Georgia Tech OMSA Capstone Project: Final Report

Needle in the Lead Stack: Targeting the Best Sales Leads using Machine Learning and Applied Analytics

Zach Olivier

Tricentis

Project Background: What is the Pain Point?

Tricentis is a sophisticated technology company aiming to sell automated software testing and DevOps services to companies looking to speed up their internal development process. Tricentis is one of the prominent leaders in automated testing software, and have earned a reputation as one of the thought leaders in this rapidly evolving space.

Due to this Tricentis has been extremely successful with products sold worldwide in a multitude of different verticals. Through trade-shows, conferences, online marketing Tricentis has been a catalyst to the DevOps recent DevOps movement. Tricentis enjoys a large funnel of potential clients that is showing no signs of slowing down.

So what is the problem?

With all these leads “in the funnel” sales person resources can be used up or wasted extremely quickly. Every minute spent trying to contact a bad lead is a minute not working with a good lead. Sales teams cannot possibly reach out to every single lead in a timely manner. This is compounded by a longer sales cycle, and the fact that many leads interact with Tricentis to learn about the DevOps movement and best practices - and are not ready-to-purchase clients.

Tricentis ultimately needs a way to score and rank users based on their likelihood of conversation to a paying customer. Having an accurate system that does this will ensure sales teams are working with the “hottest” leads, but also given management confidence that good leads are not being turned away due to a full pipeline.

This capstone project final report will chronicle the end to end development of this lead scoring system - driven by machine learning and applied analytics.

Current State

The problem of lead scoring is not a new one to Tricentis or SaaS companies. Tricentis has developed an automated heuristic-based model that current scores leads and routes them to sales teams based on their scores. Business would like to explore a model based solution for this very same scoring system. Having an accurate model sitting on “top” of the sales funnel drives the entire lifeblood of the business. Even a slight improvement to the heuristic model can mean magnitudes of value created in saved time, sales person focus, and provide a path to growing the most qualified leads.

Project Goals - What does success look like?

Goal of this project is to develop a model based lead scoring system - to be used to filter which leads should be passed to the Tricentis sales team for follow up. To deliver a great solutions we have to meet the following requirements:

- Models should be rooted in business logic - business teams need a model that allows for inference and interpretability. Standard business logic should be encoded in the model. Convoluted or “black box” models will not be successful - not only do we want an accurate scoring system - we need to understand the drivers of the best leads.
- Models should not be constrained to only the business logic features. Business wants to explore deeper solutions than a simple heuristic model and see if there exists an optimal point between accuracy and inference.
- Models should be extensible and easily integrated with current systems. Having a great model on a local computer does nothing for the business aiming to route leads to sales teams in near real time.

If these goals of an accurate yet interpretable, business relevant and extensible model are met, this solution will solve for a major business pain point located at the core of the business model.

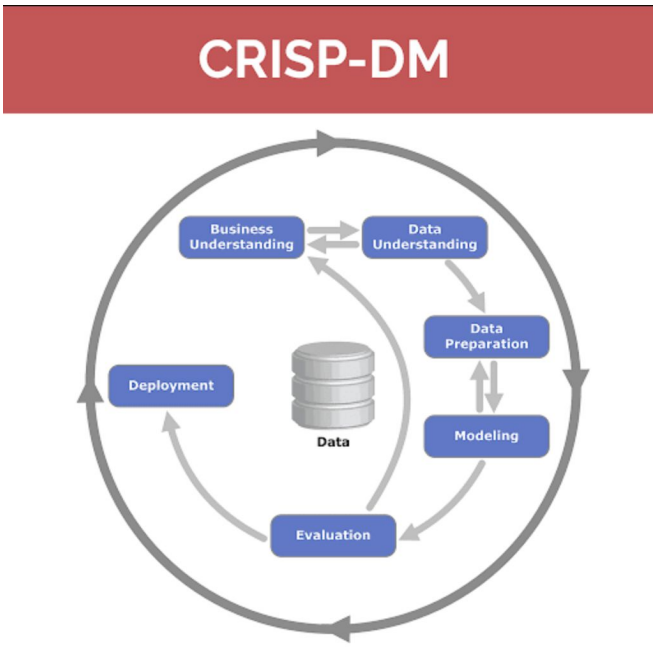
Project Overview: Summary

Below are all the steps taken to arrive at the lead scoring solution. The rest of this report will dive into each module for a detailed review of the methodology, considerations, experiments for each.

- Business and Problem Understanding
- Data Understanding
- Data Cleaning
- Model Experiments
- Business Recommendations

These steps closely follow the **Cross-Industry Standard for Data Mining (CRISP-DM)** which aims define a business problem, source data to support or solve that problem,

understand and clean the data, use the relevant data to build a model, and finally productize the system to drive automated decision making.



Business Understanding

Business understanding comes from talking with subject matter experts discussing the problem and relevant context, and also includes a healthy dose of outside research.

To get this relevant context we met with multiple Tricentis business stakeholders who discussed everything from the data landscape, current state, project objectives, business objectives and more. These meetings provide a deeply internal viewpoint of the problem that helps guide the solution. Scouring the Tricentis website and client content was extremely beneficial to understanding the market, tactics, and competition.

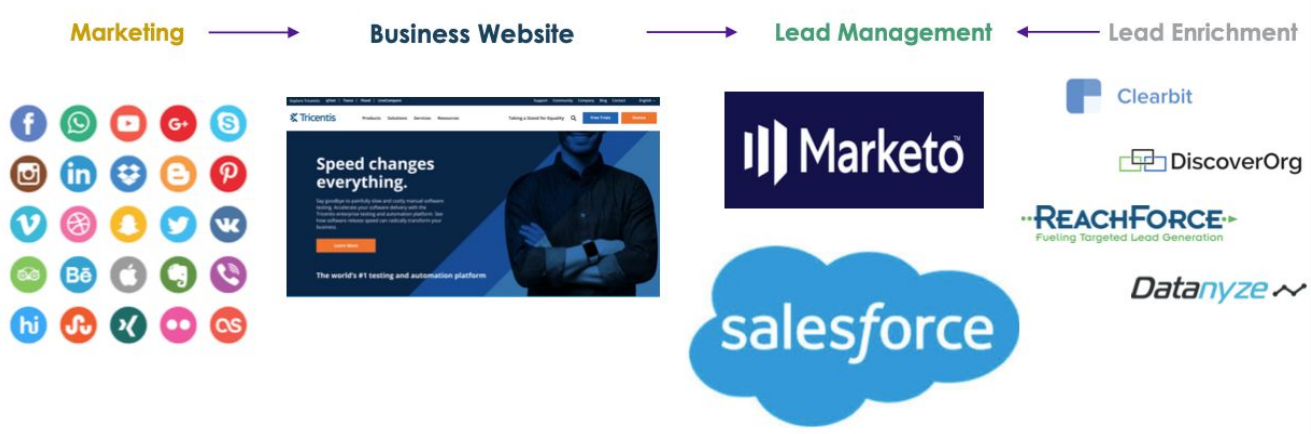
Outside research allows us to get a fresh perspective on the solution. To get this information we consulted various sources of technical and non-technical information from leading companies in the marketing consulting and data science space. Companies like Marketo, Salesforce, HubSpot are all aiming to sell CRM and sales funnel management solutions, and have great content to help understand how a sales funnel works and how lead scoring fits into the bigger picture.

One of the biggest sources of learning came from “walking the path” of a potential client. Start by finding Tricentis awareness content online, interact and click through to the conversion landing page, register your email and become a lead, and then see how the automated and personal follow-ups happen after based on other engagement with Tricentis (due to a lot of clicking on the website - I likely looked like a great lead).

All these learnings combined into a set of business fundamentals that help frame our solution:

- **Marketing:** content that engaged users and then entices them to a landing page
- **Business Website:** landing page, resources, information aiming at converting an anonymous website visitor to an official sales lead (provide email through a form)
- **Lead Management:** Once you are known as a lead, Customer Relationship Managers kick off automated marketing and eventually a sales person call to start the sales cycle
- **Lead Enrichment:** Vendors provide data to append to the lead record based on email - which is used to understand who you are as you continue down the sales path

These fundamentals combine as the **marketing-sales funnel** for almost every SaaS company in existence. Our solution will sit between the upper funnel (Marketing and Website) and the start of the lower funnel (Lead Management and Sales Person contact). **The goal is to score how leads will perform in the lower funnel (closed sale) based on their interactions in the upper funnel.**



Data Understanding

All the data we have to support our solution is generated from the marketing-sales funnel. With a great mental model of a client's path to conversion, we can tap into sources at each step to correlate to sales conversion.

- **Marketing (Upper Funnel):** data points such as the marketing channel (Google, Facebook, Linkedin etc), the marketing tactic (awareness, targeted specific, retargeting) and channel path
- **Business Website (Upper Funnel):** webpage visits, content clicks, form fills, demo registrations, page bounces, content downloads, whitepaper requests, videos watched, interactive chats, time on pages, page order, session length are all rich features that are generated by each lead and they move down the funnel
- **Lead Management (Lower Funnel):** Lead source, lead activity, automated marketing sent, automated marketing clicked, sales interactions, and custom lead segmentation are all available at this phase in the funnel
- **Lead Enrichment:** Additional attributes appended to the lead record include location, business attributes, business type, annual revenue, number of employees, years in business, industry type all add to the rich data we can use for our solution.

Data related to activity in the upper funnel is labeled as lead engagement features - these tell us what the client did on their path. Enrichment data can be labeled as the lead qualification features - these tell us who the client is and what they are like.

With this knowledge of the **marketing-sales funnel** and the combination of **lead engagement** and **lead qualification** features we can see our solution starting to form. Our goal would be to find the mapping between engagement and qualifications and a closed sale.

Data Cleaning & Feature Engineering

With our mental model of the solution in mind and access to all data within the marketing-sales funnel - our next step is to clean this data. Cleaning data means formatting the data in a way that allows for analytics and machine learning. Typically this is the most time consuming and frustrating component of a data science project.

Below is a summary of the most important data cleaning steps for this project. Highlighted are the sections that deserve a deep dive in the following sections.

Technique	Why do we need to do this?
Missing Values Strategy	Models cannot handle missing values
One Hot Encoding	Models cannot handle categorical variables
High Cardinality Encoding	Categorical Variables can have thousands of categories
Time Based Features	Calculate time between specific touchpoints, content
Channel / URL / Path Definitions	Order matters - develop a numerical version of the customer path through the funnel
Channel / URL / Path Segmentation	Websites change - using explicit URLs or page names will break the model down the road
Target Definition	Define through the data what is our target
Common Join Key	Need to link all components of the marketing sales funnel together into one dataset
Censoring	Using data that comes after a conversion will leak information into our model

Data Cleaning: Missingness as Information

Data available from the marketing-sales funnel is extremely sparse. There are many different paths a client can take: imagine all possible paths with multiple channels, web pages, content, and forms. With all these channels most customers only go through a select few, leaving most of our data missing. Enrichment data for certain variables can be up to 95% missing.

Common ways to handle missing data include imputation methods, such as filling values with mean, or modeling the values using available data as features. With such sparse data imputation techniques can fall flat - with so much missing almost every record will get the mean value. Modeling missing data using other sparse missing data causes the same problems.

To combat this we encode missingness as information to our model. Numerical missing values are filled with -1 and a secondary column is added to that indicates if that value was originally missing. Categorical columns are expanded and include a missing dummy column.

Having indications of missingness allows our model to segment the observations and handle them natively. In our model experiments it turns out that this missingness encoding was one of the richest sets of features - if a client did not complete the form completely, or if enrichment data could not find any associated information - you more likely than not were not converted to a sale.

Data Cleaning: Order Matters

Data is usually available in tables rolled up to a common identifier. This is great for querying but does not handle time or order of events well. Features such as total forms fills, number of clicks, total links downloads are easy to grab in traditional formats. However, order of content can be extremely helpful in determining accepted leads.

- **Customer A:** Facebook > Resource Page > Google > Landing Page > Form Fill
- **Customer B:** Facebook > Landing Page > Whitepaper Download > Google

In aggregate Customer A and Customer B have similar counts of aggregate touchpoints, and each filled a form. But in order Customer A may have a more engaged path. The combination of having a Tricentis resource page, then a direct search in Google, followed by the form fill is an extremely direct path indicating the customer is moving through the content. Customer B path seems to be less engaged, likely pulled out of facebook by a content ad that requires an email to download.

To take advantage of this we need to encode the path into our dataset. This allows the model to find the best paths based on eventual conversion. To do this we analyzed the paths of all customers and found **90% of customers have a path less that 5 steps** for a given channel-webpage combination. With this we expanded binary variables that indicate what marketing channel and which content happened at each path “slot”.

Example for **Customer A:**

Path Slot 1 - Facebook	Path Slot 2 - Facebook	Path Slot 3 - Facebook	Path Slot 4- Facebook	Path Slot 5 - Facebook
1	0	0	0	0

Path Slot 1 - Google	Path Slot 2 - Google	Path Slot 3 - Google	Path Slot 4- Google	Path Slot 5 - Google
0	0	1	0	0

Content Slot 1 - Resource	Content Slot 2 - Resource	Content Slot 3 - Resource	Content Slot 4- Resource	Content Slot 5 - Resource
0	1	0	0	0

These path and content slot features are extremely parse but are a robust way of providing the path information to the model. In experiments, path slot order showed that the earlier you engage with a resource or white paper, the more likely you are to convert to a sale.

Data Cleaning: Web Page / Content Segmentation

Websites and content on webpages are living documents. As time goes by, websites change, page functionality is updated, content is taken down, new content is uploaded. The fluid nature of the upper funnel makes building robust analytics solutions difficult. Including certain web pages or content in our model will work great now - but down the road will go stale or even break under the changes.

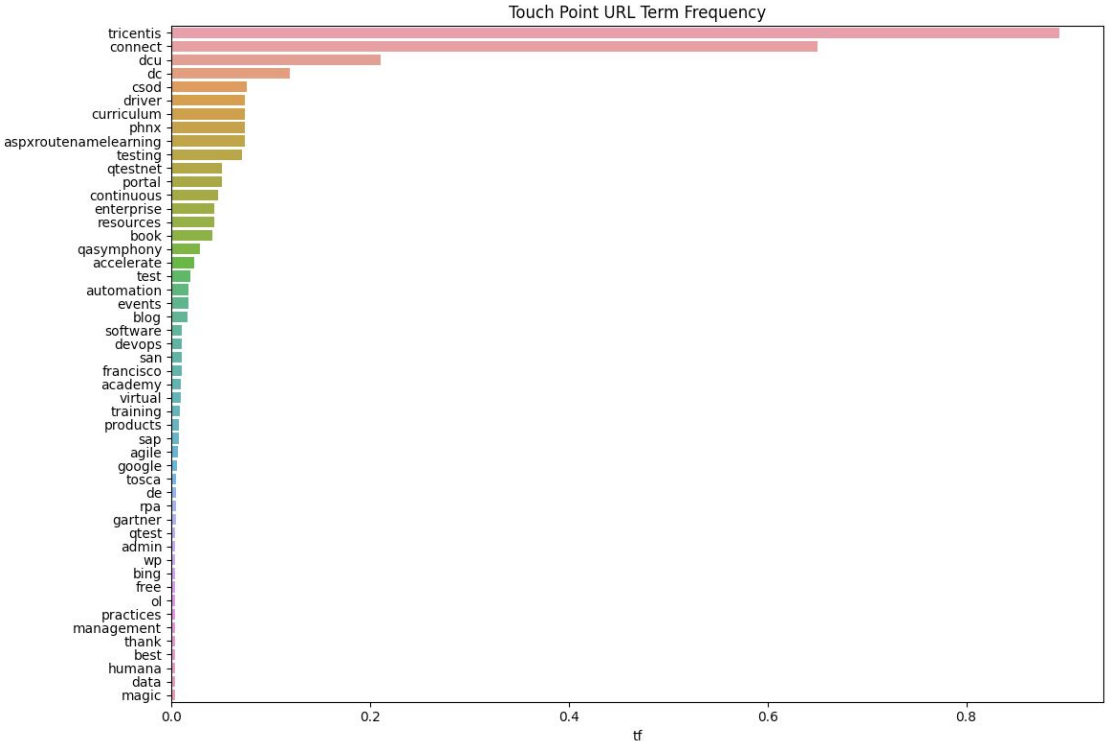
How do we counteract the future state of the website and content? We can group pages and content into meta-segments based on theme or tactic. Any future page can be bucketed into the same taxonomy of labels our model was trained on today.

To do this we can parse the urls of each webpage into specific groups like: resource, demo page, home page, conversion page, help page, contact page. For content, we

can examine the content titles and group them by theme: DevOps, Cloud, Tricentis, Marketing. In our case we employed simple Natural Language Processing (NLP) techniques such as Term Frequency (TF), and Term Frequency Inverse Document Frequency (TF-IDF) to help with the theme segmentation. Term Frequency was used to find the most common phrases out of all 100+ available content on the Tricentis website. The same methodology was applied to the set of all raw URLs from the Tricentis website. Term Frequency - Inverse Document Frequency was used to check for any rare themes that occur in both web page URLs and content titles.

With these common keywords, we now have a more robust way of handling future changes. Here are a few generic classification examples and results from the analysis:

Original	**Keyword Segmentation**
www.tricentis.com/resource/demo/SAP/	Resource web page, Demo Web Page
www.tricentis.com/e-book/devops/cloud/free	Content Page, Conversion Page, Free
What is DevOps?	DevOps content
Test Driven Development in SAP	Testing content, SAP content

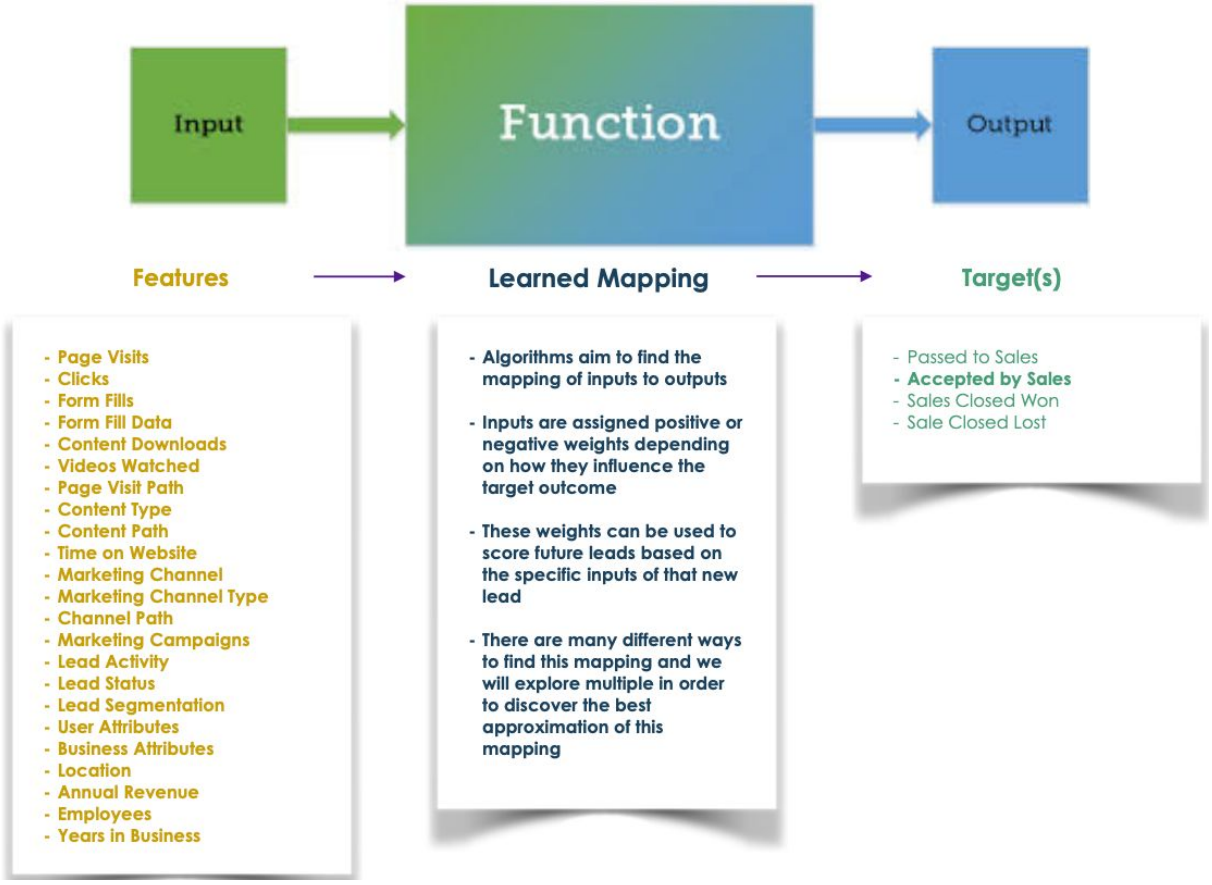


Model Definition

Now for the fun part - we have a solid business understanding, a vast amount of relevant data that is clean and ready for the modeling phase - we first need to define our model in terms of targets and features.

In a simple framework aiming at clear business understanding we propose the following:

- Any data science model, no matter how complex, is aiming to find a mapping from **inputs (features)** and **outputs (target)**
- Each model will have different assumptions and techniques to find this mapping
- **To solve our business problem - our model is defined as:**
    - **Target: If a lead was accepted by sales team as a valid lead**
    - **Features: All available data from the marketing-sales funnel**
- Once the mapping is learned from the features and the targets we can use the learned mapping to score new leads



Model Experiments

In this section we discuss all the model experiments including assumptions, performance, model tuning and final model selection. Following closely to the “No Free Lunch” Theorem - no one model is guaranteed to work best for any given data set - we set out to try a multitude of models each with different assumptions, and each offering a different take on the accuracy-interpretability trade-off.

Cross Validation: Setting up Fair Experiments

To judge multiple models we have to develop a testing strategy to objectively find the best model while guarding against overfitting. To do this we sample 80% of the entire dataset to further split into training and validation datasets. For our case, each model will be judged on a 5-Fold cross validation of the training set. The best model is selected from these experiments, then applied to the held out data for one final measure of out of sample accuracy. The final model is fit and all data available and is ready to be put into a production system.

Why do we do these steps?

Cross Validation is a way to balance model skill versus generalization. Model skill is the ability to predict correctly on the training dataset. Model generalization is the ability for the model to predict well on new data it has never seen before. Cross Validation will hold out sections of our dataset in order to estimate the generalization of our model. Our optimal goal is having a highly skilled model that also generalizes to new data extremely well - often these two ideas are at odds with each other. The more skilled a model, the more it overfits the patterns in the training data and fails to do well on the holdout data. Low skill models might not be as accurate but tend to be more stable when seeing new data. Cross validation gives us a great framework to test models and find the one that does the best in skill and generalization - one that optimizes the bias-variance tradeoff.

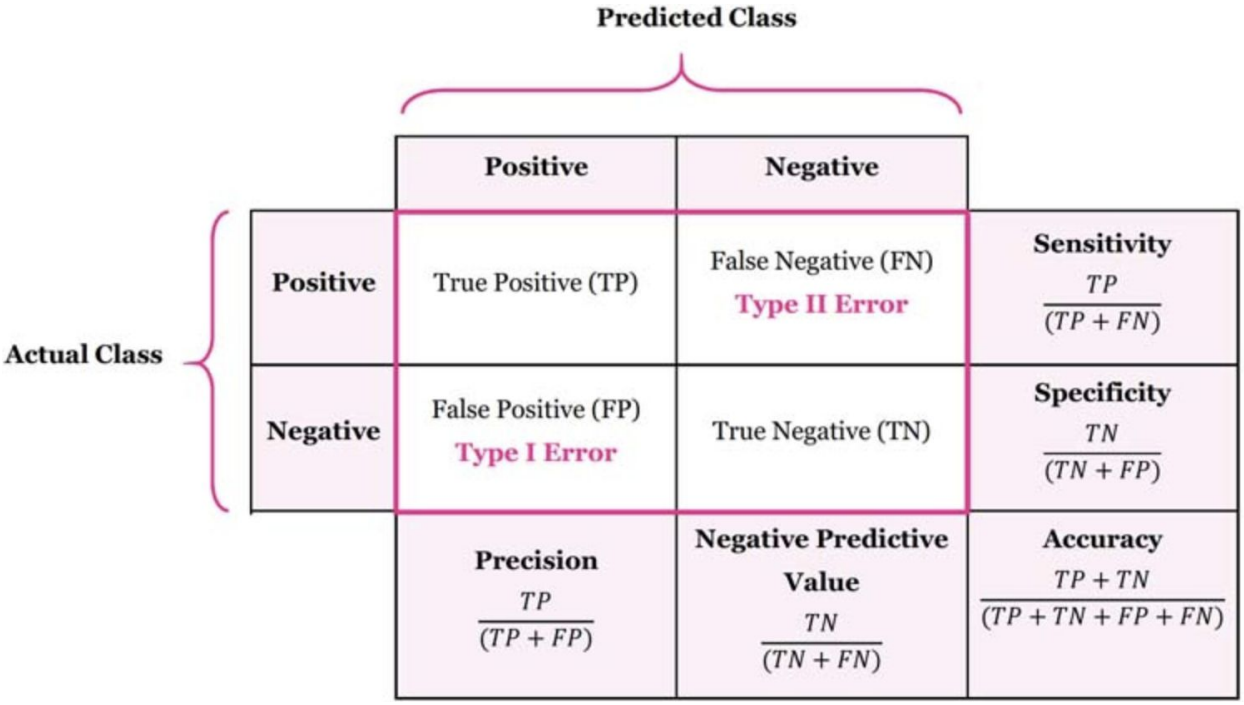
Model Judgement: Evaluation Metrics

How do we tell which model does the best through the cross validation experiments?

We will employ a host of metrics to gauge the performance of each model. Each model itself will have the same objective function - the function used to “fit” a model to the mapping - but a model can be judged by multiple metrics.

Our problem is set up as a classification problem where our model will aim to predict a label sales accepted or not sales accepted. There are several canonical metrics we can use to judge performance:

Metric	Description
Accuracy	# labels predicted correctly out of all labels
ROC-AUC	Plot TPR, FPR for classification thresholds
Precision	If we predict a label 1, how often are we correct?
Recall	How many label 1s are accurately classified?
Confusion Matrix	Summary of TP, TN, FP, FN
F1 Score	Harmonic mean between Precision and Recall



Imbalanced Data: Hard to Handle

Our data is heavily imbalanced - **the number of sales accepted leads represents less than 1% of all total leads**. Imbalanced data requires us to be especially careful in how we judge model performance.

Through the first iterations of the model tests we could easily achieve 99% accuracy - the model was completely useless. How is this possible? Models are flexible enough to take "shortcuts" in the name of getting the best accuracy - in this case the model figured out how to get an "easy" 99% accuracy by predicting every lead as not accepted by sales - by default only 1% are ever accepted! Using this model we would send no leads through to sales which is obviously not useful for business.

To handle imbalanced data we need to use several different techniques:

Technique	Description
Under Sampling	Balance dataset by random sampling the majority class
Over Sampling	Balance dataset by random sampling the minority class
SMOTE	Develop synthetic examples of the minority class
Class Weights	Penalize the model for getting minority sample wrong
Boosting Models	Handle imbalanced data natively, but requires threshold tuning

Based on model experiments - under, over, and SMOTE sampling did not generalize well in the cross validation experiments. The techniques that worked best were tuning the model class weights and using a Boosting Model to implicitly handle the missing data.

For the first three models the class weight penalty was kept the same at around 90:1 ratio of penalty of sales accepted to not accepted. This means we model will get penalized up to 90x more for misclassifying a minority class observation than a majority class observation.

Using class weights helped solve the imbalanced data issue but seemed to over-focus on the minority class. In this case the model was able to identify between 75-90% of accepted leads, but it needed to let in a large amount of False Positives to

get this accuracy! With only 1% of leads actually accepted, most models would perform badly by predicting up to 20% of leads accepted by sales.

Using these models, we would be letting in nearly 20X the amount of leads to the sales floor to capture only 80% of the accepted leads. This could be great news if the current model is strictly holding out a lot of sales accepted leads, but could be extremely detrimental to the sales team working through all of these leads.

Confusion Matrix	
61,491 (True Negatives)	13,915 (False Positives)
164 (False Negatives)	564 (True Positives)

Performance Evaluation: 5-Fold CV Average	
ROC-AUC	0.79
RECALL	0.77
PRECISION	0.04
ACCUACY	0.82
F1 SCORE	0.89
PREDICTED ACCEPTED	19%

Model Descriptions:

Data Science models aim to find the mapping between inputs and outputs. Each model has a set of assumptions and an algorithm to learn this mapping. Here we quickly explain all tested models and discuss each in terms of assumptions, expected accuracy, interpretability, and risk of overfitting.

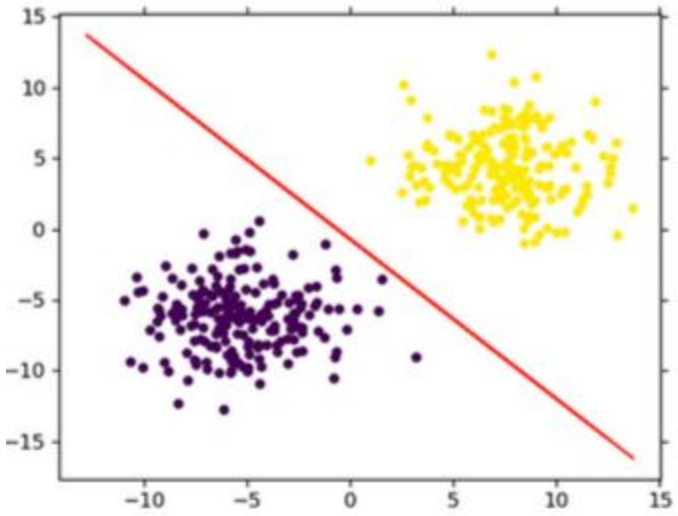
- **Assumptions:** what needs to be true for the model to work, if assumptions are not met model will give incorrect results
- **Accuracy:** Some models have allow more flexibility in mapping complex mapping functions, which can lead to better accuracy
- **Interpretability:** Select models allow insight into “why” or the specific “drivers” of its predictions, some allow no insights. There is usually a tradeoff in accuracy and model inference.
- **Overfitting Risk:** the more complex the model the more risk of overfitting to the training set and not generalizing well. More flexible models have higher risk.

Model	Assumptions	Accuracy	Interpretability	Overfitting Risk
Regularized Linear Regression	Strict	Medium	High	Low
Random Forest	Flexible	High	Medium	Medium
Neural Net	Flexible	High	Low	High
Boosted Trees	Flexible	High	Medium	Medium

Regularized Linear Regression:

Linear regression models require there to be a linear relationship between the mapping function. This is a very strict assumption in that most real world relationships are highly non-linear. If we can meet these assumptions focused on the residuals of the model predictions, linear models can perform extremely well. If assumptions are not met we will have an ill-fitting model.

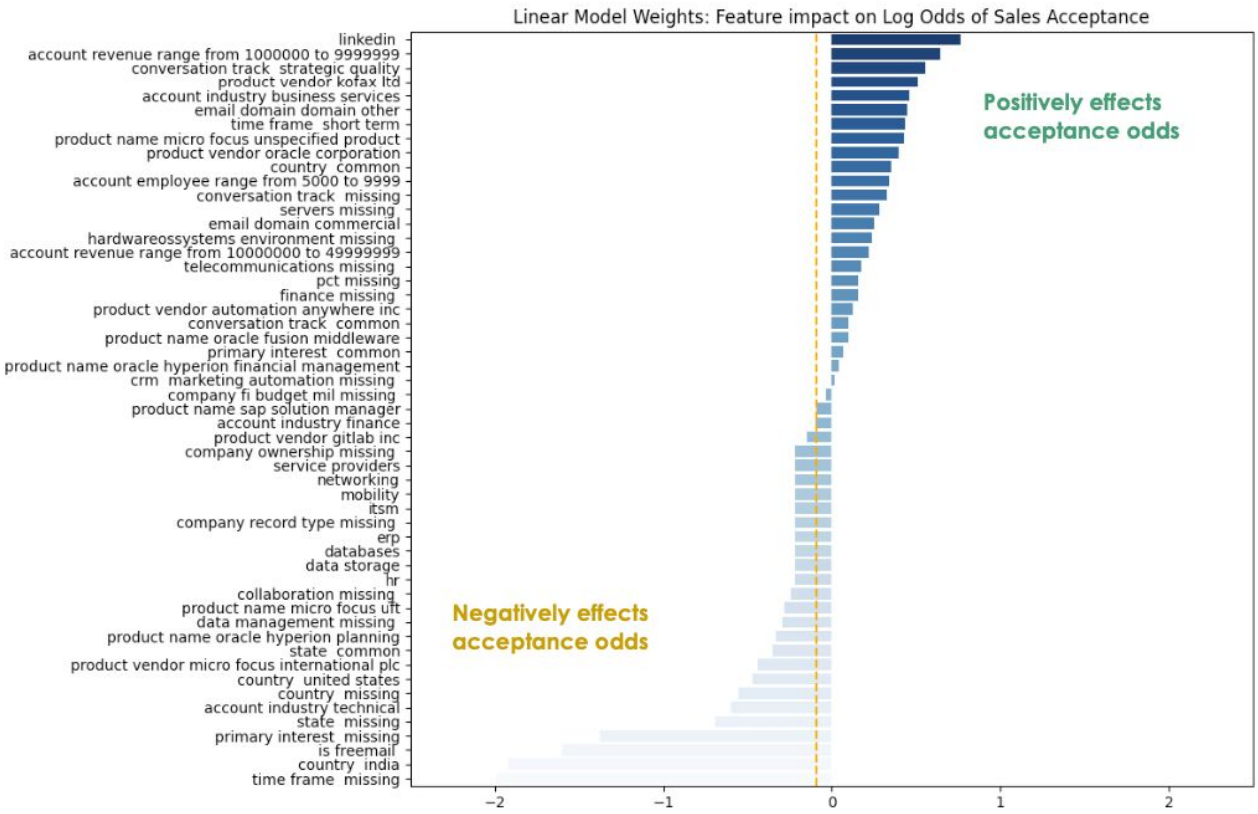
In our specific case we aim to find a straight line that best separates our two classification labels. This is done using a Logistic Regression model which generalizes the standard Linear Regression model to fit binary outcomes (classification problems).



One major benefit of linear models is their interpretability. By nature of finding a linear equation for our input-output mapping - we end up with a linear combination of the inputs to the outputs - every input feature will have a coefficient that provides the direction and magnitude of the effect on our target.

This allows for rich inference such as:

- **“Having a LinkedIn profile increases the odds of acceptance by a factor of two, holding all other variables constant”**
- **“Missing sign-up form time frame input decreases the odds of acceptance by a factor of four, holding all other variables constant”**



Inference into why the model is making certain predictions is directly applicable to the business. Sales teams can examine the features that positively affect acceptance odds as a quick-and-fast lead scoring algorithm. Marketing teams can examine the same coefficients and aim to target more leads likely to have the most important features of an acceptance.

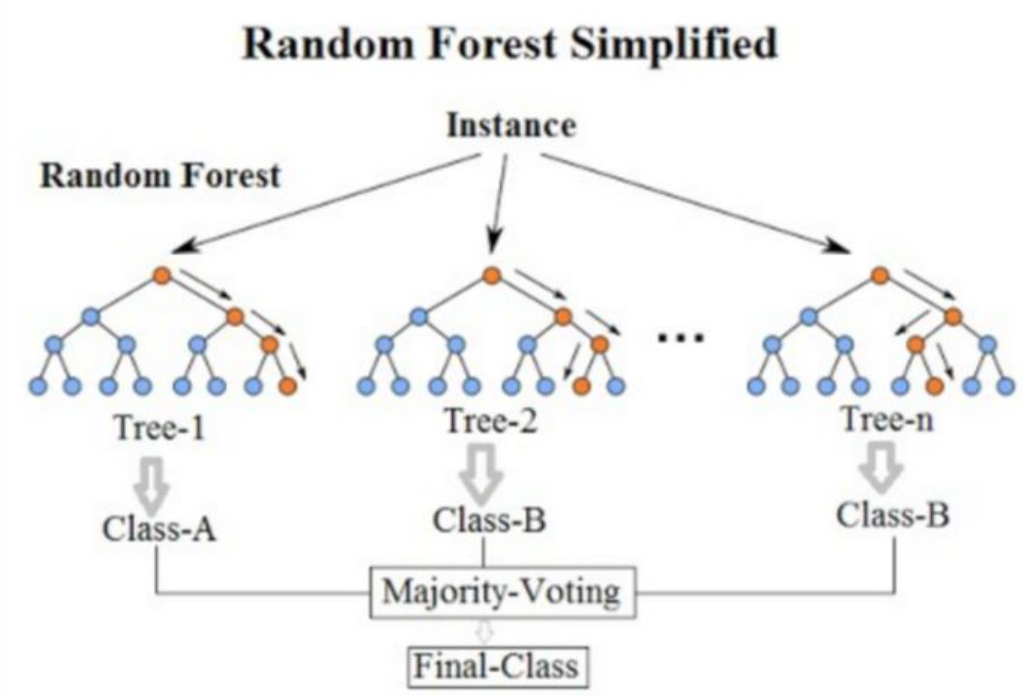
Random Forest:

Random Forest models are some of the most widely used machine learning models due to their ability to learn complex functions and provide some level of interpretability. Random Forests are non-parametric, they do not assume anything about the nature of the function mapping.

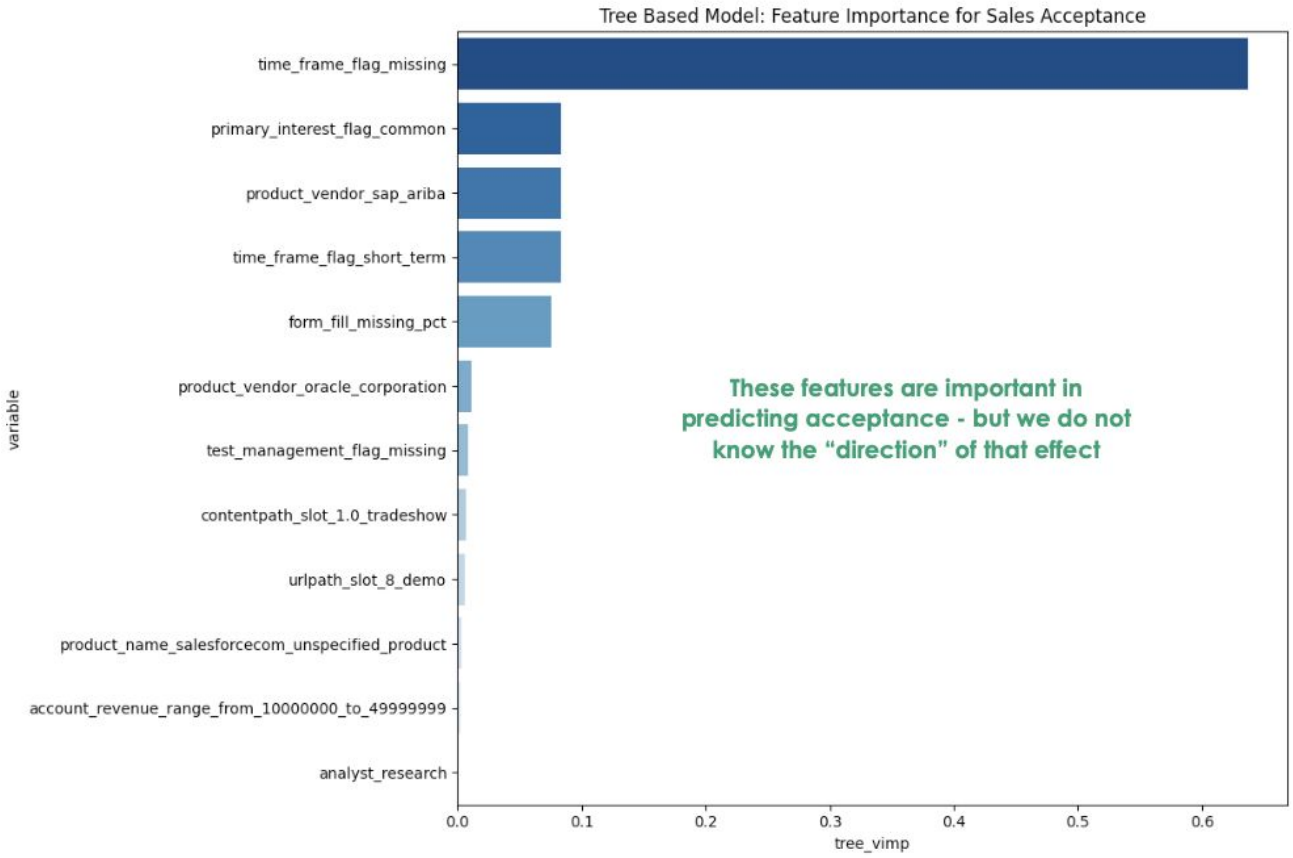
Random Forests work by randomly sampling the rows and the columns of our dataset and building the best “splits” to separate the positive and negative classes - called a decision tree. The decision tree build is analogous to a complex nested if-else statements developed by testing random splits and keeping the ones that separate the classes best. Iteratively, the columns and rows are samples and decision trees are built until there is a “forest” of unique decision trees all with their own unique decision rules. This ensemble of “experts” then all vote together on the class label, each “expert” knowing a subset of data and columns to cast their vote.

The ensemble nature of this algorithm allows for deeply flexible learning and guards against overfitting as the predictions are averaged out over a host of different decision trees.

- **“9 out of 10 decision trees in the random forest agree lead has high chance of acceptable = predict acceptance”**



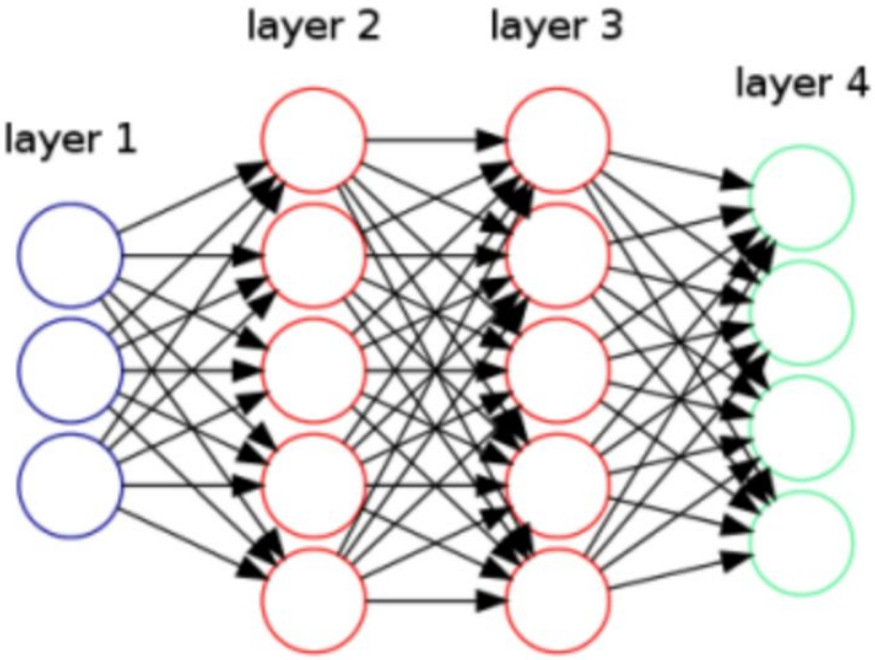
Due to how the Random Forest algorithm works we can also examine feature importance - an indication of what features are important to model predictions. Unlike Linear Regression we only have the importance, with no direction or magnitude. Here we can tell time frame is important but we do not know if it negatively or positively impacts the odds of acceptance. We only know that this variable is used a lot through many different splits of the ensemble.



Neural Networks:

Neural Networks are a class of deep learning algorithms that promise extreme flexibility in learning function mappings. However, interpretability is extremely low, and models of this sort are often called “black box” models.

Neural Networks are organized in a selection of interconnected neurons and layers of neurons. Observations with input features are passed through the network with random starting weights, but then passed backwards through the network based on how well those weights performed on the classification problem. Iterations of feed-forward and back-propogration update the weights based on the model feedback signal - the objective function the model is trying to optimize (example: classification accuracy).



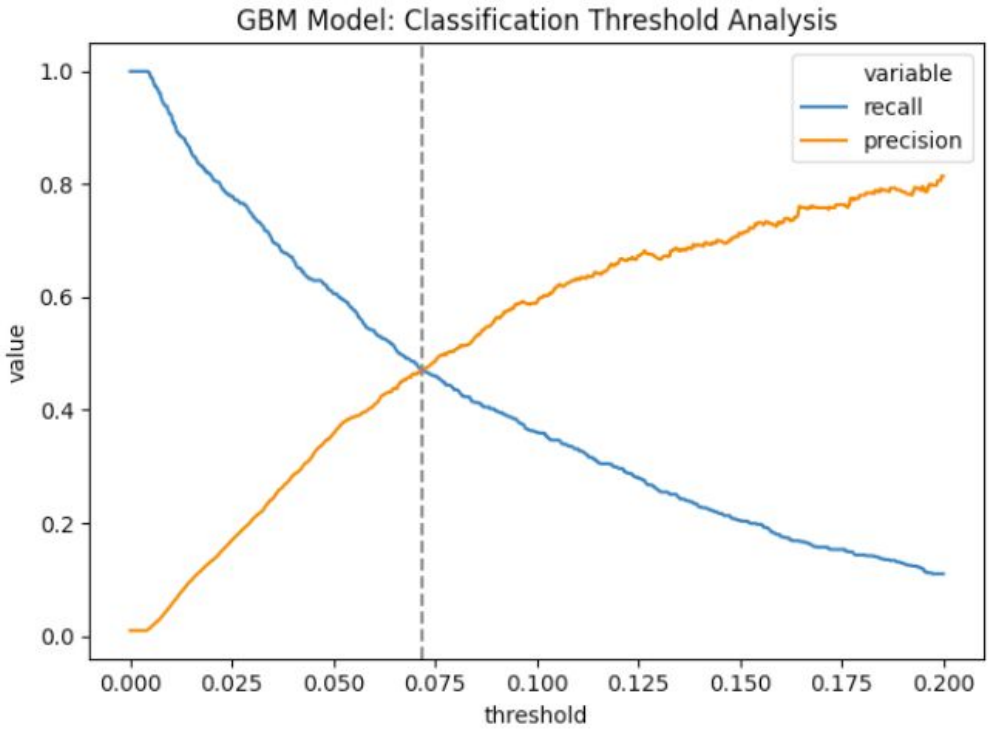
Due to the complex nature of this algorithm the risk of overfitting is extremely high. This risk combined with the lack of inference makes for a tough choice in business understanding. Most of the time, high accuracy is not enough to drive business trust of the results.

Boosted Trees:

Boosted Trees methods are very similar to Random Forest models - with huge improvement that helps natively handle imbalanced data. At each iteration of building decision trees the misclassified observations are re-weighted so that the next iteration has to “work harder” to classify the misclassified labels from the previous iteration. This re-weighting or penalizing of the misclassified observations is key to handling our imbalanced data. As iterations pass, observations that are easy to classify are mapped correctly, and the model is forced to focus on the harder-to-classify minority class.

The final step in this model is accurately tune the classification threshold. Most models output a probability of being a specific label, in our case, sales accepted or sales not accepted. Typically a threshold of 50% probability is set to flag an observation as a positive or negative label. With imbalanced data and our boosted tree model we need to find a different threshold to fit our data.

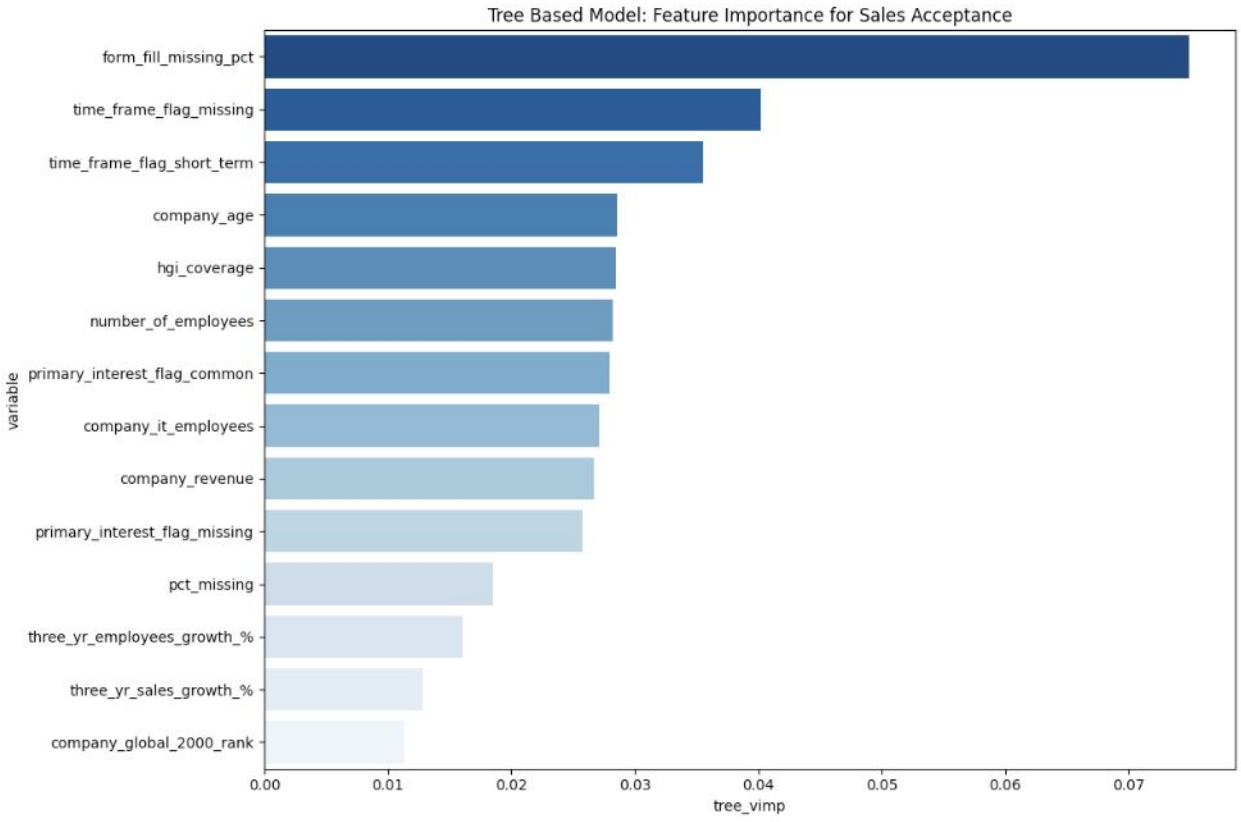
To find this threshold we can traverse the precision-recall curves to find the optimal threshold to call a label positive. The optimal threshold is around 8% which maximizes precision and recall of our model. A threshold of 8% probability of being accepted does not seem like a very strong lead, but this is eight times the true baseline rate of accepted leads (less than 1%).



The boosted tree model also provides inference into the predictions in the same way as the random forest, with a variable importance metric that highlights key variables but does not provide a direction or magnitude.

Top features that are important to the Boosted Tree model's predictions:

- **Form Fill Missing Percentage:** percentage of the form fill that has not been completed
- **Time Frame Missing:** indicator if the "looking to buy" section of the form fill was not completed
- **Time Frame Short Term:** indicator if the "looking to buy" section of the form fill was indicated in the next 1-3 months
- **Company Age:** years in business of the lead
- **HGI Coverage:** percent of enrichment variables complete for that lead



Model Comparisons:

In a side by side comparison across all models we can easily see that the Boosted Tree model performs the best on the metrics that are most relevant to the business.

- **Recall:** 75% of accepted leads were correctly identified by the model
- **Precision:** 21% of the time if the model predicts accepted the lead is actually accepted - this is nearly 5 times higher than all other models
- **Predicted Accepted:** The boosted tree model predicts 3% of all total leads will be accepted, with true baseline around 1%
- **Boosted Tree model objectively does the best job at balancing high accuracy without letting in a large amount of false positives**
- For business this means that if this model was used to send leads to the sales team:
    - **Leads sent to the sales floor will have a high probability of being accepted**
    - **We have high confidence that the leads not sent to the sales team by the model will not be strong leads - the model will not "hold out" many good leads**

Network Model		Tree Model		Linear Model		Boosted Tree Model	
Confusion Matrix		**Confusion Matrix**		**Confusion Matrix**		**Confusion Matrix**	
58,789 (True Negatives)	16,604 (False Positives)	65,559 (True Negatives)	9,830 (False Positives)	61,491 (True Negatives)	13,915 (False Positives)	73,350 (True Negatives)	2,066 (False Positives)
103 (False Negatives)	638 (True Positives)	260 (False Negatives)	485 (True Positives)	164 (False Negatives)	564 (True Positives)	182 (False Negatives)	536 (True Positives)
Performance Evaluation: 5-Fold CV Average		**Performance Evaluation: 5-Fold CV Average**		**Performance Evaluation: 5-Fold CV Average**		**Performance Evaluation: 5-Fold CV Average**	
ROC-AUC	0.91	ROC-AUC	0.88	ROC-AUC	0.79	ROC-AUC	0.86
RECALL	0.86	RECALL	0.65	RECALL	0.77	RECALL	0.75
PRECISION	0.04	PRECISION	0.05	PRECISION	0.04	PRECISION	0.21
ACCUACY	0.78	ACCUACY	0.87	ACCUACY	0.82	ACCUACY	0.97
F1 SCORE	0.87	F1 SCORE	0.92	F1 SCORE	0.89	F1 SCORE	0.98
PREDICTED ACCEPTED	30%	PREDICTED ACCEPTED	14%	PREDICTED ACCEPTED	19%	PREDICTED ACCEPTED	3%

Business Recommendations: The So What

All previous sections were a technical discussion on the progression of our model solution. Now we recap the explicit business recommendations from each step in the process. Great solutions can provide actionable recommendations for each phase of the CRISP-DM process, not just the model itself.

- **Require complete form fills:** data shows key sections of the form are important to classifying leads, more data at time of form fill will help future iterations of the lead scoring model
- **Send a random subset of all leads to the sales floor:** Currently we do not know how well the current model is performing as there is no baseline to compare. Letting in a subset of leads randomly will help judge any model and allow new models to learn real relationships not just the current model in production
- **Lead scores can be used for website optimization:** Model lead scores can be tagged on every website visitor - and those with high chances of acceptance can be retargeted - even without a form fill - by customizing web pages to change to landing pages for the highest lead scoring visitors
- **Website Engagement matters more than qualifications:** exploratory data analysis shows that website data is more important to identifying strong leads - team can invest in more content and more content paths to manufacture more touchpoints to identify strong leads
- **Offline leads need a score too:** Currently model only scores the leads through the website, tradeshow and event leads are divorced from website activity - investing in an online form for “offline” leads to complete collects more data, allows for a custom offline lead scoring, and can be normalize to the data collected on online leads
- **Segmentation of website pages, website content:** Segmentation of web content was a key piece in the project to ensure solution could work through future changes - current segmentation could be expanded to group website pages and website content by marketing tactics which could automatically update as needed
- **Use the model developed to send leads to the sales team:** Holdout performances indicate we have a great model to solve a challenging business problem - *our solution does a great job at balancing business costs (sending too many leads vs. holding good leads out) associated with finding the needles in the lead stack.*

Notes on a Production System:

Including in the project deliverables in a template to host our final model as an API to interact with other production systems. This simple API hosts our model as a python function through the Flask framework, which can then be hosted on a server with frameworks like Docker.

Here is the simple flow of our API:

- Model API is hosted online and “listening” for incoming web traffic
- When a new lead comes online, or a set threshold of activity is completed, production systems like Marketo can send a request to our online API for a score
- Upon the request, the Marketo system must include the inputs developed in the model, these need to be precomputed to be included with the request for a model score
- When the Model API receives the request, it will transform the incoming data, call the finalize model, then return a probability, a score, and a date back to the requestor
- Upon return, the Marketo system can then use the model scores to fill a field, create a custom list, or notify a salesperson