Name: Bogdan Zhurakovskyi

Location: Kyiv, Ukraine

Email: zhurak@gmail.com

Competition: Prudential Life Insurance Assessment

Summary

- **How you spent your time (Eg. what proportion on feature engineering vs. machine learning?)**

Half of time I spend on calculating probabilities for every label and different combination of them. I tried a lot of combination such as [2,3] vs all, [4,5,6] vs all and many other but without any success. So finally I left only 13 probabilities – one for each label and [1,2] vs all, [1,2,3] vs all, [1,2,3,4] vs all, [1,2,3,4,5] vs all, [1,2,3,4,5,6] vs all. Other half I spend on regression. I used only three engineered features: the product of BMI and Ins_Age, the count of med keywords and the count of NA for each sample.

- **Your most important insight into the data**

This dataset has no respect to your efforts and overfits whatever you do. So it is better to concentrate on robustness then accuracy.

- **The training methods you used**

For calculating probabilities I used an ensemble of gradient boosting, random forest and logistic regression. The weights were [0.8,0.15,0.05] respectively.

- **The tools you used**

Sklearn and XGBoost

Features Selection / Extraction

I didn't spend much time on feature engineering this time. I tried polynomial transformation and different dimension reduction technics but it did not gave me anything. The only useful thing was product of BMI Age and Medical Keywords sum. They helped to improve the model significantly. Many thanks to some good people on forum who advised it.

Training Methods

The main idea was following:

- **Calculate probabilities for each label and its combinations then add them as additional features.**

To do this I splited the train set on 10 pieces and used 9 of them to predict the last. For the test set I used all train set. Also I tried to split the train set on 50 pieces but it gave me almost nothing. During this step I built a lot of ensembles of xgboost's, random forest's, svm's and other classifiers taking into account that the more accurate the probabilities the more accurate the regression. As metric I chose the AUC score and made some huge ensembles of a lot of classifiers. But then was the surprise – the kappa score only fell. So I made a conclusion that there is a tradeoff between model complexity and connection between train and test data.

- **Apply regression and then search for the best splits for rounding predicted values.**

On this step I also tried some difficult ensembles of xgboost's and svm's but result was always worse than usual Linear Regression. I suppose that is due to the fact that all " hard work" were done during first step.

Simple Features and Methods

The model with only binary xgboost during probabilities calculation step scored 0.67762 which would be 10-th on private leaderboard.

Interesting findings

The one should always try linear models! Never underestimate it.

Background on me

- **What was your background prior to entering this challenge?**

I am Ph.D. Candidate in Statistics at the Kyiv Polytechnic Institute. * **Did you have any prior experience that helped you succeed in this competition?**

Just usual machine learning methods * **What made you decide to enter this competition?**

Transparent problem and a huge number of participants. * **How much time did you spend on the competition?**

Around 100 hours

Appendix

A1. Dependencies

Sklearn and XGBoost is enough.

A2. How To Generate the Solution

train.py to generate csv files with probabilities for decoded labels. (They are already generated and located in features folder).

predict.py to make prediction for test set using generated features from train.py.