Wine Quality Data Set

Shashi Roshan (NetID : sroshan2)

Abstract

Statistical learning models were applied to wine quality data set in order to predict the wine quality. A variety of learning techniques were explored and validated, and ultimately the predictive power of the models appears to be good.

Introduction

We use wine quality data to predict the quality of wine, using physicochemical inputs. This kind of analysis can be useful in predicting the quality of newly made wine, before it is tested with the professional wine tasters. Only physicochemical and sensory data is available. The data includes measures like fixed acidity, volatile acidity, residual sugar, chlorides, etc. The response variable is the quality variable (score between 0 and 10). We perform regression to predict the quality (on a scale of 0-10). We are also interested in the classification problem : predicting if the wine is of high quality or not (high quality means a quality score above 6).

Methods

Data

The data has been collected from UCI Machine learning repository. The data contains physicochemical measures. There is no data about grape types, wine brand, wine selling price, etc. Following are the columns in the data :
Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)

In addition, one more column has been created - high_or_low, which indicates if the wine is of high quality or not (we assume that high quality wine has score above 6).

```r
# load packages
library("tibble")
library("dplyr")
library("randomForest")
library("caret")
library("glmnet")
library('kableExtra')

# install devtools and ucidata if needed
# install.packages("devtools")
# devtools::install_github("coatless/ucidata")

# create new response variables
wine = as_tibble(ucidata::wine)
wine = wine %>%
  mutate(quality_num = quality,
         quality_cat = as.factor(quality)) %>%
  select(-quality)

wine = wine %>%  mutate(high_or_low = case_when(quality_num > 6  ~ "high" ,
                                                quality_num <= 6  ~ "low"))
wine$high_or_low = as.factor(wine$high_or_low)

# split data
set.seed(2)
idx = sample(nrow(wine), size = 0.8 * nrow(wine))
wine_trn = wine[idx, ]
wine_tst = wine[-idx, ]
```

Modelling

Regression Problem

In order predict the quality of wine on a scale of 0-10, four different statistical models were used :
- Random forest model was trained using all available predictors. The choice of mtry was chosen using cross validation.
- Decision tree model was trained using all available predictors. The choice of the complexity parameter was chosen using cross validation.
- Lasso regression model was trained using all available predictors. The choice of optimal value of lambda was chosen using cross validation.
- Ridge regression model was trained using all available predictors. The choice of optimal value of lambda was chosen using cross validation.

Classification Problem

In order predict if the wine was of high or low quality, three different statistical models were used :
- Random forest model was trained using all available predictors. The choice of mtry was chosen using cross validation.
- Decision tree model was trained using all available predictors. The choice of the complexity parameter was chosen using cross validation.
- Logistic regression was trained using all available predictors.

Evaluation

To evaluate the performance of different models, we split the data into train and test set. On the training set, we did cross validation technique to find the optimal set of parameters. To select the best model while performing hyper parameter tuning, Accuracy was used in case of classification and RMSE was used in case of Regression. Once all the model's performance were recorded using cross-validation results, we picked the best model. Then, the best model was tested on the test data set.

```
# regression using rpart
set.seed(42)
rpart_reg = train(
  quality_num ~ . - quality_cat - high_or_low,
  data = wine_trn,
  method = "rpart",
  trControl = trainControl(method = "cv",
                           number = 5)
)

rpart_reg
```

```
## CART
##
## 5197 samples
##   14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4158, 4158, 4157, 4158, 4157
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.02848989  0.7736536  0.2125911  0.6046265
##   0.04167014  0.7855984  0.1880197  0.6252948
##   0.16073518  0.8447095  0.1448791  0.6777504
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.02848989.
```

```
# regression using rf
set.seed(42)
rf_reg = train(
  quality_num ~ . - quality_cat - high_or_low,
  data = wine_trn,
  method = "rf",
  trControl = trainControl(method = "oob",
                           number = 5)
)

rf_reg
```

```
## Random Forest
##
## 5197 samples
```

```
##   14 predictor
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared
##    2    0.6070429  0.5147532
##    7    0.6085768  0.5122978
##   12    0.6116143  0.5074173
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
# regression using lasso
# create x matrix (train and test) for use with cv.glmnet()
wine_trn_x = model.matrix(quality_num ~ . - quality_cat - high_or_low, data = wine_trn)[, -1]
wine_tst_x = model.matrix(quality_num ~ . - quality_cat - high_or_low, data = wine_tst)[, -1]
set.seed(42)
cv_glmnet = cv.glmnet(
  wine_trn_x,
  wine_trn$quality_num,
  alpha = 1,
  lambda = 10 ^ seq(5, -5, length = 100),
  nfolds = 10
)
cv_glmnet$lambda.min
```

```
## [1] 0.0006579332
```

```
sqrt(cv_glmnet$cvm[cv_glmnet$lambda == cv_glmnet$lambda.1se])
```

```
## [1] 0.7486743
```

```
# regression using ridge
set.seed(42)
cv_glmnet = cv.glmnet(
  wine_trn_x,
  wine_trn$quality_num,
  alpha = 0,
  lambda = 10 ^ seq(5, -5, length = 100),
  nfolds = 10
)
cv_glmnet$lambda.min
```

```
## [1] 0.005336699
```

```
sqrt(cv_glmnet$cvm[cv_glmnet$lambda == cv_glmnet$lambda.1se])
```

```
## [1] 0.7484595
```

```r
# classification using rpart
set.seed(42)
rpart_classification = train(
  high_or_low ~ . - quality_cat - quality_num,
  data = wine_trn,
  method = "rpart",
  trControl = trainControl(method = "cv",
                           number = 5)
)

rpart_classification
```

```
## CART
##
## 5197 samples
##   14 predictor
##    2 classes: 'high', 'low'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4158, 4158, 4157, 4157, 4158
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01400000  0.8308646  0.31331267
##   0.02300000  0.8150863  0.12426931
##   0.02433333  0.8116214  0.08286067
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.014.
```

```r
# classification using glm
set.seed(42)
glm_classification = train(
  high_or_low ~ . - quality_cat - quality_num,
  data = wine_trn,
  method = "glm",
  trControl = trainControl(method = "cv",
                           number = 5)
)

glm_classification
```

```
## Generalized Linear Model
##
## 5197 samples
##   14 predictor
##    2 classes: 'high', 'low'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4158, 4158, 4157, 4157, 4158
## Resampling results:
```

```
##
##   Accuracy   Kappa
##   0.8210537  0.2762141
```

```
# classification using rf
set.seed(42)
rf_classification = train(
  high_or_low ~ . - quality_cat - quality_num,
  data = wine_trn,
  method = "rf",
  trControl = trainControl(method = "oob")
)

rf_classification
```

```
## Random Forest
##
## 5197 samples
##   14 predictor
##    2 classes: 'high', 'low'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8905138  0.5975967
##    7    0.8926304  0.6152518
##   12    0.8885896  0.6016169
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 7.
```

Results

Classification Models

```
reg_models = c('Random Forest',
               'Decision Tree',
               'Lasso Regression',
               'Ridge Regression')
reg_rmse = c(0.60, 0.77, 0.74, 0.74)
reg_description = c('mtry = 2', 'cp = 0.028', 'lambda = 0.00065', 'lambda = 0.005')

tibble('Model' = reg_models,
       'RMSE' = reg_rmse,
       'Model Description' = reg_description) %>%
  kable(digits = 2) %>%
  kable_styling("striped", full_width = FALSE)
```

Model	RMSE	Model Description
Random Forest	0.60	mtry = 2
Decision Tree	0.77	cp = 0.028
Lasso Regression	0.74	lambda = 0.00065
Ridge Regression	0.74	lambda = 0.005

Regression Models

```
classification_models = c('Random Forest', 'Decision Tree', 'Logistic Regression')
classification_accuracy = c(0.89, 0.83, 0.82)
classification_description = c('mtry = 7', 'cp = 0.014', '')

tibble(
  'Model' = classification_models,
  'Accuracy' = classification_accuracy,
  'Model Description' = classification_description
) %>%
  kable(digits = 2) %>%
  kable_styling("striped", full_width = FALSE)
```

Model	Accuracy	Model Description
Random Forest	0.89	mtry = 7
Decision Tree	0.83	cp = 0.014
Logistic Regression	0.82	

Discussion

```
# regression
p1 = predict(rf_reg, wine_tst)
test_rmse = sqrt(mean((p1 - wine_tst$quality_num) ^ 2))

# classification
p1 = predict(rf_classification, wine_tst)
test_accuracy = mean(p1 == wine_tst$high_or_low)
```

We found that the statistical models performed really well in case of classifying the wine quality as high or low. The accuracy on test data set was 88.38 %. Random Forest classifier gave the best performance.
In case of predicting the score of wine quality (regression problem), we observed that Random Forest regressor gave the best performance, with Root Mean Squared Error of 0.59 on the test data set.
We can conclude that the statistical models used in this analysis are providing good results. These models can be used in real life prediction scenarios.
Future work can include data about grape types, wine brand, wine selling price, etc, to make this analysis more interesting. Also, a larger dataset can be used, and it's optimality should be assessed. The current analysis is performed on a subset of dataset. Future work can also include using other metric for classification like Sensitivity, Specificity, or using other metric for regression problem.