CSE 587: Data Intensive Computing Phase 1

Background:

The income gap between different categories has increased significantly in recent years. A variety of inputs are used to classify them into groups of <=50K or >50K. Many policies are recommended and people are instructed on how to enroll and ensure their futures for those earning less than $50K. Citizens welfare will be better understood thereby. A reliable model for detecting overpayment can be built and used to test whether an individual is overpaying or underpaying. Also, it can be used for marketing purposes, so specific advertisements can be sent to users according to their income level.

Problem Statement:

Using census data, we are trying to predict whether income exceeds $50k per year. An objective of this model is to identify the most critical factors that contribute to increasing an individual's income. Through such an analysis, important areas of income improvement can be identified.

Data Sources:

Dataset is taken from the below kaggle source.

1. https://www.kaggle.com/datasets/uciml/adult-census-income
2. https://archive.ics.uci.edu/ml/datasets/adult

This data in Kaggle is taken from Census bureau database 1994. There are 14 features and 32561 records in the input dataset. This is primarily a binary classification problem. The main goal is to predict whether the income of the person will be <=50k or >50k based on the 14 features available.

Below is the list of features and output variables present in the dataset.(0-13 are features and column 14 is the target variable).

```
#Reading the csv formatted dataset file
adultIncome = pd.read_csv('/content/AdultCensusIncome.csv')
```

```
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
```

Below is the shape of dataset. This shows there are 32561 rows or samples and 15 columns.

```
adultIncome.shape
```

```
(32561, 15)
```

Below is the size of dataset. Size command gives the number of (rows * columns) present in the input dataset.

```
adultIncome.size   #Size of the dataset(rows*columns)
```

```
488415
```

Below is the length of the dataset. len() gives the number of rows or samples available in the dataset.

```
len(adultIncome)   #Number of rows/records of the dataset
```

```
32561
```

Below are different statistics of dataset. describe() method outputs the count, mean, standard deviation, minimum, maximum of the input data.

```
adultIncome.describe()     #describe method provides the necessary mean, std, min, max... of the numerical variables
```

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

Categorical features in the dataset:

```
print(categorical.dtypes)
```

```
workclass         object
education         object
marital.status    object
occupation        object
relationship      object
race              object
sex               object
native.country    object
income            object
dtype: object
```

Numerical features in the dataset:

```
print(num.dtypes)
```

```
age               int64
fnlwgt            int64
education.num     int64
capital.gain      int64
capital.loss      int64
hours.per.week    int64
dtype: object
```

Data Cleaning/Processing:

1. **Checking for null values:**
   We checked if there as any null values present in the dataset. The output can be seen below and we can observe that there are no null values in the dataset.

```
adultIncome.isnull().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

2. **Checking missing values:**
   We checked if there are any missing values present and got the below counts for various features. "?" value star present in occupation, workclass and native.country.

```python
adultIncome.isin(['?']).sum()
```

```
age                  0
workclass         1836
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     583
income               0
dtype: int64
```

Value_counts () gives the count of different types of workclasses present in the dataset. We can see that in “workclass”, there are 1836 samples with “?” values to handle.

```python
adultIncome['workclass'].value_counts()
```

```
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

Value_counts () gives the count of different types of occupations present in the dataset. We can see that in “occupation”, there are 1843 samples with “?” values to handle.

```python
adultIncome['occupation'].value_counts()
```

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: occupation, dtype: int64
```

Value_counts () gives the count of different types of native.country present in the dataset. We can see that in "native.country", there are 583 samples with "?" values to handle.

```python
adultIncome['native.country'].value_counts()
```

```
United-States                 29170
Mexico                          643
?                               583
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
```

Handling missing values:
Replacing all the “?” values in the entire dataset with “NaN”.

```
[ ]  adultIncome.replace("?", np.NaN, inplace = True)
```

Now we can see that all the ‘?’ values are replaced with Null as shown below.

```
adultIncome.isnull().sum()
```

```
age                  0
workclass         1836
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     583
income               0
dtype: int64
```

3. Handing missing data in the ‘workclass’ feature:

Replacing null values in the workclass with the mode by using fillna() method.

```
workMode = adultIncome['workclass'].mode()[0]
adultIncome['workclass'].fillna(workMode,inplace=True)
```

4. Handing missing data in the ‘occupation’ feature:

Using the below method, replacing all the “NaN” values in occupation field with “Other”

```
adultIncome.occupation.fillna("Other", inplace = True)
```

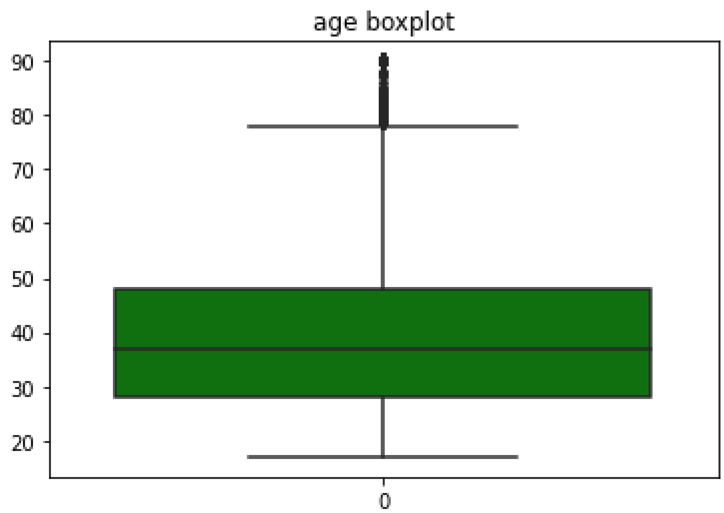
5. **Handling 'native.country' missing data:**

Using below method, Replaced all the "NaN" values of 'native.country' with "Unknown" value.

```
adultIncome["native.country"].fillna("Unknown", inplace = True)
```

6. **Outliers in the age feature:**

From the below boxplot, we can see that there are outliers in the 'age' feature. Age values greater than 80 is considered as outliers.



Using the Inter Quartile range, replaced the age outliers with corresponding median values.

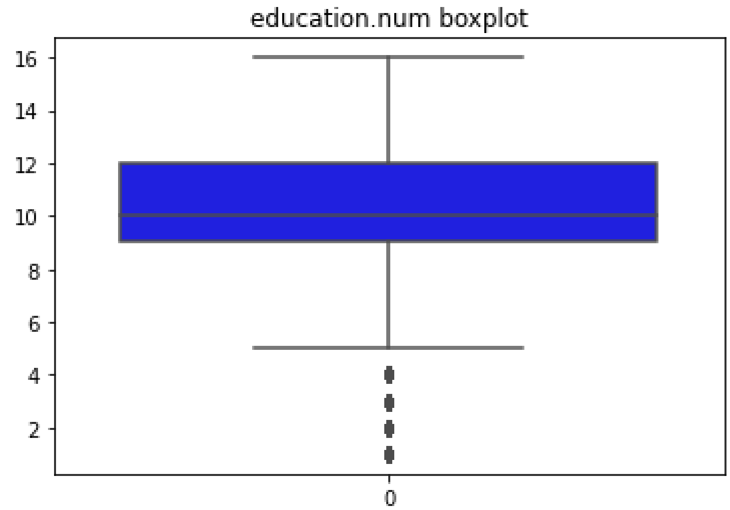
```
#Using inter Quartile range
Q1 = adultIncome['age'].quantile(0.25)
Q3 = adultIncome['age'].quantile(0.75)
upperLimit = Q3 + 1.5 * (Q3 - Q1)
lowerLimit = Q1 - 1.5 * (Q3 - Q1)
```

+ Code + Text

```
adultIncome.loc[(adultIncome['age'] < lowerLimit), 'age'] = adultIncome['age'].median()
adultIncome.loc[(adultIncome['age'] > upperLimit), 'age'] = adultIncome['age'].median()
```

7. **Outliers in the education.num feature:**

From the below boxplot we can see that there are outliers whose values range below 5 in the education.num field.



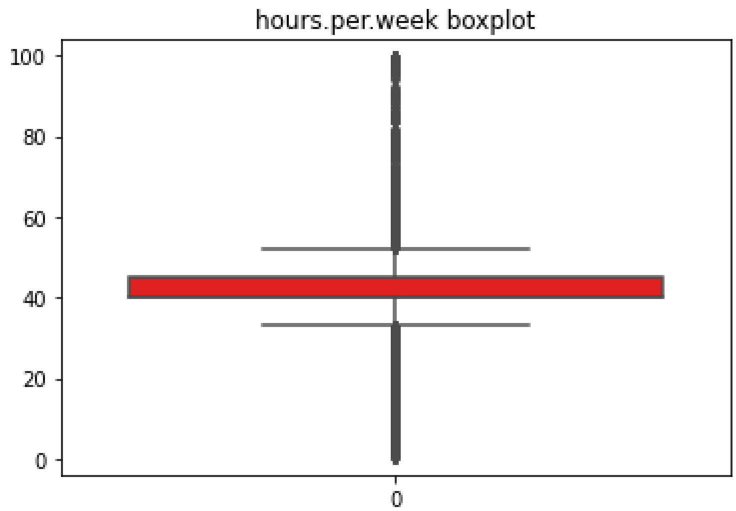
Using the Inter Quartile Range, replaced all the outliers with the corresponding median values.

```
Q1 = adultIncome['education.num'].quantile(0.25)
Q3= adultIncome['education.num'].quantile(0.75)
upperLimit = Q3 + 1.5 * (Q3 - Q1)
lowerLimit = Q1 - 1.5 * (Q3 - Q1)
```

```
adultIncome.loc[(adultIncome['education.num'] < lowerLimit), 'education.num'] = adultIncome['education.num'].median()
adultIncome.loc[(adultIncome['education.num'] > upperLimit), 'education.num'] = adultIncome['education.num'].median()
```

8. Outliers in hours.per.week feature:

From the below boxplot we can see that there are many outliers in the hours.per.week feature. Values which are above 50, below 30 are considered as outliers.



Using the Inter Quartile range, replaced the extreme values with the corresponding mean value.

```
Q1 = adultIncome['hours.per.week'].quantile(0.25)
Q3 = adultIncome['hours.per.week'].quantile(0.75)
upperLimit = Q3 + 1.5 * (Q3 - Q1)
lowerLimit = Q1 - 1.5 * (Q3 - Q1)
```

```
adultIncome.loc[(adultIncome['hours.per.week'] < lowerLimit), 'hours.per.week'] = adultIncome['hours.per.week'].mean()
adultIncome.loc[(adultIncome['hours.per.week'] > upperLimit), 'hours.per.week'] = adultIncome['hours.per.week'].mean()
```

9. Dropping 'education' feature

The feature education.num is the numerical mapping of the feature education. Thus, dropping the education feature as to reduce the redundancy in the dataset.

```
    education  education.num
      HS-grad              9
      HS-grad              9
 Some-college             10
      7th-8th              4
 Some-college             10
          ...            ...
 Some-college             10
   Assoc-acdm             12
      HS-grad              9
      HS-grad              9
      HS-grad              9
```

```
[ ] # Dropping education column as there is education.num column which is providing the same data
    del adultIncome['education']
```

10. Dropping ‘fnlwgt’ feature

The feature fnlwgt indicates the final weight. It is highly varying and discrete. It might not contribute much to the modeling of the data. Thus, dropping the attribute.

```
adultIncome['fnlwgt'].value_counts()
```

```
164190    13
203488    13
123011    13
113364    12
121124    12
          ..
183522     1
44419      1
442612     1
374833     1
257302     1
Name: fnlwgt, Length: 21648, dtype: int64
```

```
[ ] #Dropping fnlwgt as the values are highly varying and discrete
    adultIncome.drop('fnlwgt',axis=1, inplace=True)
```

11. Mapping ‘Income’(target) variable:

Using the below replace method, mapped the output variable.
Given income group <=50K as “0” and >50K as “1”.

```
adultIncome['income']=adultIncome['income'].replace('<=50K',0)
adultIncome['income']=adultIncome['income'].replace('>50K',1)
```

Handling Categorical features:

```
categorical = adultIncome.select_dtypes(include=['object'])
print(categorical.columns)
```

```
Index(['workclass', 'education', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'native.country', 'income'],
      dtype='object')
```

```
print(categorical.dtypes)
```

```
workclass         object
education         object
marital.status    object
occupation        object
relationship      object
race              object
sex               object
native.country    object
income            object
dtype: object
```

12. Handling 'Sex' feature:

```
adultIncome['sex'] = labEncoder.fit_transform(adultIncome['sex'])
sex_map = dict(zip(labEncoder.classes_, labEncoder.transform(labEncoder.classes_)))
print(sex_map)
```

```
{'Female': 0, 'Male': 1}
```

13. Handling 'race' feature:

```
adultIncome['race'] = labEncoder.fit_transform(adultIncome['race'])
race_map = dict(zip(labEncoder.classes_, labEncoder.transform(labEncoder.classes_)))
print(race_map)
```

```
{'Amer-Indian-Eskimo': 0, 'Asian-Pac-Islander': 1, 'Black': 2, 'Other': 3, 'White': 4}
```

14. Handling 'relationship' feature:

```
adultIncome['relationship'] = labEncoder.fit_transform(adultIncome['relationship'])
relation_map = dict(zip(labEncoder.classes_, labEncoder.transform(labEncoder.classes_)))
print(relation_map)
```

```
{'Husband': 0, 'Not-in-family': 1, 'Other-relative': 2, 'Own-child': 3, 'Unmarried': 4, 'Wife': 5}
```

15. Handling 'Occupation' feature:

Adm-clerical: 0
Armed-Forces: 1
Craft-repair: 2
Exec-managerial: 3
Farming-fishing: 4
Handlers-cleaners: 5
Machine-op-inspct: 6
Other: 7
Other-service: 8
Priv-house-serv: 9
Prof-specialty: 10
Protective-serv: 11
Sales: 12
Tech-support: 13
Transport-moving': 14

```
adultIncome['occupation'] = labEncoder.fit_transform(adultIncome['occupation'])
occupation_map = dict(zip(labEncoder.classes_, labEncoder.transform(labEncoder.classes_)))
print(occupation_map)
```

```
{'Adm-clerical': 0, 'Armed-Forces': 1, 'Craft-repair': 2, 'Exec-managerial': 3, 'Farming-fishing': 4,
```

16. **Handling 'marital.status' feature:**

```
adultIncome['marital.status'] = labEncoder.fit_transform(adultIncome['marital.status'])
marital_map = dict(zip(labEncoder.classes_, labEncoder.transform(labEncoder.classes_)))
print(marital_map)
```

```
{'Divorced': 0, 'Married-AF-spouse': 1, 'Married-civ-spouse': 2, 'Married-spouse-absent': 3, 'Never-married': 4, 'Separated': 5, 'Widowed': 6}
```

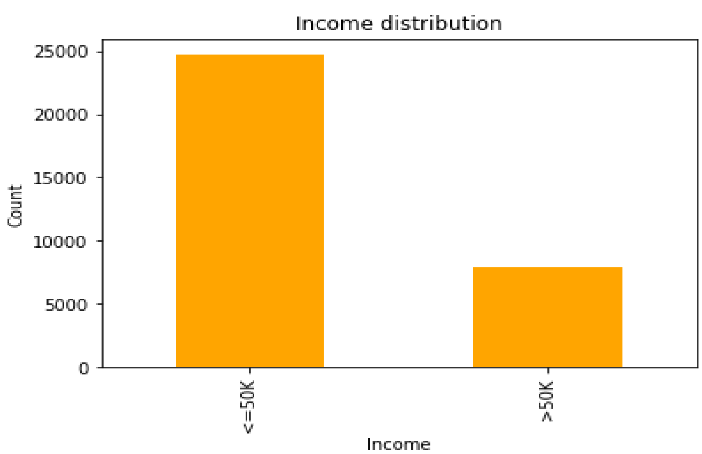
17. **Handling 'workclass 'feature:**

```python
adultIncome['workclass'] = labEncoder.fit_transform(adultIncome['workclass'])
workclass_map = dict(zip(labEncoder.classes_, labEncoder.transform(labEncoder.classes_)))
print(workclass_map)
```

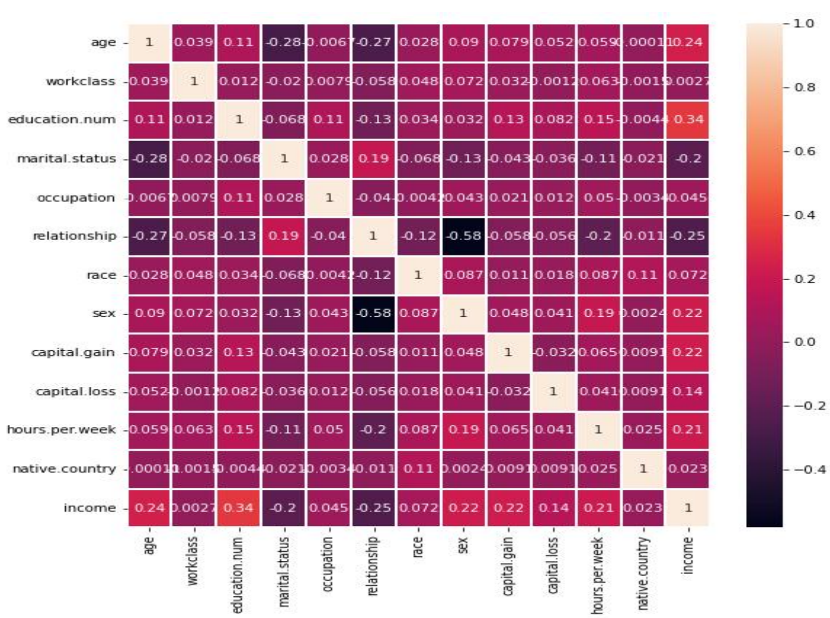
```
{'Federal-gov': 0, 'Local-gov': 1, 'Never-worked': 2, 'Private': 3, 'Self-emp-inc': 4, 'Self-emp-not-inc': 5, 'State-gov': 6, 'Without-pay': 7}
```

Exploratory Data Analysis (EDA):

1. **Bar Graph of Income feature:**
We can observe that the number of people earning less than or equal to 50K are more than twice the number of people earning greater than 50K. From the graph we can see that there are around 25000 people earning <=50K. On the other hand there are only around 9000 people earning >50K.



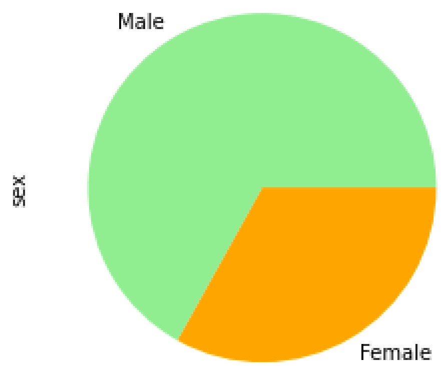
2. **HeatMap with Anootations**
Below is the correlation matrix of the features of the dataset.



As we see in the above correlation matrix, the features, income,hours.per.week; income, education.num; age, income are correlated.

3. **Pie Chart**

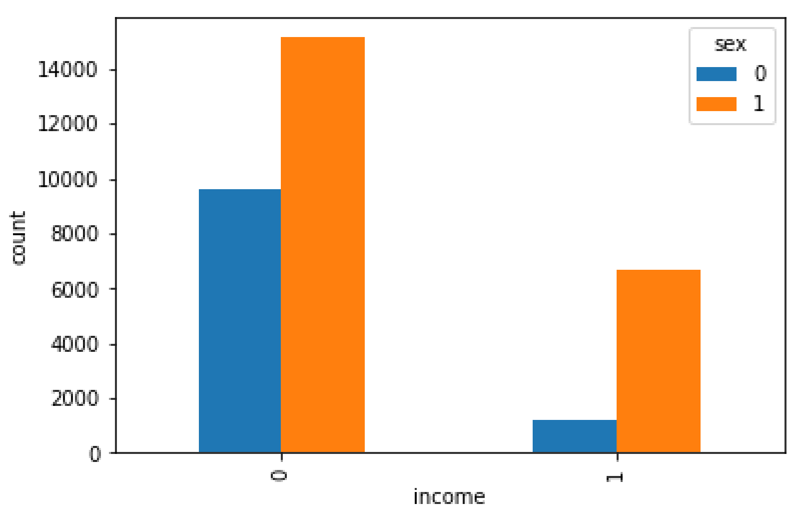
The piechart of male and female present in the dataset.



We can see that the male value has the dominant count in the sex feature.

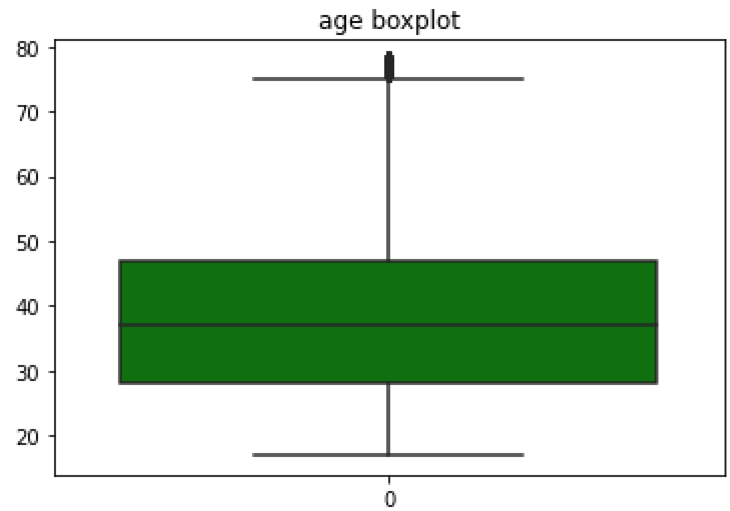
4. **Crosstab Bar graph**

We can see that overall number of females earning >50K is very less compared to number of males earning >50K. '0' of blue bar represents female and '1' of orange bar represents male. Income in the x-axis corresponds: 0 :<=50K , 1 :>50K

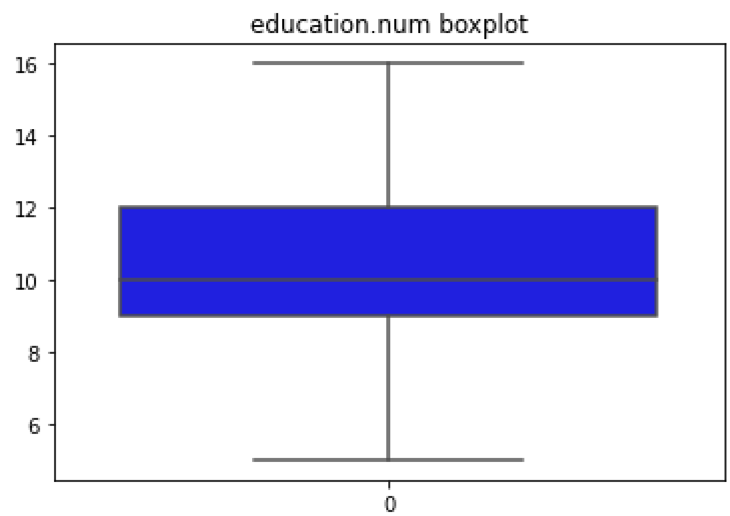


5. **Boxplot of age:**

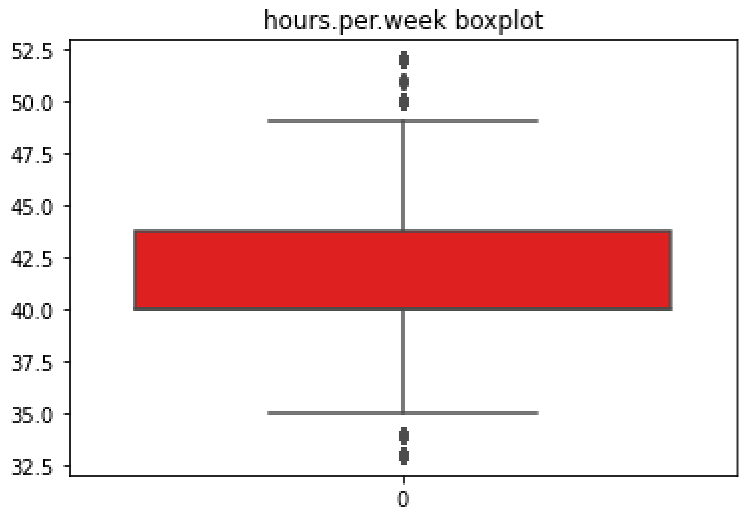
Below is age boxplot after removing outliers. Before doing pre-processing and removing outliers, there were outliers whose age is >80. Since preprocessing is done in the below boxplot we can see that outliers are reduced and all age feature values fall under 80.



6. **Boxplot of education.num**
Below is the boxplot of education.num after removing outliers. In the preprocessing step we can see that we had outliers with education.num values less than 5. Since pre-processing has been done and applied correctly in the below boxplot we can see that there are no outliers and all the values are between 6 and 16.

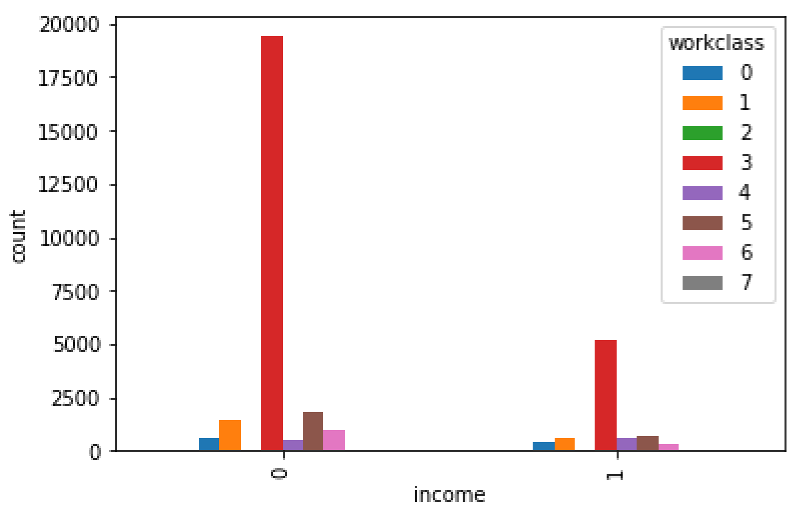


7. **Boxplot of hours.per.week:**
Below is the boxplot hours.per.week after replacing outliers with mean value. Before removing outliers we had outliers whose values are <30 or >50. After removing reasonable amount of outliers we can see that in the below boxplot many values fall under 35 and 50.



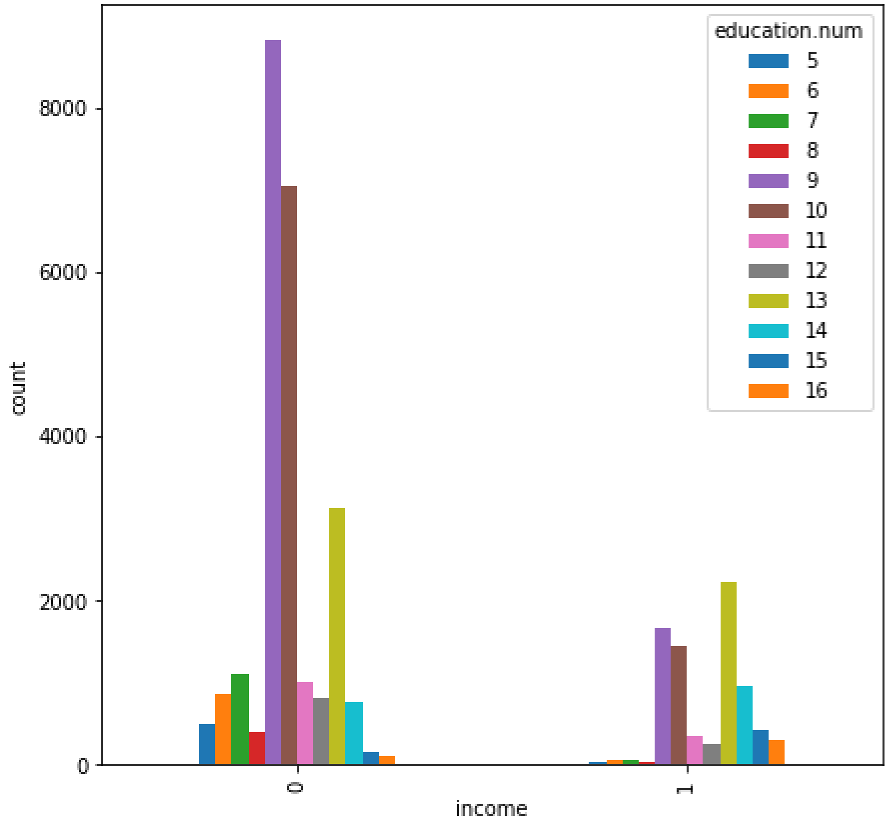
8. **Crosstab Bar Graph of Income -Workclass**

   Below is the plot of counts of different workclasses belonging to different income groups. From the graph we can see that there are many people in workclass 3 which is “private” irrespective of whether they earn <50K or >50K and almost negligible number of people belong to workclass 2 which is “Never-worked” irrespective of whether they earn <=50K or >50K.



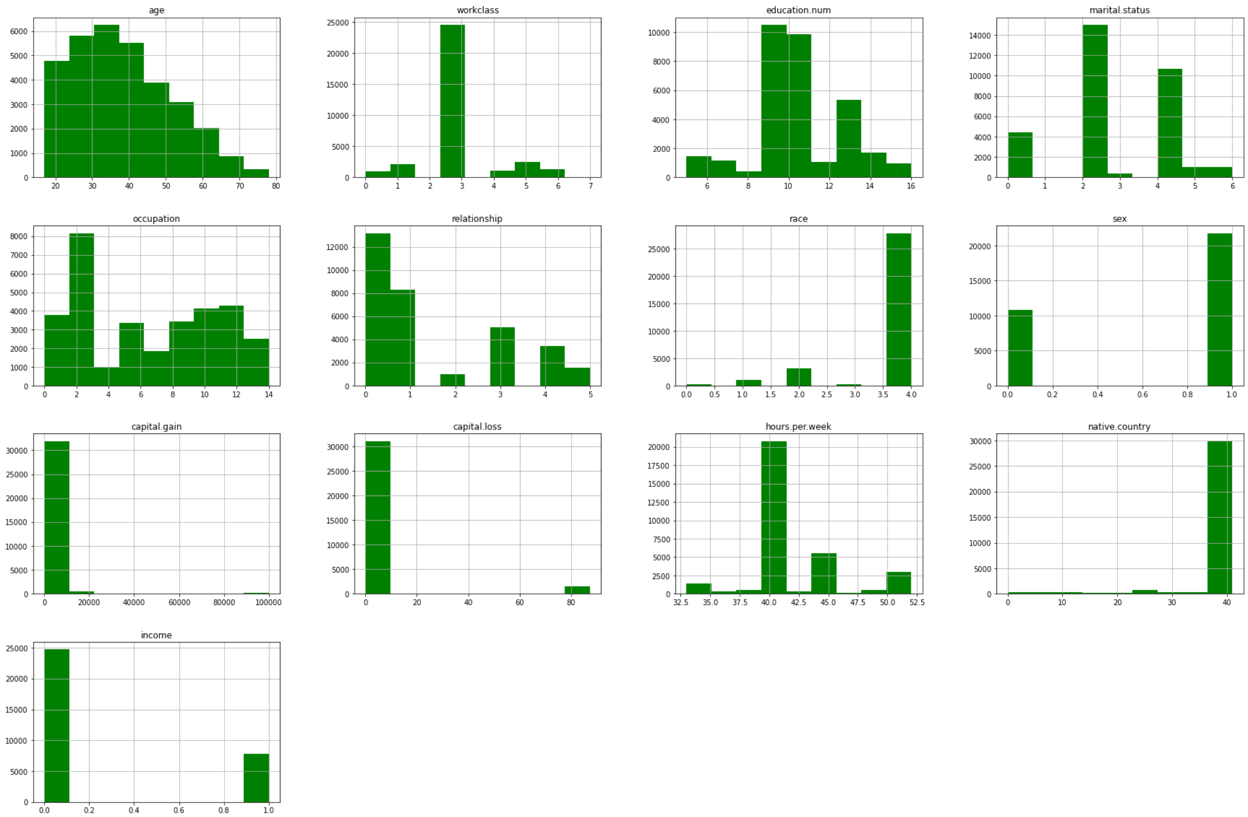
9. **Crosstab Bar Graph of Income-Education.num**

   Below is the plot of count of people belonging to different education.nums in different income groups. We can see that highest number of people with education.num value 9 belong to <=50K income group and highest number of people earning >50K belong to education.num value 13.



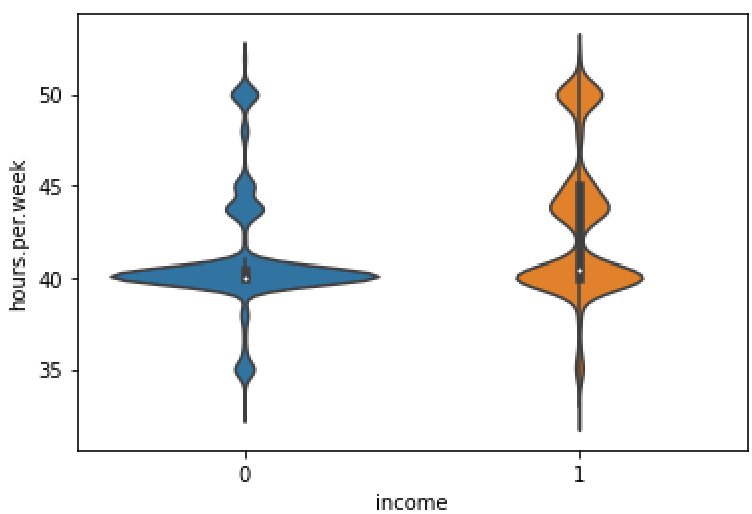
10. Histogram

Below is the histogram plot of features. From age plot we can see that there are many people between 10 to 45. In the workclass, we can see that there are many people belonging to workclass 3. In education.num we can see that there are many people between 9 and 11 values. In capital gain, most of the values are between 0 and 10000. In capital loss, most o the values are between 0 and 10. From hours.per.week we can see that most of the people work around 40 hours.



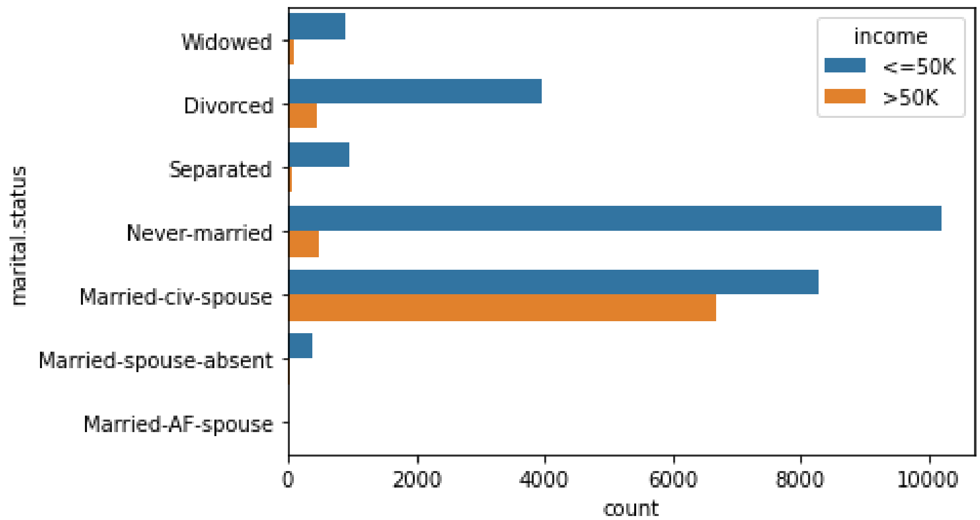
11. Violon plot of income-hours.per.week

Below is the distribution of income vs hours.per.week. In the below plot income 0 corresponds to people earning <=50K and income 1 corresponds to people earning >50K. We can see that irrespective of income groups there are many people working around 40 hours per week.



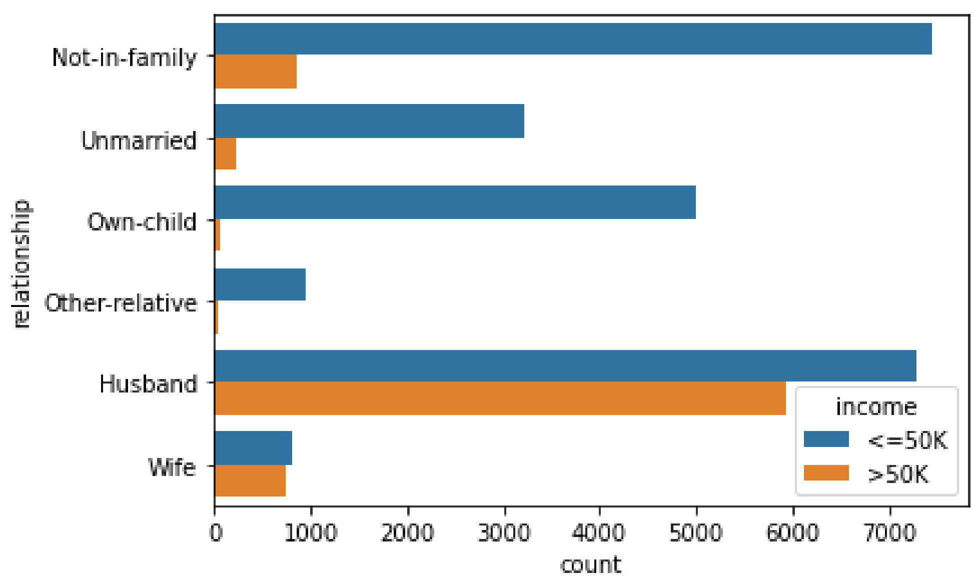
12. Countplot of 'marital.status' feature

Below is the plot of count of different marital status people belonging to different income groups. We can see that the count of "Married-AF-spouse" and "Married-spouse-absent" earning >50K is zero. Majority of people earning <=50K beong to never-married followed by married-civ-spouse category.



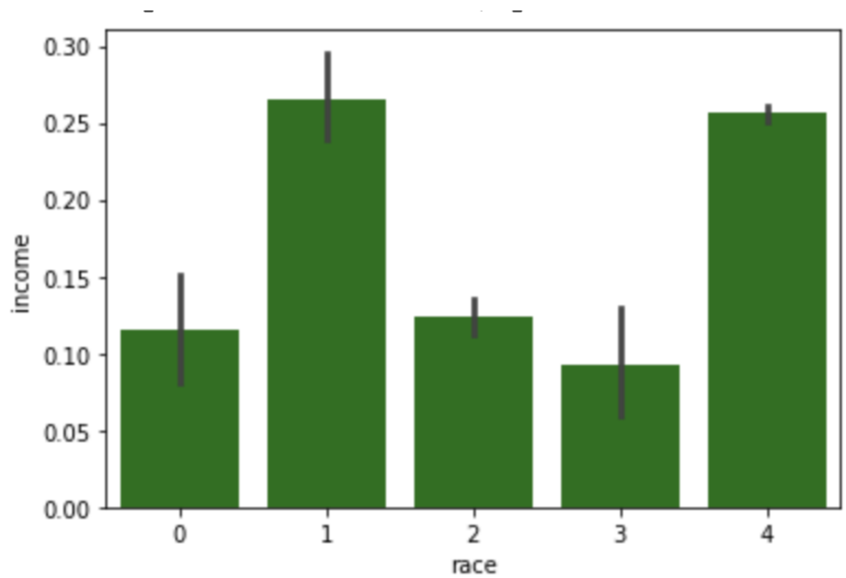
13. Countplot of 'relationship' feature

From the below plot we can see that majority of people earning <=50K belong to Not-in-family, Unmarried, Own-child or Husband category. We can also observe that the count of wives who are earning <=50K and >50K are almost equal.



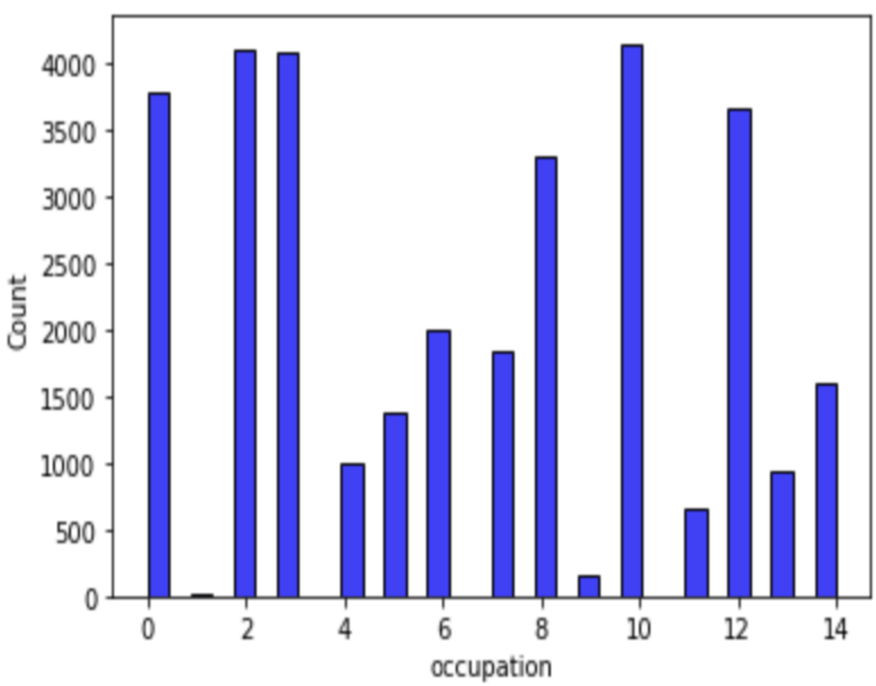
14. Barplot of Income-Race:

Below is the barplot of race vs income. Majority of the people belong to race '1' and '4'.



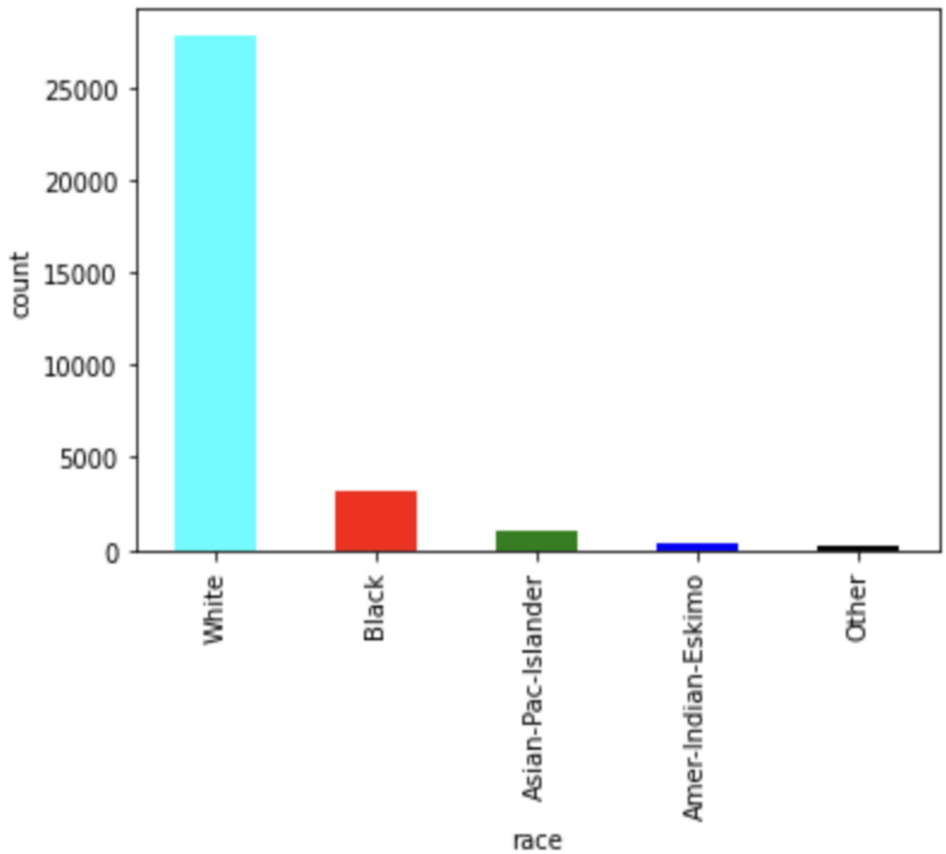
15. Histogram plot of 'occupation' feature:

Occupation classes 1 and 9 occur rarely in the dataset. Occupations 0,2,3,10 are predominant in the dataset.



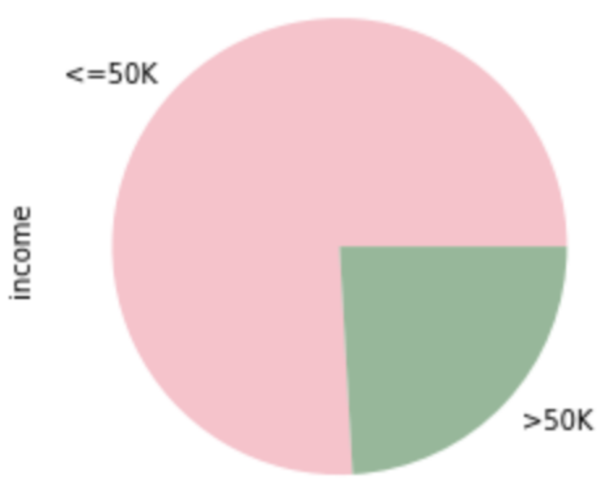
16. Barplot of ‘race’feature:

Below barplot shows the count of people belonging to various types of race. We can observe that white race is the most dominant one in the input dataset.



17. PieChart of ‘Income’(output) feature:

From the piechart we can see that the number of people earning <=50K is almost 3 times the number of people earning >50K



Name	**UBIT Name**	**UB Email**
Viditha Wudaru	vidithaw	vidithaw@buffalo.edu
Jayavani Akkiraju	jayavani	jayavani@buffalo.edu

References:

1. https://medium.com/data-warriors/eda-of-adult-census-income-dataset-cc9ac1a3d552
2. https://archive.ics.uci.edu/ml/datasets/adult
3. https://www.kaggle.com/datasets/uciml/adult-census-income
4. https://machinelearningmastery.com/imbalanced-classification-with-the-adult-income-dataset/
5. https://medium.com/analytics-vidhya/adult-census-income-dataset-using-multiple-machine-learning-models-f289c960005d
6. https://johdev.com/machine%20learning/2020/03/24/End-to-End-Data-Science-Project-with-Adult-Income-Dataset.html
7. https://pandas.pydata.org/
8. https://www.kaggle.com/code/vivymishra/eda-and-model-building
9. https://www.w3schools.com/python/matplotlib_pie_charts.asp
10. https://pandas.pydata.org/docs/user_guide/visualization.html
11. https://seaborn.pydata.org/generated/seaborn.histplot.html
12. https://stackoverflow.com/questions/42196589/any-way-to-get-mappings-of-a-label-encoder-in-python-pandas
13. https://pbpython.com/pandas-crosstab.html
14. https://www.statology.org/seaborn-heatmap-size/
15. https://godatadrive.com/blog/how-to-detect-outliers#:~:text=In%20a%20boxplot%2C%20the%20IQR,data%2C%20which%20helps%20identify%20outliers.
16. https://arxiv.org/pdf/1810.10076.pdf