Model Stacking for Kaggle

James Caldwell

2024-04-15

Kaggle Housing Price Prediction: Model Stacking

This repository contains the code and results of a Kaggle competition for predicting housing prices using model stacking. The solution utilizes three models: Random Forest, L2 Tree Boosting, and Penalized Linear Regression, combined using model stacking to improve prediction accuracy.

Kaggle Competition Details

- Kaggle User Name: JamesCaldwell1
- Best Score: 0.14824 (Ranked 2290), goal was < 0.5 RMSE.

Project Structure

- **data/**: Contains the training and test datasets (`train.csv` and `test.csv`).
- **src/**: Contains the R scripts for data cleaning, model training, and prediction.
  - `main.R` : Main script for running the entire pipeline.
  - `rf_model.R` : Script for building and saving the Random Forest model.
  - `l2_boosting_model.R` : Script for building and saving the L2 Tree Boosting model.
  - `lr_model.R` : Script for building and saving the Penalized Linear Regression model.
  - `model_stacking.R` : Script for combining predictions from the individual models using model stacking.
- **results/**: Contains the output CSV files with predictions.
  - `Z1_rf.csv` : Random Forest predictions.
  - `Z2_L2.csv` : L2 Tree Boosting predictions.
  - `Z3_lr.csv` : Penalized Linear Regression predictions.
  - `Caldwell.csv` : Stacked predictions.

The final predictions are stored in the `Caldwell.csv` file in the `results/` directory.

My solution uses model stacking of 3 models: random forests, L2 tree boosting, and penalized linear regression.

```
suppressWarnings({
library(readr)
library(glmnet)
library(tidyverse)
library(ranger)
library(janitor)
library(dplyr)
})
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ purrr     1.0.2
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.0
## ── Conflicts ──────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ tidyr::expand() masks Matrix::expand()
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ tidyr::pack()   masks Matrix::pack()
## ✖ tidyr::unpack() masks Matrix::unpack()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
##
## Attaching package: 'janitor'
##
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
# Load Data
train = read_csv('train.csv') #%>% clean_names()
```

```
## Rows: 1460 Columns: 81
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (43): MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConf...
## dbl (38): Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, Ye...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
test = read_csv('test.csv') #%>% clean_names()
```

```
## Rows: 1459 Columns: 80
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (43): MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConf...
## dbl (37): Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, Ye...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
## Data Cleaning
# Cleaning train data
# Impute missing values for numeric columns with the mean
numeric_cols <- sapply(train, is.numeric)
train[ , numeric_cols] <- lapply(train[, numeric_cols], function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x))
# Impute missing values for categorical columns with the mode
categorical_cols <- sapply(train, function(x) !is.numeric(x))
train[ , categorical_cols] <- lapply(train[, categorical_cols], function(x) ifelse(is.na(x), names(sort(table(x), decreasing = TRUE))[1], x))

# Cleaning test data
# Impute missing values for numeric columns with the mean
numeric_cols <- sapply(test, is.numeric)
test[, numeric_cols] <- lapply(test[, numeric_cols], function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x))
# Impute missing values for categorical columns with the mode
categorical_cols <- sapply(test, function(x) !is.numeric(x))
test[, categorical_cols] <- lapply(test[, categorical_cols], function(x) ifelse(is.na(x), names(sort(table(x), decreasing = TRUE))[1], x))

X = glmnet::makeX(select(train, -SalePrice), test)
X.train = X$x
X.test = X$xtest
Y.train = as.data.frame(train$SalePrice)
colnames(Y.train) <- "SalePrice"

# print(head(X.train))
```

Model #1: Random Forest

```r
# MODEL #1: Random Forest
rf_model <- ranger(SalePrice ~ ., data = train)
# Make predictions
Z1_rf <- predict(rf_model, data = test)$predictions

# Combine predictions and the X Test Id column
Z1_rf_csv <- cbind(Id = X.test[, "Id", drop = FALSE], SalePrice = Z1_rf)

# Write the submission data to a CSV file
write.csv(Z1_rf_csv, file = "Z1_rf.csv", row.names = FALSE)
```

Model #2: L2 Tree Boosting

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.3
```

```
## Loaded gbm 2.1.9
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3
```

```
gbm_train = cbind(X.train,Y.train)

# Initialize the gradient boosting model
gbm_model <- gbm(
  formula = SalePrice ~ .,
  data = gbm_train,
  distribution = "gaussian",  # For regression
  n.trees = 100,  # Number of boosting iterations
  interaction.depth = 3,  # Maximum depth of each tree
  shrinkage = 0.1,  # Learning rate
  bag.fraction = 0.5,  # Fraction of observations to be used for each tree
  train.fraction = 1,  # Fraction of data to be used for training (1 for using all data)
  n.minobsinnode = 10,  # Minimum number of observations in terminal nodes
  verbose = TRUE  # To see the progress of training
)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1 5541301263.8949             nan     0.1000 696948605.0984
##      2 4914300601.0842             nan     0.1000 603701455.8769
##      3 4379834656.7642             nan     0.1000 491809845.6934
##      4 3916473503.9772             nan     0.1000 463062611.8923
##      5 3554128730.9246             nan     0.1000 373000673.6453
##      6 3246713703.4750             nan     0.1000 315296852.7312
##      7 2961341728.5490             nan     0.1000 259341417.7149
##      8 2678801049.9194             nan     0.1000 225150443.2480
##      9 2451856621.2284             nan     0.1000 225634152.9693
##     10 2270435416.8668             nan     0.1000 153543592.2882
##     20 1273575362.3355             nan     0.1000 38578954.8312
##     40 763560568.2953             nan     0.1000 2027800.3868
##     60 604011100.5461             nan     0.1000 -3450191.9873
##     80 534669047.7984             nan     0.1000 -2107584.9870
##    100 482835711.8926             nan     0.1000 -4011466.9271
```

```
# Make predictions on the test set
Z2_L2 <- predict(gbm_model, newdata = as.data.frame(X.test), n.trees = 100)  # Assuming your test data is i
n a data frame called test_data

# Combine Id column from X.test with Z2_L2 predictions
Z2_L2_csv <- cbind(Id = X.test[, "Id", drop = FALSE], SalePrice = Z2_L2)

# Write the submission data to a CSV file
write.csv(Z2_L2_csv, file = "Z2_L2.csv", row.names = FALSE)
```

Model #3: (penalized) linear regression

```r
library(glmnet)
set.seed(2023)

# str(X.train)
# str(Y.train)
g2 = cv.glmnet(X.train, Y.train$SalePrice) # tune lambda with 10-fold cv
Z3_lr = predict(g2, X.test, s = "lambda.min") # choose lambda.min

# print(Z2_lr)
# Assign custom column names
colnames(Z3_lr) <- ("SalePrice")

write.csv(Z3_lr, file = "Z3_lr.csv", row.names = TRUE)
```

Use Model stacking to aggregate RF, L2 boosting, and LM.

Here, I use a simple weighted averaging to assemble the final model. Since the 1st two models performed better than the LR model, I'll assign a higher weight to those models

If I was to spend more time on this, I would probably use hold out data with cross validation to improve my model performance and get better estimates for the weights to use

```r
# average_predictions <- (Z1_rf + Z2_L2 + Z3_lr) / 3 #This is pure averaging
# which I ended up not using
average_predictions <- (Z1_rf*.4 + Z2_L2*.4 + Z3_lr*.2)

# Combine Id column from X.test with Z2_L2 predictions
yhat_stacked <- cbind(Id = X.test[, "Id", drop = FALSE], SalePrice = average_predictions)

# Write the submission data to a CSV file
write.csv(yhat_stacked, file = "Caldwell.csv", row.names = FALSE)
```