Assignment 4: Climbing Fatalty Predictions

Colleen McCamy

2023-05-02

Support Vector Machine (SVM) Classification Model for Topic Classification

Reading in the Data

```
# data location
urlfile ="https://raw.githubusercontent.com/MaRo406/EDS-231-text-sentiment/main/data/climbing_reports_m

wd <- setwd("~/Documents/MEDS/classes/spring/eds-231-text-analysis")

# reading in the data
incidents_df<-readr::read_csv(url(urlfile)) #|> write_csv(paste0(wd, "incident_text_dat_raw.csv"))
```

Splitting Data into Testing & Training

```
set.seed(1234)

# data pre-processing
incidents2class <- incidents_df |>
  mutate(fatal = factor(if_else(
    is.na(Deadly) ,
    "non-fatal", "fatal")))

# data splitting for training & testing
incidents_split <- initial_split(incidents2class, strata = fatal)
incidents_train <- training(incidents_split)
incidents_test <- testing(incidents_split)
```

Specifying the Predictor & Outcome Variables

```
# setting up the recipe
incidents_rec <- recipe(fatal ~ Text,
                        data = incidents_train)
```

Pre-Processing Text Data in a Recipe

```r
# pre-processing the data
recipe <- incidents_rec |>
  step_tokenize(Text) |>
  step_tokenfilter(Text, max_tokens = 1000) |>
  step_tfidf(Text)
```

Creating a Workflow

```r
incidents_wf <- workflow() |>
  add_recipe(recipe)
```

Model Selection

```r
# selecting the svm model
svm_spec <- svm_rbf() |>
  set_mode("classification") |> # set modeling context
  set_engine("kernlab") # method for fitting model
```

Fitting the Model to the Training Data

```r
# fitting our model to the training data
svm_fit <- incidents_wf |>
  add_model(svm_spec) |>
  fit(data = incidents_train)
```

Model Evaluation

```r
set.seed(363)

# creating cross validation folds
incidents_folds <- vfold_cv(incidents_train) #default is v = 10

# establishing the workflow
svm_wf <- workflow() |>
  add_recipe(recipe) |>
  add_model(svm_spec)

tic() # start time

# fitting the model to the cross validation samples
svm_results <- svm_wf |>
  fit_resamples(
  incidents_folds,
  control = control_resamples(save_pred = TRUE))

toc() # end time
```

```
## 170.429 sec elapsed
```

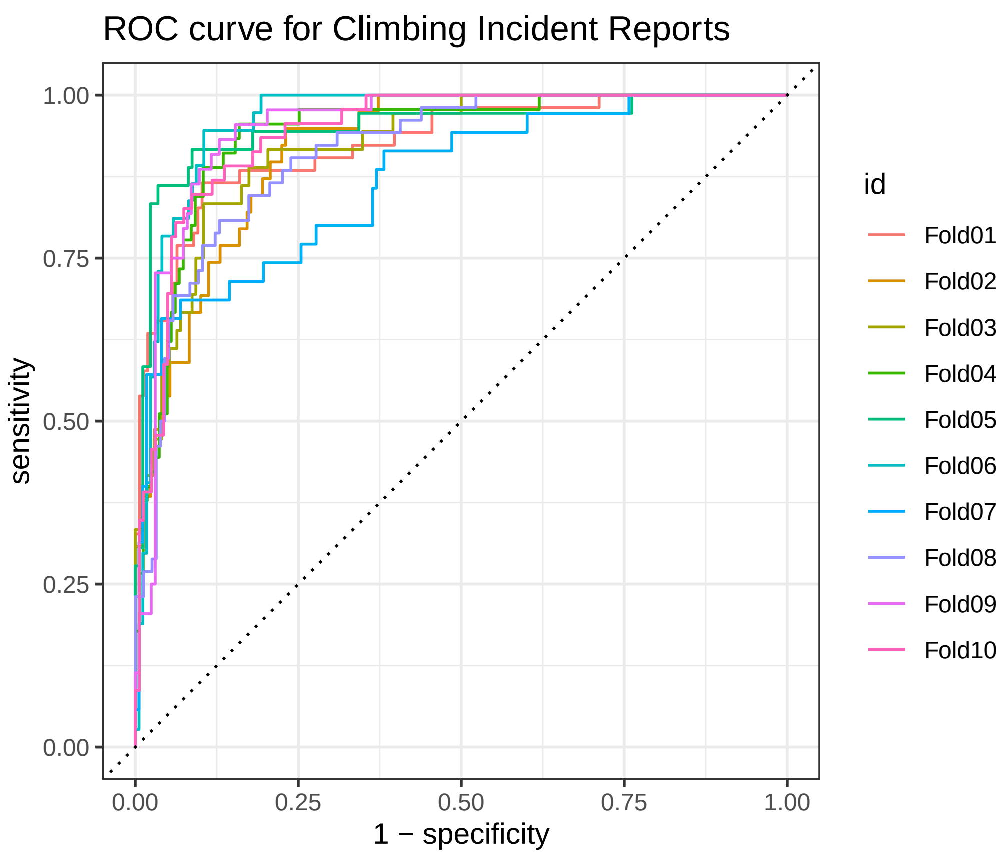
```r
# evaluating model metrics
svm_metrics <- collect_metrics(svm_results)
svm_predictions <- collect_predictions(svm_results)
svm_metrics <- svm_metrics |>
  select(.metric, mean) |>
  rename("Metric" = .metric,
         "Value" = mean)

table <- svm_metrics
```

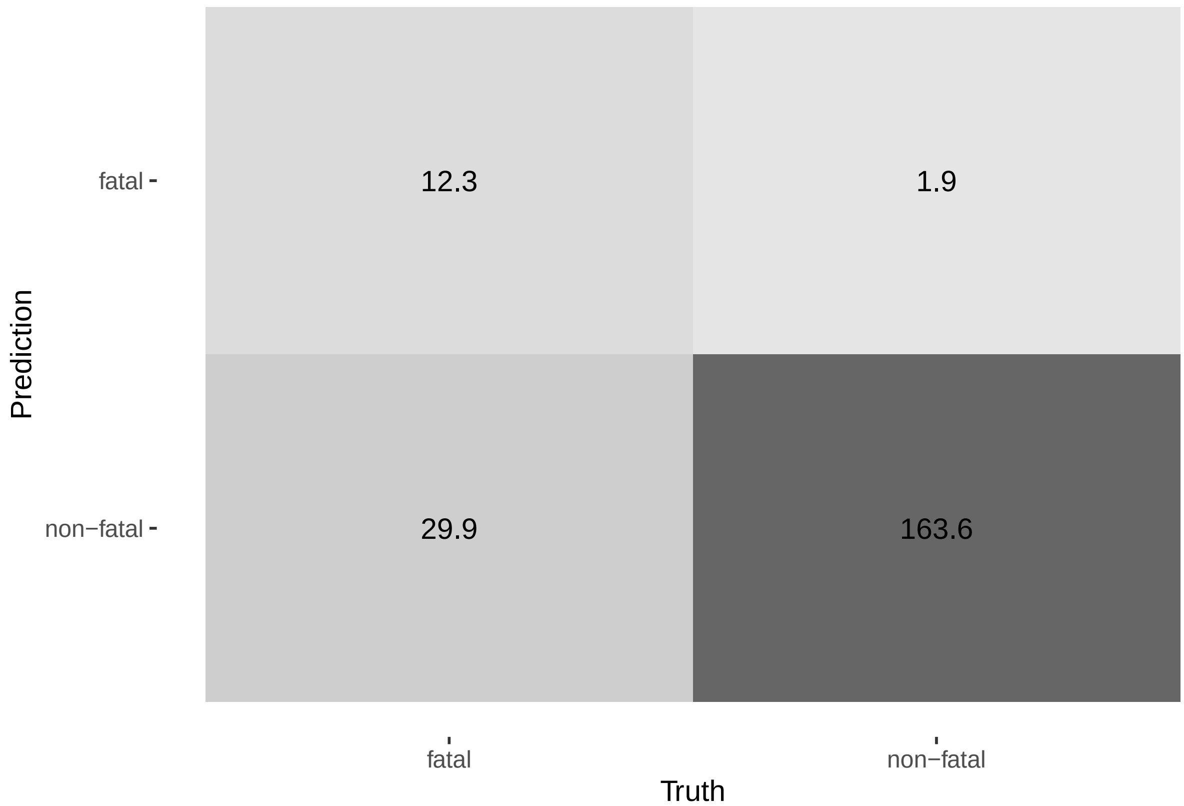
```r
# printing table results
table
```

```
## # A tibble: 2 x 2
##   Metric   Value
##   <chr>    <dbl>
## 1 accuracy 0.847
## 2 roc_auc  0.929
```

```r
# creating a performance plot
svm_predictions |>
  group_by(id) |>
  roc_curve(truth = fatal, .pred_fatal) |>
  autoplot() +
  labs(
    "Resamples",
    title = "ROC curve for Climbing Incident Reports"
  )
```




```r
# confusion matrix
conf_mat_resampled(svm_results,
                   tidy = FALSE) |>  #compute matrix for each fold then average
  autoplot(type = "heatmap")
```



Conducting a Tuned Support Vector Machine

Tuning the Model Hyperparameters

```
# new tuning specification
svm_spec_tune <- svm_rbf(cost = tune(),
                         rbf_sigma = tune()) |>
  set_mode("classification") |>
  set_engine("kernlab")

# establishing the workflow
svm_wf_tune <- workflow() |>
  add_recipe(recipe) |>
  add_model(svm_spec_tune)

# establishing parameters for the tuning
svm_param <- svm_wf_tune |>
  parameters() |>
  update(rbf_sigma = rbf_sigma(c(-7,-1)))

# getting grid to resample values
start_grid <- svm_param |>
  update(cost = cost(c(-6,1)),
         rbf_sigma = rbf_sigma(c(-6,-4))) |>
```

```r
  grid_regular(levels = 2)

# setting the seed
set.seed(363)

# start time
tic()

# fitting a model to each tuning value
svm_tune <- svm_wf_tune |>  #start with the tuning workflow
  tune_grid(resamples = incidents_folds, # giving it the cv folds
            grid = start_grid, # grid to pull numbers from
            control = control_resamples(save_pred = T)) # access to results for plotting

# end time
toc()
```

```
## 516.167 sec elapsed
```

Evaluating the Model

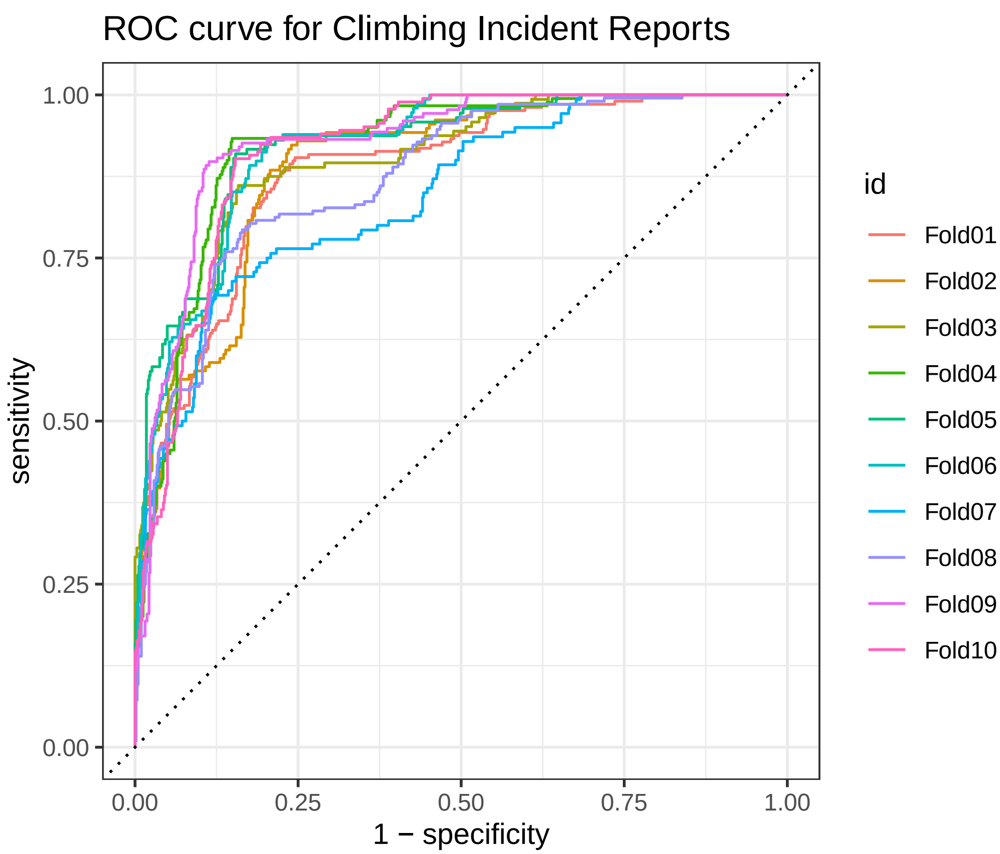
```r
# evaluating model metrics
svm_tune_metrics <- collect_metrics(svm_tune)
svm__tune_predictions <- collect_predictions(svm_tune)
svm_tune_metrics <- svm_tune_metrics |>
  select(.metric, mean) |>
  rename("Metric" = .metric,
         "Value" = mean)

# creating the tuned metrics table
table_tune <- svm_metrics |> gt::gt(caption = "Model Metrics - Tuning")
```

```r
# printing the tune table
table_tune
```

Metric	Value
accuracy	0.8468529
roc_auc	0.9290063

```r
# creating a performance plot
svm__tune_predictions |>
  group_by(id) |>
  roc_curve(truth = fatal, .pred_fatal) |>
  autoplot() +
  labs(
    "Resamples",
    title = "ROC curve for Climbing Incident Reports")
```




```r
# seeing the top auc
svm_tune |>
  show_best("roc_auc")
```

```
## # A tibble: 4 x 8
##     cost rbf_sigma .metric .estimator  mean     n std_err .config
##    <dbl>     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 2        0.0001   roc_auc binary     0.928    10 0.00723 Preprocessor1_Model4
## 2 2        0.000001 roc_auc binary     0.927    10 0.00721 Preprocessor1_Model2
## 3 0.0156   0.0001   roc_auc binary     0.927    10 0.00703 Preprocessor1_Model3
## 4 0.0156   0.000001 roc_auc binary     0.915    10 0.00764 Preprocessor1_Model1
```

```r
# seeing the top auc
svm_tune |>
  show_best("accuracy")
```

```
## # A tibble: 4 x 8
##     cost rbf_sigma .metric  .estimator  mean     n std_err .config
##    <dbl>     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 2        0.0001   accuracy binary     0.829    10  0.0107 Preprocessor1_Model4
## 2 0.0156   0.000001 accuracy binary     0.797    10  0.0100 Preprocessor1_Model1
## 3 2        0.000001 accuracy binary     0.797    10  0.0100 Preprocessor1_Model2
## 4 0.0156   0.0001   accuracy binary     0.797    10  0.0100 Preprocessor1_Model3
```

```r
# selecting the best auc
chosen_auc <- svm_tune |>
  select_best(metric = "roc_auc")
```

Finalizing Model

```r
# finalizing workflow
final_svm <- finalize_workflow(svm_wf_tune,
                                chosen_auc)

# fit final workflow the training data
final_fit <- final_svm |>
  fit(incidents_train)
```

Looking at Top Features

```r
# # fitting the model on the
# svm_model <- final_fit |>
#   pull_workflow_fit()
#
# # getting the data from the prep
# prepped_data <- final_fit |>
#   extract_fit_parsnip() |>
#   bake(new_data = incidents_train)
#
#
# svm_fit |>
#   pull_workflow_fit() |>
#   vip()
#
# svm_fit |>
#   pull_workflow_fit() |>
#   vip(method = "permute",
#       nsim = 10,
#       target = "fatal", metric = "auc",
#       reference_class = "non-fatal",
#       pred_wrapper = kernlab::predict,
#       train = incidents_train)
```

Predicting Fatality Reports on Test Data

```r
# conducting a last fit
final_metrics <- last_fit(final_fit,
         incidents_split) |>
  collect_metrics()

# selecting what we want to show in the table
```

```r
final_metrics <- final_metrics |>
  select(.metric, .estimate) |>
  rename("Metric" = .metric,
         "Value" = .estimate)

table_final <- final_metrics |> gt::gt(caption = "Final Model Metrics")
```

```r
table_final
```

Metric	Value
accuracy	0.8427128
roc_auc	0.9434294

RESPONSE: This model did not have as high area under the ROC curve or accuracy compared to the Lasso regression model. However, this model performed similarly to the Naive-Bayes as it had a slightly larger area under the ROC curve but slightly lower accuracy. All in all, since you are not able to extract feature importance with this model, I wouldn't use this model future text classification tasks.