**CENSUS INCOME PROJECT REPORT**

**By SREEKARI.I**

INTRODUCTION

Over the last few decades, humans have grown a lot of dependence on data and information in society and with this advent growth, technologies have evolved for their storage, analysis and processing on a huge scale. The fields of Data Mining and Machine Learning have not only exploited them for knowledge and discovery but also to explore certain hidden patterns and concepts which led to the prediction of future events, not easy to obtain. The problem of income inequality has been of great concern in the recent years. Making the poor better off does not seem to be the sole criteria to be in quest for eradicating this issue. People of the United States believe that the advent of economic inequality is unacceptable and demands a fair share of wealth in the society. This model actually aims to conduct a comprehensive analysis to highlight the key factors that are necessary in improving an individual's income. Such an analysis helps to set focus on the important areas which can significantly improve the income levels of individuals.

PROBLEM STATEMENT

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.

Description of **fnlwgt (final weight)**  
  
The weights on the Current Population Survey (CPS) files are controlled to independent estimates of the civilian non-institutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau. We use 3 sets of controls. These are:

1. A single cell estimate of the population 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.

UNDERSTANDING THE DATASET

There are  32560 rows and 15 columns in the dataset. The column name are 'Age', 'Workclass', 'Fnlwgt', 'Education', 'Education\_num','Marital\_status', 'occupation', 'Relationship', 'Race', 'Sex', 'Capital\_gain', 'Capital\_loss', 'Hours\_per\_week', 'Native\_country', 'Income’.

Here, Income is the target variable and there are 14 columns other than the target column. These 14 columns consists of f 8 categorical and 6 continuous attributes containing information on age, education, nationality, marital status, relationship status, occupation, work classification, gender, race, working hours per week, capital loss and capital gain.

The target variable, Income is a binomial label in the dataset that predicts whether a person earns more than 50 Thousand Dollars per year or not based on the given set of attributes.

EXPLORATORY DATA ANALYSIS

Let us check the datatype and count of the attributes(columns) of the dataset. Here, we can see if all the values are present or not.

df.info()

(Note: I have loaded the dataset in the df.)

Output:

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Age 32560 non-null int64

1 Workclass 32560 non-null object

2 Fnlwgt 32560 non-null int64

3 Education 32560 non-null object

4 Education\_num 32560 non-null int64

5 Marital\_status 32560 non-null object

6 Occupation 32560 non-null object

7 Relationship 32560 non-null object

8 Race 32560 non-null object

9 Sex 32560 non-null object

10 Capital\_gain 32560 non-null int64

11 Capital\_loss 32560 non-null int64

12 Hours\_per\_week 32560 non-null int64

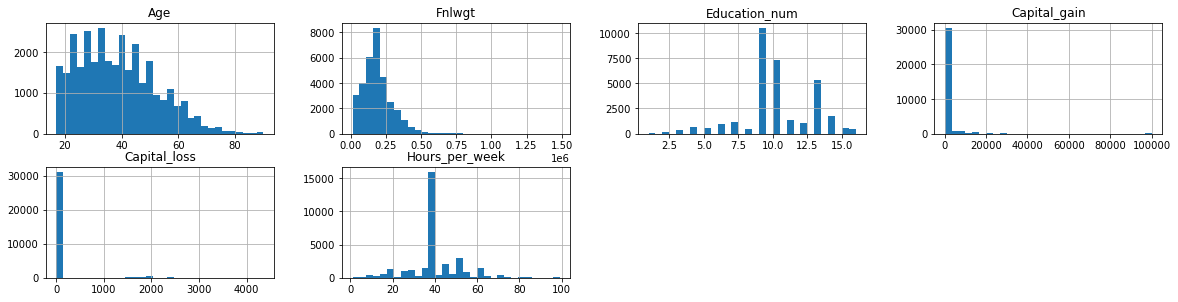
13 Native\_country 32560 non-null object

14 Income 32560 non-null object

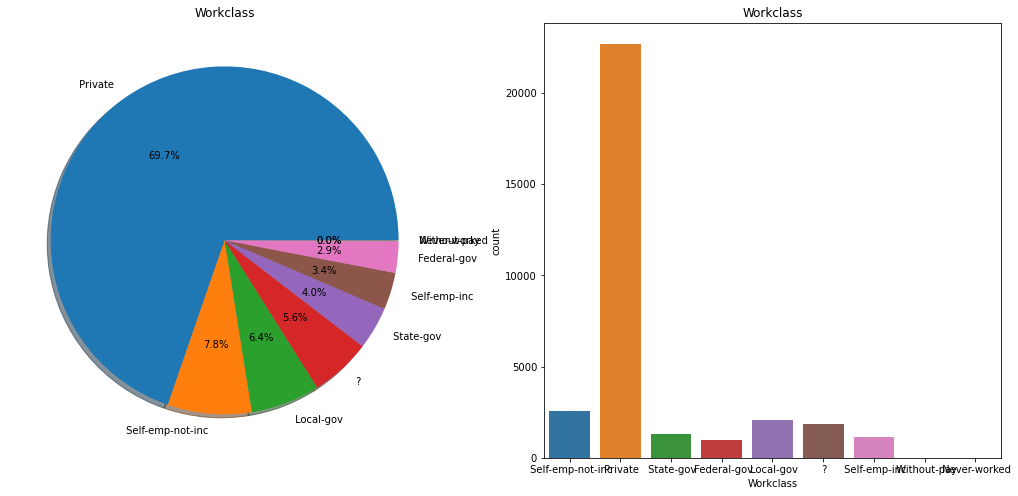
dtypes: int64 (6), object (9)

memory usage: 3.7+ MB

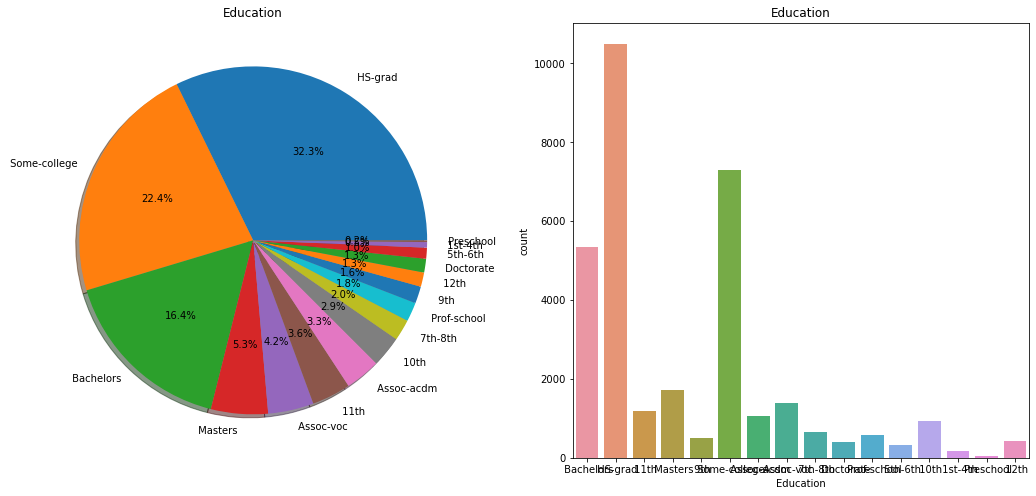
We can see that there are no null values and there are 6 attributes of int64 datatype and 9 attributes are of object datatype.



Histogram visualisation for each attribute showing what kind of distribution it is. This shows the distribution of attributes with numerical columns only.

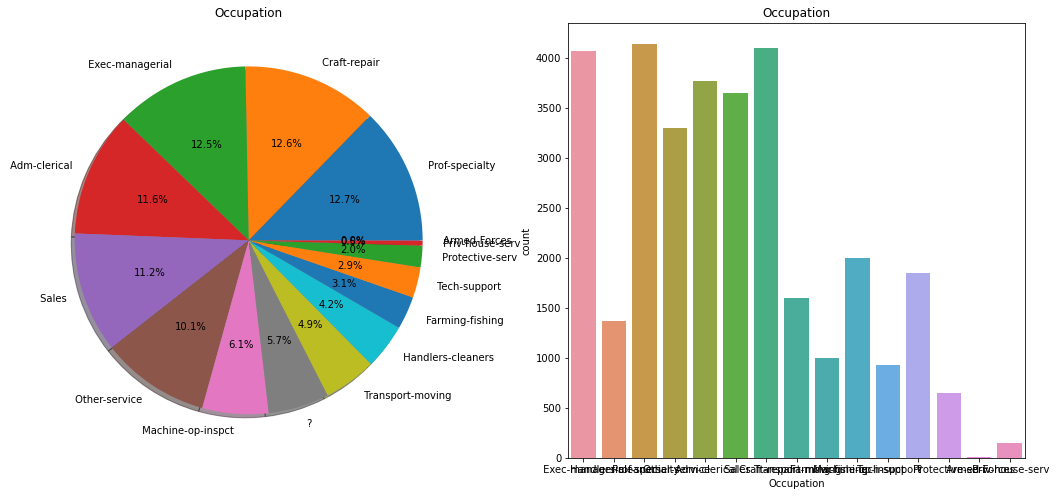


The workclass in the given dataset is divided into nine groups. 69.7% people work in Private organisations and the distribution in the remaining groups is almost the same. There are 7 people who have never worked and we do not have information of workclass of 1836 people.



The above pie-chart and barplot shows the Education level of the people of the collected dataset. The percentage of people with HS-grad, Some-college and

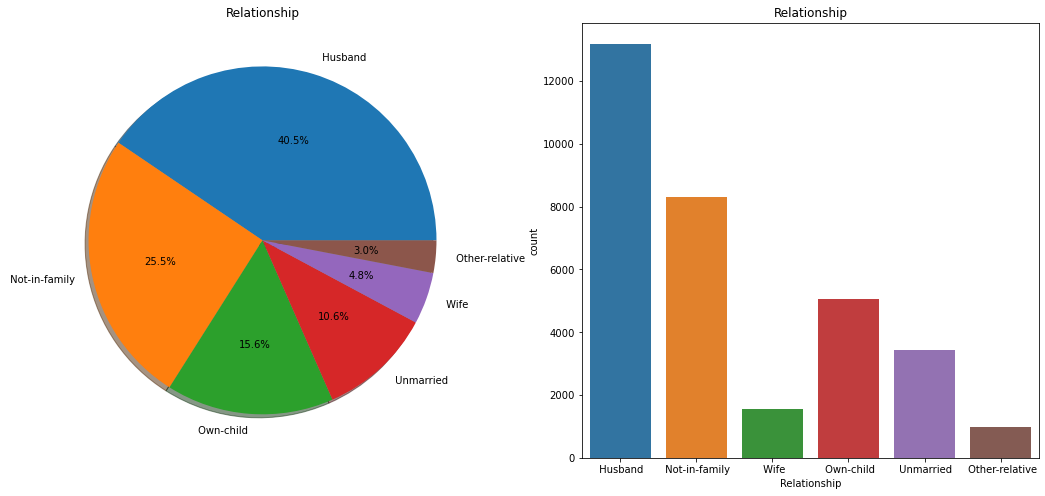
Bachelors are highest when compared to the rest while the people with only preschool education being the least.



The Occupation attribute has 15 unique categories. The people with

Prof-specialty is highest with 4140 people while the least is in Armed-Forces

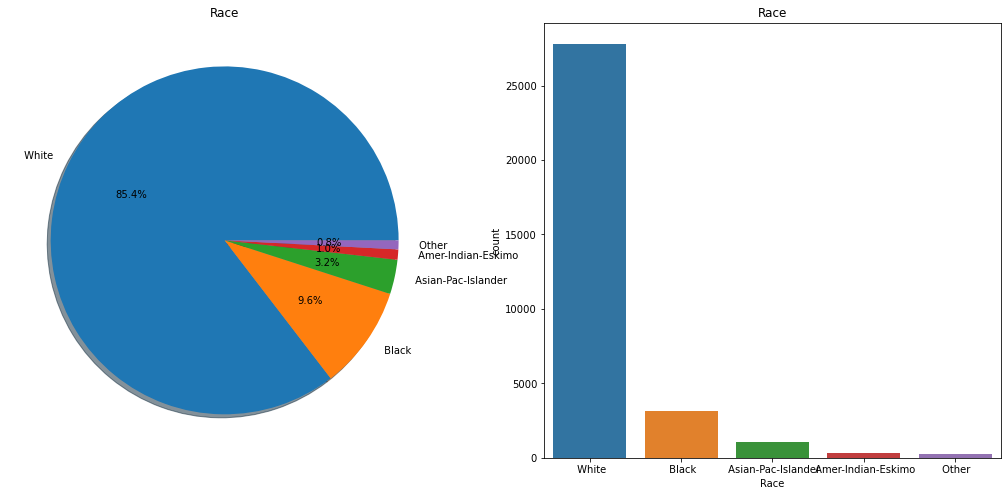
with 9 people.



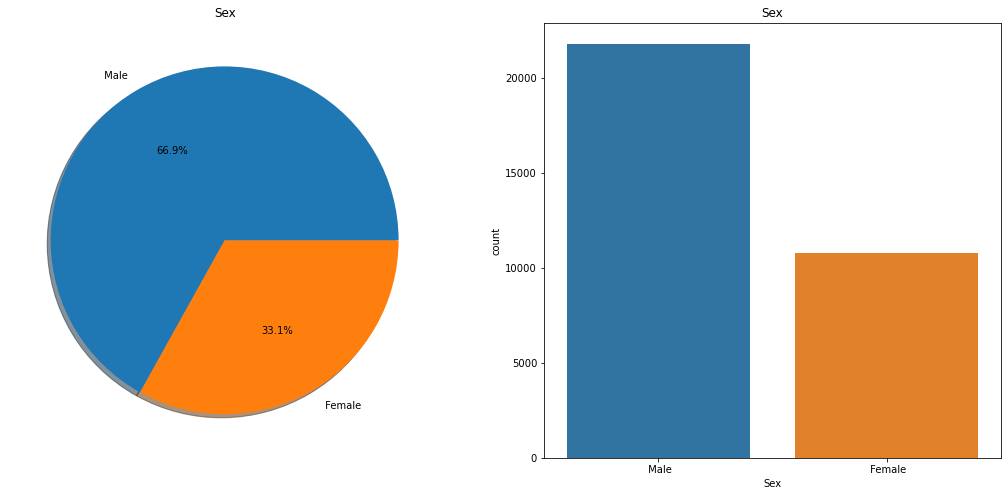
Upon checking the relationship status, we can see that the percentage of

Husband is the highest followed by Not-in-family while the least is Other-

relative.



In the surveyed population, there are more people of White Race followed by Black Race while the number of people of Other Race being the least.

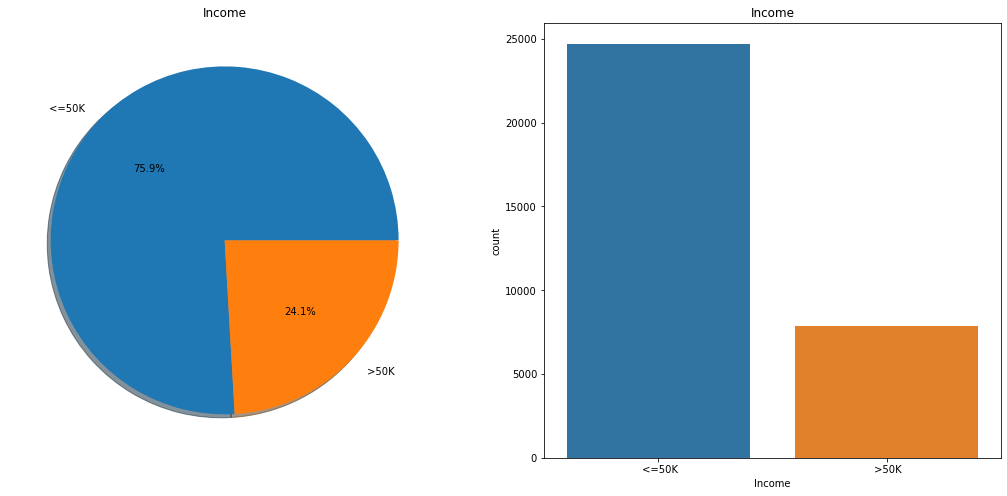


There are 66.9% Male population and the female polulation is 33.1% in the survey

Now, let us check the population in the survey based on the native country.

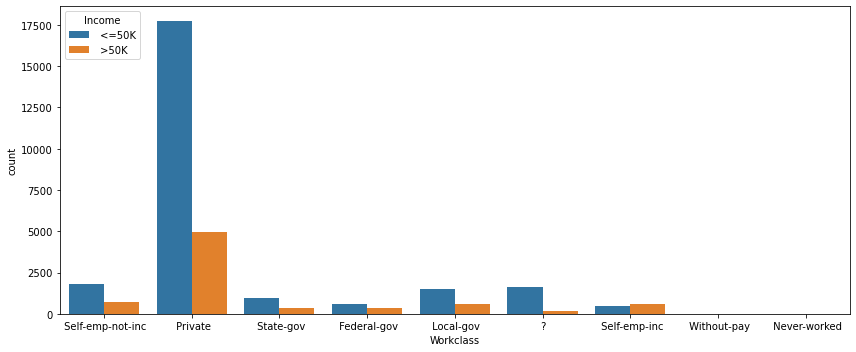


We can see that there are more people with Native\_country 'United-States' in the survey. The people with native country other than United States is very less.

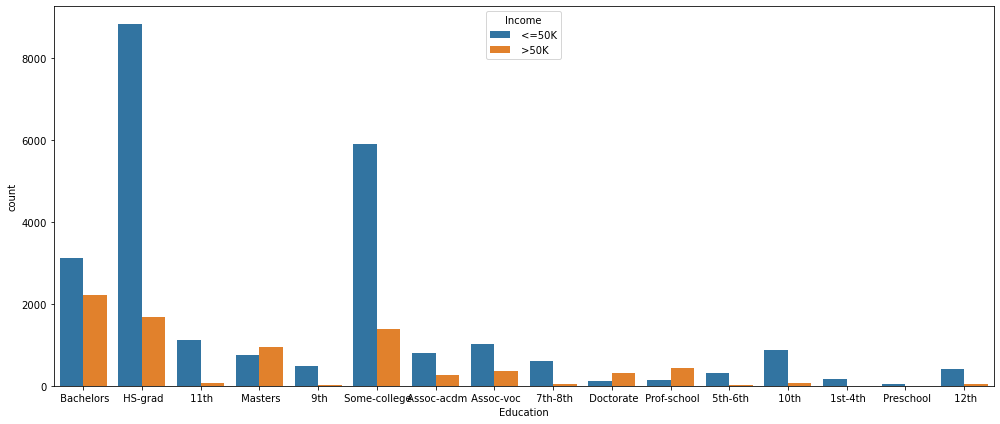


There are 75.9% population with Income less than or equal to 50K and 24.1% population with income greater than 50K..

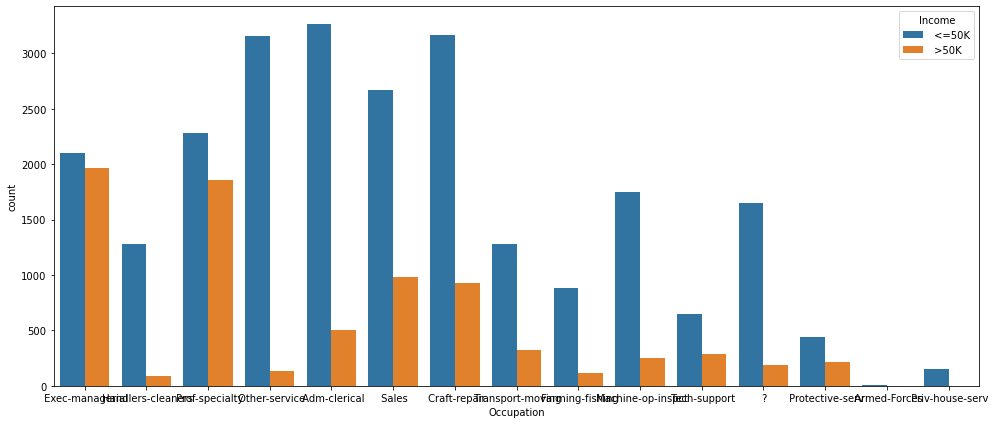
Now, I am comparing how the Categorical data varies with the target variable.



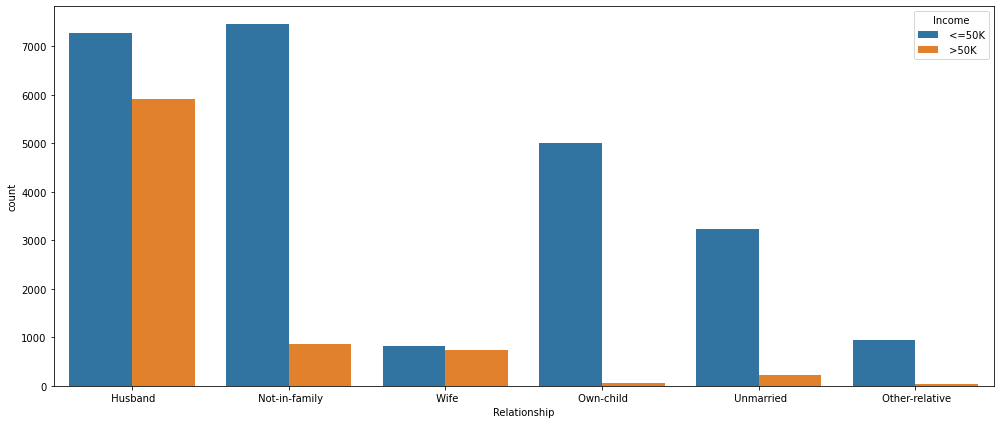
We can see the distribution of Income in various classes of Workclass here.



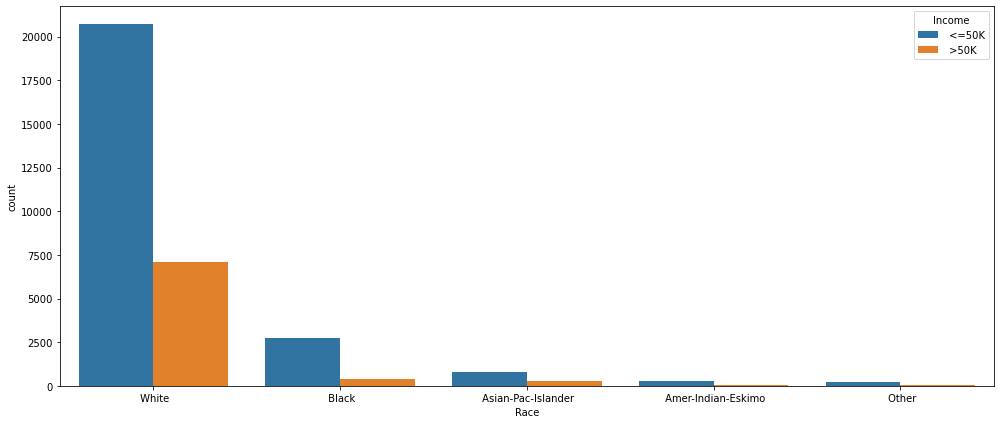
We can see the distribution of Income based on various fields of Education of population here.



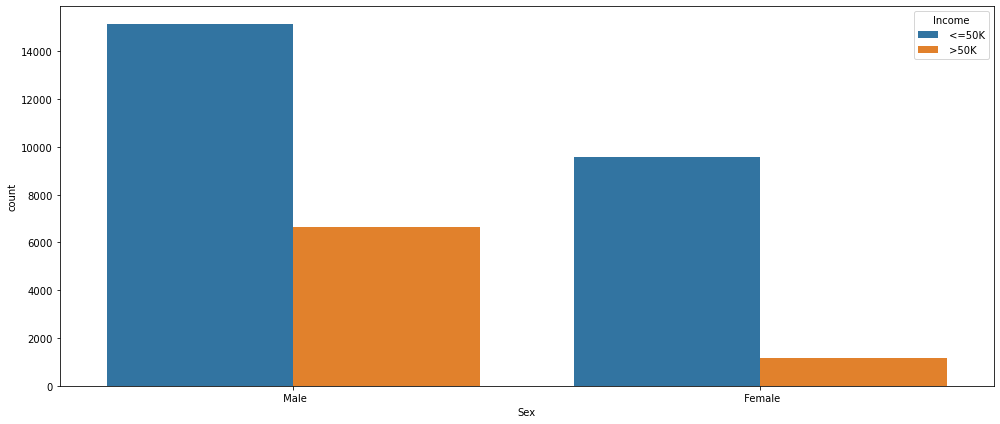
We can see the distribution of Income in various Occupations here.



We can see the distribution of Income based on the Relationship status here.



We can see the distribution of Income in different Races here.



We can see the distribution of Income based on Sex here.



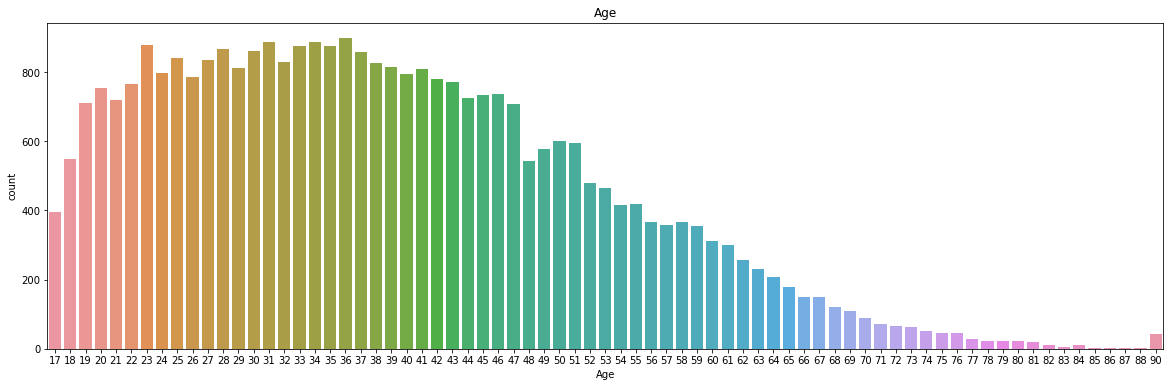
We can see the distribution of Income based on Native\_country here.

DATA PRE-PROCESSING

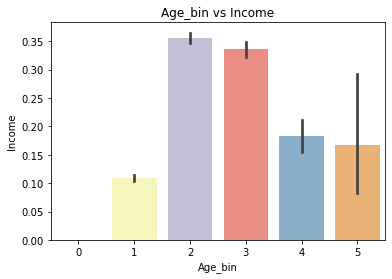
Encoding of Categorical or Non-Numeric features: As all Categorical Features are non-numeric, encoding has been done in 2 stages:

Label Encoding: All categorical features are label encoded, where alphabetically each category is assigned numbers starting from 0. This is also done before running the Extra Trees Classifier Algorithm for efficient feature selection.

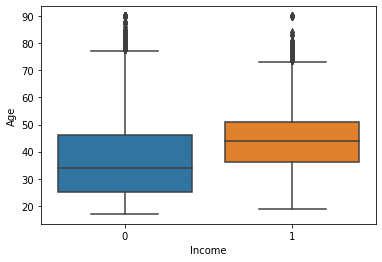
One-Hot Encoding: This involves splitting of different categorical features into its own categories where each and every category assumes a binary value i.e., 0 if it does not belong to that category and 1 if it belongs to that category. This is important for those categorical features where there exists no ordinal relationship in between them. One-Hot Encoding has been done for categorical features having more than 2 categories. Here, for all categorical features except sex attribute, all label encoded forms are transformed into One-Hot Encoded Forms. This is because sex attribute has only 2 categories i.e., male and female, which have been already represented in binary form in a single attribute and hence to avoid the curse of dimensionality, no One-Hot Encoding is done for sex attribute.

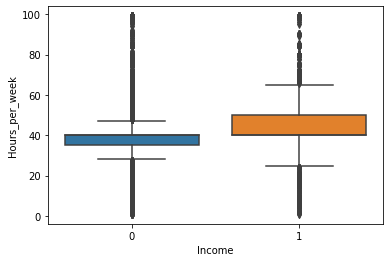


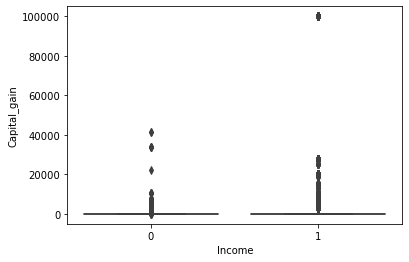
We can see that there are 73 unique values in Age. The maximum age is 90 and the minimum age is 17. So lets divide the range from 0-90 into 6 bins with size 17.

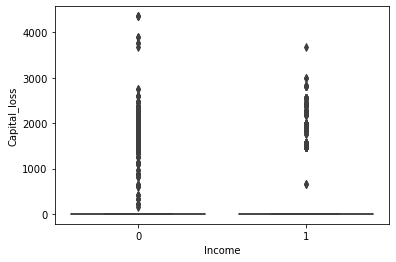


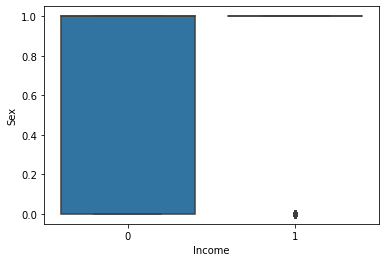
Outliers:

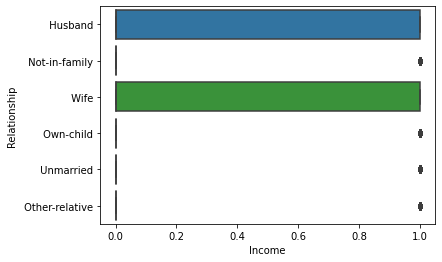


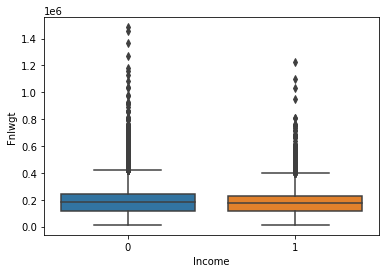


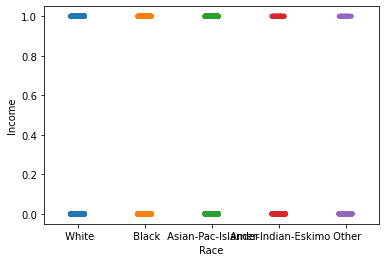






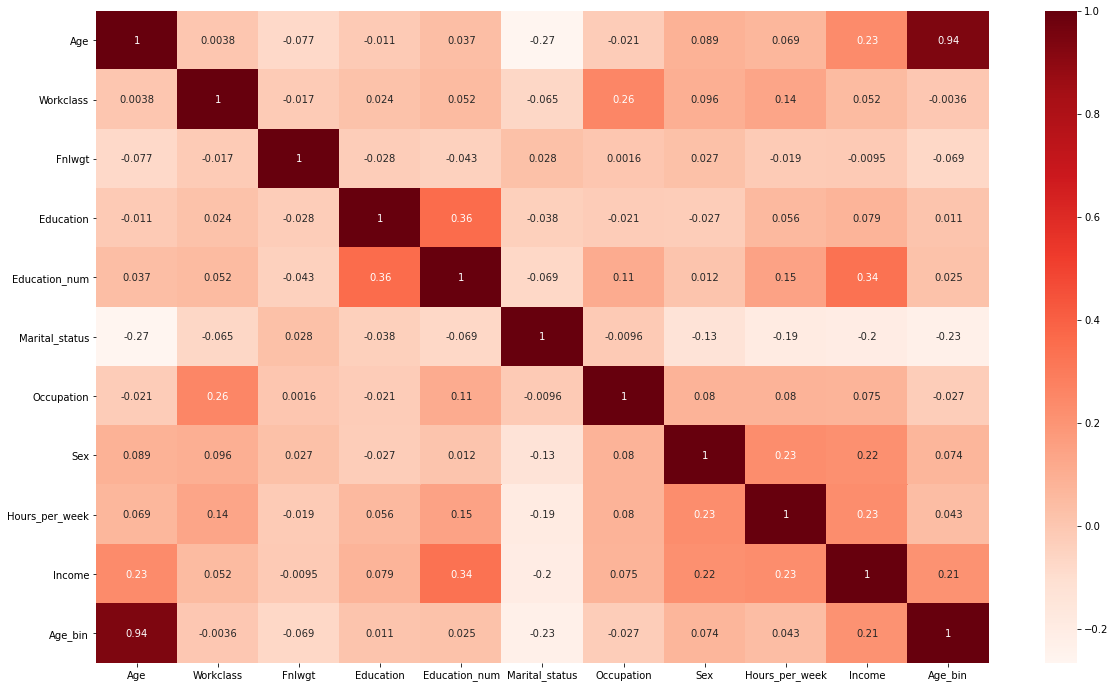








CORRELATION



A Correlation Matrix is shown in Fig 2, in the form of a HeatMap showing Feature-to-Feature and Feature-to-Label Pearson Correlations where all the features are Continuous Variables.

SKEWNESS

df.skew()

:

Age 0.558738

Workclass -0.752280

Fnlwgt 1.446972

Education -0.934063

Education\_num -0.311630

Marital\_status -0.013448

Occupation 0.114540

Sex -0.719244

Hours\_per\_week 0.227636

Income 1.212383

Age\_bin 0.629579

dtype: float64

We can see that there is moderate skewness in some of the columns of the data. I have not applied skewness correction methods as this skewness might be due to the disparity in Income distribution.

Now, we have analysed the data, made the necessary cleaning and pre-processing, let us create the model.

PREDICTIVE MODELLING

Importing the required libraries.

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn** **import** svm

**from** **sklearn.ensemble** **import** RandomForestClassifier

**from** **sklearn.neighbors** **import** KNeighborsClassifier

**from** **sklearn.naive\_bayes** **import** GaussianNB

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn** **import** metrics

**from** **sklearn.metrics** **import** confusion\_matrix

**from** **sklearn.model\_selection** **import** GridSearchCV

**Splitting data for validation**

X=df.drop([‘Income’], axis=1)

y=df[‘Income’]

Here, I am splitting the data in 70% training and 30% for testing.

X\_train, X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size = 0.3, random\_state=42)

Checking the training data shape.

print(X\_train.shape, y\_train.shape)

(22792, 10) (22792,)

We can see that there are 22792 rows and 10 columns in the X\_train and 22792 rows in y\_train.

**-----> Checking Accuracies**

**Logistic Regression**

model\_log = LogisticRegression(solver='liblinear')

model\_log.fit(X\_train, y\_train)

prediction\_log = model\_log.predict(X\_test)

print('The accuracy of the Logistic Regression is',metrics.accuracy\_score(prediction\_log, y\_test))

The accuracy of the Logistic Regression is 0.7570638820638821

**Decision Tree**

model\_tree = DecisionTreeClassifier()

model\_tree.fit(X\_train, y\_train)

prediction\_tree = model\_tree.predict(X\_test)

print('The accuracy of the Decision Tree is ', metrics.accuracy\_score(prediction\_tree, y\_test))

The accuracy of the Decision Tree is 0.7655610155610155

**K-Nearest Neighbours(KNN)**

model\_knn = KNeighborsClassifier()

model\_knn.fit(X\_train, y\_train)

prediction\_knn = model\_knn.predict(X\_test)

print('The accuracy of the K-Nearest Neighbours is ', metrics.accuracy\_score(prediction\_knn, y\_test))

The accuracy of the K-Nearest Neighbours is 0.717956592956593

s =pd.Series()

**for** i **in** list(range(1,11)):

model\_knn = KNeighborsClassifier(n\_neighbors=i)

model\_knn.fit(X\_train, y\_train)

prediction\_knn = model\_knn.predict(X\_test)

s = s.append(pd.Series(metrics.accuracy\_score(prediction\_knn, y\_test)))

plt.plot(list(range(1,11)), s)

plt.xticks([0,1,2,3,4,5,6,7,8,9,10])

plt.title('The Accuracy vs n\_neighbors K-Nearest Neighbours')

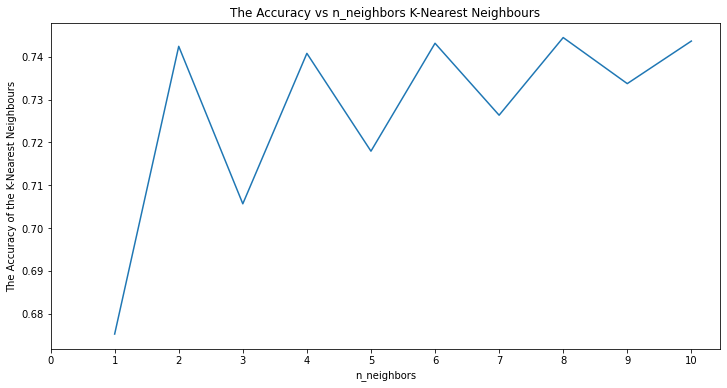
plt.xlabel('n\_neighbors')

plt.ylabel('The Accuracy of the K-Nearest Neighbours')

fig=plt.gcf()

fig.set\_size\_inches(12,6)

plt.show()



**Random Forest**

model\_random = RandomForestClassifier(n\_estimators=300)

model\_random.fit(X\_train, y\_train)

predict\_random = model\_random.predict(X\_test)

print('The accuracy of the Random Forest is ', metrics.accuracy\_score(predict\_random, y\_test))

The accuracy of the Random Forest is 0.8207411957411958

**Gaussian Naive Bayes**

model\_gaus = GaussianNB()

model\_gaus.fit(X\_train, y\_train)

prediction\_gaus = model\_gaus.predict(X\_test)

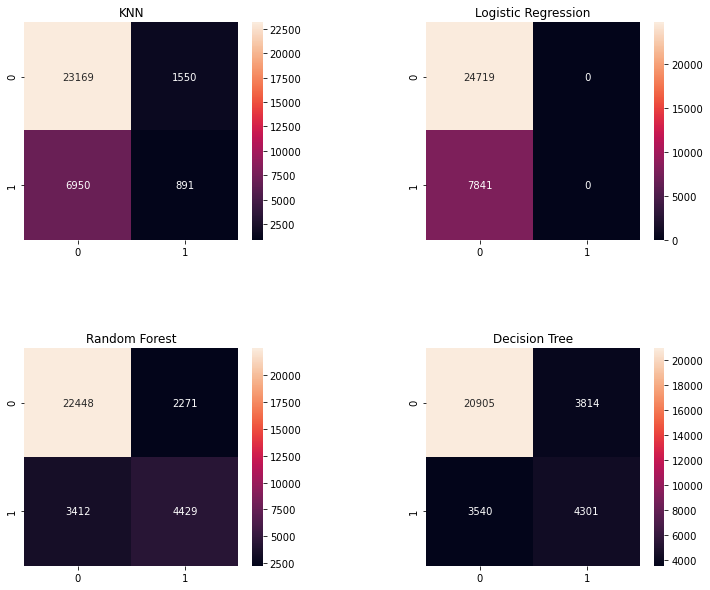
print('The accuracy of the Gaussian Naive Bayes is ', metrics.accuracy\_score(prediction\_gaus, y\_test))

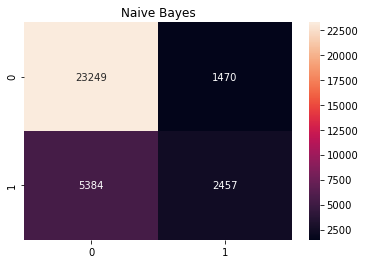
The accuracy of the Gaussian Naive Bayes is 0.7861384111384111

**-----> Checking Cross Validation Scores**

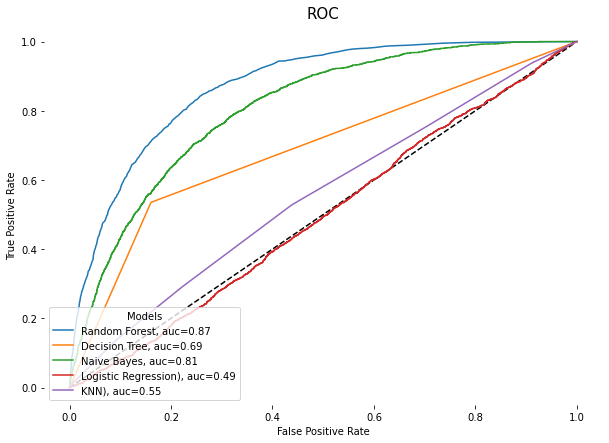
|  | **Cross Validation Score Mean** |
| --- | --- |
| **Logistic Regression** | 0.759183 |
| **KNN** | 0.740018 |
| **Decision Tree** | 0.773311 |
| **Naive Bayes** | 0.789343 |
| **Random Forest** | 0.824447 |

**Confusion Matrix for the above models**





ROC CURVE



According to the obtained Training and Validation Accuracy, it can be concluded that the model is a good fit.

The Area Under the Receiver Operator Characteristic Curve (AUROC), which is a descent one, as more the AUROC (towards 1.0), better the performance of the model.

Random Forest model got the highest AUROC score of 0.87.

HYPER PARAMETER TUNING

**---> Random Forest Model**

Fitting 5 folds for each of 19 candidates, totalling 95 fits

[Parallel(n\_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[Parallel(n\_jobs=1)]: Done 95 out of 95 | elapsed: 41.4min finished

0.8261977886977887

RandomForestClassifier(n\_estimators=800, random\_state=0)

The best parameters I have got in hyper parameter tuning for this model is 'n\_estimators': 800

The RMSE score was 0.0

**--->Logistic Regression**

Tuned Logistic Regression Parameters: {'C': 1e-05}

Best score is 0.7591830466830467

**--->Decision Tree Regression Model**

Hypertuned model of Decision Tree Regression Model has the following parameters

DecisionTreeRegressor(max\_depth=5, max\_features='auto', max\_leaf\_nodes=50, min\_samples\_leaf=2, min\_weight\_fraction\_leaf=0.1, splitter='random')



MAE: 0.31535547106783796

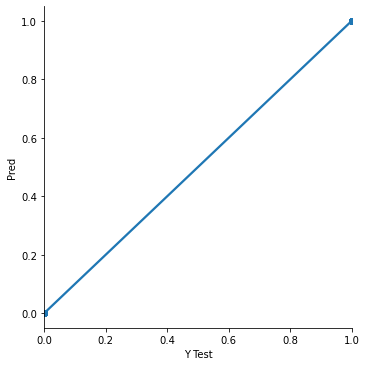
MSE: 0.1584989770456329

RMSE: 0.39811930001650625

We can see that we got best accuracy score of 0.8261977886977887 with Random Forest Model.

PREDICTIONS

|  | **Actual Data** | **Predicted Data** |
| --- | --- | --- |
| **14160** | 0 | 0 |
| **27047** | 0 | 0 |
| **28867** | 1 | 1 |
| **5667** | 1 | 1 |
| **7827** | 0 | 0 |



**SAVING THE MODEL**

**import** **joblib**

joblib.dump(model\_random, 'Census\_Income\_Prediction.pkl')

Output: ['Census\_Income\_Prediction.pkl']

*# Load the model from the file*

model\_random\_from\_joblib = joblib.load('Census\_Income\_Prediction.pkl')

*# Use the loaded model to make predictions*

model\_random\_from\_joblib.predict(X\_test)

Output:

array([0, 0, 1, ..., 1, 0, 0], dtype=int64)

CONCLUSION

This report proposed the application of various Learning Algorithms, with extensive Hyper Parameter Tuning with Grid Search on Census Data. Finally, the Validation Accuracy, so obtained, 82.61% with Random Forest Model, which is, by the best of our knowledge, has been the highest ever numeric accuracy achieved by any Income Prediction Model so far. The future scope of this work involves achieving an over-all better set of results by using hybrid models with inclusion of Machine Learning and Deep Learning together, or by applying many other advanced pre-processing techniques without further depletion in the accuracy.