	occupation	relationship	race	sex	l_Country	Salary
0	2	4	1	1	2	0	4	1	1	0
1	1	2	5	0	4	1	4	1	1	0
2	2	2	4	1	4	0	2	1	1	0
3	1	2	1	1	8	5	2	0	0	0
4	1	2	6	1	2	5	4	0	1	0
5	2	2	4	2	6	1	2	0	0	0
6	2	4	5	1	2	0	4	1	1	1
7	1	2	6	3	8	1	4	0	1	1
8	2	2	1	1	2	0	4	1	1	1

	l_age	workclass	l_education	marital-status	occupation	relationship	race	sex	l_Country	Salary
9	1	2	8	1	2	0	2	1	1	1
10	1	5	1	1	8	0	1	1	0	1
11	1	2	1	3	0	3	4	0	1	0
12	1	2	0	3	10	1	2	1	1	0
14	1	2	4	1	12	0	0	1	0	0
15	1	4	5	3	3	3	4	1	1	0
16	1	2	5	3	5	4	4	1	1	0
17	1	2	4	1	10	0	4	1	1	0
18	2	4	6	0	2	4	4	0	1	1
19	4	2	2	1	8	0	4	1	1	1
20	2	2	5	4	6	4	2	0	1	0

- I create my target Variable

In [107]:

```
y=df3[['Salary']].copy()
y
```

Out[107]:

	Salary
0	0
1	0
2	0
3	0
4	0
...	...
32555	0
32556	1
32557	0
32558	0
32559	1

30117 rows × 1 columns

In [108]:

```
X=df3.iloc[:,0:9]
```

In [109]:

```
X.columns
```

Out[109]:

```
Index(['l_age', 'workclass', 'l_education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'l_Country'],
      dtype='object')
```

Training Phase

In [110]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=324)
```

In [111]:

```
#type(X_train)
#type(X_test)
#type(y_train)
#type(y_test)
#X_train.head()
#y_train.describe()
#X_test.describe()
```

In [112]:

```
IncomeClassifier = DecisionTreeClassifier(random_state=0,max_leaf_nodes=20)
IncomeClassifier.fit(X_train, y_train)
```

Out[112]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=20,
                       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=0, splitter='best')
```

In [113]:

```
type(IncomeClassifier)
```

Out[113]:

```
sklearn.tree.tree.DecisionTreeClassifier
```

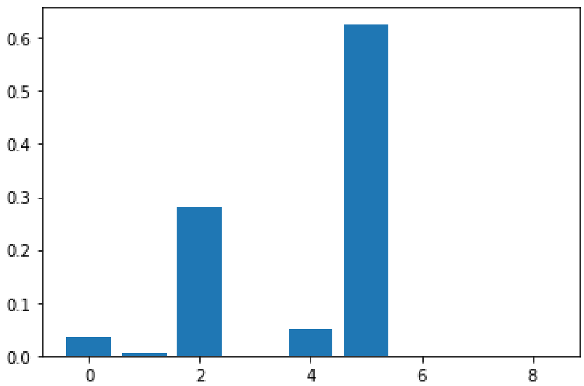
In [114]:

```
print(IncomeClassifier.feature_importances_)
```

```
[0.03562728 0.00585681 0.28184603 0.         0.05089583 0.62577404
 0.         0.         0.        ]
```

In [115]:

```
pyplot.bar(range(len(IncomeClassifier.feature_importances_)),
IncomeClassifier.feature_importances_)
pyplot.show()
```



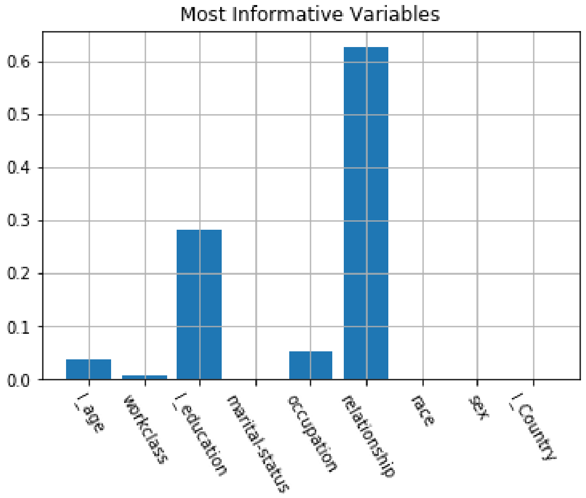
In [116]:

```
names=X.columns
```

In [117]:

```
pyplot.bar(range(len(IncomeClassifier.feature_importances_)),
IncomeClassifier.feature_importances_)
```

```
pyplot.xticks(range(X.shape[1]), names, rotation=-60)
pyplot.title('Most Informative Variables')
pyplot.grid(True)
pyplot.show()
```



I print the decision tree

In [118]:

```
dot_data = tree.export_graphviz(IncomeClassifier, out_file=None,filled = True,feature_names=names,c
lass_names=df2['Salary'].unique())
graph = graphviz.Source(dot_data)
pydot_graph = pydotplus.graph_from_dot_data(dot_data)
pydot_graph.write_png('original_tree.png')
pydot_graph.set_size('"10,10!"')
pydot_graph.write_png('resized_tree.png')
```

Out[118]:

```
True
```

In [119]:

```
graph
```

Out[119]:

In [ ]:

```

```

Prediction on Test Set

In [120]:

```
predictions = IncomeClassifier.predict(X_test)
```

In [121]:

```
predictions[-20:]
```

Out[121]:

```
array([1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [122]:

```
y_test['Salary'][-20:]
```

Out[122]:

```
3322     0
31131    0
19679    0
8480     0
9506     0
23265    1
17814    0
7524     0
24188    0
18555    0
994      1
2116     0
22198    0
6641     0
27912    0
5270     0
27457    1
28073    0
6593     0
18715    0
Name: Salary, dtype: int32
```

Measure of accuracy

In [123]:

```
accuracy_score(y_true = y_test,
               y_pred = predictions)
```

Out[123]:

```
0.820303853506389
```

In [ ]:

```

```