Hi! Some people ask me to share my solution, so I decided to write a post about it. Here are some main ideas I used in it.

At first, I used a great idea from Chahhou Mohamed: groupby features. It is a way to combine categorical and numerical features, which can greatly improve your score. You can see how it works here.

My second idea was to use the international wealth index. I encode each region with it.

My third idea is a feature, which gives me a top-1 score in the first 40 minutes of cracking the contest: decode **user_id**. When I started to explore the data, I found that **user_id** seems like a hash of some sort. So, I decided to go to https://crackstation.net and try to reverse it. Surprisingly it turned out that it is an SHA-1 hash of an integer. So I create a dictionary with the first **10_000_000** integers with their SHA-1 hashes and try to reverse all **user_id**s in the dataset. Even more surprisingly it turned out that all user_ids can be reversed in such a way, so my first solution became top-1 for almost a month. I use only default preprocessing (but nans imputed with medians instead of means) and one lightgbm model with two parameters: **class_weight="balanced"** and **boosting_type="dart"**. Later I also sorted the merged train-test dataframe by **user_id_int** and smooth CHURN by **savgol_filter**.

So I didn't push any solutions for almost three months. In the last week of the competition, I found myself outstripped by two competitors: AIC team and VKost. So on the last day I began to work on competition again. My final solution consists of three equally mixed models: two lightgbms and one catboost. I don't even try to properly optimize hyperparams, I just try some reasonable parameters and this is it:

lightgbm#1:
```
dict(n_estimators=500, boosting_type='dart',
class_weight='balanced', colsample_bytree=0.8, subsample=0.8,
reg_alpha=0.1, reg_lambda=0.1, learning_rate=0.05)
```
lightgbm#2:
```
dict(boosting_type='dart', class_weight='balanced',
n_estimators=1000, num_leaves=64, learning_rate=0.03,
reg_lambda=0.00001, reg_alpha=0.00001, subsample=0.8,
colsample_bytree=0.8)
```
catboost#1:
```
dict(iterations=1000, task_type='GPU', l2_leaf_reg=0.03,
learning_rate=0.05)
```

I also encoded all of the unique **TOP_PACK** strings by hand (at first I created a function that does it for me, but then decided that it would be better to encode every string by myself, because these strings were so inconsistent). It was tough and it wasn't worth it. Very small increment in score and 3 hours of pure pain: https://ibb.co/RyXWsK1

```
df['NANS_NUM'] = df.isna().sum(1)
```

For feature selection I use two techniques. One of them will be described in the section of libraries below and here is the other one: https://www.kaggle.com/ogrellier/feature-selection-with-null-importances. Just remember it, trust me.

Some tips and tricks:

1. You can easily revert your public score around 0.5 if you subtract your prediction from one. So instead of submitting real solutions, I used to push inverted ones: https://ibb.co/y8T2GKF. In such a way you can hide your score from other participants. And when I was outstriped by AIC team on LB, actually... I wasn't :D
2. You can easily create strong features if you perform subtraction multiplication and division of features. I created many features like these:
    - `df['ON_NET'] - (df['ZONE2'] + df['ZONE1'])`
    - `df['REVENUE'] * df['FREQUENCE']`
    - `((df['DATA_VOLUME'] + 1) / (df['ON_NET'] + 1)).apply(np.log)`

I want to mention two cool libraries that I used in my research of better solutions.

1. **boostaroota** (my second feature selection technique mentioned earlier). That library is based on boruta ideas but much much faster.
2. **memo** by one of my favorite DS guys: Vincent D. Warmerdam (he writes a lot of great small libraries and also has a number of great talks). Actually I don't use it much in this competition (because it was the last day of competition when I found it), but it is a very cool library. Just look at the tutorial and you understand what it is about.

I want to thank Zindi for such a great competition and the Expresso team for the clean dataset. It was great fun to compete here! Hope you can learn something useful from this post :P