2019

Proyecto Final – Product Development

DOCUMENTACIÓN

AXEL OMAR MEZA ARRECIS;RODOLFO ANTONIO ZEA POSADAS

Contents

Red Wine Dataset

Este dataset contiene observaciones 1599 observaciones y 12 columnas. En cada una de las observaciones se detalla varias características de diferentes clases de vino. En estas características se describen componentes que el vino contiene, específicamente el nivel de cada componente que está presente en cada observación de vino, para poder indicar a qué calidad pertenece cada una de las botellas que se listan. En el listado no hay marcas, únicamente se menciona a qué clase pertenece cada observación (botella de vino) para poder ser categorizada posteriormente con base en sus propiedades.
El dataset puede encontrarse en: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

Propósito de la data

A traves de varias características de diferentes clases de vinos se muestra la calidad que cada uno de ellos posee basado en éstas. Gracias a esta información puede determinarse la correlación que cada una de las características tiene respecto a la calidad de la botella.

Características del dataset

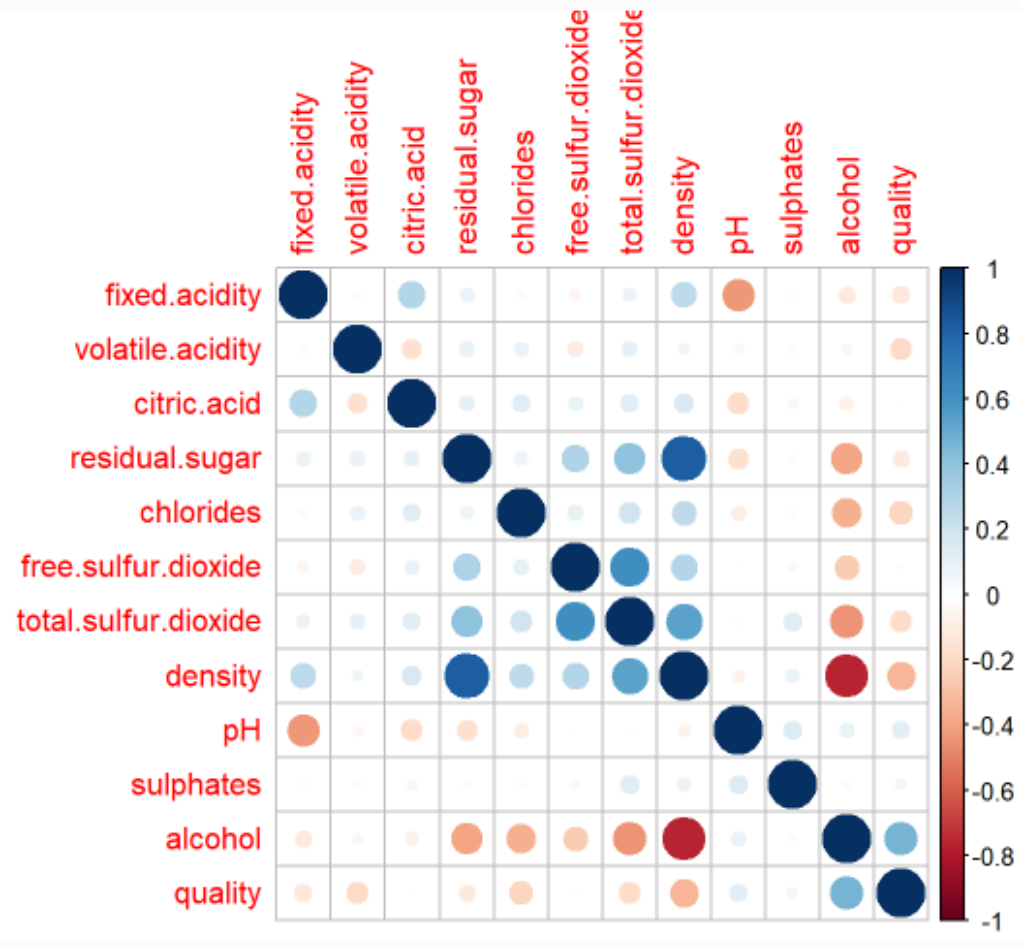
Tal cual se menciona previamente, el dataset consta de 12 columnas, la principal para nuestro caso será quality que indicará a qué calidad pertenece cada vino basado en las propiedades que cuenta y con las que fue fabricado. A continuación se da un breve detalle de las características del dataset:

No.	Característica	Descripción
1	fixed_acidity	Describe el nivel de acidez compuesto que tiene el vino
2	volatile_acidity	La cantidad de ácido acético en el vino, el cual de ser muy alto puede darle un sabor desagradable al vino, similar al vinagre.
3	citric_acid	Se encuentra en pequeñas cantidades, puede proveer un sabor de frescura al vino
4	residual_sugar	Cantidad de azúcar despues de que finaliza la fermentación, es dificil encontrar vinos con menos de 1g/L y superiores a 45g/L
5	chlorides	Cantidad presente de sal en el vino
6	free_sulfur_dioxide	Previene el crecimiento de microbios y oxidación en el vino.
7	total_sulfur_dioxide	Es la cantidad total del anterior, en pequeñas cantidades es imperceptible, pero en altas cantidades se vuelve obvio para la nariz y al sabor.
8	density	Se basa en la cantidad de agua que el vino contiene, la concentración.

9	pH	Describe qué tan ácido es el vino en la escala de 0 (muy ácido) a 14 (muy básico).
10	sulphates	Aditivo numérico del vino, SO2. Actúa como anti microbios y antioxidante.
11	alcohol	Es el porcentaje de alcohol que posee el vino.
12	quality	Para efectos de este proyecto, es la variable a clasificar, para saber a qué calidad pertenece.

Matriz de correlación

Basado en la matriz de correlación podemos observar que "quality" tiene una correlación mayor con el porcentaje de alcohol que tenga la bebida, por lo que alcohol es una característica necesaria para la elaboración de nuestros modelos, seguido por las demás en orden de correlación.



Explicación de los Modelos

Los modelos a utilizar son para clasificación de la calidad del vino basado en sus características, a continuación se detalla cada uno de ellos:

winequalitymodel.rds

```
data<-read.csv("winequality-red.csv")

dt = sort(sample(nrow(data), nrow(data)*.75))
train<-data[dt,]
test<-data[-dt,]

fit <- rpart(quality ~ fixed.acidity
                     + volatile.acidity
                     + citric.acid
                     + residual.sugar
                     + chlorides
                     + free.sulfur.dioxide
                     + total.sulfur.dioxide
                     + density
                     + pH
                     + sulphates
                     + alcohol,
             data=train,
             method="class")

fancyRpartPlot(fit)

Prediction <- predict(fit, test, type="class")
saveRDS(fit, "winequalitymodel.rds")
```

- Se carga el dataset winequality-red.csv y se separa en train y test.
- Se realiza un entrenamiento con el 75% de la data y pruebas con el 25% restante.
- Se utilizan todas features del dataset, a excepción de quality por ser la variable dependiente.
- Se realiza un entrenamiento para clasificación de la calidad del vino.
- Se guarda el modelo como *winequalitymodel.rds*

winequalitymodel_2.rds

```
data<-read.csv("winequality-red.csv")

dt = sort(sample(nrow(data), nrow(data)*.75))
train<-data[dt,]
test<-data[-dt,]

fit <- rpart(quality ~ alcohol + density
                                + total.sulfur.dioxide
                                + chlorides
                                + volatile.acidity
                                + fixed.acidity,
             data=train,
             method="class")

fancyRpartPlot(fit)

Prediction <- predict(fit, test, type="class")
saveRDS(fit, "winequalitymodel_2.rds")
```

- Se carga el dataset winequality-red.csv y se separa en train y test.
- Se realiza un entrenamiento con el 75% de la data y pruebas con el 25% restante.
- Se utilizan las features que están más correlacionadas a la calidad del vino, en este caso son 6 variables, la más correlacionada es alcohol, seguida por densidad.
- Se realiza un entrenamiento para clasificación.
- Se guarda el modelo como *winequalitymodel_2.rds*

Explicación de los Endpoints

/predict

```
# Loading model
fit <- readRDS("winequalitymodel.rds")

# Plumber Api methods

#* @apiTitle Predict Wine quality
#* @apiDescription wine quality prediction.

#' @param fixed_acidity:numeric most acids involved with wine or fixed or nonvolatile (do not
#' @param volatile_acidity:numeric the amount of acetic acid in wine, which at too high of le
#' @param citric_acid:numeric found in small quantities, citric acid can add 'freshness' and
#' @param residual_sugar:numeric the amount of sugar remaining after fermentation stops, it's
#' @param chlorides:numeric the amount of salt in the wine
#' @param free_sulfur_dioxide:numeric the free form of SO2 exists in equilibrium between mole
#' @param total_sulfur_dioxide:numeric amount of free and bound forms of S02; in low concentra
#' @param density:numeric the density of water is close to that of water depending on the per
#' @param pH:numeric describes how acidic or basic a wine is on a scale from 0 (very acidic)
#' @param sulphates:numeric a wine additive which can contribute to sulfur dioxide gas (S02)
#' @param alcohol:numeric the percent alcohol content of the wine
#' @post /predict
#' @response 200 Returns the quality variable (based on sensory data, score between 0 and 10)
```

- Se carga el modelo winequalitymodel.rds
- Especificamos cada uno de los parámetros que éste tendrá, para este caso se utiliza todas las características del dataset, a excepción de la variable a predecir –quality-
- Indicamos que el Endpoint será de tipo /POST y nombre /predict
- Si su ejecución es exitosa retornará un valor 200

```
function(req, res, fixed_acidity = NA, volatile_acidity = NA, citric_acid = NA, residual_sugar = NA, chloride
         free_sulfur_dioxide = NA, total_sulfur_dioxide = NA, density = NA, pH = NA, sulphates = NA, alcohol

  # Creating features data frame
  features <- data_frame(fixed.acidity = as.numeric(fixed_acidity),
                         volatile.acidity = as.numeric(volatile_acidity),
                         citric.acid = as.numeric(citric_acid),
                         residual.sugar = as.numeric(residual_sugar),
                         chlorides = as.numeric(chlorides),
                         free.sulfur.dioxide= as.numeric(free_sulfur_dioxide),
                         total.sulfur.dioxide = as.numeric(total_sulfur_dioxide),
                         density = as.numeric(density),
                         pH = as.numeric(pH),
                         sulphates = as.numeric(sulphates),
                         alcohol = as.numeric(alcohol))

  # Validations
  if (any(is.na(features))) {
    res$status <- 400
    res$body <- "Parameters have to be numeric or integers"
  }

  # Executing prediction
  out <- predict(fit, features, type="class")

  as.character(out)
`
```

- Se utiliza una función que recibe como parámetros la request, response y todas las features exceptuando quality, todas las features de parámetro se colocan con valor NA default.
- Todas las features son agregadas a un dataframe, el cual seguidamente se valida para determinar si hay alguna feature que no haya recibido valor o que haya recibido un valor con tipo de dato incorrecto, de ser así se retorna un mensaje de error.
- Finalmente se realiza la predicción de las features recibidas utilizando el modelo previamente cargado en la variable fit.

/batchpredict

```
#* @apiTitle Load test data
#* @apiDescription Load test data to model.

#' @param username User who makes the request.
#' @param model Model to use for testing input.
#' @param batchdata Data to be predicted
#' @post /batchpredict
#' @response 200 Returns performance metrics
```

- Especificamos cada uno de los parámetros que éste tendrá, para este caso se tiene un parámetro que recibirá la data en batch.
- Indicamos que el Endpoint será de tipo /POST y nombre /batchpredict
- Si su ejecución es exitosa retornará un valor 200

```
function(req, res, username = NA, model = NA, batchdata = NA) {

  errors <- FALSE

  if(any(is.na(batchdata)))
  {
    res$status <- 400
    res$body <- "All batchdata parameters have to be numeric or integers"
    errors <- TRUE
  }

  if (is.na(model))
  {
    res$status <- 400
    res$body <- "Parameter model is required"
    errors <- TRUE
  }

  if (!is.integer(model)) {
    res$status <- 400
    res$body <- "Parameter model must be an integer value"
    errors <- TRUE
  }

  colnames(batchdata)[which(names(batchdata) == "fixed_acidity")] <- "fixed.acidity"
  colnames(batchdata)[which(names(batchdata) == "volatile_acidity")] <- "volatile.acidity"
  colnames(batchdata)[which(names(batchdata) == "citric_acid")] <- "citric.acid"
  colnames(batchdata)[which(names(batchdata) == "residual_sugar")] <- "residual.sugar"
  colnames(batchdata)[which(names(batchdata) == "free_sulfur_dioxide")] <- "free.sulfur.dioxide"
```

- Se utiliza una función que recibe como parámetros la request, response, username, modelo y en lugar de todas las features, un solo parámetro con el batch de información.
- Se agregan los valores del batch a cada columna del dataframe.
- Se validan errores y se configura una respuesta para cualquier error de input.

```
if (!errors) {
  # Executing prediction
  if (model == 1)
  {
    # model 1
    out <- predict(fit, batchdata, type="class")
    res$body <- out

  } else if (model == 2) {
    # model 2
    out <- predict(fit2, batchdata, type="class")
    res$body <- out
  } else {
    res$status <- 400
    res$body <- "Model does not exist"
  }
```

- Finalmente se realiza la predicción del batch recibido utilizando el modelo previamente cargado en la variable fit.

/modelperformance

```
#* @apiTitle Model performance
#* @apiDescription Test model performance and return performance metrics

#' @param username User who makes the request.
#' @param model Model to use for testing input.
#' @param testdata Data to be tested for evaluating model performance
#' @post /modelperformance
#' @response 200 Returns performance metrics
```

- Se crean los parámetros necesarios, al igual que el modelo anterior, se tiene un parámetro que recibirá todos los datos de prueba, en este caso es testdata.
- Si todo se ejecuta correctamente se retornará un valor código 200.

```
if (!errors) {
  predictions = as.numeric(as.character(predictions))

  # confusion matrix
  conf.matrix <- table(testdata$quality, predictions)
  rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = "_")
  colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = "_")

  rocobj <- multiclass.roc(testdata$quality, predictions)
  auc <- rocobj$auc
  accuracy <- length(which(predictions == testdata$quality)) / length(testdata$quality)

  # Final metrics
  metrics <- list(confusion_matrix = as.data.frame(conf.matrix), auc = auc, accuracy = accuracy)
  res$body <- as.character(metrics)
}
```

- Se realizan diferentes validaciones para los parámetros y en caso haya algún error se retorna un código de error y la descripción del mismo.
- Si no hubo errores durante la ejecución del endpoint entonces se calculan las diferentes métricas y se retornan como respuesta.