**American Adult Census Research**

**Xin Li, Xi Wang, Zikun Lin**

li.xin125@husky.neu.edu

wang.xi4@husky.neu.edu

lin.zik@husky.neu.edu

**Abstract**

Acquiring the access to the adult census income data recourses provide us with the basic resources to make analysis of adult income. This dataset was extracted from the 1994 Census bureau database by Ronny Kohavi (https://www.kaggle.com/uciml/adult-census-income) and provides us with multiple features such as nationality, race, gender, education, etc. Meanwhile, the cutting-edge machine learning algorithms and models assist us to process the data to discover the adult income level classification and the features association level.

In this research, we begin with extracting adult census dataset, then training the data in order to eliminate the noise and incomplete dataset. In the end, machine learning algorithms and models will be used to generate the classification of the adult census, the association degree of the features to the income of the each adult and the prediction of adults income based on the features in the dataset.

**Introduction**

As the Data Mining and Machine Learning gradually became the world’s most popular topic. We found it extremely interesting when it comes to analyze data. After careful consideration, we decided to use the american adult census income data to do the research about the performance of different algorithms in different conditions.

We are going to utilize the first three algorithms to classify the income level of an adult with given features, and the last algorithm to extract the association rules in this data set.

* Decision Tree

This algorithm is easy to implement and convenient to explain. When utilizing decision tree, there is no need for us to make intricate preparation of the dataset comparing to other algorithms.

In addition, this algorithm does not take nonlinear relationship into account, which allows us to make a comparison with SVM.

* Naive Bayes

This algorithm is also easily to implement and convenient to explain. However it works really well in real life situation. Besides, the attributes in the dataset we have does not have strong dependency.

So, it is very suitable for our data set.

* Support Vector Machines

After analyzing the given data set, we find the data of the data set is separate obviously, there are distinct margin between each classes, this creature of our data set will not lead to a overfitting condition.

Furthermore, consider our result might not be linear regression, and Support Vector Machines can handle non-linear regression pretty well, so we choose this method, rather than other algorithm like logistic regression.

* FP-Growth algorithm

There are two good association analysis algorithm we considered—FP-Growth and Apriori.

Apriori is designed for mining frequent itemsets and association rules. This algorithm makes many searches in database to find frequent itemsets where k items are used to generate k+1-itemsets. Each k-itemset must be greater than or equal to minimum support threshold to be frequency. Otherwise, it’s called candidate itemsets.

Different with Apriori algorithm, FP-Growth allows frequent itemset discovery without candidate itemset generation. It encodes the data set using a compact data structure called FP-Tree and extracts frequent itemsets directly from this tree.

After the careful consideration, we decide to use F-P Growth instead of Apriori algorithm because of the efficiency. It’s not necessary for us to require low limited time and space to run the algorithm, because the dataset is huge, which contains more than 30000 rows..

We’re hoping to get several features to predict the income level of an adult in America. To evaluate and compare different kinds of classify algorithms, we calculated the accuracy of our model. After comparing the accuracy of the first three algorithms, we are able to know which one solve this problem better.

**Dataset**

**Introduction**

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). The prediction task is to determine whether a person makes over $50K a year.

The attributes are listed as following sheet:

|  |  |
| --- | --- |
| Column | Values |
| income | >50K, <=50K |
| age | Continuous |
| workclass | Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked |
| fnlwgt | Continuous |
| education | Bachelors, Some-college, Doctorate, 5th-6th, Preschool, etc. |
| education-num | Continuous |
| marital-status | Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse |
| occupation | Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, etc. |
| relationship | Wife, Own-child, Husband, Not-in-family, etc. |
| sex | Female, Male |
| capital-gain | continuous |
| capital-loss | continuous |
| hours-per-week | continuous |
| race | White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black |
| native-country | United-States, Cambodia, England, Puerto-Rico, Canada, etc. |

**Data Clean**

As we can see several attributes listed above are either useless or continuous. And there are many false data in the dataset, so we need to clean the data, in order to make our algorithm works more efficient.

In the cleaning section, we used the public library of python which is called pandas. It helps us process the data more effectively.

**Data Clean For Decision Tree**

The attributes—fnlwgt, capital-gain, capital-loss—have nothing to do with the income, so we decided to drop then. In the remaining attributes, the age education-num and hours-per-week are continuous. The range of education-num is from 6 to 20, so we basically do not need to change too much.

However, the range of age and hours-per-week is too wide and continuous. When we used decision tree for the first time, the tree got too big, and there were tremendous number of test cases failed because of the continuous attributes. For instance, after we trained thousand of data, there is still a possibility that there is some age number that has never been shown before, which results in the decision tree failing to find the match value in the decision node.

So we create two new attributes to discretize them. Basically, what we did is that we categorized them into several groups, the group range is 10. So, they are shown in the following sheet:

|  |  |
| --- | --- |
| Column | Values |
| agegroup | 10 - 19, 20 - 29, 30 - 39, 40 - 49, 50 - 59, 60 - 69 |
| hoursgroup | 10 - 19, 20 - 29, 30 - 39, 40 - 49, 50 - 59, 60 - 69 |

**Project description**

**Decision Tree**

The first Algorithm we implemented is the decision tree.

A decision tree represents a function that takes as input a vector of attribute values and returns a decision. The input and output values can be discrete or continuous. For this research we will concentrate on discrete values and the output is a binary, which means there will be only two different output.

For every decision tree, it has three parts: Decision Nodes, Chance Nodes and End Nodes. To create the decision tree, we need to choose every decision node carefully, which means we need to choose every attribute that can separate the remaining data most distinctly.

Meanwhile, when we try to build a decision tree, we need to find out the best tree in a short time. However, if we try every possibility in order to create the optimal decision tree, the number of distinct truth tables with rows will be . This will take a lot of time. So in order to extract attributes more quickly, we have to use *entropy*.

Entropy is a measure of the uncertainty of a random variable acquisition of information corresponds to a reduction in entropy.

Entropy:

We can define as the entropy of a Boolean random variable that is true with probability :

The entropy of the goal attribute on the whole set is:

So an attribute has *d* distinct values we need to calculate all the information from each value, so the expected remaining after testing attribute A is:

In the end, what we are going to compare is the information gain of each attribute:

The best attribute we will choose is the attribute which has the highest information gain.

Choosing the decision node using the entropy, we can create a tree layer by layer, in the end to predict the test data.

**Naïve Bayes**

The second Algorithm we chose is Naive Bayes.

In Naive Bayes model we assume that all the features are independent to each other, that is why we call it naive. However, this model is quite useful for our data set, because the target output is binary and all of the features are discrete.

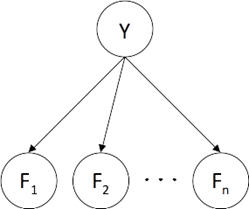


Figure-1: Naive Bayes Net

According to Figure-1, we can see the basic structure of naive bayes. For each feature we have the probability of this feature given the output:

In order to compute the possibility of the output Y. we need all the probabilities of all features. According to the bayes rule, we will have:

The prediction will be:

Before we start to predict the data, we have to train the data in order to find out the probability table. In the training process of naive bayes model, we need to calculate all the empirical probability of each parameter:

**SVM(Support Vector Machine)**

SVM is a supervised learning model, this kind of model is able to handle labeled data. Adult census income data are labeled in a way that whether individual’s annual income is more than 50K or not. Thus SVM is a suitable algorithm.

The purpose of the SVM is to find a (p-1)-dimensional hyperplane to separate data points which are viewed as p-dimensional vectors.

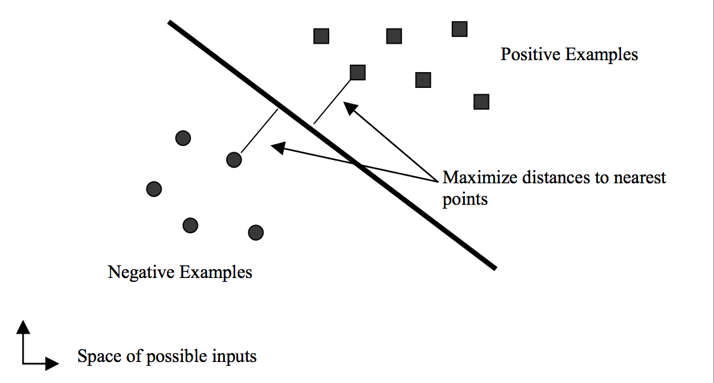
For getting the hyperplane, we need to solve the dual problem :

When we solve the dual problem, we need to satisfy the KKT condition:

As long as we find a group of , that satisfies the condition above, we find an optimal solution. Thus our aim is to find a group of optimal , when we get the , it will be easily for us to calculate the weight vector and b.

After that, the hyperplane can be gotten by and b.

Here is an example that the hyperplane separates points with two different labels:



We use the SMO(Sequential Minimal Optimization) algorithm to implement SVM, this algorithm divides a big problem into some sub-problem to solve, which makes converges faster than other algorithm.

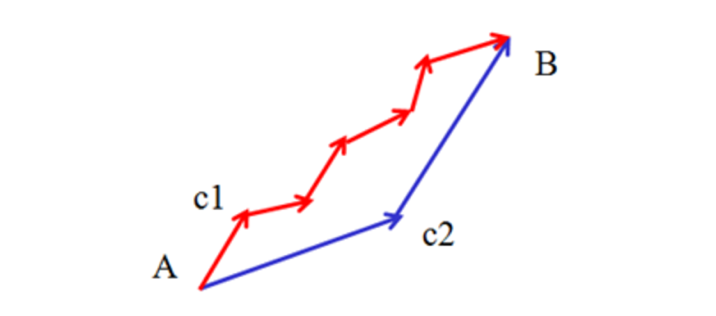
For implementing SMO, we need to repeat the following procedure until converges:

{

1. Chose
2. Only based on to update w(), and make other unchanged
3. Update the intersection b based on updated

}

In every iteration we update two , and keep other unchanged. If we have N samples to train, we will have N(N-1) ways to chose . Because our aim is to correct those samples who violate the KKT condition, if every sample satisfies KKT condition, we will finish our optimization. Thus, we will choose the who violates the KKT condition most. For choosing , if we choose the which will correct the length of step most, we can get convergence fast. The idea of choosing that we can get convergence fast is like we choose the blue way.



More specifically, the way to choose to update is, we need to find the data that violates the KKT condition most, then get the that this data correspond to. While finding the , we need to iterate those data which satisfies the condition, those data samples are on the border of the separating plane, and then we check if there is a sample violates the KKT condition. If there is no sample violates the KKT condition in the dataset that satisfies the condition, we need to iterate the whole training dataset to find the data that violates the KKT condition most. the way to find the is if we find the in the first step, we will choose the that maximize the |Ei-Ej| most. For saving calculating time, we created a cache to storage the deviance of the every sample.

Before begin the SVM classification, we need to determine the slack variable C. The value of slack variable we set, determine what extend we can accept the extend of sample’s deviance. If we set a big value for slack variable, it means we are unwilling to give up some deviant samples.

When we set the value of slack variable too big, sometimes we cannot find the solution of the SVM. However, if we set the value of variable C too small, it means we will give up too much samples, this will reduce the accuracy of SVM. During the experiment of SVM, if we set the variable C small, our program will run a longer time, because we give up too much sample, this makes SVM hard to find the hyperplane for classifying. If we give it a big value, it will lead to there is no hyperplane can separate all the samples with limited slack variable.

The training time of training 4000 samples for different C:

|  |  |
| --- | --- |
| C = 0.001 | 7.343929s |
| C = 0.1 | 26.7343929s |

Because our data is high-dimension, use kernel-function can make our SVM faster. We tested the two kernel-function “linear-kernel” and “rbf-kernel” for the performance of classification , linear-kernel was much faster than “rbf-kernel”.

The training time of training 4000 samples for differnet kernel-fucntion:

|  |  |
| --- | --- |
| linear-kernel | 7.343929s |
| rbf-kernel | 156.896534s |

**Explanation:**

Even if the linear-kernel is a degenerate version of rbf-kernel, their predict accuracy is no different for the adult census income data. Because the number of features is large, there is no need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance.

During the experiment, sometime increasing the sample size does not help the classification performance:

|  |  |
| --- | --- |
| Number of sample: | Prediction accuracy: |
| 4000 (other 100 samples for testing) | 75% |
| 5000 (other 100 samples for testing) | 73% |
| 6000 (other 100 samples for testing) | 75% |

**Explanation:**

There are some reasons may lead to this:

1- Randomness: Because in our implement in the first time that we choose randomly, this also leads to given the same number of training samples, sometimes we get different result both for prediction accuracy and the running time

2- Parameter Optimization: For example, in SVM, while increasing the training size, if the data is not linearly separable, we should increase the values of the slack variables C. Adjusting this parameter is able to help to consider any new training sample violates the linear separating hyperplane of the space.

3- Overfitting: Training some classifiers for longer time or using extra training samples, may lead to a good performance on the training data but a worse one on the testing part. This is because the classifiers could be so much fitting the training samples to an extent, and this will lead to difficult to predict the new come samples.

For training our data by using SVM, we need to transfer our data to make SVM can use them. Here is an example how we did this:

|  |  |
| --- | --- |
| Work-class Name: | After transfer: |
| Private | 0 |
| Federal-gov | 1 |
| Local-gov | 2 |
| Self-emp-not-inc | 3 |
| Self-emp-inc | 4 |
| State-gov | 5 |
| Without-pay | 6 |

The implement of SMO for SVM in this experiment is not that time efficient compared to other two classification algorithm and as the number of samples increase, the time of getting the hyperplane will increase dramatically.

|  |  |
| --- | --- |
| Number of sample: | Train time: |
| 4000 | 7.023096s |
| 8000 | 18.062998s |
| 12000 | 171.108550s |
| 16000 | 275.560671s |
| 20000 | 441.680359s |

**Explanation:**

SVM often do take a long time to train, there are several reasons that make SVM getting the result slowly. The first reason could be the linear-function and variable that we choose to train data, the second reason could be that the data were not transferred in a right way or it should be normalized.

**Experiment**

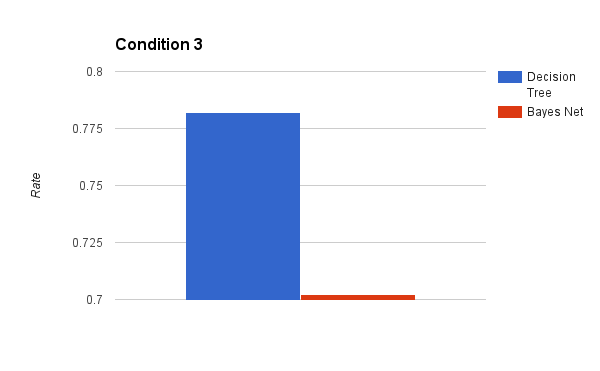
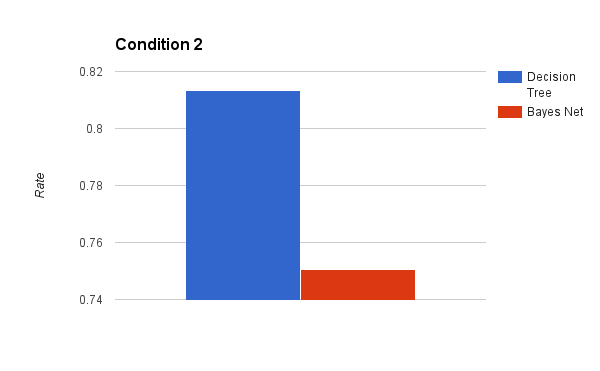
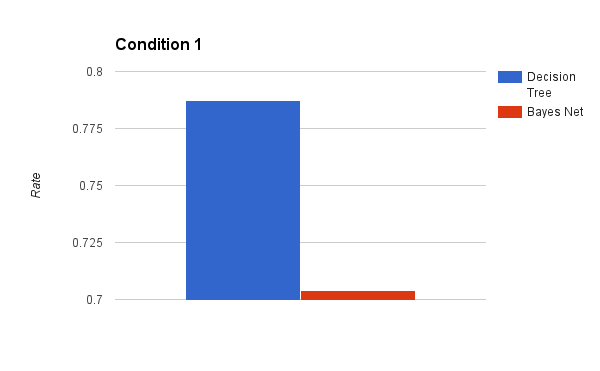
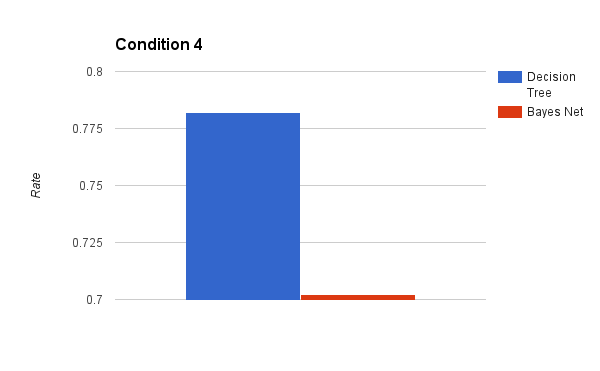
In this section we will compare these three algorithm by calculating the prediction correctness rate. We change the attributes and the training set number in multiple ways to figure out how each algorithm works in different situations.

First of all, we will test each algorithm with different number of rows of training data, starting from 4000 growing by 4000 rows every time till 20000. And the number of test cases will be 2000.

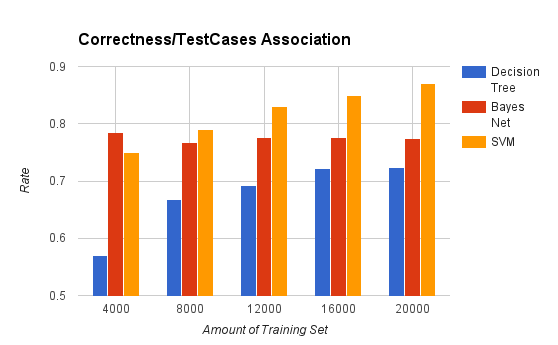
As we can see in the association analysis, the income is strongly affected by relationship, occupation, education and workclass. Therefore, we will compare these attributes with strong association to those with weak association.

All the experiments we did are listed in sheet-2:

|  |  |  |
| --- | --- | --- |
| Label | Attributes | Training Set Number |
| 1 | agegroup, native country, race, education | 20000 |
| 2 | relationship, occupation, education, workclass | 20000 |
| 3 | hoursgroup, native country, race, education | 20000 |
| 4 | hoursgroup, native country, race, education-num | 20000 |
| 5 | All | 4000 |
| 6 | All | 8000 |
| 7 | All | 12000 |
| 8 | All | 16000 |
| 9 | All | 20000 |

As We

As we can see in the four charts above, when we use hoursgroup, native country, race and education as our attributes,



As we can see from the correctness result of three classification algorithm, as the increasing number of training samples, decision tree’s and SVM’s performance become better. However, the Bayes Net meets the issue that increasing the sample size does not help the classification performance.

**Association Analysis**

In this project, we also do association analysis to find the frequent patterns of given census data.

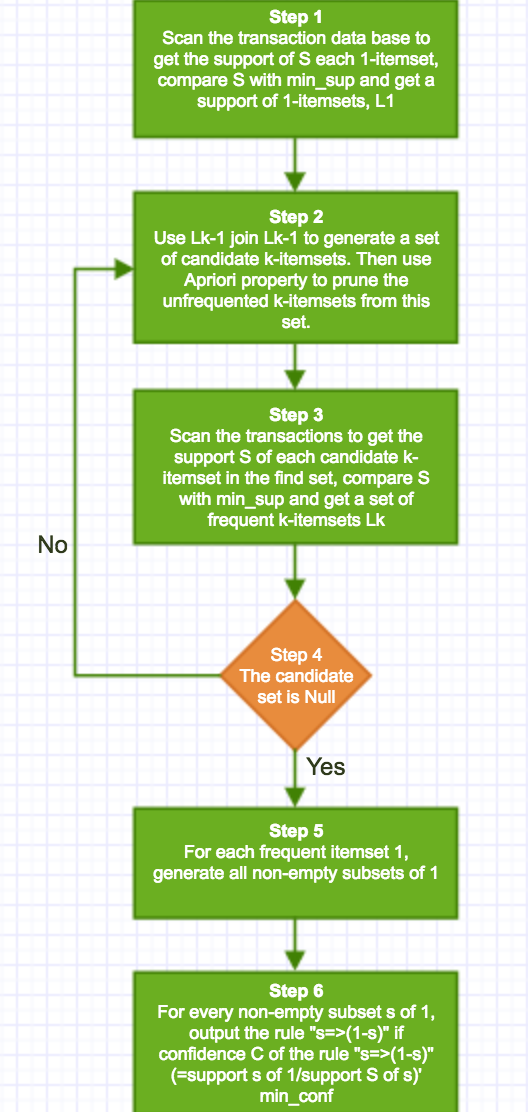
**1. Choosing Algorithm:**

When it comes to association analysis, people think about Apriori algorithm and FP-Growth algorithm first. Therefore, I’d like to introduce these two methods and then explain why we choose FP-Growth.

**Apriori algorithm:**

As mentioned above, Apriori is designed for mining frequent itemsets and association rules. This algorithm makes many searches in database to find frequent itemsets where k items are used to generate k+1-itemsets. Each k-itemset must be greater than or equal to minimum support threshold to be frequency. Otherwise, it’s called candidate itemsets.

I drew a flow chart of Apriori’s processing steps .



Clearly, this algorithm is easy to execute and very simple. However, it has a great drawback. It’s costly wasting of time to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets. In other words, Apriori will be very slow and inefficiency when memory capacity is limited with large number of transactions.

In our training database, we have approximate 30,000 transactions. Therefore, this algorithm is not a good choice in our situation

**FP-Growth algorithm:**

Different with Apriori algorithm, FP-Growth allows frequent itemset discovery without candidate itemset generation. It encodes the data set using a compact data structure called FP-Tree and extracts frequent itemsets directly from this tree.

There are two steps to go when processing FP-Growth algorithm:

1. FP-Tree construction.

The relative code of this step is in FPTreeBuilder.py file

* Scan data and find support for each item.
* Discard infrequent items.
* Sort frequent items in decreasing order based on their support.
* FP-Growth reads 1 transaction at a time and maps it to a path
* Fixed order is used, so path can overlap when transactions share items
* Pointers are maintained between nodes containing the same item, creating singly linked lists

1. Frequent itemset generation

The relative code of this step is in FPTreeMiner.py file

* Frequent itemsets extracted from the FP-Tree
* Bottom-up algorithm: from the leaves towards the root
* Using the linked lists, extract prefix path sub-trees ending in an itemset
* Each prefix path sub-tree is processed recursively to extract the frequent itemsets. Solutions are then merged.

**Compare Apriori and FP-Growth:**

|  |  |  |
| --- | --- | --- |
|  | Apriori | FP-Growth |
| Technique | Use Apriori property & join and prune property | Constructs conditional frequent pattern tree and conditional pattern base from transactions which meet minimum support |
| No. of scans | Multiple times | Twice |
| Memory | Due to large number of candidate are generated, large memory is required | Due to compact structure and no candidate generation, small memory is required |
| Time | Slow  (---Apriori using 8.60690307617 seconds --- in 3td library implementation over 30,000 transactions) | Quicker than Apriori  (---FP-Growth using 6.34852409363 seconds --- in my implementation over nearly 30,000 transactions) |

\* The code of calling of Apriori algorithm from 3rd library is just for running time comparison between Apriori and FP-Growth. It is the best evidence of why we choose FP-Growth algorithm rather than Apriori.

**2. Patterns found**

After running FP-Growth algorithm on cleaned transactions file(AdultCensus\_cleaned.csv), We got several frequent patterns.

Different minimum support makes different pattern.

|  |  |
| --- | --- |
| min\_sup | frequent patterns |
| 150 | ('nativecountry:United-States', 'income:>50K', 'hoursgroup:30 - 39') : 154  ('income:<=50K', 'occupation:Craft-repair', 'hoursgroup:30 - 39') : 150  ('sex:Male', 'income:>50K', 'occupation:Craft-repair') : 150  ('nativecountry:United-States', 'income:>50K', 'relationship:Unmarried') : 164  ('income:<=50K', 'hoursgroup:30 - 39', 'relationship:Unmarried') : 160  ('income:<=50K', 'agegroup:20 - 29', 'relationship:Unmarried') : 153  ('income:<=50K', 'relationship:Own-child', 'workclass:Local-gov') : 150  ('income:<=50K', 'agegroup:20 - 29', 'workclass:Local-gov') : 154  ('income:<=50K', 'agegroup:20 - 29', 'occupation:Sales') : 161  ('income:<=50K', 'race:Black', 'hoursgroup:20 - 29') : 150  ('nativecountry:United-States', 'income:>50K', 'race:Black') : 163  ('income:<=50K', 'agegroup:30 - 39', 'occupation:Other-service') : 156  ('income:<=50K', 'agegroup:50 - 59', 'occupation:Other-service') : 185  ('income:<=50K', 'agegroup:20 - 29', 'occupation:Other-service') : 161 |
| 200 | ('income:<=50K', 'relationship:Own-child', 'occupation:Adm-clerical') : 220  ('income:<=50K', 'agegroup:20 - 29', 'occupation:Adm-clerical') : 203  ('income:<=50K', 'occupation:Sales', 'relationship:Unmarried') : 200  ('income:<=50K', 'agegroup:50 - 59', 'relationship:Unmarried') : 204  ('income:<=50K', 'education:Some-college', 'occupation:Other-service') : 225  ('income:<=50K', 'relationship:Own-child', 'occupation:Other-service') : 236  ('income:<=50K', 'agegroup:40 - 49', 'occupation:Other-service') : 219  ('income:<=50K', 'education:Some-college', 'educationnum:10', 'occupation:Other-service') : 225  ('income:<=50K', 'educationnum:10', 'occupation:Other-service') : 225 |
| 250 | None |

**Explanation:**

The result is in the the form of A:B, which A stands by all the items in the frequent pattern, B means the frequency over all given transactions.

For every item in frequent pattern, it’s in the form of C:D again, which C means the attribute name, D refers the value of C.

**Analysis and Evaluation:**

I showed three different minimum supports in above table:

For min\_sup = 150, there are 14 frequent patterns whose frequency is between 150 to 200. Top 5 patterns in this situation are:

('income:<=50K', 'agegroup:50 - 59', 'occupation:Other-service') : 185

('nativecountry:United-States', 'income:>50K', 'relationship:Unmarried') : 164

('nativecountry:United-States', 'income:>50K', 'race:Black') : 163

('income:<=50K', 'agegroup:20 - 29', 'occupation:Other-service') : 161

('income:<=50K', 'agegroup:20 - 29', 'occupation:Sales') : 161

Since our classify target is income, we only need to learn other attributes in top 5 patterns. Therefore, as you can tell, the occupation attribute shows up 2 times, the age attribute shows up 3 times, the native country attribute shows up 2 twice and race shows once.

For min\_sup = 200, there are 10 frequent patterns whose frequency is between 200 to 250. The top 5 frequent patterns are

('income:<=50K', 'relationship:Own-child', 'occupation:Other-service') : 236

('income:<=50K', 'education:Some-college', 'occupation:Other-service') : 225

('income:<=50K', 'education:Some-college', 'educationnum:10', 'occupation:Other-service') : 225

('income:<=50K', 'educationnum:10', 'occupation:Other-service') : 225

('income:<=50K', 'relationship:Own-child', 'occupation:Adm-clerical') : 220

Since our classify target is income, we only need to learn other attributes in top 5 patterns. Therefore, as you can tell, the occupation attribute shows up 5 times, the education or educationNum attribute shows up 4 times, the relationship attribute shows up 2 twice.

For min\_sup = 250, there is no pattern meets the minimum support requirement.

The analysis of the frequency of each attribute above confirms the findings with the classifiers: age, occupation, education and relationship are good predictors of income labels “<= 50K” and “>50K”. This conclusion also matches our common sense.

**References**

Engelmore, R., and Morgan, A. eds. 1986. *Blackboard Systems.* Reading, Mass.: Addison-Wesley.

Sequential Minimal Optimization: A Fast-Algorithm for Training Support Vector Machines

*(If you use EndNote and it complains "object has been deleted" when you try to generate references, temporarily delete the heading with the copyright notice attached as a footnote, and reinsert it after EndNote finishes.)*