Contents

List of Figures

List of Tables

Chapter 3

Problem Formulation

In this thesis, we ask two questions about a new business invoice when giving instances of historical invoices and outcomes:

- Would the invoice payment delay or not?
- If it would delay, how long the delay would be?

To answer these two questions, we are going to build a classification model that identifying to which of a set of outcome categories a new invoice belongs, on the basis of a training set of data containing instances whose outcome is known.

We formulate the invoice outcome prediction task as a supervised learning problem: given instances of past invoices and their outcomes, build a model that can predict when a newly-issued invoice will be paid, if no advanced actions are taken.

And this model shall help us understand the characteristics of delayed invoices and problematic customers. In other words, it doesn't not only identify the payment delay, but also evaluate the customers.

3.1 Invoice Outcome Definition

Usually in supervised learning, each instance, which is the invoice here, is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory outcome).

In the case of invoice analytics, the input object is all relevant information except the payment result of one invoice, while the desired output value is the payment result of the invoice, which we called outcome of the invoice in this project.

Although there are various matrices to measure the outcome of invoice payment, like Days Sales Outstanding or Collection Effectiveness Index. In our case invoices are labeled based on the actual amounts collected in a specified time period.

Also, as we are interested in more than whether the invoice is going to delay or not, we shall define the outcomes for two cases, like shown in Figure 3-1.

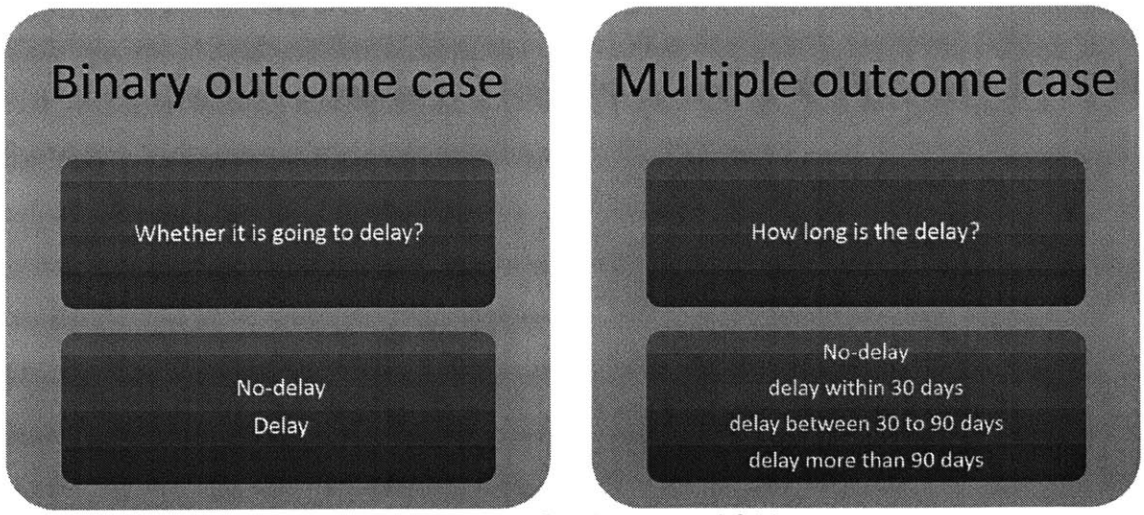

Figure 3-1: Two cases of invoice outcome definition

3.1.1 Binary outcome case

In this case, we only want to know whether a newly-issued invoice is going to be paid lately or not. The problem therefore becomes a binary classification problem: we simply need to classify the given set of invoices into two groups, no-delay invoices and delay invoices.

Accordingly, an important point is that the two groups are not symmetric rather than overall accuracy, the relative proportion of different types of errors is of interest. For example, a false positive (detecting a delay when it is no-delay) is considered differently from a false negative (fail to detect the delay when it is actually going to delay). And also, we may care about the balance of the dataset in terms of two groups, which shall be discussed in Section 4.2.1.

3.1.2 Multiple outcome case

The purpose of setting multiple (more than two) outcomes, as shown in Figure 3-1, for one invoice is to determine the delay level of the invoice. As shown in the figure, we set the outcome to be four classes:

1. No delay
2. Delay within 30 days
3. Delay between 30 to 90 days
4. Delay more than 90 days

It is worth noticing that, these four classes are commonly used in the invoice collection business, where each class corresponds to a customized collection strategy [33]. And it does not necessarily have a balanced numbers of invoices in each classes.

Chapter 4

Data and Pre-processing

The data of this project comes from a Fortune 500 firm, which specializes in oilfield services. It operates in over 90 countries, providing the oil and gas industry with products and services for drilling, formation evaluation, completion, production and reservoir consulting. And at the same time, it generates a large amount of business invoices all over the world.

The author has been able to access the databases of the company's closed invoices for three consecutive months, September to November of 2014, with around 210,000 invoices in total.

The research was first conducted on the September data, as it is the only available dataset at the very beginning of this project. And when the data of October and November came, the author already has a machine learning model, which was further tested and calibrated with the new data.

The author presents the a detailed description and preliminary analysis on the September data in Section 4.1 and Section 4.2. Data of the other two months are very similar, which shall be analyzed in the model robustness part.

4.1 Data Description

An invoice a commercial document issued by a seller, which is the oilfield service firm here, to a buyer, relating to a sale transaction and indicating the products, quantities, and agreed prices for products or services the seller had provided the buyer.

Name	Meaning
Customer Number	-
Customer Name	-
Document Number	-
Reference	Reference number in the database
Profit Center	-
Document Date	Invoice generating date
Posting Date	Invoice posting date
Document Currency	Amount of the invoice
Currency	Currency of the invoice amount
User Name	-
Clearing Date	Invoice clearing date
Entry Date	Invoice closing date
Division	-
Group	-
Payment Term	The "buffer" time of payment after the invoice issuing
Credit Representative	-

Table 4.1: Information on a typical electronic invoice

A typical electronic invoice issued by the company contains information in Table 4.1. It reveals essential information about the deal between the buyer and the seller, as well as the invoice collection mechanism of the buyer, like the profit center and the division.

It is important to notice that, all the invoices given in the format of Table 4.1, have been closed. In other words, the buyers have collected all the amount of the invoices and put that into the database. And when we are talking about the data of different month (September, October or November, 2014), it refers to the invoices closed in that month – it could be issued in any time before or within the close date.

Also, although there are lots of interpretations of "Payment Term", which usually refers to the "discounts and allowances are reductions to a basic price of goods or services", it represents in how many days the seller is expected to pay since the invoice issuing.

4.2 Preliminary Analysis

In this section, the author presents preliminary analysis on various dimensions of the invoice dataset, mostly with traditional statistics tools.

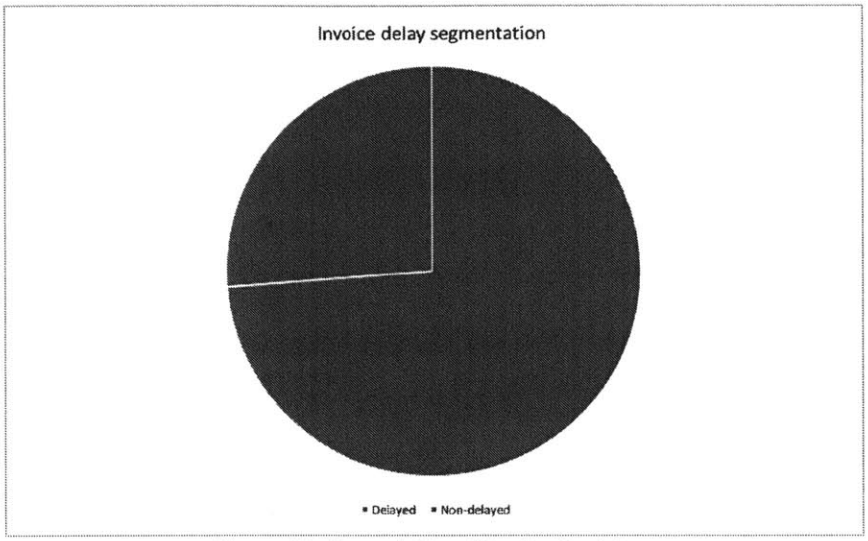

Figure 4-1: Segmentation of invoices by delays

Some statistical facts of the dataset of September invoices:

- There are totally **72464** invoices in the database, **73%** of them are paid lately (delayed invoices)(See in Figure4-1).
- There are **4291** unique customers, which means average 17 invoices per customer in that month.
- For the multi-outcome segmentation based on delay days, the distribution of different classes is shown in Figure 4-2.

4.2.1 Invoice delay

Non-delayed invoices are all alike; every delayed invoice is delayed in its own way. The average delayed dates of the delayed invoices are around **27** dates. However, out of the

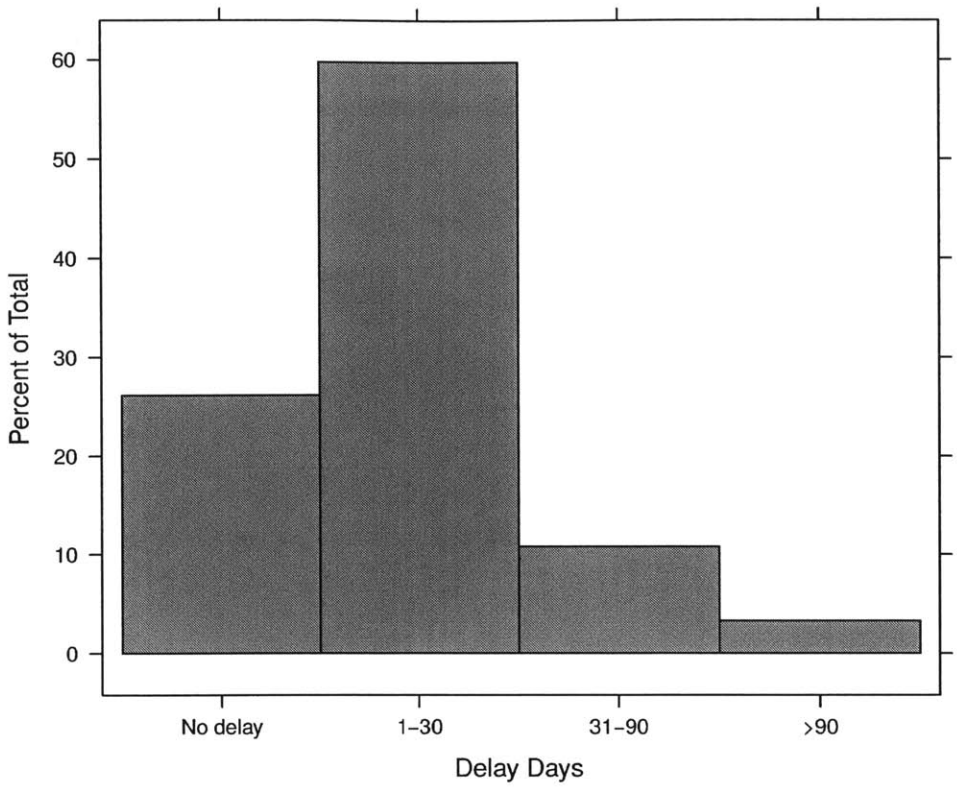

Figure 4-2: Histogram of multiple invoice outcomes

73% delayed invoices, the actual delayed dates are very different, as shown in Figure 4-3.

The distribution of the delayed dates are very similar to the famous power law distribution[34]. It shows there are large amount of delayed invoices only delayed for less than a short period, like 15 days. However, there also exits a long tail of the distribution, which represents the problematic invoices with very long delays.

This shall

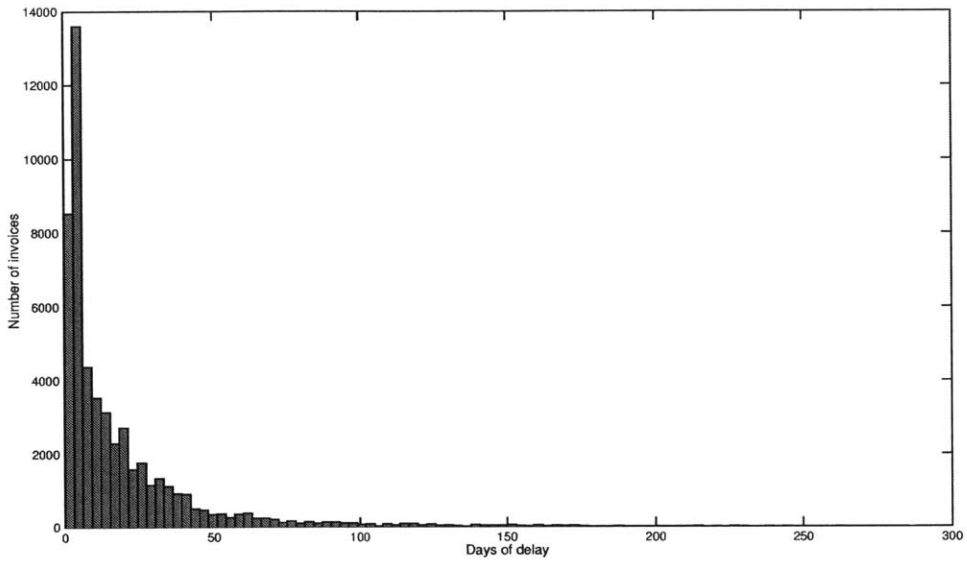

Figure 4-3: Histogram of delayed days of delayed invoices

4.2.2 Payment terms

Another interesting information of an invoice is its payment term. As mentioned before, the payment term in this database means the "buffer" time of payment after invoice issuing. It is not easy to know how the seller assign the payment term for each invoice, as it may be part of the business negotiation. However, a glimpse of the payment term distribution in Figure 4-4 can help one better understand the invoice data.

It is found in the database that, the set of possible payment terms is {30 60 10 45 90 180 35 0 120 21 50 30 60 70 75 42 20}. It is quite clear from the Figure 4-4 that most of the invoices have standard payment terms: 30 or 60 days.

4.2.3 Invoice amount & delay

One naive hypothesis on the reason of invoice delay is the invoice amount: the higher invoice amount, the more likely it is going to delay. In other words, is it true that, for the purpose of financial stability, buyers will delay the payment of invoice with large

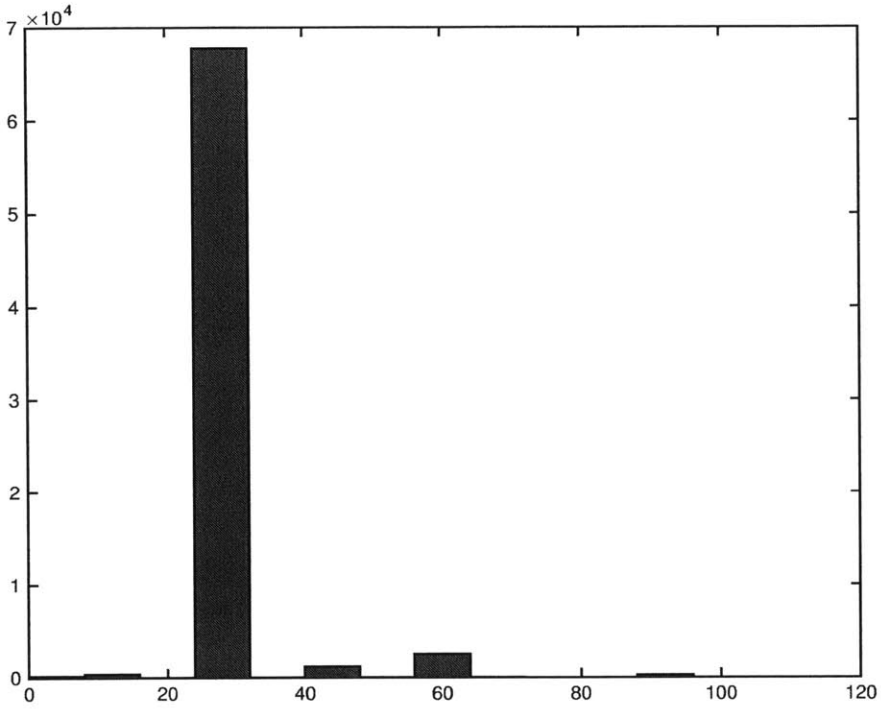

Figure 4-4: Sample

amount?

To answer this question, we first use the box plot to analyze the trends. As mentioned in Section 3, there are two cases of outcomes for the invoice delay. The binary outcome case is simply asking delay or not and is plotted in Figure4-5, while the multiple outcome case, which is asking how the delay would be, is plotted in Figure4-6.

To further verify the intuition from Figure 4-5 and Figure 4-6, we then plot the average amount of the delayed invoices versus their delayed days in Figure 4-7. It is clear that there is no obvious correlation between invoice amount and invoice delays in this figure.

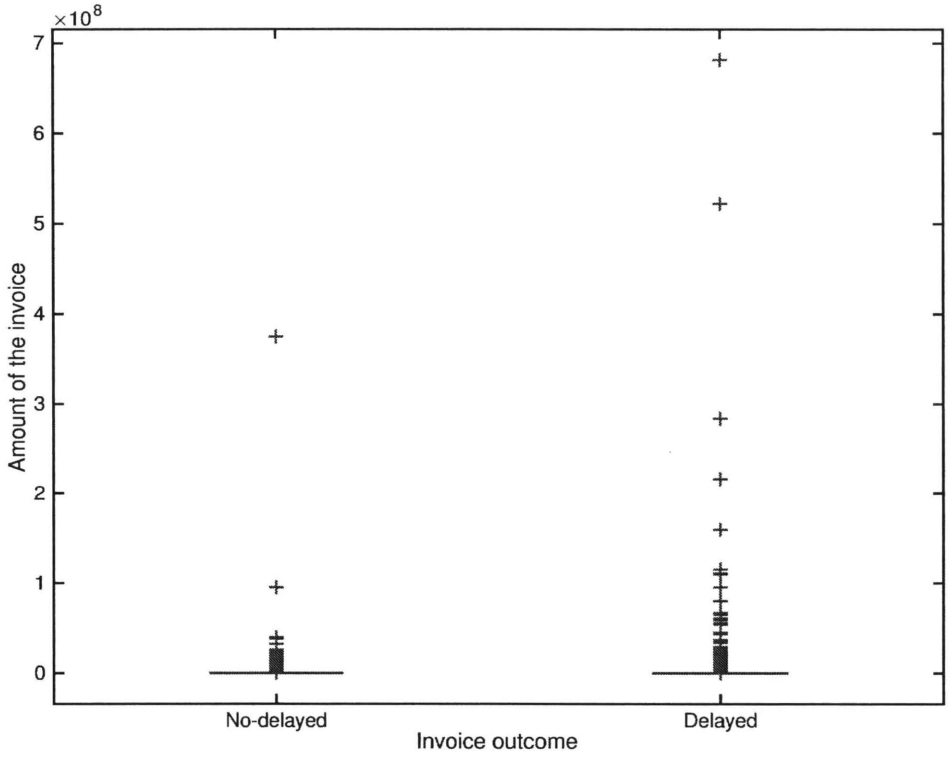

Figure 4-5: Invoice amount and delay or not

It also reminds us, it is hard to use a single variable to predict the delay of the invoice. We shall collect more information of the invoice and adopt more advanced models to understand and predict the invoices.

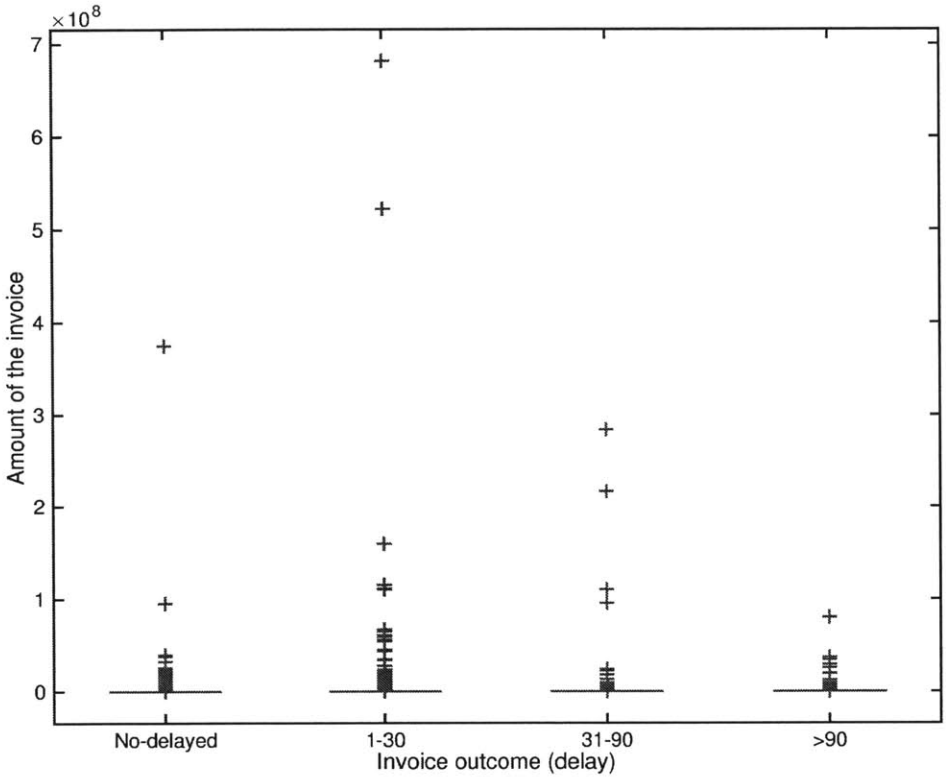

Figure 4-6: Invoice amount and delay level

4.3 Feature Construction

In the field of statistical learning, a feature (of an instance) is an individual measurable property of a phenomenon being observed. Features are usually numeric, but categrocial or structural features are also possible, such as strings and graphs are used in syntactic pattern recognition. One correlated concept of "feature" is the explanatory variable used in regression.

In the case of business invoices, the initial set of raw features are the information present in the previous section. However, there are three problems with raw features:

- Some of the information can be redundant and too large to manage.
- Some categorical information has been stored in numerical formats.

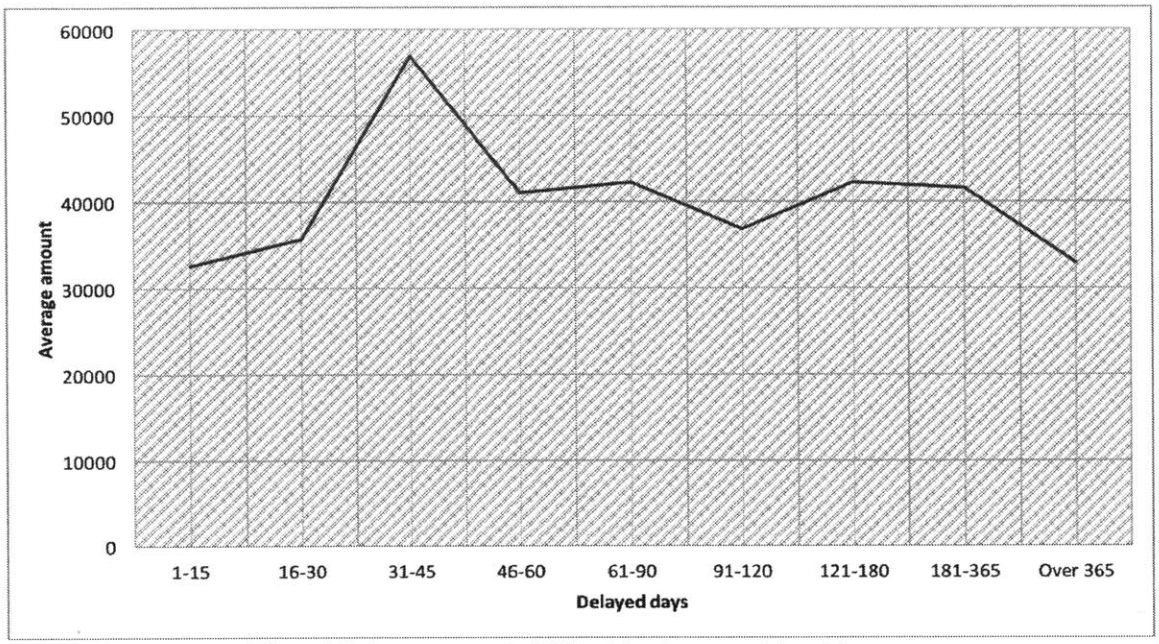

Figure 4-7: Average amount of delayed invoice versus delayed days

- Customer information is not enough, but can be extracted.

Therefore, as shown in Figure 4-8, a preliminary step in the applications of machine learning into the invoice delay prediction is to select a subset of features and construct a new and reduced set of features to facilitate learning, at the same time, to improve generalization and interpretability of the prediction result.

4.3.1 Selection of information

The first step of invoice data preprocessing is to select a subset of information from the database. It is equivalent with asking, given one invoice, what information on it might be relevant to its payment? The subset is shown in Table 4.2.

Basically, the subset in Table 4.2 keeps the amount, the owner and the dates of the invoice, as well as the handler. It contains almost all the information of one invoice, except the product or service, which unfortunately is not available due to data privacy.

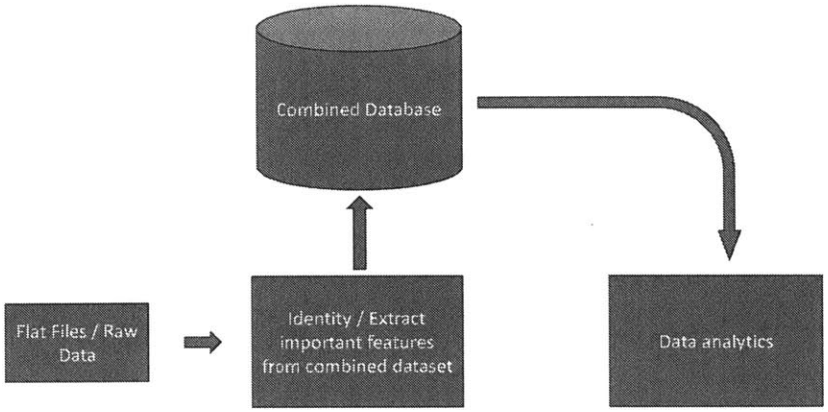

Figure 4-8: Construction of an integrated database of invoice and customer statistics used in machine learning models

Name	Meaning
Customer Number	-
Document Date	Invoice generating date
Posting Date	Invoice posting date
Document Currency	Amount of the invoice
Clearing Date	Invoice clearing date
Entry Date	Invoice closing date
Division	-
Payment Term	The "buffer" time of payment after the invoice issuing
Credit Representative	-

Table 4.2: Selected subset of invoice information

The payment term has also been kept, because it is crucial on see if the invoice is delayed or not, as one shall see later.

4.3.2 Two levels of features

One may realize that, the subset, as in Table 4.2, provides limited information, especially on the customer that the invoice belongs to. However, to understand and predict the payment of invoices, one needs to know more about the characteristics of the one pays the invoice. In this case, the payer is the customer and which customer the invoice

belongs to contains a large amount of information of what will happen on this invoice payment.

That's why there should be two levels of features of one invoice: invoice level and customer level.

Invoice level features refer to the amount, the payment term, the division and the various dates of the invoice. At the same time, the project aggregates the historical invoices for each of the customers and builds a profile accordingly. The customer profile then become the customer level features of certain invoice.

For one customer, its historical invoice data, even only for one month, can lead to a rich profile with various characteristics. Some of the elements of the customer profile include:

1. Number of paid invoices
2. Number of delayed invoices
3. Total amount of paid invoices
4. Total amount of delayed invoices
5. Delay ratio (Ratio between 1 & 2)
6. Delay amount ratio (Ratio between 3& 4)
7. Average payment term
8. Average delayed days
9. ...

They are mostly statistical facts about one customer and customer level features of the invoice in the machine learning application.

As shown in Figure 4-9 and Figure4-10, the total number of invoices of customers is similar to power law distribution: they are a large number of customer have small

number of invoices, but there is a long tail of customers with huge amount of invoices. The distribution of customers' delayed days is similar – most of guys only delayed for a short period of time, while a few delayed for really long time.

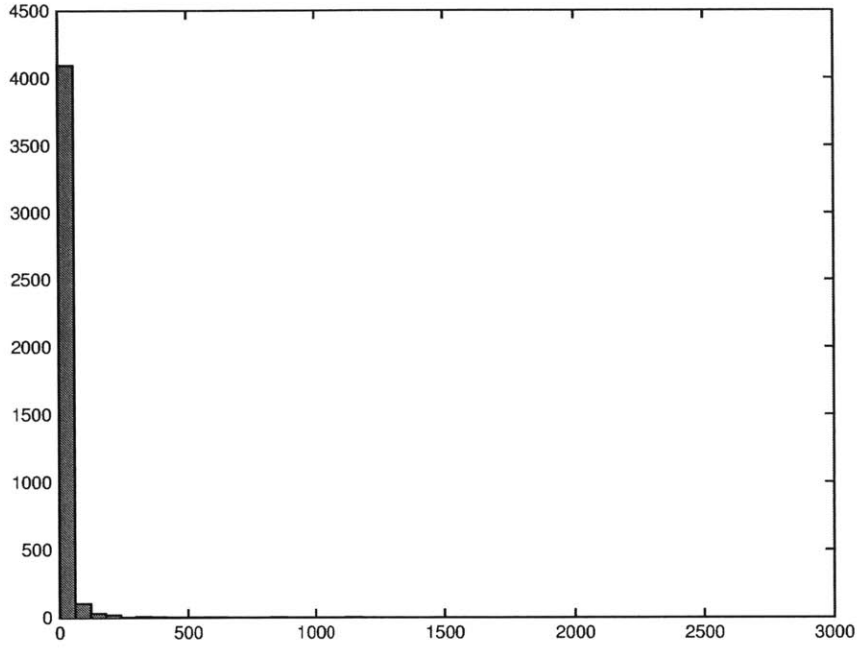

Figure 4-9: Histograms of customers' total number of invoices

Another interesting dimension of the customer is its delay ratio, which is the number of delayed invoice over the number of paid invoices, as shown in Figure 4-11. It reveals the customer's payment record: if one has a delay ratio near zero, this customer is a "good" customer – it pays every bill within the payment term. And the pattern in Figure 4-11 tells what kind of customers the machine is facing: there are large numbers of good and bad customers, but very few in between. In other words, if randomly picking one customer out the database, it is very likely it has an extreme delay ratio.

The pattern is even more obvious if we look at Figure 4-12, which plots histogram

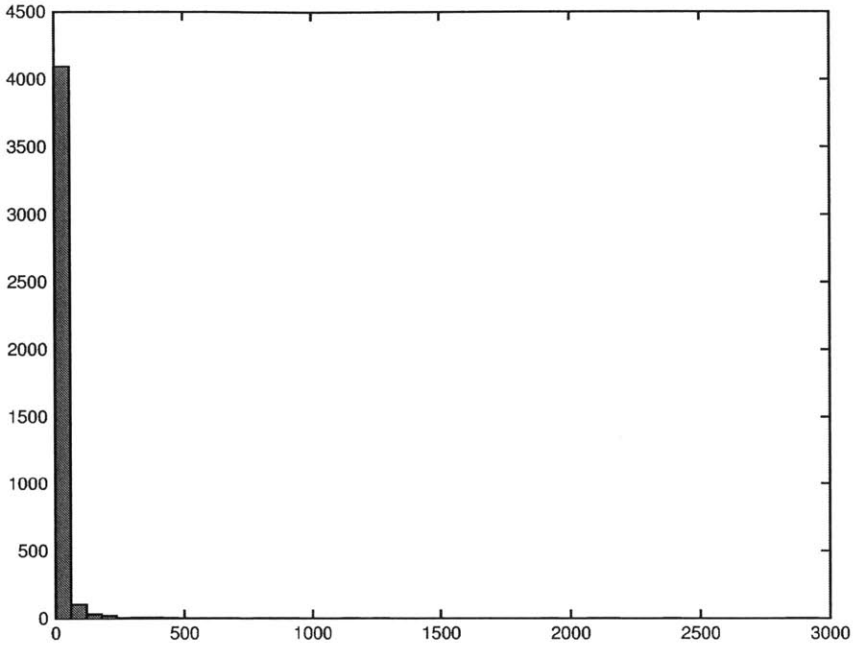

Figure 4-10: Histograms of customers' average delay days

of the amount delay ratio [1] of each customer.

Therefore, the invoice collection mechanism is dealing with extreme count-parties – they are either very well or very badly behaviored. For applying machine learning to solve this problem, it is both a challenge and an opportunity.

4.3.3 Extra information & unexpected features

In the machine learning community, extracting or selecting features is regarding as a combination of art and science. The subset selection and double-level feature structure in previous section has done the "science" part. This section introduces the "art" part of feature construction of invoices.

[1] Ratio between the customer's total amount of delayed invoice over its total amount of paid invoices

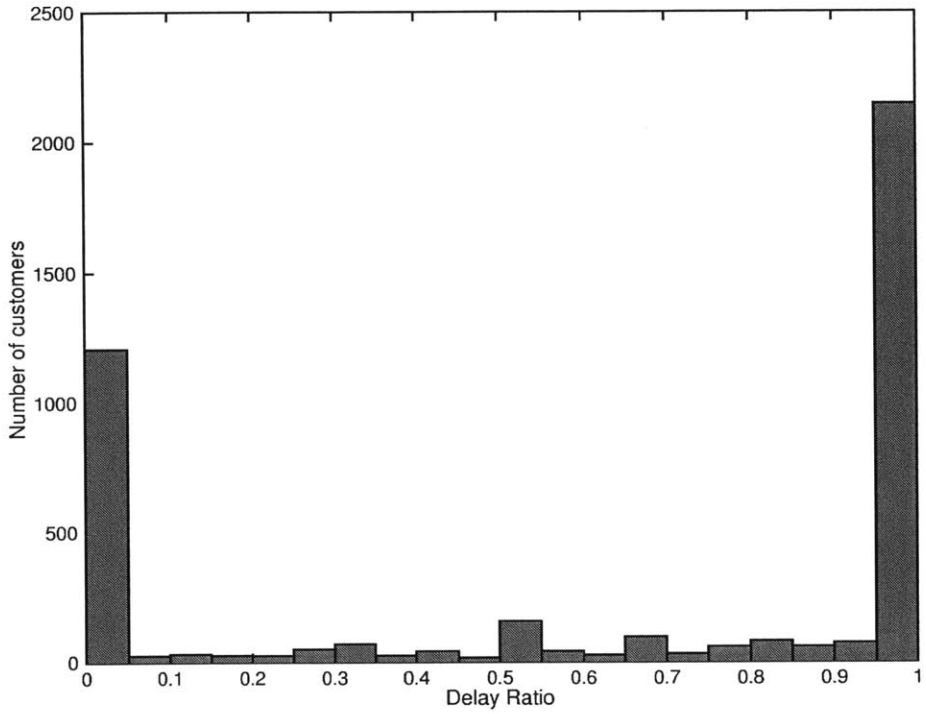

Figure 4-11: Histogram of delay ratio of customers

As the machine is supposed to detect the pattern of the invoice payments, extra information on the invoice could come from the business logic behind the invoice payments and financial stability. One element of the financial stability is the stable cash flow, especially at the end of the month, when the firm pays the salary and other fees. Therefore, if one invoice is due at the month end, it might increase its change of delay. We define a binary variable I_{ME} as follows:

$$I_{ME} = \begin{cases} 1 & \text{if the invoice is due with three days of the month end} \\ 0 & \text{otherwise} \end{cases}$$

There are 28.79% of the invoices are due at the month end (See in Figure 4-13). It is quite surprising considering the narrow range of "month end" – only three days!

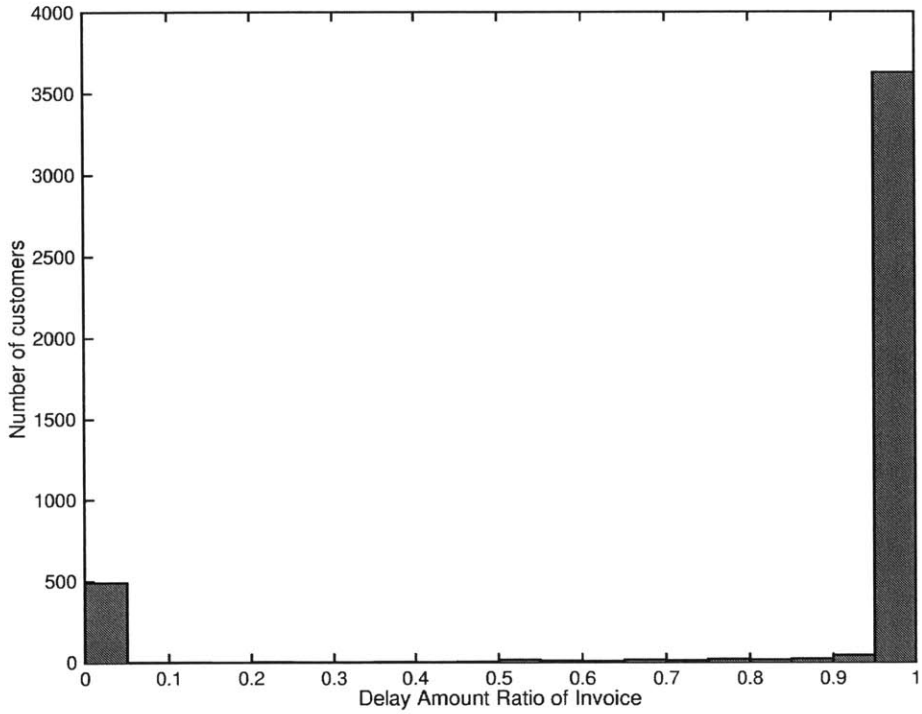

Figure 4-12: Histogram of amount delay ratio of customers

Similarly, we also build a binary indicator I_{HM} such as:

$$I_{HM} = \begin{cases} 1 & \text{if the invoice is due with first half of the month end} \\ 0 & \text{otherwise} \end{cases}$$

It turns out there are more invoices due at the second half of the month, as shown in Figure 4-13.

4.3.4 A full list of features

The full list of the features used in the machine learning algorithms is shown in Table 4.3, there are fourteen of them.

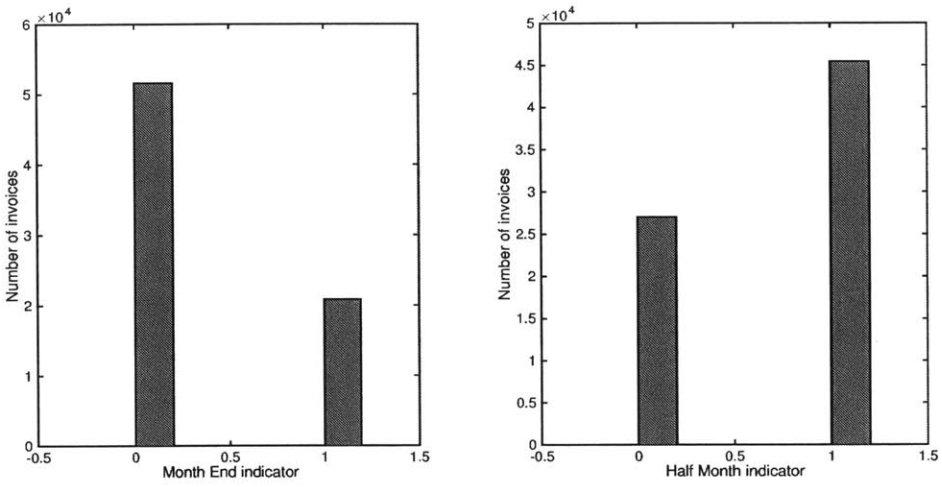

Figure 4-13: Distribution of binary indicators (features) of Month End and Half Month

It is worth noticing that, all the features are generated from the one-month data of business invoices.

Feature	Explanation
past_due	The amount of the invoice
due_date_end	Month end indicator
middle_month	Middle month indictor
no_invoice	Total number of invoice of the invoice owner
no_delay	Total number of delayed invoice of the invoice owner
sum_invoice	Total sum of invoice amount of the invoice owner
sum_delay	Total sum of delayed invoice amount of the invoice owner
ave_delay	Average delay of the delayed invoices of the invoice owner
ave_invoice	Average delay of all the invoices of the invoice owner
ratio_no	Ratio of no_delay and no_invoice
ratio_sum	Ratio of sum_delay and sum_invoice
ave_buffer	Average payment term of the invoice owner
div	Division of the invoice
sale_rep	Sales representative of the invoice

Table 4.3: The full list of features of invoices

Chapter 5

Analysis with Unsupervised Learning

Before applying supervised learning to do the invoice classification, it looks very interesting to take a look at how much all these invoices with different outcomes actually differ with each other. The way to do that is through the unsupervised learning, like clustering and principle component analysis (PCA).

5.1 Principle Component Analysis (PCA)

We then apply the other unsupervised learning method, principal component analysis, to understand the unlabeled data.

In our case, PCA analysis uses the orthogonal transformation to convert the set of invoices of possibly correlated features into a set of values of linearly uncorrelated variables called principal components (PC). The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (accounts for as much of the variability in the data as possible), and each succeeding

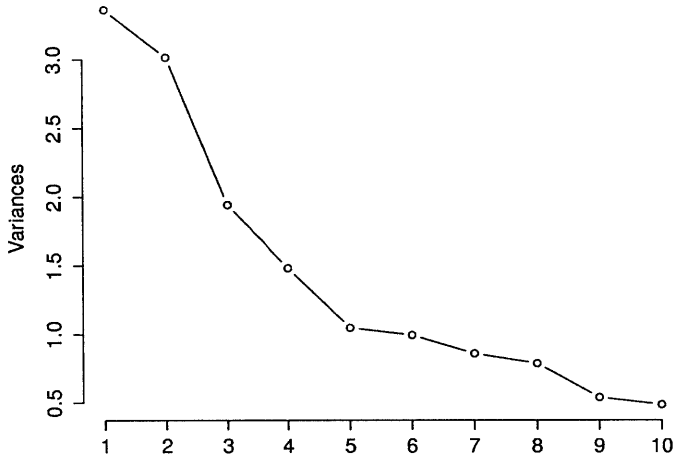

Figure 5-1: Total variance versus the number of principle components

component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components[35].

We plot the variance versus number of PC in Figure 5-1. It shows we need at lease five PCs to account most of the variability in the data.

And the first two PCs only account for 22.4% and 20.1% of the variance. With only these two PCs, one is unable to separate the two groups of invoices well, as shown in Figure 5-2.

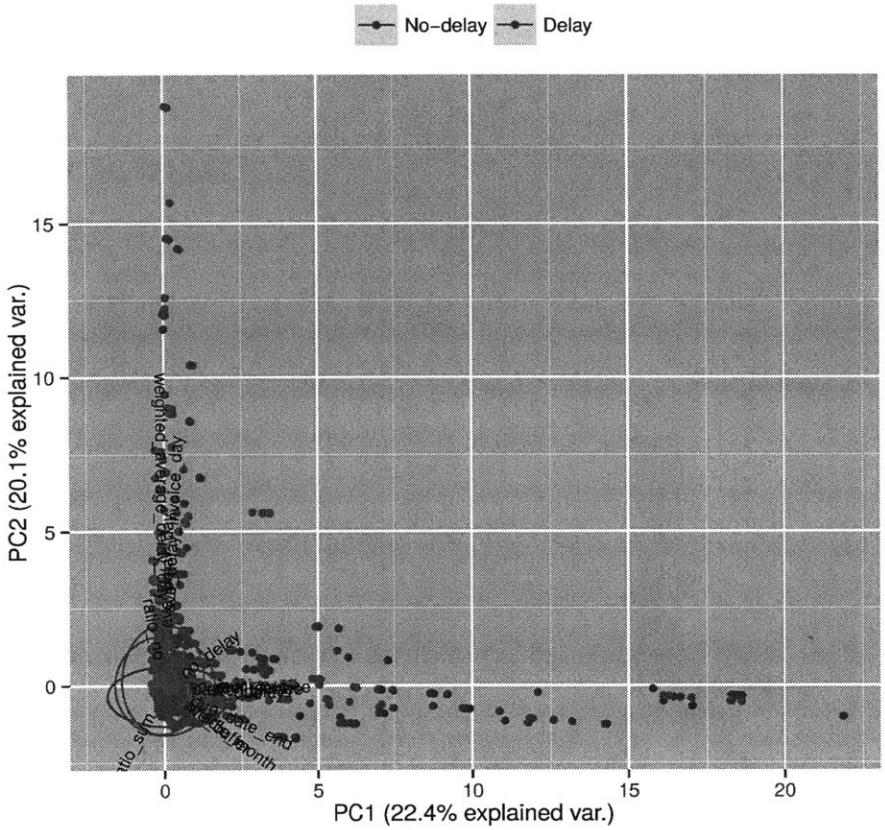

Figure 5-2: Principle component analysis of invoice features

5.2 Clustering

We also apply clustering to group a set of instances in such a way that instances in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

Usually, the similarity between two instances are represented by the feature vector distance. And therefore, to find the best number of clusters, we want the average intra-distance of the cluster to be small, but the inter-distance to be large. One intuitive way of finding the number of clusters, k, is shown in Figure 5-3:

- Try different k, looking at the change in the average distance to centroid, as k increases.
- Average falls rapidly until right k, then changes little.

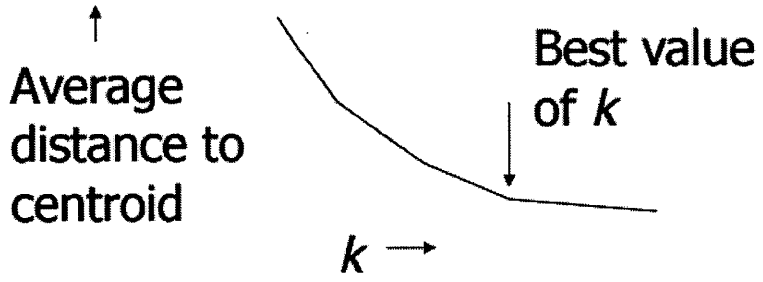

Figure 5-3: How to pick the right number of clusters in unsupervised learning

In the invoice case, we are asking, if ignoring the known outcomes of the instants, how many different types of invoices are there? And how good we can cluster them through the features? Is the optimal number of grouping/clustering them same with the number of classes we assigned in the Section 3.1?

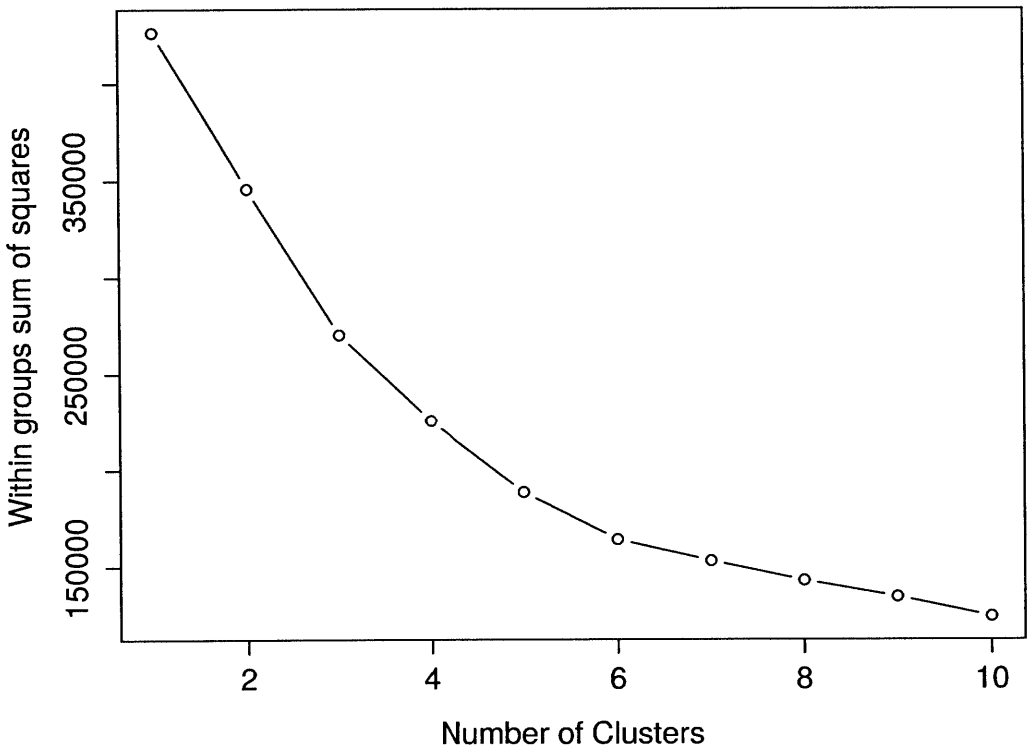

Figure 5-4: Average distance to centroid (Within groups sum of squares, also called WSS) versus number of clusters

Therefore, we plot the average distance to centroid versus number of clusters in Figure 5-4. Ideally, we shall see one "critical" point on $k = 2$, if the delay and on-time invoices are two very different groups. However, it shows there is no obvious optimal number of clusters in the unsupervised learning here, as the slope of the curve does not change abruptly in any k.

We could certainly cluster the invoice data into two or four groups without knowing the actual outcomes, which is visualized in Figure 5-5 and Figure 5-6.

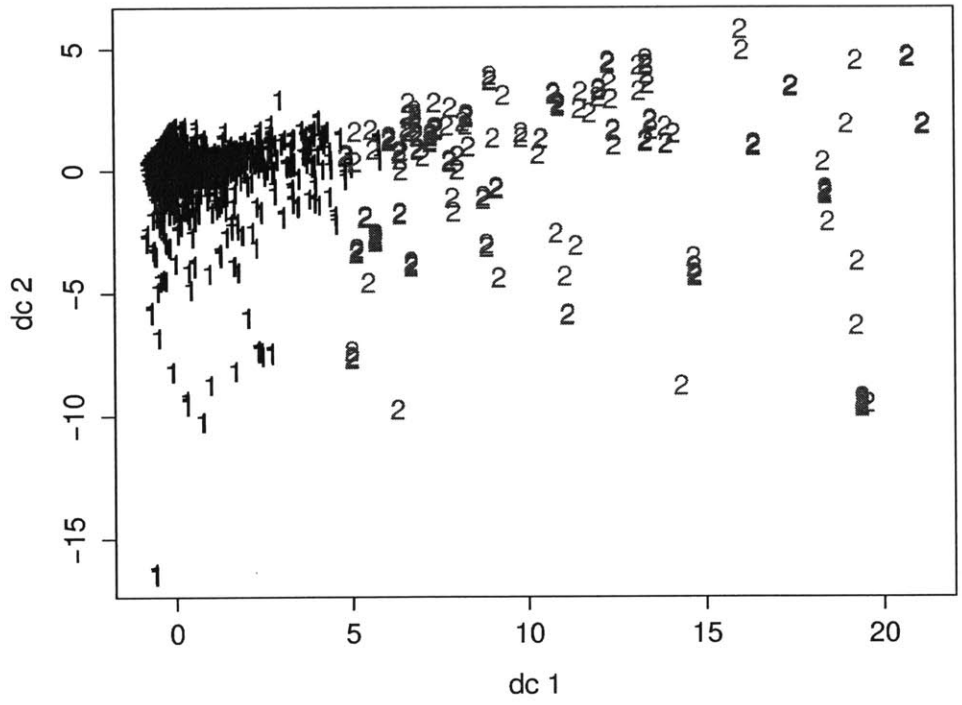

Figure 5-5: Clustering of invoices into two classes, plotted with first two discriminant components (DC)

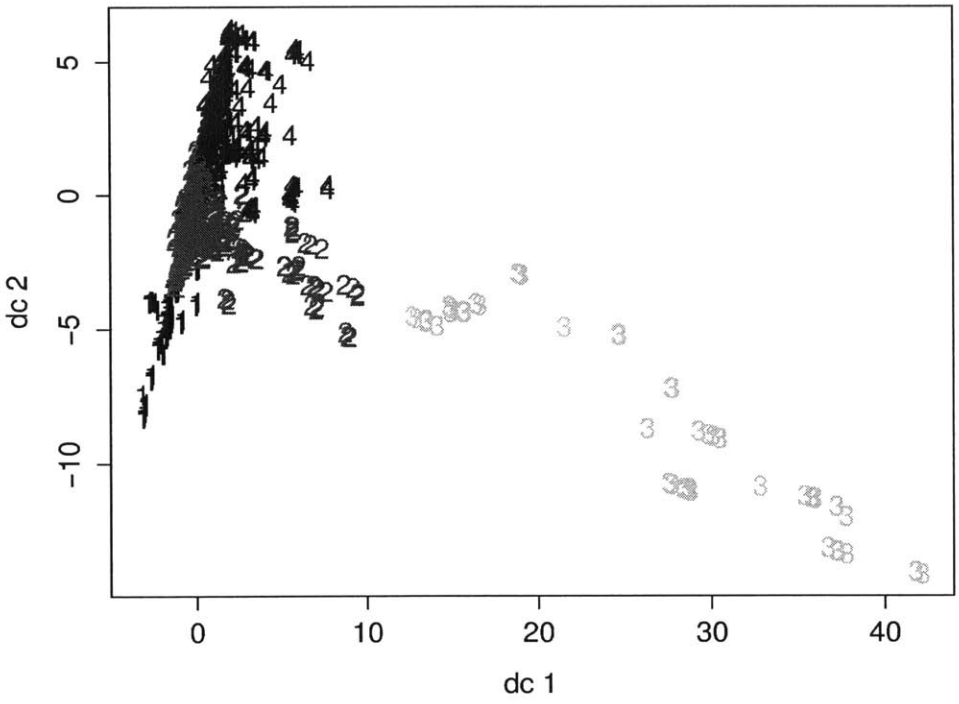

Figure 5-6: Clustering of invoices into four classes, plotted with first two discriminant components (DC)

Chapter 6

Prediction with Supervised Learning

The task we formulated in Section3 is a typical supervised classification problem: given a set of data instances (invoices) expressed by a set of features and class labels (outputs), build a model that classifying a new invoice into two (or four) outcomes (6-1).

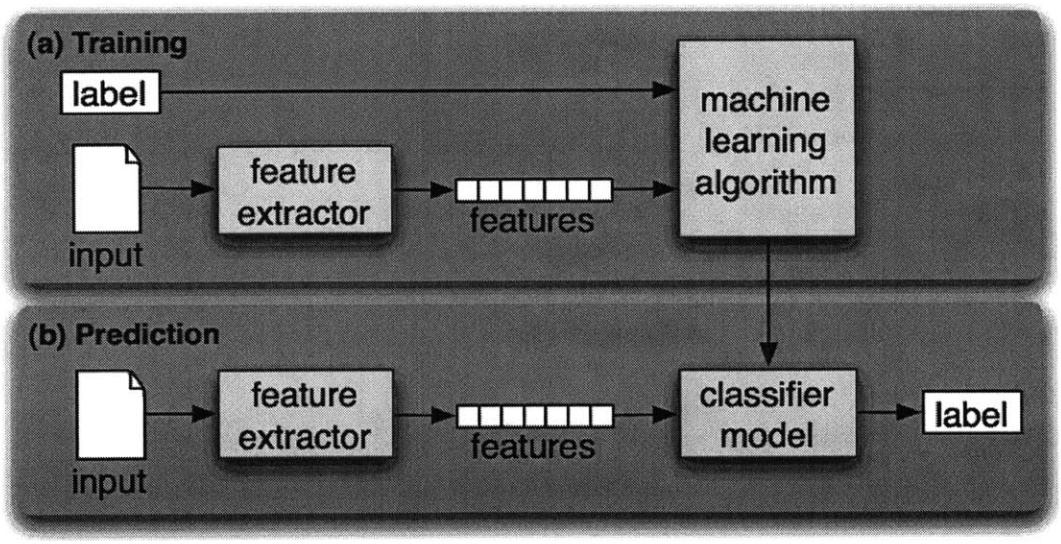

Figure 6-1: Supervised classification

For the given dataset, as shown in Figure 6-1, we divided it into two parts, training

set and test set:

- Training Set: 80% of the data, used to train and calibrate the data.
- Test (Prediction) Set: 20% of the data, the out of sample part, which is used specifically to test the performance of the calibrated model.

In other words, we shall use the training data to teach the machine different types of invoices, and then use the test day to simulate the new coming data.

6.1 Supervised Learning Algorithms

We applied following classification algorithms for this dataset:

- Classification tree[36]
- Random Forests[37]
- Adaptive boosting[38]
- Logistic regression[39]
- Support vector machine (SVM)[40]

6.2 Model Learning & Calibration

Learning, here and in other sections, were run using 10-fold cross validation. One round of 10-fold cross validation involves partitioning the training data into 10 complementary subsets, performing the analysis on 9 subset (called the training fold), and validating the analysis on the other subset (testing fold). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

We use the cross validation result to choose the right parameter for our machine learning model, before we test it in the testing set.

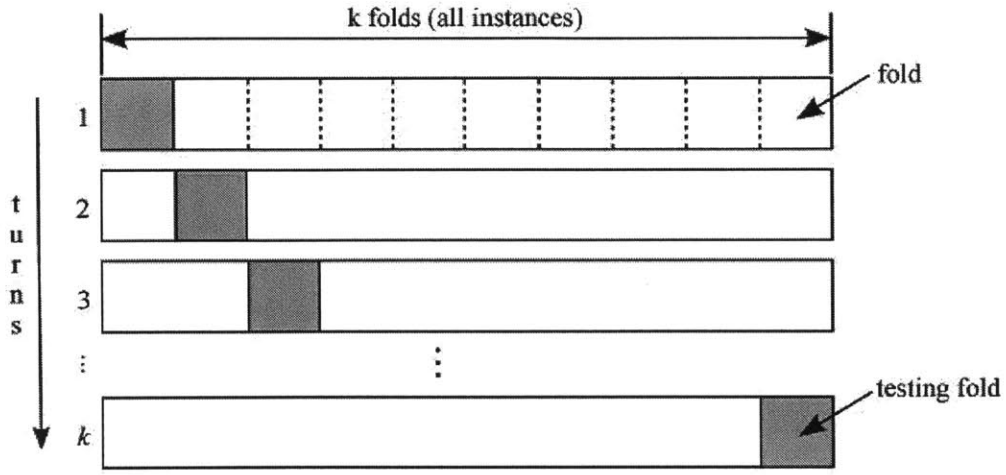

Figure 6-2: K-fold cross validation

Also, as a point of reference, we also report the accuracy of the majority-class predictor, i.e., a classifier that always predicts the class most represented in the training data. We refer to this as the Baseline.

6.3 Model Outputs

We now turn to the prediction result of various learning models.

6.3.1 Binary Outcome Case

The general results are shown in Table 6.1.

Model	Out of Sample Prediction Accuracy
Decision Tree	0.861
Random Forests	**0.892**
AdaBoost	0.863
Logistic Regression	0.864
Support Vector Machine (SVM)	0.869

Table 6.1: The prediction result of binary case with various machine learning algorithms

Decision tree We start the surprised learning with the most intuitive method - decision tree.

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value[36]. The general procedure of applying decision tree to make supervised classification includes two steps:

1. Grow the decision tree
2. Prune the grown decision tree

In the first step, we build a tree-style decision model with different complexity parameter (cp), which controls the size and levels of the decision tree. And then, we could the optimal cp by looking at Figure 6-3, which is generated by cross validation. It turns out, $cp = 0.016$ is the best decision tree complexity parameter for this problem.

We then prune the grown tree with $cp = 0.016$. And the pruned tree, also the one we used for prediction, is shown in Figure 6-4.

The detailed description of the decision process is:

```
1) root 60000 15685 1 (0.2614167 0.7385833)
  2) delay_ratio< 0.65379 20136  6701 0 (0.6672130 0.3327870)
    4) delay_ratio< 0.35322 11221  2126 0 (0.8105338 0.1894662) *
    5) delay_ratio>=0.35322 8915  4340 1 (0.4868200 0.5131800)
     10) delay_ratio< 0.50532 3911  1694 0 (0.5668627 0.4331373) *
     11) delay_ratio>=0.50532 5004  2123 1 (0.4242606 0.5757394) *
  3) delay_ratio>=0.65379 39864  2250 1 (0.0564419 0.9435581) *
```

Basically, it says, the decision tree is mostly using one feature of the invoice – the delay ratio (see in Sec 4.3.4).

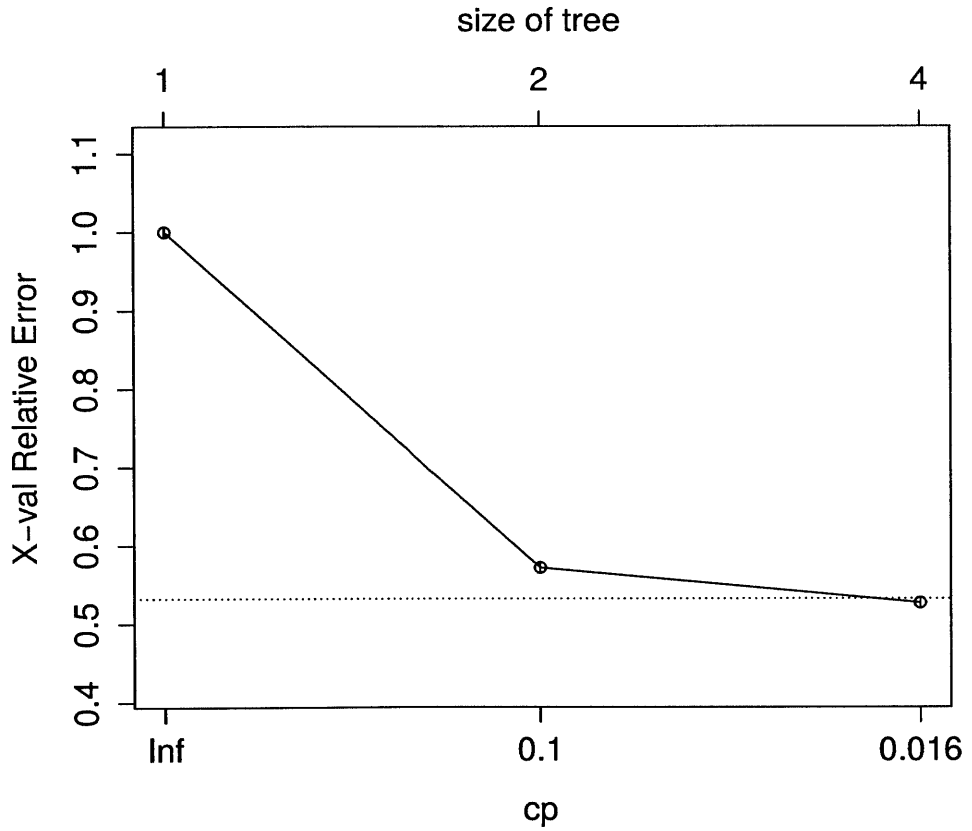

Figure 6-3: Choose the best decision tree complexity (cp) with 10-fold cross validation

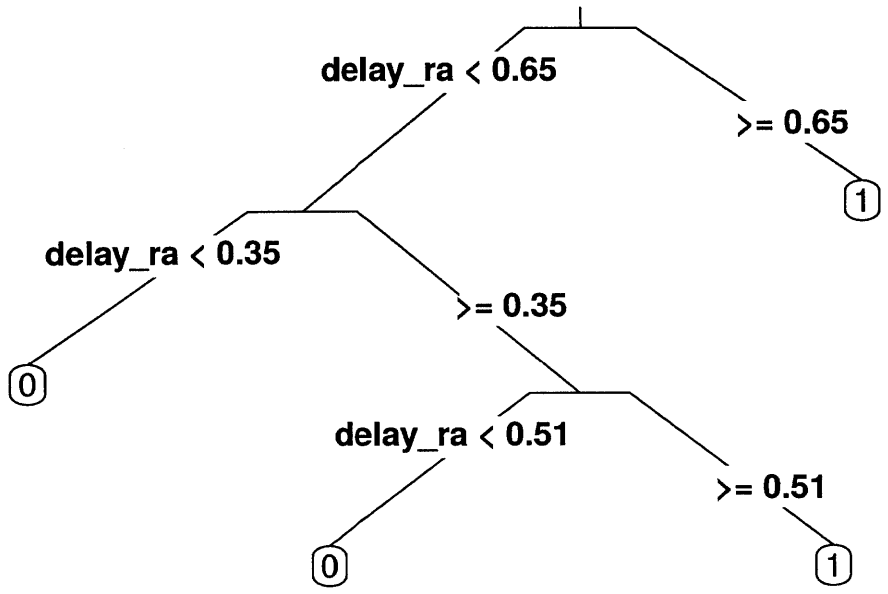

Figure 6-4: Decision tree demo on binary case

The prediction accuracy of decision tree is **0.861**. And the confusion matrix in the out of sample prediction is in Table 6.2.

	Actual no-delay	Actual delay
Predicted no-delay	1424	573
Predicted delay	463	5004

Table 6.2: Confusion matrix of decision tree in binary outcome case

And the most important three features in the decision tree models are:

1. Delay ratio of the customer

2. Average delay days of the customer

3. Number of total invoices of the customer

Random Forests Random Forests is an ensemble learning method based on decision trees. The idea behind is generate multiple decision trees and let them "vote" for the classification result [37]. In some degree, random Forests corrects for decision trees' habit of overfitting.

Therefore, one of the key parameter of random Forests algorithm is the number of decision trees to grow. Figure 6-5 shows that, the training error becomes very stable when the number of growing trees is more than 100 for the binary case.

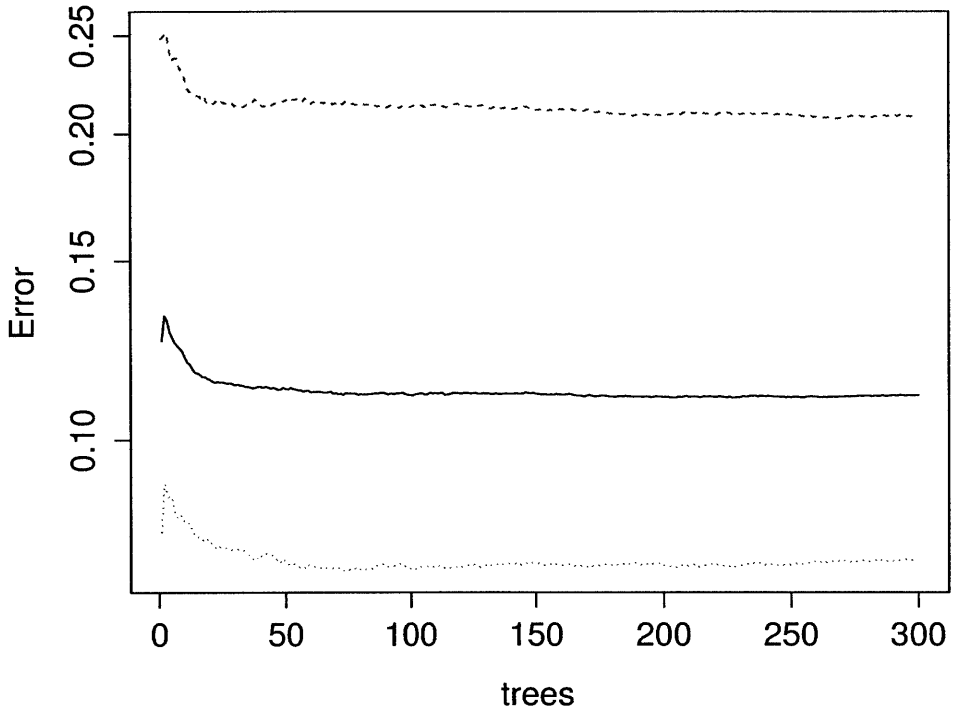

Figure 6-5: Training error versus number of tree growing in random Forests

We then apply the trained random Forests model into the out-of-sample test data. The prediction accuracy is **0.892**. And the confusion matrix in the out of sample pre-

diction is in Table 6.3.

	Actual no-delay	Actual delay
Predicted no-delay	1587	410
Predicted delay	398	5069

Table 6.3: Confusion matrix of decision tree in binary outcome case

Also, Random Forests gives us the ranking of feature importance, as shown in Figure 6-6. It shows that, the delay ratio and average delay days of the customer who the invoice belongs to are two most important features.

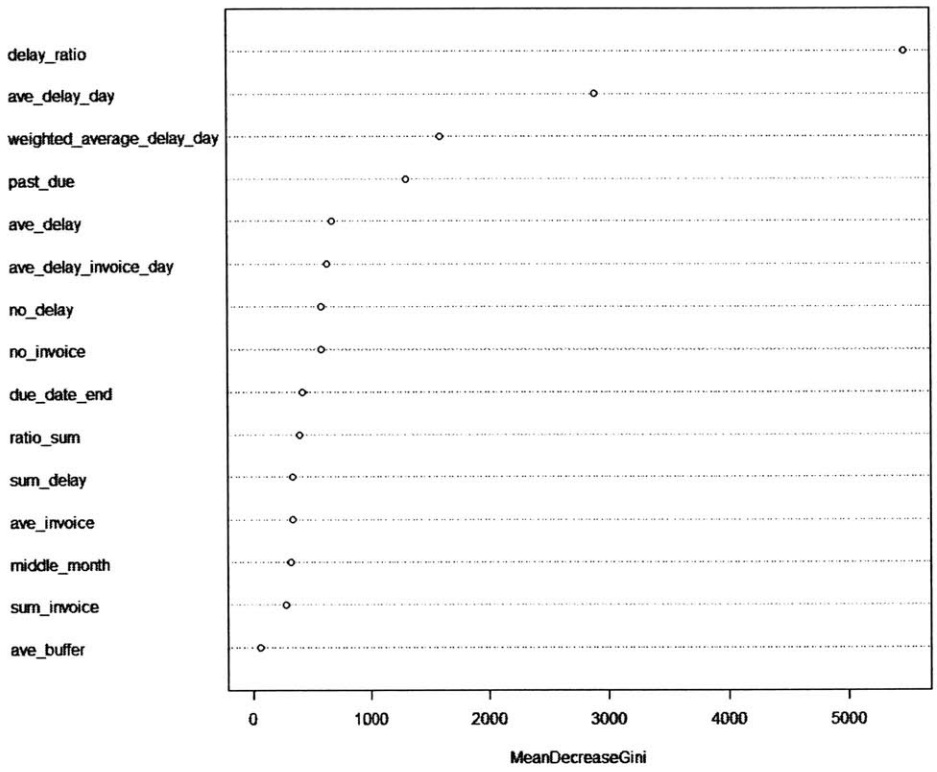

Figure 6-6: Variable importance random Forests in binary outcome case

AdaBoost AdaBoost is a method of ensemble learning. It combines the output of various algorithms (so called weak learners) into a weighted sum and give the

prediction[41].

We then apply the AdaBoost method into the out-of-sample test data. The prediction accuracy is **0.863**. And the confusion matrix in the out of sample prediction is in Table 6.4.

	Actual no-delay	**Actual delay**
Predicted no-delay	1588	617
Predicted delay	409	4850

Table 6.4: Confusion matrix of decision tree in binary outcome case

Also, Adaboost gives certainty of the predication on each invoice, which is called margin and calculated as the difference between the support of the correct class and the maximum support of an incorrect class[42]. The cumulative distribution of margins of predictions of both test and train data can be found in Figure 6-7. It shows that, the AdaBoost is quite certain for around 50% of the invoice predictions.

Logistic Regression For the binary outcome case, we could also use the classic binomial logistic regression method[43].

It shows that, the prediction accuracy is **0.864**, with the confusion matrix shown in Table 6.5.

	Actual no-delay	**Actual delay**
Predicted no-delay	1414	583
Predicted delay	430	5037

Table 6.5: Confusion matrix of logistic regression in binary outcome case

And the statistically significant features are shown in Table 6.6.

Support Vector Machine (SVM) Another natural choice of supervised learning algorithm is support vector machine (SVM), which actually turns the learning into an

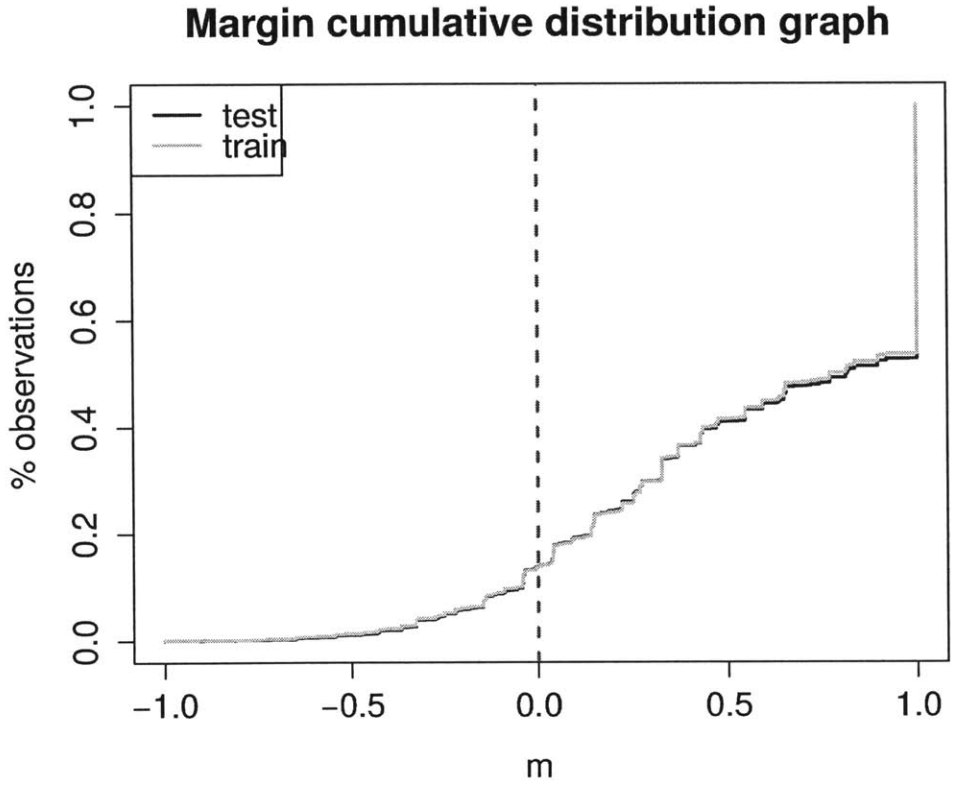

Figure 6-7: Margin (prediction certainty) cumulative distribution of the AdaBoost algorithm on invoice binary outcome prediction

optimization problem[44].

The prediction accuracy of SVM is **0.869**, with the confusion matrix shown in Table 6.7.

Feature	Explanation
due_date_end	Month end indicator
middle_month	Middle month indictor
no_invoice	Total number of invoice of the invoice owner
no_delay	Total number of delayed invoice of the invoice owner
ave_delay	Average delay of the delayed invoices of the invoice owner
ratio_no	Ratio of no_delay and no_invoice

Table 6.6: Statistically significant invoice features in logistic regression of binary outcome case

	Actual no-delay	**Actual delay**
Predicted no-delay	1398	599
Predicted delay	381	5086

Table 6.7: Confusion matrix of SVM in binary outcome case

6.3.2 Multiple Outcome Case

We then present the prediction result of multiple (four) outcome case, with same machine learning algorithms above.

The general results are shown in Table 6.8.

Model	Out of Sample Prediction Accuracy
Decision Tree	0.764
Random Forests	**0.816**
AdaBoost	0.770
Logistic Regression	0.755
Support Vector Machine (SVM)	0.773

Table 6.8: The prediction result of multiple outcome case with various machine learning algorithms

Decision tree Again, we grow the tree and prune the tree. However, the tree now need to make decisions on four outcomes: no-delay, short delay (within 30 days), medium delay (30-90 days) and long delay (more than 90 days).

As shown in Figure 6-8, the optimal $cp = 0.018$. We then use the optimal decision tree to predict the out-of-sample data, it turns out that the overall accuracy is **0.764**. How the decision tree makes classifications is shown in Figure 6-9.

The detailed confusion matrix is in Table 6.9.

	Actual no-delay	Actual short delay	Actual medium delay	Actual long delay
Predicted no-delay	1712	229	46	10
Predicted short delay	724	3378	286	8
Predicted medium delay	100	227	495	18
Predicted long delay	29	23	61	118

Table 6.9: Confusion matrix of decision tree in multiple outcome case

We can visualize how the decision tree works in the two most important features: delay ratio and average delay days. As shown in Figure 6-10, each dot in the figure is an invoice, whose color represents different outcome:

- Red: No-delay

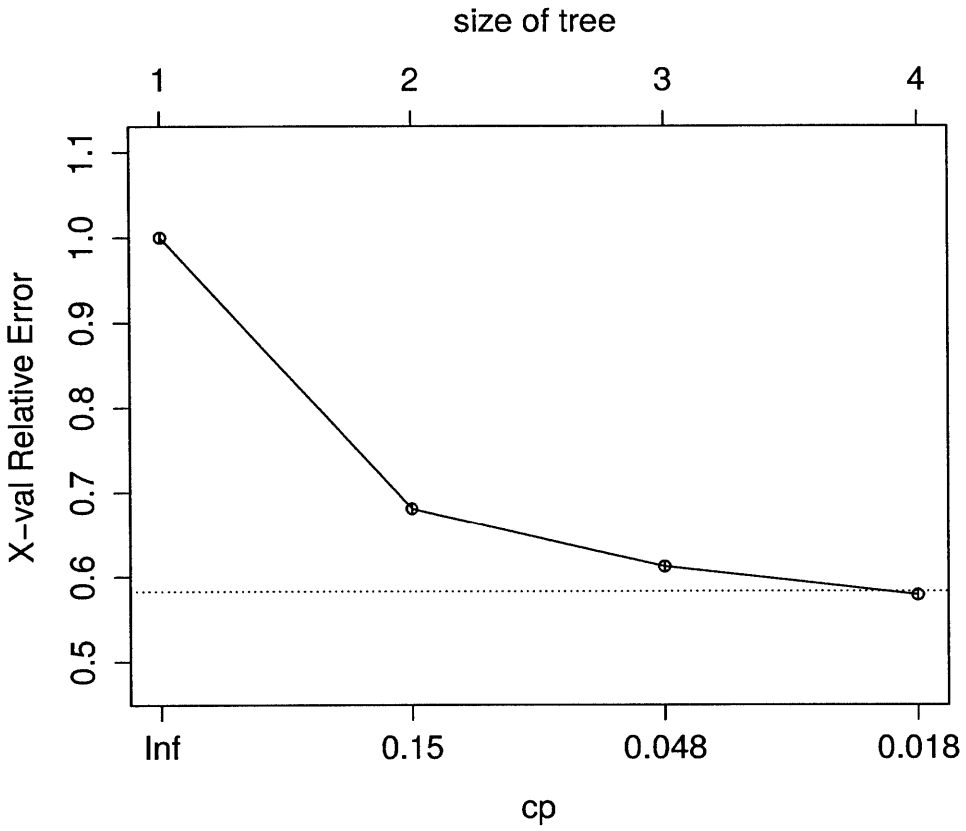

Figure 6-8: Choose the best decision tree complexity (cp) with 10-fold cross validation

- Green: Short delay
- Blue: Medium-delay
- Violet: Long-delay

Each invoice is attached to one customer, whose delay ratio and average delay days of historical invoices are known. If we take them as two axises and plot, it shows clearly the segmentation of invoices. And the black lines in Figure 6-10 is the segmentation thresholds generated by machine – the decision tree algorithm.

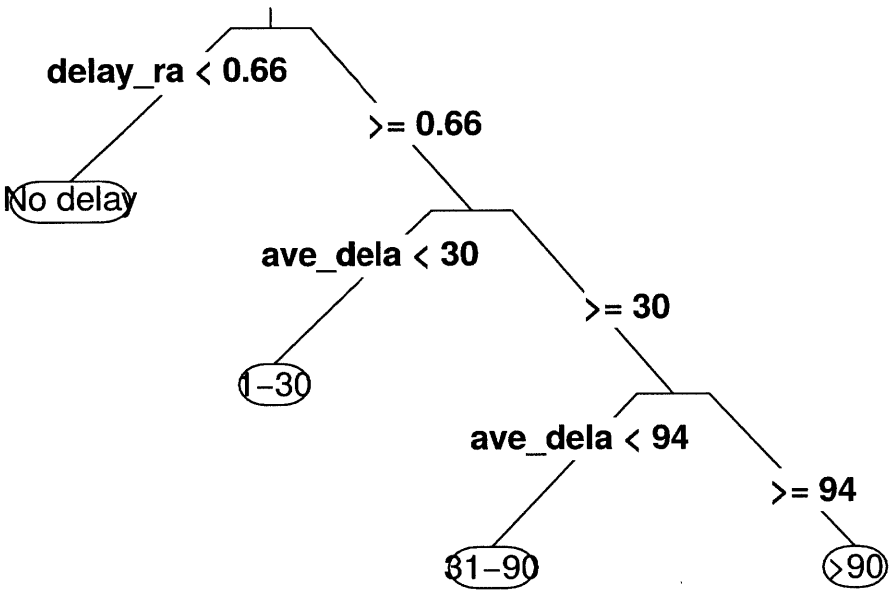

Figure 6-9: Decision tree algorithm demo on multi-outcome case

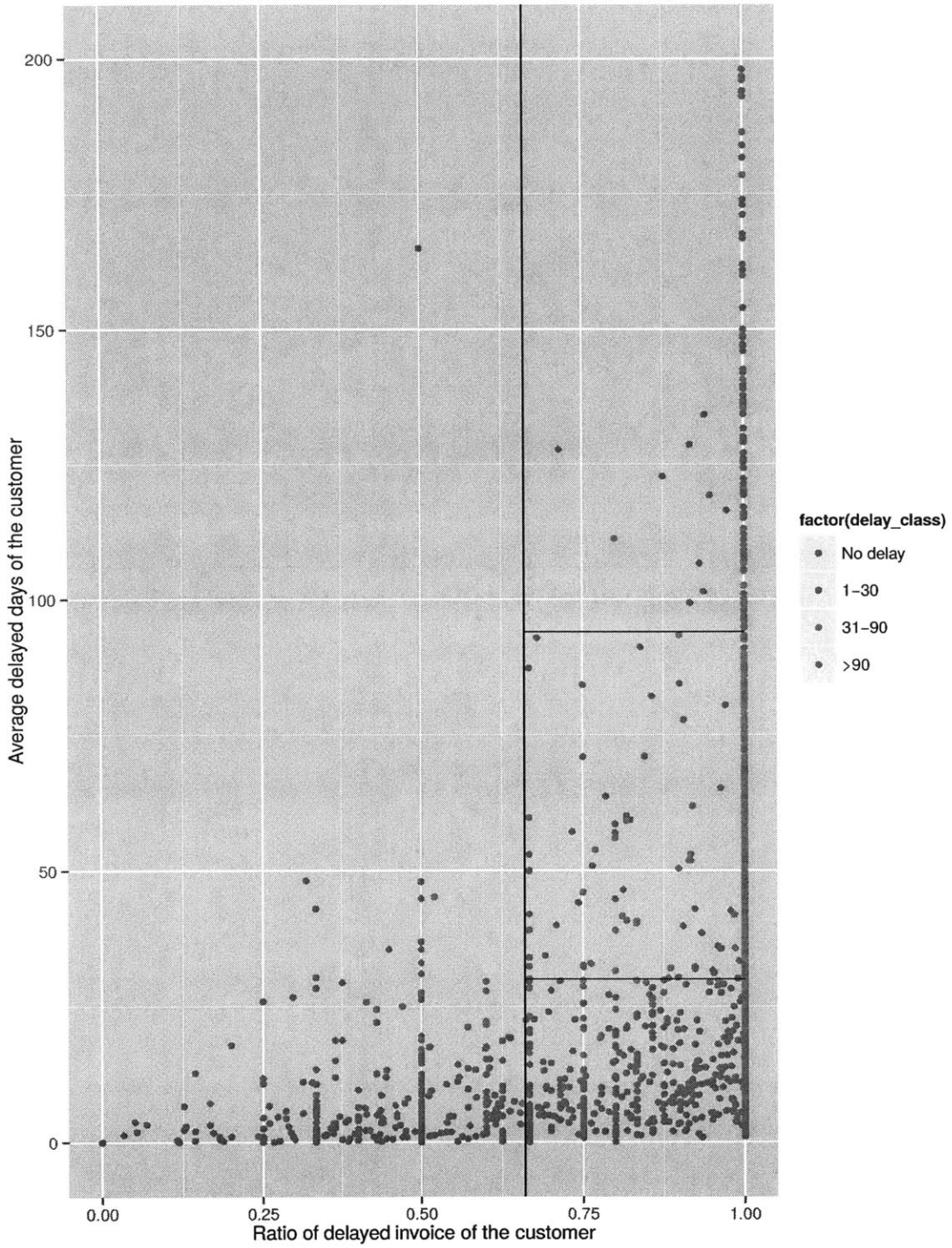

Figure 6-10: Visualization of decision tree in multiple outcome cases of invoice prediction

Random Forests We then apply the trained random Forests model into the out-of-sample test data. Again, it is found out that, the classification result becomes very stable when growing more than 100 decision trees. The prediction accuracy is **0.816**. And the confusion matrix in the out of sample prediction is in Table 6.10.

	Actual no-delay	Actual short delay	Actual medium delay	Actual long delay
Predicted no-delay	1625	350	17	5
Predicted short delay	367	3898	121	10
Predicted medium delay	64	305	449	22
Predicted long delay	18	44	49	120

Table 6.10: Confusion matrix of random Forests in multiple outcome case

Again, the ranking of feature importance, as shown in Figure 6-11. It shows that, the delay ratio and average delay days of the customer who the invoice belongs to are still two most important features.

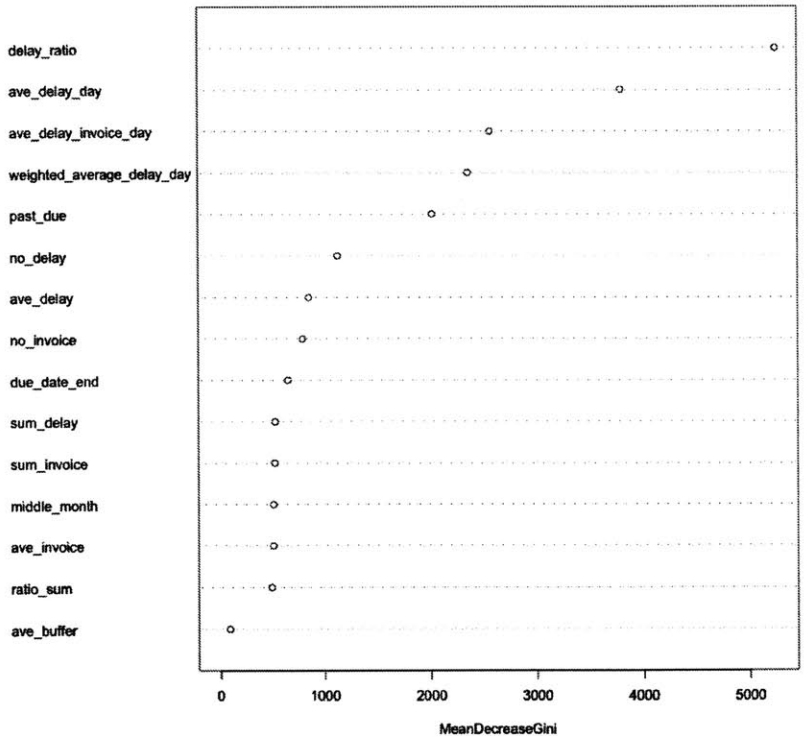

Figure 6-11: Variable importance random Forests in multiple outcome case

AdaBoost For AdaBoost, the prediction accuracy is **0.770**, with error details in Table 6.11. It doesn't work as well as Random Forests, although it is also one kind of ensemble learning methods.

	Actual no-delay	Actual short delay	Actual medium delay	Actual long delay
Predicted no-delay	1442	408	83	33
Predicted short delay	525	3824	374	36
Predicted medium delay	24	154	360	39
Predicted long delay	6	10	23	123

Table 6.11: Confusion matrix of SVM in multiple outcome case

We look at the certainty of the predication on each invoice again. The cumulative distribution of margins of predictions of both test and train data can be found in Figure 6-12. It shows that, the AdaBoost is quite certain for around 40% of the invoice predictions for multi-outcome case.

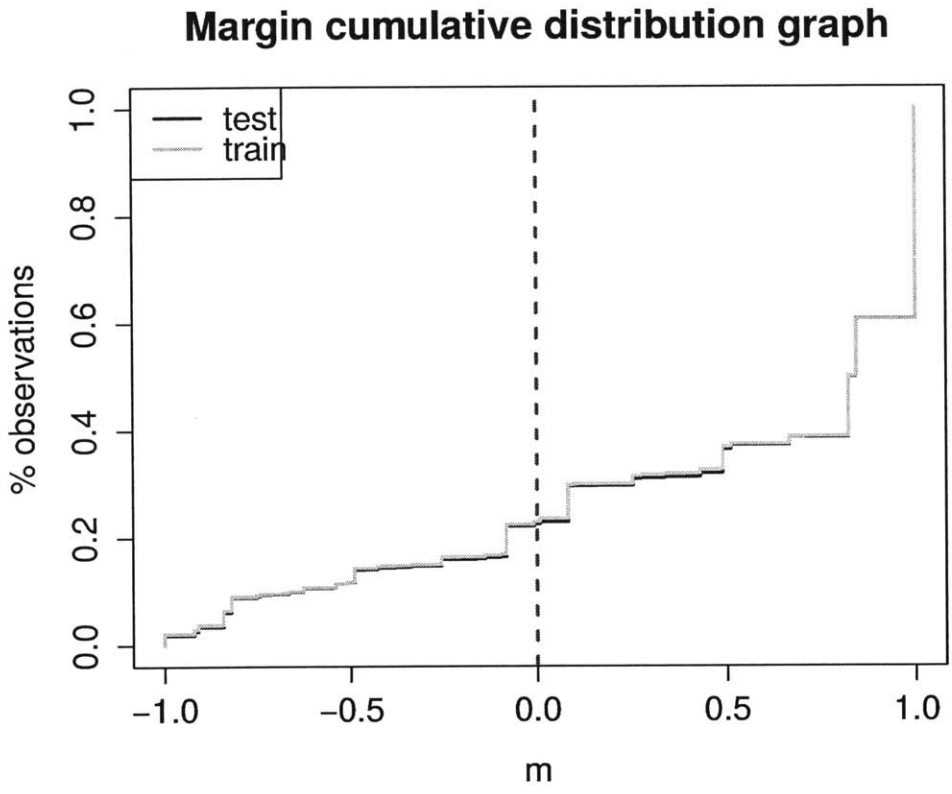

Figure 6-12: Margin (prediction certainty) cumulative distribution of the AdaBoost algorithm on invoice multi-outcome prediction

Logistic Regression We can also apply multinomial logistic regression on the multiple outcome case, which has the similar mathematical structure with the binary case.

It shows that, the prediction accuracy is **0.755**, with details in Table 6.12.

	Actual no-delay	Actual short delay	Actual medium delay	Actual long delay
Predicted no-delay	1383	577	9	28
Predicted short delay	340	3961	64	31
Predicted medium delay	61	551	204	24
Predicted long delay	13	54	73	91

Table 6.12: Confusion matrix of logistic regression in multiple outcome case

And the statistical significant variables are shown in Table 6.13, which do not differ with binary outcome case very much.

Feature	Explanation
due_date_end	Month end indicator
middle_month	Middle month indictor
no_invoice	Total number of invoice of the invoice owner
ave_delay	Average delay of the delayed invoices of the invoice owner
ratio_no	Ratio of no_delay and no_invoice

Table 6.13: Statistically significant invoice features in logistic regression of multiple outcome case

Support Vector Machine (SVM) The last method is again SVM. However, SVMs are inherently two-class classifiers. The usual way of doing multi-class classification with SVM in the practice has been to build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. In other words, it is building a voting mechanism for the classifiers[45].

The out-of-sample prediction accuracy of multi-class SVM is **0.773**. And the confusion matrix is in Table 6.14.

	Actual no-delay	Actual short delay	Actual medium delay	Actual long delay
Predicted no-delay	1521	464	5	7
Predicted short delay	425	3862	100	9
Predicted medium delay	76	471	275	48
Predicted long delay	17	54	48	112

Table 6.14: Confusion matrix of SVM in multiple outcome case

6.4 Results Analysis & Improvements

In this section, we analyze the result of our best preforming algorithm, Random Forests.

In both cases of binary outcome (Table 6.1) and multi-outcome (Table 6.8), we see quite consistent performances of various machine learning algorithms. And Random Forests has been the best algorithm in both.

If we compare the baseline and the performance of Random Forests, as shown in Figure 6.15, the general predictability is quite significant. However, we need to go to the confusion matrix of decision making to understand the algorithm performance better.

	Baseline	**Random Forests**
Binary Outcome Case	0.731	0.892
Multiple Outcome Case	0.598	0.81

Table 6.15: Measuring the performance of Random Forests

One of the key message of the confusion matrix is actually the Type 1 and Type 2 Errors[46], as demonstrated in Figure 6-13.

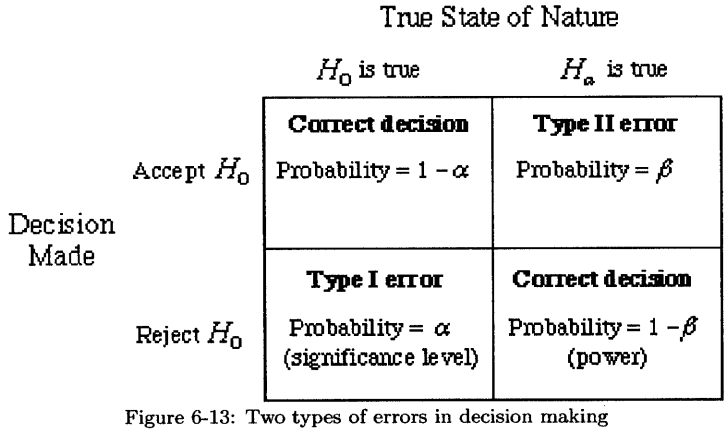

Figure 6-13: Two types of errors in decision making

In the binary outcome case of invoice prediction, Type 1 error is, given the invoice is going to be paid on time, the machine predicts it will be delay. And Type 2 error

is, given the invoice payment is going to delay, the prediction says no-delay.

Obviously, for the invoice collector, different types of errors weight differently. Usually, Type 2 error is much more "expensive" than Type 1 error. Expand the idea to multiple outcome case, it says the prediction accuracy on different classes of outcomes weights differently.

Therefore, we show the prediction accuracies of each class in both cases in Table 6.16 and Table 6.17. Table 6.16 basically tells us, given one invoice is going to be paid lately, our algorithm can detect it when issuing with around 93% accuracy.

	Prediction Accuracy
No-delay	0.794
Delay	0.927

Table 6.16: Class prediction accuracy of Random Forests in Binary Outcome Case

The Table 6.17 shows that, the algorithm (Random Forests) is quite good on detect the delay, especially the short delay. However, it has difficulties to detect the medium or long delays with very high accuracies.

	Prediction Accuracy
No-delay	0.814
Short delay	0.887
Medium delay	0.535
Long delay	0.519

Table 6.17: Class prediction accuracy of Random Forests in Multiple Outcome Case

We shall address this problem in multiple outcome case in the next section.

6.5 Imbalanced Data & Solution

One may notice from the previous section that, the prediction accuracy of our best algorithm, Random Forests, varies in different classes.

The main reason of this accuracy difference is because the data is imbalancec.There are different numbers of invoices in different classes. In other words, there are more invoice with outcome A than outcome B.

In the section, we try to address this problem in two ways. One is based sampling technique, the other one is based on cost sensitive learning. [47].

6.5.1 Weighted Sampling

As mentioned by Breiman[48], Random Forests grows its trees with a bootstrap sample of training data. However, in an imbalanced training set, there is high probability that the bootstrap sample containing few or even non of the minority class, resulting a tree with poor predicting capability of the minority class.

The intuitive way to fix this problem is with weighted sampling, which is also called stratified bootstrap[49]. That is saying, sample with replacement from within each class.

Therefore, we now sample each invoice outcome class with the weights (frequencies) proportional to their class size. It is also called Balanced Random Forests. The result is shown in Table 6.18.

The result shows that, by stratified sampling, one can significantly improve the prediction accuracy of the minority class (long delay invoices here). However, it is with the cost of the overall prediction accuracy.

	Random Forests	Balanced Random Forests
Overall	0.816	0.610
No-delay	0.814	0.758
Short delay	0.887	0.515
Medium delay	0.535	0.698
Long delay	0.519	0.8354

Table 6.18: A comparison of prediction accuracy of Random Forests and Balanced Random Forests

6.5.2 Cost-sensitive Learning

To address the different misclassification costs, we can also use instance re-weighting, which is a common approach in cost-sensitive learning.

Since the Random Forests tends to be biased towards the majority class, which is also the less important class[48], we can change the penalty function and place a heavier penalty on misclassifying the minority (and more important or expensive, like long delay invoices) class.

Therefore we assign a weight to each class, with the minority class given larger weight (i.e., higher misclassification cost), as shown in the Table 6.19

	Predicted no-delay	Predicted short delay	Predicted medium delay	Predicted long delay
Actual no-delay	0	1	1	1
Actual short delay	2	0	1	1
Actual medium delay	3	2	0	1
Actual long delay	4	3	2	0

Table 6.19: Misclassification cost matrix C for cost-sensitive Random Forests

Table 6.19 basically is a cost matrix, C. It tells us, given a invoice is actually in Class i, the penalty of classifying it into Class j is C_{ij}. For example, given a invoice has long delay, the penalty of classifying it into short delay is $C_{42} = 3$, which is larger than classifying into medium delay. $C_{43} = 2$.

The Random Forests algorithm in MATLAB, *TreeBagger*, can directly incorporate

this cost matrix C to the learning. The resulting prediction accuracy is shown in Table 6.20.

	Random Forests	Cost-sensitive Random Forests
Overall	0.816	0.812
No-delay	0.814	0.778
Short delay	0.887	0.889
Medium delay	0.535	0.600
Long delay	0.519	0.619

Table 6.20: A comparison of prediction accuracy of Random Forests and cost-sensitive Random Forests

One can see from Table 6.20 that, the overall accuracy of Cost-sensitive Random Forests is quite consistent with the original one. And we see the improvement on the accuracies on medium delay and long delay classes.

6.6 Model Robustness

We have also been able to test the performance of our Random Forests model in the new two more month data.

Each of the two month datasets has similar volume of invoices, around $70k$, and exactly same information for each invoice. Therefore, we are able to update our model parameters with new training data and test it again in the out-of-sample test data. The result is shown in Table 6.21.

	Original Dataset	New Month A	New Month B
Delay Detection Accuracy	0.927	0.914	0.909
Multi-Outcome Overall	0.816	0.794	0.803

Table 6.21: Model prediction accuracy in data of different donths

It shows that, the Random Forests model has a relatively consistent prediction accuracy in different datasets of various months.

6.7 Why Random Forests Works

So far, the algorithm Random Forests has outperformed other classifiers in the invoice prediction case. It agrees with the author's experience in machine learning – Random Forests is often the winner for lots of problems in classification (usually slightly ahead of SVMs), they're also fast and scalable.

In a paper of 2014, Fernandez-Delgado, Cernadas & Barro [50] evaluated 179 classifiers of machine learning in 121 data sets from UCI data base. They found out that the classifiers most likely to be the bests are the Random Forests versions. which achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. While the second best, the SVM with Gaussian kernel, was close, which achieved 92.3% of the maximum accuracy.

There is no definite answer why Random Forests works well widely in various cases. However, its good performance is believed to strongly associated with several of its features below[48]:

- Random Forests can handle thousands of input variables without variable deletion.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- Prototypes are computed that give information about the relation between the variables and the classification.

Chapter 7

Conclusion

7.1 Summary

This thesis discusses the analytics of business invoices and how to detect problematic payments and customers. Companies issued thousands of business invoices per month, but only a small proportion of them are paid in time. Machine learning is a natural fit to leverage the power of business analytics to understand the pattern of invoice and improve the account receivable collection by predicting the delay in advance.

In this thesis, I analyze the account receivable management (invoice collection) of a Fortune 500 company. It showed that a large proportion of the invoices issued are not paid in time. And we observe invoices with very different payments delays.

This thesis proposes a supervised (machine) learning approach to solve this problem and presented the corresponding results in invoice payment prediction. It shows that the machine learning algorithms can generate accurate prediction on the delay of the invoice payments, based the data historical invoices.

I also build a set of aggregated features of business invoices which capture the char-

acteristics of the invoice and the customer it belongs to. It shows, although no single feature of the invoice can reveal its payment outcome, the aggregate information is powerful in helping us understand the invoice outcomes. Having this set of features enhances the prediction accuracy significantly.

The algorithm Random Forests has been the best predictor in the invoice payment delay problem so far. More than that, the thesis demonstrates that by using cost-sensitive learning, we are able to improve prediction accuracy particularly for long delay invoices, which are the minorities in the data sets.

In addition, the thesis demonstrates the robustness of the machine learning model applied in this problem. It receives consistent prediction accuracies across various invoice data sets.

In general, this thesis does a comprehensive research on the invoice payment outcome prediction, from data processing to prediction model building and calibrating. It offers a framework to understand the business invoices and the customers they belong to. It also provides an actionable knowledge for the industry people to adopt.

7.2 Future work

Based on the framework built by this thesis, there are several directions one can go further on this topic.

Building a better training dataset

One way to improve the prediction accuracy of the machine learning model is to feed it better training dataset.

As for now, the training dataset is build with one-month historical invoice data. It is a relatively short period considering the data a company may have accumulate. If one can access the data with longer history, he or she may be able to improve the prediction accuracy significantly and find the seasonality pattern of the invoice payments.

Incorporating extra information of the customer, like its revenue and margin, is another way to improve the training set. It is always helpful to give the machine more information of the object, even it may seem to be not relevant at the first place.

Calibrating cost-sensitive learning

The thesis explores the idea of using cost-sensitive learning to predict the invoice payments. However, the misclassification error matrix, which is the core of this algorithm, worth more careful studying.

In this thesis, the misclassification error matrix, as defined in Table 6.19, gives a qualitative approximation of the real cost. For example, Table 6.19 says the cost of misclassifying long delay invoice into no-delay class is twice of the cost of misclassifying into medium delay class. This multiplier ($2\times$) is not necessarily true and may be improved if one can incorporate more business sense. One could even define a dynamic cost matrix, if better prediction accuracy can be achieved.

Prioritizing invoice collection

Based on the predictability we achieved, we can further explore algorithms based invoice grading to channelize work flow to maximize business value. That is saying, how to use prediction result to optimize collection performance.

Ideally, we would take actions on all the "bad" invoices, as their payments would

be late if we do nothing. However, there are always resource constraints that prevent taking actions on all. Therefore, it is of great interests to develop an algorithm to prioritize the invoices. Of course, the implicit assumption here is, taking an action on an invoice will reduce the time of delinquency.

A natural approach would be prioritize based on the prediction outcome, i.e. how bad the delay would be. However, it goes more complicated as we set the objective to be maximizing the revenue – we may need to weight the amount of the bill. Additionally, an action may have different delinquency reduction on different clients. We surely need models to quantify the result of collection actions. They are all worth modeling if we manage to increase the effectiveness of invoice collection.