

BIA Prediction Challenge

Predicting likely adopters post a music company's upcoming marketing campaign



Agenda

1. The Challenge
2. Approach
3. What we tried
4. What we couldn't
5. What worked well and what didn't

The Challenge

1. Highly imbalanced dataset
(majority-to-minority ratio of 98.3 : 1.7)
2. Features with high correlation
3. Lack of domain expertise

Our Approach

EDA and manipulations

1. Undersampling
2. Oversampling
3. SMOTE / ROSE
4. Feature Selection (PCA, IG)

Algorithms

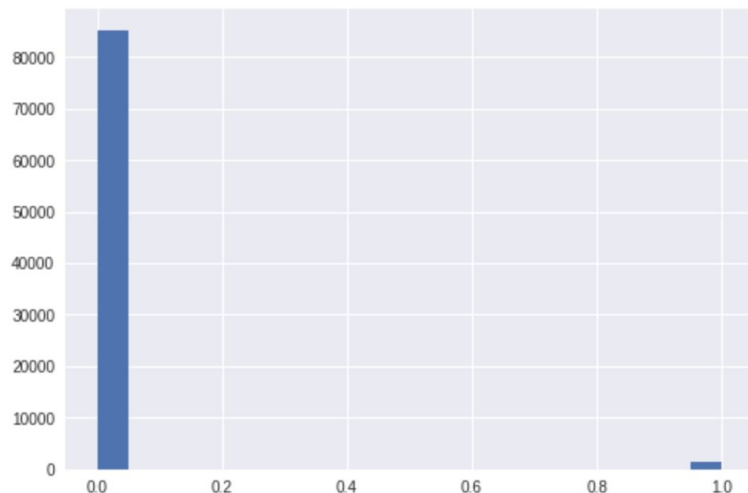
1. Binary Classification
2. Anomaly Detection
3. Neural Networks

Stack

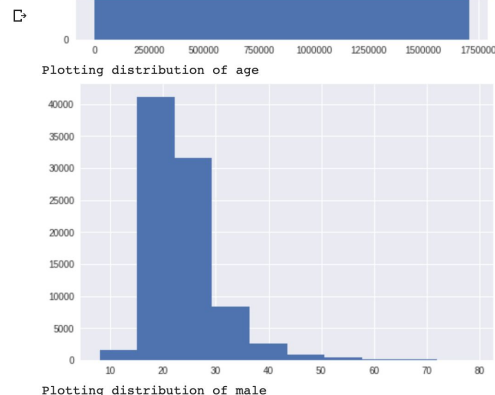
1. Python (scikit-learn, Keras, TF)
2. R
3. Azure ML Studio

What we tried

We first tried understanding the data through some preliminary EDA



```
[ ] 1 # checking each feature's distribution
    2
    3 for col in data.columns:
    4     feature_hist = data[col].hist(bins=10)
    5     print ("Plotting distribution of {}".format(col))
    6     plt.show()
```



What we tried

We thought oversampling the minority instances using algorithms such as **SMOTE** (Synthetic Minority Oversampling Technique) and packages such as **ROSE** would help as it should be beneficial to have more minority instances for learning.


We realised undersampling the majority class should provide enough instances (1540 each). All algorithms were trained on stratified samples with a 50:50 ratio.

Learning I:

Synthesizing new instances (~40k) from a small subset (1540) of instances harms the learning process as not enough variation can be obtained

What we tried

Algorithms

1. k-NN
2. Decision Tree (CART)
3. Logistic Regression
4. LogReg with Regularisation
5. Support Vector Machine
6. Random Forest
7. Extra Trees Classifier
8. Gradient Boosting
-  9. XGBoost
10. LightGBM

11. LDA
12. One Class SVM
13. Shallow Neural Network
14. Stacking of basic algorithms

Learning II:

Gradient Boosting family outperformed any other class of algorithms for small dataset with low feature variations

What we couldn't try

1. Other Ensemble learning methods such as bagging
2. More combinations in stacking
3. Deep Neural Networks
4. Different sampling ratios for majority and minority classes

Learning III:

Domain expertise to understand what features can contribute more towards service adoption could reduce experimentation time and effort

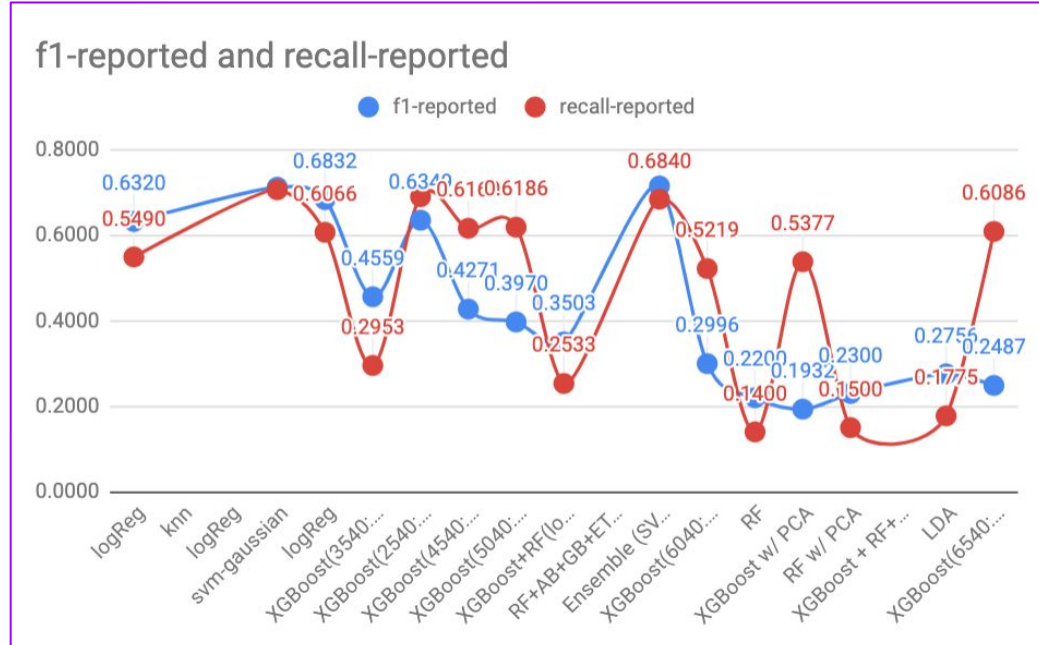
What worked well and what didn't

1. SMOTE / ROSE
2. Feature Selection (PCA)
3. k-NN, Linear SVM, CART, RF, LDA, One Class SVM, NN
 - Logistic Regression
 - SVM with RBF kernel
 - Boosting algorithms (AdaBoost, Gradient Boost, XGBoost)
 - Weighted ensemble (majority voting)

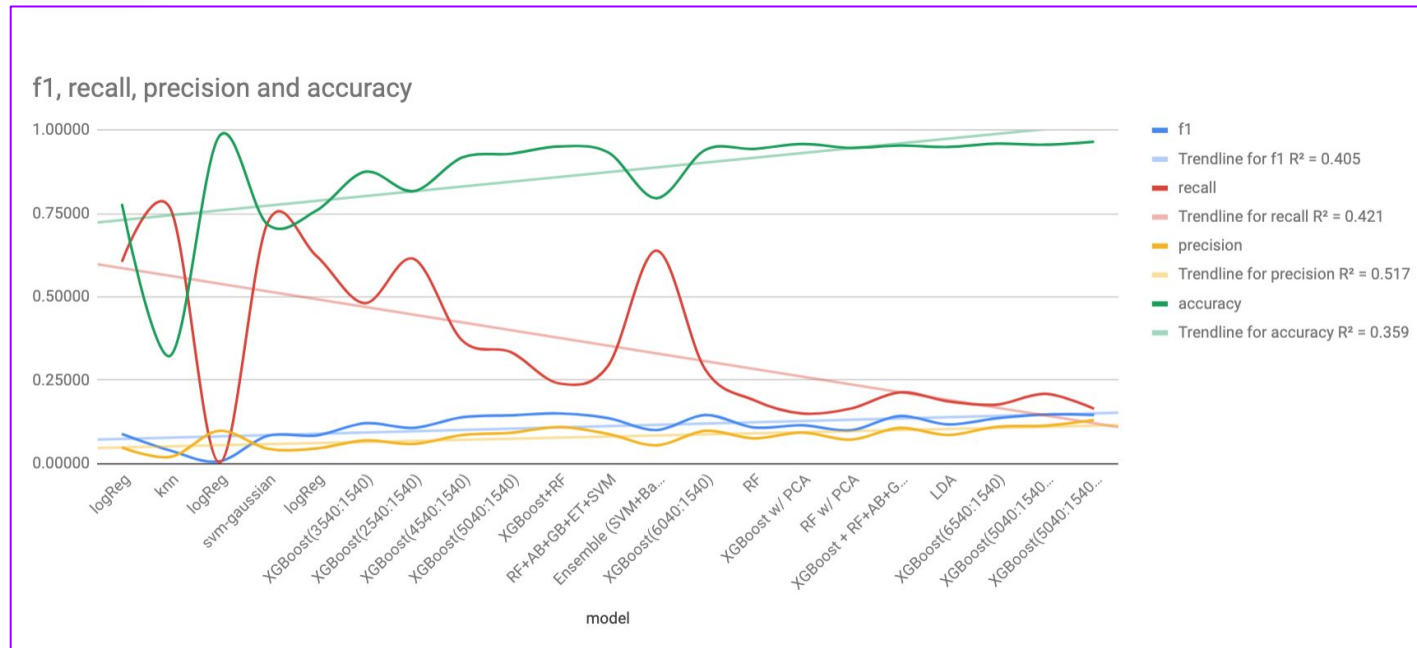
Performance

data	model	f1	recall	precision	accuracy
undersampled	logReg	0.08870	0.60519	0.04786	0.77910
ROSE	knn	0.03862	0.76429	0.01981	0.32390
ROSE	logReg	0.00629	0.00325	0.09804	0.98180
undersampled	svm-gaussian	0.08324	0.72338	0.04416	0.71690
SMOTE	logReg	0.08417	0.62338	0.04513	0.75900
undersampled(3540:1540)	XGBoost(3540:1540)	0.12089	0.48117	0.06913	0.87570
undersampled(2540:1540)	XGBoost(2540:1540)	0.10688	0.61429	0.05853	0.81760
undersampled(4540:1540)	XGBoost(4540:1540)	0.13876	0.36948	0.08542	0.91850
undersampled(5040:1540)	XGBoost(5040:1540)	0.14456	0.33442	0.09221	0.92970
undersampled(6000:1540)	XGBoost+RF	0.15028	0.24026	0.10934	0.95170
undersampled(5040:1540)	RF+AB+GB+ET+SVM	0.13640	0.29286	0.08890	0.93410
undersampled(1540:1540)	Ensemble (SVM+Bagging+RF)	0.10028	0.63896	0.05441	0.79630
undersampled(6040:1540)	XGBoost(6040:1540)	0.14566	0.28247	0.09813	0.94110
undersampled(6000:1540)	RF	0.10812	0.19026	0.07552	0.94420
undersampled(6000:1540)	XGBoost w/ PCA	0.11470	0.15000	0.09285	0.95890
undersampled(6000:1540)	RF w/ PCA	0.09988	0.16494	0.07163	0.94720
undersampled(5040:1540)	XGBoost + RF+AB+GB+ET+SVM	0.14255	0.21364	0.10696	0.95430
undersampled(4620:1540)	LDA	0.11724	0.18636	0.08552	0.95010
undersampled(6000:1540)	XGBoost(6540:1540)	0.13576	0.17662	0.11026	0.96000
undersampled(5040:1540)	XGBoost(5040:1540) w/ threshold>0.6	0.14707	0.20909	0.11342	0.95690
undersampled(5040:1540)	XGBoost(5040:1540) w/ threshold>0.6 + RF	0.14553	0.16429	0.13061	0.96570

Performance



Performance



Thank You

