CSP571 Project - Tiffany Modeling part

Tiffany Wong

2022-12-02

Contents

libraries

read in dataframe

```r
train_num <- read_csv("/Users/tiffwong/Desktop/csp571/project/datasets/train/joint_train_numeric.csv")
```

```
## New names:Rows: 14304 Columns: 210-- Column specification ------------------------------------------
## Delimiter: ","
## dbl (210): ...1, telecommuting, has_company_logo, has_questions, fraudulent,...
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# head(train_num)

test_num <- read_csv("/Users/tiffwong/Desktop/csp571/project/datasets/test/joint_test_numeric.csv")
```

```
## New names:Rows: 3576 Columns: 210-- Column specification ------------------------------------------
## Delimiter: ","
## dbl (210): ...1, telecommuting, has_company_logo, has_questions, fraudulent,...
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# head(test_num)

test_num[is.na(test_num)] = 0

train_num$X<-NULL
test_num$X<-NULL
train_num$fraudulent<-as.factor(train_num$fraudulent)
test_num$fraudulent<-as.factor(test_num$fraudulent)
train_num$department_n_first_personp<-NULL
test_num$dep_oil<-NULL
# colnames(train_num)[colSums(is.na(train_num)) > 0]
# colnames(test_num)[colSums(is.na(test_num)) > 0]

# sanity check: should be all 0's
# sapply(train_num, function(x) sum(is.na(x)))
head(train_num)
```

```
## # A tibble: 6 x 209
##    ...1 teleco~1 has_c~2 has_q~3 fraud~4 has_l~5 has_d~6 has_s~7 has_c~8 has_r~9
##   <dbl>    <dbl>   <dbl>   <dbl> <fct>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     1        0       0       0 0             1       0       0       0       1
## 2     2        0       1       1 0             1       0       0       1       1
## 3     3        0       0       0 0             1       0       0       1       0
## 4     4        0       1       0 0             1       0       0       1       1
## 5     5        0       1       1 0             1       0       0       1       1
## 6     6        0       1       0 0             1       0       0       1       1
## # ... with 199 more variables: has_benefits <dbl>, has_employment_type <dbl>,
## #   has_required_experience <dbl>, has_required_education <dbl>,
## #   has_industry <dbl>, has_fn <dbl>, department_n_chars <dbl>,
## #   department_n_extraspaces <dbl>, department_n_words <dbl>,
## #   department_n_uq_words <dbl>, department_n_caps <dbl>,
## #   department_n_nonasciis <dbl>, department_n_charsperword <dbl>,
## #   department_sent_afinn <dbl>, department_sent_bing <dbl>, ...
```

```r
head(test_num)
```

```
## # A tibble: 6 x 209
```

```
##    ...1 teleco~1 has_c~2 has_q~3 fraud~4 has_l~5 has_d~6 has_s~7 has_c~8 has_r~9
##   <dbl>    <dbl>   <dbl>   <dbl> <fct>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     1        0       1       0 0             1       1       0       1       1
## 2     2        0       1       1 0             1       0       0       1       1
## 3     3        0       0       0 0             1       0       1       0       1
## 4     4        0       1       0 0             1       1       0       1       1
## 5     5        0       1       1 0             1       1       0       1       1
## 6     6        0       1       1 0             1       0       0       1       1
## # ... with 199 more variables: has_benefits <dbl>, has_employment_type <dbl>,
## #   has_required_experience <dbl>, has_required_education <dbl>,
## #   has_industry <dbl>, has_fn <dbl>, department_n_chars <dbl>,
## #   department_n_extraspaces <dbl>, department_n_words <dbl>,
## #   department_n_uq_words <dbl>, department_n_caps <dbl>,
## #   department_n_nonasciis <dbl>, department_n_charsperword <dbl>,
## #   department_sent_afinn <dbl>, department_sent_bing <dbl>, ...
```

```
# test_num$department_n_first_personp
```

logistic regression

```
model <- suppressWarnings(glm(fraudulent~., family=binomial(link='logit'), data=train_num))
log_model_summary <- summary(model)
```

chi-sq test

This was already ran and saved its output as chisq_text.csv for coding conveinence because running it takes a long time.

```
# anova_out <- suppressWarnings(anova(model, test="Chisq") )

# summary_anova <- summary(anova_out)
```

export chi-sq results

```
# write_csv(anova_out, "/Users/tiffwong/Desktop/csp571/project/datasets/chisq_test.csv")
# write_csv(summary_anova, "/Users/tiffwong/Desktop/csp571/project/datasets/chisq_summary.csv")
```

read in chi-sq csv

```
# read in chisq_test.csv dataset
chisq <- read_csv("/Users/tiffwong/Desktop/csp571/project/datasets/chisq_test.csv")
```

```
## Rows: 209 Columns: 5-- Column specification ----------------------------------------------------------
## Delimiter: ","
## dbl (5): Df, Deviance, Resid. Df, Resid. Dev, Pr(>Chi)
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# dimension of training dataset
dim(chisq)
```

```
## [1] 209   5
```

get indices of attributes with p-value < 0.01 (significant attributes)

```
# get all p-value greater than 0.01 (meaning significance)
sig_pval_index <- which(chisq$"Pr(>Chi)" < 0.01)
length(sig_pval_index)
```

```
## [1] 83
```

```
# only select columns from training dataset with p-value
train_df <- train_num[,sig_pval_index]
head(train_df)
```

```
## # A tibble: 6 x 83
##   has_company_~1 has_q~2 fraud~3 has_d~4 has_s~5 has_c~6 has_r~7 has_b~8 has_e~9
##            <dbl>   <dbl> <fct>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1              0       0 0             0       0       0       1       1       1
## 2              1       1 0             0       0       1       1       0       0
## 3              0       0 0             0       0       1       0       0       1
## 4              1       0 0             0       0       1       1       1       1
## 5              1       1 0             0       0       1       1       1       1
## 6              1       0 0             0       0       1       1       0       0
## # ... with 74 more variables: has_required_experience <dbl>,
## #   has_industry <dbl>, has_fn <dbl>, department_n_chars <dbl>,
## #   department_n_uq_words <dbl>, department_n_caps <dbl>,
## #   department_n_charsperword <dbl>, department_sent_vader <dbl>,
## #   department_n_second_personp <dbl>, company_profile_n_hashtags <dbl>,
## #   company_profile_n_chars <dbl>, company_profile_n_exclaims <dbl>,
## #   company_profile_n_words <dbl>, company_profile_n_uq_words <dbl>, ...
```

subset test data with only useful columns too

```
test_df <- test_num[,sig_pval_index]

# check if test data has fraudulent
train_df$fraudulent <- as.factor(train_df$fraudulent)
test_df$fraudulent <- as.factor(test_df$fraudulent)

head(test_df)
```

```
## # A tibble: 6 x 83
##   has_company_~1 has_q~2 fraud~3 has_d~4 has_s~5 has_c~6 has_r~7 has_b~8 has_e~9
##            <dbl>   <dbl> <fct>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1              1       0 0             1       0       1       1       0       1
## 2              1       1 0             0       0       1       1       1       1
## 3              0       0 0             0       1       0       1       1       1
## 4              1       0 0             1       0       1       1       0       1
## 5              1       1 0             1       0       1       1       1       1
## 6              1       1 0             0       0       1       1       0       0
## # ... with 74 more variables: has_required_experience <dbl>,
## #   has_industry <dbl>, has_fn <dbl>, department_n_chars <dbl>,
## #   department_n_uq_words <dbl>, department_n_caps <dbl>,
## #   department_n_charsperword <dbl>, department_sent_vader <dbl>,
## #   department_n_second_personp <dbl>, company_profile_n_hashtags <dbl>,
## #   company_profile_n_chars <dbl>, company_profile_n_exclaims <dbl>,
```

```
## #   company_profile_n_words <dbl>, company_profile_n_uq_words <dbl>, ...
```

double check that colnames are the same in both training and testing data

```r
colnames1 <- names(train_df)
all_colnames <- paste0(paste0("'", colnames1, "'"), collapse = ", ")
colnames2 <- names(test_df)
all_colnames2 <- paste0(paste0("'", colnames2, "'"), collapse = ", ")

all_colnames == all_colnames2
```

```
## [1] TRUE
```

stochastic gradient boosting

training

```r
# training dataset is: train_df
# training labels is: train_num['fraudulent']

# Fit the model on the training set
set.seed(123)
xgbmodel <- train(
  fraudulent ~., data = train_df, method = "xgbTree",
  trControl = trainControl("cv", number = 10),
  verbose=FALSE, verbosity = 0
  )

# Best tuning parameter
xgbmodel$bestTune
```

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 99     150         3 0.4     0              0.6                1         1
```

evaluation

precition and confusion matrix

```r
# Make predictions on the test data
predicted.classes <- xgbmodel %>% predict(test_df)
confusionMatrix(predicted.classes, test_df$fraudulent)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3396   50
##          1   14  116
##
##                Accuracy : 0.9821
##                  95% CI : (0.9772, 0.9862)
##     No Information Rate : 0.9536
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.7746
##
##  Mcnemar's Test P-Value : 1.214e-05
##
##             Sensitivity : 0.9959
##             Specificity : 0.6988
##          Pos Pred Value : 0.9855
##          Neg Pred Value : 0.8923
##              Prevalence : 0.9536
##          Detection Rate : 0.9497
##    Detection Prevalence : 0.9636
##       Balanced Accuracy : 0.8473
##
##        'Positive' Class : 0
##
```

accuracy rate

```r
# Compute model prediction accuracy rate
accuracy_sgb <- mean(predicted.classes == test_num$fraudulent)
accuracy_sgb
```

```
## [1] 0.9821029
```

variables of importance

```r
variables_imp <- varImp(xgbmodel)
variables_imp
```

```
## xgbTree variable importance
##
##   only 20 most important variables shown (out of 82)
##
##                                Overall
## company_profile_n_chars         100.00
## has_company_logo                 92.88
## has_company_profile              83.93
## company_profile_n_third_person   75.12
## industry_oilenergy               74.68
## company_profile_n_polite         66.61
## company_profile_n_words          59.38
## fn_Administrative                59.29
## description_n_polite             53.41
## company_profile_sent_afinn       52.01
## description_n_caps               50.62
## YNusa                            47.26
## description_sent_vader           44.90
## company_profile_n_exclaims       44.18
## benefits_n_caps                  44.07
## company_profile_n_uq_words       42.66
## description_n_chars              39.58
## department_n_charsperword        35.56
## requirements_sent_vader          34.63
## region_cat_SW                    30.50
```

random forest classifier

```
# rename all colnames
names(train_df) <- make.names(names(train_df))
names(test_df) <- make.names(names(test_df))
```

training model

```
# training dataset is: train_df
rf <- randomForest(fraudulent~., data=train_df, proximity=TRUE, importance=TRUE)
print(rf)
```

```
##
## Call:
##  randomForest(formula = fraudulent ~ ., data = train_df, proximity = TRUE,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 1.8%
## Confusion matrix:
##       0   1  class.error
## 0 13591  13 0.0009556013
## 1   244 456 0.3485714286
```

plotting and saving png

```
# Output to be present
# As PNG file
png(file = "randomForestClassification.png")

# Plot the error vs
# The number of trees graph
plot(rf)

# Saving the file
dev.off()
```

```
## pdf
##   2
```

train data

prediction and confusion matrix

```
p1 <- predict(rf, train_df)
confusionMatrix(p1, train_df$fraudulent)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13604     0
##          1     0   700
```

```
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9511
##          Detection Rate : 0.9511
##    Detection Prevalence : 0.9511
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
length(names(train_df))
```

```
## [1] 83
```

```
length(names(test_df))
```

```
## [1] 83
```

```
janitor::compare_df_cols_same(train_num, test_num)
```

```
## [1] TRUE
```

evaluation with test data

precition and confusion matrix

```
p2 <- predict(rf, test_df)
confusionMatrix(p2, test_df$fraudulent)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3406   59
##          1    4  107
##
##                Accuracy : 0.9824
##                  95% CI : (0.9775, 0.9864)
##     No Information Rate : 0.9536
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7638
##
##  Mcnemar's Test P-Value : 1.022e-11
```

```
##
##             Sensitivity : 0.9988
##             Specificity : 0.6446
##          Pos Pred Value : 0.9830
##          Neg Pred Value : 0.9640
##              Prevalence : 0.9536
##          Detection Rate : 0.9525
##    Detection Prevalence : 0.9690
##       Balanced Accuracy : 0.8217
##
##        'Positive' Class : 0
##
```

```r
confusionmatrix <- table(p2, test_df$fraudulent)
```

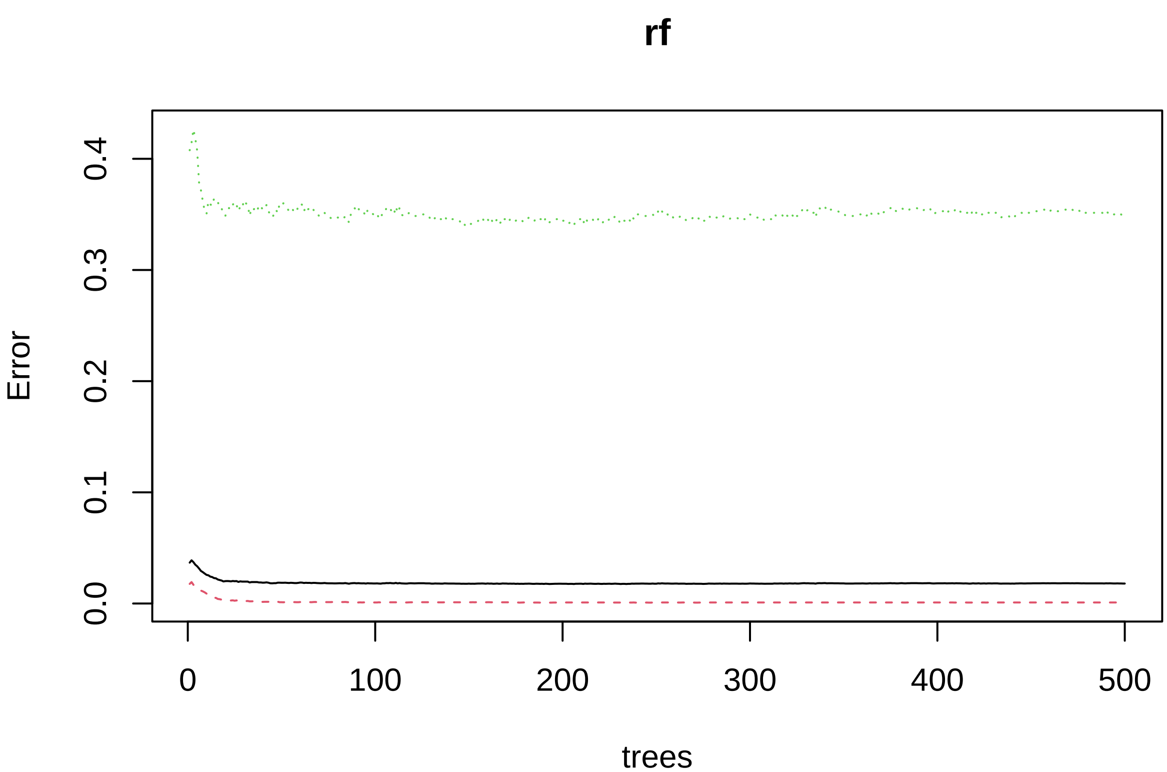
accuracy rate

```r
accuracy_random <- (sum(diag(confusionmatrix)))/sum(confusionmatrix)
accuracy_random
```

```
## [1] 0.9823826
```

error rate of rf

```r
plot(rf)
```



Ask these questions:

- does the model predict with high accuracy?
- if not, it needs further tuning -> but how?

- should we tune a number of trees and mtry basis?

classification tree

create classification tree

```
tree_model = suppressWarnings(rpart(fraudulent~., data = train_df, method = 'class') )

png(filename="classification_tree.png", height=1000, width=1800, type="cairo")

suppressWarnings(rpart.plot(tree_model) )

dev.off()
```

```
## pdf
##   2
```

evaluation

prediction and confusion matrix

```
predict_test = predict(tree_model, test_df, type = "class")
confusionMatrix(predict_test, test_df$fraudulent)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3385  103
##          1   25   63
##
##                Accuracy : 0.9642
##                  95% CI : (0.9576, 0.9701)
##     No Information Rate : 0.9536
##     P-Value [Acc > NIR] : 0.001013
##
##                   Kappa : 0.4793
##
##  Mcnemar's Test P-Value : 1.004e-11
##
##             Sensitivity : 0.9927
##             Specificity : 0.3795
##          Pos Pred Value : 0.9705
##          Neg Pred Value : 0.7159
##              Prevalence : 0.9536
##          Detection Rate : 0.9466
##    Detection Prevalence : 0.9754
##       Balanced Accuracy : 0.6861
##
##        'Positive' Class : 0
##
```

accuracy rate

```
confusionmatrix_tree <- table(predict_test, test_df$fraudulent)
confusionmatrix_tree
```

```
##
## predict_test    0    1
##            0 3385  103
##            1   25   63
```

```
accuracy_tree <- (sum(diag(confusionmatrix_tree)))/sum(confusionmatrix_tree)
accuracy_tree
```

```
## [1] 0.9642058
```

model comparison

Stochastic Gradient Boosting accuracy

```
accuracy_sgb
```

```
## [1] 0.9821029
```

Random Forest Classifier accuracy

```
accuracy_random
```

```
## [1] 0.9823826
```

Classification Tree accuracy

```
accuracy_tree
```

```
## [1] 0.9642058
```

The model with the highest accuracy is our random forest model, with 98.238255 % for accuracy.