# Class Kaggle challenge report

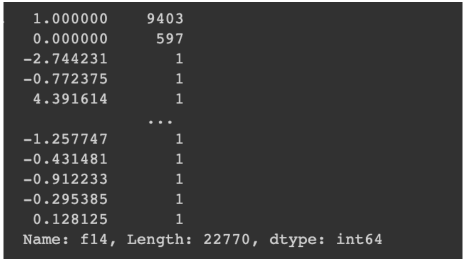
cangokalp

October 2020

## 1 Findings on features

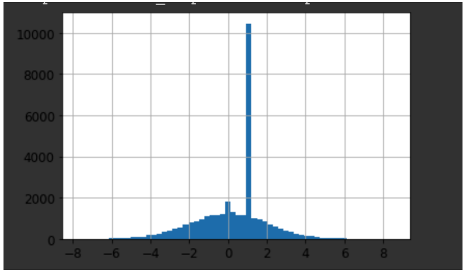
The data we had did not have feature descriptions and so we were blind in terms of what each column meant. Moreover, the numbers were arbitrary, confiscated and so it was not possible to deduce what they meant. Regardless, I investigated the histograms for each column. My most interesting finding here was column named 'f14';

Figure 1: Value Counts



```
 1.000000    9403
 0.000000     597
-2.744231       1
-0.772375       1
 4.391614       1
               ...
-1.257747       1
-0.431481       1
-0.912233       1
-0.295385       1
 0.128125       1
Name: f14, Length: 22770, dtype: int64
```

Figure 2: Histogram

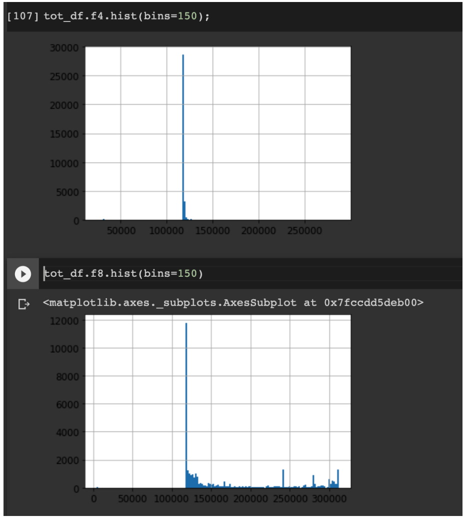


Looking at the value counts and histogram, it seems like this feature is repeating the target for certain rows and for other rows it's basically Gaussian noise. As the professor indicated in class that if we were able to find a leak, we should exploit - I decided to take the rows in the test set which had f14 of

1 or 0 and extended my training set with those. As those examples essentially have labels, I wanted to use them in the training to feed the model more data and hopefully improve my performance. After extending the training set, I then removed the f14 feature as this is now useless.

I also investigated other features, however did not make any changes. I noticed that feature f8 and f4 seemed to be capped from below at certain value;

Figure 3: Other features



```
[107] tot_df.f4.hist(bins=150);
```

```
tot_df.f8.hist(bins=150)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fccdd5deb00>
```

I also found some features with outliers. However, I did not make any changes on these features as I was going to use gradient boosting trees, and the trees are robust to outliers/feature scales.

# 2 Modeling

I used Catboost classifier as my model, and without any tuning at first I was already getting scores around 0.9s. Then I started tuning, at first I was doing the tuning manual to get a sense. Then I switched to grid search and eventually I switched to hyperopt library which supposed to explore the space depending on where in the search space it's most likely to get an improvement (given the space explored so far). I was looking to tune the hyperparameters; learning$_rate, l2_leaf_reg, andmax_depth.Inmymanualexperimentsitseemedthatforthehyperparametercolsampl$

# 3 Challenges

The biggest challenge I had was with the number of boosting rounds (iterations/trees). Catboost internally implements an overfitting detector. And so during the hyperparameter tuning phase, I let the iterations be a big number as the overfitting detector was gonna take care of it by looking at the metric (AUC) on the validation set and stop the model when the the metric on validation set stops improving for certain number of rounds (200). The issue was when I was fitting the model to the whole training data after finding the hyperparameters. The reason is that now I did not know how to set the iteration numbers, if I let it big it was going to overfit. There was no way to use an overfitting detector as there is no more unseen data.

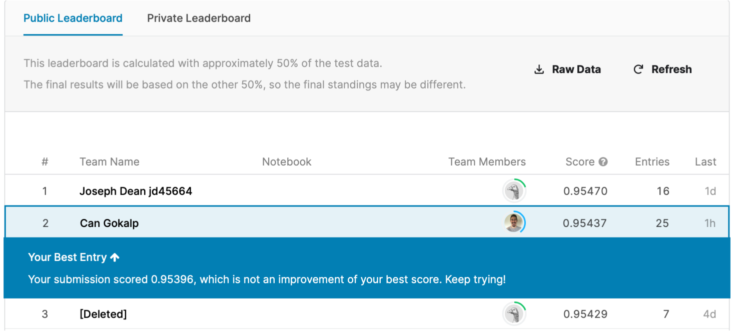
One thing I thought of was to not use the extra examples I gained from the test set as a result of the leak in the training, but use them to evaluate the hyperparameters on the full data - that way I would have a way to use the overfitting detector and the model would rollback to the number of iterations before overfitting. I did try this, however it did not improve my score on the leaderboard. I concluded that this advantage was less than the advantage gained by using the extra examples in the training.

Other idea I had was to use the hyperparameters found in the tuning phase and redefine and fit the model on train and valid sets and look at the iterations before it starts overfitting and by looking at those get an idea of the number of iterations I should use while fitting the model on the whole training data. Eventually I implemented this and looked at 10 different runs on different splits and observed the best model iteration count. In my submissions, I used the same hyperparameters except iteration numbers. I tried different values for this and made different submissions.

In hindsight, I believe I'd be better of if I tuned for both learning rate and number of trees without using the overfitting detector and that way I wouldn't have the issue I think.

I achieved the score of 0.95437 and got second on the public leaderboard.

Figure 4: Public Leaderboard



This leaderboard is calculated with approximately 50% of the test data.
The final results will be based on the other 50%, so the final standings may be different.

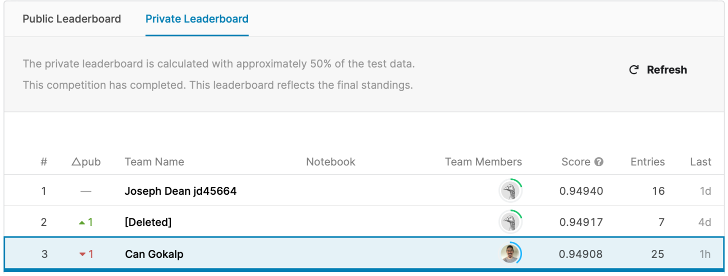
| # | Team Name | Notebook | Team Members | Score | Entries | Last |
|---|---|---|---|---|---|---|
| 1 | Joseph Dean jd45664 | | | 0.95470 | 16 | 1d |
| 2 | Can Gokalp | | | 0.95437 | 25 | 1h |
| 3 | [Deleted] | | | 0.95429 | 7 | 4d |

Your Best Entry ↑

Your submission scored 0.95396, which is not an improvement of your best score. Keep trying!

For evaluation in the private leaderboard, as instructed, I chose the 3 submissions where I tried different number of iterations while keeping the other hyperparameters the same. My rank dropped by one in the private leaderboard. I believe this is because of the different number of iteration models I submitted. One of them worked really well, whereas the other two was about 0.01 less than the best one. My guess is that their average might have reduced

Figure 5: Private Leaderboard



The private leaderboard is calculated with approximately 50% of the test data.
This competition has completed. This leaderboard reflects the final standings.

| # | △pub | Team Name | Notebook | Team Members | Score | Entries | Last |
|---|---|---|---|---|---|---|---|
| 1 | — | Joseph Dean jd45664 | | | 0.94940 | 16 | 1d |
| 2 | ▲1 | [Deleted] | | | 0.94917 | 7 | 4d |
| 3 | ▼1 | Can Gokalp | | | 0.94908 | 25 | 1h |

my score.