Predicting Late Payments: A Study in Tenant Behavior Using Data Mining Techniques

by

Mark Gschwind
Independent Consultant
mark@gschwindconsulting.com

Journal of
Real Estate Portfolio Management

Contents

Volume 13, Number 3, 2007
July–September

A Publication of the American Real Estate Society

I. Abstract:
Today, mining customer data is commonplace for banks, credit card companies and insurance companies, to name a few. This study discusses using data mining techniques at a commercial real estate manager to predict a behavior of *its* customers - its tenants. Specifically, the probability that a commercial tenant will make a late payment in the near future is estimated. The study introduces the reader to the data mining process and uses some of the more prevalent techniques for this type of problem. The result is a model with a predictive ability that is considerably better than a dartboard approach, suggesting that data mining techniques can be used by other commercial real estate managers to better understand and predict this part of tenant behavior.

I. Introduction

Paying the rent on time is a basic obligation of tenants. To monitor the timeliness of payments, the commercial real estate manager in this study used an aged receivables report. Managers made phone calls and set up reserves based on the number of days late, whether the late payment was a rent payment, and other business rules. More proactive actions included having property managers talk with tenants and check the number of cars in the parking lot to get an idea of whether a lease might get renewed, or whether payment problems might be on the horizon. But there was no significant effort to improve cash flow by proactively predicting late payments using quantitative methods, and taking corrective actions before a late payment happened. This paper discusses how basic tenant information, accounts receivable data, and government-published data can be mined to provide reasonably accurate predictions of late payments.

II. Related Research

Stoesser and Hess (2000)[1] acknowledge that there is value in attracting and keeping tenants whose cash flow is steady, "A manager may, for example…improve the overall credit quality profile of investment properties, thereby reducing the risk of the expected cash flows." But proactively identifying tenants with a stable cash flow is not simple, as Rivetti and Worzala (2003) state, "Typical strategies such as using rating agencies and analyzing company financial statements do not work if ratings can be changed over night and the audited financial statements turn out to be fabricated."[2] Research around tenant quality appears to focus on preventing defaults as in studies by Ciochetti (2003)[3] and Eichenblatt and Yeoman (1999)[4]. Less research extends to predicting and preventing delinquencies.

III. Intro to the Data Mining Process

A brief discussion about the process of data mining provides a useful background for the study. Data mining "is the exploration and analysis of large quantities of data in order to discover meaningful patterns and rules... (Alternatively) data mining is a business process for maximizing the value of data collected by the business."[5] There are several data mining methodologies in the data mining community. The two leading ones are SEMMA (Sample, Explore, Modify, Model, Assess) promoted by the SAS Institute, and CRISP-DM (CRoss Industry Standard Process for Data Mining) which is promoted by SPSS, NCR and others. A notion these methodologies both recognize is that the process is not a linear one. That is to say, the process frequently involves

---

[1] Joel W. Stoesser and Robert C. Hess, Point of View Styles of Higher Return Strategies, *Journal of Real Estate Portfolio Management*, 2000, 6:4, 417-22.

[2] Daniel Rivetti and Elaine Worzala, An Investigation into the Credit Tenant Characteristics of Department of Defense Contractors, *Journal of Real Estate Portfolio Management*, 2003, 9:2, 179-85.

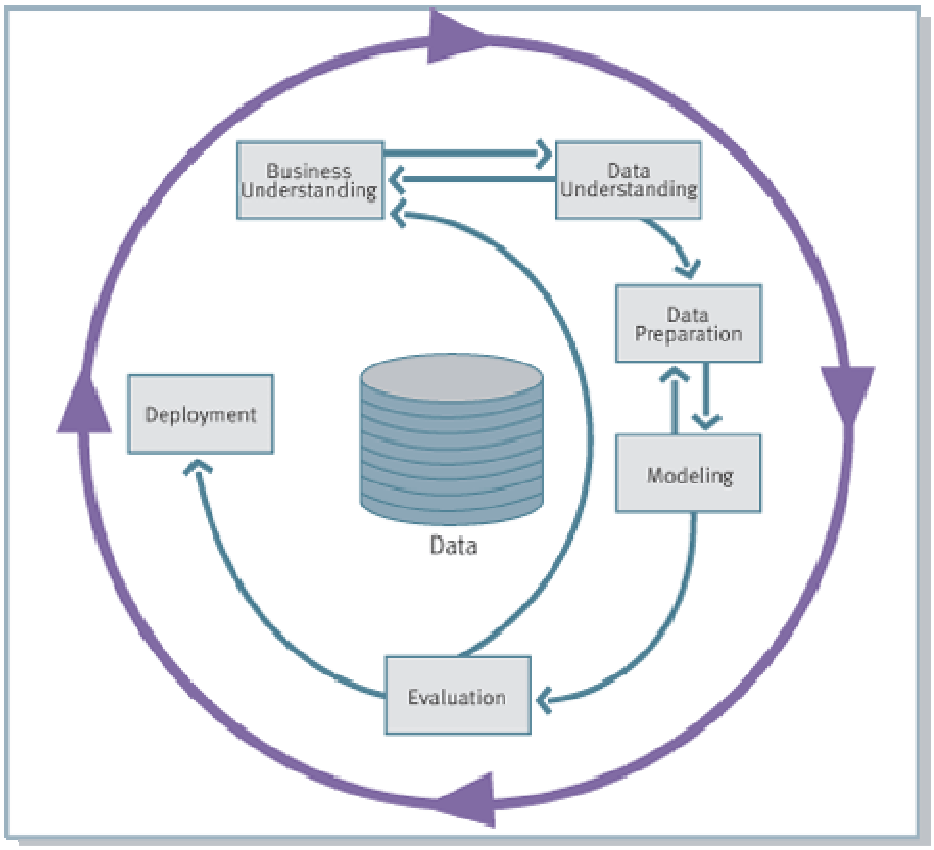
[3] Brian Ciochetti, Yongheng Deng, Gail Lee, James Shilling, Rui Yao , A Proportional Hazards Model of Commercial Mortgage Default with Originator Bias, *Journal of Real Estate Finance and Economics*, 2003, 27:1, 5-23.

[4] Eichenblatt, D. L. and J. C. Yeoman, Monte Carlo Simulation: Impact of Tenant Size and Credit Quality on Office Buildings, *Real Estate Review*, 1999, 29:3, 47-51.

[5] Michael J.A. Berry and Gordon S. Linoff, *Data Mining Techniques for Marketing, Sales, and Customer Relationship Management (Second Edition)*, Indianapolis: Wiley Publishing, Inc., 2004.

going back a step or two after discovering something in the data. The diagram of the CRISP-DM methodology below illustrates this:

Figure 1: Phases of the CRISP-DM Process Model
(source: The CRISP-DM Project)



Note that after modeling it is possible that discoveries in the data may necessitate going back a step and re-preparing the data. This was certainly the case during this study.

II. Defining the Data Mining Objective

In all data mining methodologies, choosing a measurable, actionable goal is an important first step. A goal like "understanding tenants better," would be neither measurable nor particularly actionable. The goal of this study is, "To predict the likelihood that an existing commercial tenant will have at least one late payment within the next six months." The author chose this goal because:

- More data, particularly accounts receivable history, was available for existing tenants. The lack of data on new tenants made including them too difficult.
- Several months would be required to take corrective action. Possible actions include phone calls, choosing not to renew with a tenant, or renewing with a risk premium added to the rent charge. Additionally, Fowler McNally et. al. (2001) [6] offer that, "LOCs

[6] Susan Fowler McNally, Carter H. Klein & Michael S. Abrams, How to structure a lease to protect against the risk of a bankruptcy of the tenant" *2002 Annual Survey of Letter of Credit Law & Practice*, Inst. of International Banking Law & Practice, Inc. and Real Estate Issues, Fall 2001

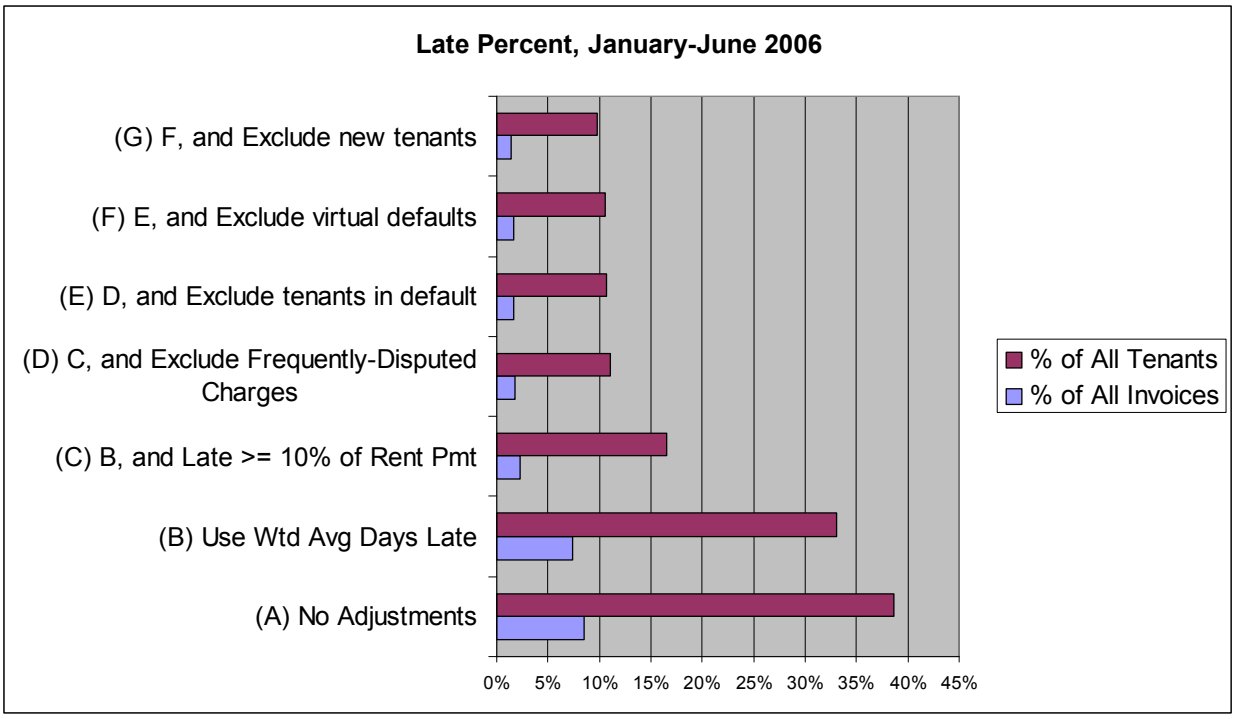
(letters of credit) will remain the security of choice for landlords leasing to tenants whose credit is deemed risky." The latter actions would require months of lead time, hence the six months.

- Commercial tenants (only) were chosen since this type of tenant presumably has different behaviors and behavioral drivers than other types of tenants.
- For purposes of doing a data mining study, the data available was well-suited to predicting late payments. Another notable tenant behavior worth studying, defaulting, is a relatively rare event. If there are not enough defaults, it is more difficult to model.

III. Refining the Data Mining Objective

After reviewing the data, it became apparent that the definition of "late" needed refinement. Figure 2 shows the adjustments made to this definition. The original definition (A in Figure 2), counted any payment over 60 days late as "late". This resulted in 38% of all tenants and 8.5% of all invoices being "late" in the next six months (from each evaluation date), which seemed unreasonably high. An initial exploration into the data found that many "lates" were due to

Figure 2: Adjustments to Definition of "Late"



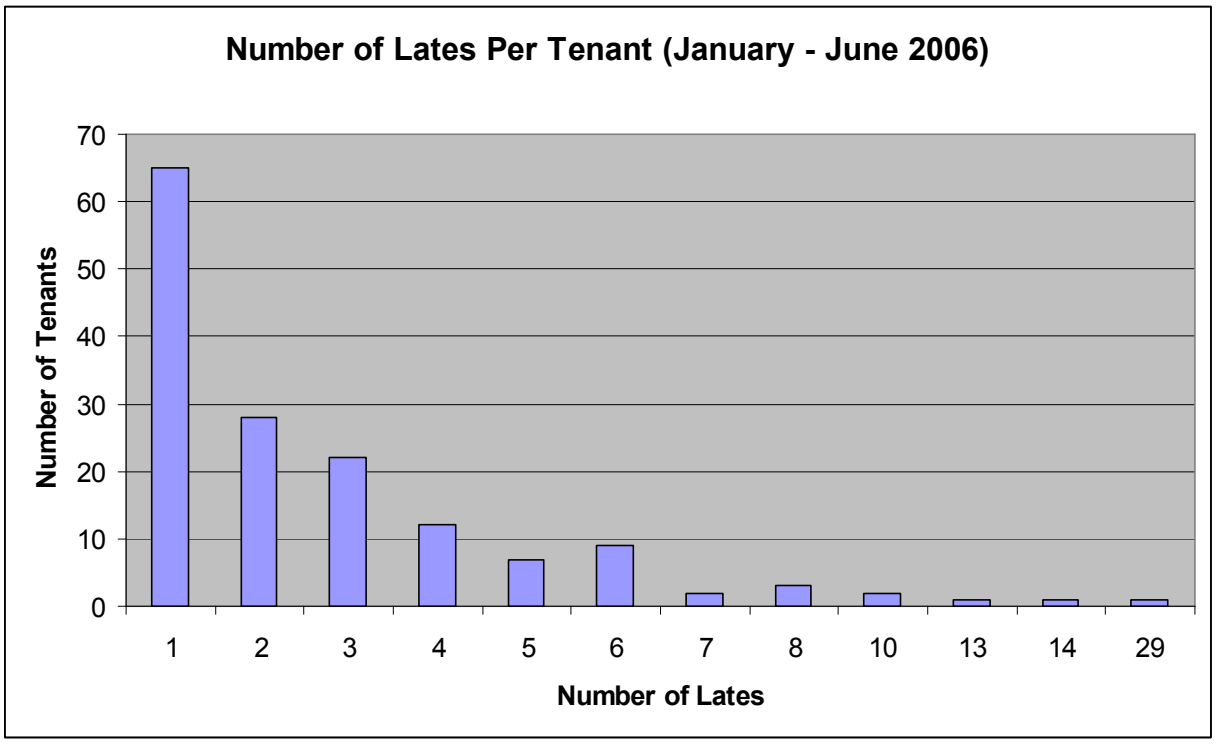
partial payments, where the bulk of the payment was on time. Further investigation revealed that these situations seemed to occur on the first invoice after a rent bump. Since this behavior appeared more an "honest mistake" than a "late", adjusting the days late to the weighted average days late better aligned the definition with the behavior under study (B in Figure 2). Still, there were "lates" on relatively small payments. These lates were far less of a hazard than a late on a monthly rent payment, and would probably not justify the cost of taking corrective action[a]. Excluding payments less than 10% of the monthly base rent focused in on behavior worthy of corrective action (C in Figure 2). Continued data profiling raised the suspicion that some "lates" were nothing more than accounting anomalies. The root of the suspicion was a late percentage that still seemed too high. More investigation found that tenants with otherwise spotless

payment records would have "lates" with only a few select charge types like year-end CAM true-ups, i.e. charges that probably required some negotiation. With no other way to determine a disputed charge, excluding these charge types was the only way to exclude "honest disputes" (D in Figure 2). Zeroing in on the target behavior more precisely would require a few more adjustments. For tenants already in default (E in Figure 2) or in "virtual default" (no payments for past 4 months) (F in Figure 2) it would be too late for corrective action. And finally, new tenants (G in Figure 2) had insufficient payment history to predict the target behavior, and needed to be excluded from the mining models[b]. More could be done to pinpoint the target behavior, but spot checks using these refinements consistently revealed the desired target behavior. It was time to set up an experimental design and prepare the input data.

IV. Sidebar to Figure 2

An interesting observation about Figure 2 is that 1.43% of all invoices were late while 9.76% of tenants had at least one late payment. The divergence of these percentages suggests that the 405 late payments were spread across many tenants, and not concentrated in a few chronic late-payers. This was the case as Figure 3 shows:

Figure 3



V. Data Preparation

V.1 The variables and their transformations

The primary data source for the study consisted of 9.5 million rows of accounts receivable data going back over 10 years. In addition there was a data warehouse with historic general ledger (GL) and leasing activity data. Basic attributes about the lease, tenant and building were available. External data was limited to unemployment rate data from the Bureau of Labor Statistics.

While reviewing the suitability of each variable as a model input, some opportunities for improvement became apparent:

- If more potential variables like SIC code had fewer missing values, more variables could be included in the mining model. This would allow for more hypothesis testing and potentially more accurate mining models.
- If historic values for attributes were captured, more variables could be included. For example, only the current building manager was stored rather than the building manager(s) during the time of each lease.
- If GL data was consistently captured at the tenant level, more hypothesis testing would be possible. For example, to test whether the amount of tenant improvements and their timing was a factor in causing a late, GL activity needed to be consistently linked to a tenant.
- As previously mentioned, if disputed charges were flagged, the target behavior would be more accurately identifiable.

Nevertheless, there were enough basic variables which could be transformed into new variables and input to the models. Figure 4 shows the final list of input variables.

Figure 4
Input Variables for Data Mining

Variable	Transforms Applied	Description
beta_slope_py		Slope of the PY days late time series using the beta algorithm***
beta_slope_times_stdev_py		An interactive term (explained later)
days_as_tenant		Number of days as a tenant - looking back at all leases
eval_yearmonth		For example 200601, one of the periods evaluated
hld_leases_exist_flag		Are any leases in "holdover" status
last_ontimeable_is_ontime		Y/N if the last payment was paid on time
amt_due_next6	Log transformed*	Amount of all scheduled charges due in the next 6 months
annual_rent_increase	Log transformed*	Dollar value increase in current annual rent
annual_rent_increase_pct_cap	Log transformed*	Pct increase in current annual rent, capped
beta_slope_py_cap	Log transformed*	Same as beta_slope_py, except the value is capped
curr_annual_rent	Log transformed*	Current annual rent for all leases
curr_annual_rent_psf	Log transformed*	Current annual rent per sq foot for all leases
days_since_last_late	Log transformed*	Number of days since the last late payment
ever_30_late	Log transformed*	Y/N if ever had a 30 day late payment - all history
ever_60_late	Log transformed*	Y/N if ever had a 60 day late payment - all history

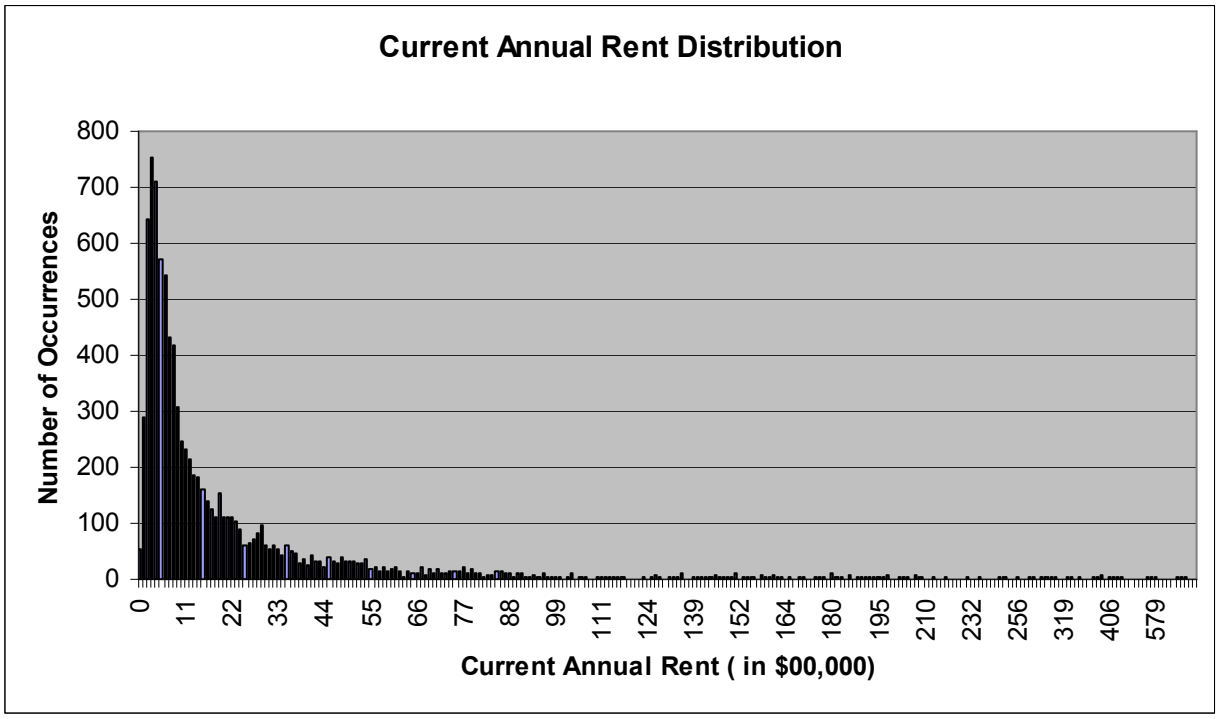
ever_90_late	Log transformed*	Y/N if ever had a 90 day late payment - all history
late_pct_ever_30	Log transformed*	Pct of payments 30 days late - all history
late_pct_ever_60	Log transformed*	Pct of payments 60 days late - all history
late_pct_ever_90	Log transformed*	Pct of payments 90 days late - all history
max_days_until_lease_end	Log transformed*	Of all leases the tenant has, the max days until lease end
mean_days_late_ever	Log transformed*	The mean of the days late for all payments
mean_days_late_py	Log transformed*	The mean days late for all payments in the PY
mean_days_late_py_20	Log transformed*	The mean days late of the most recent payments in the PY**
mean_days_late_py_8020trend	Log transformed*	The slope of the mean days late of the PY using an 80/20 method**
num_ever_30_lates	Log transformed*	Number of 30 day late payments - all history
num_ever_60_lates	Log transformed*	Number of 60 day late payments - all history
num_ever_90_lates	Log transformed*	Number of 90 day late payments - all history
num_lates_20	Log transformed*	Number of lates in the 20 portion of the 80/20 recency split**
num_ontimes_20	Log transformed*	Number of ontimes in the 20 portion of the 80/20 recency split**
osf	Log transformed*	The total occupied square feet for the tenant
pct_late_20	Log transformed*	The percent late in the 20 portion of the 80/20 recency split**
pct_late_8020trend	Log transformed*	The slope of the pct late of the PY using an 80/20 method**
stdev_days_late_ever	Log transformed*	The std deviation of the days late - all history
stdev_days_late_py	Log transformed*	The std deviation of the days late – PY
stdev_days_late_py_20	Log transformed*	The std deviation of the days late in the 20 portion of the 80/20 recency split**
stdev_days_late_py_8020trend	Log transformed*	The slope of the std deviation for the PY using an 80/20 method**
ttl_ar_reserve	Log transformed*	Total A/R reserve, using client's reserving methodology
ttl_bldg_py_repairs	Log transformed*	Total building repairs in $, for buildings the tenant occupies, for the PY
unpaid_amt_31_60	Log transformed*	Total 31-60 day aged receivables for the tenant
unpaid_amt_current	Log transformed*	Total <30 day aged receivables for the tenant
mean_days_late_py_8020trend		The slope of the mean days late for the PY using an 80/20 method**
mean_days_late_times_stdev_ever		An interactive term
mean_days_late_times_stdev_py		An interactive term
mo_chg_unempl_rate		Monthly change in the unemployment rate for the tenant's MSA
mtm_leases_exist_flag		Y/N if the tenant has month-to-month leases
new_tenant_flag		Y/N whether the tenant is new (on its first lease)
other_lates_in_bldg		Y/N whether there are other late-paying tenants in the building
pct_late_8020trend		The slope of the pct late for the PY using an 80/20 method**
priormo_unempl_rate		Prior month unemployment rate for the tenant's MSA
py_msa_mean_days_late		Mean days late for all tenants in the tenant's MSA
py_msa_pct_late		Pct late for all tenants in the tenant's MSA
py_msa_stdev_days_late		Std deviation of days late for all tenants in the tenant's MSA
beta_slope_py	squared	(see beta_slope_py)
mean_days_late_py_8020trend	squared	(see mean_days_late_py_8020trend)
num_ever_30_lates	squared	(see num_ever_30_lates)
num_ever_60_lates	squared	(see num_ever_60_lates)
num_ever_90_lates	squared	(see num_ever_90_lates)
pct_late_8020trend	squared	(see pct_late_8020trend)
stdev_days_late_ever	squared	(see stdev_days_late_ever)
stdev_days_late_py	squared	(see stdev_days_late_py)
stdev_days_late_py_8020trend	squared	(see stdev_days_late_py_8020trend)

stdev_days_late_py_8020trend		The slope of the std deviation of days late for the PY using an 80/20 method**
unpaid_31_60_div_rent		Total 31-60 day aged receivables for the tenant divided by the monthly rent
yr_chg_unempl_rate		Yearly change in the unemployment rate in the tenant's MSA

* The natural logarithm was applied to the original value, to adjust for a skewed distribution of values that would otherwise confuse the data mining algorithms.
** The 80/20 method stratifies the payment due dates of the prior year into the most recent 20% of dates and the remaining 80% of due dates. When these two groups of dates are compared, it gives more recent due dates a greater weight.
*** The beta function is a statistic that calculates the slope of a time series of values by determining the line of best fit.

Visually exploring the distributions of the underlying variables was the primary method in choosing the transforms listed in Figure 4. Many of them had skewed distributions like current annual rent (below):

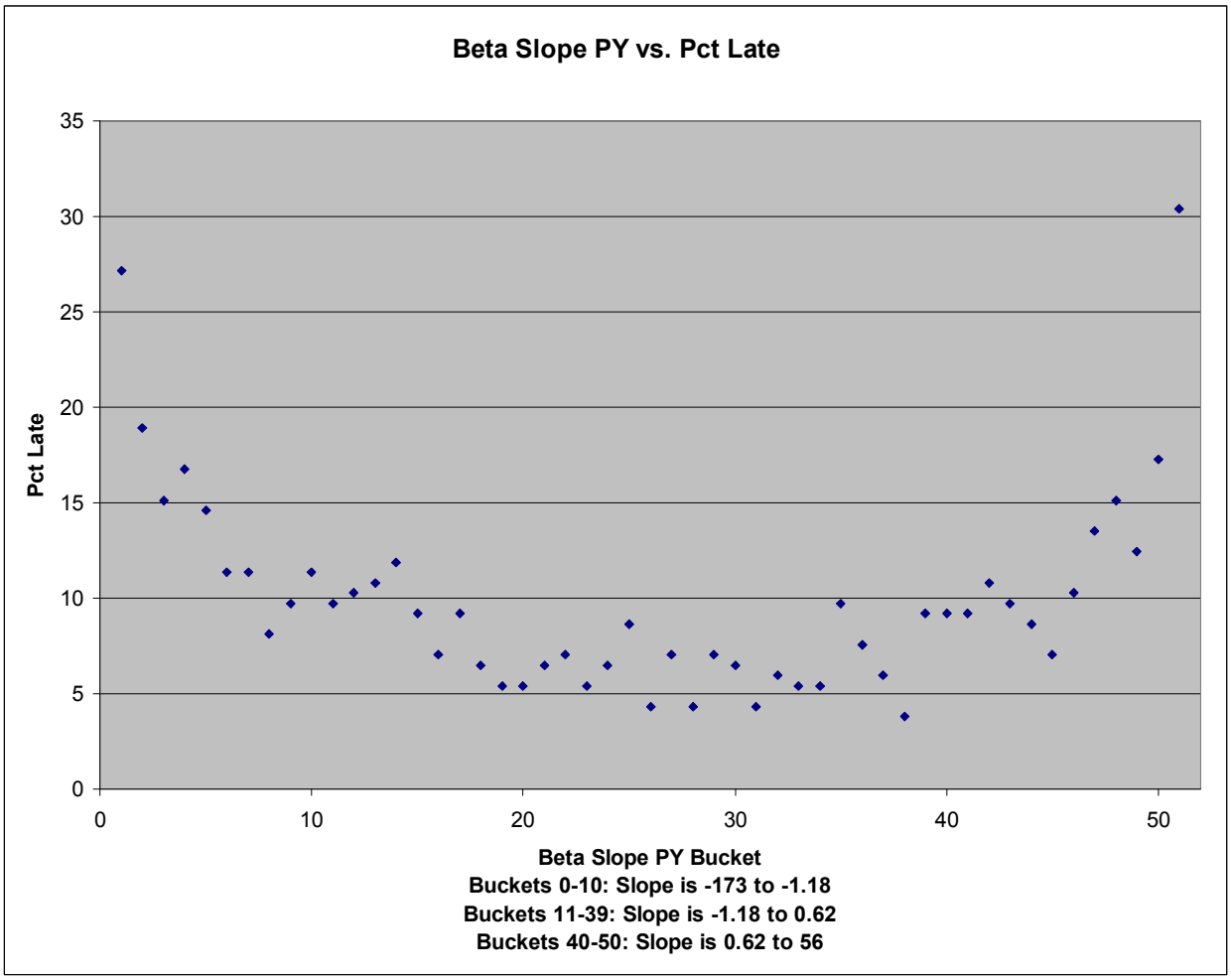
Figure 5



The median current annual rent is about $140,000 or 14 on the chart above, so clearly there is a skew to the right. When many data mining algorithms encounter a value that is out in the skewed side of a distribution, they will frequently consider the value to be "noise" and disregard it. Sometimes these values are outliers that *should* be ignored. However, my spot checks found that these values were correct, and frequently attached to lates. So instead of discarding them, the values were transformed by taking the natural logarithm. This removed some of the skewness and gave the algorithms an easier value to process[c].

Other variables displayed U-shaped (quadratic) distributions of the form of Ax^2+Bx+C. For example, the U-shaped distribution below shows that both extremely high and extremely low beta slope values are associated with high late percentages. Squaring the value of beta_slope_py and including the un-squared value captures this relationship by providing the algorithms with x and x^2, so they can calculate the values of A, B and C.

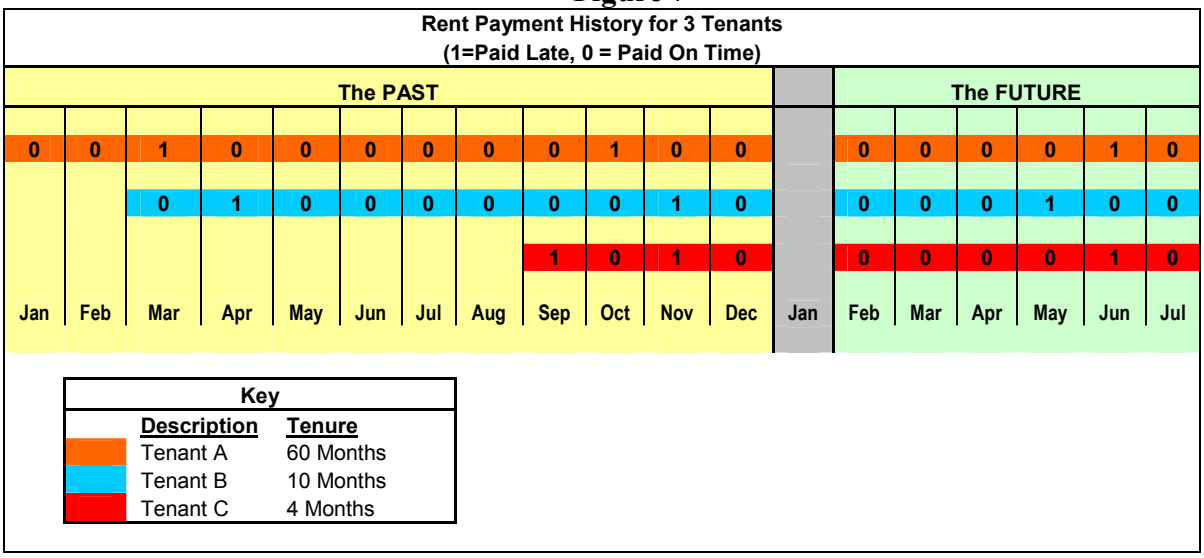
Figure 6



V.2 Capturing the Facts

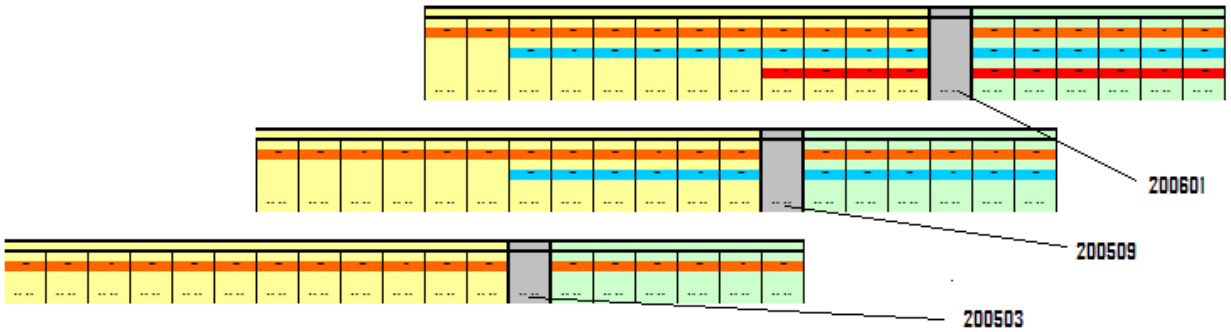
Having identified the input variables and their transformations, the next step was to devise an unbiased method to gather historical observations. Snapshots at 11 evaluation periods[d] were taken, as Figures 7 and 8 illustrate[e]. The 'Present' period (Jan in Figure 7) reflects the lag time to get data from prior periods, and the lag time of running the model before predicting a future period. The purpose of this lag period is to avoid biasing the model with data that would not be available at the time of the snapshot.

Figure 7



Notice in Figure 7 that all the tenants have lease payments for the next six months. Choosing only tenants with six months or more remaining on their lease was also necessary to avoid biasing the results. Figure 8 shows that the periods prior to an evaluation date sometimes overlapped, as the last three periods did:

Figure 8
Snapshots for Three Evaluation Periods



Allowing some overlap created more observations for the mining models, which is generally desirable. After collecting all this data using 3,200 lines of SQL code, 300 lines of SAS code and 3.5 hours of processing time, there were 9,259 observations, with a 9.52% target density (tenants with late payments as previously defined). Having identified and refined the objective,

then identifying the input variables and collecting observations, the hardest part of the process was over. Setting up the mining models was next.

VI. Building the Mining Models

The study used SAS Enterprise Miner (EM) to build mining models, pre-process model inputs and evaluate results. Three algorithms would compete to see which was best at predicting a late payment: logistic regression (logit model), decision trees and artificial neural networks (ANN). These algorithms are all suited to the task and can be found in other data mining tools. As the study found out later, determining the winner would not be straightforward. The algorithms output many statistics about goodness of fit, or the ability to predict that a tenant will have a late payment, but most of the statistics only work for one algorithm. But there is one statistic that is suitable for comparing the different models - lift. To explain lift in the context of this study, the calculation starts with the mining models calculating a probability that the tenants will have a late in the next 6 months. If tenants are then ordered into deciles based on their predicted probability of having a late, one would expect (if the mining model is doing a good job) that the top decile would have a higher proportion of observed late-payers than the general population (9.52%). The proportion in the top X deciles divided by 9.52% is considered the lift for the top X deciles[f].

The study took several precautions not to bias the results. Among them was evaluating model performance with data the models had not previously encountered. To accomplish this, the 9,259 observations were split into two sets: the training set and the validation set. Downstream from the split, the models get trained on the training set, which is a 70% random sample of the 9,259 observations then lift is calculated on the remaining 30% (the validation set). This way, lift is calculated on data that is previously "unseen" to the model. This arrangement is meant to mimic the model's performance when it is time to deploy using new data.

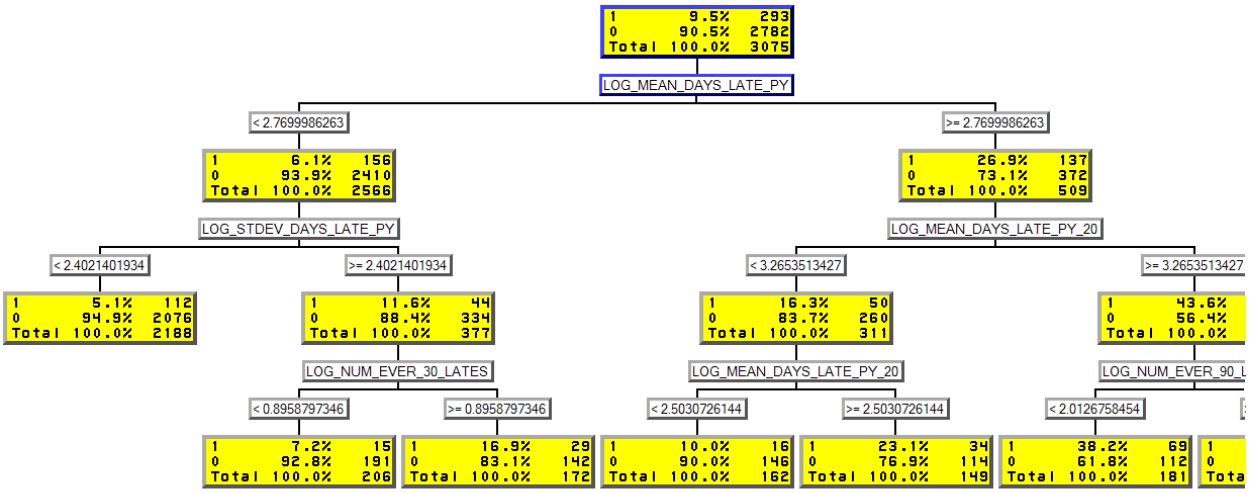
The first contestant was the logit model. The logit model is a widely used algorithm that is a close cousin to linear regression, except instead of predicting a continuous dependent variable it predicts a discreet or binary dependent variable[g] (in our case there are two discreet outcomes: the tenant had a late or the tenant didn't). Appendix A shows that of the many configurations tested to fine tune the model, the logit's best lift for the top decile of predicted late-payers, was 38.7%, i.e. the model improved the likelihood of identifying a late payer by a factor of 3.87 over a dartboard approach. This model found that the ten most predictive variables ranked in terms of their importance[h] were:

- log_num_ever_30_lates
- mean_days_late_times_stddev_py
- unpaid_31_60_div_rent
- log_unpaid_amt_current
- log_annual_rent_increase
- log_pct_late_20
- log_max_days_until_lease_end
- log_mean_days_late_py_20
- py_msa_stddev_days_late
- new_tenant_flag

One thing data miners always look for in their predictive models (in addition to lift) is results that make intuitive sense. One example in the logit's result was that the interaction between (multiplication of) the standard deviation of days late and mean days late was more significant than the two variables alone. This makes sense since a tenant with a high mean days late but low variability in the days late *should* be less likely to have a late than a tenant with a slightly lower mean days late, but a higher variability in the number of days late. Logit models have some advantages compared to the other models: the models are easy to explain, as each variable's importance is easily explained and the models tend to be stable.[i]

The next contestant was the decision tree. An easy way to introduce a decision tree is to show what one looks like. Below is a portion of the winning tree (to see all the tree configurations tested, refer to Appendix B).

Figure 9. Top Nodes of Decision Tree



The root node of the tree is showing 3,075 observations, which represents the validation data set.[j] From this set, the tree uses an algorithm to identify a variable that splits the set into groups of timely payers and late payers. Figure 9 shows the first split (top node) is made on the variable log_mean_days_late_py. If that value is >= 2.77 tenants go into one group and if it is less, tenants go into another. This brings the percent of late payers up to 26.9% in the >= 2.77 group, quite an improvement versus the 9.5% overall rate. Appendix B shows that of the many configurations tested to fine tune the model, the best lift for the top decile of predicted late-payers was 45.66%. ie the model improved the likelihood of identifying a late payer by a factor of 4.56 over a dartboard approach. This model found that the ten most predictive variables ranked by importance[k] were:

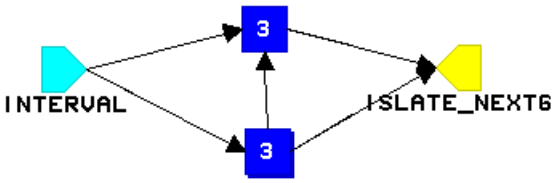
- log_mean_days_late_py
- log_mean_days_late_py_20
- log_num_ever_90_lates
- log_unpaid_amt_current
- log_ttl_bldg_py_repairs
- log_beta_slope_py_cap
- log_pct_late_20

- log_osf
- mo_chg_unempl_rate
- sq_pct_late_8020trend

One of the advantages of decision trees is that they visualize well, making them easy to explain. They are also useful for many applications besides prediction. However, a weakness of these models is that they can be unstable, meaning they may produce a different tree when retrained with new data.

The final contestant was the artificial neural network. Neural networks are a powerful but poorly understood class of models originally meant to simulate the workings of a biological brain. The winning neural network in Figure 10 below somewhat resembles a network of neurons:

Figure 10. Artificial Neural Network

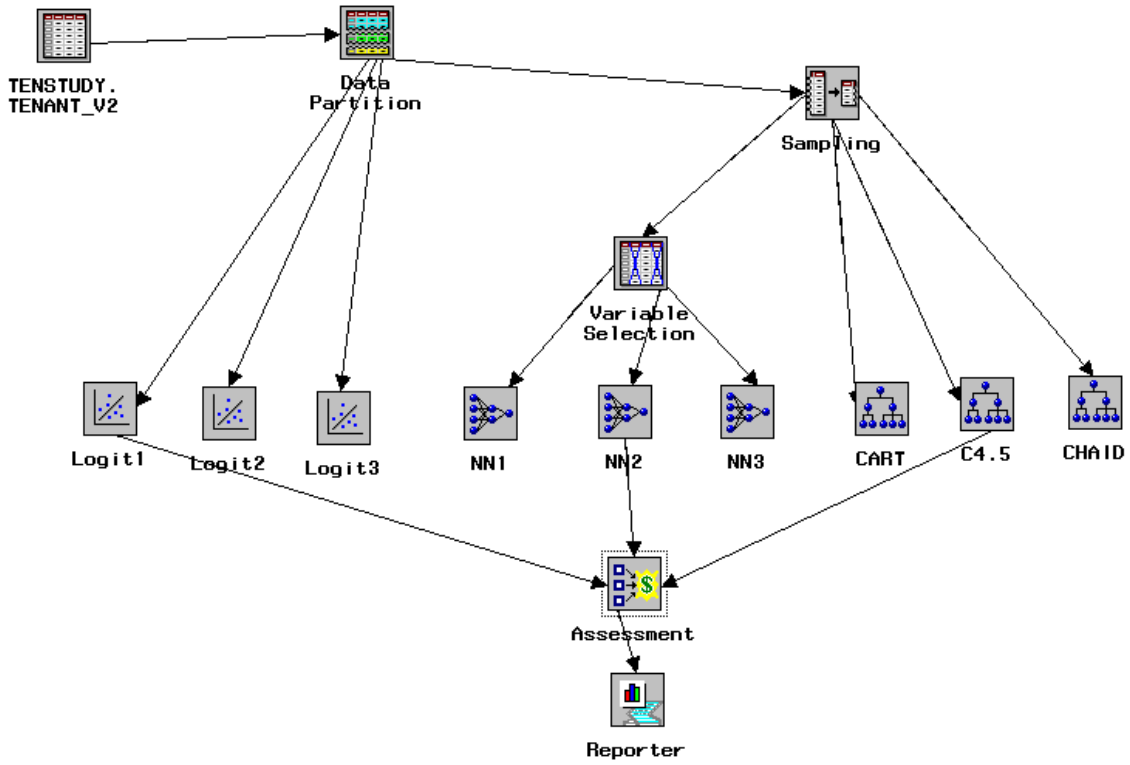


The light blue node is an input node, representing the input variables. This feeds two 'hidden' nodes that process the inputs with an algorithm[l], which in turn feed the output node. One commonality of these models is something called *back propagation* of the error. After the network determines the significant variables and calculates their weights, it compares the calculated results with the actual results, and sends the error back through the network (in an attempt to "learn" from its mistakes, in a sense). Neural networks can generate highly accurate predictions while handling noise in the data, but their difficulty is in understanding them - for most, they are a black box. Simply identifying the coefficients of variables, and the relative significance of variables is difficult. This can be a serious downfall, since users of a model must attain a certain level of comfort with a model before adopting it. As it turned out, the neural network produced the most accurate predictions in this study. Appendix C shows that of the many configurations tested to fine tune the model, the best lift for the top decile of predicted late-payers was 45.13%, with 60.98% captured in the first two deciles. This model found that nine variables had predictive value:

- log_max_days_until_lease_end
- log_mean_days_late_py_20
- log_num_ever_30_lates
- log_pct_late_20
- log_unpaid_amt_current
- mean_days_late_times_stdev_py
- py_msa_pct_late
- stdev_days_late_py_8020trend
- unpaid_31_60_div_rent

Below is the schema in Enterprise Miner used to create and compare the mining models.

Figure 11: Enterprise Miner Schema



A schema is made up of nodes that perform tasks. The Data Partition node (upper left), is where data is split into a training set and validation set. Above the Assessment Node are the Logit models, Artificial Neural Networks and Decision Trees. The schema shows that the decision trees' and neural networks' input data was oversampled[m] which the algorithms seemed to prefer. It also shows a Variable Selection node used to prune down the list of variables before input to the neural networks. Both these steps are merely a way of helping the algorithms do a better job training.

VII. Sidebar to Selected Variables

A second glance at the significant variables reveals some patterns. Figure 12 below shows where the models agreed:

Figure 12
Comparison of Significant Variables

Logistic Regression	Decision Tree	Neural Network
log_num_ever_30_lates	log_mean_days_late_py	log_max_days_until_lease_end
mean_days_late_times_stddev_py	log_mean_days_late_py_20	log_mean_days_late_py_20
unpaid_31_60_div_rent	log_num_ever_90_lates	log_num_ever_30_lates
log_unpaid_amt_current	log_unpaid_amt_current	log_pct_late_20
log_annual_rent_increase	log_ttl_bldg_py_repairs	log_unpaid_amt_current
log_pct_late_20	log_beta_slope_py_cap	mean_days_late_times_stdev_py
log_max_days_until_lease_end	log_pct_late_20	py_msa_pct_late
log_mean_days_late_py_20	log_osf	stdev_days_late_py_8020trend
py_msa_stddev_days_late	mo_chg_unempl_rate	unpaid_31_60_div_rent
new_tenant_flag	sq_pct_late_8020trend	

The green cells are variables that all models agreed upon, while the yellow and green are variables that the Logistic Regression model and Neural Network agreed upon. As the chart reveals, there was quite a bit of agreement between these two models. This does not come as a complete surprise, since the logit model can be considered a simple form of a neural network. Furthermore, if a variable is truly predictive, there should be some agreement between the models.

VIII. Evaluating Model Performance

Figure 13 below provides a graphic comparison of how well each model performed:

Figure 13: Cumulate Captured Response (Lift) for Competing Models vs. Baseline

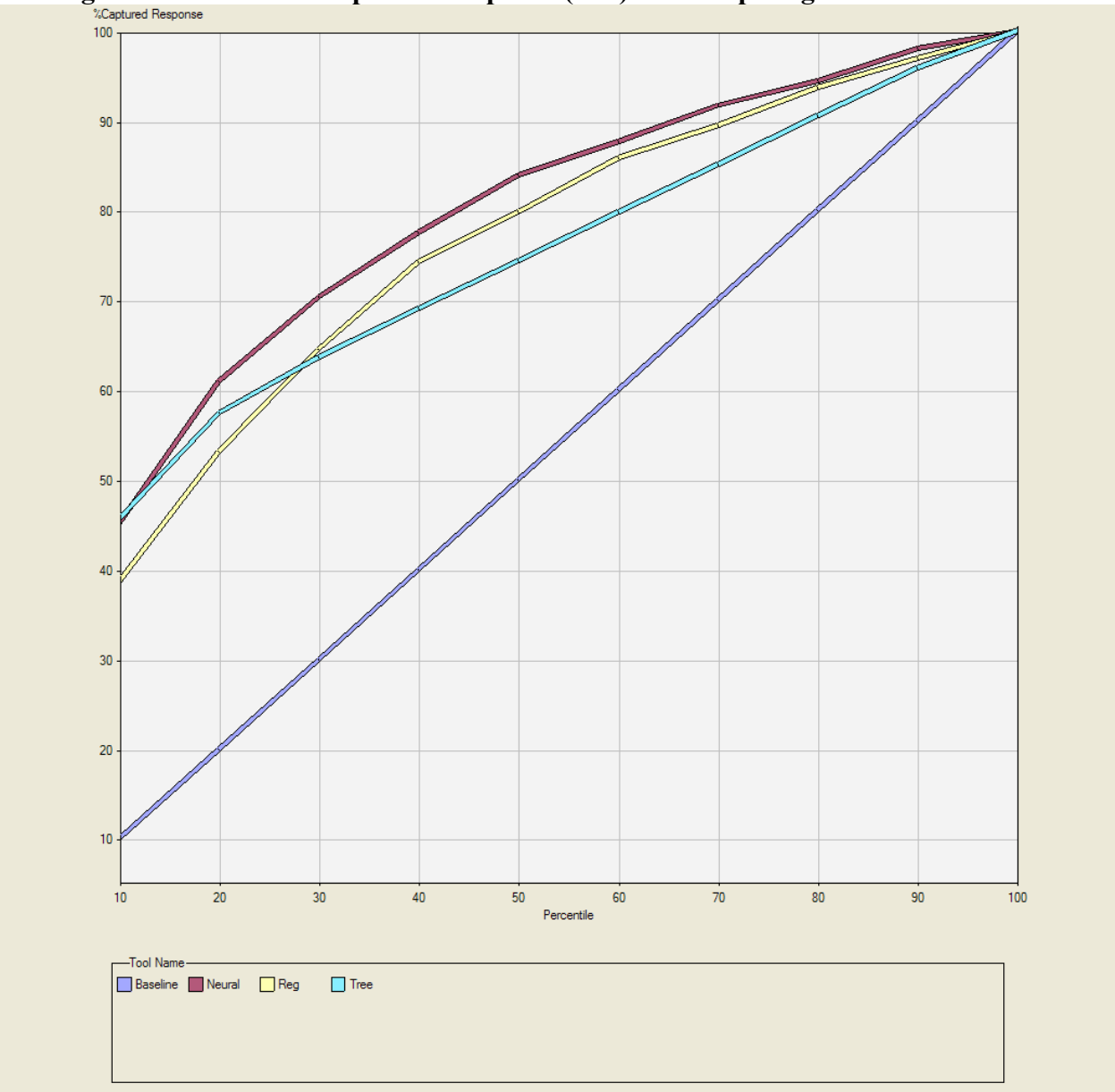


Figure 13 compares the cumulative captured response (lift) of the best configuration of the three competing models. In general, they all did a good job at predicting - they were all considerably better than the dartboard (baseline) approach. The neural network and the decision tree did an equally good job with the top 10% of tenants they were most certain about. But the neural network did a better job at guessing right on the remaining tenants.

So the winner based on lift is the neural network. However that may not be the right model for all real estate managers. Before a model is implemented, users should be comfortable with the features of the model such as stability and understandability. Considering this, if the neural network is not chosen, the winner based on lift is not clear-cut. If a real estate manager wants to take action on the top decile of tenants, the decision tree is the winner, but if the manager wants to take action on the top 40%, the logit model has the best lift. These issues would need to be discussed by each organization before choosing a model to implement.

IX. Conclusion

In conclusion, this study has demonstrated that mining basic tenant data, accounts receivable data and government-published data at a commercial real estate manager can generate predictions of late payments by tenants that are significantly better than a dartboard approach. Even better accuracy would be possible with improvements in the way tenant-level data is collected and maintained. Further study into the drivers behind the behavior under study would surely yield better explanations and predictions. But the strength of the predictions made by this study with ordinary input data suggests that real estate managers have sufficient data for making this and other useful predictions about their tenants. Real estate managers who recognize these opportunities and take advantage of them will undoubtedly find an advantage over their peers. In the future, the quality of such predictions and the ability to translate them into profit will distinguish successful real estate managers.

XI. Appendices

Appendix A - Logistic Regression Testing Results

Seq	Prior Probability	Optimization Method*	Method	Criteria	Entry	Stay	Cumul Captured Response Rate % @10%	Cumul Captured Response Rate % @20%	Cumul Captured Response Rate % @40%	Root ASE on Validation Set
1	(model - 9.52%)	1	Stepwise	Validation Misclass.	0.3	0.2	38.699	53.171	74.309	0.27856
2	(model - 9.52%)	2	Stepwise	Validation Misclass.	0.3	0.2	38.699	53.171	74.309	0.27856
3	(model - 9.52%)	3	Stepwise	Validation Misclass.	0.3	0.2	38.699	53.171	74.309	0.27856
4	(model - 9.52%)	1	Stepwise	Validation Error	0.3	0.2	36.651	53.008	73.984	0.27675
5	(model - 9.52%)	2	Stepwise	Validation Error	0.3	0.2	36.651	53.008	73.984	0.27675
6	(model - 9.52%)	3	Stepwise	Validation Error	0.3	0.2	36.651	53.008	73.984	0.27675
7	(model - 9.52%)	1	Forward	Validation Misclass.	0.2	n/a	38.699	53.171	74.309	0.27856
8	(model - 9.52%)	2	Forward	Validation Misclass.	0.2	n/a	38.699	53.171	74.309	0.27856
9	(model - 9.52%)	3	Forward	Validation Misclass.	0.2	n/a	38.699	53.171	74.309	0.27856
10	(model - 9.52%)	1	Forward	Validation Error	0.2	n/a	36.651	53.008	73.984	0.27675
11	(model - 9.52%)	2	Forward	Validation Error	0.2	n/a	36.651	53.008	73.984	0.27675
12	(model - 9.52%)	3	Forward	Validation Error	0.2	n/a	36.651	53.008	73.984	0.27675
13	20/80 Prior Vector	1	Stepwise	Validation Misclass.	0.3	0.2	30.732	48.130	70.244	0.27856
14	20/80 Prior Vector	2	Stepwise	Validation Misclass.	0.3	0.2	30.732	48.130	70.244	0.27856
15	20/80 Prior Vector	3	Stepwise	Validation Misclass.	0.3	0.2	30.732	48.130	70.244	0.27856
16	20/80 Prior Vector	1	Stepwise	Validation Error	0.3	0.2	30.702	47.767	70.894	0.27675
17	20/80 Prior Vector	2	Stepwise	Validation Error	0.3	0.2	30.702	47.767	70.894	0.27675
18	20/80 Prior Vector	3	Stepwise	Validation Error	0.3	0.2	30.702	47.767	70.894	0.27675
19	20/80 Oversampled	1	Stepwise	Validation Misclass.	0.3	0.2	38.184	55.134	74.309	
20	20/80 Oversampled	2	Stepwise	Validation Misclass.	0.3	0.2	38.184	55.134	74.309	
21	20/80 Oversampled	3	Stepwise	Validation Misclass.	0.3	0.2	38.184	55.134	74.309	
22	20/80 Oversampled	1	Stepwise	Validation Error	0.3	0.2	38.184	55.134	74.309	
23	20/80 Oversampled	2	Stepwise	Validation Error	0.3	0.2	38.184	55.134	74.309	
24	20/80 Oversampled	3	Stepwise	Validation Error	0.3	0.2	38.184	55.134	74.309	

*
Optimization Method 1 = Quasi-Newton
Optimization Method 2 = Newton-Raphson w Ridging
Optimization Method 3 = Trust-Region

Appendix B - Decision Tree Testing Results

Seq	Tree Type	Prior Probability	Min Size of Leaf	Obs's Req'd for a Split	Max # Branches from a Node	Max Depth of Tree	Cumul Captured Response Rate % @10%	Cumul Captured Response Rate % @20%	Cumul Captured Response Rate % @40%	Root ASE on Validation Set
1	CHAID	(model - 9.52%)	15	30	2	6	34.880	44.063	58.047	0.28429
2	CART	(model - 9.52%)	15	30	2	6	33.651	44.056	58.042	0.28464
3	C45	(model - 9.52%)	15	30	2	6	33.651	44.056	58.042	0.28464
4	CHAID	(model - 9.52%)	20	40	2	6	34.880	44.063	58.047	0.28260
5	CART	(model - 9.52%)	20	40	2	6	33.651	44.056	58.042	0.28464
6	C45	(model - 9.52%)	20	40	2	6	33.651	44.056	58.042	0.28464
7	CHAID	(model - 9.52%)	25	50	2	6	34.880	44.063	58.047	0.28260
8	CART	(model - 9.52%)	25	50	2	6	33.651	44.056	58.042	0.28464
9	C45	(model - 9.52%)	25	50	2	6	33.651	44.056	58.042	0.28464
10	CHAID	(model - 9.52%)	15	45	3	6	34.171	44.063	58.047	0.28325
11	CART	(model - 9.52%)	15	45	3	6	35.972	51.446	64.473	0.28195
12	C45	(model - 9.52%)	15	45	3	6	35.957	47.689	69.423	0.28217
13	CHAID	(model - 9.52%)	20	60	3	6	32.443	44.063	58.047	0.28464
14	CART	(model - 9.52%)	20	60	3	6	35.972	51.446	64.473	0.28244
15	C45	(model - 9.52%)	20	60	3	6	35.514	47.024	68.758	0.28277
16	CHAID	(model - 9.52%)	25	75	3	6	35.527	44.063	58.047	0.28308
17	CART	(model - 9.52%)	25	75	3	6	35.972	51.446	64.473	0.28236
18	C45	(model - 9.52%)	25	75	3	6	35.514	47.024	68.758	0.28264
19	CHAID	(model - 9.52%)	15	30	2	8	34.880	44.063	58.047	0.28429
20	CART	(model - 9.52%)	15	30	2	8	35.148	51.244	67.694	0.28233
21	C45	(model - 9.52%)	15	30	2	8	33.651	44.056	58.042	0.28464
22	CHAID	(model - 9.52%)	20	40	2	8	34.880	44.063	58.047	0.28261
23	CART	(model - 9.52%)	20	40	2	8	33.651	44.056	58.042	0.28464
24	C45	(model - 9.52%)	20	40	2	8	33.651	44.056	58.042	0.28464
25	CHAID	(model - 9.52%)	25	50	2	8	34.880	44.063	58.047	0.28260
26	CART	(model - 9.52%)	25	50	2	8	33.651	44.056	58.042	0.28646
27	C45	(model - 9.52%)	25	50	2	8	33.651	44.056	58.042	0.28464
28	CHAID	(model - 9.52%)	15	45	3	8	34.171	44.063	58.047	0.28325
29	CART	(model - 9.52%)	15	45	3	8	35.972	51.446	64.473	0.28195
30	C45	(model - 9.52%)	15	45	3	8	35.957	47.689	69.423	0.28217
31	CHAID	20/80 Prior Vector	15	30	2	6	34.880	44.063	58.047	0.28430
32	CART	20/80 Prior Vector	15	30	2	6	35.140	49.380	67.661	0.28371
33	C45	20/80 Prior Vector	15	30	2	6	34.263	48.847	67.387	0.02837
34	CHAID	20/80 Prior Vector	20	40	2	6	34.880	44.063	58.047	0.28261
35	CART	20/80 Prior Vector	20	40	2	6	35.140	49.380	67.661	0.28371
36	C45	20/80 Prior Vector	20	40	2	6	34.263	48.847	67.387	0.28371
37	CHAID	20/80 Prior Vector	25	50	2	6	34.880	44.063	58.047	0.28261
38	CART	20/80 Prior Vector	25	50	2	6	34.787	49.256	67.661	0.28360
39	C45	20/80 Prior Vector	25	50	2	6	33.880	48.720	67.387	0.28306
40	CHAID	20/80 Prior Vector	15	45	3	6	34.110	44.063	58.047	0.28325
41	CART	20/80 Prior Vector	15	45	3	6	34.115	49.433	70.505	0.28336
42	C45	20/80 Prior Vector	15	45	3	6	33.275	44.932	66.527	0.28257
43	CHAID	20/80 Prior Vector	20	60	3	6	32.443	44.063	58.047	0.28464
44	CART	20/80 Prior Vector	20	60	3	6	33.126	48.133	62.626	0.28270
45	C45	20/80 Prior Vector	20	60	3	6	30.968	43.549	65.144	0.28203
46	CHAID	20/80 Prior Vector	25	75	3	6	35.527	44.063	58.047	0.28309
47	CART	20/80 Prior Vector	25	75	3	6	30.968	46.741	68.188	0.28152
48	C45	20/80 Prior Vector	25	75	3	6	30.968	43.549	65.144	0.28203
49	CHAID	20/80 Prior Vector	15	30	2	8	34.880	44.063	58.047	0.28430
50	CART	20/80 Prior Vector	15	30	2	8	35.140	49.380	67.661	0.28372
51	C45	20/80 Prior Vector	15	30	2	8	34.263	48.847	67.387	0.28372
52	CHAID	20/80 Prior Vector	20	40	2	8	34.880	44.063	58.047	0.28261
53	CART	20/80 Prior Vector	20	40	2	8	35.140	49.380	67.661	0.28371
54	C45	20/80 Prior Vector	20	40	2	8	34.263	48.847	67.387	0.28372
55	CHAID	20/80 Prior Vector	25	50	2	8	34.880	44.063	58.047	0.28261

56	CART	20/80 Prior Vector	25	50	2	8	34.787	49.256	67.661	0.28359
57	C45	20/80 Prior Vector	25	50	2	8	33.880	48.720	67.387	0.28361
58	CHAID	20/80 Prior Vector	15	45	3	8	34.171	44.063	58.047	0.28325
59	CART	20/80 Prior Vector	15	45	3	8	34.115	49.433	70.505	0.28336
60	C45	20/80 Prior Vector	15	45	3	8	33.275	44.932	66.527	0.28257
61	CHAID	20/80 Oversampled**	15	30	2	6	42.062	55.112	69.531	
62	CART	20/80 Oversampled**	15	30	2	6	43.009	57.023	69.531	
63	C45	20/80 Oversampled**	15	30	2	6	42.314	55.638	68.205	
64	CHAID	20/80 Oversampled**	20	40	2	6	38.435	46.213	59.660	
65	CART	20/80 Oversampled**	20	40	2	6	38.161	49.021	61.766	
66	C45	20/80 Oversampled**	20	40	2	6	38.968	55.164	68.083	
67	CHAID	20/80 Oversampled**	25	50	2	6	38.435	46.213	59.660	
68	CART	20/80 Oversampled**	25	50	2	6	38.435	46.213	59.660	
69	C45	20/80 Oversampled**	25	50	2	6	35.634	49.167	61.875	
70	CHAID	20/80 Oversampled**	15	45	3	6	33.817	49.494	67.855	
71	CART	20/80 Oversampled**	15	45	3	6	43.089	56.513	68.289	
72	C45	20/80 Oversampled**	15	45	3	6	41.569	53.857	68.267	
73	CHAID	20/80 Oversampled**	20	60	3	6	33.817	49.494	62.498	
74	CART	20/80 Oversampled**	20	60	3	6	35.810	51.487	72.292	
75	C45	20/80 Oversampled**	20	60	3	6	37.145	50.581	68.267	
76	CHAID	20/80 Oversampled**	25	75	3	6	33.437	49.114	67.855	
77	CART	20/80 Oversampled**	25	75	3	6	33.347	49.114	67.855	
78	C45	20/80 Oversampled**	25	75	3	6	37.112	50.549	68.267	
79	CHAID	20/80 Oversampled**	15	30	2	8	42.062	55.112	69.531	
80	CART	20/80 Oversampled**	15	30	2	8	44.394	57.023	69.531	
81	C45	20/80 Oversampled**	15	30	2	8	45.661	57.494	69.063	
82	CHAID	20/80 Oversampled**	20	40	2	8	38.435	46.213	59.660	
83	CART	20/80 Oversampled**	20	40	2	8	44.171	55.491	68.083	
84	C45	20/80 Oversampled**	20	40	2	8	38.435	46.213	59.660	
85	CHAID	20/80 Oversampled**	25	50	2	8	38.435	46.213	59.660	
86	CART	20/80 Oversampled**	25	50	2	8	40.220	49.021	61.776	
87	C45	20/80 Oversampled**	25	50	2	8	38.435	46.213	59.660	
88	CHAID	20/80 Oversampled**	15	45	3	8	43.089	56.513	68.289	
89	CART	20/80 Oversampled**	15	45	3	8	41.569	53.857	68.267	
90	C45	20/80 Oversampled**	15	45	3	8	33.817	49.494	67.855	
91	CHAID	20/80 Oversampled**	15	30	2	6	32.858	47.175	60.479	
92	CART	20/80 Oversampled**	15	30	2	6	41.272	53.507	67.783	
93	C45	20/80 Oversampled**	15	30	2	6	42.298	55.638	68.205	
94	CHAID	20/80 Oversampled**	20	40	2	6	35.375	46.937	60.240	
95	CART	20/80 Oversampled**	20	40	2	6	38.617	49.167	61.875	
96	C45	20/80 Oversampled**	20	40	2	6	43.596	55.491	68.083	
97	CHAID	20/80 Oversampled**	25	50	2	6				
98	CART	20/80 Oversampled**	25	50	2	6				
99	C45	20/80 Oversampled**	25	50	2	6				
100	CHAID	20/80 Oversampled**	15	45	3	6				
101	CART	20/80 Oversampled**	15	45	3	6				
102	C45	20/80 Oversampled**	15	45	3	6				
103	CHAID	20/80 Oversampled**	20	60	3	6				
104	CART	20/80 Oversampled**	20	60	3	6				
105	C45	20/80 Oversampled**	20	60	3	6				
106	CHAID	20/80 Oversampled**	25	75	3	6				
107	CART	20/80 Oversampled**	25	75	3	6				
108	C45	20/80 Oversampled**	25	75	3	6				
109	CHAID	20/80 Oversampled**	15	30	2	8				
110	CART	20/80 Oversampled**	15	30	2	8				
111	C45	20/80 Oversampled**	15	30	2	8				
112	CHAID	20/80 Oversampled**	20	40	2	8				
113	CART	20/80 Oversampled**	20	40	2	8				
114	C45	20/80 Oversampled**	20	40	2	8				
115	CHAID	20/80 Oversampled**	25	50	2	8	31.784	47.558	70.407	
116	CART	20/80 Oversampled**	25	50	2	8	38.617	49.167	61.875	
117	C45	20/80 Oversampled**	25	50	2	8	40.220	49.021	61.766	
118	CHAID	20/80 Oversampled**	15	45	3	8	36.594	52.527	66.239	

119	CART	20/80 Oversampled**	15	45	3	8	41.972	55.693	68.099
120	C45	20/80 Oversampled**	15	45	3	8	40.047	53.704	68.267

* 20/80 Oversampled, use Prior on Split Search=N
** 20/80 Oversampled, use Prior on Split Search=Y

Appendix C - Artificial Neural Network Testing Results

Seq	Prior Probability	NN Config	Model Selection Criteria	Variable Selection R-sq Cutoff	Cumul Captured Response Rate % @10%	Cumul Captured Response Rate % @20%	Cumul Captured Response Rate % @40%	Root ASE on Validation Set
1	(model - 9.52%)	Default	Average Error	60%	36.585	55.167	72.520	0.27772
2	(model - 9.52%)	NN2	Average Error	60%	38.211	56.098	73.821	0.27776
3	(model - 9.52%)	NN3	Average Error	60%	36.098	53.171	72.195	0.27644
4	(model - 9.52%)	Default	Average Error	80%	36.585	55.167	72.520	0.27772
5	(model - 9.52%)	NN2	Average Error	80%	38.211	56.098	73.821	0.27776
6	(model - 9.52%)	NN3	Average Error	80%	36.098	53.171	72.195	0.27644
7	(model - 9.52%)	Default	Average Error	95%	36.585	55.167	72.520	0.27772
8	(model - 9.52%)	NN2	Average Error	95%	38.211	56.098	73.821	0.27776
9	(model - 9.52%)	NN3	Average Error	95%	36.098	53.171	72.195	0.27644
10	(model - 9.52%)	Default	Misclassification Rate	60%	37.073	55.610	72.358	0.27800
11	(model - 9.52%)	NN2	Misclassification Rate	60%	38.374	56.260	73.821	0.27800
12	(model - 9.52%)	NN3	Misclassification Rate	60%	37.724	55.285	73.821	0.27960
13	20/80 Prior Vector	Default	Average Error	60%	29.919	49.131	69.919	0.27717
14	20/80 Prior Vector	NN2	Average Error	60%	31.382	49.756	71.870	0.27776
15	20/80 Prior Vector	NN3	Average Error	60%	29.431	48.293	69.268	0.27644
16	20/80 Oversampled	Default	Average Error	60%	39.350	54.634	75.285	n/a
17	20/80 Oversampled	NN2	Average Error	60%	45.134	60.976	77.561	n/a
18	20/80 Oversampled	NN3	Average Error	60%	43.740	60.488	77.886	n/a

* Network architecture for all configurations is multi-layer perceptron

XII. References

[1] Joel W. Stoesser and Robert C. Hess, Point of View Styles of Higher Return Strategies, *Journal of Real Estate Portfolio Management*, 2000, 6:4, 417-22.

[2] Daniel Rivetti and Elaine Worzala, An Investigation into the Credit Tenant Characteristics of Department of Defense Contractors, *Journal of Real Estate Portfolio Management*, 2003, 9:2, 179-85.

[3] Brian Ciochetti, Yongheng Deng, Gail Lee, James Shilling, Rui Yao , A Proportional Hazards Model of Commercial Mortgage Default with Originator Bias, *Journal of Real Estate Finance and Economics*, 2003, 27:1, 5-23.

[4] Eichenblatt, D. L. and J. C. Yeoman, Monte Carlo Simulation: Impact of Tenant Size and Credit Quality on Office Buildings, *Real Estate Review*, 1999, 29:3, 47-51.

[5] Michael J.A. Berry and Gordon S. Linoff, *Data Mining Techniques for Marketing, Sales, and Customer Relationship Management (Second Edition)*, Indianapolis: Wiley Publishing, Inc., 2004.

[6] Susan Fowler McNally, Carter H. Klein & Michael S. Abrams, How to structure a lease to protect against the risk of a bankruptcy of the tenant" *2002 Annual Survey of Letter of Credit Law & Practice*, Inst. of International Banking Law & Practice, Inc. and Real Estate Issues, Fall 2001

XIII. End Notes

[a] The cost of correction is speculative since it depends on the chosen course of action. However, a proportional hazards model would more accurately account for the risk of a large payment versus a small one. An example of this being applied to default risk is Ciochetti's study, "A Proportial Hazards Model of Commercial Mortgage Default with Originator Bias," (Brian Ciochetti, Yongheng Deng, Gail Lee, James Shilling, Rui Yao) *2003 Journal of Real Estate Finance and Economics*, 27:1, 5-23.

[b] A more thorough study of this behavior would require a separate mining model for new tenants.

[c] There are other approaches to reducing skewness, such as standardizing values and putting caps and floors on values. Skewness can also be caused by bad data (outliers) and sometimes it makes sense to discard this data.

[d] 1998-11-08, 1999-05-08, 2000-07-08, 2001-04-08, 2002-10-08, 2003-06-08, 004-02-08, 2004-08-08, 2005-03-08, 2005-09-08, 2006-01-08

[e] Figure 7 is an oversimplification made to allow a visualization. In fact, history back to lease inception was evaluated for each tenant

[f] This measurement does not consider the cost of incorrectly guessing that a tenant will be a late payer. This was a simplifying assumption made for this study, but others may want to give this more careful consideration.

[g] It actually returns the log of the odds, which can be transformed into a probability of encountering the target event.

[h] Ranked in terms of the absolute value of the standardized parameter estimate.

[i] Stability refers to the significant variables and their coefficients not changing when new data is input. Stability requires rigorous design and testing.

[j] The number of observations is slightly different than 30% of 9,259 due to using a 20% stratified oversampled training data set with proportions enforced.

[k] The measure of importance is proportional to the square root of the sum over nodes of the sum of square errors in the node minus the sum of square errors in branches that are created by the single split. The measure of importance is considerable for a variable that is used in one effective split, and also for variables that are used in many splits that are less effective.

[l] The algorithm is also known as an activation function.

[m] Oversampling increases the percentage of late payers in the training set, but not the validation set. Doing this improved lift as Appendix B shows.