Interactive Learning for Efficiently Detecting Errors in Insurance Claims

Rayid Ghani
Accenture Technology Labs
Chicago, IL, USA
rayid.ghani@gmail.com

Mohit Kumar
Accenture Technology Labs
Chicago, IL, USA
mohit.x.kumar@accenture.com

ABSTRACT

Many practical data mining systems such as those for fraud detection and surveillance deal with building classifiers that are not autonomous but part of a larger interactive system with an expert in the loop. The goal of these systems is not just to maximize the performance of the classifier but to make the experts more efficient at performing their task, thus maximizing the overall Return on Investment of the system. This paper describes an interactive system for detecting payment errors in insurance claims with claim auditors in the loop. We describe an interactive claims prioritization component that uses an online cost-sensitive learning approach (more-like-this) to make the system efficient. Our interactive prioritization component is built on top of a batch classifier that has been trained to detect payment errors in health insurance claims and optimizes the interaction between the classifier and the domain experts who are consuming the results of this system. The goal is to make these auditors more efficient and effective as well as improving the classification performance of the system. The result is both a reduction in time it takes for the auditors to review and label claims as well as improving the precision of the system in finding payment errors. We show results obtained from applying this system at two major US health insurance companies indicating significant reduction in claim audit costs and potential savings of $20-$26 million/year making the insurance providers more efficient and lowering their operating costs. Our system reduces the money being wasted by providers and insurers dealing with incorrectly processed claims and makes the healthcare system more efficient.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.4.2 [**Information Systems Applications**]: Types Of Systems—*Decision Support*

KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...$10.00.

General Terms

Algorithms, Measurement, Performance

Keywords

Interactive machine learning, Health insurance claims, cost sensitive learning, budgeted learning, error detection, quality control

1. INTRODUCTION

Many deployed data mining systems such as those for fraud detection and surveillance deal with building classifiers (or predictors) that are not deployed in isolation but are part of a larger interactive system with an expert in the loop. Typically, once the models are trained, these systems score a large amount of new data and present the experts with a ranked list of cases to review and verify. These experts are often expensive, and have limited time and attention. Fraud Detection, Intrusion Detection, Medical Diagnosis, Information Filtering, and Video Surveillance are some examples where these systems are currently being used to aid experts in finding cases of interest to them. All of these applications involve a limited number of labeled examples with a high cost of labeling, and a large number (millions) of unlabeled examples, with majority of them being negative (skewed class distribution). These kinds of applications have attracted a lot of research attention from the data mining community but typical data mining research in these application areas has focused on making the classifiers more accurate when in fact, the business goal of these systems is not just to maximize the performance of the classifier but to make the experts more efficient at performing their task.

This paper deals with the problem of predicting (and reducing) payment errors when processing health insurance claims. The goal is to minimize errors by predicting which claims are likely to have errors, and presenting the highly scored ones to human auditors so that they can be corrected before finalization and payment.

Reducing payment errors in health insurance claims has a direct impact on making healthcare more affordable for consumers. Health insurance costs across the world have increased alarmingly in recent years which is forcing insurance companies to reduce their administrative costs and decrease insurance premiums. The typical process for insured healthcare in the US is that a patient goes to a service provider (medical facility) for the necessary care and the provider files a claim with the patient's health insurance company for the services provided. The insurance company then pays the service provider based on multiple complex factors including eligibility of the patient at time of service, coverage of the procedures in the benefits, contract status with the provider etc. Payment errors made by insurance companies while processing claims often result in re-processing of the claim. This extra administrative work to re-process claims is known as *rework* and accounts for a significant portion of the administrative costs of health insurance providers. These errors also have a direct monetary impact in terms of the insurance company paying more or less than what it should have. [1] estimates from a large insurance plan covering 6 million members had $400 million in identified overpayments. In our discussions with major insurance companies, we have found that these errors can result in loss of revenue of up to $1 billion each year. In addition to the direct monetary impact, there is also an indirect monetary impact since employees need to be hired to *rework* the claim and answer phone calls regarding them. According to estimates by an Accenture study, 33% of the administrative workforce is directly or indirectly related to rework processing. These statistics make the problem of rework prevention extremely important and valuable to the healthcare industry and motivated the work described in this paper.

In an earlier paper [5], we described our *batch* system for predicting errors in insurance claims. When we deployed the batch system described in [5] and tested it with auditors from a large US health insurer, we got encouraging classification results but also noticed several problems that affected the efficiency of the auditors (and hence the overall system) when interacting with our rework prediction system.

Issue 1: Context Switching Costs: The auditors were provided with the ranked list of claims that were scored highly by the trained classifier. They started from the top of that list and worked their way down. One problem they experienced was high switching costs when moving from claim to claim. This switching cost arose from the fact that the auditors spent considerable time (up to 20 minutes) reviewing a claim to figure out whether it was an error or a correctly processed claim. Once they had investigated that claim, they were given the next claim (from the ranked list) that was often completely different (both in terms of the *reason* for flagging and in terms of the procedures or diagnosis) from the previous one. At a later stage, they would be given a claim that was lower down in the ranked list but was scored highly by the classifier for *similar* reasons as an earlier claim. Unfortunately, by that time, they have already gone through enough different claims that they need to redo the investigation and the time it takes them to label it is the same as the earlier ones.

Issue 2: Intuitive Explanations of model predictions for each claim: The interface we designed for the auditors highlighted data fields in the claim form that had high scores using the hyperplane that was learned by our classifier (SVM). Even though the weights learned by the SVM result in a classifier that provides high classification accuracy, they were not necessarily intuitive (and accurate) as an explanation for the auditors.

Both of these issues stemmed from the fact that our system was designed to optimize the classification performance of the system, and not necessarily to optimize the return on investment of the overall system which includes the auditors in the loop. We were using classification accuracy (in the top n% scored claims) to measure the overall performance of the system instead of the business metric of interest which should have been related to the cost savings due to the more efficient use of the auditors. This led us to the real metric that needed to be maximized in our system (and many other similar data mining systems) - that is the number of incorrectly paid claims (both in terms of volume and dollar spend) prevented per dollar of audit spend (the ROI). In fact, it is certainly possible to have a system with lower overall classification accuracy but higher ROI. One way of achieving that is to decrease the time it would take to audit a claim. This could be done by generating intuitive explanations that make it faster (hence cheaper) to audit claims, or by ordering the scored claims so the next claim audited is not necessarily the next most likely erroneous claim but one that is similar to the previous one (possibly with a lower score) and faster to audit. Both of these approaches would result in more claims audited in a given amount of time which could result in a higher overall ROI for the system.

Taking these issues into account, we focused on building an interactive claims prioritization component that makes our batch system more efficient using an online cost-sensitive learning approach (more-like-this). Our interactive prioritization component is built on top of a batch classifier that has been trained to detect payment errors in health insurance claims and optimizes the interaction between the classifier and the auditors who are consuming the results of this system. The goal is to make these auditors more efficient and effective as well as improving the observed classification performance of the system. The result is both a reduction in time it takes for the auditors to review and label claims as well as improving the precision of the system in finding payment errors.

We show results obtained from applying this system at two major US health insurance companies indicating significant reduction in claim audit costs and multi-million dollar savings making the insurance providers more efficient and lowering their operating costs. The Rework Prevention Tool that is the basis of our work in this paper is part of Accenture's Claims Administration Offering which is currently deployed at many of the larger health insurance companies in the US. The results so far show potential savings of over $20-$26 million each year for a typical insurer. This reduces the money being wasted by providers and insurers dealing with incorrectly processed claims and makes the healthcare system more efficient. Although our work is specific to insurance claims, our interactive prioritization component is designed to be generic enough to take as input the results of any batch classifier and re-prioritize the results in order to optimize the interaction between the classifier and the domain experts, thus resulting in a higher ROI.

2. RELATED WORK

There are some vendors such as Enkata that provide tools to analyze past claims in order to discover root causes for claims rework but these tools do not predict future rework using data mining techniques and help prevent errors automatically. Typically, these tools help with understanding the factors correlated with historical rework and require managers to fix the back-end system.

In general, claims processing is not an area where data mining techniques have been widely used. A lot of the work in claims has focused on fraud detection which is a related but different area. [4] mentioned a system for detecting health-care provider fraud in electronically submitted claims developed at Travelers Insurance. Health claim fraud is very different from Rework identification because the characteristics of fraudulent claims and Rework claims are different. Often, there is not much labeled data available for fraud and the magnitude of fraud is also typically smaller. Similarly, related work in the area of credit card fraud [3] is also different from Rework identification as the data characteristics and expectations are different. There has been some work in the area of claim overpayment identification [1] which is a subproblem of Rework identification as Rework constitutes both overpayment and underpayment of claims. Also the approach taken in the system is dependent on client-specific 'edits' or rule-based scenarios by training the models for each scenario. This makes the approach less generalizable as it requires the designing of these client scenarios or rules.

Tools that are available for predicting errors in claims and more generally for interactive data mining tasks (such as fraud detection, intrusion detection, surveillance, medical diagnosis) focus on maximizing the classification performance instead of maximizing the ROI of the overall system (that includes domain experts in the loop).

The research community has focused on some aspects of tasks which have experts in the loop. There has been research work in the areas of active learning and cost-sensitive active learning [17], [8], [12] where the aim is to minimize the overall cost of training an accurate model, recognizing that the annotation costs are not constant across instances, and can vary considerably. A related area is active feature acquisition where the goal is to select the most informative features to obtain during training [14]. Budgeted learning [6], [10] where the goal is to learn the most accurate 'active classifier' based on a fixed learning budget for acquiring training data which is also related to the idea of Exploitation/Exploration from Reinforcement Learning. [16] provides an excellent survey of the latest advancements in the field of Active learning. None of the research work we are aware of has focused on the task we are tackling, that is taking the results of a batch classifier and of post-processing classification results to optimize the efficiency of domain experts in reviewing these results.

3. BACKGROUND

We give a brief overview of our batch system here and more details can be found in [5]. We formulate the problem of rework prediction as an interactive classification problem and produce a ranked list of claims that need to be manually reviewed. Figure 1 gives a generalized view of the claims processing pipeline. Claims are created by service providers and submitted to the insurance company. They go through automatic validation checks and then the appropriate pricing is applied to the claims using benefit rules and contracts. This takes place automatically in some cases and in other cases, manual intervention is required. Once the pricing is applied, claims get finalized and payment is sent to the service provider. The system we describe in this paper is the module placed after the pricing is applied to detect potential issues with the claim before it is finalized so it can be corrected before payment is sent.

3.1 System Overview

We worked with domain experts to come up with requirements that would make a rework prediction solution practical and useful for insurance companies. Our system consists of the following components: Data Collection, Feature Construction, Model Learning and Selection, Item Scoring, Explanation Generation, User Feedback, and User Interface. We only give a brief description of our baseline system here since details are given in [5]. We obtain data from health insurance companies along with the labels: **rework** or **correct**. The claims assigned the label *rework* are those that were manually examined in the past and contained errors. Those assigned the label *correct* are ones that were manually examined in the past and found to be correct. Once the data is collected and labeled, feature construction is the next step that takes place. There are four classes of information in each claim: Member information, Provider information, Claim Header, and Claim Line Details. Our features are extracted from these classes of information. Overall, we end up with approximately 15,000 categorical and numerical features to build our models.

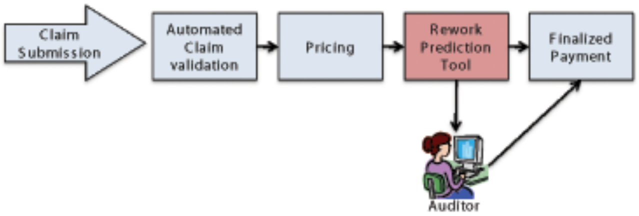

Figure 1: Claim Processing Pipeline

We chose SVMs as the main learning algorithms after experimenting with other classifiers such as decision trees and logistic regression. We also experimented with different versions of SVMs including SVMperf [9], Pegasos, LibSVM [2], and Sofia [15]. Since we have more than 238,000 features after creating binary features from the categorical data, we experimented with frequency-based feature selection techniques which made our system more efficient both in terms of execution time and storage requirements.

Accurately identifying rework and presenting it to auditors is one aspect of our system. Another equally important aspect is to give the auditors the ability to quickly determine if the claim being presented to them is rework or not. Currently, auditors in most companies are given a claim without any hints about why a claim could be rework. In our experiences working with auditors at several companies, a full audit of a claim without any hints as to why a claim is being flagged as rework could take up to an hour. We want to *explain* the predictions of the classifier to the auditor to reduce the time it takes them to determine the label of the claim. In our case, we use the feature weights learned by the SVM to generate the explanations. [13] have also used a similar approach to perform feature selection by retaining features with high weights.

We also get feedback from auditors to improve future predictions. We obtain claim level feedback (labels of examples) as well as feature-level feedback using the user interface we have developed. We worked with teams of auditors to design an interface that was similar to what auditors typically use (a claim form) when auditing the claim but enhanced it with some new features. Figure 2 shows a screenshot of the user interface for the auditors. This interface provides the auditors with a list of claims in their work queue with a rework likelihood score assigned to each claim that is generated by the classifier. In addition, we use the influence

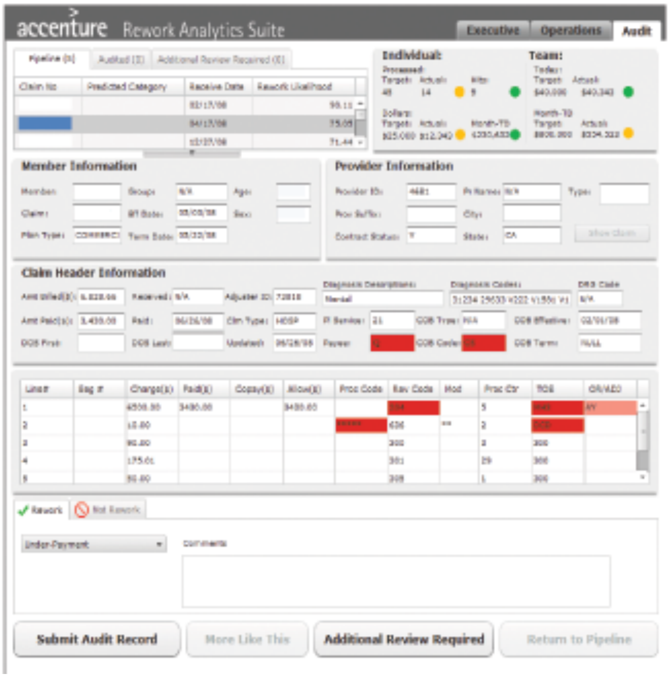

Figure 2: User Interface with the Auditor view (sensitive PHI fields are blanked out).

scores described earlier to highlight the data fields. The intensity of the highlight color is proportional to the influence score assigned by the classifier. This feature has been designed to focus the attention of the auditor to the influential data fields and help them audit the claim faster. We also allow the auditors to click on any field to *unhighlight* it (if it's highlighted) or to highlight it and give us feedback that field is influential (or not).

4. ONLINE COST-SENSITIVE LEARNING

When we did a pilot deployment of the system described in the previous section at a large US health insurance company, we got encouraging results but also noticed several problems that affected the efficiency of the auditors when interacting with our rework prediction system.

Context Switching Costs: We used the confidence of the SVM classifier to rank new claims. The auditors were then provided with the ranked list of claims and started from the top of that list and worked their way down auditing the claims. One problem they experienced was high switching costs when moving from example to example. This switching cost arose from the fact that the auditors spent considerable time (up to 20 minutes) reviewing a claim to figure out whether it was a positive or a negative example. Once they had investigated (and labeled) that claim, they were given the next claim (from the ranked list using the SVM confidence scores) that was often completely different (in terms of the *reason* for flagging) from the previous one. At a later stage, they would be given a claim that was lower down in the ranked list but was scored highly by the classifier for *similar* reasons as an earlier claim. Unfortunately, by that time, they have already gone through enough different claims that they need to redo the investigation and the time it takes them to label it is the same as the earlier ones. We hypothesize that if we can group similar claims together, it would reduce the *context switching* and help make the reviewing times shorter.

Low Precision when reviewing Top 2% scored claims: Based on the class distribution, claim volumes, auditor workload, and audit times, we determined that it's optimal to manually review top 2% of the claims that are scored by our classifier, making precision at 2% the metric we optimize[1]. The scores assigned by the SVM to the top few percent of the claims are fairly similar which results in poor ranking at the top of this list. Also, based on the feedback we got from the auditors, we found that as they went down the list reviewing the claims, they encountered several series of consecutive claims that were all negative examples (and often for similar reasons). We hypothesize that this was because the ranking was purely based on SVM scores and did not have any notion of diversity. Again, grouping similar claims together could help group a lot of these negative examples together and improve the performance of the human auditors.

Intuitive Explanations: The interface we designed for the auditors highlight data fields in the claim form that had high scores using the hyperplane that was learned by the SVM. Even though the weights learned by the SVM result in a classifier that provides high classification accuracy, they were not necessarily intuitive as an explanation for the auditors.

Batch Retraining/Learning: Because we are dealing with a system that is interactive with costly human auditors in the loop, we do not have the time to retrain the system (in real-time) after each claim is reviewed by the auditors. The long wait time that is required for retraining makes the process too expensive since the auditors have to be idle during that time. This situation forces us to deal with the tradeoff between potentially better results (by retraining after every new label) and maximizing the use of the auditor time (by retraining after a certain number of new labels have been acquired).

4.1 Interactive Claims Prioritization using Online Cost-Sensitive Learning

The four issues described above caused us to think about approaches that could take the results of the classifier and process them in order to improve the overall ROI of the system. That formed the basis of our Claims Prioritization strategy that uses online cost-sensitive learning in order to maximize the efficiency of the auditors while helping them achieve their goal of finding errors in insurance claims. The efficiency of the auditors is not just a function of the accuracy of the system but also the time it takes them to review a claim. Reducing the time it takes to review an example (claim) is an understudied area which is not often discussed in KDD research literature and is important when dealing with interactive business applications where the experts are expensive and have limited time. We describe our online cost-sensitive learning framework that focuses on reducing the overall time experts spend on labeling (verifying or correcting) examples provided to them by a classifier. We deal with the scenario where a batch classifier has been trained to predict errors in claims and focus on optimizing the interaction between the classifier and the auditors who are

[1] More details on that are in [5]

consuming the results of this predictor. The goal is to make these auditors more efficient and effective in performing their task as well as eventually improving the system over time.

4.1.1 More-Like-This Strategy

Our *More-Like-This* (MLT) strategy (Algorithm 1) is motivated by the context switching costs, low precision, and explanation issues described above. The goal of this strategy is to group the scored claims into clusters that contain claims that have been scored highly for *similar* underlying reasons. This allows us to provide the auditors with a cluster of claims that are both highly likely to be *rework* (errors) and can also be audited very quickly with small incremental audit costs. The cost gets reduced because once the auditors have invested the effort in reviewing the first couple of claims in a cluster, the rest of the claims will have similar labels due to similar reasons. At the same time, this technique also allows us to ignore a large number of claims that are similar to each other and are in fact correct claims (negative examples) but have been incorrectly ranked highly by the classifier. This is done by our online strategy where we rank the clusters using a variety of metrics (described later) and give the auditors a few claims from a cluster. If the claims are verified by the auditors as being errors, we continue giving them claims from that cluster. Otherwise, we move on the next cluster thus avoiding a large number of highly scored claims that were in fact negative. This strategy also implicitly provides the auditors with an *explanation* for the classification by providing them with other claims that are similar to what they have just labeled. MLT works as a post-processing step on new (unlabeled) claims that have already been scored by the trained classifier.

Algorithm 1: Online More-Like-This Algorithm

```
Require: a labeled set of claims $L$ and an unlabeled set $U$
1. Train classifier $C$ on $L$
2. Label $U$ using classifier $C$
3. Select the top m% scored unlabeled claims $U_T$ (we use m=10 in our case)
4. Cluster the examples $U_T \bigcup L$ into $k$ clusters
5. Rank the $k$ clusters using a cluster-based ranking function (described in the next section)
6. For each cluster in the ranked cluster list
    (a) Rank the claims within the cluster based on their informativeness of the quality of the cluster
    (b) For each claim in the ranked list within the cluster
        i. Query human auditor for the label of the claim
        ii. Move to the next cluster if the evaluation metric (we use precision in our experiments) calculated over claims labeled so far in the cluster falls below threshold $P$
```

The intuition behind our MLT approach is to use a supervised classifier like SVM to initially provide a rough partitioning of the scored examples and then use a clustering algorithm to do more finer-grained partitions. The function of clustering is to take the examples that are scored highly by the classifier and group similar ones together, hopefully creating clusters that correspond to claims that have similar underlying *reasons* for being errors. Our two-step approach of post-ranking (or classification) clustering is also potentially useful in cases where there are different underlying subgroups (or reasons) in the positive class that may be disjoint which could create issues for a binary classifier (unless multiclass labels are available).

In Step 3, we use m=10% for our experiments which selects the top 10% of the claims. Setting m=100% would be equivalent to clustering all the examples without taking into account the classifier scores. Intuitively, we would choose m based on the class priors and the budget available for reviewing the claims.

In Step 4, we vary k (number of clusters) from 200 to 2800 and settle for $|U|/100$. The higher the number of clusters, the more coherent/similar each group will be which can decrease the audit time and provide better explanations but increases the number of clusters to examine. Another tradeoff is that the larger the number of clusters the test examples may be split in different clusters thus not resulting in reduction of context-switch cost. We use Cluto[11] to perform the clustering.

In Step 5, we experiment with a variety of metrics to rank the clusters described in the next section. These ranking metrics are a function of the number of positive examples (from L in the cluster), the mean score assigned by the classifier to the unlabeled examples U in this cluster, the ratio of the positive to negative labeled examples (from L), the inter-cluster similarity score of the cluster, and the intra-cluster similarity of the cluster.

To rank the claims for review within a cluster (in Step 6a), we use the score assigned by the classifier as the metric in our experiments. This is just one example of a metric that can be used. Other metrics we plan to experiment with include distance from the centroid of the cluster (to select prototypical examples from a cluster first), and sample selection metrics from active learning literature (based on uncertainty sampling for example). In Step 6b(ii), we use precision as the metric and set the threshold to 30%. This is chosen empirically based on the baseline precision scores and the class distribution.

4.1.2 Pool-based Active Learning

The *More-Like-This* (MLT) strategy is designed to reduce the time it takes to review (or label) a claim and also to provide the auditors with immediate rewards. We also use active learning strategies to improve the future performance of the system. We use the standard pool-based active learning setting and use different sample selection strategies such as density-based and uncertainty sampling. This approach is potentially not immediately rewarding to the human auditors and focuses on future improvement of the classifier.

For density-based sampling, we calculate cosine similarity between each pair of claims and use the average similarity of each claim to all the other claims as a measure of density. For uncertainty sampling, we use the absolute value of the SVM score (distance from the SVM margin) as a measure of uncertainty.

4.1.3 Hybrid Active Learning with More-Like-This

The two strategies mentioned above have potentially competing goals. More-Like-This is designed to be of immediate benefit to the auditors by giving them claims that are highly scored and quick to audit, thus increasing their short-term

ROI. The Active Learning strategies are designed to maximize future improvement of the system and not necessarily providing claims to the auditors (in the short-term) that would be errors. In fact, the uncertainly sampling strategy is designed to present claims to the auditor for which the classifier is least confident about with the promise of improvement in the future. The tradeoff between the two previous strategies motivates a strategy that combines the benefits of both of them. We propose three different approaches to implement this hybrid strategy.

MLT-then-Active: This strategy performs MLT clustering as described in Section 4.1.1 but then uses active learning (with different sample selection metrics) to select examples to provide to the user for review and labeling.

Hybrid Sample Scoring: Hybrid Sample Scoring strategy assigns a score to every unlabeled example based on both strategies. Each example gets a score using active learning (which indicates it's utility in improving future classifier performance) as well as with the More-Like-This strategy (which indicates it's immediate utility). We can then use a weighted combination of these scores to create a final ranking.

Online Hybrid Learning: This hybrid strategy treats the two strategies independently and alternates among them based on the recent past, availability of expert resources, and business needs.

We have discussed these strategies with domain experts in claims processing and have found them to be quite receptive to the last strategy as a tradeoff. In the results with the pilot deployments presented in this paper, we don't use any hybrid strategies but we believe that these can be highly effective and are currently running experiments to evaluate their suitability in practical situations.

5. EXPERIMENTAL RESULTS

5.1 Data and Experimental Setup

We present results from two pilots with major US health insurance companies that were conducted using the system described in this paper during the past year. From the first company, we started with approximately 25 million claims spanning 2 years. Of these 25 million claims, 379,000 had been manually labeled through various audit processes. In this labeled data set, around 65% claims were Rework (errors) and the rest were Correct. It is important to remember that this distribution does not hold in the operational world since the labeled data has a skewed class distribution. For the second company, we got approximately 40 million claims spanning 12 months. Of these 40 million claims, 150000 had been manually audited and found to be either Rework or not. In this set, around 80% claims were Rework and the rest Correct. Since SVMs are not able to handle categorical data we create binary features from the categorical features resulting in around 238,500 and 215,000 features respectively for the two cases.

We ran two pilots with large US health insurance companies to evaluate the prioritization component described in this paper. The pilots involved a team of auditors from the insurance companies where we gave the auditors a set of claims to audit based on system's recommendations and the different strategies described in the previous section. In this paper, we will not describe the entire pilot but just a part of the larger evaluation that motivated and verified the More-Like-This strategy. We give detailed results with a second set of offline experiments that were designed to analyze our strategies in more detail.

5.2 Live System Deployment

We compared the improvement due to the interactive prioritization strategies described in this papers in our pilots. For the first pilot, the baseline system precision was 29% where the auditors audited 200 claims and found 58 claims to be Rework. With More-Like-This strategy, we obtained a hit rate of 55% where the auditors audited 307 claims and found 169 claims to be Rework. Thus we got a 90% improvement over the baseline system. The average audit time per claim was reduced from 3 min 55 sec for the baseline system to 2 min 52 sec for More-Like-This strategy (27% relative reduction in time). The reduction in time for auditing as well as the improvement in precision is statistically significant with $p << 0.001$ for Student's T-Test. We also ran further experiments over a period of 3 weeks with two auditors to validate some of our results.

For the second pilot, we obtained a baseline system precision of 19% where the auditors audited 221 claims and found 42 to be Rework. With More-Like-This strategy, we achieved a hit rate of 51% where 1119 claims were audited and 573 were found to be Rework. This represents a 170% relative improvement over the baseline strategy. The average time to audit was reduced from 5 min 47 sec for the baseline system to 3 min 43 sec for More-Like-This strategy (36% relative reduction in time). As observed for the first pilot, the reduction in time for auditing as well as improvement in precision is statistically significant with $p << 0.001$ for Student's T-Test.

The business benefit of these results are described in more detail in section 6 and show potential savings on $20-26$ million/year for a typical US health insurance provider.

5.3 Offline experiments

Due to the high cost and limited availability of the auditors, we limited the live audits using only the strategies that we thought would work the best. To conduct further in-depth analysis of different strategies described in this paper, we ran a large number of offline experiments. These experiments were performed 5-fold cross-validation on a claims data from last 5 months with 70,179 claims having 52% positive claims.[2]. Since the distribution of the positive class is skewed in historical data we create the test cases in each fold with 5% positive examples as follows. We select all the negative examples belonging to the fold and then exclusively sample from the remaining examples so that the test data has 5% positive examples. We keep creating test cases until we exhaust all the positive examples in the fold. Thus each test fold has same negative examples but unique positive examples. In our case, we got 4 folds with 20 test cases and 1 fold with 19 test cases with the desirable 5% positive class ratio.

[2] Since we wanted to run multiple detailed experiments, we had to select a smaller dataset for practical runtime considerations

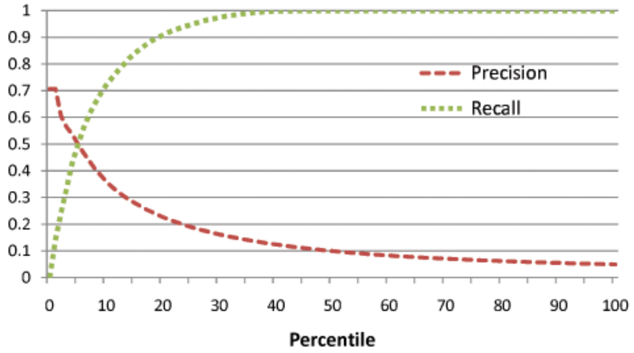

Figure 3: Precision Recall averaged over five folds with approx 20 test sets in each fold

5.3.1 Metrics

The audit capacity for typical insurance companies is approximately 2% of the total daily volume of the claims, which means that given their current audit resources they can only review up to 2% of the incoming claims every day. Standard metrics such as accuracy, precision, or recall are useful in comparing different models and approaches, but not enough to minimize the number of claim errors, which is the eventual metric. Based on the audit capacity and discussions with domain experts, we decided that we need a metric that evaluates performance for the top 2-5% scored claims to maximize benefits in real scenarios. Hence, we use Precision (or hit rate) at 2nd Percentile as our metric. This is similar to the evaluation metric, Precision at top 10 documents, popularly used in the Information Retrieval community. In our experimental framework, we do have the flexibility to modify the metric easily and do model selection based on different metrics. In the results reported below, we show precision recall graphs where appropriate, as well as our precision at top 2% metric when comparing different models.

5.3.2 Is our system accurate?

We ran five fold validation experiments where each fold has approx 20 test sets with 5% Rework ratio to show the effectiveness of our baseline system. Figure 3 shows the Precision-Recall graph for this experiment averaged over five folds with approx 20 test sets each. The average precision at 2nd percentile is 60.5%. Compared to the 2-5% hit rate that the insurance companies get in their auditing practice currently, this is a significant boost in performance.

5.3.3 How well does More-Like-This work?

We use the More-Like-This strategy on top of the baseline system. We experimented with varying the number of clusters (k ranging from 200 to 2800), the clustering algorithms (Repeated Bisections, Repeated Bisections with global optimization and direct k-way clustering), and different cluster ranking metrics as described in Section 4.1.1. We used CLUTO [11] for the clustering.

The best performance was achieved with a cluster size of 500 with repeated bisection clustering algorithm. The ranking criterion for the clusters was ratio of training Rework to Correct claims in a cluster followed by mean prediction score for test claims. Figure 4 shows the performance improvement over the baseline system. We obtain 9% relative improvement for Precision at 2nd percentile, which is the metric that we want to optimize. The improvement is statistically significant with $p << 0.001$ for paired, two-tailed Student's T-test.

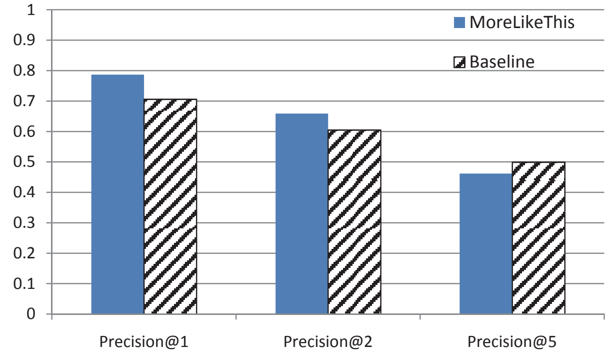

Figure 4: Performance comparison of More-Like-This strategy with Baseline strategy for different metrics

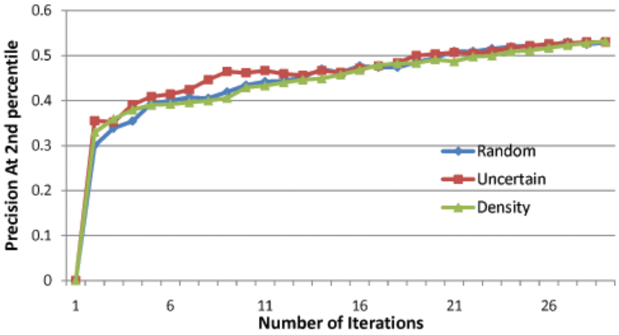

Figure 5: Performance curve for different active learning strategies

5.3.4 How well do typical Active sample selection strategies work?

We experimented with the typical pool-based Active sample selection strategies (uncertainty and density based sampling) along with the baseline random sampling in a batch learning setting. The results are shown in Figure 5. We notice that the performance is quite similar for the different strategies. This result is not too surprising as has previously pointed out by [7] that standard strategies applied myopically in batch learning setting are often worse than random querying.

6. BUSINESS BENEFITS

The work presented in this paper is targeted to increase the overall ROI of the system which we earlier defined as number of incorrectly processed claims identified per dollar of audit cost. The real business metrics behind the system are more complex and we give more details in this section.

Every claim that gets processed and paid incorrectly results in extra costs for the insurance company in various ways. If a claim is underpaid (provider is paid less than the correct amount), the service provider files a dispute with the insurance company. The dispute gets reviewed by the insurance company, and if it is indeed an error, the insurance company sends the additional payment with an interest penalty amount. While the claim is in this dispute review process, the patient and the provider continue calling the customer service center for the insurance company asking for status, resulting in call center costs. According to estimates by an Accenture study, 33% of the administrative workforce is directly or indirectly related to rework processing. Typical costs for insurance companies for these functions because of underpayments are:

- Dispute reviews/audits and adjustments: $10 million/year
- Provider Service Calls: $40 million/year
- Patient Service Calls: $50 million/year
- Interest Payments to Providers: $5 million/year

If a claim is overpaid (provider is paid more than the correct amount), in some cases the provider returns the excess money. In many cases, the insurance company ends up losing the money because they do not have a mechanism of identifying these overpayments. In cases where they do identify them, they have to go through a recovery process which results in additional costs. [1] estimates from a large insurance plan covering 6 million members had $400 million in identified overpayments making it a major source of waste in the US healthcare system.

6.1 Business Value of the Interactive Prioritization Strategy

Based on the costs we described above, we determine the business value of the prioritization strategy presented in this paper on top of the batch system. In our two pilots, the prioritization strategy resulted in 25-40% reduction in audit time per claim while at the same time increasing the relative hit rate by 90-170% (from 29% to 55% for company 1 and from 19% to 51% for company 2). Given that a typical insurance company spends $10 million/year in audits, 25-40% reduction in audit time results in $2.5-4 million/year in savings. Also, the industry estimates that 10% claims end up being paid incorrectly (which we call rework rate). Based on the rework rate of 10% and the absolute increase in hit rate results in 26-32% of additional incorrect claims being caught before being paid reducing the service calls and interest payments by $27-$33 million/year. This total of **$30-$37 million/year** in savings does not even include the medical cost savings that insurance companies will receive because they won't lose money in overpayments. That number is typically hard to quantify since the total is unknown but is known to be a large amount (in the 100s of millions). It's also important to note that these savings are estimates per year for each insurance company. Accenture is currently in discussions with most of the major health insurance companies in the US about this system that we have been developing thus increasing the overall impact on reduction of waste in the US healthcare system. The goal is to reduce the waste in the healthcare system by making claims processing more accurate and reducing the money being wasted by providers and insurers dealing with incorrectly processed claims.

7. CONCLUSIONS

This paper describes an interactive system for detecting payment errors in insurance claims with claim auditors in the loop reviewing the highly scored claims. Our main contribution is an interactive claims prioritization component that makes the system efficient using an online cost-sensitive learning approach (more-like-this). Our interactive prioritization component is built on top of a batch classifier that has been trained to detect payment errors in health insurance claims and optimizes the interaction between the classifier and the domain experts who are consuming the results of this system. The goal is to make these auditors more efficient and effective as well as improving the performance of the system over time. The result is both a reduction in time it takes for the auditors to review and label claims as well as improving the system in finding payment errors. We show results obtained from applying this system at several major US health insurance companies show significant reduction in claim audit costs and $20-$26 million each year (for typical US health insurance providers) making the insurance providers more efficient and lowering their operating costs. This reduces the money being wasted by providers and insurers dealing with incorrectly processed claims and makes the healthcare system more efficient.

In addition to the impact on the healthcare industry, we believe that this also highlights interesting research directions for the KDD community by focusing on reducing the time it takes to label an example, especially when combined with traditional active learning strategies. This setting occurs in many practical deployments of data mining systems. Although our work is specific to insurance claims, our interactive prioritization component is designed to be generic enough to take as input the results of any batch classifier and re-prioritize the results in order to optimize the interaction between the classifier and the domain experts, thus resulting in a higher ROI for many interactive data mining systems such as those for fraud detection, surveillance, intrusion detection among others.

8. REFERENCES

[1] A. Anand and D. Khots. A data mining framework for identifying claim overpayments for the health insurance industry. In *Proceedings of INFORMS Workshop on Data Mining and Health Informatics*, 2008.

[2] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] C. Curry, R. L. Grossman, D. Locke, S. Vejcik, and J. Bugajski. Detecting changes in large data sets of payment card data: a case study. In *Proceedings of KDD*, 2007.

[4] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in knowledge discovery and data mining.* AAAI, 1996.

[5] R. Ghani, M. Kumar, and Z.-S. Mei. Data mining to predict and prevent errors in health insurance claims processing. In *Proceedings of KDD*, 2010.

[6] S. Guha and K. Munagala. Approximation algorithms for budgeted learning problems. In *Proceedings of ACM symposium on Theory of computing (STOC)*, 2007.

[7] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Proceedings of NIPS*, 2008.

[8] R. Haertel, K. Seppi, E. Ringger, and J. Carroll. Return on investment for active learning. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, 2008.

[9] T. Joachims. Training linear svms in linear time. In *Proceedings of KDD*, 2006.

[10] A. Kapoor and R. Greiner. Learning and classifying under hard budgets. In *Proceedings of ECML*, 2005.

[11] G. Karypis. Cluto - a clustering toolkit. Computer sciences technical report, University of Minnesota, 2002.

[12] D. D. Margineantu. Active cost-sensitive learning. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2005.

[13] D. Mladenic and J. Brank. Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of SIGIR*, 2004.

[14] M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. In *Management Science*, 2009.

[15] D. Sculley. Combined regression and ranking. In *Proceedings of KDD*, 2010.

[16] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[17] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, 2008.