STAT 6324 Final Project

Art Tay

```
# Setup parallel computing.
#library(doParallel)
#cl <- makePSOCKcluster(24)
#registerDoParallel(cl)
```

```
# Read in the dataset.
data_full <- read.csv("heart_2020_cleaned.csv", stringsAsFactors = T)
```

```
# 70/30 stratified train-test split.
# Stratification to ensure that the proportion of positive
# heart disease case is equivalent across the training,
# testing, and raw datasets.
set.seed(123)
split <- initial_split(data_full, prop = 0.7, strata = HeartDisease)
data_train <- training(split)
data_test <- testing(split)
```

```
# Defines a recipe for data cleaning.
cleaning_recipe <- data_train %>% recipe(HeartDisease ~ .)
```

```
# Feature Engineering
cleaning_recipe %<>%
    # Adds categorical bins for Physical Health.
    step_mutate(PhysicalHealth_bin =
                    cut(PhysicalHealth,
                        breaks = c(-1, 0, 5, 10, 15, 20, 25, 30))) %>%
    # Adds categorical bins for Mental Health.
    step_mutate(MentalHealth_bin =
                    cut(MentalHealth,
                        breaks = c(-1, 0, 5, 10, 15, 20, 25, 30))) %>%
    # Takes the mid-points for AgeCategory to form a numeric variable.
    step_mutate(AgeCategory_Cont = case_when(
                    AgeCategory == "18-24" ~ 21,
                    AgeCategory == "25-29" ~ 27,
                    AgeCategory == "30-34" ~ 32,
                    AgeCategory == "35-39" ~ 37,
                    AgeCategory == "40-44" ~ 42,
                    AgeCategory == "45-49" ~ 47,
                    AgeCategory == "50-54" ~ 52,
                    AgeCategory == "55-59" ~ 57,
                    AgeCategory == "60-64" ~ 62,
                    AgeCategory == "65-69" ~ 67,
                    AgeCategory == "70-74" ~ 72,
```

```
                    AgeCategory == "75-79" ~ 77,
                    TRUE ~ 82
                )) %>%
    # Assigns 1-5 numeric values to the likert scale for General Health.
    step_mutate(GenHealth_ord = case_when(
                    GenHealth == "Poor" ~ 1,
                    GenHealth == "Fair" ~ 2,
                    GenHealth == "Good" ~ 3,
                    GenHealth == "Very Good" ~ 4,
                    TRUE ~ 5
                )) %>%
    # Categorical bins for Low Sleep, Too Much and Regular.
    step_mutate(SleepTime_bin = as.factor(case_when(
                    SleepTime < 6 ~ "Low",
                    SleepTime > 9 ~ "Too Much",
                    TRUE ~ "Regular")
                ))
```

```
# Centers and scales all numeric predictors.
cleaning_recipe %<>%
    step_normalize(all_numeric_predictors())
```

```
# Applies SMOTE method to deal with class imbalance in the response variable.
cleaning_recipe %<>% step_smotenc(HeartDisease)
```

```
cleaning_recipe %<>%
    # converts all nominal predictors into dummy variables.
    step_dummy(all_nominal_predictors()) %>%
    # creates interaction effect between select variables.
    step_interact(
        terms = ~ starts_with("Smoking"):starts_with("Asthma")
    ) %>%
    step_interact(
        terms = ~ starts_with("Smoking"):starts_with("AlcoholDrinking")
    ) %>%
    step_interact(
        terms = ~ starts_with("Race"):starts_with("Sex")
    ) %>%
    step_interact(
        terms = ~ starts_with("AlcoholDrinking"):starts_with("KidneyDisease")
    ) %>%
    step_interact(
        terms = ~ starts_with("BMI"):starts_with("PhysicalActivity")
    ) %>%
    step_interact(
        terms = ~ starts_with("Sex"):starts_with("AlcoholDrinking")
    )
```

```
cleaning_recipe %<>%
    # Filters out highly correlated predictors.
    step_corr(all_predictors()) %>%
    # Filters out near-zero variance predictors.
    step_nzv(all_predictors())
```

Baseline Logistic Regression

```
wkflow_0 <- workflow()

mod_0 <- logistic_reg() %>% set_engine("glm")

wkflow_0 %<>% add_model(mod_0) %>% add_formula(HeartDisease ~ .)
wkflow_0
```

```
## == Workflow ====================================================================
## Preprocessor: Formula
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## HeartDisease ~ .
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```
mod_0_fit <- wkflow_0 %>% fit(data = data_train)
```

```
# Extracts predicted probabilities for the 'Yes' class.
model_0_train_pred <- predict(mod_0_fit,
    new_data = data_train, type = "prob")[2]
model_0_train_pred$obs <- data_train$HeartDisease

# Calculates the optimal cutpoint for class assignment
# based on maximizing the Kappa statistics
model_0_cut <- model_0_train_pred %>%
    cutpointr(x = .pred_Yes, class = obs,
              method = maximize_metric,
              metric = cohens_kappa, pos_class = "Yes"
        )
```

```
# Exacts the model parameters for inference.
coef <- mod_0_fit %>% extract_fit_parsnip() %>% tidy()
print(coef, n = 100)
```

```
## # A tibble: 38 x 5
##    term                          estimate std.error statistic   p.value
##    <chr>                            <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                    -6.20      0.137    -45.4    0
##  2 BMI                             0.00823   0.00136    6.03   1.61e-  9
##  3 SmokingYes                      0.348     0.0171    20.3    1.15e- 91
##  4 AlcoholDrinkingYes             -0.271     0.0403    -6.73   1.70e- 11
##  5 StrokeYes                       1.05      0.0271    38.7    0
##  6 PhysicalHealth                  0.00260   0.00103    2.51   1.19e-  2
##  7 MentalHealth                    0.00502   0.00105    4.77   1.84e-  6
##  8 DiffWalkingYes                  0.216     0.0216     9.96   2.21e- 23
```

```
##  9 SexMale                            0.719     0.0174    41.4    0
## 10 `AgeCategory25-29`                -0.00652   0.153     -0.0426 9.66e-  1
## 11 `AgeCategory30-34`                 0.514     0.132      3.89   9.87e-  5
## 12 `AgeCategory35-39`                 0.619     0.127      4.88   1.05e-  6
## 13 `AgeCategory40-44`                 1.01      0.120      8.43   3.49e- 17
## 14 `AgeCategory45-49`                 1.29      0.116     11.1    8.51e- 29
## 15 `AgeCategory50-54`                 1.74      0.111     15.7    1.82e- 55
## 16 `AgeCategory55-59`                 1.99      0.109     18.2    4.16e- 74
## 17 `AgeCategory60-64`                 2.23      0.109     20.5    1.28e- 93
## 18 `AgeCategory65-69`                 2.46      0.108     22.7    1.84e-114
## 19 `AgeCategory70-74`                 2.76      0.108     25.5    2.84e-143
## 20 `AgeCategory75-79`                 2.96      0.109     27.2    1.29e-162
## 21 `AgeCategory80 or older`           3.22      0.109     29.7    1.34e-193
## 22 RaceAsian                         -0.702     0.104     -6.75   1.44e- 11
## 23 RaceBlack                         -0.389     0.0684    -5.68   1.34e-  8
## 24 RaceHispanic                      -0.263     0.0693    -3.80   1.44e-  4
## 25 RaceOther                         -0.103     0.0756    -1.37   1.72e-  1
## 26 RaceWhite                         -0.116     0.0609    -1.91   5.65e-  2
## 27 `DiabeticNo, borderline diabetes`  0.144     0.0497     2.89   3.86e-  3
## 28 DiabeticYes                        0.464     0.0200    23.2    6.15e-119
## 29 `DiabeticYes (during pregnancy)`  -0.0490    0.133     -0.368  7.13e-  1
## 30 PhysicalActivityYes               -0.00579   0.0191    -0.303  7.62e-  1
## 31 GenHealthFair                      1.51      0.0391    38.5    0
## 32 GenHealthGood                      1.05      0.0352    29.7    3.16e-194
## 33 GenHealthPoor                      1.89      0.0488    38.8    0
## 34 `GenHealthVery good`               0.461     0.0362    12.7    3.13e- 37
## 35 SleepTime                         -0.0281    0.00518   -5.42   5.98e-  8
## 36 AsthmaYes                          0.276     0.0229    12.0    2.35e- 33
## 37 KidneyDiseaseYes                   0.580     0.0291    19.9    2.17e- 88
## 38 SkinCancerYes                      0.116     0.0233     4.98   6.29e-  7
```

```
# Extracts predicted probabilities for the 'Yes' class
# using the fit model on the testing data set.
model_0_test_pred <- predict(mod_0_fit,
    new_data = data_test, type = "prob")[2]
model_0_test_pred$obs <- data_test$HeartDisease

# Assigns class prediction based on the optimal
# cutpoint calculated from the training dataset.
model_0_test_pred %<>%
    mutate(pred.class =
        ifelse(.pred_Yes >= model_0_cut$optimal_cutpoint,
        "Yes", "No")) %>%
    mutate(pred.class = as.factor(pred.class))

# Calculates a confusion matrix based on the
# testing dataset prediction results.
model_0_test_cmat <- model_0_test_pred %>%
    conf_mat(truth = obs, estimate = pred.class)
model_0_cut$optimal_cutpoint
```

```
## [1] 0.2041444
```

```
model_0_test_cmat
```

```
##           Truth
## Prediction    No   Yes
##        No  80232  4303
##        Yes  7579  3825
```

```
summary(model_0_test_cmat)
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.876
##  2 kap                  binary         0.325
##  3 sens                 binary         0.914
##  4 spec                 binary         0.471
##  5 ppv                  binary         0.949
##  6 npv                  binary         0.335
##  7 mcc                  binary         0.331
##  8 j_index              binary         0.384
##  9 bal_accuracy         binary         0.692
## 10 detection_prevalence binary         0.881
## 11 precision            binary         0.949
## 12 recall               binary         0.914
## 13 f_meas               binary         0.931
```

```
# Calculates a bier score for the testing dataset
# predictions.
test_pred_0 <- as.data.frame(model_0_test_pred)
test_pred_0$obs_dummy <- ifelse(model_0_test_pred$obs == "Yes", 1, 0)
model_0_test_bier <- (1 / nrow(test_pred_0)) *
    sum((test_pred_0$.pred_Yes - test_pred_0$obs_dummy)^2)
model_0_test_bier
```

```
## [1] 0.06519922
```

Standard Logistic Regression

```
# Uses a workflow to define the logistic regression model.
# The workflow uses the cleaning recipe defined above as
# well as the glm computation engine to compute the model
# parameters.
wkflow_1 <- workflow()

mod_1 <- logistic_reg() %>% set_engine("glm")

wkflow_1 %<>% add_model(mod_1) %>% add_recipe(cleaning_recipe)
wkflow_1
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 16 Recipe Steps
##
## * step_mutate()
## * step_mutate()
## * step_mutate()
## * step_mutate()
## * step_mutate()
## * step_normalize()
## * step_smotenc()
## * step_dummy()
## * step_interact()
## * step_interact()
## * ...
## * and 6 more steps.
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```
# Calculates the model parameters.
mod_1_fit <- wkflow_1 %>% fit(data = data_train)
```

```
# Exacts the model parameters for inference.
coef <- mod_1_fit %>% extract_fit_parsnip() %>% tidy()
print(coef, n = 100)
```

```
## # A tibble: 38 x 5
##    term                        estimate std.error statistic   p.value
##    <chr>                          <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                -4.03      0.0417    -96.7   0
##  2 BMI                         0.0614    0.00718     8.55  1.26e- 17
##  3 PhysicalHealth              0.0111    0.00767     1.45  1.48e-  1
##  4 MentalHealth               -0.0176    0.00733    -2.41  1.61e-  2
##  5 SleepTime                  -0.0767    0.00409   -18.7   2.50e- 78
##  6 AgeCategory_Cont            0.507     0.0158     32.1   1.82e-225
##  7 GenHealth_ord              -0.250     0.0263     -9.53  1.52e- 21
##  8 Smoking_Yes                 0.387     0.00861    44.9   0
##  9 Stroke_Yes                  0.780     0.0177     44.0   0
## 10 DiffWalking_Yes             0.218     0.0116     18.7   2.67e- 78
## 11 Sex_Male                    0.144     0.0223      6.43  1.24e- 10
## 12 AgeCategory_X50.54          0.740     0.0219     33.8   5.19e-250
## 13 AgeCategory_X55.59          0.941     0.0230     40.9   0
## 14 AgeCategory_X60.64          1.08      0.0255     42.3   0
## 15 AgeCategory_X65.69          1.15      0.0289     39.7   0
## 16 AgeCategory_X70.74          1.33      0.0327     40.6   0
## 17 AgeCategory_X75.79          1.44      0.0371     38.8   0
## 18 AgeCategory_X80.or.older    1.64      0.0410     40.1   0
```

```
## 19 Race_Black                   0.440     0.0285     15.4   1.13e- 53
## 20 Race_Hispanic                0.741     0.0288     25.7   1.31e-145
## 21 Race_White                   1.06      0.0256     41.3   0
## 22 Diabetic_Yes                 0.496     0.0105     47.3   0
## 23 PhysicalActivity_Yes         0.0534    0.00975     5.47  4.42e-  8
## 24 GenHealth_Fair               1.28      0.0635     20.2   3.72e- 91
## 25 GenHealth_Good               0.879     0.0435     20.2   1.18e- 90
## 26 GenHealth_Poor               1.50      0.0841     17.8   9.68e- 71
## 27 GenHealth_Very.good          0.572     0.0143     39.9   0
## 28 Asthma_Yes                   0.0895    0.0170      5.26  1.46e-  7
## 29 KidneyDisease_Yes            0.0878    0.0182      4.82  1.47e-  6
## 30 SkinCancer_Yes              -0.180     0.0121    -14.8   9.32e- 50
## 31 PhysicalHealth_bin_X.0.5.    0.0615    0.0115      5.33  9.62e-  8
## 32 PhysicalHealth_bin_X.25.30.  0.0437    0.0278      1.57  1.16e-  1
## 33 MentalHealth_bin_X.0.5.     -0.148     0.0117    -12.7   6.66e- 37
## 34 MentalHealth_bin_X.25.30.    0.294     0.0286     10.3   8.61e- 25
## 35 SleepTime_bin_Regular        0.138     0.0124     11.1   1.22e- 28
## 36 Smoking_Yes_x_Asthma_Yes    -0.0484    0.0233     -2.08  3.73e-  2
## 37 Race_White_x_Sex_Male        0.677     0.0238     28.4   6.61e-178
## 38 BMI_x_PhysicalActivity_Yes  -0.00620   0.00885    -0.700 4.84e-  1
```

```
# Extracts predicted probabilities for the 'Yes' class.
model_1_train_pred <- predict(mod_1_fit,
    new_data = data_train, type = "prob")[2]
model_1_train_pred$obs <- data_train$HeartDisease

# Calculates the optimal cutpoint for class assignment
# based on maximizing the Kappa statistics
model_1_cut <- model_1_train_pred %>%
    cutpointr(x = .pred_Yes, class = obs,
              method = maximize_metric,
              metric = cohens_kappa, pos_class = "Yes"
        )
```

```
# Extracts predicted probabilities for the 'Yes' class
# using the fit model on the testing data set.
model_1_test_pred <- predict(mod_1_fit,
    new_data = data_test, type = "prob")[2]
model_1_test_pred$obs <- data_test$HeartDisease

# Assigns class prediction based on the optimal
# cutpoint calculated from the training dataset.
model_1_test_pred %<>%
    mutate(pred.class =
        ifelse(.pred_Yes >= model_1_cut$optimal_cutpoint,
        "Yes", "No")) %>%
    mutate(pred.class = as.factor(pred.class))

# Calculates a confusion matrix based on the
# testing dataset prediction results.
model_1_test_cmat <- model_1_test_pred %>%
    conf_mat(truth = obs, estimate = pred.class)
model_1_cut$optimal_cutpoint
```

```
## [1] 0.7465393
```

```
model_1_test_cmat
```

```
##            Truth
## Prediction    No   Yes
##        No  80199  4452
##        Yes  7612  3676
```

```
summary(model_1_test_cmat)
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.874
##  2 kap                  binary         0.311
##  3 sens                 binary         0.913
##  4 spec                 binary         0.452
##  5 ppv                  binary         0.947
##  6 npv                  binary         0.326
##  7 mcc                  binary         0.316
##  8 j_index              binary         0.366
##  9 bal_accuracy         binary         0.683
## 10 detection_prevalence binary         0.882
## 11 precision            binary         0.947
## 12 recall               binary         0.913
## 13 f_meas               binary         0.930
```

```
# Calculates a bier score for the testing dataset
# predictions.
test_pred_1 <- as.data.frame(model_1_test_pred)
test_pred_1$obs_dummy <- ifelse(model_1_test_pred$obs == "Yes", 1, 0)
model_1_test_bier <- (1 / nrow(test_pred_1)) *
    sum((test_pred_1$.pred_Yes - test_pred_1$obs_dummy)^2)
model_1_test_bier
```

```
## [1] 0.1670321
```

```
# Creates a variable importance plot for the logistic
# regression model.
plot_vip <- mod_1_fit %>% extract_fit_parsnip() %>%
    vip(num_features = 10)
```

Penalized Logistic

```
# Defines a workflow for a new model.
wkflow_2 <- workflow()

# Defines the workflow's model to be a penalized
```

```r
# regression model that tune the type and amount of
# penalty using the glmnet algorithm.
mod_2 <- logistic_reg(penalty = tune(), mixture = tune()) %>%
    set_engine("glmnet")

# Define the resampling method to be 10-fold cross-validation.
resamples <- data_train %>% vfold_cv(v = 10)

# Define a grid of tunning parameters to try.
param_grid <- grid_regular(penalty(), mixture(),
                           levels = list(penalty = 10, mixture = 10))

# Adds the defined model and previously defined data cleaning to
# the workflow.
wkflow_2 %<>% add_model(mod_2) %>% add_recipe(cleaning_recipe)
```

```r
# Fits the above defined model.
model_2_fit <- wkflow_2 %>%
               tune_grid(resamples = resamples, grid = param_grid,
                         metrics = metric_set(roc_auc),
                         control = control_grid(verbose = TRUE))
```

```r
# Extracts the optimal penalty.
best_pen <- select_best(model_2_fit)
print(best_pen)
```

```
## # A tibble: 1 x 3
##        penalty mixture .config
##          <dbl>   <dbl> <chr>
## 1 0.0000000001       0 Preprocessor1_Model001
```