PHP2650: Statistical Learning and Big Data

Assignment 4 - Optimization and Prediction

Antonella Basso

April 8, 2022

Problem 1: Cross-Validation

Another researcher has emailed you with an alternative method to prune a regression tree, which they believe is more computationally efficient than what we discussed in class. The proposed procedure is outlined below. Respond to the researcher about why their proposal is a flawed use of cross-validation.

a. Iteratively build a regression tree on the full data set yielding a sequence of trees $T_1, T_2, ..., T_l$.
b. Split the data into k random folds. For each tree T_i, find the average error of T_i across the k folds.
c. Return the tree with the minimum average cross-validated error.

Solution

This proposal is a flawed use of cross-validation, primarily in that it fits each tree on the whole data. Given that the point of cross-validation is to evaluate a model's performance on unseen data, using all the data, rather than subsets of the data, to construct it prevents us from being able to make appropriate judgements regarding its skill. Moreover, the way in which the data splits (k folds) are used in this proposed method does not follow a traditional cross-validation technique. That is, this method suggests using the splits separately on each tree to obtain a corresponding average error score, which essentially amounts to obtaining a mean score for each model that has already been exposed to the data used to compute it (average error). On the other hand, cross-validation involves splitting the (randomly shuffled) data into k folds, such that each fold is used (once) as a test set for a tree/model trained on the remaining $k-1$ folds. In turn, yielding k error scores for each tree with which to evaluate its overall skill. This way, regression trees are directly judged based on their performance on a variety of unseen data, rather than on their expected performance on a variety of observed data. Pruning a regression tree accordingly thus allows us to choose the model that produces the lowest error rate based on realizations of "unseen" data we have available. Conversely, under the proposed method, each error estimate produced from the k^{th} fold will be fundamentally biased, resulting in a biased average error for each tree. So, the tree which yields the minimum average error score in this way may not be the optimal tree (realistically). For this reason, we can infer that the error scores produced under a true cross-validation paradigm capture a much wider range of a model's genuine skill, while the suggested approach would, almost surely, produce scores that overestimate its performance and are perhaps too similar to make sensible pruning decisions, resulting in an overall flawed approach to pruning regression trees (let alone cross-validation).

Problem 2: First-Order Optimization Methods

Consider the Adam algorithm introduced in class that uses momentum and an adaptive learning rate. First, define the algorithm in text. Then, implement the algorithm and compare it to vanilla gradient descent for two settings (1) a non-convex function (this may be on one variable) and (2) fitting a multiple linear regression model (at least two variables and an intercept). Show how the choice of parameters changes the behavior. Comment on your results. You may use the code from class to get started.

Solution

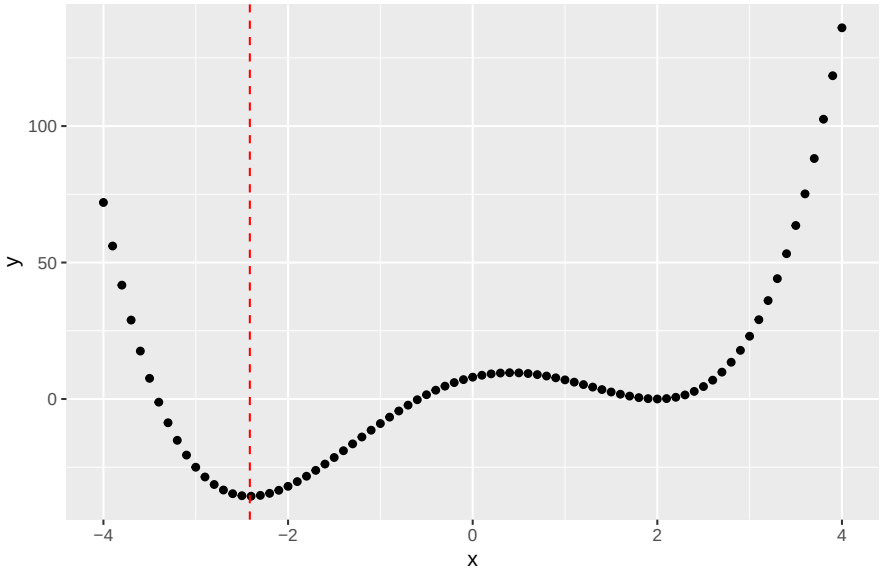
The Adam optimization algorithm, short for Adaptive Moment Estimation, can be viewed as a combination of two other stochastic gradient descent methods, namely, the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). By combining the benefits of both of these methods, it is better able to "smooth" out the steps of gradient descent, and hence, follow a quicker and less noisy path to convergence. That is, its use of adaptive learning rates (or "per-parameter" learning rates) and momentum, makes Adam well suited for handling sparse gradients that result from large noisy data, thus, making it a desired approach to problems in natural language processing and image recognition for example. Specifically, the algorithm uses continuously updated estimates of the gradient's first and second moments (the mean (m_t) and uncentered variance (v_t), respectively), called exponential moving averages, to adapt a parameter's learning rate over time (a range of steps/iterations towards some optimal solution). Moreover, in addition to controlling the exponential decay rates of these moving averages through user-defined hyper-parameters, $\beta_1, \beta_2 \in [0, 1)$, Adam corrects the existing bias within the gradient moment estimates, using $\hat{m}_t$ and $\hat{v}_t$ (rather than m_t and v_t) to determine the effective step (size) to be taken at time point t in some parameter space.

To compare the Adam optimizer to the more basic/vanilla gradient descent approach, we evaluate each algorithm's performance on the following settings:

1. Univariate Non-Convex Function: $f(x) = (x^2 - 4x + 4)(x^2 + 4x + 2)$
2. Linear Regression Models:
    a. Univariate (intercept): $f(x) = b_1 + mx$
    b. Multivariate (intercept and 2 slopes): $f(x, z) = b_1 + b_2x + b_3z$

We begin by defining each setting's objective functions and optimal solutions. Note that an optimal solution, is each algorithm's target and is approximated by optimizing/minimizing each setting's objective (or loss/cost) function. Specifically, the optimal solution (in this case) is lowest point of an objective function such that its derivative (or gradient) at that point is 0. Given that the function in the first setting is indeed the function we are trying to minimize (in being a function of the parameter of interest, x), this is our first objective function. Setting its derivative equal to 0 and solving for x, we find that the optimal solution for the first setting is $x =$ at e find that it has an optimal solution at $x = -1 - \sqrt{2} \approx -2.41421356237$. Figure 1 below provides a visual for the univariate non-convex objective function, with its optimal solution marked by the (vertical) red line (at $x = -1 - \sqrt{2}$).

Figure 1: Univariate Non−Convex Objective Function (1)



Given that the second setting involves a linear regression model, the objective function is not as straightforward. Particularly, since real data are never truly perfectly linear, it is practically impossible to fit a model that lacks any residual error. For this reason, the goal of linear regression is to fit a line that minimizes the sum of squared residuals (SSR/RSS) to provide the best approximation to the data. In two dimensions, this line is defined by an intercept and a slope. And, in the event that we want to find only the intercept which minimizes the total SSR (as in 2a), we can think of our objective function as the SSR being a function of intercept coefficients (such that the point at which its derivative equals 0 is the point/intercept for which the lowest SSR can be obtained). With this in mind, we proceed by simulating some linearly correlated data (30 points) and fitting a linear model to obtain the corresponding regression coefficients, and hence, our optimal solution for 2a; $b_1 = 1.3103$. With optimal $m = 0.9052$ (slope of x, which we consider fixed), we define our objective function for the univariate linear regression setting as follows:

$$SSR = \sum_{i=1}^{30} (y_i - f(x_i))^2$$

where y_i is the i^{th} observed value and $f(x_i)$ is the i^{th} predicted value given x, m, and unknown b_1, making this an implicit function of the intercept, b_1. Figures 2.1 and 2.2 below give the simulated data and corresponding fitted regression line (of intercept $b_1 = 1.3103$ and slope $m = 0.9052$), as well as the objective function marked with the optimal solution for the intercept (at $x = 1.3103$).

Figure 2.1: Univariate Linear Regression Simulated Data (2a)

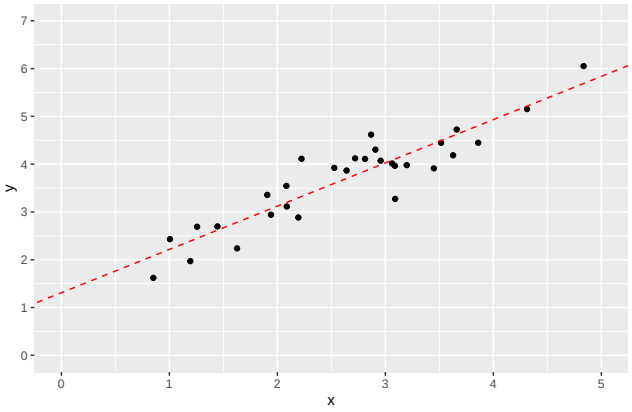
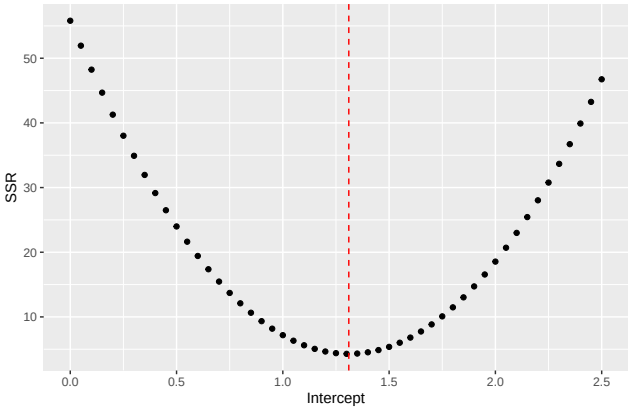


Figure 2.2: Univariate Linear Regression Objective Function (2a)



We can extend the same thinking to the multivariate case (2b). Here however, our objective function is a function of $\hat{b} = [b_1, b_2, b_3]^T$ (implicitly). That is, an intercept (b_1), a slope coefficient for the x dimension (b_2), and a slope coefficient for the z dimension (b_3). We may think of this as a three part problem in which we are trying to minimize three distinct objective functions, namely, one with respect to each b coefficient (wherein we treat the other two as constants). Simulating correlated multivariate data (30 points), and fitting a multiple linear regression model, we obtain our optimal solution; $\hat{b} = [0.02969, 0.46741, 0.49582]^T$. Figures 3.1 and 3.2 below show the simulated data (graphed in terms of x and $y/f(x)$ and colored according to values of z) with the corresponding (partial) regression line, and each (partial) objective function marked with the corresponding optimal solutions, respectively.

Figure 3.1: Multivariate Linear Regression Simulated Data (2b)

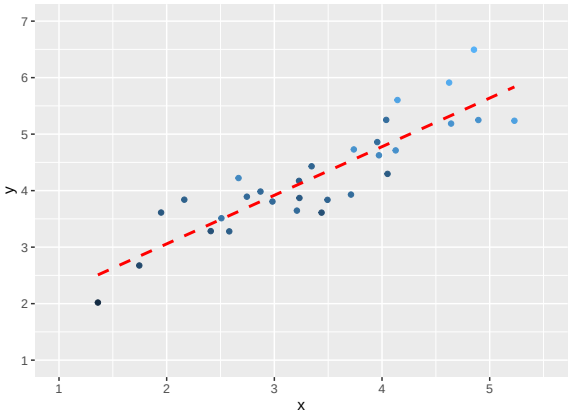
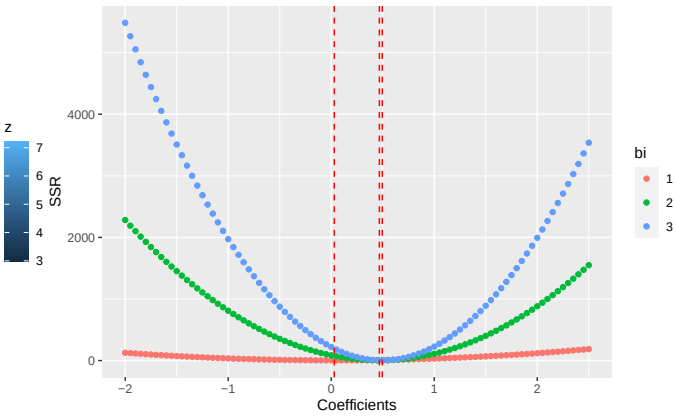


Figure 3.2: Multivariate Linear Regression Objective Function(s) (2b)



A total of 4 algorithms were created. Namely, vanilla gradient descent and Adam algorithms for handling univariate cases (settings 1 and 2a) and multivariate cases (setting 2b):

Gradient Descent Algorithm 1: Univariate

```r
gradient_desc_1 <- function(f_grad, x_init=0){
  # parameters
  max_iters <- 50 # stopping criteria 1
  learning_rate <- 0.01
  # starting conditions
  x <- rep(0, max_iters)
  x[1] <- x_init
  # descent
  for (i in 2:max_iters){
    x[i] <- x[i-1]-learning_rate*f_grad(x[i-1]) # updated x
    if (abs(x[i]-x[i-1]) <= 0.001){break} # stopping criteria 2
  }
  return(x[1:i])
}
```

Gradient Descent Algorithm 2: Multivariate

```r
gradient_desc_2 <- function(f_grad, b1_init=0, b2_init=0, b3_init=0){
  # parameters
  max_iters <- 50 # stopping criteria 1
  learning_rate <- 0.0001
  # starting conditions
  b1 <- rep(0, max_iters)
  b2 <- rep(0, max_iters)
  b3 <- rep(0, max_iters)
  b1[1] <- b1_init # initial intercept
  b2[1] <- b2_init # initial slope of x
  b3[1] <- b3_init # initial slope of z
  # descent
  for (i in 2:max_iters){
    b1[i] <- b1[i-1]-learning_rate*f_grad(b1[i-1], b2[i-1], b3[i-1])[1] # updated b1
    b2[i] <- b2[i-1]-learning_rate*f_grad(b1[i-1], b2[i-1], b3[i-1])[2] # updated b2
```

```
    b3[i] <- b3[i-1]-learning_rate*f_grad(b1[i-1], b2[i-1], b3[i-1])[3] # updated b3
    if (abs(b1[i]-b1[i-1]) <= 0.001 &
        abs(b2[i]-b2[i-1]) <= 0.001 &
        abs(b3[i]-b3[i-1]) <= 0.001){break} # stopping criteria 2
  }
  intercept <- b1[1:i]
  slope_x <- b2[1:i]
  slope_z <- b3[1:i]
  return(cbind(intercept, slope_x, slope_z)) # stopping criteria 1
}
```

Adam Algorithm 1: Univariate

```
adam_1 <- function(f_grad, x_init=0){
  # parameters
  max_iters <- 50 # stopping criteria 1
  learning_rate <- 0.1 # 0.001 (default)
  beta1 <- 0.09 # 0.9 (default)
  beta2 <- 0.6 # 0.999 (default)
  epsilon <- 1e-8
  # starting conditions
  x <- rep(0, max_iters)
  x[1] <- x_init
  # initializing first and second moment vectors
    m <- rep(0, max_iters)
    v <- rep(0, max_iters)
  # descent
  for (i in 2:max_iters){
    m[i] <- (beta1*m[i-1])+((1-beta1)*f_grad(x[i-1])) # updated first moment
        v[i] <- (beta2*v[i-1])+((1-beta2)*(f_grad(x[i-1])**2)) # updated second moment
        m_hat <- m[i]/(1-(beta1**(i))) # bias-corrected first moment
        v_hat <- v[i]/(1-(beta2**(i))) # bias-corrected second moment
    x[i] <- x[i-1]-(learning_rate*(m_hat/(sqrt(v_hat)+epsilon))) # updated x
    if (abs(x[i]-x[i-1]) <= 0.001){break} # stopping criteria 2
  }
  return(x[1:i])
}
```

Adam Algorithm 2: Multivariate

```
adam_2 <- function(f_grad, b1_init=0, b2_init=0, b3_init=0){
  # parameters
  max_iters <- 50 # stopping criteria 1
  learning_rate <- 0.01 # 0.001 (default)
  beta1 <- 0.09 # 0.9 (default)
  beta2 <- 0.5 # 0.999 (default)
  epsilon <- 1e-8
  # starting conditions
  b1 <- rep(0, max_iters)
  b2 <- rep(0, max_iters)
  b3 <- rep(0, max_iters)
  b1[1] <- b1_init # initial intercept
```

```
  b2[1] <- b2_init # initial slope of x
  b3[1] <- b3_init # initial slope of z
  # initializing first & second moment and bias-corrected moment matrices
    m <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
    v <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
    m_hat <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
    v_hat <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
  # descent
  for (i in 2:max_iters){
    for (j in 1:3){
      # updating first & second moment and bias-corrected moment matrices
      m[i,j] <- (beta1*m[i-1])+((1-beta1)*f_grad(b1[i-1], b2[i-1], b3[i-1])[j])
          v[i,j] <- (beta2*v[i-1])+((1-beta2)*(f_grad(b1[i-1], b2[i-1], b3[i-1])[j]**2))
        m_hat[i,j] <- m[i,j]/(1-(beta1**(i)))
        v_hat[i,j] <- v[i,j]/(1-(beta2**(i)))
        if (j==1) {
            b1[i] <- b1[i-1]-(learning_rate*(m_hat[i,j]/(sqrt(v_hat[i,j])+epsilon))) # updated b1
          } else if (j==2) {
            b2[i] <- b1[i-1]-(learning_rate*(m_hat[i,j]/(sqrt(v_hat[i,j])+epsilon))) # updated b2
          } else if (j==3) {
           b3[i] <- b1[i-1]-(learning_rate*(m_hat[i,j]/(sqrt(v_hat[i,j])+epsilon))) # updated b3
          }
    }
  if (abs(b1[i]-b1[i-1]) <= 0.001 &
        abs(b2[i]-b2[i-1]) <= 0.001 &
        abs(b3[i]-b3[i-1]) <= 0.001){break} # stopping criteria 2
  }
  intercept <- b1[1:i]
  slope_x <- b2[1:i]
  slope_z <- b3[1:i]
  return(cbind(intercept, slope_x, slope_z))
}
```

Noticeably, for all algorithms, in addition to setting all initial values to 0, we set the following stopping criterion:

1. Maximum number of steps/iterations: 50
2. Minimum step size: 0.001

Additionally, we see that the learning rates differ for all 4 algorithms. In the case of gradient descent for example, the learning rates which demonstrated the fastest paths to convergence were 0.01 and 0.0001 for the univariate and multivariate settings, respectively. In particular, we saw that using a larger learning rate in the latter case (such as 0.01), resulted in the algorithm's failure to converge to any particular solution (let alone an optimal one). That is, within the 50 iterations, we observed the algorithm's output oscillating between very large positive and negative values. We can assume that this is due to the fact that increasing the learning rate brings the algorithm to take larger steps at each iteration. And in such cases, these steps can become so large that the algorithm misses the optimal solution entirely (altogether failing to converge or reaching the optimal solution only after numerous iterations). This shows how sensitive gradient descent algorithms (including Adam) are to changes in the learning rate parameter.

This kind of parameter sensitivity however, extends beyond just learning rate, as is the case with the Adam algorithm. Specifically, not only does Adam's output change dramatically with small changes in the learning rate parameter (which, for best results, had to be set to 0.1 and 0.01 for the univariate and multivariate cases, respectively), but it does so for changes in the beta (hyper) parameters as well (namely, the moving averages' exponential decay rates, β_1 and β_2). As seen above, contrary to suggestions made by the algorithm's authors (and default settings in existing Adam packages), the best β_1 value (exponential decay rate for the mean/first moment) for our purposes proved to be 0.09 (very small) in both the univariate and multivariate adaptations of the Adam algorithm. On the other hand, the optimal exponential decay rates for the unsentered variances/second moments, β_2, in each of the algorithms were 0.6 and 0.5 (in between small and large) for the first and second Adam algorithms, respectively.

After implementing the algorithms on all three objective functions (or their gradients, to be exact) under such parameter conditions, we obtain the trajectories shown below. Percisely, Figures 4.1 and 4.2 give the steps taken by the vanilla gradient descent (in blue) and the Adam (in red) algorithms to obtain an optimal solution for the first setting (univariate non-convex function). Similarly, Figures 5.1 and 5.2 give each algorithm's path to convergence for the univariate linear regression model setting, 2a (intercept, b_1). And likewise, for the multivariate linear regression model setting, 2b (b_1, b_2, b_3), Figures 6.1a-c and 6.2a-c display each trajectory taken in all three dimensions by the vanilla gradient descent and Adam algorithms, respectively.

Figure 4.1: GD Algorithm for Univariate Non−Convex Function

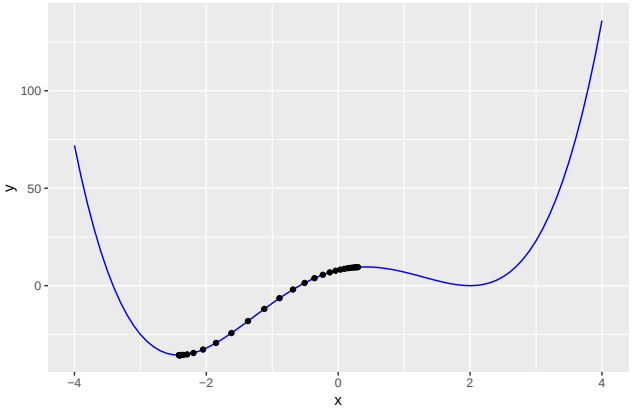


Figure 4.2: ADAM Algorithm for Univariate Non−Convex Function

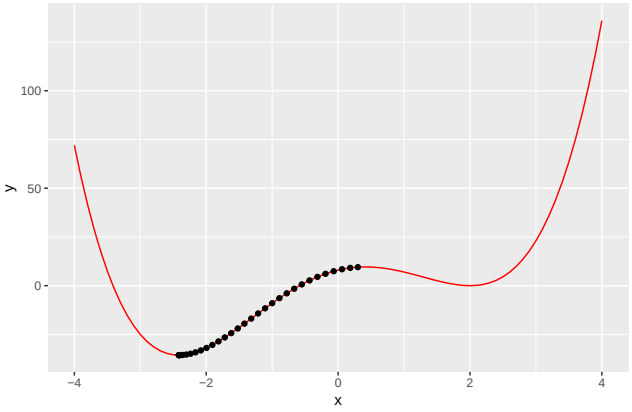


Figure 5.1: GD Algorithm for Univariate Linear Regression (intercept)

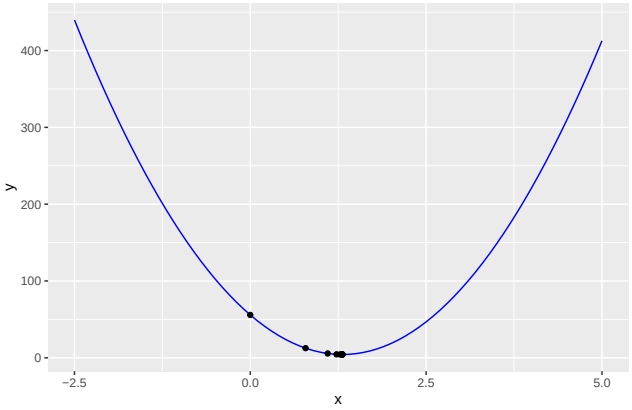
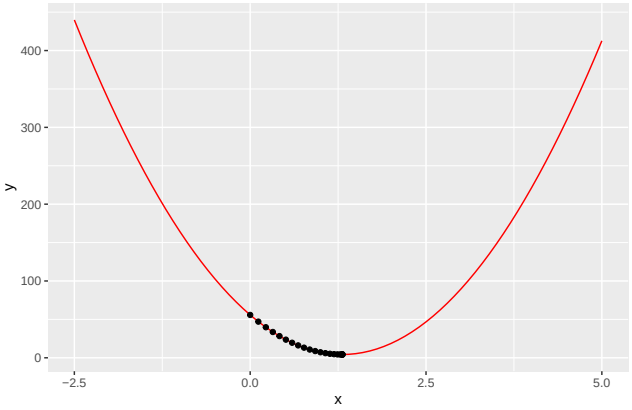
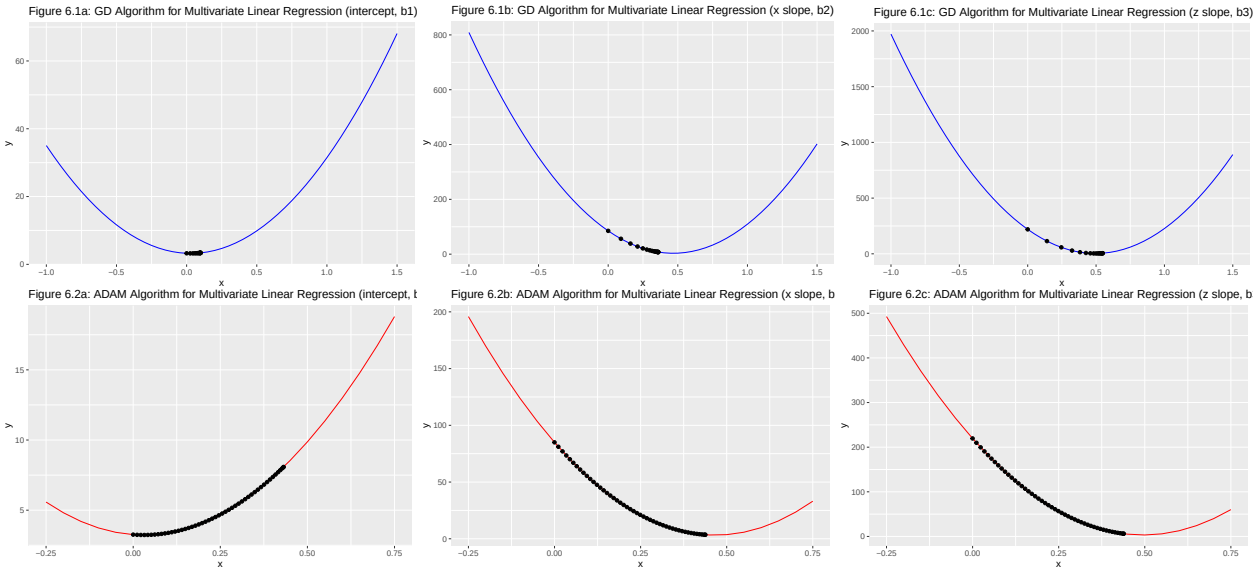


Figure 5.2: ADAM Algorithm for Univariate Linear Regression (intercept)





To better evaluate and compare the algorithms' performances on each of the settings above, we record for each, the exact solution found by the algorithm along with its absolute error/deviation from the true optimal solution, the number of iterations/steps taken by the algorithm to arrive at this solution, and the overall execution time for the algorithm. Particularly, we can see in Tables 1-5 below that, in the univariate cases, both algorithms had approximately 0 execution time, although it becomes slightly larger for Adam in the multivariate case. Moreover, aside from its use on the univariate non-convex function (first setting), the gradient descent algorithm arrived at an optimal solution sooner (or in much fewer steps) than Adam, which can also be noticed in the figures above. Lastly, with the exception of the multivariate case (2b) where it fell short (estimating coefficient values too similar to each other) the Adam algorithm did more precisely approximate the true optimal solution in both univariate settings.

Table 1: Univariate Non-Convex Function (1)

Algorithm	Steps	Solution	Absolute Error	Execution Time
Gradient Descent	30	-2.41365	0.00056	0
ADAM	32	-2.41395	0.00026	0

Table 2: Univariate Linear Regression (2a)

Algorithm	Steps	Solution	Absolute Error	Execution Time
Gradient Descent	10	1.31004	0.000264	0.001
ADAM	22	1.31031	0.000008	0

Table 3: Multivariate Linear Regression (2b) - b_1

Algorithm	Steps	Solution	Absolute Error	Execution Time
Gradient Descent	19	0.09902	0.06933	0.001
ADAM	49	0.43172	0.40203	0.005

Table 4: Multivariate Linear Regression (2b) - b_2

Algorithm	Steps	Solution	Absolute Error	Execution Time
Gradient Descent	19	0.36008	0.10733	0.001
ADAM	49	0.43762	0.02979	0.005

Table 5: Multivariate Linear Regression (2b) - b_3

Algorithm	Steps	Solution	Absolute Error	Execution Time
Gradient Descent	19	0.54877	0.05295	0.001
ADAM	49	0.43829	0.05753	0.005

With this in mind, it is clear that the vanilla gradient decent algorithm had overall better performance compared to the Adam algorithm, especially with regards to speed and linear regression. This could, in part, be due to the fact that Adam is better suited for problems involving sparse gradients and is hence, likely to perform better in non-convex settings. Additionally, we must consider the possibility that errors were made when coding the algorithm or in our overall approach to the problem itself. For instance, having used only 30 data points in the linear regression examples as well as not having sampled according to stochastic methods (having used the whole dataset) could've had considerable effects on the algorithm's estimates and overall performance. Not having chosen the correct/best (hyper) parameter values for Adam, could have also contributed to its inferior performance. However, it is without question that, despite its noticeable precision, Adam's sensitivity to its (hyper) parameters and dependence on momentum, may ultimately render it impractical under more simple settings such as those considered here.

Problem 3: Application to Diabetes Data

The data `diabetic_data.csv` on Canvas used comes from the paper "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records". It contains diabetic inpatient encounter information extracted from electronic health records of 54 hospitals across the United States from 1999 to 2008. Patients are potentially seen multiple times and you should take the first known encounter with each patient to preserve independence. Variable descriptions are in Table 1 in the paper.

Build and compare a logistic regression model and classification tree model to predict hospital readmission within 30 days. You should compare these models in terms of how they use the available variables to make predictions as well as evaluating their performance. You may want to consider some feature engineering prior to fitting your models.

Solution

Having manipulated and condensed the data according to the primary objective and information provided in the paper (reference code for exact data manipulations), we end up with the following variables (main effects) and their corresponding quantity of observations:

```
##
##                                 Overall
##   n                             69977
##   gender = Male (%)             32740 (46.8)
##   time_in_hospital (mean (SD))   4.27 (2.93)
##   num_lab_procedures (mean (SD)) 42.88 (19.89)
##   num_procedures (mean (SD))     1.43 (1.76)
##   num_medications (mean (SD))   15.67 (8.29)
```

```
##   number_outpatient (mean (SD))   0.28 (1.06)
##   number_emergency (mean (SD))    0.10 (0.51)
##   number_inpatient (mean (SD))    0.18 (0.60)
##   number_diagnoses (mean (SD))    7.22 (2.00)
##   diabetesMed = Yes (%)          53294 (76.2)
##   early_readmit = 1 (%)           6283 ( 9.0)
##   discharge_to_home = 1 (%)      44317 (63.3)
##   admission_source (%)
##      ER                          37269 (53.3)
##      Other                       10059 (14.4)
##      P/C                         22649 (32.4)
##   race_group (%)
##      Black                       12623 (18.0)
##      Other                        3136 ( 4.5)
##      White                       52302 (74.7)
##      Missing/Unknown              1916 ( 2.7)
##   age_group (%)
##      >60                         46300 (66.2)
##      0-30                         1808 ( 2.6)
##      30-60                       21869 (31.3)
##   med_specialty (%)
##      Cardiology                   4207 ( 6.0)
##      Family/GeneralPractice       4978 ( 7.1)
##      InternalMedicine            10640 (15.2)
##      Missing/Unknown             33656 (48.1)
##      Other                       12746 (18.2)
##      Surgery                      3750 ( 5.4)
##   encounter_type (%)
##      1                           57134 (81.6)
##      2                            3740 ( 5.3)
##      3                            4056 ( 5.8)
##      4                            2181 ( 3.1)
##      Other                        2866 ( 4.1)
##   diag_1_cat (%)
##      Circulatory                 21321 (30.5)
##      Diabetes                     5748 ( 8.2)
##      Digestive                    6326 ( 9.0)
##      Injury/Poisoning             4694 ( 6.7)
##      Musculoskeletal              4064 ( 5.8)
##      Neoplasms                    2538 ( 3.6)
##      Other                       18835 (26.9)
##      Respiratory                  6451 ( 9.2)
```

Where `encounter_type` is a new variable ("HbA1c" in the paper), deemed of primary interest by the authors, which combines information from the A1c test result and medication change variables. Note that prior to building our models, we perform a random 25/75 split of this data, using 25% as a hold out test set to assess and compare each model's performance, and the remaining 75% to train them (reference Table 7 for specific information regarding the data sets used).

Logistic Regression

To build our logistic regression model, we first followed an AIC-based step-wise elimination procedure to identify significant predictors. Starting with the main effects model (no variable interactions), which included all the variables from the altered dataset shown previously, it was found that all variables in the data were significant, except: `encounter_type`, `race_group`, `gender`, `num_procedures`, `num_medications`, `number_outpatient`. However, due to their discovered importance in the paper, we decide to keep the `encounter_type` and `race_group` predictors. Finally, we use a similar combined approach to obtain the following significant two-way interactions (reference code for specific model selection procedure).

Significant Interactions (12):

- `time_in_hospital: discharge_to_home, diag_1_cat, med_specialty, number_inpatient`
- `discharge_to_home: diag_1_cat, med_specialty, race_group, number_diagnoses`
- `diag_1_cat: encounter_type`
- `med_specialty: age_group`
- `age_group: number_diagnoses`
- `admission_source: diabetesMed`

With the inclusion of these interactions, we arrive at the following final logistic regression model of 25 terms (13 main effects, 12 interactions), which lowered the previous main effects model's AIC of 30948.89 by 20.95 units.

Model 1:

```
glm(early_readmit ~ time_in_hospital*(discharge_to_home + diag_1_cat
+ med_specialty + number_inpatient) + discharge_to_home*(diag_1_cat +
med_specialty + race_group + number_diagnoses) + age_group*(med_specialty
+ number_diagnoses) + diag_1_cat*encounter_type + admission_source*diabetesMed
+ num_lab_procedures + number_emergency + number_inpatient, data=diabetes_train,
family=binomial(link="logit"))
```

After training the model on the aforementioned train set, we obtain the corresponding coefficients (which include more terms than the model, due to the inclusion of dummy variables for categorical predictors of more than two groups), along with their exponentiated forms (odds ratios). Given the logit link function used for this kind of linear model, we can interpret each coefficient as the corresponding increase in the log of the odds of early readmission for: a given predictor compared to the reference group if it is categorical, or a one unit increase in the predictor if it is continuous.

Table 6: Logistic Regression Model Coefficients and Odds Ratios

Term	Coefficient	OR
(Intercept)	-2.85233	0.05771
time_in_hospital	0.07137	1.07398
discharge_to_home1	-0.38694	0.67913
diag_1_catDiabetes	-0.06000	0.94177
diag_1_catDigestive	-0.02192	0.97832
diag_1_catInjury/Poisoning	0.19209	1.21178
diag_1_catMusculoskeletal	-0.35245	0.70296
diag_1_catNeoplasms	-0.15008	0.86064
diag_1_catOther	-0.14061	0.86883
diag_1_catRespiratory	-0.06283	0.93910
med_specialtyFamily/GeneralPractice	0.79490	2.21422
med_specialtyInternalMedicine	0.59083	1.80549
med_specialtyMissing/Unknown	0.72959	2.07423
med_specialtyOther	0.76480	2.14857
med_specialtySurgery	0.91055	2.48569

Term	Coefficient	OR
number_inpatient	0.38301	1.46670
race_groupOther	0.15632	1.16920
race_groupWhite	0.00983	1.00988
race_groupMissing/Unknown	0.00574	1.00576
number_diagnoses	-0.00909	0.99095
age_group0-30	-12.56439	0.00000
age_group30-60	-0.54692	0.57873
encounter_type2	-0.09504	0.90934
encounter_type3	0.06740	1.06972
encounter_type4	-0.12257	0.88464
encounter_typeOther	-0.14423	0.86569
admission_sourceOther	-0.33003	0.71890
admission_sourceP/C	-0.03496	0.96564
diabetesMedYes	0.16387	1.17806
num_lab_procedures	0.00286	1.00287
number_emergency	0.09964	1.10477
time_in_hospital:discharge_to_home1	0.02399	1.02428
time_in_hospital:diag_1_catDiabetes	0.03294	1.03348
time_in_hospital:diag_1_catDigestive	0.00072	1.00072
time_in_hospital:diag_1_catInjury/Poisoning	-0.01723	0.98292
time_in_hospital:diag_1_catMusculoskeletal	0.06000	1.06184
time_in_hospital:diag_1_catNeoplasms	0.00173	1.00173
time_in_hospital:diag_1_catOther	0.01859	1.01876
time_in_hospital:diag_1_catRespiratory	-0.00047	0.99953
time_in_hospital:med_specialtyFamily/GeneralPractice	-0.07136	0.93112
time_in_hospital:med_specialtyInternalMedicine	-0.04716	0.95393
time_in_hospital:med_specialtyMissing/Unknown	-0.07859	0.92442
time_in_hospital:med_specialtyOther	-0.07948	0.92360
time_in_hospital:med_specialtySurgery	-0.10961	0.89618
time_in_hospital:number_inpatient	-0.01140	0.98866
discharge_to_home1:diag_1_catDiabetes	-0.14233	0.86734
discharge_to_home1:diag_1_catDigestive	-0.13314	0.87534
discharge_to_home1:diag_1_catInjury/Poisoning	-0.40262	0.66857
discharge_to_home1:diag_1_catMusculoskeletal	-0.39240	0.67543
discharge_to_home1:diag_1_catNeoplasms	0.14824	1.15980
discharge_to_home1:diag_1_catOther	-0.17623	0.83842
discharge_to_home1:diag_1_catRespiratory	-0.18372	0.83217
discharge_to_home1:med_specialtyFamily/GeneralPractice	-0.18624	0.83008
discharge_to_home1:med_specialtyInternalMedicine	-0.19120	0.82597
discharge_to_home1:med_specialtyMissing/Unknown	-0.28913	0.74892
discharge_to_home1:med_specialtyOther	-0.41557	0.65996
discharge_to_home1:med_specialtySurgery	-0.57574	0.56229
discharge_to_home1:race_groupOther	-0.55524	0.57393
discharge_to_home1:race_groupWhite	-0.07483	0.92790
discharge_to_home1:race_groupMissing/Unknown	-0.25819	0.77245
discharge_to_home1:number_diagnoses	0.02896	1.02939
med_specialtyFamily/GeneralPractice:age_group0-30	0.00985	1.00990
med_specialtyInternalMedicine:age_group0-30	11.79834	133031.36212
med_specialtyMissing/Unknown:age_group0-30	12.37166	236017.28246
med_specialtyOther:age_group0-30	11.61756	111031.02604
med_specialtySurgery:age_group0-30	11.02809	61579.68898
med_specialtyFamily/GeneralPractice:age_group30-60	-0.10322	0.90193

Term	Coefficient	OR
med_specialtyInternalMedicine:age_group30-60	-0.04531	0.95570
med_specialtyMissing/Unknown:age_group30-60	0.08014	1.08344
med_specialtyOther:age_group30-60	0.11842	1.12571
med_specialtySurgery:age_group30-60	0.25274	1.28755
number_diagnoses:age_group0-30	0.07311	1.07585
number_diagnoses:age_group30-60	0.04236	1.04327
diag_1_catDiabetes:encounter_type2	-0.12904	0.87894
diag_1_catDigestive:encounter_type2	-0.12989	0.87819
diag_1_catInjury/Poisoning:encounter_type2	-0.81380	0.44317
diag_1_catMusculoskeletal:encounter_type2	-0.08866	0.91516
diag_1_catNeoplasms:encounter_type2	-0.08955	0.91434
diag_1_catOther:encounter_type2	0.20417	1.22650
diag_1_catRespiratory:encounter_type2	-0.86977	0.41905
diag_1_catDiabetes:encounter_type3	-0.48770	0.61404
diag_1_catDigestive:encounter_type3	-0.33085	0.71832
diag_1_catInjury/Poisoning:encounter_type3	-0.61405	0.54115
diag_1_catMusculoskeletal:encounter_type3	0.42719	1.53295
diag_1_catNeoplasms:encounter_type3	-1.02973	0.35710
diag_1_catOther:encounter_type3	-0.21836	0.80384
diag_1_catRespiratory:encounter_type3	-0.34091	0.71112
diag_1_catDiabetes:encounter_type4	-0.28430	0.75254
diag_1_catDigestive:encounter_type4	0.07703	1.08007
diag_1_catInjury/Poisoning:encounter_type4	-0.35234	0.70304
diag_1_catMusculoskeletal:encounter_type4	0.27627	1.31820
diag_1_catNeoplasms:encounter_type4	0.25524	1.29077
diag_1_catOther:encounter_type4	0.01675	1.01689
diag_1_catRespiratory:encounter_type4	0.24843	1.28201
diag_1_catDiabetes:encounter_typeOther	0.25958	1.29639
diag_1_catDigestive:encounter_typeOther	0.00010	1.00010
diag_1_catInjury/Poisoning:encounter_typeOther	-0.10653	0.89895
diag_1_catMusculoskeletal:encounter_typeOther	0.03285	1.03340
diag_1_catNeoplasms:encounter_typeOther	0.34808	1.41634
diag_1_catOther:encounter_typeOther	-0.09920	0.90556
diag_1_catRespiratory:encounter_typeOther	-0.35302	0.70256
admission_sourceOther:diabetesMedYes	0.27192	1.31249
admission_sourceP/C:diabetesMedYes	0.08081	1.08417

Classification Tree

Given that a classification tree model doesn't allow for variable interactions, we construct our tree using the variables included in the main effects logistic regression model. Additionally, having specified `minsplit`, `minbucket`, and `maxdepth` values of 4, 1, and 30, respectively, we obtain a tree of a single root node with a root node error of equivalent to the proportion of positive observations in the training data (0.0905). The model itself has the following form.

Model 2:

```
rpart(early_readmit ~ encounter_type + race_group + age_group + discharge_to_home
+ admission_source + time_in_hospital + med_specialty + num_lab_procedures
+ number_emergency + number_inpatient + number_diagnoses + diabetesMed
+ diag_1_cat, method="class", data=diabetes_train, minsplit=4, minbucket=1,
maxdepth=30)
```

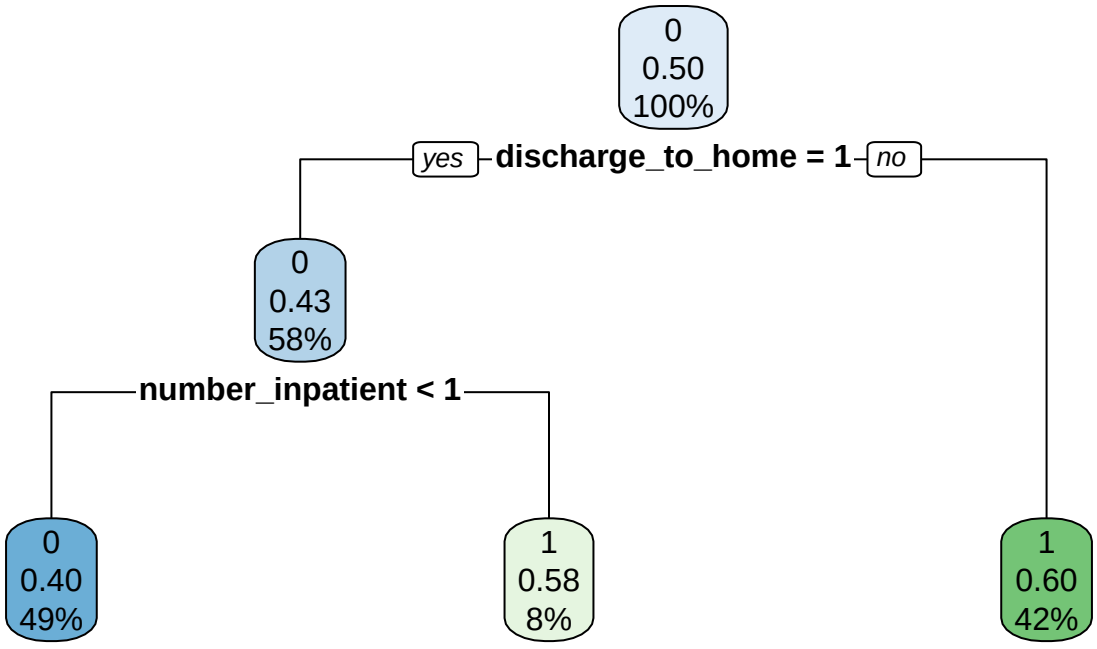
We can assume that this model produced a tree of a single root node due to the fact that it was trained on severely unbalanced data. That is, data that has such a low number of positive predictions or cases of early readmission (<10%) compared to negative ones or cases of non-early readmission. Thus, to obtain a perhaps more realistic model, we use an under-sampling technique to equalize the level of positive and negative observations. Despite having to decrease the number of negative observations to do this and hence, the size of our training data altogether, we arrive at a more substantive classification model, which has as many negative observations as it has positive ones. Table 7 below, summarizes the final datasets used for constructing the three models, which are all subsets of the original altered data discussed previously.

Table 7: Datasets Used

Dataset	Observations	Proportion	Positive	Negative
Complete	69977	1.00000	6283	63694
Test	17494	0.25000	1533	15961
Train	52483	0.75000	4750	47733
Balanced Train	9500	0.13576	4750	4750

Staying consistent with the total number of positive observations in the original training set (4,750) to retain as much data as possible, the third and final model we consider, is of the same form as the previous one, but is trained on a rather small portion of the data by contrast (as shown in the table above). With a root node error of 0.5 and inclusion of the `discharge_to_home` and `number_inpatient` predictors, Model 3, yields the following tree representation (Figure 7).

Figure 7: Classification Tree Model for Balanced Data



To assess and compare the overall performance of each of the models on the test data, we focus on the corresponding ROC curves given by Figures 8.1-8.3 as well as the measures shown in Table 8 below.

Table 8: Model Performance Measures

Model	TP	FP	TN	FN	Sensitivity	Specificity	Accuracy	AUC
Logistic	3	12	15949	1530	0.00196	0.99925	0.9119	0.63391
Classification Tree	0	0	15961	1533	0.00000	1.00000	0.9124	0.50000
Classification Tree (Balanced)	946	6732	9229	587	0.61709	0.57822	0.5816	0.59000

Looking at the values provided, we see that despite having a high specificity rate of 99.92%, the logistic regression model fell short in displaying a low sensitivity rate (0.2%), having committed several type II errors (FN). That is, despite having accurately predicted the majority of non-early readmission cases, this model demonstrated a high number of false negatives, or misclassifications of existing early readmission cases. More strikingly is the fact that out of the 15 positive predictions made (which is already low compared to the true number of positive observations (1,533), only 1/5 of them were actually correct. So, although the model had a small number of false positives (FP), this number is quite large in relation to the total number of positive predicted values. Thus, its corresponding AUC of 0.63 just barely deems it a satisfactory model, being able to predict cases of non-early readmission rather effectively, but being limited in its ability to correctly identify actual cases of early readmission (which is the overarching research objective).

Similarly, in having predicted 100% of the negative cases accurately (since only negative predictions were made), the classification tree model trained on all the available test data showed a total of 0 true and false positives, and as many false negatives as there were cases of early readmission or positive observations (1,533), hence giving it a sensitivity rate of 0% and a 100% specificity. According to this model's AUC of 0.5 (lowest possible AUC attainable), which is supported by its linear ROC trend (TP=FP), the model is inherently unsatisfactory by these standards. Namely, despite being "good" at predicting cases of non-early readmission, this model is incapable of predicting cases of early readmission (whether correct or not). In this way, both models' high classification accuracies are misleading, causing one to think they serve as effective means of predicting early readmission status, when in fact they merely reflect the low prevalence of positive cases in the data and in no way speak to the model's true predictive power.

This being the case, accuracy scores will remain high for these models when tested on data similar to that which it was trained on (since all or most of its predictions will be negative), and radically lower when tested on data with a greater number positive early-readmission cases. Conversely, we see that classification accuracy decreases significantly for the third tree model that was trained on the balanced data (subset of the original train data). Having correctly classified only 58.16% of the same test data, one would be tempted to think that this model is far less effective than the previous two. However, in balancing its train data, we've also increased the model's sensitivity remarkably. That is, its ability to classify instances of early readmission. And, although this comes at the cost of some specificity and overall accuracy (which we could attribute to a loss of information in the under-sampling procedure), we now have a less biased account of the model's true predictive ability given the variables and information available in the data. More careful inspection of these measurements reveals that the model is now almost equally as effective in identifying instances of positive cases as it is negative ones. Yet, it continues to display better performance with respect to the latter.

Looking at all three AUC values, we notice that this third model is, in a way, a half-way point between the other two models in terms of it's overall classification accuracy. It could be that including meaningful variable interactions within the tree model improves its performance to be up to par with that of the logistic regression model. Or similarly, that training the logistic regression model on more balanced data improves its performance far beyond that of the classification tree's. Still, it is not certain weather one model is significantly better than the other, although it would seem that a logistic regression approach would be preferred given the complexity of significant predictors. What can be said however, is that the results we see here are more reflective of the nature of the relationship between existing variables and our primary outcome, the structure of data available, and its influence on model performance, rather than each method's stand-alone predictive capability.

Figure 8.1: ROC Curve − Logistic Regression Model

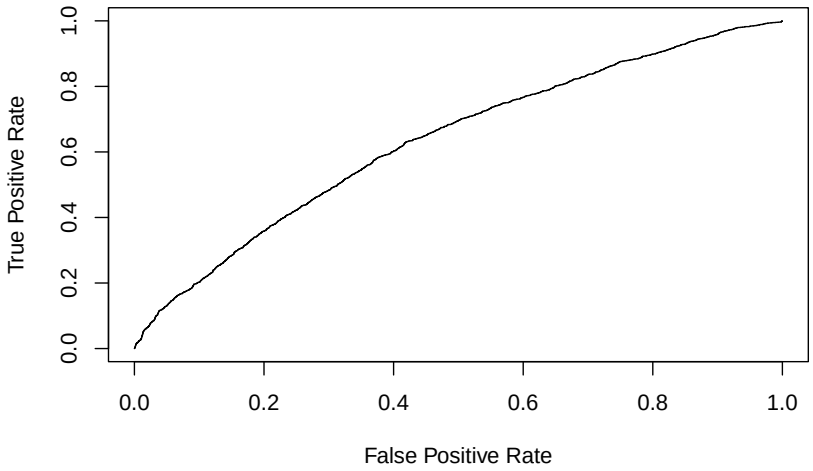


Figure 8.2: ROC Curve − Classification Tree Model 1

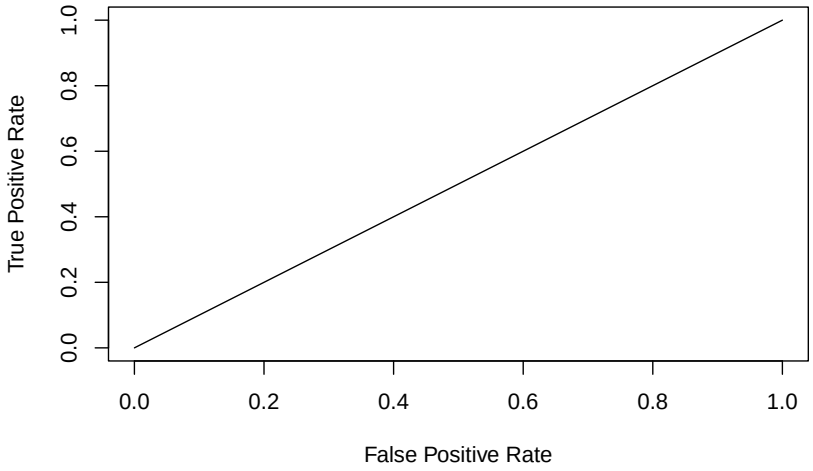
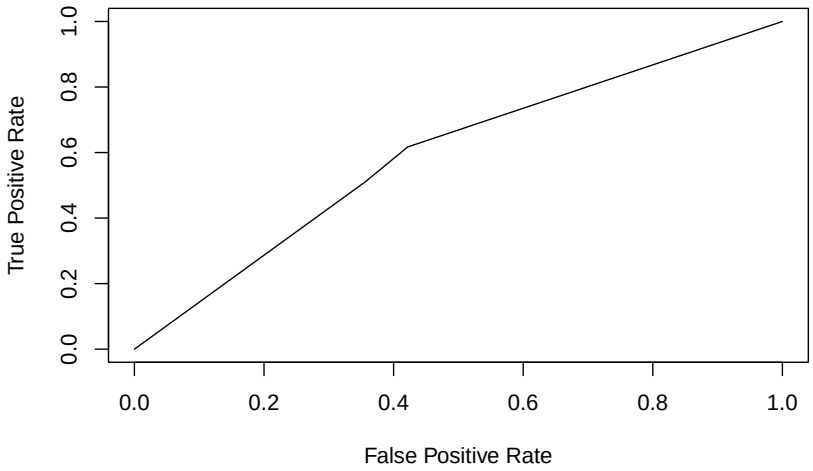


Figure 8.3: ROC Curve − Classification Tree Model 2



Code

```
################################ Question 2 ###################################

# 1. Univariate Non-Convex Function
of1 <- function(x) ((x^2-4*x+4)*(x^2+4*x+2))
x_vals <- seq(-4, 4, 0.1)
y_vals <- of1(x_vals)
df_1 <- data.frame(x=x_vals, y=y_vals)
ggplot() +
  geom_point(data=df_1, aes(x_vals, y_vals), fill="black") +
  xlim(-4, 4) +
  geom_vline(aes(xintercept=-2.41421356237), color="red", lty="dashed") +
  labs(title="Figure 1: Univariate Non-Convex Objective Function (1)", x="x", y="y")

# optimal solution: x=-1-sqrt(2)=-2.41421356237
```

```
set.seed(47)

# 2a. Univariate Linear Regression (intercept): Simulated Data
sig1 <- rbind(c(1,0.90), c(0.90, 1))
data1 <- mvrnorm(n=30, mu=c(3, 4), Sigma=sig1)
# ggplot() +
#   geom_point(aes(data1[,1], data1[,2]), color="black") +
#   labs(title="", x="x", y="y") +
#   scale_x_continuous(limits=c(0, 5), breaks=seq(0, 5, 1)) +
#   scale_y_continuous(limits=c(0, 7), breaks=seq(0, 7, 1))

# Calculating optimal solution using traditional linear regression
lm1 <- lm(data1[,2]~data1[,1]) # intercept = 1.3103
ggplot() +
  geom_point(aes(data1[,1], data1[,2]), fill="black") +
  scale_x_continuous(limits=c(0, 5), breaks=seq(0, 5, 1)) +
  scale_y_continuous(limits=c(0, 7), breaks=seq(0, 7, 1)) +
  geom_abline(aes(intercept=1.3103, slope=0.9052), color="red", lty="dashed") +
  labs(title="Figure 2.1: Univariate Linear Regression Simulated Data (2a)", x="x", y="y")

# This is what we are trying to optimize
of2a <- function(intercept, x=data1[,1], y=data1[,2]){
  SSRs <- c()
  for (i in 1:length(intercept)){
     pred <- intercept[i] + 0.9052*x
     obs <- y
     SSR <- sum((obs-pred)**2)
     SSRs <- c(SSRs, SSR)
  }
  return(SSRs)
}
intercepts <- seq(0, 2.5, 0.05)
ssrs <- of2a(intercept=intercepts)
df_2a <- data.frame(intercepts, ssrs)
ggplot(df_2a) +
  geom_point(aes(intercepts, ssrs), fill="black") +
  xlim(0, 2.5) +
```

```
  geom_vline(aes(xintercept=1.3103), color="red", lty="dashed") +
  labs(title="Figure 2.2: Univariate Linear Regression Objective Function (2a)",
       x="Intercept", y="SSR")
```

```
set.seed(47)

# 2b. Multivariate Linear Regression (intercept + 2 slopes): Simulated Data
sig2 <- rbind(c(1, 0.85, 0.75), c(0.85, 1, 0.9), c(0.75, 0.9, 1))
data2 <- mvrnorm(n=30, mu=c(3, 4, 5), Sigma=sig2)
y <- data2[,2]
x <- data2[,1]
z <- data2[,3]
data2_df <- as.data.frame(cbind(y, x, z))

# Calculating optimal solution using multiple linear regression
# y ~ b1 + b2*x + b3*z
lm2 <- lm(formula=y~., data2_df) # b1 = 0.02969, b2 = 0.46741, b3 = 0.49582
#summary(lm2)
ggplot(data2_df, aes(x, y, color=z)) +
  geom_point() +
  scale_x_continuous(limits=c(1, 5.5), breaks=seq(1, 5.5, 1)) +
  scale_y_continuous(limits=c(1, 7), breaks=seq(1, 7, 1)) +
  stat_smooth(method="lm", se=FALSE, color="red", lty="dashed") +
  labs(title="Figure 3.1: Multivariate Linear Regression Simulated Data (2b)", x="x", y="y")

# This is what we are trying to optimize
of2b <- function(b1, b2, b3, x=data2_df$x, y=data2_df$y, z=data2_df$z){
  SSRs1 <- c()
  SSRs2 <- c()
  SSRs3 <- c()
  for (i in 1:length(b1)){
     pred1 <- b1[i] + 0.46741*x + 0.49582*z
     pred2 <- 0.02969 + b2[i]*x + 0.49582*z
     pred3 <- 0.02969 + 0.46741*x + b3[i]*z
     obs <- y
     SSR1 <- sum((obs-pred1)**2)
     SSR2 <- sum((obs-pred2)**2)
     SSR3 <- sum((obs-pred3)**2)
     SSRs1 <- c(SSRs1, SSR1)
     SSRs2 <- c(SSRs2, SSR2)
     SSRs3 <- c(SSRs3, SSR3)
  }
  return(cbind(SSRs1, SSRs2, SSRs3))
}
bis <- seq(-2, 2.5, 0.05)
ssrs1 <- of2b(b1=bis, b2=bis, b3=bis)[,1]
ssrs2 <- of2b(b1=bis, b2=bis, b3=bis)[,2]
ssrs3 <- of2b(b1=bis, b2=bis, b3=bis)[,3]
ssrs123 <- as.data.frame(cbind(ssrs1, ssrs2, ssrs3, coeffs=bis)) %>%
  pivot_longer(cols=starts_with("ssrs"), names_to="bi",
               names_prefix="ssrs", values_to="SSR")
optimals_2b <- data.frame(bis=c("b1", "b2", "b3"), coeffs=c(0.02969, 0.46741, 0.49582))
ggplot(ssrs123, aes(coeffs, SSR, color=bi)) +
  geom_point() +
```

```
  #facet_wrap(~ bi) +
  geom_vline(aes(xintercept=coeffs), optimals_2b, color="red", lty="dashed") +
  labs(title="Figure 3.2: Multivariate Linear Regression Objective Function(s) (2b)",
       x="Coefficients", y="SSR")
```

```
## Gradients for 3 Objective Functions

# Gradient 1 (Derivative of Objective Function 1) - Univariate Non-Convex Function
of_grad_1 <- Deriv(of1)

# Gradient 2a (Derivative of Objective Function 2a) - Univariate LR (intercept)
of_grad_2a <- function(b1, x=data1[,1], y=data1[,2]){
  # taking the Deriv(~sum((y-(b1+0.9052*x))**2), "b1")  wrt the intercept (b1)
  gradient <- sum(-(2*(y - (0.9052*x + b1)))) # constant slope
  return(gradient)
}

# Gradient 2b (Derivative of Objective Function 2b) -  Multivariate LR (intercept + 2 slopes)
of_grad_2b <- function(b1, b2, b3, y=data2_df$y, x=data2_df$x, z=data2_df$z){
  # taking the Deriv(~sum((y-(b1+b2*x+b3*z))**2)) wrt b1, b2, & b3
  derivative1 <- sum(-(2*(y - (b1 + b2*x + b3*z)))) # wrt intercept (b1)
  derivative2 <- sum(-(2*(x*(y - (b1 + b2*x + b3*z))))) # wrt slope of x (b2)
  derivative3 <- sum(-(2*(z*(y - (b1 + b2*x + b3*z))))) #wrt slope of z (b3)
  gradient <- c(derivative1, derivative2, derivative3)
  return(gradient)
}
```

```
## (Vanilla) Gradient Descent Algorithms

# Gradient Descent 1 - Univariate
gradient_desc_1 <- function(f_grad, x_init=0){
  # parameters
  max_iters <- 50 # stopping criteria 1: maximum number of iterations/steps
  learning_rate <- 0.01
  # starting conditions
  x <- rep(0, max_iters) # stopping criteria 1
  x[1] <- x_init
  # descent
  for (i in 2:max_iters){
    x[i] <- x[i-1]-learning_rate*f_grad(x[i-1])
    if (abs(x[i]-x[i-1]) <= 0.001){break} # stopping criteria 2: step size close to 0
  }
  return(x[1:i]) # stopping criteria 1
}

# Gradient Descent 2 - Multivariate LR (intercept + 2 slopes)
gradient_desc_2 <- function(f_grad, b1_init=0, b2_init=0, b3_init=0){
  # parameters
  max_iters <- 50 # stopping criteria 1: maximum number of iterations/steps
  learning_rate <- 0.0001 # needed smaller learning rate
  # starting conditions
  b1 <- rep(0, max_iters) # stopping criteria 1
  b2 <- rep(0, max_iters) # stopping criteria 1
  b3 <- rep(0, max_iters) # stopping criteria 1
```

```
  b1[1] <- b1_init # initial intercept
  b2[1] <- b2_init # initial slope of x
  b3[1] <- b3_init # initial slope of z
  # descent
  for (i in 2:max_iters){
    b1[i] <- b1[i-1]-learning_rate*f_grad(b1[i-1], b2[i-1], b3[i-1])[1]
    b2[i] <- b2[i-1]-learning_rate*f_grad(b1[i-1], b2[i-1], b3[i-1])[2]
    b3[i] <- b3[i-1]-learning_rate*f_grad(b1[i-1], b2[i-1], b3[i-1])[3]
    if (abs(b1[i]-b1[i-1]) <= 0.001 &
        abs(b2[i]-b2[i-1]) <= 0.001 &
        abs(b3[i]-b3[i-1]) <= 0.001){break} # stopping criteria 2: step sizes close to 0
  }
  intercept <- b1[1:i]
  slope_x <- b2[1:i]
  slope_z <- b3[1:i]
  return(cbind(intercept, slope_x, slope_z)) # stopping criteria 1
}
```

```
## ADAM Optimizers/Algorithms

# ADAM 1 - Univariate
adam_1 <- function(f_grad, x_init=0){
  # parameters (values that seem to work best)
  max_iters <- 50
  learning_rate <- 0.1 # or 0.001 (default)
  beta1 <- 0.09 # also runif(1, 0, 1) or 0.9 (default)  - lower is better
  beta2 <- 0.6 # also runif(1, 0, 1) or 0.999 (default) - larger is better
  epsilon <- 1e-8
  # starting conditions
  x <- rep(0, max_iters) # stopping criteria 1
  x[1] <- x_init
  # initializing first and second moment vectors
    m <- rep(0, max_iters)
    v <- rep(0, max_iters)
  # descent
  for (i in 2:max_iters){
    m[i] <- (beta1*m[i-1])+((1-beta1)*f_grad(x[i-1])) # updated first moment
        v[i] <- (beta2*v[i-1])+((1-beta2)*(f_grad(x[i-1])**2)) # updated second moment
        m_hat <- m[i]/(1-(beta1**(i))) # bias-corrected first moment
        v_hat <- v[i]/(1-(beta2**(i))) # bias-corrected second moment
    x[i] <- x[i-1]-(learning_rate*(m_hat/(sqrt(v_hat)+epsilon))) # updated b1
    if (abs(x[i]-x[i-1]) <= 0.001){break} # stopping criteria 2: step size close to 0
  }
  return(x[1:i]) # stopping criteria 1
}

# ADAM 2 - Multivariate LR (intercept + 2 slopes)
adam_2 <- function(f_grad, b1_init=0, b2_init=0, b3_init=0){
  # parameters (values that seem to work best)
  max_iters <- 50
  learning_rate <- 0.01 # or 0.001 (default)
  beta1 <- 0.09 # also runif(1, 0, 1) or 0.9 (default)  - lower is better
  beta2 <- 0.5 # also runif(1, 0, 1) or 0.999 (default) - larger is better
  epsilon <- 1e-8
```

```
  # starting conditions
  b1 <- rep(0, max_iters) # stopping criteria 1
  b2 <- rep(0, max_iters) # stopping criteria 1
  b3 <- rep(0, max_iters) # stopping criteria 1
  b1[1] <- b1_init # initial intercept
  b2[1] <- b2_init # initial slope of x
  b3[1] <- b3_init # initial slope of z
  # initializing first & second moment and bias-corrected moment matrices
    m <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
    v <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
    m_hat <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
    v_hat <- matrix(rep(0, max_iters*3), nrow=max_iters, ncol=3)
  # descent
  for (i in 2:max_iters){
    for (j in 1:3){
      # updating first & second moment and bias-corrected moment matrices
      m[i,j] <- (beta1*m[i-1])+((1-beta1)*f_grad(b1[i-1], b2[i-1], b3[i-1])[j])
          v[i,j] <- (beta2*v[i-1])+((1-beta2)*(f_grad(b1[i-1], b2[i-1], b3[i-1])[j]**2))
        m_hat[i,j] <- m[i,j]/(1-(beta1**(i)))
        v_hat[i,j] <- v[i,j]/(1-(beta2**(i)))
        if (j==1) {
            b1[i] <- b1[i-1]-(learning_rate*(m_hat[i,j]/(sqrt(v_hat[i,j])+epsilon)))
          } else if (j==2) {
            b2[i] <- b1[i-1]-(learning_rate*(m_hat[i,j]/(sqrt(v_hat[i,j])+epsilon)))
          } else if (j==3) {
           b3[i] <- b1[i-1]-(learning_rate*(m_hat[i,j]/(sqrt(v_hat[i,j])+epsilon)))
          }
    }
  # stopping criteria 2: step sizes close to 0
  if (abs(b1[i]-b1[i-1]) <= 0.001 &
        abs(b2[i]-b2[i-1]) <= 0.001 &
        abs(b3[i]-b3[i-1]) <= 0.001){break}
  }
  intercept <- b1[1:i]
  slope_x <- b2[1:i]
  slope_z <- b3[1:i]
  return(cbind(intercept, slope_x, slope_z)) # stopping criteria 1
}
```

```
### Plotting Algorithm Trajectories for 3 Objective Functions

## 1. Univariate Non-Convex Function
# Gradient Descent
x_1i <- gradient_desc_1(of_grad_1, 0.3)
y_1i <- of1(x_1i)
df_1i <- data.frame(x=x_1i, y=y_1i, time_1i=1:length(x_1i))
of_df_1i <- data.frame(x_1i=seq(-4, 4, 0.1))
of_df_1i$y_1i <- of1(of_df_1i$x_1i)
p_1i <- ggplot() +
  geom_line(data=of_df_1i, aes(x_1i, y_1i), color="blue") +
  geom_point(data=df_1i, aes(x_1i, y_1i, frame=time_1i)) +
  xlim(-4, 4) +
  labs(title="Figure 4.1: GD Algorithm for Univariate Non-Convex Function",
       x="x", y="y")
```

```r
# ADAM
x_1ii <- adam_1(of_grad_1, 0.3)
y_1ii <- of1(x_1ii)
df_1ii <- data.frame(x=x_1ii, y=y_1ii, time_1ii=1:length(x_1ii))
of_df_1ii <- data.frame(x_1ii=seq(-4, 4, 0.1))
of_df_1ii$y_1ii <- of1(of_df_1ii$x_1ii)
p_1ii <- ggplot() +
  geom_line(data=of_df_1ii, aes(x_1ii, y_1ii), color="red") +
  geom_point(data=df_1ii, aes(x_1ii, y_1ii, frame=time_1ii)) +
  xlim(-4, 4) +
  labs(title="Figure 4.2: ADAM Algorithm for Univariate Non-Convex Function",
       x="x", y="y")

## 2a. Univariate Linear Regression (intercept)
# Gradient Descent
x_2ai <- gradient_desc_1(of_grad_2a)
y_2ai <- of2a(intercept=x_2ai)
df_2ai <- data.frame(x=x_2ai, y=y_2ai, time_2ai=1:length(x_2ai))
of_df_2ai <- data.frame(x_2ai=seq(-2.5, 5, 0.1))
of_df_2ai$y_2ai <- of2a(intercept=of_df_2ai$x_2ai)
p_2ai <- ggplot() +
  geom_line(data=of_df_2ai, aes(x_2ai, y_2ai), color="blue") +
  geom_point(data=df_2ai, aes(x_2ai, y_2ai, frame=time_2ai)) +
  xlim(-2.5, 5) +
  labs(title="Figure 5.1: GD Algorithm for Univariate Linear Regression (intercept)",
       x="x", y="y")
# ADAM
x_2aii <- adam_1(of_grad_2a)
y_2aii <- of2a(intercept=x_2aii)
df_2aii <- data.frame(x=x_2aii, y=y_2aii, time_2aii=1:length(x_2aii))
of_df_2aii <- data.frame(x_2aii=seq(-2.5, 5, 0.1))
of_df_2aii$y_2aii <- of2a(intercept=of_df_2aii$x_2aii)
p_2aii <- ggplot() +
  geom_line(data=of_df_2aii, aes(x_2aii, y_2aii), color="red") +
  geom_point(data=df_2aii, aes(x_2aii, y_2aii, frame=time_2aii)) +
  xlim(-2.5, 5) +
  labs(title="Figure 5.2: ADAM Algorithm for Univariate Linear Regression (intercept)",
       x="x", y="y")
```

```r
## 2b. Multivariate Linear Regression (intercept + 2 slopes)
# Gradient Descent
# intercept (b1)
x_2bi_1 <- gradient_desc_2(of_grad_2b)[,1]
y_2bi_1 <- of2b(b1=x_2bi_1, b2=x_2bi_1, b3=x_2bi_1)[,1]
df_2bi_1 <- data.frame(x=x_2bi_1, y=y_2bi_1, time_2bi_1=1:length(x_2bi_1))
of_df_2bi_1 <- data.frame(x_2bi_1=seq(-1, 1.5, 0.05))
of_df_2bi_1$y_2bi_1 <- of2b(b1=of_df_2bi_1$x_2bi_1,
                            b2=of_df_2bi_1$x_2bi_1,
                            b3=of_df_2bi_1$x_2bi_1)[,1]
p_2bi_1 <- ggplot() +
  geom_line(data=of_df_2bi_1, aes(x_2bi_1, y_2bi_1), color="blue") +
  geom_point(data=df_2bi_1, aes(x_2bi_1, y_2bi_1, frame=time_2bi_1)) +
  xlim(-1, 1.5) +
  labs(title="Figure 6.1a: GD Algorithm for Multivariate Linear Regression (intercept, b1)",
```

```
       x="x", y="y")
# slope of x (b2)
x_2bi_2 <- gradient_desc_2(of_grad_2b)[,2]
y_2bi_2 <- of2b(b1=x_2bi_2, b2=x_2bi_2, b3=x_2bi_2)[,2]
df_2bi_2 <- data.frame(x=x_2bi_2, y=y_2bi_2, time_2bi_2=1:length(x_2bi_2))
of_df_2bi_2 <- data.frame(x_2bi_2=seq(-1, 1.5, 0.05))
of_df_2bi_2$y_2bi_2 <- of2b(b1=of_df_2bi_2$x_2bi_2,
                            b2=of_df_2bi_2$x_2bi_2,
                            b3=of_df_2bi_2$x_2bi_2)[,2]
p_2bi_2 <- ggplot() +
  geom_line(data=of_df_2bi_2, aes(x_2bi_2, y_2bi_2), color="blue") +
  geom_point(data=df_2bi_2, aes(x_2bi_2, y_2bi_2, frame=time_2bi_2)) +
  xlim(-1, 1.5) +
  labs(title="Figure 6.1b: GD Algorithm for Multivariate Linear Regression (x slope, b2)",
       x="x", y="y")
# slope of z (b3)
x_2bi_3 <- gradient_desc_2(of_grad_2b)[,3]
y_2bi_3 <- of2b(b1=x_2bi_3, b2=x_2bi_3, b3=x_2bi_3)[,3]
df_2bi_3 <- data.frame(x=x_2bi_3, y=y_2bi_3, time_2bi_3=1:length(x_2bi_3))
of_df_2bi_3 <- data.frame(x_2bi_3=seq(-1, 1.5, 0.05))
of_df_2bi_3$y_2bi_3 <- of2b(b1=of_df_2bi_3$x_2bi_3,
                            b2=of_df_2bi_3$x_2bi_3,
                            b3=of_df_2bi_3$x_2bi_3)[,3]
p_2bi_3 <- ggplot() +
  geom_line(data=of_df_2bi_3, aes(x_2bi_3, y_2bi_3), color="blue") +
  geom_point(data=df_2bi_3, aes(x_2bi_3, y_2bi_3, frame=time_2bi_3)) +
  xlim(-1, 1.5) +
  labs(title="Figure 6.1c: GD Algorithm for Multivariate Linear Regression (z slope, b3)",
       x="x", y="y")
# ADAM
# intercept (b1)
x_2bii_1 <- adam_2(of_grad_2b)[,1]
y_2bii_1 <- of2b(b1=x_2bii_1, b2=x_2bii_1, b3=x_2bii_1)[,1]
df_2bii_1 <- data.frame(x=x_2bii_1, y=y_2bii_1, time_2bii_1=1:length(x_2bii_1))
of_df_2bii_1 <- data.frame(x_2bii_1=seq(-0.25, 0.75, 0.05))
of_df_2bii_1$y_2bii_1 <- of2b(b1=of_df_2bii_1$x_2bii_1,
                              b2=of_df_2bii_1$x_2bii_1,
                              b3=of_df_2bii_1$x_2bii_1)[,1]
p_2bii_1 <- ggplot() +
  geom_line(data=of_df_2bii_1, aes(x_2bii_1, y_2bii_1), color="red") +
  geom_point(data=df_2bii_1, aes(x_2bii_1, y_2bii_1, frame=time_2bii_1)) +
  xlim(-0.25, 0.75) +
  labs(title="Figure 6.2a: ADAM Algorithm for Multivariate Linear Regression (intercept, b1)",
       x="x", y="y")
# slope of x (b2)
x_2bii_2 <- adam_2(of_grad_2b)[,2]
y_2bii_2 <- of2b(b1=x_2bii_2, b2=x_2bii_2, b3=x_2bii_2)[,2]
df_2bii_2 <- data.frame(x=x_2bii_2, y=y_2bii_2, time_2bii_2=1:length(x_2bii_2))
of_df_2bii_2 <- data.frame(x_2bii_2=seq(-0.25, 0.75, 0.05))
of_df_2bii_2$y_2bii_2 <- of2b(b1=of_df_2bii_2$x_2bii_2,
                              b2=of_df_2bii_2$x_2bii_2,
                              b3=of_df_2bii_2$x_2bii_2)[,2]
p_2bii_2 <- ggplot() +
```

```
  geom_line(data=of_df_2bii_2, aes(x_2bii_2, y_2bii_2), color="red") +
  geom_point(data=df_2bii_2, aes(x_2bii_2, y_2bii_2, frame=time_2bii_2)) +
  xlim(-0.25, 0.75) +
  labs(title="Figure 6.2b: ADAM Algorithm for Multivariate Linear Regression (x slope, b2)",
       x="x", y="y")
# slope of z (b3)
x_2bii_3 <- adam_2(of_grad_2b)[,3]
y_2bii_3 <- of2b(b1=x_2bii_3, b2=x_2bii_3, b3=x_2bii_3)[,3]
df_2bii_3 <- data.frame(x=x_2bii_3, y=y_2bii_3, time_2bii_3=1:length(x_2bii_3))
of_df_2bii_3 <- data.frame(x_2bii_3=seq(-0.25, 0.75, 0.05))
of_df_2bii_3$y_2bii_3 <- of2b(b1=of_df_2bii_3$x_2bii_3,
                              b2=of_df_2bii_3$x_2bii_3,
                              b3=of_df_2bii_3$x_2bii_3)[,3]
p_2bii_3 <- ggplot() +
  geom_line(data=of_df_2bii_3, aes(x_2bii_3, y_2bii_3), color="red") +
  geom_point(data=df_2bii_3, aes(x_2bii_3, y_2bii_3, frame=time_2bii_3)) +
  xlim(-0.25, 0.75) +
  labs(title="Figure 6.2c: ADAM Algorithm for Multivariate Linear Regression (z slope, b3)",
       x="x", y="y")
```

```
### Running Algorithms on Gradients of 3 Objective Functions and Making Tables

## 1. Univariate Non-Convex Function
# optimal sol: x = -2.41421356237
gd1 <- gradient_desc_1(of_grad_1, 0.3)
gd1_time <- system.time(gradient_desc_1(of_grad_1, 0.3))
adam1 <- adam_1(of_grad_1, 0.3)
adam1_time <- system.time(adam_1(of_grad_1, 0.3))
# table 1
alg1 <- c("Gradient Descent", "ADAM")
iters1 <- c(length(gd1), length(adam1))
sols1 <- c(gd1[length(gd1)], adam1[length(adam1)])
error1 <- c(abs(-2.41421356237-sols1[1]), abs(-2.41421356237-sols1[2]))
runtime1 <- c(as.double(gd1_time[3]), as.double(adam1_time[3]))
table1 <- as.data.frame(cbind(alg1, iters1, sols1=round(sols1, 5),
                              error1=round(error1, 5), runtime1=round(runtime1, 5))) %>%
  rename("Algorithm"=alg1, "Steps"=iters1,
         "Solution"=sols1, "Absolute Error"=error1,
         "Execution Time"=runtime1)

## 2a. Univariate Linear Regression (intercept)
# optimal sol: b1 = 1.3103
gd2a <- gradient_desc_1(of_grad_2a)
gd2a_time <- system.time(gradient_desc_1(of_grad_2a))
adam2a <- adam_1(of_grad_2a)
adam2a_time <- system.time(adam_1(of_grad_2a))
# table 2
alg2 <- c("Gradient Descent", "ADAM")
iters2 <- c(length(gd2a), length(adam2a))
sols2 <- c(gd2a[length(gd2a)], adam2a[length(adam2a)])
error2 <- c(abs(1.3103-sols2[1]), abs(1.3103-sols2[2]))
runtime2 <- c(as.double(gd2a_time[3]), as.double(adam2a_time[3]))
table2 <- as.data.frame(cbind(alg2, iters2, sols2=round(sols2, 5),
                              error2=format(round(error2, 6), scientific=FALSE),
```

```
                              runtime2=round(runtime2,5))) %>%
  rename("Algorithm"=alg2, "Steps"=iters2,
         "Solution"=sols2, "Absolute Error"=error2,
         "Execution Time"=runtime2)

## 2b. Multivariate Linear Regression (intercept + 2 slopes)
# optimal sol: b1 = 0.02969, b2 = 0.46741, b3 = 0.49582
gd2b <- gradient_desc_2(of_grad_2b)
gd2b_time <- system.time(gradient_desc_2(of_grad_2b))
adam2b <- adam_2(of_grad_2b)
adam2b_time <- system.time(adam_2(of_grad_2b))
# these values are a lot closer together than those for GD
# tables 3a, 3b, 3c
alg3 <- c("Gradient Descent", "ADAM")
iters3 <- c(nrow(gd2b), nrow(adam2b))
sols3a <- c(as.double(gd2b[nrow(gd2b),1]), as.double(adam2b[nrow(adam2b),1]))
sols3b <- c(as.double(gd2b[nrow(gd2b),2]), as.double(adam2b[nrow(adam2b),2]))
sols3c <- c(as.double(gd2b[nrow(gd2b),3]), as.double(adam2b[nrow(adam2b),3]))
error3a <- abs(0.02969-sols3a)
error3b <- abs(0.46741-sols3b)
error3c <- abs(0.49582-sols3c)
runtime3 <- c(as.double(gd2b_time[3]), as.double(adam2b_time[3]))
table3a <- as.data.frame(cbind(alg3, iters3, sols3a=round(sols3a, 5),
                              error3a=round(error3a, 5), runtime3=round(runtime3, 5))) %>%
  rename("Algorithm"=alg3, "Steps"=iters3,
         "Solution"=sols3a, "Absolute Error"=error3a,
         "Execution Time"=runtime3)
table3b <- as.data.frame(cbind(alg3, iters3, sols3b=round(sols3b, 5),
                              error3b=round(error3b, 5), runtime3=round(runtime3, 5))) %>%
  rename("Algorithm"=alg3, "Steps"=iters3,
         "Solution"=sols3b, "Absolute Error"=error3b,
         "Execution Time"=runtime3)
table3c <- as.data.frame(cbind(alg3, iters3, sols3c=round(sols3c, 5),
                              error3c=round(error3c, 5), runtime3=round(runtime3, 5))) %>%
  rename("Algorithm"=alg3, "Steps"=iters3,
         "Solution"=sols3c, "Absolute Error"=error3c,
         "Execution Time"=runtime3)
```

```
################################## Question 3 ###################################

# Importing Data
diabetes <- read.csv("/Users/antonellabasso/Desktop/PHP2650/DATA/diabetic_data.csv")
names(diabetes)
```

```
## Data Wrangling

# replacing ?'s with NA values
diabetes[diabetes=="?"] <- NA

# changing some variables to factors
cat_vars <- c('race','gender','age','admission_type_id',
              'discharge_disposition_id','admission_source_id','payer_code',
              'medical_specialty','max_glu_serum',
              'A1Cresult','metformin','repaglinide','nateglinide','chlorpropamide',
```

```
              'glimepiride','acetohexamide','glipizide','glyburide','tolbutamide',
              'pioglitazone','rosiglitazone','acarbose','miglitol','troglitazone',
              'tolazamide','examide','citoglipton','insulin','glyburide.metformin',
              'glipizide.metformin','glimepiride.pioglitazone','metformin.rosiglitazone',
              'metformin.pioglitazone','change','diabetesMed','readmitted')
diabetes[cat_vars] <- lapply(diabetes[cat_vars], factor)

# filtering data to those who could be readmitted and first encounter
diabetes <- diabetes %>% filter(!(discharge_disposition_id %in% c(11, 13, 14, 19, 20, 21)))
diabetes <- diabetes %>%
  group_by(patient_nbr) %>%
  arrange(encounter_id, .group_by=TRUE) %>%
  slice(1) %>%
  ungroup()

# specifying primary outcome of interest (binary - subset of `readmitted`)
diabetes$early_readmit <- as.factor(ifelse(diabetes$readmitted=="<30", 1, 0))

# changing other variables according to ids:
# discharge
diabetes$discharge_to_home <- as.factor(ifelse(as.numeric(diabetes$discharge_disposition_id)==1, 1, 0))
# admission source
diabetes$admission_source <- as.factor(with(diabetes,
                                    ifelse(as.numeric(admission_source_id)==7, "ER",
                                    ifelse(as.numeric(admission_source_id) %in% 1:2, "P/C", "Other"))
# race
diabetes$race_group <- as.factor(with(diabetes,
                                ifelse(race=="AfricanAmerican", "Black",
                                ifelse(race=="Caucasian", "White",
                                ifelse(is.na(race), "Missing/Unknown", "Other")))))
# age
lt_30 <- c("[0-10)", "[10-20)", "[20-30)")
bet_30_60 <- c("[30-40)", "[40-50)", "[50-60)")
diabetes$age_group <- as.factor(with(diabetes,
                                ifelse(age %in% lt_30, "0-30",
                                ifelse(age %in% bet_30_60, "30-60", ">60"))))
# medical specialty
surgery <- c("[0-10)", "[10-20)", "[20-30)")
diabetes$med_specialty <- as.factor(with(diabetes,
                                ifelse(medical_specialty=="InternalMedicine", "InternalMedicine",
                                ifelse(medical_specialty=="Cardiology", "Cardiology",
                                ifelse(startsWith(as.character(diabetes$medical_specialty), "Surg"), "Su
                                ifelse(medical_specialty=="Family/GeneralPractice", "Family/GeneralPract
                                ifelse(medical_specialty=="PhysicianNotFound", "Missing/Unknown",
                                ifelse(is.na(medical_specialty), "Missing/Unknown", "Other")))))))))

# creating HbA1c variable (encounter type)
  # 1. no test performed,
  # 2. test performed and in normal range,
  # 3. test performed with result >8%, and no change in diabetic medications
  # 4. test performed with result >8%, and diabetic medication was changed
diabetes$encounter_type <- as.factor(with(diabetes,
                                    ifelse(A1Cresult=="None", 1,
```

```
                                    ifelse(A1Cresult=="Norm", 2,
                                    ifelse(A1Cresult==">8" & change=="Ch", 3,
                                    ifelse(A1Cresult==">8" & change=="No", 4, "Other"))))))


# % missing data
diabetes_na <- apply(diabetes, 2, function(x) sum(is.na(x))/nrow(diabetes)) # for all columns
diabetes_na[diabetes_na != 0] # all columns where % > 0

# count of unique values for each variable
diabetes_unique <- sapply(lapply(diabetes, unique), length)
diabetes_unique[diabetes_unique < 2]

# filtering data to those who could be readmitted and first encounter
diabetes <- diabetes %>% filter(!(discharge_disposition_id %in% c(11, 13, 14, 19, 20, 21)),
                                !(gender=="Unknown/Invalid"))
diabetes <- diabetes %>%
  group_by(patient_nbr) %>%
  arrange(encounter_id, .group_by=TRUE) %>%
  slice(1) %>%
  ungroup()

# diagnoses: https://en.wikipedia.org/wiki/List_of_ICD-9_codes
icd_type <- vector(length=999, mode="character")
icd_type[1:139] <- "infectious"
icd_type[140:239] <- "neoplasms"
icd_type[240:279] <- "endocrine"
icd_type[250] <- "diabetes"
icd_type[280:289] <- "blood"
icd_type[290:319] <- "mental"
icd_type[320:389] <- "nerves"
icd_type[390:459] <- "circulatory"
icd_type[460:519] <- "respiratory"
icd_type[520:579] <- "digestive"
icd_type[580:629] <- "genitounirany"
icd_type[630:679] <- "pregnancy"
icd_type[680:709] <- "skin"
icd_type[710:739] <- "musculoskeletal"
icd_type[740:759] <- "congenital_anomalies"
icd_type[760:779] <- "perinatal"
icd_type[780:799] <- "ill_defined"
icd_type[800:999] <- "injury_poison"

# creating new variables
diabetes$diag_1_cat <- icd_type[as.numeric(diabetes$diag_1)]
diabetes$diag_1_cat[substr(diabetes$diag_1, 1, 1) %in% c('V','E')] <- "external"
diabetes$diag_1_cat <- as.factor(with(diabetes,
                              ifelse(diag_1_cat=="circulatory", "Circulatory",
                              ifelse(diag_1_cat=="respiratory", "Respiratory",
                              ifelse(diag_1_cat=="digestive", "Digestive",
                              ifelse(diag_1_cat=="diabetes", "Diabetes",
                              ifelse(diag_1_cat=="injury_poison", "Injury/Poisoning",
                              ifelse(diag_1_cat=="musculoskeletal", "Musculoskeletal",
```

```
                              ifelse(diag_1_cat=="genitourinary", "Genitourinary",
                              ifelse(diag_1_cat=="neoplasms", "Neoplasms", "Other")))))))))))

diabetes$diag_2_cat <- icd_type[as.numeric(diabetes$diag_2)]
diabetes$diag_2_cat[substr(diabetes$diag_2, 1, 1) %in% c('V','E')] <- "external"
diabetes$diag_2_cat <- as.factor(with(diabetes,
                              ifelse(diag_2_cat=="circulatory", "Circulatory",
                              ifelse(diag_2_cat=="respiratory", "Respiratory",
                              ifelse(diag_2_cat=="digestive", "Digestive",
                              ifelse(diag_2_cat=="diabetes", "Diabetes",
                              ifelse(diag_2_cat=="injury_poison", "Injury/Poisoning",
                              ifelse(diag_2_cat=="musculoskeletal", "Musculoskeletal",
                              ifelse(diag_2_cat=="genitourinary", "Genitourinary",
                              ifelse(diag_2_cat=="neoplasms", "Neoplasms", "Other")))))))))))

diabetes$diag_3_cat <- icd_type[as.numeric(diabetes$diag_3)]
diabetes$diag_3_cat[substr(diabetes$diag_3, 1, 1) %in% c('V','E')] <- "external"
diabetes$diag_3_cat <- as.factor(with(diabetes,
                              ifelse(diag_3_cat=="circulatory", "Circulatory",
                              ifelse(diag_3_cat=="respiratory", "Respiratory",
                              ifelse(diag_3_cat=="digestive", "Digestive",
                              ifelse(diag_3_cat=="diabetes", "Diabetes",
                              ifelse(diag_3_cat=="injury_poison", "Injury/Poisoning",
                              ifelse(diag_3_cat=="musculoskeletal", "Musculoskeletal",
                              ifelse(diag_3_cat=="genitourinary", "Genitourinary",
                              ifelse(diag_3_cat=="neoplasms", "Neoplasms", "Other")))))))))))

# removing redundant &/or incomplete variables/data
diabetes <- diabetes %>% select(-c(readmitted, discharge_disposition_id, admission_source_id, race, age
                                   medical_specialty, weight, payer_code,
                                   examide, citoglipton, acetohexamide, troglitazone, tolbutamide,
                                   glipizide.metformin, glimepiride.pioglitazone,
                                   metformin.rosiglitazone, metformin.pioglitazone,
                                   diag_1, diag_2, diag_3))

# dropping unused levels
diabetes <- droplevels(diabetes, except=15:28)

# replacing NA (for factored variables)
diabetes$med_specialty <- factor(diabetes$med_specialty, levels=c(levels(diabetes$med_specialty)))
diabetes$med_specialty[is.na(diabetes$med_specialty)] <- "Missing/Unknown"
diabetes$race_group <- factor(diabetes$race_group, levels=c(levels(diabetes$race_group), "Missing/Unknow
diabetes$race_group[is.na(diabetes$race_group)] <- "Missing/Unknown"

# variable summary
CreateTableOne(data=diabetes)
```

```
## Data for Models

# variables of importance according to paper
m_vars <- c("early_readmit", "encounter_type", "race_group", "gender", "age_group",
            "discharge_to_home", "admission_source", "time_in_hospital", "med_specialty",
            "num_lab_procedures", "num_procedures", "num_medications", "number_outpatient",
            "number_emergency", "number_inpatient", "number_diagnoses", "diabetesMed", "diag_1_cat")
```

```r
# new data frame
# 10 rows with missing values removed
diabetes_mdata <- na.omit(diabetes[, which(colnames(diabetes) %in% m_vars)])

# variable summary
CreateTableOne(data=diabetes_mdata)
```

```r
set.seed(47)

## Splitting the Data (25/75)

# 0.25 sample of row indices
test <- sample(1:nrow(diabetes_mdata), floor(0.25*nrow(diabetes_mdata)), replace=FALSE)
# test set
diabetes_test <- diabetes_mdata[test, ]
# train set
diabetes_train <- diabetes_mdata[-test, ]
```

```r
### Logistic Regression Model

# first main effects model
d_lm_me <- glm(early_readmit ~ encounter_type + race_group + gender + age_group +
                 discharge_to_home + admission_source + time_in_hospital + med_specialty +
                 num_lab_procedures + num_procedures + num_medications + number_outpatient +
                 number_emergency + number_inpatient + number_diagnoses + diabetesMed + diag_1_cat,
               data=diabetes_train, family=binomial(link="logit"))

## variable selection: stepwise elimination

#step(na.omit(d_lm_me))
# removes: encounter_type, race_group, gender, num_procedures + num_medications + number_outpatient
d_lm_me_step <- glm(early_readmit ~ age_group + discharge_to_home + admission_source +
                      time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
                      number_inpatient + number_diagnoses + diabetesMed + diag_1_cat,
                    data=diabetes_train, family=binomial(link="logit"))

# chosen main effects model
# introducing: encounter_type, race_group (due to significance in paper)
d_lm_me2 <- glm(early_readmit ~ encounter_type + race_group + age_group + discharge_to_home + admission_
                      time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
                      number_inpatient + number_diagnoses + diabetesMed + diag_1_cat,
                    data=diabetes_train, family=binomial(link="logit")) # 13 main effects

AIC(d_lm_me) # 30956.67
AIC(d_lm_me_step) # 30953.2
AIC(d_lm_me2) # 30948.89

## interaction term selection (process done separately)

# significant interactions from paper (8):
  # time_in_hospital: discharge_to_home, diag_1_cat, med_specialty
  # discharge_to_home: diag_1_cat, med_specialty, race_group,
  # diag_1_cat: encounter_type
  # med_specialty: age_group
```

```r
# significant interactions found (21): includes all those from the paper
  # time_in_hospital: discharge_to_home, diag_1_cat, med_specialty, num_lab_procedures, number_inpatien
  # discharge_to_home: diag_1_cat, med_specialty, race_group, num_lab_procedures, number_emergency, num
  # diag_1_cat: encounter_type, age_group, num_lab_procedures
  # med_specialty: age_group, admission_source, num_lab_procedures
  # age_group: admission_source, number_diagnoses
  # admission_source: num_lab_procedures, diabetesMed

# interactions to keep (12): removed seemingly redundant and overly complex interactions
  # time_in_hospital: discharge_to_home, diag_1_cat, med_specialty, number_inpatient
  # discharge_to_home: diag_1_cat, med_specialty, race_group, number_diagnoses
  # diag_1_cat: encounter_type
  # med_specialty: age_group
  # age_group: number_diagnoses
  # admission_source: diabetesMed

# final model (including interactions and main effects chosen above)
d_lm <- glm(early_readmit ~ time_in_hospital*(discharge_to_home + diag_1_cat + med_specialty + number_i
                discharge_to_home*(diag_1_cat + med_specialty + race_group + number_diagnoses) +
                age_group*(med_specialty + number_diagnoses) + diag_1_cat*encounter_type +
                admission_source*diabetesMed + num_lab_procedures + number_emergency + number_inpatie
              data=diabetes_train, family=binomial(link="logit")) # 25 terms (13 main effects, 12 int

AIC(d_lm) # 30927.94 <- best
```

```r
## Interaction Term Selection

# significant interactions from paper (8):
  # time_in_hospital: discharge_to_home, diag_1_cat, med_specialty
  # discharge_to_home: diag_1_cat, med_specialty, race_group,
  # diag_1_cat: encounter_type
  # med_specialty: age_group

# PROCESS: (testing each possibly significant interaction)
# encounter_type
# significant: encounter_type*diag_1_cat(p)
# summary(glm(early_readmit ~ (race_group + age_group + discharge_to_home + admission_source +
#                       time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*encounter_type,
#                     data=diabetes_train, family=binomial(link="logit")))
# race_group
# significant: race_group*discharge_to_home(p)
# summary(glm(early_readmit ~ (encounter_type + age_group + discharge_to_home + admission_source +
#                       time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*race_group,
#                     data=diabetes_train, family=binomial(link="logit")))
# age_group
# significant: age_group*number_diagnoses, age_group*admission_source
# summary(glm(early_readmit ~ (encounter_type + race_group + discharge_to_home + admission_source +
#                       time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*age_group,
#                     data=diabetes_train, family=binomial(link="logit")))
# discharge_to_home
```

```
# significant: discharge_to_home*num_lab_procedures, discharge_to_home*number_emergency, discharge_to_h
# summary(glm(early_readmit ~ (encounter_type + race_group + age_group + admission_source +
#                       time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*discharge_to_ho
#                     data=diabetes_train, family=binomial(link="logit")))
# admission_source
# significant: admission_source*med_specialty (too complex), admission_source*num_lab_procedures, admis
# summary(glm(early_readmit ~ (encounter_type + race_group + age_group + discharge_to_home +
#                       time_in_hospital + med_specialty + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*admission_sourc
#                     data=diabetes_train, family=binomial(link="logit")))
# time_in_hospital
# significant: time_in_hospital*discharge_to_home(p), time_in_hospital*num_lab_procedures, time_in_hosp
# summary(glm(early_readmit ~ (encounter_type + race_group + age_group + discharge_to_home +
#                       admission_source + med_specialty + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*time_in_hospita
#                     data=diabetes_train, family=binomial(link="logit")))
# med_specialty
# significant: med_specialty*num_lab_procedures, med_specialty*discharge_to_home(p), med_specialty*age_
# summary(glm(early_readmit ~ (encounter_type + race_group + age_group + discharge_to_home +
#                       admission_source + time_in_hospital + num_lab_procedures + number_emergency +
#                       number_inpatient + number_diagnoses + diabetesMed + diag_1_cat)*med_specialty,
#                     data=diabetes_train, family=binomial(link="logit")))
# diag_1_cat
# significant: diag_1_cat*discharge_to_home(p), diag_1_cat*num_lab_procedures, diag_1_cat*age_group (an
# summary(glm(early_readmit ~ (encounter_type + race_group + age_group + discharge_to_home +
#                       admission_source + time_in_hospital + med_specialty + num_lab_procedures +
#                         number_emergency + number_inpatient + number_diagnoses + diabetesMed)*diag_1_
#                     data=diabetes_train, family=binomial(link="logit")))

# significant interactions found (21): includes all those from the paper
  # time_in_hospital: discharge_to_home, diag_1_cat, med_specialty, num_lab_procedures, number_inpatien
  # discharge_to_home: diag_1_cat, med_specialty, race_group, num_lab_procedures, number_emergency, num
  # diag_1_cat: encounter_type, age_group, num_lab_procedures
  # med_specialty: age_group, admission_source, num_lab_procedures
  # age_group: admission_source, number_diagnoses
  # admission_source: num_lab_procedures, diabetesMed

# interactions to keep (12): removed seemingly redundant and overly complex interactions
  # time_in_hospital: discharge_to_home, diag_1_cat, med_specialty, number_inpatient
  # discharge_to_home: diag_1_cat, med_specialty, race_group, number_diagnoses
  # diag_1_cat: encounter_type
  # med_specialty: age_group
  # age_group: number_diagnoses
  # admission_source: diabetesMed

# corresponding model
# d_lm <- glm(early_readmit ~ time_in_hospital*(discharge_to_home + diag_1_cat + med_specialty + number
#                   discharge_to_home*(diag_1_cat + med_specialty + race_group + number_diagnoses) +
#                   age_group*(med_specialty + number_diagnoses) + diag_1_cat*encounter_type +
#                   admission_source*diabetesMed + num_lab_procedures + number_emergency + number_inpat
#                 data=diabetes_train, family=binomial(link="logit"))
# AIC: 30927.94
```

```
## Final Model Selection

# PROCESS 1: (taking out possibly insignificant terms)
# out: discharge*diag
# summary(glm(early_readmit ~ time_in_hospital*(discharge_to_home + diag_1_cat + med_specialty + number
#                   discharge_to_home*(med_specialty + race_group + number_diagnoses) +
#                   age_group*(med_specialty + number_diagnoses) + diag_1_cat*encounter_type +
#                   admission_source*diabetesMed + num_lab_procedures + number_emergency + number_inpat
#                 data=diabetes_train, family=binomial(link="logit")))
# AIC: 30929 <- increased from original
# out: admission_source*diabetesMed
# summary(glm(early_readmit ~ time_in_hospital*(discharge_to_home + diag_1_cat + med_specialty + number
#                   discharge_to_home*(diag_1_cat + med_specialty + race_group + number_diagnoses) +
#                   age_group*(med_specialty + number_diagnoses) + diag_1_cat*encounter_type +
#                   num_lab_procedures + number_emergency + number_inpatient,
#                 data=diabetes_train, family=binomial(link="logit")))
# AIC: 30966 <- increased from original
# out: num_lab_procedures + number_emergency + number_inpatient
# summary(glm(early_readmit ~ time_in_hospital*(discharge_to_home + diag_1_cat + med_specialty + number
#                   discharge_to_home*(diag_1_cat + med_specialty + race_group + number_diagnoses) +
#                   age_group*(med_specialty + number_diagnoses) + diag_1_cat*encounter_type,
#                 data=diabetes_train, family=binomial(link="logit")))
# AIC: 30988 <- increased from original

# PROCESS 2: stepwise elimination
# step(d_lm)
# early_readmit ~ time_in_hospital + discharge_to_home + diag_1_cat +
#     med_specialty + number_inpatient + race_group + number_diagnoses +
#     age_group + admission_source + diabetesMed + num_lab_procedures +
#     number_emergency + time_in_hospital:discharge_to_home + time_in_hospital:diag_1_cat +
#     time_in_hospital:med_specialty + time_in_hospital:number_inpatient +
#     discharge_to_home:diag_1_cat + discharge_to_home:med_specialty +
#     discharge_to_home:race_group + discharge_to_home:number_diagnoses +
#     med_specialty:age_group + number_diagnoses:age_group + admission_source:diabetesMed

# since we can't remove terms not included here, the best model (based on AIC) is given by d_lm (above)
```

```
## Model Coefficients: Logistic Regression

# putting coefficients and OR into data frame
d_lm_coeffs <- coef(d_lm)
d_lm_df <- as.data.frame(d_lm_coeffs)
d_lm_df <- cbind(predictor=rownames(d_lm_df), d_lm_df)
rownames(d_lm_df) <- 1:nrow(d_lm_df)
d_lm_df <- d_lm_df %>%
  mutate(OR=round(exp(d_lm_coeffs), 5))
colnames(d_lm_df) <- c("Term", "Coefficient", "OR")
```

```
## Evaluating Model Fit/Performance: Logistic Regression

# model predictions
d_lm_probs <- predict(d_lm, newdata=diabetes_test, type="response") # probabilities
```

```r
# binary predictions (rounded)
d_lm_pred <- ifelse(d_lm_probs > 0.5, 1, 0) # or: d_lm_pred <- ifelse(round(d_lm_pred), 1, 0)
  # num positive predictions = 15, num positive observations = 1533
  sum(d_lm_pred)
  # num negative predictions = 17479, num negative observations = 15961
  nrow(diabetes_test)-sum(d_lm_pred)
# num correct predictions = 15952
sum(as.factor(d_lm_pred) == diabetes_test$early_readmit)
# classification prediction accuracy = 91.19%, misclassification error rate = 8.81%.
mean(d_lm_pred == diabetes_test$early_readmit) # -> 15952/nrow(diabetes_test)
# confusion matrix
confusionMatrix(data=as.factor(d_lm_pred), diabetes_test$early_readmit, positive="1")

# ROC curve (plot separate)
d_lm_pred_roc <- prediction(d_lm_probs, diabetes_test$early_readmit)
d_lm_perf <- performance(d_lm_pred_roc, measure="tpr", x.measure="fpr")
# AUC (>0.7 indicates good model performance)
d_lm_auc <- performance(d_lm_pred_roc, measure="auc")
d_lm_auc <- d_lm_auc@y.values[[1]] # auc = 0.63
```

```r
### Classification Tree Model

# first tree model
d_ct <- rpart(early_readmit ~ encounter_type + race_group + age_group + discharge_to_home +
                admission_source + time_in_hospital + med_specialty + num_lab_procedures +
                number_emergency + number_inpatient + number_diagnoses + diabetesMed + diag_1_cat,
              method="class", data=diabetes_train, minsplit=4, minbucket=1, maxdepth=30)
# summaries
summary(d_ct)
printcp(d_ct)
# root node error (0.0905) equivalent to the proportion of positive observations in the training data (
```

```r
## Evaluating Model Fit/Performance: Classification Tree

# model predictions
d_ct_probs <- predict(d_ct, newdata=diabetes_test) # probabilities
# binary/positive predictions (rounded)
d_ct_pred <- as.factor(round(d_ct_probs[, 2]))
levels(d_ct_pred) <- c(levels(d_ct_pred), "1")
  # num positive predictions = 0, num positive observations = 1533
  sum(as.numeric(d_ct_pred)-1)
  # num negative predictions = 17494, num negative observations = 15961
  nrow(diabetes_test)-sum(as.numeric(d_ct_pred)-1)
# num correct predictions = 15961
sum(d_ct_pred == diabetes_test$early_readmit)
# classification prediction accuracy = 91.24%, misclassification error rate = 8.76%.
mean(d_ct_pred == diabetes_test$early_readmit) # -> 15961/nrow(diabetes_test)
# confusion matrix
confusionMatrix(data=d_ct_pred, diabetes_test$early_readmit, positive="1")

# ROC curve (plot separate)
d_ct_pred_roc <- prediction(d_ct_probs[, 2], diabetes_test$early_readmit)
d_ct_perf <- performance(d_ct_pred_roc, measure="tpr", x.measure="fpr")
# AUC (>0.7 indicates good model performance, <0.6 indicates unsatisfactory model performance)
```

```r
d_ct_auc <- performance(d_ct_pred_roc, measure="auc")
d_ct_auc <- d_ct_auc@y.values[[1]] # auc = 0.5
```

```r
# NOTE: since only ~9.05% of the data contains positive predictions, the data is very unbalanced (resul

## Balancing Data

# undersampling
d_train_balanced <- ovun.sample(early_readmit~., data=diabetes_train, method="under",
                                N=sum(diabetes_train$early_readmit==1)*2, seed=47)$data
table(d_train_balanced$early_readmit) # newly balanced data
```

```r
## Classification Tree Model - Balanced Data

# new tree model
d_ct_balanced <- rpart(early_readmit ~ encounter_type + race_group + age_group + discharge_to_home +
                         admission_source + time_in_hospital + med_specialty + num_lab_procedures +
                         number_emergency + number_inpatient + number_diagnoses + diabetesMed + diag_1_
                       method="class", data=d_train_balanced, minsplit=4, minbucket=1, maxdepth=30)
# summary
summary(d_ct_balanced)
# tree plot
# alternative to plot(d_ct_balanced), text(d_ct_balanced):
rpart.plot(d_ct_balanced,
           main="Figure 7: Classification Tree Model for Balanced Data")
# complexity
printcp(d_ct_balanced)
#plotcp(d_ct_balanced)
```

```r
## Evaluating Model Fit/Performance: Classification Tree - Balanced Data

# model predictions
d_ct_balanced_probs <- predict(d_ct_balanced, newdata=diabetes_test) # probabilities
# binary/positive predictions (rounded)
d_ct_balanced_pred <- as.factor(round(d_ct_balanced_probs[, 2]))
levels(d_ct_balanced_pred) <- c(levels(d_ct_balanced_pred), "1")
  # num positive predictions = 7678, num positive observations = 1533
  sum(as.numeric(d_ct_balanced_pred)-1)
  # num negative predictions = 9816, num negative observations = 15961
  nrow(diabetes_test)-sum(as.numeric(d_ct_balanced_pred)-1)
# num correct predictions = 10175
sum(d_ct_balanced_pred == diabetes_test$early_readmit)
# classification prediction accuracy = 58.16%, misclassification error rate = 41.84%.
mean(d_ct_balanced_pred == diabetes_test$early_readmit) # -> 10175/nrow(diabetes_test)
# confusion matrix
confusionMatrix(data=d_ct_balanced_pred, diabetes_test$early_readmit, positive="1")

# ROC curve (plot separate)
d_ct_balanced_pred_roc <- prediction(d_ct_balanced_probs[, 2], diabetes_test$early_readmit)
d_ct_balanced_perf <- performance(d_ct_balanced_pred_roc, measure="tpr", x.measure="fpr")
# AUC (>0.7 indicates good model performance, <0.6 indicates unsatisfactory model performance)
d_ct_balanced_auc <- performance(d_ct_balanced_pred_roc, measure="auc")
d_ct_balanced_auc <- d_ct_balanced_auc@y.values[[1]] # auc = 0.59
```

```
## Tables

# Data Sets
dataset <- c("Complete", "Test", "Train", "Balanced Train")
obs <- c(nrow(diabetes_mdata), nrow(diabetes_test), nrow(diabetes_train), nrow(d_train_balanced))
prop <- c(obs[1]/obs[1], obs[2]/obs[1], obs[3]/obs[1], obs[4]/obs[1])
pos_obs <- c(sum(diabetes_mdata$early_readmit==1), sum(diabetes_test$early_readmit==1),
             sum(diabetes_train$early_readmit==1), sum(d_train_balanced$early_readmit==1))
pos_obs_prop <- pos_obs/obs
neg_obs <- c(sum(diabetes_mdata$early_readmit!=1), sum(diabetes_test$early_readmit!=1),
             sum(diabetes_train$early_readmit!=1), sum(d_train_balanced$early_readmit!=1))
neg_obs_prop <- neg_obs/obs
datasets <- data.frame(dataset, obs, prop, pos_obs, neg_obs) %>%
  rename("Dataset"=dataset, "Observations"=obs, "Proportion"=prop,
         "Positive"=pos_obs, "Negative"=neg_obs)

# Models (Performance)
Model <- c("Logistic", "Classification Tree", "Classification Tree (Balanced)")
TP <- c(3, 0, 946)
FP <- c(12, 0, 6732)
TN <- c(15949, 15961, 9229)
FN <- c(1530, 1533, 587)
Sensitivity <- c(0.0019569, 0, 0.61709)
Specificity <- c(0.9992482, 1, 0.57822)
Accuracy <- c(0.9119, 0.9124, 0.5816)
AUC <- c(0.6339094, 0.5, 0.59)
model_perf <- data.frame(Model, TP, FP, TN, FN, Sensitivity, Specificity, Accuracy, AUC)
```

```
## ROC curves

# logistic regression model
plot(d_lm_perf,
     main="Figure 8.1: ROC Curve - Logistic Regression Model",
     xlab="False Positive Rate",
     ylab="True Positive Rate")
# classification tree model
plot(d_ct_perf,
     main="Figure 8.2: ROC Curve - Classification Tree Model 1",
     xlab="False Positive Rate",
     ylab="True Positive Rate")
# classification tree model (balanced data)
plot(d_ct_balanced_perf,
     main="Figure 8.3: ROC Curve - Classification Tree Model 2",
     #sub="(Balanced Data)",
     xlab="False Positive Rate",
     ylab="True Positive Rate")
```

Sources:

- https://arxiv.org/pdf/1412.6980.pdf
- https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
- https://towardsdatascience.com/understanding-gradient-descent-and-adam-optimization-472ae8a78c10
- https://downloads.hindawi.com/journals/bmri/2014/781670.pdf