COMP 309 — Machine Learning Tools and Techniques
Assignment 3: Kaggle Competition Report

Objectives

The goal of this competition is to develop the best possible machine learning system to predict the house-hold income of families of newborn children.

Exploring and understanding the census data

Variable type

The census data only have three Variable type, most are binary type (eg ethnicity, income source), some of them are nominal type (eg Deprivation, qualification), others are numerical type (eg age, working hours). Is interesting that the class label house-hold income is label as nominal type, which means it aiming to predict the range of house-hold income instead of the exact amount. This idea can be imitate, to change the numerical type to nominal type by divide the numerical data into several different ranges. In census data, can use this idea to change the numerical type.

Uncertainty

The first interesting pattern fined in census datas is the description of child depend family type code. In this description, it refers a word: ‘Adult’ but it do not have further description at all. Does he/she is the brother/sister of the newborn child? What is the relationship between this adult and his/her family? The most important thing is does his/her income counts in the house-hold income? If it does, the details of this ‘Adult’ is very important. If it doesn’t, does this ‘Adult’ count as an inversely proportional attributes to income? (this adult use up the family income)

Completeness

In the census data, there are many missing valuables. The reason for this might be the provider do not willing to provide their informations (eg ladies do not willing to provide their ages) or the provider do not know those informations. One of the common phenomenon is occur on single parent families, in those single parent families it do not have the details on his/her partner, so she/he cannot provide any information, it leads to plenty of missing values on mother informations and father informations. In the statistical, there are around 4% missing values on mother informations, and around 19% missing values on father informations. In addition to divorce cause of missing data, there are some other issues leads to have around 3% missing values on other attributes. But in terms of child informations, their have good integrity.

Proportional/Inverse proportion

In the census data, there are some attributes are proportional to house-hold income, some of them are inversely proportional, others are do not have direct relationship to house-hold income. For instance, work hours is an attributes that proportional to house-hold income. If they have the same hourly pay, the more time you work, you will get more income. Moreover, qualification also is an attributes that proportional to house-hold income, the higher qualification will have higher hourly pay. On the inverse proportion side, smoke is one of the attributes to represent. Smoke will cost a lot of money, if cost $20 smoke per day, one year will cost $730. Again for example, the attributes 'single parent' is inverse proportion to house-hold. If the house-hold only depends on one parent, this obviously cannot compare with two parent family. The above discussion illustrates the relationship between some attributes and house-hold income. In fact, some attributes are haven't/undefined direct relationship to house-hold income. (eg ethnicity)

Class imbalance

In the census data, there are have 15 class labels. But the class is imbalance, the class income in range ($50,001 - $70,000, $70,001 - $100,000 and $70,001 - $100,000) those three class are over 66% of whole labeled dataset. In this situation, if run through the machine learning problem it might easily cause overfilling problem. From those 3000 labeled instances, the machine learning technique will know well for those 3 class label's characteristics, but there do not have enough instance for identify the rest of the class's characteristics, that means is hard to currently classify for the rest of the classes.

Outliers/Extreme Value

There are some outliers/Extreme values in some numerical type attributes, for example in the attribute mother age, have a 96 year old mother. That is impossible. If the maternal age is over 50, that is belongs to advanced maternal age. Another Extreme is fined from dad_work_hour attribute, is impossible for someone work 160 hour per week. Also those kind of situation is occur in m_work_hour attribute and other numerical type attributes.Those outliers and Extreme Values will affect on machine learning technique during training. Once the machine learning technique recognise those characteristics, it might lead to make a wrong classification. In other words, it might course overfitting problem.

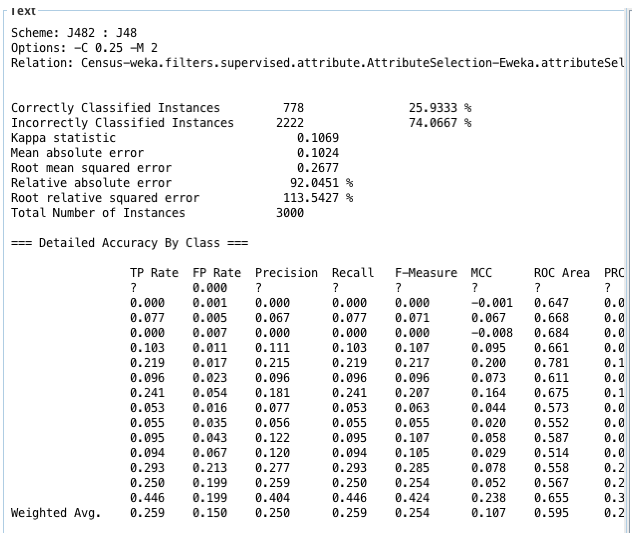
Developing and testing machine learning system

Data split

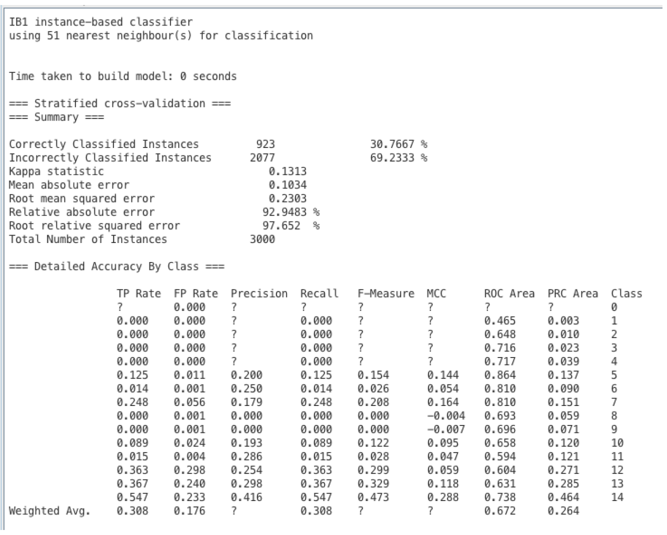
The census data have 7621 instance in total, the first 3000 instance are have class label, and the rest of them are not. Therefore, only the first 3000 instances can be use as training and testing, and base on those instances to find out the best initial system to predict the total . The rest of the instances are use for observe and predict. During building the system, I design to use 10 fold cross-validation set to test the performance of my system. Because that will given me a quite precise accurate rate.

Machine Learning technique selection

First of all, I did make attempt on depiction tree technique and K-nearest neighbour technique. Without doing any data reprocessing, the accuracy of those two techniques are shows below:



```
Text
Scheme: J482 : J48
Options: -C 0.25 -M 2
Relation: Census-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSel


Correctly Classified Instances         778               25.9333 %
Incorrectly Classified Instances      2222               74.0667 %
Kappa statistic                          0.1069
Mean absolute error                      0.1024
Root mean squared error                  0.2677
Relative absolute error                 92.0451 %
Root relative squared error            113.5427 %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC
                 ?        0.000    ?          ?        ?          ?        ?         ?
                 0.000    0.001    0.000      0.000    0.000      -0.001   0.647     0.0
                 0.077    0.005    0.067      0.077    0.071      0.067    0.668     0.0
                 0.000    0.007    0.000      0.000    0.000      -0.008   0.684     0.0
                 0.103    0.011    0.111      0.103    0.107      0.095    0.661     0.0
                 0.219    0.017    0.215      0.219    0.217      0.200    0.781     0.1
                 0.096    0.023    0.096      0.096    0.096      0.073    0.611     0.0
                 0.241    0.054    0.181      0.241    0.207      0.164    0.675     0.1
                 0.053    0.016    0.077      0.053    0.063      0.044    0.573     0.0
                 0.055    0.035    0.056      0.055    0.055      0.020    0.552     0.0
                 0.095    0.043    0.122      0.095    0.107      0.058    0.587     0.0
                 0.094    0.067    0.120      0.094    0.105      0.029    0.514     0.0
                 0.293    0.213    0.277      0.293    0.285      0.078    0.558     0.2
                 0.250    0.199    0.259      0.250    0.254      0.052    0.567     0.2
                 0.446    0.199    0.404      0.446    0.424      0.238    0.655     0.3
Weighted Avg.    0.259    0.150    0.250      0.259    0.254      0.107    0.595     0.2
```



```
IB1 instance-based classifier
using 51 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         923               30.7667 %
Incorrectly Classified Instances      2077               69.2333 %
Kappa statistic                          0.1313
Mean absolute error                      0.1034
Root mean squared error                  0.2303
Relative absolute error                 92.9483 %
Root relative squared error             97.652  %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        0.000    ?          ?        ?          ?        ?         ?         0
                 0.000    0.000    ?          0.000    ?          ?        0.465     0.003     1
                 0.000    0.000    ?          0.000    ?          ?        0.648     0.010     2
                 0.000    0.000    ?          0.000    ?          ?        0.716     0.023     3
                 0.000    0.000    ?          0.000    ?          ?        0.717     0.039     4
                 0.125    0.011    0.200      0.125    0.154      0.144    0.864     0.137     5
                 0.014    0.001    0.250      0.014    0.026      0.054    0.810     0.090     6
                 0.248    0.056    0.179      0.248    0.208      0.164    0.810     0.151     7
                 0.000    0.001    0.000      0.000    0.000      -0.004   0.693     0.059     8
                 0.000    0.001    0.000      0.000    0.000      -0.007   0.696     0.071     9
                 0.089    0.024    0.193      0.089    0.122      0.095    0.658     0.120     10
                 0.015    0.004    0.286      0.015    0.028      0.047    0.594     0.121     11
                 0.363    0.298    0.254      0.363    0.299      0.059    0.604     0.271     12
                 0.367    0.240    0.298      0.367    0.329      0.118    0.631     0.285     13
                 0.547    0.233    0.416      0.547    0.473      0.288    0.738     0.464     14
Weighted Avg.    0.308    0.176    ?          0.308    ?          ?        0.672     0.264
```

The graph shows the baseline of those two algorithms, compare to each other, IBK have slightly better performance then J48. If look in details, in TP rate, it shows IBK algorithms are doing better classification for the last 3 classes, but J48 does doing better to classify the rest of the classes. Overall, this census dataset provide a lot of instances for the last 3 classes, IBK does the better job for recognise those 3 classes's characteristics, and J48 recognise those characteristics quite evenly. Either those two technique are suitable for this dataset, they have their own advantages.

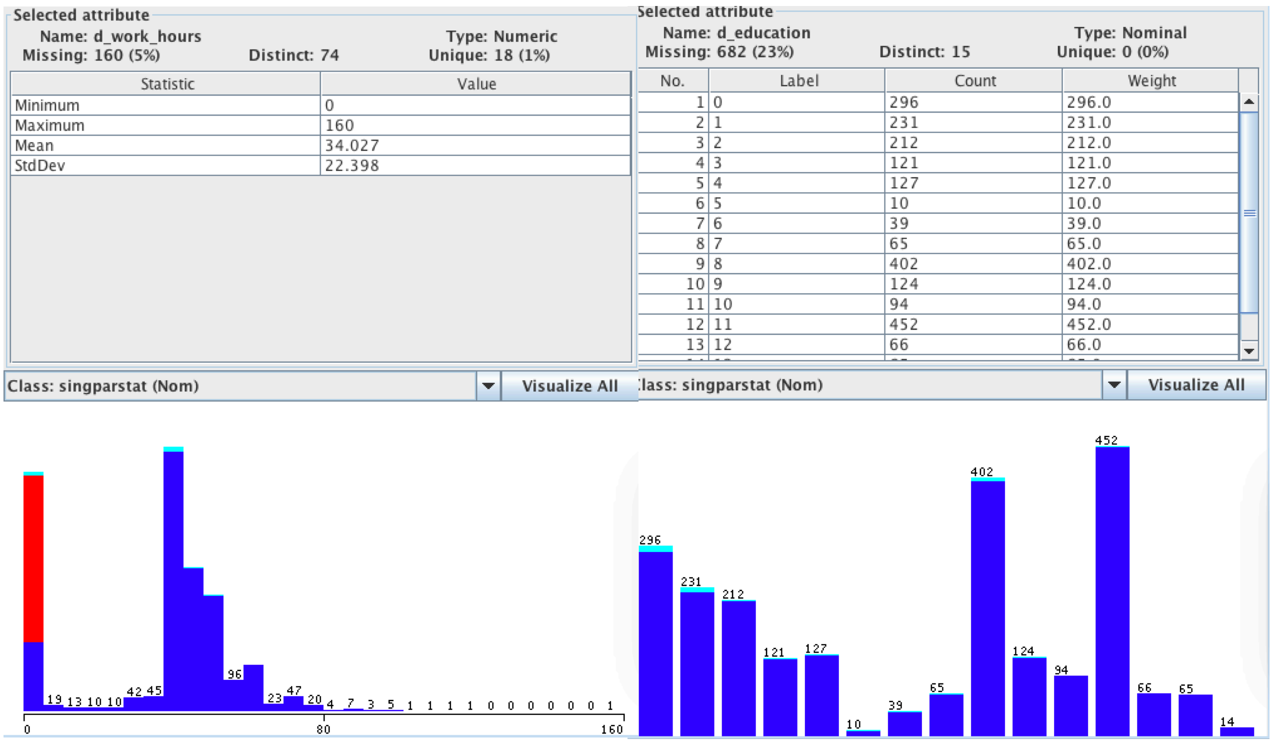
Initial Ideas

There are two ways to improve the accuracy, one is to doing data reprocessing to deal with missing values, outliers and change the variable type that is suitable for the machine learning techniques. Which helps the machine learning techniques easier to learn/recognise the characteristics. But consider in the first 3000 census data instances, there are have imbalances classes and the rest of the classes are unknown. If the rest of the classes are have quite balance, the algorithm that used for training will does not performing well. It might overfitting for recognise last 3 classes's characteristics. The second way is to deal with imbalance classes, to make the class balance by add some 'fake' class instances, or subtract some instance from last 3 classes (doing resample).

Initial System Design

missing values processing

In the first Initial system design, will try to deal with missing values, outliers and variable types to see how well it perform on the leaderboard. Firstly, replacing missing values by using k-nearest algorithm. This is because we have to according to it's actual situation to replacing values, to predict the value that is closest to the actual situation. Rather then just use mean and modes to replacing. For example, there are about 5% missing values on dad_work_hour attributes. The bottom right figure shows in the single_mother_family, dad work for 0 hours (the red bar is the single_mother_family class). If we have a missing value on dad_work_hour attributes, but it is a single_mother_family, the missing value should be replace 0 instead of the mean 34.027.



Selected attribute
Name: d_work_hours — Type: Numeric
Missing: 160 (5%) — Distinct: 74 — Unique: 18 (1%)

Statistic	Value
Minimum	0
Maximum	160
Mean	34.027
StdDev	22.398

Class: singparstat (Nom)

Selected attribute
Name: d_education — Type: Nominal
Missing: 682 (23%) — Distinct: 15 — Unique: 0 (0%)

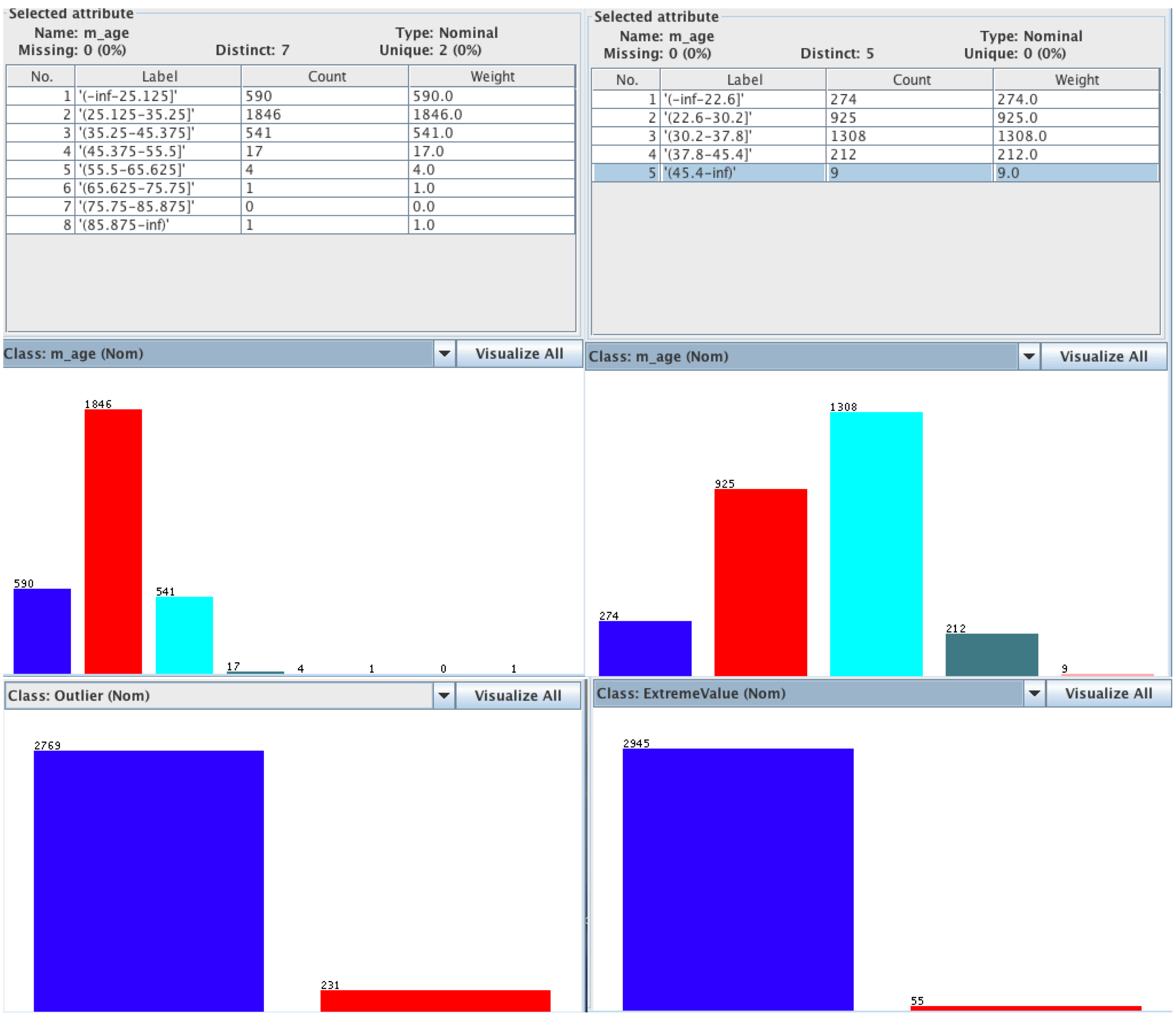
No.	Label	Count	Weight
1	0	296	296.0
2	1	231	231.0
3	2	212	212.0
4	3	121	121.0
5	4	127	127.0
6	5	10	10.0
7	6	39	39.0
8	7	65	65.0
9	8	402	402.0
10	9	124	124.0
11	10	94	94.0
12	11	452	452.0
13	12	66	66.0

Class: singparstat (Nom)

The top left figure shows in single_mother_family, it do not provide any dad education informations. If use modes to replace those missing values all the values will replace as 11 (Bachelor Degree and Level 7 Qualification), that is not real. It should consider other attributes to doing replacing, and KNN is one of the good options.

Outlier/extreme values processing

After dealing with missing values, I use a interquartile range filter to check out the Outlier/extreme values, the below red bar below shows the outlier and extreme values that I find out:



Selected attribute
Name: m_age
Missing: 0 (0%) Distinct: 7
Type: Nominal
Unique: 2 (0%)

No.	Label	Count	Weight
1	'(-inf-25.125]'	590	590.0
2	'(25.125-35.25]'	1846	1846.0
3	'(35.25-45.375]'	541	541.0
4	'(45.375-55.5]'	17	17.0
5	'(55.5-65.625]'	4	4.0
6	'(65.625-75.75]'	1	1.0
7	'(75.75-85.875]'	0	0.0
8	'(85.875-inf)'	1	1.0

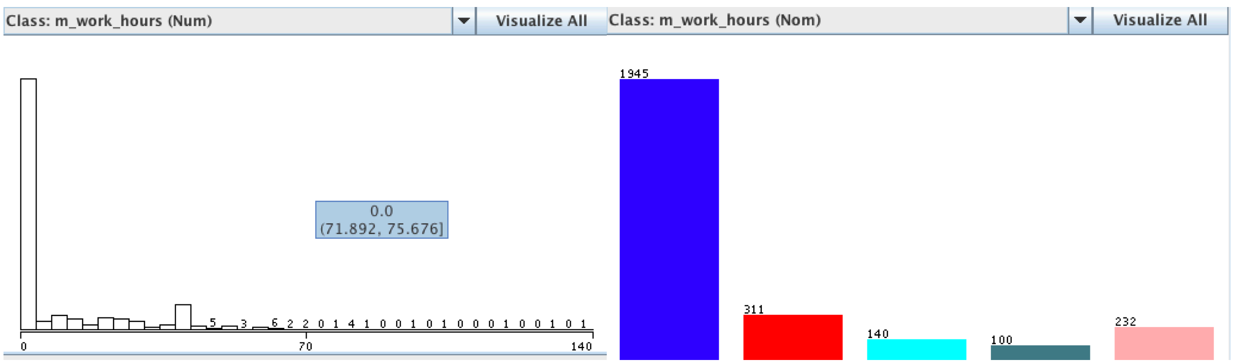
Selected attribute
Name: m_age
Missing: 0 (0%) Distinct: 5
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-22.6]'	274	274.0
2	'(22.6-30.2]'	925	925.0
3	'(30.2-37.8]'	1308	1308.0
4	'(37.8-45.4]'	212	212.0
5	'(45.4-inf)'	9	9.0

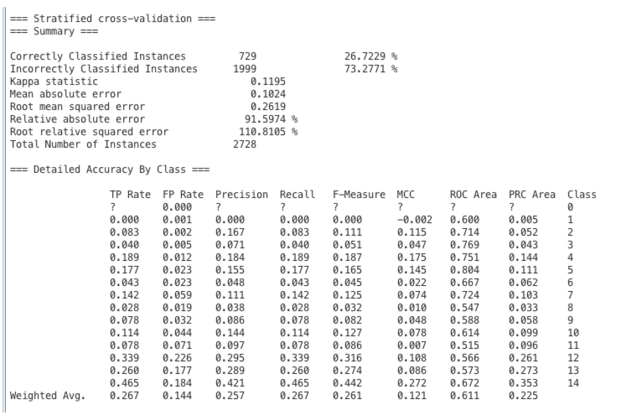
Those outlier values will affect on discretise the numeric variables, for example when I discretise the m_age attributes it will provide some useless label, in this situation it might better to set age above 55.5-inf into one label. After remove the outliers, discretise the m_age attributes shows as the top right figure.

Variable type processing

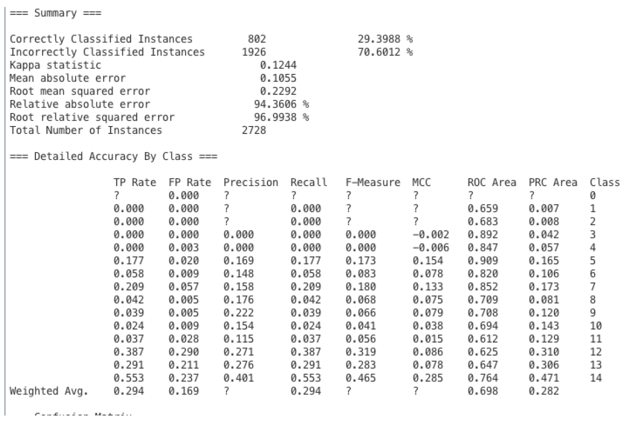
I design to use discretise filter to convert numeric type to nominal type, rather than just simply use numeric to nominal filter. This is because it can make less branches, the more branches the tree have the harder to calculate the purely of the nodes and the tree might treat this as an insignificance attributes, set it at very bottom of the tree. Also The chance for make wrong deciton (error rate) will also increases. I think 5 might be the right amount of bins for discretise the numeric attributes. The below two figure shows the m_wirk_hours attribute before and after using the discretise filter:



Initial System result



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         729               26.7229 %
Incorrectly Classified Instances      1999               73.2771 %
Kappa statistic                          0.1195
Mean absolute error                      0.1024
Root mean squared error                  0.2619
Relative absolute error                 91.5974 %
Root relative squared error            110.8105 %
Total Number of Instances             2728

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        0.000    ?          ?        ?          ?        ?         ?         0
                 0.000    0.001    0.000      0.000    0.000      -0.002   0.600     0.005     1
                 0.083    0.002    0.167      0.083    0.111      0.115    0.714     0.052     2
                 0.040    0.005    0.071      0.040    0.051      0.047    0.769     0.043     3
                 0.189    0.012    0.184      0.189    0.187      0.175    0.751     0.144     4
                 0.177    0.023    0.155      0.177    0.165      0.145    0.804     0.111     5
                 0.043    0.023    0.048      0.043    0.045      0.022    0.667     0.062     6
                 0.142    0.059    0.111      0.142    0.125      0.074    0.724     0.103     7
                 0.028    0.019    0.038      0.028    0.032      0.010    0.547     0.033     8
                 0.078    0.032    0.086      0.078    0.082      0.048    0.588     0.058     9
                 0.114    0.044    0.144      0.114    0.127      0.078    0.614     0.099     10
                 0.078    0.071    0.097      0.078    0.086      0.007    0.515     0.096     11
                 0.339    0.226    0.295      0.339    0.316      0.108    0.566     0.261     12
                 0.260    0.177    0.289      0.260    0.274      0.086    0.573     0.273     13
                 0.465    0.184    0.421      0.465    0.442      0.272    0.672     0.353     14
Weighted Avg.    0.267    0.144    0.257      0.267    0.261      0.121    0.611     0.225
```

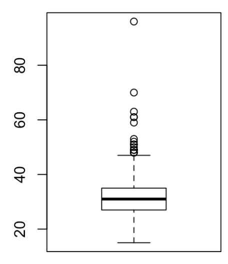


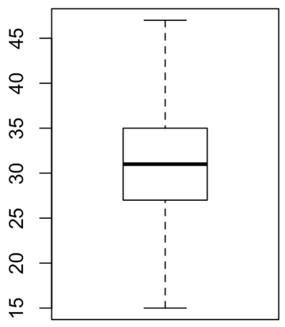
```
=== Summary ===

Correctly Classified Instances         802               29.3988 %
Incorrectly Classified Instances      1926               70.6012 %
Kappa statistic                          0.1244
Mean absolute error                      0.1055
Root mean squared error                  0.2292
Relative absolute error                 94.3606 %
Root relative squared error             96.9938 %
Total Number of Instances             2728

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        0.000    ?          ?        ?          ?        ?         ?         0
                 0.000    0.000    ?          0.000    ?          ?        0.659     0.007     1
                 0.000    0.000    ?          0.000    ?          ?        0.683     0.008     2
                 0.000    0.000    0.000      0.000    0.000      -0.002   0.892     0.042     3
                 0.000    0.003    0.000      0.000    0.000      -0.006   0.847     0.057     4
                 0.177    0.020    0.169      0.177    0.173      0.154    0.909     0.165     5
                 0.058    0.009    0.148      0.058    0.083      0.078    0.820     0.106     6
                 0.209    0.057    0.158      0.209    0.180      0.133    0.852     0.173     7
                 0.042    0.005    0.176      0.042    0.068      0.075    0.709     0.081     8
                 0.039    0.005    0.222      0.039    0.066      0.079    0.708     0.120     9
                 0.024    0.009    0.154      0.024    0.041      0.038    0.694     0.143     10
                 0.037    0.028    0.115      0.037    0.056      0.015    0.612     0.129     11
                 0.387    0.290    0.271      0.387    0.319      0.086    0.625     0.310     12
                 0.291    0.211    0.276      0.291    0.283      0.078    0.647     0.306     13
                 0.553    0.237    0.401      0.553    0.465      0.285    0.764     0.471     14
Weighted Avg.    0.294    0.169    ?          0.294    ?          ?        0.698     0.282
```

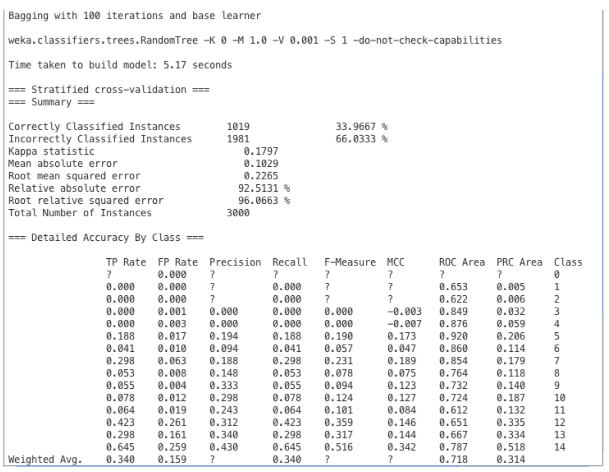
Those two accuracy are the performance of J48 and IBK.(10 fold cross validation is used). Compare to the base result, J48 actually increases it's accuracy, but IBK decreases it's accuracy. The reason might be to convert the numeric type to nominal type, the IBK is better to identify the characteristics from numeric variables. It seems like my data processing system is more suitable for using diction tree techniques, the following system design will more focus on doing data reprocessing for diction tree techniques. By the way I did make attempt on the leaderboard, use IBK algorithms, the score is 0.287. Which very close to the 10 fold cross validation test score and the J48 results also looks slimier.

Intermediary system

After summaries the experience above, I realise if I remove the outliers the whole instances also has been remove. I cannot judge the outliers value are more important then others, because of one outliers value to sacrifice the whole instances will make better progress. Instead to remove them to replace all the outliers by the bench mark value(Q3 + 1.5*IQR / Q1-1.5*IQR), that will make the instances useable. Further more, to using diction tree as the machine learning technique for the system, is better to provide the data as binary type. The more branches of the tree have, the tree will be harder to calculate the purely of the nodes, therefore binary type variables is the best for the diction tree doing classification. Again consider using diction tree techniques, using random forest diction tree technique is a better option. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. That means there is more chance for predict the right class label.





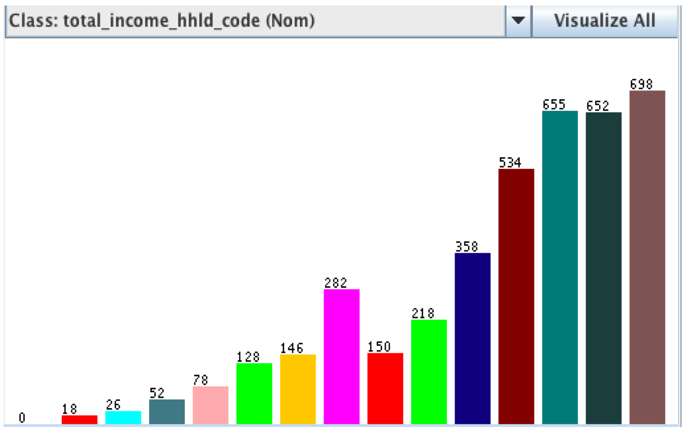


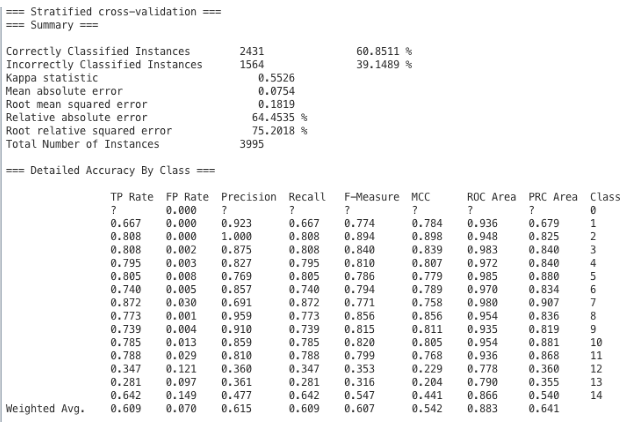
```
Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 5.17 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1019               33.9667 %
Incorrectly Classified Instances      1981               66.0333 %
Kappa statistic                          0.1797
Mean absolute error                      0.1029
Root mean squared error                  0.2265
Relative absolute error                 92.5131 %
Root relative squared error             96.0663 %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        0.000    ?          ?        ?          ?        ?         ?         0
                 0.000    0.000    ?          0.000    ?          ?        0.653     0.005     1
                 0.000    0.000    ?          0.000    ?          ?        0.622     0.006     2
                 0.000    0.001    0.000      0.000    0.000      -0.003   0.849     0.032     3
                 0.000    0.003    0.000      0.000    0.000      -0.007   0.876     0.059     4
                 0.188    0.017    0.194      0.188    0.190      0.173    0.920     0.206     5
                 0.041    0.010    0.094      0.041    0.057      0.047    0.860     0.114     6
                 0.298    0.063    0.188      0.298    0.231      0.189    0.854     0.179     7
                 0.053    0.008    0.148      0.053    0.078      0.075    0.764     0.118     8
                 0.055    0.004    0.333      0.055    0.094      0.123    0.732     0.140     9
                 0.078    0.012    0.298      0.078    0.124      0.127    0.724     0.187     10
                 0.064    0.019    0.243      0.064    0.101      0.084    0.612     0.132     11
                 0.423    0.261    0.312      0.423    0.359      0.146    0.651     0.335     12
                 0.298    0.161    0.340      0.298    0.317      0.144    0.667     0.334     13
                 0.645    0.259    0.430      0.645    0.516      0.342    0.787     0.518     14
Weighted Avg.    0.340    0.159    ?          0.340    ?          ?        0.718     0.314
```

The right figure shows before and after dealing with outliers, left figure shows the accuracy of intermediary system. The score of this system on leaderboard is 0.33350, which make some progress from last system.

Resample System

The TP rate on the above system also shows the high prediction for the last 3 classes. To improve the total accuracy, it also need to improve prediction for the rest of the classes. In theory, the best way to improve, is to provide more instances for the machine learning technique. But we cannot get any more instance from the real world. So, I design to make some 'fake' instances for using on the training dataset, the instances are randomly generate in a range between maximum and minimum values, to fill the shortage of the less instances classes. After run through the system, it did improved the accuracy for training, and from the tp rate, it shows the classification on class 0-11 are actually increase. But the score of this system on leaderboard is on the contrary, it decreases about 1%. The problem might be the 'fake' instances I created are completely useless, it do not conform to the actual situation. Perhaps, it might be the overfitting problem, the class distribution on the test set might be slimier to the training set. To increase the identify ability for class 0-11 in training set, will decrease the predicted accuracy of last 3 class.



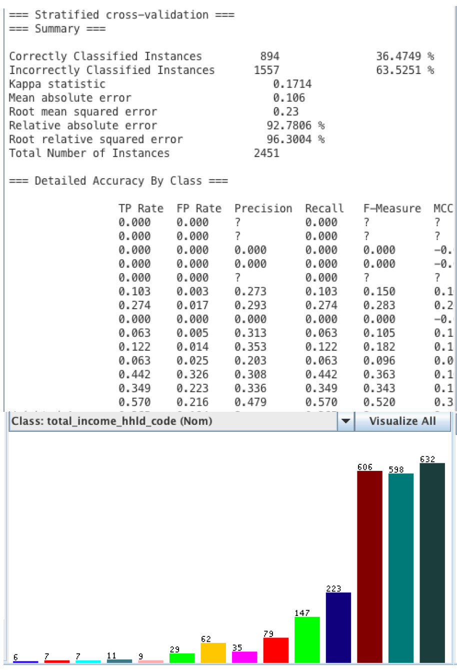


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2431               60.8511 %
Incorrectly Classified Instances      1564               39.1489 %
Kappa statistic                          0.5526
Mean absolute error                      0.0754
Root mean squared error                  0.1819
Relative absolute error                 64.4535 %
Root relative squared error             75.2018 %
Total Number of Instances             3995

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        0.000    ?          ?        ?          ?        ?         ?         0
                 0.667    0.000    0.923      0.667    0.774      0.784    0.936     0.679     1
                 0.808    0.000    1.000      0.808    0.894      0.898    0.948     0.825     2
                 0.808    0.002    0.875      0.808    0.840      0.839    0.983     0.840     3
                 0.795    0.003    0.827      0.795    0.810      0.807    0.972     0.840     4
                 0.805    0.008    0.769      0.805    0.786      0.779    0.985     0.880     5
                 0.740    0.005    0.857      0.740    0.794      0.789    0.970     0.834     6
                 0.872    0.030    0.691      0.872    0.771      0.758    0.980     0.907     7
                 0.773    0.001    0.959      0.773    0.856      0.856    0.954     0.836     8
                 0.739    0.004    0.910      0.739    0.815      0.811    0.935     0.819     9
                 0.785    0.013    0.859      0.785    0.820      0.805    0.954     0.881     10
                 0.788    0.029    0.810      0.788    0.799      0.768    0.936     0.868     11
                 0.347    0.121    0.360      0.347    0.353      0.229    0.778     0.360     12
                 0.281    0.097    0.361      0.281    0.316      0.204    0.790     0.355     13
                 0.642    0.149    0.477      0.642    0.547      0.441    0.866     0.540     14
Weighted Avg.    0.609    0.070    0.615      0.609    0.607      0.542    0.883     0.641
```

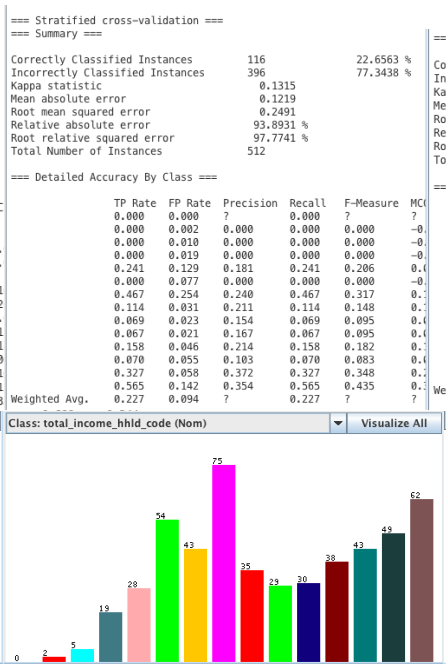
Combine System

To develop a better system, I design to ignore the resample system and I did look back on the attributes try to find out some useful correlation for the new system. Because I cannot judge whether the resample system make any performances or not. One of the correlation I find out is what I discussed before: single parent family always have huge amount of missing variables on child's mother/father. Eg (in single_mother family, the whole column attributes d_education, d_smoke, d_Ethnicity ect are all missing values) If use kNN to doing interpolation, it will try to replace with the closest attributes from single_mother class or two_parent_classes. Therefore use kNN to replacing those variables are not even close to the real situations. The better way to deal with those missing value is to divide the data into single_mother dataset, single_father dataset, and two_parent dataset. In those different dataset can use different technique to dealing with the missing values. For the single_parent families dataset, Is better to delete the whole missing attributes, and for the two_parent dataset can be replaces missing values by using kNN. And I did use the same process from Initial System dealing with outliers and variable types. Another reason for train and test each dataset separately, is try to make the system better to recognise the class characteristics. If we look for the class description of each dataset, they are all different. The following figures order are two_parent dataset, single_mother dataset and single_father dataset.

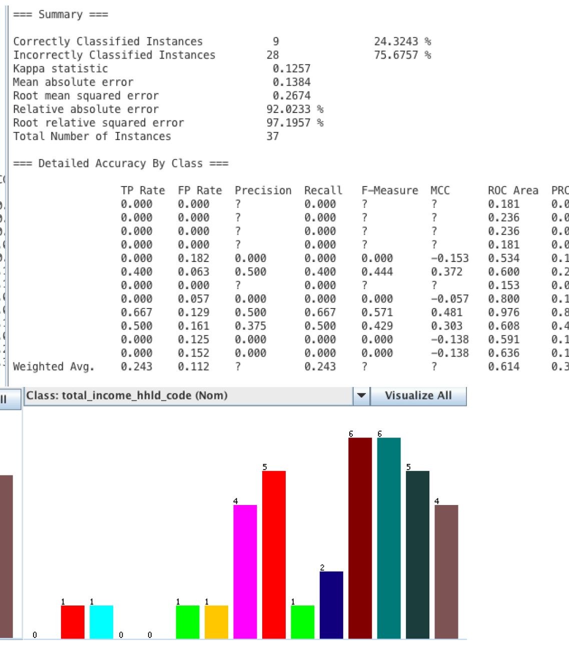
For the result, I did use random forest technique to run through those datasets. The follow figures are the accuracy and the class distribution for each dataset, when I combine all the dataset and upload to the leaderboard, the score of this system is 0.33850.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         894               36.4749 %
Incorrectly Classified Instances      1557               63.5251 %
Kappa statistic                          0.1714
Mean absolute error                      0.106
Root mean squared error                  0.23
Relative absolute error                 92.7806 %
Root relative squared error             96.3004 %
Total Number of Instances             2451

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
                 0.000    0.000    ?          0.000    ?          ?
                 0.000    0.000    ?          0.000    ?          ?
                 0.000    0.000    0.000      0.000    0.000      -0.
                 0.000    0.000    0.000      0.000    0.000      -0.
                 0.000    0.000    ?          0.000    ?          ?
                 0.103    0.003    0.273      0.103    0.150      0.1
                 0.274    0.017    0.293      0.274    0.283      0.2
                 0.000    0.000    0.000      0.000    0.000      -0.
                 0.063    0.005    0.313      0.063    0.105      0.1
                 0.122    0.014    0.353      0.122    0.182      0.1
                 0.063    0.025    0.203      0.063    0.096      0.0
                 0.442    0.326    0.308      0.442    0.363      0.1
                 0.349    0.223    0.336      0.349    0.343      0.1
                 0.570    0.216    0.479      0.570    0.520      0.3
```



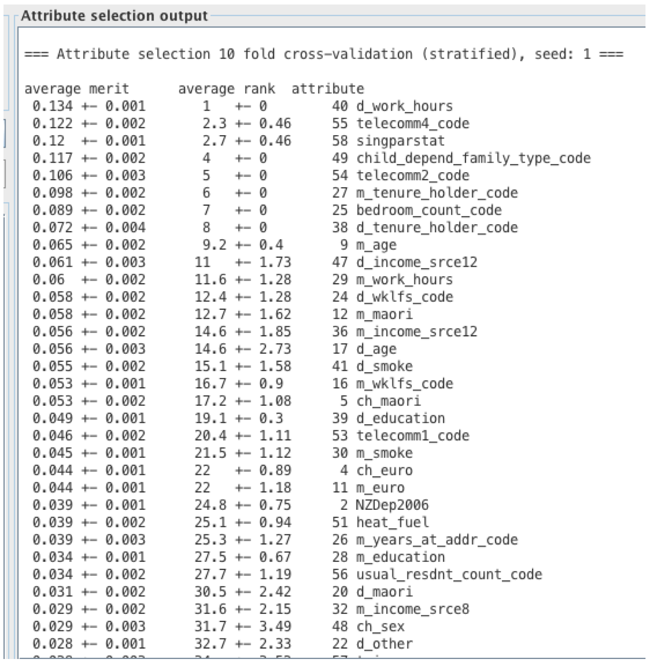
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         116               22.6563 %
Incorrectly Classified Instances       396               77.3438 %
Kappa statistic                          0.1315
Mean absolute error                      0.1219
Root mean squared error                  0.2491
Relative absolute error                 93.8931 %
Root relative squared error             97.7741 %
Total Number of Instances              512

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
                 0.000    0.000    ?          0.000    ?          ?
                 0.000    0.002    0.000      0.000    0.000      -0.
                 0.000    0.010    0.000      0.000    0.000      -0.
                 0.000    0.019    0.000      0.000    0.000      -0.
                 0.241    0.129    0.181      0.241    0.206      0.
                 0.000    0.077    0.000      0.000    0.000      -0.
                 0.467    0.254    0.240      0.467    0.317      0.
                 0.114    0.031    0.211      0.114    0.148      0.
                 0.069    0.023    0.154      0.069    0.095      0.
                 0.067    0.021    0.167      0.067    0.095      0.
                 0.158    0.046    0.214      0.158    0.182      0.
                 0.070    0.055    0.103      0.070    0.083      0.
                 0.327    0.058    0.372      0.327    0.348      0.
                 0.565    0.142    0.354      0.565    0.435      0.
Weighted Avg.    0.227    0.094    ?          0.227    ?          ?
```



```
=== Summary ===

Correctly Classified Instances           9               24.3243 %
Incorrectly Classified Instances        28               75.6757 %
Kappa statistic                          0.1257
Mean absolute error                      0.1384
Root mean squared error                  0.2674
Relative absolute error                 92.0233 %
Root relative squared error             97.1957 %
Total Number of Instances               37

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC
                 0.000    0.000    ?          0.000    ?          ?        0.181     0.0
                 0.000    0.000    ?          0.000    ?          ?        0.236     0.0
                 0.000    0.000    ?          0.000    ?          ?        0.236     0.0
                 0.000    0.000    ?          0.000    ?          ?        0.181     0.0
                 0.000    0.182    0.000      0.000    0.000      -0.153   0.534     0.1
                 0.400    0.063    0.500      0.400    0.444      0.372    0.600     0.2
                 0.000    0.000    ?          0.000    ?          ?        0.153     0.0
                 0.000    0.057    0.000      0.000    0.000      -0.057   0.800     0.1
                 0.667    0.129    0.500      0.667    0.571      0.481    0.976     0.8
                 0.500    0.161    0.375      0.500    0.429      0.303    0.608     0.4
                 0.000    0.125    0.000      0.000    0.000      -0.138   0.591     0.1
                 0.000    0.152    0.000      0.000    0.000      -0.138   0.636     0.1
Weighted Avg.    0.243    0.112    ?          0.243    ?          ?        0.614     0.3
```

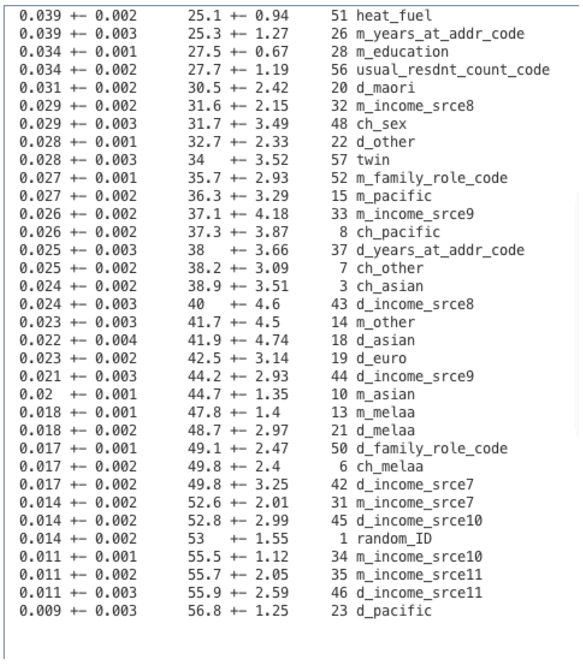
Best system

The best system is base build on Intermediary system, replacing missing value by use KNN and deal with outlier by using IQR knowledge. On the variable type conversion parts, to convert numeric type to binary is better to automatically convert in random forest technique, instead using numeric to binary filter. I tested it out by using 10 fold cross-validation. After dealing with the values and valuable type, I did use correlation attribute evaluates filters to measuring the correlation between each attributes and the class, and try to remove the insignificant attributes one by one from the lowest correaltion. One of the increasing thing I find out is if I remove the lowest correlation attributes (d_pacific, d_income_srce11, m_income_srce)it do not improve any accuracy, when I remove attribute d_income_sre10 the system actually increases 1%. How come remove a higher correlation attribute will increases? The score of this system on leaderboard is 0.34550, and this is my best score so far. The reason for I select this system is because the system is development from my previous system, it has all the advantages from the previous system, and it actually have the best performance.



Attribute selection output

```
=== Attribute selection 10 fold cross-validation (stratified), seed: 1 ===

average merit      average rank  attribute
 0.134 +- 0.001       1   +- 0        40 d_work_hours
 0.122 +- 0.002       2.3 +- 0.46     55 telecomm4_code
 0.12  +- 0.001       2.7 +- 0.46     58 singparstat
 0.117 +- 0.002       4   +- 0        49 child_depend_family_type_code
 0.106 +- 0.003       5   +- 0        54 telecomm2_code
 0.098 +- 0.002       6   +- 0        27 m_tenure_holder_code
 0.089 +- 0.002       7   +- 0        25 bedroom_count_code
 0.072 +- 0.004       8   +- 0        38 d_tenure_holder_code
 0.065 +- 0.002       9.2 +- 0.4       9 m_age
 0.061 +- 0.003      11   +- 1.73     47 d_income_srce12
 0.06  +- 0.002      11.6 +- 1.28     29 m_work_hours
 0.058 +- 0.002      12.4 +- 1.28     24 d_wklfs_code
 0.058 +- 0.002      12.7 +- 1.62     12 m_maori
 0.056 +- 0.002      14.6 +- 1.85     36 m_income_srce12
 0.056 +- 0.003      14.6 +- 2.73     17 d_age
 0.055 +- 0.002      15.1 +- 1.58     41 d_smoke
 0.053 +- 0.001      16.7 +- 0.9      16 m_wklfs_code
 0.053 +- 0.002      17.2 +- 1.08      5 ch_maori
 0.049 +- 0.001      19.1 +- 0.3      39 d_education
 0.046 +- 0.002      20.4 +- 1.11     53 telecomm1_code
 0.045 +- 0.001      21.5 +- 1.12     30 m_smoke
 0.044 +- 0.001      22   +- 0.89      4 ch_euro
 0.044 +- 0.001      22   +- 1.18     11 m_euro
 0.039 +- 0.001      24.8 +- 0.75      2 NZDep2006
 0.039 +- 0.002      25.1 +- 0.94     51 heat_fuel
 0.039 +- 0.003      25.3 +- 1.27     26 m_years_at_addr_code
 0.034 +- 0.001      27.5 +- 0.67     28 m_education
 0.034 +- 0.002      27.7 +- 1.19     56 usual_resdnt_count_code
 0.031 +- 0.002      30.5 +- 2.42     20 d_maori
 0.029 +- 0.002      31.6 +- 2.15     32 m_income_srce8
 0.029 +- 0.003      31.7 +- 3.49     48 ch_sex
 0.028 +- 0.001      32.7 +- 2.33     22 d_other
```



```
0.039 +- 0.002      25.1 +- 0.94     51 heat_fuel
0.039 +- 0.003      25.3 +- 1.27     26 m_years_at_addr_code
0.034 +- 0.001      27.5 +- 0.67     28 m_education
0.034 +- 0.002      27.7 +- 1.19     56 usual_resdnt_count_code
0.031 +- 0.002      30.5 +- 2.42     20 d_maori
0.029 +- 0.002      31.6 +- 2.15     32 m_income_srce8
0.029 +- 0.003      31.7 +- 3.49     48 ch_sex
0.028 +- 0.001      32.7 +- 2.33     22 d_other
```

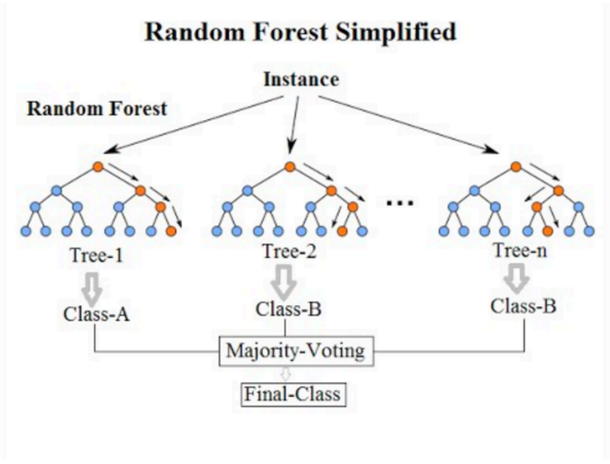
```
0.028 +- 0.003      34   +- 3.52     57 twin
0.027 +- 0.001      35.7 +- 2.93     52 m_family_role_code
0.027 +- 0.002      36.3 +- 3.29     15 m_pacific
0.026 +- 0.002      37.1 +- 4.18     33 m_income_srce9
0.026 +- 0.002      37.3 +- 3.87      8 ch_pacific
0.025 +- 0.003      38   +- 3.66     37 d_years_at_addr_code
0.025 +- 0.002      38.2 +- 3.09      7 ch_other
0.024 +- 0.002      38.9 +- 3.51      3 ch_asian
0.024 +- 0.003      40   +- 4.6      43 d_income_srce8
0.023 +- 0.003      41.7 +- 4.5      14 m_other
0.022 +- 0.004      41.9 +- 4.74     18 d_asian
0.023 +- 0.002      42.5 +- 3.14     19 d_euro
0.021 +- 0.003      44.2 +- 2.93     44 d_income_srce9
0.02  +- 0.001      44.7 +- 1.35     10 m_asian
0.018 +- 0.001      47.8 +- 1.4      13 m_melaa
0.018 +- 0.002      48.7 +- 2.97     21 d_melaa
0.017 +- 0.001      49.1 +- 2.47     50 d_family_role_code
0.017 +- 0.002      49.8 +- 2.4       6 ch_melaa
0.017 +- 0.002      49.8 +- 3.25     42 d_income_srce7
0.014 +- 0.002      52.6 +- 2.01     31 m_income_srce7
0.014 +- 0.002      52.8 +- 2.99     45 d_income_srce10
0.014 +- 0.002      53   +- 1.55      1 random_ID
0.011 +- 0.001      55.5 +- 1.12     34 m_income_srce10
0.011 +- 0.002      55.7 +- 2.05     35 m_income_srce11
0.011 +- 0.003      55.9 +- 2.59     46 d_income_srce11
0.009 +- 0.003      56.8 +- 1.25     23 d_pacific
```

(above figure shows the correlation between each attributes and the class)

Reflecting on findings

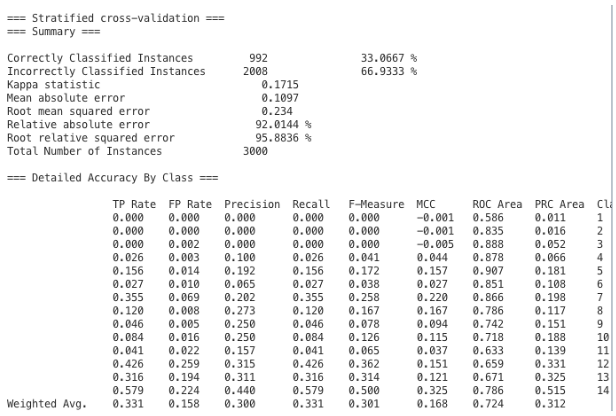
Interpretability of the best system

In my personal point of view, this is easy to interpret my machine learning model. This system can be roughly divided into two parts: firstly, choose a suitable machine learning algorithm that is suitable for the census data. The algorithm I choose is random forest, it will provide a tree structure as a human readable from. Secondly, doing data preprocessing, the aiming is to reduct the data back to the real situation, to feed the machine learning model. Once the model has been trained, the diction tree will not to change when you want to add more test datasets.

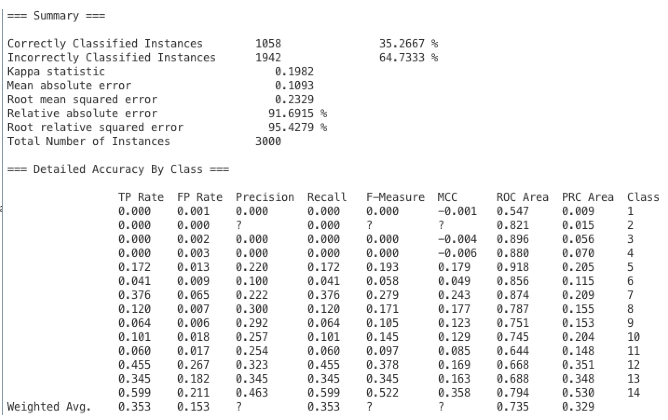


Interpretability of other systems

Compare to my best system, the resample system and combine system might harder to interpret. They all use the same machine learning technique, the hard point is on data reprocessing part. Is hard to explain to the public why resample and split data are useful, it can only explain why I make those attempts and explain from the result but cannot explain from the processing. Another example is when I try do delete the lowest correlation attributes, the accuracy will decrease, is hard to explain why I remove a higher correlation attributes will increase accuracy. In the process of find out the remove which attribute will increase the accuracy, I have to test it out one by one. If we judging from the results: (before and after remove insignificant attributes from this system) The more interpretable system is the better performance this system have. But on the customer or the public viewpoint, have an interpretable system is more important. As a designer, has to balance of those two factors.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         992               33.0667 %
Incorrectly Classified Instances      2008               66.9333 %
Kappa statistic                          0.1715
Mean absolute error                      0.1097
Root mean squared error                  0.234
Relative absolute error                 92.0144 %
Root relative squared error             95.8836 %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cl
                 0.000    0.000    0.000      0.000    0.000      -0.001   0.586     0.011     1
                 0.000    0.000    0.000      0.000    0.000      -0.001   0.835     0.016     2
                 0.000    0.002    0.000      0.000    0.000      -0.005   0.888     0.052     3
                 0.026    0.003    0.100      0.026    0.041      0.044    0.878     0.066     4
                 0.156    0.014    0.192      0.156    0.172      0.157    0.907     0.181     5
                 0.027    0.010    0.065      0.027    0.038      0.027    0.851     0.108     6
                 0.355    0.069    0.202      0.355    0.258      0.220    0.866     0.198     7
                 0.120    0.008    0.273      0.120    0.167      0.167    0.786     0.117     8
                 0.046    0.005    0.250      0.046    0.078      0.094    0.742     0.151     9
                 0.084    0.016    0.250      0.084    0.126      0.115    0.718     0.188     10
                 0.041    0.022    0.157      0.041    0.065      0.037    0.633     0.139     11
                 0.426    0.259    0.315      0.426    0.362      0.151    0.659     0.331     12
                 0.316    0.194    0.311      0.316    0.314      0.121    0.671     0.325     13
                 0.579    0.224    0.440      0.579    0.500      0.325    0.786     0.515     14
Weighted Avg.    0.331    0.158    0.300      0.331    0.301      0.168    0.724     0.312
```



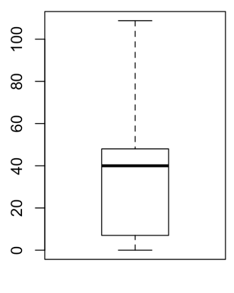
```
=== Summary ===

Correctly Classified Instances        1058               35.2667 %
Incorrectly Classified Instances      1942               64.7333 %
Kappa statistic                          0.1982
Mean absolute error                      0.1093
Root mean squared error                  0.2329
Relative absolute error                 91.6915 %
Root relative squared error             95.4279 %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.001    0.000      0.000    0.000      -0.001   0.547     0.009     1
                 0.000    0.000    ?          0.000    ?          ?        0.821     0.015     2
                 0.000    0.002    0.000      0.000    0.000      -0.004   0.896     0.056     3
                 0.000    0.003    0.000      0.000    0.000      -0.006   0.880     0.070     4
                 0.172    0.013    0.220      0.172    0.193      0.179    0.918     0.205     5
                 0.041    0.009    0.100      0.041    0.058      0.049    0.856     0.115     6
                 0.376    0.065    0.222      0.376    0.279      0.243    0.874     0.209     7
                 0.120    0.007    0.300      0.120    0.171      0.177    0.787     0.155     8
                 0.064    0.006    0.292      0.064    0.105      0.123    0.751     0.153     9
                 0.101    0.018    0.257      0.101    0.145      0.129    0.745     0.204     10
                 0.060    0.017    0.254      0.060    0.097      0.085    0.644     0.148     11
                 0.455    0.267    0.323      0.455    0.378      0.169    0.668     0.351     12
                 0.345    0.182    0.345      0.345    0.345      0.163    0.688     0.348     13
                 0.599    0.211    0.463      0.599    0.522      0.358    0.794     0.530     14
Weighted Avg.    0.353    0.153    ?          0.353    ?          ?        0.735     0.329
```

Ethical consequences

In this census dataset, it provided a lot of ethnic information. Those attributes are used in my system. For example numbers of the benches are split by ethnicity attributes, it might have a segmentation like if those who are an Europe he will spilt into a high income class, otherwise he will into a low income class. On the ethical point of view, spilt by ethnic is not ethical. We should not use any ethnic attributes to predict the incomes.

Not biased system

If to produce a not biased system, one way is to remove all the ethnicity attributes, if there are use any ethnicity informations to judge their income that is biased. For example cannot saids that if one family have higher income than other family because their ethnicity is Europe or Maori ect. Also is better to change to another way to deal with outliers, in the previous systems I design to use IQR to judge whether this value is a outlier or not, then change the outliers to a suitable value. But this is basis of some of the attributes, for instances in the attribute d_working hour there are some people work above 105 hours, we cannot judge them cannot work that long for a week. But we can divide the attributes in several groups (eg 0-20,20-40,40-60,60-80,80-100, and 100-inff), in this way bias problem is fixed. In conclusion, to provide a not biased system is better to do not change any values, (but you can use method to split them) and do not use any attributes that relate to ethical consequences.


0
20
40
60
80
100