Assignment_2

Alexis McCartney

2025-09-27

```
library(readr)
library(dplyr)
library(caret)
library(gmodels)
library(ggplot2)
```

I load the required R packages 1. readr for readings CSV files 2. dplyr for manipulating data 3. caret for machine learning and modeling 4. gmodels for summaries and cross tables 5. ggplot2 for data visualization

```
df <- read_csv("UniversalBank.csv")
names(df) <- make.names(names(df)) #make column names safe
str(df)
```

```
## spc_tbl_ [5,000 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID                : num [1:5000] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : num [1:5000] 91107 90089 94720 94112 91330 ...
##  $ Family            : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : num [1:5000] 1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : num [1:5000] 0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: num [1:5000] 1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : num [1:5000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num [1:5000] 0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : num [1:5000] 0 0 0 0 1 0 0 1 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   Age = col_double(),
##   ..   Experience = col_double(),
##   ..   Income = col_double(),
##   ..   ‘ZIP Code‘ = col_double(),
##   ..   Family = col_double(),
##   ..   CCAvg = col_double(),
##   ..   Education = col_double(),
##   ..   Mortgage = col_double(),
##   ..   ‘Personal Loan‘ = col_double(),
##   ..   ‘Securities Account‘ = col_double(),
##   ..   ‘CD Account‘ = col_double(),
##   ..   Online = col_double(),
##   ..   CreditCard = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
summary(df$Personal.Loan)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   0.000   0.000   0.096   0.000   1.000
```

I loaded the Universal Bank.csv data set and cleaned the column named make.names() so they can be reflected in R syntax error-free. The str() function conveys the structure of the data set and summary(df$Personal.Loan) provides a gimpse of the target variable I'm predicting, which in this case is if a customer accepted a personal loan.

```r
# Keep a copy of raw data just in case
raw_df <- df
#Remove ID and ZIP.Code because they are not predictive
df <- subset(df,select= -c(ID, ZIP.Code))
#Convert Personal.Loan into a factor with levels "0" and "1"
df$Personal.Loan <- factor(df$Personal.Loan,levels = c(0,1), labels = c("0", "1"))
# Double-check
str(df$Personal.Loan)
```

```
##  Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
```

```r
table(df$Personal.Loan)
```

```
## 
##    0    1 
## 4520  480
```

I removed ID and Zip Code because they are identifiers and not features that are predictive. I also converted Personal.Loan into a factor with the levels of "0" and "1". This makes the classification model treat loan approvals as categorical outcomes rather than numeric values. Lastly, I confirmed the balance of the classes with str() and table().

```r
set.seed(42)
idx <- createDataPartition(df$Personal.Loan,p=0.60, list=FALSE)
train_df <- df[idx, ]
valid_df <- df[-idx, ]

# Check class balance in each split
prop.table(table(train_df$Personal.Loan))
```

```
## 
##     0     1 
## 0.904 0.096
```

```r
prop.table(table(valid_df$Personal.Loan))
```

```
##
##     0     1
## 0.904 0.096
```

I split the data set into a training set (60%) and a validation set (40%) using the stratified sampling method to keep the proportion of loan approvals. The proportions of "0" (no loan) and "1" (loan approved) are checked in this sequence/chunk to confirm the class balance of both subsets.

```r
# Build dummy encoder on TRAIN only
dummy_model <- caret::dummyVars(Personal.Loan ~ ., data = train_df, fullRank = FALSE)
# Create numeric feature matrices for train/validation
train_X <- as.data.frame(predict(dummy_model, newdata = train_df))
valid_X <- as.data.frame(predict(dummy_model, newdata = valid_df))
#Clean column names (Education_1, Education_2,...)
colnames(train_X) <- gsub("\\.", "_", colnames(train_X))
colnames(valid_X) <- gsub("\\.","_", colnames(valid_X))
#Targets
train_y <- train_df$Personal.Loan
valid_y <- valid_df$Personal.Loan
#Sanity checks (force output with print)
print(grep("^Education_", names(train_X), value=TRUE))
```

```
## character(0)
```

```r
print(dim(train_X)); print(dim(valid_X))
```

```
## [1] 3000   11
```

```
## [1] 2000   11
```

```r
print(table(train_y)); print(table(valid_y))
```

```
## train_y
##    0    1
## 2712  288
```

```
## valid_y
##    0    1
## 1808  192
```

I built a dummy encoder on training set to categorical variables. I transformed both validation and training sets into numeric predictor matrices. I also cleaned the column names to avoid discrepancies. I then defined the target variable separately from any predictors. I also ran sanity checks to confirm that Education columns exist, the dimensions of the datasets were correct and that there was a class balance in the target variable.

```
# Convert Education to factor
train_df$Education <- factor(train_df$Education, levels = c(1,2,3) )
valid_df$Education <- factor(valid_df$Education, levels = c(1,2,3))
# Rebuild dummy encoder on TRAIN ONLY
dummy_model <- caret::dummyVars(Personal.Loan ~ ., data = train_df, fullRank = FALSE )
# Build numeric feature matrices
train_X <- as.data.frame(predict(dummy_model, newdata = train_df))
valid_X <- as.data.frame(predict(dummy_model,newdata = valid_df))
#Clean column names
colnames(train_X) <- gsub("\\.","_", colnames(train_X))
colnames(valid_X) <- gsub("\\.","_", colnames(valid_X))
#Targets
train_y <- train_df$Personal.Loan
valid_y <- valid_df$Personal.Loan
#Sanity Checks
print(grep("^Education_", names(train_X), value=TRUE))
```

```
## [1] "Education_1" "Education_2" "Education_3"
```

```
print(dim(train_X)); print(dim(valid_X))
```

```
## [1] 3000   13
```

```
## [1] 2000   13
```

```
print(table(train_y)); print(table(valid_y))
```

```
## train_y
##    0    1
## 2712  288
```

```
## valid_y
##    0    1
## 1808  192
```

Converted the Education variable into a factor with levels 1,2,3 for consistency. Rebuilt the dummy encoder on the training set to avoid the leaking of data. I transformed the training/validation sets into numeric predictor matrices. I then cleaned up the column names again so they are accurate for modeling. I also redefined the target variables. Lastly performed sanity checks to verify Education 1, Education 2, Education 3, dimensions and class distribution in the target variable.

```
#Rebuild the validation matrix with the SAME dummy model
valid_X <- as.data.frame(predict(dummy_model, newdata = valid_df))
colnames(valid_X) <-gsub("\\.","_", colnames(valid_X))
#Confirm dimensions now match train_X
print(dim(train_X)) # should be 3000 x 13
```

```
## [1] 3000   13
```

```r
print(dim(valid_X)) # should be 2000 x 13
```

```
## [1] 2000   13
```

I reapplied the dummy variable encoding to the validation set using the same dummy model as training. I cleaned my column names for consistency. I also confirmed the validation feature matrix has the same structure as the training matrix (13 predictors)

```r
#Build on predictors only (exclude Personal.Loan)
predictor_train <-subset(train_df, select = -Personal.Loan)
predictor_valid <- subset(valid_df, select = -Personal.Loan)
dummy_model_X <- caret::dummyVars(~., data = predictor_train, fullRank = FALSE)
train_X <- as.data.frame(predict(dummy_model_X, newdata = predictor_train))
valid_X <- as.data.frame(predict(dummy_model_X, newdata = predictor_valid))
colnames(train_X) <- gsub("\\.","_", colnames(train_X))
colnames(valid_X) <- gsub("\\.","_", colnames(valid_X))
train_y <- train_df$Personal.Loan
valid_y <- valid_df$Personal.Loan
#Sanity Check
print(dim(train_X)); print(dim(valid_X))
```

```
## [1] 3000   13
```

```
## [1] 2000   13
```

```r
print(grep("^Education_", names(train_X), value = TRUE))
```

```
## [1] "Education_1" "Education_2" "Education_3"
```

Removed the target column (Personal.Loan) before encoding so only predictor variables are transformed. I used dummy Vars() to create dummy variables for categorical reasons. I applied the dummy model to train/validation sets to create feature matrices.

```r
#Train k=1 with standardization
knn_k1 <-caret::train(
x = train_X, y = train_y,
method = "knn",
preProcess = c("center","scale"),
tuneGrid = data.frame(k=1),
trControl = caret::trainControl(method = "none"),
metric = "Accuracy"
)

#Given customer (Education must be a factor with same levels 1/2/3)
new_customer_raw <- data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  Family = 2,
  CCAvg = 2,
  Education = factor(2, levels = c(1,2,3)),
  Mortgage = 0,
  Securities.Account = 0,
  CD.Account = 0,
  Online = 1,
  CreditCard = 1
  )
#Transform with the SAME dummy model (built on predictors only)
new_customer_X <- as.data.frame(predict(dummy_model_X,newdata=new_customer_raw))
colnames(new_customer_X) <- gsub("\\.","_",colnames(new_customer_X))
#Align columns with training matrix
missing_cols <- setdiff(colnames(train_X), colnames(new_customer_X))
for (mc in missing_cols) new_customer_X[[mc]] <- 0
new_customer_X <- new_customer_X[, colnames(train_X)]
#Predict class for k=1
k1_prediction <- predict(knn_k1, new_customer_X)
k1_prediction
```

```
## [1] 0
## Levels: 0 1
```

I trained a k-Nearest Neighbors classifier with k=1 using standardized predictors like center,scale. I also created a hypothetical new customer record with values for all predictors. I transformed this record with same dummy encoding used on the training data set to keep consistency throughout. I used the fitted model to predict whether the new customer would take a personal loan. k1_prediction shows the result (0= no loan, 1 = loan).

```
# Build on predictors only (exclude Personal.Loan)
predictor_train <- subset(train_df, select = -Personal.Loan)
predictor_valid <-subset(valid_df, select = -Personal.Loan)

# Dummy encoder on TRAIN predictors only
dummy_model_X <-caret::dummyVars(~.,data = predictor_train, fullRank = FALSE)

# Create numeric feature matrices from the predictor-only encoder
train_X <- as.data.frame(predict(dummy_model_X, newdata = predictor_train))
valid_X <- as.data.frame(predict(dummy_model_X, newdata = predictor_valid))
# Clean column names
colnames (train_X) <- gsub("\\.","_", colnames(train_X))
colnames(valid_X) <- gsub("\\.","_", colnames(valid_X))
#Targets (unchanged)
train_y <- train_df$Personal.Loan
valid_y <- valid_df$Personal.Loan

#Sanity Checks
print(dim(train_X)); print(dim(valid_X))
```

```
## [1] 3000   13
```

```
## [1] 2000   13
```

```
print(grep("^Education_", names(train_X), value = TRUE))
```

```
## [1] "Education_1" "Education_2" "Education_3"
```

Removed the target (Personal.Loan) so only predictors are encoded. I fit a predictor-only dummy encoder on the training predictors. I applied the encoder to both training and validation predictors to get numeric feature matrices. Kept train_y/valid_y as target vectors.Performed sanity checks to confirm matrices dimensions and the presence of Education 1/2/3.

```
# Train k=1 with standardization
knn_k1 <- caret::train(
  x = train_X,
  y = train_y,
  method = "knn",
  preProcess = c("center","scale"),
  tuneGrid = data.frame(k=1),
  trControl = caret::trainControl(method = "none"),
  metric = "Accuracy"
)
# Given Customer
new_customer_raw <- data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  Family = 2,
  CCAvg = 2,
  Education = factor(2, levels = c(1,2,3)),
  Mortgage = 0,
  Securities.Account = 0,
  CD.Account = 0,
  Online = 1,
  CreditCard = 1
  )
# Transform with predictor-only dummy model
new_customer_X <- as.data.frame(predict(dummy_model_X, newdata = new_customer_raw))
colnames(new_customer_X) <-gsub("\\.","_", colnames(new_customer_X))
#Align columns with training matrix
missing_cols <- setdiff(colnames(train_X), colnames(new_customer_X))
for (mc in missing_cols) new_customer_X[[mc]] <-0
new_customer_X <- new_customer_X[, colnames(train_X), drop = FALSE]
#Predict class
k1_prediction <- predict(knn_k1, new_customer_X)
k1_prediction
```

```
## [1] 0
## Levels: 0 1
```

Trained a k-NN model with k=1 using predictors. Built a single new customer row with all predictors to make sure Education is a factor with levels 1,2,3 to match training. Aligned columns to train_X so the model actually accepts the input. Predicted the customer's class with the trained model, which is k1_prediction prints 0 or 1, which would be no loan vs. loan.

```r
# Try odd k from 1 to 31; bootstrap resampling keeps it simple

grid <- expand.grid(k = seq(1,31, by =2))

ctrl_cv <- caret::trainControl(method = "boot", number = 25)

set.seed(42)
knn_tuned <- caret::train(
x = train_X,
y = train_y,
method = "knn",
preProcess = c("center","scale"),
tuneGrid = grid,
trControl = ctrl_cv,
metric = "Accuracy"
)
knn_tuned    #shows the accuracy results for each k
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   13 predictor
##    2 classes: ’0’, ’1’
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9555110  0.7146209
##    3  0.9494612  0.6630523
##    5  0.9490174  0.6484904
##    7  0.9499975  0.6438568
##    9  0.9485029  0.6289884
##   11  0.9478940  0.6188275
##   13  0.9473394  0.6103692
##   15  0.9464368  0.5991102
##   17  0.9452317  0.5839829
##   19  0.9442958  0.5746793
##   21  0.9430526  0.5593454
##   23  0.9421451  0.5512865
##   25  0.9414679  0.5414406
##   27  0.9410261  0.5356584
##   29  0.9401167  0.5260284
##   31  0.9397534  0.5201584
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```r
best_k <- knn_tuned$bestTune$k
best_k       # prints the chosen best k
```

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

Defined a tuning grid of odd values of k. Used bootstrap re-sampling as the cross-validation method. Trained k-NN models across the grid with standardization. The caret automatically reported accuracy for each k and selects the best k, with the highest accuracy.

```
# Predictions on validation holdout
valid_pred <- predict(knn_tuned, newdata = valid_X)

# Confusion matrix; success class is "1"
cm_valid <- caret::confusionMatrix(valid_pred, valid_y, positive = "1")
cm_valid
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1788   64
##          1   20  128
##
##                Accuracy : 0.958
##                  95% CI : (0.9483, 0.9664)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7304
##
##  Mcnemar’s Test P-Value : 2.71e-06
##
##             Sensitivity : 0.6667
##             Specificity : 0.9889
##          Pos Pred Value : 0.8649
##          Neg Pred Value : 0.9654
##              Prevalence : 0.0960
##          Detection Rate : 0.0640
##    Detection Prevalence : 0.0740
##       Balanced Accuracy : 0.8278
##
##        ’Positive’ Class : 1
##
```

Used the tuned k-NN model to generate predictions. Stored those predictions in valid_X. Compared predictions against true labels with a confusion matrix. Set the positive class as “1”, which means that a customer accepts a loan. The confusion matrix provided accuracy, recall, specificity, kappa.

```
# Given customer (Education must match factor levels used earlier)
new_customer_raw <- data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  Family = 2,
  CCAvg = 2,
  Education = factor(2, levels = c(1,2,3)), # Matches train levels
```

```r
  Mortgage = 0,
  Securities.Account = 0,
  CD.Account = 0,
  Online = 1,
  CreditCard = 1
)
# Transform with the SAME dummy model (predictor-only)
new_customer_X <- as.data.frame(predict(dummy_model_X, newdata = new_customer_raw))
colnames(new_customer_X) <-gsub("\\.","_", colnames(new_customer_X))
# Align columns with training matrix
missing_cols <- setdiff(colnames(train_X), colnames(new_customer_X))
for (mc in missing_cols) new_customer_X[[mc]] <- 0
new_customer_X <- new_customer_X[, colnames(train_X), drop = FALSE]
# Predict class using tuned k-NN model
best_pred <- predict(knn_tuned, new_customer_X)
best_pred
```

```
## [1] 0
## Levels: 0 1
```

Made a new customer profile with all predictor fields matching the training data. Used the same dummy encoder to transform that row to make sure it matches training features. Applied the tuned k-NN model to classify that customer. The output best prediction (best_pred) provides the predicted loan approval class (0 = not approved, 1 = approved)

```r
set.seed(42)

# 50% train
idx50 <- createDataPartition(df$Personal.Loan, p= 0.50, list = FALSE)
train2 <- df[idx50, ]
rest <- df[-idx50, ]

# Of the left over 50%, take 60% as validation (=> 30% overall, 40% as test (=> 20% overall)
idx_val <- createDataPartition(rest$Personal.Loan, p = 0.60, list = FALSE)
valid2 <- rest[idx_val, ]
test2 <- rest[-idx_val,]
# Make sure Education is a factor in all three
train2$Education <- factor(train2$Education, levels = c(1,2,3))
valid2$Education <- factor(valid2$Education, levels = c(1,2,3))
test2$Education <- factor(test2$Education, levels = c(1,2,3))

# Predictor-only dummy encoder on the NEW train2
pred_train2 <- subset(train2, select = -Personal.Loan)
pred_valid2 <- subset(valid2, select = -Personal.Loan)
pred_test2 <- subset(test2, select = -Personal.Loan)

dummy2 <-caret::dummyVars(~ ., data = pred_train2, fullRank = FALSE)

X_tr2 <- as.data.frame(predict(dummy2, newdata = pred_train2)); colnames(X_tr2) <- gsub("\\.","_", colna
X_va2 <- as.data.frame(predict(dummy2, newdata = pred_valid2)); colnames(X_va2) <- gsub("\\.","_", colna
X_te2 <- as.data.frame(predict(dummy2, newdata = pred_test2)); colnames (X_te2) <- gsub("\\.","_", colna

y_tr2 <- train2$Personal.Loan
```

```r
y_va2 <- valid2$Personal.Loan
y_te2 <- test2$Personal.Loan

# Fit with fixed best_k from Part 2 (your knn_tuned$bestTune$k, which was 1)
knn_best_fixed <- caret::train(
x = X_tr2, y = y_tr2,
method = "knn",
preProcess = c("center", "scale"),
tuneGrid = data.frame(k = best_k),
trControl = caret::trainControl(method = "none"),
metric = "Accuracy"
)

# Confusion matrics on train/valid/test
pred_tr2 <- predict(knn_best_fixed, X_tr2)
pred_va2 <- predict (knn_best_fixed, X_va2)
pred_te2 <- predict(knn_best_fixed, X_te2)

cm_tr2 <- caret::confusionMatrix(pred_tr2, y_tr2, positive = "1")
cm_va2 <- caret::confusionMatrix(pred_va2, y_va2, positive = "1")
cm_te2 <- caret::confusionMatrix(pred_te2, y_te2, positive = "1")

cm_tr2; cm_va2; cm_te2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar’s Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.096
##          Detection Rate : 0.096
##    Detection Prevalence : 0.096
##       Balanced Accuracy : 1.000
##
##        ’Positive’ Class : 1
##
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    0    1
##          0 1335   48
##          1   21   96
##
##                Accuracy : 0.954
##                  95% CI : (0.9421, 0.964)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 3.461e-13
##
##                   Kappa : 0.7107
##
##  Mcnemar's Test P-Value : 0.001748
##
##             Sensitivity : 0.6667
##             Specificity : 0.9845
##          Pos Pred Value : 0.8205
##          Neg Pred Value : 0.9653
##              Prevalence : 0.0960
##          Detection Rate : 0.0640
##    Detection Prevalence : 0.0780
##       Balanced Accuracy : 0.8256
##
##        'Positive' Class : 1
##
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 893  27
##          1  11  69
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 2.14e-12
##
##                   Kappa : 0.7634
##
##  Mcnemar's Test P-Value : 0.01496
##
##             Sensitivity : 0.7188
##             Specificity : 0.9878
##          Pos Pred Value : 0.8625
##          Neg Pred Value : 0.9707
##              Prevalence : 0.0960
##          Detection Rate : 0.0690
##    Detection Prevalence : 0.0800
##       Balanced Accuracy : 0.8533
##
##        'Positive' Class : 1
##
```

Re partitioned the data into 50% train, 30% validation, 20% test using stratified splits to keep the 9.6% positive rate stable. Converted Education to a factor (levels 1-3) in all three partitions. Training = Accuracy was 1.000. Possible overfitting Validation = Accuracy ~ 0.954, sensitivity ~ 0.667, specificity ~0.985. Overall strong, but recall on positives seems lower, which is common because of the data not being balanced. Test Accuracy ~ 0.962, sensitivity ~ 0.719, specificity ~0.988, slightly better than validation.