# MSDS 6372 - Project 2

## Logistic Regression

Christian Nava, Kevin Thompson, Mel Schwan
December 7, 2019

## Objective 1

### Introduction

Predicting whether a client will open a bank term deposit account can be an important business need for a bank's marketing department. Understanding a bank's customer type can aid in efficiently targeting customers most likely to open term deposit accounts thereby increasing the number of accounts opened and reducing the costs of acquiring these customers. Using data from a Portuguese banking institution, we aim to predict term deposit subscriptions. We also use an empirical model to conduct inference on variables of interest to find associations between these variables and account openings.

### Data Description

The Bank Marketing Data Set consists of 40,787 observations of 21 variables: 6 numeric, 14 categorical, and a binary response variable that has values of "yes" and "no". The data were originally used by Moro et al and come from direct marketing campaigns of a Portuguese banking institution.[1] A detailed description taken from the UCI Machine Learning Repository is available in the appendix.[2]

Per the dataset's authors note, the $duration$ variable "highly affects the output target (e.g., if duration=0 then y="no"). Yet, the duration is not known before a call is performed." We

excluded the $duration$ variable from our analysis to prevent target leakage because the duration of a phone call is not known prior to its completion.

[1] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014
[2] S. Moro, P. Cortez and P. Rita (2014). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Bank+Marketing]. Irvine, CA: University of California, School of Information and Computer Science.

## Exploratory Data Analysis

### *Missing and Unknown Values*

Missingness is a key issue in prediction and inference. We are primarily concerned with how outcomes change with missingness in the case of prediction. We do this by using the naniar R package to construct a shadow matrix.

Certain variables contained missing data in the form of “unknown” entries. These “unknown” entries were converted to explicitly missing values as shown in Table 1.

Table 1: Missing and Unknown Values

| **Variables** | **Missing or Unknown Values** | **% of Total Values** |
|---|---:|---:|
| default | 8597 | 20.9 |
| education | 1731 | 4.2 |
| housing | 990 | 2.4 |
| loan | 990 | 2.4 |
| job | 330 | 0.8 |
| marital | 80 | 0.2 |

We can see that approximately 20.9% of the $default$ variable is missing. Among the non-missing values for the $default$ variable, there are only 3 individuals who have shared a defaulted status on a loan. We can infer from this that clients who have defaulted are likely unwilling to share their loan default status with the bank.

We will remove the rows with missing and unknown values and say that our scope of inference and prediction is reduced to those individuals whom we know did not default.

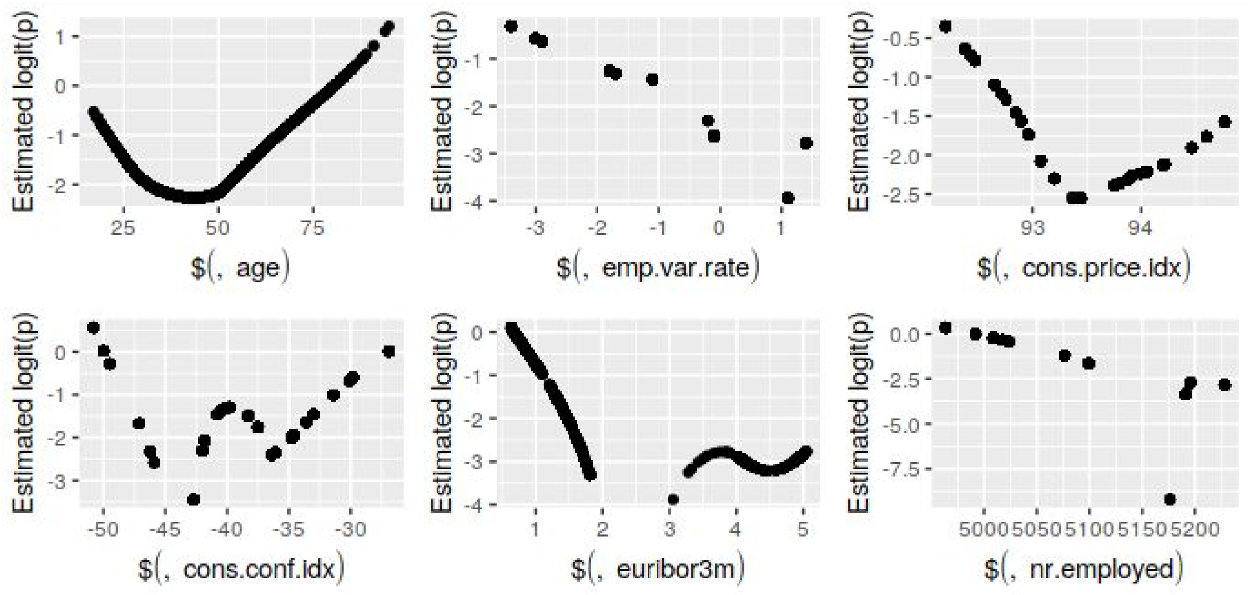
### *Missing Data Impacts on Prediction*

Because responses are directly observable, we can view how the response changes across the missingness of various variables. We found that only missingness in education and missingness in defaults were associated with the response variable. The proportion of “yes” responses stayed the same for groups of missing and non-missing individuals in other variables with missing values.

*Feature Importance*

Using a LOESS (Locally Estimated Scatterplot Smoothing) -based estimation of logit(p), we can construct a matrix of scatterplots that allows to compare the individual associations of various continuous predictors with the log-odds. Figure 1 shows a scatterplot matrix with all of our continuous predictors.

**Figure 1: Scatterplot Matrix of Continuous Predictors**



It is important to not take the plots too literally as these estimates are sensitive to the degree of smoothing and consequently the number of observations in a given neighborhood of the predictor. This is a drawback of LOESS, but we gain some insight into what types of associations our variables have with the log-odds of opening a bank term deposit account. All of our continuous variables appear to have some noticeable association with the log-odds.

We next investigate possible multicollinearity issues and take a deeper look at the importance of our features. Multicollinearity is particularly troublesome for prediction because substitution effects can lead us to underestimate important variables. While it is possible for us to correct for linear substitution effects using principal components, we prefer to maintain the interpretability of our model.

In order to further ascertain the relative importance of variables, we can fit random forest classifiers to find the Mean Decrease in Impurity (MDI) and the Mean Decrease in Accuracy (MDA) for all of our variables. We estimate the degree of multicollinearity by computing the condition number for our variance-covariance matrix, which is computed using the ratio between the largest and smallest singular values.

**Figure 2: Mean Decrease in Impurity by Variable**

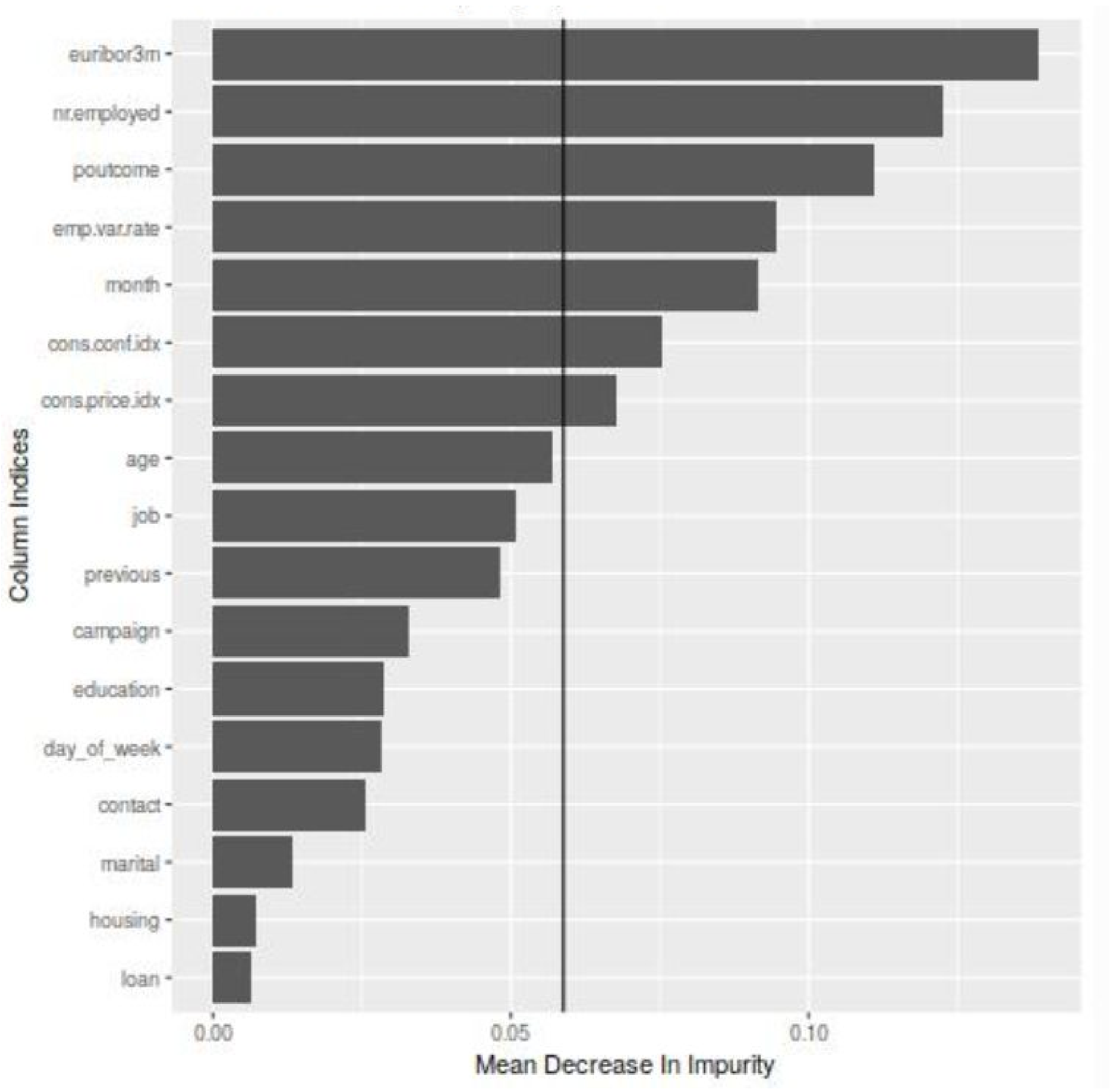
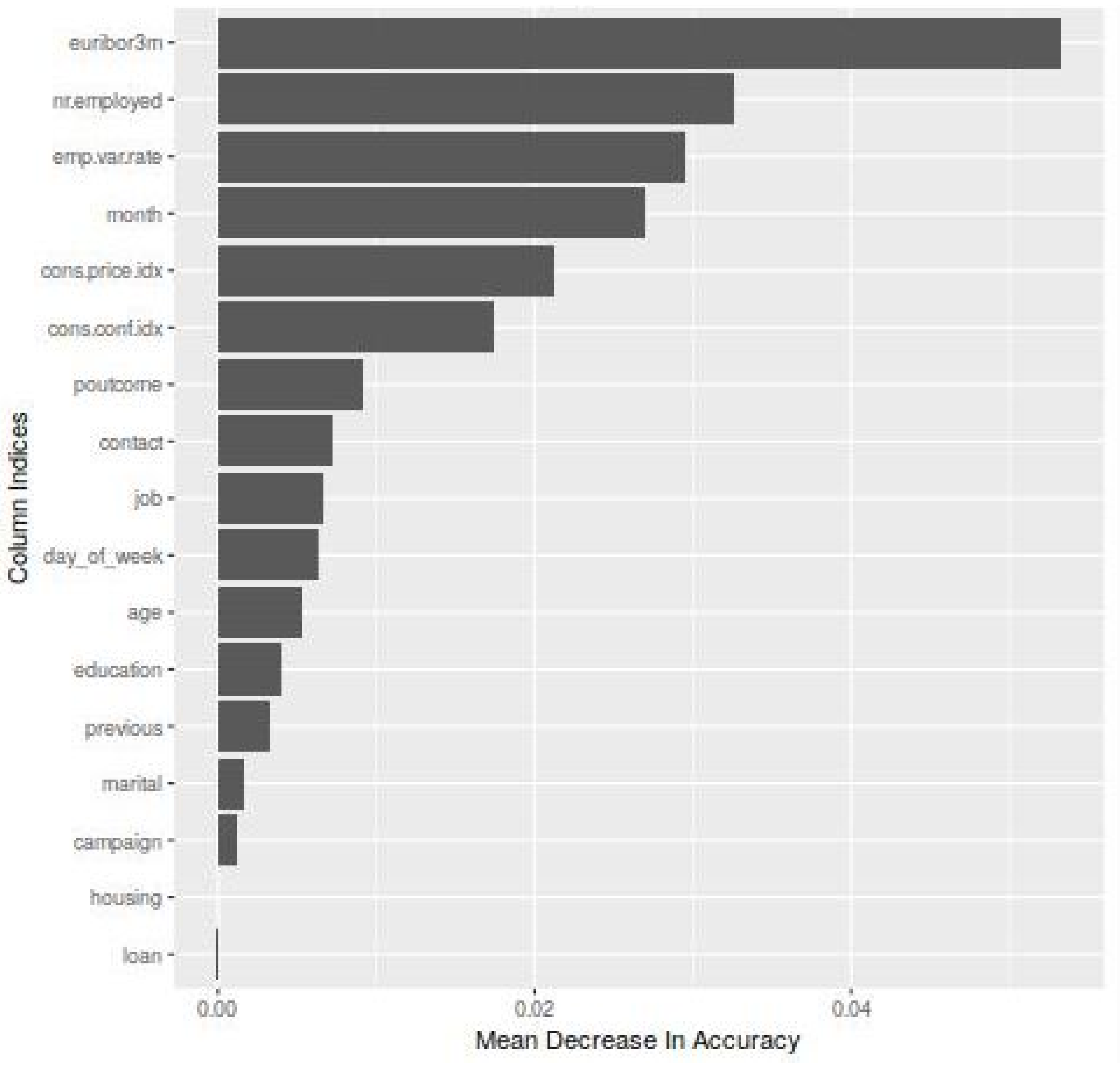


**Figure 3: Mean Decrease in Accuracy by Variable**



We further compute a condition number of 3.04, which is quite small, thus we find little evidence of multicollinearity and we can see that our features appear to be decently strong.

Figure 2 and Figure 3 further validates the information we received from our scatterplot matrix. The vertical line on the MDI plot is a threshold that is computed by determining how far the bars would extend if every variable were equally important. When variables exceed this threshold, it is a sign that they are particularly important. It should be noted, however, that MDI uses in-sample data. MDA is the out-of-sample counterpart that measures importance by seeing how much accuracy we lose by permuting the variables. We can see that the MDA graph agrees in large part with our MDI plot.

## Logistic Regression Model

### *Restatement of Problem*

Using the “Bank Marketing Data Set” available from UCI’s Machine Learning Repository, we will use various classification techniques to predict whether a client will open a term deposit account. We estimated our parameters with the following model:

$$logit(p) = \beta_0 + \beta_1 \cdot contact + \beta_2 \cdot month + \beta_3 \cdot poutcome + \beta_4 \cdot emp.var.rate + \beta_5 \cdot cons.price.idx$$

Where $logit(p)$ denotes the log odds of opening a bank term deposit account versus not opening one, $contact$ refers to the medium through which the client was contacted, $month$ refers to the last month in which the client was contacted, $poutcome$ refers to whether the client was previously contacted and what the outcome of this contact was, $emp.var.rate$ refers to the Portugeuse national employment rate, $cons.price.idx$ refers to the Portugeuse consumer price index, and epsilon denotes the error term.

### *Solution Outline*

We aim to accomplish our task with logistic regression techniques, decision-tree based classifiers, and linear discriminant analysis. The dataset will be preprocessed by removing any unnecessary or redundant variables. We will use the initial logistic regression model as a benchmark for subsequent model fits.

## Parameter Interpretation

Table 3 shows the parameter estimates. The respective confidence intervals are shown in Table 4.

### *Interpretation of Parameters*

$\beta_0$ : **Intercept** - This is the odds ratio of a client saying yes to a term deposit subscription when all other variables are theoretically zero. The odds ratio is $e^{-109.82} = 2.01 \times 10^{-48}$ with a 95% confidence level that the value for the odds ratio is contained in the interval [$5.61 \times 10^{-53}$, $7.13 \times 10^{-44}$]. The odds ratio and interval values are essentially zero, which makes sense, in theory, if a client has never been contacted (month=0) via phone or cellular (contact=0) there would be no previous outcome (poutcome=0). There also would be full employment in the country (var.emp.rate = 0) and no inflation (cons.price.idx = 0). The odds of signing up for an account under these circumstances over the odds of not signing up are very small.

$\beta_1$ : **contactcellular** - Holding all other variables at a fixed value, the odds of a client contacted via their cellular phone subscribing to a term deposit account over the odds of a client contacted by telephone (reference group) are $e^{0.46235}$ = 1.59, with a 95% confidence level that the value for the odds ratio is contained in the interval [1.41, 1.79].

$\beta_2$ : **monthapr -** Holding all other variables at a fixed value, the odds of a client last contacted in April subscribing to a term deposit account over the odds of a client last contacted in March (reference group) subscribing to a term deposit account are $e^{-1.29936}$ = 0.27, with a 95% confidence level that the value for the odds ratio is contained in the interval [0.22, 0.34].

$\beta_3$ through $\beta_{10}$ : **monthmay** through **monthdec** -The interpretation for these parameters is identical to $\beta_2$ where the month and odds would change respectively with each $\beta$ coefficient as per Table 4 in the appendix.

$\beta_{11}$ : **poutcomefailure -** Holding all other variables at a fixed value, the odds of a client subscribing to a term deposit account who was previously contacted for a marketing campaign with a failure outcome over the odds of a client subscribing to a term deposit account who was not previously contacted for a marketing campaign (reference group) are $e^{-0.46769}$ = 0.63 , with a 95% confidence level that the value for the odds ratio is contained in the interval [0.55, 0.71].

$\beta_{12}$ : **poutcomesucces -** Holding all other variables at a fixed value, the odds of a client subscribing to a term deposit account who was previously contacted for a marketing campaign with a successful outcome over the odds of a client subscribing to a term deposit account who was not previously contacted for a marketing campaign (reference group) is $e^{1.32374}$ = 3.76 , with a 95% confidence level that the value for the odds ratio is contained in the interval [3.22, 4.39].

$\beta_{13}$ : **emp.var.rate -** Holding all other variables at a fixed value, for a one-unit increase in $emp.var.rate$ , the odds ratio will change by $e^{-0.82187}$ = 0.44, with a 95% confidence level that the value for the expected change in the odds ratio is contained in the interval [0.42, 0.46].

$\beta_{14}$ : **cons.price.idx -** Holding all other variables at a fixed value, for a one-unit increase $cons.price.idx$ , the odds ratio will change by $e^{1.15816}$ = 3.18, with a 95% confidence level that the value for the expected change in the odds ratio is contained in the interval [2.85, 3.56].

## Model Selection

We began with the full model and then manually selected the predictors most significantly associated to the response, which included $contact$ , $month$ , $poutcome$ , $emp.var.rate$ , and $cons.price.index$ . To ensure our model fits well, we will perform a prediction accuracy test using both an unbalanced and a balanced training set.

Table 1: Logistic Model Select Confusion Matrix Output

| Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Unbalanced | 90.0% | 15.8% | 99.3% |
| Balanced | 83.3% | 63.1% | 86.2% |

### *Prediction Accuracy - Unbalanced Training Set*

When a dataset has significantly larger number of either the positive or negative classes it is considered unbalanced. In an unbalanced dataset with a binary response variable, we run the risk of disproportionately sampling observations (i.e., we may sample too many with a "no" and too few with a "yes" for the response variable). This can result in misleading accuracy metrics.

Per the confusion matrix for our model (Figure 4), we get a 90.0% accuracy, a 15.8% sensitivity and a 99.3% specificity. Sensitivity is the proportion of the positive class that was classified correctly. Specificity is the proportion of the negative class that was classified correctly. The positive class for this model is “yes ” and the negative class is a "no" for term deposit subscriptions.

Since we are interested in a client subscribing to a term deposit account (i.e., a “yes” response) we are more interested in the sensitivity metric. The model does a poor job of classifying the positive class, correctly classifying only 15.8% of “yes” responses as shown in Table 1 and the confusion matrix in the appendix, Table 6.

### *Prediction Accuracy - Balanced Training Set*

When we balance the sampling, it gives us a training set with similar proportions of responses as the entire dataset. Per the confusion matrix for the balanced training set, we get an 83.3% accuracy, a sensitivity of 63.1% and a specificity of 86.2%. From the sensitivity metric we can see that using a balanced training set improved the model’s ability to correctly classify the “yes” responses.

### *Note on Data*

The response variable was downsampled prior to fitting all of the following models. This allowed us to achieve a better balanced accuracy. Prior to downsampling, the estimated sensitivity of our algorithms were consistently too low to be usable in a production setting. After downsampling, all of our algorithms performed significantly better. While this did reduce our specificity, the trade-off was insignificant.

## Conclusions

Our results suggest that the odds of opening a bank term deposit account for a client who was contacted by cellular phone are 1.59 times higher than that of a client contacted by telephone.

Also, the odds of a client opening a bank term deposit account who was previously contacted for a marketing campaign with a successful outcome are 3.76 times higher than for a client not previously contacted. The consumer price index (CPI) is a measure of inflation. As $cons.price.idx$ increases, we would expect to see a 218% increase in the odds of a client opening a term deposit account for every one unit increase in $cons.price.idx$.

Future campaigns can consider targeting clients who have previously opened bank term deposit accounts, and clients who have a cellular phone on file. Marketing campaign leaders can also pay closer attention to CPI to time the deployment of a campaign.

## Objective 2

### Competing Models

We will use a logistic ridge regression, linear discriminant analysis, and a random forest model in an attempt to increase the prediction performance over the simple logistic regression from objective one. Ridge regression adds just enough bias to make the estimates reasonably reliable approximations to actual population values. Random forest builds multiple decision trees and merges their predictions resulting in a more accurate and stable prediction.

The banking marketing dataset is an unbalanced dataset with a majority negative class and a minority positive class. To create a more balanced dataset we have downsampled the original cleaned dataset from objective one and will be comparing performance to the balanced logistic benchmark model.

#### *Ridge Regression Model*

Ridge regression models are essentially linear models that introduce additional skepticism into the significance of predictors by inflating the variance-covariance matrix by some scalar multiple of the identity matrix. Logistic ridge regression is a special case of this class of linear models in a setting where we believe that the likelihood is binomial rather than gaussian.

We employed the R function model.matrix(), which helps create a matrix of predictors and automatically converts categorical predictors to appropriate dummy variables required for the glmnet() function.

Our single hyper-parameter of interest is lambda, the scalar value that inflates our variance-covariance matrix. We tuned this hyper-parameter through 5-fold cross-validation.

The ridge regression model performed better than the benchmark model in terms of sensitivity and fared worse in terms of accuracy and specificity. We can see from Table 7 that our ridge model using the balanced dataset is better at classifying the positive class but not as good at classifying the negative class as is the benchmark logistic model.

### *Linear Discriminant Analysis Model*

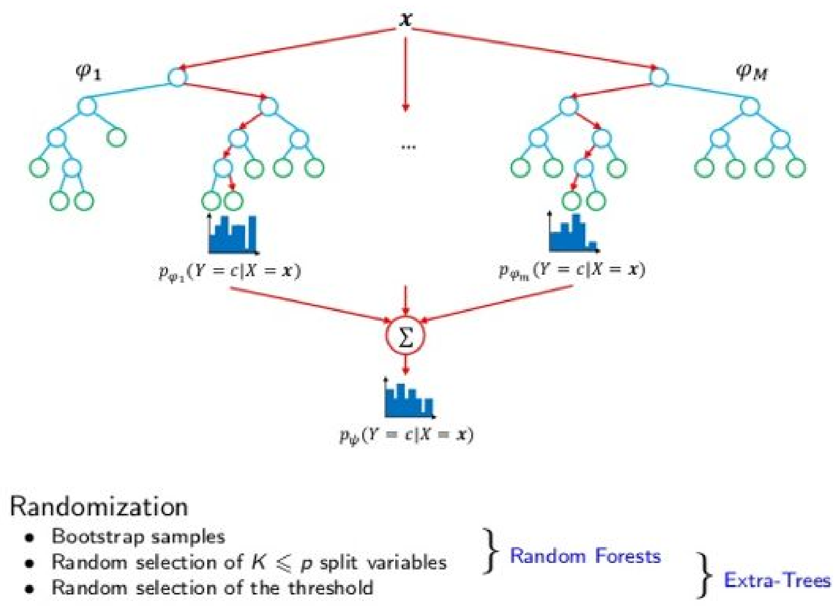
We fit a linear discriminant analysis ("LDA") model since LDA models tend to work well in classification problems with a categorical output variable. LDA models assume that the decision boundaries for the classes of interest can be modeled as a linear combination of gaussian predictors and that there is a constant variance-covariance matrix across classes. We fit our LDA model using all available continuous predictors with sufficient variance for the numerical computations in the LDA algorithm. When we compare the overall accuracy of the ridge and LDA models, the ridge model has better sensitivity results than the LDA or benchmark models.

### *Random Forest Model*

We finally fit a random forest classifier to compare its performance to the other models. The random forest model yielded a reasonable compromise between sensitivity (69.3%) and specificity (81.3%). Its overall performance was similar to that of the logistic ridge regression model (sensitivity = 70.2%, specificity = 79.5%) as shown in Table 2.

The random forest classifier is an ensemble method of learning for classification and is potentially more powerful than logistic regression. The random forest algorithm takes a dataset of N observations and produces K datasets of size N/K by sampling with replacement from the original dataset. Each tree then randomly samples features from the feature space at each partition to reduce the correlation between trees and then seeks to minimize the impurity of its classifications. The mode of the trees' outputs is the chosen classification for a new observation.

**Figure 4:Random Forest Model**



$x$
$\varphi_1$
$\varphi_M$
...
$p_{\varphi_1}(Y = c|X = x)$
$p_{\varphi_m}(Y = c|X = x)$
$\Sigma$
$p_{\psi}(Y = c|X = x)$
Randomization
Bootstrap samples
Random selection of $K \leqslant p$ split variables
Random selection of the threshold
Random Forests
Extra-Trees

Source: KDNuggets

The basic procedure for the algorithm involves the following steps illustrated above in Figure 4.

1. Select $\phi_M$ bootstrap samples from the training data, $\chi$.
2. For each of the bootstrap samples $\phi_M$, grow an unpruned classification or regression tree to the largest extent possible (i.e., split the node into two daughter nodes until the minimum node size is reached).
3. Predict new data by aggregating the predictions of the $\phi_M$ trees. These predictions can be assembled into a probability distribution, whose mode (i.e., majority votes for classification) or average (i.e., aggregation for regression) yields the random forest's proposed rating $p_\psi(Y = c|X = \chi)$.

Given the small number of features, we focused on tuning the mtry hyper-parameter shown in the code in the appendix. The number of features resampled at each node-split was tuned using out-of-bag samples.

## *Performance Comparison*

Table 2 provides the relevant performance metrics for our binary classifiers. The best overall performers were the logistic ridge regression model (sensitivity = 70.2%) and the random forest classifier (sensitivity = 69.2%). Given the similarity in performance, it is unclear whether one model is necessarily better than the other; however, the fact that it is easier to fit and tune a logistic regression model makes it a more appealing choice for deployment.

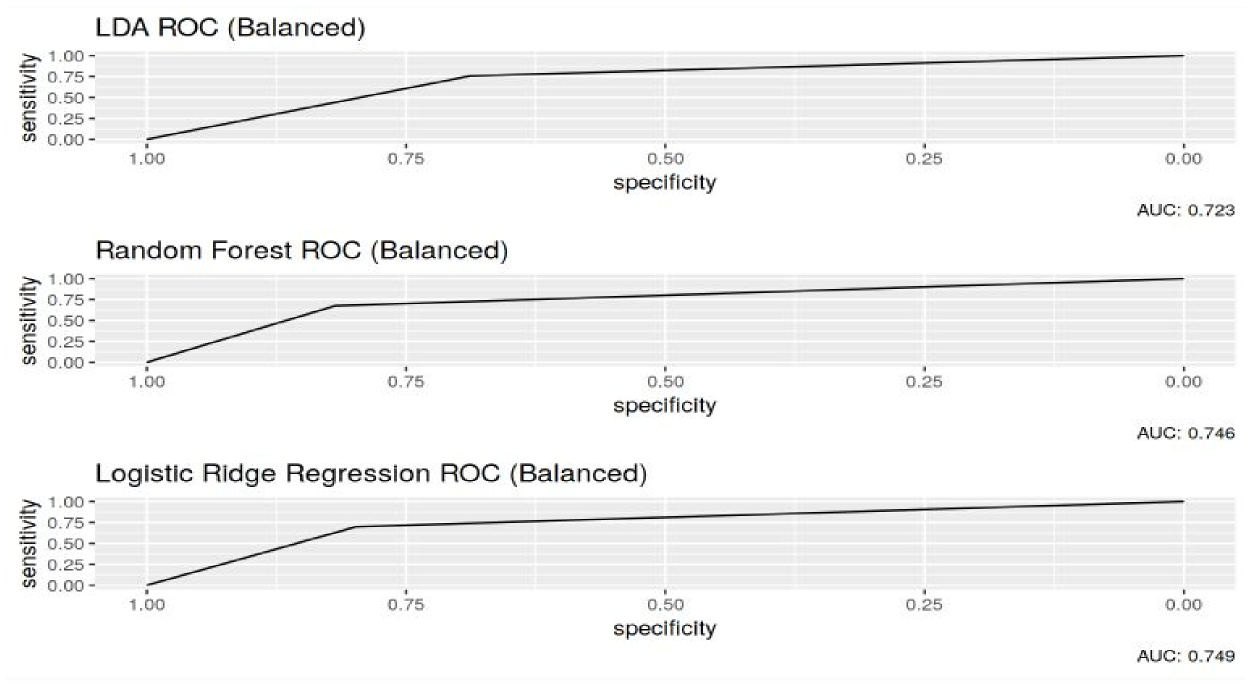
Table 2: Model Comparative Performance

| Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Benchmark Logistic | 84.5% | 62.9% | 87.6% |
| Logistic Ridge | 78.3% | 70.2% | 79.5% |
| LDA | 81.8% | 66.4% | 84.0% |
| Random Forest | 79.8% | 69.2% | 81.3% |

We will also consider the area under the receiver operating characteristic (ROC) curve as a performance metric. The ROC curve graphs the performance of a classification model at all classification thresholds; it plots the true sensitivity against the specificity as shown in Figure 5. Movement along the curve indicates changing the threshold used for classifying a positive instance.

The area under the ROC curve measures the two-dimensional area under the ROC curve. This metric is measured between 0 and 1. A higher score indicates a better model. Visually, a superior model will graph as a curve above and to the left of another curve. A straight diagonal line beginning at 0 on the bottom left and ending at 1 on the top right indicates a naive model (random guessing) where the AUC score would be 0.50.

**Figure 5: Comparative ROC AUC Results**



Per the ROC AUC results, we see that the logistic ridge model performed better (AUC = 0.749) than the LDA or random forest models.

## Project Conclusions

In order to determine whether a client will sign up for a term deposit account we would use a logistic ridge regression model as this has shown to have the best compromise between sensitivity and specificity, i.e., it is the model that most correctly predicts a “yes” for the response variable while having sufficient specificity to avoid unnecessarily wasting resources by over-calling clients who are not interested in an account.

All three of our models performed similarly, though the logistic ridge regression model and the random forest model performed the best. The AUC estimates are so similar that it is difficult to tell whether one model is necessarily better at prediction than the other. In a production environment, it would make more sense to use the logistic ridge regression model because it is easier to fit and maintain.

We have reason to believe that an AUC of around 0.75 is the best we can achieve with these predictors because the paper associated with this data achieved similar results with different models.

There are several predictors that would have been helpful for our models, particularly an employee ID for those employees that call clients.  Different employees will have different levels of charisma and persuasive abilities; that information is not captured in the feature set because we have no knowledge of which calls were made by the same employees. Additional research using boosted tree algorithms such as LightGBM or XGBoost can be considered as these classifiers tend to do well in binary classification tasks.

# Appendix

## Variable Descriptions

### Bank client data:
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')

### Related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

### Other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

### Social and economic context attributes:
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)

18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

### Output variable (desired target):

21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

## Tables

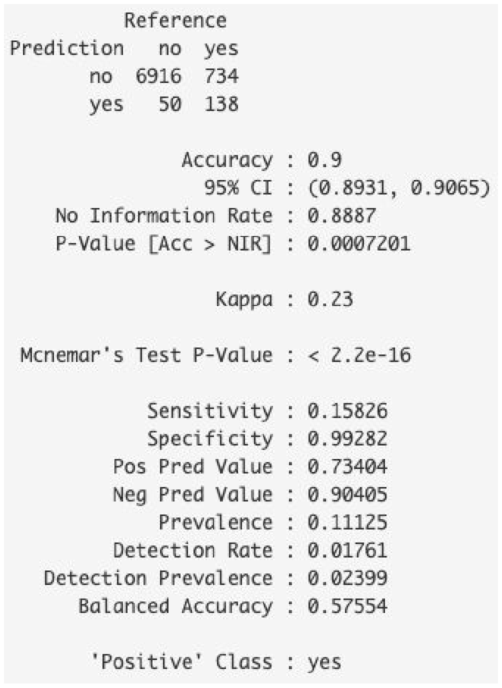
**Table 3: Logistic Regression Output**

| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|
| (Intercept) | -108.1556055 | 4.9039026 | -22.055007 | 0 |
| contactcellular | 0.492937 | 0.0595 | 8.288417 | 0.00000 |
| monthjun | 0.3843162 | 0.0748 | 5.136619 | 0.00000 |
| monthjul | 0.8365575 | 0.0732 | 11.431654 | 0.00000 |
| monthaug | 1.0618362 | 0.0702 | 15.132686 | 0.00000 |
| monthoct | 0.9082919 | 0.0984 | 9.227821 | 0.00000 |
| monthnov | 0.3524237 | 0.0749 | 4.70293 | 0.00000 |
| monthdec | 1.0695659 | 0.1794 | 5.963281 | 0.00000 |
| monthmar | 1.8337679 | 0.1038 | 17.673948 | 0.00000 |
| monthapr | 0.528272 | 0.0690 | 7.653731 | 0.00000 |
| monthsep | 0.8140786 | 0.1077 | 7.555386 | 0.00000 |
| poutcomefailure | -0.4861733 | 0.0588302 | -8.264008 | 0 |
| poutcomesuccess | 1.3166609 | 0.0736063 | 17.887895 | 0 |
| emp.var.rate | -0.8143643 | 0.0215868 | -37.725178 | 0 |
| cons.price.idx | 1.1217132 | 0.0523706 | 21.418746 | 0.00000 |

**Table 4: Odds Ratios and 95% Confidence Intervals**

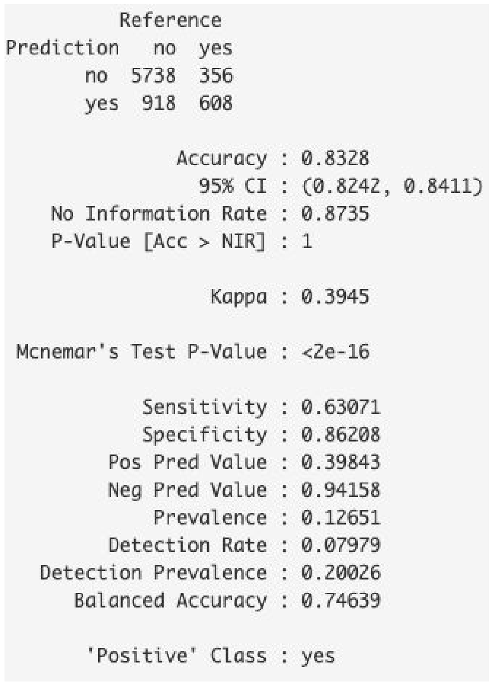
| Parameter | odds ratio | lower bound | upper bound |
|---|---:|---:|---:|
| (Intercept) | 0.00 | 0.00 | 0.00 |
| contactcellular | 1.59 | 1.41 | 1.79 |
| monthapr | 0.27 | 0.22 | 0.34 |
| monthmay | 0.18 | 0.14 | 0.22 |
| monthjun | 0.25 | 0.20 | 0.32 |
| monthjul | 0.39 | 0.31 | 0.50 |
| monthaug | 0.54 | 0.42 | 0.68 |
| monthsep | 0.43 | 0.32 | 0.58 |
| monthoct | 0.46 | 0.35 | 0.61 |
| monthnov | 0.24 | 0.19 | 0.31 |
| monthdec | 0.53 | 0.35 | 0.80 |
| poutcomefailure | 0.63 | 0.55 | 0.71 |
| poutcomesuccess | 3.76 | 3.22 | 4.39 |
| emp.var.rate | 0.44 | 0.42 | 0.46 |
| cons.price.idx | 3.18 | 2.85 | 3.56 |

**Table 5: Confusion Matrix,**
**Unbalanced Logistic Model**



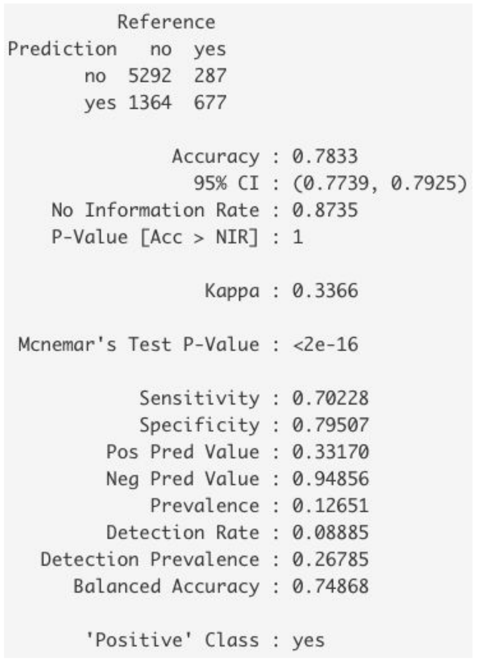
```
          Reference
Prediction   no  yes
       no  6916  734
       yes   50  138

               Accuracy : 0.9
                 95% CI : (0.8931, 0.9065)
    No Information Rate : 0.8887
    P-Value [Acc > NIR] : 0.0007201

                  Kappa : 0.23

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.15826
            Specificity : 0.99282
         Pos Pred Value : 0.73404
         Neg Pred Value : 0.90405
             Prevalence : 0.11125
         Detection Rate : 0.01761
   Detection Prevalence : 0.02399
      Balanced Accuracy : 0.57554

       'Positive' Class : yes
```

**Table 6: Confusion Matrix,**
**Balanced Logistic Model**



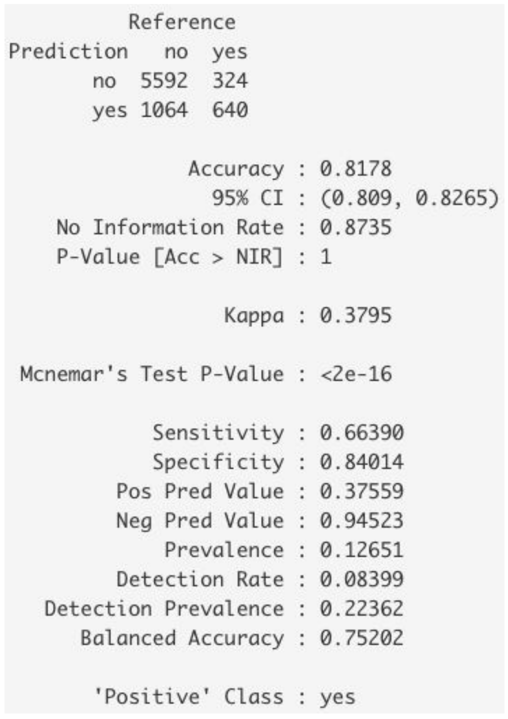
```
          Reference
Prediction   no  yes
       no  5738  356
       yes  918  608

               Accuracy : 0.8328
                 95% CI : (0.8242, 0.8411)
    No Information Rate : 0.8735
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3945

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.63071
            Specificity : 0.86208
         Pos Pred Value : 0.39843
         Neg Pred Value : 0.94158
             Prevalence : 0.12651
         Detection Rate : 0.07979
   Detection Prevalence : 0.20026
      Balanced Accuracy : 0.74639

       'Positive' Class : yes
```

**Table 7:  Confusion Matrix,**
**Balanced Logistic Ridge Model**



```
          Reference
Prediction   no  yes
       no  5292  287
       yes 1364  677

               Accuracy : 0.7833
                 95% CI : (0.7739, 0.7925)
    No Information Rate : 0.8735
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3366

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.70228
            Specificity : 0.79507
         Pos Pred Value : 0.33170
         Neg Pred Value : 0.94856
             Prevalence : 0.12651
         Detection Rate : 0.08885
   Detection Prevalence : 0.26785
      Balanced Accuracy : 0.74868

       'Positive' Class : yes
```

**Table 8: Confusion Matrix,**
**Balanced LDA Model**



```
          Reference
Prediction   no  yes
       no  5592  324
       yes 1064  640

               Accuracy : 0.8178
                 95% CI : (0.809, 0.8265)
    No Information Rate : 0.8735
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3795

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.66390
            Specificity : 0.84014
         Pos Pred Value : 0.37559
         Neg Pred Value : 0.94523
             Prevalence : 0.12651
         Detection Rate : 0.08399
   Detection Prevalence : 0.22362
      Balanced Accuracy : 0.75202

       'Positive' Class : yes
```

## Code for Objective 1

```
library(tidyverse)
library(gmodels) #cross tables and logistic regression
library(caret)
library(ggplot2)
library(data.table)

# Read in data
df_raw <- read.csv("data/bank-additional-full.csv",sep = ";")

##################################
###  Exploratory Data Analysis  ##
##################################

# Check for missing values
# Report the number of NA's in each column.
a = colnames(df_raw)
b = colSums(is.na(df_raw))  %>% as.data.table

missing_value_table = cbind(a, b)
colnames(missing_value_table) = c("Variables","Missing_values")
missing_value_table = missing_value_table  %>%
  filter(Missing_values>0)  %>%
  mutate("% of Total Values" = round(100 * (Missing_values /
nrow(df_raw)), 1))  %>%
  arrange(desc(Missing_values))

head(missing_value_table, 10)

# Report the number of "unknown"'s in each column.
a = colnames(df_raw)
b = colSums(df_raw=="unknown")  %>% as.data.table

unknown_value_table = cbind(a, b)
colnames(unknown_value_table) = c("Variables","Unknown_values")
unknown_value_table = unknown_value_table  %>%
  filter(Unknown_values>0)  %>%
  mutate("% of Total Values" = round(100 * (Unknown_values /
nrow(df_raw)), 1))  %>%
  arrange(desc(Unknown_values))
```

```
head(unknown_value_table, 10)

# The following EDA code based on and adapted from Kaggle user
# Psqrt available from:
https://www.kaggle.com/psqrtpsqrt/bank-marketing-eda-classification-p
r-f-score

# Correlation Matrix of Continuous Variables
library(corrplot)
df_raw %>%
  dplyr::select(emp.var.rate, cons.price.idx, cons.conf.idx,
euribor3m, nr.employed) %>%
  cor() %>%
  corrplot(method = "number",
           type = "upper",
           tl.cex = 0.8,
           tl.srt = 45,
           tl.col = "black")
# Per the correlation matrix, there do not appear to be any variables
of concern. We will keep all numeric variables.

# Age
df_raw %>%
  ggplot() +
  aes(x = age) +
  geom_bar() +
  geom_vline(xintercept = c(30, 60),
             col = "red",
             linetype = "dashed") +
  facet_grid(y ~ .,
             scales = "free_y") +
  scale_x_continuous(breaks = seq(0, 100, 5))

# Cross Tables
CrossTable(df_raw$age, df_raw$y)          # Age
CrossTable(df_raw$job, df_raw$y)          # Job
CrossTable(df_raw$marital, df_raw$y)      # Marital Status
CrossTable(df_raw$education, df_raw$y)    # Education
CrossTable(df_raw$default, df_raw$y)      # Default
CrossTable(df_raw$housing, df_raw$y)      # Housing
CrossTable(df_raw$loan, df_raw$y)         # Loan
CrossTable(df_raw$contact, df_raw$y)      # Contact
CrossTable(df_raw$month, df_raw$y)        # Month
CrossTable(df_raw$day_of_week, df_raw$y)  # Day of Week
CrossTable(df_raw$campaign, df_raw$y)     # Campaign
CrossTable(df_raw$pdays, df_raw$y)        # Pdays
CrossTable(df_raw$previous, df_raw$y)     # Previous
CrossTable(df_raw$poutcome, df_raw$y)     # Poutcome


###############################
##    Data Pre-processing    ##
###############################

# Split ages into three groups: "17-30", "30-60", ">60"
df_raw = df_raw %>%
  mutate(age = if_else(age > 60, "high", if_else(age > 30, "mid",
"low")))

# Remove observations with "unknown" values for job, marital, and
education variables
df_raw = df_raw %>%
  filter(job != "unknown") %>%
  filter(marital != "unknown") %>%
  filter(education != "unknown")

# Drop variables
df_raw = df_raw %>%
  select(-c(duration, default, pdays, housing, loan))

# Set reference levels for factors
df_raw$contact <- factor(df_raw$contact, order = FALSE, levels
=c('telephone', 'cellular'))
df_raw$education <- factor(df_raw$education, order = FALSE, levels
=c('illiterate','basic.4y', 'basic.6y','basic.9y',
'high.school','professional.course','university.degree'))
df_raw$age <- factor(df_raw$age, order = FALSE, levels
=c('low','mid', 'high'))
df_raw$job <- factor(df_raw$job, order = FALSE, levels
=c('blue-collar', 'services','entrepreneur', 'housemaid',
'self-employed','technician', 'management','admin.','unemployed',
'retired','student'))
df_raw$marital <- factor(df_raw$marital, order = FALSE, levels
=c('married', 'divorced', 'single'))
df_raw$month <- factor(df_raw$month, order = FALSE, levels =c('mar',
'apr','may', 'jun','jul', 'aug', 'sep','oct', 'nov','dec'))
```

```
df_raw$poutcome <- factor(df_raw$poutcome, order = FALSE, levels
=c('nonexistent', 'failure','success'))
df_raw$y <- factor(df_raw$y, order = FALSE, levels = c('no', 'yes'))

# make copy of dataframe to avoid messing up original
df_bank.clean <- df_raw

# Split the data into training and test set
set.seed(123)
training.samples <- createDataPartition(df_bank.clean$y, p = 0.8,
list = FALSE)
train.data  <- df_bank.clean[training.samples, ]
test.data <- df_bank.clean[-training.samples, ]


################################
##    Creation of Datasets    ##
################################

# deal with missing values: see missing data file in analysis folder
for reference

bank <- map_df(bank, na_if, y = "unknown")
bank <- bank %>% drop_na(default, housing, loan, job, education,
duration, marital) %>% filter(default == "no")
bank <- bank %>% dplyr::select(-c(default, duration, pdays))
bank <- bank %>% droplevels()
set.seed(193204)
#create training and test sets
idx <- createDataPartition(bank$y, p = 0.75, list = FALSE)
training.data <- bank[idx, ]
test.data <- bank[-idx, ]
balanced.training.data <- training.data %>% downsample(y)
balanced.training.data_X <- balanced.training.data %>%
dplyr::select(-c(y))
test.data_X <- test.data %>% select(-c(y))

###########################
##    Model Selection    ##
###########################

# Fit full model using all of the potential predictors
model_full <- glm(y ~ ., data = train.data, family = "binomial")
```

```
summary(model_full)

# Fit a model using the most relevant variables from the full model
model_final <- glm(y ~ contact + month + poutcome + emp.var.rate +
cons.price.idx,
                   family = binomial(link = 'logit'),
                   data = train.data)
summary(model_final)

# Coefficient confidence intervals
confint.default(model_final)

# odds ratios and 95% CI
exp(cbind(OR = coef(model_final), confint(model_final)))

##################################
##    Unbalanced Predictions    ##
##################################

library(caret)
predictions <- predict(model_final, newdata = test.data, type =
"link")
predictions <- as.factor(map_chr(predictions, function(x){if(x >=
0.5){return("yes")} else{return("no")}}))

# confusion matrix
confusionMatrix(predictions, test.data$y, positive = "yes")

###############################
##   Balanced Prediction     ##
###############################

load("data/datasets.RData")

# Fit balanced model
model_balanced <- glm(y ~ contact + month + poutcome + emp.var.rate +
cons.price.idx,
                      family = binomial(link = 'logit'),
                      data = balanced.training.data)

# make predictions
predictions.balanced <- predict(model_balanced, newdata = test.data,
type = "link")
```

```
predictions.balanced <- as.factor(map_chr(predictions.balanced,
function(x){if(x >= 0.5){return("yes")} else{return("no")}}))

# confusion matrix
confusionMatrix(predictions.balanced, test.data$y, positive = "yes")
```

## Code for Objective 2

```
####################################
##    Ridge Regression Model     ##
####################################

# Create prediction matrix
x = model.matrix(y~., balanced.training.data)[,-1]
y = balanced.training.data$y
xtest <- model.matrix(y~.,test.data)[,-1]
ytest <- test.data$y

# Fit ridge model
library(glmnet)
balanced.ridge.mod <- cv.glmnet(x, y, alpha = 0, type.measure =
"class", nfolds = 5, family = "binomial")

# Lambda plot
plot(balanced.ridge.mod)
balanced.bestlambda <- balanced.ridge.mod$lambda.min
balanced.ridge.pred <- as.factor(predict(balanced.ridge.mod, s =
balanced.bestlambda, newx = xtest, type = "class"))

# confusion matrix
confusionMatrix(data = balanced.ridge.pred, reference = ytest,
positive = "yes")

#####################
##    LDA Model    ##
#####################

library(MASS)
# Fit LDA Model
lda.model.balanced <- lda(y ~ contact + month + poutcome +
emp.var.rate + cons.price.idx,
                          data = balanced.training.data)
```

~~~
# Make predictions
predmodel.train.lda = predict(lda.model.balanced, newdata =
test.data, type = "class")

# Confusion matrix
confusionMatrix(data = predmodel.train.lda$class, reference =
test.data$y, positive = "yes")

###############################
##    Random Forest Model    ##
###############################

```{r rfmodel}
library(naniar)
library(FactoMineR)
features <- bank %>% select(-c(y))
model <- randomForest(y ~ ., data = bank, mtry  = 1, importance =
TRUE)
model2 <- randomForest(y ~ ., mtry = 1, data = bank, importance =
TRUE)
~~~
```{r MDI}
source("../src/featureImportance/MDA.R")
source("../src/featureImportance/MDI.R")
plotImpMDI(model)
```
```{r multicollinearity}
source("../src/multicollinearity/conditionNumber.R")
paste("Condition Number:",conditionNumber(features))
```
Low condition number suggests that we do not have to worry about
multicollinearity.
```{r mda}
plotImpMDA(model2)
```
```{r balancedRFModel}
library(caret)
library(doMC)
registerDoMC(cores = 4)
set.seed(825)
oob = trainControl(method = "oob")
```

```
  rf_grid =  expand.grid(mtry = 1:18, splitrule = "gini", min.node.size
  = 500)
  balanced_rf_model = train(y ~ ., data = balanced.training.data,
                       method = "ranger",
                       trControl = oob,
                       verbose = FALSE,
                       tuneGrid = rf_grid,
                       metric = "Accuracy")
  ```
  ```{r balancedResults}
  plot(balanced_rf_model)
  ```
  ```{r balancedFinal}
  new_balanced_rf_model <- randomForest(x = balanced.training.data_X, y
  = balanced.training.data$y, mtry = 15, nodesize = 25)
  ```
  ```{r balancedResults2}
  balanced.pred <- predict(new_balanced_rf_model, newdata = test.data)
  confusionMatrix(data = balanced.pred, reference = test.data$y,
  positive = "yes")
  new_balanced_rf_model.roc <- roc(test.data$y,
  as.ordered(balanced.pred))
  ggroc(new_balanced_rf_model.roc) + labs(title = "Random Forest ROC
  (Balanced)",caption =
  paste("AUC:",round(new_balanced_rf_model.roc$auc, 3)))
  ```

  ##############################
  ##    LOESS Scatterplots    ##
  ##############################
```{r, echo=TRUE}

logistic_scatter <- function(data, response, ..., smoothing = 0.7,
jitter=FALSE){

  predictors <- map(enexprs(...), function(x){return(expr(`$`(!!data,
!!(x))))})

  response <- expr(`$`(!!enexpr(data), !!enexpr(response)))

  graph_list <- map(predictors, loess_plot, response = response, span =
smoothing, jitter=jitter)
```

~~~~
  names(graph_list) <- as.character(eval(enexprs(...)))

}

loess_plot <- function(predictor, response, span, jitter){

  response <- as.numeric(eval(response)) - 1

  loess.mod <- loess(response ~ eval(predictor), span = span)

  loess.pred <- predict(loess.mod)

  prob <- pmax(pmin(loess.pred, 0.9999), 0.0001)

  logit.loess.pred <- logit(prob)

  if(jitter == FALSE){

    return(ggplot(NULL,aes(x = eval(predictor), y = logit.loess.pred))
+ geom_point() +

             labs(y = "Estimated logit(p)", x = predictor))

  }

  else{

    return(ggplot(NULL,aes(x = predictor, y = logit.loess.pred)) +
geom_point(position = "jitter") +

             labs(y = "Estimated logit(p)", x = predictor))

  }

}

logit <- function(p){

  log(p/(1-p))

}

```
~~~~