An Insurance Recommendation System Using Bayesian Networks

Maleeha Qazi
American Family Insurance, Strategic Data & Analytics
mqazi@amfam.com

Glenn M. Fung
American Family Insurance, Strategic Data & Analytics
gfung@amfam.com

Katie J. Meissner
American Family Insurance, Strategic Data & Analytics
kmeissne@amfam.com

Eduardo R. Fontes
American Family Insurance, Strategic Data & Analytics
erfontes@gmail.com

ABSTRACT

In this paper we describe a deployed recommender system to predict insurance products for new and existing customers. Our goal is to give our customers personalized recommendations based on what other similar people with similar portfolios have, in order to make sure they were adequately covered for their needs. Our system uses customer characteristics in addition to customer portfolio data. Since the number of possible recommendable products is relatively small, compared to other recommender domains, and missing data is relatively frequent, we chose to use Bayesian Networks for modeling our system. Experimental results show advantages of using probabilistic graphical models over the widely used low-rank matrix factorization model for the insurance domain.

KEYWORDS

Recommender systems; Bayesian Networks; Insurance domain; Structure Learning; Deployed system

1 MOTIVATION

Our motivation for creating a recommendation system for the insurance domain stems from wanting to provide value for our customer and our company. We strive to offer products and options that are relevant to our customers, to help narrow down their set of choices to those most appropriate for them, and to improve the overall customer experience. This will increase trust and customer loyalty for the company. The company, in turn, can use the obtained knowledge of the customer base to help create actions to maximize specific company objectives.

The aim is to predict relevant insurance products for our customers based on what other similar people with similar portfolios have. We want to be able to do this for both our current customer base & the prospective customer base. Due to the shifting nature of how people choose to buy insurance, we cannot assume that customers will always have or take time to meet with agents for this purpose. The varying knowledge level of each agent for different products can also influence the customer experience. The current version of the recommender system hopes to aid our agents in providing value to our customers by generating recommendations based on customer portfolio data, and then allowing agents to act on these recommendations as they see fit. The human interaction aspect is relied on for the best possible experience for the customer. Future versions of the system will allow for direct customer interaction & offers.

RecSys '17, August 27–31, 2017, Como, Italy
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4652-8/17/08…$15.00.
DOI: http://dx.doi.org/10.1145/3109859.3109907

2 RELATED WORK

To our knowledge, there are not many documented instances of recommender systems for the insurance domain. Some include: [16], [17], and [8]. [16] describes the unique properties of the insurance domain, and is focused on a system for call center reps servicing Life & Annuity policies with very limited knowledge of the products or customers. Our current system is for agents, who have higher degrees of knowledge about their customers, & is for auto, property, life, and umbrella policies (for which they might not have all the detailed information). In contrast with our system, [17] suggests that their system is based on a standard user-user collaborative filtering approach. [8] is focused on life insurance & uses association rules to create recommendations which are served directly to customers via the web.

There is documentation of a knowledge-based interactive recommender application in the financial industry [7], which was built to assist sales representatives in determining personalized financial service portfolios for their customers. The system walks a sales rep through a series of questions during a conversation with a customer, and outputs an automated summary of the advisory session.

It caught our attention, that a paper that compared some collaborative filtering algorithms [4] showed that for a wide range of conditions, Bayesian networks outperformed other methods. The preferred method depended on the nature of the data and the application, but the paper highlights that Bayesian networks have smaller memory requirements & allow for faster predictions, but require a learning phase that can take a significant amount of time (multiple hours, depending on the amount of data and features) which is not a problem in our case.

Low Rank Matrix Factorization (LRMF) models are a widely used algorithm for recommender systems. We experimented with the Boosted Inductive Matrix Completion algorithms (BIMC) as described in [15], [23], [21], and [2].

3 DESIGN CONSIDERATIONS

The following sections will describe an overview of the whole deployed system and the key design decisions made during the design process.

3.1 System Overview

Our deployed recommender system consists of the following modules, executed in-order:

(1) **The recommendation module:** This is the core recommender model, it takes the input customer data (portfolio, policy, etc.) and generates predictions for likely recommendations.
(2) **The optimization module:** it takes as an input a set of possible recommendations from the recommendation module and prioritizes recommendations based on an optimization criteria predefined by the business. Possible choices for optimization strategies are: customer retention, customer satisfaction, & customer lifetime value (prediction or estimation of the net profit attributed to the entire future relationship with a customer).
(3) **The business rule module:** This module filters recommendations according to business rules to ensure that the final recommendations are appropriate for the customer. For example, it filters out a recommendation if a similar recommendation has been presented to the customer recently (this is based on the system's business requirement).

The final goal is to have the system deployed as a service that can be used in different company applications: As a tool to assist agents (our focus in this paper), as a customer facing recommendation engine, and as a tool to assist our marketing department, among others.

3.2 Models' Scope

For the initial phase of the project, we chose to focus on 3 states: Ohio (OH), Utah (UT), & Nebraska (NE). Since the model features are line specific, we chose to create a property and an auto model per state - for a total of 6 models. All recommendations are made at the policy level.

The purpose of the system for this phase is to suggest new product offers (for both cross-sell and up-sell) for every existing customer. "Cross-sell" means a model recommends a different product line (e.g.: auto model recommends property, life, or umbrella coverage), vs. "up-sell" which means a model recommends an additional coverage in the same product line (e.g. property model recommends id fraud coverage).

Both the property and auto models have 3 cross-sell targets/products: for the auto models they are property, umbrella, and life, and for the property models they are auto, umbrella, and life. The final recommendation for the two cross-sell targets (life and umbrella) that have predictions from both the auto and property models is made by combining both predictions in an optimal way.

In addition, the auto models have 7 up-sell targets/products (comprehensive coverage, collision coverage, emergency road-side service, medical expense, rental reimbursement, death & dismemberment, and lease/loan gap coverage), and the property models have 3 up-sell targets/products (sewer/sump pump overflow coverage, itemized personal property coverage, and id fraud coverage).

3.3 Data Used by the Models

We chose to use data from four product lines: auto, property, umbrella, & life.

The auto data consisted of coverage details (limits, deductibles, discounts, etc.), customer demographics (age, gender, marital status, etc.), household characteristics, and various risk factors. The property data consisted of policy data, coverage details, options and endorsements, dwelling characteristics, household characteristics, customer demographics, sewer and wildfire risk, and geological data. In addition, we used other related policy indicators (for umbrella and life).

As described above, the data for this project came from multiple disparate sources, and the first task was to consolidate the data into a single database. We created a series of jobs to then wrangle the data into the granularity we desired for modeling: from transactional data to tables where each row contained a single policy's information. Then, we did some business analysis to determine the features that are most likely to be available at the time of execution, and contained the most relevant information for the task at hand. This analysis included data source analysis, raw/derived feature analysis, business work-flow analysis, statistical data checks, etc.

Our data contained both discrete and continuous features. Some of the continuous features were manually discretized based on business logic because that made that most sense, but most were binned using the K-Means algorithm [10]. Due to the nature of the data, missing values were kept as separate states for each node, where they were found, because their absence was considered significant.

3.4 Modeling Techniques

During the system design process, we experimented with two algorithms: (a) Bayesian networks (BN) and (b) Low rank matrix factorization (LRMF) models. We chose not to use collaborative filtering because our missing-values would cause issues with defining similarities between users which is a requirement of such algorithms.

A Bayesian network is a probabilistic generative graphical model that represents a set of variables of interest and their conditional dependencies via a directed acyclic graph (DAG). These models are intuitively easy for a human to understand due to the direct dependencies and local distributions vs. a complete joint distribution. And they have both causal and probabilistic semantics that are ideal to represent the combination of prior knowledge and data. In addition, they deal gracefully with missing values. This is one of the main reasons we considered this algorithm for our system since we want to use the methodology for both current and prospective customers for which missing data is common. A detailed introduction to graphical models can be found in [13].

The LRMF models are content-independent of the items being recommended, and they scale well with respect to the number of products added for consideration. In particular, we used a Python implementation of the Inductive Matrix Completion algorithm (IMC), the Boosted Inductive Matrix Completion algorithm (BIMC) ([15], [23], [21], and [2]). This version of LRMF allows for the incorporation of features (customer and policy characteristics) in addition to the label matrix (portfolio features).

There are several characteristics of our problem that made the Bayesian Network approach a better fit in our case:

- Most of our features derived from policy and customer data have discrete values.
- Traditional recommendation systems deal with many products, but in our case we had a limited number of products to work with.
- We had a number of missing values in our dataset that needed to be dealt with.

As we will show later in Section 4, in our initial experiments, we obtained better results with the BN approach.

3.5 Model Training

As described above, for the initial phase of the project, we created a total of 6 models: an auto and property model for each of the 3 states (OH, UT, NE) we were focusing on. The auto and property models were kept separate due to the distinct types of features used for each - the feature sets did not overlap. Also, the value ranges of the features varied by state so we choose to keep each state separate as well.

We tried various tools for BN learning: Weka BN tools [9], bnlearn for R [14, 18–20], the Python library libpgm [11], Netica [3], & BayesiaLab [1]. Due to its rich feature set, we chose BayesiaLab as the tool of choice for this phase of our system. All learning/training was done using the BayesiaLab GUI, since most of the necessary features are not provided via API. But all predictions from learned models were generated using the provided Java API.

Most available algorithms for structure learning are either unsupervised or only consider one target variable in the supervised case. Since we had multiple targets for each of the networks we wanted to learn, we tried multiple structure learning methodologies:

1. Creating a non-empty start point by manual addition of arcs based on business & logical relationships and then running the Taboo algorithm (unsupervised learning).
2. Learning the Augmented Markov blanket of each target separately (supervised learning), concatenating all the learned networks together, setting no targets, and using that as a non-empty starting point for running the Taboo algorithm.
3. Running various unsupervised algorithms from a totally empty starting point (i.e. only nodes, no added arcs).
4. Allowing for more links to the target nodes (by dropping the local structural coefficients to 0) and restricted links among other variables (by leaving the default structural coefficients), and applying various unsupervised algorithms to get the final structure.

The final auto models used the first methodology, and the final property models used a combination of the other three methodologies.

We made a 70/30 train/test split with each dataset. Recommendations were made by running inference on each target in the BN, which made it a multi-label classification problem. For each target, we evaluated the models based on the ROC (Receiver Operating Characteristic) curves [6]. The networks that gave the best test-set AUCs (Area Under Curve) across all targets were chosen as the final networks. In general, the property models had more features/nodes and were more complex than the auto models. Precision@N was not a relevant metric for us since: (a) we had a small number of possible recommendations per household and (b) for each possible recommendation type there are two possibles outcomes (recommend or not) to be shown to the agent. Hence we did not need to rank the recommendations. Since our recommendations are meant to be of the form "if you like A, then you will like B" (e.g. Amazon), time was also not a relevant factor.

Target/ State	NE Test AUC	OH Test AUC	UT Test AUC
Gap	86.99%	86.08%	82.00%
ERS	70.83%	76.99%	74.00%
RR lmt (mean)	80.06%	81.06%	77.50%
Med lmt (mean)	70.07%	74.04%	68.33%
D&D lmt (mean)	70.36%	68.77%	66.75%
Comp ded (mean)	89.87%	87.76%	84.25%
Coll ded (mean)	96.70%	94.96%	93.33%
Umbrella	90.68%	88.97%	87.00%
Property	81.89%	82.11%	88.00%
Life	97.02%	96.44%	93.00%

Table 1: Auto Test Set AUCs (BN)

Target/ State	NE Test AUC	OH Test AUC	UT Test AUC
Sewer lmt (mean)	77.77%	76.37%	79.20%
Itemized Pers Prop	74.72%	75.98%	75.87%
ID Fraud	70.38%	72.54%	74.58%
Umbrella	87.34%	85.89%	85.84%
Life	65.30%	64.65%	70.36%
Auto	64.79%	65.72%	65.11%

Table 2: Property Test Set AUCs (BN)

4 MODEL RESULTS

4.1 Pre-deployment Off-line Validation

Next, we show model validation results obtained before model deployment. The final auto networks' test-set AUCs across all targets are shown in Table 1. "Ded" stands for deductible, and "lmt" stands for limit. The final property networks' test-set AUCs across all targets are shown in Table 2. For the multi-state targets in both tables, AUCs were an average of the calculated AUC per state.

Table 3 shows comparisons between the low rank models and the BN on the UT property data (with the same data binning, and train/test data split). The best results are in bold. As it can be seen from the table, in general, the BN approach performed better. The same trend was observed for all the 6 models we built. Figure 1 shows an example of the auto network's structure for the state of OH.

4.2 Post-deployment On-line Validation

The system went live on 17 Nov 2016, and since then 4 check-points have been taken of how well the system is performing according to the operational metrics defined by business (check-points taken: 14, 56, 89, & 112 days from launch). The initial/beta roll-out went

Target/ State	IMC AUC	BIMC AUC	BN AUC
Sewer lmt	75.63%	76.51%	**79.20%**
Itemized Pers Prop	68.68%	75.08%	**75.87%**
ID Fraud	67.51%	73.49%	**74.58%**
Umbrella	81.99%	84.55%	**85.84%**
Life	71.82%	**73.14%**	70.36%
Auto	66.44%	**85.91%**	65.11%

Table 3: Low Rank Model UT Property Test Set AUCs (vs. BN). Best results in bold.

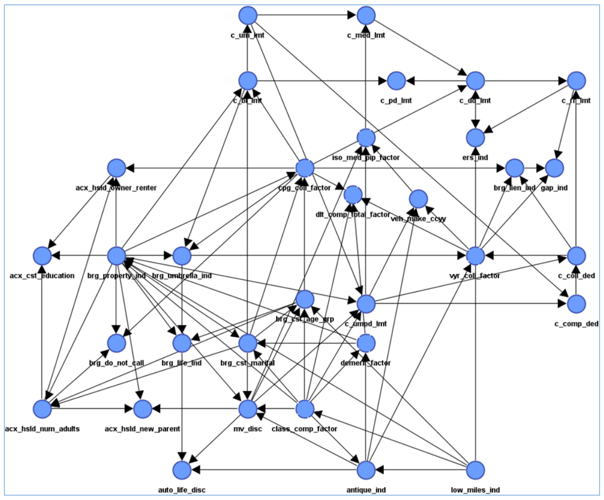


Figure 1: Example of the auto network for the state of OH. Complexity: 33 nodes, 89 arcs, avg. of 2.69 parents/children per node.

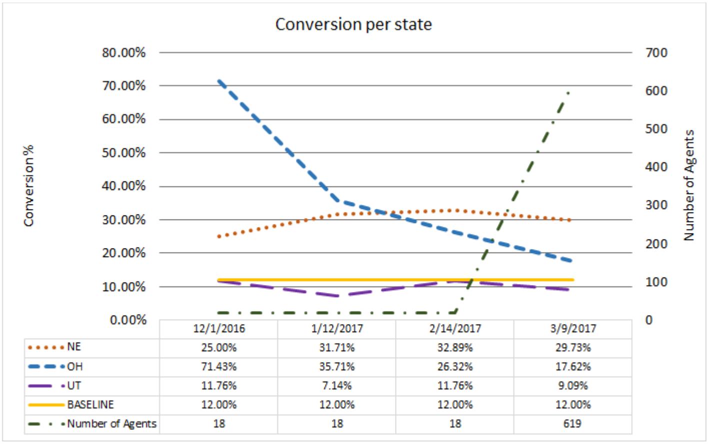
to 18 agents across the 3 states (OH, UT, NE). These individuals were the district sales representatives of those states. As of 3 Feb 2017, all remaining agents in those 3 states were given access to the recommendations generated for their accounts (only the 89 & 112 day checkpoints would be affected by this increase in number of agents; total of 619 agents licensed to sell in those 3 states).

At the last checkpoint taken (date: 3/9/2017), there were 366,998 active recommendations, 737 received agent action/feedback (accept: 104, decline: 465, defer: 56, quote initiated: 112).

Our deployed recommender has been well received in general by the agents. Some of the good feedback we have received from them includes:

- "I've increased premium volume with 3 customers in only 2 days"
- "[it] brings attention to details to every customer."
- "..is a valuable teaching tool for ourselves and our staff."

For this stage of the system, success is measured by conversion, which is the percentage of offers presented that get accepted by the customer. The conversion per state for offers generated by the system was calculated at each checkpoint, and compared to the standard industry conversion baseline of 12%, as shown in Figure 2. As expected, because NE & OH have higher recommendation



	12/1/2016	1/12/2017	2/14/2017	3/9/2017
NE	25.00%	31.71%	32.89%	29.73%
OH	71.43%	35.71%	26.32%	17.62%
UT	11.76%	7.14%	11.76%	9.09%
BASELINE	12.00%	12.00%	12.00%	12.00%
Number of Agents	18	18	18	619

Figure 2: Conversion Across All States vs. Baseline

counts than UT, the conversion percent is higher in those states vs. UT, which is closer to the baseline.

We will wait for a year of feedback to calculate more sophisticated performance indicators (like retention) to assess the recommender's impact on the business using a test & learn methodology [12].

5 CONCLUSIONS AND LESSONS LEARNED

The recommender system we deployed aims to help the entire customer base by uniformly bringing coverage options to the attention of agents so that our customers are adequately covered for their needs. Clearly an agent tailors the recommendation coverage for each customer, but the system helps prompt this action, whereas before there were no prompts for agents.

We learned that business rules were very important. For example, the gap target didn't make sense because based on underwriting criteria this could only be added at the initial point of car purchase, and adjusting the model's thresholds for giving recommendations are very sensitive for business success.

6 FUTURE WORK

We have many things we'd like to try to improve our system, among them are:

- Create unified models for all lines instead of having separate models for auto and property
- Extend the system to prospective customers. This was one of the main reasons for our choice of Bayesian networks. There is a lot of information missing for prospective customers so the model should be able to gracefully adapt and make predictions with little available data.
- Compare our results to the deep learning approaches proposed in [5, 22].
- Roll-out recommendation models for the remaining states.
- Go beyond cross-sell and up-sell targets to "next best action" for any customer at any time. E.g., send a timely communication to the customer or grant an additional discount.

REFERENCES

[1] Bayesialab. http://www.bayesia.com/. Versions: 5.4.3 & 6.0.2; Accessed: 1-Feb-2017.

[2] IMC software. http://bigdata.ices.utexas.edu/software/inductive-matrix-completion/. Online; Accessed 1-Feb-2017.

[3] Netica application. https://www.norsys.com/netica.html. Accessed: 1-Feb-2017.

[4] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[5] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. Wide and deep learning for recommender systems. *arXiv:1606.07792*, 2016.

[6] T. Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[7] A. Felfernig and A. Kiener. Knowledge-based interactive selling of financial services with fsadvisor. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3*, IAAI'05, pages 1475–1482. AAAI Press, 2005.

[8] A. Gupta and A. Jain. Life insurance recommender system based on association rule mining and dual clustering method for solving cold-start problem. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3, Oct 2013.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[10] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, July 2002.

[11] K. R. Karkera. *Building Probabilistic Graphical Models with Python*. Packt Publishing, 2014.

[12] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2006.

[13] K. P. Murphy. An introduction to graphical models. Technical report, 2001.

[14] R. Nagarajan and M. Scutari. *Bayesian Networks in R with Applications in Systems Biology*. Springer, New York, 2013. ISBN 978-1-4614-6445-7, 978-1-4614-6446-4.

[15] N. Natarajan and I. S. Dhillon. Inductive matrix completion for predicting gene-disease associations. *Bioinformatics*, 30, jun 2014.

[16] L. Rokach, G. Shani, B. Shapira, E. Chapnik, and G. Siboni. Recommending insurance riders.

[17] B. P. Sanghamitra Mitra, Nilendra Chaudhari. Leveraging hybrid recommendation system in insurance domain. *International Journal of Engineering and Computer Science*, 3, Oct 2014.

[18] M. Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[19] M. Scutari. Bayesian network constraint-based structure learning algorithms: Parallel and optimized implementations in the bnlearn R package. *Journal of Statistical Software*, 77(2):1–20, 2017.

[20] M. Scutari and J.-B. Denis. *Bayesian Networks with Examples in R*. Chapman and Hall, Boca Raton, 2014. ISBN 978-1-4822-2558-7, 978-1-4822-2560-0.

[21] D. Shin, S. Cetintas, K.-C. Lee, and I. S. Dhillon. Tumblr blog recommendation with boosted inductive matrix completion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 203–212, New York, NY, USA, 2015. ACM.

[22] F. Strub, R. Gaudel, and J. Mary. Hybrid Recommender System based on Autoencoders. In *the 1st Workshop on Deep Learning for Recommender Systems*, pages 11 – 16, Boston, United States, Sept. 2016.

[23] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–593–I–601. JMLR.org, 2014.