Name : Christian Vargas
Institution: UCSC Silicon Valley
Course: Python for programmers
Date: Sep-09-17

Final project – Data Analysis and Machine learning Income Data (census)

1. Introduction:

Machine learning (ML) has evolved during the last few years with great advances in compute capacity and availability. ML refers to the process by which software programs learn from experience from data and use this to make predictions and classifications mostly.

The goal of this project is to train a binary classifier to predict the column <income> which has two possible values ">50K" and "<=50K" using the following ML algorithms:

- **Logistic regression** using python libraries **Sklearn** and **Statsmodels**

  "Logistic Regression is a mathematical model used in statistics to estimate (guess) the probability of an event occurring having been given some previous data. Logistic Regression works with binary data, where either the event happens (1) or the event does not happen (0). (Wikipedia)

- **Decision tree classifier** and **regression** using python library **Sklearn**

  "Decision Trees (DTs) *are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features." (Wikipedia)*

After the ML algorithms made the prediction is necessary evaluate the accuracy of the classifier, for this process I will use two methods:

- **Classification accuracy (ACC):**

  "ACC is the number of correct predictions made divided by the total number of predictions made." (Wikipedia)

- **Area under ROC Curve (AUC):**

  "AUC is a performance metric for binary classification problems. The AUC represents a model's ability to discriminate between positive and negative classes." (Wikipedia)"

For this project, I will use **US Adult Census data** relating income to social factors such as Age, Education, race etc. **Adult dataset** is made up of categorical (classes) and continuous features (numeric) which include missing values, each row is labelled as either having a salary greater than ">50K" or "<=50K".

- The **categorical** columns are: *workclass, education, marital_status, occupation, relationship, race, gender, native_country.*
- The **continuous** columns are: *age, education_num, capital_gain, capital_loss, hours_per_week.*

2. Requirements

This project will use **Spyder** from **Anaconda Python 2.7** as main programs. To run the code is necessary:

- **Data file** = adult.csv file (Raw Data)

- **Python Libraries:**

```
#==================== Data preparation (cleansing) =========================#
#Numerics and Data Analysis
import pandas as pd #Pandas
import numpy as np #Numpy

#To store results as objects
from sklearn.datasets.base import Bunch #Dictionary

# Graphics and Vizualization
import seaborn as sns
import matplotlib.pyplot as plt

# Categorization of data
from sklearn.preprocessing import LabelEncoder

#==================== Machine Learning algorithms ==========================#
#logistict regression.
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm # statsmodel is chosen because it outputs descriptive stats for the model

# Decision Trees
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor

# Accuracy validation
from scipy.stats import pointbiserialr, spearmanr
from sklearn.cross_validation import cross_val_score
import sklearn.cross_validation as cross_validation
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.cross_validation import StratifiedShuffleSplit # split data test and train
```

3. Python program description

The python program I wrote is organized of the following form:

3.1. Importing needed libraries:

3.2. Data cleansing, visualization and preparation function definitions:

- **3.2.1.** def load_csv(filename): load csv file
- **3.2.2.** def checking_data(df): missing values
- **3.2.3.** def clean_data(df): cleaning data
- **3.2.4.** def visualize_data(data): plotting and visualization
- **3.2.5.** def number_encode_features(data): making data numeric
- **3.2.6.** def prepare_data(data): splitting data training, test and targets

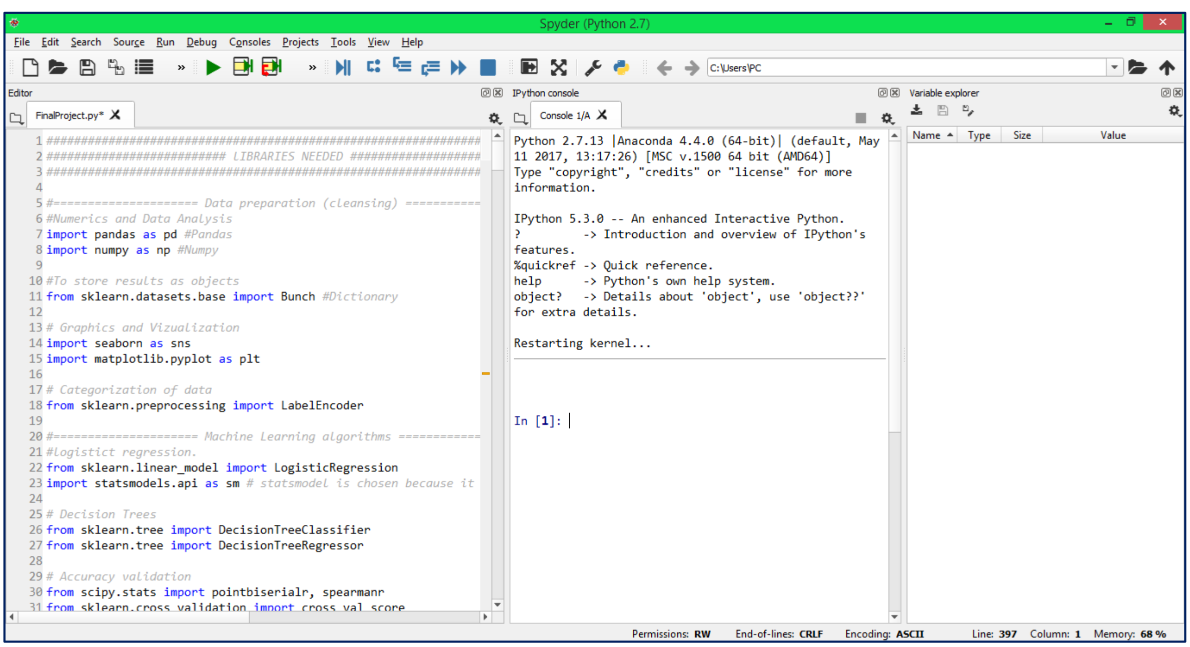
3.3. Machine Learning function definitions:

- **3.3.1.** def correlation (data):
- **3.3.2.** def log_regre_statsmodels():
- **3.3.3.** def log_regre_sklearn():
- **3.3.4.** def decision_tree_classifier(data,target):
- **3.3.5.** def decision_tree_regre(data,target):

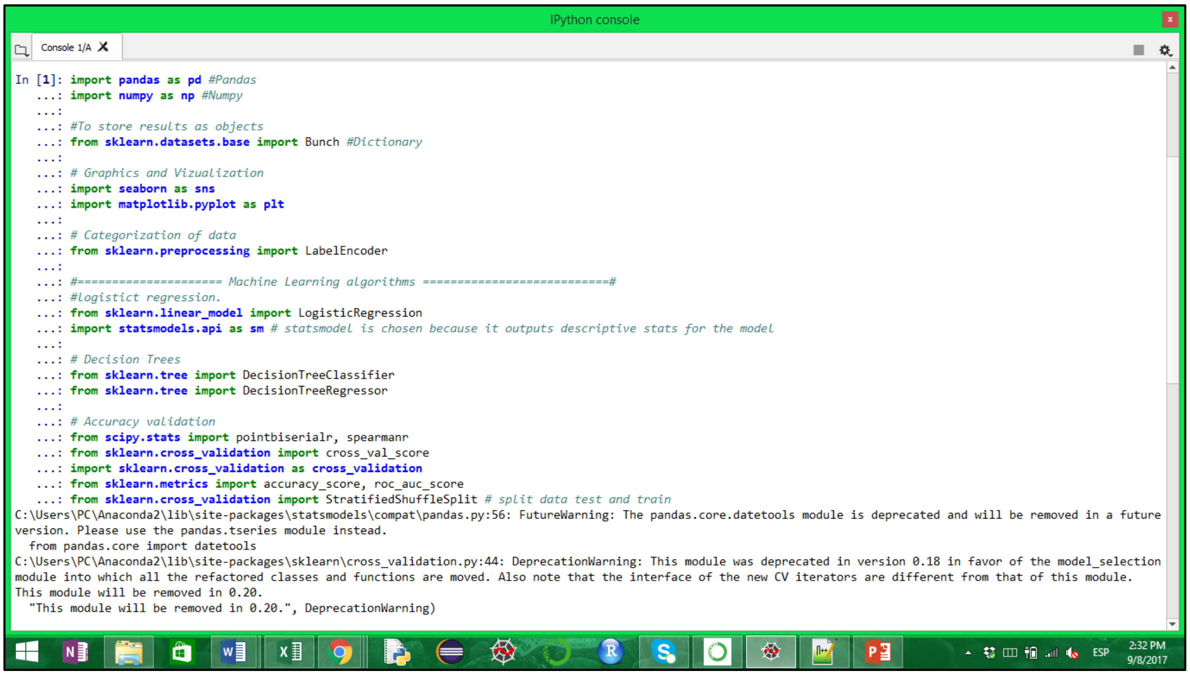
3.4. Main code execution:

4. Screenshots of the program output

4.1. Initial view from Spyder without any variables and with the kernel restarted



4.2. Library import process



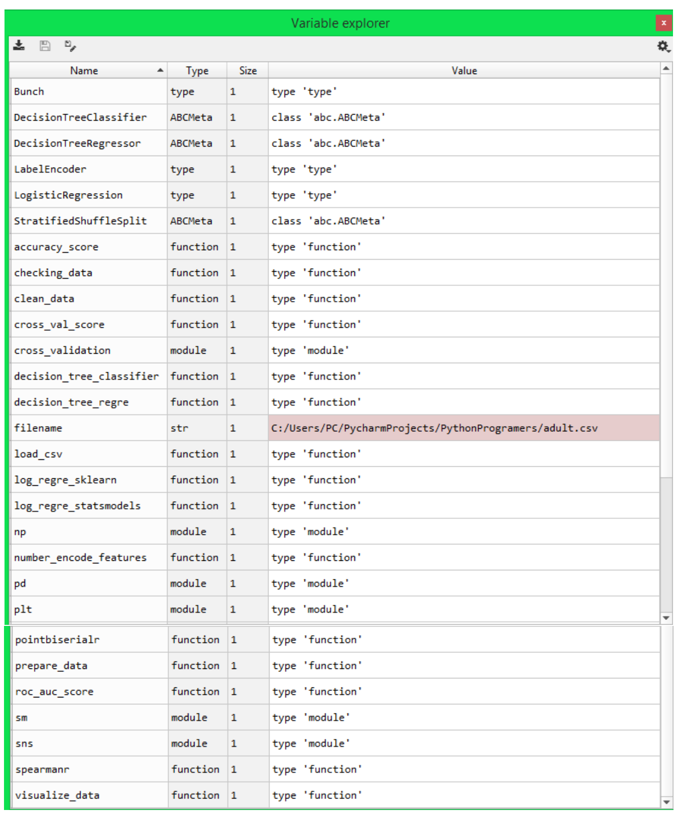
IPython console

Console 1/A

```
In [1]: import pandas as pd #Pandas
   ...: import numpy as np #Numpy
   ...:
   ...: #To store results as objects
   ...: from sklearn.datasets.base import Bunch #Dictionary
   ...:
   ...: # Graphics and Vizualization
   ...: import seaborn as sns
   ...: import matplotlib.pyplot as plt
   ...:
   ...: # Categorization of data
   ...: from sklearn.preprocessing import LabelEncoder
   ...:
   ...: #==================== Machine Learning algorithms ==========================#
   ...: #Logistict regression.
   ...: from sklearn.linear_model import LogisticRegression
   ...: import statsmodels.api as sm # statsmodel is chosen because it outputs descriptive stats for the model
   ...:
   ...: # Decision Trees
   ...: from sklearn.tree import DecisionTreeClassifier
   ...: from sklearn.tree import DecisionTreeRegressor
   ...:
   ...: # Accuracy validation
   ...: from scipy.stats import pointbiserialr, spearmanr
   ...: from sklearn.cross_validation import cross_val_score
   ...: import sklearn.cross_validation as cross_validation
   ...: from sklearn.metrics import accuracy_score, roc_auc_score
   ...: from sklearn.cross_validation import StratifiedShuffleSplit # split data test and train
C:\Users\PC\Anaconda2\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core.datetools module is deprecated and will be removed in a future
version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
C:\Users\PC\Anaconda2\lib\site-packages\sklearn\cross_validation.py:44: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection
module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module.
This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

2:32 PM
9/8/2017

4.3. Reading function definitions

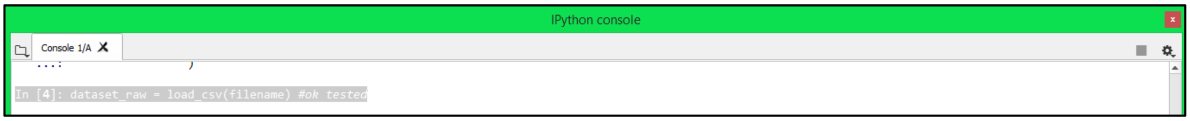


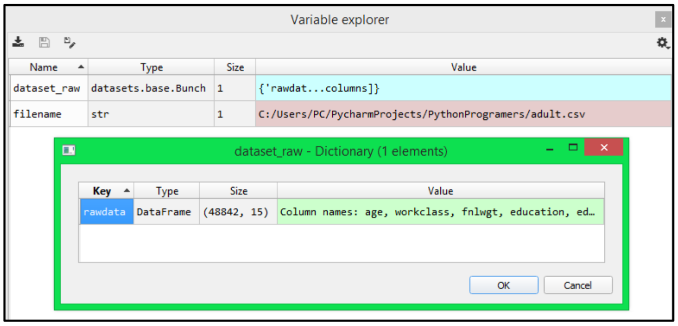
Variable explorer

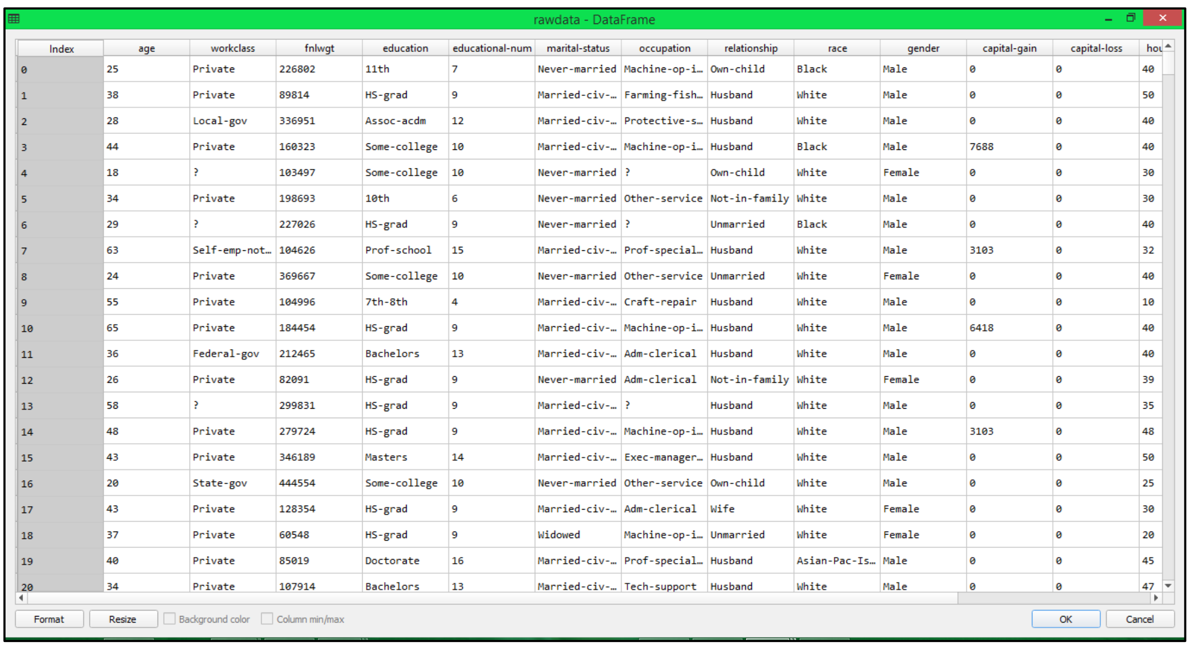
Name	Type	Size	Value
Bunch	type	1	type 'type'
DecisionTreeClassifier	ABCMeta	1	class 'abc.ABCMeta'
DecisionTreeRegressor	ABCMeta	1	class 'abc.ABCMeta'
LabelEncoder	type	1	type 'type'
LogisticRegression	type	1	type 'type'
StratifiedShuffleSplit	ABCMeta	1	class 'abc.ABCMeta'
accuracy_score	function	1	type 'function'
checking_data	function	1	type 'function'
clean_data	function	1	type 'function'
cross_val_score	function	1	type 'function'
cross_validation	module	1	type 'module'
decision_tree_classifier	function	1	type 'function'
decision_tree_regre	function	1	type 'function'
filename	str	1	C:/Users/PC/PycharmProjects/PythonProgramers/adult.csv
load_csv	function	1	type 'function'
log_regre_sklearn	function	1	type 'function'
log_regre_statsmodels	function	1	type 'function'
np	module	1	type 'module'
number_encode_features	function	1	type 'function'
pd	module	1	type 'module'
plt	module	1	type 'module'
pointbiserialr	function	1	type 'function'
prepare_data	function	1	type 'function'
roc_auc_score	function	1	type 'function'
sm	module	1	type 'module'
sns	module	1	type 'module'
spearmanr	function	1	type 'function'
visualize_data	function	1	type 'function'

4.4. Main Execution

4.4.1. Loading csv file



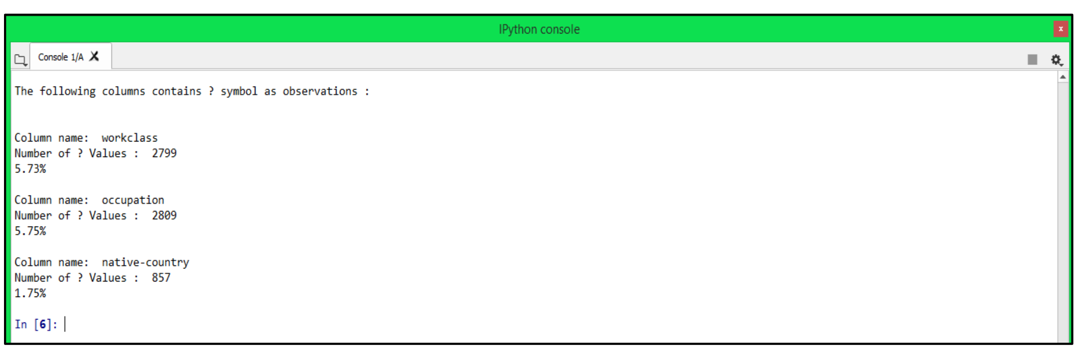




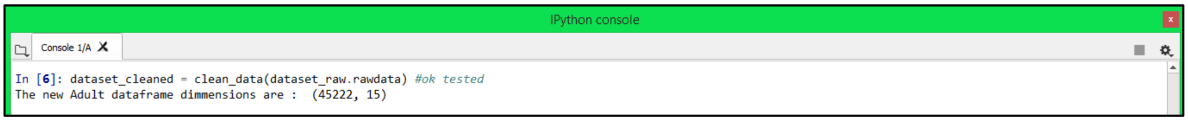
rawdata - DataFrame

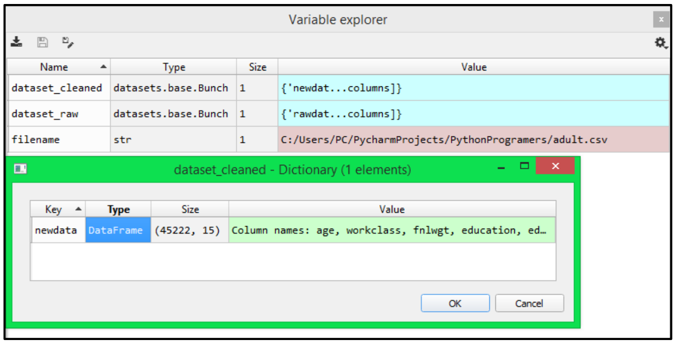
Index	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hou
0	25	Private	226802	11th	7	Never-married	Machine-op-i…	Own-child	Black	Male	0	0	40
1	38	Private	89814	HS-grad	9	Married-civ-…	Farming-fish…	Husband	White	Male	0	0	50
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-…	Protective-s…	Husband	White	Male	0	0	40
3	44	Private	160323	Some-college	10	Married-civ-…	Machine-op-i…	Husband	Black	Male	7688	0	40
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30
6	29	?	227026	HS-grad	9	Never-married	?	Unmarried	Black	Male	0	0	40
7	63	Self-emp-not…	104626	Prof-school	15	Married-civ-…	Prof-special…	Husband	White	Male	3103	0	32
8	24	Private	369667	Some-college	10	Never-married	Other-service	Unmarried	White	Female	0	0	40
9	55	Private	104996	7th-8th	4	Married-civ-…	Craft-repair	Husband	White	Male	0	0	10
10	65	Private	184454	HS-grad	9	Married-civ-…	Machine-op-i…	Husband	White	Male	6418	0	40
11	36	Federal-gov	212465	Bachelors	13	Married-civ-…	Adm-clerical	Husband	White	Male	0	0	40
12	26	Private	82091	HS-grad	9	Never-married	Adm-clerical	Not-in-family	White	Female	0	0	39
13	58	?	299831	HS-grad	9	Married-civ-…	?	Husband	White	Male	0	0	35
14	48	Private	279724	HS-grad	9	Married-civ-…	Machine-op-i…	Husband	White	Male	3103	0	48
15	43	Private	346189	Masters	14	Married-civ-…	Exec-manager…	Husband	White	Male	0	0	50
16	20	State-gov	444554	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	25
17	43	Private	128354	HS-grad	9	Married-civ-…	Adm-clerical	Wife	White	Female	0	0	30
18	37	Private	60548	HS-grad	9	Widowed	Machine-op-i…	Unmarried	White	Female	0	0	20
19	40	Private	85019	Doctorate	16	Married-civ-…	Prof-special…	Husband	Asian-Pac-Is…	Male	0	0	45
20	34	Private	107914	Bachelors	13	Married-civ-…	Tech-support	Husband	White	Male	0	0	47

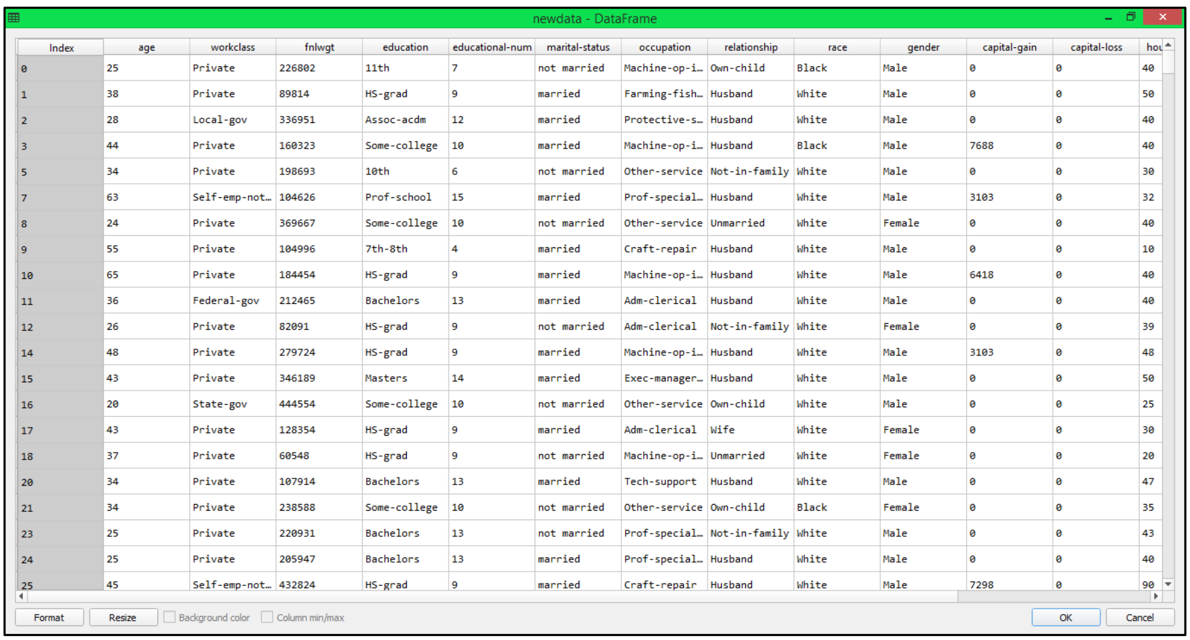
4.4.2. Checking data NA values “?”



4.4.3. Cleaning dataset, eliminating missing values







newdata - DataFrame

Index	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hou
0	25	Private	226802	11th	7	not married	Machine-op-i…	Own-child	Black	Male	0	0	40
1	38	Private	89814	HS-grad	9	married	Farming-fish…	Husband	White	Male	0	0	50
2	28	Local-gov	336951	Assoc-acdm	12	married	Protective-s…	Husband	White	Male	0	0	40
3	44	Private	160323	Some-college	10	married	Machine-op-i…	Husband	Black	Male	7688	0	40
5	34	Private	198693	10th	6	not married	Other-service	Not-in-family	White	Male	0	0	30
7	63	Self-emp-not…	104626	Prof-school	15	married	Prof-special…	Husband	White	Male	3103	0	32
8	24	Private	369667	Some-college	10	not married	Other-service	Unmarried	White	Female	0	0	40
9	55	Private	104996	7th-8th	4	married	Craft-repair	Husband	White	Male	0	0	10
10	65	Private	184454	HS-grad	9	married	Machine-op-i…	Husband	White	Male	6418	0	40
11	36	Federal-gov	212465	Bachelors	13	married	Adm-clerical	Husband	White	Male	0	0	40
12	26	Private	82091	HS-grad	9	not married	Adm-clerical	Not-in-family	White	Female	0	0	39
14	48	Private	279724	HS-grad	9	married	Machine-op-i…	Husband	White	Male	3103	0	48
15	43	Private	346189	Masters	14	married	Exec-manager…	Husband	White	Male	0	0	50
16	20	State-gov	444554	Some-college	10	not married	Other-service	Own-child	White	Male	0	0	25
17	43	Private	128354	HS-grad	9	married	Adm-clerical	Wife	White	Female	0	0	30
18	37	Private	60548	HS-grad	9	not married	Machine-op-i…	Unmarried	White	Female	0	0	20
20	34	Private	107914	Bachelors	13	married	Tech-support	Husband	White	Male	0	0	47
21	34	Private	238588	Some-college	10	not married	Other-service	Own-child	Black	Female	0	0	35
23	25	Private	220931	Bachelors	13	not married	Prof-special…	Not-in-family	White	Male	0	0	43
24	25	Private	205947	Bachelors	13	married	Prof-special…	Husband	White	Male	0	0	40
25	45	Self-emp-not…	432824	HS-grad	9	married	Craft-repair	Husband	White	Male	7298	0	90

Format Resize Background color Column min/max OK Cancel

4.4.4. The data preparation to machine learning process needs to create train and test datasets respectively.

IPython console

Console 1/A

```
In [7]: dataset_prepared = prepare_data(dataset_cleaned.newdata) #ok tested
```

dataset_prepared - Dictionary (10 elements)

Key	Type	Size	Value
categorical_features	dict	9	{'education': ['11th', 'HS-grad', 'Assoc-acdm', 'Some-coll…
data	DataFrame	(45222, 14)	Column names: age, workclass, fnlwgt, education, education…
data_target	Series	(45222L,)	class 'pandas.core.series.Series'
feature_names	list	15	['age', 'workclass', 'fnlwgt', 'education', 'educational-n…
rawdata	DataFrame	(45222, 15)	Column names: age, workclass, fnlwgt, education, education…
target_names	list	2	['<=50K', '>50K']
target_test	Series	(11306L,)	class 'pandas.core.series.Series'
test	DataFrame	(11306, 14)	Column names: age, workclass, fnlwgt, education, education…
train	DataFrame	(33916, 14)	Column names: age, workclass, fnlwgt, education, education…
train_target	Series	(33916L,)	class 'pandas.core.series.Series'

Data without <income> column

data - DataFrame

Index		fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country
0		226802	11th	7	not married	Machine-op-i…	Own-child	Black	Male	0	0	40	United-States
1		89814	HS-grad	9	married	Farming-fish…	Husband	White	Male	0	0	50	United-States
2		336951	Assoc-acdm	12	married	Protective-s…	Husband	White	Male	0	0	40	United-States
3		160323	Some-college	10	married	Machine-op-i…	Husband	Black	Male	7688	0	40	United-States
5		198693	10th	6	not married	Other-service	Not-in-family	White	Male	0	0	30	United-States
7	ot…	104626	Prof-school	15	married	Prof-special…	Husband	White	Male	3103	0	32	United-States
8		369667	Some-college	10	not married	Other-service	Unmarried	White	Female	0	0	40	United-States
9		104996	7th-8th	4	married	Craft-repair	Husband	White	Male	0	0	10	United-States
10		184454	HS-grad	9	married	Machine-op-i…	Husband	White	Male	6418	0	40	United-States
11	v	212465	Bachelors	13	married	Adm-clerical	Husband	White	Male	0	0	40	United-States
12		82091	HS-grad	9	not married	Adm-clerical	Not-in-family	White	Female	0	0	39	United-States
14		279724	HS-grad	9	married	Machine-op-i…	Husband	White	Male	3103	0	48	United-States
15		346189	Masters	14	married	Exec-manager…	Husband	White	Male	0	0	50	United-States
16		444554	Some-college	10	not married	Other-service	Own-child	White	Male	0	0	25	United-States
17		128354	HS-grad	9	married	Adm-clerical	Wife	White	Female	0	0	30	United-States
18		60548	HS-grad	9	not married	Machine-op-i…	Unmarried	White	Female	0	0	20	United-States
20		107914	Bachelors	13	married	Tech-support	Husband	White	Male	0	0	47	United-States
21		238588	Some-college	10	not married	Other-service	Own-child	Black	Female	0	0	35	United-States
23		220931	Bachelors	13	not married	Prof-special…	Not-in-family	White	Male	0	0	43	Peru
24		205947	Bachelors	13	married	Prof-special…	Husband	White	Male	0	0	40	United-States
25	ot…	432824	HS-grad	9	married	Craft-repair	Husband	White	Male	7298	0	90	United-States

Format Resize Background color Column min/max OK Cancel

data_target - Series

Index	income
0	<=50K
1	<=50K
2	>50K
3	>50K
5	<=50K
7	>50K
8	<=50K
9	<=50K
10	>50K
11	<=50K
12	<=50K

4.4.5. Data Preparation

IPython console

Console 1/A

```
In [8]: dataset_prepared = prepare_data(dataset_cleaned.newdata) #ok tested

In [9]:
```

4.4.6. Convert categorical data to numerical data

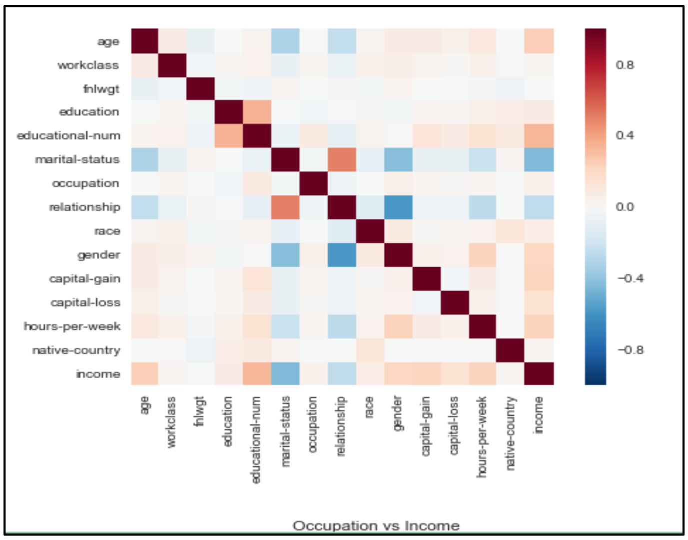
IPython console

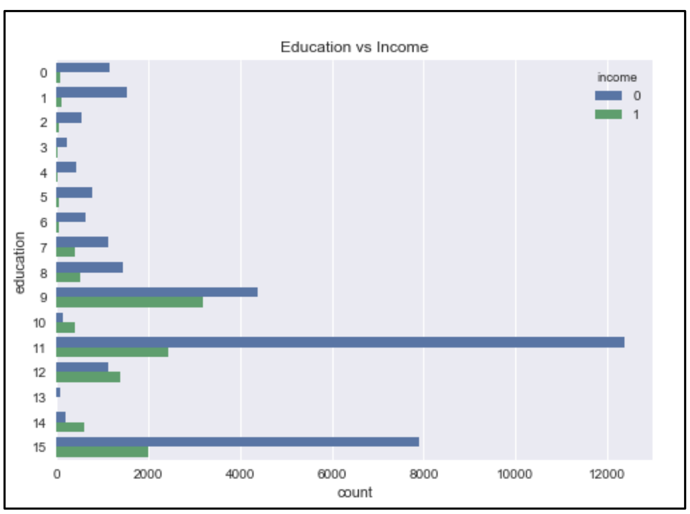
Console 1/A

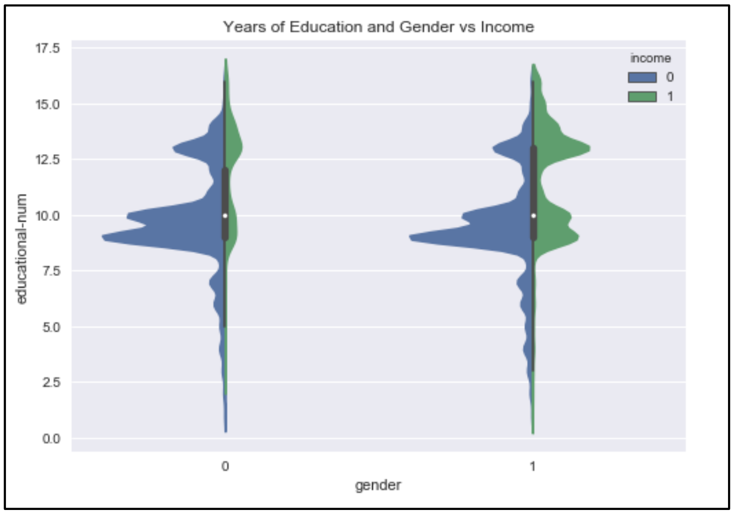
```
In [9]: dataset_encoded = number_encode_features(dataset_cleaned.newdata) #ok tested

In [10]:
```

4.4.7. Plotting

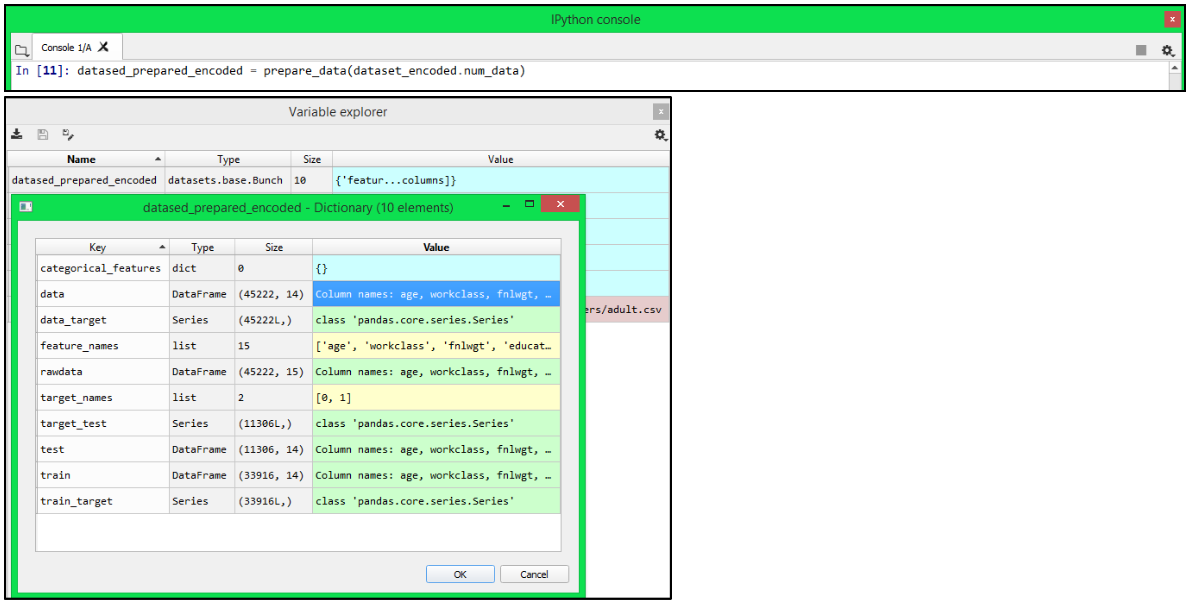
```
In [10]: visualize_data(dataset_encoded.num_data) #
ok tested
```





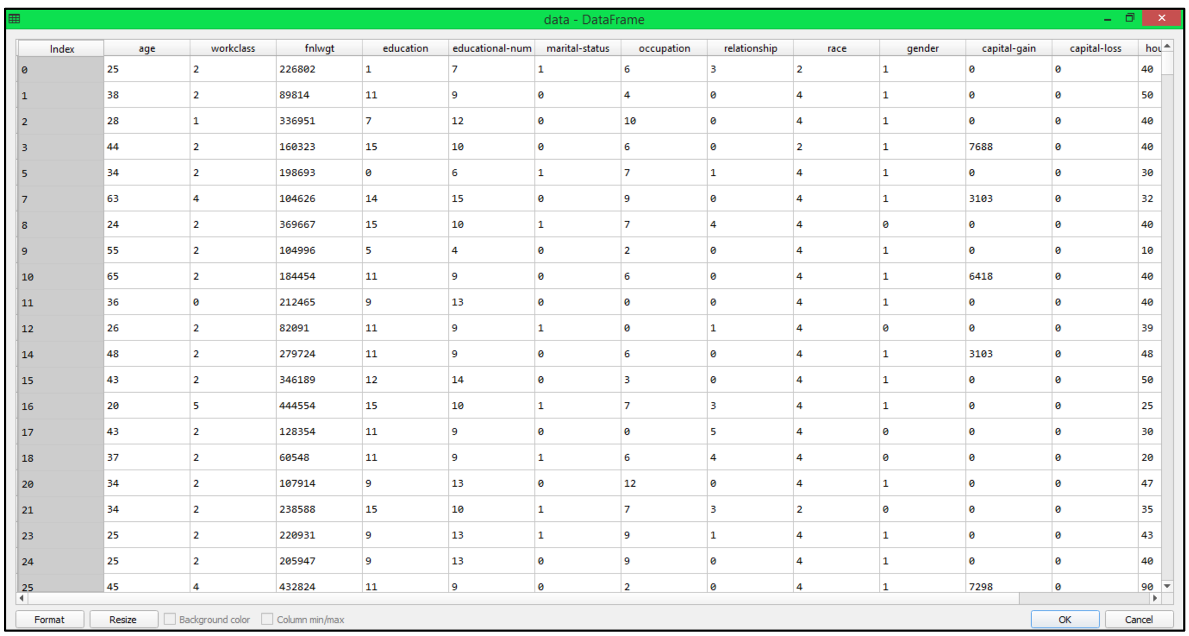

Years of Education and Gender vs Income
income
0
1
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
educational-num
0
1
gender

4.4.8. Splitting data test, train, and targets.



Name	Type	Size	Value
datased_prepared_encoded	datasets.base.Bunch	10	{'featur...columns]}

Key	Type	Size	Value
categorical_features	dict	0	{}
data	DataFrame	(45222, 14)	Column names: age, workclass, fnlwgt, …
data_target	Series	(45222L,)	class 'pandas.core.series.Series'
feature_names	list	15	['age', 'workclass', 'fnlwgt', 'educat…
rawdata	DataFrame	(45222, 15)	Column names: age, workclass, fnlwgt, …
target_names	list	2	[0, 1]
target_test	Series	(11306L,)	class 'pandas.core.series.Series'
test	DataFrame	(11306, 14)	Column names: age, workclass, fnlwgt, …
train	DataFrame	(33916, 14)	Column names: age, workclass, fnlwgt, …
train_target	Series	(33916L,)	class 'pandas.core.series.Series'



Index	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hou
0	25	2	226802	1	7	1	6	3	2	1	0	0	40
1	38	2	89814	11	9	0	4	0	4	1	0	0	50
2	28	1	336951	7	12	0	10	0	4	1	0	0	40
3	44	2	160323	15	10	0	6	0	2	1	7688	0	40
5	34	2	198693	0	6	1	7	1	4	1	0	0	30
7	63	4	104626	14	15	0	9	0	4	1	3103	0	32
8	24	2	369667	15	10	1	7	4	4	0	0	0	40
9	55	2	104996	5	4	0	2	0	4	1	0	0	10
10	65	2	184454	11	9	0	6	0	4	1	6418	0	40
11	36	0	212465	9	13	0	0	0	4	1	0	0	40
12	26	2	82091	11	9	1	0	1	4	0	0	0	39
14	48	2	279724	11	9	0	6	0	4	1	3103	0	48
15	43	2	346189	12	14	0	3	0	4	1	0	0	50
16	20	5	444554	15	10	1	7	3	4	1	0	0	25
17	43	2	128354	11	9	0	0	5	4	0	0	0	30
18	37	2	60548	11	9	1	6	4	4	0	0	0	20
20	34	2	107914	9	13	0	12	0	4	1	0	0	47
21	34	2	238588	15	10	1	7	3	2	0	0	0	35
23	25	2	220931	9	13	1	9	1	4	1	0	0	43
24	25	2	205947	9	13	0	9	0	4	1	0	0	40
25	45	4	432824	11	9	0	2	0	4	1	7298	0	90

4.4.9. Machine Learning

4.4.9.1. Logistic Regression

IPython console

Console 1/A

```
------Correlation Table------------
               abs_corr  correlation
parameter
marital-status  0.437678    -0.437678
educational-num 0.332800     0.332800
relationship    0.253402    -0.253402
age             0.237040     0.237040
hours-per-week  0.227199     0.227199
capital-gain    0.221034     0.221034
gender          0.215760     0.215760
capital-loss    0.148687     0.148687
education       0.081196     0.081196
race            0.070844     0.070844
occupation      0.049787     0.049787
native-country  0.020103     0.020103
workclass       0.015659     0.015659
fnlwgt          0.007264    -0.007264
```

4.4.9.2. Logistic Regression training data

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 income   No. Observations:                33916
Model:                          Logit   Df Residuals:                    33902
Method:                           MLE   Df Model:                           13
Date:                Fri, 08 Sep 2017   Pseudo R-squ.:                  0.3415
Time:                        15:51:40   Log-Likelihood:                -12520.
converged:                       True   LL-Null:                       -19011.
                                        LLR p-value:                     0.000
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
age                 0.0075      0.001      6.019      0.000       0.005       0.010
workclass          -0.2455      0.016    -15.273      0.000      -0.277      -0.214
fnlwgt          -8.336e-07    1.5e-07     -5.557      0.000   -1.13e-06    -5.4e-07
education          -0.0351      0.005     -7.445      0.000      -0.044      -0.026
educational-num     0.2708      0.007     40.927      0.000       0.258       0.284
marital-status     -2.6510      0.045    -59.380      0.000      -2.738      -2.563
occupation         -0.0130      0.004     -3.303      0.001      -0.021      -0.005
relationship       -0.1831      0.015    -12.451      0.000      -0.212      -0.154
race               -0.1744      0.018     -9.503      0.000      -0.210      -0.138
gender             -0.6286      0.060    -10.518      0.000      -0.746      -0.511
capital-gain        0.0003   9.48e-06     34.419      0.000       0.000       0.000
capital-loss        0.0007   3.41e-05     21.063      0.000       0.001       0.001
hours-per-week      0.0112      0.001      8.345      0.000       0.009       0.014
native-country     -0.0439      0.002    -19.580      0.000      -0.048      -0.039
===================================================================================
Model Evaluation Statistics Accuracy - Area under the curve AUC.
Training :
Accuraccy score: 82.93%
ROC AUC score: 73.69%
test set result

Optimization terminated successfully.
         Current function value: 0.372052
         Iterations 8
```

4.4.9.3. Logistic Regression testing data

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 income   No. Observations:                11306
Model:                          Logit   Df Residuals:                    11292
Method:                           MLE   Df Model:                           13
Date:                Fri, 08 Sep 2017   Pseudo R-squ.:                  0.3334
Time:                        15:51:40   Log-Likelihood:                -4206.4
converged:                       True   LL-Null:                       -6310.6
                                        LLR p-value:                     0.000
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
age                 0.0070      0.002      3.265      0.001       0.003       0.011
workclass          -0.2479      0.028     -8.800      0.000      -0.303      -0.193
fnlwgt          -7.788e-07   2.63e-07     -2.960      0.003   -1.29e-06   -2.63e-07
education          -0.0061      0.008     -0.754      0.451      -0.022       0.010
educational-num     0.2394      0.011     21.075      0.000       0.217       0.262
marital-status     -2.7215      0.078    -34.851      0.000      -2.875      -2.568
occupation         -0.0110      0.007     -1.637      0.102      -0.024       0.002
relationship       -0.1737      0.025     -6.879      0.000      -0.223      -0.124
race               -0.1755      0.033     -5.388      0.000      -0.239      -0.112
gender             -0.5295      0.103     -5.121      0.000      -0.732      -0.327
capital-gain        0.0003   1.65e-05     19.910      0.000       0.000       0.000
capital-loss        0.0007   5.83e-05     11.385      0.000       0.001       0.001
hours-per-week      0.0092      0.002      3.840      0.000       0.005       0.014
native-country     -0.0425      0.004    -10.837      0.000      -0.050      -0.035
===================================================================================
Model Evaluation Statistics Accuracy - Area under the curve AUC.
Testing :
Accuraccy score: 82.36%
ROC AUC score: 72.05%
```

4.4.9.4. Model evaluation statistics logistic regression

```
Model Evaluation Statistics Accuracy - Area under the curve AUC.
Testing :
Accuraccy score: 82.36%
ROC AUC score: 72.05%

In [16]: result_log_reg_sklearn = log_regre_sklearn()

Model Evaluation Statistics Accuracy - Area under the curve AUC.
Training :
Accuraccy score: 79.5%
ROC AUC score: 62.17%

Model Evaluation Statistics Accuracy - Area under the curve AUC.
Testing :
Accuraccy score: 78.47%
ROC AUC score: 61.22%
```

4.4.9.5. Decision Tree Classifier

```
In [17]: result_decision_tree_clasif =
decision_tree_classifier(datased_prepared_encoded.raw
data, datased_prepared_encoded.data_target)
Accuraccy score: 85.13%
ROC AUC score: 81.54%
```

4.4.9.6. Decision Tree Regression

```
In [18]: result_decision_tree_regre =
decision_tree_regre(datased_prepared_encoded.rawdata,
datased_prepared_encoded.data_target)
Accuraccy score: 84.2%
ROC AUC score: 79.51%
```

5. Conclusion

5.1. Data Cleansing and Preparation

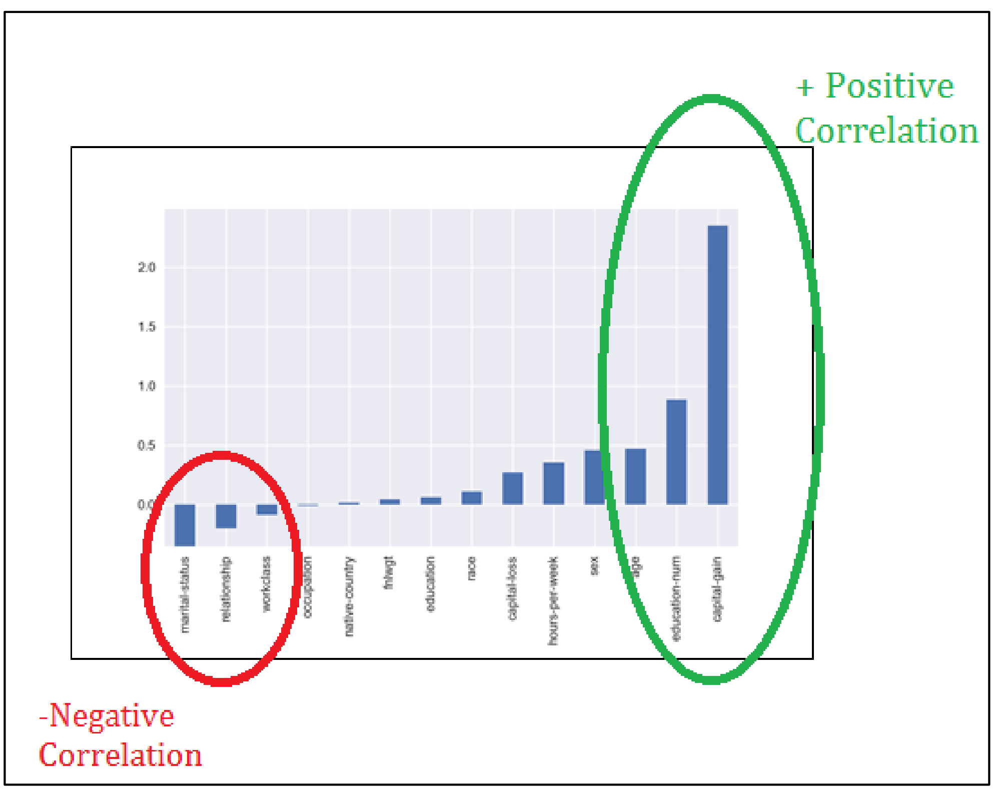
The project consisted of an initial stage of data preparation, this stage was the most time-consumer for whole the project, approximately 70% of the total time.

Categorical features like work-class, education, marital-status, occupation, relationship, race, sex and native-country were encoded (made them numeric) due to the fact that is necessary to Machine Learning processes. It was necessary to discard some information (rows), due to some of the columns like work-class, native-country and occupation had missing values.

5.2. Data Visualization

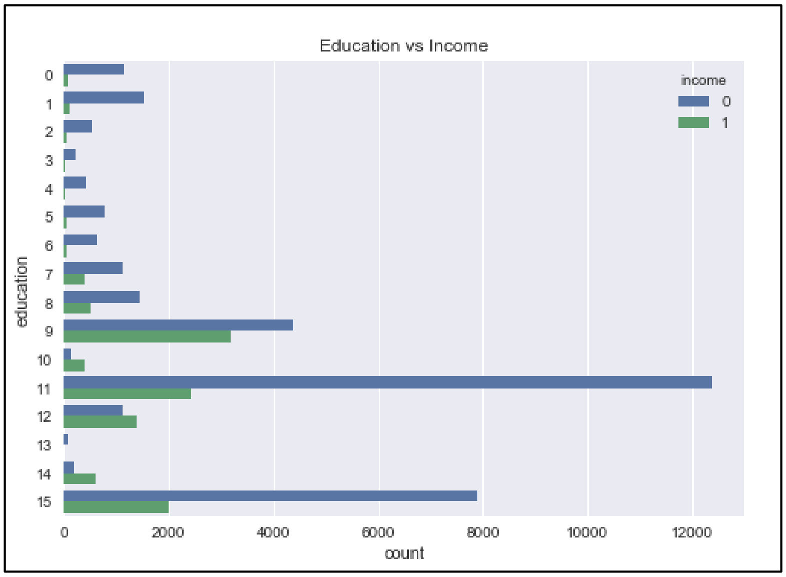
The correlation results allow us to choose which features we should use to implement our machine learning algorithms.

Among the correlation results, the features which have a strong positive correlation to income are: capital gain, education, age and sex. On the other hand, the features which has a negative correlation to income are: marital-statues, relationship, work class

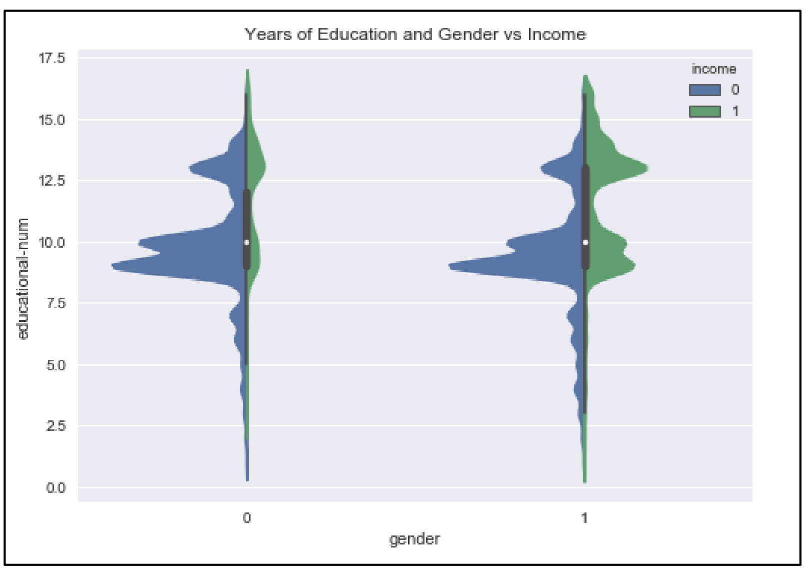


- People with people having higher education like master, doctorate or prof-school are likely to gain more than 50K.

- Top level occupations like executive managerial have highest income ratio, while handlers-cleaners and service employees get lower incomes.

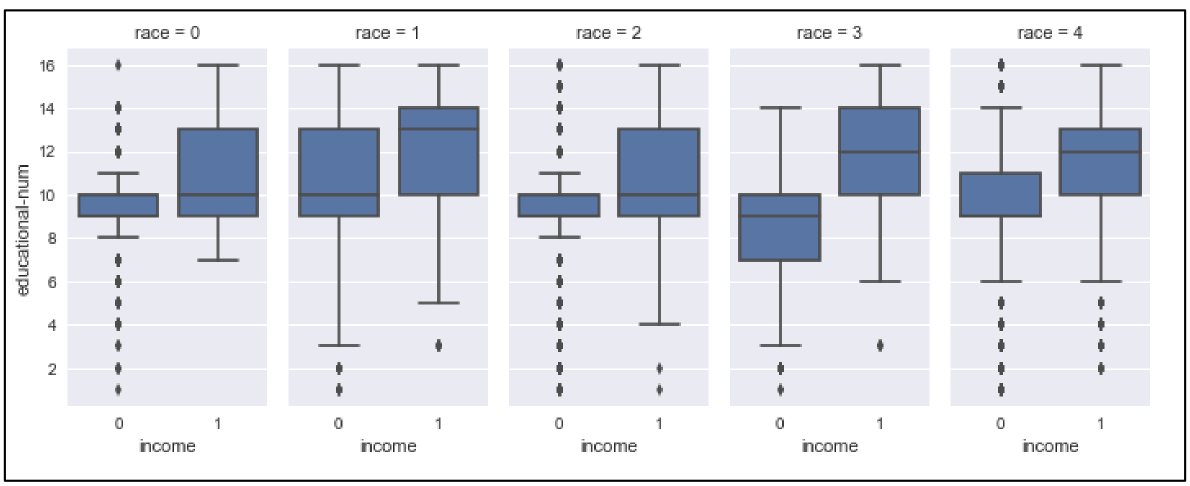


- More education does not result in the same gains in income for women (0) compared to men (1). High income (Green)



- High-income (green) rate is reached at Middle Ages for both women (0) and men (1).

- For some races, such as; Asian Americans/Pacific Islanders, Blacks and Native Americans, more education does not result in the same gains in income compared to Whites.



5.3. Machine Learning algorithms

- I implemented four machine learning algorithms to predict the income label > 50k or <50 k

- To check the accuracy of our models we used two methods: Classification Accuracy (ACC) and Area under ROC Curve (AUC). Our average prediction percentage was 77%.

Model	***Validation***	***%***
result_decision_tree_clasif	ACC_test	85%
result_decision_tree_regre	AUC_test	84%
result_log_reg_sm	ACC_test	82%
result_decision_tree_clasif	AUC_test	82%
result_log_reg_sm	AUC_test	72%
result_decision_tree_regre	ACC_test	80%
result_log_reg_sklearn	ACC_test	78%
result_log_reg_sklearn	AUC_test	61%
	Average	77%

Among Classification Accuracy validation method (ACC), the Model which gave us a higher percentage is Decision tree classifiers

Model	***Validation***	***%***
result_decision_tree_clasif	ACC_test	85%

Among Area under ROC Curve validation method (AUC), the Model who gave us a higher percentage is **Decision tree regression**

Model	***Validation***	***%***
result_decision_tree_regre	AUC_test	84%

6. Appendices

6.1. Functions importing process

6.1.1. Data Preparation

IPython console

Console 1/A

```
In [2]: filename = 'C:/Users/PC/PycharmProjects/PythonProgramers/adult.csv'
   ...:
   ...: def load_csv(filename):
   ...:     data = pd.read_csv(filename, sep="\s*,", engine='python')
   ...:     return Bunch(
   ...:         rawdata = data.copy())
   ...:
   ...:
   ...: def checking_data(df):
   ...:
   ...:     print (df.head())   # Printing how data was read it
   ...:     print '{} {}'.format("\nAdult dataframe dimmensions are : ", df.shape) #will print 48842 Rows 15 Columns
   ...:
   ...:     #Will print number unique values per column df checking data
   ...:     print '{}'.format("\nWe will check for each column, number of observations : \n")
   ...:     for i in df:
   ...:         print '{} {}'.format("\nColumn name : ", i)
   ...:         print df[i].value_counts()
   ...:
   ...:     # Data Cleansing
   ...:     col_names_adultdf = df.columns
   ...:     num_data_adultdf = df.shape[0]
   ...:
   ...:     #The dataframe contains some values represented with a ? character
   ...:     #This loop will check each who has character "?"
   ...:     print '{}'.format("\nThe following columns contains ? symbol as observations : \n")
   ...:
   ...:     columns_With_NA = []
   ...:     for i in col_names_adultdf:
   ...:         num_NA_Values = df[i].isin(["?"]).sum()
   ...:         if num_NA_Values > 0:
   ...:             columns_With_NA.append(i)
   ...:             print '{} {}'.format("\nColumn name: ", i)  #will pring each column name which contains missing values
   ...:             print '{} {}'.format("Number of ? Values : ", num_NA_Values)
   ...:             print ("{0:.2f}%".format(float(num_NA_Values) / num_data_adultdf * 100))
   ...:
```

IPython console

Console 1/A

```
   ...:
   ...: def clean_data(df):
   ...:     # We are getting some categorical we can make them meaningful,
   ...:     # We will replace this categories 6 categories into two married or not married
   ...:     df.replace(['Divorced', 'Married-AF-spouse',
   ...:                  'Married-civ-spouse', 'Married-spouse-absent',
   ...:                  'Never-married','Separated','Widowed'],
   ...:                 ['not married','married','married','married',
   ...:                  'not married','not married','not married'], inplace = True)
   ...:
   ...:     col_names_adultdf = df.columns
   ...:
   ...:     columns_With_NA = []
   ...:     for i in col_names_adultdf:
   ...:         num_NA_Values = df[i].isin(["?"]).sum()
   ...:         if num_NA_Values > 0:
   ...:             columns_With_NA.append(i)
   ...:
   ...:     # Deleting data with ? missing information
   ...:     for i in columns_With_NA:
   ...:         df = df[df[i] != "?"]
   ...:
   ...:     # Printing comparison between old - and new dataframe dimensions
   ...:     #print '{} {}'.format("Adult dataframe original dimmensions were: ", df_copy.shape) #will print 48842 Rows 15 Columns
   ...:     print '{} {}'.format("The new Adult dataframe dimmensions are : ", df.shape) #will print 48842 Rows 15 Columns
   ...:
   ...:     return Bunch(
   ...:     newdata = df.copy())
   ...:
   ...:
   ...: def visualize_data(data):
   ...:     #encoded_data, _ = number_encode_features(data)
   ...:     sns.heatmap(data.corr(), square=True)
   ...:     plt.show()
   ...:
   ...:     sns.countplot(y='occupation', hue='income', data=data, )
   ...:     sns.plt.title('Occupation vs Income')
```

IPython console

Console 1/A

```
...:     sns.plt.show()
...:
...:     sns.countplot(y='education', hue='income', data=data, )
...:     sns.plt.title('Education vs Income')
...:     sns.plt.show()
...:
...:     # How years of education correlate to income, disaggregated by race.
...:     # More education does not result in the same gains in income
...:     # for Asian Americans/Pacific Islanders and Native Americans compared to Caucasians.
...:     g = sns.FacetGrid(data, col='race', size=4, aspect=.5)
...:     g = g.map(sns.boxplot, 'income', 'educational-num')
...:     #sns.plt.title('Years of Education vs Income, disaggregated by race')
...:     sns.plt.show()
...:
...:     # How years of education correlate to income, disaggregated by sex.
...:     # More education also does not result in the same gains in income for women compared to men.
...:     g = sns.FacetGrid(data, col='gender', size=4, aspect=.5)
...:     g = g.map(sns.boxplot, 'income', 'educational-num')
...:     #sns.plt.title('Years of Education vs Income, disaggregated by sex')
...:     sns.plt.show()
...:
...:     # How age correlates to income, disaggregated by race.
...:     # Generally older people make more, except for Asian Americans/Pacific Islanders.
...:     g = sns.FacetGrid(data, col='race', size=4, aspect=.5)
...:     g = g.map(sns.boxplot, 'income', 'age')
...:     #sns.plt.title('Age vs Income, disaggregated by race')
...:     sns.plt.show()
...:
...:     # How hours worked per week correlates to income, disaggregated by marital status.
...:     g = sns.FacetGrid(data, col='marital-status', size=4, aspect=.5)
...:     g = g.map(sns.boxplot, 'income', 'hours-per-week')
...:     #sns.plt.title('Hours by week vs Income, disaggregated by marital status')
...:     sns.plt.show()
...:
...:     sns.violinplot(x='gender', y='educational-num', hue='income', data=data, split=True, scale='count')
...:     sns.plt.title('Years of Education and Gender vs Income')
...:     sns.plt.show()
```

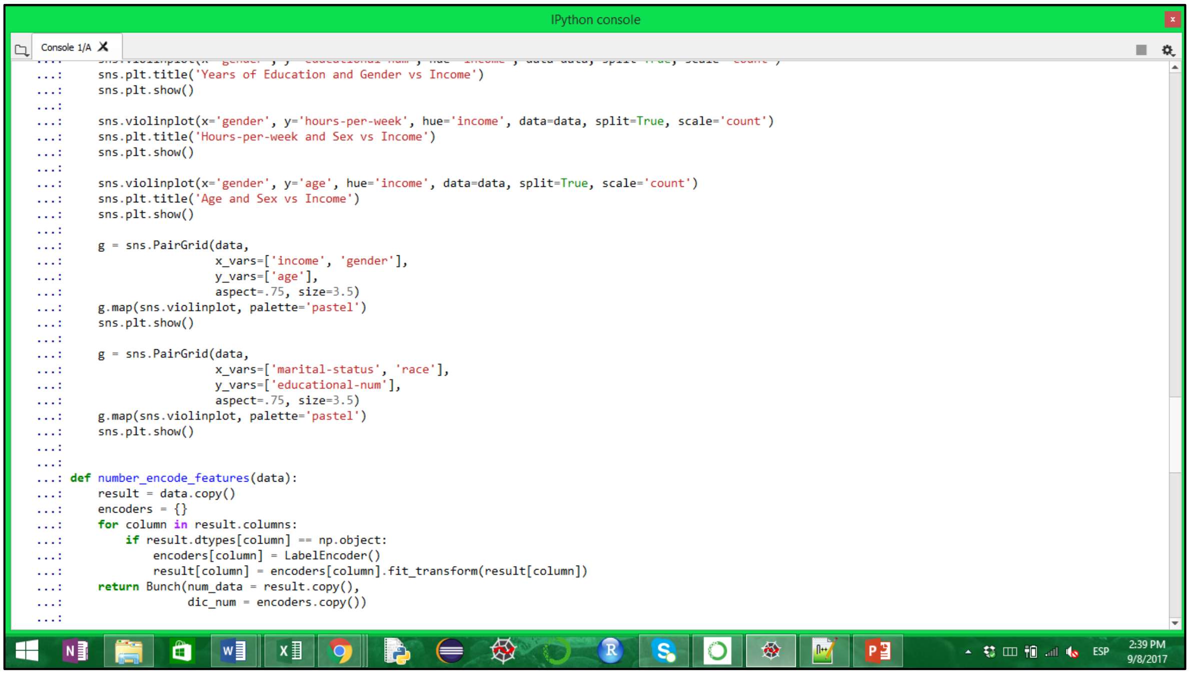
ESP 2:38 PM 9/8/2017

IPython console

Console 1/A

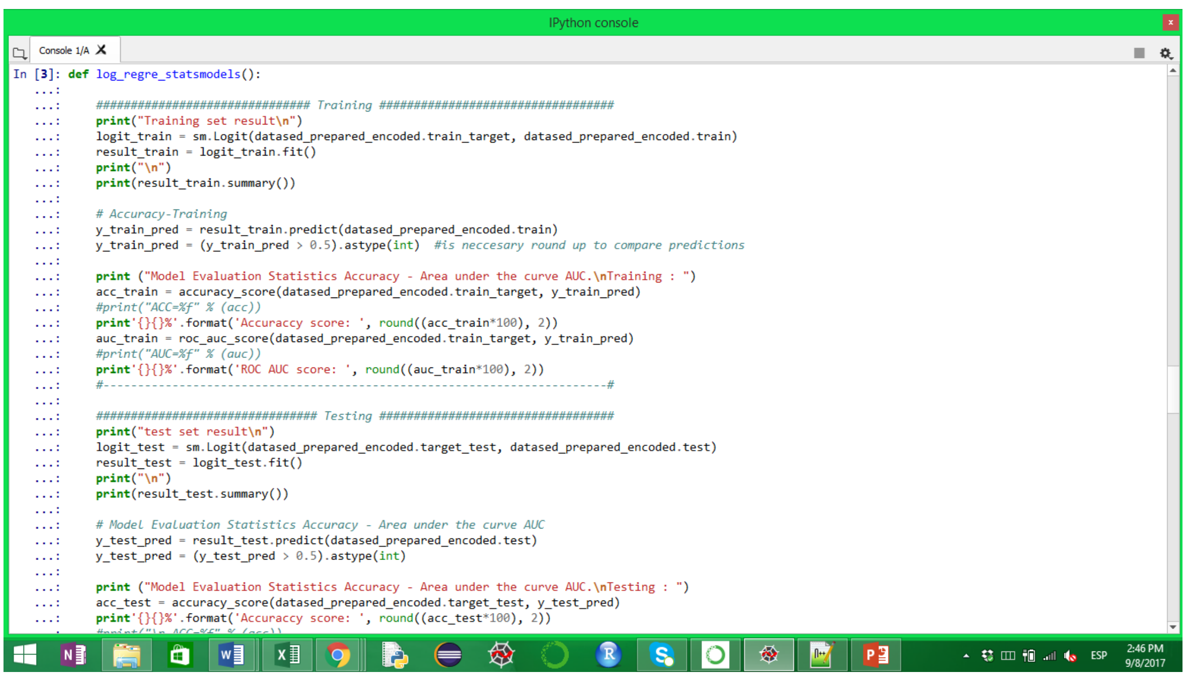
```
...:     sns.violinplot(x='gender', y='educational-num', hue='income', data=data, split=True, scale='count')
...:     sns.plt.title('Years of Education and Gender vs Income')
...:     sns.plt.show()
...:
...:     sns.violinplot(x='gender', y='hours-per-week', hue='income', data=data, split=True, scale='count')
...:     sns.plt.title('Hours-per-week and Sex vs Income')
...:     sns.plt.show()
...:
...:     sns.violinplot(x='gender', y='age', hue='income', data=data, split=True, scale='count')
...:     sns.plt.title('Age and Sex vs Income')
...:     sns.plt.show()
...:
...:     g = sns.PairGrid(data,
...:                      x_vars=['income', 'gender'],
...:                      y_vars=['age'],
...:                      aspect=.75, size=3.5)
...:     g.map(sns.violinplot, palette='pastel')
...:     sns.plt.show()
...:
...:     g = sns.PairGrid(data,
...:                      x_vars=['marital-status', 'race'],
...:                      y_vars=['educational-num'],
...:                      aspect=.75, size=3.5)
...:     g.map(sns.violinplot, palette='pastel')
...:     sns.plt.show()
...:
...:
...: def number_encode_features(data):
...:     result = data.copy()
...:     encoders = {}
...:     for column in result.columns:
...:         if result.dtypes[column] == np.object:
...:             encoders[column] = LabelEncoder()
...:             result[column] = encoders[column].fit_transform(result[column])
...:     return Bunch(num_data = result.copy(),
...:                  dic_num = encoders.copy())
...:
```

ESP 2:39 PM 9/8/2017



```
IPython console
Console 1/A
   ...:     sns.violinplot(x='gender', y='educational-num', hue='income', data=data, split=True, scale='count')
   ...:     sns.plt.title('Years of Education and Gender vs Income')
   ...:     sns.plt.show()
   ...:
   ...:     sns.violinplot(x='gender', y='hours-per-week', hue='income', data=data, split=True, scale='count')
   ...:     sns.plt.title('Hours-per-week and Sex vs Income')
   ...:     sns.plt.show()
   ...:
   ...:     sns.violinplot(x='gender', y='age', hue='income', data=data, split=True, scale='count')
   ...:     sns.plt.title('Age and Sex vs Income')
   ...:     sns.plt.show()
   ...:
   ...:     g = sns.PairGrid(data,
   ...:                      x_vars=['income', 'gender'],
   ...:                      y_vars=['age'],
   ...:                      aspect=.75, size=3.5)
   ...:     g.map(sns.violinplot, palette='pastel')
   ...:     sns.plt.show()
   ...:
   ...:     g = sns.PairGrid(data,
   ...:                      x_vars=['marital-status', 'race'],
   ...:                      y_vars=['educational-num'],
   ...:                      aspect=.75, size=3.5)
   ...:     g.map(sns.violinplot, palette='pastel')
   ...:     sns.plt.show()
   ...:
   ...:
   ...: def number_encode_features(data):
   ...:     result = data.copy()
   ...:     encoders = {}
   ...:     for column in result.columns:
   ...:         if result.dtypes[column] == np.object:
   ...:             encoders[column] = LabelEncoder()
   ...:             result[column] = encoders[column].fit_transform(result[column])
   ...:     return Bunch(num_data = result.copy(),
   ...:                  dic_num = encoders.copy())
   ...:
```

6.1.2. Machine learning



```
IPython console
Console 1/A
In [3]: def log_regre_statsmodels():
   ...:
   ...:     ############################## Training ##################################
   ...:     print("Training set result\n")
   ...:     logit_train = sm.Logit(datased_prepared_encoded.train_target, datased_prepared_encoded.train)
   ...:     result_train = logit_train.fit()
   ...:     print("\n")
   ...:     print(result_train.summary())
   ...:
   ...:     # Accuracy-Training
   ...:     y_train_pred = result_train.predict(datased_prepared_encoded.train)
   ...:     y_train_pred = (y_train_pred > 0.5).astype(int)  #is neccesary round up to compare predictions
   ...:
   ...:     print ("Model Evaluation Statistics Accuracy - Area under the curve AUC.\nTraining : ")
   ...:     acc_train = accuracy_score(datased_prepared_encoded.train_target, y_train_pred)
   ...:     #print("ACC=%f" % (acc))
   ...:     print'{}{}%'.format('Accuraccy score: ', round((acc_train*100), 2))
   ...:     auc_train = roc_auc_score(datased_prepared_encoded.train_target, y_train_pred)
   ...:     #print("AUC=%f" % (auc))
   ...:     print'{}{}%'.format('ROC AUC score: ', round((auc_train*100), 2))
   ...:     #----------------------------------------------------------------------------#
   ...:
   ...:     ############################### Testing ##################################
   ...:     print("test set result\n")
   ...:     logit_test = sm.Logit(datased_prepared_encoded.target_test, datased_prepared_encoded.test)
   ...:     result_test = logit_test.fit()
   ...:     print("\n")
   ...:     print(result_test.summary())
   ...:
   ...:     # Model Evaluation Statistics Accuracy - Area under the curve AUC
   ...:     y_test_pred = result_test.predict(datased_prepared_encoded.test)
   ...:     y_test_pred = (y_test_pred > 0.5).astype(int)
   ...:
   ...:     print ("Model Evaluation Statistics Accuracy - Area under the curve AUC.\nTesting : ")
   ...:     acc_test = accuracy_score(datased_prepared_encoded.target_test, y_test_pred)
   ...:     print'{}{}%'.format('Accuraccy score: ', round((acc_test*100), 2))
   ...:     #print("\n ACC=%f" % (acc))
```

IPython console

Console 1/A

```
...:     #print("\n ACC=%f" % (acc))
...:     auc_test = roc_auc_score(datased_prepared_encoded.target_test, y_test_pred)
...:     print'{}{}%'.format('ROC AUC score: ', round((auc_test*100), 2))
...:     #print("\n AUC=%f" % (auc))
...:
...:     return Bunch(train_results = y_train_pred.copy(),
...:                  ACC_train = acc_train.copy(),
...:                  AUC_train = auc_train.copy(),
...:                  test_results = y_test_pred.copy(),
...:                  ACC_test = acc_test.copy(),
...:                  AUC_test = auc_test.copy()
...:                  )
...:
...:
...: #==============================================================================#
...: #  2. LOGISTIC REGRESSION using sklearn.linear_model
...: #------------------------------------------------------------------------------#
...: def log_regre_sklearn():
...:     logr = LogisticRegression()
...:     logr.fit( datased_prepared_encoded.train, datased_prepared_encoded.train_target )
...:     results_train = logr.predict(datased_prepared_encoded.train)
...:
...:     print ("\nModel Evaluation Statistics Accuracy - Area under the curve AUC.\nTraining : ")
...:     acc_train = accuracy_score(datased_prepared_encoded.train_target, results_train)
...:     print'{}{}%'.format('Accuraccy score: ', round((acc_train*100), 2))
...:     auc_train = roc_auc_score(datased_prepared_encoded.train_target, results_train)
...:     print'{}{}%'.format('ROC AUC score: ', round((auc_train*100), 2))
...:
...:     logr.fit( datased_prepared_encoded.test, datased_prepared_encoded.target_test )
...:     results_test = logr.predict(datased_prepared_encoded.test)
...:     #results_test = (results_test > 0.5).astype(int)
...:
...:     print ("\nModel Evaluation Statistics Accuracy - Area under the curve AUC.\nTesting : ")
...:     acc_test = accuracy_score(datased_prepared_encoded.target_test, results_test)
...:     print'{}{}%'.format('Accuraccy score: ', round((acc_test*100), 2))
...:     auc_test = roc_auc_score(datased_prepared_encoded.target_test, results_test)
```

IPython console

Console 1/A

```
...:     print'{}{}%'.format('ROC AUC score: ', round((auc_test*100), 2))
...:
...:     return Bunch(train_results = results_train.copy(),
...:                  ACC_train = acc_train.copy(),
...:                  AUC_train = auc_train.copy(),
...:                  test_results = results_test.copy(),
...:                  ACC_test = acc_test.copy(),
...:                  AUC_test = auc_test.copy()
...:                  )
...:
...:
...: #==============================================================================#
...: #  2. DECISION TREE CLASSIFIER using sklearn
...: #------------------------------------------------------------------------------#
...: def decision_tree_classifier(data,target):
...:     predictors = ['age','workclass','education','educational-num',
...:                   'marital-status', 'occupation','relationship','race','gender',
...:                   'capital-gain','capital-loss','hours-per-week', 'native-country']
...:
...:     tree_count = 10
...:     bag_proportion = 0.6
...:     predictions = []
...:
...:     sss = StratifiedShuffleSplit(target, 1, test_size=0.25, random_state=1)
...:     for train_index, test_index in sss:
...:         train_data = data.iloc[train_index]
...:         test_data = data.iloc[test_index]
...:         for i in range(tree_count):
...:             bag = train_data.sample(frac=bag_proportion, replace = True, random_state=i)
...:             X_train, X_test = bag[predictors], test_data[predictors]
...:             y_train, y_test = bag["income"], test_data["income"]
...:             clf = DecisionTreeClassifier(random_state=1, min_samples_leaf=75)
...:             clf.fit(X_train, y_train)
...:             predictions.append(clf.predict_proba(X_test)[:,1])
...:
...:     combined = np.sum(predictions, axis=0)/10
```

IPython console

Console 1/A

```
   ...:     rounded = np.round(combined) # we have to round prediction to the ceeling
   ...:
   ...:     acc_test = accuracy_score(rounded, y_test)
   ...:     auc_test = roc_auc_score(rounded, y_test)
   ...:
   ...:     print'{}{}%'.format('Accuraccy score: ', round(accuracy_score(rounded, y_test) * 100 , 2))
   ...:     print'{}{}%'.format('ROC AUC score: ', round(roc_auc_score(rounded, y_test)* 100 , 2))
   ...:
   ...:     return Bunch(predic = predictions,
   ...:                  test_results = rounded,
   ...:                  ACC_test = acc_test.copy(),
   ...:                  AUC_test = auc_test.copy()
   ...:                  )
   ...:
   ...:
   ...:
   ...: #==========================================================================#
   ...: #  2. DECISION TREE REGRESSION using sklearn
   ...: #--------------------------------------------------------------------------#
   ...: def decision_tree_regre(data,target):
   ...:     predictors = ['age','workclass','education','educational-num',
   ...:               'marital-status', 'occupation','relationship','race','gender',
   ...:               'capital-gain','capital-loss','hours-per-week', 'native-country']
   ...:
   ...:     tree_count = 10
   ...:     bag_proportion = 0.6
   ...:     predictions = []
   ...:
   ...:     sss = StratifiedShuffleSplit(target, 1, test_size=0.25, random_state=1)
   ...:     for train_index, test_index in sss:
   ...:         train_data = data.iloc[train_index]
   ...:         test_data = data.iloc[test_index]
   ...:         for i in range(tree_count):
   ...:             bag = train_data.sample(frac=bag_proportion, replace = True, random_state=i)
   ...:             X_train, X_test = bag[predictors], test_data[predictors]
   ...:             y_train, y_test = bag["income"], test_data["income"]
   ...:             clf = DecisionTreeRegressor()
   ...:             clf.fit(X_train, y_train)
```

ESP 2:48 PM 9/8/2017

IPython console

Console 1/A

```
   ...:             predictions.append(clf.predict(X_test))
   ...:
   ...:     combined = np.sum(predictions, axis=0)/10
   ...:     rounded = np.round(combined) # we have to round prediction to the ceeling
   ...:
   ...:     acc_test = accuracy_score(rounded, y_test)
   ...:     auc_test = roc_auc_score(rounded, y_test)
   ...:
   ...:     print'{}{}%'.format('Accuraccy score: ', round(accuracy_score(rounded, y_test) * 100 , 2))
   ...:     print'{}{}%'.format('ROC AUC score: ', round(roc_auc_score(rounded, y_test)* 100 , 2))
   ...:
   ...:     return Bunch(predic = predictions,
   ...:                  test_results = rounded,
   ...:                  ACC_test = acc_test.copy(),
   ...:                  AUC_test = auc_test.copy()
   ...:                  )

In [4]:
```

ESP 2:48 PM 9/8/2017