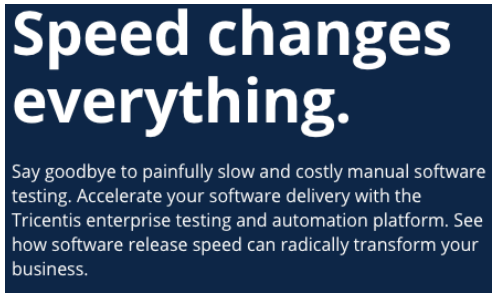
CAPSTONE PROGRESS REPORT

ZACHARY OLIVIER

JUNE 2020

PROJECT BACKGROUND: WHAT IS THE PAIN POINT?

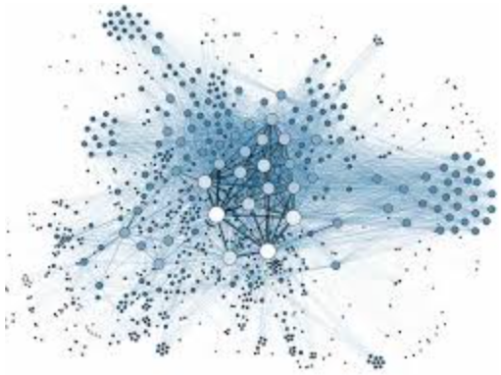
Tricentis



- **Tricentis** is a sophisticated technology company aiming to sell automated software testing and DevOps services to companies looking to speed up their development process
- **Good News:** through trade shows, online marketing, and thought leadership in the DevOps space - Tricentis has a huge amount of potential clients
- **Bad News:** sales teams cannot reach out to every lead - sales resources are wasted if not contacting leads that are "ready to buy"
- **Solution:** automated scoring system to rank and pass through only the highest qualified leads to sales
- **Current State:** scoring system is automated but based on heuristics - this is working great - but business wants to explore a model based solution for this extremely important scoring system - this system drives the entire lifeblood of the business

PROJECT GOALS: WHAT DOES SUCCESS LOOK LIKE?

Tricentis



- 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
- **Requirements:**
  - Model should be rooted in business logic - "standard" features should be included
  - Model should not be constrained to only the most common features, but also take advantage of deeper relationships
  - Model should be interpretable - business would like to understand what drives a specific high or low lead score
  - Model should be accurate, but not aim to exclude all leads in the pursuit of accuracy
  - Model should be extensible, easily integrated with current systems
- **If these goals are met - we will have an excellent solution to a critical business pain point**

PROJECT PLAN: ACTIVITY TIMELINE

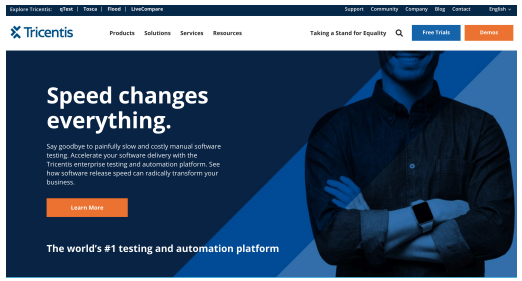
Tricentis

CRISP-DM

- Activity Timeline:
  - **Business and problem understanding** (1-2 weeks)
  - **Data Profiling**: understand how data is collected, explore available data, understand how to tie available data to specific solutions (2-5 weeks)
  - **Data Cleaning:** put relevant data into an analytics-ready format, derive key metrics and features to inform modeling process (2-5 weeks)
  - **Model Experiments:** use data to test different model solutions, validate the model and select the best - exploratory data analysis (1-2 weeks)
  - **Package Model:** develop a quick REST API to serve model scores to end users - will be the basis for our model to interact with current systems (2-4 weeks)

DATA PROFILING: HOW DOES THE DATA SYSTEM WORK?

Marketing → Business Website → Lead Management ← Lead Enrichment



Marketo

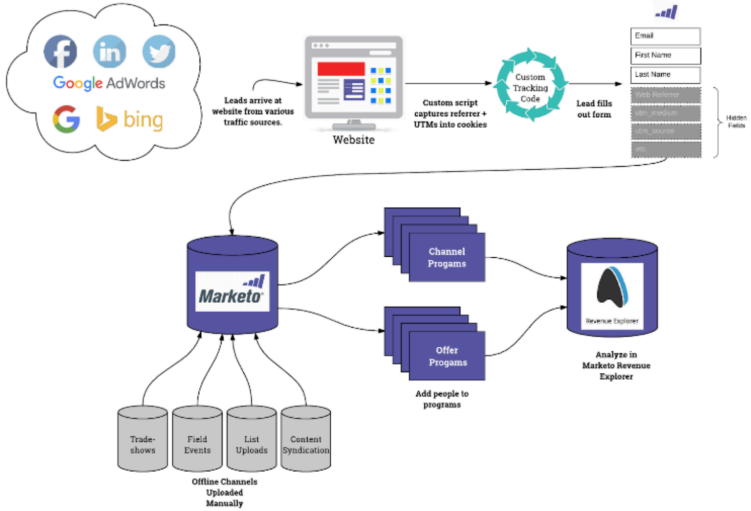
Clearbit

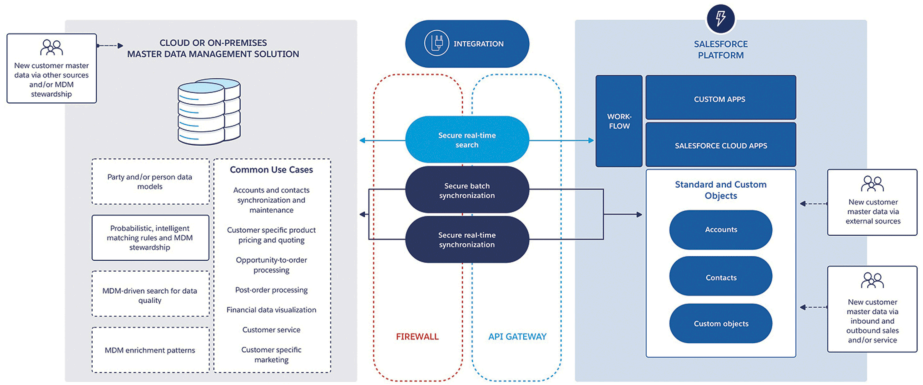
DiscoverOrg

ReachForce
Fueling Targeted Lead Generation

Datanyze

salesforce

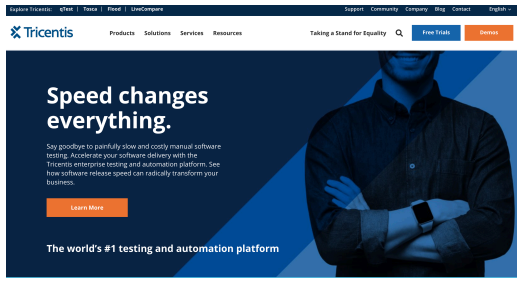




- Generic view of a typical sales funnel for a SaaS company - relies on the integration of **Marketo and SalesForce**
- In this layout - marketing drives leads to the company website / landing page, and these visitors are tracked by Marketo, if a lead provides an email (form fill, content interaction), the lead is passed to SalesForce, where the sales team can view visitor data and follow up with that lead
- **Currently the business rule model sits with Marketo to control which leads are passed to SalesForce and the sales team - this is the same function our model will aim to solve**
- Once a lead fills out a form - various lead enrichment vendors can append certain attributes to the lead record
- Examining the lead funnel shows how data is generated and gives insights into what type of data is available to use with our model

DATA PROFILING: WHAT DOES OUR DATA LOOK LIKE?

Marketing → **Business Website** → **Lead Management** ← **Lead Enrichment**



Marketo

Clearbit

DiscoverOrg

ReachForce
Fueling Targeted Lead Generation

Datanyze

salesforce

- Marketing Channel
- Marketing Channel Type
- Channel Path

- Page Visits
- Clicks
- Form Fills
- Form Fill Data
- Content Downloads
- Sign-ups
- Videos Watched
- Page Visit Path
- Content Type
- Content Path
- Time on Website
- Time Between Sessions

- Marketing Campaigns
- Lead Activity
- Lead Status
- Sales Status
- Interactions with Sales
- Retargeting
- Lead Segmentation

- User Attributes
- Business Attributes
- Location
- Business Type
- Annual Revenue
- Employees
- Years in Business

Lead Activity / Engagement

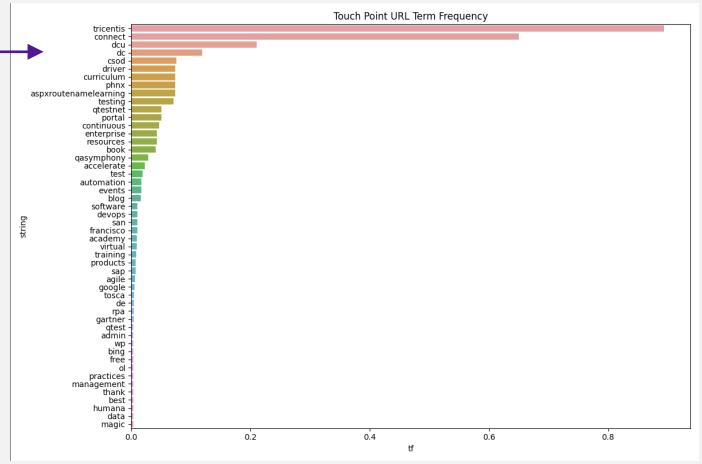
Lead Qualifications

- Each section of the funnel provides unique data to score leads that are most likely to convert
- Data available for modeling includes **marketing channels** (Google, Facebook, Paid Search, CPC), **website engagement touch-points** (clicks, content downloads, form fills), **lead activity** (demos, webinars, sales contacts, retargeting campaigns) and **lead attributes** (business type, location, revenue)
- Current state: heuristic model housed in Marketo scores leads based on **engagement metrics** and lead **qualification metrics** across the sales funnel
- These same metrics are available to use for our solution - but we are free to recode, recategorize, and combine together in a data driven model

DATA CLEANING: TECHNIQUES USED TO GET DATA IN MODELING SHAPE

Data Processing	Why do we need to do this? (Detailed Description)
One Hot Encoding	All categorical data points must be converted to numeric for modeling - each level of a categorical variable but be given a separate column and flagged as 1 or 0 as an indicator if that lead has that specific category
High Cardinality Encoding	Categorical variables with an extreme amount of levels (like the enrichment data provides) is infeasible to one hot encode - in this case we will impute each category with the count of total leads with that category - and provide a dictionary lookup for future scoring
Time Based Features	Time is a rich feature - the intensity a lead clicks through content on a web page may be important to predicting if a lead will convert - this includes calculating time between events, total time on website, total time between content types for all leads in the data set
Channel / Url / Content Path Definitions	The order in which a lead completes their journey matters - here we develop order slots and flag is a certain type of content was consumed, channel was visited, or touch point was visited in the order of which it appears - this means calculating a unique path for each lead: {slot 1: Google, slot 2: Home Page, slot 3: Demo} into sets of column binary variables
Channel / Url / Content Path Keyword Segmentation	URLs for visited webpages, and titles of available content are not static attributes - these change as the business evolves - it is important to group these attributes in flexible segments in order to preserve resilience (if a specific url is used in the model and that url changes - model breaks) - here we employ term frequency and TF-IDF to grab a set of common keywords to segment our webpages, channel names, and urls into generalizable attributes
Target Definition	Using the lead records - we filter out certain leads that do not meet web visitor criteria to ensure our model will be targeting a relevant business outcome - this includes making sure data appended after a deal is closed is not kept as a feature, this will "leak" information into our model
Common Join Key	Cleaning each dataset from the sales funnel does not matter if we cannot link them together - here we use a unique key to connect all features (touch-points and attributes) to our targets
Remove Censoring	Touch-points or any other activity after a lead converts will leak information into our model - giving biased results - to guard against this - we filter out any activity that happened after conversion date - this is done for all leads in the dataset

- Summary of some but not all of the challenging data cleaning and feature building are shown here
- **Key Points:**
  - **Models expect numeric variables**, all categorical variables must be reformatted (including all 100+ enrichment attributes)
  - **Model leakage** happens when our model is trained on data that we know after conversion, but did not know at the time of scoring
  - **Censoring** is related to leakage - data must all be filtered to be before conversion date
  - **Connecting all datasets together** and handling edge cases of dropped rows, duplicated rows, non-unique join keys are paramount to rapidly iterating in the model experimentation phase
  - **Website features can be brittle** - need to encode these in a way that captures general intent of the page to not use the specific page in our model
- **With data cleaning and feature building complete - we have a rich feature library from each step of the sales funnel to use to predict lead conversion**



MODEL EXPERIMENTS: A SIMPE MODEL FRAMEWORK

Features → **Learned Mapping** → **Target(s)**

- Page Visits
- Clicks
- Form Fills
- Form Fill Data
- Content Downloads
- Videos Watched
- Page Visit Path
- Content Type
- Content Path
- Time on Website
- Marketing Channel
- Marketing Channel Type
- Channel Path
- Marketing Campaigns
- Lead Activity
- Lead Status
- Lead Segmentation
- User Attributes
- Business Attributes
- Location
- Annual Revenue
- Employees
- Years in Business

- **Algorithms aim to find the mapping of inputs to outputs**
- **Inputs are assigned positive or negative weights depending on how they influence the target outcome**
- **These weights can be used to score future leads based on the specific inputs of that new lead**
- **There are many different ways to find this mapping and we will explore multiple in order to discover the best approximation of this mapping**

- Passed to Sales
- **Accepted by Sales**
- Sales Closed Won
- Sale Closed Lost

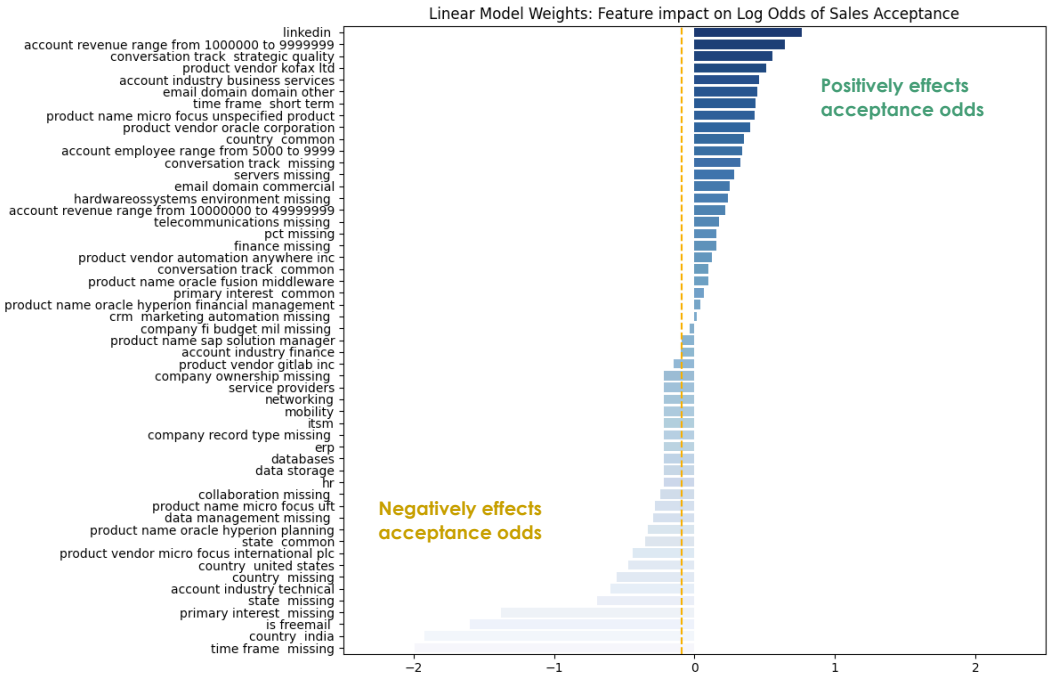
- Model algorithms and processes can be complex and counter productive to building understanding and trust with business stakeholders - here we provide a frame of reference for what we try to accomplish
- At the core - models simply aim to find the **mapping of inputs and outputs** - we know the inputs (form fills, web activity etc.) and the outputs (accepted by sales, closed sales etc.) but we would like to find how the inputs are linked to the outputs
- There are many different techniques to find this mapping, each with key assumptions and unique ways of finding the mappings
- Here is our model in simplest form - all models will have the same objective of finding the input-output mapping, but will find the mapping in unique ways - no matter how complex the method - all can still be reduced to this simple function form
    - **Inputs (Features):** marketing channel, website touch-points, content engagement, product demos, lead attributes, business attributes - all data found in ll categorical variables must be reformatted (including all 100+ enrichment attributes)
    - **Output (Target):** indicator if a lead has been accepted by sales as a viable sales prospect
- **Once the mapping is learned - we can use this to score new leads based on their inputs**

MODEL EXPERIMENTS: REGULARIZED LINEAR MODEL

Assumptions: Strict | **Accuracy: Medium** | **Interpretability: High** | **Overfitting Risk: Low**

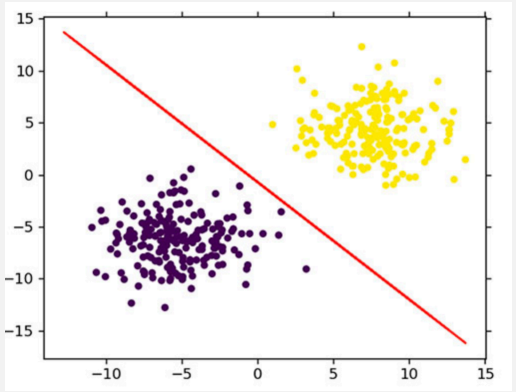
Confusion Matrix	
61,491 (True Negatives)	13,915 (False Positives)
164 (False Negatives)	564 (True Positives)

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 Performance is tricky working with unbalanced data - with sales acceptance only 1% of all leads - **model can predict no acceptance and get "free" 99% accuracy!** But this model will be worthless to the business…
- To model this problem we can track performance against a host of metrics that determine the balance between positive and negative classes
- **Model performance is fairly strong - accurately identifying 77% of sales accepted leads - but model also scores large amount of False Positives (~14K)**
- **Model is confident with leads that will definitely not convert, but not optimal on leads that are "close" to profile of conversion**

- A regularized linear model aims to fit a line that best separates the accepted by sales and not accepted by sales labels.
- Lasso Regularization will throw out features that have a small effect on the odds of sales acceptance - **of our original 1000+ feature dataset, the regularized model only kept around ~60 variables**
- **Linear models are extremely interpretable** - they provide greater detail into the learned mapping. Each feature is given a coefficient that provides the magnitude and direction of the features effect on our labels.
- **Features that *positively effect* the odds of sales acceptance:** LinkedIn Profile, High Account Revenue, Strategic Quality Conversations, Product Vendor Kofax, Other Email Domain, Form Fill Time Frame - Short Term, Product Vendor - Oracle, Common Country Location
- **Features that *negatively effect* the odds of sales acceptance:** Form Fill Time Frame Missing, Country India, Freemail Flag, Primary Interest - Missing, State - Missing, Account Industry - Technical

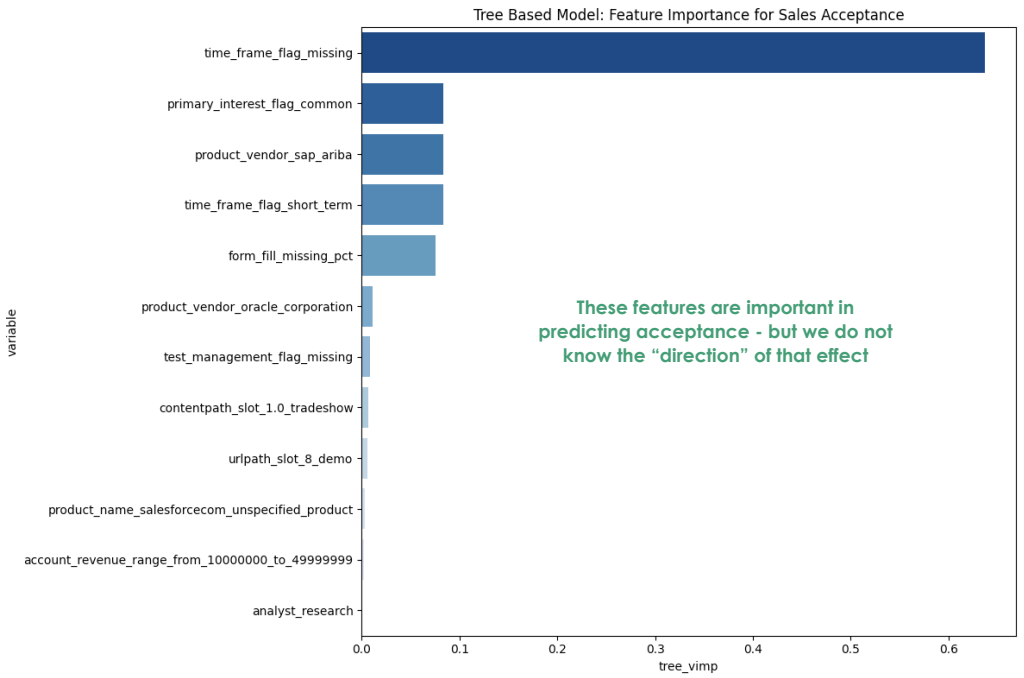


MODEL EXPERIMENTS: RANDOM FOREST

Assumptions: Flexible | **Accuracy: High** | **Interpretability: Medium** | **Overfitting Risk: Medium**

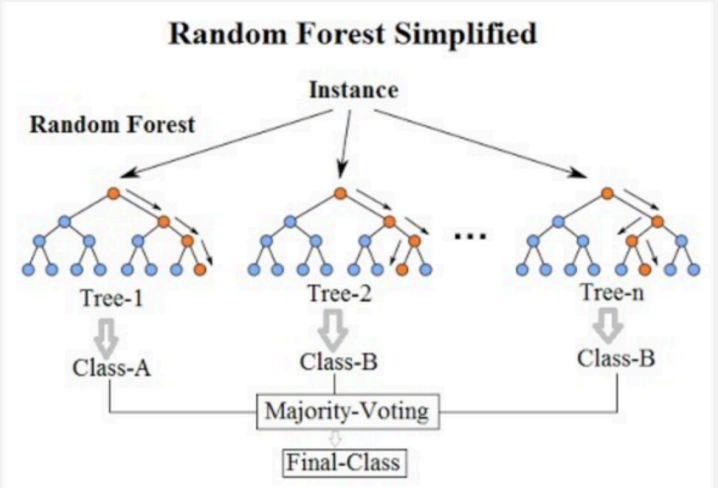
Confusion Matrix	
65,559 (True Negatives)	9,830 (False Positives)
260 (False Negatives)	485 (True Positives)

Performance Evaluation: 5-Fold CV Average	
ROC-AUC	0.88
RECALL	0.65
PRECISION	0.05
ACCUACY	0.87
F1 SCORE	0.92
PREDICTED ACCEPTED	14%



- Model Performance looks strong but still as low precision - **we need a lot of false positives to predict identify a high percentage of accepted leads…**
- Model accurately identifies **65% of sales accepted leads but also scores ~10K false positives** - the **accuracy of predicted accepted but truly accepted is only 5%**
- Model is confident with leads that will definitely not convert, but not optimal on leads that are "close" to profile of conversion
- **Using this model would flag 14% of all leads as accepted by sales**

- Random Forest learns the mapping by fitting multiple decision trees (analogous to nested if-statements) on different subsets of observations and features. All the decision trees combine together in an ensemble that votes on the predicted label based on the subset of data they are "experts" in. Ensemble methods have been shown to fit many flexible function mappings.
- **Tree based models allows for some interpretability** - we can count the times a feature is used across all the ensemble to get a feature importance metric. **This metric tells us which variables have a large effect on prediction but does not tell us if the features impact prediction positively or negatively.**
- Our tree model picks out **Form Fill Time Frame, Primary Interest, SAP ARIBA Product, Trade-shows (slot 1) and Demos (slot 8) as the most important features** driving prediction performance from the Feature Importance metric.

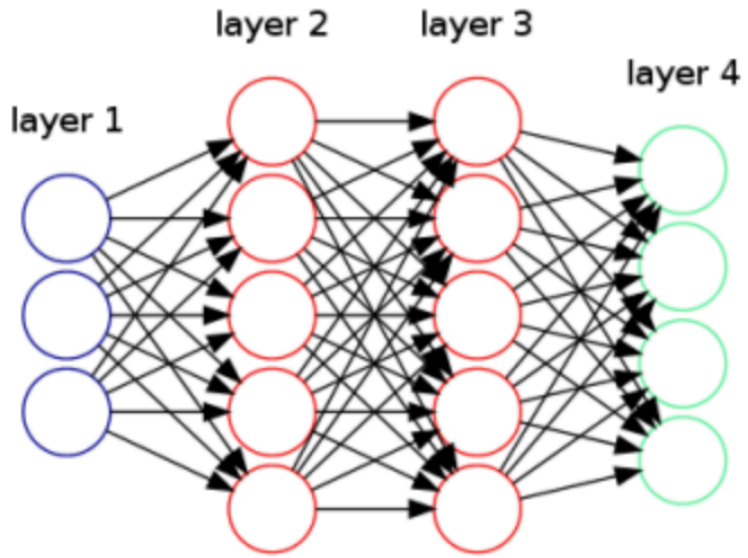


MODEL EXPERIMENTS: NEURAL NETWORK

Assumptions: Flexible | **Accuracy: High** | **Interpretability: Low** | **Overfitting Risk: High**

Confusion Matrix	
58,789 (True Negatives)	16,604 (False Positives)
103 (False Negatives)	638 (True Positives)

Performance Evaluation: 5-Fold CV Average	
ROC-AUC	0.91
RECALL	0.86
PRECISION	0.04
ACCUACY	0.78
F1 SCORE	0.87
PREDICTED ACCEPTED	30%



- **Model performance on identifying leads that are accepted by sales is strong - accurately identifying 86% of sales accepted leads - but model also scores large amount of False Positives (~17K) - this means the precision (accuracy of model picking accepted vs. actual accepted) is very low**

- **Estimate that using this model will identify ~30% of leads that show signs of sales acceptance - we know true acceptance is only around 1% - showing that the higher classification accuracy of our positive class comes with more false positives**

- Neural Networks are extremely popular in classification tasks due to their high flexibility and ability to get high accuracy quickly
- The tradeoff is that **Neural Networks offer very little on "why" they are making their predictions** - interpretability is minimal compared to linear and tree based models - there is no estimate of impact or feature importance metrics available with Neural Networks
- Neural Networks can have multiple layers of connected neurons - each transforming the inputs into more complex representations - this is great for mapping complex functions but difficult to determine which features and transformations are driving the outputs
- **Risk of overfitting is also extremely high** - models with many neurons and layers can find patterns in the data that are too specific to the training data, but those patterns will not hold on new data - **balancing the search for accuracy against overfitting means fitting smaller networks, randomly dropping out connections, or even pruning back layers of the network**

MODEL EXPERIMENTS: COMPARE MODELS

Network Model

Confusion Matrix	
58,789 (True Negatives)	16,604 (False Positives)
103 (False Negatives)	638 (True Positives)

Performance Evaluation: 5-Fold CV Average	
ROC-AUC	0.91
RECALL	0.86
PRECISION	0.04
ACCUACY	0.78
F1 SCORE	0.87
PREDICTED ACCEPTED	30%

Tree Model

Confusion Matrix	
65,559 (True Negatives)	9,830 (False Positives)
260 (False Negatives)	485 (True Positives)

Performance Evaluation: 5-Fold CV Average	
ROC-AUC	0.88
RECALL	0.65
PRECISION	0.05
ACCUACY	0.87
F1 SCORE	0.92
PREDICTED ACCEPTED	14%

Linear Model

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%

- Comparing all models shows some common patterns:
  - **All models are good at figuring out who will not be accepted by sales**
  - **All models have difficulty cleanly separating accepted sales leads (precision low across all)**
  - **Tradeoff: the better we get at classifying accepted leads, the more false positives we let in**
- All models can be tuned further to help balance identifying leads accurately and letting in false positives - this can be done based on business goals
  - **Does business want to let in as few leads as possible, the leads model is most sure about - but might leave potential leads out?**
  - **Does business want to capture as many leads as possible but have our sales time work through more false positives?**
- Network model does the best at classifying accepted by sales leads (86% recall) but lets in the most false positives (~17K)
- Tree model does the worst at classifying accepted by sales leads (65% recall) but lets in the least false positives (~10K)
- Linear model accurately identifies about 77% of accepted by sales leads but lets in 14K false positives
- The fact that all models with varying complexity run into this same tradeoff means the separation between our two classes is likely a data issue, given the same data all models are finding it difficult to identify sales acceptances without an increase in false positives - solution likely means more detailed form fill data collection, more possible touch-points and content interactions to better identify these leads

NEXT STEPS

- **All models can be further refined to find the best balance between identifying accepted leads (recall) and letting in false positives (precision)**

- **Discuss with business these trade-offs and find out the "cost" on each side of our trade-off** - this will help determine the best possible model for this heavily imbalanced data set

    - What is the cost of *False Positives*? Leads sent to sales but not accepted
    - What is the cost of *False Negatives*? Leads not sent to sales but actually accepted

- **Finalize the model and analyze the outputs to determine which features drive predictions** - this can be used by business to optimize data collection, marketing strategy

    - Linear Model: magnitude and sign of the coefficients
    - Tree Model: feature importance
    - Network Model: summary statistics of key variables to show differences in prediction

- **Build a quick API to serve the scores of the best model - this will allow systems and users to actually use the scores of our model**