Jennifer Mead

IST 718

June 2020

Comparing models, encoding, and evaluation metrics on an imbalanced binary classification dataset

Problem statement

I have three questions I want to answer.

Which machine learning model works best for binary classification?

There won't be a conclusive answer to this question, but I am curious about how different models perform on the same dataset.

My data is the IBM Telco customer churn dataset ([https://www.kaggle.com/blastchar/telco-customer-churn/data#]). It classifies about 7000 customers into "renew" (churn = 0) or "leave" (churn = 1). One benefit of this dataset is that it is realistic because it is imbalanced: there are many more people who stay than leave.

Does one-hot encoding impact the results?

For binary classification, column values need to be numbers. It is also generally recommended to bin continuous values (categorize groups of numbers) and to one-hot encode columns that have more than two states (e.g. values of 0, 1, 2, and 3 instead of just 0 and 1).

This is straightforward for a dataset with a small number of columns, but a big hassle for a dataset with a large number of columns. Do these steps actually improve the results? I'm going to run the same models on two sets of fields: one hasn't been binned and encoded, and the other has.

Which metrics are useful in evaluating an imbalanced binary classification dataset?

Accuracy is the default metric of many models, but accuracy is terrible on imbalanced datasets because many models accidentally classify as the larger subset, artificially inflating the accuracy.

(In this case, a model that predicted everyone would stay would have high accuracy because there are many more "stay" cases in the dataset.)

Based on suggestions, I'm going to look at AUC-ROC (area under the curve), F1 score (which tries to balance performance on positive and negative values), and time. Ideally we'd have a highly accurate model that was also fast.

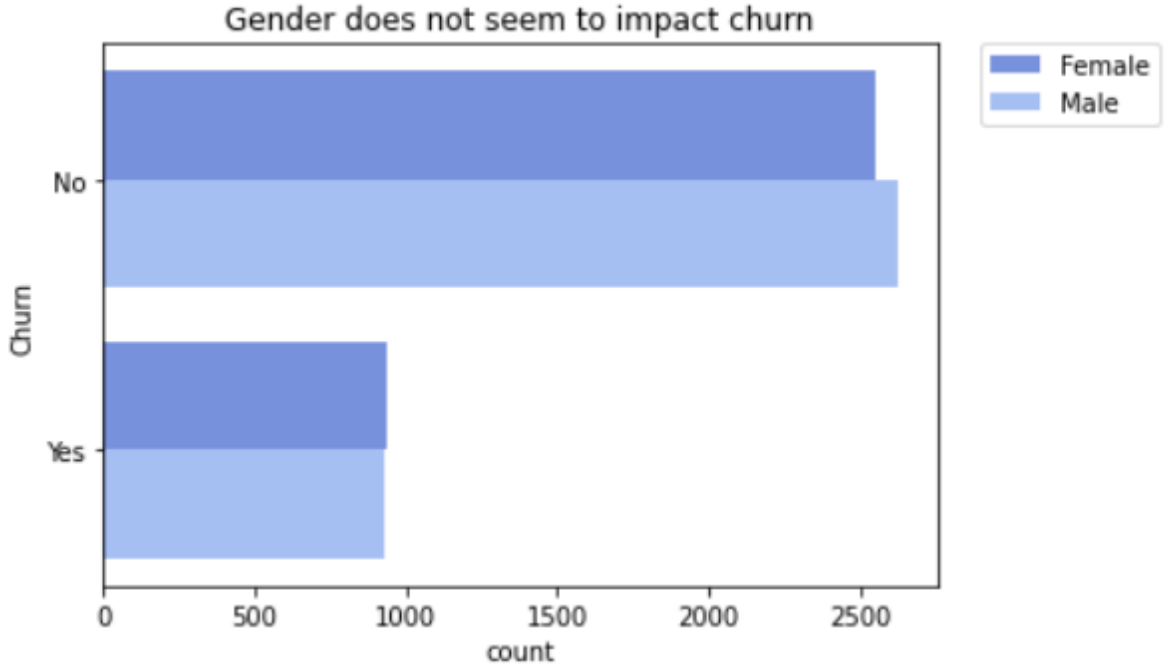
The data

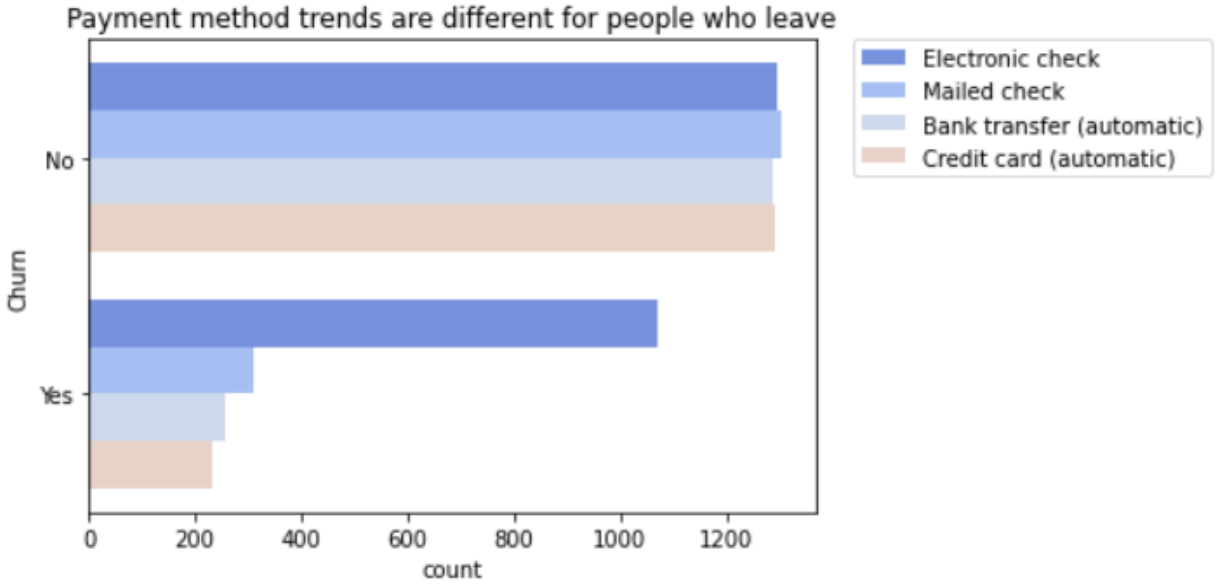
There are 7043 rows and 20 columns.

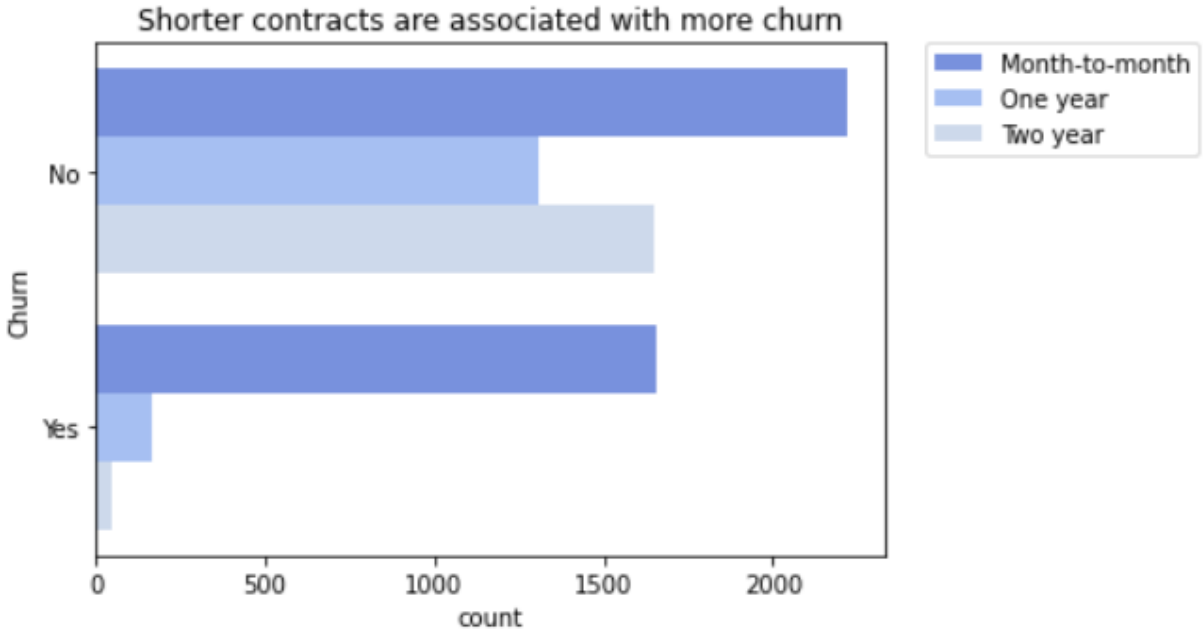
#	Column	Non-Null Count	Dtype
0	customerID	7043 non-null	object
1	gender	7043 non-null	object
2	SeniorCitizen	7043 non-null	int64
3	Partner	7043 non-null	object
4	Dependents	7043 non-null	object
5	tenure	7043 non-null	int64
6	PhoneService	7043 non-null	object
7	MultipleLines	7043 non-null	object
8	InternetService	7043 non-null	object
9	OnlineSecurity	7043 non-null	object
10	OnlineBackup	7043 non-null	object
11	DeviceProtection	7043 non-null	object
12	TechSupport	7043 non-null	object
13	StreamingTV	7043 non-null	object
14	StreamingMovies	7043 non-null	object
15	Contract	7043 non-null	object
16	PaperlessBilling	7043 non-null	object
17	PaymentMethod	7043 non-null	object
18	MonthlyCharges	7043 non-null	float64
19	TotalCharges	7043 non-null	object
20	Churn	7043 non-null	object

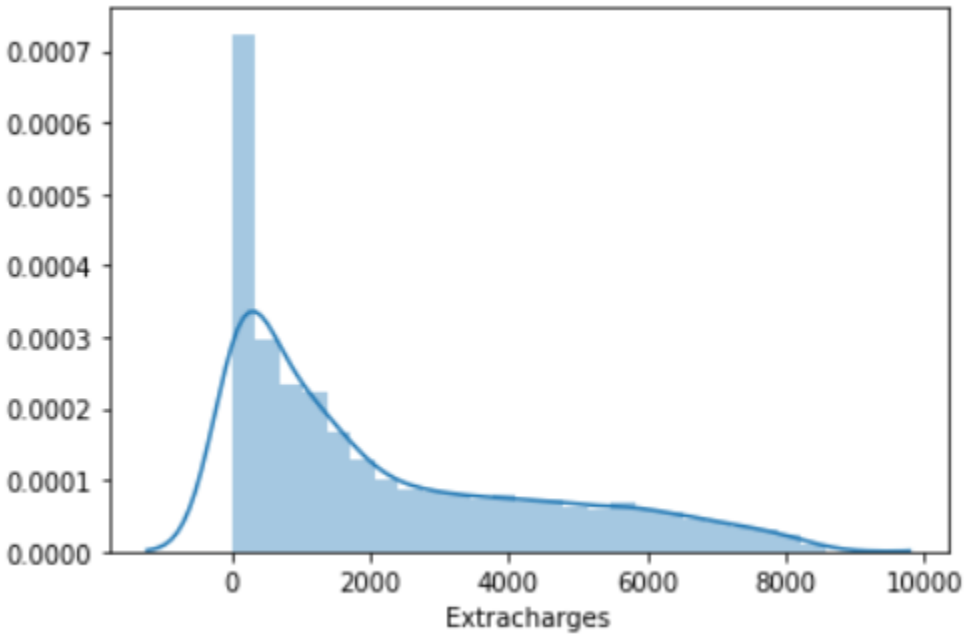
Exploratory data analysis

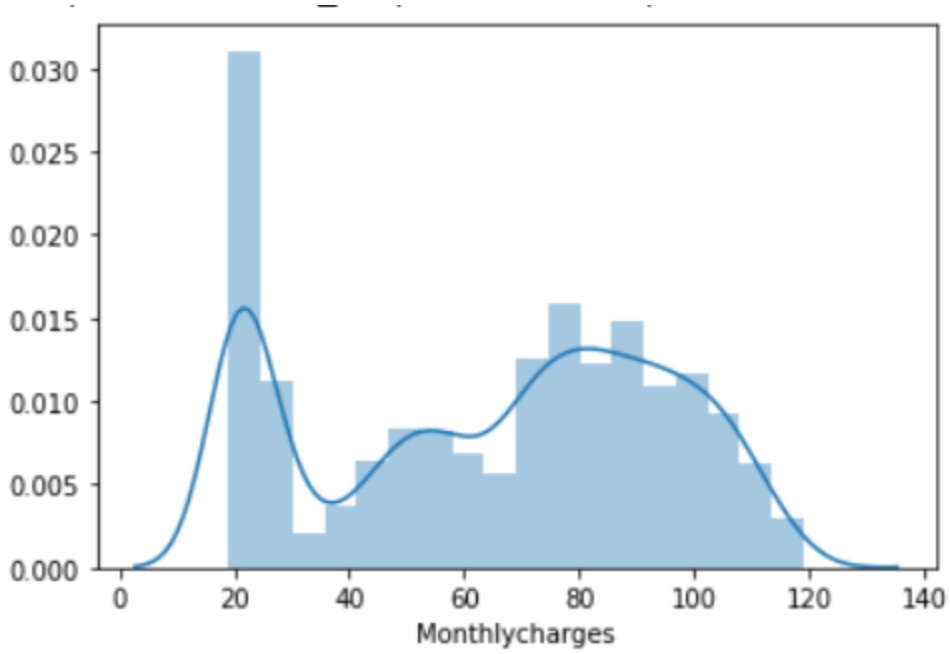
I looked at the data a bit to see if there were some obvious influences to churn, and to see which fields might need to be changed before they are useful.


Gender does not seem to impact churn
Female
Male
No
Yes
Churn
0
500
1000
1500
2000
2500
count

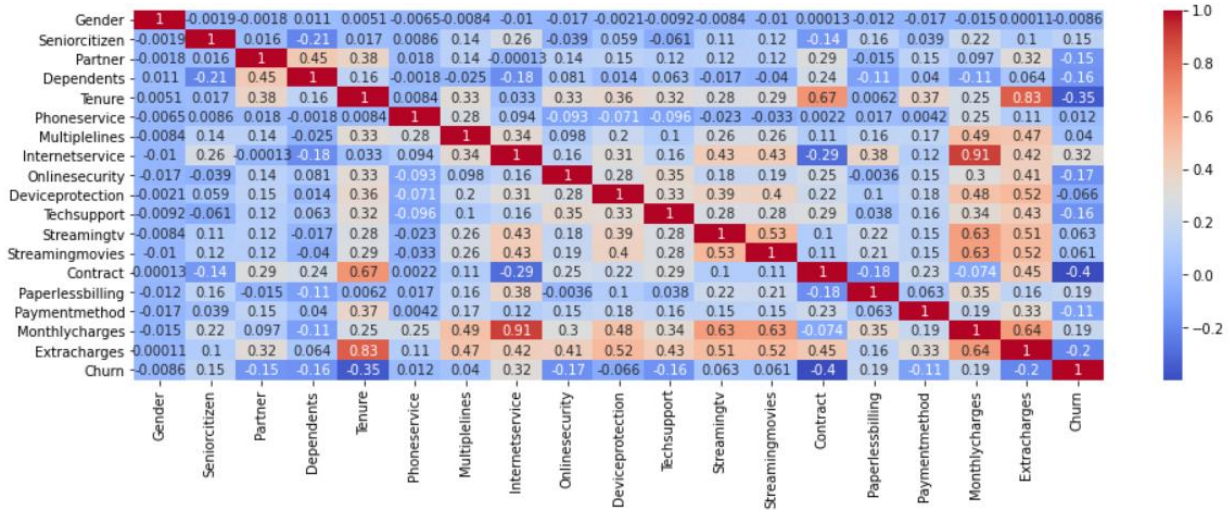

Payment method trends are different for people who leave
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
No
Yes
Churn
0
200
400
600
800
1000
1200
count


Shorter contracts are associated with more churn
Month-to-month
One year
Two year
No
Yes
Churn
0
500
1000
1500
2000
count


0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0
2000
4000
6000
8000
10000
Extracharges



Correlation plots are useful to see if the data is varied enough to be included in the machine learning models. If all of these boxes were the same color, the models would find nothing. In general, the columns that have light, dark, blue, and orange will be more useful to the models.



The models

Model	Tuning Parameter
Logistic regression	max_iter
K nearest neighbors	k
Random forest	n_estimators
MLP	max_iter
Naive Bayes (Gaussian)	var_smoothing
Naive Bayes (Bernoulli)	alpha
Decision trees	max_depth
Support vector machines classifier	max_iter

I've made functions for eight models, plus functions to tune each of these. I haven't comprehensively tried all tuning parameters, due to time constraints. However, I did try 9 variations of one tuning parameter for each model on each dataset. This adds up to 144 variations.

The time metric is calculated on the prediction step, not the tuning as a whole.

Round 1: data that has not been binned and one-hot encoded

Name	Time	F1	AucRoc	TP	FP	FN	TN
Round 1: Naive Bayes Gaussian	0.01	0.617	0.828	1321	393	163	448
Round 1: Naive Bayes Bernoulli	0.013	0.607	0.827	1416	298	215	396
Round 1: Decison tree	0.015	0.56	0.828	1499	215	290	321
Round 1: Decison tree	0.021	0.565	0.775	1465	249	272	339
Round 1: Logistic regression	0.11	0.61	0.843	1530	184	262	349
Round 1: K-nearest neighbors	0.114	0.448	0.776	1607	107	404	207
Round 1: Support vector	0.248	0.584	0.84	1544	170	289	322
Round 1: Multilayer perceptron	0.704	0.589	0.829	1546	168	286	325
Round 1: Random forest	1.083	0.557	0.82	1534	180	306	305

Conclusions for round 1

Of the eight models:

- The fastest is Naive Bayes Gaussian
- The best area under the curve is logistic regression
- The highest F1 score is logistic regression
- The overall winner is logistic regression: third fastest, best AUC, highest F1 score.
- The worst overall is random forest which had the third lowest AUC, the slowest time, and the second lowest F1 score.

Round 2: data has been binned and one-hot encoded

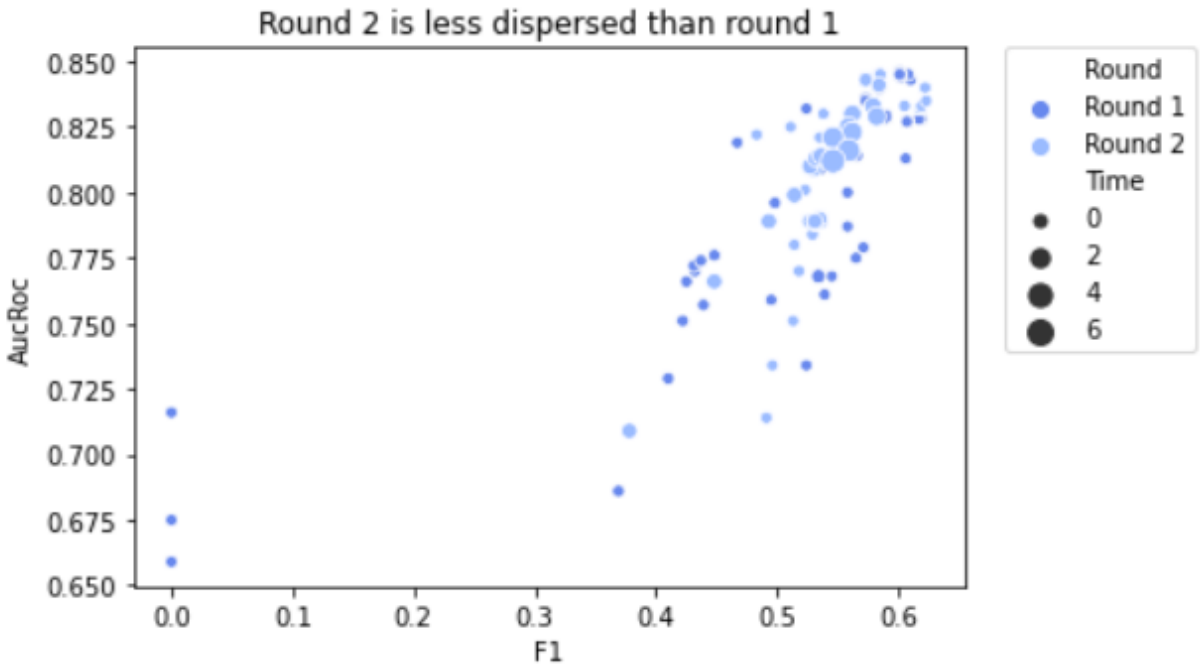
Name	Time	F1	AucRoc	TP	FP	FN	TN
Round 2: Naive Bayes Bernoulli	0.014	0.622	0.84	1337	377	165	446
Round 2: Naive Bayes Gaussian	0.014	0.623	0.835	1249	465	124	487
Round 2: Decison tree	0.019	0.538	0.83	1559	155	329	282
Round 2: Logistic regression	0.044	0.584	0.845	1557	157	294	317
Round 2: Support vector	0.084	0.605	0.833	1516	198	260	351
Round 2: Random forest	0.823	0.536	0.789	1469	245	298	313
Round 2: Random forest	0.94	0.535	0.79	1473	241	300	311
Round 2: K-nearest neighbors	1.476	0.547	0.819	1517	197	307	304
Round 2: Multilayer perceptron	2.02	0.562	0.83	1519	195	296	315

Conclusions for round 2

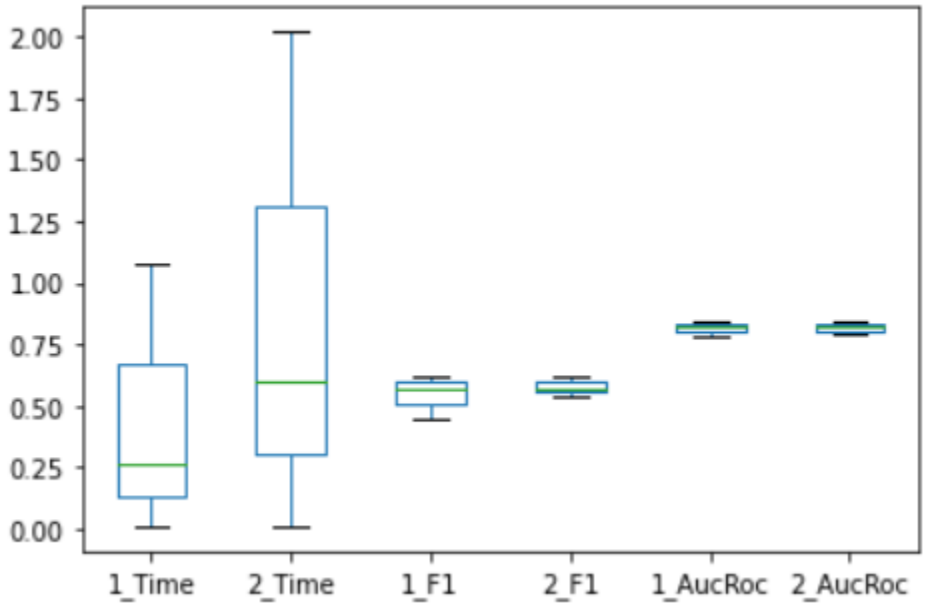
- Best area under the curve: logistic regression
- Best F1 score: Naive Bayes Gaussian
- Best time: Naive Bayes Gaussian
- Best overall: Naive Bayes Gaussian, which is first in third in area under the curve, and first with F1 score and time.

Comparing round 1 and round 2

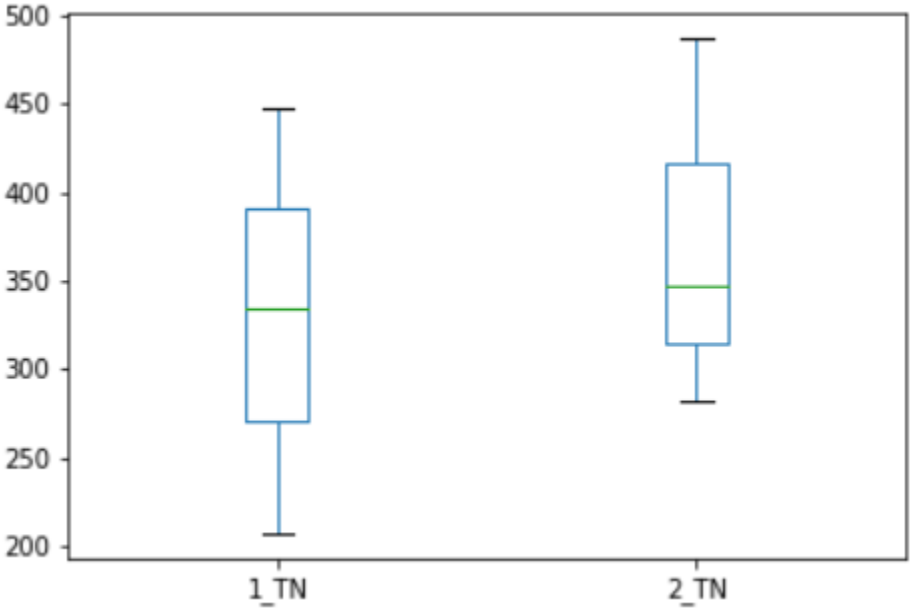
There are good models in both round 1 and round 2. Round 2 results are clustered more closely together.



Some of the round 2 models take longer. F1 and area under the curve are similar.



The big difference is that round 2 is better at identifying the people who churn.



Conclusions

Which machine learning model works best for binary classification?

This depends on how you define "best", but assuming that you care about F1 score and area under the curve, Naive Bayes (Gaussian), Logistic regression, and MLP are all good.

Does one-hot encoding impact the results?

Yes, but not as much as I expected. And the increased number of columns slowed several of the models down; if speed were an issue it might not be worth it to encode the variables.

Which metrics are useful in evaluating an imbalanced binary classification dataset?

F1 scored ended up being more useful than area under the curve. I also found it valuable to look at true negative classifications explicitly (they are part of F1 score), and would continue to use this for imbalanced datasets.

Next steps

New questions:

- If the dataset is balanced, how does that impact the results? Does up-sampling, down-sampling, or a combination work best to even out the cases?
- I'd like to explore Navie Bayes, MLP, and Logistic regression more. Next time, I can use cross validation and GridSearch to try more tuning parameters.