BUDT758T
Group 17
Zizhao Liu, Yuhan Cui, Fangao Fu
5/15/2019

Executive Summary

This report is demonstrated by a professional consulting team from the University of Maryland who focuses on the healthcare industry. All data are collecting and using for the purpose of predicting patients who are readmitted or return to a hospital within 30 days of being discharged. In the hospital industry, the 30-day readmission rate is one of the most important measurements about hospitals' performances. Therefore, this information is designed to predict future performance and see whether any hospitals need to improve their services.

We made some changes and updates after previous stage of our report. First, we splitted training data into two datasets based on whether patients have ADMIT_RESULT records, and this allows us to also keep RISK and SEVERITY. Then, we optimize our data cleaning process by changing null value into Unknown which allows us to keep 95% of information in our training dataset. Finally, we combined levels from DC_RESULT and ED_RESULT based on definition, and it would eliminate errors because of level deficiency.

Since we reorganized our dataset, we explored the Hospital dataset using a number of different classification analyses compare with previous. First, we developed a boosting logistic regression model to let us get a more flexible decision boundary than other ensemble methods. Next, we use PCA to apply dimension reduction and generated a logistic regression based on that. Later, we used Stepwise approach to it gives us an objective, data-driven way to sift through data. Last, we built a random forest trees to give us more information and variability by using not often used variables. we evaluate the model by testing models on testing dataset and compare the TPR based on our purpose.

However, we are still facing some issues such as there may be some errors in the datasets we have not found and we might split the dataset in the wrong way. In data processing, we simply eliminate variables with a huge amount of null values. There might other ways to process the data, which might improve our models overall. In the modeling, we have encountered Rank Deficiency error due to lacking enough information for a certain level of a categorical variable.

Our analysis can be used as a guideline for those hospitals which want to increase their service level ,and who looking for overall improvement.

Exploratory Data Analysis

Before we start modeling, it's important for us to understand the dataset, and to explore the relationship between variables, which is an essential step for us to choose which variables to use in order to have an optimal result. There are 26 predictive variables and one response variables in this dataset (Appendix A: data type).

In our previous analysis, we eliminate INDEX, departure time, RISK, SEVERITY, ADMIT_RESULT, ETHNICITY, CONSULT_CHARGE and CONSULT_IN_ED. However, there is a mistake we made in our previous report which is that we didn't realize some variables' types are wrong in our dataset. In the current stage, we start our progress with transferring those variables from numeric to categorical and make some updates in the data processing for better use of the structure of the data.

First, we keep the decision of not using INDEX in our modeling process since is a unique identifier to distinguish each patient. However, we will keep the variable in our test dataset in order to match each patient with their predicted returned sort patients into relevant order for prediction purpose.

The reason that we need INDEX as identifier is that we split training data into two datasets based on whether patients have ADMIT_RESULT records. We find most patients who have admit result record definitely have records of risk and severity. By splitting the dataset, we are able to keep the information of RISK and SEVERITY we previously eliminated. Both dataset eliminate departure time and ETHNICITY. For the dataset that has admit records will also include risk and severity, and we named it as df_ip. However, for the rest of data, we just eliminate the ADMIT, RISK and SEVERITY since patients who are not admitted by the hospital will not have evaluation of sickness at all, and we named the dataset as df_op.

As we go through the datasets, we find that CONSULT_ORDER, CONSULT_CHARGE, and CONSULT_ IN_ED which we eliminated before since high proportion of zero might become useful after splitting. All three of them have relatively small proportion of one, but 98.24% of CONSULT_ORDER, 85.27% of CONSULT_CHARGE, and 95.11% CONSULT_ IN_ED of them falling into df_ip dataset.

In our previous analysis, it was quite a challenge using DC_RESULT in our model since it has 36 levels. We had to try several seeds to get a good partition in order to avoid validation data being inherently different from training data. We believe there is a more appropriate way for us to use this information. After we go through all 36 levels, we find that we can combine them based on definition similarities. Eventually, we transfer 36 levels into 5 levels after combination process, which are 'further treatment'. 'no treatment', 'discharge', 'left' and 'other'. Similarly, we combine 14 of 16 levels of ED_RESULT into 5 levels: 'further treatment', 'leaving without completing treatment', ‘left without permission’, ‘L&D’, ‘Discharged’. We keep ‘deceased’, and ‘observation’ as the original because each of them carries unique information; ‘deceased’ indicates patient would never return to the hospital whereas ‘observation’ is an uncertain instruction, which cannot be placed with other levels. However, the downside of this combination is that none of us have strong domain knowledge in the hospital industry, and there is possibility that our combination is not ideal.

We have to strictly repeated the update process we described above for the testing data to maintain the consistency, and it causes issues.There are many null values in the testing data, and we cannot process it the same way as how we processed the training data because we have to make prediction for each patient in the testing data rather than deleting the patient's record. Therefore, we set all null value as ‘Unknown’, a variable that does not contain any values. We process the training data in the same way which allow us to keep more than 95% of the information. Eventually, we remove 144 ‘#NA’ records in our dependent variable, RETURN.

Modeling and Evaluation

In our previous analysis, we tried three models: logistic regression, classification tree, LDA, and kNN. In our current stage, since we apply several changes to our data cleaning process and split our dataset based on ADMIT_RESULT, it’s not appropriate to use those four models any longer. Unlike our previous analysis, we do not need to use simple partition or cross-validation this time. We will use testing data for evaluation. Therefore, the more training data we have the more information we can capture and use in our models.

When we start modeling, we consider the purpose of building models. Our research purpose is predicting patients who are readmitted or return to a hospital within 30 days of being discharged. Since we are more concern about patients who return to the hospital which only accounts for small portion of the data, we decide to start with a boosting logistic regression.

- **XGBoosting Logistic Regression**

Logistic model is able to predict the probability of a patient will return to a hospital, and boosting method allows the model to reweight every data point and assign higher weight to misclassified points, which let us get a more flexible decision boundary than other ensemble methods.

We try to run two different XGboosting logistic regressions for each dataset to predict RETURN using all other variables we choose from the last stage. Unlike our previous models such as kNN, we do not need to apply cross-validation to choose the parameter that gives us the highest validation accuracy. As you can see in both of our relatively influence graphs, CHARGES are the most recently used variables to predict the hard predicted points and other variables importances are relatively agree with each others. This is not what we expected when we split the data into two datasets since we were hoping that two datasets would have different results which means variables ADMIT_RESULT, RISK and SEVERITY don't give us additional power to predict 1 value of RETURN. Therefore, if we run the whole dataset together, our result may not have significant change. (Appendix A: XGBoost Variable Importance Plot)

- **Stepwise**

Next, since we have a relatively large dataset, we decide to use stepwise variable selection to build logistic regressions and generate different logistic model for each data set, df_ip and df_op. Stepwise is one of the variable selection procedure, and the reason we use variable selection is that it gives us an objective, data-driven way to sift through data, and it allows us to have the advantage that won't miss anything. Moreover, stepwise can solve omitted variable bias: we leave variable out of the model that should have been include.

Overall, stepwise gives us the best testing accuracy for both inpatient and outpatient dataset among all other models we tried. We also tried to use different cut-off from 0.2 to x in order to get the best result, and 0.5 is our final choice. (Appendix B: Summary on Stepwise Model)

- **Principal Component Analysis**

PCA is a good technique for dimension reduction that works well. Since we have 26 variables in our dataset, we think it may be a good option for our analysis. The main advantage of PCA compared with other dimension reduction procedures is that it will not get rid of any variables but, instead, capture as much information as there in the dataset.

We have applied PCA procedure on both df_ip and df_op, almost 100% of information has been put into the first principal component whereas the rest of principal components contain nearly nothing, which rarely happens.Moreover, PCA is an unsupervised approach, and it can not guarantee to give us a predictions. It indeed captures as much information as it can, but when we try to use it on our testing dataset to predict, it doesn't work. We didn't find the reason for that, but we assume that might because of inconsistency between training data and testing data or principal component which R generates do not contain the same information. We evaluate our PCA result by comparing with simple regression and stepwise regression, and the performance of PCA is not as good as both of them. (Appendix C: PCA Variance Plot)

- **Random Forest**

However, in our previous analysis the classification tree didn’t give us the ideal result-accuracy was the same as the baseline, we want to retry once with ensemble method, random forest with 500 trees.

Tree model measures impurity to split the data, so the model will always choose the “best” split which has the lowest total impurity score. Random forest will force trees to use variables that are not used very often in single trees, which gives more information and variability. It might give us a better result and more prediction power.

Given the predictor importance graphs, the most important variable in terms of predicting power are AGE, CHARGES, and FINANCIAL CLASS on df_ip whereas on df_op, variables that have strong predicting power are FINANCIAL CLASS, AGE, and GENDER. As far as inference power, HOUR_ARR appears to be the most important variable in building the trees on both datasets. Even though, two sets of predictor importance graphs do have similarities, they do not agree with each other since orders of variable importance are different as well, which totally makes sense because the two datasets have different characteristics. (Appendix D: RandomForest variable importance plot)

- **Evaluation and Conclusion**

As we stated before, we are mainly focusing on ‘1’, which means patients who will return to the hospital within 30 days after discharged. Therefore, besides over accuracy, True Positive Rate should also be considered.

We choose our model based on testing data accuracy. Other than that, We tried different cutoff values to see which value can give us highest testing accuracy. Results are shown in the table down below. The one with highest accuracy is RandomForest, which is 82.60% and 76.83% with cutoff values of 0.5 for both inpatient and outpatient datasets. Other than that, we were thinking about use different models for inpatient and outpatient dataset which ever give us the best result, but in this case, RandomForest is the best for both dataset. One problem with this procedure is that we do not know what cutoff values are the most appropriate in this case, which means domain knowledge might be needed here.

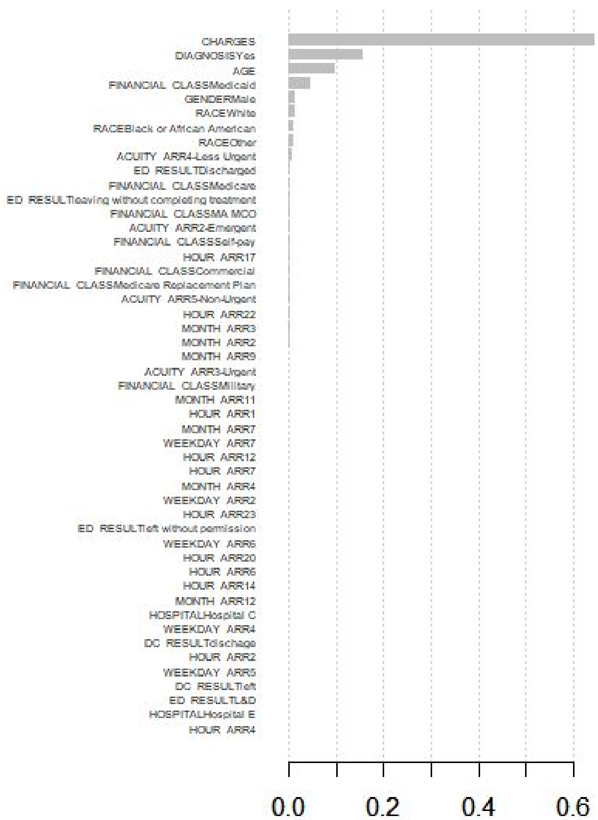
As we mentioned before, True Positive Rate is also another measurement we considered. XGBoost has the highest TPR for both of inpatient and outpatient testing data. Based on this, we might use this model for certain circumstance, but since its accuracy is below baseline accuracy, we will still choose RandomForest.(Appendix E: Test Accuracy and TPR on Models)

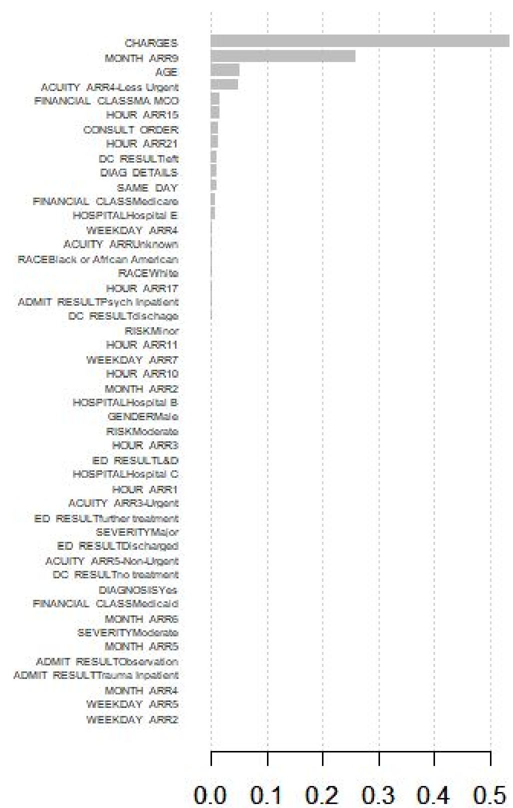
- **Recommendation**

The purpose of our analysis should be focus on 1 of the return, Therefore, this information is designed to predict future performance and see whether any hospitals need to improve their services. However, maybe we didn't choose the best model or find the pattern of those data, there's a trade-off we have to make between accuracy and TPR.

However, hospitals can still use our random forest model which provide a relatively high true positive rate with outpatient dataset, which means it performs better with patients who don't have admit result. Since this dataset is much larger than the other one, we would say that it can give the hospital a good insight of their service. For example is there anything unusual happened in the emergency room to cause patients transfer to another facility, or since many patients have been sent to police station after therapy, is location the main reason that cause high return rate. Those are questions can be thought about and evaluate in the future.

Appendix A





Appendix B

```
> summary(op_both)

Call:
glm(formula = RETURN ~ ED_RESULT + FINANCIAL_CLASS + GENDER +
    HOUR_ARR + RACE + MONTH_ARR + ACUITY_ARR + AGE +
WEEKDAY_ARR +
    DIAGNOSIS + DC_RESULT + CHARGES, family = "binomial", data =
op_nonull[,
    -1])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8990  -0.7951  -0.5889   0.8926   3.3746

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -1.594e+01  7.709e+01  -0.207 0.836170
ED_RESULTObservation          1.256e+01  7.709e+01   0.163
0.870548
ED_RESULTUnknown              1.252e+01  7.709e+01   0.162
0.871008
ED_RESULTfurther treatment    1.233e+01  7.709e+01   0.160
0.872921
ED_RESULTleaving without completing treatment  1.312e+01  7.709e+01
0.170 0.864826
ED_RESULTleft without permission    1.345e+01  7.709e+01   0.174
0.861492
ED_RESULTL&D                  1.266e+01  7.709e+01   0.164 0.869519
ED_RESULTDischarged           1.272e+01  7.709e+01   0.165
0.868886
FINANCIAL_CLASSCommercial    -5.165e-03  1.026e-01  -0.050
0.959867
FINANCIAL_CLASSGlobal Contracts    4.279e-01  1.132e+00   0.378
0.705529
FINANCIAL_CLASSMA MCO         1.135e+00  7.867e-02  14.423 <
2e-16 ***
FINANCIAL_CLASSMedicaid       1.321e+00  9.253e-02  14.273 <
2e-16 ***
FINANCIAL_CLASSMedicaid Pending    2.161e+00  8.482e-01   2.547
0.010854 *
```

```
FINANCIAL_CLASSMedicare                       1.130e+00  8.499e-02  13.296  <
2e-16 ***
FINANCIAL_CLASSMedicare Replacement Plan       7.632e-01  1.177e-01
6.485 8.86e-11 ***
FINANCIAL_CLASSMilitary                  1.153e+00  1.807e-01   6.379
1.79e-10 ***
FINANCIAL_CLASSOther                    -5.563e-01  3.466e-01  -1.605
0.108478
FINANCIAL_CLASSOut of State Medicaid         -2.389e-01  4.819e-01  -0.496
0.620009
FINANCIAL_CLASSSelf-pay                   1.706e-01  9.388e-02   1.817
0.069219 .
FINANCIAL_CLASSWorker's Comp               -1.480e-01  2.927e-01  -0.506
0.613075
GENDERMale                             6.372e-01  2.925e-02  21.787  < 2e-16 ***
HOUR_ARR1                             -8.649e-03  9.352e-02  -0.092 0.926315
HOUR_ARR2                              9.959e-02  9.514e-02   1.047 0.295212
HOUR_ARR3                              1.860e-01  9.483e-02   1.961 0.049853 *
HOUR_ARR4                              1.408e-01  9.303e-02   1.513 0.130249
HOUR_ARR5                              3.553e-01  9.198e-02   3.862 0.000112 ***
HOUR_ARR6                              4.090e-01  8.962e-02   4.564 5.02e-06 ***
HOUR_ARR7                              4.544e-01  9.650e-02   4.709 2.49e-06 ***
HOUR_ARR8                              2.114e-01  9.951e-02   2.124 0.033677 *
HOUR_ARR9                              2.586e-01  9.866e-02   2.621 0.008767 **
HOUR_ARR10                            -1.601e-02  9.737e-02  -0.164 0.869386
HOUR_ARR11                             2.384e-02  9.488e-02   0.251 0.801633
HOUR_ARR12                             1.379e-02  9.446e-02   0.146 0.883947
HOUR_ARR13                            -1.767e-01  9.818e-02  -1.800 0.071827 .
HOUR_ARR14                            -1.865e-01  9.543e-02  -1.954 0.050667 .
HOUR_ARR15                            -1.625e-01  9.306e-02  -1.746 0.080844 .
HOUR_ARR16                            -2.334e-01  9.466e-02  -2.466 0.013664 *
HOUR_ARR17                            -1.562e-01  9.213e-02  -1.696 0.089907 .
HOUR_ARR18                            -1.288e-01  9.123e-02  -1.411 0.158102
HOUR_ARR19                            -1.847e-01  9.497e-02  -1.945 0.051788 .
HOUR_ARR20                            -1.707e-01  9.653e-02  -1.768 0.077076 .
HOUR_ARR21                            -1.729e-01  9.496e-02  -1.821 0.068672 .
HOUR_ARR22                            -1.676e-01  9.562e-02  -1.753 0.079574 .
HOUR_ARR23                            -1.261e-01  9.638e-02  -1.308 0.190794
RACEAsian                          -2.250e-01  5.778e-01  -0.389 0.696940
RACEBlack or African American               6.346e-01  5.061e-01   1.254
0.209924
RACEDeclined to Answer                   -1.127e+01  3.477e+02  -0.032
0.974133
RACEHispanic                          -1.167e+01  2.059e+02  -0.057 0.954780
RACENative Hawaiian or Other Pacific Islander  3.225e-01  1.195e+00   0.270
0.787288
RACEOther                             3.946e-01  5.190e-01   0.760 0.447145
RACETwo or More Races                    2.070e+00  1.012e+00   2.046
0.040797 *
RACEUnknown                           -2.193e+00  6.362e-01  -3.446 0.000568
***
RACEWhite                             5.205e-01  5.070e-01   1.027 0.304645
MONTH_ARR2                             1.408e-01  6.432e-02   2.190 0.028536 *
MONTH_ARR3                             4.069e-02  6.324e-02   0.643 0.519945
MONTH_ARR4                             1.300e-02  6.355e-02   0.205 0.837940
MONTH_ARR5                             3.905e-02  6.252e-02   0.625 0.532264
MONTH_ARR6                            -4.764e-02  6.422e-02  -0.742 0.458209
MONTH_ARR7                             1.531e-01  6.248e-02   2.450 0.014282 *
MONTH_ARR8                             1.995e-02  6.410e-02   0.311 0.755578
MONTH_ARR9                            -4.991e-01  6.930e-02  -7.202 5.95e-13 ***
MONTH_ARR11                            2.837e-02  6.531e-02   0.434 0.663967
MONTH_ARR12                            5.827e-02  6.427e-02   0.907 0.364591
ACUITY_ARR2-Emergent                   -3.797e-01  4.282e-01  -0.887
0.375173
ACUITY_ARR3-Urgent                     -3.193e-01  4.238e-01  -0.753 0.451230
ACUITY_ARR4-Less Urgent                -2.689e-01  4.242e-01  -0.634
0.526240
ACUITY_ARR5-Non-Urgent                  3.143e-02  4.269e-01   0.074
0.941309
ACUITY_ARRUnknown                       2.779e-02  4.319e-01   0.064
0.948695
AGE                               7.183e-03  1.035e-03   6.938 3.98e-12 ***
WEEKDAY_ARR2                          -1.489e-01  5.178e-02  -2.876 0.004021
**
WEEKDAY_ARR3                          -2.808e-01  5.275e-02  -5.323 1.02e-07
***
WEEKDAY_ARR4                          -1.881e-01  5.224e-02  -3.600 0.000318
***
WEEKDAY_ARR5                          -1.422e-01  5.235e-02  -2.715 0.006622
**
WEEKDAY_ARR6                          -1.848e-01  5.255e-02  -3.517 0.000436
***
WEEKDAY_ARR7                          -1.163e-01  5.288e-02  -2.200 0.027809
*
DIAGNOSISYes                          -2.719e-01  6.705e-02  -4.056 4.99e-05 ***
DC_RESULTleft                          5.138e-01  1.873e-01   2.744 0.006076 **
DC_RESULTother                         6.320e-01  2.041e-01   3.096 0.001961 **
DC_RESULTdischage                      4.427e-01  1.760e-01   2.516 0.011872
*
DC_RESULTno treatment                  -1.545e-01  3.513e-01  -0.440
0.660136
CHARGES                               1.014e-05  4.104e-06   2.472 0.013453 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 35308  on 30606  degrees of freedom
Residual deviance: 31596  on 30526  degrees of freedom
AIC: 31758

Number of Fisher Scoring iterations: 12
```

```
> summary(ip_both)

Call:
glm(formula = RETURN ~ DC_RESULT + FINANCIAL_CLASS + AGE +
ED_RESULT +
    MONTH_ARR + CHARGES + RACE + SAME_DAY + GENDER +
ADMIT_RESULT +
    ACUITY_ARR + DIAGNOSIS + CONSULT_CHARGE, family = "binomial",
    data = ip_nonull[, -1])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3550  -0.6661  -0.5204  -0.3269   2.9605

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                          -1.222e+01  1.028e+03  -0.012 0.990514
DC_RESULTleft                         1.226e+00  2.067e-01   5.932 2.99e-09 ***
DC_RESULTother                       -2.455e+00  1.013e+00  -2.425 0.015302
*
DC_RESULTdischage                     5.269e-01  1.219e-01   4.323 1.54e-05
***
DC_RESULTno treatment                 5.584e-01  2.986e-01   1.870
0.061481 .
FINANCIAL_CLASSCommercial              -6.597e-02  1.606e-01  -0.411
0.681310
FINANCIAL_CLASSGlobal Contracts          -5.248e-01  7.499e-01  -0.700
0.484091
FINANCIAL_CLASSMA MCO                   5.032e-01  1.259e-01   3.998
6.40e-05 ***
FINANCIAL_CLASSMedicaid                 2.087e-01  1.723e-01   1.211
0.225757
FINANCIAL_CLASSMedicaid Pending            6.491e-01  1.171e+00   0.554
0.579390
FINANCIAL_CLASSMedicare                 5.985e-01  1.297e-01   4.616
3.92e-06 ***
```

FINANCIAL_CLASSMedicare Replacement Plan 5.756e-01 1.835e-01
3.137 0.001707 **
FINANCIAL_CLASSMilitary -3.254e-01 4.895e-01 -0.665
0.506146
FINANCIAL_CLASSOther -1.583e+00 1.048e+00 -1.510
0.131114
FINANCIAL_CLASSOut of State Medicaid 7.475e-01 8.154e-01 0.917
0.359307
FINANCIAL_CLASSSelf-pay -1.226e+00 4.401e-01 -2.786
0.005342 **
FINANCIAL_CLASSWorker's Comp -1.372e+01 3.255e+02 -0.042
0.966364
AGE -2.112e-02 2.335e-03 -9.045 < 2e-16 ***
ED_RESULTObservation 1.086e+01 1.028e+03 0.011
0.991569
ED_RESULTUnknown -3.379e+00 1.092e+03 -0.003
0.997531
ED_RESULTfurther treatment 1.091e+01 1.028e+03 0.011
0.991534
ED_RESULTleaving without completing treatment 1.018e+01 1.028e+03
0.010 0.992101
ED_RESULTleft without permission 1.148e+01 1.028e+03 0.011
0.991087
ED_RESULTL&D 9.665e+00 1.028e+03 0.009 0.992497
ED_RESULTDischarged 1.063e+01 1.028e+03 0.010
0.991747
MONTH_ARR2 2.152e-02 1.550e-01 0.139 0.889562
MONTH_ARR3 2.216e-01 1.503e-01 1.474 0.140347
MONTH_ARR4 2.604e-01 1.470e-01 1.772 0.076459 .
MONTH_ARR5 3.145e-01 1.453e-01 2.165 0.030404 *
MONTH_ARR6 3.542e-01 1.449e-01 2.445 0.014479 *
MONTH_ARR7 1.165e-01 1.511e-01 0.771 0.440469
MONTH_ARR8 7.781e-02 1.500e-01 0.519 0.603857
MONTH_ARR9 -6.573e-01 1.723e-01 -3.814 0.000137
***
MONTH_ARR11 5.234e-02 1.529e-01 0.342 0.732162
MONTH_ARR12 3.288e-01 1.483e-01 2.217 0.026652 *
CHARGES -4.996e-06 1.276e-06 -3.915 9.04e-05 ***
RACEAsian -1.161e+00 5.578e-01 -2.081 0.037442 *
RACEBlack or African American -7.501e-01 4.419e-01 -1.697
0.089619 .
RACEDeclined to Answer -1.465e+01 1.455e+03 -0.010
0.991971
RACEHispanic -1.376e+01 1.455e+03 -0.009 0.992455
RACENative Hawaiian or Other Pacific Islander 9.051e-02 9.562e-01 0.095
0.924591
RACEOther -1.426e+00 5.051e-01 -2.823 0.004764 **
RACETwo or More Races -1.516e+01 1.455e+03 -0.010
0.991690
RACEUnknown -1.910e+00 1.129e+00 -1.691 0.090742 .
RACEWhite -1.008e+00 4.442e-01 -2.270 0.023222 *
SAME_DAY -1.978e-01 7.211e-02 -2.743 0.006085 **
GENDERMale 1.694e-01 6.435e-02 2.632 0.008477 **
ADMIT_RESULTObservation 5.513e-02 1.012e-01 0.545
0.586027
ADMIT_RESULTPsych Inpatient -1.333e+00 3.808e-01 -3.501
0.000464 ***
ADMIT_RESULTTrauma Inpatient 8.255e-02 2.119e-01 0.390
0.696848
ACUITY_ARR2-Emergent 7.678e-01 3.806e-01 2.017
0.043668 *
ACUITY_ARR3-Urgent 8.302e-01 3.788e-01 2.192 0.028380
*
ACUITY_ARR4-Less Urgent 1.155e+00 4.160e-01 2.775
0.005512 **
ACUITY_ARR5-Non-Urgent 2.541e+00 7.152e-01 3.552
0.000382 ***
ACUITY_ARRUnknown 7.764e-01 4.852e-01 1.600
0.109610
DIAGNOSISYes 1.715e-01 6.717e-02 2.553 0.010669 *
CONSULT_CHARGE -5.362e-01 2.408e-01 -2.227
0.025962 *

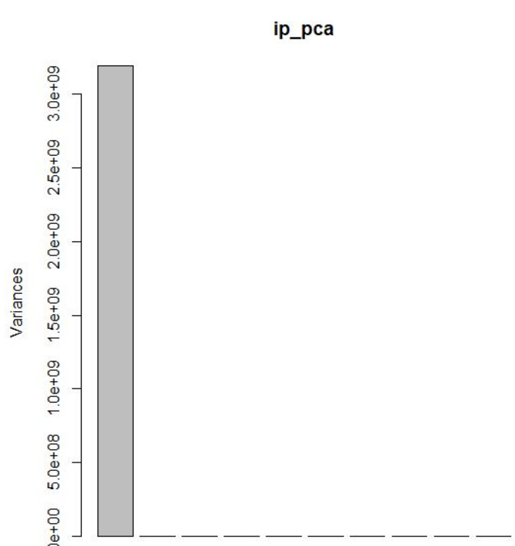
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

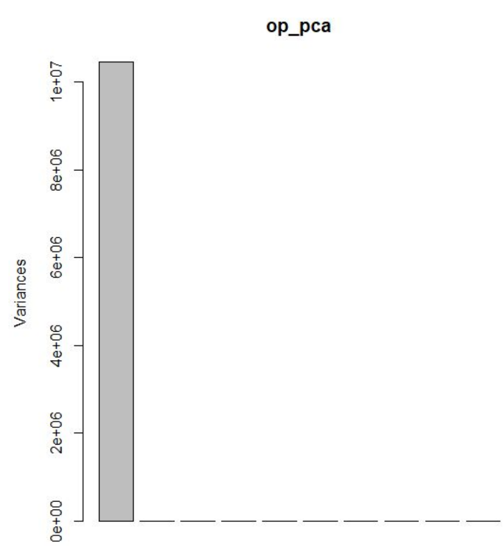
(Dispersion parameter for binomial family taken to be 1)

Null deviance: 7006.1 on 7610 degrees of freedom
Residual deviance: 6489.9 on 7554 degrees of freedom
AIC: 6603.9

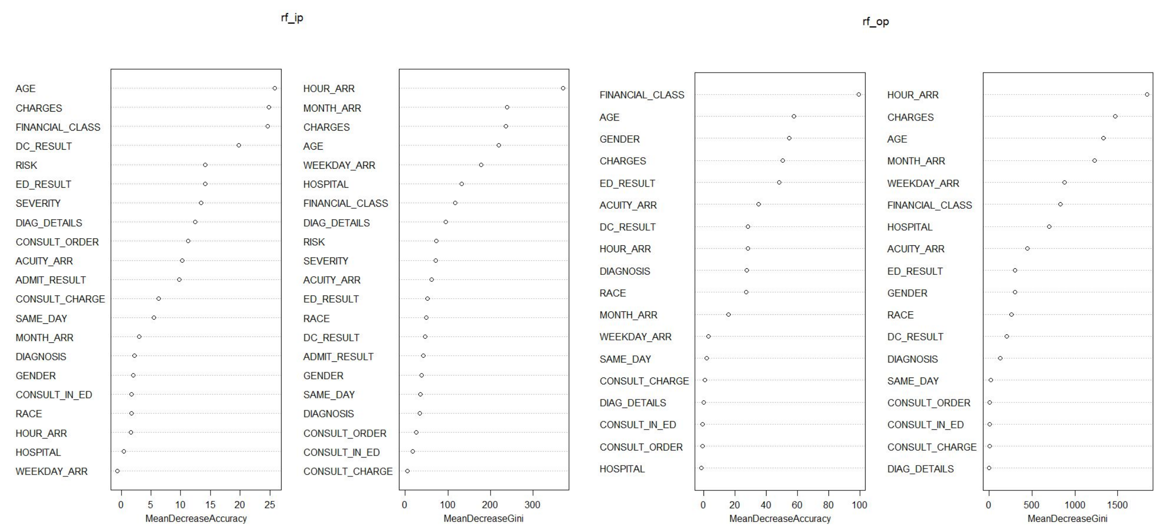
Number of Fisher Scoring iterations: 14

Appendix C





Appendix D



Appendix E

		XGBoost	Stepwise	RandomForest
Highest accuracy	**Inpatient**	**75.33%**	**82.40%**	**82.60%**
	Outpatient	**68.31%**	**75.70%**	**76.83%**
Corresponding TPR	**Inpatient**	**11.03%**	**0.69%**	**10.56%**
	Outpatient	**38.66%**	**10.56%**	**30.09%**
cutoff	**Inpatient**	**0.4**	**0.5**	**0.5**
	Outpatient	**0.5**	**0.45**	**0.5**

R Code:

```
#import the training data set
library(readr)
df<- read_csv("D:/BUDT758T/Project Files/Hospitals_Train.csv")

#fillin the null in certain columns
df$ACUITY_ARR[is.na(df$ACUITY_ARR)]<-'Unknown'
df$ED_RESULT[is.na(df$ED_RESULT)]<-'Unknown'
df$RACE[is.na(df$RACE)]<-'Unknown'
df$RISK[is.na(df$RISK)]<-'Unknown'
df$SEVERITY[is.na(df$SEVERITY)]<-'Unknown'

#change the type of some variable
df$RETURN=ifelse(df$RETURN=='Yes',1,0)
df$MONTH_ARR=as.factor(df$MONTH_ARR)
df$WEEKDAY_ARR=as.factor(df$WEEKDAY_ARR)
df$HOUR_ARR=as.factor(df$HOUR_ARR)
df$CHARGES=as.numeric(as.character(df$CHARGES))
df$CHARGES[is.na(df$CHARGES)]<-mean(as.numeric(df[!is.na(df$CHARGES),][['CHARGES']]))
df$DC_RESULT=as.factor(df$DC_RESULT)
df$ACUITY_ARR=as.factor(df$ACUITY_ARR)
df$ED_RESULT=as.factor(df$ED_RESULT)
df$DIAGNOSIS=as.factor(df$DIAGNOSIS)
df$FINANCIAL_CLASS=as.factor(df$FINANCIAL_CLASS)
df$RACE=as.factor(df$RACE)
df$HOSPITAL=as.factor(df$HOSPITAL)
df$GENDER=as.factor(df$GENDER)
df$ADMIT_RESULT=as.factor(df$ADMIT_RESULT)
df$RISK=as.factor(df$RISK)
df$SEVERITY=as.factor(df$SEVERITY)
df$CONSULT_IN_ED[is.na(df$CONSULT_IN_ED)]<-0
df$DIAG_DETAILS=as.numeric(df$DIAG_DETAILS)

sapply(df, function(x) sum(is.na(x)))


#check the structure the df
str(df)
library(rockchalk)
#reduce the levels in ED_RESULTS and DC_RESULTS
df$DC_RESULT = combineLevels(df$DC_RESULT,levs=c(2,4,5,7,8,9,10,12,16,28,30,31,32,33,34,35,36),newLabel = c("further treatment"))
df$DC_RESULT = combineLevels(df$DC_RESULT,levs = c(2,14,15,16,17,18),newLabel = c("left"))
df$DC_RESULT = combineLevels(df$DC_RESULT,levs = c(7,13),newLabel = c("other"))
df$DC_RESULT = combineLevels(df$DC_RESULT,levs = c(1,7,8,9,10),newLabel = ("dischage"))
df$DC_RESULT = combineLevels(df$DC_RESULT,levs = c(1,2,3,4,5,6),newLabel = c("no treatment"))
#reduce the levels in ED_RESULTS and DC_RESULTS
df$ED_RESULT=combineLevels(df$ED_RESULT,levs=c(1,2,3,16),newLabel = c("further treatment"))
df$ED_RESULT=combineLevels(df$ED_RESULT,levs=c(1,6,7,8),newLabel = c("leaving without completing treatment"))
df$ED_RESULT=combineLevels(df$ED_RESULT,levs=c(4,5),newLabel = c("left without permission"))
df$ED_RESULT=combineLevels(df$ED_RESULT,levs=c(6,5),newLabel = c("L&D"))
df$ED_RESULT=combineLevels(df$ED_RESULT,levs=c(1,3),newLabel = c("Discharged"))
```

```
#combine the
df$ACUITY_ARR=combineLevels(df$ACUITY_ARR,levs=c(5,6),newLabel = c("5-Non-Urgent"))
df$ACUITY_ARR=combineLevels(df$ACUITY_ARR,levs=c(5),newLabel = c("Unknown"))
levels(df$DC_RESULT)
levels(df$ED_RESULT)
#seperate the data set by the ADMIT_RESULTS
df_ip <- df[!is.na(df$ADMIT_RESULT),]
df_op <-df[is.na(df$ADMIT_RESULT),]
#check the existence null values in the df_ip and df_op
sapply(df_ip, function(x) sum(is.na(x)))
sapply(df_op, function(x) sum(is.na(x)))
#elininated "ADMIT_RESULT", "RISK", "SEVERITY", "WEEKDAY_DEP", "HOUR_DEP", "MONTH_DEP", "ETHNICITY" from df_op
df_ip<-subset(df_ip,select = -c(WEEKDAY_DEP, HOUR_DEP, MONTH_DEP, ETHNICITY))
#elininated "ADMIT_RESULT", "RISK", "SEVERITY", "WEEKDAY_DEP", "HOUR_DEP", "MONTH_DEP", "ETHNICITY" from df_op
df_op<-subset(df_op,select = -c(WEEKDAY_DEP, HOUR_DEP, MONTH_DEP, ETHNICITY,ADMIT_RESULT, RISK, SEVERITY))
#check the structure of df_ip and df_op
str(df_ip)
str(df_op)
#create functions for train and validation set splitting
#seeds=908765
#perc=0.8
#gen_train<-function(df_whole){
#  set.seed(seeds)
#  ip_num=nrow(df_whole)
#  obs <-sample(ip_num,perc*ip_num)
#  trainset_alias <-df_whole[obs,]
#  return(trainset_alias)
#}

#gen_val<-function(df_whole){
#  set.seed(seeds)
#  ip_num=nrow(df_whole)
#  obs <-sample(ip_num,perc*ip_num)
#  valset_alias <-df_whole[-obs,]
#  return(valset_alias)
#}

#Remove all the nulls in the df_ip and df_op
ip_nonull <- na.omit(df_ip)
op_nonull <- na.omit(df_op)

###After having the testing data, we no longer need the validation data set!
#generate the train and validation sets in df_ip
#ip_nonull_train <-gen_train(ip_nonull)
#ip_nonull_val<-gen_val(ip_nonull)
#generate the train and validation sets in df_ip
#op_nonull_train <-gen_train(op_nonull)
#op_nonull_val<-gen_val(op_nonull)
#generate a sparse matrix on ip_nonull and op_nonull
library(Matrix)
ip_sparse <- sparse.model.matrix(RETURN ~ ., data = ip_nonull[,-1])
op_sparse <- sparse.model.matrix(RETURN ~ ., data = op_nonull[,-1])
#ip_sparse_train <- sparse.model.matrix(RETURN ~ ., data = ip_nonull_train[,-1])
#op_sparse_train <- sparse.model.matrix(RETURN ~ ., data = op_nonull_train[,-1])
#ip_sparse_val <- sparse.model.matrix(RETURN ~ ., data = ip_nonull_val[,-1])
#op_sparse_val <- sparse.model.matrix(RETURN ~ ., data = op_nonull_val[,-1])
```

```
###deal with the testing data set
test <- read_csv("D:/BUDT758T/Project Files/Hospitals_Test.csv")
#change the type of some variable

test$RETURN=ifelse(test$RETURN=='Yes',1,0)
test$MONTH_ARR=as.factor(test$MONTH_ARR)
test$WEEKDAY_ARR=as.factor(test$WEEKDAY_ARR)
test$HOUR_ARR=as.factor(test$HOUR_ARR)
test$CHARGES=as.numeric(as.character(test$CHARGES))
test$CHARGES[is.na(test$CHARGES)]<-mean(as.numeric(test[!is.na(test$CHARGES),][['CHARGES']]))
test$DC_RESULT=as.factor(test$DC_RESULT)
test$ACUITY_ARR[is.na(test$ACUITY_ARR)]<-'Unknown'
test$ACUITY_ARR=as.factor(test$ACUITY_ARR)
test$ED_RESULT[is.na(test$ED_RESULT)]<-'Unknown'
test$ED_RESULT=as.factor(test$ED_RESULT)
test$DIAGNOSIS=as.factor(test$DIAGNOSIS)
test$FINANCIAL_CLASS=as.factor(test$FINANCIAL_CLASS)
test$RACE[is.na(test$RACE)]<-'Unknown'
test$RACE=as.factor(test$RACE)
test$HOSPITAL=as.factor(test$HOSPITAL)
test$GENDER=as.factor(test$GENDER)
test$ADMIT_RESULT=as.factor(test$ADMIT_RESULT)
test$RISK[is.na(test$RISK)]<-'Unknown'
test$SEVERITY[is.na(test$SEVERITY)]<-'Unknown'
test$RISK=as.factor(test$RISK)
test$SEVERITY=as.factor(test$SEVERITY)
test$CONSULT_IN_ED[is.na(test$CONSULT_IN_ED)]<-0
test$DIAG_DETAILS=as.numeric(test$DIAG_DETAILS)

sapply(test, function(x) sum(is.na(x)))

#check the structure the test
str(test)
library(rockchalk)
#reduce the levels in ED_RESULTS and DC_RESULTS
test$DC_RESULT = combineLevels(test$DC_RESULT,levs=c(1,3,4,5,6,7,9,10,11,12,27,28,29,30,31,32,33),newLabel = c("further
treatment"))
test$DC_RESULT = combineLevels(test$DC_RESULT,levs = c(1,11,12,13,14,15),newLabel = c("left"))
test$DC_RESULT = combineLevels(test$DC_RESULT,levs = c(5,10),newLabel = c("other"))
test$DC_RESULT = combineLevels(test$DC_RESULT,levs = c(5,6,7,8),newLabel = ("dischage"))
test$DC_RESULT = combineLevels(test$DC_RESULT,levs = c(1,2,3,4),newLabel = c("no treatment"))
#reduce the levels in ED_RESULTS and DC_RESULTS
test$ED_RESULT=combineLevels(test$ED_RESULT,levs=c(1,2,15),newLabel = c("further treatment"))
test$ED_RESULT=combineLevels(test$ED_RESULT,levs=c(1,6,7,8),newLabel = c("leaving without completing treatment"))
test$ED_RESULT=combineLevels(test$ED_RESULT,levs=c(4,5),newLabel = c("left without permission"))
test$ED_RESULT=combineLevels(test$ED_RESULT,levs=c(5,6),newLabel = c("L&D"))
test$ED_RESULT=combineLevels(test$ED_RESULT,levs=c(1,3),newLabel = c("Discharged"))

levels(test$DC_RESULT)
levels(test$ED_RESULT)
#seperate the data set by the ADMIT_RESULTS
test_ip <- test[!is.na(test$ADMIT_RESULT),]
test_op <-test[is.na(test$ADMIT_RESULT),]

#elininated "ADMIT_RESULT", "RISK", "SEVERITY", "WEEKDAY_DEP", "HOUR_DEP", "MONTH_DEP", "ETHNICITY" from
test_op
test_ip<-subset(test_ip,select = -c(WEEKDAY_DEP, HOUR_DEP, MONTH_DEP, ETHNICITY))
```

```
#elininated "ADMIT_RESULT", "RISK", "SEVERITY", "WEEKDAY_DEP", "HOUR_DEP", "MONTH_DEP", "ETHNICITY" from
test_op
test_op<-subset(test_op,select = -c(WEEKDAY_DEP, HOUR_DEP, MONTH_DEP, ETHNICITY,ADMIT_RESULT, RISK,
SEVERITY))

test_op$DC_RESULT[is.na(test_op$DC_RESULT)]<-'other'


library(Matrix)
testip_sparse <- sparse.model.matrix(RETURN ~ ., data = test_ip[,-1])
testop_sparse <- sparse.model.matrix(RETURN ~ ., data = test_op[,-1])

test
sapply(test_ip,function(x) sum(is.na(x)))
sapply(test_op,function(x) sum(is.na(x)))



#classification regression
reg_ip_all <- glm(RETURN~.,data=ip_nonull[,-1], family="binomial")
reg_op_all <-glm(RETURN~.,data=op_nonull[,-1], family="binomial")
reg_ip_null <- glm(RETURN~1,data=ip_nonull[,-1], family="binomial")
reg_op_null <-glm(RETURN~1,data=op_nonull[,-1], family="binomial")
#stepwise
ip_both = step(reg_ip_null, scope=list(upper=reg_ip_all), direction="both", trace=1)
summary(ip_both)
op_both = step(reg_op_null, scope=list(upper=reg_op_all), direction="both", trace=0)
summary(op_both)
#predict the ip train and test set with classification regresssion
reg_pred_ip <- predict(ip_both,newdata=test_ip[,-1],type="response")
reg_pred_op <- predict(op_both,newdata=test_op[,-1],type="response")

#find the best cutoff on ip
cutoffs=c(0.01,0.05,0.1,0.15,0.2,0.3,0.4,0.45,0.5,0.53,0.55,0.6,0.63)
reg_ip_acc=rep(0,13)
reg_ip_TPR=rep(0,13)
reg_ip_TNR=rep(0,13)
for (i in 1:13){
  reg_class_ip=ifelse(reg_pred_ip>cutoffs[i],1,0)
  reg_confuse_test=table(test_ip$RETURN,reg_class_ip)
  reg_ip_TPR[i]=(reg_confuse_test[2,2]/sum(reg_confuse_test[2,]))
  reg_ip_TNR[i]=(reg_confuse_test[1,1]/sum(reg_confuse_test[1,]))
  reg_ip_acc[i]=(reg_confuse_test[1,1]+reg_confuse_test[2,2])/sum(reg_confuse_test)
}
print(paste('For ip ,stepwise model at best cutoff=',toString(cutoffs[which.max(reg_ip_acc)]),', the acc =',toString(max(reg_ip_acc))))


#find the best cutoff on op
cutoffs=c(0.03,0.05,0.1,0.15,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)
reg_op_acc=rep(0,12)
reg_op_TPR=rep(0,12)
reg_op_TNR=rep(0,12)
for (i in 1:13){
  reg_class_op=ifelse(reg_pred_op>cutoffs[i],1,0)
  reg_confuse_test_op=table(test_op$RETURN,reg_class_op)
  reg_op_TPR[i]=(reg_confuse_test_op[2,2]/sum(reg_confuse_test_op[2,]))
  reg_op_TNR[i]=(reg_confuse_test_op[1,1]/sum(reg_confuse_test_op[1,]))
```

```
  reg_op_acc[i]=(reg_confuse_test_op[1,1]+reg_confuse_test_op[2,2])/sum(reg_confuse_test_op)
}
print(paste('For op, stepwise model at best cutoff=',toString(cutoffs[which.max(reg_op_acc)]),', the acc
=',toString(max(reg_op_acc))))

#combine the training and testing data together
all_ip=rbind(ip_sparse,testip_sparse)
all_op=rbind(op_sparse,testop_sparse)
ip_pca = prcomp(all_ip)
summary(ip_pca)
plot(ip_pca)
#combine the training and testing data
all_op=rbind(op_sparse,testop_sparse)
all_op=rbind(op_sparse,testop_sparse)
op_pca = prcomp(all_op)
summary(op_pca)
plot(op_pca)
#we only choose pc1
dfip_pca=prcomp(ip_sparse)
dfop_pca=prcomp(op_sparse)
dfip_pca1=dfip_pca$x[,1]
dfop_pca1=dfop_pca$x[,1]

#train the model with pc1
ip_pca1_model=glm(ip_nonull$RETURN~dfip_pca1,family='binomial')
op_pca1_model=glm(op_nonull$RETURN~dfop_pca1,family='binomial')
#compare the pca model and the logistic one
summary(ip_pca1_model)
summary(op_pca1_model)
summary(ip_both)
summary(op_both)

###xgBoost
library('xgboost')
traindataip <- ip_sparse
label_ip=ip_nonull$RETURN
bst_ip<-xgboost(data=as.matrix(traindataip),label=label_ip,max.depth=7,eta=4,nrounds=130,objective = "binary:logistic")
ip_pred_xgb<-predict(bst_ip,newdata = as.matrix(testip_sparse),type='response')
#plot the importance matrix of ip
ip_importance_matrix <- xgb.importance(colnames(as.matrix(traindataip)), model = bst_ip)
xgb.plot.importance(ip_importance_matrix)
#plot the importance matrix of op
op_importance_matrix <- xgb.importance(colnames(as.matrix(traindataop)), model = bst_op)
xgb.plot.importance(op_importance_matrix)
#choose a best cutoff on ip data
cutoffs_xgb_ip=c(0.01,0.09,0.1,0.15,0.2,0.3,0.4,0.45,0.5,0.6,0.7,0.8,0.9)
xgb_ip_acc=rep(0,13)
xgb_ip_TPR=rep(0,13)
xgb_ip_TNR=rep(0,13)
for (i in 1:13){
  xgb_class=ifelse(ip_pred_xgb>cutoffs_xgb_ip[i],1,0)
  xgb_confuse_test=table(test_ip$RETURN,xgb_class)
  xgb_ip_TPR[i]=(xgb_confuse_test[2,2]/sum(xgb_confuse_test[2,]))
  xgb_ip_TNR[i]=(xgb_confuse_test[1,1]/sum(xgb_confuse_test[1,]))
  xgb_ip_acc[i]=(xgb_confuse_test[1,1]+xgb_confuse_test[2,2])/sum(xgb_confuse_test)
}
```

```
print(paste('For ip data, XGboost model at cutoff=',toString(cutoffs_xgb_ip[which.max(xgb_ip_acc)],),'will have a highest acc',',which is =',toString(max(xgb_ip_acc))))
#do the same on the op data
traindataop <- op_sparse
label_op=op_nonull$RETURN
bst_op<-xgboost(data=as.matrix(traindataop),label=label_op,max.depth=7,eta=4,nround=521,objective = "binary:logistic")
op_pred_xgb<-predict(bst_op,newdata = as.matrix(testop_sparse),type='response')
cutoffs_xgb_op=c(0.01,0.09,0.1,0.15,0.2,0.3,0.4,0.5,0.65,0.7,0.8,0.9,0.95)
xgb_op_acc=rep(0,13)
xgb_op_TPR=rep(0,13)
xgb_op_TNR=rep(0,13)
for (i in 1:13){
  xgb_class=ifelse(op_pred_xgb>cutoffs_xgb_op[i],1,0)
  xgb_confuse_test=table(test_op$RETURN,xgb_class)
  xgb_op_TPR[i]=(xgb_confuse_test[2,2]/sum(xgb_confuse_test[2,]))
  xgb_op_TNR[i]=(xgb_confuse_test[1,1]/sum(xgb_confuse_test[1,]))
  xgb_op_acc[i]=(xgb_confuse_test[1,1]+xgb_confuse_test[2,2])/sum(xgb_confuse_test)
}

print(paste('For op data, XGboost model at cutoff=',toString(cutoffs_xgb_ip[which.max(xgb_op_acc)],),
      'will have a highest acc',',which is =',toString(max(xgb_op_acc))))

###randomForest
rf_ip=randomForest(as.factor(RETURN)~.,data=ip_nonull[,-1],ntree=500,mtry=4,importance=TRUE)
ip_pred_rf=predict(rf_ip,newdata=test_ip[,-1],type="prob")
ip_probs_rf=ip_pred_rf[,2]
cutoffs_rf_ip=c(0.03,0.09,0.1,0.15,0.2,0.3,0.4,0.5,0.6)
rf_ip_acc=rep(0,9)
rf_ip_TPR=rep(0,9)
rf_ip_TNR=rep(0,9)
for (i in 1:13){
  rf_class=ifelse(ip_probs_rf>cutoffs_rf_ip[i],1,0)
  rf_confuse_test=table(test_ip$RETURN,rf_class)
  rf_ip_TPR[i]=(rf_confuse_test[2,2]/sum(rf_confuse_test[2,]))
  rf_ip_TNR[i]=(rf_confuse_test[1,1]/sum(rf_confuse_test[1,]))
  rf_ip_acc[i]=(rf_confuse_test[1,1]+rf_confuse_test[2,2])/sum(rf_confuse_test)
}
print(paste('For ip data, rf model at cutoff=',toString(cutoffs_rf_ip[which.max(rf_ip_acc)],),
      'will have a highest acc',',which is =',toString(max(rf_ip_acc))))

rf_op=randomForest(as.factor(RETURN)~.,data=op_nonull[,-1],ntree=500,mtry=4,importance=TRUE)
op_pred_rf=predict(rf_op,newdata=test_op[,-1],type="prob")
op_probs_rf=op_pred_rf[,2]
cutoffs_rf_op=c(0.03,0.09,0.1,0.15,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95)
rf_op_acc=rep(0,13)
rf_op_TPR=rep(0,13)
rf_op_TNR=rep(0,13)
for (i in 1:13){
  rf_class=ifelse(op_probs_rf>cutoffs_rf_op[i],1,0)
  rf_confuse_test=table(test_op$RETURN,rf_class)
  rf_op_TPR[i]=(rf_confuse_test[2,2]/sum(rf_confuse_test[2,]))
  rf_op_TNR[i]=(rf_confuse_test[1,1]/sum(rf_confuse_test[1,]))
  rf_op_acc[i]=(rf_confuse_test[1,1]+rf_confuse_test[2,2])/sum(rf_confuse_test)
}
print(paste('For op data, rf model at cutoff=',toString(cutoffs_rf_op[which.max(rf_op_acc)],),
      'will have a highest acc',',which is =',toString(max(rf_op_acc))))
importance(rf_ip)
```

```
varImpPlot(rf_ip)
importance(rf_op)
varImpPlot(rf_op)
```