Classifier for Hand-labeled Data

Kunhao Ni

2024-04-09

Intro:

The purpose of this project is to explore the way to implement a classifier for categorical data with missing value (being specific, we are classifying the 'Directors102b7' column).The data set is retrieved from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/3S1XRN.

Problem Summary:

After conducting some research, and analysis to the original data set, there are some facts of this data set need to be considered carefully:

1. The majority of the data are nominal and categorical.
2. There are a lot of missing value in the data set.
3. For some specific col(Charter_ID, Date_Coded, Date_Filing), unique values are large.
4. For the most of the column, the unique values are limited(Y/N or Null)
5. The column we want to classify is a binary col with response of Yes or No.

Basing facts above, we have to divide this problem into 3 parts: processing data, encoding categorical data, and fitting a model for the classifier.

Data Processing:

In order to process the data for the model training in next step, we need to deal with the missing data at first. Generally speaking, we want to avoid using data with more than 10% of data missing. Unfortunately, there are only 4 columns of data has less than 10% of data missing. Thus, in this study we use 20% as the threshold. The following code loaded the data, dropped columns with more than 20% of data loss, and split-ted the data into training set and prediction sets. The training set is consist of rows that does not have empty value in the 'Directors102b7' column. The prediction set is consist of rows that has missing value in 'Directors102b7' column.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# Load Data
data <- read.csv("LabeledDataCCG.csv")

# Drop columns that has missing more than 20% values.
missing_data_percentages <- sapply(data, function(x) sum(is.na(x)) / length(x)) * 100
columns_to_drop <- names(missing_data_percentages[missing_data_percentages > 20])
data_cleaned <- data %>% select(-one_of(columns_to_drop))

# Split the data into training and prediction
training_data <- data_cleaned[data_cleaned$Directors102b7 == 'Y' | data_cleaned$Directors102b7 == 'N', ]
prediction_data <- data_cleaned[!(data_cleaned$Directors102b7 == 'Y' | data_cleaned$Directors102b7 == '
```

Encoding Categorical Data Considering the fact that the data set is consist of categorical data, we need to encode the data into numerical data for the model training. My initial attempt was to label each unseen variable with a specific integer. Then I realize that there are limitation of this method: machine learning models may misinterpreted that there is some sort of hierarchy in them. Thus, I used one-hot encoding to avoid the problem of originality.

```r
# One hot encoding to transfer the categorical data into numerical data for the next step
data_encoded <- data_cleaned %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.factor, function(x) as.integer(as.factor(x)))

training_data_encoded <- training_data %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.factor, function(x) as.integer(as.factor(x)))

prediction_data_encoded <- prediction_data %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.factor, function(x) as.integer(as.factor(x)))
```

Model Fitting There are many potential methods of classification problems:Logistic Regression, K-NN, Decision Trees, Random Forest and etc. After a brief research, I decided to use Random Forest as the method to fit the model for these reasons: 1. These methods are suitable for this data set size and can handle non-linear relationships. Random forests, in particular, can provide good accuracy and handle over-fitting better than individual decision trees. 2. Comparing methods like Logistic Regression, Random Forest performs better as the amount of data columns getting large(Robustness to Noise). 3. Random Forest approach is non-parametric, which means it does not assume anything about the distribution of the data at the root level or the correlation between the target variable and characteristics. 4. Random Forest approach has a relative higher tolerance to the categorical data.

The code implementation shown as follow:

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(123)

training_data_encoded$Directors102b7 <- as.factor(training_data_encoded$Directors102b7)

# Train the Random Forest model
rf_model <- randomForest(Directors102b7 ~ ., data=training_data_encoded, ntree=100)
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Directors102b7 ~ ., data = training_data_encoded,      ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 3.26%
## Confusion matrix:
##     1    2 class.error
## 1 760  130 0.146067416
## 2  36 4164 0.008571429
```

After the model fitting with training data, I use the RF model as the classifier to predict the 'Directors102b7' column value of rows with data loss.

```
# Predicting with the Random Forest model
predictions <- predict(rf_model, newdata=prediction_data_encoded)
prediction_data_encoded$Directors102b7 <- predictions
write.csv(prediction_data_encoded, "prediction.csv",row.names = FALSE, quote=FALSE)
```

Examination: To check for the accuracy of the model, I conduct a cross-validation. The result shown that the performance of the rf_model has a good performance.

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# Conduct a cross-validation to check for accuracy of the model

X <- training_data_encoded[, -which(names(training_data_encoded) == "Directors102b7")]
y <- training_data_encoded$Directors102b7

ctrl <- trainControl(method = "cv", number = 5)

model <- train(x = X, y = y, method = "rf", trControl = ctrl)

print(model)
```

```
## Random Forest
##
## 5090 samples
##   39 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4072, 4072, 4072, 4072, 4072
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9239686  0.6879559
##   20    0.9693517  0.8895613
##   39    0.9677800  0.8841072
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 20.
```

Summary:

In this project, I used Open AI as a search engine to find related essay and document as resources. Also I used chain-of-thought prompting strategies to implement the code in a more efficient ways. While this project deployed the Open AI as a source, I wrote the majority part of code and documents. Moreover, I do have knowledge about all statistical methods I mentioned above.

Speak about the project itself and the result, I feel that it is a interesting challenge to apply machine learning method on a poorly labeled data set. I made several attempts before coming up with this method(include K-NN method and Missing Forest imputation), where this method I'm writing about has the optimal performance.

Reference:

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/3S1XRN

https://chat.openai.com/share/2ea7fe1a-3b69-4bf4-951d-fc3aaa7f9e54

https://chat.openai.com/share/c2079a02-3452-4d78-b3e4-843e3fd69cd0

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00516-9#Sec32

https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/

https://www.geeksforgeeks.org/ml-one-hot-encoding/