RISK CLASSIFICATION FOR LIFE INSURANCE

Kaggle project
Larry (Jianfeng) Yan

Contents

- Objective
- Exploratory data analysis
- Modeling
- Result and evaluation
- Visualization

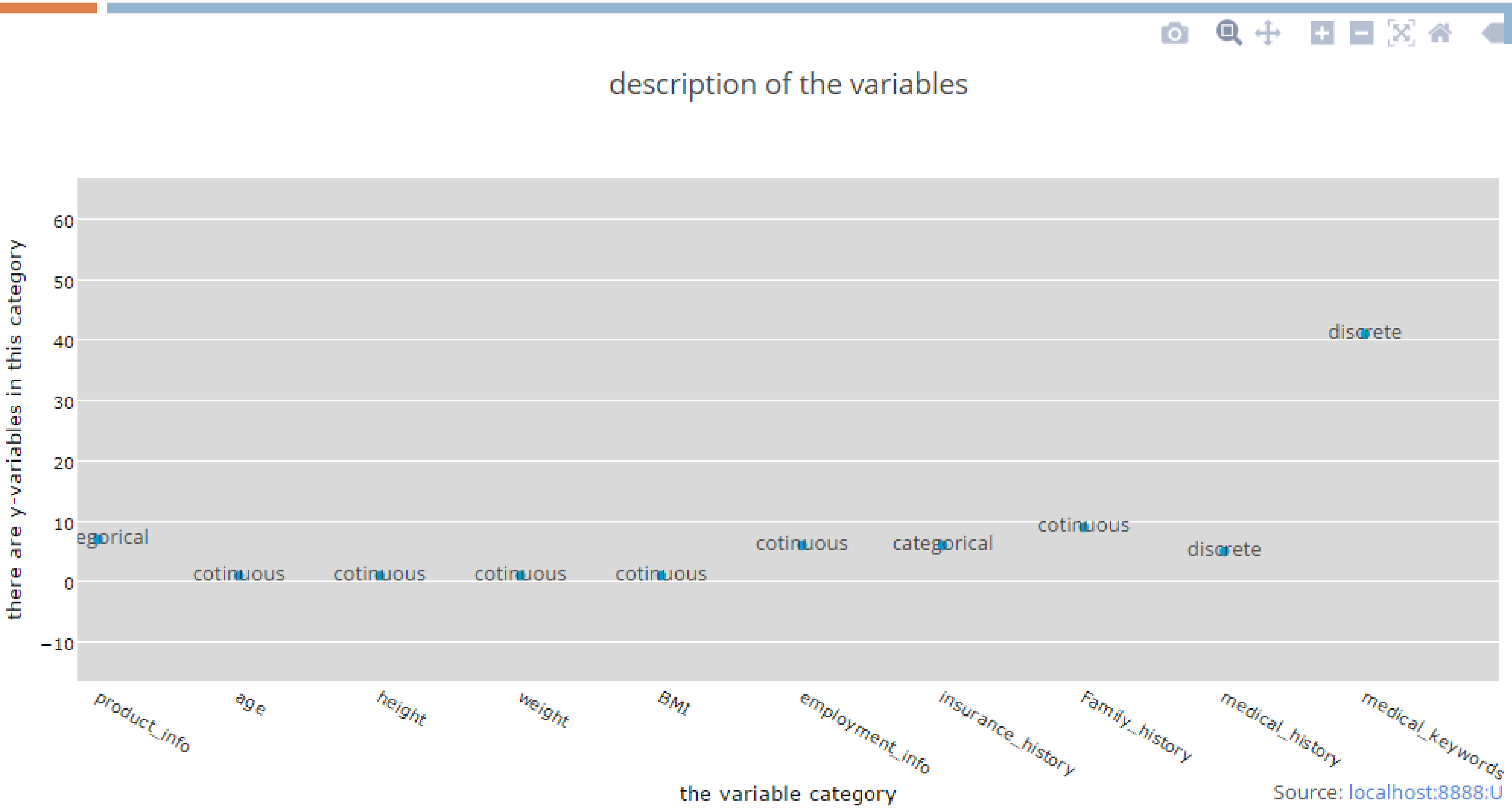
Objective

- Problem: Identifying risk classification and eligibility is labor intensive and slow, for life insurance.
- Solution: Automatically classifying the risk level, given the data of clients.

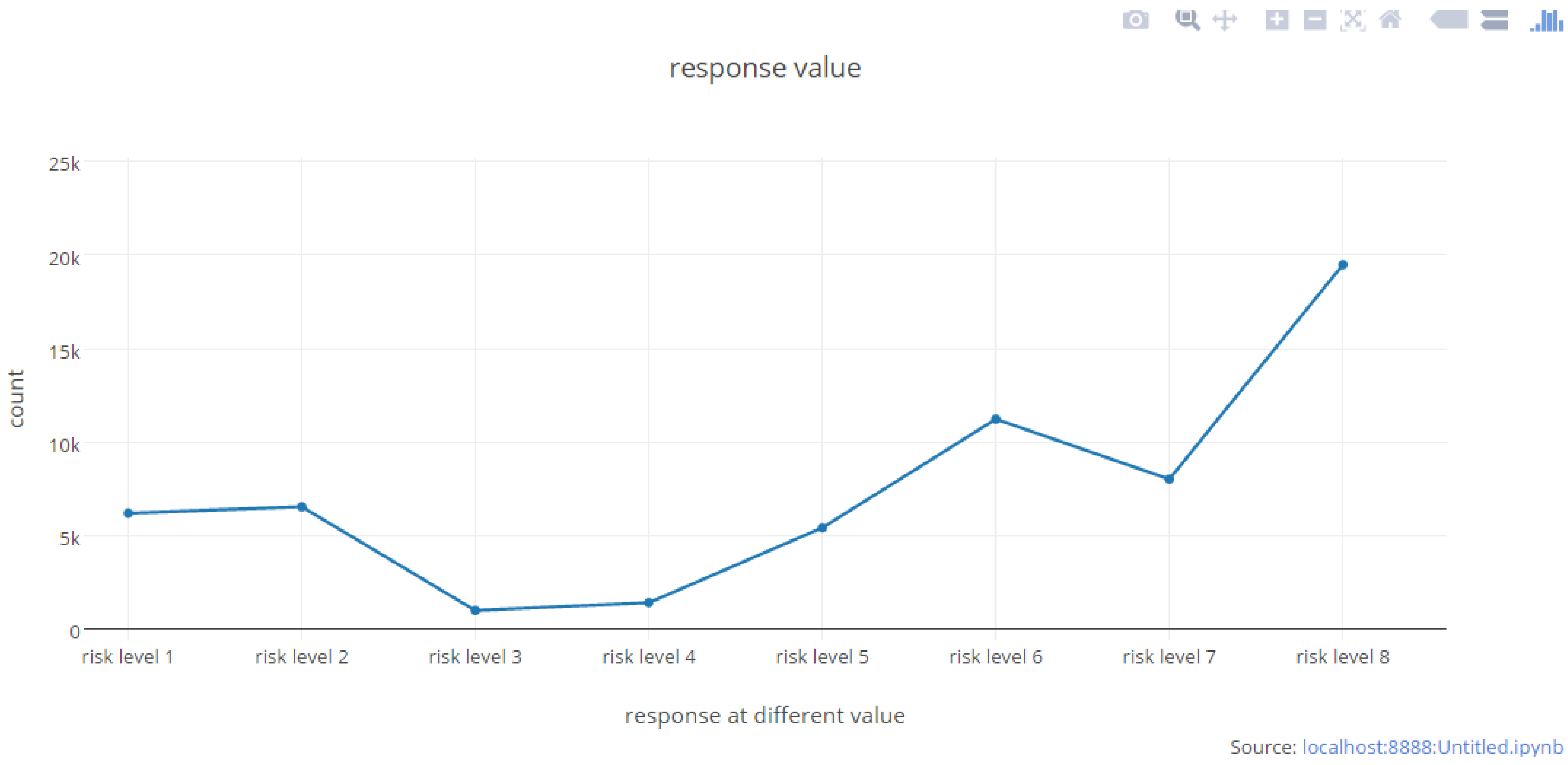
Exploratory data analysis (EDA)

- Data with 126 features, and a Y variable.
- Y variable is the risk level, from level 1 to level 8.
- Among the 126 features, 60 are categorical, 13 continuous, 53 are discrete.
- So the objective is to make a classification of the risk level according to the 126 features.
- For better understanding of the variables, visualize it with two graph.

A first look at the features (EDA)



A first look at Y (EDA)



Data manipulation (EDA)

- Create dummy values for categorical variables
- Missing value:

  continuous: using mean to replace

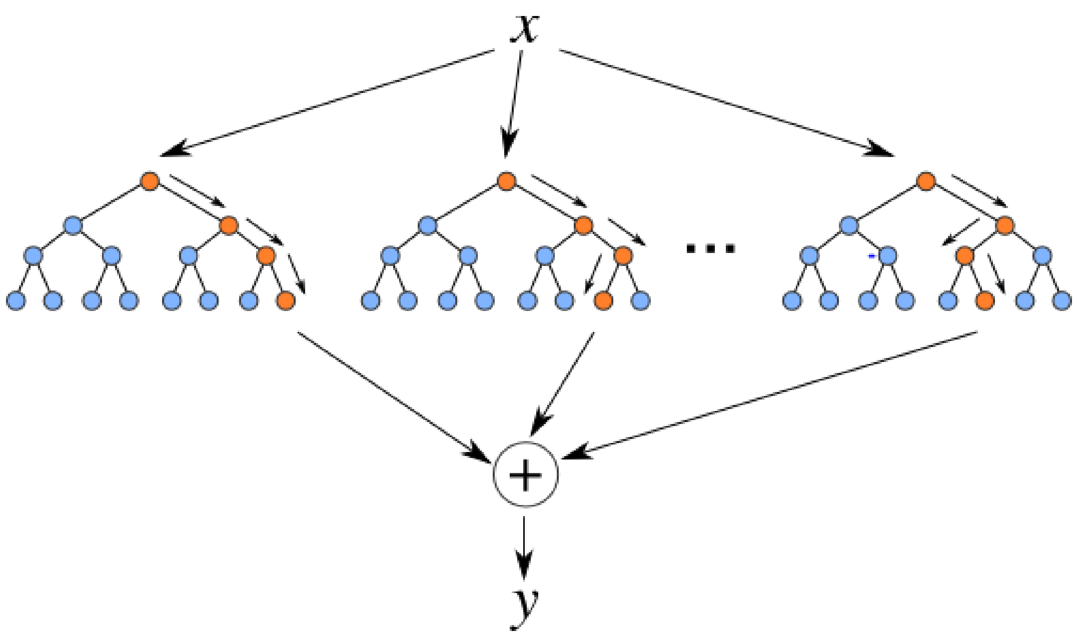
  categorical: create dummy variable,

  0-missing, 1- non-missing
- Normalization.
- Split the data into train data and test data for cross validation

Models

- Random forest
- Adaptive boosting tree (Adaboosting)
- Gradient boosting Linear Regression
- Gradient boosting Possion Regression
- Artificial neural network

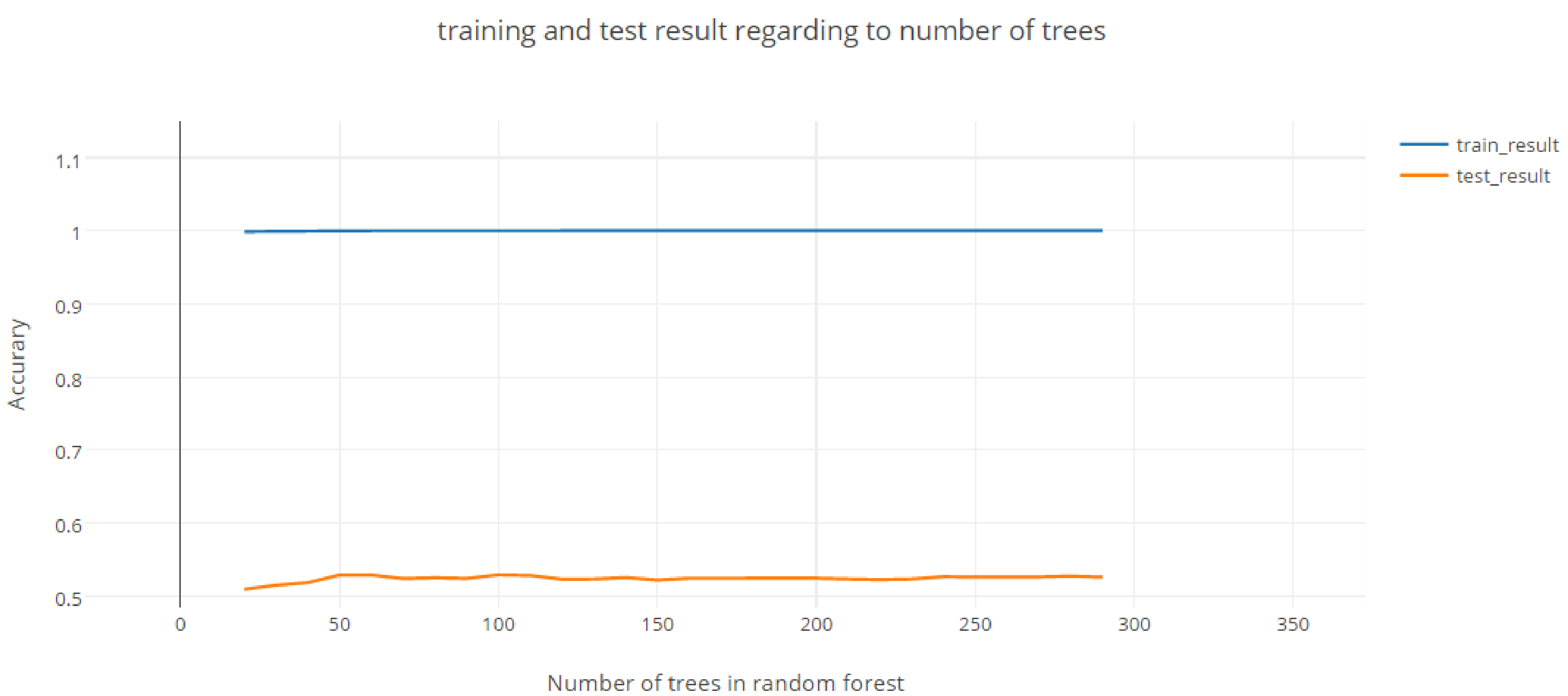
Model: Random Forest



- Generate N trees
- Using Bootstraping to train the trees
- Ensemble the result, make prediction.

Model: Random Forest

- Result



Model: Adaboosting

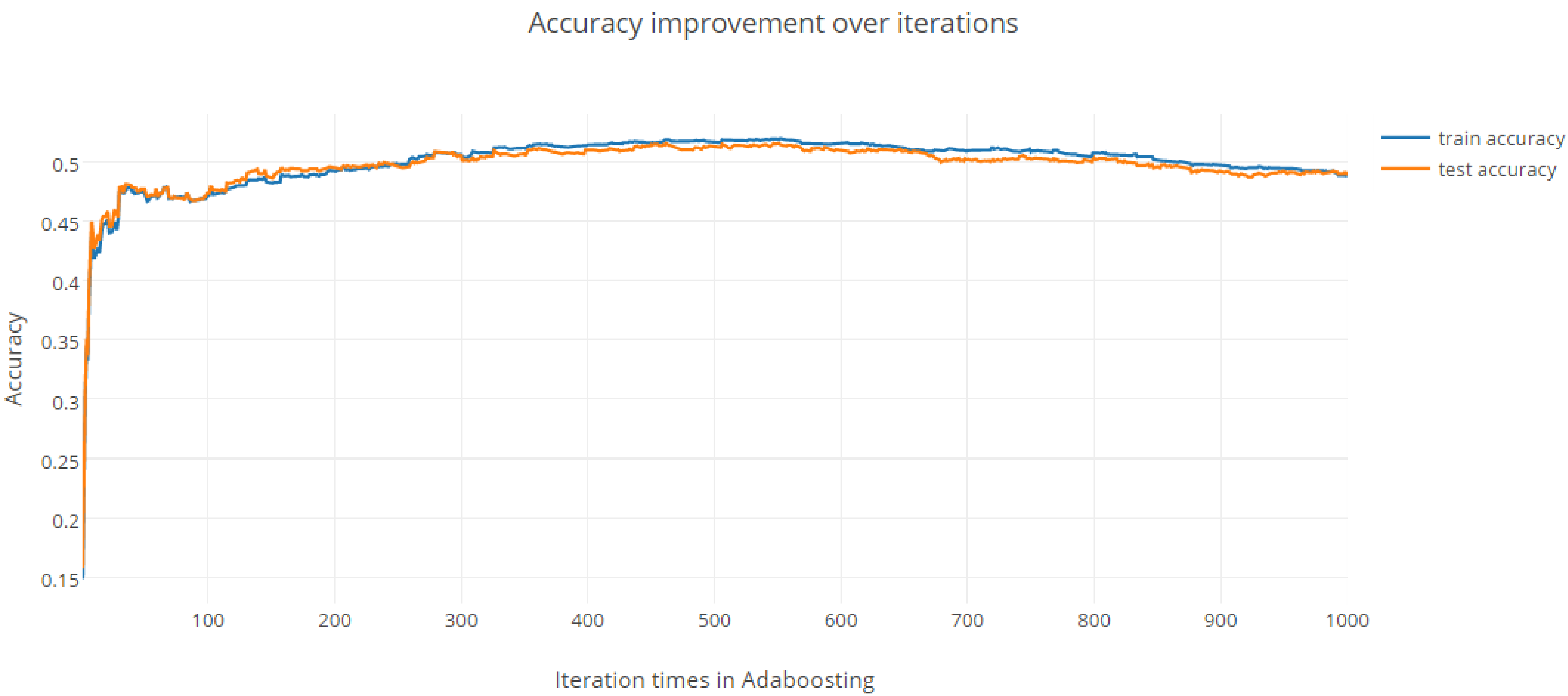
- Combine weak learners to get a strong learner.
- Iteratively adjust the weight of samples.
- Add more weight to wrongly classified samples.

Algorithm 16.2: Adaboost.M1, for binary classification with exponential loss

$w_i = 1/N$;
for $m = 1 : M$ **do**
    Fit a classifier $\phi_m(\mathbf{x})$ to the training set using weights $\mathbf{w}$;
    Compute $\text{err}_m = \frac{\sum_{i=1}^{N} w_{i,m}\mathbb{I}(\tilde{y}_i \neq \phi_m(\mathbf{x}_i))}{\sum_{i=1}^{N} w_{i,m})}$;
    Compute $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$;
    Set $w_i \leftarrow w_i \exp[\alpha_m \mathbb{I}(\tilde{y}_i \neq \phi_m(\mathbf{x}_i))]$;
Return $f(\mathbf{x}) = \text{sgn}\left[\sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x})\right]$;

Model: Adaboosting

- After 1000 iterations, the improvement of prediction accuracy for training and testing data:



Stochastic Gradient Boosting Regression

Why I use regression ?

1. Risk level from 1 to 8, its rank cardinal variable, it's meaningful!
2. Classification don't consider ranking.
3. Regression may fit better for continuous variable, for this discrete variable, using poisson regression.
4. Too many features (over 900): using stochastic gradient boosting.

Stochastic Gradient Boosting Regression

My parameter setting. (Using xgboost package)

```
def get_params():

    """
    eta:  actually shrinks the feature weights afte each iteration of boosting,
     to make the boosting process more conservative
    objective: I tried linear regression and poisson regression, poission is better.
    min_child_weight: minimum sum of instance weight needed in a child

    """
    params = {}
    params["objective"] = "reg:linear"
    params["eta"] = 0.06
    params["min_child_weight"] = 80
    params["subsample"] = 0.85
    params["colsample_bytree"] = 0.30
    params["max_depth"] = 9
    plst = list(params.items())
    return plst
```

Stochastic Gradient Boosting Regression

SGB Linear Regression tree

- Before using offset

  Train vs test

  0.6910 vs 0.6169

- After using offset:

  Train vs test

  0.7410 vs 0.6440

SGB Poisson Regression tree

- Before using offset

  Train vs test

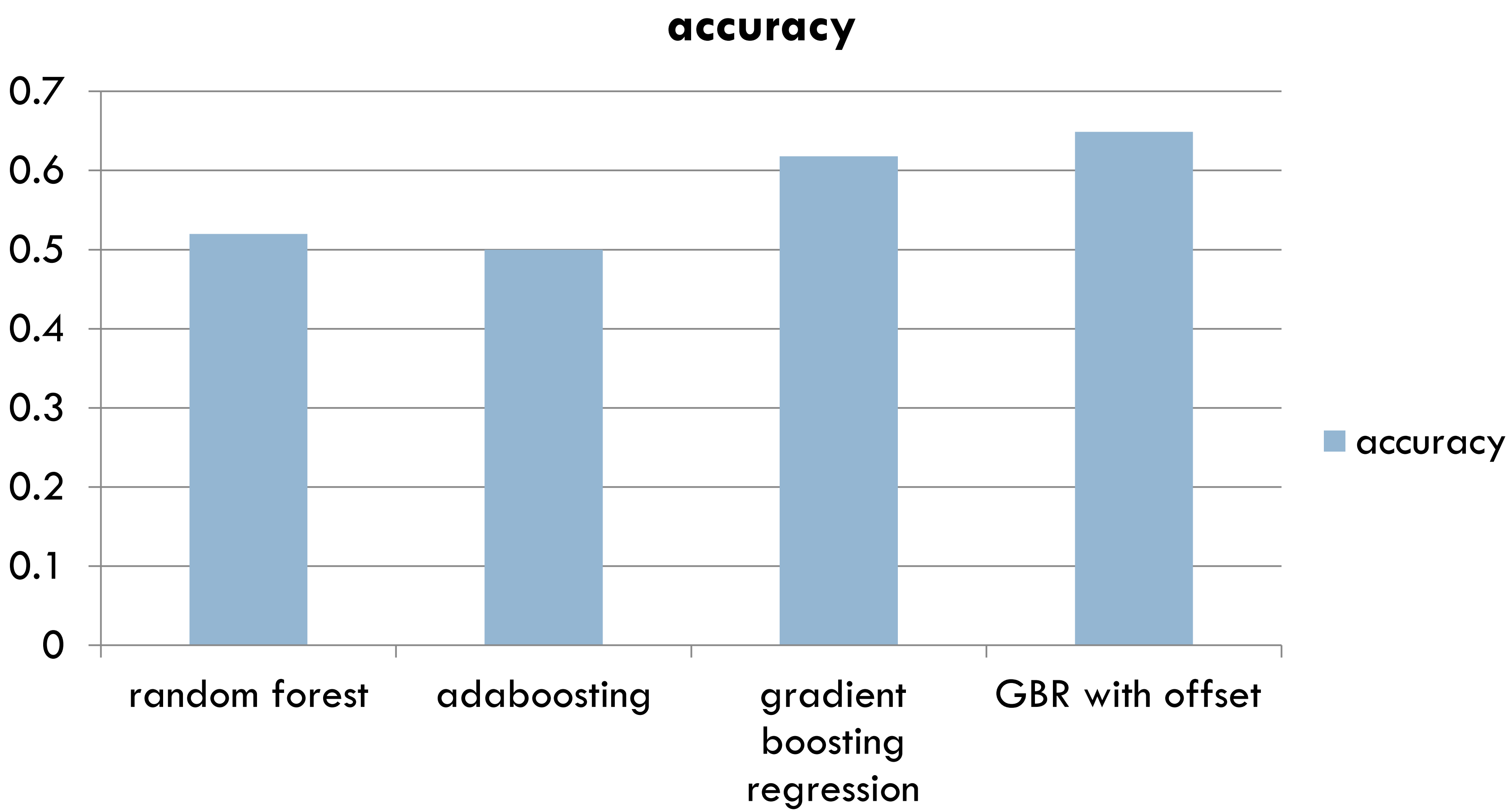
  0.7090 vs 0.6179

- After using offset:

  Train vs test

  0.7669 vs 0.6493

compare models



Stochastic Gradient Boosting Regression

- What is offset?

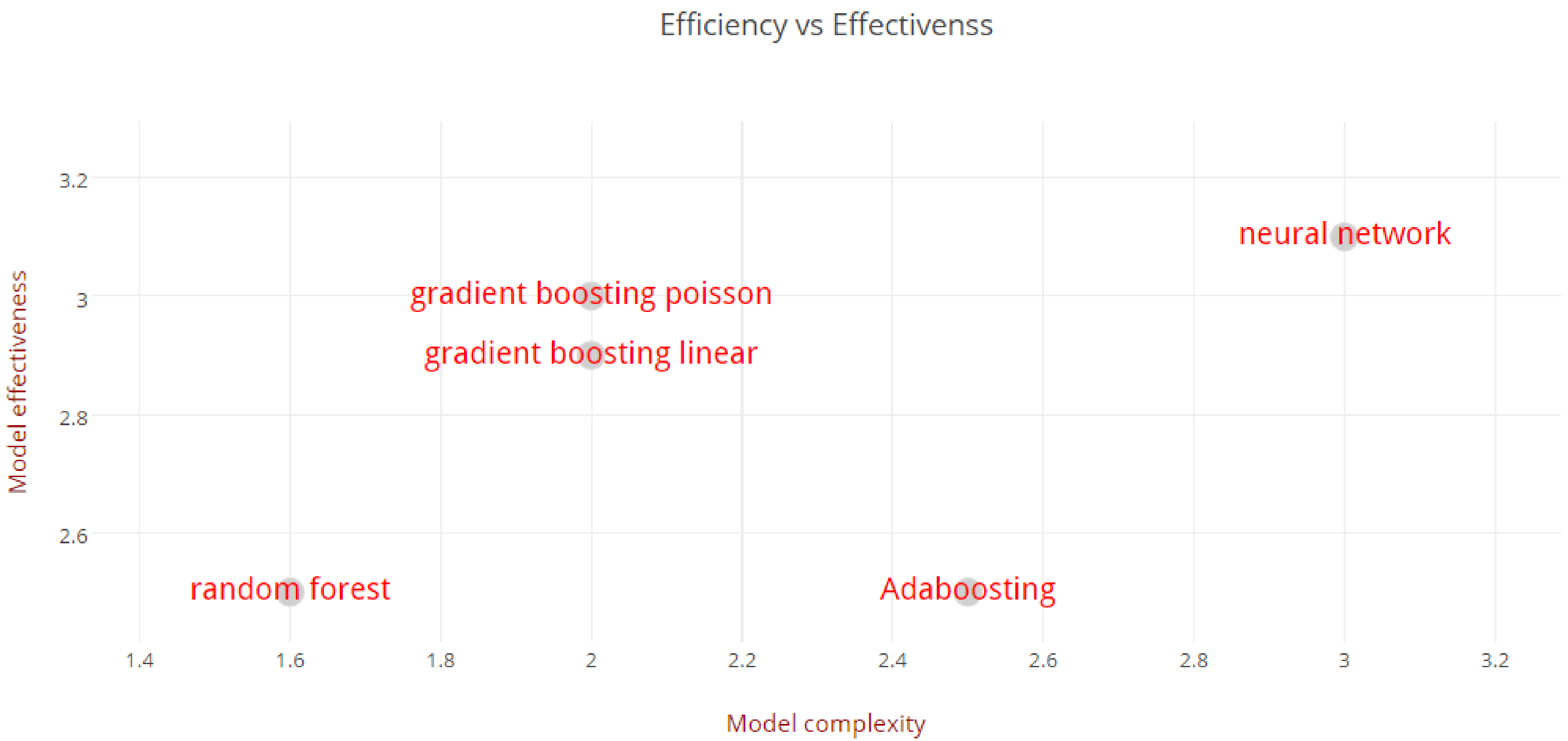
Adjust the predicted value by adding a constant to make it fitting the real value better.

For example.

Predicted value {2.71, 2.84, 2.91, 3.1, 3.43, 3.45} is labeled as class 3.

If the real value is {3, 3, 3, 3, 4, 4}. Then adding 0.2 to the predicted value can make the overrall result better.

Evaluation



Efficiency vs Effectivenss
neural network
gradient boosting poisson
gradient boosting linear
random forest
Adaboosting
3.2
3
2.8
2.6
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
Model effectiveness
Model complexity

The end

Thank you for your time.

Sorry for the lack of neural network in my ppt.

Some Links:

https://plot.ly/~jy2641/ (plots for this ppt)

https://www.kaggle.com/c/prudential-life-insurance-assessment (link for the data)

https://github.com/Larryjianfeng (my github)