FINAL PROJECT

Sebastián Ballesteros

2023-03-20

Contents

Project Guidelines

- Load and clean the data as needed. Divide your dataset into test and train sets.
- Build three predictive models to predict which calls will be successful using the following algorithms:
  - Logistic Regression optimized with backward stepwise regression
  - SVM - you are asked to figure out which Kernel best fits our data
  - Decision Tree - you are asked to figure out whether a cost matrix would be appropriate for this DT.
- We will use a novel prediction combination methodology. We will consider a prediction "1" only if ALL the three models above predict "1". Combine the three predictions from the three models above using this methodology to come to a final prediction for your test data.

- Create a Confusion Matrix for the final combined prediction. Does the combined prediction meet the requirements of the client? If you are the Call Center Manager, would you consider this prediction model useful?
- Finally, you, the call center manager, has just received a list of 10 people to be called today. You need to decide which of these 10 people to call. Use the prediction methodology outlined above to decide which of these 10 people to call.
- Feel free to make any additional assumptions you may need. You are the boss here and you are asked to decide the best way forward.

loading libraries

```
library(class)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ggplot2)
library(gmodels)
library(neuralnet)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
library(stringr)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(C50)
```

Data Cleaning

```
#Downloading and Prepping the Data
tele <- read.csv("tele.csv", stringsAsFactors = TRUE)

#We are deleting the "duration" variable because it is an after the fact measurement. We only should be
tele$duration <- NULL

# Deleting the column X
tele$X <- NULL

# Changing pdays to a dummy and deleting pdays
tele$pdaysdummy <- ifelse(tele$pdays == 999, 0, 1)
tele$pdays <- NULL
```

Getting Data Ready for Analysis

```
# Using model.matrix to convert all the factors to dummy variables
# We are converting all of the factors into dummy variables as the input into knn has to be numeric

telemm <- as.data.frame(model.matrix(~.-1,tele))

# Randomize the rows in the data (shuffling the rows)
set.seed(12345)
tele_random <- telemm[sample(nrow(telemm)),]

#Normalize the data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# we are going to normalize everything
tele_norm <- as.data.frame(lapply(tele_random, normalize))
```

Getting Train and Test Samples

```
set.seed(12345)
test_set <- sample(1:nrow(tele_norm), nrow(tele_norm)/2)

tele_train <- tele_norm[-test_set,]
tele_test  <- tele_norm[test_set,]
```

Logit Model

Backwards Regression

```
logit_model<- step(glm(yyes ~ . ,data=tele_train, family = "binomial"),
                   direction = c("backward"))

saveRDS(logit_model,"logit_model")
```

Summary Logit

```
logit_model <- readRDS("logit_model")

summary(logit_model)
```

```
##
## Call:
## glm(formula = yyes ~ jobadmin. + jobblue.collar + jobentrepreneur +
##     jobhousemaid + jobservices + jobtechnician + jobunemployed +
##     defaultunknown + contacttelephone + monthaug + monthjun +
##     monthmar + monthmay + monthnov + day_of_weekmon + campaign +
##     poutcomenonexistent + poutcomesuccess + emp.var.rate + cons.price.idx +
##     cons.conf.idx + euribor3m + pdaysdummy, family = "binomial",
##     data = tele_train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9953  -0.3837  -0.3158  -0.2689   2.8484
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.26870    0.13417  -9.456  < 2e-16 ***
## jobadmin.           -0.14022    0.07012  -2.000 0.045534 *
## jobblue.collar      -0.32357    0.08302  -3.898 9.72e-05 ***
## jobentrepreneur     -0.36580    0.16275  -2.248 0.024600 *
## jobhousemaid        -0.26777    0.16795  -1.594 0.110853
## jobservices         -0.37923    0.10700  -3.544 0.000394 ***
## jobtechnician       -0.15380    0.08113  -1.896 0.057986 .
## jobunemployed       -0.24587    0.16412  -1.498 0.134101
## defaultunknown      -0.34374    0.08229  -4.177 2.95e-05 ***
## contacttelephone    -0.56054    0.08399  -6.674 2.49e-11 ***
## monthaug             0.35853    0.10740   3.338 0.000843 ***
## monthjun            -0.53659    0.10593  -5.066 4.07e-07 ***
## monthmar             1.22214    0.14188   8.614  < 2e-16 ***
## monthmay            -0.50605    0.07521  -6.729 1.71e-11 ***
## monthnov            -0.44632    0.10088  -4.424 9.68e-06 ***
## day_of_weekmon      -0.21377    0.06338  -3.373 0.000744 ***
## campaign            -1.17324    0.66192  -1.772 0.076318 .
## poutcomenonexistent  0.42172    0.07880   5.352 8.69e-08 ***
## poutcomesuccess      0.78292    0.26495   2.955 0.003127 **
## emp.var.rate        -5.69445    0.64338  -8.851  < 2e-16 ***
## cons.price.idx       3.94268    0.32332  12.194  < 2e-16 ***
## cons.conf.idx        0.41821    0.14662   2.852 0.004341 **
## euribor3m            1.39188    0.44254   3.145 0.001660 **
## pdaysdummy           0.92492    0.26143   3.538 0.000403 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 14466  on 20593  degrees of freedom
## Residual deviance: 11345  on 20570  degrees of freedom
## AIC: 11393
##
## Number of Fisher Scoring iterations: 6
```

Logit Model Results

An activation threshold of 0.21 will be used from now on as it has proven to maximize profits and maximize kappa from other homeworks and assignmets done on this dataset

```
logit_pred<-predict(logit_model, newdata = tele_test , type = "response")

pred<-as.factor(ifelse(logit_pred>=0.21,1,0))
test<-as.factor(tele_test$yyes)

confusionMatrix(pred,test, positive = "1")
```

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

```
##           Reference
## Prediction     0     1
##          0 16578  1036
##          1  1688  1292
##
##                Accuracy : 0.8677
##                  95% CI : (0.863, 0.8723)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4122
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.55498
##             Specificity : 0.90759
##          Pos Pred Value : 0.43356
##          Neg Pred Value : 0.94118
##              Prevalence : 0.11304
##          Detection Rate : 0.06274
##    Detection Prevalence : 0.14470
##       Balanced Accuracy : 0.73129
##
##        'Positive' Class : 1
##
```

Support Vector Machines

Training an SVM with every kernel

```
kernels <- c("rbfdot", "polydot", "tanhdot", "vanilladot", "laplacedot", "besseldot", "anovadot", "spli
svms <- c()

for (i in kernels){
  svm_model <- ksvm(as.factor(yyes) ~ ., data = tele_train, kernel = i)
  svms <- append(svms,svm_model)
}

names(svms) <- kernels

saveRDS(svms,"svms")
```

Predicting with SVM's

```
SVMs <- readRDS("svms")

SVM_df <- data.frame(rbfdot = predict(SVMs$rbfdot,tele_test))

SVM_df$polydot    <- predict(SVMs$polydot,tele_test)
SVM_df$tanhdot    <- predict(SVMs$tanhdot,tele_test)
SVM_df$vanilladot <- predict(SVMs$vanilladot,tele_test)
SVM_df$laplacedot <- predict(SVMs$laplacedot,tele_test)
SVM_df$besseldot  <- predict(SVMs$besseldot,tele_test)
```

```
SVM_df$anovadot   <- predict(SVMs$anovadot,tele_test)
SVM_df$splinedot  <- predict(SVMs$splinedot,tele_test)

saveRDS(SVM_df, "svm_pred")
```

SVM's performance

```
svms     <- readRDS("svms")
svm_pred <- readRDS("svm_pred")
```

rbfdot

```
confusionMatrix(as.factor(svm_pred$rbfdot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 18066  1883
##          1   200   445
##
##                Accuracy : 0.8989
##                  95% CI : (0.8947, 0.9029)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 2.328e-08
##
##                   Kappa : 0.2632
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9891
##             Specificity : 0.1912
##          Pos Pred Value : 0.9056
##          Neg Pred Value : 0.6899
##              Prevalence : 0.8870
##          Detection Rate : 0.8772
##    Detection Prevalence : 0.9687
##       Balanced Accuracy : 0.5901
##
##        'Positive' Class : 0
##
```

polydot

```
confusionMatrix(as.factor(svm_pred$polydot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 18000  1855
##          1   266   473
##
```

```
##                Accuracy : 0.897
##                  95% CI : (0.8928, 0.9011)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 2.087e-06
##
##                   Kappa : 0.2686
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9854
##             Specificity : 0.2032
##          Pos Pred Value : 0.9066
##          Neg Pred Value : 0.6401
##              Prevalence : 0.8870
##          Detection Rate : 0.8740
##    Detection Prevalence : 0.9641
##       Balanced Accuracy : 0.5943
##
##        'Positive' Class : 0
##
```

tanhdot

```
confusionMatrix(as.factor(svm_pred$tanhdot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16419  1875
##          1  1847   453
##
##                Accuracy : 0.8193
##                  95% CI : (0.8139, 0.8245)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.094
##
##  Mcnemar's Test P-Value : 0.6581
##
##             Sensitivity : 0.8989
##             Specificity : 0.1946
##          Pos Pred Value : 0.8975
##          Neg Pred Value : 0.1970
##              Prevalence : 0.8870
##          Detection Rate : 0.7973
##    Detection Prevalence : 0.8883
##       Balanced Accuracy : 0.5467
##
##        'Positive' Class : 0
##
```

vanilladot

```
confusionMatrix(as.factor(svm_pred$vanilladot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 18000  1855
##          1   266   473
##
##                Accuracy : 0.897
##                  95% CI : (0.8928, 0.9011)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 2.087e-06
##
##                   Kappa : 0.2686
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9854
##             Specificity : 0.2032
##          Pos Pred Value : 0.9066
##          Neg Pred Value : 0.6401
##              Prevalence : 0.8870
##          Detection Rate : 0.8740
##    Detection Prevalence : 0.9641
##       Balanced Accuracy : 0.5943
##
##        'Positive' Class : 0
##
```

laplacedot

```
confusionMatrix(as.factor(svm_pred$laplacedot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 18023  1881
##          1   243   447
##
##                Accuracy : 0.8969
##                  95% CI : (0.8926, 0.901)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 2.884e-06
##
##                   Kappa : 0.2579
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9867
##             Specificity : 0.1920
```

```
##          Pos Pred Value : 0.9055
##          Neg Pred Value : 0.6478
##              Prevalence : 0.8870
##          Detection Rate : 0.8752
##    Detection Prevalence : 0.9665
##       Balanced Accuracy : 0.5894
##
##        'Positive' Class : 0
##
```

besseldot

```
confusionMatrix(as.factor(svm_pred$besseldot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16156  2264
##          1  2110    64
##
##                Accuracy : 0.7876
##                  95% CI : (0.782, 0.7932)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : -0.0906
##
##  Mcnemar's Test P-Value : 0.0207
##
##             Sensitivity : 0.88448
##             Specificity : 0.02749
##          Pos Pred Value : 0.87709
##          Neg Pred Value : 0.02944
##              Prevalence : 0.88696
##          Detection Rate : 0.78450
##    Detection Prevalence : 0.89444
##       Balanced Accuracy : 0.45599
##
##        'Positive' Class : 0
##
```

anovadot

```
confusionMatrix(as.factor(svm_pred$anovadot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17985  1832
##          1   281   496
##
##                Accuracy : 0.8974
```

```
##                  95% CI : (0.8932, 0.9015)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 8.614e-07
##
##                   Kappa : 0.2787
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9846
##             Specificity : 0.2131
##          Pos Pred Value : 0.9076
##          Neg Pred Value : 0.6384
##              Prevalence : 0.8870
##          Detection Rate : 0.8733
##    Detection Prevalence : 0.9623
##       Balanced Accuracy : 0.5988
##
##        'Positive' Class : 0
##
```

splinedot

```
confusionMatrix(as.factor(svm_pred$splinedot), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15345  1714
##          1  2921   614
##
##                Accuracy : 0.7749
##                  95% CI : (0.7692, 0.7806)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0847
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8401
##             Specificity : 0.2637
##          Pos Pred Value : 0.8995
##          Neg Pred Value : 0.1737
##              Prevalence : 0.8870
##          Detection Rate : 0.7451
##    Detection Prevalence : 0.8283
##       Balanced Accuracy : 0.5519
##
##        'Positive' Class : 0
##
```

Decision Tree

The first model built is the Decision Tree. The C5.0 function from the C50 library is used to build the model. The formula used in this case is as.factor(yyes) ~ . which means that the model will use all the other columns to predict the "yyes" column, which is the outcome of interest.

DT

```
DT_model <- C5.0(as.factor(yyes) ~ ., data = tele_train)
```

Decision Tree results

```
DT_pred <- as.numeric(predict(DT_model, tele_test))-1

confusionMatrix(as.factor(DT_pred),as.factor(tele_test$yyes), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17652  1555
##          1   614   773
##
##                Accuracy : 0.8947
##                  95% CI : (0.8904, 0.8988)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 0.0002151
##
##                   Kappa : 0.3623
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.33204
##             Specificity : 0.96639
##          Pos Pred Value : 0.55732
##          Neg Pred Value : 0.91904
##              Prevalence : 0.11304
##          Detection Rate : 0.03754
##    Detection Prevalence : 0.06735
##       Balanced Accuracy : 0.64922
##
##        'Positive' Class : 1
##
```

DT Cost Matrix

```
#Given the original prompt for the project, and the fact that the tune function for tuning does not run

cost_matrix <- matrix(c(0,-1,0,5),ncol = 2)

DT_errorcost <- C5.0(as.factor(yyes) ~ ., data = tele_train, costs = cost_matrix)
```

```
## Warning: no dimnames were given for the cost matrix; the factor levels will be
```

```
## used
```

Cost Matrix DT results

```
errorcost_pred <- predict(DT_errorcost, tele_test)

confusionMatrix(as.factor(errorcost_pred), as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17652  1555
##          1   614   773
##
##                Accuracy : 0.8947
##                  95% CI : (0.8904, 0.8988)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 0.0002151
##
##                   Kappa : 0.3623
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9664
##             Specificity : 0.3320
##          Pos Pred Value : 0.9190
##          Neg Pred Value : 0.5573
##              Prevalence : 0.8870
##          Detection Rate : 0.8571
##    Detection Prevalence : 0.9327
##       Balanced Accuracy : 0.6492
##
##        'Positive' Class : 0
##
```

Final Prediction

```
combined_pred <- ifelse((ifelse(logit_pred>0.21,1,0) * DT_pred * as.numeric(svm_pred$vanilladot)-1) ==
confusionMatrix(as.factor(combined_pred),as.factor(tele_test$yyes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 18090  1905
##          1   176   423
##
##                Accuracy : 0.899
##                  95% CI : (0.8948, 0.903)
##     No Information Rate : 0.887
##     P-Value [Acc > NIR] : 1.8e-08
##
```

```
##                   Kappa : 0.2545
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9904
##             Specificity : 0.1817
##          Pos Pred Value : 0.9047
##          Neg Pred Value : 0.7062
##              Prevalence : 0.8870
##          Detection Rate : 0.8784
##    Detection Prevalence : 0.9709
##       Balanced Accuracy : 0.5860
##
##        'Positive' Class : 0
##
```

10 Person List

Selecting 10 people

```
set.seed(12345)
today_list <- tele_norm[sample(1:nrow(tele_norm),10),]
today_list$yyes
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

Ensamble prediction

```
pred_today_logit <- ifelse(predict(logit_model, today_list, type = "response")>0.21,1,0)
pred_today_svm   <- predict(svms$vanilladot, today_list)
pred_today_DT    <- predict(DT_model, today_list)
```

Who to call

call people with position n in the list as indicated below

```
(pred_today_logit * (as.numeric(pred_today_DT)-1) * (as.numeric(pred_today_svm)-1)) == 1
```

```
## 10904 33373  9986    75 30311 26378 21415 25548 35656 31756
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
which((pred_today_logit * (as.numeric(pred_today_DT)-1) * (as.numeric(pred_today_svm)-1)) == 1)
```

```
## named integer(0)
```

In this case we dont call any of the 10 randomly sampled people

Conclusion

After running all three level one models, It was evident that the best performing SVM was the Anovadot kernel, but due to the small number of observations, and debugging problems with prediction I opted to run Vanilladot for the 10 person list due to time constraints. Backwards Step-wise regression converged at 23 variables, and the Decision tree was better off without the cost matrix, as kappa did not change from one model to the other. The best model overall, including level one models was the logit with a Kappa of 0.4122.

When it comes to combining models, there are various factors to consider beyond just model accuracy. While the approach taken in this project may not have been optimal, there may be better ways to combine models that can improve prediction results. It's important to evaluate the strengths and weaknesses of each model and determine the best way to leverage their strengths.

Moreover, it's important to consider whether it's even necessary to combine models, it may be more effective to rely on a single model rather than combining multiple models given the guidelines of this project. Depending on the uses of this model one could be favored over another. If you are looking for robustness/never needing to train data, SVM's might be the best. If you are looking to interpret and find relationships in the data, then regression and decision tree might be better. In my opinion either one is more usefull and better at predicting than the combined model given that we only activate if all three models suggest a positive.

In terms of activation point, there are various approaches that can be used to optimize performance. While reducing false positives is important, which is what the combined model is very good at, it's equally crucial to avoid missing out on true positives. One approach I have used in the past is to use a cost-based approach that takes into account the costs of false positives and false negatives. However, this cost function will depend to the specific call center and the costs associated with different types of errors. as well as the revenue generated by successful calls

It's also important to keep in mind that model accuracy is not always the most important factor to consider. Depending on the specific use case, there may be other factors to prioritize such as profits, and variability of cashflows.

I would consider that this model would be useful to a call center manager because there is an improvement in prediction than picking at random. The increased revenues are not only significant but given the cost function in our last project, make the difference between profitability and bankruptcy. That isnt to say that this model leaves a lot to be desired out of the data, and in terms of maximizing prediction accuracy.