Stats Project 2

Theodoro Gasperin Terra Camargo & Tarek Chammaa El Rifai Mashmoushi

2024-05-13

Question 1

Study and describe the predictor variables. Do you see any issues that are relevant for making predictions? Make sure to discuss the dimensionality of the data and the implication on fitting models.

The dataset has 102 columns, which include 101 predictor variables and 1 response variable (LET_IS). This high dimensionality presents several challenges, including increased computational cost and a high risk of overfitting. Consequently, it is crucial to select a model capable of handling many predictors, such as Lasso or Ridge regression. The predictor variables exhibit considerable variability in their data types, with some being continuous (e.g., IBS_NASL) and others discrete (e.g., Sex). The response variable, LET_IS, is binary, indicating whether a patient survived (False) or died (True). Another issue to consider is multi-collinearity. Given the large number of predictors, it is likely that some variables are highly correlated, which can complicate model interpretation and degrade performance. The dataset also contains several instances of missing data, each affecting the same five individuals across different variables. The affected variables and their missing values are: nr03 (5 missing values), zab_leg_02 (5 missing values), FIB_G_POST (5 missing values), post_im (5 missing values), and fibr_ter_02 (5 missing values). To handle these missing values, we used predictive imputation approach that used the mice() function from the mice library. This function fills in the missing values by using the information from the other variables to make the best guess, creating a complete dataset for analysis.

```
#head(MI)
#summary(MI)
dim(MI)
```

```
## [1] 1700  101
```

```
# checking for missing values
sum(is.na(MI))
```

```
## [1] 25
```

```
# Identify columns with missing data
missing_columns <- names(MI)[apply(MI, 2, function(x) any(is.na(x)))]
print(missing_columns)
```

```
## [1] "nr03"        "zab_leg_02"  "FIB_G_POST"  "post_im"     "fibr_ter_02"
```

```
# Identify rows with missing data
rows_with_na <- apply(MI, 1, function(x) any(is.na(x)))

# Get the indices of rows with missing data
rows_with_missing_data <- which(rows_with_na)

# Print the indices of rows with missing data
print(rows_with_missing_data)
```

```
## [1]  220  428 1147 1157 1572
```

```
# Print the rows with missing data
#print(MI[rows_with_missing_data, ])
```

Question 2

Fit and compare an appropriate unconstrained linear model, as well as lasso and ridge regression models. Discuss what you find. What is an appropriate base-level of performance to compare your models to?

We fitted and compared three models: an unconstrained linear model, a Lasso regression model, and a Ridge regression model, with a focus on identifying a model with high sensitivity, which measures the proportion of actual positives correctly identified by the model and thus, allowing the identification of the true death cases in our data.

The unconstrained linear model achieved an accuracy of 0.81, a specificity of 0.89, and a sensitivity of 0.43. While this model performed reasonably well in terms of overall accuracy and specificity, its low sensitivity indicates a significant shortcoming in correctly identifying true positives. This makes the model less suitable for scenarios where correctly identifying true deaths (positives) is crucial.

```
unconstrained_model <- glm(LET_IS ~ ., data = MI_train, family = binomial)
#summary(unconstrained_model)

# Predictions
unconstrained_pred <- predict(unconstrained_model, newdata = MI_test, type = "response")
unconstrained_pred_class <- ifelse(unconstrained_pred > 0.5, TRUE, FALSE)

# Confusion matrix
unconstrained_model.table <- table(MI_test$LET_IS, unconstrained_pred_class)
confusionMatrix(rotate_table_180(unconstrained_model.table))
```

```
## Confusion Matrix and Statistics
##
##        unconstrained_pred_class
##         TRUE FALSE
##   TRUE    25    29
##   FALSE   33   252
##
##                Accuracy : 0.8171
##                  95% CI : (0.7718, 0.8568)
##     No Information Rate : 0.8289
##     P-Value [Acc > NIR] : 0.7448
##
##                   Kappa : 0.3371
##
##  Mcnemar's Test P-Value : 0.7032
##
##             Sensitivity : 0.43103
##             Specificity : 0.89680
##          Pos Pred Value : 0.46296
##          Neg Pred Value : 0.88421
##              Prevalence : 0.17109
##          Detection Rate : 0.07375
##    Detection Prevalence : 0.15929
##       Balanced Accuracy : 0.66392
```

```
##
##        'Positive' Class : TRUE
##
```

The Lasso regression model, which applies L1 regularization, showed an improved overall accuracy of 0.885 and a slightly higher specificity of 0.90. More importantly, it achieved a significantly higher sensitivity of 0.7273 compared to the unconstrained linear model. This indicates that Lasso regression was more effective in correctly identifying true positives cases, which is aligned with our objective of fiding a high sensitivity model.

```
# Lasso Model with Cross-Validation of 5 folds on the Training data
lasso_model <- cv.glmnet(as.matrix(select(MI_train, -LET_IS)), MI_train$LET_IS, type.measure = "class",

# Best lambda
#plot(lasso_model)

# Best lambda values
lasso_model$lambda.min
```

```
## [1] 0.006482767
```

```
lasso_model$lambda.1se
```

```
## [1] 0.01643618
```

```
# Coefficients of variables at lambda min and lambda 1se
#coef(lasso_model, s = "lambda.min")
#coef(lasso_model, s = "lambda.1se")

# Predictions
lasso_pred.min <- predict(lasso_model, s= lasso_model$lambda.min, newx = as.matrix(select(MI_test, -LET
lasso_pred.1se <- predict(lasso_model, s= lasso_model$lambda.1se, newx = as.matrix(select(MI_test, -LET

# Confusion matrix
lasso_model.table <- table(MI_test$LET_IS, lasso_pred.min)
#table(MI_test$LET_IS, lasso_pred.min)
confusionMatrix(rotate_table_180(lasso_model.table))
```

```
## Confusion Matrix and Statistics
##
##        lasso_pred.min
##         TRUE FALSE
##   TRUE    24    30
##   FALSE    9   276
##
##                Accuracy : 0.885
##                  95% CI : (0.8461, 0.9169)
##     No Information Rate : 0.9027
##     P-Value [Acc > NIR] : 0.881493
##
##                   Kappa : 0.4901
##
##  Mcnemar's Test P-Value : 0.001362
##
##             Sensitivity : 0.72727
##             Specificity : 0.90196
##          Pos Pred Value : 0.44444
```

```
##          Neg Pred Value : 0.96842
##              Prevalence : 0.09735
##          Detection Rate : 0.07080
##    Detection Prevalence : 0.15929
##       Balanced Accuracy : 0.81462
##
##        'Positive' Class : TRUE
##
```

The Ridge regression model, using L2 regularization, achieved an accuracy of 0.8791, maintaining a specificity of 0.89, and a sensitivity of 0.7241. While the Ridge model also improved sensitivity compared to the unconstrained linear model, it performed slightly worse than the Lasso model in terms of sensitivity.

```
# Ridge Model with Cross-Validation of 5 folds on the Training data
ridge_model <- cv.glmnet(as.matrix(select(MI_train, -LET_IS)), MI_train$LET_IS, type.measure = "class",

# Best lambda
#plot(ridge_model)

# Best lambda values
ridge_model$lambda.min
```

```
## [1] 0.05638381
```

```
ridge_model$lambda.1se
```

```
## [1] 0.3977766
```

```
# Coefficients of variables at lambda min and lambda 1se
lasso_coef.min <- coef(lasso_model, s = "lambda.min")
lasso_coef.1se <- coef(lasso_model, s = "lambda.1se")

# Predictions
ridge_pred.min <- predict(ridge_model, s= ridge_model$lambda.min, newx = as.matrix(select(MI_test, -LET
ridge_pred.1se <- predict(ridge_model, s= ridge_model$lambda.1se, newx = as.matrix(select(MI_test, -LET

# Confusion matrix
ridge_model.table <- table(MI_test$LET_IS, ridge_pred.min)
#table(MI_test$LET_IS, ridge_pred.1se)
confusionMatrix(rotate_table_180(ridge_model.table))
```

```
## Confusion Matrix and Statistics
##
##        ridge_pred.min
##         TRUE FALSE
##   TRUE    21    33
##   FALSE    8   277
##
##                Accuracy : 0.8791
##                  95% CI : (0.8395, 0.9118)
##     No Information Rate : 0.9145
##     P-Value [Acc > NIR] : 0.9897731
##
##                   Kappa : 0.4442
##
##  Mcnemar's Test P-Value : 0.0001781
##
```

```
##             Sensitivity : 0.72414
##             Specificity : 0.89355
##          Pos Pred Value : 0.38889
##          Neg Pred Value : 0.97193
##              Prevalence : 0.08555
##          Detection Rate : 0.06195
##    Detection Prevalence : 0.15929
##       Balanced Accuracy : 0.80884
##
##        'Positive' Class : TRUE
##
```

Given our goal of achieving high sensitivity, the Lasso regression model stands out as the most suitable choice. It provides a good balance between high accuracy and specificity while significantly improving sensitivity. Thus, while both Lasso and Ridge regression techniques enhanced sensitivity compared to the unconstrained linear model, Lasso regression was particularly effective, making it the best model for our needs. The appropriate base-level of performance to compare our models to would be the unconstrained linear model, which serves as a benchmark for evaluating improvements in sensitivity and overall performance.

Question 3

Among your top predictors, do you see evidence of non-linear effects? How could you accommodate non-linear effects and still use a regularized regression approach? Does adding non-linear effects improve your model?

We identified the top predictors using Lasso regression and observed evidence of non-linear effects in some of these predictors. To assess the relevance of these non-linear effects, we fitted a Generalized Additive Model (GAM) which allows for non-linear relationships by incorporating smooth terms for continuous predictors. However, when comparing the performance of the GAM model to the standard Lasso regression model, which does not account for non-linear effects, we found that the inclusion of non-linear terms did not improve the model's performance at the threshold of 0.5. In fact, it decreased the sensitivity from 72% (lasso) to 64% (GAM).

```
# GAM model at threshold 0.5
gam_model <- gam(formula.adjusted, data = MI_train, family = "binomial")

#summary(gam_model)
#plot(gam_model, se = TRUE, col = "purple")

# Predictions
gam_pred <- predict(gam_model, MI_test, type = "response")

threshold <- 0.5
gam_pred_binary <- ifelse(gam_pred > threshold, TRUE, FALSE)

# Confusion matrix
gam_model.table <- table(MI_test$LET_IS, gam_pred_binary)
confusionMatrix(rotate_table_180(gam_model.table))
```

```
## Confusion Matrix and Statistics
##
##        gam_pred_binary
##         TRUE FALSE
##   TRUE    32    22
##   FALSE   18   267
##
```

```
##                Accuracy : 0.882
##                  95% CI : (0.8428, 0.9144)
##     No Information Rate : 0.8525
##     P-Value [Acc > NIR] : 0.06972
##
##                   Kappa : 0.5458
##
##  Mcnemar's Test P-Value : 0.63526
##
##             Sensitivity : 0.6400
##             Specificity : 0.9239
##          Pos Pred Value : 0.5926
##          Neg Pred Value : 0.9368
##              Prevalence : 0.1475
##          Detection Rate : 0.0944
##    Detection Prevalence : 0.1593
##       Balanced Accuracy : 0.7819
##
##        'Positive' Class : TRUE
##
```

However, with the threshold of 0.8, the GAM model performed better overall. It achieved an accuracy of 89%, a sensitivity of 85%, and a specificity of 90%. This suggests that the GAM model correctly captured the non linear effects in the data and performed overall better than the lasso and ridge models.

```
# GAM model at threshold 0.8
threshold <- 0.8
gam_pred_binary <- ifelse(gam_pred > threshold, TRUE, FALSE)

# Confusion matrix
gam_model.table <- table(MI_test$LET_IS, gam_pred_binary)
confusionMatrix(rotate_table_180(gam_model.table))
```

```
## Confusion Matrix and Statistics
##
##        gam_pred_binary
##         TRUE FALSE
##   TRUE    23    31
##   FALSE    4   281
##
##                Accuracy : 0.8968
##                  95% CI : (0.8593, 0.927)
##     No Information Rate : 0.9204
##     P-Value [Acc > NIR] : 0.9515
##
##                   Kappa : 0.5166
##
##  Mcnemar's Test P-Value : 1.109e-05
##
##             Sensitivity : 0.85185
##             Specificity : 0.90064
##          Pos Pred Value : 0.42593
##          Neg Pred Value : 0.98596
##              Prevalence : 0.07965
##          Detection Rate : 0.06785
```

```
##    Detection Prevalence : 0.15929
##       Balanced Accuracy : 0.87625
##
##        'Positive' Class : TRUE
##
```

```
# Re-seting the threshold back to 0.5
threshold <- 0.5
gam_pred_binary <- ifelse(gam_pred > threshold, TRUE, FALSE)
```

Question 4

Fit an appropriate Random Forest model. Report a comparison of performance to your linear model and explain any differences in performance. Do you see an important difference in how variables are used for predictions?

When comparing Random Forests and Linear Models, there are significant differences in how variables are used for predictions. Linear models assume a linear relationship between the predictors and the response. This means they only capture linear interactions. In contrast, Random Forests allow us to build multiple decision trees to capture complex, non-linear relationships and interactions between variables. Variable importance in Random Forests is determined by the contribution of each variable to the reduction in impurity (Gini impurity) across all trees. Variables frequently used to split nodes and providing better splits are considered more important. Random Forests are more flexible than Lasso, Ridge, and GLMs, particularly when it comes to capturing non-linear relationships and interactions between predictors, whereas linear models offer more straightforward interpretations and are easier to understand.

```
set.seed(33)

# Random Forest model
forest_model <- randomForest(LET_IS ~ ., mtry= 18, data = MI_train, ntree = 500)

# Predictions
forest_pred <- predict(forest_model, MI_test, type = "class")
forest_model.table <- table(MI_test$LET_IS, forest_pred)

# Confusion matrix
confusionMatrix(rotate_table_180(forest_model.table))
```

```
## Confusion Matrix and Statistics
##
##        forest_pred
##         TRUE FALSE
##   TRUE    19    35
##   FALSE    1   284
##
##                Accuracy : 0.8938
##                  95% CI : (0.856, 0.9245)
##     No Information Rate : 0.941
##     P-Value [Acc > NIR] : 0.9997
##
##                   Kappa : 0.4677
##
##  Mcnemar's Test P-Value : 3.798e-08
##
##             Sensitivity : 0.95000
##             Specificity : 0.89028
```

```
##          Pos Pred Value : 0.35185
##          Neg Pred Value : 0.99649
##              Prevalence : 0.05900
##          Detection Rate : 0.05605
##    Detection Prevalence : 0.15929
##       Balanced Accuracy : 0.92014
##
##        'Positive' Class : TRUE
##
```

The ROC curve comparison reveals significant differences in the performance of the models based on the Area Under the Curve (AUC) metric. The Generalized Additive Model (GAM) demonstrates the highest AUC of 0.765, indicating superior discriminative ability across various thresholds when compared to the other models. The Lasso regression follows closely with an AUC of 0.706, suggesting that it also performs well in differentiating between classes. The Ridge regression, with an AUC of 0.680, performs moderately, surpassing the Unconstrained linear model and the Random Forest, both of which have an AUC of 0.674. This indicates that while regularization techniques like Lasso and Ridge improve model performance over the basic linear model, the GAM's incorporation of non-linear effects provides the most substantial enhancement in predictive capability. The Random Forest model's AUC performance being on par with the Unconstrained model suggests that it may not be capturing complex relationships in the data as effectively as expected. However, the Random forest model showed the highest sensitivity among all predictors (95%), correctly identified 19 out of 20 cases which it predicted to be positive, and 89% accuracy in par with the GAM model at threshold 0.8. Overall, these findings highlight the importance of considering non-linear effects and regularization techniques to enhance model accuracy and reliability in predictive tasks.