Customer Churn Modeling

Chetana Vyas
November 2021

Agenda

1 Introduction

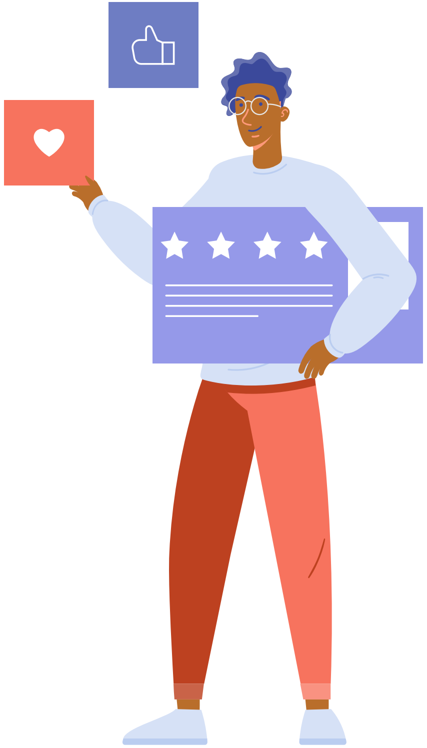
2 Data

3 Workflow

4 Classification Models

5 Result Analysis

6 Future Work



1. Introduction

Why do we care about the Customer Churn?

- 80% of your future profits will come from just 20% of your existing customers
- Acquiring a new customer is five times as expensive as retaining an existing customer
- Increasing customer retention rates by 5% increases profits by 25-95%

Source: Gartner Group, Bain & Company

2. Data

It’s all about the data!

1. **Kaggle Data set** — 10,000 data points
2. **Features** — 14 features: Balance, Credit Score, Age, # of Products, etc.
3. **Target** — Churn / Retained
4. **Goal** — Binary Classification Model to predict Churn

3. Workflow

Machine Learning Workflow

1

Data

- Macro level explorations
- Performance metric

2

Train-Test Split

- No Data Leakage
- Train: 80%
- Test: 20%

3

Feature Engineering

- Encode Categorical Features
- Standardize Features

4

ML Classification Models

- Dummy Classifier
- Logistic Regression
- KNN
- Decision Trees
- Random Forest
- Gradient Boosting

5

Finalize & Interpret

- Takeaways and recommendations
- Next Steps

Data

Sanity Check

Are feature values plausible?

Metric - Recall

Goal

Identify customers most likely to churn

EDA: Breaking down the Churn

20%	70%	64%	55%	56%
Churn	Use only 1 Bank Product	Inactive Users	Females	Age: 50-60 years

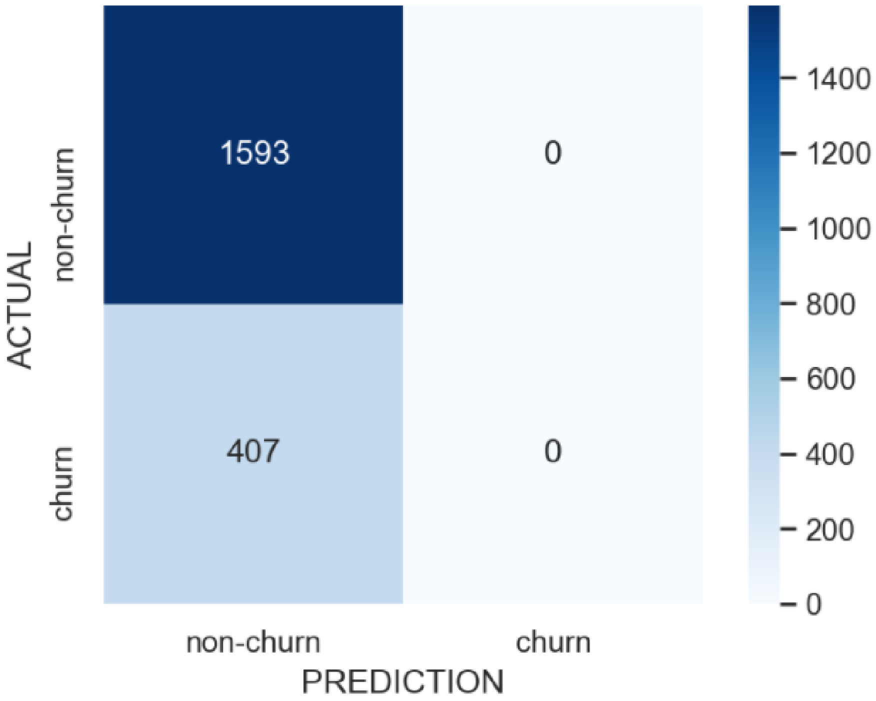
Feature Engineering

1. Categorical predictors encoded using One-Hot Encoding
2. Standardized features
3. Drop irrelevant / confidential features - CustomerID, Surname

4. Classification Models

Baseline Model - Dummy Classifier

- ❖ Classify everything as Majority Class (Not-Churn)
- ❖ Customers Lost: **20.4%**



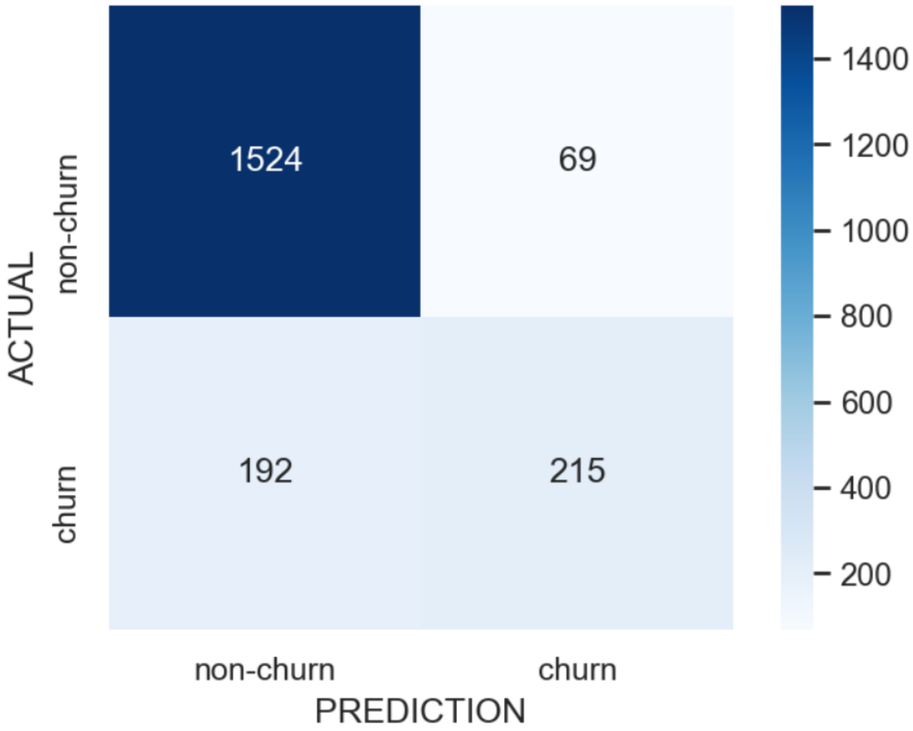
Building Classification Models

- Cross Validation
- Hyperparameters tuning
- Scoring the model

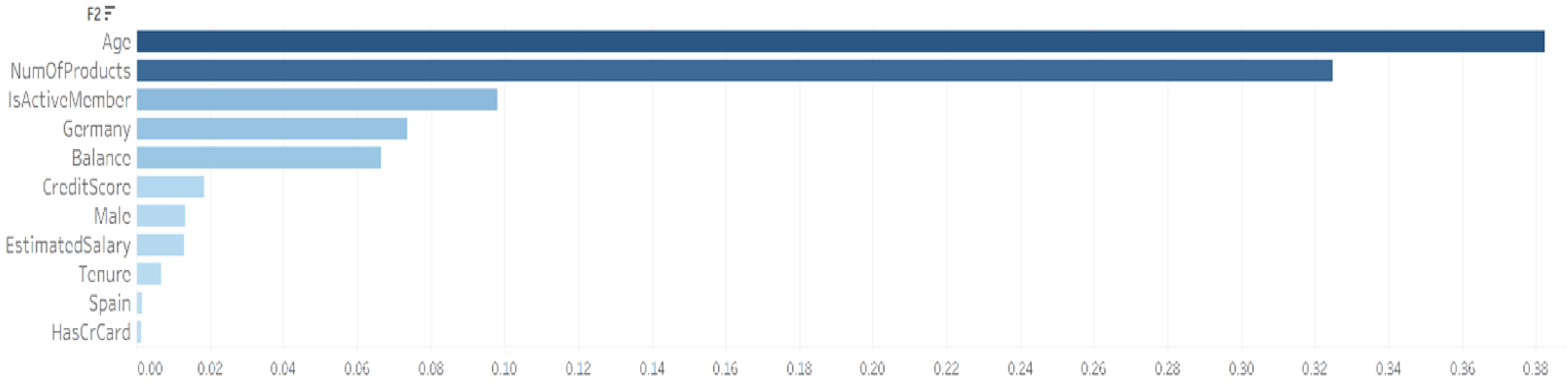
Model	Recall
Logistic Regression	0.82
KNN	0.83
Decision Tree	0.82
Random Forest	0.84
Gradient Boosting	0.87

XGBoost

❖ Recall: **0.87**

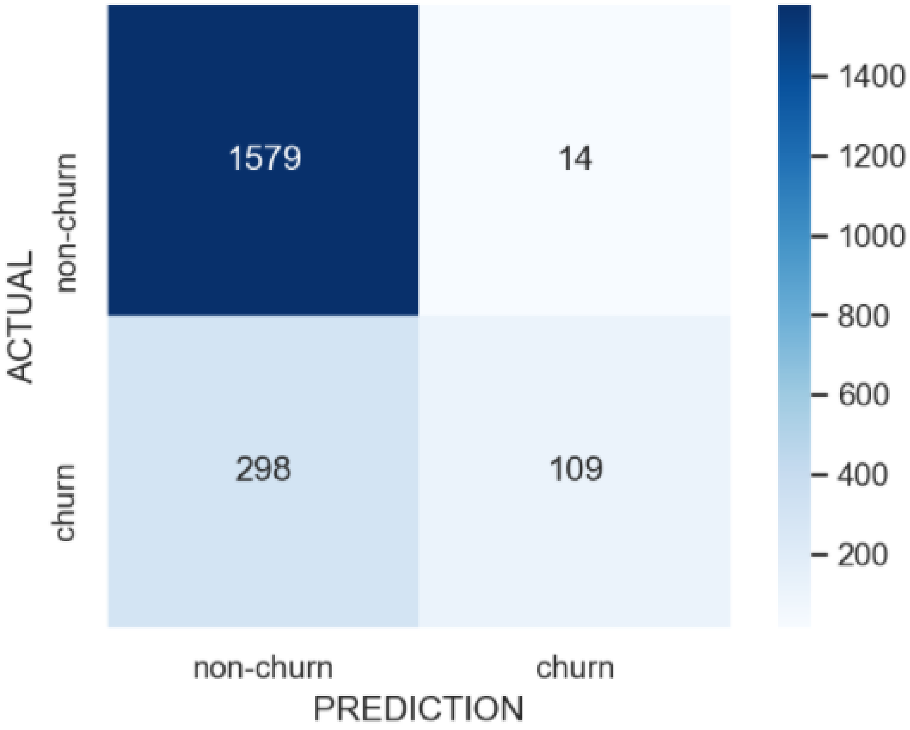


XGBoost - Feature Importance

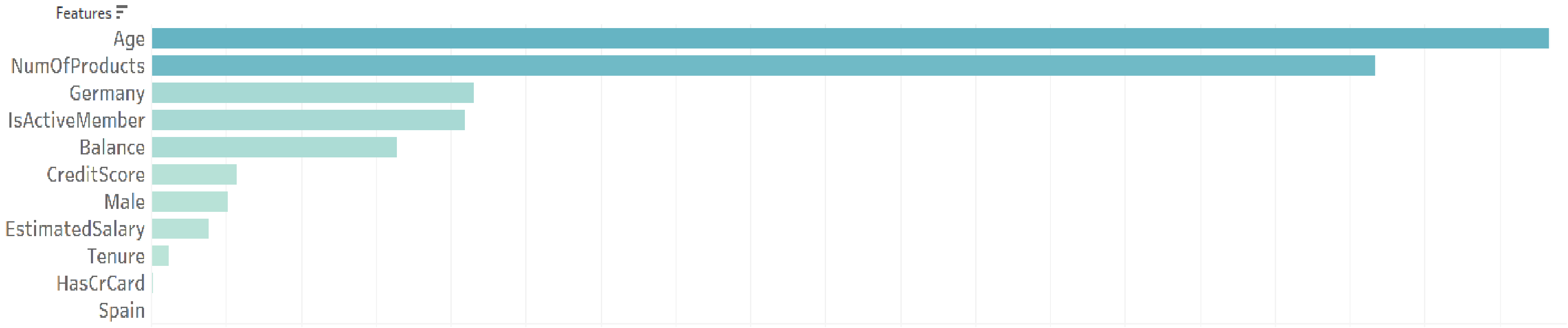


Final Model - Random Forest

- Recall: **0.84**

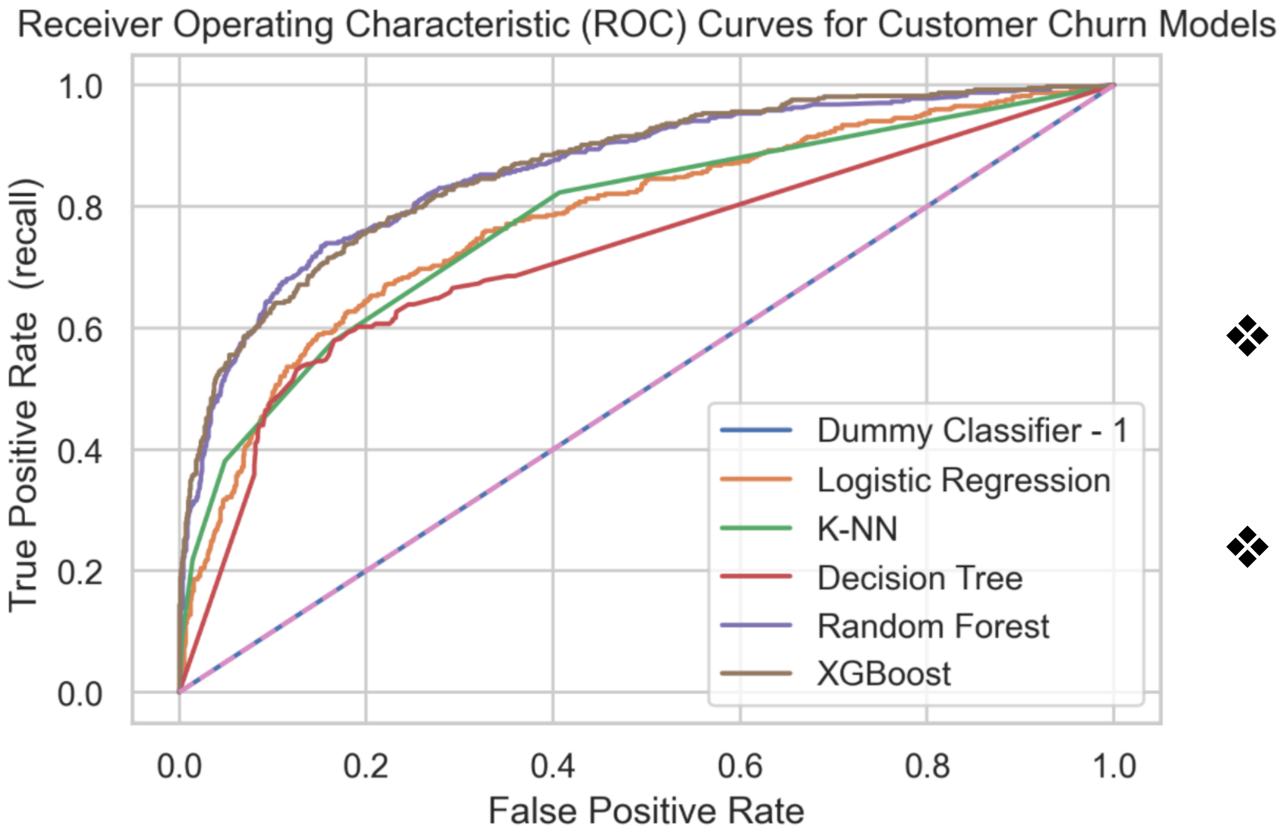


Random Forest - Feature Importance



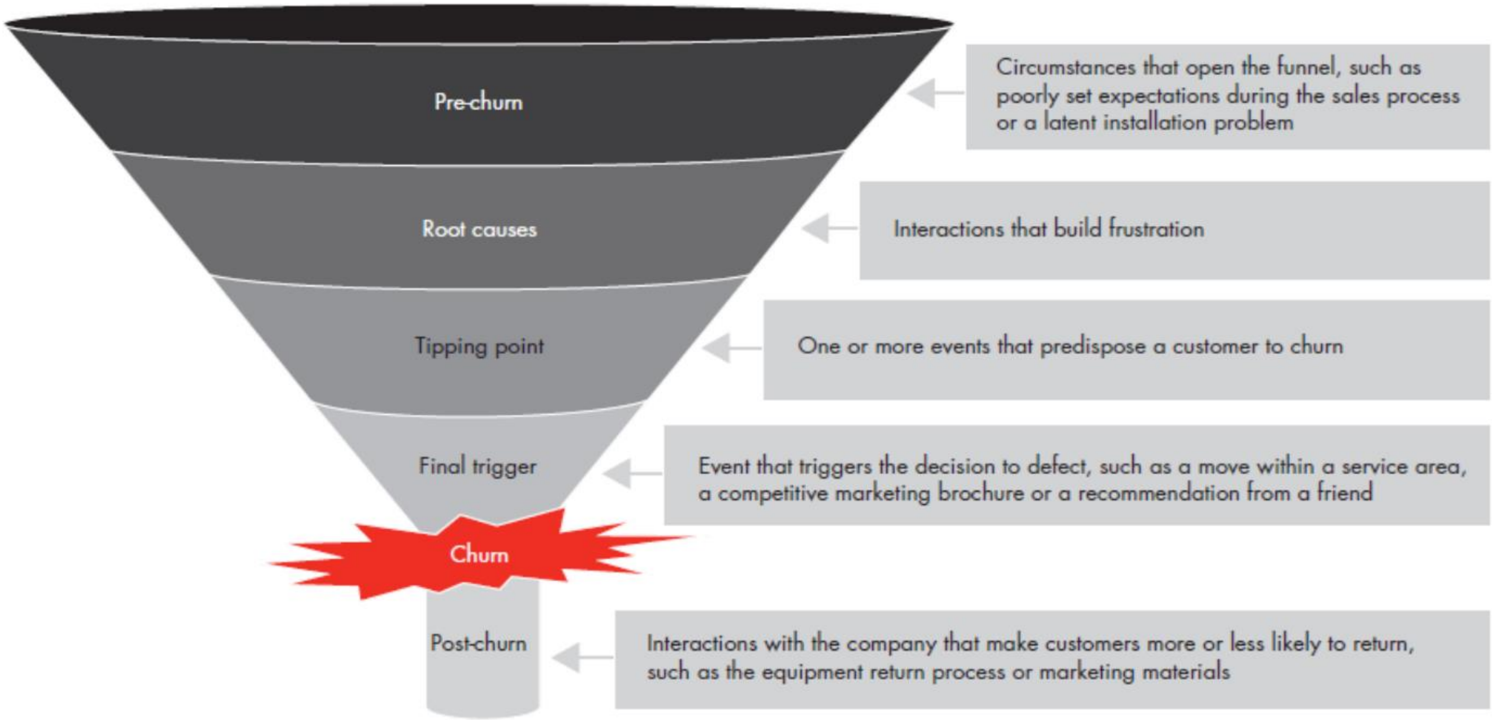
5. Result Analysis

Comparison - Model Performance



- ❖ Random Forest: **0.859**
- ❖ XGBoost: **0.863**

Understanding the Churn



Source: Bain & Company

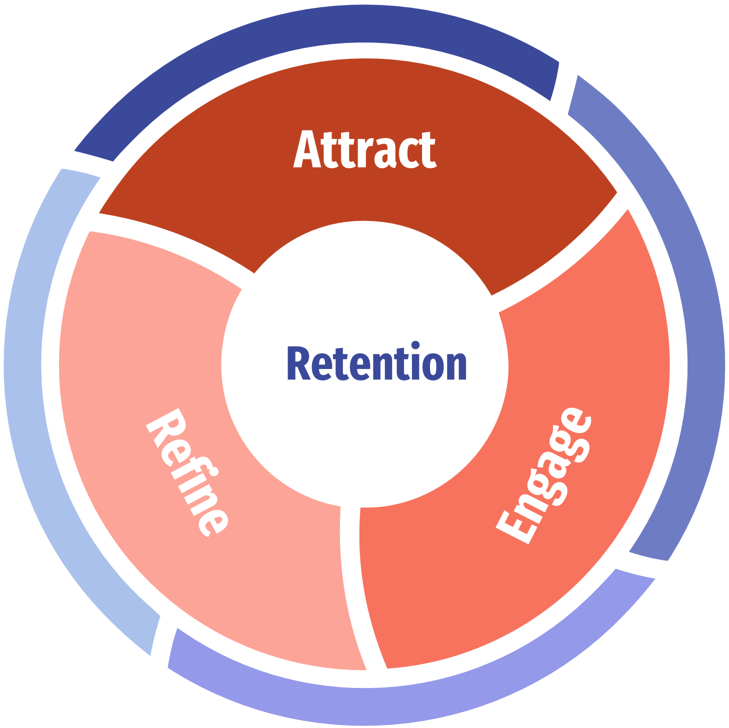
Recommendations

Churn Detection

Weekly run the Churn Model to identify customers at risk of churning

Churn Prevention

Loyalty and Customer Retention Programs



Continuous Optimization

Refine offered services

Marketing Techniques

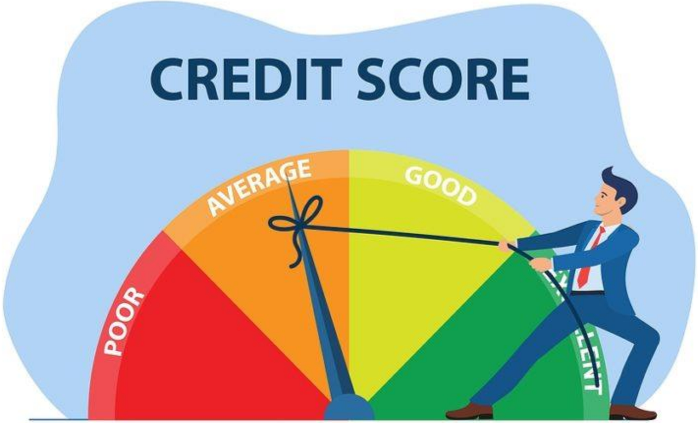
Digital / offline marketing

6. Future Work

Next Steps

Build combination of Models:

1. **Target Profitable / Elite Customers**
   Focus on maximizing **Recall**
   Eg. Bank Balance > 500,000
   Credit Score > 750
1. **Handling non-elite Customers**
   Focus on minimizing **Precision**
   Eg. Bank Balance < 500
   Credit Score < 580



Next Steps

Gather additional information

- Churn Date
- Frequency and time of user logins
- Customer background (education, employment, etc.)
- Bank services
- Mobile banking, app reviews
- New govt. policies (demonetization)

Re-train models

- Learn more about hyperparameter tuning
- Class Imbalance

Thank you!

References

Prescription for Cutting Costs | Bain & Company

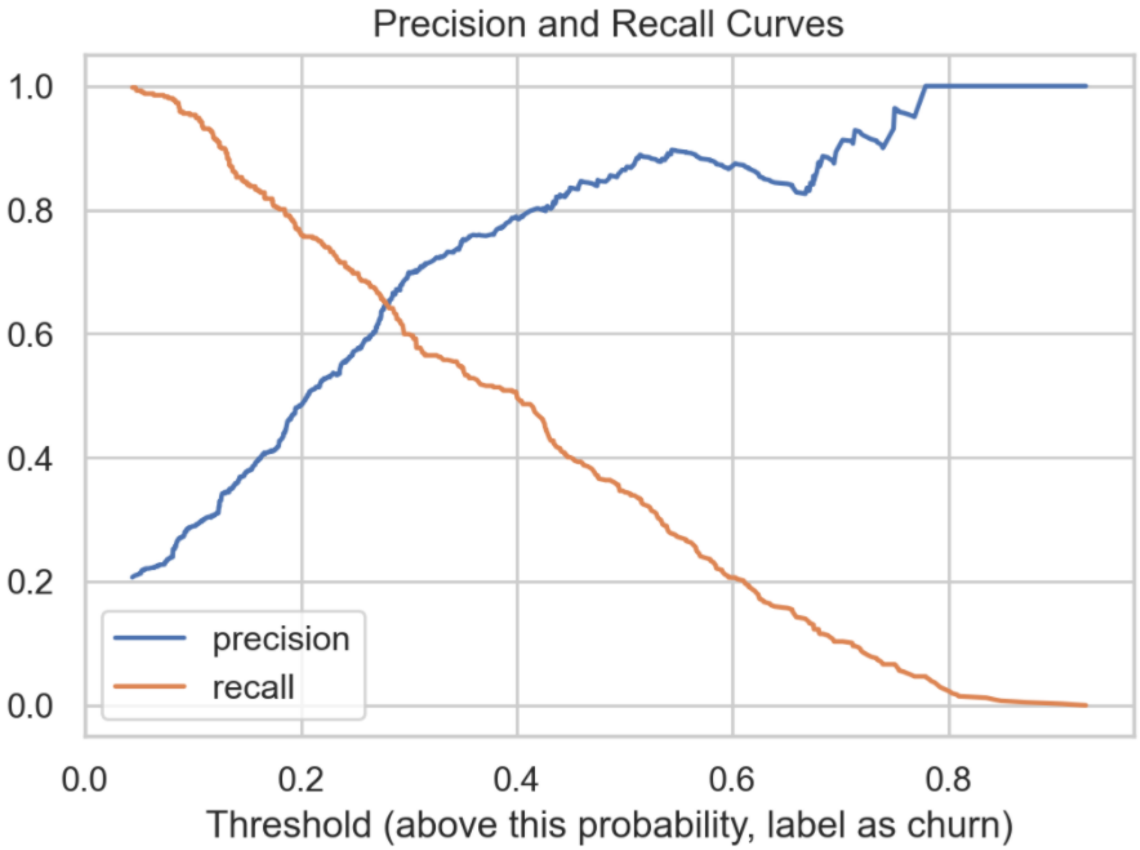
Breaking the Back of Customer Churn | Bain & Company

80/20 principle in Customer Churn | Gartner Group

Bank Customer Churn | Kaggle data set

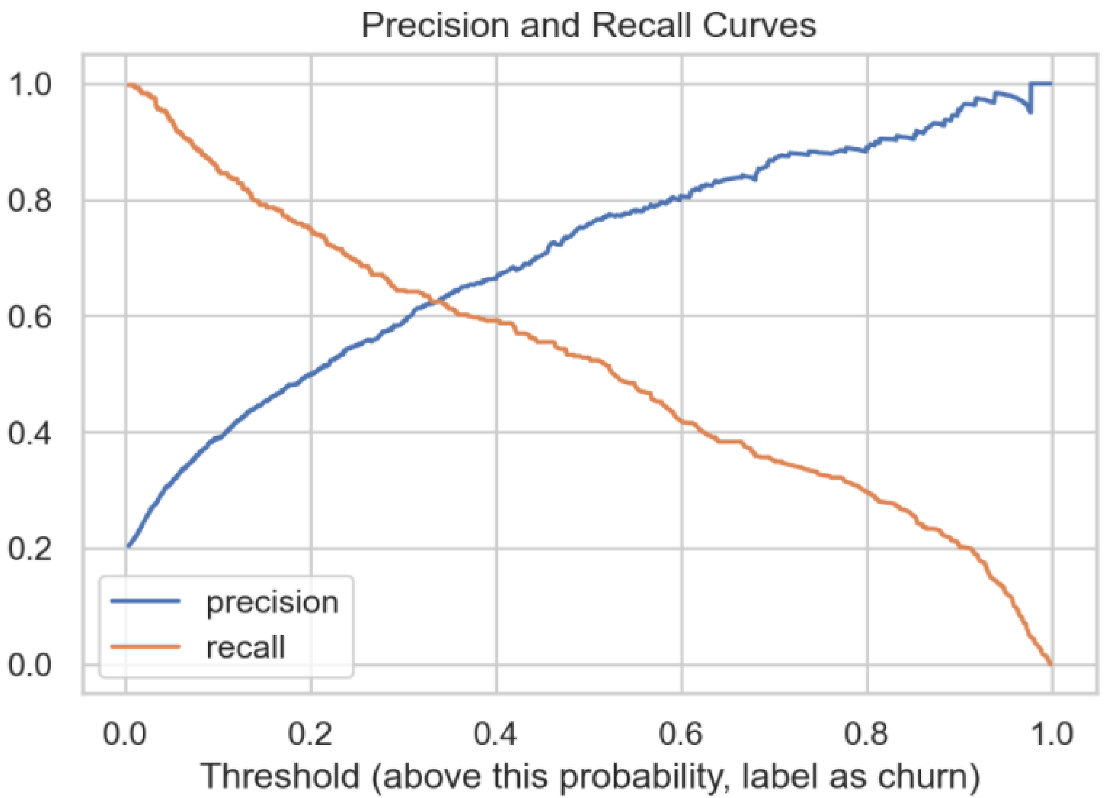
Appendix

Precision Recall curve - Random Forest



	Coefficients	Features
1	0.382248	Age
4	0.324913	NumOfProducts
6	0.098136	IsActiveMember
8	0.073533	Germany
3	0.066545	Balance
0	0.018418	CreditScore
10	0.013252	Male
7	0.013119	EstimatedSalary
2	0.006787	Tenure
9	0.001706	Spain
5	0.001341	HasCrCard

Precision Recall curve - XGBoost



	Coefficients	Features
4	0.301469	NumOfProducts
6	0.169982	IsActiveMember
1	0.163032	Age
8	0.089339	Germany
3	0.056229	Balance
9	0.047886	Spain
10	0.044737	Male
2	0.033435	Tenure
0	0.032193	CreditScore
7	0.031748	EstimatedSalary
5	0.029949	HasCrCard

Sample Data set

RowNumber	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
50	776	Germany	Female	37	2	103769.22	2	1	0	194099.12	0
2328	644	France	Male	30	5	44928.88	1	1	1	10771.46	0
9425	689	France	Female	40	1	0.00	2	1	1	119446.64	0
8438	781	France	Male	29	9	0.00	2	0	0	172097.40	0
5102	622	Spain	Female	58	2	0.00	2	1	1	33277.31	0
8847	571	France	Female	53	2	0.00	2	1	0	28045.77	0
2707	696	France	Male	22	9	149777.00	1	1	1	198032.93	0
8087	593	France	Male	50	6	171740.69	1	0	0	20893.61	0
8900	584	France	Female	41	3	0.00	2	1	1	160095.48	0
8428	753	France	Female	40	0	3768.69	2	1	0	177065.24	1
7789	551	Spain	Male	76	2	128410.71	2	1	1	181718.73	0
7091	601	France	Male	47	1	64430.06	2	0	1	96517.97	0
8296	722	France	Male	40	6	0.00	2	1	1	111893.09	0
7801	698	Germany	Female	52	1	107906.75	1	1	0	168886.39	1
7473	448	France	Female	36	6	83947.12	2	1	0	81999.53	0
2330	850	France	Male	35	3	162442.35	1	1	0	183566.78	0

XGBoost Hyperparameter Tuning

In [57]:

```
random_cv.fit(X_trainsc,y_train)
print("Best params: ", random_cv.best_params_)
print("Best estimator: ", random_cv.best_estimator_)
print("Best score: ", random_cv.best_score_)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best params:  {'min_child_weight': 7, 'max_depth': 4, 'learning_rate': 0.3, 'gamma': 0.1}
Best estimator:  XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='auc', gamma=0.1, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=4, min_child_weight=7, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', use_label_encoder=False,
              validate_parameters=1, verbosity=None)
Best score:  0.48098159509202454
```