Using KNN to do Predictions in the Adult Dataset

PMR3508: Machine Learning and Pattern Recognation

Student name: Rodolfo Lima (PMR3508-2021-6)

0- Libraries

In [31]:
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))

import ssl
ssl._create_default_https_context = ssl._create_unverified_context

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sea

#Sklearn functions
from sklearn.preprocessing import RobustScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
```

1- Data Cleaning

In [93]:
```
#Training data
df_train = pd.read_csv('train_data.csv')
df_train = df_train.set_index('Id')

#testing Data
df_test = pd.read_csv("adult.test.txt",
        names=[
        'age', 'workclass', 'fnlwgt', 'education', 'education.num',
       'marital.status', 'occupation', 'relationship', 'race', 'sex',
       'capital.gain', 'capital.loss', 'hours.per.week', 'native.country',
       'income'],
        sep=r'\s*,\s*',
        engine='python')
df_test.drop(df_test.index[0], axis = 0, inplace = True)
df_train.head()
```

Out[93]:

Id	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.los
16280	34	Private	204991	Some-college	10	Divorced	Exec-managerial	Own-child	White	Male	0	
16281	58	Local-gov	310085	10th	6	Married-civ-spouse	Transport-moving	Husband	White	Male	0	
16282	25	Private	146117	Some-college	10	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	
16283	24	Private	138938	Some-college	10	Divorced	Adm-clerical	Not-in-family	White	Female	0	
16284	57	Self-emp-inc	258883	HS-grad	9	Married-civ-spouse	Transport-moving	Husband	White	Male	5178	

In [60]:

```python
# First, I've to check the labels of categorical variables and values of quantitative variables in orde
r to assess the integrity of my dataset
features = df_train.columns

for feature in features:
    print('Feature '+feature + ' values:',set(df_train[feature]))
```

```
Feature age values: {17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 6
2, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 8
7, 88, 90}
Feature workclass values: {'Federal-gov', 'Without-pay', '?', 'Private', 'Self-emp-inc', 'Self-emp-no
t-inc', 'State-gov', 'Never-worked', 'Local-gov'}
Feature fnlwgt values: {262153, 262158, 131088, 131091, 393248, 131117, 393264, 262196, 262208, 26223
3, 262241, 262244, 262245, 131178, 131180, 262280, 393354, 393357, 262285, 393360, 131224, 131230, 39
3376, 131239, 131258, 262345, 131275, 262352, 131288, 131298, 131302, 131309, 131310, 262402, 393480,
262409, 262425, 262439, 262446, 131379, 262478, 131414, 131417, 131425, 131435, 262511, 262515, 13145
9, 131461, 131463, 131473, 262552, 262570, 131519, 393673, 262601, 131534, 262617, 131552, 131568, 39
3712, 262642, 393715, 131573, 131584, 393728, 262656, 131588, 131591, 262664, 131608, 131611, 262684,
131615, 262688, 131620, 393768, 131631, 131650, 262744, 262749, 131679, 131681, 131686, 131699, 26277
8, 131714, 262802, 262819, 656036, 262841, 131776, 131777, 262872, 393945, 131808, 262882, 131811, 26
2885, 393965, 131826, 131827, 131852, 131869, 131899, 262978, 131916, 262994, 263000, 263005, 131934,
263015, 263024, 131976, 131982, 131991, 263081, 263094, 132026, 394176, 132053, 132055, 263128, 13205
7, 263130, 132078, 263150, 132125, 263200, 132130, 263224, 132178, 132191, 132204, 132222, 263300, 26
3307, 132243, 263338, 263339, 263340, 132300, 263374, 394447, 132304, 132320, 132326, 132327, 394474,
394484, 132341, 394503, 263431, 263444, 394534, 132393, 132412, 263498, 263502, 394612, 263552, 13248
1, 263561, 263568, 394669, 263600, 132529, 263608, 263612, 263614, 132551, 132563, 132565, 263637, 26
3641, 132572, 394727, 263670, 132601, 132606, 132626, 132633, 132636, 132652, 263728, 263729, 132661,
525878, 263746, 394820, 132680, 132683, 132686, 263773, 132704, 132705, 132716, 394860, 132717, 13272
8, 132737, 132749, 132755, 263831, 263836, 394927, 263855, 263871, 132805, 132806, 263886, 132819, 26
3899, 132832, 263908, 132847, 263925, 263932, 132870, 132874, 395022, 132879, 395026, 132887, 263970,
263982, 132912, 132917, 132925, 132930, 395078, 264012, 526164, 132963, 132972, 132973, 264052, 26405
5, 132986, 132996, 133014, 264095, 133025, 395170, 264102, 133043, 133050, 133055, 133060, 133061, 26
4148, 264166, 657397, 264188, 133122, 133126, 133136, 264210, 133144, 395297, 133164, 133166, 133169,
264244, 133201, 133219, 395368, 264300, 133239, 264314, 133248, 133278, 264357, 133299, 526528, 26439
0, 133327, 133328, 133336, 133352, 133355, 133359, 264436, 395512, 133373, 264457, 133403, 133420, 39
5567, 133425, 264498, 264503, 133436, 133449, 133454, 264526, 133461, 133471, 264544, 133495, 133503,
133515, 526734, 133520, 264593, 264600, 133536, 264627, 133569, 264651, 133582, 133584, 133586, 26466
3, 395736, 133602, 133616, 133625, 264710, 133654, 133655, 264738, 264740, 133669, 395831, 133692, 13
3694, 133696, 133729, 133736, 526968, 133756, 133758, 264834, 133766, 133770, 264864, 264874, 133819,
133821, 133833, 133846, 264924, 133853, 133861, 264936, 264939, 133876, 264961, 264968, 133902, 13390
6, 133917, 133929, 133935, 133937, 133938, 527162, 396099, 133963, 265038, 133969, 396116, 133974, 13
3983, 133985, 134004, 265077, 265083, 265086, 265097, 134026, 265099, 265105, 265116, 134048, 134069,
265148, 134088, 134113, 134120, 265192, 396270, 265201, 134130, 134152, 134153, 265230, 134181, 13419
2, 265266, 134195, 265275, 134220, 265295, 265314, 265356, 134286, 134287, 396467, 134331, 396482, 26
5434, 134367, 134378, 396538, 265477, 265509, 134444, 134447, 396595, 265535, 134475, 134480, 265554,
396633, 265567, 265576, 134509, 134524, 265618, 265628, 134566, 265638, 396722, 265657, 265661, 26566
2, 396745, 134613, 396758, 265698, 265706, 396790, 265737, 134671, 134699, 134727, 396878, 265807, 13
4737, 134746, 396895, 134756, 134766, 134768, 134782, 134787, 134793, 134797, 134808, 265881, 134813,
134821, 265917, 134854, 265954, 134886, 134888, 134890, 134906, 134935, 266015, 134945, 266022, 13496
0, 266037, 266043, 134974, 659273, 134997, 266070, 266072, 135001, 266080, 266081, 266083, 266084, 26
6091, 135020, 135028, 135039, 135044, 266119, 266126, 135056, 266134, 266135, 266138, 266150, 135102,
135105, 266189, 135134, 397280, 135138, 135162, 397307, 397317, 266255, 135190, 397346, 266275, 26628
1, 266287, 659504, 266324, 266325, 266337, 659558, 266343, 135285, 135289, 135293, 135296, 135308, 13
5312, 135315, 397466, 397467, 266400, 135339, 135342, 266439, 135388, 266461, 266463, 266467, 528616,
135416, 266489, 135436, 266510, 135439, 266525, 266529, 266530, 135458, 135465, 135470, 135480, 13550
0, 135520, 135525, 266598, 266624, 266635, 135566, 135567, 135568, 266639, 266645, 266668, 135601, 13
5603, 135606, 135607, 266681, 135617, 266707, 266710, 135643, 135645, 135647, 397831, 266764, 135716,
266792, 791084, 266803, 397877, 266820, 135776, 266855, 135785, 135786, 266860, 135796, 135803, 39796
2, 397963, 135839, 266912, 135840, 135845, 266926, 398001, 266945, 135874, 398019, 529104, 266973, 39
8067, 135924, 267012, 267034, 267040, 267044, 529216, 529223, 267085, 267086, 136017, 136028, 267101,
267107, 1184622, 136063, 267138, 398212, 267147, 136075, 136077, 398220, 136081, 267161, 136094, 2671
74, 136107, 136121, 267198, 267205, 136137, 136162, 660461, 136177, 267252, 136198, 136204, 267284, 1
36218, 136224, 136226, 136244, 267325, 136262, 136277, 267352, 136306, 136309, 136310, 136314, 13632
2, 267396, 267399, 398473, 136331, 267404, 267408, 136343, 267426, 267431, 136363, 136398, 267471, 13
6413, 136419, 136448, 267521, 136450, 136455, 267540, 136472, 267546, 136480, 398625, 398626, 267556,
398652, 267581, 267588, 398662, 136546, 267652, 660870, 267661, 267663, 136610, 136629, 267706, 26771
7, 136651, 267736, 398827, 136684, 136687, 267763, 398843, 267776, 267790, 136721, 267798, 398904, 13
6767, 267843, 398918, 267859, 398931, 136787, 267866, 136819, 267891, 267893, 136823, 136824, 136836,
267912, 136841, 398986, 398988, 136862, 136866, 136873, 267945, 399020, 136878, 399022, 267952, 26795
5, 530099, 267965, 267966, 267967, 399052, 136913, 267989, 136935, 136939, 399087, 399088, 268022, 13
6951, 136975, 268051, 136982, 136985, 136986, 136996, 136997, 399155, 268083, 137018, 268090, 268098,
137042, 268127, 137059, 137063, 137065, 268145, 268147, 137076, 137088, 399246, 268183, 137126, 26821
3, 137142, 268222, 268234, 268252, 137184, 137192, 268276, 137205, 268281, 137214, 268292, 137223, 13
7225, 137229, 137232, 530454, 399386, 399387, 137250, 137253, 268358, 137290, 137296, 137300, 137301,
137304, 399449, 137310, 137314, 268392, 137354, 137363, 137367, 399522, 268451, 137390, 268482, 13742
1, 137428, 268514, 137444, 268524, 268525, 268545, 137476, 268553, 268571, 137499, 268575, 137506, 13
7510, 137522, 268598, 137527, 137537, 137547, 268620, 137554, 268639, 137578, 268661, 137591, 137604,
137606, 137613, 137616, 137618, 268700, 137629, 137645, 137646, 137651, 268726, 137658, 137674, 13767
8, 137681, 137691, 137698, 137707, 137722, 268797, 268804, 137733, 137753, 268831, 399904, 268832, 26
8840, 268861, 137795, 137814, 137815, 531055, 137843, 400004, 137862, 137875, 137876, 268952, 268954,
268965, 137895, 137898, 137900, 137907, 400061, 137917, 137940, 269015, 137951, 137952, 137953, 26902
8, 269042, 269045, 137974, 269060, 400132, 137994, 269068, 269080, 138022, 269095, 138026, 138054, 13
8069, 138077, 400225, 269168, 138107, 269182, 269186, 400285, 138142, 138145, 269221, 138152, 138153,
138157, 138162, 269243, 269246, 138179, 138184, 138190, 138192, 400356, 269284, 269300, 269318, 26932
3, 138251, 138253, 269329, 138269, 138270, 400416, 269354, 138283, 138285, 138293, 400443, 138332, 13
8342, 269417, 138352, 138358, 138370, 269444, 269455, 400535, 269474, 138416, 269499, 269512, 138441,
269527, 400616, 400630, 400635, 138497, 138502, 138504, 138507, 269583, 138513, 138514, 269604, 13853
7, 138542, 138557, 138575, 269652, 269654, 269657, 269665, 138594, 138597, 138611, 269687, 138621, 13
8626, 138634, 269708, 269722, 269723, 269733, 138662, 138667, 138692, 138705, 138714, 269786, 138744,
138760, 138765, 138768, 400943, 138816, 400966, 138845, 138847, 138852, 138892, 401051, 138907, 13891
7, 269991, 138938, 138940, 138944, 401104, 138962, 138966, 138970, 270043, 401118, 138975, 270056, 27
0059, 401134, 138991, 138992, 138999, 139000, 270075, 139003, 270079, 139012, 270092, 139049, 401198,
139057, 270142, 270147, 270151, 139086, 139098, 663394, 139116, 270194, 139126, 139127, 139145, 27021
8, 270228, 139161, 532379, 139176, 139180, 139183, 139187, 401333, 139190, 401335, 139193, 270276, 27
0278, 139209, 139212, 270324, 270333, 270335, 270339, 139268, 139272, 139290, 270366, 139296, 401451,
270379, 139324, 139338, 139344, 139347, 270421, 139364, 401508, 270436, 401531, 270460, 139391, 27050
2, 270517, 270522, 139452, 139455, 139464, 139466, 270544, 270546, 401623, 270551, 270565, 270572, 27
0577, 139514, 270587, 401690, 139568, 139576, 401723, 139586, 139616, 401762, 270693, 532845, 139647,
270721, 270728, 139660, 139671, 401832, 139703, 139730, 139734, 401886, 139743, 139753, 139770, 27084
2, 401930, 270859, 270872, 270886, 270887, 270889, 139822, 139843, 139850, 401998, 270942, 139883, 13
9889, 139890, 270968, 270973, 139903, 139906, 139907, 270985, 139916, 533147, 271012, 271013, 402089,
139960, 139978, 139989, 139992, 140001, 140011, 140012, 271092, 140027, 140035, 140045, 271118, 27112
2, 664366, 271160, 140092, 140108, 140117, 140121, 402306, 140164, 271243, 140176, 271262, 271274, 27
1276, 140206, 402361, 140219, 402367, 140237, 140242, 402397, 271328, 271346, 271352, 140282, 271354,
140300, 271379, 402462, 140319, 271393, 140359, 271431, 140363, 140365, 271446, 664670, 271466, 40253
9, 140399, 140414, 140426, 271521, 140456, 140459, 140462, 795830, 271544, 140474, 140477, 140478, 27
1550, 271567, 271571, 271572, 271579, 140516, 271603, 664821, 140544, 140558, 140559, 140564, 140569,
140571, 140576, 140581, 140583, 140590, 140592, 271665, 402778, 271710, 271714, 140644, 140664, 40281
2, 140673, 271749, 271753, 271767, 140711, 140713, 271792, 271795, 140729, 271807, 140741, 140752, 27
1828, 140764, 271837, 140782, 140790, 140798, 271901, 140830, 402975, 140845, 140849, 140851, 140852,
140854, 402998, 271933, 271935, 140863, 271936, 140869, 140886, 271962, 403037, 140915, 403072, 27201
9, 140957, 403107, 403112, 140985, 140988, 272063, 272069, 141003, 272090, 141028, 141029, 141040, 40
3187, 141058, 141085, 272165, 272166, 272182, 272185, 141113, 141118, 403276, 272209, 141165, 272240,
272248, 141181, 141185, 141186, 141218, 141221, 141229, 141245, 403391, 141253, 141264, 272338, 27234
3, 141272, 272359, 403433, 141297, 141301, 272376, 141308, 141323, 403467, 403468, 141326, 141327, 27
2411, 403489, 141350, 272425, 272428, 141363, 403519, 272451, 141388, 272471, 272476, 403550, 403552,
141409, 141410, 141418, 141420, 141453, 272531, 141459, 403625, 141481, 141483, 141490, 141511, 40367
1, 141537, 272615, 141545, 272618, 141549, 141550, 272625, 141558, 141570, 141580, 141583, 141584, 27
2656, 141590, 272667, 272669, 272671, 141626, 141637, 403782, 141642, 403788, 141645, 141649, 141657,
141663, 141698, 272778, 141706, 272780, 403860, 403865, 141727, 272800, 141742, 141745, 141758, 40391
0, 403911, 272856, 141795, 141797, 141802, 141807, 272896, 141824, 272902, 272910, 141838, 272913, 14
1841, 272918, 141858, 141868, 272944, 141875, 141876, 141877, 272950, 404023, 272960, 141896, 272986,
404062, 141918, 273010, 404085, 141944, 141957, 273051, 273084, 142022, 142030, 142038, 142061, 14207
3, 142075, 142076, 142081, 273194, 273206, 142158, 273230, 142166, 273241, 142169, 142171, 273243, 14
2182, 273269, 142210, 273285, 273287, 142219, 142227, 142233, 273308, 273324, 142264, 404416, 404421,
142282, 142287, 273362, 404453, 142326, 273399, 273402, 273403, 273425, 273435, 142370, 273454, 14238
3, 404547, 142404, 142411, 142424, 404573, 273514, 142443, 142444, 404599, 404601, 142457, 273536, 14
2470, 404616, 273558, 142494, 273569, 142501, 273575, 142506, 404661, 142519, 142528, 273604, 142535,
273612, 142547, 142555, 273629, 142566, 273640, 142573, 142579, 142581, 142616, 142621, 142624, 27370
1, 142646, 142647, 142657, 273734, 142673, 142675, 142682, 142689, 273771, 142707, 142712, 142714, 14
2717, 142719, 273792, 142723, 404868, 142725, 273803, 273818, 142751, 273828, 142756, 142757, 142760,
535978, 142764, 142766, 142791, 273876, 404951, 142828, 273905, 142835, 404998, 142856, 273929, 14287
1, 142874, 142886, 142897, 142912, 142914, 142922, 142924, 274010, 405083, 142964, 142977, 274057, 14
3003, 405155, 274096, 143030, 274103, 143032, 405177, 143034, 274111, 143046, 143058, 143062, 143068,
143069, 143083, 274158, 143098, 143110, 143118, 143123, 274198, 274200, 143129, 405284, 274222, 14315
2, 274228, 143162, 405309, 143175, 143189, 274276, 405362, 405374, 405386, 143266, 143267, 143280, 14
3281, 274363, 143299, 143312, 274398, 143327, 143331, 143342, 274424, 12285, 143360, 143368, 143372,
274451, 405526, 143385, 143392, 274475, 274502, 405577, 143436, 143437, 143445, 274528, 143459, 14348
2, 143486, 274562, 405644, 274579, 536725, 143516, 143517, 143526, 143533, 143538, 143540, 405684, 14
3542, 143547, 143554, 143570, 405723, 143582, 143583, 274657, 143589, 143604, 274679, 274683, 274689,
274690, 405765, 274720, 405793, 274724, 143653, 274731, 143664, 274746, 143699, 405855, 143716, 27479
7, 274800, 143730, 143731, 143732, 274809, 143741, 274830, 143766, 405913, 143771, 143774, 143791, 14
3800, 143804, 143807, 274883, 143822, 143828, 143833, 274913, 274916, 274917, 143849, 143851, 143857,
143865, 143867, 143868, 274964, 274969, 406051, 143910, 143912, 799281, 143931, 143932, 406078, 14393
9, 143943, 143949, 143953, 275034, 275051, 143995, 144002, 275074, 537222, 275093, 275094, 275095, 66
8319, 144032, 275110, 275116, 144056, 144063, 144064, 144067, 144071, 668362, 144084, 144094, 275181,
144114, 144116, 275190, 144122, 144125, 144133, 144137, 275223, 144154, 275232, 144161, 275236, 14416
5, 144169, 275244, 144172, 144182, 144218, 275291, 144236, 144259, 275338, 275357, 275361, 275364, 27
5369, 144301, 275385, 406463, 144322, 275395, 406468, 275421, 144351, 144354, 144361, 275438, 144371,
```

```
Feature education values: {'Assoc-voc', 'HS-grad', 'Bachelors', '12th', '10th', 'Masters', '7th-8th',
 'Some-college', '1st-4th', 'Preschool', '11th', 'Assoc-acdm', 'Doctorate', '9th', '5th-6th', 'Prof-s
chool'}
Feature education.num values: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
Feature marital.status values: {'Married-spouse-absent', 'Divorced', 'Separated', 'Widowed', 'Married
-AF-spouse', 'Never-married', 'Married-civ-spouse'}
Feature occupation values: {'Machine-op-inspct', 'Handlers-cleaners', 'Craft-repair', 'Priv-house-ser
v', 'Farming-fishing', '?', 'Sales', 'Exec-managerial', 'Tech-support', 'Armed-Forces', 'Protective-s
erv', 'Adm-clerical', 'Prof-specialty', 'Other-service', 'Transport-moving'}
Feature relationship values: {'Husband', 'Own-child', 'Not-in-family', 'Unmarried', 'Wife', 'Other-re
lative'}
Feature race values: {'Asian-Pac-Islander', 'Black', 'Other', 'Amer-Indian-Eskimo', 'White'}
Feature sex values: {'Female', 'Male'}
Feature capital.gain values: {0, 2050, 4101, 7688, 14344, 2062, 2580, 22040, 3103, 1055, 25124, 2597,
 4650, 18481, 2105, 5178, 1086, 3137, 6723, 2635, 4687, 594, 20051, 1111, 5721, 3674, 2653, 1639, 676
7, 114, 2174, 1151, 2176, 7298, 25236, 1173, 2202, 99999, 9386, 15020, 15024, 4787, 27828, 2228, 684
9, 3781, 3273, 7896, 6360, 3818, 13550, 2290, 3325, 4865, 14084, 1797, 7430, 2829, 6418, 7443, 10520,
 2329, 4386, 1831, 2346, 7978, 34095, 3887, 2354, 1848, 4416, 4931, 3908, 2885, 10566, 4934, 5455, 34
11, 2387, 9562, 2907, 3418, 41310, 6497, 3942, 2407, 3432, 10605, 2414, 6514, 2936, 3456, 1409, 3464,
 3471, 1424, 2961, 401, 914, 2964, 5013, 4508, 11678, 2463, 2977, 8614, 1455, 2993, 5556, 1471, 5060,
 6097, 15831, 2009, 991, 4064, 1506, 2538, 2036}
Feature capital.loss values: {0, 2051, 1539, 2057, 1564, 2080, 1573, 1579, 2603, 1590, 1594, 1602, 10
92, 1617, 2129, 1628, 3683, 2149, 1648, 625, 1138, 1651, 2163, 2174, 2179, 1668, 1669, 1672, 653, 220
1, 155, 2205, 2206, 1719, 2231, 1721, 3770, 2238, 1726, 2754, 2246, 1735, 1740, 1741, 2258, 213, 226
7, 1755, 1762, 1258, 2282, 4356, 2824, 1816, 1825, 2339, 810, 2352, 1844, 1848, 3900, 1340, 323, 237
7, 1876, 2392, 1887, 1380, 1902, 2415, 880, 1408, 1411, 2444, 1944, 2457, 419, 2467, 2472, 1974, 197
7, 2489, 1980, 3004, 1485, 974, 2001, 2002, 1504, 2547, 2042, 2559}
Feature hours.per.week values: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 4
5, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 70, 7
2, 73, 74, 75, 76, 77, 78, 80, 81, 82, 84, 85, 86, 87, 88, 89, 90, 91, 92, 94, 95, 96, 97, 98, 99}
Feature native.country values: {'Scotland', 'Japan', 'Hungary', 'Dominican-Republic', 'Ireland', 'Nic
aragua', 'Taiwan', 'Puerto-Rico', 'Laos', 'India', 'England', 'United-States', 'Yugoslavia', 'Hong',
 'Thailand', 'Peru', 'Cambodia', 'South', 'Guatemala', 'France', 'Ecuador', 'Canada', 'Italy', 'Germa
ny', 'Honduras', 'Philippines', 'Greece', 'Iran', 'Outlying-US(Guam-USVI-etc)', 'Mexico', 'Jamaica',
 'China', 'Cuba', 'Trinadad&Tobago', '?', 'Holand-Netherlands', 'Portugal', 'Columbia', 'Poland', 'Vi
etnam', 'El-Salvador', 'Haiti'}
Feature income values: {'>50K', '<=50K'}
```

In [61]:

```
# I have to deal with "?" strings in categorical variables and possible missing data
df_train = df_train.replace('?', np.nan)
df_test = df_test.replace('?', np.nan)
rows_nan = []
for index,row in df_train.isnull().iterrows():
    if (row.values).any():
        rows_nan.append(index)
len(rows_nan)  #number of rows  that have any missing data
```

Out[61]: 2399

In [62]:

```
print('Rows that have any missing value: ',str((len(rows_nan)/df_train.shape[0])*100) + '%')
```

```
Rows that have any missing value:  7.367936117936118%
```

In [63]:

```
#Owing to the small % of rows having missing data, I'll get rid of them
df_train = df_train.dropna(axis = 0)
display(df_train)

#df_test = df_test.dropna(axis = 0)
```

Id	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.los
16280	34	Private	204991	Some-college	10	Divorced	Exec-managerial	Own-child	White	Male	0	
16281	58	Local-gov	310085	10th	6	Married-civ-spouse	Transport-moving	Husband	White	Male	0	
16282	25	Private	146117	Some-college	10	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	
16283	24	Private	138938	Some-college	10	Divorced	Adm-clerical	Not-in-family	White	Female	0	
16284	57	Self-emp-inc	258883	HS-grad	9	Married-civ-spouse	Transport-moving	Husband	White	Male	5178	
...	...	...	...	...	...	...	...	...	...	...	...	
48835	42	Private	384236	Masters	14	Married-civ-spouse	Prof-specialty	Husband	White	Male	7688	
48836	23	Private	129042	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	Black	Female	0	
48837	30	Private	195488	HS-grad	9	Never-married	Priv-house-serv	Own-child	White	Female	0	
48838	18	Private	27620	HS-grad	9	Never-married	Adm-clerical	Not-in-family	White	Female	0	
48839	47	Local-gov	203067	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Male	0	

30161 rows × 15 columns

In [64]:

```
df_train.columns
```

Out[64]:

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

In [65]:

```
#I will delete education columns once I already have 'education.num'
df_train.drop(labels = 'education', axis = 1, inplace = True)
df_test.drop(labels = 'education', axis = 1, inplace = True)
```

2- Exploratory Data Analysis (EDA) and Data Preparation

2.1 - Correlation Matrix

- Our target is the income, it's a categorical variable with two label
- I'm changing the target in order to have two quantiative value: 0 to <=50 and >50 to 1. Then, I'll able to measure the correlations among the variables

In [66]:

```
df_train_corr = df_train.copy()
df_train_corr['income']  =df_train_corr['income'].apply(lambda x: 0 if x == '<=50K' else 1 )
set(df_train_corr['income'])
```

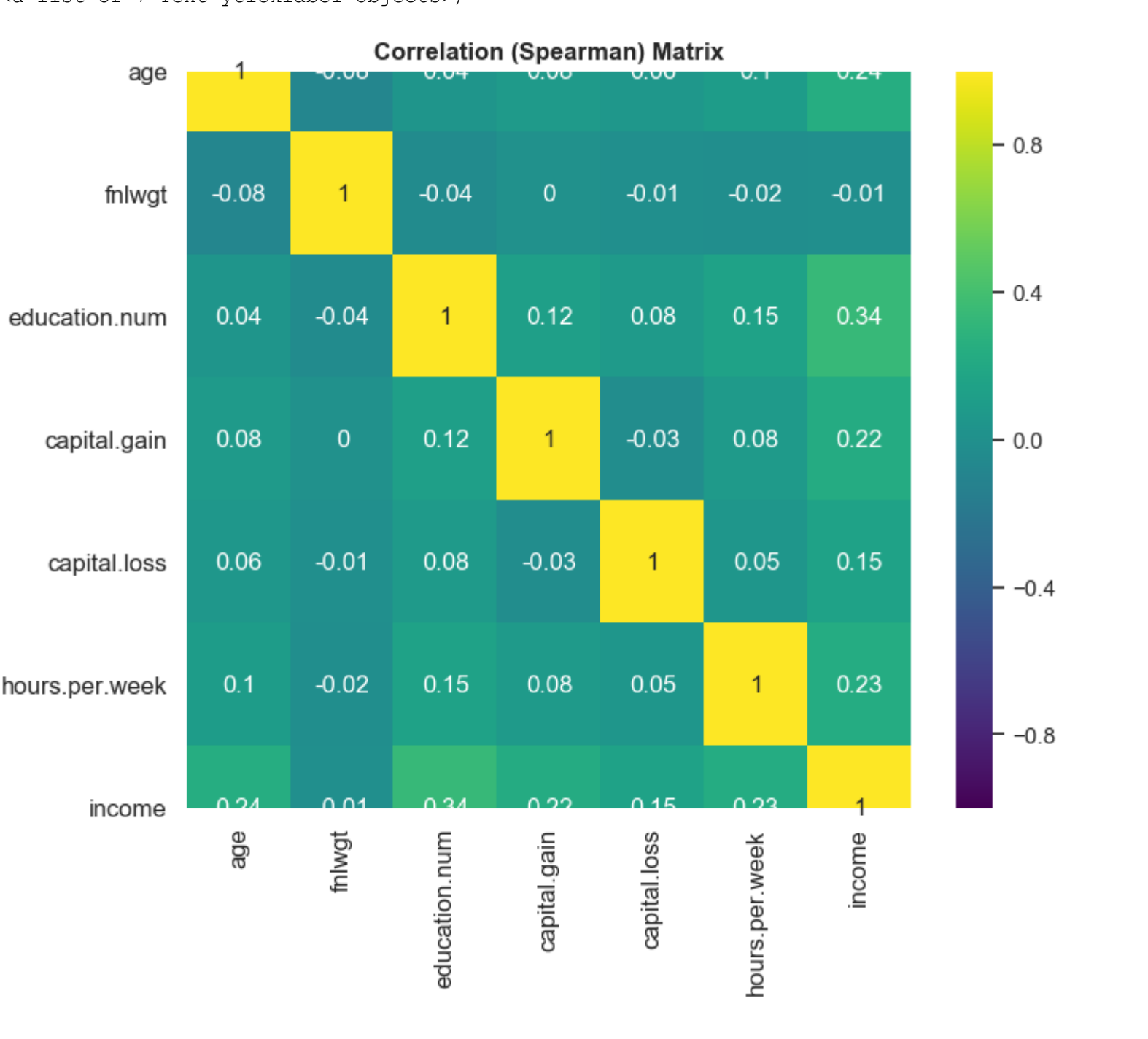
Out[66]: {0, 1}

In [67]:

```
sea.heatmap(round(df_train_corr.corr(),2), annot = True,cmap = 'viridis',vmin = -1, vmax = 1)
plt.title("Correlation (Spearman) Matrix", fontweight = 'bold', fontsize = 12)

#Setting
plt.gcf().set_size_inches(8,8)
plt.gcf().set_dpi(100)
plt.subplots_adjust(bottom = .25)
plt.xticks(size = 12)
plt.yticks(size = 12)
```

Out[67]:

```
(array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5]),
 <a list of 7 Text yticklabel objects>)
```



- Comparing the target with other features, we can see the pearson correlation between income and fnlwgtg is close to 0. So we can considerer take this feature out

In [68]:

```
df_train.drop(labels = 'fnlwgt', axis =1, inplace = True)
df_test.drop(labels = 'fnlwgt', axis =1, inplace = True)
```

In [69]:

```
df_train.head()
```

Out[69]:

Id	age	workclass	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week
16280	34	Private	10	Divorced	Exec-managerial	Own-child	White	Male	0	0	44
16281	58	Local-gov	6	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	40
16282	25	Private	10	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	0	42
16283	24	Private	10	Divorced	Adm-clerical	Not-in-family	White	Female	0	0	40
16284	57	Self-emp-inc	9	Married-civ-spouse	Transport-moving	Husband	White	Male	5178	0	60

2.2 - Analyzing quantitative features

In [70]:

```
display(df_train.head())
quant_features = ['age','education.num','capital.gain', 'capital.loss','hours.per.week']
```

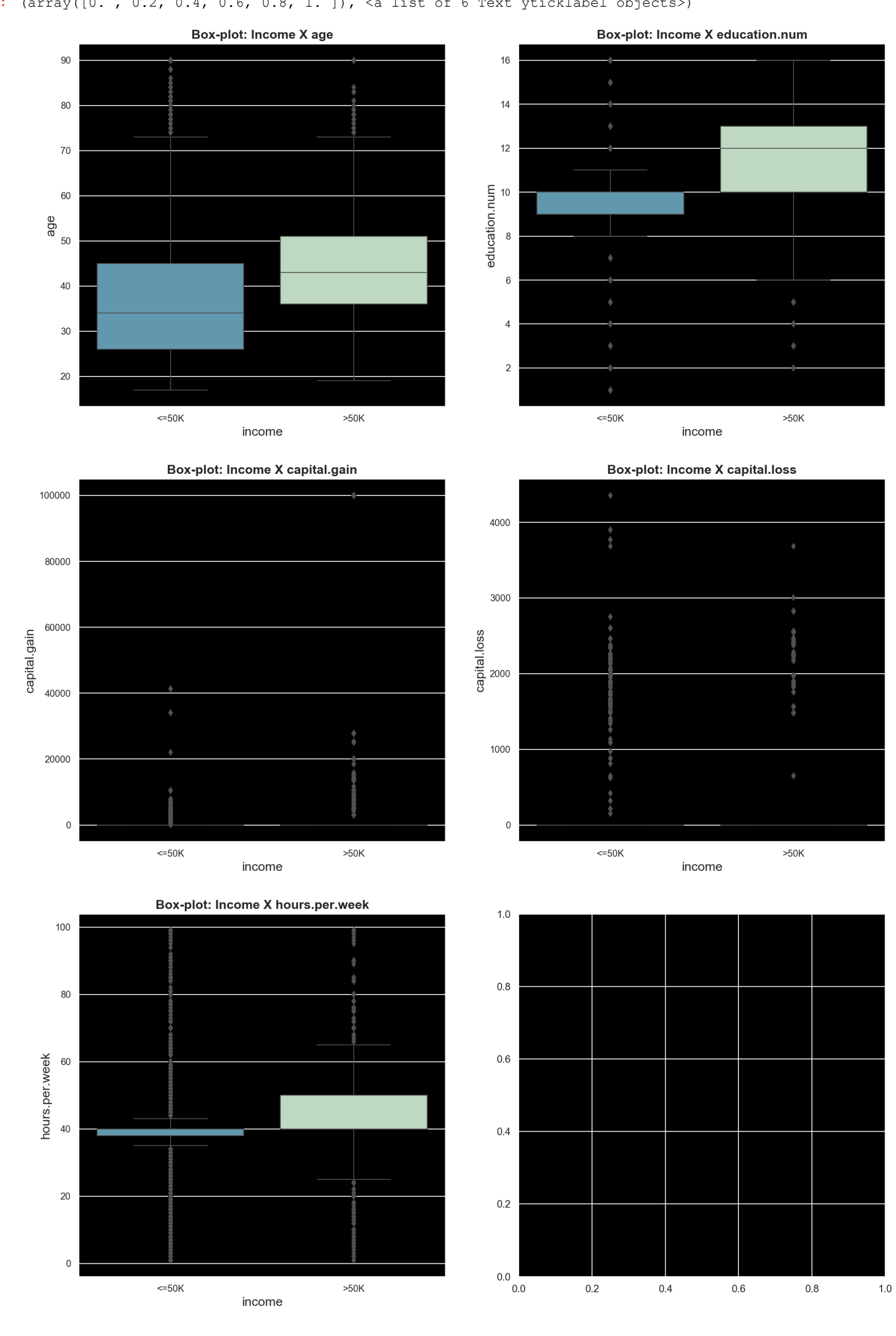
Id	age	workclass	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week
16280	34	Private	10	Divorced	Exec-managerial	Own-child	White	Male	0	0	44
16281	58	Local-gov	6	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	40
16282	25	Private	10	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	0	42
16283	24	Private	10	Divorced	Adm-clerical	Not-in-family	White	Female	0	0	40
16284	57	Self-emp-inc	9	Married-civ-spouse	Transport-moving	Husband	White	Male	5178	0	60

In [71]:

```
sea.set(rc = {'axes.facecolor':'black'} )
fig, ((ax1,ax2),(ax3,ax4),(ax5,ax6)) = plt.subplots(3,2)
axes = [ax1,ax2,ax3,ax4,ax5,ax6]


for feature,ax in zip(quant_features,axes):

    #Plotting
    sea.boxplot(data = df_train, x = 'income', y =feature, palette = "GnBu_r", orient = "v", linewidth
= 1, saturation= 0.56, ax= ax)
    #sea.swarmplot(data = df_train, x = 'income', y = feature, palette = 'GnBu_r', orient = 'v', size =
 7,alpha= 1, edgecolor= 'grey', ax = ax)
    ax.set_xlabel('income', fontsize = 15)
    ax.set_ylabel(feature, fontsize = 15)
    ax.set_title("Box-plot: Income X {}".format(feature) ,size = 15, fontweight = 'bold')


#Setting
plt.gcf().set_size_inches(17,12)
plt.gcf().set_dpi(130)
plt.subplots_adjust(bottom = -0.8)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
```

Out[71]:

```
(array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]), <a list of 6 Text yticklabel objects>)
```



- We can realize there are many outliers from the boxplots that need to be handled
- I'm using Robust Scaler method to deal with outliers

In [72]:

```
df_train[['capital.gain', 'capital.loss', 'hours.per.week']] = RobustScaler().fit_transform(df_train[[
'capital.gain', 'capital.loss', 'hours.per.week']])
df_test[['capital.gain', 'capital.loss', 'hours.per.week']] = RobustScaler().fit_transform(df_test[['ca
pital.gain', 'capital.loss', 'hours.per.week']])
```

2.3 - Analyzing qualitative features

In [73]:

```
display(df_train.head())
qual_features = ['workclass', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.co
untry']
```

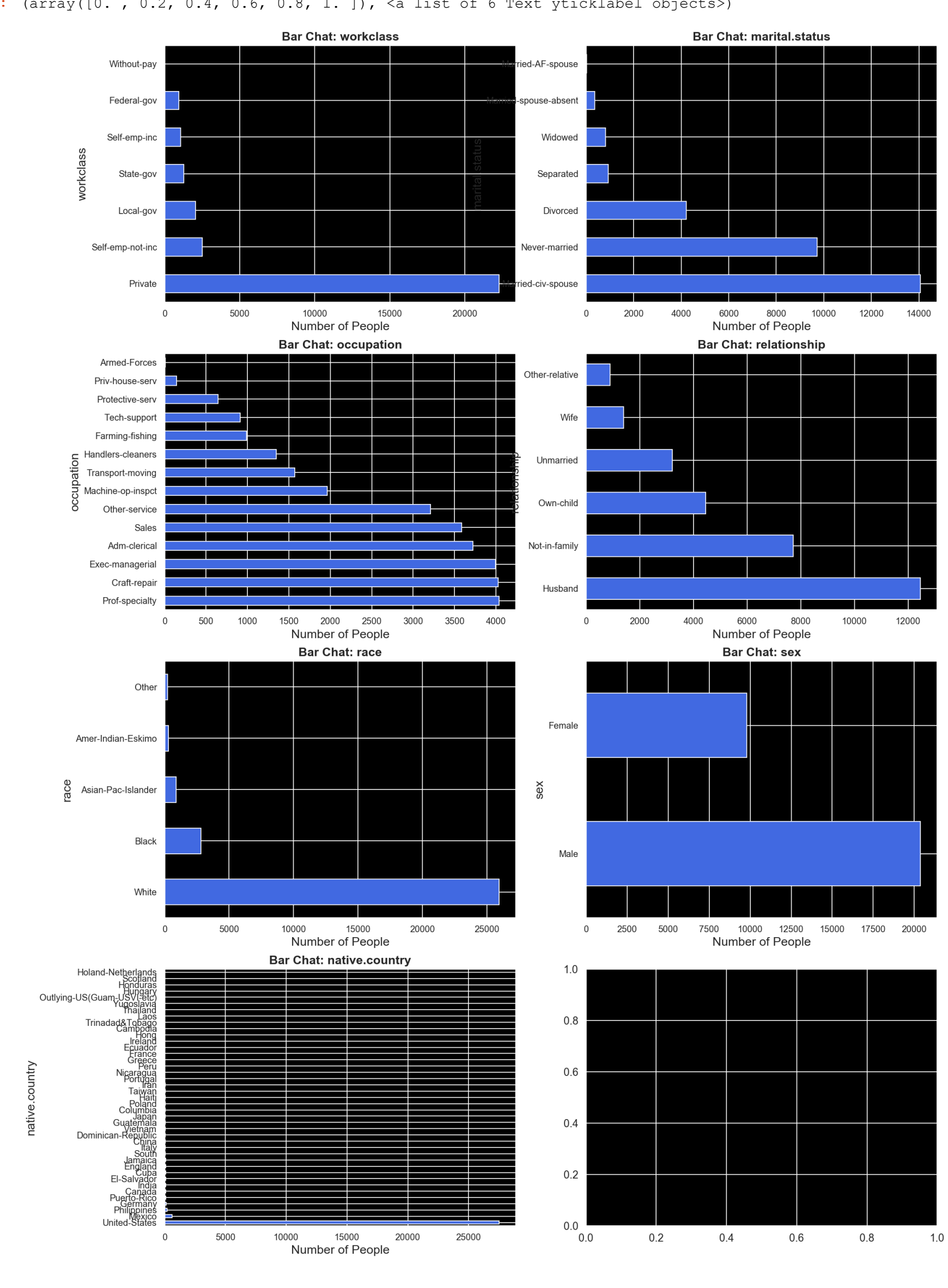
Id	age	workclass	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week
16280	34	Private	10	Divorced	Exec-managerial	Own-child	White	Male	0.0	0.0	0.8
16281	58	Local-gov	6	Married-civ-spouse	Transport-moving	Husband	White	Male	0.0	0.0	0.0
16282	25	Private	10	Never-married	Machine-op-inspct	Not-in-family	White	Male	0.0	0.0	0.4
16283	24	Private	10	Divorced	Adm-clerical	Not-in-family	White	Female	0.0	0.0	0.0
16284	57	Self-emp-inc	9	Married-civ-spouse	Transport-moving	Husband	White	Male	5178.0	0.0	4.0

In [74]:

```
sea.set(rc = {'axes.facecolor':'black'} )
fig, ((ax1,ax2),(ax3,ax4),(ax5,ax6), (ax7,ax8)) = plt.subplots(4,2)
axes = [ax1,ax2,ax3,ax4,ax5,ax6, ax7, ax8]


for feature,ax in zip(qual_features,axes):

    #Plotting
    df_train[feature].value_counts().plot.barh( color = 'royalblue', ax = ax)
    ax.set_xlabel('Number of People', fontsize = 15)
    ax.set_ylabel(feature, fontsize = 15)
    ax.set_title("Bar Chat: {}".format(feature) ,size = 15, fontweight = 'bold')


#Setting
plt.gcf().set_size_inches(17,12)
plt.gcf().set_dpi(130)
plt.subplots_adjust(bottom = -0.8)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
```

Out[74]:

```
(array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]), <a list of 6 Text yticklabel objects>)
```



- The fact that catch my attention the most is the too way greater number of records from the US

In [75]:

```
df_train.groupby(by = 'native.country').agg({'native.country': lambda x: str(round(np.count_nonzero(x)/
len(df_train) * 100,2)) + ' %'})
```

Out[75]:

native.country	native.country
Cambodia	0.06 %
Canada	0.35 %
China	0.23 %
Columbia	0.19 %
Cuba	0.31 %
Dominican-Republic	0.22 %
Ecuador	0.09 %
El-Salvador	0.33 %
England	0.29 %
France	0.09 %
Germany	0.42 %
Greece	0.1 %
Guatemala	0.21 %
Haiti	0.14 %
Holand-Netherlands	0.0 %
Honduras	0.04 %
Hong	0.06 %
Hungary	0.04 %
India	0.33 %
Iran	0.14 %
Ireland	0.08 %
Italy	0.23 %
Jamaica	0.27 %
Japan	0.2 %
Laos	0.06 %
Mexico	2.02 %
Nicaragua	0.11 %
Outlying-US(Guam-USVI-etc)	0.05 %
Peru	0.1 %
Philippines	0.62 %
Poland	0.19 %
Portugal	0.11 %
Puerto-Rico	0.36 %
Scotland	0.04 %
South	0.24 %
Taiwan	0.14 %
Thailand	0.06 %
Trinadad&Tobago	0.06 %
United-States	91.19 %
Vietnam	0.21 %
Yugoslavia	0.05 %

- We can notice that ony the United-States accounts for more than **91%** of the dataset records. We can conclude that the dataset is totally unballanced
- Very well. I'll handle it placing 1 to people from the US and 0 to non people from the US

In [76]:

```
df_train['USA'] = df_train['native.country'].apply(lambda x: 1 if x == 'United-States' else 0 )
df_test['USA'] = df_test['native.country'].apply(lambda x: 1 if x == 'United-States' else 0 )
```

- Now I have to convert the labels to numerical values in order to apply the KNN classifier

In [78]:

```
df_test
```

Out[78]:

	age	workclass	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week
1	25	Private	7.0	Never-married	Machine-op-inspct	Own-child	Black	Male	0.0	0.0	0.0
2	38	Private	9.0	Married-civ-spouse	Farming-fishing	Husband	White	Male	0.0	0.0	2.0
3	28	Local-gov	12.0	Married-civ-spouse	Protective-serv	Husband	White	Male	0.0	0.0	0.0
4	44	Private	10.0	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688.0	0.0	0.0
5	18	NaN	10.0	Never-married	NaN	Own-child	White	Female	0.0	0.0	-2.0
...	...	...	...	...	...	...	...	...	...	...	...
16277	39	Private	13.0	Divorced	Prof-specialty	Not-in-family	White	Female	0.0	0.0	-0.8
16278	64	NaN	9.0	Widowed	NaN	Other-relative	Black	Male	0.0	0.0	0.0
16279	38	Private	13.0	Married-civ-spouse	Prof-specialty	Husband	White	Male	0.0	0.0	2.0
16280	44	Private	13.0	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455.0	0.0	0.0
16281	35	Self-emp-inc	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	4.0

16281 rows × 14 columns

In [81]:

```
df_test['workclass']
```

Out[81]:

```
1              Private
2              Private
3            Local-gov
4              Private
5                  NaN
             ...
16277          Private
16278              NaN
16279          Private
16280          Private
16281     Self-emp-inc
Name: workclass, Length: 16281, dtype: object
```

In [83]:

```
#training dataset
for feature in qual_features:
    df_train[feature] = LabelEncoder().fit_transform(df_train[feature])

#Test dataset
for feature in qual_features:
    print(feature)
    df_test[feature] = LabelEncoder().fit_transform(df_test[feature].astype(str))
```

```
workclass
marital.status
occupation
relationship
race
sex
native.country
```

3- Using KNN (K-Nearest Nighbor) Cassifier to Predict

3.1- Feature Selection

In [84]:

```
df_train.columns
```

Out[84]:

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

In [85]:

```
feature_select = ['age', 'workclass', 'education.num', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'capital.gain', 'capital.loss',
       'hours.per.week', 'USA']
target = 'income'

x_train = df_train[feature_select]
y_train = df_train[target]
x_test = df_test[feature_select]
y_test = df_test[target]
```

3.2 Applyig the Classifier to the training dataset

In [86]:

```
#I'm using cross validation method with 10 folders to select K that presents more accuracy
KN = np.arange(10, 45, 5)
for K in KN:
    score = cross_val_score(KNeighborsClassifier(n_neighbors = K), x_train, y_train, cv = 10, scoring =
 'accuracy').mean()
    print(f'K-Neighbor: {K} | Accuracy: {score}')
```

```
K-Neighbor: 10 | Accuracy: 0.85103284531966
K-Neighbor: 15 | Accuracy: 0.8510327683393977
K-Neighbor: 20 | Accuracy: 0.8507676592987716
K-Neighbor: 25 | Accuracy: 0.8499058320670976
K-Neighbor: 30 | Accuracy: 0.8503368666227574
K-Neighbor: 35 | Accuracy: 0.8483142101443745
K-Neighbor: 40 | Accuracy: 0.8473857512304079
```

- I pick up the K that presents the best accuracy

3.3- Applyig the Classifier to the testing dataset

In [87]:

```
K = 10
knn = KNeighborsClassifier(n_neighbors = K)
knn.fit(x_test,y_test)
y_predict= knn.predict(x_test) #classification in practice
```

In [88]:

```
y_predict
```

Out[88]:

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

In [89]:

```
#Classification accuracy
accuracy_score(y_test, y_predict)
```

Out[89]: 0.8766046311651619

3.4 Saving Results

In [90]:

```
df_test
```

Out[90]:

	age	workclass	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	nativ
1	25	3	7.0	4	11	3	2	1	0.0	0.0	0.0	
2	38	3	9.0	2	9	0	4	1	0.0	0.0	2.0	
3	28	1	12.0	2	2	0	4	1	0.0	0.0	0.0	
4	44	3	10.0	2	11	0	2	1	7688.0	0.0	0.0	
5	18	8	10.0	4	6	3	4	0	0.0	0.0	-2.0	
...	...	...	...	...	...	...	...	...	...	...	...	
16277	39	3	13.0	0	14	1	4	0	0.0	0.0	-0.8	
16278	64	8	9.0	6	6	2	2	1	0.0	0.0	0.0	
16279	38	3	13.0	2	14	0	4	1	0.0	0.0	2.0	
16280	44	3	13.0	0	0	3	1	1	5455.0	0.0	0.0	
16281	35	4	13.0	2	8	0	4	1	0.0	0.0	4.0	

16281 rows × 14 columns