# *Statistical Learning Algorithms Applied to Automobile Insurance Ratemaking*

Charles Dugas, Yoshua Bengio,
Nicolas Chapados, Pascal Vincent,
Germain Denoncourt, FCAS, FCIA, MAAA,
and Christian Fournier, FCAS, FCIA

# Statistical Learning Algorithms Applied to Automobile Insurance Ratemaking

**Charles Dugas, Yoshua Bengio, Nicolas Chapados and Pascal Vincent**
{DUGAS,BENGIO,CHAPADOS,VINCENT}@APSTAT.COM
*Apstat Technologies Inc.*

**Germain Denoncourt, F.C.A.S., F.C.I.A.**
GDENONCOURT@ASSURANCE-ALPHA.COM
*L'Alpha Compagnie d'Assurance Inc.*

**Christian Fournier, F.C.A.S., F.C.I.A.**
CHRISTIAN.FOURNIER@ICBC.COM
*Insurance Corporation of British Columbia*

## Abstract

We recently conducted a research project for a large North American automobile insurer. This study was the most exhaustive ever undertaken by this particular insurer and lasted over an entire year. We analyzed the discriminating power of each variable used for ratemaking. We analyzed the performance of several models within five broad categories: linear regressions, generalized linear models, decision trees, neural networks and support vector machines. In this paper, we present the main results of this study. We qualitatively compare models and show how neural networks can represent high-order nonlinear dependencies with a small number of parameters, each of which is estimated on a large proportion of the data, thus yielding low variance. We thoroughly explain the purpose of the nonlinear sigmoidal transforms which are at the very heart of neural networks' performances. The main numerical result is a statistically significant reduction in the out-of-sample mean-squared error using the neural network model and our ability to substantially reduce the median premium by charging more to the highest risks. This in turn can translate into substantial savings and financial benefits for an insurer. We hope this paper goes a long way towards convincing actuaries to include neural networks within their set of modeling tools for ratemaking.

## 1. Introduction

Ratemaking is one of the main mathematical problems faced by actuaries. They must first estimate how much each insurance contract is expected to cost. This conditional expected claim amount is called the *pure premium* and it is the basis of the *gross premium* charged to the insured. This expected value is conditioned on information available about the insured and about the contract, which we call the *input profile.*

Automobile insurance ratemaking is a complex task for many reasons. First of all, many factors are relevant. Taking account of each of them individually, i.e., making independence assumptions, can be hurtful (Bailey and Simon (1960)). Taking account of all interactions is intractable and is sometimes referred to as the *curse of dimensionality* (Bellman (1957)). In practice, actuarial judgment is used to discover the most relevant of these interactions and feed them explicitly to the model. Neural networks, on the other hand, are well-known for their ability to represent high-order nonlinear interactions with a small number of parameters, i.e., they can automatically detect those most relevant interactions between variables (Rumelhart et al. (1986)). We explain how and why in section 4.

A second difficulty comes from the distribution of claims: asymmetric with fat tails with a large majority of zeros and a few unreliable and very large values, i.e., an asymmetric heavy tail extending out toward high positive values. Modeling data with such a distribution is essentially difficult because *outliers*, which are sampled from the tail of the distribution, have a strong influence on parameter estimation. When the distribution is symmetric around the mean, the problems caused by outliers can be reduced using *robust* estimation techniques (Huber (1982), Hampel et al. (1986), Rousseeuw and Leroy (1987)) which basically intend to ignore or down-weight outliers. Note that these techniques do not work for an asymmetric distribution: most outliers are on the same side of the mean, so down-weighting them introduces a strong bias on its estimation: the conditional expectation would be systematically underestimated. Recent developments for dealing with asymmetric heavy-tail distributions have been made (Takeuchi et al. (2002)).

The third difficulty is due to the non-stationary nature of the relationship between explanatory variables and the expected claim amount. This has an important effect on the methodology to use, in particular with respect to the task of *model selection*. We describe our methodology in section 3.

Fourth, from year to year, the general level of claims may fluctuate heavily, in particular in states and provinces where winter plays an important role in the frequency and severity of accidents. The growth of the economy and the price of gas can also affect these figures.

Fifth, one needs sufficient computational power to develop models: we had access to a large database of $\approx 8 \times 10^6$ records, and the training effort and numerical stability of some algorithms can be burdensome for such a large number of training examples. In particular, neural networks are computationally very demanding.

Sixth, the data may be of poor quality. In particular, there may be missing fields for many records. An actuary could systematically discard incomplete records but this leads to loss of information. Also, this strategy could induce a bias if the absence of a data is not random but rather correlated to some particular feature which affects the level of risk. Alternatively one could choose among known techniques for dealing with missing values (Dempster et al. (1977), Ghahramani and Jordan (1994), Bengio and Gingras (1996)).

Seventh, once the pure premiums have been established the actuary must properly allocate expenses and a reserve for profit among the different contracts in order to obtain the gross premium level that will be charged to the insureds. Finally, an actuary must account for competitive concerns: his company's strategic goals, other insurers' rate changes, projected renewal rates and market elasticity.

In this paper, we address the task of setting an appropriate pure premium level for each contract, i.e., difficulties one through four as described above. Our goal is to compare different models with respect to their performance in that regard, i.e., how well they are able to forecast the claim level associated to each contract. We chose several models within five broad categories: linear regressions, generalized linear models (McCullagh and Nelder (1989)), decision trees (Kass (1980)), neural networks and support vector machines (Vapnik (1998a)).

The rest of the paper is organized as follows: we start by describing the mathematical criteria underlying insurance premium estimation (section 2). Our methodology is described in section 3, followed by a review of the statistical learning algorithms that we consider in this study, including our best-performing mixture of positive-output neural networks (section 4). We then highlight our most important experimental results (section 5), and in view of them conclude with an examination of the prospects for applying statistical learning algorithms to insurance modeling (section 7).

## 2. Mathematical Objectives

The first goal of insurance premium modeling is to estimate the *expected claim amount* for a given insurance contract for a future period (usually one year). Here we consider that the amount is 0 when no claim is filed. Let $X \in \mathbf{R}^n$ denote the customer and contract *input profile*, a vector representing all the information known about the customer and the proposed insurance policy before the beginning of the contract. Let $A \in \mathbf{R}^+$ denote the amount that the customer claims during the contract period; we shall assume that $A$ is non-negative. Our objective is to estimate this claim amount, which is the *pure premium* $p_{pure}$ of a given contract $x$:[1]

$$p_{pure}(x) = E[A|X = x]. \tag{1}$$

where $E[\cdot]$ denotes expectation, i.e. the average over an infinite population, and $E[A|X = x]$ is a conditional expectation, i.e. the average over a subset of an infinite population, comprising only the cases satisfying the condition $X = x$.

### 2.1 The Precision Criterion

In practice, of course, we have no direct access to the quantity (1), which we must estimate. One possible criterion is to seek the *most precise* predictor, which minimizes the expected squared error (ESE) over the unknown distribution:

$$E[(p(X) - A)^2], \tag{2}$$

where $p(X)$ is a pure premium predictor and the expectation is taken over the random variables $X$ (input profile) and $A$ (total claim amount). Since the true joint distribution of $A$ and $X$ is unknown, we can **unbiasedly** estimate the ESE performance of an estimator $p(X)$ on a data set $D_{test} = \{\langle x_i, a_i\rangle\}_{i=1}^N$, as long as this data set is not used to choose $p$, using the **mean-squared error** on that data set:

$$\frac{1}{N} \sum_{\langle x_i, a_i\rangle \in D_{test}} (p(x_i;\theta) - a_i)^2, \tag{3}$$

1. The pure premium is distinguished from the premium actually charged to the customer, which must account for the underwriting costs (marketing, commissions, premium tax), administrative overhead, risk and profit loadings and other costs.

where $\theta$ is the vector of parameters of the model used to compute the premiums. The vector $x_i$ represents the $i^{\text{th}}$ input profile of dataset $D_{test}$ and $a_i$ is the claim amount associated to that input profile. Thus, $D_{test}$ is a set of $N$ insurance policies. For each policy, $D_{test}$ holds the input profile and associated incurred amount. We will call the data set $D_{test}$ a **test set**. It is used only to independently assess the performance of a predictor $p$. To **choose** $p$ from a (usually infinite) set of possible predictors, one uses an *estimator* $L$, which obtains a predictor $p$ from a given **training set** $D$. Such an estimator is really a **statistical learning algorithm** (Hastie et al. (2001)), yielding a predictor $p = L_D$ for a given data set $D$. What we call the **squared bias** of such an estimator is $(E[A|X] - E[L_D(X)])^2$, where $E[L_D(X)]$ is the average predictor obtained by considering all possible training sets $D$ (sampled from $P(A,X)$). It represents how far the average estimated predictor deviates from the ideal pure premium, $E[A|X]$. What we call the **variance** of such an estimator is $E[(L_D(X) - E[L_D(X)])^2]$. It represents how the particular predictor obtained with some data set $D$ deviates from the average of predictors over all data sets, i.e. it represents the sensitivity of the estimator to the variations in the training data and is related to the classical measure of **credibility**.

Is the mean-squared error (MSE) on a test set an appropriate criterion to evaluate the predictive power of a predictor $p$? First one should note that if $p_1$ and $p_2$ are two predictors of $E[A|X]$, then the MSE criterion is a good indication of how close they are to $E[A|X]$, since by the law of iterated expectations,

$$\begin{aligned} E[(p_1(X)-A)^2] - E[(p_2(X)-A)^2] &= E[(p_1(X)-E[A|X])^2] \\ &\quad -E[(p_2(X)-E[A|X])^2], \end{aligned}$$

and of course the expected MSE is minimized when $p(X) = E[A|X]$.

For the more mathematically-minded readers, we show that minimizing the expected squared error optimizes simultaneously both the precision (low bias) and the variance of the estimator. We denote $E_D$ the expectation over the training set $D$. The expected squared error of an estimator $L_D$ decomposes as follows:

$$\begin{aligned} E[(A-L_D(X))^2] &= E[((A-E[A|X]) + (E[A|X]-L_D(X)))^2] \\ &= \underbrace{E[(A-E[A|X])^2]}_{\text{noise}} + E[(E[A|X]-L_D(X))^2] \\ &\quad +2\underbrace{E[(A-E[A|X])(E[A|X]-L_D(X))]}_{\text{zero}} \\ &= \text{noise} + E[((E[A|X]-E_D[L_D(X)]) + (E_D[L_D(X)]-L_D(X)))^2] \\ &= \text{noise} + E[(E[A|X]-E_D[L_D(X)])^2] + E[(E_D[L_D(X)]-L_D(X))^2] \\ &\quad +2\underbrace{E[(E[A|X]-E_D[L_D(X)])(E_D[L_D(X)]-L_D(X))]}_{\text{zero}} \\ &= \text{noise} + \underbrace{E[(E[A|X]-E_D[L_D(X)])^2]}_{\text{bias}^2} + \underbrace{E[(E_D[L_D(X)]-L_D(X))^2]}_{\text{variance}}. \end{aligned}$$

Thus, algorithms that try to minimize the expected squared error simultaneously reduce both the bias and the variance of the estimators, striking a tradeoff that minimizes the sum of both (since the remainder is the noise, which cannot be reduced by the choice of predictor). On the other hand, with a rule such as minimum-bias used with table-based methods, cells are merged up to a point where each cell has sufficient credibility, i.e., where the variance is sufficiently low. Then, once the credibility (and variance) level is set fixed, the bias is minimized. On the contrary, by targeting minimization of the expected squared error one avoids this arbitrary setting of a credibility level.

In comparison to parametric approaches, this approach avoids distributional assumptions. Furthermore, it looks for an optimal trade-off between bias and variance, whereas parametric approaches typically focus on the unbiased estimators (within a class that is associated with a certain variance). Because of the above trade-off possibility, it is always possible (with a finite data set) to improve an unbiased estimator by trading a bit of bias increase for a lot of variance reduction (Hastie et al. (2001)).

## 2.2 The Fairness Criterion

In insurance policy pricing, the precision criterion is not the sole part of the picture; just as important is that the estimated premiums do not systematically discriminate against specific segments of the population. We call this objective the *fairness criterion*, sometimes referred to as *actuarial fairness*. We define the *bias of the premium* $b(P)$ to be the difference between the average pure premium and the average incurred amount, in a given sub-population $P$ of dataset $D$:

$$b(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} p(x_i) - a_i, \tag{4}$$

where $|P|$ denotes the cardinality of the sub-population $P$, and $p(\cdot)$ is some premium estimation function. The vector $x_i$ represents the $i^{\text{th}}$ input profile of sub-population $P$ and $a_i$ is the claim amount associated to that input profile. A possible fairness criterion would be based on minimizing the sum, over a certain set of critical sub-populations $\{P_k\}$ of dataset $D$, of the square of the biases:

$$\sum_{k,P_k\in D} b^2(P_k) \tag{5}$$

In the particular case where one considers all sub-populations, then both the fairness and precision criterions lead to the same optimal solution, i.e., they are minimized when $p(x_i) = E[A_i|x_i]$, $\forall i$, i.e., for every insurance policy, the premium is equal to the conditional expectation of the claim amount. The proof is given in appendix A.

In order to measure the fairness criterion, we used the following methodology: after training a model to minimize the MSE criterion (3), we define a finite number of disjoint subsets (sub-populations) of test set $D$: $P_k \subset D, P_k \cap P_{j\neq k} = \emptyset$, and *verify* that the absolute bias is not significantly different from zero. The subsets $P_k$ can be chosen at convenience; in our experiments, we considered 10 subsets of equal size delimited by the deciles of the test set premium distribution. In this way, we verify that, for example, for the group of contracts with a premium between the 5th and the 6th decile, the average premium matches, within statistical significance, the average claim amount.

## 2.3 Penalized Training Criterion and Bias-Variance Tradeoff

Although our objective is to minimize the expected out-of-sample squared error (ESE), it does not mean that we should minimize the in-sample (*training set*) mean-squared error (MSE):

$$\frac{1}{L} \sum_{\langle x_t, a_t \rangle \in D_{train}} (p(x_i;\theta) - a_i)^2$$

in order to achieve that goal. The reason for that apparent discrepancy has to do with the bias-variance trade-off in generalization error (Geman et al. (1992)), and the fundamental principles of statistical learning theory (Vapnik (1998b)). To illustrate these ideas, let us consider the simple case of linear regression, which becomes ridge regression when the training criterion is penalized. Consider a class of linear predictive functions of the input $x$,

$$p(x) \;=\; \sum_{i=1}^{n} \beta_i x_i.$$

Instead of minimizing the training set mean-squared error (MSE), consider the following penalized criterion:

$$\frac{1}{L} \sum_{\langle x_t, a_t \rangle \in D_{train}} (p(x_i;\theta) - a_i)^2 \;+\; \lambda \sum_i \beta_i^2$$

with $\lambda \geq 0$ and a minimum achieved at $\hat{\beta}_\lambda$. Thus $\hat{\beta}_0$ is the Ordinary Least Squares estimator. This minimum is always achieved with **shrinked** solutions, i.e. $||\hat{\beta}_\lambda|| < ||\hat{\beta}_0||$ for $\lambda > 0$. Note that this solution is generally **biased**, unlike $\hat{\beta}_0$, in the sense that if the data is generated from a multivariate normal distribution, the expected value of $\hat{\beta}_\lambda$ is smaller than the true value $\beta$ from the underlying distribution. Note that the set of functions effectively allowed for a solution is smaller when $\lambda$ is larger.

In the case where linear regression is the proper model (normally distributed data, with output variance $\sigma^2$), and the *amount of data $l$ is finite*, it is easy to prove that the optimal fixed value of $\lambda$ (in expectation over different training sets) is

$$\lambda^* = \frac{\sigma^2}{l\,||\beta||^2}.$$

Note therefore that **the optimal model is biased** (optimal in the sense of minimizing out-of-sample error).

This example illustrates the more general principle of bias-variance trade-off in generalization error, well discussed by Geman et al. (1992). Increasing $\lambda$ corresponds to "smoothing more" in non-parametric statistics (choosing a simpler function) or to the choice of a smaller capacity ("smaller" class of functions) in Vapnik's VC-theory (Vapnik (1998b)). A too large value of $\lambda$ corresponds to **underfitting** (too simple model, too much bias), whereas a too small value corresponds to **overfitting**) (too complex model, too much variance). Which value of $\lambda$ should be chosen? It should be the one that strikes the optimal balance between bias and variance. This is the question that **model selection** algorithms address. Fortunately, the expected out-of-sample error has a unique minimum as a function of $\lambda$ (or more generally of the capacity, or complexity of the class of functions). Concerning the above formula, note that unfortunately the data is generally not normal, and $\sigma^2$ and $\beta$ are both unknown, so the above formula can't be used directly to choose $\lambda$. However, using a separate held-out data set (also called a *validation set*, here), and taking advantage of that unique minimum property (which is true for any data distribution), we can quickly select a good value of $\lambda$ (essentially by searching), which approximately minimizes the estimated out-of-sample error on that validation set.

Note that we arrive at the conclusion that a biased model is preferable because we set as our goal to minimize out-of-sample error. If our goal was to **discover** the underlying "truth", and if we could make very strong assumptions about the true nature of the data distribution, then the more classical statistical approach based on minimum variance unbiased estimators would be more appropriate. However, in the context of practical insurance premium estimation, we don't really know the form of the true data distribution, and we really care about how the model is going to perform in the future (at least for ratemaking).

## 3. Methodology

A delicate problem to guard against when applying statistical learning algorithms is that of *overfitting*. It has precisely to do with striking the right trade-off between bias and variance (as introduced in the previous section), and is known in technical terms as *capacity control*. Figure 1 illustrates the problem: the two plots show empirical data points (black dots) that we are trying to approximate with a function (solid red curve). All points are sampled from the same underlying function (dashed blue curve), but are corrupted with noise; the dashed curve may be seen as the "true" function we are seeking to estimate.

The left plot shows the result of fitting a very flexible function, i.e. a high-order polynomial in this case, to the available data points. We see that the function fits the data points perfectly: there is zero error (distance) between the red curve and each of the black dots. However, the function oscillates wildly between those points; it has not captured any of the fundamental features of the underlying function. What is happening here is that the function has mostly *captured the noise* in the data: it overfits.

The right plot, on the other hand, shows the fitting of a less flexible function, i.e. a 2nd-order polynomial, which exhibits a small error with respect to each data point. However, by not fitting the noise (because it does not have the necessary degrees of freedom), the fitted function far better conveys the structural essence of the matter.

*Capacity control* lies at the heart of a sound methodology for data mining and statistical learning algorithms. The goal is simple: to choose a function class flexible enough (with enough capacity) to express a desired solution, but constrained enough that it does not fit the noise in the data points. In other words, we want to avoid overfitting **and** underfitting.

Figure 2 illustrates the basic steps that are commonly taken to resolve this issue: these are not the only means to prevent overfitting, but are the simplest to understand and use.

1. The full data set is randomly split into three *disjoint* subsets, respectively called the training, validation, and test sets.

2. The training set is used to fit a model with a chosen initial capacity.

3. The validation set is used to *evaluate the performance* of that fitted function, on **different data points** than used for the fitting. The key here is that a function

overfitting the training set will exhibit a low performance on the validation set, if it does not capture the underlying structure of the problem.

4. Depending on the validation set performance, the capacity of the model is adjusted (increased or reduced), and a new training phase (step 2) is attempted. This training–validation cycle is repeated multiple times and the capacity that provides the best validation performance is chosen.

5. Finally, the performance of the "ultimate" function (that coming out of the validation phase) is evaluated on data points never used previously—those in the test set—to give a completely unbiased measure of the performance that can be expected when the system is deployed in the field. This is called **generalization performance**.

## 4. Models

In this section, we describe various models that have been implemented and used for the purpose of ratemaking. We begin with the simplest model: charging a flat premium to every insured. Then, we gradually move on towards more complex models.

### 4.1 Constant Model

For benchmark evaluation purposes, we implemented the constant model. This consists of simply charging every single insurance policy a flat premium, regardless of the associated variable values. The premium is the mean of all incurred amounts as it is the constant value that minimizes the mean-squared error.

$$p(x) = \beta_0, \tag{6}$$

where $\beta_0$ is the mean and the premium $p(x)$ is independent of the input profile $x$. In figure 3, the constant model is viewed as a flat line when the premium value is plotted against one of the input variables.

### 4.2 Linear Model

We implemented a linear model which consists of a set of coefficients, one for each variable plus an intercept value, that minimize the mean-squared error,

$$p(x) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i. \tag{7}$$

Figure 4 illustrates a linear model where the resulting premiums form a line, given one input variable value. With a two dimensional input variable space, a plane would be drawn. In higher dimension, the corresponding geometrical form is referred to as a hyper-plane.

There are two main ways to control the capacity of linear models when in presence of noisy data:

- using a subset of input variables; this directly reduces the number of coefficients (but choosing the best subset introduces another level of choice which is sometimes detrimental).

- penalizing the norm of the parameters (in general, intercept parameter $\beta_0$ is excluded from the penalty term); this is called *ridge regression* in statistics, and *weight decay* in the neural networks community. This was the main method used to control capacity of the linear model in our experiments (see above subsection 2.3).

It should be noted that the premium computed with the linear model can be negative (and negative values are indeed sometimes obtained with the trained linear models). This may happen even if there are no negative amounts in the data, simply because the model has no built-in positivity constraint (unlike the GLM and the softplus neural network described below).

## 4.3 Table-based methods

These more traditional ratemaking methods rely mainly on a classification system, base rates and relativities. The target function is approximated by constants over regular (finite) intervals. As shown on the figure, this gives rise to a typical staircase-like function, where each level of the staircase is given by the value in the corresponding cell in the table. A common refinement in one dimension is to perform a linear interpolation between neighboring cells, to smooth the resulting function somewhat. The table is not limited to two variables; however, when adding a new variable (dimension), the number of cells increases by a factor equal to the number of discretization steps in the new variable.

In order to use table-based methods to estimate a pure premium, find a certain number of variables deemed useful for the prediction, and discretize those variables if they are continuous. To fill out the table, compute over a number of years (using historical data) the total incurred claim amount for all customers whose profiles *fall within a given cell* of the table, and average the total within that cell. This gives the pure premium associated with each cell of the table.

Assuming that the $i^{\text{th}}$ variable of profile $x$ belongs to the $j^{\text{th}}$ category, we obtain,

$$p(x) = \beta_0 \prod_{i=1}^{m} \beta_{i,j} + \sum_{i=m+1}^{n} \beta_{i,j}, \tag{8}$$

where $\beta_{i,j}$ is the relativity for the $j^{\text{th}}$ category of the $i^{\text{th}}$ variable and $\beta_0$ is the standard premium. We consider the case where the first $m$ factors are multiplicative and the last $n-m$ factors are additive.

The formula above assumes that all variables have been analyzed individually and independently. A great deal of effort is often put in trying to capture dependencies (or interactions) between some variables and to encode them into the premium model.

An extension of the above is to multiplicatively combine multiple tables associated to different subsets of variables. This is in effect a particular form of generalized linear model (see below), where each table represents the interdependence effects between some variables.

## 4.4 Greedy Multiplicative Model

Greedy learning algorithms "grow" a model by gradually adding one "piece" at a time to the model, but keeping the already chosen pieces fixed. At each step, the "piece" that is most helpful to minimize the training criterion is "added" to the model. This is how decision trees are typically built. Using the validation set performance we can decide when to stop adding pieces (when the estimated out-of-sample performance starts degrading).

The GLM described in the next section is a multiplicative model because the final premium function can be seen as a product of coefficients associated with each input variable. The basic idea of the Greedy Multiplicative Model is to add one of these multiplicative coefficients at a time. At each step, we have to choose one among the input variables. We choose the variable which would reduce most the training MSE. The coefficient for that component is easily obtained analytically by minimizing the MSE when all the previously obtained coefficients are kept fixed.

In the tables we use the short-hand name "CondMean" for this model because it estimates and combines many conditional means. Note that like the GLM, this model provides positive premiums.

### 4.5 Generalized Linear Model

Bailey and Simon (1960) introduced generalized linear models (GLM) to the actuarial community four decades ago. More recently, Brown (1988), Holler et al. (1999), Murphy et al. (2000) conducted experiments using such models. GLMs, at their roots, are simple linear models that are composed with a fixed nonlinearity (the so-called *link function*); a commonly-used link function is simply the exponential function $e^x$. GLMs (with the exponential link) are sometimes used in actuarial modeling since they naturally represent *multiplicative effects*, for example risk factors whose effects should combine multiplicatively rather than additively. They are attractive since they incorporate *problem-specific knowledge* directly into the model. These models can be used to obtain a pure premium, yielding such a formula,

$$p(x) \quad = \quad \exp\left(\beta_0 + \sum_{i=1}^{n} \beta_i x_i\right), \qquad (9)$$

where, the exponentiation ensures that the resulting premiums are all positive. In figure 5, we can see that the model generates an exponential function in terms of the input variable.

In their favor, GLMs are quite easy to estimate[2], have interpretable parameters, can be associated to parametric noise models, and are not so affected when the number of explanatory variables increases, as long as the number of observations used in the estimation remains sufficient. Unfortunately, they are fairly restricted in the shape of the functions they can estimate.

The capacity of a GLM model can be controlled using the same techniques as those mentionned above (4.2) in the context of linear models. Again, note that the GLM always provides a positive premium.

### 4.6 CHAID decision trees

Decision trees split the variable space in smaller subspaces. Any input profile $x$ fits into one and only one of those subspaces called *leaves*. To each leaf is associated a different premium

2. We have estimated the parameters to minimize the mean-squared error, but other training criteria have also been proposed in the GLM literature and this could be the subject of further studies.

level,

$$p(x) = \sum_{i=1}^{n_l} \mathbf{I}_{\{x \in l_i\}} \beta_i, \tag{10}$$

where $\mathbf{I}_{\{x\in l_i\}}$ is an indicator function equal to 1 if and only if $x$ belongs to the $i^{\text{th}}$ leaf. In that case, $\mathbf{I}_{\{x\in l_i\}} = 1$ and $p(x) = \beta_i$. Otherwise, $\mathbf{I}_{\{x\in l_i\}}$ is equal to zero, meaning $x$ belongs to another leaf. The number of leaves is $n_l$. The premium level $\beta_i$ is set equal to the average incurred amount of the policies for which the profile $x$ belongs to the $i^{\text{th}}$ leaf. In figure 6, the decision tree is viewed as generating a piecewise constant function. The task of the decision tree is to choose the "best" possible partition of the input variable space.

The basic way in which capacity is controlled is through several hyper-parameters: minimum population in each leaf, minimum population to consider splitting a node, maximum height of the decision tree and, in the case of CHAID decision trees (Kass (1980)), Chi-square statistic threshold value.

## 4.7 Combination of CHAID and Linear Model

This model is similar to the previous one except that, in each leaf, we have replaced the associated constant premium value with a linear regression. Each leaf has its own set of regression coefficients. There are thus $n_l$ different linear regressions of $n + 1$ coefficients each.

$$p(x) = \sum_{i=1}^{n_l} \mathbf{I}_{\{x \in l_i\}} \left( \beta_{i,0} + \sum_{j=1}^{n} \beta_{i,j} x_j \right). \tag{11}$$

Each linear regression was fit to minimize the mean-squared error on the training cases that belong to its leaf. For reasons that are clear in the light of learning theory, a tree used in such a combination should have less leaves than an ordinary CHAID tree. In our experiments we have chosen the size of the tree based on the validation set MSE.

In these models, capacity is controlled with the same hyper-parameters as CHAID, and there is also the question of finding the right *weight decay* for the linear regression. Again, the validation set is used for this purpose.

## 4.8 Ordinary Neural Network

Ordinary neural networks consist of the clever combination and simultaneous training of a group of units or *neurons* that are individually quite simple. Figure 8 illustrates a typical *multi-layer feedforward architecture* such as the ones that were used for the current project.

We describe here the steps that lead to the computation of the final output of the neural network. First, we compute a series of linear combinations of the input variables:

$$v_i = \alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} x_j, \tag{12}$$

where $x_j$ is the $j^{\text{th}}$ out of $n$ variables, $\alpha_{i,0}$ and $\alpha_{i,j}$ are the slope intercept and the weights of the $i^{\text{th}}$ linear combination. The result of the linear combination, $v_i$, is often referred to as the *level of activation* in analogy to the neurons in the brain.

Then, a non-linear transform (called a *transfer function*) is applied to each of the linear combinations in order to obtain what are called the *hidden units*. We used the hyperbolic tangent function:

$$\begin{aligned} h_i &= \tanh(v_i) \\ &= \frac{e^{v_i} - e^{-v_i}}{e^{v_i} + e^{-v_i}}, \end{aligned} \tag{13}$$

where $h_i$ is the $i^{\text{th}}$ hidden unit. The use of such a transfer function with infinite expansion in its terms has an important role in helping the neural network capture nonlinear interactions and this is the subject of subsection 4.9.

Finally, the hidden units are linearly combined in order to compute the final output of the neural network:

$$p(x) = \beta_0 + \sum_{i=1}^{n_h} \beta_i h_i, \tag{14}$$

where $p(x)$ is the premium computed by the neural network, $n_h$ is the number of hidden units and $\beta_0$ and $\beta_i$ are the slope intercept and the weights of the final linear combination.

Put all together in a single equation, we obtain:

$$p(x) = \beta_0 + \sum_{i=1}^{n_h} \beta_i \tanh\left(\alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} x_j\right). \tag{15}$$

Figure 9 depicts a smooth non-linear function which could be generated by a neural network.

The number of hidden units ($n_h$ above) plays a crucial role in our desire to control the capacity of the neural network. If we choose a too large value for $n_h$, then the number of parameters of the model increases and it becomes possible, during the parameter optimization phase, for the neural network to model noise or spurious relationships present in the data used for optimzation but that do not necessarily exist in other datasets. Conversely, if $n_h$ is set to a low value, the number of parameters could be too small and the neural network would not capture all of the relevant interactions in order to properly compute the premiums. Choosing the optimal number of hidden units is an important part of modelling with neural networks. Another technique for controlling the capacity of a neural network is to use weight decay, i.e., a penalized training criterion as described in subsection 2.3 that limits the size of the parameters of the neural network.

Choosing the optimal values for the parameters is a complex task and out of the scope of this paper. Many different optimization algorithms and refinements have been suggested (Bishop (1995), Orr and Müller (1998)) but in practice, the simple stochastic gradient descent algorithm is still very popular and usually gives good performance.

Note that like the linear regression, this model can potentially yield negative premiums in some cases. We have observed much fewer such cases than with the linear regression.

## 4.9 How can Neural Networks Represent Nonlinear Interactions?

For the more mathematically-minded readers, we present a simple explanation of why neural networks are able to represent **nonlinear interactions between the input variables.** To simplify, suppose that we have only two input variables, $x_1$ and $x_2$. In classical linear regression, a common trick is to include fixed nonlinear combinations among the regressors, such as $x_1^2, x_2^2, x_1x_2, x_1^2x_2, \ldots$ However, it is obvious that this approach adds exponentially many terms to the regression, as one seeks higher powers of the input variables.

In contrast, consider a single hidden unit of a neural network, connected to two inputs. The adjustable network parameters are named, for simplicity, $\alpha_0, \alpha_1$ and $\alpha_2$. A typical function computed by this unit is given by

$$\tanh(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2).$$

Here comes the central part of the argument: performing a Taylor series expansion of $\tanh(y + \alpha_0)$ in powers of $y$, and letting $\alpha_1 x_1 + \alpha_2 x_2$ stand for $y$, we obtain (where $\beta \equiv \tanh \alpha_0$),

$$\begin{aligned}
&\tanh(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2) = \\
&\quad \beta + (1-\beta^2)(\alpha_1 x_1 + \alpha_2 x_2) + (-\beta + \beta^3)(\alpha_1 x_1 + \alpha_2 x_2)^2 + \\
&\quad \left(-\tfrac{1}{3} + \tfrac{4\beta^2}{3} - \beta^4\right)(\alpha_1 x_1 + \alpha_2 x_2)^3 + \\
&\quad \left(\tfrac{2\beta}{3} - \tfrac{5\beta^3}{3} + \beta^5\right)(\alpha_1 x_1 + \alpha_2 x_2)^4 + O(\alpha_1 x_1 + \alpha_2 x_2)^5.
\end{aligned}$$

In fact the number of terms is infinite: the nonlinear function computed by this single hidden unit includes **all powers of the input variables**, but they cannot all be independently controlled. The terms that will ultimately stand out depend on the coefficients $\alpha_0, \alpha_1$, and $\alpha_2$. Adding more hidden units increases the flexibility of the overall function computed by the network: each unit is connected to the input variables with its own set of coefficients, thereby allowing the network to capture as many (nonlinear) relationships between the variables as the number of units allows.

The coeffients linking the input variables to the hidden units can also be interpreted in terms of **projections** of the input variables. Each set of coefficients for one unit represents a direction of interest in input space. The values of the coefficients are found during the **network training** phase using iterative nonlinear optimization algorithms.

## 4.10 Softplus Neural Network

This new type of model was introduced precisely to make sure that positive premiums are obtained. The softplus function was recently introduced in Dugas et al. (2001) as a means to model a convex relationship between an output and one of its inputs. We modified the neural network architecture and included a softplus unit as a final transfer function. Figure 10 illustrates this new architecture we have introduced for the purpose of computing insurance premiums. The corresponding formula is as such:

$$p(x) \;=\; F\left(\beta_0 + \sum_{i=1}^{n_h} \beta_i \, \tanh\left(\alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} x_j\right)\right), \qquad (16)$$

where $F(\cdot)$ is the softplus function which is actually simply the primitive (integral) function of the "sigmoid" function. Thus

$$F(y) \quad = \quad \log\left(1 + e^{y}\right). \tag{17}$$

The softplus function is convex and monotone increasing w.r.t. to its input and always strictly positive. Thus, as can be seen in Figure 11, this proposed architecture leads to strictly positive premiums.

In preliminary experiments we have also tried to use the exponential function (rather than the softplus function) as the final transfer function. However we obtained poor results due to difficulties in the optimization (probably due to the very large gradients obtained when the argument of the exponential is large).

The capacity of the softplus neural network is tuned just like that of an ordinary neural network. Note that this kind of neural network architecture is not available in commercial neural network packages.

## 4.11 Regression Support Vector Machine

Support Vector Machines (SVM) have recently been introduced as a very powerful set of non-parametric statistical learning algorithms (see Vapnik (1998a) and Schölkopf et al. (1998)). They have been very successful in classification tasks, but the framework has also been extended to perform regression. Like other *kernel methods* the class of functions has the following form:

$$p(x) \quad = \quad \sum_i \alpha_i K(x, x_i) \tag{18}$$

where $x_i$ is the input profile associated with one of the training records, and $\alpha_i$ is a scalar coefficient that is learned by the algorithm and $K$ is a *kernel function* that satisfies the Mercer condition (Cristianini and Shawe-Taylor (2000)):

$$\int_C K(x,y)\, g(x)\, g(y)\, dx\, dy \quad \geq \quad 0 \tag{19}$$

for any square integrable function $g(x)$ and compact subset $C$ of $\mathbf{R}^n$. This Mercer condition ensures that the kernel function can be represented as a simple dot product:

$$K(x,y) \quad = \quad \phi(x) \cdot \phi(y) \tag{20}$$

where $\phi()$ is a function that projects the input profile vector into a (usually very) high-dimensional "feature" space, usually in a nonlinear fashion. This leads us, to a simple expression for the premium function:

$$\begin{aligned} p(x) &= \sum_i \alpha_i \phi(x) \cdot \phi(x_i) \\ &= \phi(x) \cdot \left(\sum_i \alpha_i \phi(x_i)\right) \\ &= w \cdot \phi(x). \end{aligned} \tag{21}$$

Thus, in order to compute the premium, one needs to project input profile $x$ in its feature space and compute a dot product with vector $w$. This vector $w$ depends only on a certain number of input profiles from the training dataset and their associated coefficients. These input profiles are referred to as the *support vectors* and have been selected, along with their associated coefficients by the optimization algorithm.

SVMs have several very attractive theoretical properties, including the fact that an exact solution to the optimization problem of minimizing the training criterion can be found, and the capacity of the model is automatically determined from the training data. In many applications, we also find that most of the $\alpha_i$ coefficients are zero.

However, in the case of insurance data, an important characteristic of regression SVMs is that they are NOT trained to minimize the training MSE. Instead they minimize the following criterion:

$$J = \frac{1}{2}||w||^2 + \lambda \sum_i |a_t - p(x_i)|_\epsilon \tag{22}$$

where $|e|_\epsilon = \max(0, |e| - \epsilon)$, $\lambda$ and $\epsilon$ trade-off accuracy with complexity, $a_i$ is the observed incurred claim amount for record $i$, $x_i$ is the input profile for record $i$, and the vector $w$ is defined in terms of the $\alpha_i$ coefficients above. It can therefore be seen that this algorithm minimizes something close to the *absolute value of the error* rather than the *squared error*. As a consequence, the SVM tends to find a solution that is close to the *conditional median* rather than the *conditional expectation*, the latter being what we want to evaluate in order to set the proper value for a premium. Furthermore, note that the insurance data display a highly asymmetric distribution, so the median and the mean are very different. In fact, the conditional median is often exactly zero. Capacity is controlled through the $\epsilon$ and $\lambda$ coefficients.

## 4.12 Mixture Models

The *mixture of experts* has been proposed Jacobs et al. (1991) in the statistical learning litterature in order to decompose the learning problem, and it can be applied to regression as well as classification. The conditional expectation is expressed as a linear combination of the predictions of **expert models**, with weights determined by a **gater model**. The *experts* are specialized predictors that each estimate the pure premium for insureds that belong to a certain class. The *gater* attempts to predict to which class each insured belongs, with an estimator of the conditional probability of the class given the insured's input profile. For a mixture model, the premium can be expressed as

$$p(x) = \sum_c p(c|x)p_c(x) \tag{23}$$

where $p(c|x)$ is the probability that an insured with input profile $x$ belongs to class $c$. This value is determined by the gater model. Also, $p_c(x)$ is the premium, as computed by the expert model of class $c$, associated to input profile $x$.

A trivial case occurs when the class $c$ is deterministically found for any particular input profile $x$. In that case, we simply split the training database and train each expert model on a subset of the data. The gater then simply assigns a value of $p_c(x) = 1$ if $c$ is the appropriate

| Model | Train MSE | Valid MSE | Test MSE |
|---|---|---|---|
| Constant | 56.1108 | 56.5744 | 67.1192 |
| Linear | 56.0780 | 56.5463 | 67.0909 |
| GLM | 56.0762 | 56.5498 | 67.0926 |
| NN | 56.0706 | 56.5468 | 67.0903 |
| Softplus NN | 56.0704 | 56.5480 | 67.0918 |
| CHAID | 56.0917 | 56.5657 | 67.1078 |
| CondMean | 56.0827 | 56.5508 | 67.0964 |
| Mixture | 56.0743 | **56.5416** | **67.0851** |

Table 1: Comparison between the main models, with MSE on the training set, validation set, and test sets. The MSE is with respect to claim amounts and premiums expressed in thousand of dollars.

class for input profile $x$ and zero otherwise. This is in fact fundamentally equivalent to other techniques such as decision trees or table-based methods. A more general and powerful approach is to have the learning algorithm discover a relevant decomposition of the data into different regions of the input space which then become the classes and are encoded in the gater model. In that case, both the gater and the experts are trained together.

In this study both the experts and the gater are softplus neural networks, but any other model can be used. In Figure 12, we schematically illustrate a mixture model as the one that was used in the framework of this project.

## 5. Experimental Results

### 5.1 Mean-Squared Error Comparisons

Table 1 summarizes the main results concerning the comparison between different types of statistical machine learning algorithms. All the models have been trained using the same input profile variables. For each insurance policy, a total of 33 input variables were used and the total claims for an accident came from five main coverages: bodily injury, accident benefit, property damage, collision and comprehensive. Two other minor coverages were also included: death benefit and loss of use. In the table, *NN* stands for neural network, *GLM* for generalized linear model, and *CondMean* for the Greedy Multiplicative Model. The MSE on the training set, validation set and test set are shown for all models. The MSE is with respect to claim amounts and premiums expressed in **thousand of dollars**. The model with the lowest MSE is the "Mixture model", and it is the model that has been selected for the comparisons with the insurer's current rules for determining insurance premiums to which we shall refer as the *Rule-Based Model*.

One may wonder from the previous table why the MSE values are so similar across various models for each dataset and much different across the datasets. In particular, all models perform much worse on the testset (in terms of their MSE). There is a very simple explanation. The maximum incurred amount on the test set and on the validation set is around 3 million dollars. If there was one more such large claim in the test set than in

| Model #1 | Model #2 | Mean | Standard Error | Z | p-value |
|---|---|---|---|---|---|
| Constant | Mixture | 3.40709e-02 | 3.32724e-03 | 10.240000 | **0** |
| Linear | Mixture | 5.82350e-03 | 1.32211e-03 | 4.404700 | **5.29653e-06** |
| GLM | Mixture | 7.54013e-03 | 1.15020e-03 | 6.555500 | **2.77278e-11** |
| NN | Mixture | 5.23885e-03 | 1.41112e-03 | 3.712540 | **1.02596e-04** |
| Softplus NN | Mixture | 6.71066e-03 | 1.09351e-03 | 6.136810 | **4.20977e-10** |
| CHAID | Mixture | 2.35891e-02 | 2.57762e-03 | 9.151520 | **0** |

Table 2: Statistical Comparison Between Different Learning Models and the Mixture Model. The p-value is for the null hypothesis of no difference between Model #1 and the best mixture model. Note that ALL differences are statistically significant.

the validation set, one would expect the test MSE (calculated for premiums and amounts in thousand of dollars) to be larger by about 7 (these are in units of squared thousand dollars). Thus a difference of 11 can easily be explained by a couple of large claims. This is a reflection of the very thick right-hand tail of the incurred amount distribution (whose standard deviation is only of about 8 thousand dollars). Conversely, this also explains why all MSE are very similar across models for one particular dataset. The MSE values are all mainly driven by very large claims which no model could reliably forecast (no model could lead the insurer to charge one million dollars to a particular insured!) Consequently, truly significant differences between model performances are shadowed by the effect of very large claims on the MSE values. Although the differences between model performance are relatively small, we shall see next that careful statistical analysis allows us to discover that some of them are significant.

Figure 13 illustrates graphically the results of the table, with the models ordered according to the validation set MSE. One should note that within each class of models the capacity is tuned according to the performance on the validation set. On the test and validation sets, the Mixture model dominates all the others. Then come the ordinary neural network, linear model, and softplus neural network. Only slightly worse are the GLM and CondMean (the Greedy Multiplicative model). CHAID fared poorly on this dataset. Note that the CHAID + linear model described in section 4.7 performed worse than ordinary CHAID. Finally, the constant model is shown as a baseline (since it corresponds to assigning the same premium to every 1-year policy). It is also interesting to note from the figure that the model with the lowest training MSE is not necessarily the best out-of-sample (on the validation or test sets). The SVM performance was appalling and is not shown here; it did much worse than the constant model, because it is aiming for the conditional median rather the conditional expectation, which are very different for this kind of data.

Table 2 shows a statistical analysis to determine whether the differences in MSE between the Mixture model and each of the other models are significant. The *Mean* column shows the difference in MSE with the Mixture model. The next column shows the *Standard Error* of that mean. Dividing the mean by the standard error gives *Z* in the next column. The last column gives the *p-value* of the null hypothesis according to which the true expected squared errors for both models are the same. Conventionally, a value below 5% or 1% is interpreted as indicating a significant difference between the two models. The p-values

| Model #1 | Model #2 | Mean | Standard Error | Z | p-value |
|---|---|---|---|---|---|
| Constant | CHAID | 1.04818e-02 | 2.62416e-03 | **3.994350** | **3.24368e-05** |
| CHAID | GLM | 1.60490e-02 | 2.15109e-03 | **7.460850** | **4.29823e-14** |
| GLM | Softplus NN | 8.29468e-04 | 8.94764e-04 | 0.927025 | 1.76957e-01 |
| Softplus NN | Linear | 8.87159e-04 | 1.08802e-03 | 0.815392 | 2.07424e-01 |
| Linear | NN | 5.84651e-04 | 1.33283e-03 | 0.438653 | 3.30457e-01 |
| NN | Mixture | 5.23885e-03 | 1.41112e-03 | **3.712540** | **1.02596e-04** |

Table 3: Statistical Comparison Between Pairs of Learning Models. Models are ordered from worst to best. The test is for comparing the sum of MSEs. The p-value is for the null hypothesis of no difference between Model #1 and Model #2.

and Z corresponding to significant differences are highlighted. Therefore the differences in performance between the mixture and the other models are all statistically significant. As mentionned above, the MSE values are very much affected by large claims. Does such a sensitivity to very large claims make statistical comparisons between models incorrect? No. Fortunately all the comparisons are performed on **paired data** (the squared error for each individual policy), which cancel out the effect of these very large claims (since, for these special cases, the squared error will be huge for all models and of very close magnitude)

Table 3 has similar columns, but it provides a comparison of pairs of models, where the pairs are consecutive models in the order of validation set MSE. What can be seen is that the ordinary neural network (NN) is significantly better than the linear model, but the latter, the softplus neural network and GLM are not statistically distinguishable. Finally GLM is significantly better than CHAID, which is significantly better than the constant model. Note that although the softplus neural network alone is not doing very well here, it is doing very well within the Mixture model (it is the most successful one as a component of the mixture). The reason may be that within the mixture, the parameter estimation for model of the low incurred amounts is not polluted by the very large incurred amounts (which are learned in a separate model).

## 5.2 Evaluating Model Fairness

Although measuring the predictive accuracy—as done with the MSE in the previous section—is a useful first step in comparing models, it tells only part of the story. A given model could appear significantly better than its competitors *when averaging over all customers*, and yet perform miserably when restricting attention to a subset of customers.

We consider a model to be *fair* if different cross-sections of the population are not significantly biased against, compared with the overall population. Model fairness implies that the average premiums within each sub-group should be statistically close to the average incurred amount within that sub-group.

Obviously, it is nearly impossible to correct for any imaginable bias since there are *many* different criteria to choose from in order to divide the population into subgroups; for instance, we could split according to any single variable (e.g. premium charged, gender, rate group, territory) but also *combinations of variables* (e.g. all combinations of gender and

|  | Mixture Model |  | Rule-Based Model |  |
|---|---|---|---|---|
|  | Low | High | Low | High |
| Subgroup 1 | 50.81 | 166.24 | 139.27 | 245.0145 |
| Subgroup 2 | 166.24 | 214.10 | 245.01 | 297.0435 |
| Subgroup 3 | 214.10 | 259.74 | 297.04 | 336.7524 |
| Subgroup 4 | 259.74 | 306.26 | 336.75 | 378.4123 |
| Subgroup 5 | 306.27 | 357.18 | 378.41 | 417.5794 |
| Subgroup 6 | 357.18 | 415.93 | 417.58 | 460.2658 |
| Subgroup 7 | 415.93 | 490.34 | 460.26 | 507.0753 |
| Subgroup 8 | 490.35 | 597.14 | 507.07 | 554.2909 |
| Subgroup 9 | 597.14 | 783.90 | 554.29 | 617.1175 |
| Subgroup 10 | 783.90 | 4296.78 | 617.14 | 3095.7861 |

Table 4: Subgroups used for evaluating model fairness, for the Mixture and Rule-Based Models. The lowest and highest premiums in the subgroups are given. Each subgroup contains the same number of observations, $\approx 28,000$.

territory, etc.). Ultimately, by combining enough variables, we end up identifying individual customers, and give up any hope of statistical reliability.

As a first step towards validating models and ensuring fairness, we choose the subgroups corresponding to the location of the deciles of the premium distribution. The $i$-th decile of a distribution is the point immediately above $10i\%$ of the individuals of the population. For example, the 9-th decile is the point such that 90% of the population come below it. In other words, the first subgroup contains the 10% of the customers who are given the lowest premiums by the model, the second subgroup contains the range 10%–20%, and so on.

The subgroups corresponding to the Mixture Model (the proposed model) differ slightly from those in the *Rule-Based Model* (the insurer's current rules for determining insurance premiums). Since the premium distribution for both models is not the same. The subgroups used for evaluating each model are given in Table 4. Since they correspond to the deciles of a distribution, all the subgroups contain approximately the same number of observations ($\approx 28,000$ on the 1998 test set).

The bias within each subgroup appears in Figure 14. It shows the average difference between the premiums and the incurred amounts, within each subgroup (recall that the subgroups are divided according to the premiums charged by each model, as per Table 4). A positive difference implies that the average premium within a subgroup is higher than the average incurred amount within the same subgroup. 95% confidence intervals on the mean difference are also given, to assess the statistical significance of the results.

Since subgroups for the two models do not exactly represent the same customers, we shall refrain from directly comparing the two models on a given subgroup. We note the following points:

- For most subgroups, the two models are being fair: the bias is usually not statistically significantly different from zero.

- More rarely, the bias is significantly positive (the models overcharge), but never significantly negative (models undercharge).
- The only subgroup for which both models undercharge is that of the highest-paying customers, the 10-th subgroup. This can be understood, as these customers represent the highest risk; a high degree of uncertainty is associated with them. This uncertainty is reflected in the huge confidence intervals on the mean difference, wide enough not to make the bias significantly different from zero in both cases. (The bias for the Rule-Based Model is nearly significant.)

From these results, we conclude that both models are usually fair to customers in all premium subgroups. A different type of analysis could also be pursued, asking a different question: "In which cases do the Mixture and the Rule-Based Models differ the most?" We address this issue in next section.

### 5.3 Comparison with Current Premiums

For this comparison, we used the best (on the validation set) *Mixture model* and compare it on the test data of 1998 against the insurer's Rule-Based Model. Note that for legislative reasons, the Rule-Based Model did not use the same variables as the proposed Mixture Model.

Histograms comparing the distribution of the premiums between the Rule-Based and the Mixture models appear in Figure 15. We observe that the premiums from the Mixture model is smoother and exhibits fatter tails (more probability mass in the right-hand side of the distribution, far from the mean). The Mixture model is better able to recognize risky customers and impose an appropriately-priced premium.

This observation is confirmed by looking at the distribution of the *premium difference* between the Rule-Based and Mixture models, as shown in Figure 16.

We note that this distribution is extremely skewed to the left. This means that for some customers, the Rule-Based model considerably under-charges with respect to the Mixture model. Yet, the median of the distribution is above zero, meaning that the *typical customer pays more under the Rule-Based model than under the Mixture model.* At the same time, the Mixture model achieves better prediction accuracy, as measured by the *Mean-Squared Error* (MSE) of the respective models, all the while remaining fair to customers in all categories.

Our overriding conclusion can be stated plainly: the Mixture model correctly charges less for typical customers, and correctly charges more for the "risky" ones. This may be due in part to the use of more variables, and in part to the use of a statistical learning algorithm which is better suited to capturing the dependencies between many variables.

## 6. Taking Advantage of Increased Discriminant Power

Neural networks have been known to perform well in tasks where discrimination is an important aspect of the task at hand and this has lead to many commercially successful application of these modelling tools (Keller (1997)). We have shown that, when applied properly while taking into account the particulars of insurance data, that ability to discriminate is also revealed with insurance data. When applied to automobile insurance ratemaking, they allow us to identify more precisely the true risk associated to each insured.

## 6.1 Application to Underwriting

Completely changing the rate structure of an insurer can be a costly enterprise, in particular when it involves significant changes in the computer systems handling transactions, or the relations with brokers. There are other applications of systems which improve the estimation of pure premium. In the States of Massachusetts, New Hampshire, North Carolina and the provinces of Québec and Ontario, improved discrimination can be used for the purpose of choosing the risks to be ceeded to the *risk-sharing pools* (actual terminology varies from one jurisdiction to another). According to these pool plans, an insurer can choose to ceed a portion of its book of business (5%-10%) to the pool by paying a portion of the gross premium that was charged to the insured. Then, in case an accident occurs, the pool assumes all claim payments. The losses in the pool are then shared between the insurers. Thus, for an insurer, the goal is to identify the risks that have been underpriced the most (i.e. those for which the difference between the true risk and the current premium is largest). There are a few reasons why such inadequately rated risks can be identified:

- legislation related to ratemaking could be more restrictive than the one that pertains to the risk-sharing pool,
- strategic marketing concerns may have forced the insurer to underprice a certain part of its book of business and,
- other concerns may not allow the insurer to use highly discriminative models for the purpose of ratemaking.

Better discrimination of risks can be used to identify, with higher confidence, the worst risks in a population and therefore improve the performance of an insurance company's underwriting team.

## 6.2 Application to Ratemaking and Marketing

The greatest benefit from an improved estimation of pure premium derives by considering its application to ratemaking. The main reason for these benefits is that a more discriminant predictor will identify a group of insureds that are significantly undercharged and a (much larger) group that is significantly overcharged. Identifying the *undercharged* will yield **increased profits**: increasing their premiums will either directly increase revenues (if they stay) or reduce underwriting losses (if they switch to another insurer). The advantage of identifying the insured profiles which correspond to *overcharged* premiums can be coupled with a marketing strategy in order attract new customers and **increase market share**, a very powerful engine for increased profitability of the insurer (because of the fixed costs being shared by a larger number of insureds).

To decide on the appropriate change in premium, one also needs to consider market effects. An elasticity model can be independently developped in order to characterize the relation between premium change and the probability of losing current customers or acquiring new customers. A pure premium model such as the one described in this paper can then be combined with the elasticity model, as well as pricing constraints (e.g. to prevent too much rate dislocation in premiums, or to satisfy some jurisdiction's regulations), in order to obtain a function that "optimally" chooses for each insured profile an appropriate change in gross premium, in order to maximize a financial criterion. We have successfully tested such an idea and the detailed analysis of these results will be the subject of a further paper.

## 7. Conclusion

In this paper, we have argued in favor of the use of statistical learning algorithms such as neural networks for automobile insurance ratemaking. We have described various candidate models and compared them qualitatively and numerically. We have found that the selected model has significantly outperformed all other models, including the current premium structure. We believe that their superior performance is mainly due to their ability to capture high-order dependencies between variables and to cope with the fat tail distribution of the claims. Other industries have adopted statistical learning algorithms in the last decade and we have shown them to be suited for the automobile insurance industry as well.

## Appendix A. Proof of the equivalence of the fairness and precision criterions

In this section, we show that, when all subpopulations are considered to evaluate fairness, the precision criterion and the fairness criterion, as they were defined in section 2, both lead to the same premium function.

**Theorem 1** *The premium function which maximizes precision (in the sense of equation 2) also maximizes fairness (in the sense of equation 5, when all subpopulations are considered), and it is the only one that does maximize it.*

**Proof:**

Let $P$ be a subset of the domain of input profiles. Let $q$ be a premium predictor function. The bias in $P$ is defined by

$$b_q(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} (q(x_i) - a_i).$$

Let $F_q = -E[\sum_P b_q(P)^2]$ be the expected "fairness" criterion using premium function $q$, to be maximized (by choosing $q$ appropriately).

Let $p(x) = E[a|x]$ be the optimal solution to the precision criterion, i.e. the minimizer of

$$E[(p(X) - A)^2].$$

Consider a particular population $P$. Let $q(P)$ denote the average premium for that population using the premium function $q(x)$,

$$q(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} q(x_i)$$

and similarly, define $a(P)$ the average claim amount for that population,

$$a(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} a_i$$

Then the expected squared bias for that population, using the premium function $q$, is

$$E[b_q(P)^2] = E[(q(P) - a(P))^2]$$

which is minimized for any $q$ such that $q(P) = E[a(P)]$.

Note in particular that the optimal ESE solution, $p$, is such a minimizer of $F_q$, since

$$p(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} E[a_i|x_i] = E[\frac{1}{|P|} \sum_{(x_i,a_i)\in P} a_i] = E[a(P)]$$

We know therefore that $q = p$ is a minimizer of $F_q$, i.e. $\forall q, F_p \leq F_q$.

Are there other minimizers? Consider a function $q \neq p$, that is a minimizer for a particular population $P_1$. Since $q \neq p$, $\exists x$ s.t. $q(x) \neq p(x)$. Consider the particular singleton population $P_x = \{x\}$. On singleton populations, the expected squared bias is the same as the expected squared error. In fact, there is a component of $F$ which contains only the squared biases for the singleton populuations, and it is equal to the expected squared error. Therefore on that population (and any other singleton population for which $q \neq p$) there is only one minimizer of the expected squared bias, and it is the conditional expectation $p(x)$. So $E[(q(x)-A)^2|X = x] > E[(p(x)-A)^2|X = x]$ and therefore $E[b_q(P_x)] > E[b_p(P_x)]$. Since $p$ is a maximiser of fairness for all populations, it is enough to prove that $q$ is sub-optimal on one population to prove that the overall fairness of $q$ is less than that of $p$, which is the main statement of our theorem:

$$\forall q \neq p, F_q > F_p.$$

## References

R.A. Bailey and L. Simon. Two studies in automobile insurance ratemaking. *ASTIN Bulletin*, 1(4):192–217, 1960.

R.E. Bellman. *Dynamic Programming.* Princeton University Press, NJ, 1957.

Y. Bengio and F. Gingras. Recurrent neural networks for missing or asynchronous data. In M. Mozer, D.S. Touretzky, and M. Perrone, editors, *Advances in Neural Information Processing System*, volume 8, pages 395–401. MIT Press, Cambridge, MA, 1996.

C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

R.L. Brown. Minimum bias with generalized linear models. In *Proceedings of the Casualty Actuarial Society*, 1988.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge Press, 2000.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. A universal approximator of convex functions applied to option pricing. In *Advances in Neural Information Processing Systems*, volume 13, Denver, CO, 2001.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, San Mateo, CA, 1994. Morgan Kaufmann.

F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel. *Robust Statistics, The Approach based on Influence Functions.* John Wiley & Sons, 1986.

T. Hastie, R. Tibshirani, and J. Friedman. *Data Mining, Inference and Prediction.* Springer, 2001.

K.D. Holler, D. Sommer, and G. Trahair. Something old, something new in classification ratemaking with a novel use of glms for credit insurance. *Casualty Actuarial Society Forum*, pages 31–84, 1999.

P.J. Huber. *Robust Statistics.* John Wiley & Sons Inc., 1982.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3:79–87, 1991.

G.V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

Paul E. Keller. Neural networks: Commercial applications, 1997. http://www.emsl.pnl.gov:2080/proj/neuron/neural/products.

P. McCullagh and J. Nelder. *Generalized Linear Models.* Chapman and Hall, London, 1989.

K. Murphy, M.J. Brockman, and P.K.W. Lee. Using generalized linear models to build dynamic pricing systems. *Casualty Actuarial Society Forum*, pages 107–139, 2000.

G.B. Orr and K.-R. Müller. *Neural Networks: Tricks of the Trade.* Springer, 1998.

P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons Inc., 1987.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

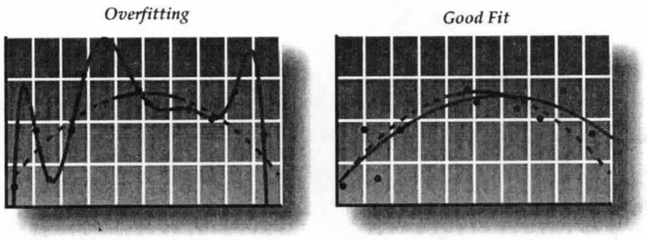
B. Schölkopf, C.J.C. Burges, and A.J. Smola. *Advances in kernel methods: support vector learning.* MIT Press, 1998.

I. Takeuchi, Y. Bengio, and T. Kanamori. Robust regression with asymmetric heavy-tail noise distributions. *Neural Computation*, 2002. to appear.

V. Vapnik. *Statistical Learning Theory.* John Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454, 1998a.

V. Vapnik. *Statistical Learning Theory.* Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454, 1998b.

Figure 1: Illustration of **overfitting**. The solid left curve fits the noise in the data points (black dots) and has not learned the underlying structure (dashed). The right curve, with less flexibility, does not overfit.



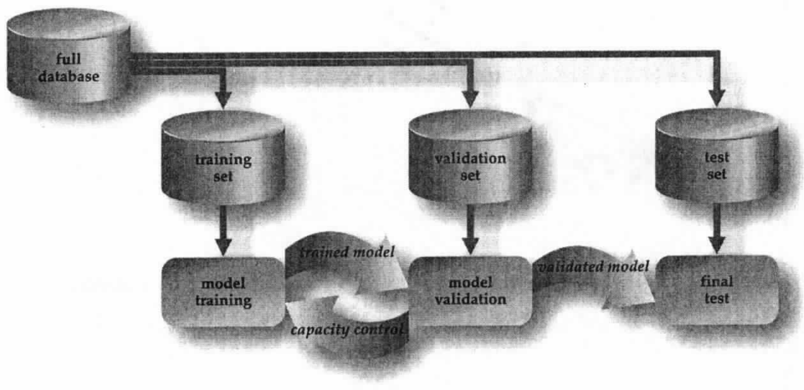


Figure 2: Methodology to prevent overfitting. Model capacity is controlled via a validation set, disjoint from the training set. The generalization performance estimator is obtained by final testing on the test set, disjoint from the first two.

Figure 3: The constant model fits the best horizontal line through the training data.

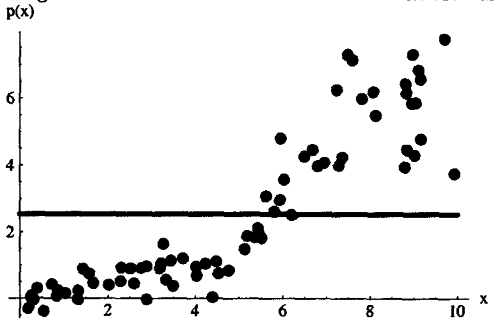


Figure 4: The linear model fits a straight line through the training data.

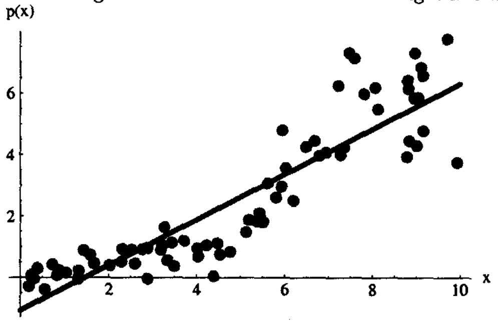


Figure 5: The generalized linear model fits an exponential of a linear transformation of the variables.

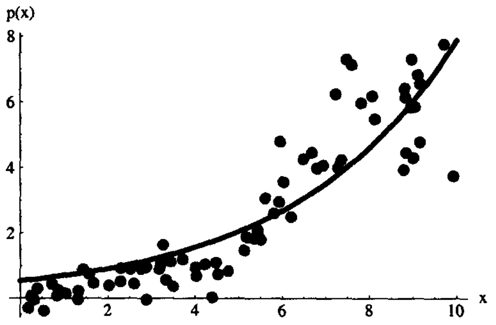


Figure 6: The CHAID model fits constants to partitions of the variables. The dashed lines in the figure delimit the partitions, and are found automatically by the CHAID algorithm.

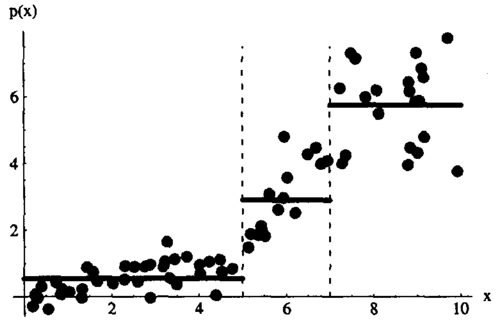


Figure 7: The CHAID+Linear model fits a straight line within each of the CHAID partitions of the variable space.

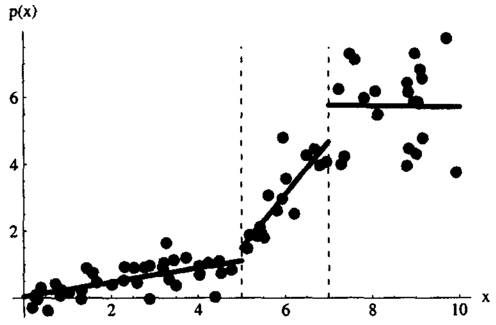


Figure 8: Topology of a one-hidden-layer neural network. In each unit of the hidden layer, the variables are linearly combined. The network then applies a non-linear transformation on those linear combinations. Finally, the resulting values of the hidden units are linearly combined in the output layer.

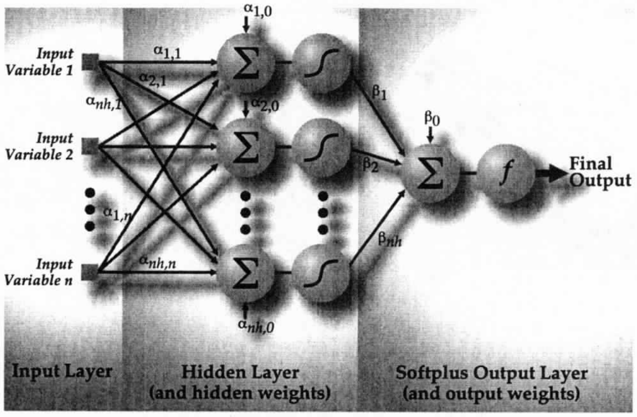


Figure 9: The neural network model learns a smooth non-linear function of the variables.

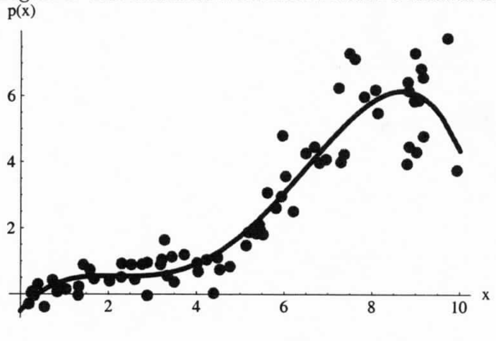


Figure 10: Topology of a one-hidden-layer softplus neural network. The hidden layer applies a non-linear transformation of the variables, whose results are linearly combined by the output layer. The softplus output function forces the function to be positive. To avoid cluttering, some weights linking the variables to the hidden layer are omitted on the figure.

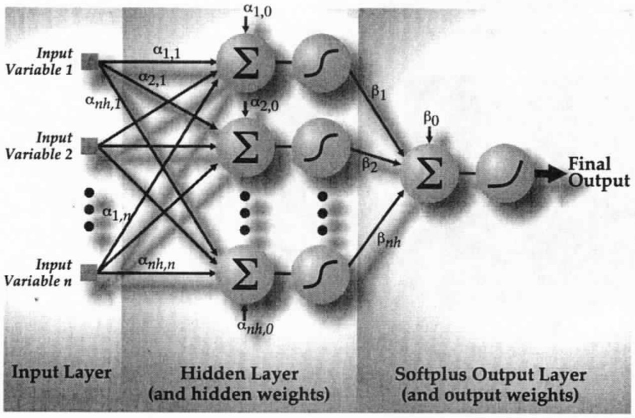


Figure 11: The softplus neural network model learns a smooth non-linear **positive** function of the variables. This positivity is desirable for estimating insurance premiums.

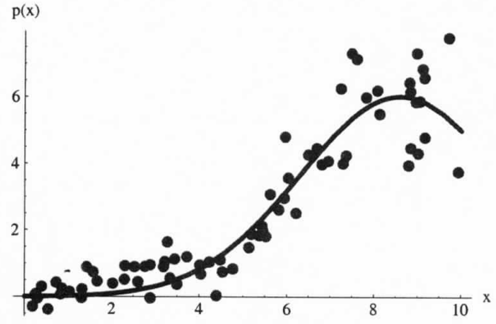


Figure 12: Schematic representation of the mixture model. The first-stage models each make an independent decision, which are linearly combined by a second-stage **gater**.

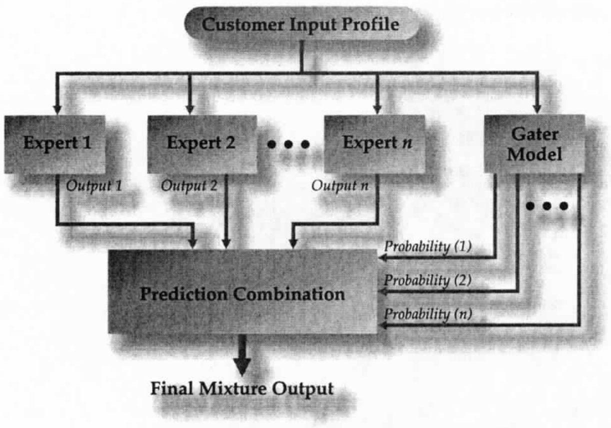
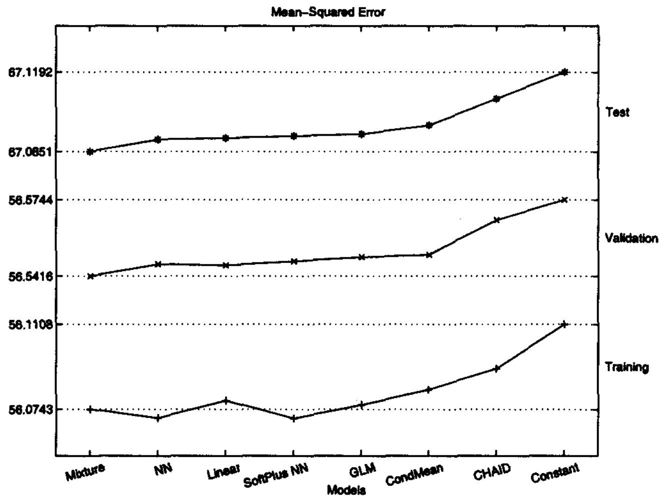


Figure 13: MSE results (from table 1) for eight models . Models have been sorted in ascending order of test results. The training, validation and test curves have been shifted closer together for visualization purposes. The out-of-sample test performance of the mixture model is significantly better than any of the other. Validation based model selection is confirmed on test results.



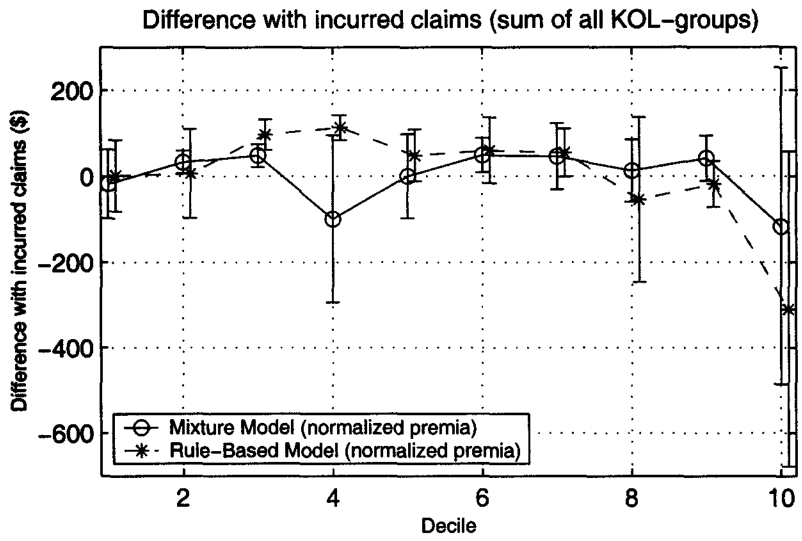


Figure 14: Average difference between premiums and incurred amounts (on the sum over all coverage groups), for the Mixture and Rule-Based models, for each decile of the models' respective premium distribution. We observe that both models are being fair to most customers, except those in the last decile, the highest-risk customers, where they appear to under-charge. The error bars represent 95% confidence intervals. (Each decile contains $\approx 28,000$ observations.)

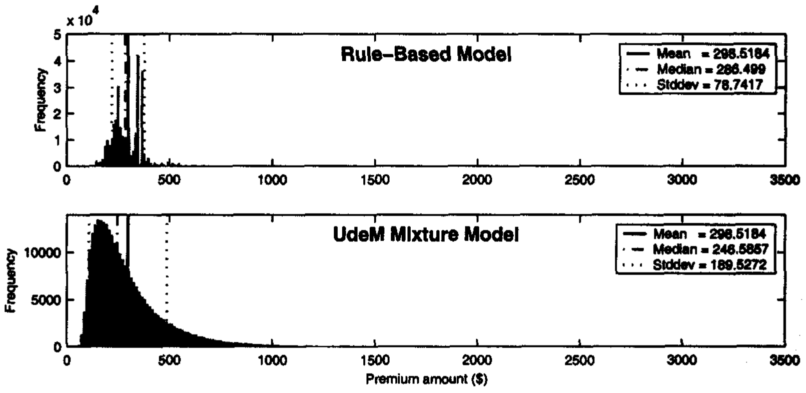
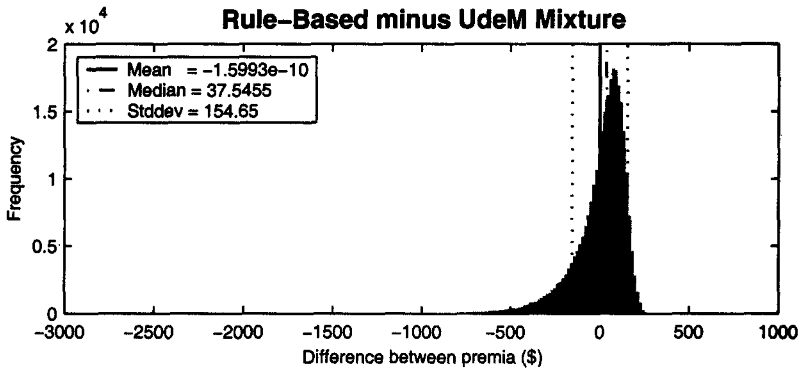


Figure 15: Comparison of the premium distribution for the current Rule-Based model and the Mixture model. The distributions are normalized to the same mean. The Mixture model distribution has fatter tails and is much smoother.

Figure 16: Distribution of the premium difference between the Rule-Based and Mixture models, for the sum of the first three coverage groups. The distribution is negatively skewed: the Rule-Based model severely under-charges for some customers.


Rule–Based minus UdeM Mixture
x 10^4
Mean = -1.5993e-10
Median = 37.5455
Stddev = 154.65
Frequency
Difference between premia ($)