Habits

Contents

Task 1: Predicion modelling

Part 1: Inspecting, cleaning and imputing the data set

In this section, we will train and rund three different prediction models based on a data set containing financial records of a list of companies. In total, there are 24 variables in the data set and xxx observations. Prior to training our prediction models, we load and examine the data set.

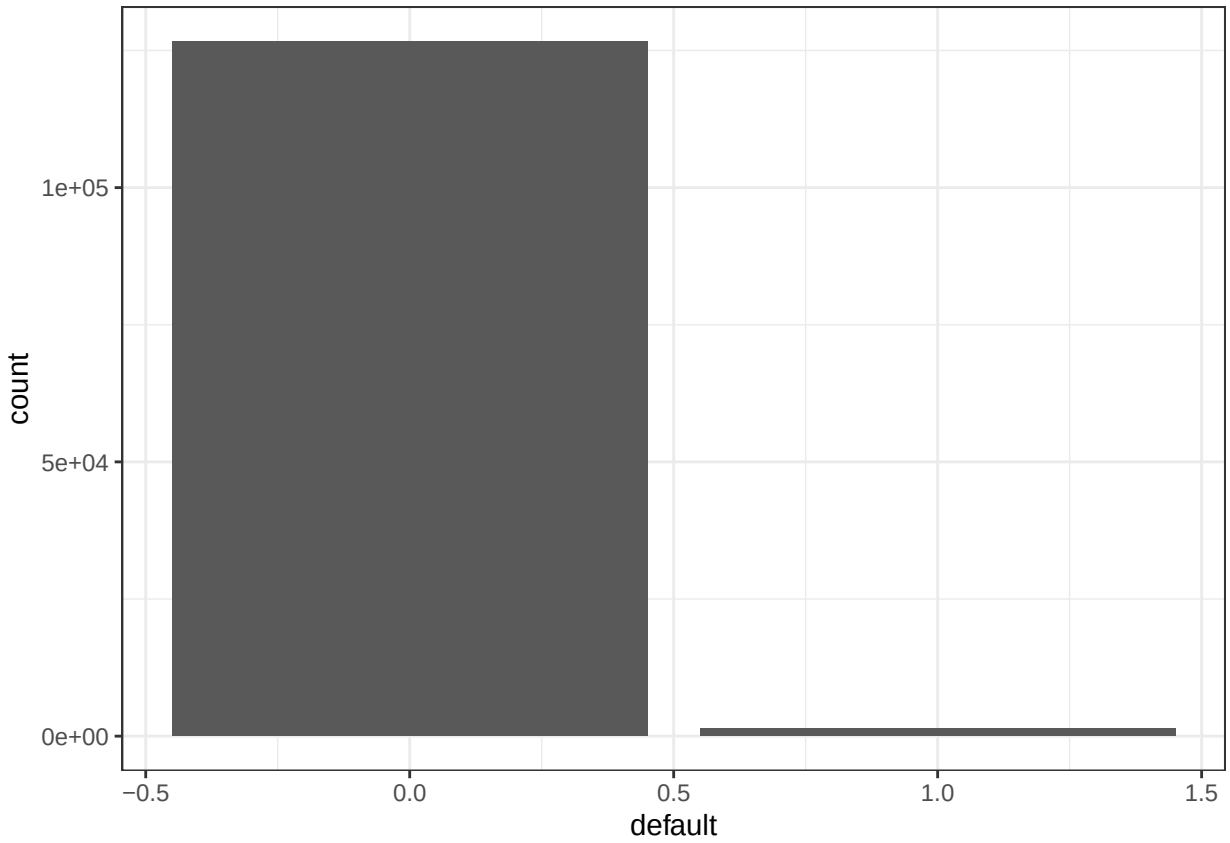
The first thing we notice is that some variables are classified as numeric when they should be factor variables, and vice versa. We reclassify these variables.

Table 1:

Statistic	Min	Max
default	0	1
profit_margin	−10,000,000,000,000,000,000.000	25,553.330
gross_operating_inc_perc	0.000	1.000
operating_margin	−10,000,000,000,000,000,000.000	586.175
EBITDA_margin	−10,000,000,000,000,000,000.000	586.171
interest_coverage_ratio	−10,000,000,000,000,000,000.000	10,000,000,000,000,000,000.000
cost_of_debt	−8,131.722	165,944.600
interest_bearing_debt	−10,000,000,000,000,000,000.000	7,523.000
revenue_stability	−10,000,000,000,000,000,000.000	4,473.242
equity_ratio	−10,000,000,000,000,000,000.000	168.999
equity_ratio_stability	−10,000,000,000,000,000,000.000	168.000
liquidity_ratio_1	−2,473.828	10,000,000,000,000,000,000.000
liquidity_ratio_2	−2,473.790	10,000,000,000,000,000,000.000
liquidity_ratio_3	−2,473.809	10,000,000,000,000,000,000.000
equity	−771,200	182,466,000
total_assets	−9,685	544,267,000
revenue	−2,883,588	588,422,000
age_of_company	2	22
unpaid_debt_collection	−5.000	10,000,000,000,000,000,000.000
paid_debt_collection	−5.500	10,000,000,000,000,000,000.000
adverse_audit_opinion	0	6
industry	0	11
amount_unpaid_debt	−121,226	10,000,000,000,000,000,000
payment_reminders	0	3

Table xx displays summary statistics of our data. As shown in the table, the value xx is present in many of the observations. Most likely, these values represent measurement errors or missing variables, and we therefore choose to replace these values with NAs. After completing this step, there are still some quite extreme outliers remaining in the data. Our assessment is that these values do not represent real observations, as they deviate from typical values for the financial key figures represented in the variables in question. Applying a threshold of 2,5 percent at each end of the variables' distribution, we replace values exceeding this threshold with NAs.

We also notice that the variable default, which will be the dependent variable in our prediction models, is imbalanced. That is, observations of defaulting companies are underrepresented, which will make it difficult to predict defaults. The underrepresentation of defaults is illustrated in figure **??**. we will handle this imbalance when we implement the prediction models in section xx.


1e+05
5e+04
0e+00
count
−0.5
0.0
0.5
1.0
1.5
default

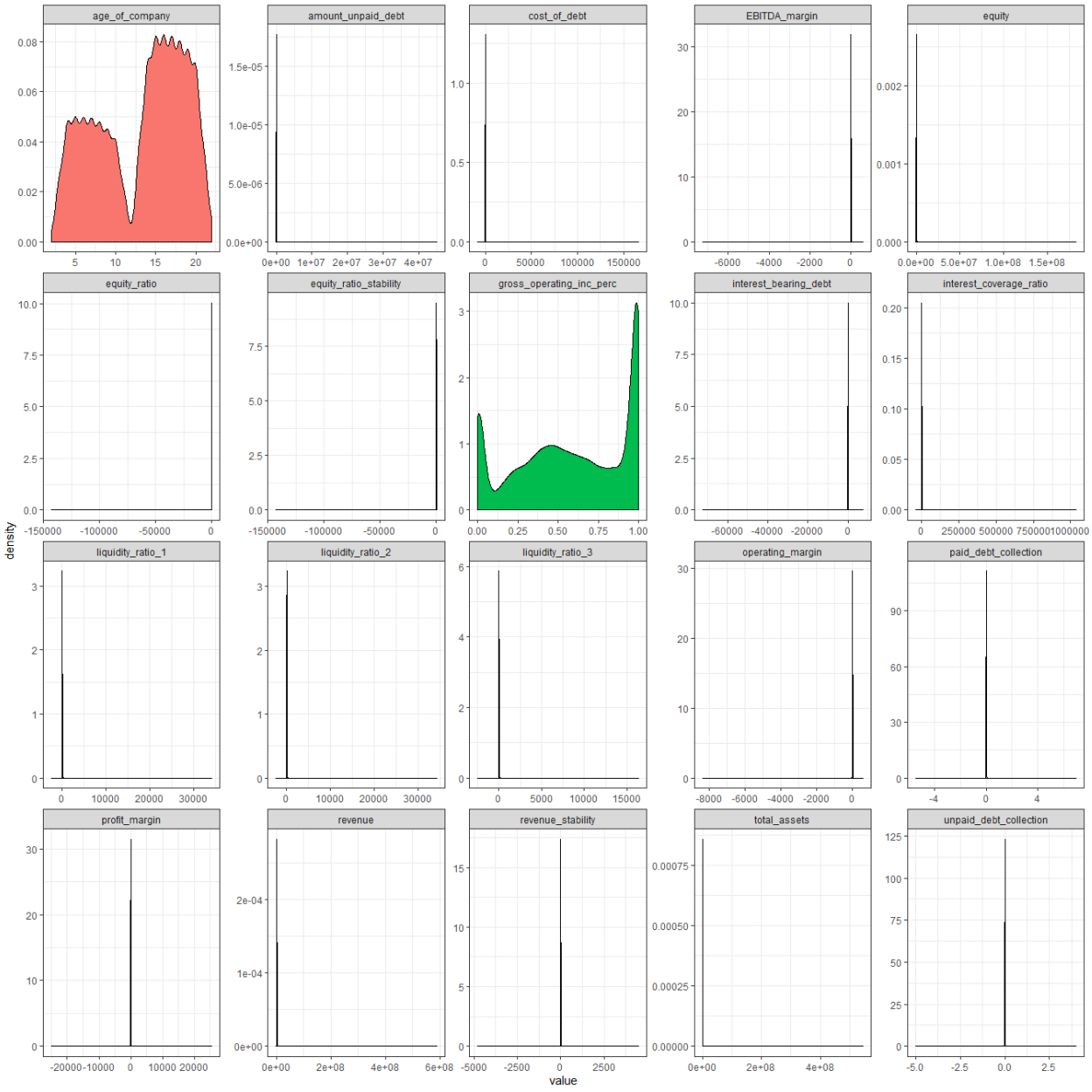
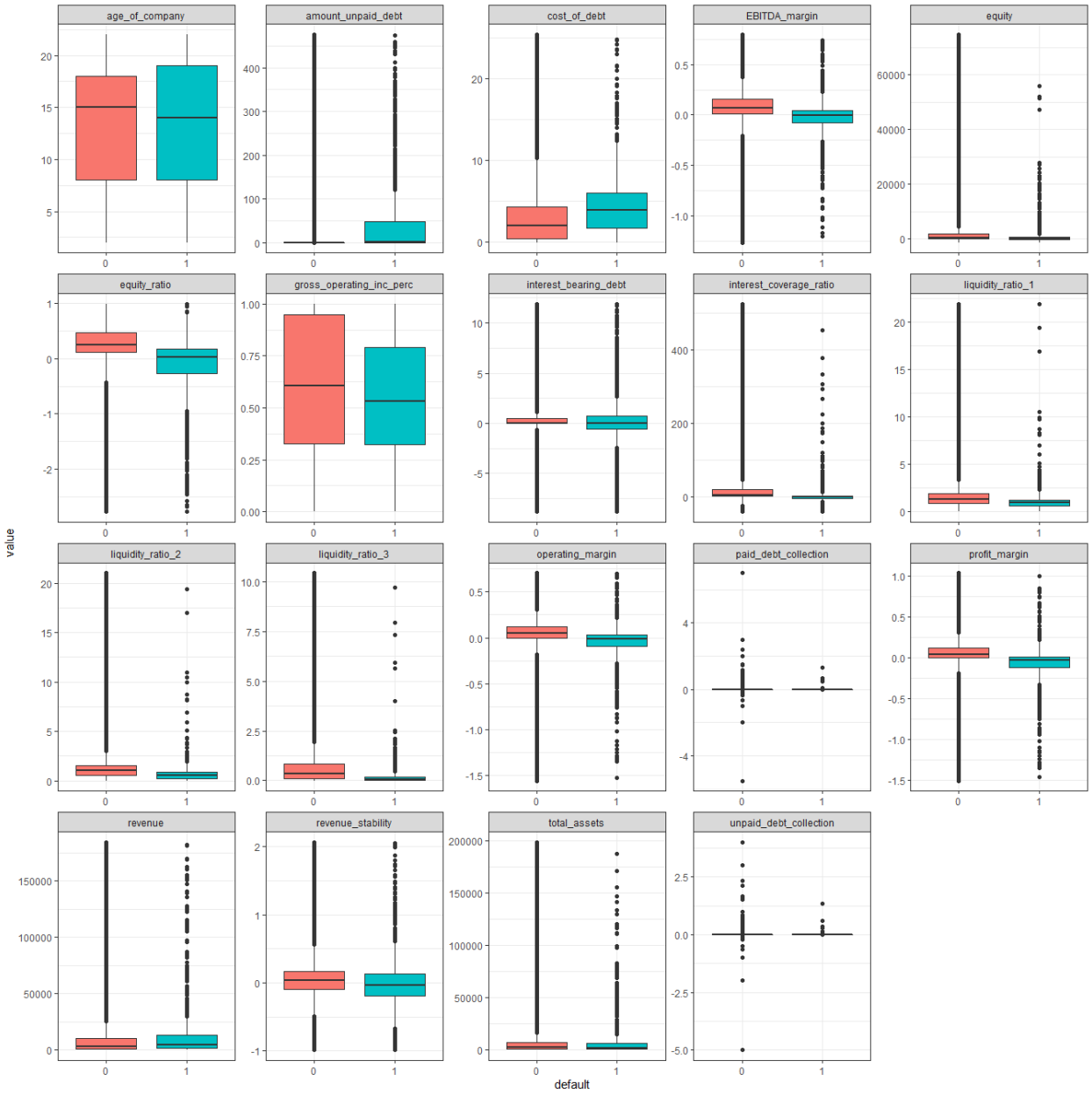
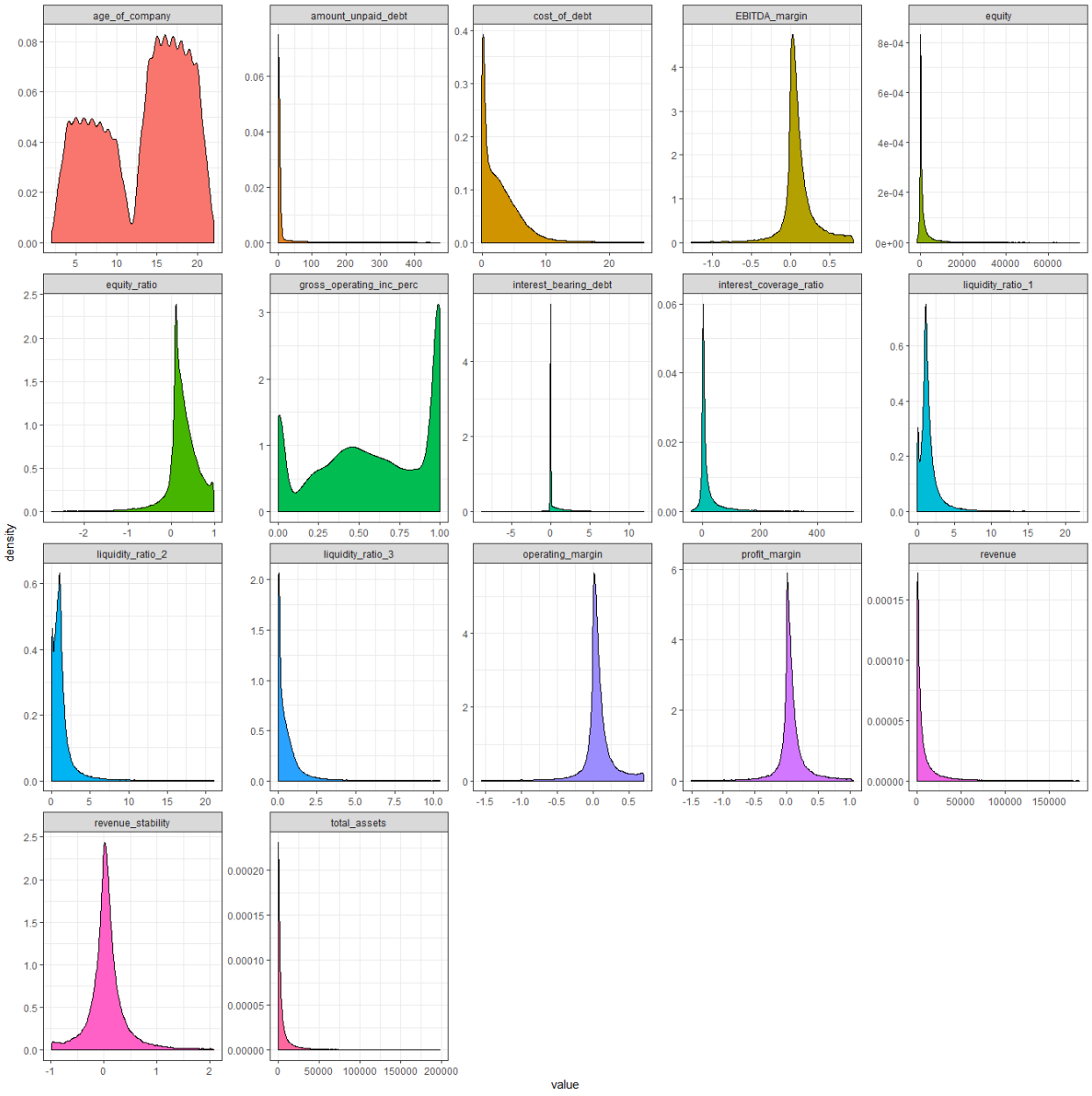


Figure **??** and **??** contain box- and density plots of the numeric variables in our data set after cleaning:


age_of_company
amount_unpaid_debt
cost_of_debt
EBITDA_margin
equity
equity_ratio
gross_operating_inc_perc
interest_bearing_debt
interest_coverage_ratio
liquidity_ratio_1
liquidity_ratio_2
liquidity_ratio_3
operating_margin
paid_debt_collection
profit_margin
revenue
revenue_stability
total_assets
unpaid_debt_collection
value
default



After cleaning the data set, we are left with xx missing values. As these account for a substantial fraction of the total number of observations in the data set, we choose to impute these using the MICE package. An illustration and explanation of the imputation added to the appendix.

##Prediction modelling

Preparations

Prior to training our prediction models, we run the data set through a function which detects whether any of the variables correlate. It turns out that the variables *total assets*, *revenue*, *industry* and *paid debt collection* correlate with one or more of the other variables. To avoid multicollinearity, these variables are removed from the data set.

Further, the data is split into a training set and a test set.

Logistic Regression Model

Our first prediction model is a logistic regression model. - Using cross validation - Oversampling method implemented here

Summary statistics from the logistic regression model are presented in table xxxx. - comments: which variables are significant, and do they make economically sense

Figure xxx gives and overview of the variable importance.
- Comments:

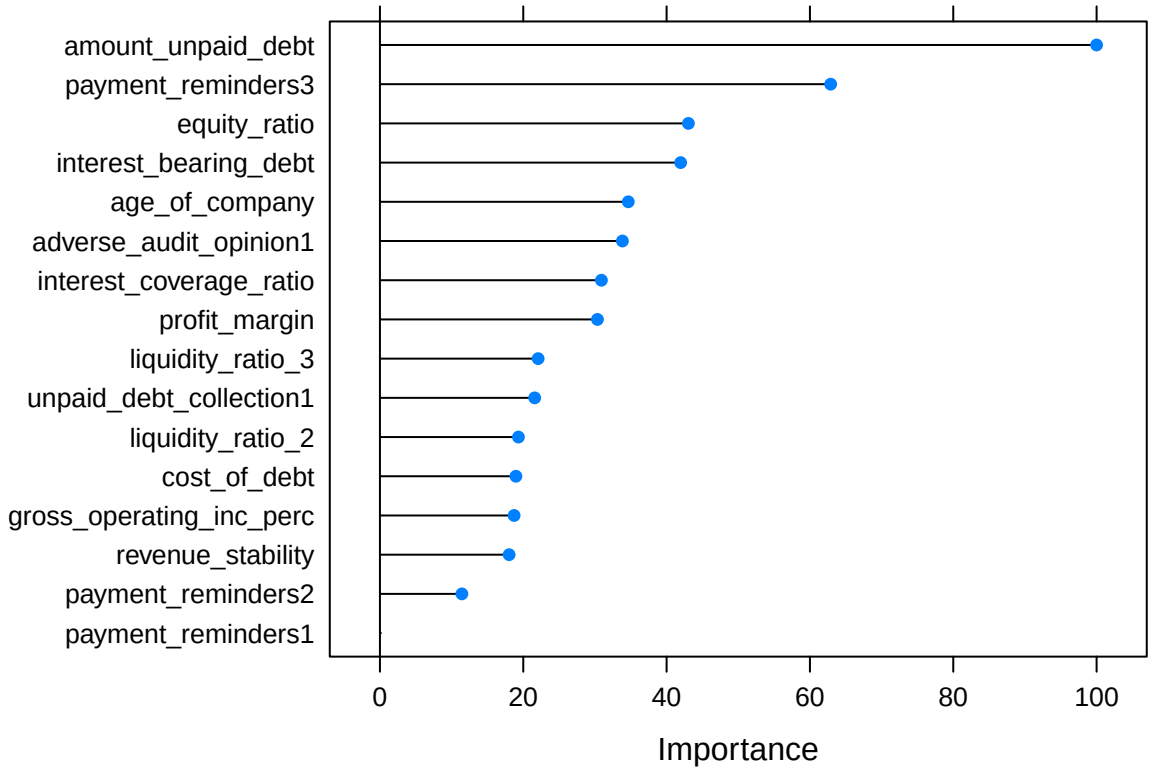
A confusion matrix from the glm model is presented in table xx. - Comments:

Figure xx shows the AUC for the glm model. - Comments:

Random forest

Our second predicion model is a random forest model. Comments: - We have implemented cross validation - Fine tuning? - Oversampling technique is also implemented here.

The model's confusion matrics is presented in figure xx, and the ROC curve is presented in figure xx. - Discuss performance



Confusion Matrix and Statistics

```
      Reference
```

Prediction 0 1 0 34738 139 1 3262 281

```
           Accuracy : 0.9115
             95% CI : (0.9086, 0.9143)
No Information Rate : 0.9891
P-Value [Acc > NIR] : 1

              Kappa : 0.1247


Mcnemar's Test P-Value : <2e-16

        Sensitivity : 0.91416
        Specificity : 0.66905
     Pos Pred Value : 0.99601
     Neg Pred Value : 0.07931
         Prevalence : 0.98907
     Detection Rate : 0.90416

Detection Prevalence : 0.90778
Balanced Accuracy : 0.79160

   'Positive' Class : 0
```

-

Show ROC here # -

model_xgb <- readRDS("xgb.Rdata")

model_xgb

plot(varImp(model_xgb))

xgb_pred <- data.frame(actual = test_data$default, predict(model_xgb, newdata = test_data, type = "prob"))

rf_pred$predict < -ifelse(xgb_pred$X1 > 0.5, 1, 0) xgb_pred$predict < -as.factor(xgb_pred$predict)

cm_xgb <- confusionMatrix(xgb_pred$predict, test_data$default) cm_xgb

ROC curve xgb

result.predicted.prob <- predict(model_xgb, test_data, type="prob") # Prediction

result.roc <- roc(test_data$default, result.predicted.prob$1) # Draw ROC curve.

plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")

result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy")) print(result.coords)#to get threshold and accuracy

Look at at difference all together

Look at the performance

models <- list(glm = model_glm, rf = model_rf, xgb = model_xgb)

resampling <- resamples(models)

bwplot(resampling)

density plots of accuracy

scales <- list(x=list(relation="free"), y=list(relation="free")) densityplot(resampling, scales=scales, pch = "|", allow.multiple = TRUE)

Other snacks for comparing

splom(resampling)

xyplot(resampling, models=c("rf", "xgb"))

summary(resampling)