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

html_notebook: default

Introduction

About this set

[Bank marketing](#) was downloaded from [archive.ics.uci.edu](#). The set includes direct marketing campaign phone calls from a portugese banking institution.

80:20 Training and Test Sets

In the code block below users can obtain the code used to read a dataset in csv format and install the accompanying tools to split dataset into training and testing sets.

```
``{r}
```

Code to split data into training and test datasets

Importing data sets

```
dataset = read.csv('bank-additional-full.csv')
```

Installing tools to ensure we can split dataset

```
install.packages('caTools') library(caTools)
```

Splitting dataset into the training set and test set

```
split = sample.split(dataset$deposit, SplitRatio = 0.8) training_set = subset(dataset, split == TRUE) test_set = subset(dataset, split == FALSE)
```

```
# R Functions for Data Exploration
```


Now that we have training and test data we're going to use some built in R functions to explore our

- Dim()
- Glimpse()
- Summary()
- Create_Report()
- Skim()

```
### Dim Function
```

The dim function outputs the dimensions of the dataset.

```
```{r}
dim(training_set)
```



## Glimpse Function

The dplyr package's **glimpse** function will show a vertical preview of the dataset. Users can preview data types and sample data.

```
```{r}
```

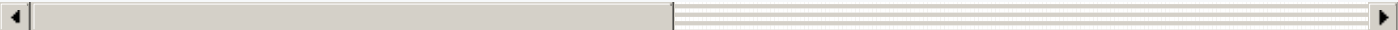
Installing Dplyr package

```
install.packages('dplyr') library(dplyr) glimpse(training_set)
```

```
### Summary Function
```

The **summary** function displays the data type and a few additional details for each column in the data

```
```{r}
summary(training_set)
```



## Create\_Report Function

The DataExplorer package's **create\_report** function is a one-line function that will retrieve the whole data profile of your data frame. It will generate an html page containing your data frame's fundamental statistics, structure, missing data, distribution visualizations, correlation matrix, and principal component analysis.

```
```{r}
```

Installing DataExplorer package

```
install.packages('DataExplorer') library(DataExplorer) DataExplorer::create_report(training_set)
```

Skim Function

The **skim** function from the **skimr** package is a useful addition to the **summary** function. It presents

```
```{r}
#Installing DataExplorer package
install.packages('skimr')
library(skimr)
skim(training_set)
```

## Informative Graph

---

### Creating a ggplot

```
```{r} install.packages('ggplot2') library(ggplot2)
```

geom_point

```
ggplot(data = training_set) + geom_point(mapping = aes(x = age, y = job))
```

geom_smooth

```
ggplot(data = training_set) + geom_smooth(mapping = aes(x = age, y = duration))
```

Creating a Simple Bar Chart

```
```{r}
ggplot(data = training_set) +
 geom_bar(mapping = aes(x = marital, y = stat(prop), group = 1))
```

## Logistic Regression Model

We're going to estimate the deposit odds using marital status and whether or not they have a housing and/or personal loan.

```
```{r} training_set$housing <- as.factor(training_set$housing) training_set$marital<- as.factor(training_set$marital)
training_set$loan <- as.factor(training_set$loan)
```

```
log_model <- glm(deposit ~ marital+housing+loan, family = "binomial", data = training_set)
```

```
summary(log_model)
```

```

### kNN Regression

```{r}
library(caret)

train_x = training_set[, -14]
train_x = scale(train_x)[,]
train_y = training_set[,14]

test_x = test_set[, -14]
test_x = scale(test[, -14])[,]
test_y = test_set[,14]

knnmodel = knnreg(train_x, train_y)
str(knnmodel)

pred_y = predict(knnmodel, data.frame(test_x))

mse = mean((test_y - pred_y)^2)
mae = MAE(test_y, pred_y)
rmse = RMSE(test_y, pred_y)

cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)

x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l", lwd=2)
lines(x, pred_y, col = "blue", lwd=2)
legend("topright", legend = c("original-medv", "predicted-medv"),
 fill = c("red", "blue"), col = 2:3, adj = c(0, 0.6))
grid()

```

## Decision Regression Tree

```

{r} install.packages('party') library(party) tree <- ctree(deposit ~ age+duration+campaign,
data=training_set, controls = ctree_control(mincriterion = 0.9, minsplit = 200)) plot(tree)

```

# Conclusion

Logistic regression is a statistical analysis approach that seeks to predict a data value based on previous observations. The purpose of logistic regression is to discover the model that best describes the connection between a dichotomous feature of interest (dependent variable = response or outcome variable) and a collection of independent (predictor or explanatory) factors. The k-nearest-neighbors algorithm is a supervised classification system that takes a set of labeled points and utilizes them to learn how to categorize other points. To label a new point, it looks at the labelled points closest to it (those are its nearest neighbors) and lets those neighbors vote on which label the majority of the neighbors have (the "k" is the number of neighbors it checks). A decision tree is a graphical depiction of certain choice scenarios that is employed in structured decision processes where complicated branching occurs. It progressively develops an associated decision tree while breaking down a data set into smaller and smaller subgroups. The end result is a tree containing leaf nodes and decision nodes. A branch node reflects a categorization or decision, whereas a decision node has two or more branches. The root node is the topmost decision node in a tree that corresponds to the best predictor. Both category and numerical data may be handled by decision trees. By comparing the models we can see that knn regression is the best approach as it results in higher accuracy.