A Data-Driven Approach to Predict the Success of Bank Telemarketing

Rob Pruette, Manuel Michel Corona
Master of Science in Applied Statistics and Data Analytics, Southern Methodist University
Dallas, TX 75275 USA
{rpruette, mmichelcorona}@smu.edu

Abstract

Nowadays, the creation of new financial intermediaries has become essential due to the need of finding solutions that will bring the most people into the formal banking system. At the same time, this has generated an aggressive competition among these institutions that want to attract the resources of the population. Therefore, banks, which are the most important institutions in any economy, now need to analyze and improve their marketing strategies in order to identify potential customers in the most efficient manner. This project examines a data set that contains the results of direct marketing campaigns of a Portuguese banking institution, and the goal is to predict if the contacted clients will make a term deposit in this bank. Different predictive and statistical techniques as Logistic Regression, Penalized Logistic Regression, Logistic Regression with Principal Component Analysis (PCA), Random Forest, and Neural Networks were implemented and compared utilizing different the metrics: accuracy, kappa statistic, sensitivity and specificity. "Up-sampling" and optimal cut point methods were also used to deal with the highly imbalanced data set. Overall, the model that seemed to perform the best in this data set was the Neural Network, even though, the accuracy (86.5%) was not the best, the kapa statistic (0.40), sensitivity (90.4%) and specificity (55%) combined, outperformed the other models when the goal was to predict if the customer made a term deposit.

Introduction

All over the world, banks are constantly having to find new ways to attract people's attention and innovate new products so that people will trust them with their money. With new customer deposits, banks accomplish one of their primordial goals of keeping the economy stable as it grows.

Given the data from a marketing campaign performed by a Portuguese bank, we ask ourselves, "Can the implementation of statistical and machine learning models better predict whether a customer will invest in a short-term deposit?"

As the financial services ecosystem constantly changes, banks are continually updating how their funds are allocated. If banks can more accurately predict who will invest or at least gain a better understanding of what factors make a client more likely to invest, they could optimize marketing expenses and profits.

Background

How do banks make money and what is their role in the economy?

Simply put, commercial banks make money by providing loans. Their income is generated by the interest customers pay on these loans. The capital to provide loans comes from customer deposits such as checking accounts, savings accounts, money market accounts, CDs, term deposits, and other bank products.

Customers who decide to deposit their money into these accounts, are lending money to the bank and will be paid an interest for as long as the bank keeps their money. However, the interest rate that the bank is paying them will always be less than the interest rate they charge on the money they lend. As a result, an important portion of the profits of a commercial bank is determined by the gap between the interest paid to customers on their deposits and the interest earned on the loans issued. This is called net interest income [1].

From here we see the importance of deposits made to the bank. They are by far the largest source of funds received, even though in most cases, the deposits have very short terms. This is because customers reserve the right to withdraw the full amount of their deposits at any time.

Banks have an important role in the economy since they can directly influence the supply of money, inflation, and economic growth of a country. They are a key component of the financial system since they are in charge of allocating funds from people to borrowers in a very efficient manner. They also can provide specialized financial services that help to make the economy more efficient.

It is important to note that banks are only one type of financial intermediary. Many new financial products and services have been developed, as well as new institutions that include credit unions, insurance companies, mutual funds, pension funds, finance companies, and real estate investment trusts.

As a result, there is a very strong competition between financial intermediaries to capture customer's attention and convince them to make deposits or open new accounts. Therefore, it has become essential for them to implement different and efficient marketing campaigns that can help them to identify potential customers.

Data

As previously mentioned, the data set used for this project is based on the marketing campaigns of a Portuguese banking institution. This data set was accessed from the UCI Machine Learning Repository website. In the campaign, clients were called, often more than once, to determine if they would subscribe to a bank term deposit. The responses were recorded as "yes" or "no". 20 variables are included in the data set including personal information, data related to the last contact with the client, and social and economic context variables. The data set contains 41,188 observations.

Table 1 provides a dictionary of all the variables included in the dataset.

Table 1: Data Dictionary

Variable	Description
age	Age of prospective customer
job	Occupation of prospective customer
marital	Marital status of prospective customer
education	Education level of prospective customer
default	Does the prospective customer have credit in default?
housing	Does the prospective customer have a home loan?
loan	Does the prospective have a personal loan?
contact	How was the prospective customer contacted?
month	In which month was prospective customer contacted?
day_of_week	On which day of the week was the prospective customer contacted?

duration	Duration of last contact
campaign	Number of times prospective customer was contacted during campaign
previous	Number of times prospective customer was contacted before the campaign
poutcome	Outcome of the previous marketing campaign
emp.var.rate	Employment variation rate – quarterly indicator
cons.price.idx	Consumer price index – monthly indicator
cons.conf.idx	Consumer confidence index – monthly indicator
euribor3m	Euribor 3-month rate – daily indicator
nr.employed	Numer of employees – quarterly indicator

Additionally, *Table 2* provides summary statistics for all the variables in this dataset. The medians are reported for variables that have skewed distributions.

Table 2: Summary Statistics

Variable	%	Mean (Median)
Age		40.02
Job		
Admin	25.30%	
Entrepreneur	22.47%	
Housemaid	2.57%	
Management	7.10%	
Retired	4.18%	
Self-Employed	3.45%	
Services	9.64%	
Student	2.12%	
Technician	16.37%	
Unemployed	2.46%	
Unknown	0.80%	
Marital		
Divorced	11.20%	
Married	60.52%	
Single	28.09%	
Unknown	0.19%	
Education		
Basic 4 year	10.14%	
Basic 6 year	5.56%	
Basic 9 year	14.68%	
High School	23.10%	
Illiterate	0.04%	
Professional Course	12.73%	
University Degree	29.54%	
Unknown	4.20%	
Default		
No	79.12%	
Yes	0.01%	
Unknown	20.87%	

Variable	%	Mean (Median)
Housing		
No	45.21%	
Yes	52.38%	
Unknown	2.40%	
Loan		
No	82.43%	
Yes	15.17%	
Unknown	2.40%	
Contact		
Cellular	63.47%	
Telephone	36.53%	
Month		
March	1.33%	
April	6.39%	
May	33.43%	
June	12.91%	
July	17.42%	
Aug	15.00%	
September	1.38%	
October	1.74%	
November	9.96%	
December	0.44%	
Day of Week		
Monday	20.67%	
Tuesday	19.64%	
Wednesday	19.75%	
Thursday	20.94%	
Friday	19.00%	
Duration		258.29
Campaign		(2)
Previous		(0)

Previous Outcome		
Failure	10.32%	
Non-existent	86.34%	
Success	3.33%	
Emp. Var. Rate		0.08
Cons. Price Index		93.58
Cons. Conf. Index		-40.50
Euribor 3 Month Index		3.62
Number Employed		5167.04
Outcome		
No	88.73%	
Yes	11.27%	

Methods

Predicting whether or not a client will choose to subscribe to a bank term deposit is a problem of classification. In this project, we fit models using Logistic Regression, Penalized Logistic Regression, Principal Component Analysis, Random Forests, and Neural Networks.

Before fitting these models, we first had to address missing data. The observations that contained an "unknown" level for any variable, were recoded to be missing. The following variables then contained missing information: job, marital, education, default, housing, and loan. Using the *NearZeroVariance* function from the *caret* package in R we identified variables that had few unique values relative to the number of observations and variables that had large ratios of the most common value to the second most common value. These variables were *pdays* and *default*. These variables were removed from the data before analysis. The remaining missing values were imputed using *k* Nearest Neighbors, with *k* = 10.

The *duration* variable was also removed from the data. The duration of the last contact highly affects the output target (e.g., if duration=0 then response="no"). Yet, the duration is not known before a call is performed. Also, after the end of the call, the response is obviously known. This variable could be useful for benchmark purposes but won't be useful for predictive purposes [2].

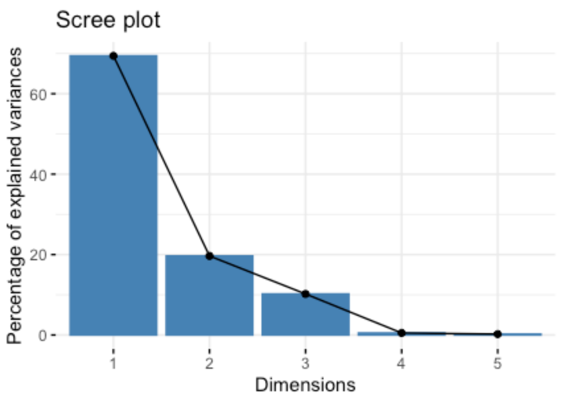
The data were also separated into training and testing data. 80% of the total data was randomly selected to be the training data and the remaining 20% was set aside to measure the performance of our models. All models were trained using the *train* function from the caret package. When using cross validation techniques, we used repeated 10-fold cross validation with 5 repeats.

Since the outcome variable in this dataset contains nearly 89% "no" responses, we recognize the data is imbalanced. To account for this imbalance, we implemented the use of optimal cut points, maximizing Cohen's Kappa Statistic. This was performed using the *optimal.cutpoints* function from the *OptimalCutpoints* package. However, results are reported with and without the use of optimal cut points. We also implemented up-sampling while training our models. Using the observations in the training data, we up-sampled from the "yes" class to have a balanced response variable. This was done with the *upSample* function from the *caret* package. This resulted in a final training dataset with 58,478 observations. The testing data contained 8,238 observations.

The first model was fit using cross validated logistic regression on the entire data set (after the removal of the variables previously mentioned).

The second model incorporated the use of principal component analysis (PCA). The economic variables (employment variation rate, consumer price index, consumer confidence index, Euribor rate, and number of employees) were there only variables used in the PCA analysis. Two components were retained, as with only two components, 89% of the cumulative variance was explained. (See scree plot in *Figure 1* below.) These components were then used in place of the economic variables in another cross validated logistic regression model.

Figure 1: PCA Loadings



The third model implemented the use of penalization in a logistic regression model. The *alpha* and *lambda* parameters were tuned using the following values given in *Table 3*.

Table 3: Tuning Values

alpha	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10
lambda	0.01	$0.03\overline{1}$	$0.05\overline{2}$	$0.07\overline{3}$	$0.09\overline{4}$	$0.11\overline{5}$	$0.13\overline{6}$	$0.15\overline{7}$	$0.17\overline{8}$	0.2	

The fourth model applied to the final data set was a random forest algorithm that creates a collection of decision tree models that were trained utilizing a 10-fold cross validation technique repeated 5 times. Accuracy was used to select the optimal model using the largest value, the tuning length or maximum number of iterations for this model was 5.

Finally, the fifth model trained was a neural network where the training data set was centered and scaled, and a 10-fold cross validation with no repeats was implemented. This is the only model where the k-fold cross validation was not repeated 5 times because of computing power limitations. This model used 10 hidden layers and the values 0, .1, 1 and 2 as regularization parameters to avoid over-fitting.

Results

The cross validated logistic regression model using optimal cut points (OPC) resulted in a Kappa statistic of 0.4101 and an overall accuracy of 0.8717. The 95% DeLong confidence interval for the area under the curve is [0.77220, 0.8075]. *Figure 2* and *Figure 3* below, show additional results.

Figure 2

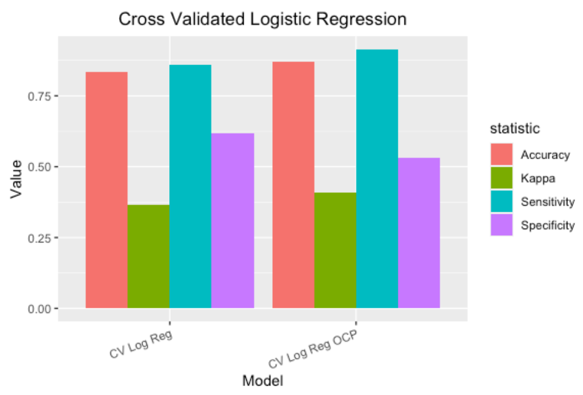
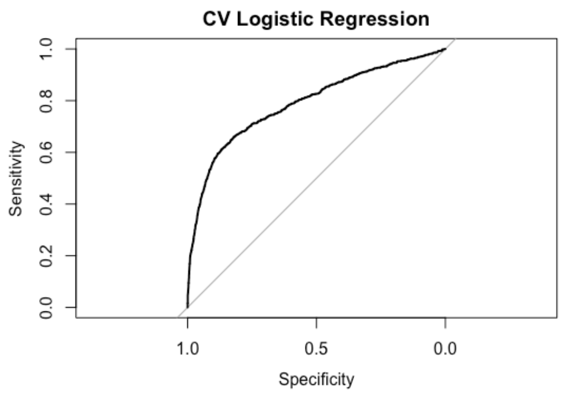


Figure 3



The cross validated logistic regression model using principal components analysis and an optimal cut point resulted in a Kappa statistic of 0.4025 and an accuracy of 0.8716. The 95% DeLong confidence interval for the area under the curve is [0.7686, 0.8035]. Additional results are given in *Figure 4* and *Figure 5*.

Figure 4

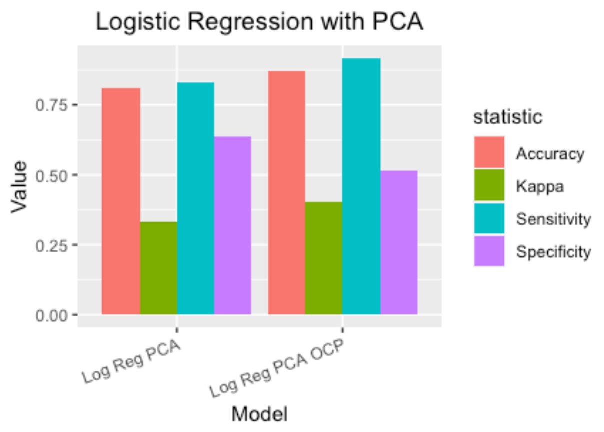
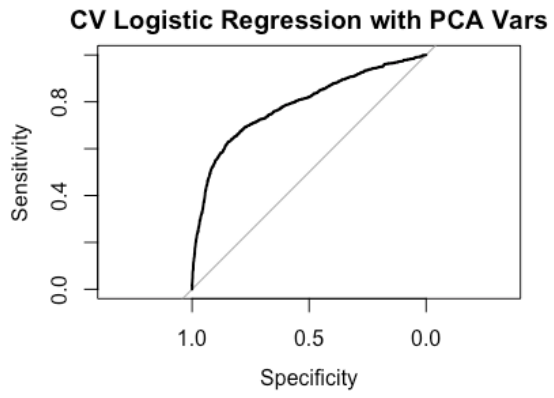


Figure 5



Out of the three statistical models, the penalized logistic regression model performed the worst. The optimal tuning parameters were *alpha* = 0.4 and *lambda* = 0.01. Using an optimal cut point, this model achieved a Kappa statistic of 0.3325 and an overall accuracy of 0.8105. The 95% DeLong confidence interval for the area under the curve is [0.7567, 0.7936]. Additional results are given in *Figure 6* and *Figure 7* below. Final set of variables selected by this model can be found in table 4.

Table 4: Penalized Logistic Regression Coefficients

Variable	***Coefficient***
(Intercept)	0.0923
age	0.0072
job	0.0187
marital	0.0575
education	0.0438
housing	(0.0113)
loan	-
contact	0.2613

month	(0.1747)
day_of_week	0.0428
campaign	(0.0862)
previous	0.1592
poutcome	0.3665
emp.var.rate	(0.5115)
cons.price.idx	0.3178
cons.conf.idx	0.1666
euribor3m	(0.0758)
nr.employed	(0.5057)

Figure 6

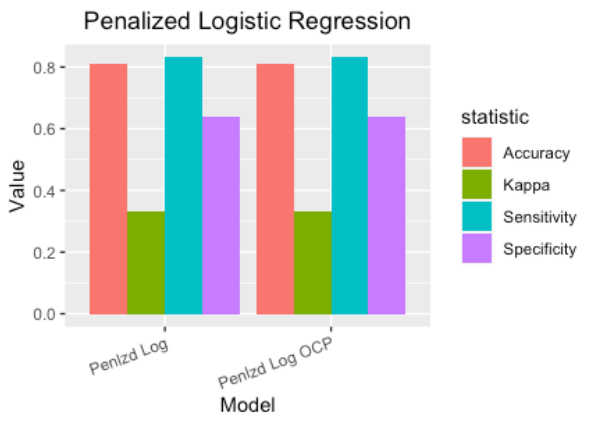
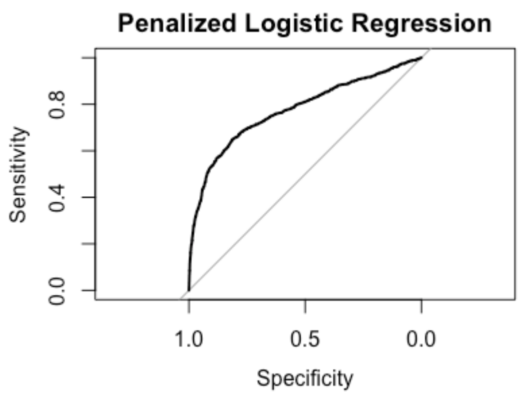


Figure 7



We also implemented the use of machine learning models. The first model implemented was a random forest model. The optimal tuning value for the number of variables (*mtry*) was 9. With an optimal cut point, the Kappa statistic from this model was 0.3530 and the overall accuracy was 0.8780. The 95% DeLong confidence interval for the area under the curve is [0.7498, 0.7860].

Figure 8

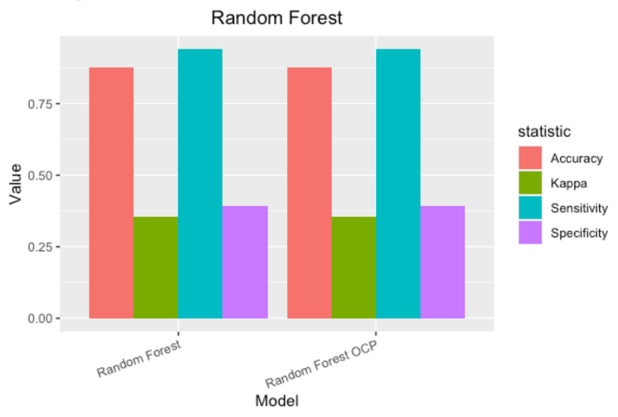


Figure 9

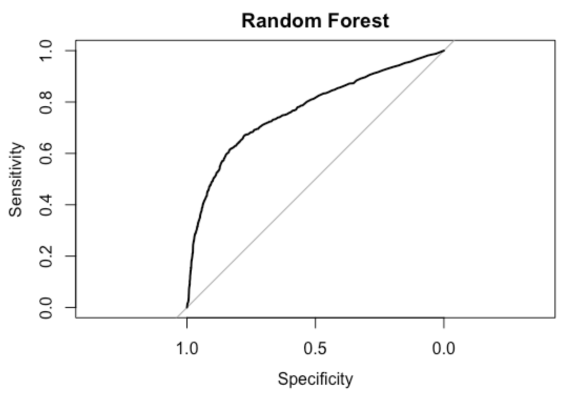
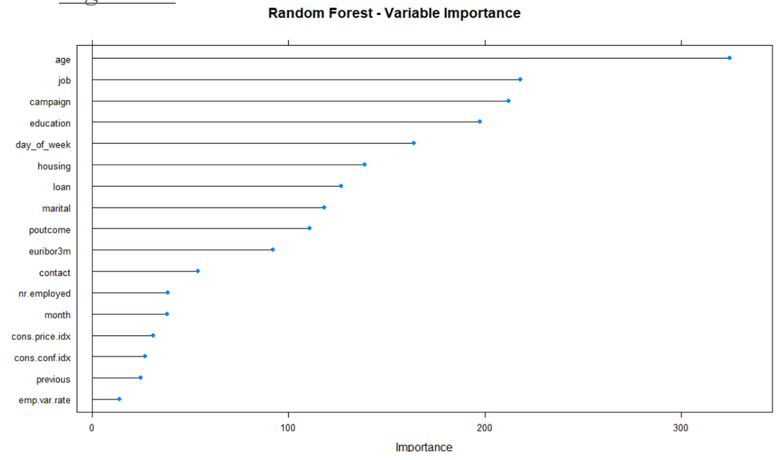


Figure 10



The last model used in this analysis was a neural network model. The optimal tuning values for this model were size = 9 and decay = 0. With an optimal cut point, the Kappa statistic achieved was 0.4034 and the accuracy was 0.8650.The 95% confidence interval for the area under the curve is [0.7614, 0.7976]. More results are given in *Figure 11* and *Figure 12*.

Figure 11

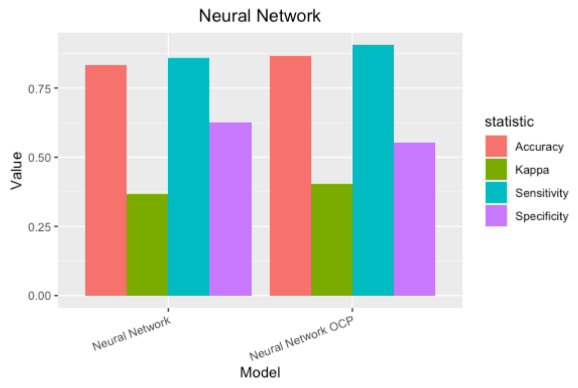


Figure 12

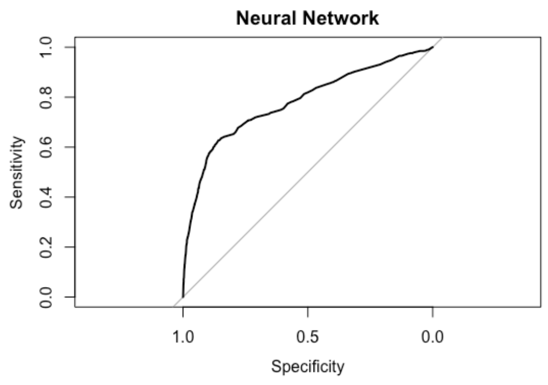
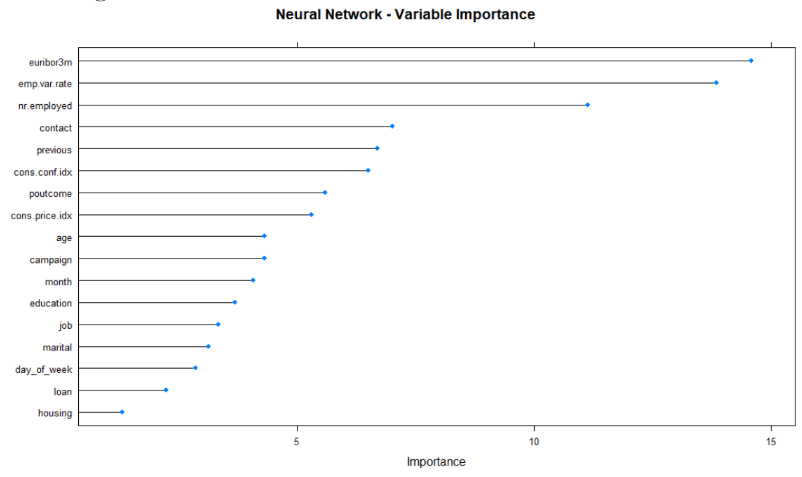


Figure 13



Conclusions and Future Work

Overall, the Kappa statistic for all models was reasonable since all of them are above 0.3 and usually a high accuracy of 90% and high expected accuracy of 85% provide a moderate Kappa of 0.3. As we can see, having an accuracy above 80% for all models can be misleading and it is important to always review specificity and sensitivity. For this specific data set, the high accuracy rates are because it was easier for the models to correctly predict if a customer won't make a term deposit, even though this information could be useful for marketing techniques, it will be more important for the bank to be able to identify characteristics of the people that could potentially become customers. Due to the challenges of this data set, we were unable to confirm which features are the most relevant, but according to the random forest model, the most important predictors are "age", "job", and "campaign", and according to the neural network model "euribor3m", "emp.var.rate", and "nr.employed", which are the economic variables, are the most important.

This leads us to conclude that additional techniques to deal with highly imbalanced data need to be considered and evaluated with these models. In general, the results of the Neural Network, even though it's accuracy and Kappa statistic were not the best, the sensitivity (90.4%) and specificity (55%) rates combined, outperformed the other models. Also, it is likely that other machine learning methods could perform better in this data set, and a deeper study of the configuration of the optimal parameters or hyperparameters for each method will need to be evaluated in the future.

References

1. Federal Reserve Bank. (2001, July 1). What is the economic function of a bank? Retrieved May 3, 2020, from https://www.frbsf.org/education/publications/doctor-econ/2001/july/bank-economic-function/
2. [Moro et al., 2014] S. Moro, P. Cortez, and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, In press, http://dx.doi.org/10.1016/j.dss.2014.03.001
3. Kagan, J. (2020, April 21). Understanding Commercial Banks. Retrieved May 3, 2020, from https://www.investopedia.com/terms/c/commercialbank.asp
4. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: with applications in R. New York: Springer.

Appendix

```
library(VIM)
library(caret)
library(ggplot2)
library(corrplot)
library(Hmisc)
library(DMwR)
library(factoextra)
library(OptimalCutpoints)
library(cluster)
library(pROC)
library(tidyr)
library(randomForest)


# Read in file
market <- read.csv("/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/bank-additional/bank-additional-full.csv",
                sep = ";")
# 21 variables and 41,188 observations
dim(market)

# There are 39,673 observations that have the value 999 for the pdays variable
# These are missing values
table(market$pdays)

# pdays variable identified as near zero variance
# this variable should be removed
nearZeroVar(market, names = TRUE)

# Remove the pdays variable from data
market <- market[, -which(colnames(market) == "pdays")]
colnames(market)

###### JOB ########
table(market$marital)
# set the "unknown" levels to missing
for (i in 1:nrow(market)){
  if(market$job[i] == "unknown"){
    market$job[i] <- NA
  }
}
# 330 missing values
length(which(is.na(market$job) == TRUE))

###### MARITAL ########
table(market$marital)
# set the "unknown" levels to missing
for (i in 1:nrow(market)){
  if(market$marital[i] == "unknown"){
    market$marital[i] <- NA
  }
}
# 80 missing values
length(which(is.na(market$marital) == TRUE))

###### EDUCATION ########
table(market$education)
# set the "unknown" levels to missing
for (i in 1:nrow(market)){
  if(market$education[i] == "unknown"){
    market$education[i] <- NA
  }
}
# 1,731 missing values
length(which(is.na(market$education) == TRUE))

###### DEFAULT ########
table(market$default)
# set the "unknown" levels to missing
for (i in 1:nrow(market)){
  if(market$default[i] == "unknown"){
    market$default[i] <- NA
  }
}
# 8,597 missing values
```

```
length(which(is.na(market$default) == TRUE))

###### HOUSING ########
table(market$housing)
# set the "unknown" levels to missing
for (i in 1:nrow(market)){
  if(market$housing[i] == "unknown"){
    market$housing[i] <- NA
  }
}
# 990 missing values
length(which(is.na(market$housing) == TRUE))

###### LOAN ########
table(market$loan)
# set the "unknown" levels to missing
for (i in 1:nrow(market)){
  if(market$loan[i] == "unknown"){
    market$loan[i] <- NA
  }
}
# 990 missing values
length(which(is.na(market$loan) == TRUE))

nearZeroVar(market, names = TRUE)

# Remove the default variable from data
market <- market[, -which(colnames(market) == "default" | colnames(market) ==
"duration")]
colnames(market)

# Use K Nearest Neighbors to impute the missing data
market2 <- VIM::kNN(market, variable = c("job", "marital", "education", "housing",
"loan"), k = 10)
#saveRDS(market2, "/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/market2.rds")
#market2 <- readRDS("/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/market2.rds")
```

```
# Drop the boolean columns created from imputation
colnames(market2)
market2 <- market2[, -c(19:23)]

# Now there are no missisng values
market2[!complete.cases(market2),]

age_plot <- ggplot(data = market2, aes(x = age)) +
  geom_histogram(bins = 30) +
  labs(title = "Distribution of Age", x = "Age")

job_plot <- ggplot(data = market2, aes(x = job)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 20, hjust = 1)) +
  labs(title = "Job", x = "Job Type")

marital_plot <- ggplot(data = market2, aes(x = marital)) +
  geom_bar() +
  labs(title = "Marital Status", x = "Marital Status")

education_plot <- ggplot(data = market2, aes(x = education)) +
  geom_bar()+
  labs(title = "Education", x = "Education Level") +
  theme(axis.text.x = element_text(angle = 20, hjust = 1))

housing_plot <- ggplot(data = market2, aes(x = housing)) +
  geom_bar() +
  labs(title = "Housing", x = "Has a Home Loan")

loan_plot <- ggplot(data = market2, aes(x = loan)) +
  geom_bar() +
  labs(title = "Personal Loan", x = "Has a Personal Loan")

contact_plot <- ggplot(data = market2, aes(x = contact)) +
  geom_bar() +
  labs(title = "Contact", x = "Type of Contact")
```

```
month_plot <- ggplot(data = market2, aes(x = month)) +
  geom_bar() +
  labs(title = "Month", x = "Month of Contact")

day_plot <- ggplot(data = market2, aes(x = day_of_week)) +
  geom_bar() +
  labs(title = "Day of Weed", x = "Day of Contact")

campaign_plot <- ggplot(data = market2, aes(x = campaign)) +
  geom_histogram(bins = 25) +
  labs(title = "Number of Contacts During Campaign", x = "# Contacts")

previous_plot <- ggplot(data = market2, aes(x = previous)) +
  geom_histogram(bins = 8) +
  labs(title = "Number of Contacts Prior to Campaign", x = "# Contacts")

poutcome_plot <- ggplot(data = market2, aes(x = poutcome)) +
  geom_bar() +
  labs(title = "Outcome of Previous Contact", x = "Outcome")

emp.var.rate_plot <- ggplot(data = market2, aes(x = emp.var.rate)) +
  geom_histogram(bins = 10) +
  labs(title = "Employment Variation Rate", x = "Rate")

cons.price.idx_plot <- ggplot(data = market2, aes(x = cons.price.idx)) +
  geom_histogram(bins = 20) +
  labs(title = "Consumer Price Index", x = "Index")

cons.conf.idx_plot <- ggplot(data = market2, aes(x = cons.conf.idx)) +
  geom_histogram(bins = 10) +
  labs(title = "Consumer Confidence Index", x = "Index")

euribor3m_plot <- ggplot(data = market2, aes(x = euribor3m)) +
  geom_histogram(bins = 10) +
  labs(title = "Euribor 3 Month Rate", x = "Rate")

nr.employed_plot <- ggplot(data = market2, aes(x = nr.employed)) +
  geom_histogram(bins = 11) +
  labs(title = "Number of Employees at Bank", x = "Number of Employees")

outcome_plot <- ggplot(data = market2, aes(x = y)) +
  geom_bar() +
  labs(title = "Did the customer subscribe to bank term deposit?", x = "Response")


# Change binary variables to levels of 0 and 1 instead of character levels
market2$housing <- ifelse(market2$housing == "no", 0, 1)
market2$loan <- ifelse(market2$loan == "no", 0, 1)
market2$contact <- ifelse(market2$contact == "telephone", 0, 1)

# Change outcome variable to a factor variable
market2$y <- as.factor(market2$y)
levels(market2$y)

# Subset only the numeric variables for correlation plot
numeric <- c("age", "housing", "loan", "contact", "campaign", "previous", "emp.var.rate",
            "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed")
market2_num <- market2[, numeric]
correlation_df <- data.frame(market2_num, "outcome" = ifelse(market2$y == "yes", 1,
0))
correlation_df <- correlation_df[, -12]
head(correlation_df)
# Matrix of correlations, rounded to 4 decimal places
res <- round(cor(correlation_df), 4)
# Correlation plot
corrplot(res, type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)



# Create training and testing data
set.seed(869)
train_index <- sample(1:nrow(market2), 0.8 * nrow(market2))
test_index <- setdiff(1:nrow(market2), train_index)

pre_market_train <- market2[train_index, ]
market_test <- market2[test_index, ]
```

```
# Up-Sampling
set.seed(155)
market_train <- upSample(x = pre_market_train[, -which(colnames(pre_market_train) ==
"y")],
                         y = pre_market_train[, which(colnames(pre_market_train) == "y")])
colnames(market_train)[18] <- "y"

table(market_train$y)
dim(market_test)

# Logistic Regression Using Cross Validation
ctrl1 <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 5,
                      savePredictions = TRUE,
                      classProbs = TRUE)
set.seed(869)
logisticRegCV <- train(y ~ ., data = market_train,
                       method = "glm",
                       family = "binomial",
                       trControl = ctrl1)
saveRDS(logisticRegCV, "/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/logisticregcv.rds")
#logisticRegCV <- readRDS("/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/logisticregcv.rds")



# Create dataframe with predictions
cvLog_results <- data.frame(obs = market_test$y)
cvLog_up_results <- data.frame(obs = market_test$y)
# probabilities
cvLog_results$prob <- predict(logisticRegCV, market_test, type = "prob")[,2]
cvLog_up_results$prob <- predict(logisticRegCV_up, market_test, type = "prob")[,2]
# predictions
cvLog_results$pred <- predict(logisticRegCV, market_test)
cvLog_up_results$pred <- predict(logisticRegCV_up, market_test)
```

```
# Confusion matrix using original predictions
model1cm <- confusionMatrix(data = cvLog_results$pred, reference =
cvLog_results$obs)
model1cm_up <- confusionMatrix(data = cvLog_up_results$pred, reference =
cvLog_up_results$obs)
model1_accuracy <- model1cm$overall["Accuracy"]
model1_kappa <- model1cm$overall["Kappa"]
model1_sens <- model1cm$byClass["Sensitivity"]
model1_spec <- model1cm$byClass["Specificity"]

# New cutpoint
cutpoint1 <- summary(optimal.cutpoints(X = "prob", status = "obs", cvLog_results,
                        tag.healthy = "no", methods = "MaxKappa"))
final_cutpoint1 <- cutpoint1$MaxKappa$Global$optimal.cutoff$cutoff
final_cutpoint1
# Create new predictions using optimal cutpoint
cvLog_results$new_pred <- as.factor(ifelse(cvLog_results$prob > final_cutpoint1, "yes",
"no"))
head(cvLog_results)
# New confusion matrix
model1cm2 <- confusionMatrix(data = cvLog_results$new_pred, reference =
cvLog_results$obs)
model1_accuracy2 <- model1cm2$overall["Accuracy"]
model1_kappa2 <- model1cm2$overall["Kappa"]
model1_sens2 <- model1cm2$byClass["Sensitivity"]
model1_spec2 <- model1cm2$byClass["Specificity"]

########## Principal Component Analysis ###########

# economic variables
econ <- c("emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed")
econ_df <- market2[, econ]

# Principal Component Analysis
econ_pca <- prcomp(econ_df, center = T, scale = T)
summary(econ_pca)

# Scree Plot
```

```
fviz_eig(econ_pca)

# Visualization -- Dimension Reduction
fviz_pca_ind(econ_pca, label = "none", habillage = market2$y,
          addEllipses = TRUE, ellipse.level = 0.95, palette = "Dark1", axes = c(1,2))

# Merge 2 columns of PCA variables with market data
colnames(market2)
pca_data <- data.frame(market2[, -c(13:17)], econ_pca$x[, 1:2])
pre_pca_train <- pca_data[train_index,]
pca_test <- pca_data[test_index, ]

pca_train <- upSample(x = pre_pca_train[, -which(colnames(pre_pca_train) == "y")],
                   y = pre_pca_train[, which(colnames(pre_pca_train) == "y")])
colnames(pca_train)[15] <- "y"
colnames(pca_train)

# Train logistic regression model with the new pca variables
set.seed(869)
logistic_pca <- train(y ~ ., data = pca_train,
                     method = "glm",
                     family = "binomial",
                     trControl = ctrl1)

saveRDS(logistic_pca, "/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/logistic_pca.rds")
#logistic_pca <- readRDS("/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/logistic_pca.rds")

# Create dataset with results
log.pca_results <- data.frame(obs = pca_test$y)
# Predicted probabilities
log.pca_results$prob <- predict(logistic_pca, newdata = pca_test, type = "prob")[,2]
# Predicted outcomes
log.pca_results$pred <- predict(logistic_pca, pca_test)
# Confusion matrix
model2cm <- confusionMatrix(data = log.pca_results$pred, reference =
log.pca_results$obs)
```

```
model2_accuracy <- model2cm$overall["Accuracy"]
model2_kappa <- model2cm$overall["Kappa"]
model2_sens <- model2cm$byClass["Sensitivity"]
model2_spec <- model2cm$byClass["Specificity"]

# Find optimal cutpoint
cutpoint2 <- summary(optimal.cutpoints(X = "prob", status = "obs", log.pca_results,
                                tag.healthy = "no", methods = "MaxKappa"))
final_cutpoint2 <- cutpoint2$MaxKappa$Global$optimal.cutoff$cutoff
final_cutpoint2
# Re-label predictions based on new cutpoint
log.pca_results$new_pred <- as.factor(ifelse(log.pca_results$prob > final_cutpoint2,
"yes", "no"))
# Confusion matrix with cutpoint
model2cm2 <- confusionMatrix(data = log.pca_results$new_pred, reference =
log.pca_results$obs)
model2_accuracy2 <- model2cm2$overall["Accuracy"]
model2_kappa2 <- model2cm2$overall["Kappa"]
model2_sens2 <- model2cm2$byClass["Sensitivity"]
model2_spec2 <- model2cm2$byClass["Specificity"]



########## Penalized Logistic Regression ############

glmnGrid <- expand.grid(alpha = seq(0, 1, length = 11),
                lambda = seq(0.01, 0.2, length = 10))
set.seed(869)
penalize.log <- train(x = data.matrix(market_train[, -which(colnames(market_train) ==
"y")]),
              y = market_train[, which(colnames(market_train) == "y")],
              method = "glmnet",
              tuneGrid = glmnGrid,
              preProc = c("center", "scale"),
              family = "binomial",
              trControl = ctrl1)
penalize.log$bestTune
```

```
saveRDS(penalize.log, "/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/penalize.log.rds")
#penalize.log <- readRDS("/Users/rob.pruette/Documents/SMU Spring 2020/STAT
6309/Project1/penalize.log.rds")

# Create dataframe with predictions
pn.log_results <- data.frame(obs = market_test$y)
# probabilities
pn.log_results$prob <- predict(penalize.log, data.matrix(market_test), type = "prob")[,2]
# predictions
pn.log_results$pred <- predict(penalize.log, data.matrix(market_test))
# Confusion matrix using original predictions
model3cm <- confusionMatrix(data = pn.log_results$pred, reference =
pn.log_results$obs)
model3_accuracy <- model2cm$overall["Accuracy"]
model3_kappa <- model2cm$overall["Kappa"]
model3_sens <- model2cm$byClass["Sensitivity"]
model3_spec <- model2cm$byClass["Specificity"]


# optimal cutpoint
cutpoint3 <- summary(optimal.cutpoints(X = "prob", status = "obs", pn.log_results,
                              tag.healthy = "no", methods = "MaxKappa"))
final_cutpoint3 <- cutpoint3$MaxKappa$Global$optimal.cutoff$cutoff
final_cutpoint3
# make new predictions using cutpoint
pn.log_results$new_pred <- as.factor(ifelse(pn.log_results$prob > final_cutpoint3, "yes",
"no"))
# Confusion matrix with new predictions
model3cm2 <- confusionMatrix(data = pn.log_results$new_pred, reference =
pn.log_results$obs)
model3_accuracy2 <- model2cm$overall["Accuracy"]
model3_kappa2 <- model2cm$overall["Kappa"]
model3_sens2 <- model2cm$byClass["Sensitivity"]
model3_spec2 <- model2cm$byClass["Specificity"]


# Random Forest Model
```

```
ctrlRF <- trainControl(method = "repeatedcv",
                    classProbs = TRUE,
                    number = 10,
                    repeats=5,
                    savePredictions = TRUE,
                    search = 'random')
set.seed(869)
 Random.Forest <- train(x = data.matrix(market_train[, -which(colnames(market_train)
== "y")]),
                    y = market_train[, which(colnames(market_train) == "y")],
                    method = "rf",
                    ntree = 500,
                    tuneLenght = 5,
                    importance = TRUE,
                    metric = "Accuracy",
                    trControl = ctrlRF)

Random.Forest$bestTune
# Create dataframe with predictions
Random.Forest_results <- data.frame(obs = market_test$y)
# probabilities
Random.Forest_results$prob <- predict(Random.Forest, data.matrix(market_test), type =
"prob")[,2]
# predictions
Random.Forest_results$pred <- predict(Random.Forest, data.matrix(market_test))
# Confusion matrix using original predictions
model4cm <- confusionMatrix(data = Random.Forest_results$pred, reference =
Random.Forest_results$obs)
model4_accuracy <- model4cm$overall["Accuracy"]
model4_kappa <- model4cm$overall["Kappa"]
model4_sens <- model4cm$byClass["Sensitivity"]
model4_spec <- model4cm$byClass["Specificity"]
model4cm


# optimal cutpoint
cutpoint4 <- summary(optimal.cutpoints(X = "prob", status = "obs",
Random.Forest_results,
                             tag.healthy = "no", methods = "MaxKappa"))
final_cutpoint4 <- cutpoint4$MaxKappa$Global$optimal.cutoff$cutoff
final_cutpoint4

# make new predictions using cutpoint
Random.Forest_results$new_pred <- as.factor(ifelse(Random.Forest_results$prob >
final_cutpoint4, "yes", "no"))
# Confusion matrix with new predictions
confusionMatrix(data = Random.Forest_results$new_pred, reference =
Random.Forest_results$obs)
model4cm2 <- confusionMatrix(data = Random.Forest_results$pred, reference =
Random.Forest_results$obs)
model4_accuracy2 <- model4cm2$overall["Accuracy"]
model4_kappa2 <- model4cm2$overall["Kappa"]
model4_sens2 <- model4cm2$byClass["Sensitivity"]
model4_spec2 <- model4cm2$byClass["Specificity"]
model4cm2


# Neural Networks

 nnetGrid <- expand.grid(size = c(1:10), decay = c(0, .1, 1, 2))
 maxSize <- max(nnetGrid$size)

 ctrlNN <- trainControl(method = "repeatedcv",
                  summaryFunction = multiClassSummary,
                  classProbs = TRUE,
                  savePredictions = TRUE,
                  search = 'grid')

 set.seed(869)
 Neural.Network <- train(data.matrix(market_train[, -which(colnames(market_train) ==
"y")]),
                  y = market_train[, which(colnames(market_train) == "y")],
                  method = "nnet",
                  metric = "Accuracy",
                  preProc = c("center", "scale"),
                  tuneGrid = nnetGrid,
                    repeats = 10,
                    trace = FALSE,
                    maxit = 2000,
                    MaxNWts = 10*(maxSize * (ncol(market_train) + 1) + maxSize + 1),
                    allowParallel = FALSE,
                    trControl = ctrlNN)

Neural.Network$bestTune
# Create dataframe with predictions
Neural.Network_results <- data.frame(obs = market_test$y)
# probabilities
Neural.Network_results$prob <- predict(Neural.Network, data.matrix(market_test), type
= "prob")[,2]
# predictions
Neural.Network_results$pred <- predict(Neural.Network, data.matrix(market_test))
# Confusion matrix using original predictions
model5cm <- confusionMatrix(data = Neural.Network_results$pred, reference =
Neural.Network_results$obs)
model5_accuracy <- model5cm$overall["Accuracy"]
model5_kappa <- model5cm$overall["Kappa"]
model5_sens <- model5cm$byClass["Sensitivity"]
model5_spec <- model5cm$byClass["Specificity"]
model5cm

# optimal cutpoint
cutpoint5 <- summary(optimal.cutpoints(X = "prob", status = "obs",
Neural.Network_results,
                              tag.healthy = "no", methods = "MaxKappa"))
final_cutpoint5 <- cutpoint5$MaxKappa$Global$optimal.cutoff$cutoff
final_cutpoint5

# make new predictions using cutpoint
Neural.Network_results$new_pred <- as.factor(ifelse(Neural.Network_results$prob >
final_cutpoint5, "yes", "no"))
# Confusion matrix with new predictions
model5cm2 <- confusionMatrix(data = Neural.Network_results$new_pred, reference =
Neural.Network_results$obs)
model5_accuracy2 <- model5cm2$overall["Accuracy"]
```

```
model5_kappa2 <- model5cm2$overall["Kappa"]
model5_sens2 <- model5cm2$byClass["Sensitivity"]
model5_spec2 <- model5cm2$byClass["Specificity"]
model5cm2


# Calibration plots
cal_data <- data.frame(LogReg = predict(logisticRegCV, market_test, type = "prob")[,2])
cal_data$LogPCA <- predict(logistic_pca, pca_test, type = "prob")[,2]
cal_data$LogPenalized <- predict(penalize.log, data.matrix(market_test), type =
"prob")[,2]
cal_data$RandomForest <- predict(Random.Forest, data.matrix(market_test), type =
"prob")[,2]
cal_data$NeuralNetwork <- predict(Neural.Network, data.matrix(market_test), type =
"prob")[,2]
cal_data$class <- market_test[, 18]

calibration_results <- calibration(class ~ LogReg + LogPCA + LogPenalized +
RandomForest + NeuralNetwork, data = cal_data, cuts = 10, class = "yes")
xyplot(calibration_results, auto.key = list(columns = 3))

model1ROC <- roc(cvLog_results$obs, cvLog_results$prob)
plot(model1ROC, main ="CV Logistic Regression")
auc(model1ROC)
ci.auc(model1ROC)

model2ROC <- roc(log.pca_results$obs, log.pca_results$prob)
plot(model2ROC, main = "CV Logistic Regression with PCA Vars")
auc(model2ROC)
ci.auc(model2ROC)

model3ROC <- roc(pn.log_results$obs, pn.log_results$prob)
plot(model3ROC, main = "Penalized Logistic Regression")
auc(model3ROC)
ci.auc(model3ROC)

model4ROC <- roc(Random.Forest_results$obs, Random.Forest_results$prob)
plot(model4ROC, main = "Random Forest")
```

```
auc(model4ROC)
ci.auc(model4ROC)

model5ROC <- roc(Neural.Network_results$obs, Neural.Network_results$prob)
plot(model5ROC, main = "Neural Network")
auc(model5ROC)
ci.auc(model5ROC)

# Results dataframe
# Results dataframe
results_df <- data.frame(Model = c("CV Log Reg", "CV Log Reg OCP", "Log Reg
PCA", "Log Reg PCA OCP", "Penlzd Log", "Penlzd Log OCP", "Random Forest",
"Random Forest OCP", "Neural Network", "Neural Network OCP"))
results_df$Kappa <- c(model1_kappa, model1_kappa2, model2_kappa, model2_kappa2,
model3_kappa, model3_kappa2, model4_kappa, model4_kappa2, model5_kappa,
model5_kappa2)
results_df$Accuracy <- c(model1_accuracy, model1_accuracy2, model2_accuracy,
model2_accuracy2, model3_accuracy, model3_accuracy2, model4_accuracy,
model4_accuracy2, model5_accuracy, model5_accuracy2)
results_df$Sensitivity <- c(model1_sens, model1_sens2, model2_sens, model2_sens2,
model3_sens, model3_sens2, model4_sens, model4_sens2, model5_sens, model5_sens2)
results_df$Specificity <- c(model1_spec, model1_spec2, model2_spec, model2_spec2,
model3_spec, model3_spec2, model4_spec, model4_spec2, model5_spec, model5_spec2)
results_df$Cutpoint <- c("Default", "Optimal", "Default", "Optimal", "Default",
"Optimal", "Default", "Optimal", "Default", "Optimal")
results_df

gather.df <- gather(results_df, key = "statistic", value = "value", Accuracy, Sensitivity,
Specificity, Kappa, Cutpoint)
gather.df2 <- gather.df[-which(gather.df$statistic == "Cutpoint"),]
#gather.df2 <- gather.df2[-which(gather.df2$Cutpoint == "Default"),]

model1_gather <- gather.df2[which(gather.df2$Model == "CV Log Reg" |
gather.df2$Model == "CV Log Reg OCP"),]
model2_gather <- gather.df2[which(gather.df2$Model == "Log Reg PCA" |
gather.df2$Model == "Log Reg PCA OCP"),]
model3_gather <- gather.df2[which(gather.df2$Model == "Penlzd Log" |
gather.df2$Model == "Penlzd Log OCP"),]
model4_gather <- gather.df2[which(gather.df2$Model == "Random Forest" |
gather.df2$Model == "Random Forest OCP"),]
```

```
model5_gather <- gather.df2[which(gather.df2$Model == "Neural Network" |
gather.df2$Model == "Neural Network OCP"),]


ggplot(model5_gather, aes(fill = statistic, x = Model, y = as.numeric(value)))+
  geom_bar(position = "dodge", stat = "identity")+
  scale_y_continuous() +
  theme(axis.text.x = element_text(angle = 20, hjust = 1)) +
  labs(title = "Neural Network", y = "Value")+
  theme(plot.title = element_text(hjust = 0.5))
```