A Hedge Fund Machine Learning Modeling Challenge

Team: More or Less by Ziqiao Liu, Lei Zhang

- ❏ Background Knowledge
- ❏ Explore Data Visualization
- ❏ Model Development Process
- ❏ Approach with “Less”
- ❏ Approach with “More”

Background Knowledge

- Numerai is a hedge fund which manages an institutional grade long/short equity strategy for their investors to make trades. https://numer.ai/
- Provided data has been encrypted for developing machine learning models.
- Numerai releases a new dataset each week and the competition resets.

Background Knowledge

- Problem: How to predict the stock market by using Machine Learning Models
- Data:

  Train--- Numerai_trainning_data:

  21 features(0-1), 1 target(0,1), 136573 observations

  Test--- Numerai_tournament_data:

  21 features, 13518 observations
- Model Performance Measurement:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right)$$

Explore Data Visualization

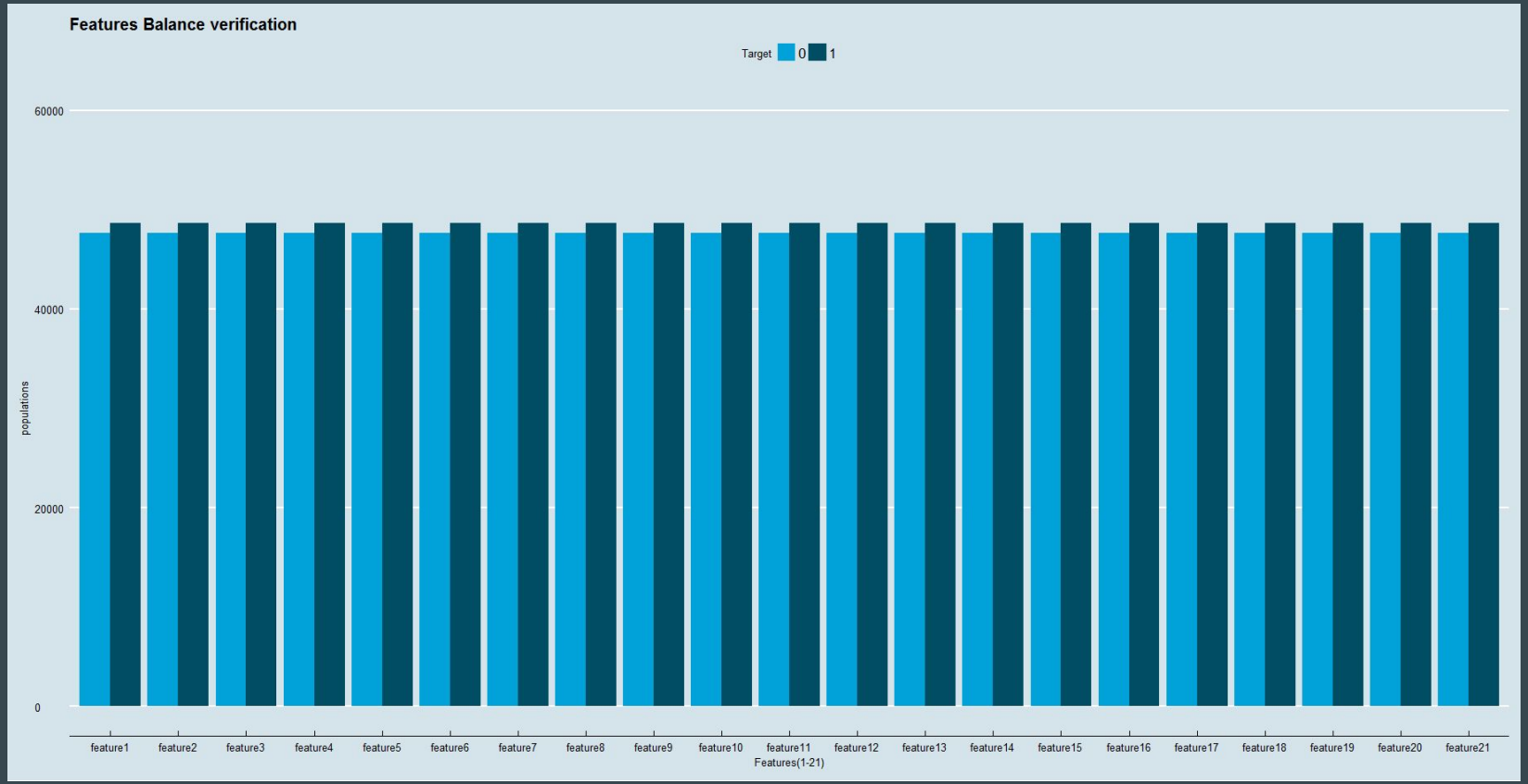
- A sample of a training data

feature11	feature12	feature13	feature14	feature15
Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.0000
1st Qu.:0.2665	1st Qu.:0.2527	1st Qu.:0.2431	1st Qu.:0.2764	1st Qu.:0.2308
Median :0.5041	Median :0.4878	Median :0.4940	Median :0.5370	Median :0.4954
Mean :0.5079	Mean :0.4936	Mean :0.4926	Mean :0.5247	Mean :0.4907
3rd Qu.:0.7730	3rd Qu.:0.7390	3rd Qu.:0.7529	3rd Qu.:0.7774	3rd Qu.:0.7432
Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000

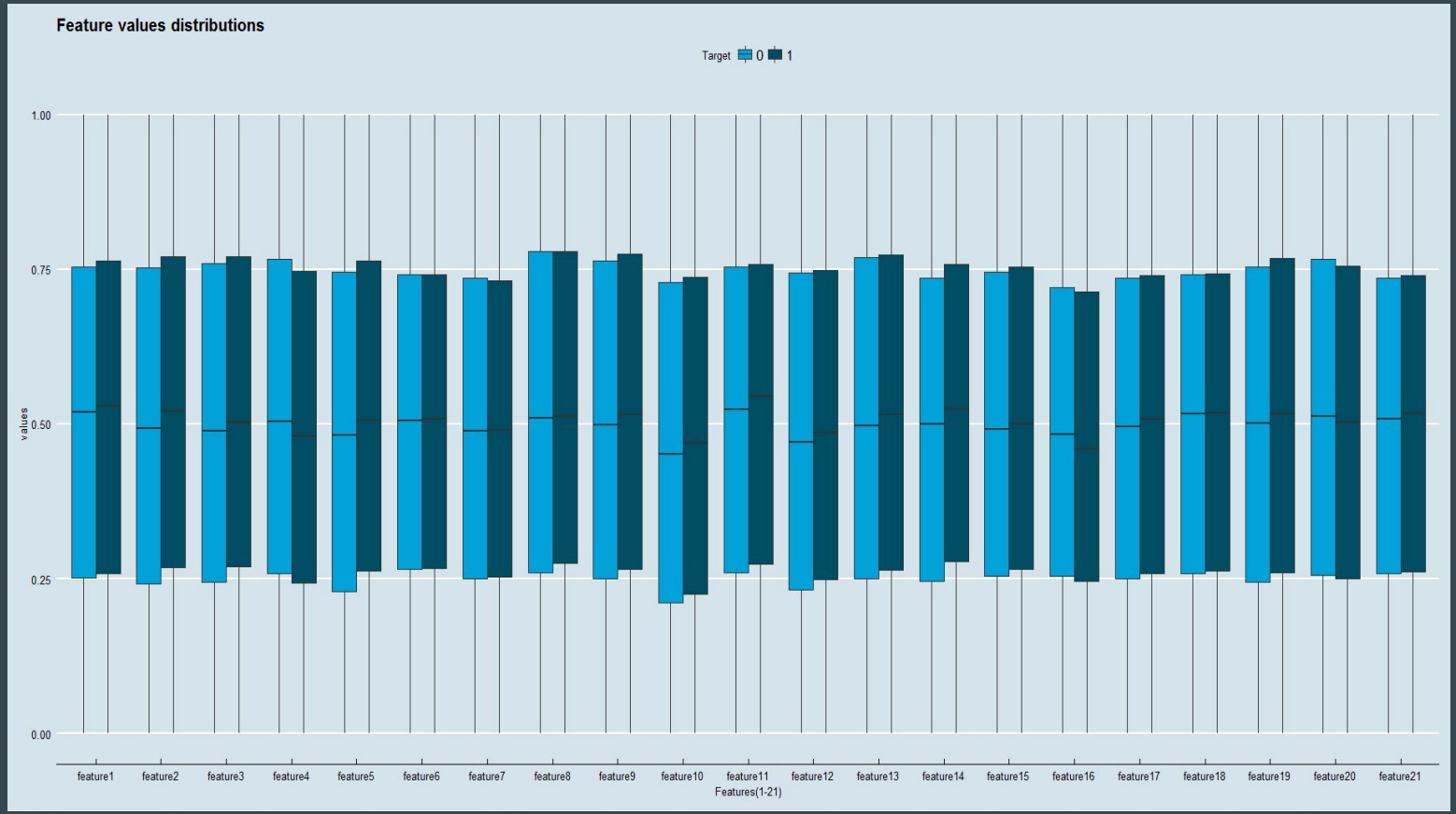
feature16	feature17	feature18	feature19	feature20
Min. :0.0000	Min. :0.0000001	Min. :0.0000	Min. :0.0000	Min. :0.0000
1st Qu.:0.2616	1st Qu.:0.2405456	1st Qu.:0.2532	1st Qu.:0.2857	1st Qu.:0.2423
Median :0.4900	Median :0.5129868	Median :0.4768	Median :0.5509	Median :0.4897
Mean :0.4996	Mean :0.5072219	Mean :0.4918	Mean :0.5308	Mean :0.4870
3rd Qu.:0.7487	3rd Qu.:0.7624780	3rd Qu.:0.7301	3rd Qu.:0.7916	3rd Qu.:0.7380
Max. :1.0000	Max. :1.0000000	Max. :1.0000	Max. :1.0000	Max. :1.0000

feature21	target
Min. :0.0000	Min. :0.0000
1st Qu.:0.2783	1st Qu.:0.0000
Median :0.5221	Median :1.0000
Mean :0.5157	Mean :0.5033
3rd Qu.:0.7575	3rd Qu.:1.0000
Max. :1.0000	Max. :1.0000

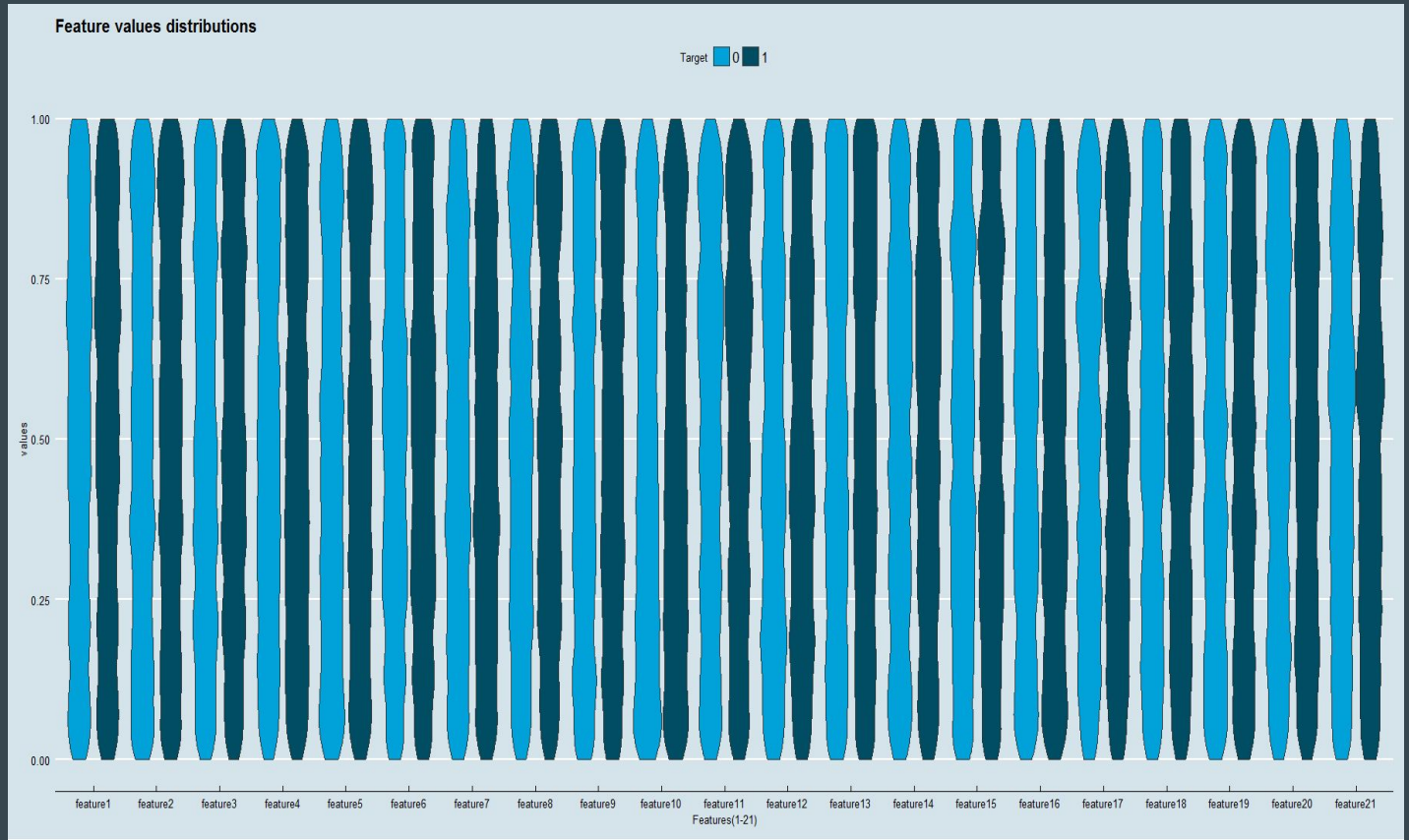
Explore Data Visualization — Feature Balance



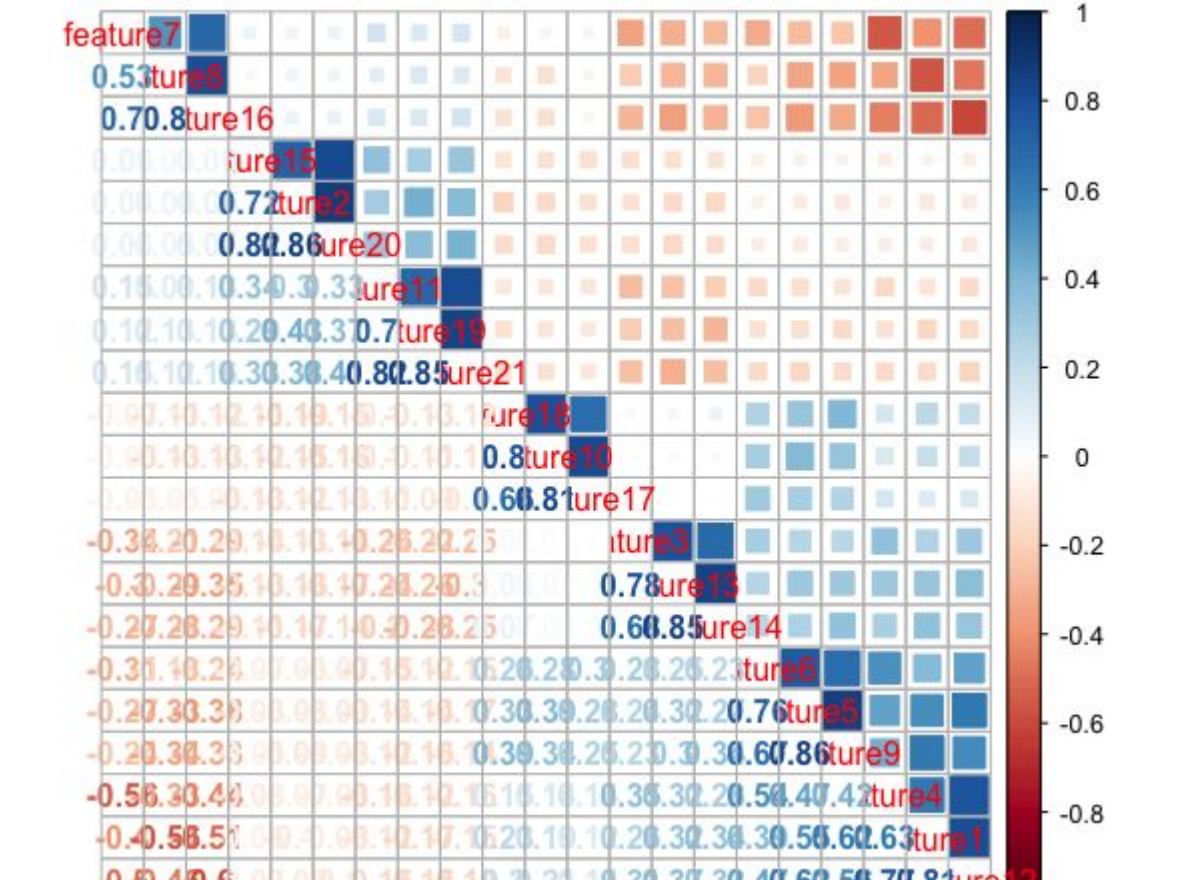
Explore Data Visualization – Feature Distributions



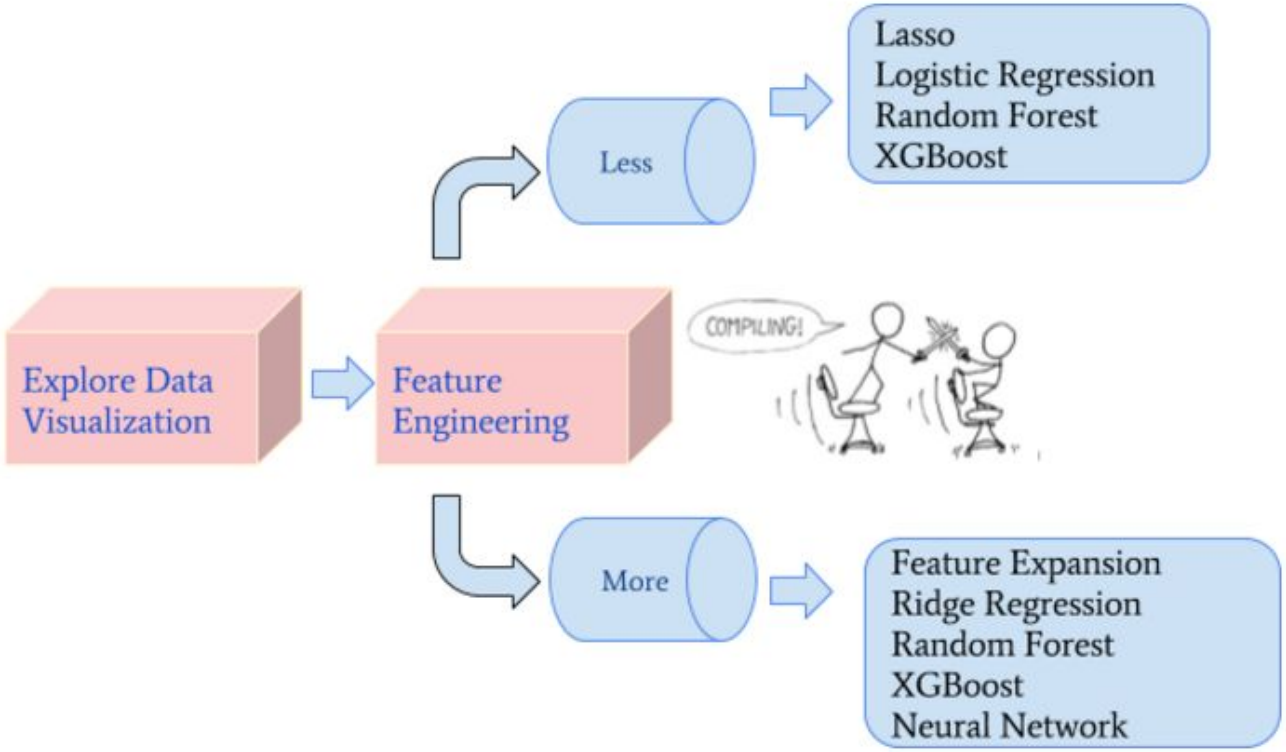
Explore Data Visualization – Feature Distributions



Explore Data Visualization – Correlation



Model Development Process



Less: Model Selection

- Feature Exploration:

Lasso, Random Forest

- Prediction Model Training:

Logistic Regression: basic, easy to implement, efficient to train, initial try

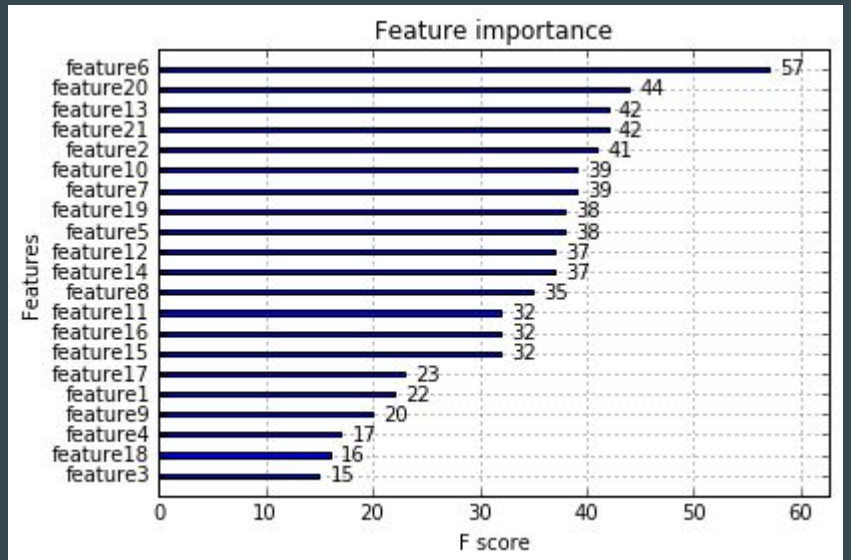
Random Forest: a highly accurate classifier, flexible, for large dataset

XGBoost: flexible, has more costume parameters, suitable for competition

Less: Feature Exploration

```
In [188]: lasso.coef_
Out[188]:
array([-0.        ,  0.        , -0.        ,  0.00254346,  0.        ,
        0.03424556, -0.        ,  0.        , -0.        ,  0.02140864,
        0.        ,  0.        , -0.01470154, -0.        ,  0.        ,
        0.        ,  0.        ,  0.0015325 ,  0.02632659,  0.03243665,
        0.01840176])
```

feature 4,6,10,13,18,19.20,21
alpha: 1e-3



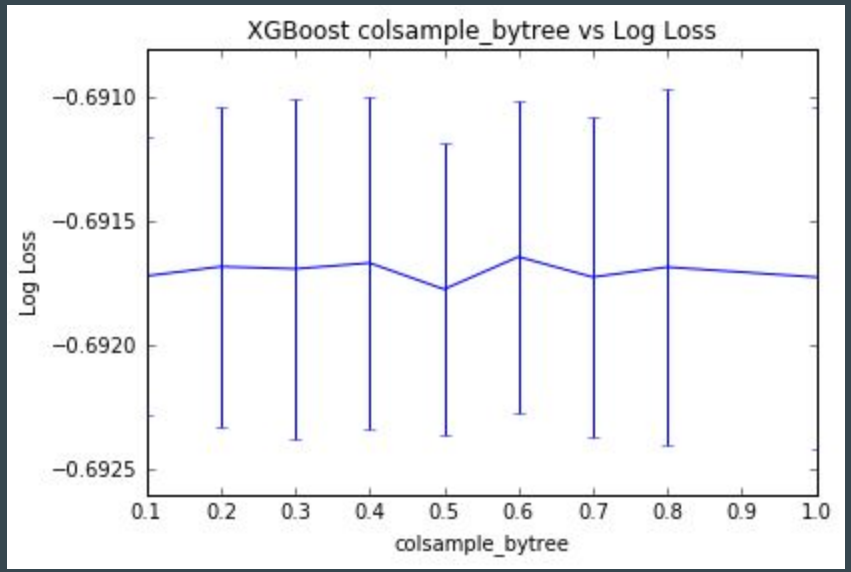
feature 6, 20, 13, 21, 10, 2, 7, 9, 5, 12
14, 8, 11, 16, 15, 17, 1, 9, 4, 18, 3

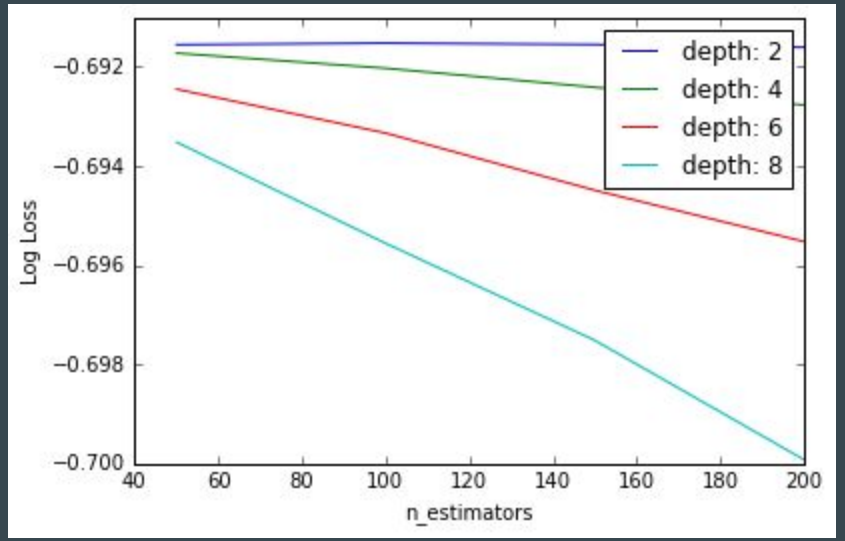
Less: Logistic Regression, Random Forest

- Logistic Regression: 0.68910
- Random Forest: 0.6950144177748 (n_estimators=300, 500, 800)
- XGBoost Parameters Tuning with Cross Validation: 0.690

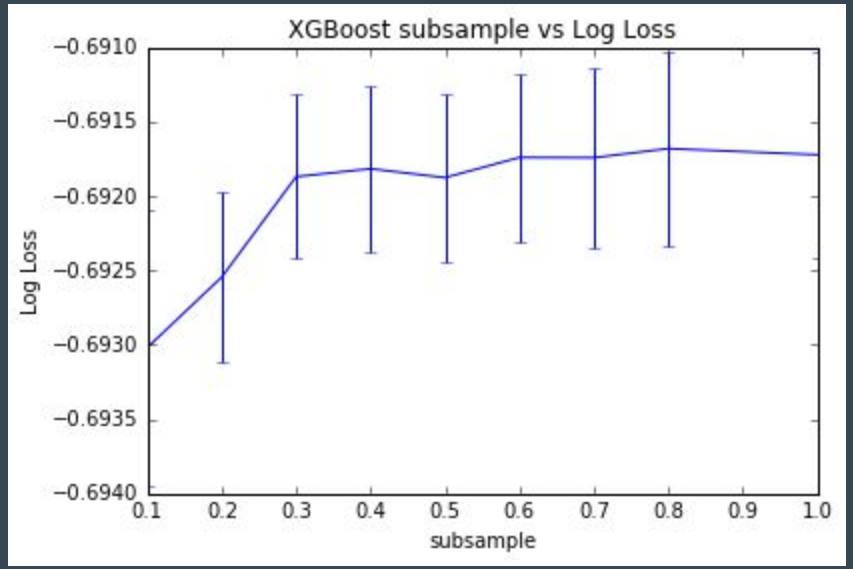
Number of trees	range(50, 50, 300)
Colsample by tree	[0.1, 0.2, 0.4, 0.6, 0.8, 1.0]
Max depth	[2, 4, 6, 8]
Sub sample	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
Learning rate	[0.001, 0.1, 0.2, 0.3]

Less: XGBoost Parameter Tuning Process





Less: XGBoost Parameter Tuning Process

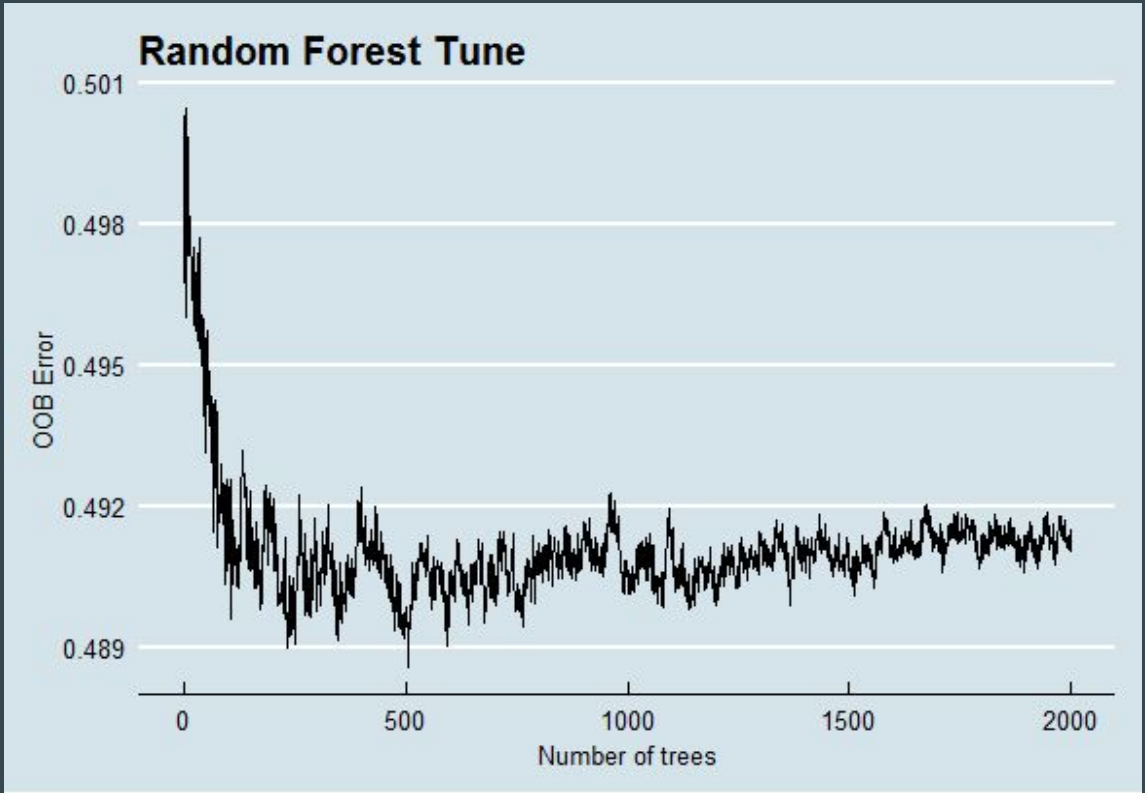
- 

Conclusion

- Tree Based Models' performance does not meet expectations
- Dataset is more suitable to Logistic Regression models
- Implement feature reduction results in model training
- Model ensemble in future steps

Approach : More

Random Forest — Parameter Selection



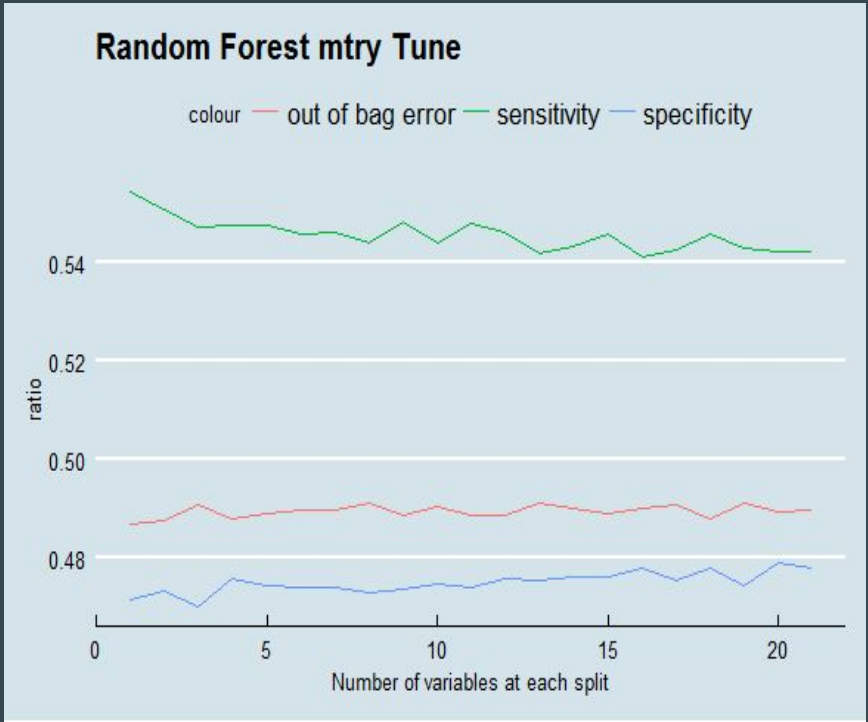
Step 1 : Determine the range of trees' number

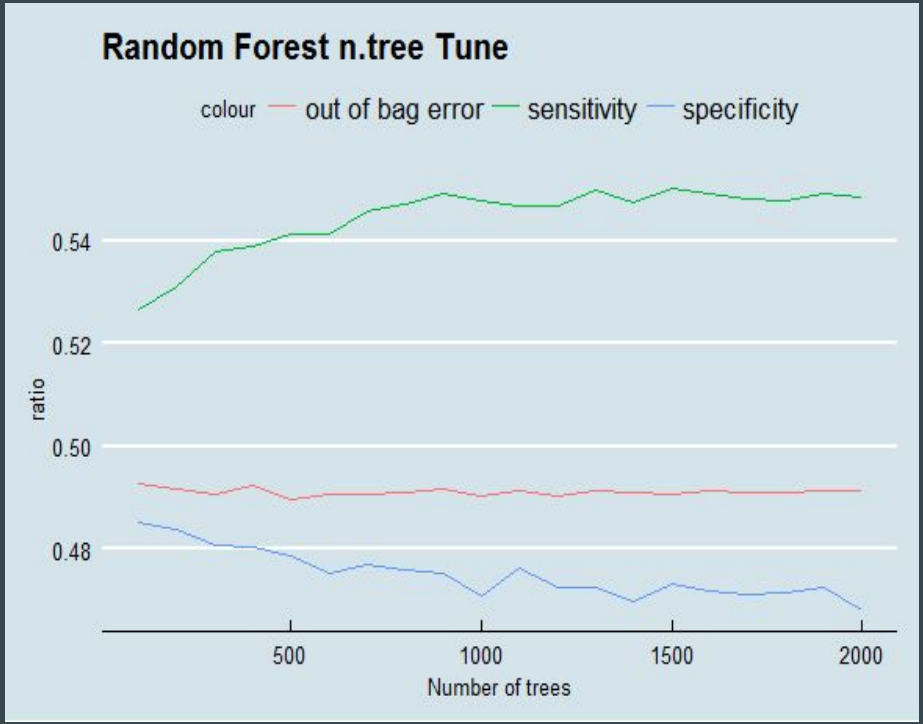
Step 2 : Determine the best parameter of mtry and number of trees

Step 3 : Try different kinds of threshold (such as sensitivity,

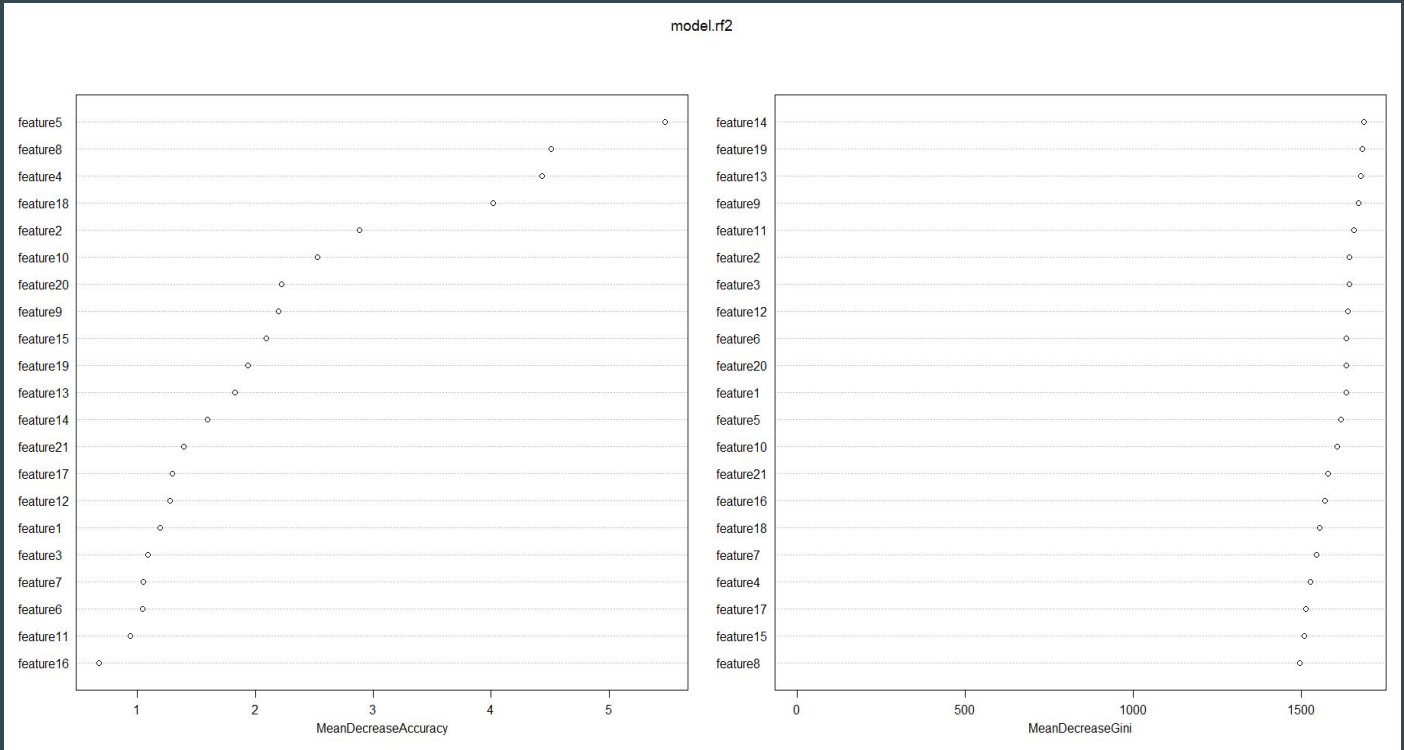
Specificity, out of bags error)

Random Forest — Parameter Selection





Random Forest — Feature Importance



Conclusion :

Hard to do feature selection

Random Forest — Result and Problem

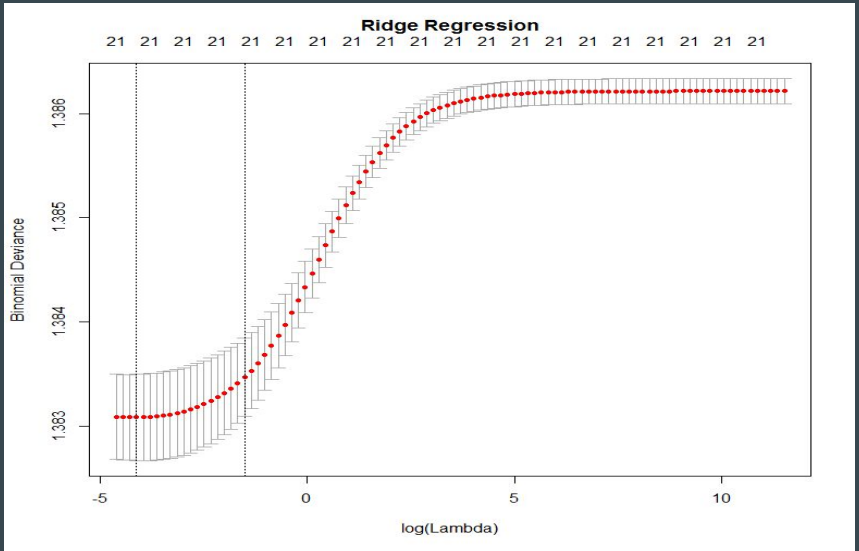
Pros:

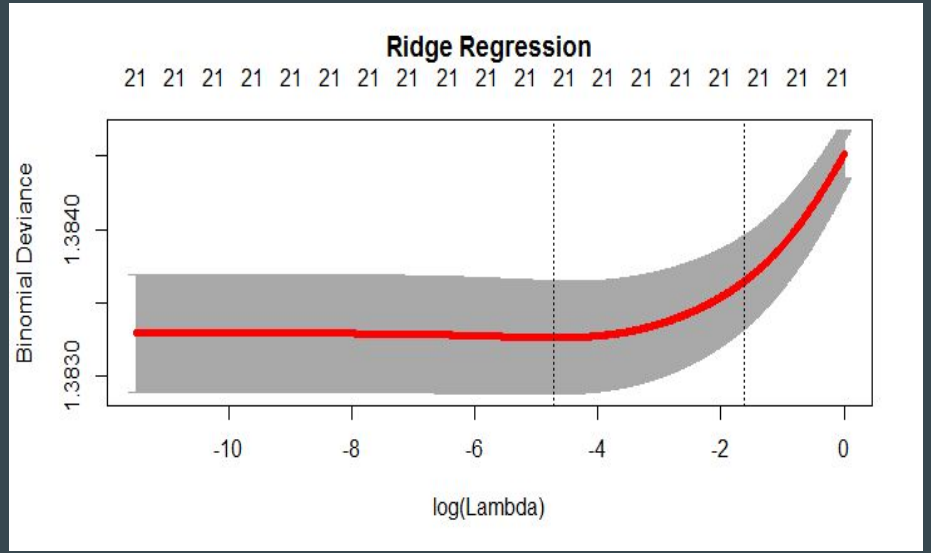
- Fast
- Easy to cross validation, not sensitive to overfit (too few features and possible to overfit with large number of trees);

Cons:

- Low accuracy

Ridge Regression — Lambda by Deviance

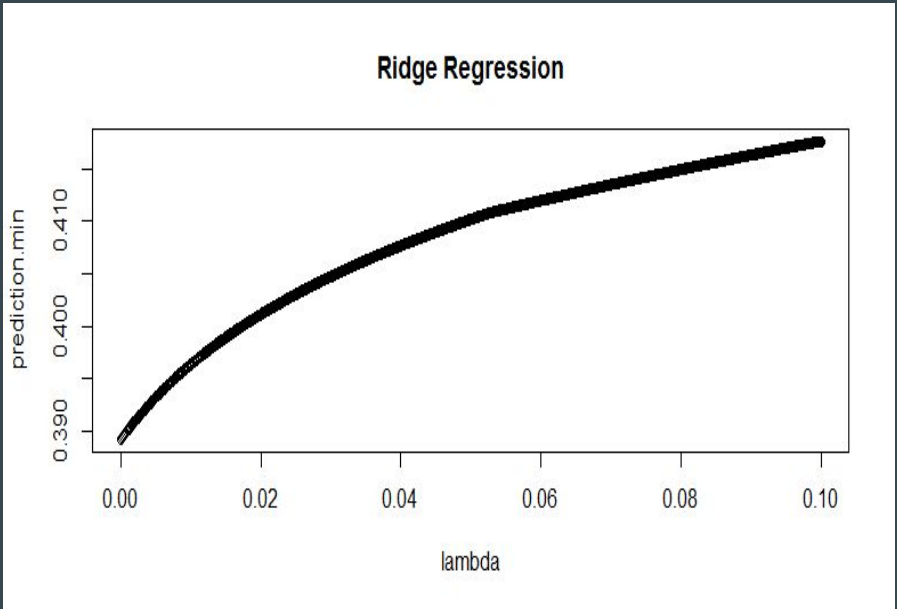


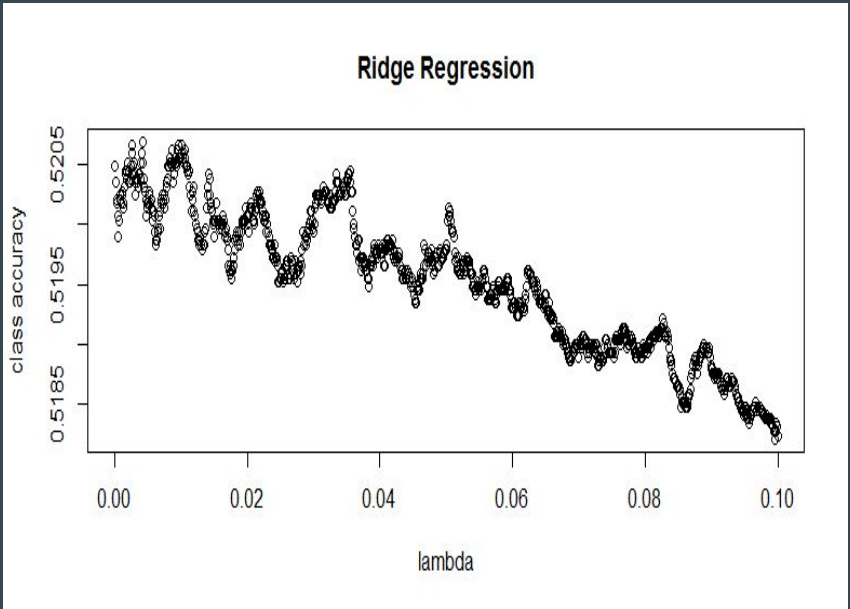


Question: why the lambda is close to 0 ?

Hypothesis: Feature is not enough, high bias, import feature expansion

Ridge Regression — Lambda by Different Cost Function

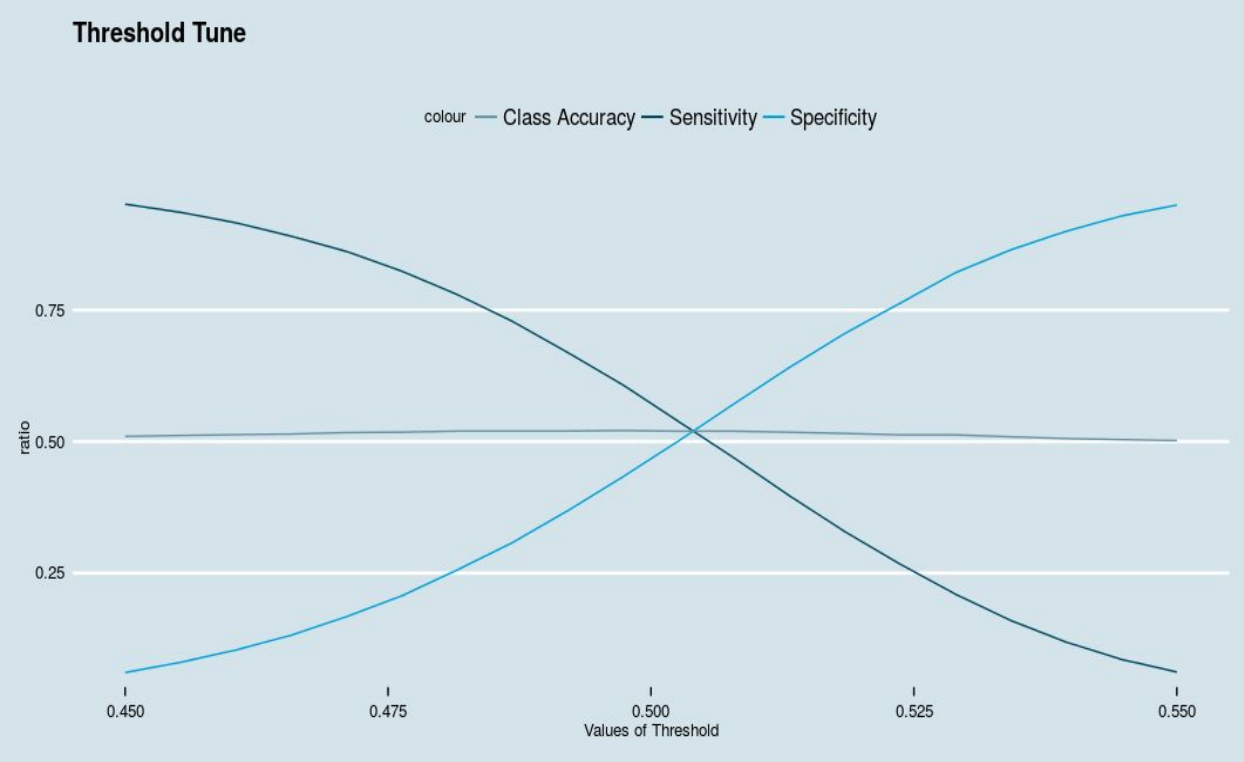




Question: why the lambda is close to 0 ?

Hypothesis: Feature is not enough, high bias, import feature expansion

Ridge Regression — Threshold Tune



Max class accuracy at

0.4975

Specificity, Sensitivity, Class Accuracy are equal at 0.5045

Features Engineering

1. Neural Network “expand features” automatically
2. Expand features from 21 to 42 by ***exp(-feature)***

Taylor expansion: $exp(-x) = 1 + (-x) + (-x)^2/2! + (-x)^3/3! + ... + (-x)^n/n! + ...$

3. Expand features from 21 to 126 with the response kept within (0,1)

$log(1+x) = x - x^2/2 + x^3/3 - x^4/4 + ...(-1)^{n-1}x^n/n + ...$

$sin(x) = x - x^3/3! + x^5/5! - ..... + (-1)^{n-1}x^{2n-1}/(2n-1)! + ...$

$cos(x) = 1 - x^2/2! + x^4/4! - ..... + (-1)^n x^{2n}/(2n)! + ...$

$tanh(x) = x + x^3/3 + x^5/5 + ... + x^{2n-1}/(2n-1) + ...$

4. Expand features from 126 to 864, multiply every two features to create cross term

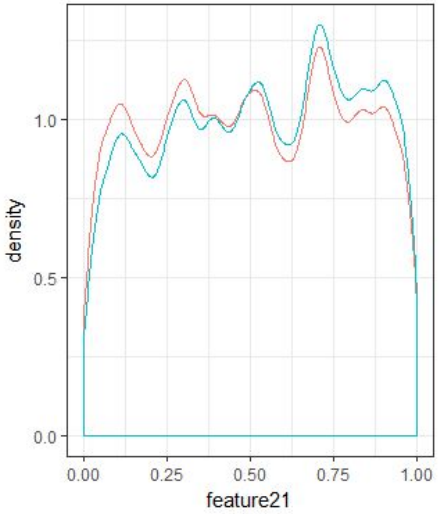
Neural Network

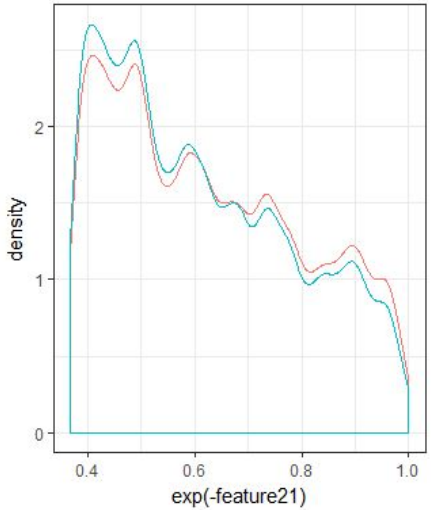
Pro:

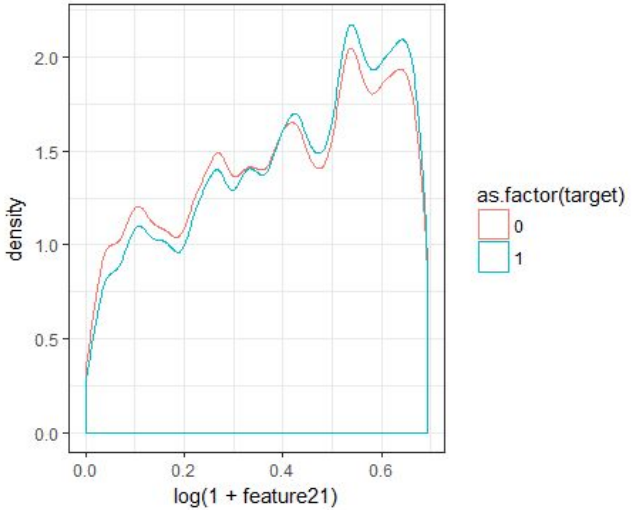
Feature expansion automatically

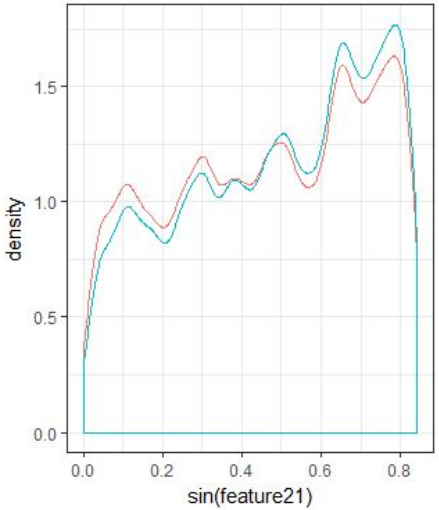
Con:

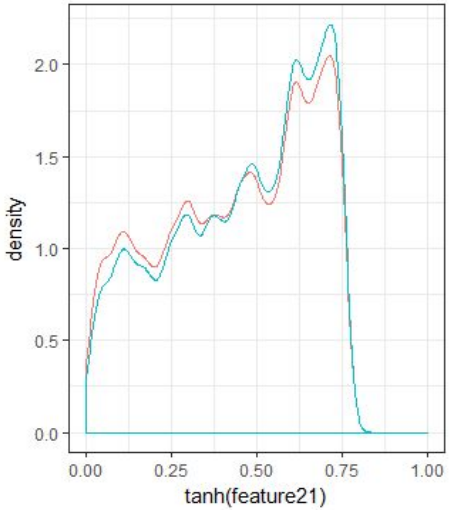
Low accuracy

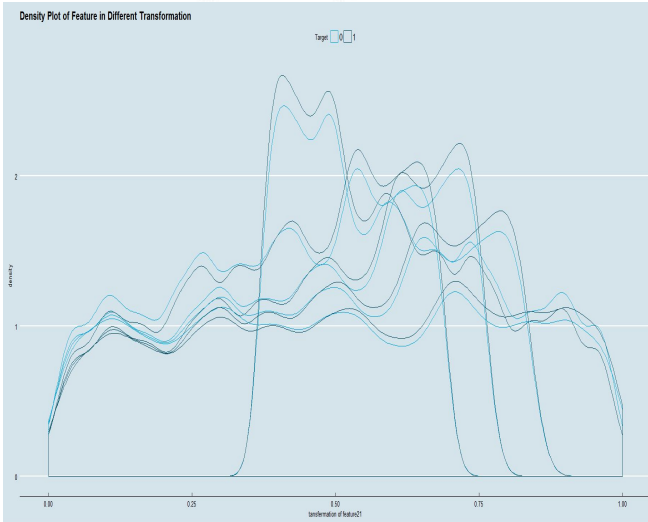

density
feature21
as.factor(target)
0
1


density
exp(-feature21)
as.factor(target)
0
1


density
log(1 + feature21)
as.factor(target)
0
1


density
sin(feature21)
as.factor(target)
0
1


density
tanh(feature21)
as.factor(target)
0
1


Density Plot of Feature in Different Transformation
Target 0 1

Result of Feature Engineering – Ridge Regression

Lambda :

0.008277857 0.01942 0.09637935 0.5305481

Class accuracy of test set:

0.5206971, 0.521893, 0.5221127, 0.5237723

Future work — Xgboost

Applying Xgboost to new data with 42, 126, 864 features:

By now the logloss of new data with cross validation:

21 features : 0.691

42 features : 0.56(incomplete,best score by now, could be overfit)

126 features : accuracy (0.72) (incomplete, best class accuracy by now)

864 features : future work (maybe not necessary and overfit)