~~~
---
title: "final"
author: "Xinyang Zhou"
date: "4/21/2023"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```


```{r, echo=FALSE, error=FALSE}
library(MASS)
library(class)
library(gmodels)
library(caret)
library(gmodels)
library(kernlab)
library(C50)
```


# Logistic Regression

## Data cleaning
```{r}
tele_logistic <- read.csv("tele.csv")
tele_logistic$duration <- NULL
tele_logistic$X <- NULL
tele_logistic$campaign <- NULL
tele_logistic$default <- NULL

#Making the y a dummy variable
tele_logistic$y = ifelse(tele_logistic$y == "yes", 1, 0)

#Factoring out the columns with characters
tele_logistic$job <- as.factor(tele_logistic$job)
tele_logistic$marital <- as.factor(tele_logistic$marital)
tele_logistic$education <- as.factor(tele_logistic$education)
tele_logistic$previous <- as.factor(tele_logistic$previous)
tele_logistic$pdays <- as.factor(tele_logistic$pdays)
tele_logistic$housing <- as.factor(tele_logistic$housing)
tele_logistic$loan <- as.factor(tele_logistic$loan)
tele_logistic$contact <- as.factor(tele_logistic$contact)
tele_logistic$month <- as.factor(tele_logistic$month)
tele_logistic$day_of_week <- as.factor(tele_logistic$day_of_week)
tele_logistic$poutcome <- as.factor(tele_logistic$poutcome)
tele_logistic$nr.employed <- as.factor(tele_logistic$nr.employed)

tele_logistic$pdaysdummy <- ifelse(tele_logistic$pdays == 999, 0, 1)
tele_logistic$pdays <- NULL
tele_logistic$previous = NULL
```

## Getting Train and Test Samples
```{r}
set.seed(12345)
test_set_logistic <- sample(1:nrow(tele_logistic), 0.5*nrow(tele_logistic))
tele_train_logistic <- tele_logistic[-test_set_logistic, ]
tele_test_logistic <- tele_logistic[test_set_logistic, ]

tele_train_labels_logistic <- tele_logistic[-test_set_logistic, "y"]
tele_test_labels_logistic <- tele_logistic[test_set_logistic, "y"]
~~~

~~~~
```

## Building Initial Logistics Regression Model
```{r, cache=TRUE}
logistic <- glm(y ~ ., data = tele_train_logistic, family="binomial")
```

We want to build an optimized logistic regression model using backward regression.

## Logistic Regression Optimized with Backward Stepwise Regression
```{r}
optimized_model <- stepAIC(logistic, direction = "backward")
```

Our stepwise backward elimination result is shown above. After implementing this algorithm
to our logistic regression, we have 10 predictors, which is a lot less than our original
model. This method improve the model's performance by removing irrelevant or redundant
variables from the model.

## Prediction with Optimized Logistics Regression Model
```{r, cache=TRUE}
prediction = predict(optimized_model, newdata = tele_test_logistic)
logistic_predictions <- ifelse(prediction >= 0.5, "1", "0")
CrossTable(logistic_predictions, tele_test_labels_logistic)
confusionMatrix(as.factor(logistic_predictions), as.factor(tele_test_labels_logistic),
positive ="1")
```

Our Logistic Regression optimized with backward stepwise regression gives us an accuracy
of 0.8985 and Kappa 0.2375. Then we want to try Support Vector Machines (SVM) and Decision
Tree models.

## Downloading and Prepping the Data

```{r}
#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 using variables that we know before the call
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

```{r}
# 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

```{r}
# Selects 10000 random rows for test data
set.seed(12345)
test_set <- sample(1:nrow(tele_norm), 0.5*nrow(tele_norm))
# Depending on R-version and computer, different rows may be selected.
# If that happens, results are different.

# Create a train set and test set
#First the predictors - all columns except the yyes column
tele_train <- tele_norm[-test_set, -match("yyes",names(tele_norm))]
tele_test <- tele_norm[test_set, -match("yyes",names(tele_norm))]

#Now the response (aka Labels) - only the yyes column
tele_train_labels <- tele_norm[-test_set, "yyes"]
tele_test_labels <- tele_norm[test_set, "yyes"]

```

# SVM

```{r}
set.seed(12345)
simple_model = ksvm(tele_train_labels ~ ., data = tele_train, kernel = "rbfdot")
simplepred = predict(simple_model, tele_test)
svm_prediction <- ifelse(simplepred >= 0.5, 1, 0)
CrossTable(svm_prediction, tele_test_labels)
confusionMatrix(as.factor(svm_prediction), as.factor(tele_test_labels), positive ="1")
```

The Kappa of this SVM model is 0.2528, which is not optimal. We want to find out the best
kernel for our dataset.

```{r}
set.seed(12345)
simple_model2 = ksvm(tele_train_labels ~ ., data = tele_train, kernel = "polydot")
simplepred2 = predict(simple_model2, tele_test)
svm_prediction2 <- ifelse(simplepred2 >= 0.5, 1, 0)
CrossTable(svm_prediction2, tele_test_labels)
confusionMatrix(as.factor(svm_prediction2), as.factor(tele_test_labels), positive ="1")
```

```{r}
set.seed(12345)
simple_model3 = ksvm(tele_train_labels ~ ., data = tele_train, kernel = "laplacedot")
simplepred3 = predict(simple_model3, tele_test)
svm_prediction3 <- ifelse(simplepred3 >= 0.5, 1, 0)
CrossTable(svm_prediction3, tele_test_labels)
confusionMatrix(as.factor(svm_prediction3), as.factor(tele_test_labels), positive ="1")
```


# Best SVM

```{r}
set.seed(12345)
improved_simple_model = ksvm(tele_train_labels ~ ., data = tele_train, kernel =
"vanilladot")
~~~

~~~
improved_simplepred = predict(improved_simple_model, tele_test)
improved_svm_prediction <- ifelse(improved_simplepred >= 0.5, 1, 0)
CrossTable(improved_svm_prediction, tele_test_labels)
confusionMatrix(as.factor(improved_svm_prediction), as.factor(tele_test_labels), positive
="1")
```

After trying all the kernels, the prediction model with kernel "vanilladot" performs the
best as Kappa is the biggest among all. Here, we have a Kappa of 0.2686. Hence, we will
use this as our kernel (Note that the outcome of "polydot" is exactly the same as
"vanilladot").


# Decision Tree Using Combined Predictions
```{r}
set.seed(12345)
tree = C5.0(as.factor(tele_train_labels)~., data = tele_train)
tree_pred = predict(tree, tele_test)
confusionMatrix(as.factor(tree_pred), as.factor(tele_test_labels))
```

We have a Kappa of 0.3623. We want to check whether a cost matrix would be appropriate for
this decision tree.

We want to build the error cost matrix as above because we will have no cost if the model
give us correct predictions. If the prediction is 0 and the truth is 1, which means we do
not call such client and this client will accept the deal if we do, we loss 6 - 1 = 5
dollars as it is the revenue we could have earned. Similarly, if the prediction is 1 and
the truth is 0, which means we call the client but the client does not accept the deal, we
loss 1 dollar as it is the cost of calling.

# Improved Decision Tree

```{r}
set.seed(12345)
error_cost = matrix(c(0,1,5,0), nrow = 2)
improved_tree = C5.0(as.factor(tele_train_labels) ~., data = tele_train, cost =
error_cost)
improved_tree_prediction = predict(improved_tree, tele_test)
CrossTable(improved_tree_prediction, tele_test_labels)
confusionMatrix(as.factor(improved_tree_prediction), as.factor(tele_test_labels))
```

We have a Kappa 0.3886, which is larger than the Kappa in our previous decision tree
model. Hence, we will want to proceed using this decision tree model in reality. In
addition, this model incorporate the business problem we are interested in.

# Combined Model
```{r}
final = as.numeric(logistic_predictions) + as.numeric(improved_svm_prediction) +
as.numeric(improved_tree_prediction)
final_prediction = ifelse(final==3, "1", "0")
confusionMatrix(as.factor(final_prediction), as.factor(tele_test_labels))
```

Let us assume the requirement is less than 5% False Positive rate and at least 90% total
accuracy rate.

False Positive Rate = 272 / (272+17994) = 0.01489 < 0.05.

Our False Positive Rate is extremely low and it is lower than 5%. However, even though the
accuracy rate is above 89%, it is still not 90%. Therefore, our combined model did not
fulfill the requirements.

Our Kappa is 0.272 here, which is lower than the decision tree model with cost matrix, but
~~~

higher than optimized logistic regression or optimized SVM models. In my point of view, the nature of prediction in our final model is a little bit arbitrary, as the only way to get "1" is when all three models agree and predict "1"s. Under such conditions, we are conservative in giving "1". In other words, unless we are sure, the final prediction will be "0".

Putting this into our business problem, we are conservative to predict a successful call. Rather, unless we are sure about it, we will predict such a client will refuse our offer.s

Therefore, we could expect to have a significant increase in cases where our prediction is negative but actually the truth is positive, indicating a client would accept our offer if we make the call; we missed the chance because we are conservative on predicting a client will accept the offer.

On the other hand, making such a strict bar did not destroy our model. The Kappa of 0.272 is still acceptable. The result indicates that in most cases, our three individual models agree with each other. Therefore, this model is useful for me if I'm the call center manager.

# Final Predictions

~~~
```{r, error=FALSE}
set.seed(12345)
sample_rows <- sample(nrow(tele_test), 10)
final_test <- tele_test_logistic[sample_rows, ]

final_prediction = predict(optimized_model, newdata = final_test)
final_logistic_prediction <- ifelse(final_prediction >= 0.5, 1, 0)

final_test = tele_test[sample_rows, ]
final_svm = predict(improved_simple_model, final_test)
final_svm_prediction <- ifelse(final_svm >= 0.5, 1, 0)

final_tree_prediction = predict(improved_tree, final_test)

final_new = as.numeric(final_logistic_prediction) + as.numeric(final_svm_prediction) +
as.numeric(final_tree_prediction)
final_prediction_new = ifelse(final_new ==3, "1", "0")
final_prediction_new
```
~~~

If we have these 10 randomly selected clients to be called today, based on our final methodology and model, we will call none of them today as all predictions are "0"s.

In conclusion, if we are interested in a high Kappa, where a higher Kappa value indicates better agreement between the classifier and the true labels, we should use the decision tree model with the cost matrix instead of the combined model. If our company is relatively conservative and lacks employees in the call center, we can choose the combined model as the majority of the predictions will be "0"s, and hence no phone call is needed, which is resulted from the strict decision criteria.

For future works, it is worthwhile to try switching back to the majority voting scheme, where 2 out of 3 models say 1 then the combined prediction is 1. This is a middle ground, allowing deficiency in individual models and making the final decision with an easier bar.