CS699 Project Report – Classification of Bank Costumers' Credit Scores

By

Xinyue Zeng & Chao Gao

Boston University

Nov, 1st, 2022

Introduction

This project is mainly about classifying bank customers' credit scores into three levels "Good", "Standard" and "Poor" in order to help with banks to make appropriate decisions towards different new and old customers. The dataset contains information related to a person's credit pattern that forms a credit score as a criterion for accepting new credit applications.

The dataset contains 23930 tuples and 28 different attributes. The following are specific descriptions of each attribute:

- ID: Represents a unique identification of an entry
- Customer_ID: Represents a unique identification of a person
- Month: Represents the month of the year
- Name: Represents the name of a person
- Age: Represents the age of the person
- SSN: Represents the social security number of a person
- Occupation: Represents the occupation of the person
- Annual_Income: Represents the annual income of the person
- Monthly_Inhand_Salary: Represents the monthly base salary of a person
- Num_Bank_Accounts: Represents the number of bank accounts a person holds
- Num_Credit_Card: Represents the number of other credit cards held by a person
- Interest_Rate: Represents the interest rate on credit card
- Num_of_Loan: Represents the number of loans taken from the bank
- Type_of_Loan: Represents the types of loan taken by a person
- Delay_from_due_date: Represents the average number of days delayed from the payment date
- Num_of_Delayed_Payment: Represents the average number of payments delayed by a person
- Changed_Credit_Limit: Represents the percentage change in credit card limit
- Num_Credit_Inquiries: Represents the number of credit card inquiries
- Credit_Mix: Represents the classification of the mix of credits
- Outstanding_Debt: Represents the remaining debt to be paid (in USD)
- Credit_Utilization_Ratio: Represents the utilization ratio of credit card
- Credit_History_Age_Months: Represents the age of credit history of the person
- Payment_of_Min_Amount: Represents whether only the minimum amount was paid by the person
- Total_EMI_per_month: Represents the monthly EMI payments (in USD)
- Amount_invested_monthly: Represents the monthly amount invested by the customer (in USD)
- Payment_Behaviour: Represents the payment behavior of the customer (in USD)

- Monthly_Balance: Represents the monthly balance amount of the customer (in USD)
- Credit_Score: Represents the bracket of credit score (Poor, Standard, Good)

Statement of Our Data Mining Goal

As we mentioned in Introduction part, our dataset will divide customers' credit scores (Credit_Score) into three levels ("Good", "Standard", "Poor") based on their information. Based on the results of our best model, we are going to predict a new bank customer's credit score and correctly divide it into one of these three levels.

Detailed Description of Data Mining Tools We Used

We used three data mining tools to achieve our classification.

- **Python:**
    i. Import the raw dataset to deal with the missing values
    ii. Output the results as a new file

- **R:**
    i. Removed the outliers for every attribute and outputted it as a new file
    ii. Converted the continuous variables such as monthly income or interest rate into categorical variables and outputted the dataset
    iii. Divided the final preprocessed dataset into training set and testing set with a ratio of 66%:34%

- **Weka:**
    i. Used five different Attribution Selection Methods to create five different training reduced sets and five testing reduced sets
    ii. Chose five different algorithms to make 25 different prediction models and test them. with 10-fold cross-validation and collected the performances' results

Brief Description of Classification Algorithms We Used

- **OneR:**
  One Rule is a simple but accurate classification algorithm that generates one rule for each predictor in the data, then selects the rule with the smallest total error as its "one rule". For each predictor, for each value of that predictor, make a rule as follows: Count how often each value of class appears; find the most frequent class; make the rule assign that class to this value of the predictor. Then calculate the total error of the rules of each predictor and choose the predictor with the smallest total error.

- **Naïve Bayes:**
  Naïve Bayes classifiers based on applying Bayes' theorem with strong independence assumptions between the features. It is highly scalable, requiring a number of parameters linear in the number of variables in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

- **KNN:**
  K-nearest algorithm is a non-parametric supervised learning method used for classification and regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

- **J48:**
  J48 is a classification algorithm which produces decision trees based on information theory. It builds decision trees based on a set of training data in the same way the ID3 algorithm does, by using the concept of information entropy.

- **Random Forest:**
  Random forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging to tree learners.

Brief Description of Attribute Selected by Each Attribute Selection Method

Attribute Evaluator

ClassifierSubsetEval: Evaluates attribute subsets on training data or a separately hold out testing set. Uses a classifier to estimate the 'merit' of a set of attributes.

ClassifierAttributeEval: Evaluates the worth of an attribute by using a user-specified classifier.

InfoGainAttributeEval: Evaluates the worth of an attribute by measuring the information gain with respect to the class.

InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute).

CfsSubsetEval: Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

Subsets of features that are highly correlated with the class while having low intercorrelation are preferred.

WrapperSubsetEval: Evaluates attribute sets by using a learning scheme. Cross validation is used to estimate the accuracy of the learning scheme for a set of attributes.

Search Method

GreedyStepwise: Performs a greedy forward or backward search through the space of attribute subsets. Can also produce a ranked list of attributes by traversing the space from one side to the other and recording the order that attributes are selected.

Ranker: Ranks attributes by their individual evaluations. Use in conjunction with attribute evaluators (ReliefF, GainRatio, Entropy etc).

What parameters we chose for this attribute selection method?

ClassifierSubsetEval: For the attribute evaluator: for the user-specified classifier, we used J48 with default settings, and everything else is on default

For the search method, we used GreedyStepwise with generateRanking: True, numToSelect= 10, and everything else on default

ClassifierAttributeEval: For the attribute evaluator: for the user-specified classifier, we used J48 with default settings, and everything else is on default

For the search method, we used Ranker with generateRanking: True, numToSelect= 10, and everything else on default

InfoGainAttributeEval: For the attribute evaluator: we used default settings

For the search method, we used Ranker with generateRanking: True, numToSelect= 10, and everything else on default

CfsSubsetEval: For the attribute evaluator: we used default settings

For the search method, we used GreedyStepwise with generateRanking: True, numToSelect= 10, and everything else on default

WrapperSubsetEval: For the attribute evaluator: for the user-specified classifier, we used J48 with default settings, and everything else is on default

For the search method, we used GreedyStepwise with generateRanking: True, numToSelect= 10, and everything else on default

Detailed Description of Data Mining Procedure

Python:

For the raw dataset, our first step is to deal with the NA values. Because the data contained enough tuples to make the model and the raw dataset itself had only a small fraction of null values, we decided to delete the rows that contain any null values. We chose Python to complete the first step and the specific steps are below:

```
import pandas as pd

data = pd.read_excel('CreditScoreData.xlsx', sheet_name='Credit_Score_Classification')
data.dropna(inplace=True)


data.to_excel("Preprocessed_Credit_Score_Data.xlsx")
```

After the first step, our dataset reduced from 23929 tuples to 21403 tuples.

As for the current dataset, we found that there were some attributes that uncorrelated with the class attribute. They were “Name”, “ID”, “Customer_ID”, “SSN”, and “Month”. In this case, we decided to delete them directly. Then the dataset became 23 attributes and 21404 tuples.

R:

Part1:

Input the dataset and for each numeric attribute, using IQR method to find the outliers. Removed all rows that contained outliers for every numeric attribute. The dataset was reduced from 21403 tuples to 15255 tuples. The example code is appended below:

```r
# Remove the rows that contain outliers of Age

quartiles <- quantile(data$Age, probs=c(.25, .75), na.rm = FALSE)
IQR1 <- IQR(data$Age)

Lower1 <- quartiles[1] - 1.5*IQR1
Upper1 <- quartiles[2] + 1.5*IQR1


# Remove the rows that contain outliers of 'Annual_Income'

quartiles <- quantile(data$Annual_Income, probs=c(.25, .75), na.rm = FALSE)
IQR2 <- IQR(data$Annual_Income)

Lower2 <- quartiles[1] - 1.5*IQR2
Upper2 <- quartiles[2] + 1.5*IQR2


# Remove the rows that contain outliers of 'Monthly_Inhand_Salary'

quartiles <- quantile(data$Monthly_Inhand_Salary, probs=c(.25, .75), na.rm = FALSE)
IQR3 <- IQR(data$Monthly_Inhand_Salary)

Lower3 <- quartiles[1] - 1.5*IQR3
Upper3 <- quartiles[2] + 1.5*IQR3


# Remove the rows that contain outliers of 'Num_Bank_Accounts'
quartiles <- quantile(data$Num_Bank_Accounts, probs=c(.25, .75), na.rm = FALSE)
IQR4 <- IQR(data$Num_Bank_Accounts)

Lower4 <- quartiles[1] - 1.5*IQR4
Upper4 <- quartiles[2] + 1.5*IQR4
```

```r
# Remove the rows that contain outliers of 'Num_Credit_Card'
quartiles <- quantile(data$Num_Credit_Card, probs=c(.25, .75), na.rm = FALSE)
IQR5 <- IQR(data$Num_Credit_Card)

Lower5 <- quartiles[1] - 1.5*IQR5
Upper5 <- quartiles[2] + 1.5*IQR5


# Remove the rows that contain outliers of 'Interest_Rate'
quartiles <- quantile(data$Interest_Rate, probs=c(.25, .75), na.rm = FALSE)
IQR6 <- IQR(data$Interest_Rate)

Lower6 <- quartiles[1] - 1.5*IQR6
Upper6 <- quartiles[2] + 1.5*IQR6


# Remove the rows that contain outliers of 'Num_of_Loan'
quartiles <- quantile(data$Num_of_Loan, probs=c(.25, .75), na.rm = FALSE)
IQR7 <- IQR(data$Num_of_Loan)

Lower7 <- quartiles[1] - 1.5*IQR7
Upper7 <- quartiles[2] + 1.5*IQR7


# Remove the rows that contain outliers of 'Delay_from_due_date'
quartiles <- quantile(data$Delay_from_due_date, probs=c(.25, .75), na.rm = FALSE)
IQR8 <- IQR(data$Delay_from_due_date)

Lower8 <- quartiles[1] - 1.5*IQR8
Upper8 <- quartiles[2] + 1.5*IQR8


# Remove the rows that contain outliers of 'Num_of_Delayed_Payment'
quartiles <- quantile(data$Num_of_Delayed_Payment, probs=c(.25, .75), na.rm = FALSE)
IQR9 <- IQR(data$Num_of_Delayed_Payment)

Lower9 <- quartiles[1] - 1.5*IQR9
Upper9 <- quartiles[2] + 1.5*IQR9
```

```r
# Remove the rows that contain outliers of 'Changed_Credit_Limit'
quartiles <- quantile(data$Changed_Credit_Limit, probs=c(.25, .75), na.rm = FALSE)
IQR10 <- IQR(data$Changed_Credit_Limit)

Lower10 <- quartiles[1] - 1.5*IQR10
Upper10 <- quartiles[2] + 1.5*IQR10


# Remove the rows that contain outliers of 'Num_Credit_Inquiries'
quartiles <- quantile(data$Num_Credit_Inquiries, probs=c(.25, .75), na.rm = FALSE)
IQR11 <- IQR(data$Num_Credit_Inquiries)

Lower11 <- quartiles[1] - 1.5*IQR11
Upper11 <- quartiles[2] + 1.5*IQR11


# Remove the rows that contain outliers of 'Outstanding_Debt'
quartiles <- quantile(data$Outstanding_Debt, probs=c(.25, .75), na.rm = FALSE)
IQR12 <- IQR(data$Outstanding_Debt)

Lower12 <- quartiles[1] - 1.5*IQR12
Upper12 <- quartiles[2] + 1.5*IQR12


# Remove the rows that contain outliers of 'Credit_Utilization_Ratio'
quartiles <- quantile(data$Credit_Utilization_Ratio, probs=c(.25, .75), na.rm = FALSE)
IQR13 <- IQR(data$Credit_Utilization_Ratio)

Lower13 <- quartiles[1] - 1.5*IQR13
Upper13 <- quartiles[2] + 1.5*IQR13


# Remove the rows that contain outliers of 'Credit_History_Age_Months'
quartiles <- quantile(data$Credit_History_Age_Months, probs=c(.25, .75), na.rm = FALSE)
IQR14 <- IQR(data$Credit_History_Age_Months)

Lower14 <- quartiles[1] - 1.5*IQR14
Upper14 <- quartiles[2] + 1.5*IQR14


# Remove the rows that contain outliers of 'Total_EMI_per_month'
quartiles <- quantile(data$Total_EMI_per_month, probs=c(.25, .75), na.rm = FALSE)
IQR15 <- IQR(data$Total_EMI_per_month)

Lower15 <- quartiles[1] - 1.5*IQR15
Upper15 <- quartiles[2] + 1.5*IQR15


# Remove the rows that contain outliers of 'Amount_invested_monthly'
quartiles <- quantile(data$Amount_invested_monthly, probs=c(.25, .75), na.rm = FALSE)
IQR16 <- IQR(data$Amount_invested_monthly)

Lower16 <- quartiles[1] - 1.5*IQR16
Upper16 <- quartiles[2] + 1.5*IQR16


# Remove the rows that contain outliers of 'Monthly_Balance'
quartiles <- quantile(data$Monthly_Balance, probs=c(.25, .75), na.rm = FALSE)
IQR17 <- IQR(data$Monthly_Balance)

Lower17 <- quartiles[1] - 1.5*IQR17
Upper17 <- quartiles[2] + 1.5*IQR17
```

We calculated for all the upper bounds and lower bounds, and then find the intersection of them and generate the dataset without any outlier.

```
data <- data[data$Age < Upper1 & data$Age > Lower1 & data$Annual_Income < Upper2 &
               data$Annual_Income > Lower2 & data$Monthly_Inhand_Salary < Upper3 &
               data$Monthly_Inhand_Salary > Lower3 & data$Num_Bank_Accounts < Upper4 &
               data$Num_Bank_Accounts > Lower4 & data$Num_Credit_Card < Upper5 &
               data$Num_Credit_Card > Lower5 & data$Interest_Rate < Upper6 &
               data$Interest_Rate > Lower6 & data$Num_of_Loan < Upper7 &
               data$Num_of_Loan > Lower7 & data$Delay_from_due_date < Upper8 &
               data$Delay_from_due_date > Lower8 & data$Num_of_Delayed_Payment < Upper9 &
               data$Num_of_Delayed_Payment > Lower9 & data$Changed_Credit_Limit < Upper10 &
               data$Changed_Credit_Limit > Lower10 & data$Num_Credit_Inquiries < Upper11 &
               data$Num_Credit_Inquiries > Lower11 & data$Outstanding_Debt < Upper12 &
               data$Outstanding_Debt > Lower12 & data$Credit_Utilization_Ratio < Upper13 &
               data$Credit_Utilization_Ratio > Lower13 & data$Credit_History_Age_Months < Upper14 &
               data$Credit_History_Age_Months > Lower14 & data$Total_EMI_per_month < Upper15 &
               data$Total_EMI_per_month > Lower15 & data$Amount_invested_monthly < Upper16 &
               data$Amount_invested_monthly > Lower16 & data$Monthly_Balance < Upper17 &
               data$Monthly_Balance > Lower17,]

write.csv(data,"C:/Users/ZXY/Desktop/Dataset_Without_Outliers.csv", row.names = FALSE)
```

Part2:

After removing all the outliers that may influence the performances of models, we decided to convert all the continuous attributes into categorical by using binning method to divide the range into N intervals with equal size:

```
### PART 2 ###

## Data rearranging ##

data_no_outliers <- read.csv(file = 'C:/Users/ZXY/Desktop/Dataset_Without_Outliers.csv')

# Convert 'Age' to categorical variable from Continuous variable

summary(data_no_outliers$Age)

# Binning method: divide the range into 6 intervals with equal size

data_no_outliers$Age <- cut(data_no_outliers$Age, breaks=6)



# Convert 'Annual_Income' to categorical variable from Continuous variable

summary(data_no_outliers$Annual_Income)

# Binning method: divide the range into 6 intervals with equal size

data_no_outliers$Annual_Income <- cut(data_no_outliers$Annual_Income,
                                      breaks=c(7000, 22000, 44000, 66000, 88000, 110000, 140000),
                                      labels=c('7~22k', '22~44k', '44~66k', '66~88k', '88~110k', '110~140k'))



# Convert 'Monthly_Inhand_Salary' to categorical variable from Continuous variable

summary(data_no_outliers$Monthly_Inhand_Salary)

# Binning method: divide the range into 6 intervals with equal size

data_no_outliers$Monthly_Inhand_Salary <- cut(data_no_outliers$Monthly_Inhand_Salary,
                                             breaks=c(300, 2250, 4200, 6150, 8100, 10050, 12000),
                                             labels=c('300~2250', '2500~4200', '4200~6150', '6150~8100',
                                                      '8100~10050', '10050--12000'))
```

```r
# Convert 'Num_Bank_Accounts' to categorical variable from Continuous variable

summary(data_no_outliers$Num_Bank_Accounts)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Num_Bank_Accounts <- cut(data_no_outliers$Num_Bank_Accounts,
                                         breaks=c(0, 2, 4, 6, 8, 10),
                                         labels=c('0~2', '3~4', '5~6', '7~8', '9~10'))


# Convert 'Num_Credit_Card' to categorical variable from Continuous variable

summary(data_no_outliers$Num_Credit_Card)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Num_Credit_Card <- cut(data_no_outliers$Num_Credit_Card,
                                       breaks=c(0, 2, 4, 6, 8, 10),
                                       labels=c('0~2', '3~4', '5~6', '7~8', '9~10'))



# Convert 'Interest_Rate' to categorical variable from Continuous variable

summary(data_no_outliers$Interest_Rate)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Interest_Rate <- cut(data_no_outliers$Interest_Rate,
                                     breaks=c(0, 7, 14, 21, 28, 35),
                                     labels=c('0~7', '7~14', '14~21', '21~28', '28~35'))


# Convert 'Num_of_Loan' to categorical variable from Continuous variable

summary(data_no_outliers$Num_of_Loan)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Num_of_Loan <- cut(data_no_outliers$Num_of_Loan,
                                   breaks=c(0, 2, 4, 6, 8, 10),
                                   labels=c('0~2', '3~4', '5~6', '7~8', '9~10'))



# Convert 'Delay_from_due_date' to categorical variable from Continuous variable

summary(data_no_outliers$Delay_from_due_date)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Delay_from_due_date <- cut(data_no_outliers$Delay_from_due_date,
                                           breaks=c(-1, 11, 22, 33, 44, 55),
                                           labels=c('0~11', '12~22', '23~33', '34~44', '45~55'))


# Convert 'Num_of_Delayed_Payment' to categorical variable from Continuous variable

summary(data_no_outliers$Num_of_Delayed_Payment)

# Binning method: divide the range into 4 intervals with equal size

data_no_outliers$Num_of_Delayed_Payment <- cut(data_no_outliers$Num_of_Delayed_Payment,
                                              breaks=c(-1, 7, 14, 21, 28),
                                              labels=c('0~7', '8~14', '15~21', '22~28'))
```

```r
# Convert 'Changed_Credit_Limit' to categorical variable from Continuous variable

summary(data_no_outliers$Changed_Credit_Limit)

# Binning method: divide the range into 3 intervals with equal size

data_no_outliers$Changed_Credit_Limit <- cut(data_no_outliers$Changed_Credit_Limit,
                                            breaks=c(-7, 6, 18, 32),
                                            labels=c('-6~6', '7~18', '19~32'))


# Convert 'Num_Credit_Inquiries' to categorical variable from Continuous variable

summary(data_no_outliers$Num_Credit_Inquiries)

# Binning method: divide the range into 3 intervals with equal size

data_no_outliers$Num_Credit_Inquiries <- cut(data_no_outliers$Num_Credit_Inquiries,
                                            breaks=c(-1, 6, 12, 18),
                                            labels=c('0~6', '7~12', '13~18'))



# Convert 'Outstanding_Debt' to categorical variable from Continuous variable

summary(data_no_outliers$Outstanding_Debt)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Outstanding_Debt <- cut(data_no_outliers$Outstanding_Debt,
                                        breaks=c(0, 920, 1840, 2760, 3680, 4600),
                                        labels=c('0~920', '920~1840', '1840~2760', '2760~3680', '3680~4600'))


# Convert 'Credit_Utilization_Ratio' to categorical variable from Continuous variable

summary(data_no_outliers$Credit_Utilization_Ratio)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Credit_Utilization_Ratio <- cut(data_no_outliers$Credit_Utilization_Ratio,
                                                breaks=c(20, 25, 30, 35, 40, 44),
                                                labels=c('20~25', '25~30', '30~35', '35~40', '40~44'))


# Convert 'Credit_History_Age_Months' to categorical variable from Continuous variable

summary(data_no_outliers$Credit_History_Age_Months)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Credit_History_Age_Months <- cut(data_no_outliers$Credit_History_Age_Months,
                                                 breaks=c(0, 80, 160, 240, 320, 404),
                                                 labels=c('0~80', '80~160', '160~240', '240~320', '320~404'))


# Convert 'Total_EMI_per_month' to categorical variable from Continuous variable

summary(data_no_outliers$Total_EMI_per_month)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Total_EMI_per_month <- cut(data_no_outliers$Total_EMI_per_month,
                                           breaks=c(4, 76, 148, 220, 292, 364),
                                           labels=c('4~76', '76~148', '148~220', '220~292', '292~364'))
```

```
# Convert 'Amount_invested_monthly' to categorical variable from Continuous variable

summary(data_no_outliers$Amount_invested_monthly)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Amount_invested_monthly <- cut(data_no_outliers$Amount_invested_monthly,
                                               breaks=c(-1, 90, 180, 270, 360, 450),
                                               labels=c('0~90', '90~180', '180~270', '270~360', '360~450'))


# Convert 'Monthly_Balance' to categorical variable from Continuous variable

summary(data_no_outliers$Monthly_Balance)

# Binning method: divide the range into 5 intervals with equal size

data_no_outliers$Monthly_Balance <- cut(data_no_outliers$Monthly_Balance,
                                       breaks=c(5, 144, 283, 422, 561, 700),
                                       labels=c('5~144', '144~283', '283~422', '422~561', '561~700'))

write.csv(data_no_outliers ,"C:/Users/ZXY/Desktop/data_final.csv", row.names = FALSE)
```

After this, we got a dataset without any outlier and all continuous variable converted into categorical.

Part3:

The last part is to divided the dataset into training set and testing set, with a ratio of 66%:34%

```
### PART 3 ###

# Split the preprocessed dataset into training set and testing set

data_no_outliers <- read.csv(file = 'C:/Users/ZXY/Desktop/data_final.csv')

set.seed(123)
split1 <- sample(c(rep(0, 0.66 * nrow(data)), rep(1, 0.34 * nrow(data))))

train <- data_no_outliers[split1 == 0, ]
test <- data_no_outliers[split1 == 1, ]

write.csv(train,"C:/Users/ZXY/Desktop/training_set.csv", row.names = FALSE)
write.csv(test,"C:/Users/ZXY/Desktop/testing_set.csv", row.names = FALSE)
```

At this time, we got a training set with 10069 tuples and 23 attributes, and a testing set with 5186 tuples and 23 attributes.

Data Mining Result and Evaluation

- **OneR:**

Dataset after preprocessing:

Correctly Classified Instances: 12992, 60.7018%
Incorrectly Classified Instances: 8411, 39.2982%
Mean absolute error: 0.262
Root mean squared error: 0.5118

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.335	0.139	0.290	0.335	0.311	0.185	0.598	0.194	Good
0.766	0.470	0.652	0.766	0.704	0.306	0.648	0.625	Standard
0.466	0.082	0.728	0.466	0.568	0.443	0.692	0.510	Poor
0.607	0.298	0.624	0.607	0.603	0.332	0.655	0.525	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	1040	1595	947
	Standard	1962	8764	2713
	Poor	107	1087	3188

Reduced Dataset 1 (CfsSubsetEval):

Correctly Classified Instances: 2875, 55.4377%
Incorrectly Classified Instances: 2311, 44.5623%
Mean absolute error: 0.2971
Root mean squared error: 0.5451

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.369	0.227	0.199	0.369	0.258	0.112	0.571	0.157	Good

0.697	0.430	0.656	0.697	0.676	0.268	0.633	0.621	Standard
0.394	0.076	0.716	0.394	0.509	0.389	0.659	0.481	Poor
0.554	0.288	0.615	0.554	0.566	0.287	0.633	0,514	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	253	622	399
	Standard	397	1953	628
	Poor	36	229	669

Reduced Dataset 2 (ClassifierAttributeEval):

Correctly Classified Instances: 3304, 63.71%
Incorrectly Classified Instances: 1882, 36.29%
Mean absolute error: 0.2419
Root mean squared error: 0.4919

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.797	0.119	0.506	0.797	0.619	0.566	0.839	0.430	Good
0.681	0.340	0.702	0.681	0.691	0.340	0.670	0.651	Standard
0.499	0.154	0.612	0.499	0.550	0.366	0.673	0.469	Poor
0.637	0.250	0.647	0.637	0.636	0.379	0.693	0.562	Wei.Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	547	373	161
	Standard	123	1910	688
	Poor	16	521	847

Reduced Dataset 3 (ClassifierSubsetEval):

Correctly Classified Instances: 3304, 63.71%

Incorrectly Classified Instances: 1882, 36.29%
Mean absolute error: 0.2419
Root mean squared error: 0.4919

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.797	0.119	0.506	0.797	0.619	0.566	0.839	0.430	Good
0.681	0.340	0.702	0.681	0.691	0.340	0.670	0.651	Standard
0.499	0.154	0.612	0.499	0.550	0.366	0.673	0.469	Poor
0.637	0.250	0.647	0.637	0.636	0.379	0.693	0.562	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	547	373	161
	Standard	123	1910	688
	Poor	16	521	847

Reduced Dataset 4 (InfoGainAttributeEval):

Correctly Classified Instances: 2875, 55.4377%
Incorrectly Classified Instances: 2311, 44.5623%
Mean absolute error: 0.2971
Root mean squared error: 0.5451

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.369	0.227	0.199	0.369	0.258	0.112	0.571	0.157	Good
0.697	0.430	0.656	0.697	0.676	0.268	0.633	0.621	Standard
0.394	0.076	0.716	0.394	0.509	0.389	0.659	0.481	Poor
0.554	0.288	0.615	0.554	0.566	0.287	0.633	0,514	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	253	622	399

	Standard	397	1953	628
	Poor	36	229	669

Reduced Dataset 5 (WrapperSubsetEval):

Correctly Classified Instances: 3304, 63.71%
Incorrectly Classified Instances: 1882, 36.29%
Mean absolute error: 0.2419
Root mean squared error: 0.4919

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.797	0.119	0.506	0.797	0.619	0.566	0.839	0.430	Good
0.681	0.340	0.702	0.681	0.691	0.340	0.670	0.651	Standard
0.499	0.154	0.612	0.499	0.550	0.366	0.673	0.469	Poor
0.637	0.250	0.647	0.637	0.636	0.379	0.693	0.562	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	547	373	161
	Standard	123	1910	688
	Poor	16	521	847

- **Naïve Bayes:**

Dataset after preprocessing:

Correctly Classified Instances: 13631, 63.6873%
Incorrectly Classified Instances: 7772, 36.3127%
Mean absolute error: 0.258
Root mean squared error: 0.465

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.819	0.188	0.425	0.819	0.560	0.495	0.863	0.471	Good
0.530	0.143	0.810	0.530	0.641	0.405	0.723	0.758	Standard
0.732	0.200	0.633	0.732	0.679	0.514	0.779	0.593	Poor
0.637	0.168	0.698	0.637	0.641	0.453	0.761	0.663	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	2546	2572	867
	Standard	457	6069	965
	Poor	106	5016	5016

Reduced Dataset 1 (CfsSubsetEval):

Correctly Classified Instances: 3591, 69.2441%
Incorrectly Classified Instances: 1595, 30.7559%
Mean absolute error: 0.2255
Root mean squared error: 0.4247

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.800	0.127	0.489	0.800	0.607	0.554	0.890	0.595	Good
0.615	0.141	0.837	0.615	0.709	0.483	0.769	0.799	Standard
0.777	0.197	0.658	0.777	0.712	0.559	0.811	0.661	Poor
0.692	0.157	0.732	0.692	0.697	0.517	0.799	0.727	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	549	415	158
	Standard	116	1724	220
	Poor	21	665	1318

Reduced Dataset 2 (ClassifierAttributeEval):

Correctly Classified Instances: 3560, 68.6464%
Incorrectly Classified Instances: 1626, 31.3536%
Mean absolute error: 0.2318
Root mean squared error: 0.4349

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.773	0.122	0.492	0.773	0.601	0.544	0.872	0.520	Good
0.630	0.173	0.811	0.630	0.709	0.462	0.739	0.777	Standard
0.745	0.191	0.654	0.745	0.696	0.537	0.791	0.605	Poor
0.686	0.172	0.718	0.686	0.691	0.497	0.774	0.687	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	530	389	158
	Standard	136	1767	275
	Poor	80	648	1263

Reduced Dataset 3 (ClassifierSubsetEval):

Correctly Classified Instances: 3602, 69.4562%
Incorrectly Classified Instances: 1584, 30.5438%
Mean absolute error: 0.2511
Root mean squared error: 0.4107

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.797	0.120	0.503	0.797	0.617	0.564	0.869	0.444	Good
0.625	0.148	0.833	0.625	0.714	0.484	0.744	0.773	Standard
0.768	0.198	0.653	0.768	0.706	0.550	0.792	0.615	Poor
0.695	0.161	0.730	0.695	0.699	0.516	0.776	0.678	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	547	379	161
	Standard	120	1752	232
	Poor	19	673	1303

Reduced Dataset 4 (InfoGainAttributeEval):

Correctly Classified Instances: 3604, 69.4948%
Incorrectly Classified Instances: 1582, 30.5052%
Mean absolute error: 0.2235
Root mean squared error: 0.429

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.792	0.123	0.495	0.792	0.609	0.554	0.885	0.561	Good
0.627	0.149	0.832	0.627	0.715	0.486	0.764	0.796	Standard
0.768	0.193	0.659	0.768	0.709	0.555	0.808	0.657	Poor
0.695	0.160	0.731	0.695	0.699	0.517	0.794	0.719	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	543	393	162
	Standard	122	1759	232
	Poor	21	652	1302

Reduced Dataset 5 (WrapperSubsetEval):

Correctly Classified Instances: 3559, 68.6271%
Incorrectly Classified Instances: 1627, 31,3729%
Mean absolute error: 0.2386
Root mean squared error: 0.4215

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.796	0.123	0.497	0.796	0.612	0.558	0.881	0.550	Good
0.631	0.175	0.809	0.631	0.709	0.460	0.749	0.779	Standard
0.733	0.188	0.654	0.733	0.691	0.530	0.797	0.620	Poor
0.686	0.172	0.717	0.686	0.691	0.496	0.782	0.697	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	546	393	160
	Standard	124	1770	293
	Poor	16	641	1243

- **KNN (with KNN=10):**

Dataset after preprocessing:

Correctly Classified Instances: 10793, 70.7506%
Incorrectly Classified Instances: 4462, 29.2494%
Mean absolute error: 0.2685
Root mean squared error: 0.3645

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.591	0.069	0.565	0.591	0.578	0.513	0.890	0.561	Good
0.786	0.340	0.731	0.786	0.758	0.450	0.799	0.820	Standard
0.624	0.114	0.728	0.624	0.672	0.533	0.855	0.721	Poor
0.708	0.230	0.708	0.708	0.706	0.486	0.829	0.753	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	1183	634	277
	Standard	781	6486	1604

	Poor	37	1129	3124

Reduced Dataset 1 (CfsSubsetEval):

Correctly Classified Instances: 3642, 70.2275%
Incorrectly Classified Instances: 1544, 29.7725%
Mean absolute error: 0.2615
Root mean squared error: 0.3658

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.659	0.086	0.539	0.659	0.593	0.528	0.889	0.570	Good
0.730	0.275	0.757	0.730	0.743	0.453	0.791	0.813	Standard
0.675	0.144	0.695	0.675	0.685	0.535	0.848	0.696	Poor
0.702	0.207	0.708	0.702	0.704	0.490	0.822	0.742	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	452	276	110
	Standard	214	2046	442
	Poor	20	482	1144

Reduced Dataset 2 (ClassifierAttributeEval):

Correctly Classified Instances: 3619, 69.784%
Incorrectly Classified Instances: 1567, 30.216%
Mean absolute error: 0.2661
Root mean squared error: 0.3679

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.622	0.079	0.545	0.622	0.581	0.514	0.881	0.555	Good
0.743	0.306	0.741	0.743	0.742	0.437	0.787	0.810	Standard
0.654	0.138	0.697	0.654	0.675	0.525	0.847	0.695	Poor
0.700	0.221	0.701	0.700	0.700	0.477	0.818	0.738	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	427	260	97
	Standard	239	2083	490
	Poor	20	461	1109

Reduced Dataset 3 (ClassifierSubsetEval):

Correctly Classified Instances: 3683, 71.0181%
Incorrectly Classified Instances: 1503, 28.9819%
Mean absolute error: 0.2625
Root mean squared error: 0.3681

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.665	0.095	0.515	0.665	0.581	0.513	0.878	0.497	Good
0.741	0.266	0.766	0.741	0.754	0.474	0.789	0.809	Standard
0.677	0.126	0.723	0.677	0.699	0.561	0.839	0.692	Poor
0.710	0.198	0.719	0.710	0.713	0.508	0.817	0.730	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	456	303	126
	Standard	212	2079	422
	Poor	18	422	1148

Reduced Dataset 4 (InfoGainAttributeEval):

Correctly Classified Instances: 3644, 70.2661%
Incorrectly Classified Instances: 1542, 29.7339%
Mean absolute error: 0.2642
Root mean squared error: 0.3666

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.672	0.088	0.538	0.672	0.598	0.533	0.889	0.567	Good
0.731	0.276	0.757	0.731	0.744	0.454	0.789	0.808	Standard
0.669	0.140	0.699	0.669	0.683	0.535	0.843	0.692	Poor
0.703	0.207	0.709	0.703	0.705	0.491	0.820	0.738	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	461	285	111
	Standard	206	2049	451
	Poor	19	470	1134

Reduced Dataset 5 (WrapperSubsetEval):

Correctly Classified Instances: 3680, 70.9603%
Incorrectly Classified Instances: 1506, 29.0397%
Mean absolute error: 0.2582
Root mean squared error: 0.3638

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.659	0.082	0.550	0.659	0.599	0.535	0.890	0.578	Good
0.757	0.297	0.750	0.757	0.753	0.460	0.796	0.811	Standard
0.652	0.123	0.721	0.652	0.685	0.544	0.849	0.703	Poor
0.710	0.212	0.714	0.710	0.711	0.497	0.826	0.745	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	452	272	98
	Standard	216	2122	492
	Poor	18	410	1106

- **Random Forest:**

Preprocessed dataset with bagSizePercent=10, numIterations=10
Reduced dataset with default settings

Dataset after preprocessing:

Correctly Classified Instances: 10545, 69.1249%
Incorrectly Classified Instances: 4710, 30.8751%
Mean absolute error: 0.303
Root mean squared error: 0.3753

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.377	0.035	0.617	0.377	0.468	0.424	0.888	0.564	Good
0.824	0.440	0.688	0.824	0.750	0.401	0.785	0.809	Standard
0.598	0.113	0.721	0.598	0.653	0.511	0.846	0.704	Poor
0.691	0.280	0.689	0.691	0.681	0.440	0.818	0.742	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	754	343	126
	Standard	1194	6799	1887
	Poor	53	1107	2992

Reduced Dataset 1 (CfsSubsetEval):

Correctly Classified Instances: 3788, 73.0428%
Incorrectly Classified Instances: 1398, 26.9572%
Mean absolute error: 0.2487
Root mean squared error: 0.3553

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.576	0.044	0.667	0.576	0.618	0.567	0.906	0.660	Good
0.801	0.335	0.738	0.801	0.768	0.472	0.806	0.821	Standard
0.676	0.115	0.740	0.676	0.707	0.575	0.871	0.752	Poor
0.730	0.225	0.729	0.730	0.728	0.518	0.841	0.777	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	395	172	25
	Standard	274	2246	524
	Poor	17	386	1147

Reduced Dataset 2 (ClassifierAttributeEval):

Correctly Classified Instances: 3709, 71.5195%
Incorrectly Classified Instances: 1477, 28.4805%
Mean absolute error: 0.2296
Root mean squared error: 0.3684

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.573	0.064	0.579	0.573	0.576	0.512	0.888	0.573	Good
0.751	0.279	0.760	0.751	0.756	0.472	0.800	0.825	Standard
0.713	0.151	0.697	0.713	0.705	0.559	0.866	0.730	Poor
0.715	0.209	0.715	0.715	0.715	0.506	0.833	0.761	Wei.Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	393	227	59
	Standard	237	2107	428
	Poor	56	470	1209

Reduced Dataset 3 (ClassifierSubsetEval):

Correctly Classified Instances: 3733, 71.9823%
Incorrectly Classified Instances: 1453, 28.0177%
Mean absolute error: 0.2719
Root mean squared error: 0.3627

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.692	0.099	0.515	0.692	0.591	0.526	0.886	0.498	Good
0.743	0.241	0.784	0.743	0.763	0.501	0.800	0.816	Standard
0.693	0.124	0.731	0.693	0.711	0.577	0.848	0.717	Poor
0.720	0.184	0.731	0.720	0.723	0.529	0.827	0.741	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	475	311	136
	Standard	188	2083	385
	Poor	23	410	1175

Reduced Dataset 4 (InfoGainAttributeEval):

Correctly Classified Instances: 3792, 73.1199%
Incorrectly Classified Instances: 1394, 26.8801%
Mean absolute error: 0.2401
Root mean squared error: 0.3574

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.595	0.044	0.674	0.595	0.632	0.581	0.903	0.670	Good
0.794	0.327	0.741	0.794	0.766	0.471	0.803	0.818	Standard
0.683	0.119	0.735	0.683	0.708	0.575	0.867	0.745	Poor
0.731	0.222	0.730	0.731	0.729	0.519	0.837	0.775	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	408	176	21
	Standard	263	2226	517
	Poor	15	402	1158

Reduced Dataset 5 (WrapperSubsetEval):

Correctly Classified Instances: 3733, 71.9823%
Incorrectly Classified Instances: 1453, 28.0177%
Mean absolute error: 0.2592
Root mean squared error: 0.3576

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.688	0.090	0.539	0.688	0.604	0.541	0.897	0.589	Good
0.740	0.250	0.777	0.740	0.758	0.488	0.806	0.821	Standard
0.699	0.130	0.724	0.699	0.711	0.575	0.859	0.728	Poor
0.720	0.190	0.728	0.720	0.722	0.524	0.836	0.760	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	472	299	105
	Standard	191	2075	405
	Poor	23	430	1186

- **J48(with default settings):**

Dataset after preprocessing:

Correctly Classified Instances: 10571, 69.2953%
Incorrectly Classified Instances: 4684, 30.7047%
Mean absolute error: 0.277
Root mean squared error: 0.3722

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.807	0.117	0.509	0.807	0.625	0.574	0.871	0.454	Good
0.615	0.135	0.843	0.615	0.711	0.489	0.757	0.783	Standard
0.775	0.213	0.640	0.775	0.701	0.540	0.810	0.636	Poor
0.693	0.158	0.732	0.693	0.697	0.517	0.789	0.691	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	1615	1059	496
	Standard	319	5076	629
	Poor	67	2114	3880

Reduced Dataset 1 (CfsSubsetEval):

Correctly Classified Instances: 3592, 69.2634%
Incorrectly Classified Instances: 1594, 30.7366%
Mean absolute error: 0.2735
Root mean squared error: 0.3707

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.720	0.100	0.524	0.720	0.607	0.545	0.884	0.541	Good
0.620	0.154	0.826	0.620	0.708	0.473	0.761	0.768	Standard

0.801	0.223	0.636	0.801	0.709	0.551	0.825	0.629	Poor
0.693	0.169	0.724	0.693	0.695	0.508	0.798	0.692	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	494	325	124
	Standard	153	1739	740
	Poor	39	740	1359

Reduced Dataset 2 (ClassifierAttributeEval):

Correctly Classified Instances: 3726, 71.8473%
Incorrectly Classified Instances: 1460, 28.1527%
Mean absolute error: 0.2497
Root mean squared error: 0.3726

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.627	0.071	0.574	0.627	0.599	0.536	0.858	0.526	Good
0.738	0.254	0.774	0.738	0.756	0.483	0.777	0.775	Standard
0.723	0.154	0.696	0.723	0.709	0.564	0.832	0.655	Poor
0.718	0.197	0.722	0.718	0.720	0.516	0.806	0.703	Wei.Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	430	244	75
	Standard	210	2070	395
	Poor	46	490	1226

Reduced Dataset 3 (ClassifierSubsetEval):

Correctly Classified Instances: 3709, 71.5159%
Incorrectly Classified Instances: 1477, 28.4805%

Mean absolute error: 0.259
Root mean squared error: 0.3633

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.765	0.116	0.501	0.765	0.606	0.548	0.872	0.462	Good
0.729	0.229	0.789	0.729	0.758	0.498	0.796	0.802	Standard
0.673	0.117	0.736	0.673	0.703	0.569	0.845	0.704	Poor
0.715	0.177	0.734	0.715	0.720	0.528	0.822	0.725	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	525	364	158
	Standard	148	2043	397
	Poor	13	397	1141

Reduced Dataset 4 (InfoGainAttributeEval):

Correctly Classified Instances: 3592, 69.2634%
Incorrectly Classified Instances: 1594, 30.7366%
Mean absolute error: 0.2735
Root mean squared error: 0.3707

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.720	0.100	0.524	0.720	0.607	0.545	0.884	0.541	Good
0.620	0.154	0.826	0.620	0.708	0.473	0.761	0.768	Standard
0.801	0.223	0.636	0.801	0.709	0.551	0.825	0.629	Poor
0.693	0.169	0.724	0.693	0.695	0.508	0.798	0.692	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	494	325	124

	Standard	153	1739	213
	Poor	39	740	1359

Reduced Dataset 5 (WrapperSubsetEval):

Correctly Classified Instances: 3739, 72.098%
Incorrectly Classified Instances: 1447, 27.902%
Mean absolute error: 0.2556
Root mean squared error: 0.3597

Detailed Accuracy By Class:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.662	0.084	0.544	0.662	0.597	0.532	0.886	0.563	Good
0.743	0.253	0.776	0.743	0.759	0.488	0.799	0.803	Standard
0.709	0.133	0.721	0.709	0.715	0.579	0.854	0.706	Poor
0.721	0.191	0.727	0.721	0.723	0.524	0.829	0.739	Wei. Avg

Confusion Matrix:

	True Class			
Predicted Class		Good	Standard	Poor
	Good	454	286	94
	Standard	203	2082	399
	Poor	29	436	1203

Discussion and conclusion

According to the 25 results, we decided to use the Accuracy and Root Mean Square Error to select our best model. Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition: Accuracy = (Num of correct predictions)/(Total number of predictions). And for Root Mean Square Error, it is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance. It is extremely helpful to give a single number to judge a model's performance in

machine learning, whether it be during training, cross-validation or monitoring after deployment. Root Mean Square Error is one of the most widely used measures for this.

25 Models Summary

Model	Accuracy	RMSE
OneR Cfs	55.44%	0.5451
OneR ClassifierAttribute	63.71%	0.4919
OneR ClassifierSubset	63.71%	0.4919
OneR InfoGainAttribute	55.44%	0.5451
OneR WrapperSubset	63.71%	0.4919
Naive Bayes Cfs	69.24%	0.4247
Naive Bayes ClassifierAttribute	68.65%	0.4349
Naive Bayes ClassifierSubset	69.46%	0.4107
Naive Bayes InfoGainAttribute	69.49%	0.429
Naive Bayes WrapperSubset	68.63%	0.4215
KNN Cfs	70.23%	0.3658
KNN ClassifierAttribute	69.78%	0.3679
KNN ClassifierSubset	71.02%	0.3681
KNN InfoGainAttribute	70.27%	0.3666
KNN WrapperSubset	70.96%	0.3638
Random Forest Cfs	73.04%	0.3553
Random Forest ClassifierAttribute	71.52%	0.3684
Random Forest ClassifierSubset	71.98%	0.3627
Random Forest InfoGainAttribute	73.12%	0.3574
Random Forest WrapperSubset	71.98%	0.3576
J48 Cfs	69.26%	0.3707
J48 ClassifierAttribute	71.85%	0.3726
J48 ClassifierSubset	71.52%	0.3633
J48 InfoGainAttribute	69.26%	0.3707
J48 WrapperSubset	72.10%	0.3597

From the chart above, we chose out best model as the one using Random Forest with CfsSubsetEval attribute selection method, it has relatively highest Accuracy 73.04% and lowest RMSE 0.3553.

What We Learn

Data Preprocessing and Preparation:

Data processing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning. From this project, we learned the importance of it. If we have bad data in our train model, it will lead to an improperly trained model that won't actually be relevant to the analysis. Depending on our dataset, first we need to do the data quality assessment, such as finding whether the data types are matched or whether there are mixed data types in one feature. We also need to observe the Missing values and calculate for the outliers. Second, in data cleaning process, we need to deal with the above problems. For the missing values, we could ignore them or fill in missing values, depends on the dataset we have. And for outliers we need to delete them in order to make sure they will not influence out model, because outliers can have a huge impact on data analysis. We chose the IQR method to detect the outliers in out project, but there are also other ways to find out the outliers such as Cook's distance. We also learned how to fixing noisy data such as using binning method to set the data into smaller groups of more similar data.

Model Selection:

From choosing our own models, we learned we could choose the good algorithm for a dataset from the below conditions. First, the size of our dataset. If the dataset is small, we could choose algorithms with high bias or low variance like linear regression. If the dataset is sufficiently large such as the one we used in our project, we can go for a low bias/high variance algorithms such as KNN, or Decision trees and our project's results also proved this. Second, we need to follow each algorithm's logic and requirements. For example, if we decide to use Naïve Bayes, we need to confirm that all the features are independent. If they are not independent, the performance of the model will be quite influenced. We understand this deeply from our project by realizing that the models came from Naïve Bayes worked not as good as others such as Decision Tree. Last, when the dataset is relatively complicate such as includes many features or tuples, we need to choose relatively complicate and multidimensional algorithms such as Random Forest. Algorithms like OneR will have a relatively bad performance.

Attribute Selection:

For selecting the optimal attributes for modelling, we learned data reduction reduces the size of data so that it could be used for analysis more efficiently. The data set may have a large number of attributes, but some of those attributes can be irrelevant or redundant. The goal of attribute subset selection is to find a minimum set of attributes such that dropping of those irrelevant attributes does not much affect the utility of data and the cost of data analysis could be reduced. Mining on a reduced data set also makes the discovered pattern easier to understand. Additionally, a search strategy is needed for any attribute selection technique that is based on evaluating attribute subsets rather than single attributes. This includes primarily attribute selection wrappers, but also some of attribute selection filters. Two major flavors of filtering attribute selection algorithms can be distinguished depending on whether they evaluate individual attributes or candidate attribute subsets. Attribute selection has to be always considered as a part of the modeling process in a broader sense. Its effects have to be therefore evaluated using an independent data subset.

Contributions of the Project

Data Part:

- Data preprocessing, preparation (Xinyue Zeng)
- Creating training and testing sets (Xinyue Zeng)
- Creating 10 reduced datasets for training and testing (Chao Gao)
- Creating 30 models and collecting their results (Chao Gao: 18/30, Xinyue Zeng: 12/30)

Report Part:

- Introduction (Xinyue Zeng)
- Statement of Our Mining Goal (Xinyue Zeng)
- Detailed Description of Data Mining Tools We Used (Xinyue Zeng)
- Brief Description of Classification Algorithms We Used (Xinyue Zeng)
- Brief Description of Attribute Selected By Each Attribute Selection Method (Chao Gao)
- Detailed Description of Data Mining Procedure (Xinyue Zeng)
- Data Mining Results and Evaluation (Chao Gao, Xinyue Zeng)
- What we Learn (Chao Gao, Xinyue Zeng)

References

https://www.saedsayad.com/oner.htm#:~:text=OneR%2C%20short%20for%20%22One%20Rule,each%20predictor%20against%20the%20target

https://en.wikipedia.org/wiki/Random_forest

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://www.kdnuggets.com/2020/05/guide-choose-right-machine-learning-algorithm.html

https://monkeylearn.com/blog/data-preprocessing/

https://developers.google.com/machine-learning/crash-course/classification/accuracy

https://c3.ai/glossary/data-science/root-mean-square-error-rmse/#:~:text=What%20is%20Root%20Mean%20Square,true%20values%20using%20Euclidean%20distance