Proceedings of the Twenty-Sixth Annual Conference on Innovative Applications of Artificial Intelligence

Predictive Models for Determining If and When to Display Online Lead Forms

Timothy Chan, Joseph I, Carlos Macasaet, Daniel Kang, Robert M. Hardy, Carlos Ruiz, Rigel Porras, Brian Baron, Karim Qazi, Padraic Hannon, Tomonori Honda
Propensity Modeling Team
Edmunds.com
1620 26th St, 4th Floor
Santa Monica, CA 90404

Abstract

This paper will demonstrate a machine learning application for predicting positive lead conversion events on the Edmunds.com website, an American destination for car shopping. A positive conversion event occurs when a user fills out and submits a lead form interstitial. We used machine learning to identify which users might want to fill out lead forms, and where in their sessions to present the interstitials. There are several factors that make these predictions difficult, such as (a) far more negative than positive responses (b) seasonality effects due to car sales events near holidays, which require the model to be easily tunable and (c) the need for computationally fast predictions for real-time decision-making in order to minimize any impact on the website's usability. Rather than develop a single highly complex model, we used an ensemble of three simple models: Naive Bayes, Markov Chain, and Vowpal Wabbit. The ensemble generated significant lift over random predictions and demonstrated comparable accuracy to an external consulting company's model.

1 Introduction

Edmunds.com is a car-shopping website committed to helping people find and purchase the car that meets their every need. Almost 18 million visitors use our research, shopping, and buying tools every month to make an easy and informed decision on their next new or used car. We provide comprehensive car reviews, shopping tips, photos, videos, and feature stories offering a friendly and informative car shopping experience.

One key conversion activity is enabling visitors to contact dealers for price quotes. In the past, in order to request a price quote from a dealer, visitors needed to navigate to particular pages. We call price quote requests, *dealer leads*, and we call the pages that allow visitors to make such requests, *static lead forms*.

Through a series of experiments, we found that some visitors who wanted price quotes were unable to locate the appropriate lead form pages. So, we created a lead form that could appear at the most appropriate time for the user. To accomplish this, we implemented an interstitial or pop-up style lead form. We call a visitor who requests a quote from a dealer using an interstitial or static lead form a *lead submitter*.

Given that interstitials tend to annoy visitors who are not interested in submitting leads, we prefer to show lead forms only to visitors who are likely to respond positively. Annoying our visitors could lead to fewer page views, lower dwell times, and lower advertising revenue. We need to minimize annoyance while generating revenue through dealer leads.

This is a difficult problem because: (a) a small percentage of our visitors request price quotes from dealers; so we have an extreme class imbalance, which makes building good models difficult (Gao et al. 2008; He et al. 2009), (b) interstitials are a new feature on our site, so we have limited training data, (c) our website is dynamic; content and page routing can change at any time,[1] (d) to maintain website usability, we had to ensure that pages loaded within 500 milliseconds, and (e) half of our traffic consists of new visitors.

Because of these limitations, we needed a computationally efficient yet robust and adjustable AI-based model that will predict who is most likely to react favorably to the lead form interstitial and when to show it.

2 Data

We used two main sources of data for our modeling efforts, internal clickstream data and third-party demographic information. The primary source – our clickstream – is the page view history of each visitor to our site (Moe 2003; Moe and Fader 2004). From this, we can glean insight into the pages and content that interest each visitor, as well as the progression of page views reflecting each visitor's thought process. We organize our pages by assigning them to categories such as new car, used car, inventory, reviews, tips & advice, news, calculator, etc. These categories help us determine, at a high level, the type of information visitors are seeking. This information is especially useful when visitors stay on the site for a long time and view a lot of content. However, many of our visitors have short clickstream ses-

[1] We assume that under-fitted models will be more robust to website changes than over-fitted models.

sions. More than half of our visitors are "new"[2] and the average number of page views per visitor is fewer than four. Nevertheless, a small number of visitors have very long sessions, making for a long-tailed pages per session distribution.

Because clickstream data contains limited information, we augment our dataset using external demographic information provided by third parties. This demographic information comes in two distinct forms. The first comes from a cookie-based provider (Eirinaki and Vazirgiannis 2003; Chester 2012). With this, we infer household-level information about our visitors because each cookie is associated with a particular computer. The main drawback of this type of data is that it is not available for most visitors. Only visitors with the third-party cookie display this information.

The other type of demographic information is zip-code based data (United States Census Bureau 2010). In this case, we map the IP address of each visitor to a zip-code and we obtain aggregate demographic information about each geographic area. Because we only have to ensure that a particular IP address maps to a zip-code, the coverage for this type of data is much better, but the information obtained is less targeted.

3 Models

One of the difficulties associated with clickstream prediction models is that the length of browsing history varies drastically from session to session. We must either aggregate the data into summary statistics, or the model must be able to handle variable length data. These considerations impacted our choice of models. We settled on Naive Bayes, Markov Chain, and Vowpal Wabbit (VW).

The Naive Bayes model assumes that each page view is a separate disjoint event and that the overall propensity for filling out a lead form is the sum of incremental propensities for each page viewed. The Markov Chain model focuses on transitions between pages. Finally, Vowpal Wabbit is an online logistic regression model into which we pass aggregated session statistics as well as third-party demographic information. We designed all three models to predict the likelihood of a *conversion event*, which, in our case, is a visitor filling out and submitting an interstitial lead form.

Naive Bayes

We chose Naive Bayes (Langley, Iba, and Thompson 1992) as the first model because of its simplicity and ability to handle large numbers of categorical variables as separate features. Despite its simplicity, it has been shown to have accuracy comparable to other classification models. (Rish 2001; Hand and Yu 2001). In our application, we treat each page on our website (after cleaning) as a distinct feature, as in a "bag of words" or "bag of features" (Zhang et al. 2007). Our Naive Bayes model only considers webpages without reference to demographic information or the order of the page views. Thus, we assume that every page a visitor views increments the propensity to submit a lead.[3]

Our first step in building a Naive Bayes model was to clean the recorded URLs for each visitor. This is crucial because there can be over 5 million distinct URLs for one month's traffic of only 12.5 million visitors. Thus, without cleaning, there would be too few statistically significant URLs. We cleaned URLs by removing query parameters and by combining similar, infrequently visited, pages together into page types.

For example, we aggregated all pages that contain dealerships such as:

http://www.edmunds.com/dealerships/all/NewYork/

http://www.edmunds.com/dealerships/Acura/California/

http://www.edmunds.com/dealerships/Hyundai/Kansas/Wichita.html

into www.edmunds.com/dealerships. This puts rarely visited pages into broader categories like car-care, car-reviews, etc. By doing this, we cut down the number of unique features to fewer than 9,000 pages.

To simplify the Naive Bayes calculation, we computed the weight of evidence (Goodman and Royall 1988) for each page as follows:

$$p_{page_i} = \frac{\text{Count of lead submitters who saw page } i}{\text{Count of all visitors who saw page } i} \quad (1)$$

$$p_{avg} = \frac{\text{Count of lead submitters}}{\text{Count of all visitors}} \quad (2)$$

$$w_{page_i} = \log\left(\frac{p_{page_i}}{p_{avg}}\right) \quad (3)$$

where p_{page_i} is the likelihood of conversion given that a visitor saw page i. p_{avg} is the average conversion rate, and w_{page_i} is the log-likelihood ratio, or weight of evidence, for page i. Given these weights of evidence, we compute the Naive Bayes score for the k-th visitor as:

$$W_k^{NB} = \sum_{i \in \{page_1^k, \ldots page_N^k\}} w_i \quad (4)$$

where $page_i^k$ is the i-th webpage that the k-th visitor viewed and N is the total number of pages that the k-th visitor viewed.

With this Naive Bayes model, we can analyze how viewing particular pages on Edmunds.com impact lead submissions. Pages like consumer reviews for older models, car maintenance pages, and discontinued vehicle pages exhibit negative weights of evidence. This supports our intuition that viewers of these pages most likely are not new car shoppers wanting price quotes from dealers. On the other hand, pages like dealership info, car leasing info, and the homepage for new cars carry positive weights of evidence. Another interesting observation is that for new car pages, the direction of the weight of evidence depends on make and model. This shows that visitor intent is linked to the types of cars they view. Some cars with negative weights of evidence are Aston Martin, Lamborghini, Tesla, and the Ford

[2]This means that we cannot associate a visitor's session with any previous sessions. This can happen for various reasons including visitors clearing their caches or using multiple browsers.

[3]After a threshold is surpassed, a lead form is displayed.

Mustang. This tells us either that buyers of these cars do not need help obtaining pricing information or that they jare aspirational window shoppers or car enthusiasts rather than buyers who are close to purchasing.

The assumption that page views are independent and that page order is not important are not realistic. Yet, the Naive Bayes model is intuitive and computationally efficient.

Markov Chain

The second model we explored was a first order Markov Chain model. The Markov Chain model has been utilized in various domains including anomaly detection (Ye 2000) and classification (Liu and Selfridge-Field 2002; Kaliakatsos-Papakostas, Epitropakis, and Vrahatis 2011; Li and Wang 2012). Because lead submission are rare events, an algorithm that is able to detect anomalies seemed a good choice. The model detects differences in page transition patterns between lead and non-lead submitters. Unlike Naive Bayes, this model focuses on the order of events, examing that a visitor who views Page A before Page B may have different lead submission propensity than a visitor who views Page B followed by Page A.

One limitation of Markov Chain model is that it requires significantly more training data than the Naive Bayes model. Using 9,000 different URLs like we did with Naive Bayes resulted in 81 million possible transitions. Many of these transitions rarely happen, so confidence in the transition probabilities will be low. To reduce the number of possible transitions, we used our internal page categories such as home_page, model_review, new_model_mydp_pricing, etc. These categories describe pages at a high level and lack details such as car make and model. Because we have about 90 different page categories, the number of transitions reduces to 8,000.

With this page category data, we created the Markov Chain model in the following manner. First, we calculated page transition probabilities for lead submitters and non-lead visitors separately.

$$T^{lead}_{cat_i,cat_j} = \frac{CNT^{lead}_{cat_i,cat_j}}{\sum_{cat_k} CNT^{lead}_{cat_i,cat_k}} \tag{5}$$

$$T^{non\text{-}lead}_{cat_i,cat_j} = \frac{CNT^{non\text{-}lead}_{cat_i,cat_j}}{\sum_{cat_k} CNT^{non\text{-}lead}_{cat_i,cat_k}} \tag{6}$$

where $CNT^{lead}_{cat_i,cat_j}$ is the count of lead submitters who saw page category i followed by page category j and $CNT^{non\text{-}lead}_{cat_i,cat_j}$ is the count of non-lead submitters who saw page category i followed by page category j. We also calculated the transition probability for a visitor's initial page category (the category of the page on which a visitor lands) in a similar manner.

$$T^{lead}_{cat^0_i} = \frac{CNT^{lead}_{cat^0_i}}{\sum_{cat^0_j} CNT^{lead}_{cat^0_j}} \tag{7}$$

$$T^{non\text{-}lead}_{cat^0_i} = \frac{CNT^{non\text{-}lead}_{cat^0_i}}{\sum_{cat^0_j} CNT^{non\text{-}lead}_{cat^0_j}} \tag{8}$$

where $CNT^{lead}_{cat^0_i}$ is the count of lead submitters who landed on page category i and $CNT^{non\text{-}lead}_{cat^0_i}$ is the count of non-lead submitters who landed on page category i. Given these transition probabilities, the Markov Chain model becomes:

$$W^{MC}_k = \log\left(\frac{T^{lead}_{cat^0_{k1}}}{T^{non\text{-}lead}_{cat^0_{k1}}}\right) + \sum_{j=cat_{k2}}^{cat_{k(N-1)}} \log\left(\frac{T^{lead}_{j,j+1}}{T^{non\text{-}lead}_{j,j+1}}\right) \tag{9}$$

where cat_{ki} is the category of the i-th webpage that the k-th visitor viewed.

The Markov Chain model provided insight into the types of pages on which lead submitters tend to land. Intuitively, visitors who land on dealership, incentives, inventory, or pricing related pages are more likely to be close to purchasing a car and therefore more likely to request a price quote. Visitors who land on pages that provide advice on new car buying are also more likely to submit leads. On the other hand, visitors who land on pages about used cars, repair & maintenance, or advice about warranty and safety, are less likely to submit leads. These intuitive results provided some assurance that the model was correctly tuned.

The Markov Chain model also provided insight into browsing patterns of lead submitters. Consider these two visitors who saw the following, as their first two pages on Edmunds.com:

Visitor A: new model pricing, maintenance index

Visitor B: maintenance index, new model pricing

The new model pricing page has a *positive* weight of evidence, while the maintenance index page has a *negative* weight of evidence. In our Naive Bayes formulation, because order does not matter, both visitor A and B will have the *same* likelihood to submit a lead. However, according to the Markov Chain model, a visitor who saw the new model pricing page first has a *positive* weight of evidence. The visitor is 1.91 times more likely to submit a lead than on average. In contrast, if a visitor saw the maintenance page, followed by new model pricing page, there is a *negative* weight of evidence. That visitor is 20% less likely to submit a lead compared to the average. Thus, this model learns the ordering of events, in this case, sequences of page views.

Vowpal Wabbit

One notable thing about our visitors is that their behavior reflects seasonality. Because car manufacturers tend to have sales events near holidays, we have more visitors and higher conversion rates around holidays like Memorial Day, Independence Day, and Labor Day. Ideally, our model should account for this seasonality. One way to capture seasonality is to use online learning (Bottou 1998; Kivinen, Smola, and Williamson 2004; Smale and Yao 2006; Saffari et al. 2009; 2010).

We chose to use Vowpal Wabbit (*VW*), an online learning system developed by John Langford and others (Langford 2007; Agarwal et al. 2011). It is regarded as a computationally fast machine learning system. We felt that *VW* was an appropriate choice for our third model because it focuses on

statistics and other information not used by the other two models.

The main features explored by this online model were: (a) clickstream statistics, e.g., page count, new car visit, etc., (b) vehicle price information, (c) zip code information, (d) search keywords, and (e) quadratic term that combines clickstream statistics and zip code information.

We normalized these features before building the model. For example, the number of pages visited was transformed using a $log(x+1)$ function to compress long-tailed distributions. Return visit counts were converted into indicator variables, showing either a statistically significant increase or decrease in the conversion rate compared to the average conversion rate.[4] We transformed zip code and search keyword information using weight of evidence as in the Naive Bayes formulation. We also measured whether visitors looked at similarly priced cars or whether they transitioned to higher or lower priced cars. We hypothesized that visitors shopping within narrow price ranges were deeper into the buying funnel and readier to request quotes from dealers. We tried to choose the most appropriate transformation for each variable on a case-by-case basis.

One difficulty in training *VW* was its sensitivity to the order of training cases. Some sensitivity was expected because training used stochastic gradient decent. However, because our conversion rate was so low, this sensitivity was magnified. Thus, we used batch learning for feature selection before training with *VW*. For simplicity, we utilized R's logistic regression package (The R Development Core Team 2012) and backward feature selection (Guyon and Elisseeff 2003) to narrow down the key variables.

After feature selection, we saw some interesting results. The most important features were the statistics from clickstream behavior such as number of pages visited, the fraction of visits containing new car pages, the number of car review pages visited (showing negative correlation), etc. The weight of evidence for zip code and search keywords were also important. On the other hand, third-party demographic information added some value, but not with high importance.[5] This may have been caused by the fact our demographic variables were zip code based, and probably overlapped with our zip code weight of evidence variables. Finally, we thresholded the *VW* model to target the same percentage of visitors as the Naive Bayes and Markov Chain models.

Simple Ensemble for Final Predictions

Given our three different models, we needed to form an appropriate ensemble (Dietterich 2000; Wang et al. 2011) based on stacking (also called blending) (Ting and Witten 1999; Dzeroski and Zenko 2004). Even though the stacking approach can be hit-or-miss, and hard to justify theoretically, it gained popularity following its success in the Netflix (Koren 2009) and Heritage Health Prize (Vogel, Brierley, and Axelrod 2011) competitions.

Although we could have taken more time to devise a better stacking algorithm, we decided to use a logical "OR" function to aggregate the predictions from Naive Bayes, Markov Chain, and Vowpal Wabbit. We took this approach to limit false interstitial signals that would reduce dwell time and ad revenue.

To retrain our models, a Hadoop workflow aggregates the required metrics using the latest clickstream data. Then, Python scripts process the aggregated data to produce a weight of evidence CSV file for the Naive Bayes model and transition probability CSV files for the Markov Chain model. With the CSV files and a holdout set of clickstream data, a validation harness incorporating our business rules determines both target rate and accuracy of the model. Upon verification, we publish the updated CSVs to our lead prediction service.

4 Results

We verified the effectiveness of our predictive model using three different baselines: (a) conversion when interstitial was shown to everyone, (b) conversion when interstitial was shown randomly at 5% for every page viewed, and (c) conversion when using a consulting company's model

The first baseline provides an upper bound on conversion rate lift. When we showed the interstitial to everyone, we got about a 70% lift. However, by showing an interstitial to everyone, we saw a significant increase in complaints, with visitors stating that the user-friendliness of our website had declined. This demonstrated the necessity for a "smart" solution that would target the lead form interstitial properly.

As a second baseline, we compared our model to random interstitial display. Rather than randomly dividing the population for determining who will see the interstitial, we randomly decided to show or not show interstitials on each page. Because visitors who see more pages have a higher likelihood to see the interstitial, this "random" model exploits the correlation between page views and the interstitial conversion rate. Thus, the conversion rate was higher for this model than had we just randomly divided the population into interstitial and non interstitial visits. When we chose a 5% likelihood for showing the interstitial on each page, we observed a 20% conversion rate lift.[6]

For the last baseline, we compared our blended model to a consulting company's model which was built with the same goal: to increase lead conversions. Because the consulting company did not reveal many details about their model, we treated it as a black box. Also, since the consulting company was working on building and improving their model for eight months, it provided a rough upper bound estimate for the performance we might expect on this problem.

Figure 1 compares the conversion rate between our model, the consulting company's model, and a control group. Over 1.5 months, the weighted average conversion (accounting for daily traffic variation) was 0.210% for the control group,

[4]The main reason for this transformation is that the lead conversion rate as a function of the count of return visits shows a nonlinear relationship that logistic regression and Vowpal Wabbit cannot easily capture.

[5]Validation AUC for the *VW* model showed improvement from 0.703 to 0.725 by adding the third party demographics information.

[6]For reference, the average page views per visitor was 4.6.

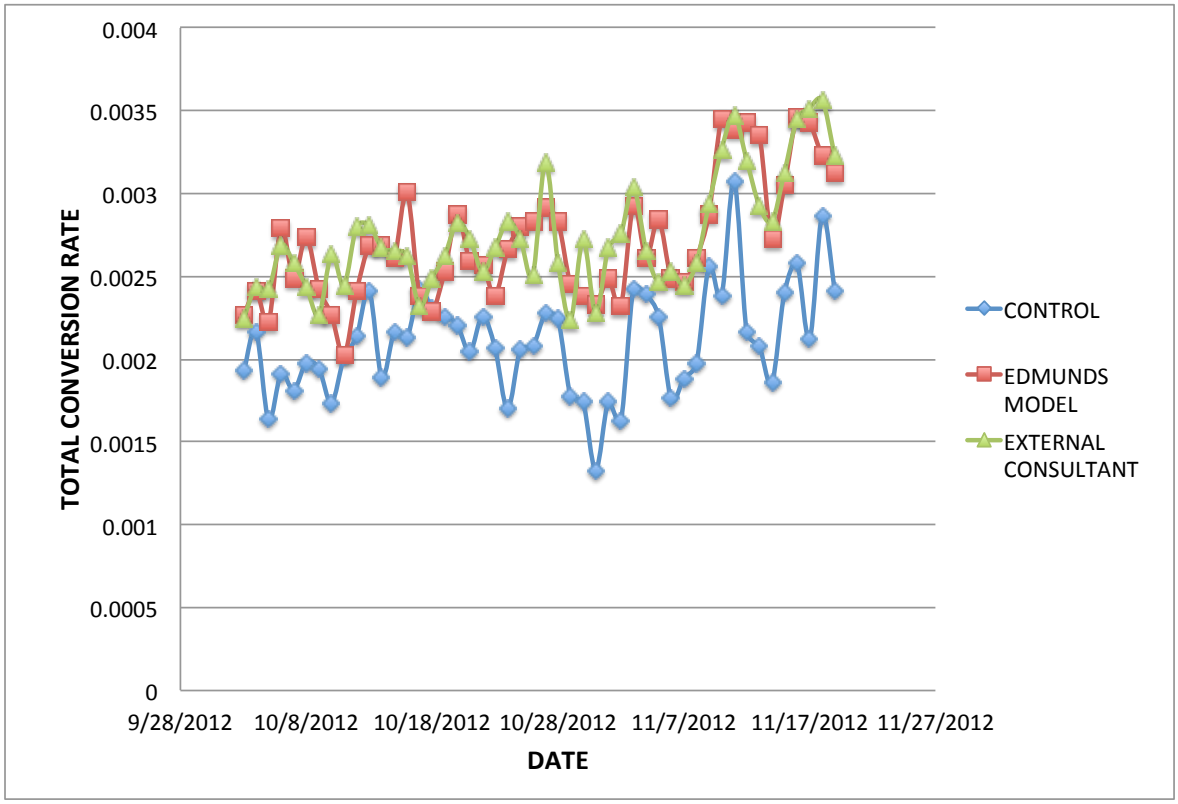

Figure 1: Comparison of Total Conversion Rate

0.273% for the consulting company's model, and 0.270% for our final internal model. This indicates a 29% boost in conversions for the Edmunds.com model and a 30% increase for the consulting company's model, demonstrating that the two models perform similarly. Given that the first baseline showed at most a 70% increase in conversion and because we targeted less than 15% of the population, with random targeting we would expect a 10.5% increase in conversion. Both our stacked model and the consulting company's model also outperformed the second baseline which exploits the number of pages viewed by our visitors.

To understand the differences between our model and the consulting company's model, we examined lead conversion and return visitors. Our internal model actually had a slightly higher (by about 5%) lead conversion rate per visitor than the consulting company's model. Though lead conversion per interstitial was higher, total lead submissions were lower, suggesting that our model may have cannibalized static lead form submissions. That is, our model may have shown the interstitial to visitors who were already going to submit a lead via the static form. Figure 2 shows that lead conversion depends on the percentage of return visitors. The consulting company's model performs better with return visitors, while our model shows more promise for first time visitors. This suggests potential for further improvement by aggregating our model with the consulting company's model.

5 Level of Exposure

We deployed our model as an A/B test. We randomly bucketed each of our visitors into one of several recipes. Although the recipes and allocations varied over time, we usually reserved at least ten percent of our traffic for the default site experience, which had no lead form interstitials. The rest of the traffic was typically split between our predictive model and the one developed by our consultant. The model recipes were just the regular site experience, with the possibility that a lead form interstitial might be shown. Ultimately, our model was deployed to production for three months at a high traffic level. The model scored 14.5 million unique visitors and recommended lead form interstitials for 2 million of them. Of those, 12,600 thousand submitted leads.

6 Comparison to a Third Party Vendor

As described previously, Edmunds.com partnered with a consulting company to build a lead prediction model. While developing and maintaining their model, a representative from the company attended our daily project scrums. We also held several meetings with their developers to discuss the nature of our data and give them some domain expertise.

For visitors who were bucketed into the consultant's A/B test recipe, we used the consultant's REST service to determine whether or not to show a lead form. The REST call included, among other things, the URL visited, our assigned

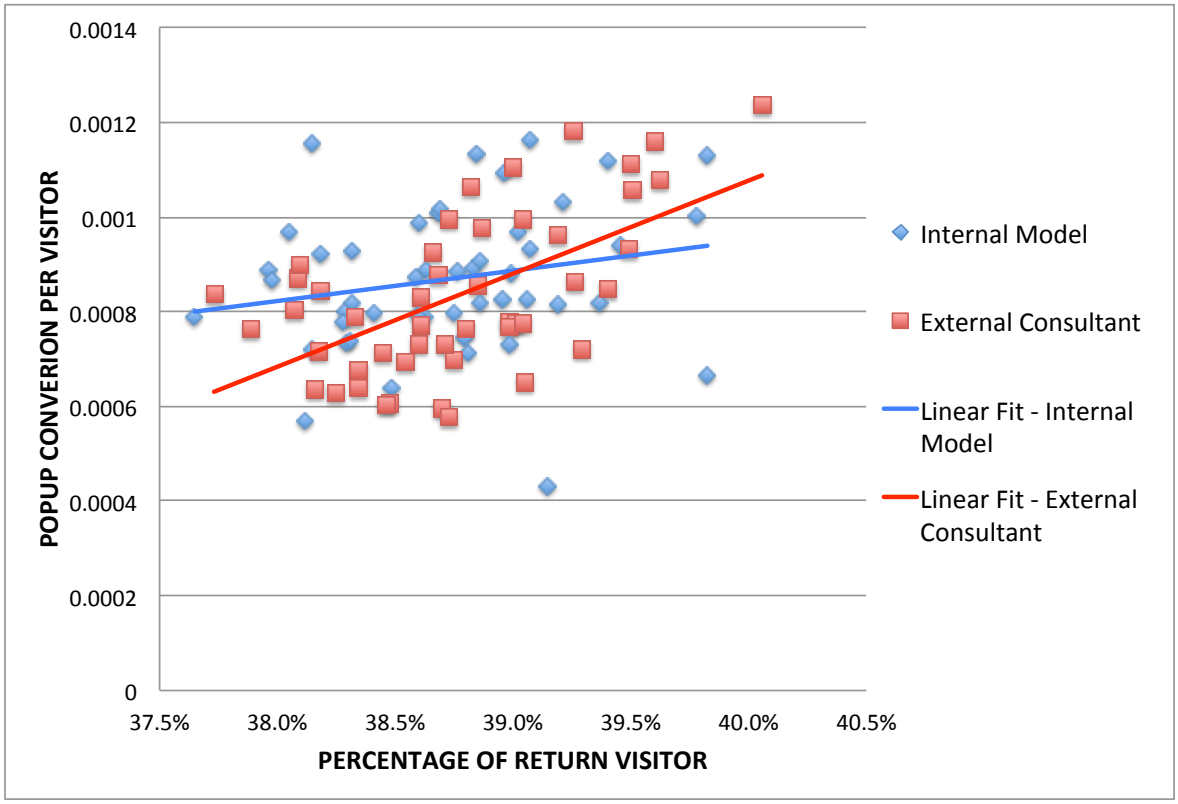

Figure 2: Percentage of Return Visitor vs. Lead Conversion per Visitor

page category and page name, and any vehicle data, such as make or model, associated with the page.

Aside from providing us with a prediction, this REST call also afforded the vendor live updates of the visitors in their recipe. Through separate REST calls, we also notified the vendor, in real time, every time a lead form interstitial was presented, dismissed, or submitted. In addition to live updates, every month we sent them the past month's clickstream history for all visitors so they could retune their model with more comprehensive data sets.

The consulting company did not provide us too many details about their model, but they did indicate it was a tree-based ensemble.

7 Discussion

We learned several lessons from this case study and we identified opportunities for future research.

1. *Systematic Feature Creation*

   Even though this research is more focused on improving model accuracy and less on feature generation, creating a good set of features is critical to building good predictive models. By cleaning the data and creating good features, we can use simpler, more computationally-efficient learning models. For example, our models are simpler than state of the art algorithms that utilize bagging and boosting like AdaBoost, Random Forest, and Gradient Boosting Machine. Simpler algorithms can be computed more quickly, enabling us to minimize user wait times. Therefore further advances in generating useful features will be beneficial.[7]

2. *Improving the Aggregation Process using Stacking*

   Because of the cost of model building, it is practical to start with simpler models and build more complex models only if they are warranted. Stacking potentially leverages the strengths of several simple models. In light of recent work that has shown that stacking tends to be hit-or-miss (Graczyk et al. 2010), future work should be done to determine conditions under which stacking will work effectively. Work could also be done to determine which additional model types will improve an existing stack.

3. *Early Estimation of Trade-off between Improving Prediction vs. Improving Product*

   In website AI applications, data may not represent the implementation environment as well as we would like. Changes in usage patterns, seasonality, and changes in the website itself result in a non-stationary problem domain. For example, aside from optimizing interstitial timing, we have noticed significant improvement in conversion lift by enhancing the aesthetic design of the interstitials them-

[7] Although there have been some studies involving automatic feature generation (Leather, Bonilla, and O'Boyle 2009), more effort has been devoted to improving feature selection techniques (Saeys, Inza, and Larranaga 2007; Deisy et al. 2010)

selves. Thus we should weigh the benefits of making better interstitial timing predictions against making better interstitials.[8]

We reaped a number of benefits by developing a predictive model ourselves. First, we freed up resources from having to communicate with the consulting company. Due to security concerns, we could not provide the consultants direct access to our data warehouse; because of this, each month, we needed to prepare training data to upload to a shared location accessible by the consultants. Finally, it was necessary to spend hours explaining domain specific terminology and Edmunds.com business practices to the consultants.

Second, because the consultant company's model was a black box to us and was reliant on data provided by us, it often lagged behind changes to the website. When we made layout or navigation changes to our site, we had to wait for the consultant to re-tune their model instead of being proactive about model changes.

Third, the consulting company's model implementation sometimes failed to handle Edmunds.com's traffic volume. In addition to providing a model, the consulting company also managed a REST layer for scoring visitors live. At times, their service became unavailable or did not return within an acceptable delay. We then had to rely on them to fix the problem in a timely manner.

8 Conclusion

Through our project to improve lead form conversions for Edmunds.com, we showed that stacking three simple models can effectively predict classification of a small minority class. Model building focused on data transformation and feature creation, rather than on model complexity. The project demonstrated a need for further research in the following areas: (1) systematic feature generation, (2) formalization of the ensemble stacking algorithm to enable estimation of offline performance, (3) knowing which additional models have the best chance of further improving the ensemble, and (4) a method for determining when to improve our website product offerings versus using predictive modeling to optimize timing.

Towards the end of 2013, we removed the lead form interstitial. Edmunds.com underwent an extensive site redesign that included a new lead product. Product teams needed to test the effectiveness of incremental improvements to the site without metrics for traditional leads diluted by the interstitial.

In the first quarter of 2014, after the changes to the new site stabilized, we decided to reintroduce the lead form interstitial in order to increase our overall lead volume. We made some slight modifications to avoid interfering with our new lead product. We also simplified our deployment by removing Vowpal Wabbit from the ensemble. This way, all model scoring was performed by a single web application artifact. Finally, we reduced the target rate for the interstitial. We found that by targeting only the top 5% of visitors who were most likely to submit a lead, we increased our lead conversion rate by 26% without negatively impacting bounce rate, dwell time, or advertising revenue.

[8] All products encompass design and implementation issues. Effective products must be executed efficiently. (Minderhoud 1999). In our case, we are trading off effort to get the right product (aesthetic of interstitial) with getting the product right (accuracy of interstitial prediction).

9 Acknowledgments

We would like to thank Andy Wadhwa for contributing to our team before moving to Newegg.com and Steven Spitz for his invaluable assistance in the final editing of this paper. We would also like to acknowledge the invaluable contribution that Punnoose Isaac, Brian Terr, and Philip Potloff provided to the Propensity Modeling Team. Finally, we would like to thank Ely Dahan, Sanjog Misra, and Anand Bodapati from UCLA for helpful discussions.

References

Agarwal, A.; Chapelle, O.; Dudik, M.; and Langford, J. 2011. A reliable effective terascale linear learning system. *Computing Research Repository* abs/1110.4198.

Bottou, L. 1998. *On-Line Learning in Neural Networks*. New York, NY: Cambridge University Press. chapter Online Learning and Stochastic Approximation, 9–42.

Chester, J. 2012. *European Data Protection: In Good Health?* Springer Science+Business Media. chapter Cookie Wars: How New Data Profiling and Targeting Techniques Threaten Citizens and Consumers in the "Big Data" Era.

Deisy, C.; Baskar, S.; Ramraj, N.; Koori, J. S.; and Jeevanandam, P. 2010. A novel information theoretic-interact algorithm (it-in) for feature selection using three machine learning algorithms. *Expert Systems with Applications* 37(12):7589–7597.

Dietterich, T. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems, First International Workshop*. IAPR.

Dzeroski, S., and Zenko, B. 2004. Is combining classifiers with stacking better than selecting best one? *Machine Learning* 54(3):255–273.

Eirinaki, M., and Vazirgiannis, M. 2003. Web mining for web personalization. *ACM Transactions on Internet Technology* 3(1):1–27.

Gao, J.; Ding, B.; Fan, W.; Han, J.; and Yu, P. S. 2008. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing* 12(6):37–49.

Goodman, S. N., and Royall, R. 1988. Commentary: Evidence and scientific research. *American Journal of Public Health* 78(12):1568–1574.

Graczyk, M.; Lasota, T.; Trawinski, R.; and Trawinski, K. 2010. Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In *Second international conference on Intelligent information and database systems: Part II*. ACIIDS.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research* 3:1157–1182.

Hand, D. J., and Yu, K. 2001. Idiot's bayes – not so stupid after all? *International Statistical Review* 69(3):385–398.

He, X.; Duan, L.; Zhou, Y.; and Dom, B. 2009. Threshold selection for web-page classification with highly skewed class distribution. In *18th International World Wide Web Conference*. ACM.

Kaliakatsos-Papakostas, M. A.; Epitropakis, M. G.; and Vrahatis, M. N. 2011. Weighted markov chain model for musical composer identification. *Applications of Evolutionary Computation* 334–343.

Kivinen, J.; Smola, A. J.; and Williamson, R. C. 2004. Online learning with kernels. *IEEE Transaction on Signal Processing* 52(8):2165–2176.

Koren, Y. 2009. The bellkor solution to the netflix grand prize. Technical report, Yahoo! Research.

Langford, J. 2007. Wiki for vowpal wabbit project. https://github.com/JohnLangford/vowpal_wabbit/wiki.

Langley, P.; Iba, W.; and Thompson, K. 1992. An analysis of bayesian classifier. In *The Tenth National Conference on Artificial Intelligence*. AAAI.

Leather, H.; Bonilla, E.; and O'Boyle, M. 2009. Automatic feature generation for machine learning based optimizing compilation. In *International Symposium on Code Generation and Optimization*. IEEE.

Li, D., and Wang, H.-Q. 2012. A markov chain model-based method for cancer classification. In *8th International Conference on Natural Computation*. IEEE.

Liu, Y.-W., and Selfridge-Field, E. 2002. Modeling music as markov chains: Composer identification. Technical report, Stanford.

Minderhoud, S. 1999. Quality and reliability in product creation - extending the traditional approach. *Quality and Reliability Engineering International* 15(6):417–425.

Moe, W. W., and Fader, P. S. 2004. Capturing evolving visit behavior in clickstream data. *Journal of Interactive Marketing* 18(1):5–19.

Moe, W. W. 2003. Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of Consumer Psychology* 13(1&2):29–39.

Rish, I. 2001. An empirical study of the naive bayes classifer. *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence* 3(22):41–46.

Saeys, Y.; Inza, I.; and Larranaga, P. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics* 23(19):2507–2517.

Saffari, A.; Leistner, C.; Santner, J.; Godec, M.; and Bischof, H. 2009. On-line random forest. In *Conference on Computer Vision and Pattern Recognition*. IEEE.

Saffari, A.; Godec, M.; Pock, T.; Leistner, C.; and Bischof, H. 2010. Online multi-class lpboost. In *Conference on Computer Vision and Pattern Recognition*. IEEE.

Smale, S., and Yao, Y. 2006. Online learning algorithm. *Foundations of Computational Mathematics* 6(2):145–170.

The R Development Core Team. 2012. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Ting, K. M., and Witten, I. H. 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research* 10:271–289.

United States Census Bureau. 2010. Zip code statistics. http://www.census.gov/epcd/www/zipstats.html.

Vogel, D.; Brierley, P.; and Axelrod, R. 2011. Heritage provider network health prize: How we did it. Technical report, Team Market Makers.

Wang, G.; Hao, J.; Ma, J.; and Jiang, H. 2011. A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications* 38(1):223–230.

Ye, N. 2000. A markov chain model of temporal behavior for anomaly detection. In *Workshop on Information Assurance and Security*. IEEE.

Zhang, J.; Marszałek, M.; Lazebnik, S.; and Schmid, C. 2007. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision* 73(2):213–238.