2019 UKC Prototype Study

Albert Lee

2019-08-10

Meta

Author: Albert Lee **Date Created**: 2019-07-24 **Date Updated**: 2019-08-10 **Environment** :

Introduction

- **Dataset**: https://data.cityofchicago.org/
- **Hackathon Note**: https://docs.google.com/document/u/1/d/1d8tgkLKcJwN7oy-W9h0R0IHJSFlz0H2tUJtG9LUn1hw
  edit?ouid=101681315319651182806&usp=docs_home&ths=true
- **data description**: https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD

Questions to ask

- What is the crime rate?
- The most common name in Chicago?
- What is the most useful predictor to predict the type of crime?
- etc

Data Science Questions

Can you predict a crime type based on the feature?

Data

Crime data

- Issue: the dataset is huge. takes a long time to download (1.8G)

So will use the reduced version of it. See

```
df_top10_samples <- read_rds(here::here("df_top10_samples.csv"))
```

```
df_crime_types <- df_top10_samples %>%
  count(primary_type) %>%
  arrange(desc(n)) %>%
  mutate(`percent_crime` = scales::percent(n/sum(n)))

knitr::kable(df_crime_types)
```

primary_type	n	percent_crime
ASSAULT	20000	10.0%
BATTERY	20000	10.0%
BURGLARY	20000	10.0%
CRIMINAL DAMAGE	20000	10.0%
DECEPTIVE PRACTICE	20000	10.0%
MOTOR VEHICLE THEFT	20000	10.0%
NARCOTICS	20000	10.0%
OTHER OFFENSE	20000	10.0%
ROBBERY	20000	10.0%
THEFT	20000	10.0%

Location

EDA: Is there a difference in crime type and rate at different times?

```
df_top10_samples %>%
  select(primary_type, date) %>%
  mutate(time=hour(date)) %>%
  select(-date) %>%
  count(primary_type, time) %>%
  group_by(primary_type) %>%
  mutate(rate=n/sum(n)) %>%
  ungroup() %>%
  ggplot(aes(x=time, y=rate,
             color=primary_type,
             group=primary_type
             )) +
  geom_point() +
  geom_line() -> p


ggplotly(p)
```

```
# Y axis:
# X-axis Hours
```

Modeling - Machine Learning

Feature engineering / Split

```
set.seed(628)

data_in <- df_top10_samples %>%
  mutate(hour=lubridate::hour(date)) %>%
  select(primary_type, x_coord, y_coord, hour)

# Training/Testing Split --------------------
data_split <- initial_split(data_in, strata = "primary_type", p = 0.75)
```

```
## Warning: Too little data to stratify. Unstratified resampling will be used.
```

```r
train_data <- training(data_split)
test_data  <- testing(data_split)

model_rec <- recipe(primary_type ~ ., data = train_data) %>%
                step_center(x_coord,y_coord) %>%
                step_scale(x_coord,y_coord)

summary(model_rec, original = TRUE)
```

```
## # A tibble: 4 x 4
##   variable     type    role      source
##   <chr>        <chr>   <chr>     <chr>
## 1 x_coord      numeric predictor original
## 2 y_coord      numeric predictor original
## 3 hour         numeric predictor original
## 4 primary_type nominal outcome   original
```

Preprocessing before machine learning

The following is the prepping done before fitting the ML model

```r
model_prepped <- prep(model_rec, training = train_data)
tidy(model_prepped)
```

```
## # A tibble: 2 x 6
##   number operation type   trained skip  id
##    <int> <chr>     <chr>  <lgl>   <lgl> <chr>
## 1      1 step      center TRUE    FALSE center_dEN1Y
## 2      2 step      scale  TRUE    FALSE scale_I7wJv
```

```r
juice(model_prepped) %>%
  head()
```

```
## # A tibble: 6 x 4
##   x_coord y_coord  hour primary_type
##     <dbl>   <dbl> <int> <fct>
## 1  -2.49    1.58     13 ASSAULT
## 2   0.636  -0.716    20 ASSAULT
## 3  -0.897   1.05      9 ASSAULT
## 4   0.118  -0.754     6 ASSAULT
## 5  -1.42    1.02      0 ASSAULT
## 6   0.568   0.433    11 ASSAULT
```

Apply Preprocessing

> During the process of preparing the recipe, each step is estimated via prep and then applied to the training set using bake before proceeding to the next step. After the recipe has been prepared, bake can be used with any data set to apply the preprocessing to those data. https://cran.r-project.org/web/packages/recipes/vignettes/Skipping.html

```r
baked_train_data <- bake(model_prepped, new_data = train_data)
baked_test_data  <- bake(model_prepped, new_data = test_data)
```

```r
model_ctrl <- fit_control(verbosity=2L)

model_fit <- boost_tree(trees = 5) %>%
               set_mode("classification") %>%
               set_engine("xgboost",
                          silent=0,
                          verbose = 1L # verbose = 1 means print evaluation metric
                          ) %>%
               fit(formula(model_prepped),
                   data = baked_train_data,
                   control=model_ctrl)

write_rds(model_fit, "basic_model.rds")
```

Model Performance

```r
df_pred <- predict(model_fit, new_data=baked_test_data, type=c("prob")) %>%
  mutate(actual=baked_test_data$primary_type) %>%
  select(actual, everything())

# Cross entropy
df_metrics_crossentropy <- df_pred %>%
    mn_log_loss(actual, 2:ncol(.))

# Accuracy and Kappa
df_pred_class <- predict(model_fit, new_data=baked_test_data, type=c("class")) %>%
  mutate(actual=baked_test_data$primary_type) %>%
  select(actual, everything())
df_metrics_classes <- metrics(df_pred_class, truth = actual, estimate=.pred_class)

bind_rows(df_metrics_classes, df_metrics_crossentropy)
```

```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    multiclass     0.192
## 2 kap         multiclass     0.103
## 3 mn_log_loss multiclass     2.21
```

Conclusion

Appendix

IUCR

IUCR : Illinois Uniform Crime Reporting (IUCR) codes are four digit codes that law enforcement agencies use to classify criminal incidents when taking individual reports. … The Chicago Police Department currently uses more than 350 IUCR codes to classify criminal offenses, divided into “Index” and “Non-Index” offenses.

https://data.cityofchicago.org/Public-Safety/Chicago-Police-Department-Illinois-Uniform-Crime-R/c7ck-438e/data

FBI code

FBI Code Indicates the crime classification as outlined in the FBI's National Incident-Based Reporting System (NIBRS). See the Chicago Police Department listing of these classifications at http://gis.chicagopolice.org/clearmap_crime_sums/crime_types.html. Plain Text

Community area

Indicates the community area where the incident occurred. Chicago has 77 community areas. See the community areas at https://data.cityofchicago.org/d/cauq-8yn6.

Session Information

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] yardstick_0.0.3  rsample_0.0.4    recipes_0.1.4    parsnip_0.0.2
##  [5] infer_0.4.0.1    dials_0.0.2      scales_1.0.0     broom_0.5.1
##  [9] tidymodels_0.0.2 tictoc_1.0       lubridate_1.7.4  plotly_4.9.0
## [13] here_0.1         glue_1.3.1       forcats_0.3.0    stringr_1.4.0
## [17] dplyr_0.8.0.1    purrr_0.3.2      readr_1.3.1      tidyr_0.8.2
## [21] tibble_2.1.3     ggplot2_3.1.0    tidyverse_1.2.1  nvimcom_0.9-83
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.2.0        backports_1.1.4     tidytext_0.2.1
##   [4] plyr_1.8.4          igraph_1.2.4.1      lazyeval_0.2.2
##   [7] splines_3.5.2       crosstalk_1.0.0     SnowballC_0.6.0
##  [10] rstantools_1.5.1    inline_0.3.15       digest_0.6.20
##  [13] htmltools_0.3.6     rsconnect_0.8.13    fansi_0.4.0
##  [16] magrittr_1.5        modelr_0.1.2        gower_0.1.2
##  [19] matrixStats_0.54.0  xts_0.11-2          prettyunits_1.0.2
##  [22] colorspace_1.3-2    rvest_0.3.2         haven_2.1.0
##  [25] xfun_0.8            callr_3.3.0         crayon_1.3.4
##  [28] jsonlite_1.6        lme4_1.1-21         zeallot_0.1.0
##  [31] survival_2.43-3     zoo_1.8-6           gtable_0.2.0
##  [34] ipred_0.9-8         pkgbuild_1.0.2      rstan_2.18.2
##  [37] miniUI_0.1.1.1       Rcpp_1.0.1         viridisLite_0.3.0
##  [40] xtable_1.8-3         stats4_3.5.2       lava_1.6.4
##  [43] StanHeaders_2.18.0-1 prodlim_2018.04.18 DT_0.5
##  [46] htmlwidgets_1.3      httr_1.4.0         threejs_0.3.2
##  [49] pkgconfig_2.0.2      loo_2.0.0          nnet_7.3-12
##  [52] utf8_1.1.4           labeling_0.3       tidyselect_0.2.5
##  [55] rlang_0.4.0          reshape2_1.4.3     later_0.7.5
##  [58] munsell_0.5.0        cellranger_1.1.0   tools_3.5.2
##  [61] xgboost_0.90.0.2     cli_1.1.0          generics_0.0.2
##  [64] ggridges_0.5.1       evaluate_0.14      yaml_2.2.0
##  [67] processx_3.4.0       knitr_1.23         nlme_3.1-137
##  [70] mime_0.7             rstanarm_2.18.2    xml2_1.2.0
##  [73] tokenizers_0.2.1     compiler_3.5.2     bayesplot_1.7.0
##  [76] shinythemes_1.1.2    rstudioapi_0.10    tidyposterior_0.0.2
##  [79] stringi_1.4.3        highr_0.8          ps_1.3.0
##  [82] lattice_0.20-38      Matrix_1.2-15      nloptr_1.2.1
##  [85] markdown_1.0         vctrs_0.1.0        shinyjs_1.0
##  [88] pillar_1.4.2         data.table_1.12.0  httpuv_1.4.5.1
##  [91] R6_2.4.0             promises_1.0.1     gridExtra_2.3
##  [94] janeaustenr_0.1.5    codetools_0.2-15   boot_1.3-20
##  [97] colourpicker_1.0     MASS_7.3-51.1      gtools_3.8.1
## [100] assertthat_0.2.1     rprojroot_1.3-2    withr_2.1.2
## [103] shinystan_2.5.0      parallel_3.5.2     hms_0.4.2
## [106] grid_3.5.2           rpart_4.1-13       timeDate_3043.102
## [109] class_7.3-14         minqa_1.2.4        rmarkdown_1.14
## [112] pROC_1.15.0          tidypredict_0.3.0  shiny_1.2.0
## [115] base64enc_0.1-3      dygraphs_1.1.1.6
```

Time to Knit

```
## Knitting the document: 1.467 sec elapsed
```