Assignment-1 Report

Data Mining
CSE-5334-004

Student Name:
Mehul Ganjude-1001990551,
Gagan Ujjini Mallikarjuna -1001851247,
Venkata Nagendar Shantiswaroop Adibhatla-1001862413

Professor: Dr Elizabeth D Diaz
Teaching Assistant: Pralobh Lokhande

Language: Python

Introduction

Data mining is a process of extracting and discovering patterns in large data sets. In this assignment we are analyzing Income Dataset.
So let us see the way how we can apply the techniques of data mining to come up with the more insights about the given data.

Retrieving the Data

Data retrieval is the process of converting the data which is present in the file to the data frame which is further used for analysis. The Income dataset is loaded to the Jupiter notebook using the below code.

```
#read the csv file into a Pandas data frame
df_data = pd.read_csv('income_dataset.csv', encoding='latin1')
```

Here, we are reading the data from the csv file and then saving it into the Data Frame. Data frame is the data structure which is used in the project to do further process like data exploration and visualization.

Glimpse of Data

The dataset which we have for this assignment is Income data which has the information about the Income depend upon the education in various countries. The data contains 43, 957 rows and 15 columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43957 entries, 0 to 43956
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              43957 non-null  int64
 1   workclass        41459 non-null  object
 2   final-weight     43957 non-null  int64
 3   education        43957 non-null  object
 4   educational-num  43957 non-null  int64
 5   marital-status   43957 non-null  object
 6   occupation       41451 non-null  object
 7   relationship     43957 non-null  object
 8   race             43957 non-null  object
 9   gender           43957 non-null  object
 10  capital-gain     43957 non-null  int64
 11  capital-loss     43957 non-null  int64
 12  hours-per-week   43957 non-null  int64
 13  native-country   43957 non-null  object
 14  income > 50K     43957 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.0+ MB
```

Reading Dataset

```
#return the first 5 rows of the dataset
df_data.head()
```

	age	workclass	final-weight	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income > 50K
0	67	Private	366425	Doctorate	16	Divorced	Exec-managerial	Not-in-family	White	Male	99999	0	60	United-States	Yes
1	17	Private	244602	12th	8	Never-married	Other-service	Own-child	White	Male	0	0	15	United-States	No
2	31	Private	174201	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States	Yes
3	58	State-gov	110199	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	40	United-States	No
4	25	State-gov	149248	Some-college	10	Never-married	Other-service	Not-in-family	Black	Male	0	0	40	United-States	No

Task 1: Statistical Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing data. It's where the researcher takes a bird's eye view of the data and tries to make some sense of it. It's often the first step in data analysis, implemented before any formal statistical techniques are applied.

```
###########
# Task 1 ##
###########

# 2.5 points
#Task 1-a: Print the details of the df_data data frame (information such as number of rows,columns,
#name of columns, etc)
print(">>Task 1-a: Details of df_data data frame are: \n")
print("Number of Rows:", df_data.shape[0])
print("Number of Columns:", df_data.shape[1])
print("Column Names: ")
for col in df_data.columns:
    print(col)
```

```
>>Task 1-a: Details of df_data data frame are:

Number of Rows: 43957
Number of Columns: 15
Column Names:
age
workclass
final-weight
education
educational-num
marital-status
occupation
relationship
race
gender
capital-gain
capital-loss
hours-per-week
native-country
income > 50K
```


```
>>Task 1-a: Details of df_data data frame are:

Number of Rows: 43957
Number of Columns: 15
Column Names:
age
workclass
final-weight
education
educational-num
marital-status
occupation
relationship
race
gender
capital-gain
capital-loss
hours-per-week
native-country
income > 50K
```


```
# 2.5 points
#Task 1-b: Find the number of rows and columns in the df_data data frame.
num_rows = df_data.shape[0]
num_cols = df_data.shape[1]
print ("\n\n>>Task 1-b: Number of rows:%s and number of columns:%s"% (num_rows, num_cols ))
```

```
>>Task 1-b: Number of rows:43957 and number of columns:15
```

```
# 2.5 points
#Task 1-c: Print the descriptive detail (count, unique, top, freq etc) for 'educational-num' column of the df_data

Unique_Value=df_data['educational-num']
Count=df_data['educational-num'].count()
top_val = df_data['educational-num'][0]
end_value= df_data['educational-num'].iloc[-1]
frequency = df_data['educational-num'].value_counts()
print ("\n\n>>Task 1-c: Descriptive details of 'educational-num' column are:-\n",
       Unique_Value,"\nCount is:- ",Count,"\nTop value is:- ",top_val,
       "\nEnd value is:- ",end_value,"\nFrequency is:- \n",frequency)
```

```
>>Task 1-c: Descriptive details of 'educational-num' column are:-
 0        16
1         8
2        13
3         4
4        10
         ..
43952    13
43953     9
43954    10
43955    13
43956     9
Name: educational-num, Length: 43957, dtype: int64
Count is:-  43957
Top value is:-  16
End value is:-  9
Frequency is:-
 9     14197
10     9790
13     7219
14     2392
11     1831
7      1647
12     1447
6      1250
4       862
15      748
5       684
8       587
16      536
3       468
2       223
1        76
Name: educational-num, dtype: int64
```

```
# 10 points
#Task 1-d: Print ALL the unique values of Capital-gain and Print ALL the unique values of Native-Country .


# create new dataframe, repeating or chaining as appropriate

num_uniq_capital_gain = df_data['capital-gain'].unique()
num_uniq_native_country = df_data['native-country'].unique()


print ("\n\n >>Task 1-d:")
print(num_uniq_capital_gain)

print("##################################################")
print(num_uniq_native_country)
```

```
 >>Task 1-d:
[99999     0  2653  4386  6849  5178 15024  4416  2964  2829  2176  5013
   594  2174  3137  7688  1086  3674 14344  4865  7298  9386 27828  7978
  3471 14084  1797  6497 10520  2414  2580  3103  4650  3942  3325  2354
  2597 13550  2407  4931 20051  4787  1455  1831  5060  2202  1173  6418
  2105  4101  8614  4064  3411  4508  3464  2885 25124  7443  3908   914
  9562  6514  3887  2050  2463  3418  5455   114   991  2290  2907 15831
  1151  2036  2961 10605  2329  1506  2977 34095  7430  1055  2993 22040
   401  6612  2936  6723  2538 10566  1848 41310  2346  2228  6767  1471
 15020 25236  3781  2009  3273  5721  3432  3818  4934  2635  1424  5556
  2062  4687  6360  1409  6097  1264 11678  1639 18481  7896  3456  7262
  1111  1731]
##################################################
['United-States' 'Japan' 'South' 'Portugal' 'Italy' 'Mexico' 'Ecuador'
 'England' 'Philippines' 'China' 'Germany' 'Unknown' 'Dominican-Republic'
 'Jamaica' 'Vietnam' 'Thailand' 'Puerto-Rico' 'Cuba' 'India' 'Cambodia'
 'Yugoslavia' 'Iran' 'El-Salvador' 'Poland' 'Greece' 'Ireland' 'Canada'
 'Guatemala' 'Scotland' 'Columbia' 'Outlying-US(Guam-USVI-etc)' 'Haiti'
 'Peru' 'Nicaragua' 'Taiwan' 'France' 'Trinadad&Tobago' 'Laos' 'Hungary'
 'Honduras' 'Hong' 'Holand-Netherlands']
```

Task 2: Aggregation & Filtering & Rank

In this task, we will perform some very high-level aggregation and filtering operations. Then, we will apply ranking on the results for some tasks. Pandas has a convenient and powerful syntax for aggregation, filtering, and ranking.

```
###########
# Task 2 ##
###########

#Task 2-a: Find out the race with largest number of records

#Largest_race_val  = df_data['race'].value_counts().max       37572
Largest_race  = df_data['race'].value_counts().idxmax()
print (">>Task 2-a: The Race with the largest number of records is %s"% (Largest_race))
```

```
>>Task 2-a: The Race with the largest number of records is White
```

```
#Task 2-b: #Task 2-b: Find out the total number of doctorate who are married

num_doctorate = df_data[(df_data['education']=='Doctorate') & (df_data['marital-status']=='married')]
print ("\n\n>>Task 2-b: The total number of doctorate who are married is :- %s"% (num_doctorate))
```

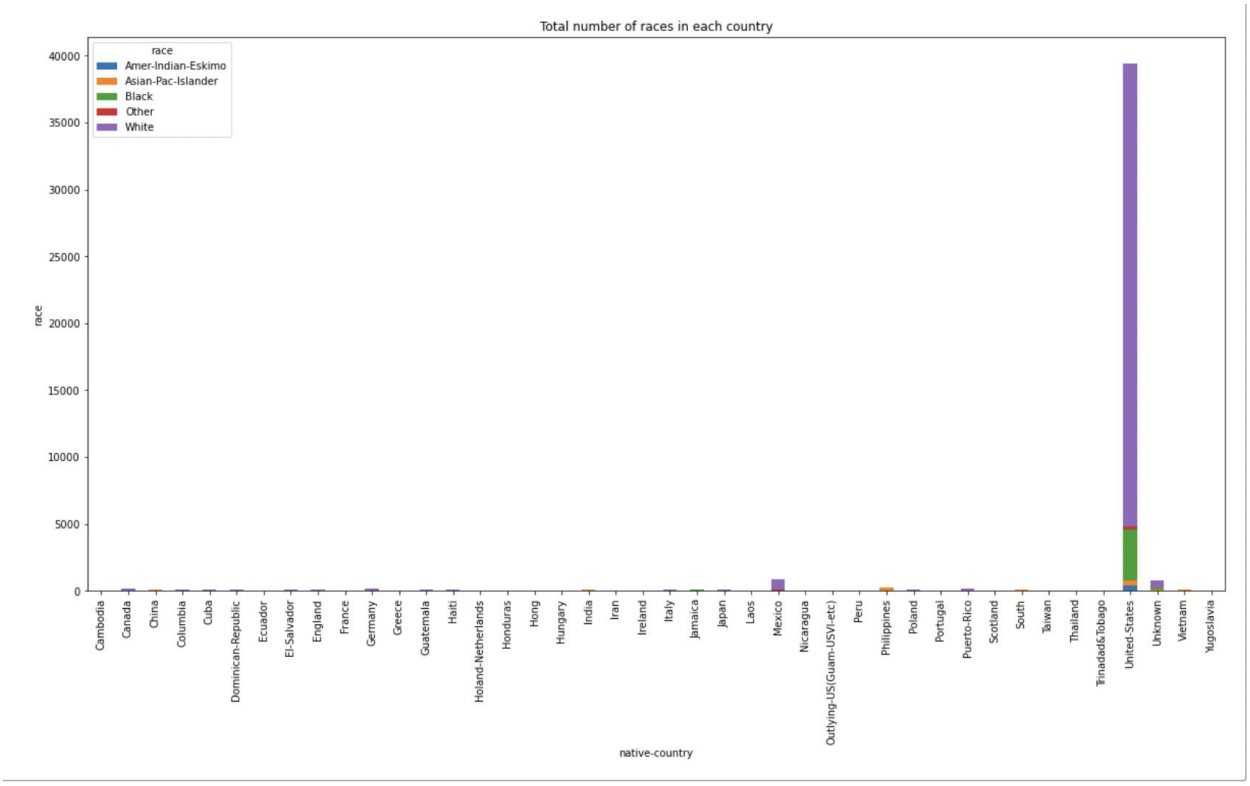
```
>>Task 2-b: The total number of doctorate who are married is :- Empty DataFrame
Columns: [age, workclass, final-weight, education, educational-num, marital-status, occupation, relationship, race,
gender, capital-gain, capital-loss, hours-per-week, native-country, income > 50K]
Index: []
```

```
#Task 2-c: Find out the top 10 countries with the highest income.

n = 10
top10_countries=(df_data.loc[df_data['income > 50K'] == "Yes"].groupby(df_data['native-country']).size().sort_value
top10_male=(df_data.loc[df_data['gender'] == "Male"].groupby(df_data['native-country']).size().sort_values(ascendin
print ("\n\n>>Task 2-c: top 10 countries with the highest income: \n%s" %
(top10_countries))
print ("\n\n>>Task 2-c: top 10 counties with the most male \n%s" % (top10_male))
```

```
>>Task 2-c: top 10 countries with the highest income:
native-country
United-States    9641
Unknown           195
Philippines        79
Canada             56
India              56
Germany            51
Mexico             42
England            39
China              34
Cuba               31
dtype: int64


>>Task 2-c: top 10 counties with the most male
native-country
United-States    26299
Mexico             681
Unknown            545
Philippines        165
India              117
Germany            110
Puerto-Rico        100
Canada              99
El-Salvador         95
China               84
dtype: int64
```

Task 3: Visualization

```
###########
# Task 3 ##
###########

# Task 3-a: Plot the declaration count for each rating .
# Think of a way to nicely visualize all the ratings!
#########################begin code for Task 3-a

plt.rcParams['figure.figsize']=(20,10)
df_data.groupby(['native-country','race']).size().unstack().plot(kind='bar', stacked=True)
plt.title('Total number of races in each country')
plt.ylabel('race')
plt.xlabel('native-country')
```



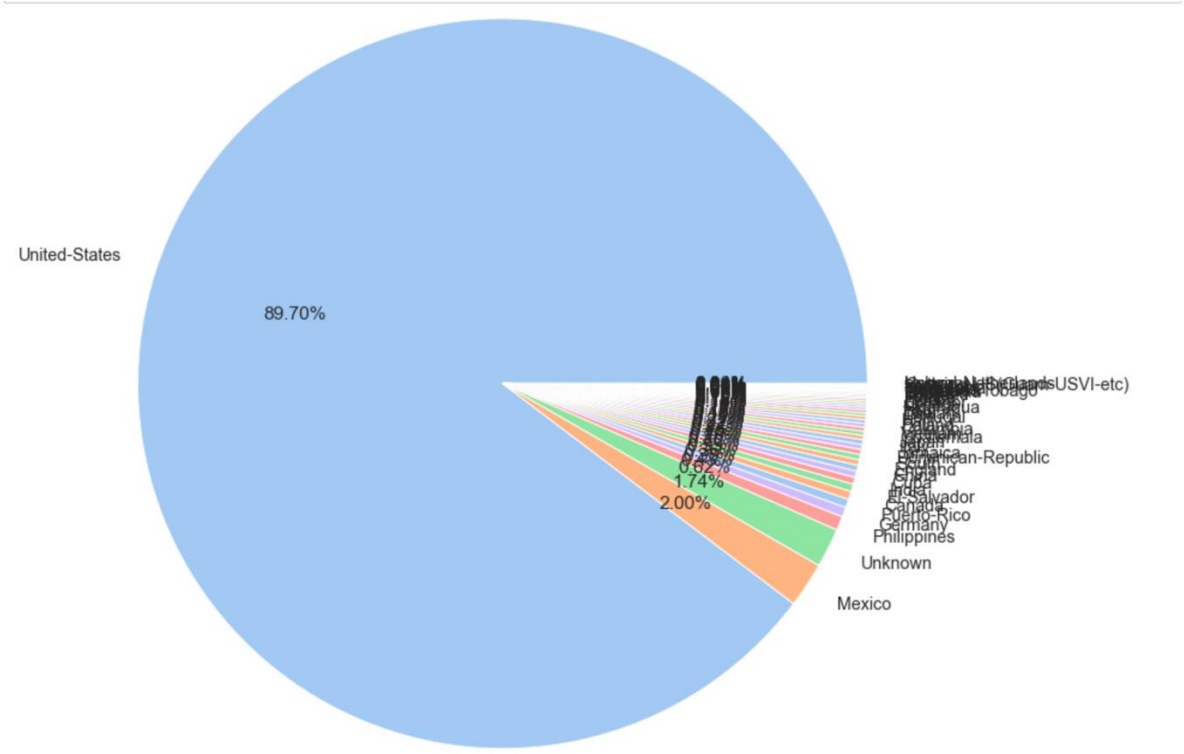
```
# Task 3-b: Draw a pie chart that represents native country

##########################begin code for Task 3-b

country_count = df_data['native-country'].value_counts().to_dict()
print(country_count)
```

```
{'United-States': 39429, 'Mexico': 880, 'Unknown': 763, 'Philippines': 273, 'Germany': 188, 'Puerto-Rico': 167, 'Canada': 158, 'El-Salvador': 145, 'India': 134, 'Cuba': 124, 'China': 113, 'England': 109, 'South': 105, 'Dominican-Republic': 97, 'Jamaica': 97, 'Italy': 94, 'Japan': 83, 'Guatemala': 79, 'Vietnam': 77, 'Columbia': 75, 'Poland': 72, 'Haiti': 71, 'Portugal': 59, 'Taiwan': 58, 'Iran': 52, 'Nicaragua': 46, 'Greece': 44, 'Ecuador': 42, 'Peru': 40, 'France': 32, 'Ireland': 32, 'Thailand': 29, 'Hong': 29, 'Cambodia': 24, 'Trinadad&Tobago': 22, 'Honduras': 20, 'Yugoslavia': 19, 'Scotland': 19, 'Laos': 19, 'Outlying-US(Guam-USVI-etc)': 19, 'Hungary': 18, 'Holand-Netherlands': 1}
```

```
labels = []
data = []
for i in country_count:
    labels.append(i)
    data.append(country_count[i])

print(labels)
print(data)
```

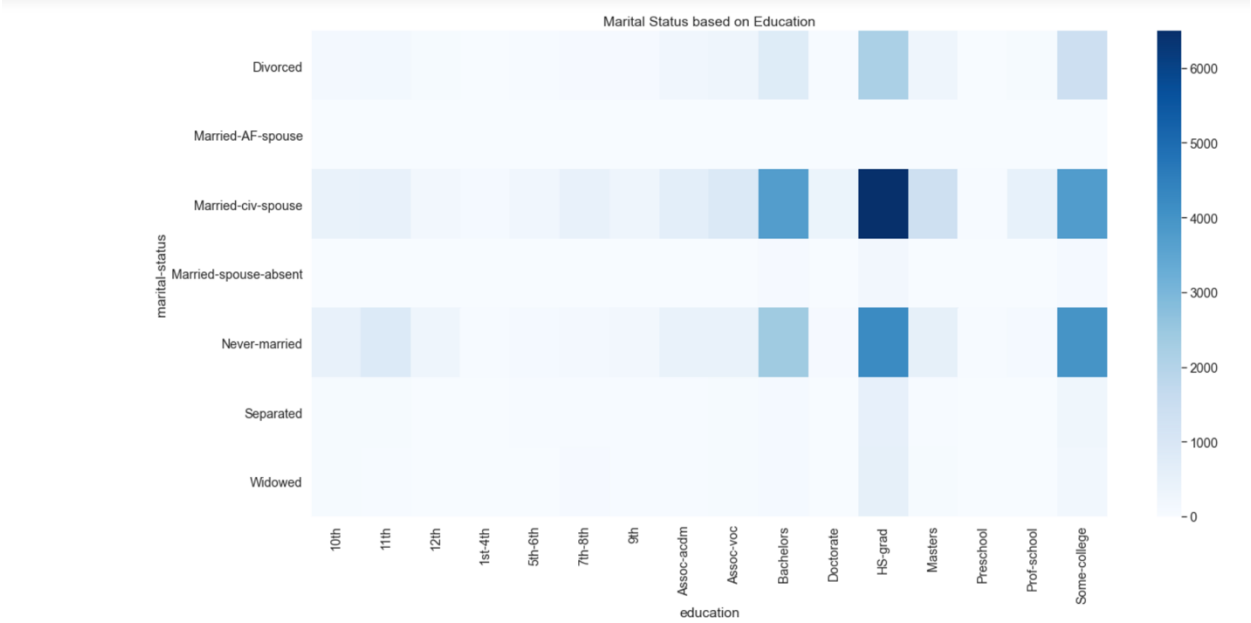
```
['United-States', 'Mexico', 'Unknown', 'Philippines', 'Germany', 'Puerto-Rico', 'Canada', 'El-Salvador', 'India', 'Cuba', 'China', 'England', 'South', 'Dominican-Republic', 'Jamaica', 'Italy', 'Japan', 'Guatemala', 'Vietnam', 'Columbia', 'Poland', 'Haiti', 'Portugal', 'Taiwan', 'Iran', 'Nicaragua', 'Greece', 'Ecuador', 'Peru', 'France', 'Ireland', 'Thailand', 'Hong', 'Cambodia', 'Trinadad&Tobago', 'Honduras', 'Yugoslavia', 'Scotland', 'Laos', 'Outlying-US(Guam-USVI-etc)', 'Hungary', 'Holand-Netherlands']
[39429, 880, 763, 273, 188, 167, 158, 145, 134, 124, 113, 109, 105, 97, 97, 94, 83, 79, 77, 75, 72, 71, 59, 58, 52, 46, 44, 42, 40, 32, 32, 29, 29, 24, 22, 20, 19, 19, 19, 19, 18, 1]
```

```
colors = sns.color_palette('pastel')[0:5]
plt.pie(data, labels = labels, colors = colors,autopct = "%0.2f%%",radius = 3.5)
plt.show()

##########################end code for Task 3-b
```



Task 4:

Find out an 'interesting' information from each one of the datasets. Create a visualization for it and explain in a few lines you're reasoning.



From the above heatmap we can see that most of the HS-grad have the marital-status of married-civ-spouse which is having the Dark Blue color whereas most of the other values are in light color.

References:-

- https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673
- https://towardsdatascience.com/data-visualization-with-pandas-1571bbc541c8
- https://towardsdatascience.com/heatmap-basics-with-pythons-seaborn-fb92ea280a6c

- https://www.analyticsvidhya.com/blog/2020/03/groupby-pandas-aggregating-data-python/