NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION AFFILIATED TO VISHVESHWARYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

Department of Computer Science and Engineering

Academic Year: 2017-2018

Data Analytics using R programming Report on

'WINE QUALITY PREDICTION'

Submitted by

ANKIT DATTA	**1NT15CS028**
HARSHITH NARAHARI	**1NT15CS064**
ROSHAN BADRINATH	**1NT15CS140**

Under the able guidance of

Dr. Sarojadevi N

Professor, Dept. of CSE

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

YELAHANKA, BANGALORE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the course project in Data Analytics in R in VI semester report entitled

Wine Quality Prediction

is an authentic record of the project carried out by

TEAM MEMBERS:

Ankit Datta (1NT15CS028)
Harshith Narahari (1NT15CS064)
Roshan Badrinath (1NT15CS140)

SIGNATURE OF GUIDE	**SIGNATURE OF HOD**
……	……...
DR. SAROJADEVI N	**DR. THIPPESWAMY M N**
PROFESSOR	**HEAD OF DEPARTMENT**
CSE DEPT, NMIT	**CSE DEPT, NMIT**

ACKNOWLEDGEMENT

We are extremely grateful to our HOD, **Dr. Thippeswamy** who extended his support towards our project. We remain indebted to our teacher **Dr. Sarojadevi N** for her constant support in the Design, Implementation and Evaluation of the project. We are thankful to her for constructive criticism and valuable suggestions, which benefited us a lot while developing the project, also without whom we might not have been able to accomplish this project. Finally, we gratefully acknowledge the support, encouragement and patience of our friends.

ABSTRACT

Human wine tasting is a sensory examination and evaluation of wine. There are many properties that decide the quality of wine such as color, swirl, smell and savor. There are also various physiochemical properties that decide the quality.

We propose a machine learning approach to predict human wine tasting preferences. Our project focuses on some of the physiochemical properties that will be used to predict the quality of wine. A large data set, from Portugal, with white *Vinho Verde* wine data sample is considered for training and testing. We have used a Fully Connected Multi Perceptron Neural Network to classify the wine data.

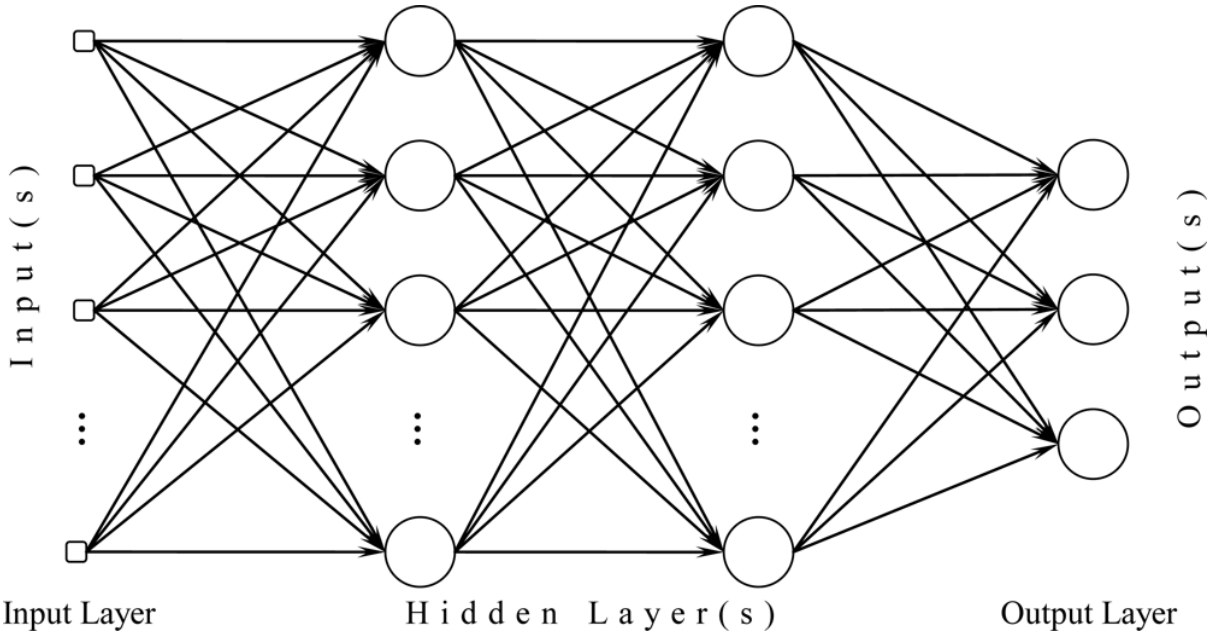
Our model helps in supporting the oenologist in wine tasting evaluation and production of wine. Similar techniques can help in target marketing by modeling consumer tastes from niche markets.

CONTENTS OF THE PROJECT

INTRODUCTION

R programming language is used for implementation of our project. R provides wide variety of tools for statistical and data analytics. *R Studio*, an open source IDE for R programming, provides powerful coding tools and an interacting graphic environment.

Tensorflow and *Keras* API, written in Python, are used for the implementation of the Neural Network. A package called *reticulate* is used which acts as an interface to Python modules. A *Fully Connected Neural Network* model is created using the Keras API.



The above figure shows a Fully Connected Neural Network. Terminologies used in a Neural Network are:

Neuron: A neuron forms the basic structure of a neural network. It receives an input, processes it, and generates an output. It is sometimes called as a node.

Layer: A neural network consists of an input layer, set of hidden layers, and an output layer. Each layer consists of set of nodes. The number of attributes in the data set used for prediction, i.e. set of explanatory variables, decide the number of nodes in the input layer. Number of nodes in the output layer depends on the values of the predictor variable.

Weight: Synaptic weight refers to the strength or amplitude of a connection between two nodes. It is considered as a linear component.

Bias: Another linear component is applied to the input, called as the bias. Bias is added to the product of the weights of the input

Activation Function: It is a non-linear component applied to the linear combination of the input. It converts the input signal to an output signal based on the function. Various activation function we have used are:

- **ReLU**: It stands for *Rectified Linear Unit*. It is usually applied to the *hidden layers*. The function is defined as:

  $$f(x) = \max(x,0)$$

- **Softmax**: It is generally applied to the *output layer* for classification problem.

Optimizers: Optimization algorithms help in minimizing the loss in the training process. One of the optimizer is *Adam*, which stands for *Adaptive Moment Estimation.*

Dropout: Dropout is a technique used to prevent over-fitting of the network. In this technique, certain weights are randomly frozen in the hidden layers during the training process. In other words, certain neurons are dropped. Hence, it allows the neural network to find new paths.

Epochs: Single iteration of all the batches of the training process is called as epochs. The number of epochs to train can be chosen. As the number of epochs increases, the accuracy increases, provided over-fitting does not occur.

DESCRIPTION OF PACKAGES

Different packages that have been used are:

Tensorflow: Tensorflow is an open-source software library for numerical communication and data flow programming using data flow graphs. It is used for machine learning applications such as neural networks. It was developed by Google Brain team for Internal use and was later launched under Apache 2.0 open source license for common use.

Reticulate: Interface to Python modules, classes and functions are provided by the Reticulate function. Reticulate acts as a wrapper package for R as Tensorflow and Keras are written in Python.

Keras: Keras is a high level neural network API. It enables fast experimentation of both convolutional and recurrent neural network. It's written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It runs seamlessly on CPU as well as and CUDA supported Nvidia graphic cards.

caTools: caTools provides basic utility functions like moving window statistic function, read and write for GIF,ENVI binary files, etc and functionality for splitting data set into training and testing set. It also provides functionality for fast calculation of AUC, LogitBoost classifier, base64 encoder/decoder, round-off-error-free sum and cumulative sum, etc.

ggplot2: ggplot2 is a data visualization package for R. It was created by Hadley Wickham in 2005. In contrast to R base graphics, ggplot2 provides functionality to add, remove and alter components in a plot with a high level of abstraction. It contains a number of defaults for web and print display of common scales and can serve as a replacement for base graphics in R

corrplot: corrplot provides graphical display of correlation matrix. Calculation of correlation between attributes of the data set can be done with the help of corrplot and the same can be used for plotting correlation plots.

DESCRIPTION OF THE DATA SET

A large data set, from Portugal, with white *Vinho Verde* wine data sample is considered for training and testing. The data set consists of 12 attributes, out of which 11 are considered as explanatory variables. The 12^{th} attribute, quality, is considered as the response variable. Various attributes available in the data set are:

- *fixed.acidity* (Tartaric acid – g/dm^3): Most acids involved with wine or fixed or nonvolatile (do not evaporate readily).
- *volatile.acidity* (Acetic acid – g/dm^3): The amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste.
- *citric.acid* (g/dm^3): Found in small quantities, citric acid can add 'freshness' and flavor to wines.
- *residual.sugar* (g/dm^3): The amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet.
- *chlorides* (Sodium Chloride – g/dm^3): The amount of salt in wine.
- *free.sulfur.dioxide* (mg/dm^3): The free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine.
- *total.sulphur.dioxide* (mg/dm^3): Amount of free and bound forms of S02; in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine.
- *density* (g/cm^3): The density of water is close to that of water depending on the percent alcohol and sugar content.
- *pH*: Describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale.
- *sulphates* (Potassium Sulphate – g/dm^3): A wine additive which can contribute to sulfur dioxide gas (S02) levels, wich acts as an antimicrobial and antioxidant.
- *alcohol* (% by Volume): The percent alcohol content of wine.
- *quality* : It is the response variable, scores between 0 to 10.

CODE DESCRIPTION AND OUTPUT

Packages to include:

```
setwd("/home/5atego/workspace/R Studio/R-Pr/")
library(tensorflow)
use_virtualenv("/home/5atego/.virtualenvs/r-tensorflow")
library(reticulate)
wn <- import("warnings")
wn$filterwarnings("ignore")
library(keras)
library(caTools)
library(ggplot2)
library(corrplot)
```

Import Data Set:

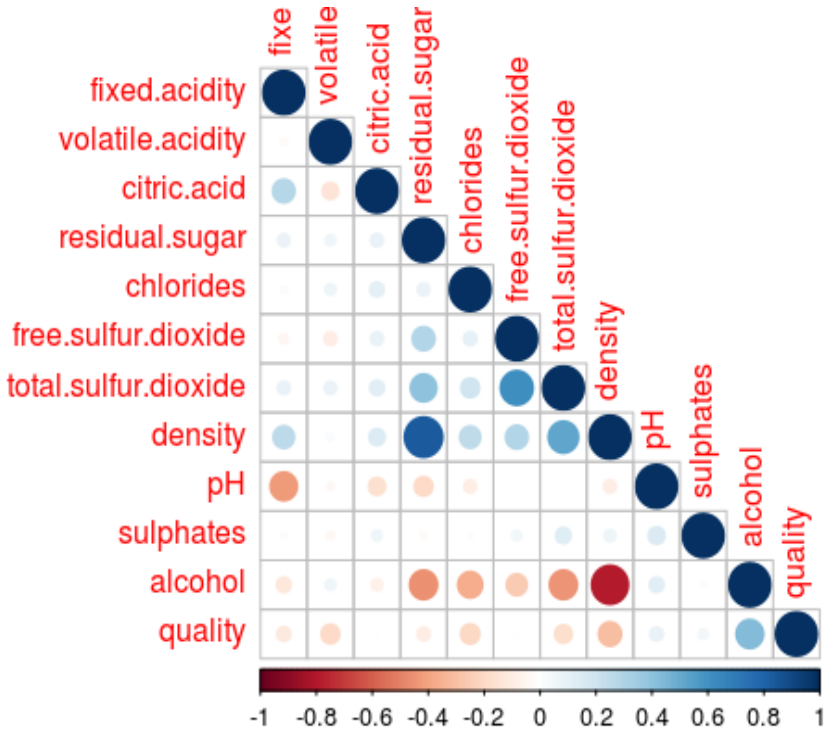
Data set to be imported is stored in *wineQualityWhites.csv*. It is stored in a variable called *data*.

```
data <- read.csv(file = "wineQualityWhites.csv", sep = ",", header = TRUE)
```

Feature Selection using Correlation Plots:

There is no correlation between *quality* and *citric.acid* and *quality* and *free.sulphur.dioxide*. Therefore citric.acid and free.sulfur.dioxide are removed during training.

```
cor <- cor(data[c(-1)])
corrplot(cor, type = "lower")
```



Find Quality Levels:

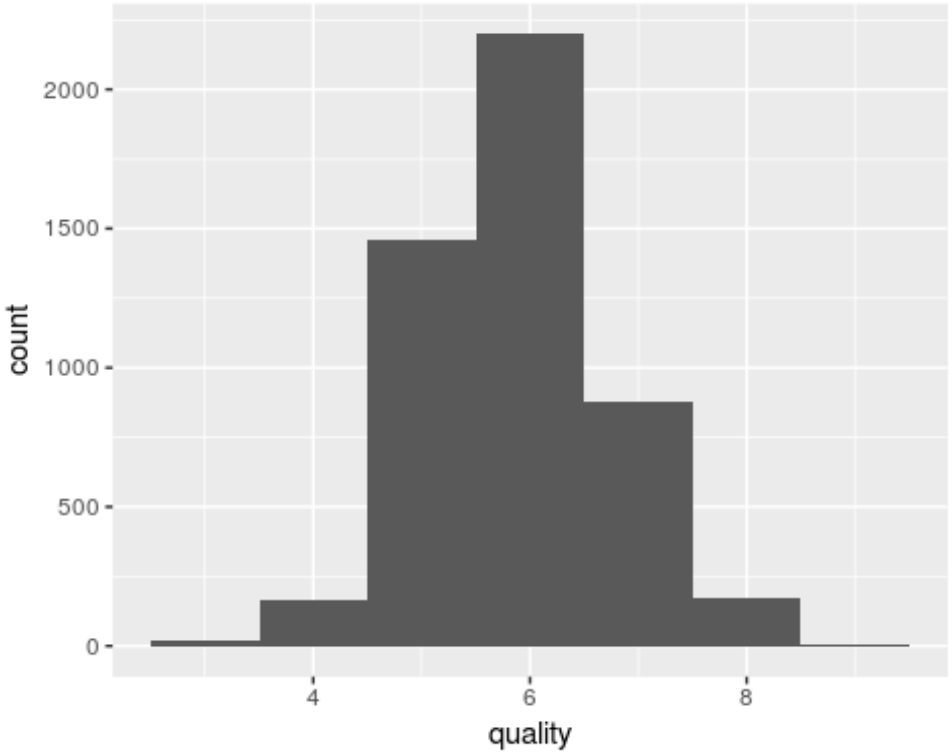
The response variable, i.e. *quality* attribute, ranges from 0 to 10, out of which values in the data set ranges from 3 to 9.

```
data$quality.factor <- as.factor(data$quality)

## [1] "3" "4" "5" "6" "7" "8" "9"
```

Histogram of Quality:

```
ggplot(data = data, aes(x = quality)) +
  geom_histogram(binwidth = 1)
```



Add Binary Ordered Quality Attribute:

```
for(i in 1:nrow(data)){
  if(data$quality[[i]] < 6){
    data$quality.order[[i]] <- "Bad"
  }else{
    data$quality.order[[i]] <- "Good"
  }
}
```

Count of Good and Bad quality wine data:

```
i <- j <- 0
for(q in data$quality.order){
  if(q == "Bad"){
    i <- i+1
  }else{
    j <- j+1
  }
}
print(i)

## [1] 1640

print(j)

## [1] 3258

data$quality.order <- factor(factor(data$quality.order), levels = c("Bad",
"Good"))
```

Convert Quality To Binary Attribute:

The *quality.order* consists of only two values, namely *Good* and *Bad*. The machine learning model does not understand character attributes. Hence, it is converted to binary values.

```
for(i in 1:nrow(data)){
  if(data$quality.order[[i]] == "Bad"){
    data$quality.num[[i]] <- 0
  }else{
    data$quality.num[[i]] <- 1
  }
}
```

Train and Testing Data Sets:

The data set is divided into training and testing sets. The package *caTools* provides the functionality to split the data set. Training sample consists of 70% of the data and testing sample consists of the remaining 30%. The data set is then written into two csv files.

```
value <- sample.split(data$X, SplitRatio = 0.7)
train.data <- subset(data, value == TRUE)
test.data <- subset(data, value == FALSE)

write.csv(train.data, file = "train_data.csv")
write.csv(test.data, file = "test_data.csv")
```

Loading existing training and testing data sets:

The training and testing data sets are loaded. All the explanatory variables are stored in *x_train* and *x_test* variables, in the form of a matrix. The response variables are stored in y_train and y_test. The response variables are given to a one-hot encoded function (to_catagorical). One-hot encoding is a process of converting categorical attribute to a form understood by ML algorithm.

```
train.data <- read.csv(file = "train_data.csv", sep = ",", header = TRUE)
test.data <- read.csv(file = "test_data.csv", sep = ",", header = TRUE)

train.data <- train.data[,-1]
test.data <- test.data[,-1]

xtrain <- as.matrix(train.data[,c(2,3,5,6,8:12)])
ytrain <- train.data[,16]
xtest  <- as.matrix(test.data[,c(2,3,5,6,8:12)])
ytest  <- test.data[,16]

ytrain <- to_categorical(y_train,2)
ytest <- to_categorical(y_test,2)
```

Creating Model:

A fully-connected feed-forward neural network is created. The model has 5 layers (3 hidden layers). It contains 2 dropout layers. For the input layer and the hidden dense layers, *ReLU* activation function is used. *Softmax* activation function is applied to the output layer.

```
model1 <- NULL
model1 <- keras_model_sequential()
model1 %>%
  layer_dense(units = 256, input_shape = c(9), activation = "relu") %>%
  layer_dropout(0.2) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(0.3) %>%
  layer_dense(units = 2,  activation = "softmax")
```

Compile the Model:

The model is compiled for the training process. *Adam* optimizer is used to reduce the loss.

```
model1 %>% compile(
  loss = "binary_crossentropy",
  optimizer = optimizer_adam(0.001),
  metrics = c('accuracy')
)
```

Train the Model:

The training data is further split into validation sample which is used for increasing the efficiency of the training process. The model is saved for future use.

```
model1 %>% fit(
  xtrain, ytrain,
  validation_split = 0.3,
  epochs = 100,
  batch_size = 87
)

save_model_hdf5(model1, "model_name.h5")
```

Load the data model:

```
model1 <- load_model_hdf5("model8_tracc_0.7850.h5")
```

Evaluate the Quality of Wine for testing data set:

The accuracy of the model is found to be 78.5% for the test sample.

```
model1 %>% evaluate(xtest,ytest)

## $loss
## [1] 0.5228955
##
## $acc
## [1] 0.785034
```

Predict Values and plot Confusion Matrix:

```
classes <- model1 %>% predict_classes(x_test)
table(Actual = test.data[,16], Predicted = classes)

##       Predicted
## Actual   0   1
##      0 299 193
##      1 123 855
```

BIBLIOGRAPHY

Data set is obtained from:

https://docs.google.com/document/d/1qEcwltBMlRYZT-l699-71TzInWfk4W9q5rTCSvDVMpc/pub

Links referred:

https://keras.rstudio.com/

https://tensorflow.rstudio.com/

https://www.r-bloggers.com/

https://cran.r-project.org/web/packages/

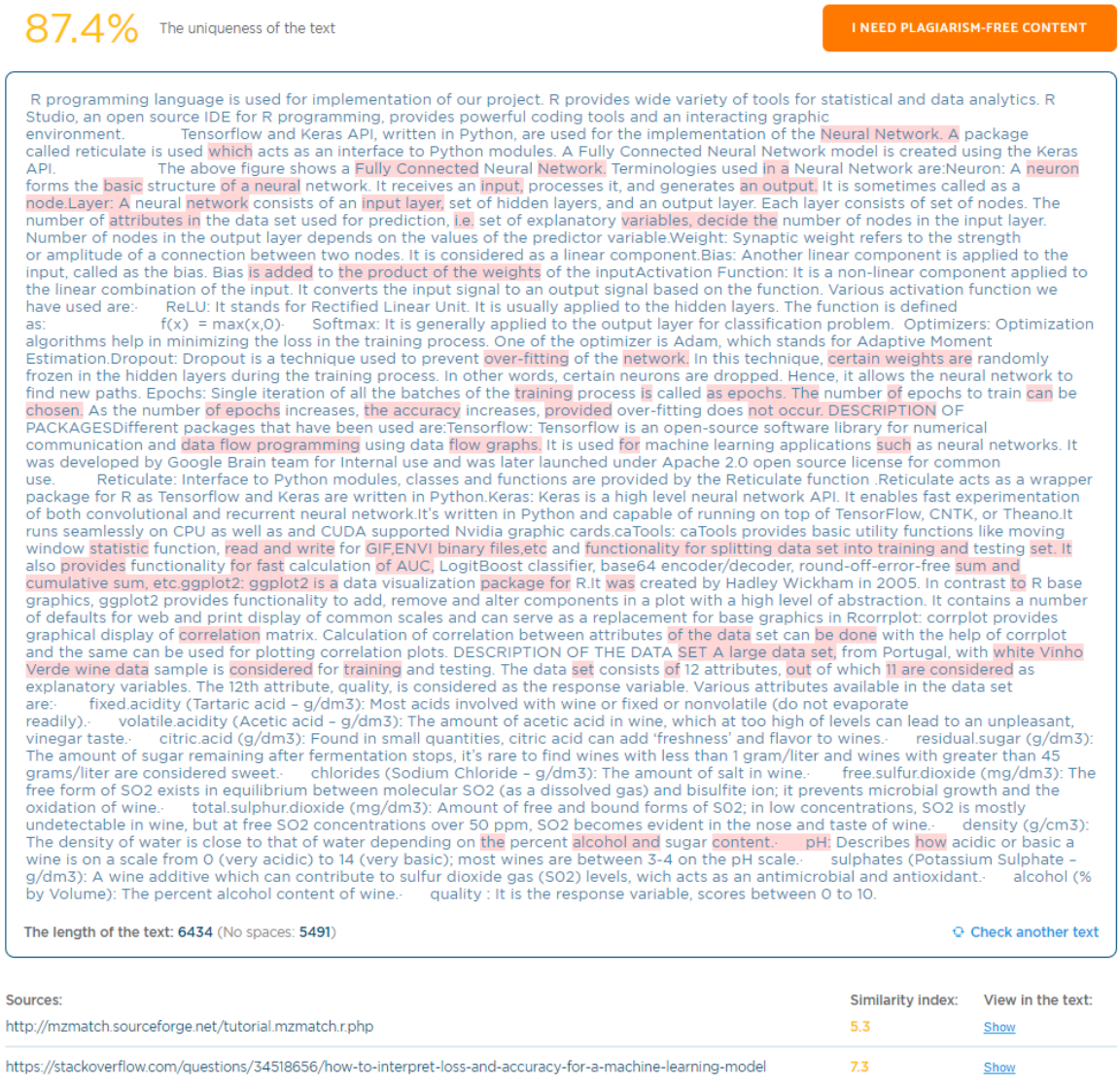
PLAGIARISM CHECK

Website: https://papersowl.com/free-plagiarism-checker

Plagiarism check for:

- Introduction
- Description of Packages
- Description of Data Set

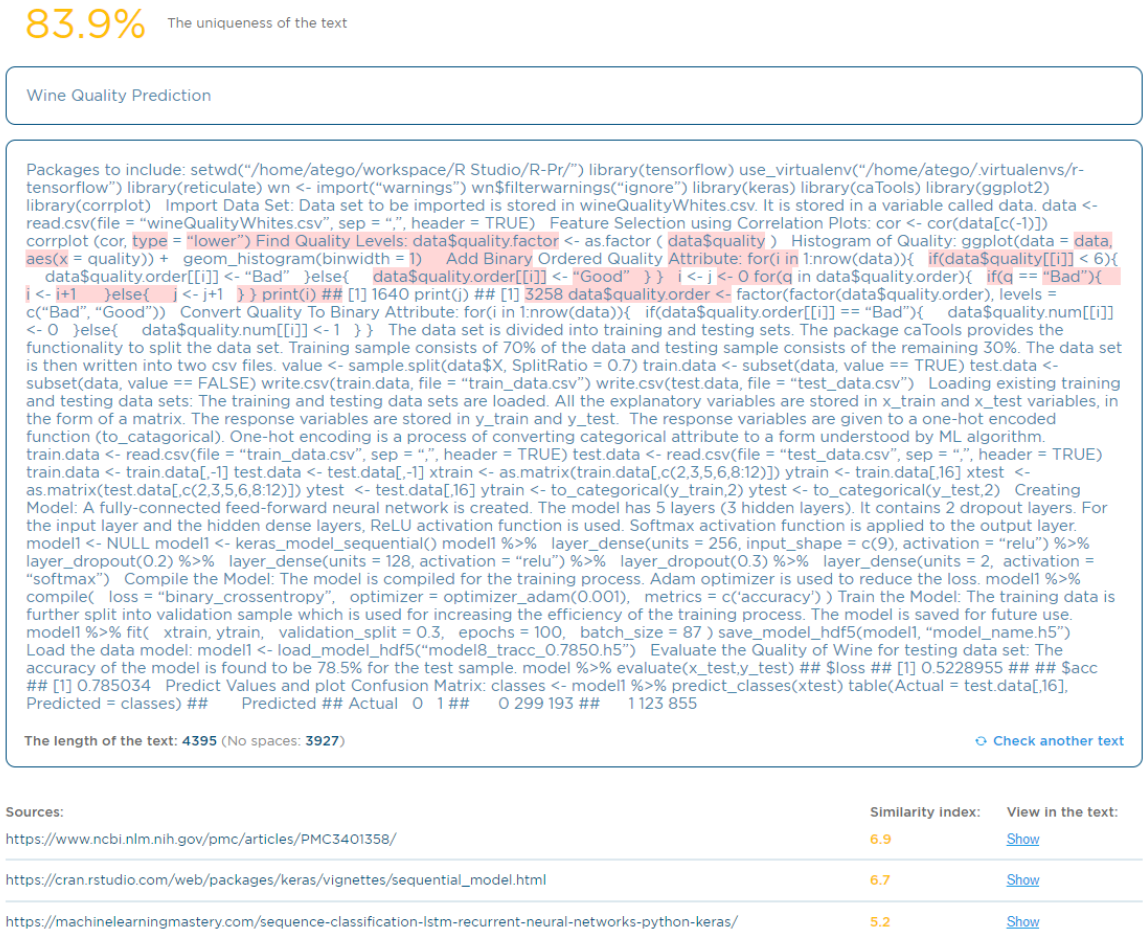
87.4% Unique



Plagiarism check for:

- Code Description and Output

83.9% Unique



83.9% The uniqueness of the text

Wine Quality Prediction

Packages to include: setwd("/home/atego/workspace/R Studio/R-Pr/") library(tensorflow) use_virtualenv("/home/atego/.virtualenvs/r-tensorflow") library(reticulate) wn <- import("warnings") wn$filterwarnings("ignore") library(keras) library(caTools) library(ggplot2) library(corrplot) Import Data Set: Data set to be imported is stored in wineQualityWhites.csv. It is stored in a variable called data. data <- read.csv(file = "wineQualityWhites.csv", sep = ",", header = TRUE) Feature Selection using Correlation Plots: cor <- cor(data[c(-1)]) corrplot (cor, type = "lower") Find Quality Levels: data$quality.factor <- as.factor (data$quality) Histogram of Quality: ggplot(data = data, aes(x = quality)) + geom_histogram(binwidth = 1) Add Binary Ordered Quality Attribute: for(i in 1:nrow(data)){ if(data$quality[[i]] < 6){ data$quality.order[[i]] <- "Bad" }else{ data$quality.order[[i]] <- "Good" } } i <- j <- 0 for(q in data$quality.order){ if(q == "Bad"){ i <- i+1 }else{ j <- j+1 } } print(i) ## [1] 1640 print(j) ## [1] 3258 data$quality.order <- factor(factor(data$quality.order), levels = c("Bad", "Good")) Convert Quality To Binary Attribute: for(i in 1:nrow(data)){ if(data$quality.order[[i]] == "Bad"){ data$quality.num[[i]] <- 0 }else{ data$quality.num[[i]] <- 1 } } The data set is divided into training and testing sets. The package caTools provides the functionality to split the data set. Training sample consists of 70% of the data and testing sample consists of the remaining 30%. The data set is then written into two csv files. value <- sample.split(data$X, SplitRatio = 0.7) train.data <- subset(data, value == TRUE) test.data <- subset(data, value == FALSE) write.csv(train.data, file = "train_data.csv") write.csv(test.data, file = "test_data.csv") Loading existing training and testing data sets: The training and testing data sets are loaded. All the explanatory variables are stored in x_train and x_test variables, in the form of a matrix. The response variables are stored in y_train and y_test. The response variables are given to a one-hot encoded function (to_catagorical). One-hot encoding is a process of converting categorical attribute to a form understood by ML algorithm. train.data <- read.csv(file = "train_data.csv", sep = ",", header = TRUE) test.data <- read.csv(file = "test_data.csv", sep = ",", header = TRUE) train.data <- train.data[,-1] test.data <- test.data[,-1] xtrain <- as.matrix(train.data[,c(2,3,5,6,8:12)]) ytrain <- train.data[,16] xtest <- as.matrix(test.data[,c(2,3,5,6,8:12)]) ytest <- test.data[,16] ytrain <- to_categorical(y_train,2) ytest <- to_categorical(y_test,2) Creating Model: A fully-connected feed-forward neural network is created. The model has 5 layers (3 hidden layers). It contains 2 dropout layers. For the input layer and the hidden dense layers, ReLU activation function is used. Softmax activation function is applied to the output layer. model1 <- NULL model1 <- keras_model_sequential() model1 %>% layer_dense(units = 256, input_shape = c(9), activation = "relu") %>% layer_dropout(0.2) %>% layer_dense(units = 128, activation = "relu") %>% layer_dropout(0.3) %>% layer_dense(units = 2, activation = "softmax") Compile the Model: The model is compiled for the training process. Adam optimizer is used to reduce the loss. model1 %>% compile(loss = "binary_crossentropy", optimizer = optimizer_adam(0.001), metrics = c('accuracy')) Train the Model: The training data is further split into validation sample which is used for increasing the efficiency of the training process. The model is saved for future use. model1 %>% fit(xtrain, ytrain, validation_split = 0.3, epochs = 100, batch_size = 87) save_model_hdf5(model1, "model_name.h5") Load the data model: model1 <- load_model_hdf5("model8_tracc_0.7850.h5") Evaluate the Quality of Wine for testing data set: The accuracy of the model is found to be 78.5% for the test sample. model %>% evaluate(x_test,y_test) ## $loss ## [1] 0.5228955 ## ## $acc ## [1] 0.785034 Predict Values and plot Confusion Matrix: classes <- model1 %>% predict_classes(xtest) table(Actual = test.data[,16], Predicted = classes) ## Predicted ## Actual 0 1 ## 0 299 193 ## 1 123 855

The length of the text: 4395 (No spaces: 3927) Check another text

Sources:	Similarity index:	View in the text:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3401358/	6.9	Show
https://cran.rstudio.com/web/packages/keras/vignettes/sequential_model.html	6.7	Show
https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/	5.2	Show