CITS3401 Data Warehousing

Final-project submission

Student – GAURAV CHAKRAVERTY (22750993)

Date – June 5, 2020

Index

Data Cleaning and Transformation

- I first downloaded the suggested "adult-training.csv" dataset from LMS.
- After that I noticed that the columns had no headers so after a bit of research I managed to find out the meaning and the possible values for each column. (reference - http://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf)
- Before I could use Weka to carry out Association rule mining, I had to solve a few problems –
    - Dataset has no header/Labels – I need to add header to the dataset.
    - Dataset contains a mixture of categorical data and numeric data – for Association rule mining to work properly in Weka I need to convert any numeric data I am using to categorical data. (Such as convert age number to "old" or "young" categories.
    - The dataset has missing values as "?" – as "?" is not a valid category and does not really give any valuable insight, I decided to remove all the rows with missing or "?" values. [To explain this further – imagine I gave categories for age such as "young","middleAge" and "old" and then I suddenly encounter a "?". That would increase the number of categories from 3 to 4 types of age values and that is not what I want and would output incorrect results when I execute the Apriori Algorithm later for example.]
    - Also the dataset included a lot of irrelevant columns that I did not need or find interesting so I would need to remove them as well.
    - Some of the columns such as education had too many categories and I am going to group them such that there are lesser categories, so it is easier to understand.
- I decided to use python (with numpy and pandas library) along with Jupiter Notebook to remove all the rows with "?" values.

```
import numpy as np
import pandas as pd
from numpy import nan
from pandas import read_csv

df=pd.read_csv('adult-training.csv',header=None, sep=',\s', na_values=["?"])
print(df.shape)
```

```
(32561, 15)
```

Had to include this as the csv file had space before the commas

```
df.replace('?', np.NaN, inplace=True)
df=df.dropna(axis='rows')
print(df.shape)
```

```
(30162, 15)
```

Managed to remove the rows with "?"/missing values.

```
df.to_csv("adult-trainingcleaned.csv", header=None);
```

- I was initially facing some problems while trying to read the csv file in python. After a bit of research online I found out that I had to use sep=',\s' because there is a space after each comma in the csv file. (reference - https://stackoverflow.com/questions/29247712/pandas-how-to-replace-with-nan-handling-non-standard-missing-values)

```
for x in range(0,(len(df[1]))):
    if (df[1][x] == 'Federal-gov') or (df[1][x] == 'Local-gov') or (df[1][x] == 'State-gov'):
        df[1][x] = 'Gov-Sector'
    elif (df[1][x] == 'Self-emp-not-inc') or (df[1][x] == 'Self-emp-inc') or (df[1][x] == 'Without-pay') or (df[1][x] == 'Never-
        df[1][x] = 'Self-Emp/No-Pay'
    elif (df[1][x] == 'Private'):
        df[1][x] = 'Prv-Sector'
```

- I then wrote a small program to group the items in the workclass column by either Gov-Sector, Prv-Sector or Self-Emp/No-Pay.
- Next, I have grouped the age column according to different age ranges and the education Level according to LowEdu(HighSchool and below), MedEdu(ex.Bachelors Degree) and HighEdu(Masters, Phd etc).

```
for x in range(0,(len(df[0]))):
    if (17<=df[0][x]<=30):
        df[0][x] = 'YoungAdult'
    elif (31<=df[0][x]<=55):
        df[0][x] = 'MiddleAged'
    elif (56<=df[0][x]<=90):
        df[0][x] = 'Old/Senior'

for x in range(0,(len(df[4]))):
    if (0<=df[4][x]<=9):
        df[4][x] = 'LowEdu'
    elif (10<=df[4][x]<=13):
        df[4][x] = 'MedEdu'
    elif (14<=df[4][x]<=16):
        df[4][x] = 'HighEdu'
```

Age was grouped according to different ranges as seen in the python program on the left. Education Level was also grouped into three categories.

Output can be seen on the right.

	A	E
1	MiddleAged	MedEdu
2	MiddleAged	MedEdu
3	MiddleAged	LowEdu
4	MiddleAged	LowEdu
5	YoungAdult	MedEdu
6	MiddleAged	HighEdu
7	MiddleAged	LowEdu
8	MiddleAged	LowEdu
9	MiddleAged	HighEdu
10	MiddleAged	

- Lastly, I grouped the number of hours worked into 4 different categories as seen below.

```
for x in range(0,(len(df[12]))):
    if (0<=df[12][x]<=20):
        df[12][x] = 'LowWorkHrs'
    elif (21<=df[12][x]<=40):
        df[12][x] = 'MedWorkHrs'
    elif (41<=df[12][x]<=60):
        df[12][x] = 'HighWorkHrs'
    elif (61<=df[12][x]<=120):
        df[12][x] = 'ExtremeWorkHrs'
```

M
MedWorkHrs
LowWorkHrs
MedWorkHrs
MedWorkHrs
MedWorkHrs
MedWorkHrs
LowWorkHrs
HighWorkHrs
HighWorkHrs
MedWorkHrs
ExtremeWorkHrs

- After that I just opened the file in Excel and added a header labels for each column. This could have been done in Python as well but I felt doing it in Excel is easier and faster.
- I also deleted the following columns which I will not be using or felt like were not relevant – fnlwgt, marital-status, relationship, race , sex, capital-gain, capital-loss ,native-country.

  [To explain this further – If I was giving someone advice after doing Association rule mining on my data – I cannot ask the person to change their gender or race so there is no point in taking such factors into consideration.

- The final result was as follows –

Age	Work-Sector	Education	Occupation	Work-Hours	Income
MiddleAged	Gov-Sector	MedEdu	Adm-clerical	MedWorkHrs	<=50K
MiddleAged	Self-Emp/No-Pay	MedEdu	Exec-managerial	LowWorkHrs	<=50K
MiddleAged	Prv-Sector	LowEdu	Handlers-cleaners	MedWorkHrs	<=50K
MiddleAged	Prv-Sector	LowEdu	Handlers-cleaners	MedWorkHrs	<=50K
YoungAdult	Prv-Sector	MedEdu	Prof-specialty	MedWorkHrs	<=50K
MiddleAged	Prv-Sector	HighEdu	Exec-managerial	MedWorkHrs	<=50K
MiddleAged	Prv-Sector	LowEdu	Other-service	LowWorkHrs	<=50K
MiddleAged	Self-Emp/No-Pay	LowEdu	Exec-managerial	HighWorkHrs	>50K

Association Rule Mining in Weka (Apriori Algorithm)

- I opened the csv file that we had created in the previous step. Then I double checked that all the data is of type Nominal and that there are no missing values.

No.		Name
1	☐	Age
2	☐	Work-Sector
3	☐	Education
4	☐	Occupation
5	☐	Work-Hours
6	☐	Income

Selected attribut

Name: Work-Hours Type: Nominal
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

No.	Label	Count	Weight
1	MedWorkHrs	18577	18577.0
2	LowWorkHrs	2388	2388.0
3	HighWorkHrs	8145	8145.0
4	ExtremeWorkHrs	1052	1052.0

- After trying out the Apriori Algorithm from the Associate tab in Weka – I carefully observed the results with the different confidence and lift values and realised that too many results included the “Age” and I cannot really ask someone to grow younger or older if suppose I found out that YoungAdults earned more money. Hence, I decided to go back and remove Age from my dataset. I felt that my results were more meaningful this way.
- Before we proceed to the next step we must first understand what the terms Support, Confidence and Lift really mean and how they are related to each other.

- Support tells us how popular an itemset is and is measured by the proportion of transactions (or rows in our case) in which an itemset appears.
  Support(X) = (Transactions containing (X))/(Total Transactions)
  Our total transactions here will be **30162** which is the number of rows in our dataset.
- Confidence tells us how likely an item Y will also be there when X is chosen. It can be calculated by finding the number of transactions(or rows in our case) where X and Y both exist, divided by total number of transactions which have X.
  Confidence(X→Y) = (Transactions containing both (X and Y))/(Transactions containing X)
  One problem with this as discussed in the lecture notes is that imagine that item X and Y both occur very frequently in our dataset. This would naturally mean what if we look at any transaction, both X and Y have a high chance to be present. This makes us think that X and Y are related to each other, but this might not be the case.
- To fix the above problem with Confidence, we have our third measure known as Lift. Lift tells us how likely item Y is likely to be there when X is there while controlling for how often Y occurs.
  Lift(X→Y) = (Confidence (X→Y))/(Support (Y))
  A lift value of more than 1 means that two items are likely to occur together.
- Now we can proceed to set the various parameters in Weka under the "Associate tab". We will be using the Apriori Algorithm which is there by default in Weka.
- I will first start by looking at Lift as we have already seen that Lift is a more reliable measure of how likely two items are likely to appear together than the confidence value alone.
- Before we begin, we much also determine our minimum support value. To get an idea of what an appropriate value for min support is – I went back and looked that the ratio of the different values compared to the total size of the dataset. We must also remember that Apriori principle allows us to prune all the supersets of an itemset which does not satisfy the minimum threshold condition for support.
- We know that we have a total of 30162 rows in our dataset. Out of those we can see below that only 7508 entries have a value of more than 50K. So only 24.89% of our entries have a income of more than 50K. Hence, lets say that for now we choose a min support value of around 0.2.

No.	Label	Count	Weight
1	<=50K	22654	22654.0
2	>50K	7508	7508.0

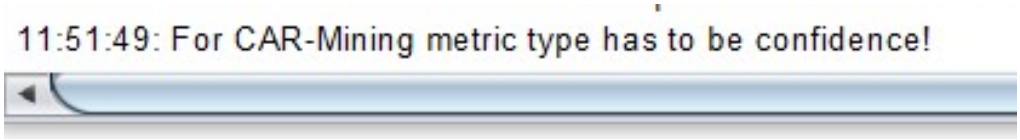
- If we go and take a look at some of the other values we can see that the number of entries where education level is high is only 2544 which is just 8.43% of the total. So, let's say that our min support now is 0.08.

No.	Label	Count	Weight
1	MedEdu	14037	14037.0
2	LowEdu	13581	13581.0
3	HighEdu	2544	2544.0

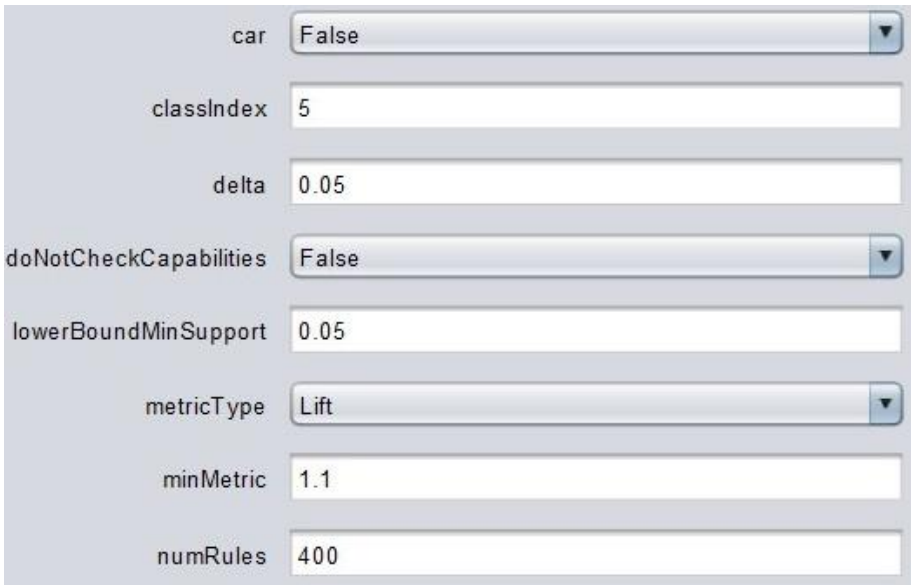
- Next I looked at the occupation column. After trying out a lot of different values for min support and observing how the results change – I have decided to proceed with a min support value of 0.05 or 5%. I believe that this number is low enough that we take into consideration the majority of our data and yet high enough to ignore some of the lower or insignificant number of occurrences.

No.	Label	Count	Weight
1	Adm-clerical	3721	3721.0
2	Exec-managerial	3992	3992.0
3	Handlers-cleaners	1350	1350.0
4	Prof-specialty	4038	4038.0
5	Other-service	3212	3212.0
6	Sales	3584	3584.0
7	Transport-moving	1572	1572.0
8	Farming-fishing	989	989.0
9	Machine-op-inspct	1966	1966.0
10	Tech-support	912	912.0
11	Craft-repair	4030	4030.0
12	Protective-serv	644	644.0
13	Armed-Forces	9	9.0
14	Priv-house-serv	143	143.0

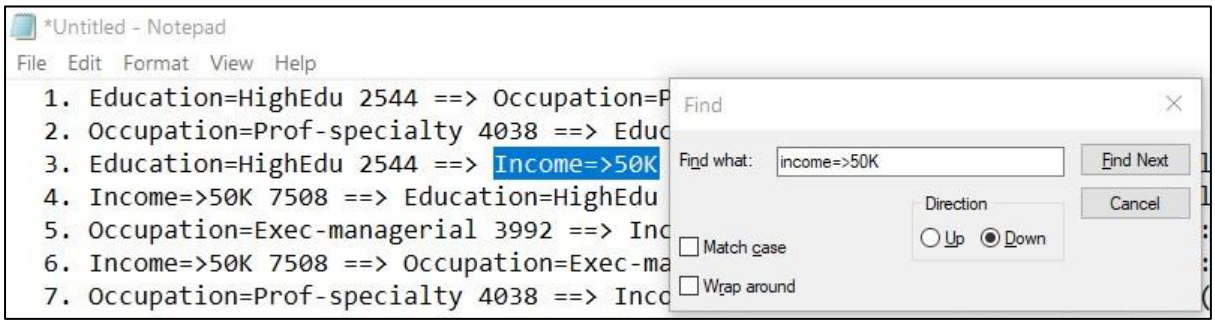
- The teacher had told us about “car” and “classIndex” attributes in Weka – basically if we set “car” as true and set the classIndex as the Index for Income, we can make sure that Income always appears on the right side when we generate rules. Thus, saving us the effort of going through each rule individually. Unfortunately, I found out that we are only able to use the above way for confidence and not Lift. Error message can be seen below.



- I set car as false, min support as 0.05 as we had decided before, metric type to Lift and min Lift value to 1.1 as any Lift value above 1 means that there is a positive correlation between two items.



- I had set the max number of rules as 400 as the number of rules with Income on the right were quite less and appeared less frequently.
- I was able to get 298 rules when I pressed the start button. However, now I had to search for the rules that were relevant to me and had Income on the right side. For this I copy pasted the entire output inside the notepad application in Windows and used the "Find" function to search and highlight rules where Income is more than 50K.



- These are the rules I found most interesting after going through all the rules –

Education=HighEdu 2544 ==> Income=>50K 1604 conf:(0.63) < lift:(2.53)>

Education=MedEdu 14037 ==> Income=>50K 4062 conf:(0.29) < lift:(1.16)>

[Notice how even though there is a positive association between having a medium level of education and earning a higher income, the Lift and confidence values are much lower than that of the previous rule. Therefore, a higher amount of education is more likely to result in a higher income.]

Occupation=Exec-managerial 3992 ==> Income=>50K 1937 conf:(0.49) < lift:(1.95)>

Occupation=Prof-specialty 4038 ==> Income=>50K 1811 conf:(0.45) < lift:(1.8)>

Education=MedEdu Work-Hours=HighWorkHrs 4170 ==> Income=>50K 1859 conf:(0.45) < lift:(1.79)>

Work-Hours=HighWorkHrs 8145 ==> Income=>50K 3349 conf:(0.41) < lift:(1.65)>

Work-Sector=Prv-Sector Work-Hours=HighWorkHrs 5739 ==> Income=>50K 2238 conf:(0.39) < lift:(1.57)>

Work-Sector=Prv-Sector Education=MedEdu 10184 ==> Income=>50K 2792 conf:(0.27) < lift:(1.1)>

- Then Top 5 rules according to me are as follows –

Education=HighEdu 2544 ==> Income=>50K 1604 conf:(0.63) < lift:(2.53)>

Hight level of education (Masters, PHD etc) makes it more likely to have a high income.

Work-Hours=HighWorkHrs 8145 ==> Income=>50K 3349 conf:(0.41) < lift:(1.65)>

People who work between 41 and 60 hours per week are likely to have a higher income. This could be because even though someone may not each that much per hour – they may be able to work extra hours or overtime to get additional income.

Occupation=Exec-managerial 3992 ==> Income=>50K 1937 conf:(0.49) < lift:(1.95)>

Occupation=Prof-specialty 4038 ==> Income=>50K 1811 conf:(0.45) < lift:(1.8)>

I would like to group these two rules together as they basically say that the above two occupations are most likely to earn an income of more than 50K.

Work-Sector=Prv-Sector Education=MedEdu 10184 ==> Income=>50K 2792 conf:(0.27) < lift:(1.1)>

If someone works in the private sector and at least has a medium level of education (ex.Bachelors Degree) then they are more likely to have a higher income.

Work-Sector=Prv-Sector Work-Hours=HighWorkHrs 5739 ==> Income=>50K 2238 conf:(0.39) < lift:(1.57)>

While it is true that working for more than 41 hours per week will increase the likelihood of having a higher income – working in the private sector would be the best way to do so. This might be because for example if you are self employed then working for longer may not always mean then you earn more.

- Based on the above rules, if I had to provide a recommendation for a person who wants to improve their income, my recommendation would be as follows –
  "Education is very important. High level of education(ex. Masters/PHD) increases the likelihood of having a high income. Also, we must be prepared to work for more than 41 hours per week if we want to achieve a high income. Lastly, even if you only have a medium level of education, then being in the private sector would provide us with the most amount of opportunity to grow our income."

Attribute Selection

- First, I went and added a few more columns to the data set that we have been using so far. The dataset that we will be using for our comparison in following sections have the following attributes –
    - Age
    - Work-Sector
    - Education
    - Marriage-State
    - Occupation
    - Family-Relation
    - Race
    - Gender
    - Work-Hours
    - Location
    - Income
- The attributes that I have chosen based on my understanding/Intuition are as follows –
    - Age
    - Work-Sector
    - Education
    - Occupation
    - Work-Hours
    - Income
- Next I had to use Weka to choose the top 5 attributes based on Information Gain. Before we proceed, we should first understand what Information Gain means and how attributes are ranked/chosen based on Information Gain.
- Entropy is measure of uncertainty associated with a random variable. Information Gain calculates the effective change in entropy after making a decision based on the value of an attribute.

  Information gain = (Entropy of distribution before the split) – (entropy of distribution after it)

- When we rank attributes based on Information Gain, attributes with the highest information gain are at the top.
- Weka allows us to rank attributes based on Information gain using "InfoGainAttributeEval" under the "Select attributes" tab. The Search Method is Ranker by default.
- As seen in the next page, the top 5 attributes based on Information Gain are as follows –
    - Family-Relation
    - Marriage-State
    - Occupation
    - Age
    - Education

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 11 Income):
        Information Gain Ranking Filter

Ranked attributes:
 0.16618    6 Family-Relation
 0.15747    4 Marriage-State
 0.09319    5 Occupation
 0.06918    1 Age
 0.06764    3 Education
 0.04752    9 Work-Hours
 0.03741    8 Gender
 0.01025    2 Work-Sector
 0.00933   10 Location
 0.00829    7 Race

Selected attributes: 6,4,5,1,3,9,8,2,10,7 : 10
```

It seems that Work-Sector and Work-Hours are not as important when we rank attributes by Information Gain.

- I attempted to manually calculate the Information Gain value for the “Education” to cross check if I get the same value using Weka.

```
EntropyWhole: 0.8095658329614156
MedEduEntropy: 0.8679165054605544
LowEduEntropy: 0.5726805129831835
HighEduEntropy: 0.9502849813044842
EntropyEducation: 0.7419283544331619
EducationInformationGain: 0.06763747852825375
```

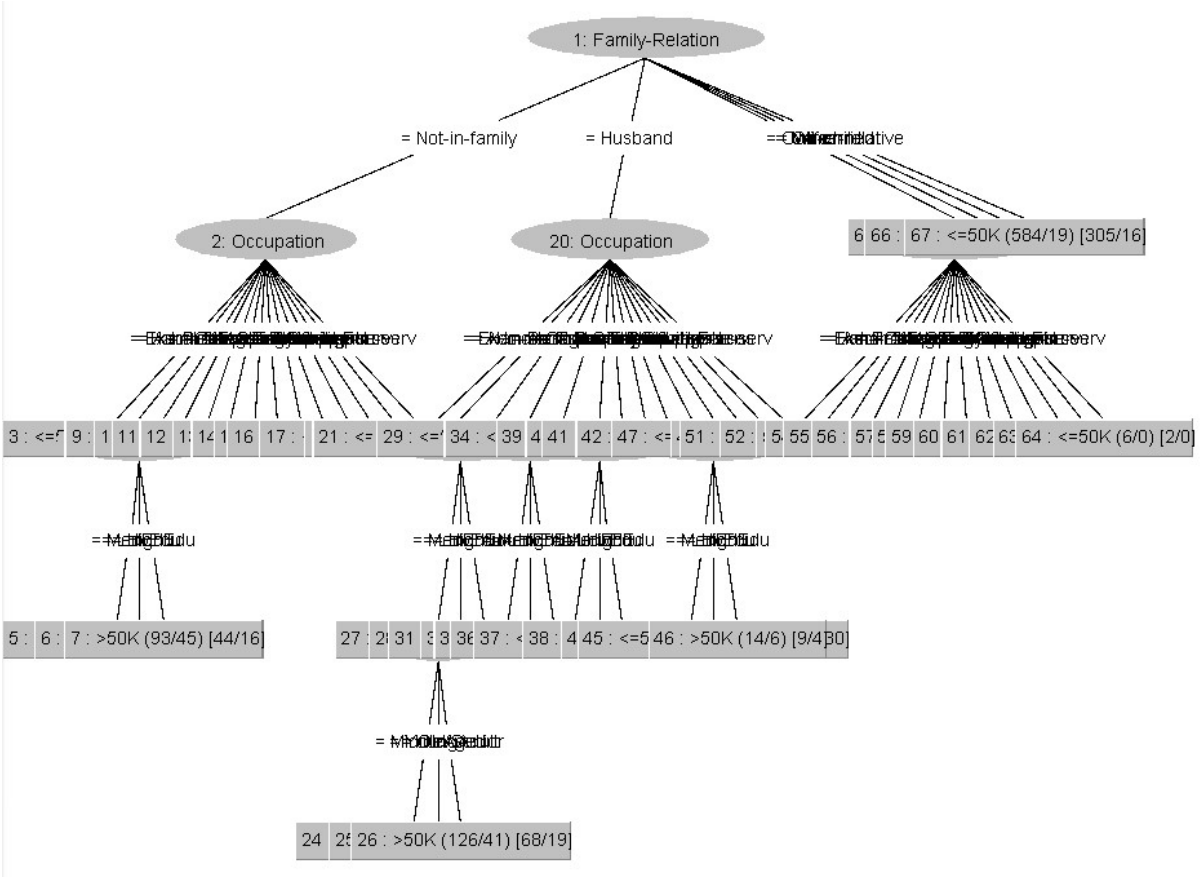
- The value for the Information Gain for Education that I obtained matches the value from Weka. The detailed steps can be seen on the next page.

```
import math
# First I calculate the entrophy of the whole data set
# entries with low income/total entries
x = 22654/30162
# entries with high income/total entries
y = 7508/30162
EntropyWhole = -(x*math.log2(x) + y*math.log2(y))
print("EntropyWhole:",EntropyWhole)

# Then I calculate the expected new entropy for Education
# Education has 3 distinct types of values (MedEdu,LowEdu and HighEdu)
# MedEdu has 14037 records 4062 of which corrospond to HighIncome.
x = 9975/14037
y = 4062/14037
MedEduEntropy = -(x*math.log2(x) + y*math.log2(y))
print("MedEduEntropy:",MedEduEntropy)

# LowEdu has 13581 records, 1842 of which corrospond to HighIncome.
x = 11739/13581
y = 1842/13581
LowEduEntropy = -(x*math.log2(x) + y*math.log2(y))
print("LowEduEntropy:",LowEduEntropy)

# HighEdu has 2544 records, 1604 of which corrospond to HighIncome.
x = 940/2544
y = 1604/2544
HighEduEntropy = -(x*math.log2(x) + y*math.log2(y))
print("HighEduEntropy:",HighEduEntropy)

EntropyEducation = (14037/30162*MedEduEntropy + 13581/30162*LowEduEntropy
                    + 2544/30162*HighEduEntropy)
print("EntropyEducation:",EntropyEducation)

EducationInformationGain = EntropyWhole - EntropyEducation
print("EducationInformationGain:",EducationInformationGain)
```

- Next we will proceed to build two separate Decision Trees based on the top 5 attributes chosen by Information gain vs my 5 chosen attributes and compare and interpret the results.

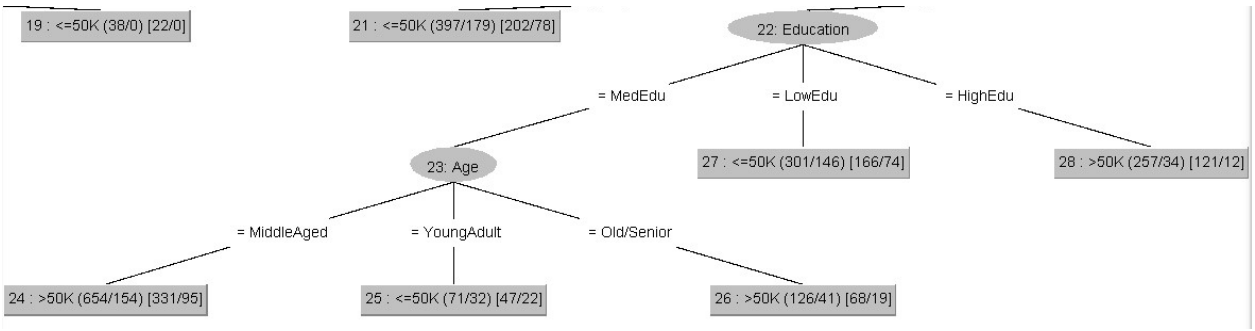
Decision Tree Model

- First, we will create the decision tree model based on the top 5 attributes selected through Information Gain –
    - Family-Relation
    - Marriage-State
    - Occupation
    - Age
    - Education
    - Income (This is the Target Variable)
- I selected REPTree under the “Classify” tab in Weka and changed the Test option to !0 Fold Cross-validation. REPTree builds a decision tree based on Information Gain. I then pressed the start button and the following decision tree was generated –



- The tree unfortunately is very wide and hence is a bit hard to understand in the screenshot above. The main reason why the tree is so wide is because of the Occupation attribute which has 14 distinct possible values. I tried to increase the “minNum” value to 100 which is the minimum total weight of the instances in a leaf. The reee did look slightly better but still too wide.
- However, when I zoom into each of the individual sections of the tree I can see that the tree at the top has Family-Relation which has the highest information gain value. Then the tree branches out to show the various Occupations and how the income is distributed. Then for some of the Occupations the tree splits further to show the how

the income is distributed amongst different Education Levels. Then in one of the branches, MidEdu splits even further to show how different age groups earn differently.



- In the above section of the tree that I have zoomed into, we can observe how for one of the occupations it shows how HighEdu is related to higher income and then within MedEdu we can further see how YoungAdults seem to earn lesser compared to Middle aged or Senior people.

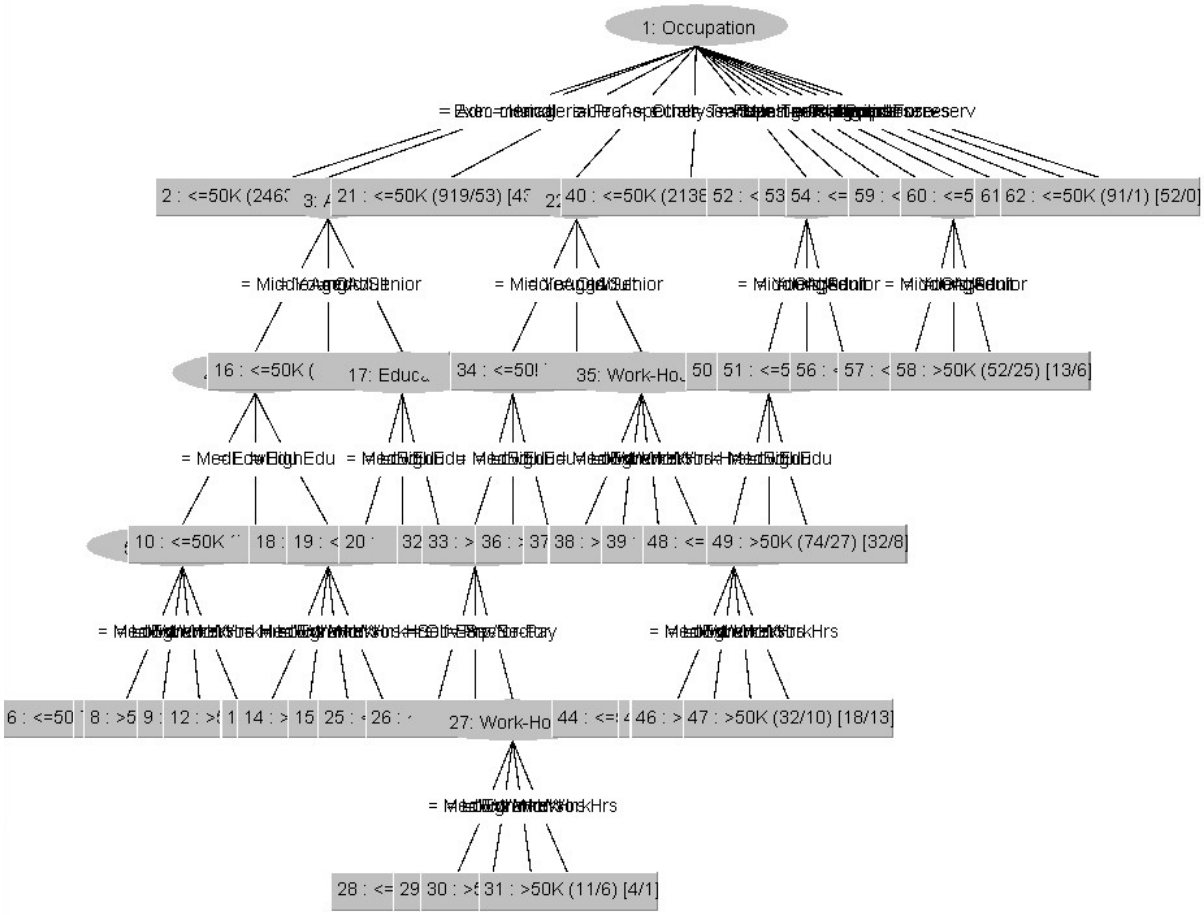
Correctly Classified Instances	24739	82.0204 %
Incorrectly Classified Instances	5423	17.9796 %

	TP Rate	FP Rate	Precision	Recall
	0.930	0.511	0.846	0.930
	0.489	0.070	0.698	0.489
Weighted Avg.	0.820	0.401	0.809	0.820

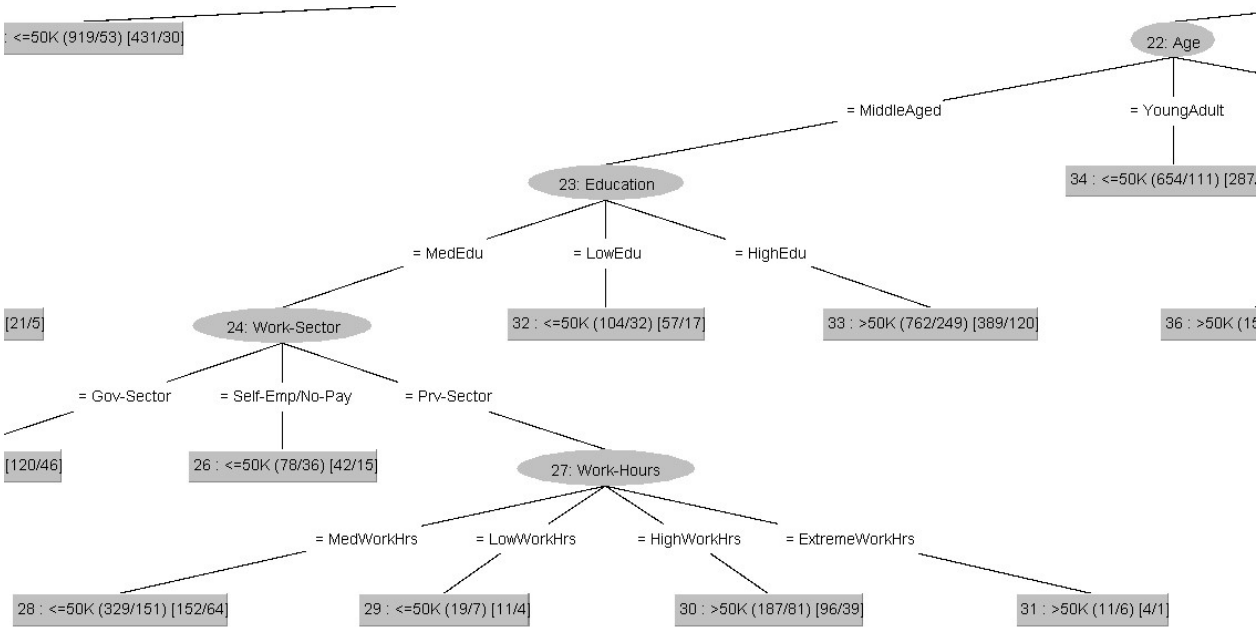
- Weka also shows us some useful information as seen above that we can use to compare our models. Precision tells us what % of instances that the classifier labelled as positive are actually positive – also known as the exactness of the model. Recall tells us what % of positive instances the labelled as positive – also known as the completeness of the model.
- We will now proceed to create a decision tree based on the top 5 attributes that I selected –
    - Age
    - Work-Sector
    - Education
    - Occupation
    - Work-Hours
    - Income (This is the Target Variable)

```
Ranked attributes:
 0.0932  4 Occupation
 0.0692  1 Age
 0.0676  3 Education
 0.0475  5 Work-Hours
 0.0103  2 Work-Sector
```

- I have also ranked my selected attributes according to Information Gain because I know that the REPTree model I am going to generate next is going to make the decision tree based on information gain.



- The above is the tree that I was able to generate based on my selected attributes. When I zoom into one of the sections of the tree I can see that Occupation is at the top of the tree (as it has the highest Information gain value) after that the tree branches out according to the Age then Education level then the Work-Sector and lastly the Work-Hours.



- It seems that the people with the highest education level and are older are likely to earn more as seen in the previous tree. This tree also shows that if someone has a Medium level of education and works in the Private Sector then it is more likely that they can earn more money if they work for High or Extreme number of hours per week. (41 to 120 hrs per week).

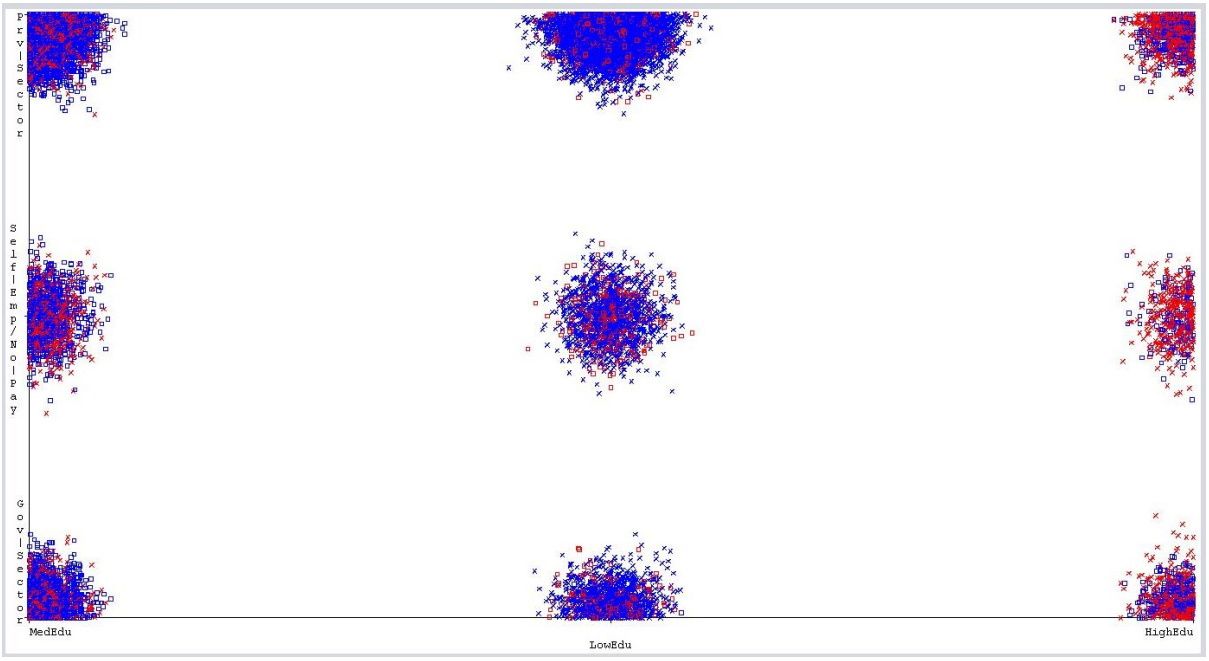
Correctly Classified Instances	23721	78.6453 %
Incorrectly Classified Instances	6441	21.3547 %

	TP Rate	FP Rate	Precision	Recall
	0.935	0.663	0.810	0.935
	0.337	0.065	0.633	0.337
Weighted Avg.	0.786	0.514	0.766	0.786

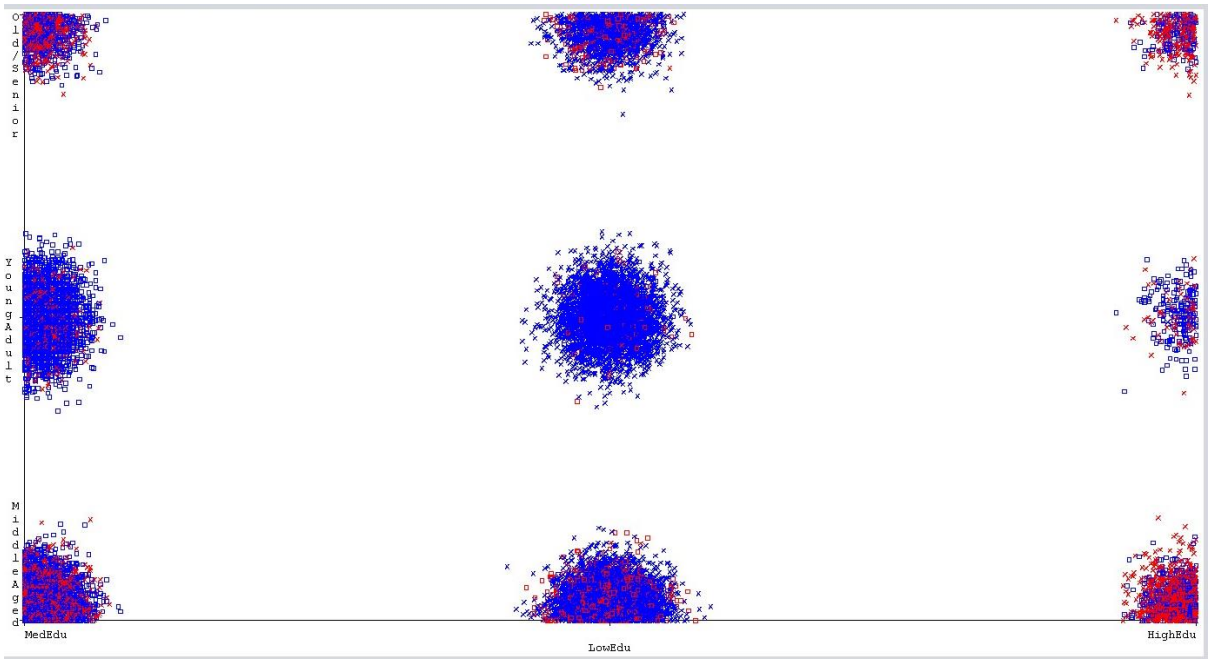
- As seen above it seems that the % of correctly classified instances in lower for this tree. The Precision and Recall vales are also lower compared to the previous decision tree.
- This shows that the decision tree will give a more accurate output if we select attributes based on information gain when possible.

Clustering

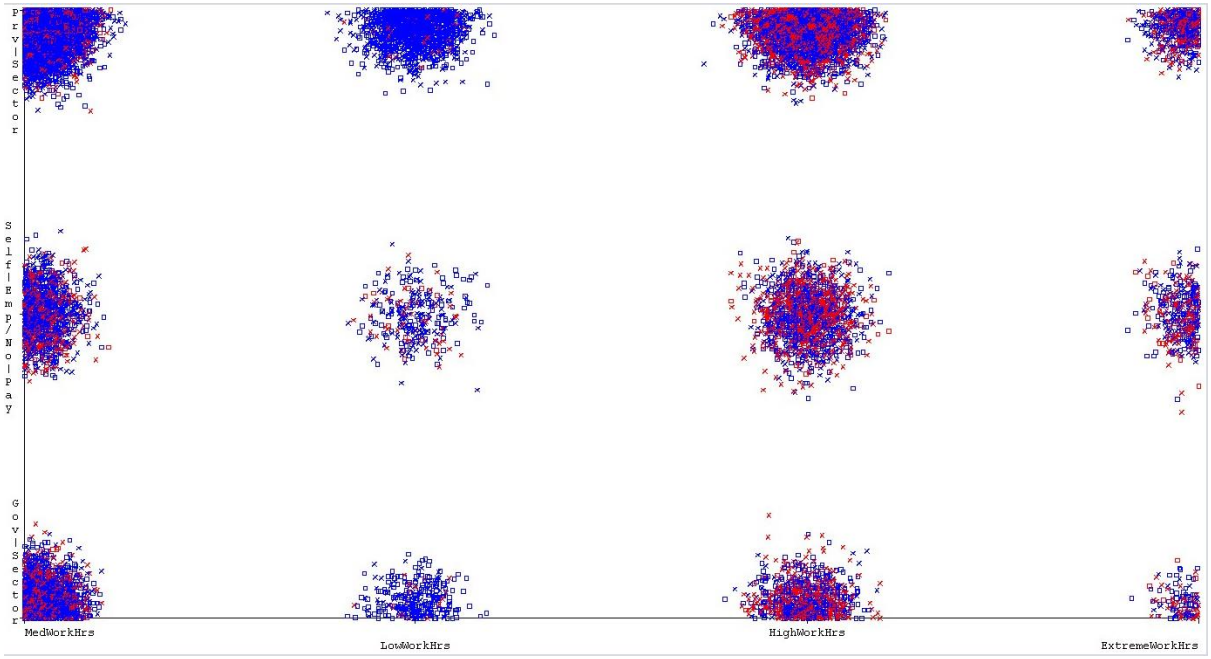
- Clustering enables us group objects similar to one another within the same cluster and dissimilar to the objects in other clusters.
- I will use my previous dataset which had the following attributes – (All the attributes were converted to Nominal).
    - Age
    - Work-Sector
    - Education
    - Occupation
    - Work-Hours
    - Income
- Under the “Cluster” tab in Weka, I chose SimpleKMeans. I set my number of clusters as 2. (For HighIncome and LowIncome). I set the cluster mode as “Classes to clusters evaluation” and selected the Income attribute.
- I then visualised the Cluster assignments as seen below –



- I selected Education for the X-axis, Work-Sector for the Y-axis and Red color represents a HighIncome and blue represents LowIncome. We can clearly see in the visualisation above that people with High amount or Education also are more likely to get higher Pay(red color) regardless of the Work-Sector.



- In the next visualisation I changed the Y-axis to represent Age and we can see that there are 2 observations –
    1. Middle-Aged and Old/Senior people seem to earn more than Young-Adults.
    2. Low amount of Education leads to a lower income regardless of the age.



- In the above visualization I chose Work-Hrs as the X-axis and Work-Sector as the Y-axis. It can clearly be seen than people who work for a high or extreme number of hours tend to earn a higher income(red) across all sectors.

```
Class attribute: Income
Classes to Clusters:

     0     1  <-- assigned to cluster
 11119 11535 | <=50K
  5716  1792 | >50K

Cluster 0 <-- >50K
Cluster 1 <-- <=50K

Incorrectly clustered instances :       12911.0  42.8055 %
```

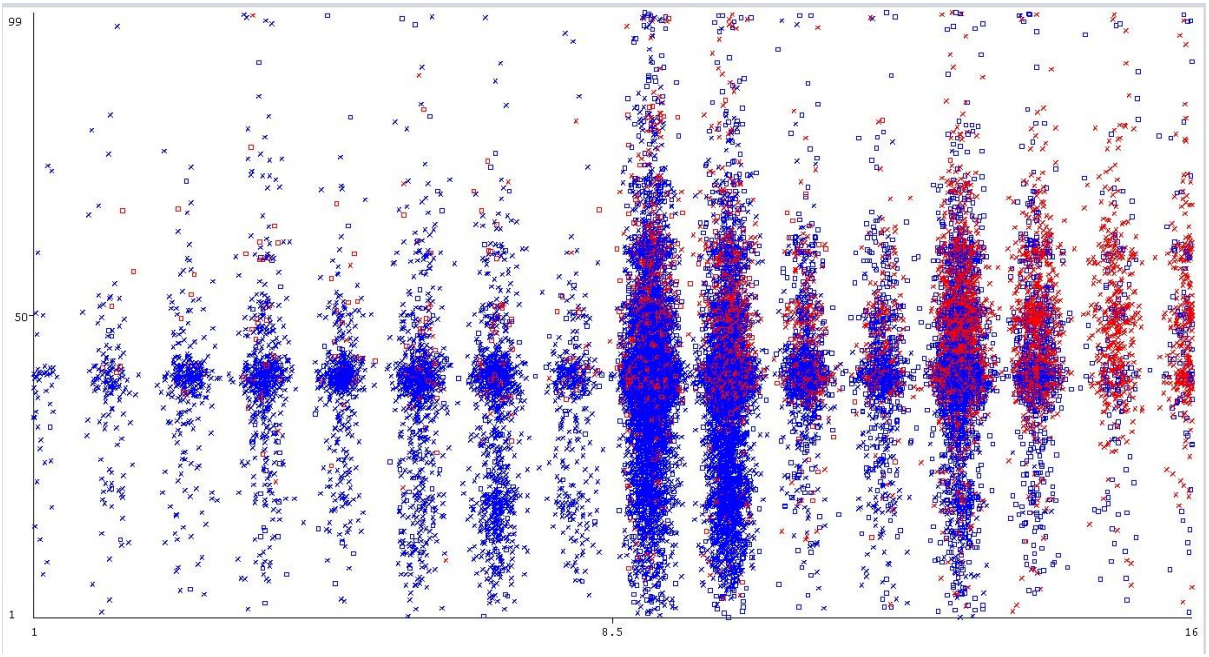
- Here we can see that the Incorrectly clustered instances in 42.8%. We can get the same number my looking at the confusion matrix above – 11119 instances that were supposed to be LowIncome were identified as HighIncome and 1792 instances that were supposed represent HighIncome were identified as LowIncome. If we add 11119 + 1792 then we get 12911 which is the number of incorrectly clustered instances.
- A reason for this is because in Weka, only Euclidean Distance and Mahattan Distance are supported, so k-means in Weka works best for numerical values.
- I wanted to find out if I could get a lower value for the Incorrectly clustered instances, so I went back to the pre-processing tab and converted all the nominal attributes to Binary using “NominalToBinary”.

```
     0     1  <-- assigned to cluster
 20959  1695 | <=50K
  7274   234 | >50K

Cluster 0 <-- <=50K
Cluster 1 <-- >50K

Incorrectly clustered instances :       8969.0   29.7361 %
```

- I was able to decrease the number of Incorrectly clustered instances but then I realised that I could no longer make sense of the data while visualising it. This is because for example now Education and Work-Sector was split into 3 different attributes each and Weka only allowed me to choose a single attribute for the X and Y axis. This made it very difficult to compare the data.
- Lastly, I went back to my CSV file and replaced the nominal Age, Education and Work-Hrs values to the numeric ones found in the original dataset.

```
    0     1  <-- assigned to cluster
 8616 14038 | <=50K
 6066  1442 | >50K

Cluster 0 <-- >50K
Cluster 1 <-- <=50K

Incorrectly clustered instances :       10058.0   33.3466 %
```

- The accuracy was still better compared to having all Nominal Values and I was able to compare and make sense of the visualised data.



- I set the Work hours as the Y-axis and Education as the X-axis like before but this time we have numeric values. We can still see that as increase in Work-Hrs and Education correspond to higher Income value(red color).
- Therefore, I would say that if we want to do clustering and have numeric values then we should not convert it to categorical values as this would lead to a lower accuracy.

Conclusion

(Interpret 3 types of learning and their relations in the context of the dataset)

Association rule mining and Clustering were actually a type of unsupervised learning. When we do unsupervised learning, we do not specify a target variable. When we executed the Apriori Algorithm, it was able to find various association rules based on support, confidence and lift even though we did not specify the target variable. We then chose the rules that were of interest to us. Classification was a type of supervised learning and used “income” as the target variable. Supervised learning always requires a target variable to train the model on.

Overall, even though the 3 types or learning processed the dataset differently (unsupervised vs supervised), lift vs information gain etc. The final result I got was similar. In all three types of learning I found common patterns such as High Education leads to an increased likelihood of increased Income. Thus, depending on the type of data we have I think that we could choose a learning method accordingly. (For example, we say that Clustering does not work very well with Categorical/Nominal data).