Import Libraries and dataset

I will import `numpy`, `pandas` and `matplotlib` for working with the dataset. Next, I will use `sklearn` for using Machine Learning models. Finally, I'll import the dataset.

```
# Working with data
import numpy as np
import pandas as pd

# Visualizations
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
%matplotlib inline

# Ignore warnings
import warnings
warnings.filterwarnings('ignore');
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Next, I will import the dataset.

```
columns = ['Age', 'Work Class', 'Final Weight', 'Education', 'Education Number', 'Marital Status', 'Occupation',
          'Relationship', 'Race', 'Sex', 'Capital Gain', 'Capital Loss', 'Hours per Week', 'Country', 'Income']
dataset = pd.read_csv('/content/adult.csv', names = columns)
```

Before any analysis, let's convert the target column into numerical classes.

```
from sklearn.preprocessing import LabelEncoder

labelEncoder = LabelEncoder()
dataset['Income'] = labelEncoder.fit_transform(dataset['Income'])
```

Exploratory Data Analysis and Data Processing

I'll next take a look at the data and draw visualizations to understand it better. I'll first take a look at the data collectively as a whole and then try to infer each column one by one.

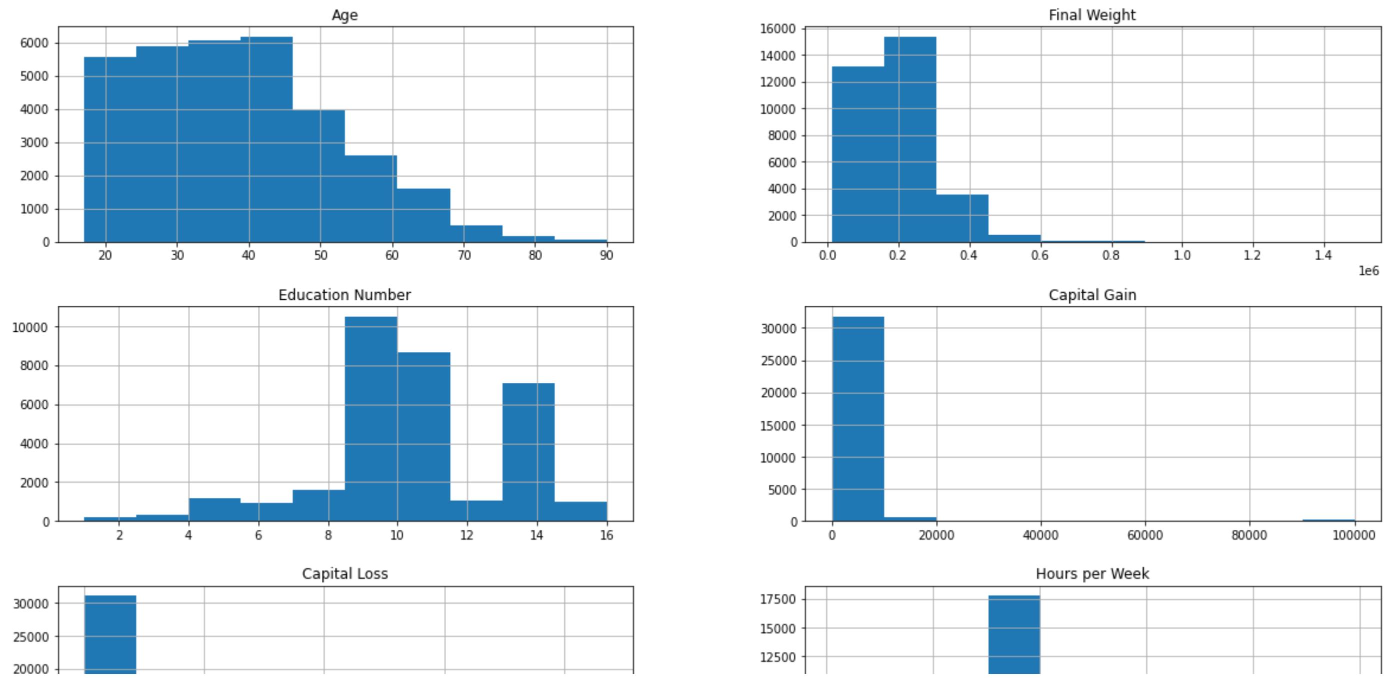
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Age               32561 non-null  int64 
 1   Work Class        32561 non-null  object
 2   Final Weight      32561 non-null  int64 
 3   Education         32561 non-null  object
 4   Education Number  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  Country           32561 non-null  object
 14  Income            32561 non-null  int64 
dtypes: int64(7), object(8)
memory usage: 3.7+ MB
```

The information above reveals that there are no missing values in the datatset.

```
rcParams['figure.figsize'] = 20, 12
dataset[['Age', 'Final Weight', 'Education Number', 'Capital Gain', 'Capital Loss', 'Hours per Week']].hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f9c40a9fc40>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f9c40a76130>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f9c40a354f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f9c409e18e0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f9c40a0fcd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f9c409bc160>]],
      dtype=object)
```



From the histograms above, I can infer the following:

1. I can group the **Age** column into bins.
2. For **Capital Gain** and **Capital Loss** the data is highly left skewed which needs to be tackled.
3. We need to analyse **Education Number** further as it might align with **Education** information.
4. **Final Weight** is also left skewed.
5. The **Hours per Week** can also be split into bins.

```
plt.matshow(dataset.corr())
plt.colorbar()
plt.xticks(np.arange(len(dataset.corr().columns)), dataset.corr().columns.values, rotation = 45)
plt.yticks(np.arange(len(dataset.corr().columns)), dataset.corr().columns.values)
for (i, j), corr in np.ndenumerate(dataset.corr()):
    plt.text(j, i, '{:0.1f}'.format(corr), ha='center', va='center', color='white', fontsize=14)
```

I'll now analyse the categorical features using `CountPlot`.

Age

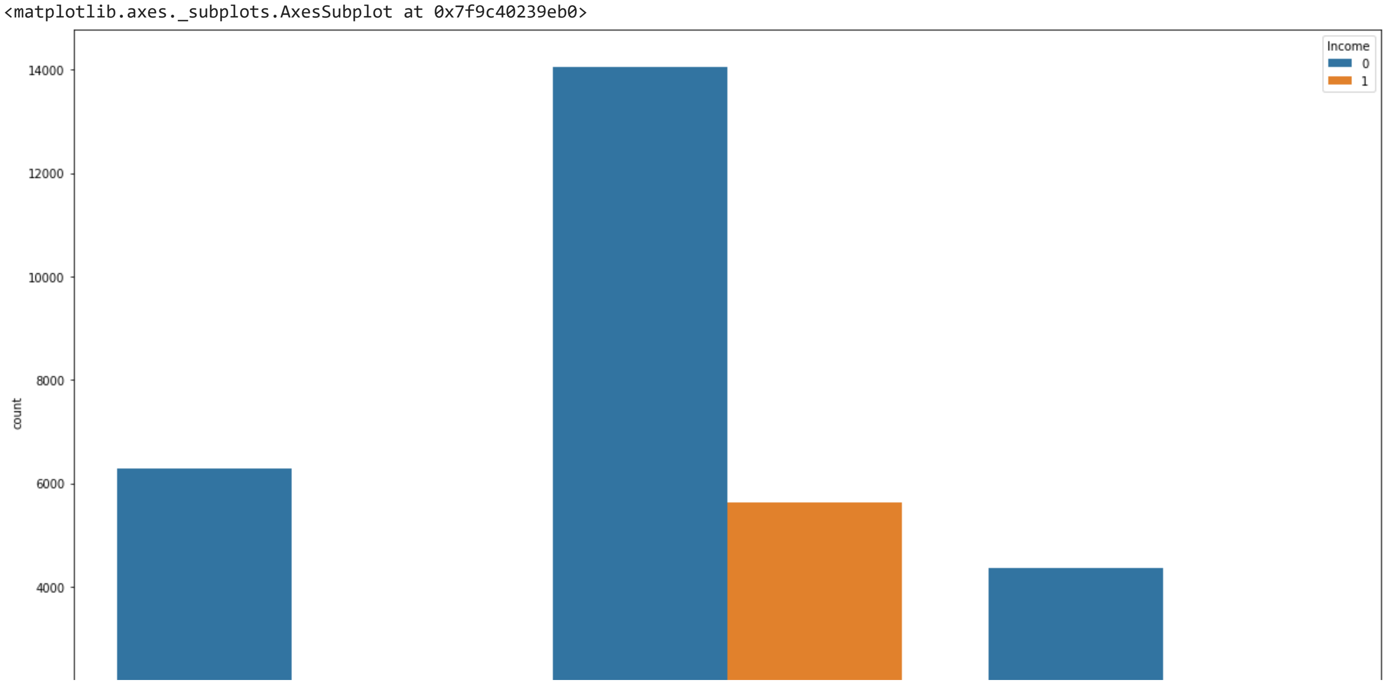
Here, I'll bucket the age into separate bins.

1. 0-25: Young
2. 25-50: Adult
3. 50-100: Old

```
dataset['Age'] = pd.cut(dataset['Age'], bins = [0, 25, 50, 100], labels = ['Young', 'Adult', 'Old'])
```

```
sns.countplot(x = 'Age', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c40239eb0>
```



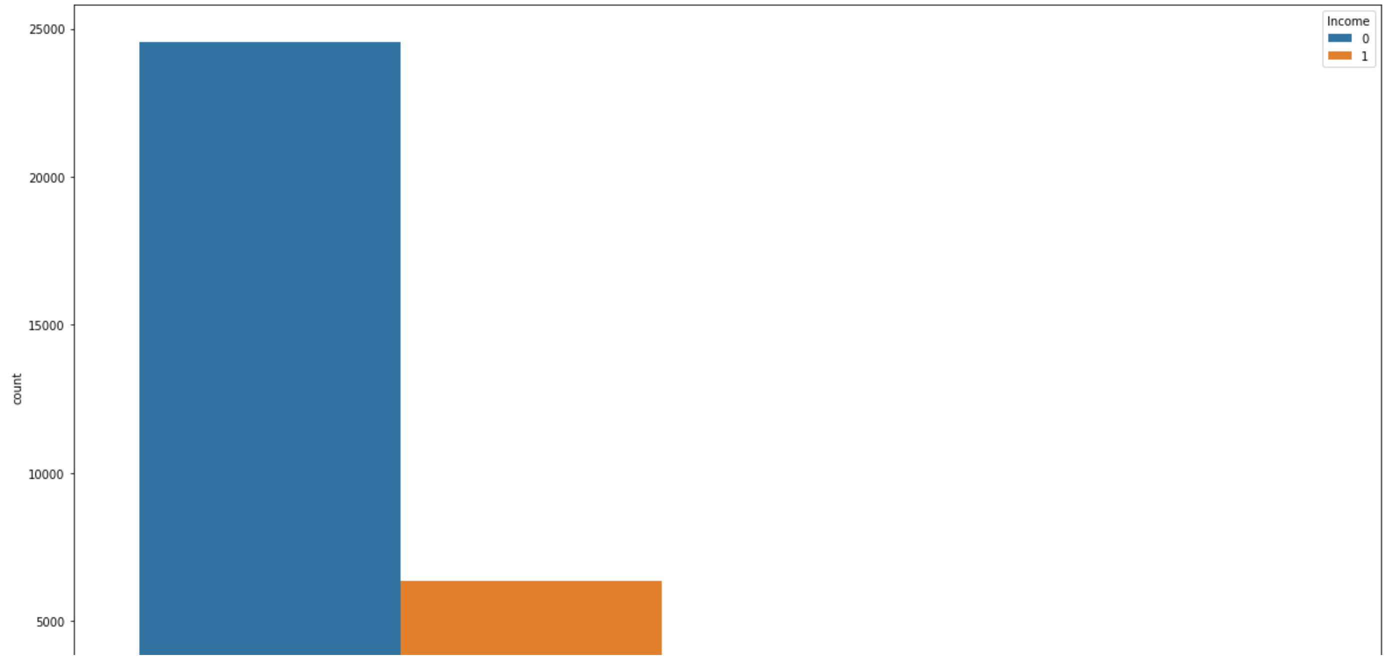
Capital Gain and Capital Loss

Rather than having both `Capital Gain` and `Capital Loss`, I will use their difference as that is more relevant and gives the change.

```
dataset['Capital Diff'] = dataset['Capital Gain'] - dataset['Capital Loss']
dataset.drop(['Capital Gain'], axis = 1, inplace = True)
dataset.drop(['Capital Loss'], axis = 1, inplace = True)
```

```
dataset['Capital Diff'] = pd.cut(dataset['Capital Diff'], bins = [-5000, 5000, 100000], labels = ['Minor', 'Major'])
sns.countplot(x = 'Capital Diff', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c401be430>
```



On taking a look at the result, I can see that the for `Minor` there are more people with Income less than `$50K` and for `Major` there are more people with Income greater than `$50K`. This is in complete agreement with the fact that people who have large `Capital Gain` compared to `Capital Loss` have `Income` more than `$50K`.

Final Weight

As seen above, there is no correlation between Income and Final Weight, so I will drop this column.

```
dataset.drop(['Final Weight'], axis = 1, inplace = True)
```

Hours per Week

Taking a look at the histogram for `Hours per Week`, I can see that the dataset is aligned around the center. I can still create buckets from this data. As usually, the work hours are close to 30-40 hours, I create the buckets as `0-30`, `30-40`, and `40-100`.

```
dataset['Hours per Week'] = pd.cut(dataset['Hours per Week'],
                                   bins = [0, 30, 40, 100],
                                   labels = ['Lesser Hours', 'Normal Hours', 'Extra Hours'])
```

```
sns.countplot(x = 'Hours per Week', hue = 'Income', data = dataset)
```

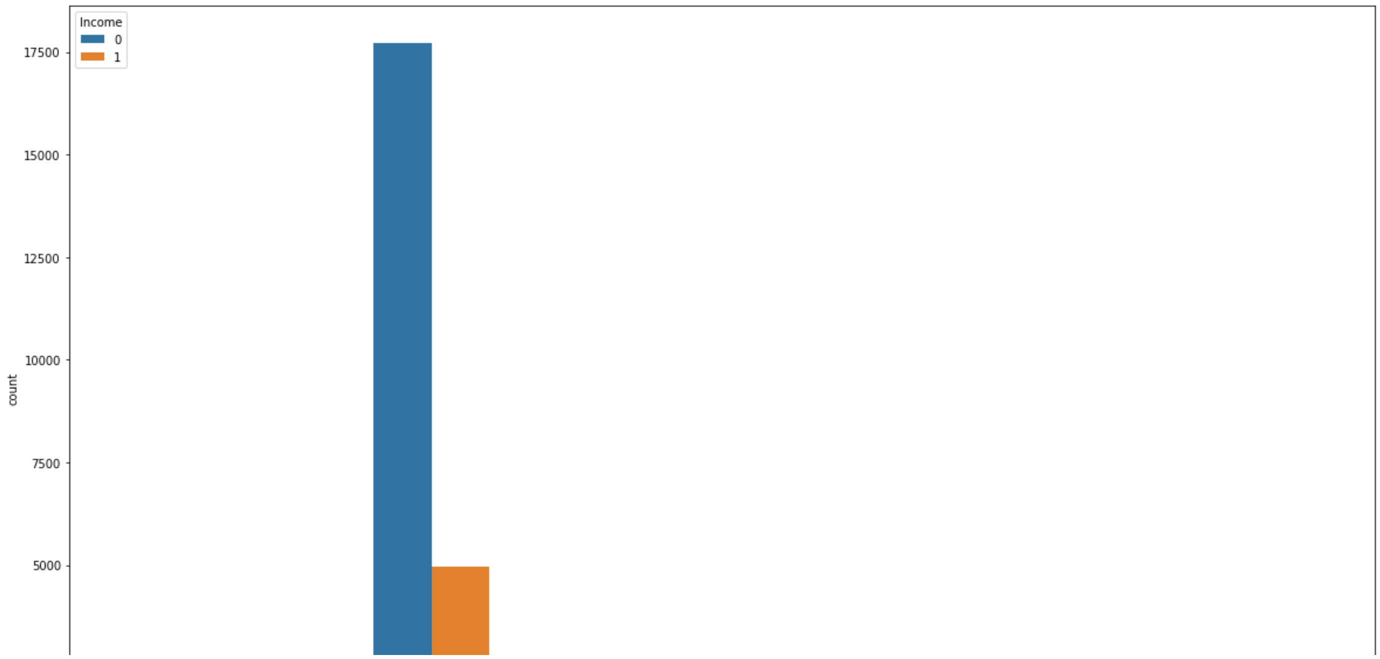
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c40115a00>
```

Taking a look at the plot above, we can see a trend. As the number of hours increase, the number of people earning more than `$50K` increases in comparison to the people earning less.

Work Class

```
sns.countplot(x = 'Work Class', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c400f2e80>
```



Taking a look at the plot above, I can see that there are `Work Class` values defined as `?` which appears to be error data. As it is very less, I'll simply remove these records. Also, the two values `Without-pay` and `Never-worked` are negligible and hence it is safe to drop them too.

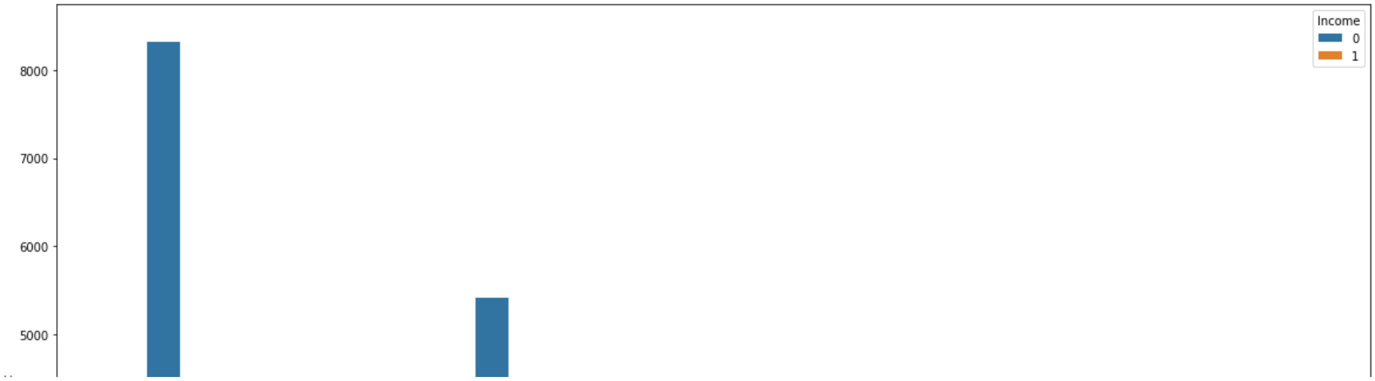
```
dataset = dataset.drop(dataset[dataset['Work Class'] == ' ?'].index)
dataset = dataset.drop(dataset[dataset['Work Class'] == ' Without-pay'].index)
dataset = dataset.drop(dataset[dataset['Work Class'] == ' Never-worked'].index)
```

Education and Education Number

It's a good time to check if there is any relation between Education and Education Number.

```
sns.countplot(x = 'Education', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c40021970>
```




```
education_classes = dataset['Education'].unique()
for edu_class in education_classes:
    print("For {}, the Education Number is {}"
          .format(edu_class, dataset[dataset['Education'] == edu_class]['Education Number'].unique()))
```

```
For  Bachelors, the Education Number is [13]
For  HS-grad, the Education Number is [9]
For  11th, the Education Number is [7]
For  Masters, the Education Number is [14]
For  9th, the Education Number is [5]
For  Some-college, the Education Number is [10]
For  Assoc-acdm, the Education Number is [12]
For  Assoc-voc, the Education Number is [11]
For  7th-8th, the Education Number is [4]
For  Doctorate, the Education Number is [16]
For  Prof-school, the Education Number is [15]
For  5th-6th, the Education Number is [3]
For  10th, the Education Number is [6]
For  Preschool, the Education Number is [1]
For  12th, the Education Number is [8]
For  1st-4th, the Education Number is [2]
```

From the analysis above, I discovered that **Education Number** and **Education** are just the same. So, I can drop any one column. Also, I'll combine all information from **Preschool** to **12th** as they can be considered of one class who have no college/university level education.

```
dataset.drop(['Education Number'], axis = 1, inplace = True)
dataset['Education'].replace([' 11th', ' 9th', ' 7th-8th', ' 5th-6th', ' 10th', ' 1st-4th', ' Preschool', ' 12th'],
                             ' School', inplace = True)
dataset['Education'].value_counts()
```

```
 HS-grad         9959
 Some-college    6772
 Bachelors       5182
 School          3820
 Masters         1675
 Assoc-voc       1321
 Assoc-acdm      1019
 Prof-school      558
 Doctorate        398
Name: Education, dtype: int64
```

Marital Status and Relationship

```
dataset['Marital Status'].value_counts()
```

```
 Married-civ-spouse       14331
 Never-married             9908
 Divorced                  4258
 Separated                  959
 Widowed                    839
 Married-spouse-absent      388
 Married-AF-spouse           21
Name: Marital Status, dtype: int64
```

```
dataset['Relationship'].value_counts()
```

```
 Husband          12700
 Not-in-family     7865
 Own-child         4520
```

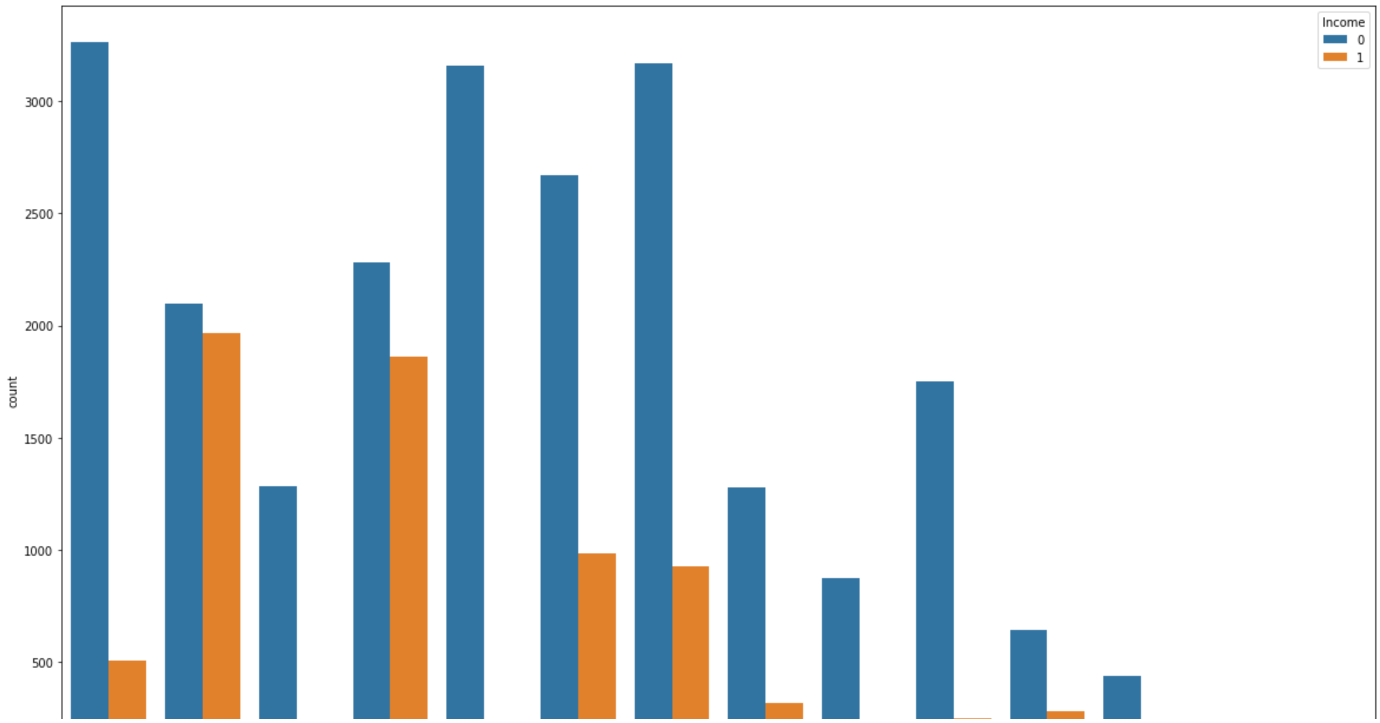
```
 Unmarried          3269
 Wife               1432
 Other-relative      918
Name: Relationship, dtype: int64
```

Both of them have no missing values. There is some overlap between the two such as if the person is **Husband** or **Wife**, then their marital status would be **Married**. However, as there is no complete overlap, I'll keep both these columns.

Occupation

```
plt.xticks(rotation = 45)
sns.countplot(x = 'Occupation', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c4008bcd0>
```



The data has no missing values. The categories have already been uniquely defined and we can keep it as is.

Race

```
sns.countplot(x = 'Race', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c3fe9e5e0>
```

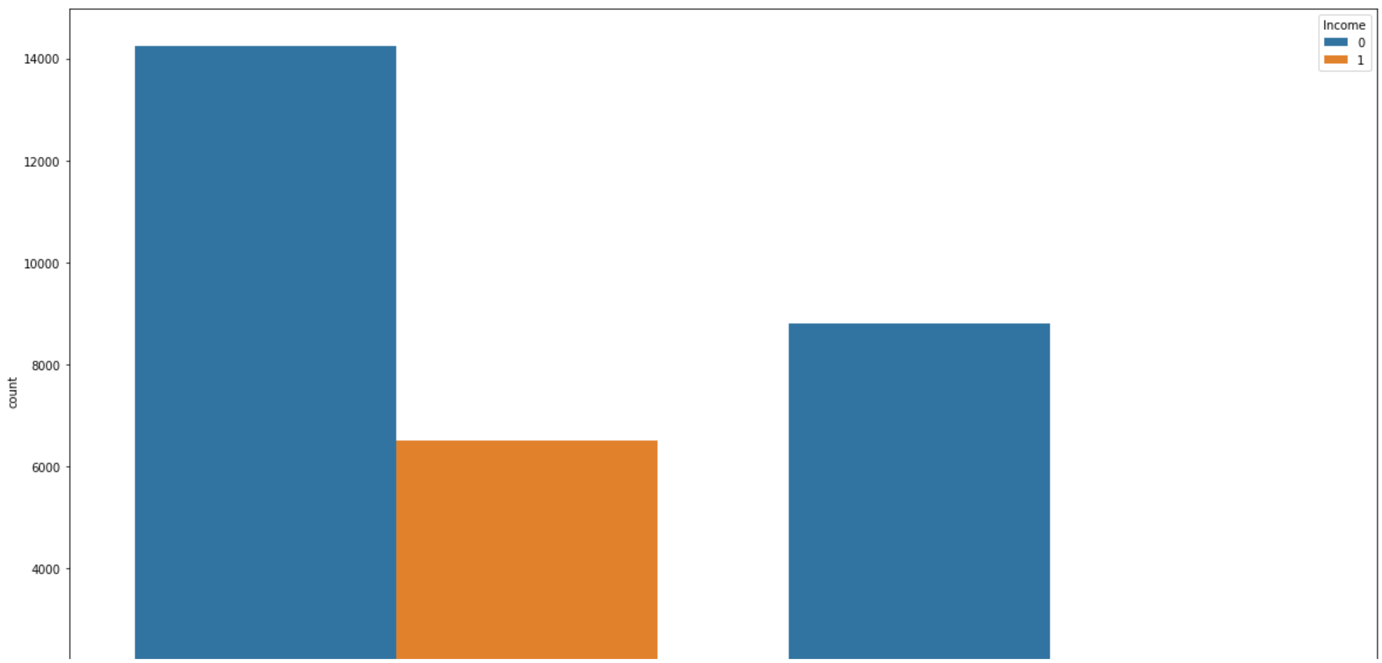
The dataset includes majority of information about White race while all other races are lesser in number. I'll combine all other race data into one class as `Other`.

```
dataset['Race'].unique()
dataset['Race'].replace([' Black', ' Asian-Pac-Islander', ' Amer-Indian-Eskimo', ' Other'],' Other', inplace = True)
```

Sex

```
sns.countplot(x = 'Sex', hue = 'Income', data = dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c4164af40>
```



From the plot above, it is clear that

1. There are more Male participants than Female participants
2. When we compare the two genders and the corresponding income distribution, more percentage of Males have an Income of more than \$50K than Females.

Country

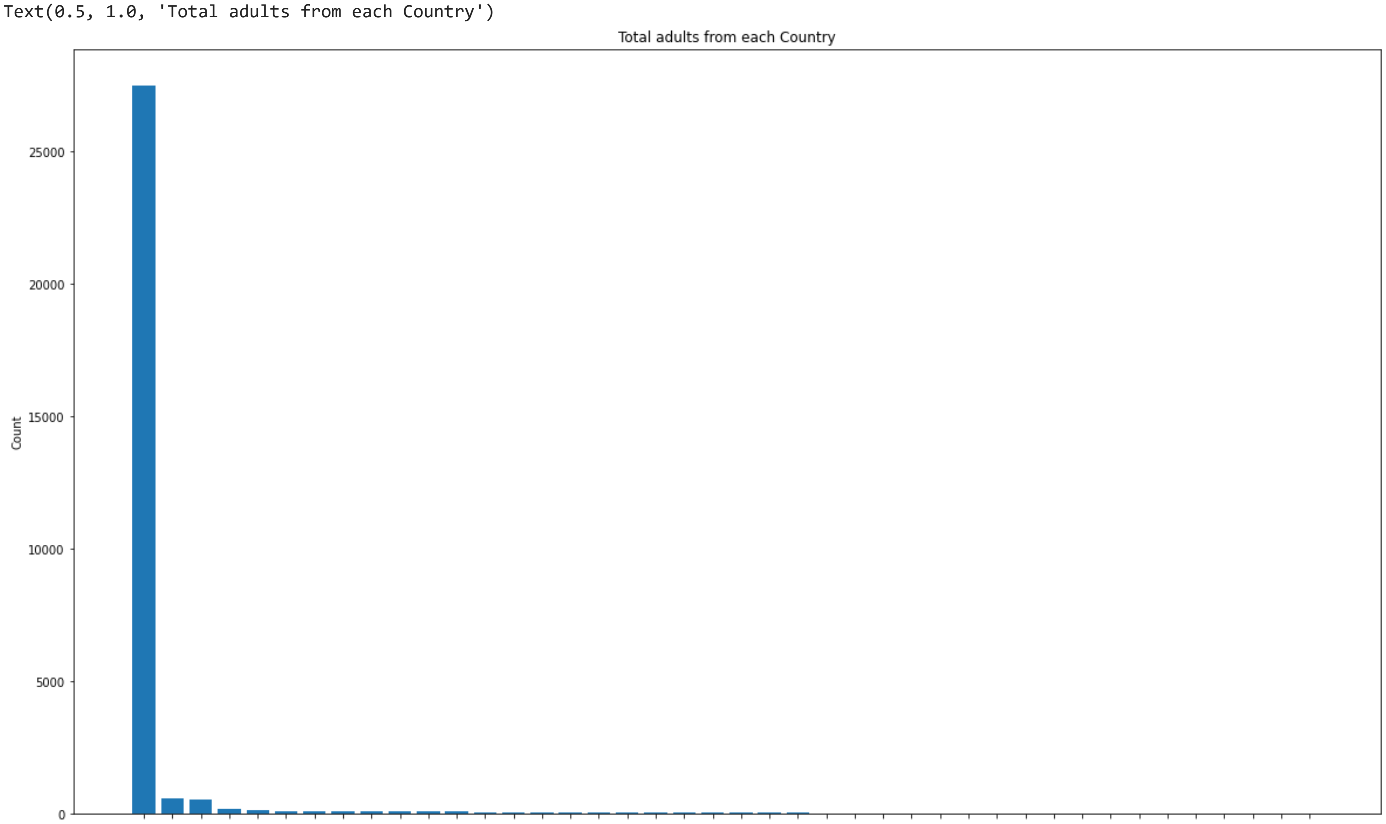
```
country_count = dataset['Country'].value_counts()
country_count
```

```
United-States                 27491
Mexico                          610
?                               556
Philippines                     187
Germany                         128
Puerto-Rico                     109
Canada                          107
India                           100
El-Salvador                     100
Cuba                             92
England                          86
Jamaica                          80
South                            71
China                            68
Italy                            68
Dominican-Republic               67
Vietnam                          64
```

```
    Guatemala                         63
    Japan                             59
    Poland                            56
    Columbia                          56
    Iran                              42
    Taiwan                            42
    Haiti                             42
    Portugal                          34
    Nicaragua                         33
    Peru                              30
    Greece                            29
    France                            27
    Ecuador                           27
    Ireland                           24
    Hong                              19
    Cambodia                          18
    Trinadad&Tobago                   18
    Thailand                          17
    Laos                              17
    Yugoslavia                        16
    Outlying-US(Guam-USVI-etc)        14
    Hungary                           13
    Honduras                          12
    Scotland                          11
    Holand-Netherlands                 1
   Name: Country, dtype: int64
```

```
plt.bar(country_count.index, country_count.values)
plt.xticks(rotation = 90)
plt.xlabel('Countries')
plt.ylabel('Count')
plt.title('Total adults from each Country')
```

```
Text(0.5, 1.0, 'Total adults from each Country')
```



There are two things that I noticed:

1. There are some missing values in Country column denoted by ?. As they are very less, I'll drop these rows.
2. The majority of adults are from United States. Thus, we can distribute the column with values as either **United States** or **Other**.

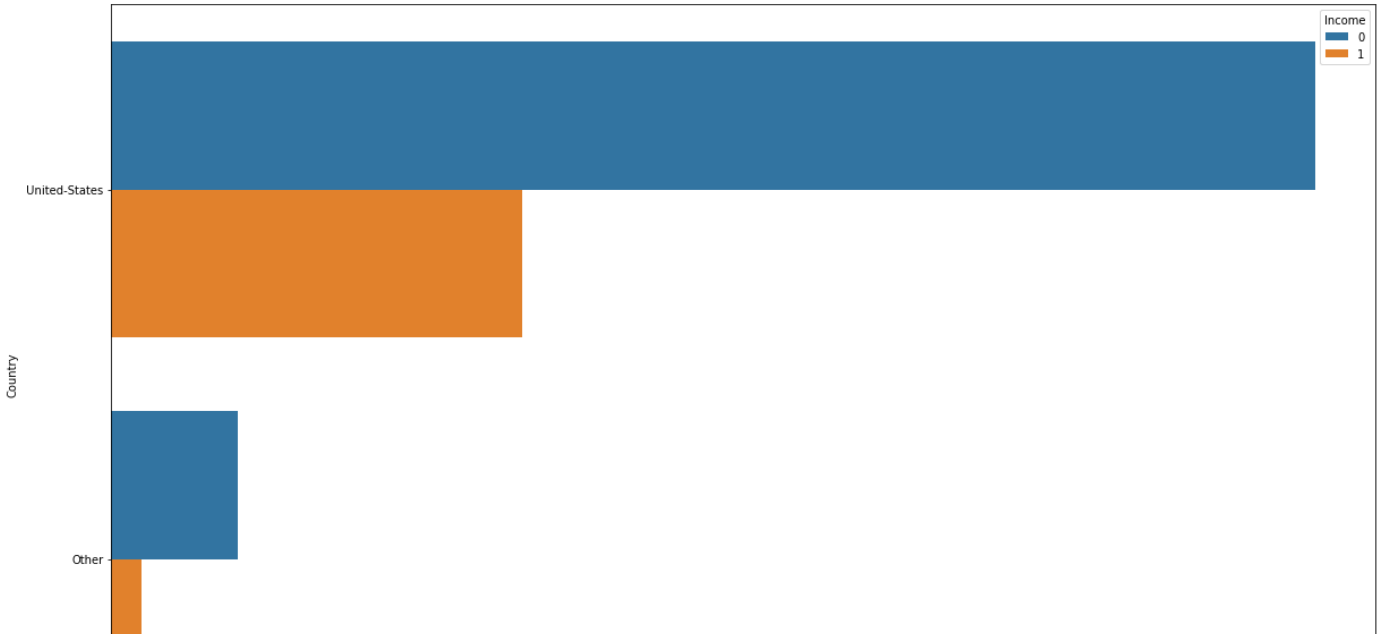
```
dataset = dataset.drop(dataset[dataset['Country'] == ' ?'].index)
```

```
countries = np.array(dataset['Country'].unique())
countries = np.delete(countries, 0)
```

```
dataset['Country'].replace(countries, 'Other', inplace = True)
```

```
sns.countplot(y = 'Country', hue = 'Income', data = dataset)
```

    <matplotlib.axes._subplots.AxesSubplot at 0x7f9c402c8220>



The data now appears much better.

I've analysed all columns. I'll simply convert categorical columns to numerical.

I will use the `get_dummies` method of pandas to get separate columns for each feature based on the unqiue values in the dataset.

```
y = dataset['Income']
X = dataset.drop(['Income'], axis = 1)
X = pd.get_dummies(X)
print("Total features: {}".format(X.shape[1]))
```

    Total features: 56

Next, I will split the dataset into the training and testing data using `train_test_split`.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
```

We're now ready to start with Machine Learning.

Appyting Machine Learning

I'll apply 5 algorithms to make the classification including **Naive Bayes Classifier**, **Support Vector Classifier**, **Decision Tree Classifier**, **Random Forest Classifier** and **Gradient Boosting Classifier**.

```
from sklearn.metrics import f1_score, accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

```
classifiers = [GaussianNB(),
               SVC(kernel = 'rbf', probability = True),
               DecisionTreeClassifier(random_state = 0),
               RandomForestClassifier(n_estimators = 100, random_state = 0),
```

```
              GradientBoostingClassifier(random_state = 0)]
classifier_names = ["Gaussian Naive Bayes",
                    "Support Vector Classifier",
                    "Decision Tree Classifier",
                    "Random Forest Classifier",
                    "Gradient Boosting Classifier"]
accuracies = []

for i in range(len(classifiers)):
    classifier = classifiers[i]
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)
    print("{}:".format(classifier_names[i]))
    print("F1 score: {:.2f}".format(f1_score(y_test, y_pred)))
    accuracy = accuracy_score(y_test, y_pred)*100
    accuracies.append(accuracy)
```

```
Gaussian Naive Bayes:
F1 score: 0.64
Support Vector Classifier:
F1 score: 0.65
Decision Tree Classifier:
F1 score: 0.62
Random Forest Classifier:
F1 score: 0.64
Gradient Boosting Classifier:
F1 score: 0.65
```

Analysing Results

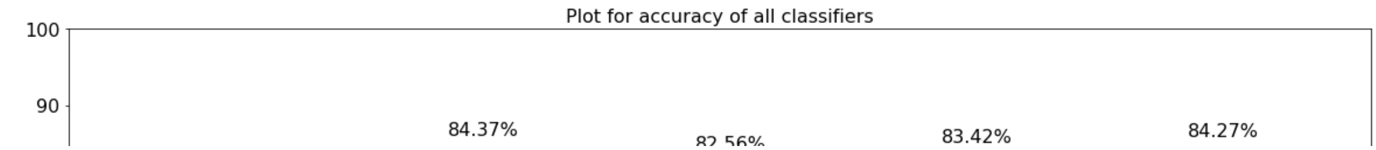
I use **Accuracy Plot** and **ROC Curve** to analyse the results.|

From the cells above, we can see that **GradientBoostingClassifier** performed the best with an **F1 score of 0.65**.

Accuracy Plot

```
from matplotlib.cm import rainbow

plt.figure(figsize = (20, 12))
colors = rainbow(np.linspace(0, 1, len(classifiers)))
barplot = plt.bar(classifier_names, accuracies, color = colors)
plt.yticks([0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100], fontsize = 16)
plt.xticks(fontsize = 14)
plt.xlabel("Classifiers", fontsize = 16)
plt.ylabel("Accuracy", fontsize = 16)
plt.title("Plot for accuracy of all classifiers", fontsize = 16)
for i, bar in enumerate(barplot):
    plt.text(bar.get_x() + bar.get_width()/2 - 0.1,
             bar.get_height()*1.02,
             s = '{:.2f}%'.format(accuracies[i]),
             fontsize = 16)
```



As it can be seen from the plot above, the **Gradient Boosting Classifier** had the best accuracy. Graphs make representing information really easy and intuitive.

ROC Curve

Let's also analyse the ROC Curve for the predictions for income more than $50K.

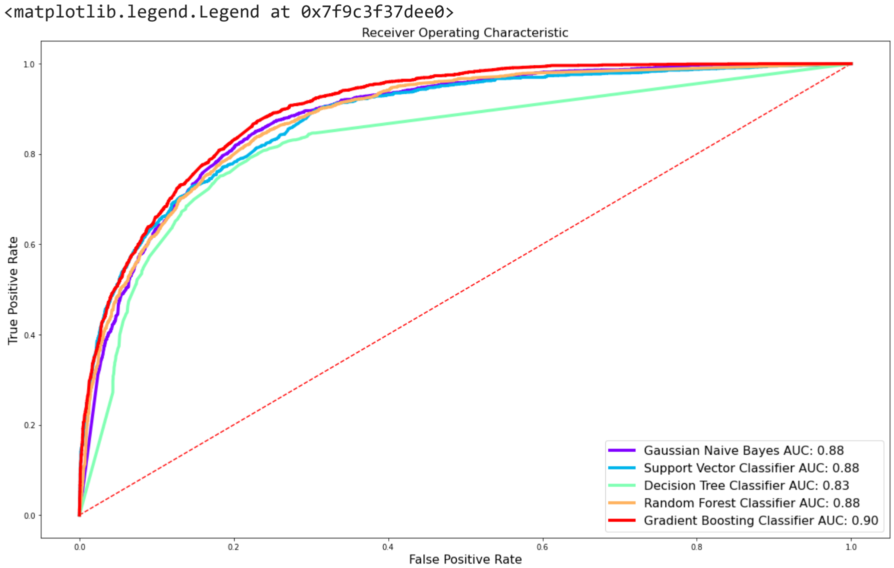
```
from sklearn.metrics import roc_curve, auc

plt.figure(figsize = (20, 12))
plt.plot([0,1], [0,1], 'r--')

for i in range(len(classifiers)):
    classifier = classifiers[i]
    probs = classifier.predict_proba(X_test)
    # Reading probability of second class
    probs = probs[:, 1]
    fpr, tpr, thresholds = roc_curve(y_test, probs)
    roc_auc = auc(fpr, tpr)
    label = classifier_names[i] + ' AUC:' + ' {0:.2f}'.format(roc_auc)
    plt.plot(fpr, tpr, c = colors[i], label = label, linewidth = 4)

plt.xlabel('False Positive Rate', fontsize = 16)
plt.ylabel('True Positive Rate', fontsize = 16)
plt.title('Receiver Operating Characteristic', fontsize = 16)
plt.legend(loc = 'lower right', fontsize = 16)
```

```
<matplotlib.legend.Legend at 0x7f9c3f37dee0>
```




```
**Gradient Boosting Classifier** has the maximum
**Area Under Curve** with a value of **0.90**.
```

Gradient Boosting Classifier has the maximum **Area Under Curve** with a value of **0.90**.

3s completed at 11:01 PM