Census Income Project

I always look forward to sharing my best Favorite project with all of you.

Census Income

census money income is defined as income received on a regular basic exclusive of certain money receipts such as capital gains before payments for personal income taxes social security union dues medicare, deductions in additions some nonfare residents which may take from of the use of business transportations and facilities full or partial payments by business for retirements program medical and educations expenses.

Data users should consider these elements when comparing income levels moreover users should be aware that for many different reasons there is a tendency in household surveys for respondents to underreports their income.

1. Problem Definition.

This data was extracted from the 1994 census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of

reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). The prediction task is to determine whether a person makes over $50k a year.

1) A single cell estimate of the populations 16+ for each state
2) Controls for Hispanic origin by age and sex
3) Controls by Race age and sex

We use all three sets of controls in our weighting program and "rake" through them 6 times so that by the end we come back to all the controls we used. The term estimate refers to population totals derived from CPS by creating "weighted tallies" of any specified socio-economic characteristics of the population. People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Loading the basic libraries this is how we start with the project so we load the basic libraries so now I come to like you can import pandas as pd and numpy as np for data visualization we will import matplotlib . pyplot as plt And I will import seaborn sns and in the meantime I will import warnings just because it will not give me much warnings and filter these warnings and just right here ignore I will ignore there warnings

```
data = pd.read_csv(r"D:\New folder\census_income.csv")
```

```
data
```

Now we will go to load the dataset so we will load the data with the help of pandas so data is equal to I will go and tell pandas to read the csv file and my csv file was cansus_income.csv and then I will tell it to read the first all entries so that I can see how does this loaded.

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	Race	Sex	Capital_gain	Capital_loss
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0

32560 rows × 15 columns

Now you can see this data set as you are age work class education marital_status relationship something called weight and education number Income and this is the income this is the last feature so the output has already been included so this makes it supervised machine learnings and as you can see this is has got binary this thing less than equal to 50000 and this is more than fifty thousand according to these feature we have to predict whether this person makes over 50000 or not a year

```
#Lets check the shape of dataset
data.shape
```

```
(32560, 15)
```

Let’s see the shape of my data set so we will see the shape and see there are like 32560 rows and 15 columns

2. Data Analysis.

```
#Lets check the data type of dataset
data.dtypes
```

```
Age                int64
Workclass         object
Fnlwgt             int64
Education         object
Education_num      int64
Marital_status    object
Occupation        object
Relationship      object
Race              object
Sex               object
Capital_gain       int64
Capital_loss       int64
Hours_per_week     int64
Native_country    object
Income            object
dtype: object
```

data . d types in this we can just see the data types and we find that this is a mix of integer and object that is categorical and numeric variable and this work class marital status occupations all these are in categorical format and rest you can see that education is again object hours per week and capital gain capital loss this flyweight this is tell you what this is something called data types

```
# Checking for null values

data.isna().sum()
```

```
Age                  0
Workclass         1836
Fnlwgt               0
Education            0
Education_num        0
Marital_status       0
Occupation        1843
Relationship         0
Race                 0
Sex                  0
Capital_gain         0
Capital_loss         0
Hours_per_week       0
Native_country     582
Income               0
dtype: int64
```

Let's one more thing and that is whether this is is null . sum that means how many I want to find out the total number of missing values and other means 0 entries now this is showing me that there are no null values in either of the feature.

```
data['Workclass'].value_counts()
```

```
Private             22673
Self-emp-not-inc     2540
Local-gov            2093
?                    1836
State-gov            1297
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: Workclass, dtype: int64
```

```
data['Occupation'].value_counts()
```

```
Prof-specialty       4136
Craft-repair         4094
Exec-managerial      4065
Adm-clerical         3767
Sales                3650
Other-service        3291
Machine-op-inspct    2000
?                    1843
Transport-moving     1597
```

```
data['Native_country'].value_counts()
```

```
United-States    29152
Mexico             639
?                  582
Philippines        198
Germany            137
Canada             121
```

One thing there are no null value but if you have observed here we got certain question marks (?) you see this entry for and you see this question marks here in work class you see this in occupation question marks you see this Native country here again question marks so now we have to figure out how many question marks are there is it really an we drop these and we have deal with these question marks is cut up data so either we drop it or we fill it with something and we also have determine in how many columns do we have these questions marks so just study this data table that this work class this questions marks all right occupations has these question marks and we have also found out that this is all about question marks.

```
data.describe()
```

	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week	Income
count	31646.000000	31646.000000	31646.000000	31646.000000	31646.000000	31646.000000	31646.000000
mean	38.383872	185321.719301	10.085603	1071.922455	87.341117	40.079125	0.240504
std	13.347389	94528.515467	2.559842	7368.643817	402.016224	11.181322	0.427397
min	17.000000	12285.000000	1.000000	0.000000	0.000000	4.000000	0.000000
25%	28.000000	117606.000000	9.000000	0.000000	0.000000	40.000000	0.000000
50%	37.000000	177667.000000	10.000000	0.000000	0.000000	40.000000	0.000000
75%	47.000000	234442.250000	12.000000	0.000000	0.000000	45.000000	0.000000
max	79.000000	506436.000000	16.000000	99999.000000	4356.000000	77.000000	1.000000

Now we some statistical calculation and we do this data describe and in this describe what we do is take this transpose of it so in transport what happens is that rows are converted into columns suppose if don't give this this will like this and if give dot transpose you see the count of this the same number of rows and the mean of age is 38.3 and the minimum age is 17 years and maximum age is 79 years so our data set is spread in the age group of 17 to 79 years you can see the education number it is like something minimum is 1 and maximum is 16 that means the number we are talking about and it is almost redundant information like of education capital gain and capital loss again can be seen here and hours per week so we got maximum hours per week is 77 and minimum is 4 that all data describe.

We come to this questions marks this thing to deal with some coding.

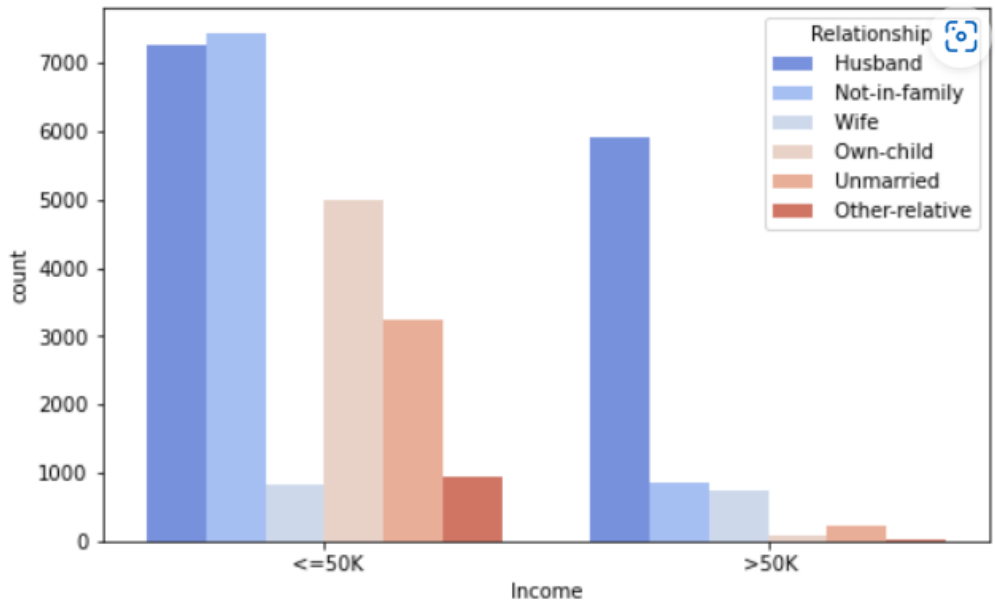
```
# Replacing value to nan value
```

```
data.Native_country.replace(' ?', np.nan, inplace = True)
data.Workclass.replace(' ?', np.nan, inplace = True)
data.Occupation.replace(' ?', np.nan, inplace = True)
```

Now we come to fixing of this data exploration part being take further but first just fill I would fill the question marks filling question marks value let's just deal with this first

and how do I fill this I will replace it with null value. I will convert this into hash to comment so that it does throw an error now what I do is I would just simply write here the work class that was having and how do I write it so in this work class you go to this work class and replace what you should replace this question marks with what replace this question marks with null value and next I come to the occupation I write here occupation and what I do again is right I again write here occupation dot replace the null value and next columns are and that was a native country and here I will write data what is it native country dot replace the question marks again and this replace this with null value so I will replace it that is all about filling question marks columns.

```
plt.figure(figsize=(8,5))
sns.countplot(data['Income'],palette='coolwarm',hue='Relationship',data=data);
```



Sns dot count plot to plot this and show you I will just data and income and I could also used it here in fact

and let's palette equals to I will take cool warn and let's make it a little interesting adding hue and I will add hue as relationship and from where I will take this will it from data and this data is and that does not include any semicolon so now you see this relationship income 1 and income 2 and the make having less than equal to 50000 as you can see that comes here it is grater than 7000 that all about relationship with income label columns.

Replace label columns

```
from sklearn.preprocessing import LabelEncoder
```

```
lab_enc=LabelEncoder()
```

```
df2 = lab_enc.fit_transform(data['Income'])
pd.Series(df2)
```

I want to convert it into numeric data type that all in binary only two outcomes.

```
data['Income'].value_counts()
```

```
0    24697
1     7839
Name: Income, dtype: int64
```

Now we do is we just check it so we just check the value now and you can see here that the income is convert now into zero and one more understandable for me.

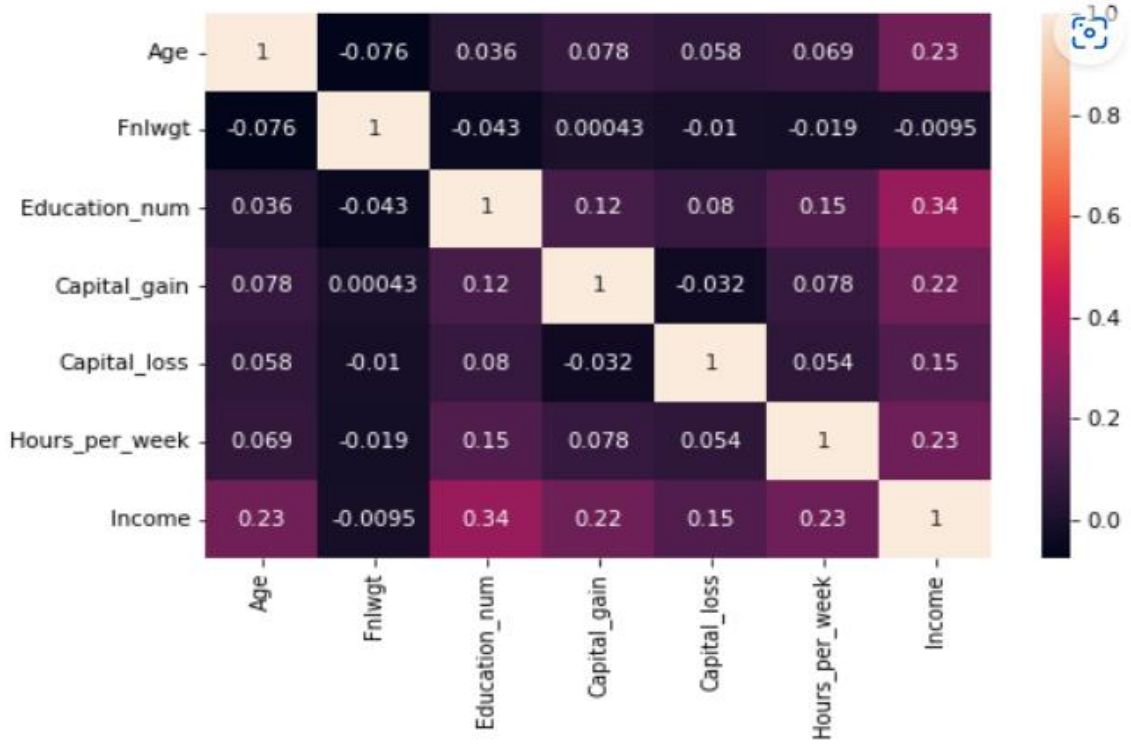
3. EDA Concluding Remark.

```
data.corr()
```

	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week	Income
Age	1.000000	-0.076448	0.036224	0.077676	0.057745	0.068515	0.234039
Fnlwgt	-0.076448	1.000000	-0.043353	0.000433	-0.010267	-0.018900	-0.009521
Education_num	0.036224	-0.043353	1.000000	0.122661	0.079901	0.148426	0.335299
Capital_gain	0.077676	0.000433	0.122661	1.000000	-0.031638	0.078408	0.223340
Capital_loss	0.057745	-0.010267	0.079901	-0.031638	1.000000	0.054229	0.150498
Hours_per_week	0.068515	-0.018900	0.148426	0.078408	0.054229	1.000000	0.229659
Income	0.234039	-0.009521	0.335299	0.223340	0.150498	0.229659	1.000000

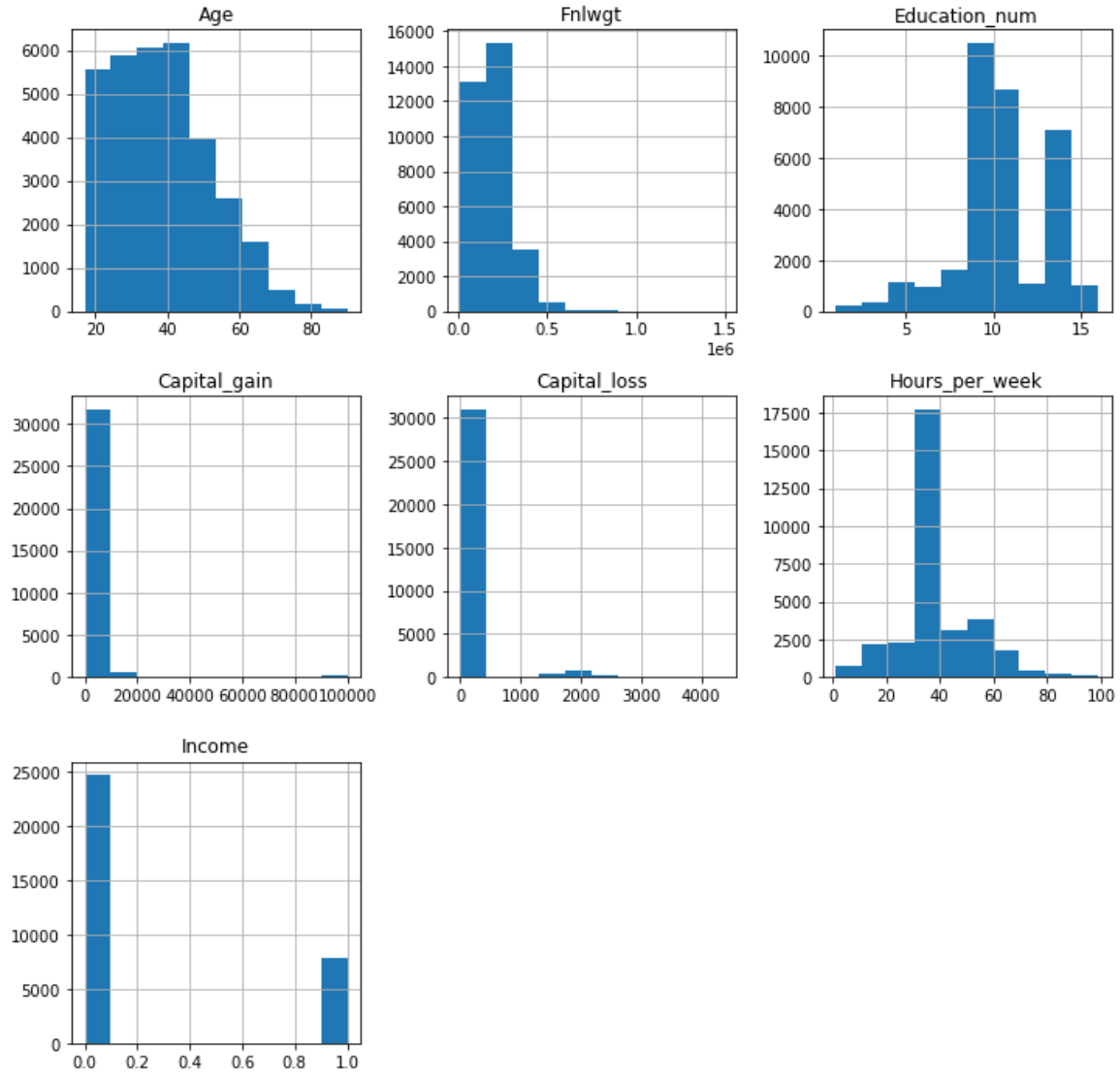
Correlation let’s figure out what is the correlation and this correlation is for the numeric data numeric these are the numeric integers which we are seeing and see that the figure that there is no as such high positive correlation just normal to moderate correlation variable with income and age and as you can see this is negatively correlated negative sigh because this is not so much of a concern and if you want to plot this we will use heat map for this so we will use this heat map and in this I will write data dot corr I’ll use this and let’s see this

<AxesSubplot:>



You can see this the correlation all the value have been written inside and how each feature is linked with another income and age has 0.23 correlation se this is figuring somewhere here which is weak that means is like If you got greater age your income might possibly be into the brackets of 50000 and above but we have to figure t out is not that evident from this so this is just given you’re an overview about the correlation that there not so much strong correlation variables but yes there is like weakly correlated correlation that we can see here now

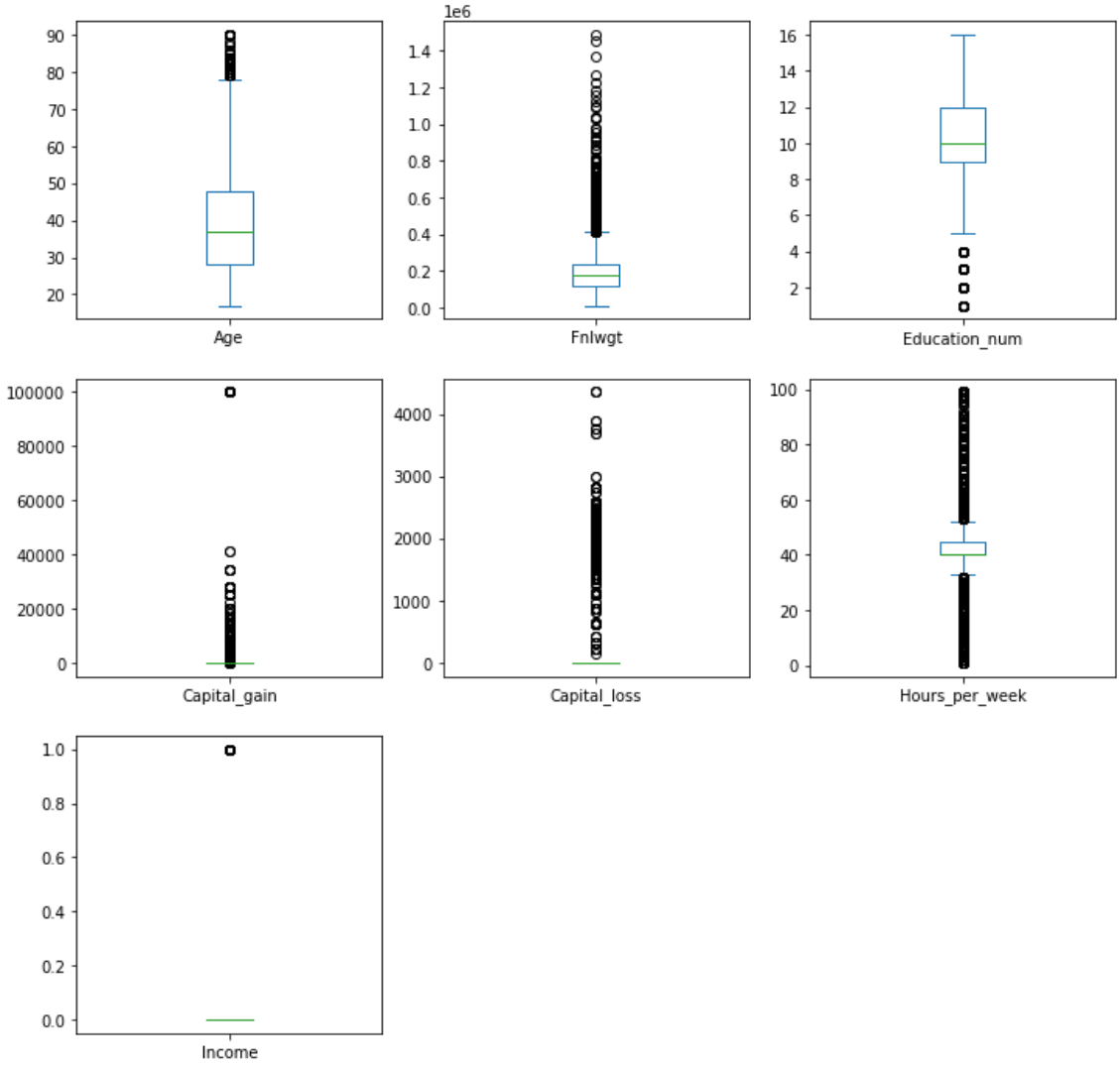
The shape of the data and that is through histogram



How does this comes out this histogram so you can see the histogram now you see that we've got around seven variable which are being plotted age and you can see the distribution of eight here this age is spread from like around age from 17 because Use we know that from describe it is 17 to 90 years and the most of the age group lies in the braket of 20 to around 50 capital gain it is from

0 to somewhere around thousand ten thousand or so 1500 capital loss and these the values you see here which are really small these are also outliers and will see these outliers separately with box plot and similarly you can see this hours per week so maximum people working in this bracket in this range around 25 to 40 hours per week so they are really very hard workers income people who have got like less than or greater then less than is equal to fifty thousand want to observe this is histogram.

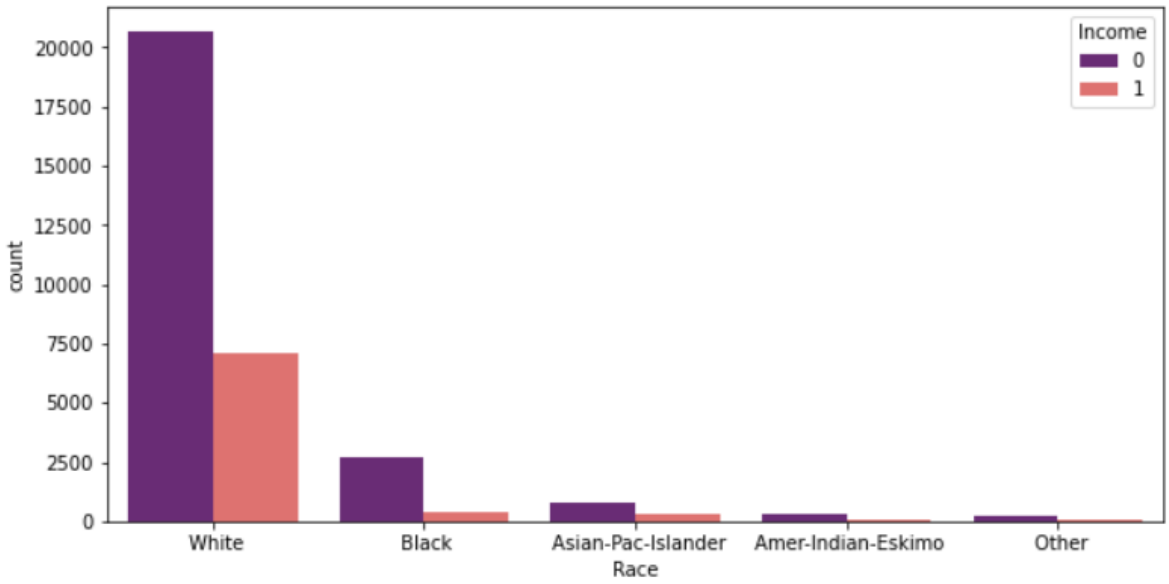
Now for the box plot



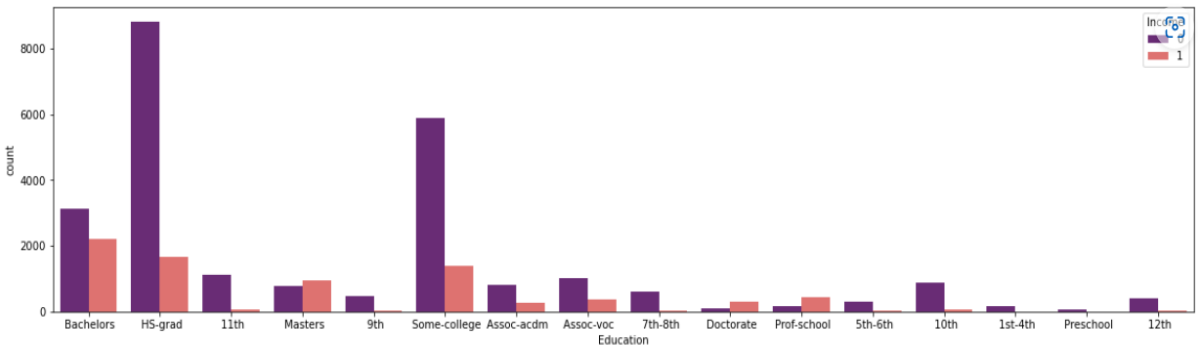
Now I get the outliers these are the extreme values in my data set so I have almost all numeric data like age I have this extreme value which is like around the most of the age is lying in the age group of like as you discussed in histogram also from 20yo around d50 but you have people who were in the working group were above 50 to exceeding to 90 or so and few very people here like in the 80 to 90 similarly you can find here all box plot columns fnlwgt , education number ,

capital gain , capital loss , hours per week , also outliers .

```
<AxesSubplot:xlabel='Race', ylabel='count'>
```



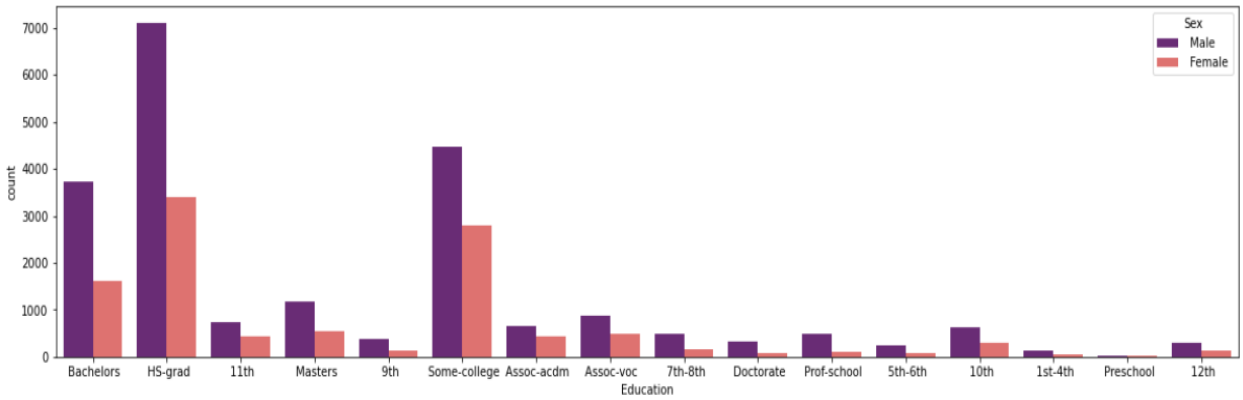
Count want to see the count and I see that divide white have got the highest income and there are they are highest in the number in this data set so we can find that their income Is variable is high in both the cases then black in the population and then we got Asian pacific island race American Indian eskimo is the lowest this is the data you can see here.



Education and income

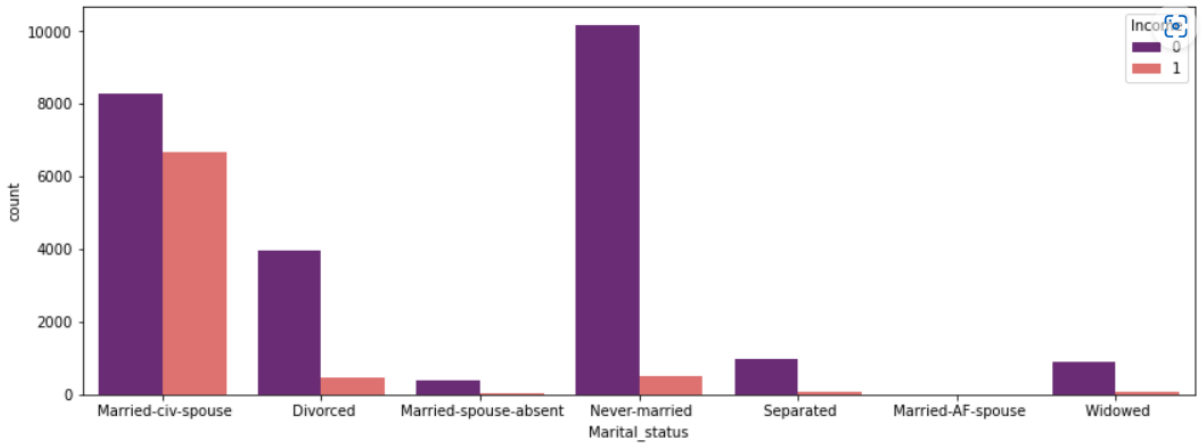
So you can see in education you got you income now

Education and sex



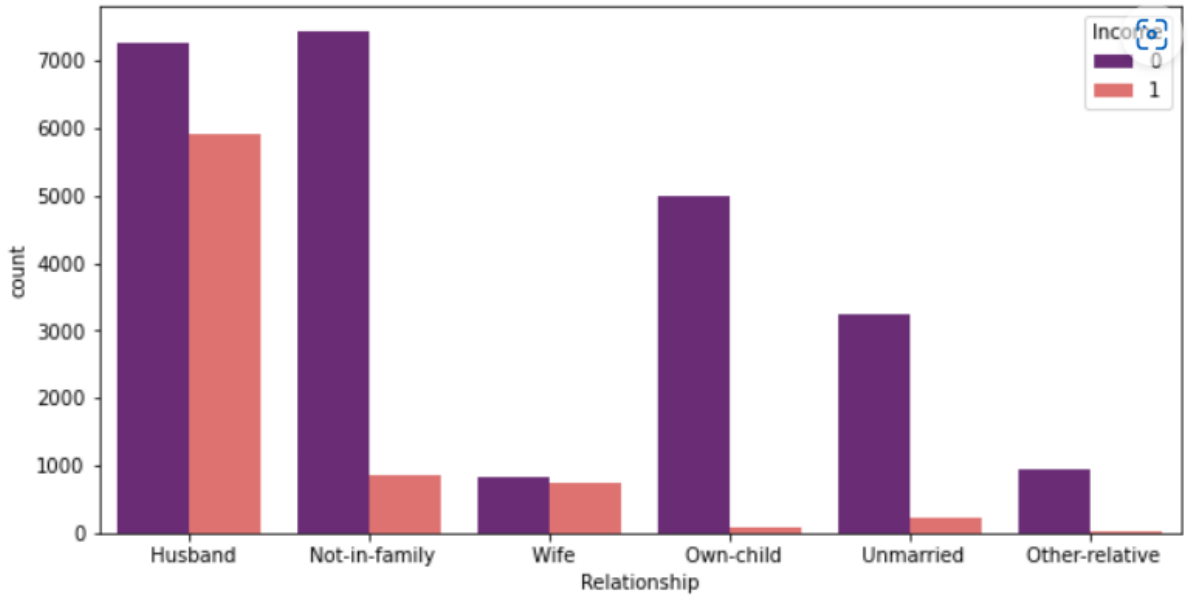
So you can see that male and gender and their count in each of the education category and you can see that females are much lower as compared to males and you see in hs-grad you’re got the highest number of males and then followed by followed by this in highest education.

Marital status



So you can see in marital status you got you income now highest are never married marital status.

Relationship



So you can see in Relationship you got you income now highest are husband and not in family income relationship.

4. Pre-Processing Pipeline.

Removing Outliers:

```
from scipy.stats import zscore
z_score=zscore(data[['Age','Fnlwgt','Hours_per_week']])
abs_z_score=np.abs(z_score)
```

```
filtering_entry=(abs_z_score<3).all(axis=1)
data=data[filtering_entry]
data.describe()
```

Removing Outliers present in our dataset in age, fnlwgt, hours per week are maximum outliers in this columns we use the z-score method to remove outliers

Removing Skewness:

```
power_transform(data[['Education_num','Capital_gain','Capital_loss','Hours_per_week']])
```

So you can see we use power transform to removing skewness our data set education number, capital-gain, capital-loss, hours per week, this columns are likely to skewness problem then we removing skewness.

Encoding:

```
df1 = data.copy()
df1 = df1.apply(LabelEncoder().fit_transform)
df1.head()
```

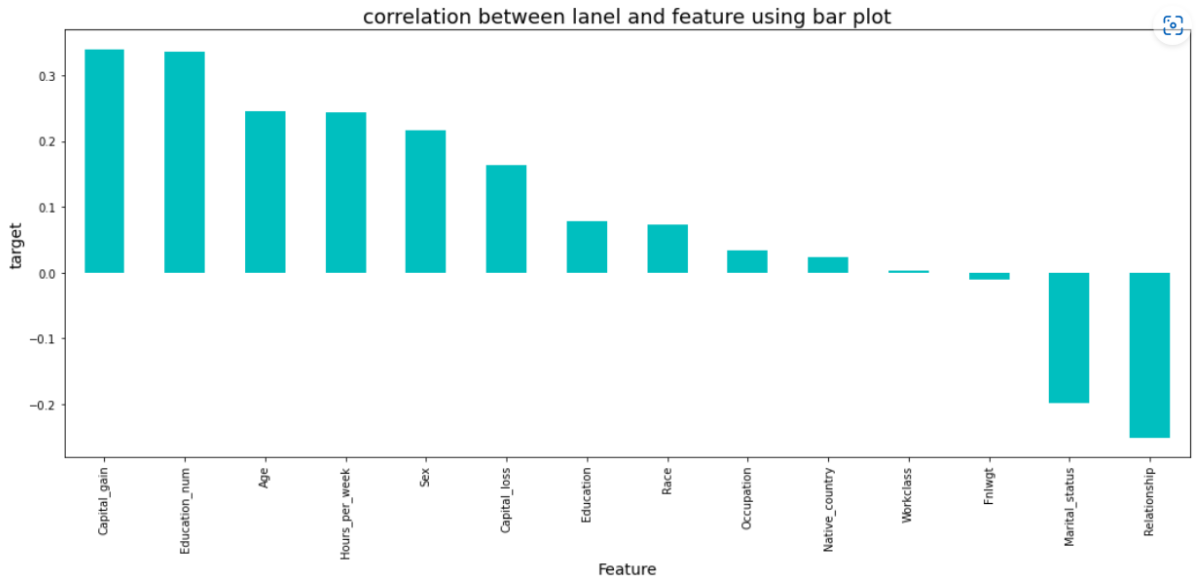
	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship
0	33	5	2875	9	12	2	3	0
1	21	3	13888	11	8	0	5	1
2	36	3	15126	1	6	2	5	0
3	11	3	19089	9	12	2	9	5
4	20	3	17451	12	13	2	3	5

So you can see encoding method to use all categorical columns to numeric format because model building only need to numeric format number it has assigned numbers to each categorical variable you can see marital status 0 and 1 2 this how it works the label encoder just fits your data into machine readable format .

```
cor['Income'].sort_values(ascending=False)
```

```
Income            1.000000
Capital_gain      0.339778
Education_num     0.336189
Age               0.245367
Hours_per_week    0.243065
Sex               0.215579
Capital_loss      0.162781
Education         0.077845
Race              0.072690
Occupation        0.034063
Native_country    0.022830
Workclass         0.003723
Fnlwgt           -0.010923
Marital_status   -0.198533
Relationship     -0.250493
Name: Income, dtype: float64
```

You can see this the correlation all the value have been written inside and how each feature is linked with target variable strong relationship with capital-gain, education-number etc less correlation with income columns are marital status and relationship columns.



Visualizing the correlation between label and features using bar plot so as you can see marital status and relationship.

```
x=df1.drop(columns='Income')#Feature
y=df1.Income#Target
```

Dividing data in feature and label so as you can see now. Y this is the dependent variable and X is the independent variable.

```
#Lets import standardscaler
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)
x_scaled
```

```
array([[ 0.87030597,  1.72614868, -1.25134   , ..., -0.20511256,
        -2.43136463,  0.26069089],
       [-0.02876051, -0.08037626,  0.61858631, ..., -0.20511256,
        -0.00545122,  0.26069089],
       [ 1.09507259, -0.08037626,  0.82878959, ..., -0.20511256,
        -0.00545122,  0.26069089],
       ...,
       [ 1.46968363, -0.08037626, -0.42224256, ..., -0.20511256,
        -0.00545122,  0.26069089],
       [-1.22751581, -0.08037626,  0.41670286, ..., -0.20511256,
        -1.80242412,  0.26069089],
       [ 1.02015039,  0.82288621,  1.24478155, ..., -0.20511256,
        -0.00545122,  0.26069089]])
```

We are going to do our columns to unit less and then we are going to standard size this data them we called standard scaler.

VIF - > Variance Inflation Factor

This is how our data looks now after scaling great now we will check for multicollinearity using VIF (variance inflation factor)

```
#Lets import VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif=pd.DataFrame()
vif['vif']=[variance_inflation_factor(x_scaled,i)for i in range(x_scaled.shape[1])]
vif['feature']=x.columns
```

```
vif
```

	vif	feature
0	1.179915	Age
1	1.008464	Workclass
2	1.013152	Fnlwgt
3	1.156132	Education
4	1.227840	Education_num
5	1.138493	Marital_status
6	1.015752	Occupation
7	1.685572	Relationship
8	1.033357	Race
9	1.565482	Sex
10	1.057905	Capital_gain
11	1.022074	Capital_loss
12	1.144347	Hours_per_week
13	1.029001	Native_country

All the VIF values are less than 5 and very low that means no multicollinearity now we can go ahead with fitting our data.

5. Building Machine Learning Models.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

```
from sklearn.metrics import accuracy_score,classification_report
from sklearn.model_selection import train_test_split
```

```
for i in range(0,1000):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=0.20)
    lr.fit(x_train,y_train)

    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    if round(accuracy_score(y_train,pred_train)*100,1)==round(accuracy_score(y_test,pred_test)*100,1):
        print("At Radom State ",i,"The Model Performs Very Well")
        print("At random State ",i,"The Model performs very well")
        print("Training Accuracy Score is :- ",accuracy_score(y_train,pred_train)*100)
        print("Testing Accuracy Score is : - ",accuracy_score(y_test,pred_test)*100,"\n")
```

I will be using logistic regression in this and because classifiers that is logistic regression and so we will go with logistic regression it is use for binary classification in the staring because I wanted to show you from where you want to really import so from sk learn dot linear model you find this logistic regression I will import logistic regression and thewn you can go ahead again with from sk lean dot metric I will also measure this performance of my classifier that is logistic regression I will see the accuracy of it so I will go for accuracy score and I will crate an object called lr in this I will store the logistic regression classifier so I will write here create a object pred what lr dot predict x train and also one more create object pred test equal to lr dot predict x test print the accuracy of everything so that I m using this and again will making prediction on x test and y test and let’s just run this.

```
At Radom State  218 The Model Performs Very Well
At random State  218 The Model performs very well
Training Accuracy Score is :-  80.73155316795703
Testing Accuracy Score is : -  80.71090047393365
```

So this you can say that on training accuracy score perming something around 80.73 and testing accuracy score is 80.71 this is my logistic regression.

Lets check with KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

```
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=i)
    knn.fit(x_train,y_train)
    x_pred=knn.predict(x_train)
    y_pred=knn.predict(x_test)
    print(f" At random state {i} , the testing accuracy is :- {accuracy_score(y_train,x_pred)}")
    print(f" At random state {i} , the testing accuracy is :- {accuracy_score(y_test,y_pred)}")
    print("\n")
```

I will be using Kneighbors classifier in this and because classifiers that is Kneighbors classifier and so we will go with Kneighbors classifier it is use for binary classification in the staring because I wanted to show you from where you want to really import so from sk learn dot neighbors you find this Kneighbors classifier I will import Kneighbors classifier and thewn you can go ahead again with from sk lean dot metric I will also measure this performance of my classifier that is Kneighbors classifier I will see the accuracy of it so I will go for accuracy score and I will crate an object called knn in this I will store the Kneighbors classifier so I will write here create a object x pred what knn dot predict x train and also one more create object y pred test equal to knn dot predict x test print the accuracy of everything so that I m using this and again will making prediction on x test and y test and let’s just run this.

```
knn.score(x_train,y_train)#Training Score
```

```
0.843751872333702
```

```
knn.score(x_test,y_test)#Testing Score
```

```
0.768404423807267
```

So this you can say that on training accuracy score perming something around 84 and testing accuracy score is 76 this is my KNeighbors Classifier.

Lets check with AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
```

```
ada = AdaBoostClassifier()
```

```
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=i)
    ada.fit(x_train,y_train)
    x_pred=ada.predict(x_train)
    y_pred=ada.predict(x_test)
    print(f" At random state {i} , the testing accuracy is :- {accuracy_score(y_train,x_pred)}")
    print(f" At random state {i} , the testing accuracy is :- {accuracy_score(y_test,y_pred)}")
    print("\n")
```

Another algorithm use I will be using AdaBoost Classifier in this and because classifiers that is AdaBoost Classifier and so we will go with AdaBoost Classifier it is use for binary classification in the staring because I wanted to show you from where you want to really import so from sk learn dot ensemble you find this AdaBoost Classifier I will import AdaBoost Classifier and the you can go ahead again with from I will also measure this performance of my classifier that is AdaBoost Classifier I will see the accuracy of it so I will go for accuracy score and I will crate an object called ada in this I will store the AdaBoost

Classifier so I will write here create a object x pred what ada dot predict x train and also one more create object y pred test equal to ada dot predict x test print the accuracy of everything so that I m using this and again will making prediction on x test and y test and let's just run this.

```
ada.score(x_train,y_train)#Training Score
```

```
0.8615500079001422
```

```
ada.score(x_test,y_test)#Testing Score
```

```
0.8609794628751974
```

So this you can say that on training accuracy score perming something around 86 and testing accuracy score also is 86 this is my AdaBoost Classifier.

Lets check with Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
```

```
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=i)
    rf.fit(x_train,y_train)
    x_pred=rf.predict(x_train)
    y_pred=rf.predict(x_test)
    print(f" At random state {i} , the testing accuracy is :- {accuracy_score(y_train,x_pred)}")
    print(f" At random state {i} , the testing accuracy is :- {accuracy_score(y_test,y_pred)}")
    print("\n")
```

Another algorithm use Random Forest classifier from the module from sk learn again dot n symbol they are under ensembl and i will import Random Forest Classifier then I create an onject for that rf in that I will store this random forest classifier then create and this I will use this rf this random forest classifier and I will fit I will fit x train and y train then so I

will write here create a object x pred what rf dot predict x train and also one more create object y pred test equal to rf dot predict x test print the accuracy of everything so that I m using this and again will making prediction on x test and y test and let's just run this.

```
rf.score(x_train,y_train)
```

```
0.9999604992889872
```

```
rf.score(x_test,y_test)
```

```
0.8628751974723539
```

So this you can say that on training accuracy score perming something around 99 and testing accuracy score is 86 this is my Random Forest Classiier.

Do Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
```

```
grid_param={
    'criterion':['ginni','entropy'],
    'max_depth': range(10,15),# The Maximum depth of the tree
    'min_samples_leaf':range(2,6),#The maximum number of samples required to be at a leaf node
    'min_samples_split':range(3,8),#The minimum numbers of sample required to split an internal node
    'max_leaf_nodes':range(5,10)}#Best nodes are defined as them unlimited number of leaf nodes
```

```
grid_search=GridSearchCV(estimator=rf,
                         param_grid=grid_param,
                         cv=5,
                        n_jobs=-1)
```

```
grid_search.fit(x_train,y_train)
```

```
: cnn.score(x_train,y_train)
: 0.8195212513825249

: cnn.score(x_test,y_test)
: 0.8243285939968404
```

Hyperparameter Tuning

Hyperparameter Tuning So this you can say that on training accuracy score perming something around 81 and testing accuracy score is 82 this is my Random Forest Classifier Hyperparameter Tuning.

6. Concluding Remarks.

Conclusion

```
loaded_model=pickle.load(open('Census','rb'))
result=loaded_model.score(x_test,y_test)
print(result*100)
```

```
86.2875197472354
```

Submitted by :

Yash Jaiswal

Batch No :

Ds0522