Group 12 - R.markdown

2024-05-22

I - EXPLORING AND PREPARING DATA

Step 1: Set up all the libraries we used (already installed at the console part)

```
library(ggplot2)
library(GGally)
library(tidyverse)
library(dplyr)
library(purrr)
library(ROSE)
library(caret)
library(kknn)
library(rpart)
library(rpart.plot)
library(rsample)
library(ranger)
library(MLmetrics)
library(broom)
library(yardstick)
library(e1071)
```

Step 2: Import data by clicking “Import Dataset” (Our data file was already been excluded the first row, which is from X1-X23 by using Excel)

- Our dataset’s name is imported as “data”
- Store data to a new dataframe “cccard”
- View the data type of each variable

```
cccard <- data
View(cccard)
str(cccard)
```

Step 3: Explore data type and covert categorical columns to factor to ensure they are correctly interpreted and handled in statistical modeling and analysis (using as.factor())

```
# Define categorical columns
catagorical_columns <- c('SEX','EDUCATION','MARRIAGE','PAY_0','PAY_2','PAY_3','PAY_4','PAY_5','P
AY_6')

## Transform categorical columns to factor data type
cccard$SEX <- as.factor(as.character(cccard$SEX))
cccard$EDUCATION <- as.factor(as.character(cccard$EDUCATION))
cccard$MARRIAGE <- as.factor(as.character(cccard$MARRIAGE))
cccard$PAY_0 <- as.factor(as.character(cccard$PAY_0))
cccard$PAY_2 <- as.factor(as.character(cccard$PAY_2))
cccard$PAY_3 <- as.factor(as.character(cccard$PAY_3))
cccard$PAY_4 <- as.factor(as.character(cccard$PAY_4))
cccard$PAY_5 <- as.factor(as.character(cccard$PAY_5))
cccard$PAY_6 <- as.factor(as.character(cccard$PAY_6))
cccard$defaultpaymentnextmonth <- as.factor(as.character(cccard$defaultpaymentnextmonth))
```

Step 4: Count unidentify values in dataset
All columns below (from *pay_0* to *marriage*) have unidentified values, which are *0* and *-2*

```
pay_0 <- sum(cccard$PAY_0 == 0 | cccard$PAY_0 == -2)
pay_2 <- sum(cccard$PAY_2 == 0 | cccard$PAY_2 == -2)
pay_3 <- sum(cccard$PAY_3 == 0 | cccard$PAY_3 == -2)
pay_4 <- sum(cccard$PAY_4 == 0 | cccard$PAY_4 == -2)
pay_5 <- sum(cccard$PAY_5 == 0 | cccard$PAY_5 == -2)
pay_6 <- sum(cccard$PAY_6 == 0 | cccard$PAY_6 == -2)
education <- sum(cccard$EDUCATION == 5 | cccard$EDUCATION == 6)
marriage <- sum(cccard$MARRIAGE == 0)

#Count all unidentified values
count <- data.frame(pay_0,pay_2,pay_3,pay_4,pay_5,pay_6,education,marriage)

count
```

Step 5: Transform value *0* and *-2* from the ***PAY_0*** -> ***PAY_6*** columns to *-1* since those 2 values are not documented in the official website

```
indices <- which(cccard$PAY_0 == 0 | cccard$PAY_0 == -2)
cccard$PAY_0[indices] <- -1
indices <- which(cccard$PAY_2 == 0 | cccard$PAY_2 == -2)
cccard$PAY_2[indices] <- -1
indices <- which(cccard$PAY_3 == 0 | cccard$PAY_3 == -2)
cccard$PAY_3[indices] <- -1
indices <- which(cccard$PAY_4 == 0 | cccard$PAY_4 == -2)
cccard$PAY_4[indices] <- -1
indices <- which(cccard$PAY_5 == 0 | cccard$PAY_5 == -2)
cccard$PAY_5[indices] <- -1
indices <- which(cccard$PAY_6 == 0 | cccard$PAY_6 == -2)
cccard$PAY_6[indices] <- -1
```

Do the same thing in ***EDUCATION*** and ***MARRIAGE*** column

```
indices <- which(cccard$EDUCATION == 5 | cccard$EDUCATION == 6 | cccard$EDUCATION == 0)
cccard$EDUCATION[indices] <- 4
indices <- which(cccard$MARRIAGE == 0)
cccard$MARRIAGE[indices] <- 3
```

Step 6: Define duplicated values and remove *ID* column(as this column is uneccessary) in the dataset

```
#Define dupplicated values in the data set
sum(duplicated(cccard))

#Summary of the NA value
colSums(is.na(cccard))

#Remove first column (ID column)
cccard <- cccard[,-c(1)]
```

Step 7: Explore data distribution trends of variables to see whether they are imbalanced or not by visualization
1. Box-plot for ***Age*** and ***Limit_Bal***

```
#Box-plot for *AGE*
ggplot(data = cccard, mapping=aes(x = defaultpaymentnextmonth, y = AGE, fill = defaultpaymentnex
tmonth)) +
  geom_boxplot() +
  stat_summary(fun = "mean", geom = "point", shape = 8,
               size = 2, color = "white")

#Box-plot for *LIMIT_BAL*
ggplot(data = cccard, mapping=aes(x = defaultpaymentnextmonth, y = LIMIT_BAL, fill = defaultpaym
entnextmonth)) +
  geom_boxplot() +
  stat_summary(fun = "mean", geom = "point", shape = 8,
               size = 2, color = "white")
```

2. Bar chart for ***Sex, Education, Marriage***, and ***defaultpaymentnextmont***

```
ggplot(data = cccard, mapping=aes(x = SEX, fill = defaultpaymentnextmonth)) + geom_bar()
ggplot(data = cccard, mapping=aes(x = EDUCATION, fill = defaultpaymentnextmonth)) + geom_bar()
ggplot(data = cccard, mapping=aes(x = MARRIAGE, fill = defaultpaymentnextmonth)) + geom_bar()
ggplot(data = cccard, mapping=aes(x = defaultpaymentnextmonth, fill = defaultpaymentnextmonth))
+ geom_bar()
```

Step 8: Outlier Removal

```
#Calculate 1st, 3rd quartiles and IQR
cleanOutliers <- function(data, var_name) {
  q1 <- quantile(data[[var_name]], 0.25)
  q3 <- quantile(data[[var_name]], 0.75)
  iqr <- q3 - q1

  #Define lower and upper bounds for outlier detection
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr

  #Identify outliers
  outliers <- data[[var_name]] < lower_bound | data[[var_name]] > upper_bound

  #Remove outliers from the original dataset
  cleaned_data <- data[!outliers, ]

  summary(cleaned_data)
  return(cleaned_data)
}

cccard <- cleanOutliers(cccard, "LIMIT_BAL")
```

After exploring, there is a significant imbalance in dataset, therefore, we prepared by removing outlier and upscaling data.
Step 9: Upscaling data

```
target <- "defaultpaymentnextmonth"
predictors <- c("LIMIT_BAL","SEX","EDUCATION", "MARRIAGE","AGE","PAY_0","PAY_2","PAY_3","PAY_
4","PAY_5","PAY_6","BILL_AMT1",
                "BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL_AMT5","BILL_AMT6","PAY_AMT1","PAY_AMT
2","PAY_AMT3","PAY_AMT4",
                "PAY_AMT5","PAY_AMT6")
formula <- as.formula(paste(target, "~", paste(predictors, collapse = "+")))
```

Perform Ovun resampling then view dataset after preparing

```
cccard_bal <- ovun.sample(formula, cccard, method = "both", p = 0.5,
                          N = 29833)$data
View(cccard_bal)
summary(cccard_bal)
```

II - DATA SAMPLING

Spliting dataset into a training set and a testing set (70:30)

```
set.seed(123)
trainIndex <- sample(1:nrow(cccard_bal),0.70*nrow(cccard_bal))
cc_train <- data[trainIndex, ]
cc_test <- data[-trainIndex, ]
```

Convert train and test dataset to factor

```
cc_train$defaultpaymentnextmonth <- as.factor(cc_train$defaultpaymentnextmonth)

cc_test$defaultpaymentnextmonth <- as.factor(cc_test$defaultpaymentnextmonth)
```

III - BUILDING MODELS

This section proposes 5 models for data predicttion, including:

- Decision Tree
- Random Forest
- Knn
- SVM
- Logistic regression

Our models will be trained by historical data

3.1. Decision Tree

Build Decision Tree model to predict defaultpaymentnextmonth based on all other columns

```
tree_model <- rpart(defaultpaymentnextmonth ~., data = cc_train)
```

Visualize decision tree model

```
rpart.plot(tree_model, extra = 106)
title(main = "Decision Tree Model of Credit Card Data")
```

Make Decision Tree prediction

```
dtree.predictions <- predict(tree_model, cc_test, type = "class")
```

Compare the actual values and predictions

```
dtree.comparison <- cc_test
dtree.comparison$Predictions <- dtree.predictions
dtree.comparison[ , c("defaultpaymentnextmonth", "Predictions")]
```

Plot confusion matrix (Decision Tree)

```
actual_dt <- factor(cc_test$defaultpaymentnextmonth)
pred_dt <- factor(dtree.predictions)

confusion_matrix_dt <- table(data.frame(actual_dt, pred_dt))
confusion_matrix_dt
```

Calculating accuracy, precision, recall, f1_score (Decision Tree)

```
dt_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.comparis
on$Predictions))

dt_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(dtre
e.comparison$Predictions))

dt_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.comp
arison$Predictions))

dt_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.co
mparison$Predictions))
```

Display the metrics

```
cat("Accuracy of Decision Tree model:", dt_accuracy, "\n")
cat("Precision of Decision Tree model:", dt_precision, "\n")
cat("Recall of Decision Tree model:", dt_recall, "\n")
cat("F1-score of Decision Tree mode:", dt_f1_score, "\n")
```

3.2. Random Forest

Build Random Forest model to predict defaultpaymentnextmonth based on all other columns

```
rf_model <- ranger(defaultpaymentnextmonth ~ .,
                   data = cc_train,
                   mtry = floor((ncol(cc_train) - 1) / 5),
                   importance = "impurity",
                   num.trees = 500,
                   classification = TRUE)
```

Make random forest prediction

```
rf_pred <- predict(rf_model, data = cc_test)
rf_pred_label <- rf_pred$predictions
```

Calculating accuracy, precision, recall, f1_score (Random Forest)

```
rf_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predic
tions))

rf_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_p
red$predictions))

rf_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$pr
edictions))

rf_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred
$predictions))
rf_f1_score
```

Display the metrics

```
cat("Accuracy of Random Forest model:", rf_accuracy, "\n")
cat("Precision of Random Forest model:", rf_precision, "\n")
cat("Recall of Random Forest model:",rf_recall, "\n")
cat("F1-score of Random Forest mode:", rf_f1_score, "\n")
```

3.3 Knn Model

Build kNN model to predict defaultpaymentnextmonth based on all other columns

- Set K value to 50
- distance = 2

```
knn_model <- kknn(defaultpaymentnextmonth ~ .,
                        train = cc_train,
                        test = cc_test,
                        k = 50,
                        distance = 2)
```

Make Knn prediction

```
knn.predictions <- predict(knn_model, newdata = cc_test)
```

Comparison table for KNN actual values and predictions for defaultpaymentnextmonth

```
knn.compare <- cc_test
knn.compare$Predictions <- knn.predictions
knn.compare[, (c("defaultpaymentnextmonth", "Predictions"))]
```

Calculating accuracy, precision, recall, f1_score (Knn)

```
knn_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$P
redictions))

knn_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(kn
n.compare$Predictions))

knn_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compa
re$Predictions))

knn_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.com
pare$Predictions))
```

Display the metrics

```
cat("Accuracy of Knn model:", knn_accuracy, "\n")
cat("Precision of Knn model:", knn_precision, "\n")
cat("Recall of Knn model:",knn_recall, "\n")
cat("F1-score of Knn mode:", knn_f1_score, "\n")
```

3.4. SVM model

Build SVM model to predict defaultpaymentnextmonth based on all other columns

```
svm_model <- svm(defaultpaymentnextmonth ~ ., data = cc_train)
```

Make SVM prediction

```
svm_pred <- predict(svm_model, newdata = cc_test)
```

Comparison table for SVM actual values and predictions for defaultpaymentnextmonth

```
svm.compare <- cc_test
svm.compare$Predictions <- svm_pred
svm.compare[ , c("defaultpaymentnextmonth", "Predictions")]
```

Calculating accuracy, precision, recall, f1_score (SVM)

```
svm_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.compare$P
redictions))

svm_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(sv
m.compare$Predictions))

svm_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.compa
re$Predictions))

svm_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.com
pare$Predictions))
```

Display the metrics

```
cat("Accuracy of SVM model:", svm_accuracy, "\n")
cat("Precision of Knn model:", svm_precision, "\n")
cat("Recall of Knn model:",svm_recall, "\n")
cat("F1-score of Knn mode:", svm_f1_score, "\n")
```

3.5. Logistic regression model

Build Logistic Regression model to predict defaultpaymentnextmonth based on all other columns

```
glm_model <- glm(defaultpaymentnextmonth ~ .,
                  data = cc_train,
                  family = binomial(link = "logit"))
```

Make logistic regression prediction

```
glm_pred <- cc_test %>%
  mutate(propensity = predict(glm_model, ., type = 'response'),
         glm_est = factor(as.numeric(propensity > 0.5), labels = c("0", "1"), levels = c(0, 1)),
         index = 1:n())

glm_pred %>% select(propensity, glm_est, defaultpaymentnextmonth)
str(glm_pred)
```

Plot confusion matrix (Logistic Regression)

```
glm_pred %>%
  conf_mat(defaultpaymentnextmonth, glm_est) %>%
  pluck("table") %>%
  addmargins()
```

Calculating accuracy, precision, recall, f1_score (Logistic Regression)

```
glm_accuracy <- accuracy_vec(truth = cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_es
t)

glm_precision <- precision_vec(cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)

glm_recall <- recall_vec(cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)

glm_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est,
beta = 1)
```

Display the metrics

```
cat("Accuracy of LR model:", glm_accuracy, "\n")
cat("Precision of Knn model:", glm_precision, "\n")
cat("Recall of Knn model:",glm_recall, "\n")
cat("F1-score of Knn mode:", glm_f1_score, "\n")
```