TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

INTRODUCTION TO DATA MINING

DEFAULT OF CREDIT CARD CLIENTS DATA SET

17011601
İPEK UYANIK

INTRODUCTION TO DATA MINING PROJECT REPORT

LECTURER
SONGÜL VARLI

ARALIK,2018

1.Analysis of Data

1.1The Business Context

A Taiwan-based credit card issuer wants to better predict the likelihood of default for its customers, as well as identify the key drivers that determine this likelihood. This would inform the issuer's decisions on who to give a credit card to and what credit limit to provide. It would also help the issuer have a better understanding of their current and potential customers, which would inform their future strategy, including their planning of offering targeted credit products to their customers.

1.2.The Data

The credit card issuer has gathered information on 30000 customers. The dataset contains information on 24 variables, including demographic factors, credit data, history of payment, and bill statements of credit card customers from April 2005 to September 2005, as well as information on the outcome: did the customer default or not?

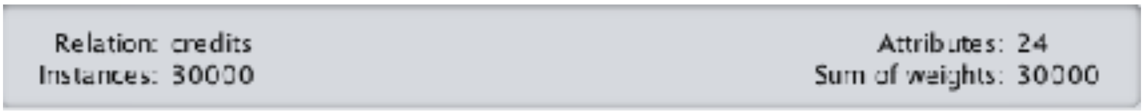
1.3.Attribute Information

This research employed a binary variable, default payment (Yes = 1, No = 0), as the response variable. This study reviewed the literature and used the following 23 variables as explanatory variables:

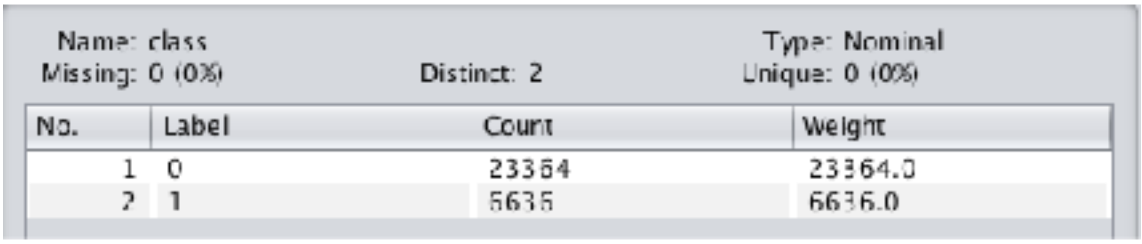
LIMIT_BAL : Amount of given credit in NT dollars (includes individual and family/supplementary credit)
SEX : Gender (1=male, 2=female)
EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE : Marital status (1=married, 2=single, 3=others)
AGE : Age in years
PAY_0 : Repayment status in September, 2005 (-2=no consumption, -1=pay duly, 0=the use of revolving credit, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
PAY_2 : Repayment status in August, 2005 (scale same as above)
PAY_3 : Repayment status in July, 2005 (scale same as above)
PAY_4 : Repayment status in June, 2005 (scale same as above)
PAY_5 : Repayment status in May, 2005 (scale same as above)
PAY_6 : Repayment status in April, 2005 (scale same as above)
BILL_AMT1 : Amount of bill statement in September, 2005 (NT dollar)
BILL_AMT2 : Amount of bill statement in August, 2005 (NT dollar)
BILL_AMT3 : Amount of bill statement in July, 2005 (NT dollar)

BILL_AMT4 : Amount of bill statement in June, 2005 (NT dollar)
BILL_AMT5 : Amount of bill statement in May, 2005 (NT dollar)
BILL_AMT6 : Amount of bill statement in April, 2005 (NT dollar)
PAY_AMT1 : Amount of previous payment in September, 2005 (NT dollar)
PAY_AMT2 : Amount of previous payment in August, 2005 (NT dollar)
PAY_AMT3 : Amount of previous payment in July, 2005 (NT dollar)
PAY_AMT4 : Amount of previous payment in June, 2005 (NT dollar)
PAY_AMT5 : Amount of previous payment in May, 2005 (NT dollar)
PAY_AMT6 : Amount of previous payment in April, 2005 (NT dollar)
CLASS (default.payment.next.month) : Default payment (1=yes, 0=no)

There is no missing value in this data set.



Training data set information


Name: class
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	0	23364	23364.0
2	1	6636	6636.0

Class information

1.4.Outlier Analysis

The data has 4263 outliers.We chose 3 classification methods to decide if we should the outliers or not.Random forest, naive bayes and bayes net are applied.

Database visualization

```
Correctly Classified Instances        24497               81.6567 %
Kappa statistic                          0.3723
Mean absolute error                      0.2776
Root mean squared error                  0.3724
Relative absolute error                 78.8304 %
Root relative squared error             89.7265 %
Total Number of Instances             30000
```

Random forest with outlier

```
Correctly Classified Instances        20757               80.6504 %
Kappa statistic                          0.3732
Mean absolute error                      0.2819
Root mean squared error                  0.3796
Relative absolute error                 78.5528 %
Root relative squared error             89.6117 %
Total Number of Instances             25737
```

Random forest without outlier

Correctly Classified Instances	18623	62.0767 %
Kappa statistic	0.2306	
Mean absolute error	0.403	
Root mean squared error	0.5122	
Relative absolute error	115.9766 %	
Root relative squared error	173.4027 %	
Total Number of Instances	30000	

Naive bayes with outlier

Correctly Classified Instances	13802	53.6271 %
Kappa statistic	0.1711	
Mean absolute error	0.4344	
Root mean squared error	0.5477	
Relative absolute error	121.0428 %	
Root relative squared error	129.2848 %	
Total Number of Instances	25737	

Naive bayes without outlier

Correctly Classified Instances	23480	78.2667 %
Kappa statistic	0.3737	
Mean absolute error	0.2435	
Root mean squared error	0.4198	
Relative absolute error	70.6819 %	
Root relative squared error	101.1432 %	
Total Number of Instances	30000	

Bayes net with outlier

Correctly Classified Instances	20002	77.7169 %
Kappa statistic	0.3726	
Mean absolute error	0.249	
Root mean squared error	0.4242	
Relative absolute error	69.3752 %	
Root relative squared error	100.1431 %	
Total Number of Instances	25737	

Bayes net without outlier

As we see the results above, when we removed the outliers the results of the data with outliers is more accurate.The accuracy has decreased %1 for random forest classification, %8.44 for naive bayes classification, %0.54 for bayes net classification.Therefore we used the data which is with outlier.

1.5.Feature Selection

We use InfoGainAttributeEval that evaluates the worth of an attribute by measuring the information gain with respect to the class for ranking attributes.We can see how important every attribute for the data.
And the results are:

```
Ranked attributes:
 0.09563      23 pay_amt6
 0.0837569    21 pay_amt4
 0.0805345    22 pay_amt5
 0.0745463    18 pay_amt1
 0.0740547    19 pay_amt2
 0.0725449    20 pay_amt3
 0.0265581     1 limit_bal
 0.0235847    17 bill_amt6
 0.0176443    16 bill_amt5
 0.0146199    15 bill_amt4
 0.0132438     8 pay_3
 0.0125285     7 pay_2
 0.0113598    14 bill_amt3
 0.0111748     9 pay_4
 0.0110138    10 pay_5
 0.0109488    11 pay_6
 0.0093496    13 bill_amt2
 0.0089742     6 pay_0
 0.0045149    24 class
 0.0042832    12 bill_amt1
 0.0021422     5 age
 0.0019526     3 education
 0.0002554     4 marriage
 0.0000157     2 sex
 0            25 Outlier

Selected attributes: 23,21,22,18,19,20,1,17,16,15,8,7,14,9,10,11,13,6,24,12,5,3,4,2,25 : 25
```

2.1.Classification

2.1.1.1.Logistic (Functions.Logistic)

There are two 'the best results' for classification in WEKA. One of them is Logistic, correctly classifying %82.0567 of the data.Taken directly from WEKA's page:

"Class for building and using a multinomial logistic regression model with a ridge estimator.
There are some modifications, however, compared to the paper of leCessie and van Houwelingen(1992):
If there are k classes for n instances with m attributes, the parameter matrix B to be calculated will be an m*(k-1) matrix.
The probability for class j with the exception of the last class is

Pj(Xi) = exp(XiBj)/((sum[j=1..(k-1)]exp(Xi*Bj))+1)

The last class has probability

1-(sum[j=1..(k-1)]Pj(Xi))
    = 1/((sum[j=1..(k-1)]exp(Xi*Bj))+1)

The (negative) multinomial log-likelihood is thus:

L = -sum[i=1..n]{
    sum[j=1..(k-1)](Yij * ln(Pj(Xi)))
    +(1 - (sum[j=1..(k-1)]Yij))
    * ln(1 - sum[j=1..(k-1)]Pj(Xi))
    } + ridge * (B^2)

In order to find the matrix B for which L is minimised, a Quasi-Newton Method is used to search for the optimized values of the m*(k-1) variables. Note that before we use the optimization procedure, we 'squeeze' the matrix B into a m*(k-1) vector. For details of the optimization procedure, please check weka.core.Optimization class. Although original Logistic Regression does not deal with instance weights, we modify the algorithm a little bit to handle the instance weights."

```
Correctly Classified Instances       24617               82.0567 %
Kappa statistic                          0.3715
Mean absolute error                      0.2716
Root mean squared error                  0.3695
Relative absolute error                 78.8252 %
Root relative squared error             89.0271 %
Total Number of Instances            30000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,953    0,645    0,839      0,953    0,892      0,400    0,758     0,902     0
                 0,355    0,047    0,681      0,355    0,467      0,400    0,758     0,538     1
Weighted Avg.    0,821    0,513    0,804      0,821    0,798      0,400    0,758     0,821

=== Confusion Matrix ===

     a     b   <-- classified as
 22261  1103 |     a = 0
  4280  2356 |     b = 1
```

2.1.1.2.Multi Class Classifier(meta.MultiClassClassifier)

The 2nd best algorithm in Weka is Multi Class Classifier. It correctly classified %82.0567 too.The description taken directly from WEKA's page also:

"A metaclassifier for handling multi-class datasets with 2-class classifiers. This classifier

```
Correctly Classified Instances       24617               82.0567 %
Kappa statistic                          0.3715
Mean absolute error                      0.2716
Root mean squared error                  0.3695
Relative absolute error                 78.8252 %
Root relative squared error             89.0371 %
Total Number of Instances            30000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,953    0,645    0,839      0,953    0,892      0,400    0,758     0,902     0
                 0,355    0,047    0,681      0,355    0,467      0,400    0,758     0,538     1
Weighted Avg.    0,821    0,513    0,804      0,821    0,798      0,400    0,758     0,821

=== Confusion Matrix ===

     a     b   <-- classified as
 22261  1103 |     a = 0
  4280  2356 |     b = 1
```

is also capable of applying error correcting output codes for increased accuracy."

2.1.2.1.SGD (functions.SGD)

There are two algorithms for the 2nd best algorithms also SGD is one of them which correctly classified %81.9567.The description taken directly from WEKA:
"Implements stochastic gradient descent for learning various linear models (binary class SVM, binary class logistic regression, squared loss, Huber loss and epsilon-insensitive loss linear regression). Globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes, so the coefficients in the output are based on the normalized data.
For numeric class attributes, the squared, Huber or epsilon-insensitve loss function must be used. Epsilon-insensitive and Huber loss may require a much higher learning rate."

```
Correctly Classified Instances       24587               81.9567 %
Kappa statistic                          0.3539
Mean absolute error                      0.1804
Root mean squared error                  0.4248
Relative absolute error                 52.3674 %
Root relative squared error            102.3418 %
Total Number of Instances            30000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,959    0,673    0,834      0,959    0,892      0,390    0,643     0,832     0
                 0,327    0,041    0,694      0,327    0,445      0,390    0,643     0,377     1
Weighted Avg.    0,820    0,533    0,803      0,820    0,793      0,390    0,643     0,731

=== Confusion Matrix ===

     a     b   <-- classified as
 22414   960 |     a = 0
  4453  2173 |     b = 1
```

2.1.2.2.Multi Class Classifier Updateable

As 2nd one Multi Class Classifier Updateable also correctly classified %81.9567.The description taken from WEKA:
"A metaclassifier for handling multi-class datasets with 2-class classifiers. This classifier is also capable of applying error correcting output codes for increased accuracy. The base classifier must be an updateable classifier."

```
Correctly Classified Instances       24587               81.9567 %
Kappa statistic                          0.3539
Mean absolute error                      0.1804
Root mean squared error                  0.4248
Relative absolute error                 52.3674 %
Root relative squared error            102.3418 %
Total Number of Instances            30000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,959    0,673    0,834      0,959    0,893      0,390    0,643     0,832     0
                 0,327    0,041    0,696      0,327    0,445      0,390    0,643     0,377     1
Weighted Avg.    0,820    0,533    0,803      0,820    0,793      0,390    0,643     0,731

=== Confusion Matrix ===

     a     b   <-- classified as
 22414   950 |     a = 0
  4463  2173 |     b = 1
```

2.1.3.OneR (rules.OneR)

The 3rd best classifier correctly classified %81.92 of the data. A program that learns 1−rules from examples.Class for building and using a 1R classifier; in other words, uses the minimum-error attribute for prediction, discretizing numeric attributes.Program 1R is ordinary in most respects. It ranks attributes according to error rate (on the training set), as opposed to the entropy-based measures used in C4. It treats all numerically-valued attributes as continuous and uses a straightforward method to divide the range of values into several disjoint intervals. It handles missing values by treating "missing" as a legitimate value. Appendix A gives pseudocode for 1R.

In datasets with continuously-valued attributes there is a risk of overfitting. In dividing the continuous range of values into a finite number of intervals it is tempting to make each interval "pure", i.e. contain examples that are all of the same class. But just as overfitting may result from deepening a decision tree until all the leaves are pure, so too overfitting may result from subdividing an interval until all the subintervals are pure. To avoid this, 1R requires all intervals (except the rightmost) to contain more than a predefined number of examples in the same class. Based on the results in Holte et al. (1989), the threshold was set at 6 for all datasets except for the datasets with fewest examples (LA,SO) where the threshold was set at 3.

A similar difficulty sometimes arises with nominal attributes. For example, consider a dataset in which there is a nominal attribute that uniquely identifies each example, such as the name of a patient in a medical dataset. Using this attribute, one can build a 1−rule that classifies a given training set 100% correctly: needless to say, the rule will not perform well on an independent test set. Although this problem is uncommon, it did arise in two of the datasets in this study (GL,HO); the problematic attributes have been manually deleted from the datasets.

```
Correctly Classified Instances       24576               81.92   %
Kappa statistic                          0.3515
Mean absolute error                      0.1808
Root mean squared error                  0.4252
Relative absolute error                 52.4738 %
Root relative squared error            102.4457 %
Total Number of Instances            30000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,968    0,675    0,833      0,968    0,892      0,388    0,642     0,831     0
                 0,325    0,048    0,695      0,325    0,443      0,388    0,642     0,375     1
Weighted Avg.    0,819    0,535    0,803      0,819    0,793      0,388    0,642     0,730

=== Confusion Matrix ===

     a     b   <-- classified as
 22420   944 |     a = 0
  4480  2156 |     b = 1
```

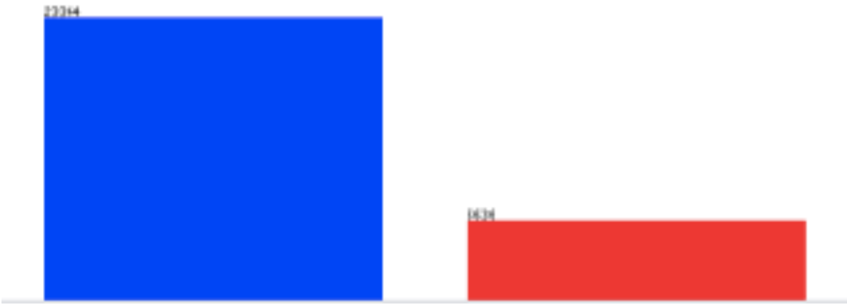
2.2.CLUSTERING

Before show the comparison of the algorithms, the initial values are like below:

Name: class Missing: 0 (0%) Distinct: 2 Type: Nominal Unique: 0 (0%)

No.	Label	Count	Weight
1	0	23364	23364.0
2	1	6636	6636.0

Class: class (Nom) Visualize All



2.2.1.Farthest First Traversal

The best clustering method is Farhest First Traversal with %27.95 incorrectly clustered result.Description taken directly from WEKA's page: "A clustering heuristic that selects as centers the first k points of a farthest-first traversal, and then assigns each of the input points to its nearest center. ... Again, this can be approximated by choosing the first k points of a farthest-first traversal."

```
=== Clustering model (full training set) ===

FarthestFirst
==============

Cluster centroids:

Cluster 0
        60000.0 2 2 2 44.0 0 0 0 0 0 0 43271.0 41539.0 [illegible] no no
Cluster 1
        500000.0 2 1 1 35.0 -2 -2 -1 -1 -2 -1 325.0 -15085.0 [illegible] yes yes


Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      27687 ( 92%)
1       2313 (  8%)


Class attribute: class
Classes to Clusters:

    0     1  <-- assigned to cluster
 21333  2031 | 0
  6354   282 | 1

Cluster 0 <-- 0
Cluster 1 <-- 1

Incorrectly clustered instances :       8385.0   27.95   %
```

2.2.2.Simple K Means

We can show the simple K means clustering as 2nd best clustering. Percentage of the incorrectly clustered instances is 40.43. Definition of algorithm taken from directly from “http://jcsites.juniata.edu/faculty/rhodes/ida/kmeans.html".

This is an effective way to identify clusters

1. Choose a value for K, the number of clusters to be determined
2. Randomly choose K instances within the dataset as the initial cluster centers
3. For each instance
    1. calculate the Euclidean distance between the instance and each of the cluster centers
    2. assign the instance to the cluster with smallest distance
4. For each cluster, calculate a new mean based on the instances now in the cluster
5. If there are any changes to a cluster mean, repeat steps 3-5 with the new set of means

Reassignment of an instance to another cluster will result in a new iteration
Algorithm works for any number of attributes. Two attributes can be graphed on a plane, three in a cube, n attributes in n-space.
Euclidean distances for 4 attributes are generalized as follows: Let the cluster mean, or initial value be (a,b,c,d) and an instance be (i,j,k,l), then
distance = $\mathrm{sqrt}((i-a)^2+(j-b)^2+(k-c)^2+(l-d)^2)$
Step #4 recalculates new a, b, c and d values.

```
kMeans
======

Number of iterations: 9
Within cluster sum of squared errors: 106989.69603625558

Initial starting points (random):

Cluster 0: 130000,2,2,1,31,0,0,0,0,0,0,32215,33690,34859,34851,32827,16732,2333,2000,2000,1000,2000,7131,no,no
Cluster 1: 50000,1,2,2,26,0,0,-1,-1,0,0,46008,3756,195,4225,8355,9351,1007,1000,4776,5000,700,600,no,no

Missing values globally replaced with mean/mode

Final cluster centroids:
                            Cluster#
Attribute      Full Data           0           1
               (30000.0)   (20729.0)    (9271.0)
==================================================
limit_bal    167484.3227 145054.7388 217634.5594
sex                    2           2           2
education              2           2           1
marriage               2           2           2
age              35.4855     34.9361     36.7138
pay_0                  0           0          -1
pay_2                  0           0          -1
pay_3                  0           0          -1
pay_4                  0           0          -1
pay_5                  0           0          -1
pay_6                  0           0          -1
bill_amt1     51223.3309  69372.8253  10642.9329
bill_amt2     49179.0752  66938.6544   9470.4872
bill_amt3     47013.1548  63730.9591   9633.8682
bill_amt4      43262.949  58200.7786   9863.5022
bill_amt5      40311.401   53806.221   9959.5032
bill_amt6     38871.7604  51772.2965   10027.492
pay_amt1       5663.5805   5782.3569   5390.0086
pay_amt2       5921.1635   5705.1913   6404.0551
pay_amt3       5225.6815   4846.0352   6074.5314

=== Model and evaluation on training set ===

Clustered Instances

0      20729 ( 69%)
1       9271 ( 31%)


Class attribute: class
Classes to Clusters:

     0     1  <-- assigned to cluster
 15982  7382 | 0
  4747  1889 | 1

Cluster 0 <-- 0
Cluster 1 <-- 1

Incorrectly clustered instances :      12129.0  40.43   %
```

2.2.3.Make Density Based Clusterer

The 3rd best algorithm is Make Density Based Clusterer with %45.0967 incorrectly clustered data. The description of algorithm taken directly from WEKA is: “Class for wrapping a Clusterer to make it return a distribution and density. Fits normal distributions and discrete distributions within each cluster produced by the wrapped clusterer. Supports the NumberOfClustersRequestable interface only if the wrapped Clusterer does.”

```
=== Model and evaluation on training set ===

Clustered Instances

0      18667 ( 62%)
1      11333 ( 38%)


Log likelihood: -166.50327


Class attribute: class
Classes to Clusters:

     0     1  <-- assigned to cluster
 14751  9813 | 0
  4416  3220 | 1

Cluster 0 <-- 0
Cluster 1 <-- 1

Incorrectly clustered instances :      13529.0   45.0967 %
```