title: 'Notebook 2: Regression' author: "Group 10 (Umar, Cory, Carolline, Benji)" date: October 08, 2022 output: pdf_document: default

# html_notebook: default

## Packages & Libraries

```{r} library(corrplot) library(ggplot2) library(caret) library(DMwR) library(mlbench) library(ROCR)

```
### Preprocessing

```{r}
data <- read.csv("bank-additional-full.csv",sep=";")
head(data)
```

showing row count ```{r} cat("Original row count is",nrow(data))

```
dropping 'duration','pdays','default'
```{r}
data <- subset(data, select = -c(duration, pdays, default))
colnames(data)
```

find the columns that have "unknown" as a value ```{r} has_unknown <- function(df, col) { (sum(df[,col]=="unknown")>0) } where_is_unknown <- function(df) { c <- 0 for(col in colnames(df)) { if (has_unknown(df,col)) { cat(col) c <- c + 1 } } if(c == 0) { print(paste("There are no unknown values in the dataframe")) } } cat("These columns have unknown values") where_is_unknown(data)

```
replace those "unknown" values with a sample of the other values in those columns
```{r}
replace_unknowns <- function(df) {
  for(col in colnames(df)) {
    if(has_unknown(df,col)) {
      n_unk <- sum(df[,col]=="unknown")
      idx <- which(df[,col]=="unknown")
      df[idx,col] <- sample(col[!col=="unknown"],n_unk,replace=TRUE)
    }
  }
  df
}
data <- replace_unknowns(data)
head(data)
```

just to make sure those values were actually replaced and now there's no more unknowns ```{r} cat("# of unknowns:", sum(data=="unknown"))

```
show the categorical columns
```{r}
cats <- names(data)[sapply(data,is.character)]
cats
```

encode all of the categorical values into numericals ```{r} encode <- function(df,col) { as.numeric(factor(df[,col]))-1 } for(cat in cats) { data[,cat] <- encode(data,cat) } head(data)

```
 count plot for target to check for imbalance
```{r}
plt <- ggplot(data, aes(x = y)) + geom_bar(size=6)
plt + ggtitle("\t\t\t\tClass Imbalance in y")
```

Fixing the class imbalance by oversampling & resetting dataframe ```{r} data$y <- factor(data$y) data <- upSample(x=data[,-ncol(data)],y=data$y) names(data)[names(data)=='Class'] <- 'y'

```
 count plot for target to ensure classes have been balanced
```{r}
plt <- ggplot(data, aes(x = y)) + geom_bar(size=6)
plt + ggtitle("\t\t\t\tClass Balance in y")
```

checking for NA's ```{r} print(paste("# of NA's", sum(is.na(data))))

```
checking for unknowns
```{r}
where_is_unknown(data)
```

since there are no unknown values and no NA's, lastly checking for increase in row count in dataframe ```{r} cat("New row count is",nrow(data))

```
#### Train Test Split

splitting 80/20 into train and test
```{r}
set.seed(3)
i <- sample(1:nrow(data), nrow(data)*0.80, replace = FALSE)
train <- data[i,]
test <- data[-i,]
dim(train)
dim(test)
```

## General EDA

plotting correlation matrix ```{r} train$y <- encode(train,'y') C <- cor(train) corrplot(C, method='color', order = 'alphabet')
```

```
looking more specifically relative to 'y'
```{r}
C[,'y']
```

it looks like euribor3m, nr.employed, emp.var.rate have the strongest negative correlation with y, and previous and poutcome, have the strongest positive correlation.

## Linear Regression

to explore more, making base model for all independents against 'y' ```{r} set.seed(10) base <- lm(y~.,data=train) summary(base)

```
the insignificants appear to be age, job, housing, and loan, removing these and redoing model
```{r}
set.seed(11)
m1 <- lm(y~.,data=subset(train,select = -c(age,job,housing,loan)))
summary(m1)
```

every predictor is significant to the regression, but still R^2 is low. let's see if scaling the independents help at all ```{r} minMaxScaler <- function(v) { (v-min(v))/(max(v)-min(v)) }

```
```{r}
train_scaled <- subset(train,select = -c(age,job,housing,loan))
train_scaled[,-ncol(train_scaled)] <- lapply(train_scaled[, -ncol(train_scaled)],minMaxScaler)
head(train_scaled)
```

```{r} set.seed(13) m2 <- lm(y~.,data=train_scaled) summary(m2)

```
```{r}
par(mfrow=c(2,2))
plot(m2)
par(mfrow=c(1,1))
```

true to form, scaling does not impact the regression results. Both R^2 and the adjusted R^2 are staying low. we will use the model 'm1' for prediction as 'm2' doesn't appear to be doing much better.

### Linear Regression Evaluation

create model for 'm1' once again (no scaling) ```{r} set.seed(11) m1 <- lm(y~.,data=subset(train,select = -c(age,job,housing,loan))) summary(m1)

```
get predictions and measure accuracy
```{r}
set.seed(21)
probas <- predict(m1, newdata = test, type="response")
pred <- ifelse(probas>0.5, 1, 0)
acc <- mean(pred==test$y)
cat("Accuracy = ", round(acc,3))
cat("Correlation = ",round(cor(probas,as.numeric(test$y)-1),3))
```

construct confusion matrix for easy visualization ```{r} confusionMatrix(as.factor(pred), reference=test$y)

```
construct ROC plot & show auc
```{r}
set.seed(31)
pred <- prediction(probas,test$y)
roc <- performance(pred, measure="tpr", x.measure="fpr")
plot(roc)
cat("auc =", round(performance(pred,measure="auc")@y.values[[1]],3))
```

Overall, the linear regression model was predicting at around 73% accuracy. The correlation is around 51%. The model seemed to be able to distinguish between positive and negative classes for the 'y' feature though, with auc of about 78%.

Let's see if KNN regression can do more justice.

## KNN Regression

For KNN we will use the scaled version of the training sample. We will need to scale everything in test as well except for the the target 'y'

```{r} test_scaled <- subset(test,select = -c(age,job,housing,loan)) test_scaled[,-ncol(test_scaled)] <- lapply(test_scaled[, -ncol(test_scaled)],minMaxScaler)

```
Fit the scaled training data to the model

```{r}
kr1 <- knnreg(train_scaled[,-ncol(train_scaled)], train_scaled[,'y'])
pred <- predict(kr1, test_scaled[,-ncol(test_scaled)])
cor_kr1 <- cor(pred,as.numeric(test$y)-1)
cat("correlation=", round(cor_kr1,3))
```

Correlation went up, let's see if we can select the best k to maximize it & minimize mse

`{r} cor_kr <- rep(0,40) mse_kr <- rep(0,40) test_y_num <- encode(test,'y') i <- 1 for(k in seq(1,39,1)) { fit_k <- knnreg(train_scaled[,-ncol(train_scaled)], train_scaled$y, k=k) pred_k <- predict(fit_k, test_scaled[,-ncol(test_scaled)]) cor_kr[i] <- cor(pred_k,test_y_num) mse_kr[i] <- mean((pred_k - test_y_num)^2) print(paste("k = ", k, cor_kr[i], mse_kr[i])) i <- i+1 }`

While k=1 produces the largest correlation, k = 3, produces the smallest mse. See that k = 3, produces the second highest correlation. We will use k = 3 to train again.
```