

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

## Group 9 Project Report – Teach For America Case Study

### Introduction

In this project, we worked with the Teach For America admissions dataset to predict if an applicant will complete the admissions process. The target variable is Completed Admissions Process, coded as 0 and 1. A value of 1 means the applicant followed through with the full process. A value of 0 means they did not complete it.

The main goal of our analysis was to use the methods we learned in class to build several predictive models and compare how well they perform. We focused on:

- kNN (k nearest neighbors)
- Naive Bayes
- Decision Tree with C5 point zero
- SVM (Support Vector Machine) with radial kernel
- ANN (Artificial Neural Network)

One big challenge is that the data is very imbalanced. Around eighty percent of the observations have Completed Admissions Process = 1. Only about twenty percent have 0. Because of this, a model that simply predicts everyone as 1 already gets about 0.8000 accuracy. So if we only look at accuracy, many models will look good but actually do nothing useful for the minority group.

For that reason we looked at accuracy, kappa, sensitivity, specificity, and confusion matrices. We also tried both baseline models and tuned models using cross validation. In the end we chose the best performing method and gave some interpretation of the inputs

### 1. Data Preparation

#### Reading & Selecting Variables:

We started by loading the Excel file using `read_excel`. The original dataset contains many columns. We created a new data frame called `CleanedTFAData.df` and selected variables that we felt had a clear connection to the admissions process and that we could handle with the tools from class.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

The main variables we kept were:

- App Started Year
- Cumulative GPA
- Is Math Sci or Eng Major Minor
- School Selectivity
- Essay 1 Length, Essay 2 Length, Essay 3 Length
- Essays Unique Words and Essays Sentiment
- Sign up Date, Started Date, Submitted Date, Application Deadline
- Region Preference Level
- Attended Event
- Completed Admissions Process

Our idea was that academic strength, school background, essays, timing, and engagement with Teach For America events might all be related to whether an applicant stays in the process until the end.

```
> # New dataframe w/o unnecessary variables
> CleanedTFADATA.df <- TFADATA.df[,c("App Started Year (RTAT)", "Cumulative GPA",
+ "Is Math, sci, or Eng Major Minor", "School selectivity",
+ "Essay 1 Length", "Essay 2 Length", "Essay 3 Length", "Essays Unique words",
+ "Essays Sentiment", "Sign-up Date", "Started Date", "Application Deadline", "Submitted Date",
+ "Region Preference Level", "Attended Event", "Completed Admissions Process")]
> head(CleanedTFADATA.df)
# A tibble: 6 × 16
`App Started Year (RTAT)` `Cumulative GPA` `Is Math, sci, or Eng Major Minor` `School selectivity` <dbl> <dbl> <chr>
1 2015 2.63 0 less_selective
2 2016 3.18 0 more_selective
3 2016 2.78 0 less_selective
4 2015 2.69 0 more_selective
5 2016 0 1 unknown
6 2015 2.75 0 selective
# i 12 more variables: `Essay 1 Length` <dbl>, `Essay 2 Length` <dbl>, `Essay 3 Length` <dbl>,
# `Essays Unique Words` <dbl>, `Essays Sentiment` <dbl>, `Sign-up Date` <dttm>,
# `Started Date` <dttm>, `Application Deadline` <dttm>, `Submitted Date` <dttm>,
# `Region Preference Level` <dbl>, `Attended Event` <dbl>, `Completed Admissions Process` <dbl>
> str(CleanedTFADATA.df)
tibble [74,839 × 16] (S3:tbl_df/tbl/data.frame)
$ App Started Year (RTAT) : num [1:74839] 2015 2016 2016 2015 2016 ...
$ Cumulative GPA : num [1:74839] 2.63 3.18 2.78 2.69 0 2.75 3.1 3.14 3.34 2.83 ...
$ Is Math, sci, or Eng Major Minor: num [1:74839] 0 0 0 1 0 0 0 0 0 ...
$ School selectivity : chr [1:74839] "less_selective" "more_selective" "less_selective" "more_selective" ...
$ Essay 1 Length : num [1:74839] 207 288 110 219 47 162 190 226 294 297 ...
$ Essay 2 Length : num [1:74839] 295 289 82 88 78 265 290 212 295 290 ...
$ Essay 3 Length : num [1:74839] 291 274 23 227 20 168 295 153 289 289 ...
$ Essays Unique words : num [1:74839] 378 424 119 280 86 269 331 250 351 419 ...
$ Essays Sentiment : num [1:74839] -0.6316 -0.328 0.1958 0.0722 0.1778 ...
$ Sign-up Date : POSIXct[1:74839], format: "2014-08-21" "2015-08-10" "2015-10-23" ...
$ Started Date : POSIXct[1:74839], format: "2014-08-21" "2015-08-10" "2015-10-23" ...
$ Application Deadline : POSIXct[1:74839], format: "2014-10-24" "2015-08-21" "2015-10-30" ...
$ Submitted Date : POSIXct[1:74839], format: "2014-10-23" "2015-08-21" "2015-10-24" ...
$ Region Preference Level : num [1:74839] 1 1 1 1 2 1 1 1 1 1 ...
$ Attended Event : num [1:74839] 0 0 0 1 0 1 0 0 1 0 ...
$ Completed Admissions Process : num [1:74839] 0 1 1 1 1 1 1 1 1 1 ...
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

### **Creating Time-Based & Essay-Based Variables:**

The dataset gives several dates, but raw date stamps are not easy for models to use. So we created new numeric variables that measure time gaps in days.

We used:

- days\_signup\_to\_start = Started Date minus Sign up Date
- days\_start\_to\_submit = Submitted Date minus Started Date
- days\_submit\_to\_deadline = Application Deadline minus Submitted Date
- days\_signup\_to\_submit = Submitted Date minus Sign up Date

The idea is that how long someone takes at each stage may show how serious or organized they are. For example, someone who submits right before the deadline might be different from someone who submits very early.

We then created two new essay features:

- avg\_essay\_length as the mean of Essay 1, Essay 2, and Essay 3 lengths
- essay\_density as average essay length divided by unique words plus one

The plus one in the denominator avoids division by zero in case unique words is zero. It makes the ratio stable. These features roughly capture how long the essays are and how repetitive or dense the writing might be.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
 Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
> # Days between signup and start
> CleanedTFADATA.df$days_signup_to_start <- as.numeric(
+   difftime(CleanedTFADATA.df$'Started Date', CleanedTFADATA.df$`Sign-up Date`, units = "days")
+ )
> # Days between start and submit
> CleanedTFADATA.df$days_start_to_submit <- as.numeric(
+   difftime(CleanedTFADATA.df$'Submitted Date', CleanedTFADATA.df$'Started Date', units = "days")
+ )
> # Days between submit and deadline
> CleanedTFADATA.df$days_submit_to_deadline <- as.numeric(
+   difftime(CleanedTFADATA.df$`Application Deadline`, CleanedTFADATA.df$`Submitted Date`, units = "days")
+ )
> # Days between signup and submit
> CleanedTFADATA.df$days_signup_to_submit <- as.numeric(
+   difftime(CleanedTFADATA.df$'Submitted Date', CleanedTFADATA.df$`Sign-up Date`, units = "days")
+ )
> # Average essay length
> CleanedTFADATA.df$avg_essay_length <- rowMeans(CleanedTFADATA.df[, c("Essay 1 Length", "Essay 2 Length", "Essay 3 Length")],
+ na.rm = TRUE
+ )
> # Essay density - avg length per unique word
> CleanedTFADATA.df$essay_density <- CleanedTFADATA.df$avg_essay_length / (CleanedTFADATA.df$`Essays Unique Words` + 1)
> # Top-choice region preference (focus on 1st choice - Convert others to zero)
> CleanedTFADATA.df$region_pref_1 <- ifelse(CleanedTFADATA.df$`Region Preference Level` == 1, 1, 0)
> # Submitted before deadline
> CleanedTFADATA.df$submitted_before_deadline <- ifelse(CleanedTFADATA.df$days_submit_to_deadline > 0, 1, 0)
> # Check the new variables
> summary(CleanedTFADATA.df[, c("days_signup_to_start",
+                                 "days_start_to_submit",
+                                 "days_submit_to_deadline",
+                                 "days_signup_to_submit",
+                                 "avg_essay_length",
+                                 "essay_density",
+                                 "region_pref_1",
+                                 "submitted_before_deadline")])
  days_signup_to_start days_start_to_submit days_submit_to_deadline days_signup_to_submit
Min.   : -246.000   Min.   : 0.00   Min.   :-879.00   Min.   : 0.00
1st Qu.:  0.000     1st Qu.: 1.00   1st Qu.: 0.00     1st Qu.: 1.00
Median :  0.000     Median : 8.00   Median : 2.00     Median : 9.00
Mean   :  8.002     Mean   : 22.81  Mean   : 10.58    Mean   : 30.82
3rd Qu.:  0.000     3rd Qu.: 28.00  3rd Qu.: 14.00    3rd Qu.: 37.00
Max.   : 516.000    Max.   : 897.00  Max.   : 214.00    Max.   : 897.00
  avg_essay_length essay_density region_pref_1 submitted_before_deadline
Min.   : 1.0       Min.   : 0.2381  Min.   :0.0000  Min.   : 0.0000
1st Qu.: 152.7     1st Qu.: 0.6444  1st Qu.:0.0000  1st Qu.: 0.0000
Median : 245.0     Median : 0.7135  Median :1.0000  Median :1.0000
Mean   : 212.0     Mean   : 0.7003  Mean   :0.7201  Mean   : 0.6301
3rd Qu.: 281.8     3rd Qu.: 0.7690  3rd Qu.:1.0000  3rd Qu.: 1.0000
Max.   : 309.0     Max.   : 15.8889  Max.   :1.0000  Max.   :1.0000
> head(CleanedTFADATA.df[, c("Sign-up Date", "Started Date", "Submitted Date", "Application Deadline",
+                            "days_signup_to_start", "days_start_to_submit", "days_submit_to_deadline",
+                            "avg_essay_length", "essay_density", "region_pref_1", "submitted_before_deadline")])
# A tibble: 6 x 11
  `Sign-up Date`    `Started Date`    `Submitted Date`    `Application Deadline` 
  <dttm>           <dttm>           <dttm>           <dttm>
1 2014-08-21 00:00:00 2014-08-21 00:00:00 2014-10-23 00:00:00 2014-10-24 00:00:00
2 2015-08-10 00:00:00 2015-08-10 00:00:00 2015-08-21 00:00:00 2015-08-21 00:00:00
3 2015-10-23 00:00:00 2015-10-23 00:00:00 2015-10-24 00:00:00 2015-10-30 00:00:00
4 2014-10-14 00:00:00 2014-10-14 00:00:00 2014-10-23 00:00:00 2014-10-24 00:00:00
5 2015-12-04 00:00:00 2015-12-04 00:00:00 2015-12-04 00:00:00 2016-01-15 00:00:00
6 2014-10-06 00:00:00 2014-10-06 00:00:00 2014-10-25 00:00:00 2014-10-24 00:00:00
# i 7 more variables: days_signup_to_start <dbl>, days_start_to_submit <dbl>,
# days_submit_to_deadline <dbl>, avg_essay_length <dbl>, essay_density <dbl>,
# region_pref_1 <dbl>, submitted_before_deadline <dbl>
```

## Region and deadline features:

From Region Preference Level we created region\_pref\_1. This is a simple dummy variable:

- 1 if Region Preference Level equals 1
- 0 otherwise

This captures whether the applicant has a clear first choice or not.

We also created submitted\_before\_deadline, set to 1 if days\_submit\_to\_deadline is greater than zero, and 0 otherwise. This variable tells us if someone submitted earlier than the deadline, which could show more planning and motivation.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

## Factorizing Variables & Scaling:

We converted categorical columns to factors so models like Naive Bayes can use them properly and immediately followed with scaling numeric variables with the scale function so that methods like kNN and SVM work better.

```
> # Factorize Categorical Variables
> CleanedTFAData.df$`App Started Year (RTAT)` <- as.factor(CleanedTFAData.df$`App Started Year (RTAT)` )
> CleanedTFAData.df$`Is Math, Sci, or Eng Major Minor` <- as.factor(CleanedTFAData.df$`Is Math, Sci, or Eng Major Minor` )
> CleanedTFAData.df$`Region Preference Level` <- as.factor(CleanedTFAData.df$`Region Preference Level` )
> CleanedTFAData.df$`Attended Event` <- as.factor(CleanedTFAData.df$`Attended Event` )
> CleanedTFAData.df$region_pref_1 <- as.factor(CleanedTFAData.df$region_pref_1)
> CleanedTFAData.df$submitted_before_deadline <- as.factor(CleanedTFAData.df$submitted_before_deadline)
> CleanedTFAData.df$`School selectivity` <- as.factor(CleanedTFAData.df$`School selectivity` )
> CleanedTFAData.df$`Completed Admissions Process` <- as.factor(CleanedTFAData.df$`Completed Admissions Process` )
> #Scale Data
> CleanedTFAData.df[sapply(CleanedTFAData.df, is.numeric)] <-
+   scale(CleanedTFAData.df[sapply(CleanedTFAData.df, is.numeric)])
> str(CleanedTFAData.df)
tibble [74,839 x 24] (S3:tbl_df/tbl/data.frame)
$ App Started Year (RTAT) : Factor w/ 2 levels "2015","2016": 1 2 2 1 2 1 1 1 1 1 ...
$ Cumulative GPA : num [1:74839] -0.94 0.0103 -0.6809 -0.8364 -5.4843 ...
$ Is Math, Sci, or Eng Major Minor: Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 ...
$ School Selectivity : Factor w/ 6 levels "least_selective"...: 2 3 2 3 6 5 2 5 4 4 ...
$ Essay 1 Length : num [1:74839] -0.0524 0.837 -1.1174 0.0794 -1.8092 ...
$ Essay 2 Length : num [1:74839] 0.959 0.895 -1.337 -1.273 -1.38 ...
$ Essay 3 Length : num [1:74839] 0.841 0.645 -2.251 0.103 -2.286 ...
$ Essays Unique Words : num [1:74839] 0.858 1.317 -1.725 -0.119 -2.054 ...
$ Essays Sentiment : num [1:74839] -2.303 -1.395 0.172 -0.197 0.119 ...
$ Sign-up Date : POSIXct[1:74839], format: "2014-08-21" "2015-08-10" "2015-10-23" "2014-10-14" ...
$ Started Date : POSIXct[1:74839], format: "2014-08-21" "2015-08-10" "2015-10-23" "2014-10-14" ...
$ Application Deadline : POSIXct[1:74839], format: "2014-10-24" "2015-08-21" "2015-10-30" "2014-10-24" ...
$ Submitted Date : POSIXct[1:74839], format: "2014-10-23" "2015-08-21" "2015-10-24" "2014-10-23" ...
$ Region Preference Level : Factor w/ 3 levels "1","2","3": 1 1 1 2 1 1 1 1 ...
$ Attended Event : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 2 1 ...
$ Completed Admissions Process : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 ...
$ days_signup_to_start : num [1:74839] -0.246 -0.246 -0.246 -0.246 -0.246 ...
$ days_start_to_submit : num [1:74839] 1.101 -0.324 -0.598 -0.378 -0.625 ...
$ days_submit_to_deadline : num [1:74839] -0.459 -0.507 -0.219 -0.459 1.507 ...
$ days_signup_to_submit : num [1:74839] 0.637 -0.392 -0.59 -0.432 -0.61 ...
$ avg_essay_length : num [1:74839] 0.632 0.865 -1.693 -0.41 -1.974 ...
$ essay_density : num [1:74839] -0.0236 -0.2745 -0.8619 -0.5589 -1.2103 ...
$ region_pref_1 : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 2 ...
$ submitted_before_deadline : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 2 2 1 1 ...
```

## 2. Model 1 – kNN (k nearest neighbors)

We used stratified sampling to create an 80 percent training set and 20 percent testing set. This keeps the same class proportions in both sets.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
 Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
> # Predictive Variables
> Pred_Variables <- c("Cumulative GPA",
+                         "School selectivity",
+                         "Is Math, Sci, or Eng Major Minor",
+                         "Essays Sentiment",
+                         "avg_essay_length", "essay_density",
+                         "days_signup_to_start", "days_start_to_submit",
+                         "days_submit_to_deadline", "days_signup_to_submit",
+                         "Attended Event", "region_pref_1",
+                         "submitted_before_deadline", "Region Preference Level")
> # Revert target variable back to simple 0/1 factor (not scaled)
> CleanedTFAData.df$`Completed Admissions Process` <- as.factor(TFAData.df$`Completed Admissions Process`)
> # 80/20 stratified split
> idx <- createDataPartition(y = CleanedTFAData.df$`Completed Admissions Process`,
+                             p = 0.8, list = FALSE)
> # Training and testing sets
> train.df <- CleanedTFAData.df[idx, Pred_variables]
> test.df <- CleanedTFAData.df[-idx, Pred_variables]
> # Labels for train/test
> train_labels <- CleanedTFAData.df$`Completed Admissions Process`[idx]
> test_labels <- CleanedTFAData.df$`Completed Admissions Process`[-idx]
> # Convert all factor columns in train/test to numeric
> train.df[] <- lapply(train.df, function(x) if(is.factor(x)) as.numeric(x) else x)
> test.df[] <- lapply(test.df, function(x) if(is.factor(x)) as.numeric(x) else x)
> # Check label distribution (in proportion)
> proportions(table(train_labels))
train_labels
      0      1 
0.1925441 0.8074559 
> proportions(table(CleanedTFAData.df$`Completed Admissions Process`))

      0      1 
0.1925467 0.8074533
```

We used kNN as our first model. We tried k values from 1 to 10 and selected k = 4 because it gave the best balance. We saw that kNN had:

- Accuracy around 0.7583
- Kappa around 0.0724
- Sensitivity around 0.9010
- Specificity around 0.1596

```
> # Knn → K = 4
> Knn_K <- knn(train = train.df, test = test.df, cl = train_labels, k = 4)
> confusionMatrix(Knn_K, factor(test_labels), positive = "1")
Confusion Matrix and Statistics

             Reference
Prediction      0      1
      0   460  1196
      1  2422 10889

Accuracy : 0.7583
95% CI  : (0.7513, 0.7651)
No Information Rate : 0.8074
P-Value [Acc > NIR] : 1

Kappa : 0.0724

McNemar's Test P-Value : <2e-16

Sensitivity : 0.9010
Specificity : 0.1596
Pos Pred Value : 0.8180
Neg Pred Value : 0.2778
Prevalence : 0.8074
Detection Rate : 0.7275
Detection Prevalence : 0.8894
Balanced Accuracy : 0.5303

'Positive' Class : 1
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

It was better than a random guess, but still struggled because of the imbalance. We also ran a ten fold sensitivity check with lapply and the sensitivity averaged around 0.2778.

```
> #####Create 10 folds#####
> CleanedCabData.folds <- createFolds(CleanedTFAData.df$"Completed Admissions Process", k = 10)
> str(CleanedCabData.folds) #View 10 folds
List of 10
$ Fold01: int [1:7484] 24 26 28 32 33 34 39 45 50 68 ...
$ Fold02: int [1:7484] 7 8 9 19 48 65 69 82 98 100 ...
$ Fold03: int [1:7483] 3 5 6 36 40 51 74 79 80 111 ...
$ Fold04: int [1:7484] 13 18 22 23 27 31 38 47 58 62 ...
$ Fold05: int [1:7484] 10 15 25 37 41 52 59 60 70 94 ...
$ Fold06: int [1:7484] 2 35 54 64 71 81 83 96 109 119 ...
$ Fold07: int [1:7484] 1 16 46 75 84 106 110 113 128 130 ...
$ Fold08: int [1:7484] 4 29 42 44 49 86 90 114 127 134 ...
$ Fold09: int [1:7484] 12 17 30 53 55 56 57 61 73 87 ...
$ Fold10: int [1:7484] 11 14 20 21 43 66 67 72 76 77 ...
> #Cross-Validation - (80% train | 20% test)
> Knn_K_results <- lapply(CleanedCabData.folds, function(x){
+   train.df
+   test.df
+   confusionMatrix(Knn_K, factor(test_labels), positive = "1")
+   sns <- sensitivity(factor(test_labels), Knn_K)
+   return(sns)
+ })
> str(Knn_K_results)
List of 10
$ Fold01: num 0.278
$ Fold02: num 0.278
$ Fold03: num 0.278
$ Fold04: num 0.278
$ Fold05: num 0.278
$ Fold06: num 0.278
$ Fold07: num 0.278
$ Fold08: num 0.278
$ Fold09: num 0.278
$ Fold10: num 0.278
> mean(unlist(Knn_K_results))
[1] 0.2777778
```

For kNN it is harder to interpret specific inputs, because it does not give clear coefficients or rules. What we can say is that the model uses a combination of GPA, school selectivity, essay sentiment, essay length and density, and timing variables to define distances between applicants. Applicants that look similar on those features will tend to get the same prediction.

In plain terms, people with similar GPAs, essay style, region preferences, and time patterns are seen as neighbors. However, based on the weak performance, kNN is not the best tool for this problem.

### 3. Model 2 – Naive Bayes

We ran a baseline Naive Bayes model and resulted in the following:

- Accuracy about 0.8074
- Kappa about 0.0009
- Sensitivity almost 0.9998
- Specificity about 0.000694

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
 Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
> ## Train Naive Bayes model
> model_nb <- train(`Completed Admissions Process` ~
+ `cumulative GPA` + `School Selectivity` +
+ `Is Math, sci, or Eng Major Minor` + `Essays sentiment` +
+ avg_essay_length + essay_density +
+ days_signup_to_start + days_start_to_submit +
+ days_submit_to_deadline + days_signup_to_submit +
+ `Attended Event` + region_pref_1 +
+ submitted_before_deadline + `Region Preference Level`,
+ data = NB_TrainData,
+ method = "naive_bayes",
+ trControl = trainControl(method = "cv", number = 10))

> # Predictions
> predictions_NB <- predict(model_nb, NB_TestData)
> # Confusion matrix
> confusionMatrix(predictions_NB, NB_TestData$`Completed Admissions Process`, positive = "1")
Confusion Matrix and Statistics

Reference
Prediction      0      1
      0   2   2
      1 2880 12083

Accuracy : 0.8074
95% CI : (0.801, 0.8137)
No Information Rate : 0.8074
P-value [Acc > NIR] : 0.505

Kappa : 9e-04

McNemar's Test P-Value : <2e-16

Sensitivity : 0.999835
Specificity : 0.000694
Pos Pred Value : 0.807525
Neg Pred Value : 0.500000
Prevalence : 0.807443
Detection Rate : 0.807309
Detection Prevalence : 0.999733
Balanced Accuracy : 0.500264

'Positive' Class : 1

> # View model summary
> print(model_nb)
Naive Bayes

59872 samples
14 predictor
 2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 53885, 53885, 53885, 53884, 53885, 53885, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE       0.7836384  0.0776717309
  TRUE        0.8073223 -0.0001601796

Tuning parameter 'laplace' was held constant at a value of 0

Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE
and adjust = 1.
```

We then executed Naive Bayes model with a manual tune approach with a grid over laplace, usekernel, and adjust to achieved the following:

- Accuracy about 0.8075
- Kappa about 0.001
- Sensitivity almost 0.9999
- Specificity about 0.000694

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

The model is almost always predicting 1 and almost never predicting 0.

For the tuned Naive Bayes model, even after searching the grid, the results did not change much. Accuracy is around 0.8075, kappa is around 0.0010, sensitivity is nearly 1, and specificity stays close to zero.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

We can say that the best Naive Bayes model is the tuned one because it gives slightly better metrics, but in practice it behaves the same as always predicting the majority class.

Naive Bayes gives conditional probabilities for each predictor given the class. We did not deeply analyze every probability table, but one important point is that the model heavily follows the prior class distribution. Because 1 is very common, the model prefers to assign class 1 in most cases.

From a Teach For America point of view, this model is not very useful on its own because it does not distinguish well between people who will complete the process and those who will not.

#### 4. Model 3 – Decision Tree

We trained both a baseline tree and a tuned model using a grid of different combinations. Both models gave:

- Accuracy around 0.8074
- Kappa at 0.0000
- Sensitivity at 1.0000
- Specificity at 0.0000

*Decision Tree Baseline*

```
> # Predictions on test data
> Predictions_DT <- predict(DT_model, newdata = DT_TestData)
> # Confusion Matrix
> ConfusionMatrix(Predictions_DT, DT_TestData$'Completed Admissions Process', positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction      0      1
      0      0      0
      1  2882 12085

Accuracy : 0.8074
95% CI : (0.801, 0.8137)
No Information Rate : 0.8074
P-Value [Acc > NIR] : 0.505

Kappa : 0

Mcnemar's Test P-Value : <2e-16

Sensitivity : 1.0000
Specificity : 0.0000
Pos Pred Value : 0.8074
Neg Pred Value :    NAN
Prevalence : 0.8074
Detection Rate : 0.8074
Detection Prevalence : 1.0000
Balanced Accuracy : 0.5000

'Positive' class : 1
```

*Decision Tree Tuned*

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
> # Predictions for tuned model
> Predictions_DT_tuned <- predict(DT_model_tuned, newdata = DT_TestData)
> confusionMatrix(Predictions_DT_tuned, DT_TestData$`Completed Admissions Process`, positive =
+ "1")
Confusion Matrix and Statistics

Reference
Prediction      0      1
      0     0     0
      1   2882 12085

Accuracy : 0.8074
95% CI  : (0.801, 0.8137)
No Information Rate : 0.8074
P-Value [Acc > NIR] : 0.505

Kappa : 0

McNemar's Test P-Value : <2e-16

Sensitivity : 1.0000
Specificity : 0.0000
Pos Pred value : 0.8074
Neg Pred value :    NAN
Prevalence : 0.8074
Detection Rate : 0.8074
Detection Prevalence : 1.0000
Balanced Accuracy : 0.5000

'Positive' class : 1
```

This means the tree always predicts the positive class and does not learn useful splits.

We did not print the full rule set, since the important point is that the model learned to always choose the majority class. That means the tree is basically not using any predictor in a helpful way.

From a Teaching For America perspective, this method in our current setup does not help identify risky candidates. It just confirms the imbalance.

## 5. Model 4 – SVM (Support Vector Machine)

We created dummy variables and scaled everything for SVM. We tried both tune length and a custom tuning grid for sigma and C. Both models performed the same:

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
 Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
> ### Model: SVM (radial kernel)
> # Convert to Categorical variables to Dummy variables - Don not drop first dummy!
> SVMctr_data <- dummy_cols(cleanedTFAData.df,
+ select_columns = c("School.selectivity",
+ "Is.Math.Sci.or.Eng.Major.Minor",
+ "Attended.Event",
+ "region_pref_1",
+ "submitted_before_deadline",
+ "Region.Preference.Level"),
+ remove_first_dummy = FALSE,)
> # Normalize names so spaces become dots in formulas
> names(SVMctr_data) <- make.names(names(SVMctr_data))
> str(SVMctr_data)
tibble [74,839 x 41] (S3:tbl_df/tbl/data.frame)
$ App.Started.Year..RTAT. : Factor w/ 2 levels "2015","2016": 1 2 2 1 2 1 1 1 1 1 ...
$ cumulative.GPA : num [1:74839] -0.94 0.0103 -0.6809 -0.8364 -5.4843 ...
$ Is.Math..Sci..or..Eng.Major.Minor : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 ...
$ School.selectivity : Factor w/ 6 levels "least_selective",...: 2 3 2 3 6 5 2 5 4 4 ...
$ Essay.1.Length : num [1:74839] -0.0524 0.837 -1.1174 0.0794 -1.8092 ...
$ Essay.2.Length : num [1:74839] 0.959 0.895 -1.337 -1.273 -1.38 ...
$ Essay.3.Length : num [1:74839] 0.841 0.645 -2.251 0.103 -2.286 ...
$ Essays.unique.words : num [1:74839] 0.858 1.317 -1.725 -0.119 -2.054 ...
$ Essays.Sentiment : num [1:74839] -2.303 -1.395 0.172 -0.197 0.119 ...
$ Sign.up.Date : POSIXct[1:74839], format: "2014-08-21" "2015-08-10" "2015-10-23" "2014-10-14" ...
$ Started.Date : POSIXct[1:74839], format: "2014-08-21" "2015-08-10" "2015-10-23" "2014-10-14" ...
$ Application.Deadline : POSIXct[1:74839], format: "2014-10-24" "2015-08-21" "2015-10-30" "2014-10-24" ...
$ Submitted.Date : POSIXct[1:74839], format: "2014-10-23" "2015-08-21" "2015-10-24" "2014-10-23" ...
$ Region.Preference.Level : Factor w/ 3 levels "1","2","3": 1 1 1 2 1 1 1 1 ...
$ Attended.Event : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 1 ...
$ Completed.Admissions.Process : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 ...
$ days_signup_to_start : num [1:74839] -0.246 -0.246 -0.246 -0.246 -0.246 ...
$ days_start_to_submit : num [1:74839] 1.101 -0.324 -0.598 -0.378 -0.625 ...
$ days_submit_to_deadline : num [1:74839] -0.459 -0.507 -0.219 -0.459 1.507 ...
$ days_signup_to_submit : num [1:74839] 0.637 -0.392 -0.59 -0.432 -0.61 ...
$ avg_essay_length : num [1:74839] 0.632 0.865 -1.693 -0.41 -1.974 ...
$ essay_density : num [1:74839] -0.0236 -0.2745 -0.8619 -0.5589 -1.2103 ...
$ region_pref_1 : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 2 ...
$ submitted_before_deadline : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 2 2 1 ...
$ School.selectivity_least_selective: int [1:74839] 0 0 0 0 0 0 0 0 0 ...
$ School.selectivity_less_selective: int [1:74839] 1 0 1 0 0 0 1 0 0 ...
$ School.selectivity_more_selective: int [1:74839] 0 1 0 1 0 0 0 0 0 ...
$ School.selectivity_most_selective: int [1:74839] 0 0 0 0 0 0 0 0 1 ...
$ School.selectivity_selective: int [1:74839] 0 0 0 0 0 1 0 1 0 ...
$ School.selectivity_unknown: int [1:74839] 0 0 0 1 0 0 0 0 0 ...
$ Is.Math..Sci..or..Eng.Major.Minor_0: int [1:74839] 1 1 1 0 1 1 1 1 ...
$ Is.Math..Sci..or..Eng.Major.Minor_1: int [1:74839] 0 0 0 1 0 0 0 0 ...
$ Attended.Event_0 : int [1:74839] 1 1 0 1 0 1 1 0 1 ...
$ Attended.Event_1 : int [1:74839] 0 0 0 1 0 0 0 1 0 ...
$ region_pref_1_0 : int [1:74839] 0 0 0 1 0 0 0 0 0 ...
$ region_pref_1_1 : int [1:74839] 1 1 1 0 1 1 1 1 ...
$ submitted_before_deadline_0 : int [1:74839] 0 1 0 0 0 1 0 0 1 ...
$ submitted_before_deadline_1 : int [1:74839] 1 0 1 1 1 0 1 1 0 ...
$ Region.Preference.Level_1 : int [1:74839] 1 1 1 0 1 1 1 1 ...
$ Region.Preference.Level_2 : int [1:74839] 0 0 0 1 0 0 0 0 ...
$ Region.Preference.Level_3 : int [1:74839] 0 0 0 0 0 0 0 0 ...
```

- Accuracy around 0.8074
- Sensitivity exactly 1.0000
- Specificity exactly 0.0000

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

*SVM TuneLength - Model Performance & Confusion Matrix*

```
> # View model performance
> SVM_model
Support Vector Machines with Radial Basis Function Kernel

59872 samples
 23 predictor
 2 classes: '0', '1'

Pre-processing: centered (23), scaled (23)
Resampling: Cross-validated (3 fold)
Summary of sample sizes: 39914, 39916, 39914
Resampling results across tuning parameters:

      C    Accuracy   Kappa
0.25  0.8074559  0.0000000000
0.50  0.8074559  0.0000000000
1.00  0.8074726  0.0001400497

Tuning parameter 'sigma' was held constant at a value of 0.03656592
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.03656592 and C = 1.

> # Predictions on test data
> confusionMatrix(predict(SVM_model, SVM_TestData),
+                   SVM_TestData$Completed.Admissions.Process,
+                   positive = "1")
Confusion Matrix and Statistics

Reference
Prediction   0     1
          0     0     1
          1 2882 12084

          Accuracy : 0.8074
          95% CI : (0.801, 0.8137)
          No Information Rate : 0.8074
          P-value [Acc > NIR] : 0.5132

          Kappa : -1e-04

          Mcnemar's Test P-Value : <2e-16

          Sensitivity : 0.9999
          Specificity : 0.0000
          Pos Pred Value : 0.8074
          Neg Pred Value : 0.0000
          Prevalence : 0.8074
          Detection Rate : 0.8074
          Detection Prevalence : 0.9999
          Balanced Accuracy : 0.5000

          'Positive' class : 1
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

*SVM TuneGrid – Tune Parameter, Model Performance, & Confusion Matrix*

```
> SVM_grid <- expand.grid(  
+   sigma = c(0.02, 0.03, 0.0367, 0.045),  
+   C     = c(0.5, 1, 2)  
+ )
```

```
> # Tuned results and confusion matrix  
> SVM_model_Tuned  
Support Vector Machines with Radial Basis Function Kernel  
  
59872 samples  
 23 predictor  
 2 classes: '0', '1'  
  
Pre-processing: centered (23), scaled (23)  
Resampling: Cross-validated (3 fold)  
Summary of sample sizes: 39914, 39915, 39915  
Resampling results across tuning parameters:  
  
  sigma    C    Accuracy    Kappa  
  0.0200  0.5  0.8074559  0.000000e+00  
  0.0200  1.0  0.8074559  0.000000e+00  
  0.0200  2.0  0.8074559  0.000000e+00  
  0.0300  0.5  0.8074559  0.000000e+00  
  0.0300  1.0  0.8074559  0.000000e+00  
  0.0300  2.0  0.8074058  4.326907e-04  
  0.0367  0.5  0.8074559  0.000000e+00  
  0.0367  1.0  0.8074392  7.327009e-05  
  0.0367  2.0  0.8074392  1.350895e-03  
  0.0450  0.5  0.8074559  0.000000e+00  
  0.0450  1.0  0.8074058  1.130738e-04  
  0.0450  2.0  0.8073557  2.349874e-03  
  
Accuracy was used to select the optimal model using the largest value.  
The final values used for the model were sigma = 0.045 and C = 0.5.  
> SVM_model_Tuned$bestTune  
  sigma    C  
10 0.045 0.5
```

```
> confusionMatrix(predict(SVM_model_Tuned, SVM_TestData),  
+                   SVM_TestData$Completed.Admissions.Process,  
+                   positive = "1")  
Confusion Matrix and Statistics  
  
Reference  
Prediction      0      1  
      0      0      0  
      1  2882 12085  
  
          Accuracy : 0.8074  
          95% CI : (0.801, 0.8137)  
No Information Rate : 0.8074  
P-Value [Acc > NIR] : 0.505  
  
Kappa : 0  
  
McNemar's Test P-Value : <2e-16  
  
Sensitivity : 1.0000  
Specificity : 0.0000  
Pos Pred Value : 0.8074  
Neg Pred Value :      NaN  
Prevalence : 0.8074  
Detection Rate : 0.8074  
Detection Prevalence : 1.0000  
Balanced Accuracy : 0.5000  
  
'Positive' class : 1
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

SVM had the same issue as the tree and Naive Bayes. No differences were found between baseline and tuned, and both resulted in poor specificity.

Because all predictions are positive, we cannot say much about which inputs shape the boundary. The SVM is basically not separating the classes in a meaningful way. It focuses on minimizing error in the majority class, which dominates the loss.

From a practical perspective this means SVM in this configuration is not a helpful model for Teach For America if they want to flag applicants who are likely to drop out.

## **6. Model 5 – ANN (Artificial Neural Network)**

ANNs are not as easy to interpret as trees, but we can still reason about what is happening. Because the ANN performs slightly better on the minority class, it suggests that some combination of features carries a signal about dropping out.

Likely useful features are:

- Timing features like how long it takes to move through the steps
- Essay sentiment and length, which may reflect motivation and communication skills
- School selectivity and GPA, which may relate to overall performance
- Attendance at events and region preference:

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

*ANN Baseline*

```
> # View model performance
> ANN_model
Neural Network

59872 samples
 23 predictor
 2 classes: '0', '1'

Pre-processing: centered (23), scaled (23)
Resampling: Cross-validated (5 fold)
Summary of sample sizes: 47897, 47897, 47898, 47897, 47899
Resampling results across tuning parameters:

  size  decay  Accuracy  Kappa
  1    0e+00  0.8074893  0.0002799659
  1    1e-04  0.8076062  0.0015778738
  1    1e-03  0.8075728  0.0012988220
  1    1e-02  0.8076062  0.0015778738
  1    1e-01  0.8074893  0.0005988020
  3    0e+00  0.8080572  0.0189184875
  3    1e-04  0.8082075  0.0223735430
  3    1e-03  0.8085249  0.0287706996
  3    1e-02  0.8077398  0.0123502407
  3    1e-01  0.8083412  0.0215460877
  5    0e+00  0.8080572  0.0209255574
  5    1e-04  0.8084080  0.0300244205
  5    1e-03  0.8083579  0.0316624435
  5    1e-02  0.8083578  0.0310467444
  5    1e-01  0.8078902  0.0239469063
  7    0e+00  0.8082075  0.0339617850
  7    1e-04  0.8081574  0.0365106138
  7    1e-03  0.8084246  0.0278223740
  7    1e-02  0.8081741  0.0275923695
  7    1e-01  0.8083412  0.0317887970
  9    0e+00  0.8082409  0.0317807008
  9    1e-04  0.8076563  0.0259691953
  9    1e-03  0.8083578  0.0362480998
  9    1e-02  0.8077064  0.0281134579
  9    1e-01  0.8088589  0.0392334316

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 9 and decay = 0.1.
> # Predictions on test data
> confusionMatrix(predict(ANN_model, ANN_TestData),
+                   ANN_TestData$Completed.Admissions.Process,
+                   positive = "1")
Confusion Matrix and Statistics

             Reference
Prediction      0      1
      0     63     59
      1   2819   12026

          Accuracy : 0.8077
          95% CI : (0.8013, 0.814)
          No Information Rate : 0.8074
          P-Value [Acc > NIR] : 0.4719

          Kappa : 0.0267

McNemar's Test P-Value : <2e-16

          Sensitivity : 0.99512
          Specificity : 0.02186
          Pos Pred Value : 0.81010
          Neg Pred Value : 0.51639
          Prevalence : 0.80744
          Detection Rate : 0.80350
          Detection Prevalence : 0.99185
          Balanced Accuracy : 0.50849

'Positive' Class : 1
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
 Nityam Sharma, Soham Mukherjee, Neeyati Anand

*ANN TuneGrid – metri: kappa | maxit: 500 | CV: 10 | Size: c(7, 15, 2) | decay: swq(0.001, 0.01, 0.003)*

```
> ANN_model_tuned <- train(Completed.Admissions.Process ~
+                                         Cumulative.GPA + Essays.Sentiment +
+                                         avg_essay_length + essay_density +
+                                         days_signup_to_start + days_start_to_submit +
+                                         days_submit_to_deadline + days_signup_to_submit +
+                                         School.Selectivity_least_selective +
+                                         School.selectivity_less_selective +
+                                         School.selectivity_more_selective +
+                                         School.selectivity_most_selective +
+                                         Is.Math.Sci..or.Eng.Major.Minor_0 +
+                                         Is.Math.Sci..or.Eng.Major.Minor_1 +
+                                         Attended.Event_0 + Attended.Event_1 +
+                                         region_pref_1_0 + region_pref_1_1 +
+                                         submitted_before_deadline_0 + submitted_before_deadline_1 +
+                                         Region.Preference.Level_1 + Region.Preference.Level_2 + Region.Preference.Level_3,
+                                         data = ANN_TrainData,
+                                         method = "nnet",
+                                         metric = "kappa",
+                                         maxit = 500,
+                                         trControl = trainControl(method = "cv", number = 10),
+                                         preprocess = c("center", "scale"),
+                                         tuneGrid = expand.grid(
+                                             size = seq(7, 15, 2),
+                                             decay = seq(0.001, 0.01, 0.003)),
+                                         trace = FALSE)
> ## See which size/decay was chosen
> ANN_model_tuned
Neural Network

59872 samples
 23 predictor
 2 classes: '0', '1'

Pre-processing: centered (23), scaled (23)
Resampling: cross-validated (10 fold)
Summary of sample sizes: 53885, 53885, 53884, 53885, 53884, 53885, ...
Resampling results across tuning parameters:

size  decay  Accuracy   Kappa
7    0.001  0.8077065  0.04336500
7    0.004  0.8082576  0.03885029
7    0.007  0.8083746  0.03968371
7    0.010  0.8080071  0.03943822
9    0.001  0.8082744  0.04328579
9    0.004  0.8088088  0.04512892
9    0.007  0.8082910  0.04057569
9    0.010  0.8082744  0.04722295
11   0.001  0.8077733  0.04892086
11   0.004  0.8075060  0.04469172
11   0.007  0.8086752  0.05135121
11   0.010  0.8084247  0.05079417
13   0.001  0.8075061  0.05073172
13   0.004  0.8084581  0.05287988
13   0.007  0.8074393  0.04913672
13   0.010  0.8078735  0.05081399
15   0.001  0.8076230  0.05328049
15   0.004  0.8075228  0.05269931
15   0.007  0.8079069  0.05775073
15   0.010  0.8083245  0.05371695

Kappa was used to select the optimal model using the largest value.
The final values used for the model were size = 15 and decay = 0.007.
> ANN_model_tuned$bestTune
  size decay
19  15  0.007
> # Predictions on test data
> confusionMatrix(predict(ANN_model_tuned, ANN_TestData),
+                   ANN_TestData$Completed.Admissions.Process,
+                   positive = "1")
Confusion Matrix and Statistics

             Reference
Prediction      0      1
      0   127   131
      1  2755 11954

Accuracy : 0.8072
95% CI : (0.8008, 0.8135)
No Information Rate : 0.8074
P-Value [Acc > NIR] : 0.538

Kappa : 0.0509

McNemar's Test P-value : <2e-16

Sensitivity : 0.98916
Specificity : 0.04407
Pos Pred Value : 0.81270
Neg Pred Value : 0.49225
Prevalence : 0.80744
Detection Rate : 0.79869
Detection Prevalence : 0.98276
Balanced Accuracy : 0.51661

'Positive' Class : 1
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

- Accuracy around 0.8072
- Kappa around 0.0510
- Sensitivity around 0.9908
- Specificity around 0.0436

Even though the improvements look small, this model has the highest kappa and the best balanced accuracy among all models we tried. It is the only method that gets any meaningful lift over the trivial rule of predicting everyone as 1.

ANNs are not as easy to interpret as trees, but we can still reason about what is happening. Because the ANN performs slightly better on the minority class, it suggests that some combination of features does carry signal about dropping out.

Likely useful features are:

- Timing features like how long it takes to move through the steps
- Essay sentiment and length, which may reflect motivation and communication skills
- School selectivity and GPA, which may relate to overall performance
- Attendance at events and region preference

## 7. Overall Comparison

When we compare all models side by side, we see the following patterns:

1. The class imbalance dominates almost everything. Many methods simply push all predictions to the majority class because that already gives about 0.8000 accuracy.
2. kNN gets lower accuracy, around 0.7583, and only modest improvements in balanced accuracy.
3. Naive Bayes, Decision Tree, and SVM all reach around 0.8070 accuracy but have kappa near zero and specificity near zero. They do not really learn how to spot the minority class.

ANN stands out a bit. It has similar accuracy around 0.8072, but gets a kappa around 0.0510 and a balanced accuracy around 0.5172. It is still not perfect, but it shows some ability to handle the imbalance better than the other methods.

Because of this, we chose the tuned ANN [metri: kappa | maxit: 500 | CV: 10 | Size: c(7, 15, 2) | decay: swq(0.001, 0.01, 0.003)] as our best model. It is the only one that provides any improvement over a pure majority class rule in a consistent way.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

## 8. Practical insights for Teach For America

Even though the imbalance limits the power of our models, we can still draw some qualitative lessons:

- Timing seems important. Applicants who move faster from signing up to starting and from starting to submitting likely show more commitment. The day difference variables capture this and are used by all models.
- Essay-related features like average length, unique words, and sentiment probably separate more engaged candidates from less engaged ones.
- Event attendance and clear region preference also may signal stronger interest and follow-through. Applicants who attend events and have a clear first-choice region might be more likely to finish the process.

If Teach For America wants better predictive power, a next step could be to handle class imbalance with resampling, class weights, or alternative metrics. That would allow the models to focus more on the smaller group that does not complete the process and maybe highlight which patterns are most risky.

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

## 9. Appendix: Full R-Programming Code

#Predictive Analytics (Group 9 Project)

# Student Names:

```
# Jason Lee, Fredrick Swingle Joshua Minami
```

```
# Nityam Sharma, Soham Mukherjee, Neeyati Anand
```

```
# Install Read Excel pkgs
```

```
install.packages("readxl")
```

```
# Load required packages
```

```
library(caret)
```

```
library(class)
```

```
library(naivebayes)
```

```
library(C50)
```

```
library(NeuralNetTools)
```

```
library(fastDummies)
```

```
library(kernlab)
```

```
library(arules)
```

```
library(readxl)
```

```
set.seed(1947)
```

```
#Importing provided Dataset
```

```
TFAData.df<- read_excel("People Analytics at Teach For America Data Set.xlsx")
```

```
str(TFAData.df)
```

```
# Check a few begining records from dataset
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

head(TFAData.df)

# New dataframe w/o unnecessary variables

```
CleanedTFAData.df<- TFAData.df[,c("App Started Year (RTAT)", "Cumulative GPA",
  "Is Math, Sci, or Eng Major Minor", "School Selectivity",
  "Essay 1 Length", "Essay 2 Length", "Essay 3 Length", "Essays
  Unique Words",
  "Essays Sentiment", "Sign-up Date", "Started Date", "Application
  Deadline", "Submitted Date",
  "Region Preference Level", "Attended Event", "Completed
  Admissions Process")]
```

head(CleanedTFAData.df)

str(CleanedTFAData.df)

# Days between signup and start

```
CleanedTFAData.df$days_signup_to_start <- as.numeric(
  difftime(CleanedTFAData.df`Started Date`, CleanedTFAData.df`Sign-up Date`, units
  = "days")
)
```

# Days between start and submit

```
CleanedTFAData.df$days_start_to_submit <- as.numeric(
  difftime(CleanedTFAData.df`Submitted Date`, CleanedTFAData.df`Started Date`,
  units = "days")
)
```

# Days between submit and deadline

```
CleanedTFAData.df$days_submit_to_deadline <- as.numeric(
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
difftime(CleanedTFAData.df$`Application Deadline`, CleanedTFAData.df$`Submitted Date`, units = "days")
```

```
)
```

```
# Days between signup and submit
```

```
CleanedTFAData.df$days_signup_to_submit <- as.numeric(
```

```
difftime(CleanedTFAData.df$`Submitted Date`, CleanedTFAData.df$`Sign-up Date`, units = "days")
```

```
)
```

```
# Average essay length
```

```
CleanedTFAData.df$avg_essay_length <- rowMeans(CleanedTFAData.df[, c("Essay 1 Length", "Essay 2 Length", "Essay 3 Length")],
```

```
na.rm = TRUE
```

```
)
```

```
# Essay density → avg length per unique word
```

```
CleanedTFAData.df$essay_density <- CleanedTFAData.df$avg_essay_length / (CleanedTFAData.df$`Essays Unique Words` + 1)
```

```
# Top-choice region preference (focus on 1st choice → Convert others to zero)
```

```
CleanedTFAData.df$region_pref_1 <- ifelse(CleanedTFAData.df$`Region Preference Level` == 1, 1, 0)
```

```
# Submitted before deadline
```

```
CleanedTFAData.df$submitted_before_deadline <- ifelse(CleanedTFAData.df$days_submit_to_deadline > 0, 1, 0)
```

```
# Check the new variables
```

```
summary(CleanedTFAData.df[, c("days_signup_to_start",
```

```
"days_start_to_submit",
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
"days_submit_to_deadline",  
"days_signup_to_submit",  
"avg_essay_length",  
"essay_density",  
"region_pref_1",  
"submitted_before_deadline")])
```

```
head(CleanedTFAData.df[, c("Sign-up Date","Started Date","Submitted  
Date","Application Deadline",  
"days_signup_to_start","days_start_to_submit","days_submit_to_deadline",  
"avg_essay_length","essay_density","region_pref_1","submitted_before_deadline")])
```

# Factorize Categorical Variables

```
CleanedTFAData.df$`App Started Year (RTAT)` <- as.factor(CleanedTFAData.df$`App  
Started Year (RTAT)`)
```

```
CleanedTFAData.df$`Is Math, Sci, or Eng Major Minor` <-  
as.factor(CleanedTFAData.df$`Is Math, Sci, or Eng Major Minor`)
```

```
CleanedTFAData.df$`Region Preference Level` <-  
as.factor(CleanedTFAData.df$`Region Preference Level`)
```

```
CleanedTFAData.df$`Attended Event` <- as.factor(CleanedTFAData.df$`Attended  
Event`)
```

```
CleanedTFAData.df$region_pref_1 <- as.factor(CleanedTFAData.df$region_pref_1)
```

```
CleanedTFAData.df$submitted_before_deadline <-  
as.factor(CleanedTFAData.df$submitted_before_deadline)
```

```
CleanedTFAData.df$`School Selectivity` <- as.factor(CleanedTFAData.df$`School  
Selectivity`)
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
CleanedTFAData.df$`Completed Admissions Process`<-  
as.factor(CleanedTFAData.df$`Completed Admissions Process`)
```

```
#Scale Data
```

```
CleanedTFAData.df[sapply(CleanedTFAData.df, is.numeric)] <-  
scale(CleanedTFAData.df[sapply(CleanedTFAData.df, is.numeric)])  
str(CleanedTFAData.df)
```

```
#### Model: KNN
```

```
# Predictive Variables
```

```
Pred_Variables <- c("Cumulative GPA",  
"School Selectivity",  
"Is Math, Sci, or Eng Major Minor",  
"Essays Sentiment",  
"avg_essay_length", "essay_density",  
"days_signup_to_start", "days_start_to_submit",  
"days_submit_to_deadline", "days_signup_to_submit",  
"Attended Event", "region_pref_1",  
"submitted_before_deadline", "Region Preference Level")
```

```
# Revert target variable back to simple 0/1 factor (not scaled)
```

```
CleanedTFAData.df$`Completed Admissions Process` <-  
as.factor(TFAData.df$`Completed Admissions Process`)
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

# 80/20 stratified split

```
idx <- createDataPartition(y = CleanedTFAData.df$`Completed Admissions Process`,  
                           p = 0.8, list = FALSE)
```

# Training and testing sets

```
train.df <- CleanedTFAData.df[idx, Pred_Variables]
```

```
test.df <- CleanedTFAData.df[-idx, Pred_Variables]
```

# Labels for train/test

```
train_labels <- CleanedTFAData.df$`Completed Admissions Process`[idx]
```

```
test_labels <- CleanedTFAData.df$`Completed Admissions Process`[-idx]
```

# Convert all factor columns in train/test to numeric

```
train.df[] <- lapply(train.df, function(x) if(is.factor(x)) as.numeric(x) else x)
```

```
test.df[] <- lapply(test.df, function(x) if(is.factor(x)) as.numeric(x) else x)
```

# Check label distribution (in proportion)

```
proportions(table(train_labels))
```

```
proportions(table(CleanedTFAData.df$`Completed Admissions Process`))
```

# Knn → K = 4

```
Knn_K <- knn(train = train.df, test = test.df, cl = train_labels, k = 4)
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
confusionMatrix(Knn_K, factor(test_labels), positive = "1")
```

#####Create 10 folds#####

```
CleanedCabData.folds <- createFolds(CleanedTFAData.df$`Completed Admissions Process`, k = 10)
```

```
str(CleanedCabData.folds) #View 10 folds
```

#Cross-Validation → (80% train | 20% test)

```
Knn_K_results <- lapply(CleanedCabData.folds, function(x){
```

```
  train.df
```

```
  test.df
```

```
  confusionMatrix(Knn_K, factor(test_labels), positive = "1")
```

```
  sns <- sensitivity(factor(test_labels), Knn_K)
```

```
  return(sns)
```

```
})
```

```
str(Knn_K_results)
```

```
mean(unlist(Knn_K_results))
```

### Model: Naive Bayes

```
## Train Naive Bayes Model | 80/20 partition
```

```
NB_TrainData <- CleanedTFAData.df[idx, ]
```

```
NB_TestData <- CleanedTFAData.df[-idx, ]
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
# Check proportions to confirm stratified split  
  
proportions(table(CleanedTFAData.df$`Completed Admissions Process`))  
  
proportions(table(NB_TrainData$`Completed Admissions Process`))  
  
proportions(table(NB_TestData$`Completed Admissions Process`))
```

## Train Naive Bayes model

```
model_nb <- train(`Completed Admissions Process` ~  
  `Cumulative GPA` + `School Selectivity` +  
  `Is Math, Sci, or Eng Major Minor` + `Essays Sentiment` +  
  avg_essay_length + essay_density +  
  days_signup_to_start + days_start_to_submit +  
  days_submit_to_deadline + days_signup_to_submit +  
  `Attended Event` + region_pref_1 +  
  submitted_before_deadline + `Region Preference Level`,  
  data = NB_TrainData,  
  method = "naive_bayes",  
  trControl = trainControl(method = "cv", number = 10))
```

# Predictions

```
predictions_NB <- predict(model_nb, NB_TestData)
```

# Confusion matrix

```
confusionMatrix(predictions_NB, NB_TestData$`Completed Admissions Process`,  
  positive = "1")
```

# View model summary

```
print(model_nb)
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
## Naive Bayes - TuneGrid

# Get tuning parameters

modelLookup('naive_bayes')

# Define tuning grid

NBtune_grid <- expand.grid(
  laplace = c(0, 0.5, 1, 1.5),
  usekernel = c(TRUE, FALSE),
  adjust = seq(0.3, 3, 0.3)
)

model_nb_tuned <- train(`Completed Admissions Process` ~
  `Cumulative GPA` + `School Selectivity` +
  `Is Math, Sci, or Eng Major Minor` + `Essays Sentiment` +
  avg_essay_length + essay_density +
  days_signup_to_start + days_start_to_submit +
  days_submit_to_deadline + days_signup_to_submit +
  `Attended Event` + region_pref_1 +
  submitted_before_deadline + `Region Preference Level`,
  data = NB_TrainData,
  method = "naive_bayes",
  tuneGrid = NBtune_grid,
  trControl = trainControl(method = "cv", number = 10))
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

# Predictions

```
predictions_NB_Tuned <- predict(model_nb_tuned, NB_TestData)
```

# Confusion matrix

```
confusionMatrix(predictions_NB_Tuned, NB_TestData$`Completed Admissions Process`, positive = "1")
```

# View model summary

```
print(predictions_NB_Tuned)
```

### Model: Decision Tree (C5.0)

## Train Decision Tree model | 80/20 partition

```
DT_TrainData <- CleanedTFAData.df[idx, ]
```

```
DT_TestData <- CleanedTFAData.df[-idx, ]
```

# Check proportions (confirm stratified sampling worked)

```
proportions(table(CleanedTFAData.df$`Completed Admissions Process`))
```

```
proportions(table(DT_TrainData$`Completed Admissions Process`))
```

```
proportions(table(DT_TestData$`Completed Admissions Process`))
```

# Base Decision Tree (C5.0) with 10-fold CV | TuneLength

```
DT_model <- train(
```

```
`Completed Admissions Process` ~
```

```
`Cumulative GPA` + `School Selectivity` +
```

```
`Is Math, Sci, or Eng Major Minor` + `Essays Sentiment` +
```

```
avg_essay_length + essay_density +
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
days_signup_to_start + days_start_to_submit +  
days_submit_to_deadline + days_signup_to_submit +  
'Attended Event' + region_pref_1 +  
submitted_before_deadline + 'Region Preference Level',  
data = DT_TrainData,  
method = "C5.0",  
trControl = trainControl(method = "cv", number = 10),  
tuneLength = 10  
)
```

```
# View model performance  
DT_model  
  
# Predictions on test data  
Predictions_DT <- predict(DT_model, newdata = DT_TestData)  
  
# Confusion Matrix  
confusionMatrix(Predictions_DT, DT_TestData$`Completed Admissions Process`,  
positive = "1")  
  
# Decision Tree (C5.0) with 10-fold CV | TuneGrid  
  
# Define tuning grid  
DT_Grid <- expand.grid(  
model = c("tree", "rules"),  
winnow = c(TRUE, FALSE),  
trials = c(1, 5, 10, 15, 20)  
)
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
DT_model_tuned <- train(  
  `Completed Admissions Process` ~  
  `Cumulative GPA` + `School Selectivity` +  
  `Is Math, Sci, or Eng Major Minor` + `Essays Sentiment` +  
  avg_essay_length + essay_density +  
  days_signup_to_start + days_start_to_submit +  
  days_submit_to_deadline + days_signup_to_submit +  
  `Attended Event` + region_pref_1 +  
  submitted_before_deadline + `Region Preference Level`,  
  data = DT_TrainData,  
  method = "C5.0",  
  trControl = trainControl(method = "cv", number = 10),  
  tuneGrid = DT_Grid  
)  
  
# View tuning results  
DT_model_tuned  
DT_model_tuned$bestTune  
  
  
# Predictions for tuned model  
Predictions_DT_tuned <- predict(DT_model_tuned, newdata = DT_TestData)  
confusionMatrix(Predictions_DT_tuned, DT_TestData$`Completed Admissions  
Process`, positive = "1")  
  
  
#### Model: SVM (radial kernel)  
# Convert to Categorical Variables to Dummy Variables - Don not drop first dummy!
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
SVMctr_data <- dummy_cols(CleanedTFAData.df,  
select_columns = c("School Selectivity",  
"Is Math, Sci, or Eng Major Minor",  
"Attended Event",  
"region_pref_1",  
"submitted_before_deadline",  
"Region Preference Level"),  
remove_first_dummy = FALSE,)
```

```
# Normalize names so spaces become dots in formulas
```

```
names(SVMctr_data) <- make.names(names(SVMctr_data))  
  
str(SVMctr_data)
```

```
# Train SVM model
```

```
SVM_TrainData <- SVMctr_data[idx,]
```

```
SVM_TestData <- SVMctr_data[-idx,]
```

# SVM with radial kernel - TuneLength

```
SVM_model <- train(Completed.Admissions.Process ~
```

## Cumulative.GPA + Essays.Sentiment +

avg\_essay\_length + essay\_density +

days\_signup\_to\_start + days\_start\_to\_submit +

days\_submit\_to\_deadline + days\_signup\_to\_submit +

School.Selectivity\_least\_selective +

### School.Selectivity less selective +

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
School.Selectivity_more_selective +  
School.Selectivity_most_selective +  
Is.Math..Sci..or.Eng.Major.Minor_0 +  
Is.Math..Sci..or.Eng.Major.Minor_1 +  
Attended.Event_0 + Attended.Event_1 +  
region_pref_1_0 + region_pref_1_1 +  
submitted_before_deadline_0 + submitted_before_deadline_1 +  
Region.Preference.Level_1 + Region.Preference.Level_2 +  
Region.Preference.Level_3,  
  
data = SVM_TrainData,  
method = "svmRadial",  
trControl = trainControl(method = "cv", number = 3),  
preProcess = c("center", "scale"),  
tuneLength = 3)  
  
# View model performance  
  
SVM_model  
  
# Predictions on test data  
  
confusionMatrix(predict(SVM_model, SVM_TestData),  
SVM_TestData$Completed.Admissions.Process,  
positive = "1")  
  
# SVM with radial kernel - TuneGrid  
  
SVM_grid <- expand.grid(  
sigma = c(0.02, 0.03, 0.0367, 0.045),  
C = c(0.5, 1, 2))
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
SVM_model_Tuned <- train(Completed.Admissions.Process ~  
  Cumulative.GPA + Essays.Sentiment +  
  avg_essay_length + essay_density +  
  days_signup_to_start + days_start_to_submit +  
  days_submit_to_deadline + days_signup_to_submit +  
  School.Selectivity_least_selective +  
  School.Selectivity_less_selective +  
  School.Selectivity_more_selective +  
  School.Selectivity_most_selective +  
  Is.Math..Sci..or.Eng.Major.Minor_0 +  
  Is.Math..Sci..or.Eng.Major.Minor_1 +  
  Attended.Event_0 + Attended.Event_1 +  
  region_pref_1_0 + region_pref_1_1 +  
  submitted_before_deadline_0 + submitted_before_deadline_1 +  
  Region.Preference.Level_1 + Region.Preference.Level_2 +  
 Region.Preference.Level_3,  
 data = SVM_TrainData,  
 method = "svmRadial",  
 trControl = trainControl(method = "cv", number = 3),  
 preProcess = c("center", "scale"),  
 tuneGrid = SVM_grid)  
  
# Tuned results and confusion matrix  
  
SVM_model_Tuned  
SVM_model_Tuned$bestTune
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
confusionMatrix(predict(SVM_model_Tuned, SVM_TestData),  
  SVM_TestData$Completed.Admissions.Process,  
  positive = "1")  
  
#### Model: ANN - TuneLength  
  
## Reuse SVM data for ANN  
  
ANN_TrainData <- SVM_TrainData  
  
ANN_TestData <- SVM_TestData  
  
## Baseline ANN model  
  
ANN_model <- train(Completed.Admissions.Process ~  
  Cumulative.GPA + Essays.Sentiment +  
  avg_essay_length + essay_density +  
  days_signup_to_start + days_start_to_submit +  
  days_submit_to_deadline + days_signup_to_submit +  
  School.Selectivity_least_selective +  
  School.Selectivity_less_selective +  
  School.Selectivity_more_selective +  
  School.Selectivity_most_selective +  
  Is.Math..Sci..or.Eng.Major.Minor_0 +  
  Is.Math..Sci..or.Eng.Major.Minor_1 +  
  Attended.Event_0 + Attended.Event_1 +  
  region_pref_1_0 + region_pref_1_1 +  
  submitted_before_deadline_0 + submitted_before_deadline_1 +  
  Region.Preference.Level_1 + Region.Preference.Level_2 +  
  Region.Preference.Level_3,  
  data = ANN_TrainData,
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
method = "nnet",
metric = "Kappa",
maxit = 300,
trControl = trainControl(method = "cv", number = 5),
preProcess = c("center", "scale"),
tuneLength = 7,
trace = FALSE)

# View model performance

ANN_model

# Predictions on test data

confusionMatrix(predict(ANN_model, ANN_TestData),
ANN_TestData$Completed.Admissions.Process,
positive = "1")

### Model: ANN - TuneGrid

# Get tuning parameters

modelLookup("nnet")

ANN_model_tuned <- train(Completed.Admissions.Process ~
Cumulative.GPA + Essays.Sentiment +
avg_essay_length + essay_density +
days_signup_to_start + days_start_to_submit +
days_submit_to_deadline + days_signup_to_submit +
School.Selectivity_least_selective +
School.Selectivity_less_selective +
School.Selectivity_more_selective +
School.Selectivity_most_selective +
```

Student Names: Jason Lee ,Fredrick Swingle, Joshua Minami  
Nityam Sharma, Soham Mukherjee, Neeyati Anand

```
Is.Math..Sci..or.Eng.Major.Minor_0 +
Is.Math..Sci..or.Eng.Major.Minor_1 +
Attended.Event_0 + Attended.Event_1 +
region_pref_1_0 + region_pref_1_1 +
submitted_before_deadline_0 + submitted_before_deadline_1 +
Region.Preference.Level_1 + Region.Preference.Level_2 +
Region.Preference.Level_3,
data = ANN_TrainData,
method = "nnet",
metric = "Kappa",
maxit = 500,
trControl = trainControl(method = "cv", number = 10),
preProcess = c("center", "scale"),
tuneGrid = expand.grid(
  size = seq(7, 15, 2) ,
  decay = seq(0.001, 0.01, 0.003)),
trace = FALSE)
## See which size/decay was chosen
ANN_model_tuned$bestTune
# Predictions on test data
confusionMatrix(predict(ANN_model_tuned, ANN_TestData),
  ANN_TestData$Completed.Admissions.Process,
  positive = "1")
```