Top-down Forecasting Using a CRM Database

Gino Rooney | Tom Bauer

Abstract

More often than not sales forecasting in modern companies is poorly implemented despite the wealth of data that is readily available to the sales executives. Third party services such as Salesforce can be used to track and monitor this data. Given access to this data, it is possible to implement machine-learning algorithms that can generate more accurate and more statistically significant predictions.

Introduction

ROAM is an early stage start up company that uses a big data platform along with a company's CRM (Customer Relationship Management) system to make predictions on sales opportunities in the pipeline that spans from the point of contact until the closing of a sale. In general, all of this information is contained in the CRM database. With access to this CRM data, ROAM uses machine learning algorithms in order to predict what will happen to unclosed sales within a company's pipeline. This is useful because it informs a company about the deals that deserve the most attention (i.e. which are more likely to close) and allows them to then forecast better. For our project, we assumed that the uncertainty of close problem had been solved. Some pertinent predictions for given opportunities are whether or not a sale will close and when, as well as the amount that sale will close for. While ROAM already has the capability to make predictions on the likelihood and timing of an opportunity closing, they have not yet been able to make efficient predictions on the value of a closing deal. Once this value can be predicted, forecasts for a period can then be constructed by aggregating the predicted values of sales that will close in that time frame. Thus, our work for this project focuses primarily around solving this value regression problem. Ultimately, constructing more accurate forecasting models would provide a company with a more statistically significant indicator of how to properly budget their finances and make strategic decisions.

Opportunity Pipeline

For data acquisition purposes, it's important to understand the basics of how an opportunity evolves within the pipeline of the system. Figure 1 shows an example of the evolvement of an opportunity in the pipeline.

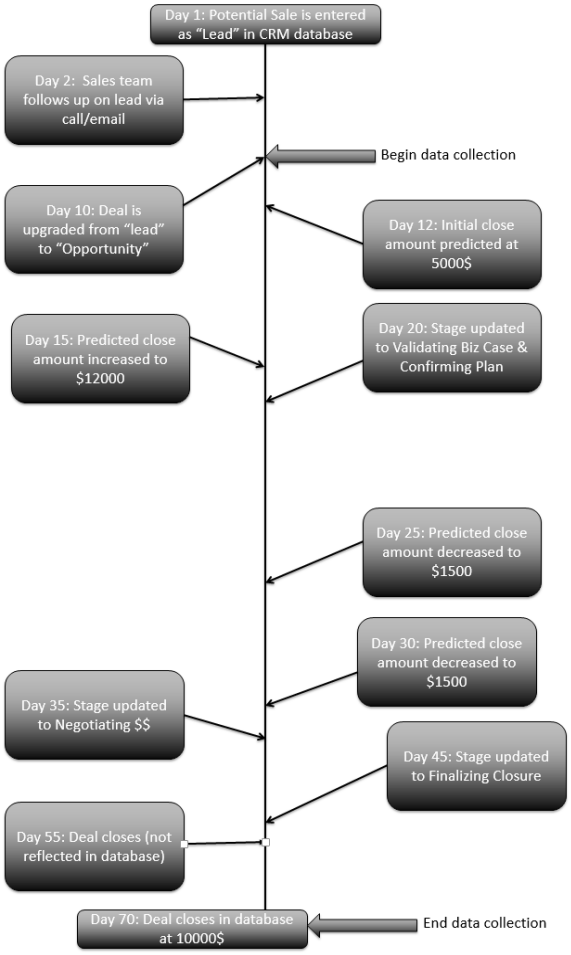


Figure 1. Timeline of an opportunity in the CRM pipeline.

Initially, every opportunity enters the system as a "lead". Leads can be acquired through various forms, such as marketing, customer referral, or cold calls. The sales staff personally follows up on these leads and if a lead looks as if it is going to develop into a potential sale then it is converted to opportunity in the pipeline. Data acquisition begins once the lead has been converted; this is considered

the start date for an opportunity. At this time, the deal is given an initial close amount prediction, which is one of the first features we considered important to make predictions. From this conversion point on, the database tracks a large amount of information, corresponding to events associated with the evolvement of this opportunity. Eventually the opportunity will close with a "won" or a "loss" status, indicating the sale was either completed or fell through, respectively. For the purposes of predicting close amounts, we were only concerned with opportunities that closed as "won." When examining opportunities in the database, it quickly became apparent that the data had inaccuracies that are a direct result of the environment that the sales team has to work in. As shown in figure 1, a deal can close in reality, but if the sales team has already met their quota, they will often sit on the deal and put off updating it in the system. This is one of the many types of discrepancies that the database is riddled with that inhibit models from making more accurate predictions.

Data Acquisition

CRM databases are renowned for being unorganized and prone to data discrepancies introduced by the inconsistencies of data entry by the humans entering information into the database. This type of database is intended for humans to read, not for automated systems to interpret. This human element introduces pervasive noise in the database that must be addressed in order to collect meaningful data. To make meaningful predictions using machine-learning algorithms, data must be extracted using a method that takes this sort of database environment into account. Our data extraction pipeline implements carefully constructed python scripts to filter through the CRM database and extract features. The chosen features are selected by starting from a date of interest in the past. This allows us to obtain information about what will actually happen to current opportunities in the pipeline and construct training and test set to compare our predictions against.

Machine Learning Algorithms

L1 Lasso Regression

The first regression algorithm that was tried was Lasso (Least Absolute Shrinkage and Selection Operator) regression. This linear modeling method minimizes the mean squared error subject to a constraint on (or a penalizing regularization term alpha) the sum of the absolute size of the regression coefficients. This method is useful when needing automatic feature selection because the constraint on the sum of the coefficients can force some of them to zero. Because our initial feature size was much greater than number of examples that we have ($n>>m$), we thought Lasso regression would be useful in finding a model based only on the most important features.

k-Nearest Neighbors

K-Nearest Neighbors (kNN) classification was examined due to its simplicity and the complexity of the distribution of the data. At its core, k-NNR assigns values based on the average close value of the k training examples that are closest in the feature space. In testing, we searched for the optimal k that produced the lowest root mean squared error (RMSE).

Decision Tree (DT) Regression

This algorithm was used because it does not require much data preprocessing, and can perform efficiently with large data sets. DT regression uses interior nodes based on the input features to classify predicted values. In using this model, we varied the maximum depth of the tree (proportional to # of interior nodes) to find the optimal model. A consequence of using this regression technique is it produces discontinuous piecewise constant models as seen in our results.

Feature Analysis

The initial training and test set that was analyzed pulled thousands of features (>number of training examples) corresponding to each open opportunity in the database at a given date in the past. While some of the entries reflect actual values (predicted close amount, probability of close, etc.), many of the

features are a discretization of updates to the opportunity pipeline. For such sequences of updates, n-grams are used to assign value to these transitions.

Predictive models were developed using this initial training and test set and the machine learning algorithms above. Figures 2 and 3 below show the test and training error for one of our lasso regression models. Our models were computed in log-space (as opposed to dollar space). This was done to transform our target variable to be approximately Gaussian distributed, which fits the assumption of our squared loss linear regression models. In this case, our target variable in dollar space was very non-Gaussian, hence the transformation. When examining errors, predicted values can be easily converted back to dollar space as seen in the Analysis section.

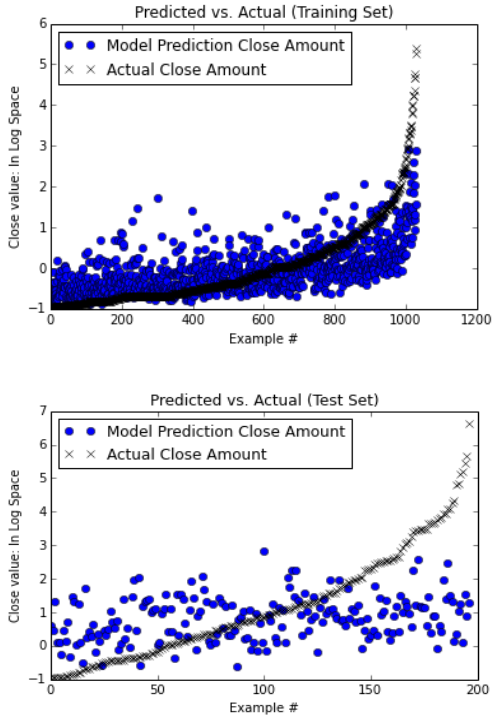


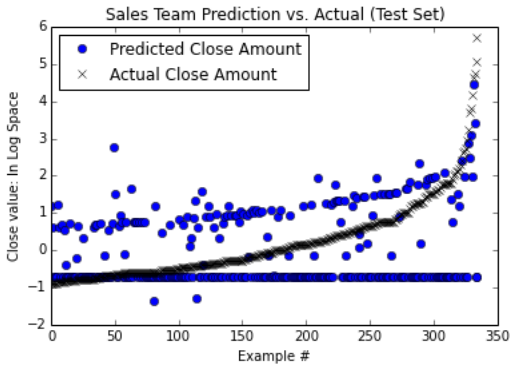
Figure 2-3. Model Predictions using Lasso Regression

Our initial tests from our different models led us to believe that our models were suffering from an under-fitting problem. To resolve this, we sought out to reduce the number of features we used and apply more rigorous data preprocessing to reduce the sparsity and noisiness of our input features. Features were refined to remove human discrepancies or database inconsistencies. One key refinement we made was to adjust the way the database pulled predicted close values. We noticed that many database entries had initial predicted amounts of zero in the system. Usually, these entries were an artifact of the system and a real entry would follow with a created time that was minutely different. To account for this, we selected the first nonzero predicted close value. Also, the close amounts would occasionally be negative, and, assuming the rest of the data associated with this opportunity looked normal, the absolute value of the close amount was used. We also limited ourselves to a deals with a greater close amount than $5000; this was recommended by ROAM because removing the multitude of smaller deals from our analysis would have little effect on predicting the company's bottom line.

Baseline for improvement

An important feature in predicting the close value for a single opportunity was the initial predicted close amount for a deal in the pipeline. This represents the amount of money for which the salesman thinks an opportunity will close. This measure provides a good baseline test to compare our predictive models' results. After all, if the model does not make more accurate predictions than the salesman on the opportunity, than it is not any more useful than the salesman's predictions. Figure 4 below show the current predicted amount in the CRM and the actual close amount in both our training and test set of opportunity in log-space. The horizontal line represents opportunities that employees have entered with the same predicted close values but end up having different (in some case much different) actual close values.

Figure 4. Entered Predicted and Actual Close Amount for Training/TestSet



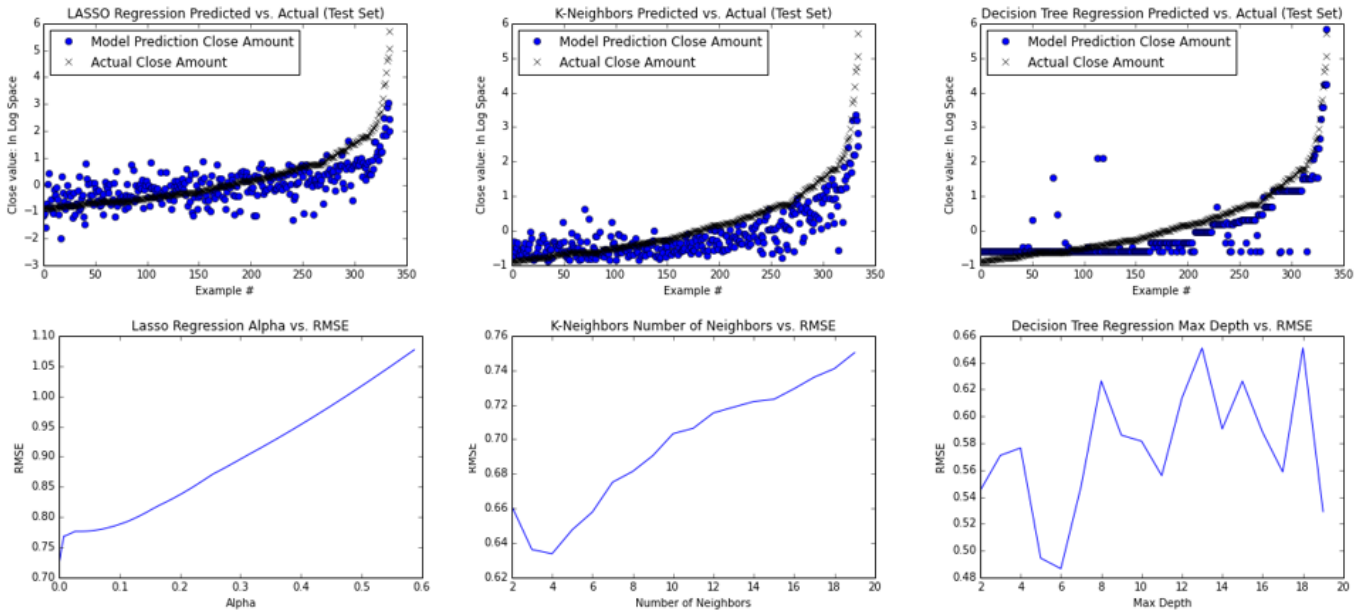


Figure 5. Results from predictive models and RMSE curves for Lasso, k-NNR, and DT regression.

The root mean squared error (RMSE) between predicted close amount and actual close was computed as an accuracy metric. Thus, in evaluating whether or not a new model is useful, we compare this baseline test RMSE and the RMSE of our new model on the test set.

Results

With our refined featured set, we reevaluated our different models and optimized to find the best regularization parameters for each algorithm. The results are shown in figure 5. For the Lasso Regression method, we used cross validation to find the optimal alpha in terms of RMS, which resulted in the plotted model below.

In k-NNR, we sought to find the ideal number of nearest neighbors k to consider, and, in decision tree regression, we varied the maximum tree depth to find the model that would produce the smallest RMSE.

Analysis

In refining the features, we went from an extremely large feature set (>2000) to a feature set of 8. We wanted to be certain of the validity of each feature we added to our model and worked diligently to remove any discrepancies or noise in each of the features we were adding. Ultimately, we put together a data extraction and filtering script to pull 8 clean features we thought would be relevant to solving this value regression problem. The majority of these features were derived from predicted close amounts, the probability of a sale closing, the time the opportunity had been open, and the owner of the opportunity (salesman). In hopes of further improving models with our refined feature set, we tried feature expansion via inner products. To do this, the feature set was expanded by taking the inner product of each feature with itself and the rest of the features, effectively bringing us from 8 features to 64 features. By doing this, all of the machine learning algorithms we implemented saw an improvement when using this expanded feature set.

The table in figure 6 shows the results of feature expansion across our optimized models in terms of both log-space and dollar-space. For dollar-space analysis, we used the mean error because we thought it would be more tangible for executives who analyzed such predictions. From our models, we found that decision tree regression provided the most improvement in terms of both error metrics. In comparison to the baseline test of employee predictions, our decision tree regression model was able to reduce the mean error by a factor of two in dollar space.

	Original Feature Set(8)		Expanded Feature Set(64)	
	Test Set RMSE	Dollar Space Mean Error	Test Set RMSE	Dollar Space Mean Error
Base Line	1.2607	$10,512.22	1.2607	$10,512.22
Decision Tree Regression	0.515	$5,072.50	0.4867	$3,909.50
K-Nearest Neighbors	0.6389	$6,936.88	0.6338	$6,806.48
Lasso Regression	0.7691	$7,993.64	0.7239	$7,644.67

Figure 6. Table showing results from original feature set and expansion

Conclusion

Ultimately, we were able to significantly improve upon the employee predictions for the closing value of a sale. This was done by going through several iterations of acquiring data, implementing machine learning algorithms, and then refining those algorithms. In going through this development process we became aware that working with real data has certain inherent challenges and that these can often be a product of the environment in which the data was created. In going through the database, we were able to provide ROAM with some useful information that helped them find discrepancies in their own data extraction pipeline. We believe there is still room to improve our results in the future. Despite the amount of time we spent on data extraction and noise filtering, there still seems to be some outliers in the data. These outliers could be actual outliers or an undiscovered database dependency. These results could possibly be further refined if these outliers were discovered to be a discrepancy in the database. Also, these predictions might be improved with the incorporation of additional clean features.

Originally, we started off by assuming the close problem was solved. Now, ROAM can include this work in combination with the solution to that problem to make top down forecast predictions.

Acknowledgements

We would like to thank Roam Analytics, Andrew Maas, Kevin Reschke, and Joe Barefoot, for providing us with an opportunity to work on a real problem that involved using real world data.

References

Duda, R., Hart, P., and David Stork, (2001), *Pattern Classification*, 2nd ed. John Wiley & Sons.

Hastie, T., Friedman, J, and Tibshirani, R., (2001), *The Elements of Statistical Learning*, New York: Springer-Verlag.

Tibshirani, R (1996). *Regression shrinkage and selection via the lasso*. J. Royal. Statist. Soc., Vol. 58, N0. 1, pages 267-288)