HW_KNN

Riccardo Cappi

2024-03-27

KNN Homework

```
#TODO:
# - Feature engineering to reduce the feature dimensionality.
# - Perform k-fold cross-validation choosing a k value
# - Perform KNN on training set and on cross-validation set, by increasing K.
```

Importing required libraries

```
library(caret)
```

```
## Warning: il pacchetto 'caret' è stato creato con R versione 4.3.3
```

```
## Caricamento del pacchetto richiesto: ggplot2
```

```
## Warning: il pacchetto 'ggplot2' è stato creato con R versione 4.3.3
```

```
## Caricamento del pacchetto richiesto: lattice
```

```
library(ggplot2)
# library(class)
library(FNN)
```

```
## Warning: il pacchetto 'FNN' è stato creato con R versione 4.3.3
```

Import dataset

```
data = read.csv("wineq_train.csv")
summary(data)
```

```
##  fixed.acidity   volatile.acidity  citric.acid     residual.sugar
##  Min.   : 4.20   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.30   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.800
##  Median : 6.80   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.86   Mean   :0.2791   Mean   :0.3348   Mean   : 6.441
##  3rd Qu.: 7.30   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.:10.000
##  Max.   :14.20   Max.   :1.1000   Max.   :1.0000   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9938
##  Mean   :0.04562   Mean   : 35.51      Mean   :138.6        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:168.0        3rd Qu.:0.9962
##  Max.   :0.29000   Max.   :289.00      Max.   :440.0        Max.   :1.0390
```

```
##        pH          sulphates         alcohol         quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.40   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.186   Mean   :0.4899   Mean   :10.52   Mean   :5.879
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Feture engineering

```
# TODO: feature engineering

X <- data[, -12]
y <- data$quality

X <- as.data.frame(scale(X)) #scaling data
```

RMSE function

```
RMSE <- function(y_true, y_pred){
  return(sqrt(mean((y_true - y_pred)^2)))
}
```

KNN function

```
fit_knn <- function(X_train, y_train, X_test, k_val){
  y_hat = knn.reg(train = X_train, test=X_test, y=y_train, k = k_val)
  y_hat = y_hat$pred
  return(y_hat)
}
```

Fit KNN on training set

```
k_range = seq(1, 50, by = 1)
train_rmse <- c()
for (k in k_range){
  y_hat = fit_knn(X, y, X, k)
  train_rmse <- append(train_rmse, RMSE(y, y_hat))
}
train_rmse
```

```
##  [1] 0.0000000 0.3882746 0.4871092 0.5371067 0.5649488 0.5859896 0.6024887
##  [8] 0.6123310 0.6217223 0.6324769 0.6407542 0.6462169 0.6504363 0.6537753
## [15] 0.6585097 0.6618090 0.6657744 0.6683665 0.6707874 0.6729203 0.6759784
## [22] 0.6774624 0.6791875 0.6804889 0.6820559 0.6841196 0.6852982 0.6865546
## [29] 0.6871816 0.6879484 0.6894601 0.6917271 0.6924031 0.6933596 0.6943612
## [36] 0.6946163 0.6950260 0.6963632 0.6975669 0.6982346 0.6992055 0.6999242
## [43] 0.7009269 0.7017697 0.7026195 0.7031184 0.7039087 0.7047570 0.7054662
## [50] 0.7059670
```

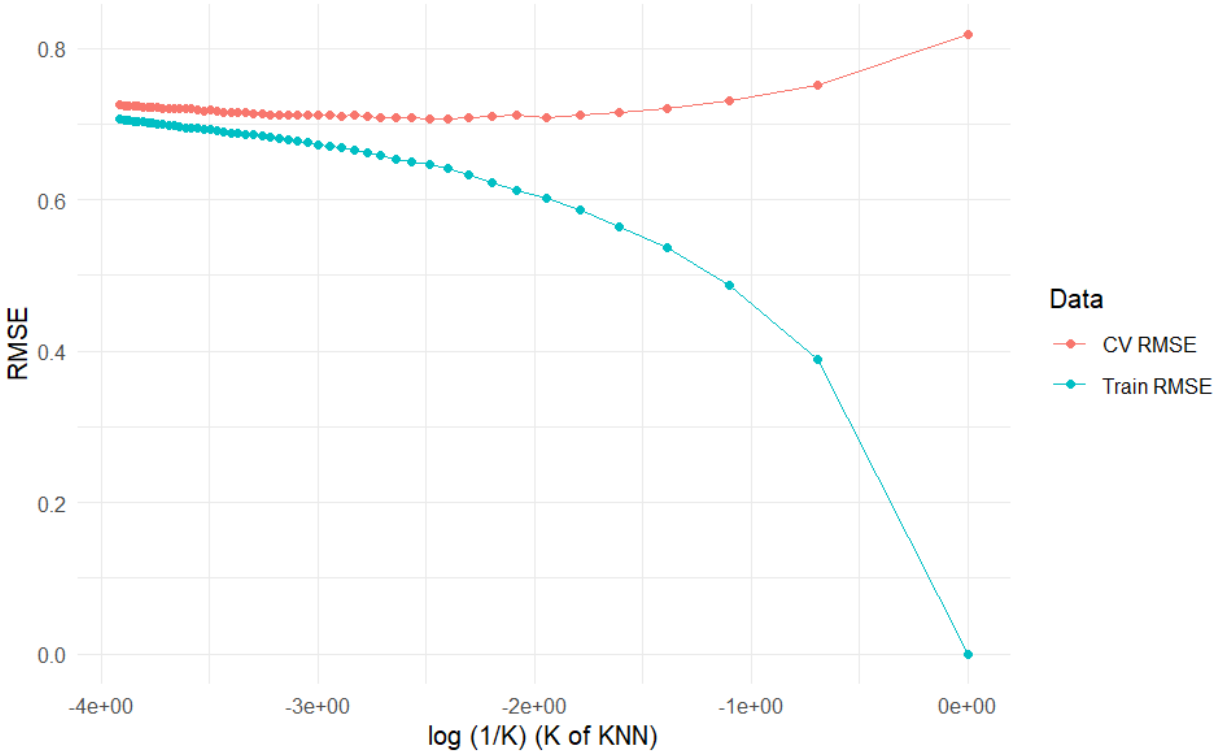
k-fold cross validation

```
set.seed(42)
folds <- createFolds(y, k = 5, list = TRUE)
cv_rmse <- c()

for (k in k_range){
  cv_rmse_fold <- c()
  for (i in 1:length(folds)) {
    #unlist function flatten the list. Remember that folds is a list of lists
    train_indexes <- unlist(folds[-i])
    test_indexes <- unlist(folds[i])

    X_train <- X[train_indexes, ]
    y_train <- y[train_indexes]
    X_test <- X[test_indexes, ]
    y_test <- y[test_indexes]

    y_hat <- fit_knn(X_train, y_train, X_test, k)
    cv_rmse_fold <- append(cv_rmse_fold, RMSE(y_test, y_hat))
  }
  cv_rmse <- append(cv_rmse, mean(cv_rmse_fold))
}
cv_rmse
```

```
##  [1] 0.8175592 0.7513217 0.7298486 0.7197063 0.7157152 0.7116402 0.7087802
##  [8] 0.7109160 0.7092348 0.7084734 0.7067160 0.7068999 0.7082562 0.7080684
## [15] 0.7088415 0.7103726 0.7111282 0.7106457 0.7114006 0.7115652 0.7118155
## [22] 0.7121506 0.7120951 0.7119807 0.7123931 0.7130063 0.7133882 0.7143712
## [29] 0.7153951 0.7150090 0.7157212 0.7170293 0.7177516 0.7177353 0.7182535
## [36] 0.7195023 0.7199831 0.7199234 0.7202568 0.7207065 0.7209610 0.7216148
## [43] 0.7218227 0.7221640 0.7226323 0.7235589 0.7236659 0.7239220 0.7245744
## [50] 0.7251670
```

Plots

```
results <- data.frame(k = k_range, train_rmse = train_rmse, cv_rmse = cv_rmse)
ggplot(results, aes(x = log(1/k), y = train_rmse, color = "Train RMSE")) +
  geom_line() +
  geom_point() +
  geom_line(aes(x = log(1/k), y = cv_rmse, color = "CV RMSE")) +
  geom_point(aes(x = log(1/k), y = cv_rmse, color = "CV RMSE")) +
  scale_x_continuous(labels = scales::scientific_format()) +
  labs(x = "log (1/K) (K of KNN)", y = "RMSE", color = "Data") +
  theme_minimal()
```



TODO: test the model with the lowest cv error