Project Stage 3 Report

Radiation Therapy Team B: Abby Johnson, Abigail McManus, Leah Rohde

ST 495

Using Natural Language Processing to Classify Medical Errors

Executive Summary

In this report, we investigated creating a classification model that categorizes errors made in the radiation therapy process solely using the text description of those errors. In the United States, half of all cancer patients receive radiation therapy treatment at 2,200 centers nationwide [3]. It is estimated that between 44,000 and 98,000 Americans die every year due to medical errors [2], and the radiation therapy process in particular includes over 300 steps [3] and leaves lots of room for error. Identifying and correcting those errors is a manual process that is time consuming, inefficient, and costly. We received the USLCC data in 12 data files that we condensed down into one singular master file, performed Latent Semantic Analysis (LSA) on the error description, and created a support vector machine (SVM) classification model based on our results from the LSA procedure.

Introduction

Death by preventable medical errors is a significant problem for healthcare organizations and patients alike. The World Health Organization reports that adverse events under medical care are likely to be within the top ten leading causes of death and disability worldwide [5]. Medical errors occur either when a planned action is carried out incorrectly or the treatment plan itself is incorrect. Medical errors can stem from measurement errors, misdiagnoses, failure to report, adverse drug reactions, patient misidentification, and more [1, 2, 4]. Additionally, medical errors come with a massive financial burden. While treating the medical error is a large part of the cost, one must also consider the financial impact of disability, resulting loss of income, and general productivity [2]. A widely referenced article, "To Err is Human," from the Institute of Medicine reports a loss of between $17 billion and $29 billion for national hospitals annually [2].

Our area of interest, oncology, is particularly susceptible to medical errors due to its lengthy and complicated process, which involves a large number of clinicians. In addition, the nature of radiation therapy requires intense precision with high risk adverse effects. Varying data is collected across all stages of the radiation therapy process. Upon imaging and diagnosis, the patient is first entered into the database with personal and demographic information, as well as scheduling of future appointments. Billing codes are then created before the patient meets with a financial counselor. Then begins the pre-radiation CT simulation where clinician notes and other data are entered. From there, the clinicians create a computer treatment plan and calculate varying dose calculations both computerized and manual, where data is collected each step of the process. Calculations are particularly prone to error, and small inaccuracies in measurements can have drastic adverse effects. The next step, also particularly prone to error, is measurement and manufacturing of electric cutouts used in the radiation process. Furthering research and development data as well as patient doctor notes are entered up until the radiation treatment. Post-treatment planning and follow up visits are recorded and entered into the database following the radiation. Post-treatment data is particularly crucial as many adverse events arise due to poor communication and follow-up after a procedure [1]. Finally, quality assurance and radiation safety measures and tests are performed and put into the database as an effective regulatory measure. In this brief overview of the radiation therapy process, any number of errors are likely to occur. Thus, with medical errors being so hazardous and the radiation therapy process being so error-prone, it is becoming increasingly necessary to automate the process for handling these errors.

In this study, our goal was to use natural language processing (NLP) techniques to parse free-text error reports and to use classification modeling to accurately categorize those errors. With the resulting model, our team hopes to improve patient safety and save hospitals time and money. A systematic review of previous attempts to categorize error reports using a range of NLP techniques found that across the 35 articles, models were met with a favorable performance [6]. Continued research and development of NLP-based models is a promising approach to automating adverse event categorization. Improvements in event categorization will help oncologists better understand the nature of reported medical errors. Having automated analysis alongside expert knowledge can ideally be used to reduce the number of adverse events during the radiation process. On a larger scale, NLP and classification techniques can be used across a wide array of specialities, helping hospitals manage risk associated with medical errors.

Description of Raw Data

The data used in our analysis was compiled from the USLCC health center's error reporting program and was made available to us by MERP. The data contains information on medical process errors including those that occurred pre-treatment and post-treatment. The data is spread across 12 files and connected by a unique error identifier, called DEVID. The additional variables contained in the data set describe the error that occurred, how the error was corrected, and how corrections were approved.

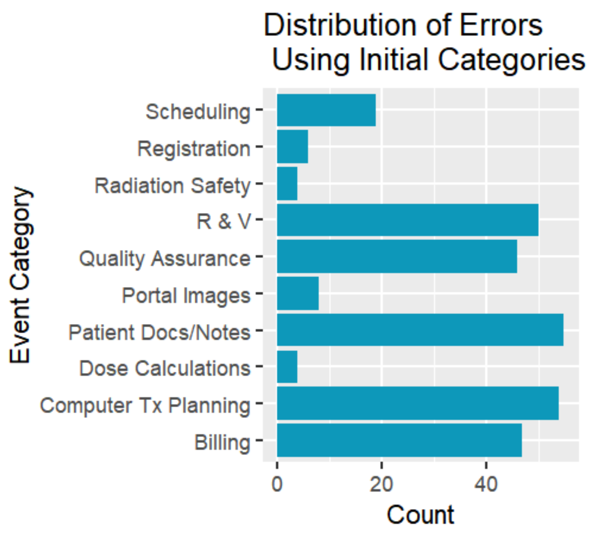
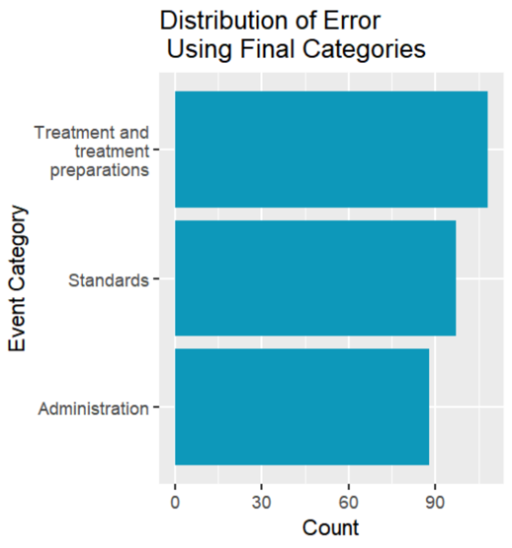
Methods

Data Wrangling

Before we could begin the process of using NLP and classification techniques, it was first necessary to clean the raw data. Our first task, after reading all 12 files into R, was to remove unnecessary variables. Given that we had a list of the variables we needed to keep, we were easily able to identify and filter out the variables that were of no use to us. Performing this task first allowed us to focus on cleaning only the data that was relevant to our analysis. Our second task was to rename the variables. Certain variables across the several files had identical names, like DESCRIPTION, but captured different information. To keep this information distinct, these variables were given more descriptive names, like DESCRIPTION (ERROR_DESCRIPTION). Our next task was to merge the files. To connect the 12 files, we needed to find unique record identifiers that would connect one data set to the other. One variable that connected most of the data sets together was DEVID, which contained unique ID codes for errors that had occurred. While this was an effective way to connect data sets, there were some observations that were missing DEVID values. For these observations, we simply removed them while merging as we would be unable to connect these observations with data from other files. Other variables used to merge data sets were ACTION_ID, which was an identifier for corrective actions, USERNAME, which was the login of the person who reported the error, and ERROR_CODE, which denoted the numbering convention associated with an error. Our final task was to condense the data such that the final data set contained only one record per DEVID. Since it was possible for an error to be corrected multiple times and for these corrections to be approved multiple times, our merged data set contained several records per error. To resolve this, values that caused this to occur were concatenated into comma separated strings. For example, if an error was corrected once by rgood and once by ekline, the value for “CORRECTED_BY” would be “rgood, ekline”. Once this task had been completed, we had a functioning master data set with 293 observations (one per DEVID) and all variables that were required by our project description.

Text Processing and Tokenization

While the master data set contained clean data, the text variables required further processing in preparation for the classification model. To do this we started by isolating the two variables that were needed for our model: ERROR_DESCRIPTION, the model’s only feature, and EVENT_CAT, the model’s response. There were 10 different adverse event categories in our analysis. However, since the cleaned data had only 293 observations, several categories contained too few observations to make meaningful analysis. To resolve this, the 10 event categories were grouped into 3 larger categories: administration, standards, and treatment/treatment preparations. See Figures 1 and 2 below for a visualization of this transformation. Next, we normalized the ERROR_DESCRIPTION variable by removing any punctuation and special characters from the text, ensuring that those characters would not interfere with the text analysis to come. We also converted the ERROR_DESCRIPTION values to be lowercase so that words with different capitalizations would not contribute to errors in the analysis. Then, we tokenized the ERROR_DESCRIPTION variable at ngram levels 1, 2, and 3 and converted the tokens into numerical features by using tf-idf, assigning a weight for each term. We converted the table containing the EVENT_CAT and ERROR_DESCRIPTION variables into a Corpus, which creates an individual document for every entry in our table. Additionally, a document term matrix (DTM) was constructed using the tf-idf weights. A known limitation of DTM is that it creates a high dimensional matrix with very "noisy" data. To reduce the amount of noise in the DTM, sparse terms were removed to create a new document term matrix by a reduction of 80% of terms. We then applied LSA to reduce the dimensions of the DTM using singular value decomposition. The resulting table was the input for our classification model.

Figure 1	Figure 2
	

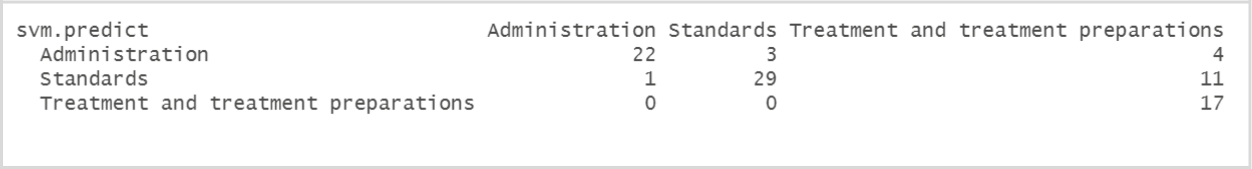
Modeling

To begin modeling the data, we split the data into a training set and a testing set, using 70% of the data for the former and the remaining 30% for the latter. For our classification model, we chose the supervised learning technique Linear Support Vector Machine (SVM). Linear SVM is known to be both efficient and accurate and applied well to our data set. We used our training data to evaluate the performance of this model, and the results of this evaluation are discussed in the following section.

Results and Discussion

Out of 293 observations, 87 observations were used to test the performance of our SVM model. Of these 87 observations, 68 were correctly classified, giving our model an accuracy of 78.16%. See Figure 3 below for the confusion matrix documenting these results. Overall, this model performed well and demonstrates the value of using natural language processing and classification models to improve the error handling process for medical errors. However, our team believes that improvements could be made to increase the accuracy of future models.

Figure 3



```
svm.predict                            Administration Standards Treatment and treatment preparations
  Administration                                   22         3                                    4
  Standards                                         1        29                                   11
  Treatment and treatment preparations              0         0                                   17
```

Many of the challenges we faced while working on this study were as a result of the data formatting. Some of the values in the data contained special characters, oddly placed capitalizations, and other confounding input that needed to be changed before we could properly conduct analysis on it. We also recognize that data from other sources may hold their own unique issues with the formatting of variables or they may not have quality data points that allow for the same kind of analysis to be done. Thus, while we tried to make our data as clean as we could, we recommend that future research employ more data cleaning techniques than we have carried out here. For example, future studies could make efforts to account for misspelled words or even synonymous words in the data. Another challenge we encountered was the limited amount of data we received. With only 293 observations, it was simply infeasible to use the original 10 event categories to conduct our analysis. As said previously, some categories contained too few observations, and to resolve this, we combined the original categories into 3 much larger categories. While this made our analysis easier to conduct, doing so may not be practical for all situations. For example, if the users of our model needed to differentiate between treatment errors and treatment preparation errors, our solution would be ineffective. To improve upon this, we recommend that further research is conducted on much larger data sets. The last challenge we faced is related to the simplicity of our methods. While our SVM model produced favorable results, we believe that more complex techniques, such as deep learning methods, could result in a much more in-depth and accurate classification.

In conclusion, preventable medical errors are extremely hazardous for patients, and the process for handling these errors leaves much to be desired. However, the results of our study show that it is possible to automate the error categorization process, and we hope that our classification model and the classification models of future studies will be used as tools to increase patient safety.

References

1. Bari, A., Khan, R. A., & Rathore, A. W. (2016). Medical errors; causes, consequences, emotional response and resulting behavioral change. *Pakistan journal of medical sciences*, *32*(3), 523–528. https://doi.org/10.12669/pjms.323.9701
2. Institute of Medicine (US) Committee on Quality of Health Care in America, Kohn, L. T., Corrigan, J. M., & Donaldson, M. S. (Eds.). (2000). *To Err is Human: Building a Safer Health System*. National Academies Press (US).
3. NC State - Error Prop Modeling Class Description. https://docs.google.com/document/d/16jcmPueNTMNje4uG8YWojMqWMI7kWbdH/
4. Ulanimo, V. M., O'Leary-Kelley, C., & Connolly, P. M. (2007). Nurses' perceptions of causes of medication errors and barriers to reporting. *Journal of nursing care quality*, *22*(1), 28–33. https://doi.org/10.1097/00001786-200701000-00007
5. World Health Organization. (2019, September 13). *Patient safety*. World Health Organization. Retrieved December 2, 2022, from https://www.who.int/news-room/fact-sheets/detail/patient-safety
6. Young, I. J., Luz, S., & Lone, N. (2019). A systematic review of natural language processing for classification tasks in the field of incident reporting and Adverse Event Analysis. *International Journal of Medical Informatics*, *132*, 103971. https://doi.org/10.1016/j.ijmedinf.2019.103971

Code

```
setwd("/Users/leahrohde/Documents/Classes/st495/Project")

library(readxl)

library(dplyr)

library(tidytext)

library(SnowballC)

library(tm)

library(XML)

library(lsa)

############################################### Reading in the data


Master_Chart_ID <- read_excel("Master_Chart_ID.xlsx")

Pat_Approvals <- read_excel("Pat_Approvals.xlsx")

Pat_Category_Codes <- read_excel("Pat_Category_Codes.xlsx")

Pat_Corrections <- read_excel("Pat_Corrections.xlsx")

Pat_Corrective_Node <- read_excel("Pat_Corrective_Node.xlsx")

Pat_Daily_Portions <- read_excel("Pat_Daily_Portions.xlsx")

Pat_Documents <- read_excel("Pat_Documents.xlsx")

Pat_Error_Lookup <- read_excel("Pat_Error_Lookup.xlsx")

Pat_Roles <- read_excel("Pat_Roles.xlsx")

Pat_Tasks <- read_excel("Pat_Tasks.xlsx")

Pat_Users <- read_excel("Pat_Users.xlsx")

Patient_Master <- read_excel("Patient_Master.xlsx")
```

```
############################################### Selecting variables and fixing variable names


# Pat_Category_Codes

# with DATASHEET VARIABLE NAMES

Fixed_Pat_Category_Codes <- Pat_Category_Codes[, c("ID", "CATLEVEL", "DESCRIPTION")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Pat_Category_Codes) <- c("CATEGORY1_ID", "ERROR_LEVEL_NUMBER",
"ERROR_LEVEL_NAME")




# Master_Chart_ID

# with DATASHEET VARIABLE NAMES

Fixed_Master_Chart_ID <- Master_Chart_ID[, c("DEVID", "DESCR_IF_MISC", "DESCRIPTION",

                                     "IDENTIFIED_BY", "MODIFIED_BY",

                                     "AFFECTED_TREATMENT", "CORRECTED",

                                     "WHYNOTCORRECTED", "TX_INTENT", "TX_METHOD",

                                     "DEV_TYPE", "ASSIGNED_USER", "ASSIGNED_ROLE",

                                     "ERROR_CODE")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Master_Chart_ID) <- c("DEVID", "CUSTOM_ERROR", "ERROR_DESCRIPTION",
"IDENTIFIED_BY",

                          "ASSIGNED_PERSON", "AFFECTED_TREATMENT", "CORRECTED",

                          "WHYNOTCORRECTED", "TX_INTENT", "TX_METHOD",
"MAIN_ERROR_TYPE",

                          "ASSIGNED_USER", "ASSIGNED_ROLE", "ERROR_CODE")
```

```
# Pat_Approvals
# with DATASHEET VARIABLE NAMES
Fixed_Pat_Approvals <- Pat_Approvals[, c("DEVID", "STAGE", "ROLE", "DESCRIPTION",
                                 "APPROVED_BY", "ISARCHIVED")]
# with EQUIVALENT VARIABLE NAMES
names(Fixed_Pat_Approvals) <- c("DEVID", "APPROVAL_STAGE", "LAST_CORRECTION_ROLE",
                           "APPROVING_DESCRIPTION","APPROVER_NAME", "APPROVAL_STATUS")


# Pat_Corrections
# with DATASHEET VARIABLE NAMES
Fixed_Pat_Corrections <- Pat_Corrections[, c("DEVIATION_ID", "DESCRIPTION", "ACTION_ID")]
# with EQUIVALENT VARIABLE NAMES
names(Fixed_Pat_Corrections) <- c("DEVID", "CUSTOM_CORRECTION", "ACTION_ID")


# Pat_Corrective_Node
# with DATASHEET VARIABLE NAMES
Fixed_Pat_Corrective_Node <- Pat_Corrective_Node[, c("ID", "DESCRIPTION")]
# with EQUIVALENT VARIABLE NAMES
names(Fixed_Pat_Corrective_Node) <- c("ACTION_ID", "SELECTED_CORRECTION")
```

```
# Pat_Documents

# with DATASHEET VARIABLE NAMES

Fixed_Pat_Documents <- Pat_Documents[, c("DEVIATION_ID","DOC_TYPE")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Pat_Documents) <- c("DEVID","DOCUMENT_TYPE")



# Pat_Error_Lookup

# with DATASHEET VARIABLE NAMES

Fixed_Pat_Error_Lookup <- Pat_Error_Lookup[, c("ID","CATEGORY1_ID","CATEGORY2_ID","SL",

                                   "IS_POST", "ROLE_CODES", "REG_VIOLATION",

                                   "POSSIBLE_VIOLATION", "CLIN_SIGNIFICANT",

                                   "SIGNIFICANT_ERROR", "MISADMIN", "RECORDABLE",

                                   "INTENT_METHOD_APPLICABLE", "CORRECTION_ROLES")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Pat_Error_Lookup) <- c("ERROR_CODE", "CATEGORY1_ID","CATEGORY2_ID",
"SEVERITY_LEVEL",

                          "BEFORE/AFTER_TX", "APPROVAL_ORDER", "VIOLATION",

                          "POSSIBLE_VIOLATION", "CLINICALLY_SIGNIFICANT", "SIGNIFICANT",

                          "MISADMINISTRATION", "RECORDABLE_EVENT",

                          "METHOD_OF_TREATMENT", "USUAL_CORRECTION_ROLE")



# Pat_Roles
# with DATASHEET VARIABLE NAMES

Fixed_Pat_Roles <- Pat_Roles[, c("ID","RNAME")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Pat_Roles) <- c("ROLE_ID","ROLE_NAME")




# Pat_Tasks

# with DATASHEET VARIABLE NAMES

Fixed_Pat_Tasks <- Pat_Tasks[, c("ARTIFACTID", "MOREINFO","USERID", "GROUPID",
"ACTION",

                         "STATUS")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Pat_Tasks) <- c("DEVID", "ERROR","USERNAME", "NAME", "ACTION", "STATUS")




# Pat_Users

# with DATASHEET VARIABLE NAMES

Fixed_Pat_Users <- Pat_Users[, c("ID","ROLE_CODES")]

# with EQUIVALENT VARIABLE NAMES

names(Fixed_Pat_Users) <- c("USER_ID","ROLE_CODES")




################################################# Combining data sets
```

```
# Merging Master_Chart_ID and Pat_Approvals to make master

Fixed_Pat_Approvals_Condensed <- Fixed_Pat_Approvals %>%

  group_by(DEVID) %>%

  mutate(APPROVAL_STAGE = paste(APPROVAL_STAGE, collapse = ", "),

         LAST_CORRECTION_ROLE = paste(LAST_CORRECTION_ROLE, collapse = ", "),

         APPROVING_DESCRIPTION = paste(APPROVING_DESCRIPTION, collapse = ", "),

         APPROVER_NAME = paste(APPROVER_NAME, collapse = ", "),

         APPROVAL_STATUS = paste(APPROVAL_STATUS, collapse = ", "))


Fixed_Pat_Approvals_Condensed <-

  Fixed_Pat_Approvals_Condensed[!duplicated(Fixed_Pat_Approvals_Condensed$DEVID), ]


master <- merge(Fixed_Master_Chart_ID, Fixed_Pat_Approvals_Condensed, by = "DEVID", all.x =
TRUE)




# Merging Pat_Corrections and Pat_Corrective_Node with master

combine1 <- merge(Fixed_Pat_Corrections, Fixed_Pat_Corrective_Node, by = "ACTION_ID", all.x =
TRUE)


combine1_Condensed <- combine1 %>%

  group_by(DEVID) %>%

  mutate(ACTION_ID = paste(ACTION_ID, collapse = ", "),

         CUSTOM_CORRECTION = paste(CUSTOM_CORRECTION, collapse = ", "),
         SELECTED_CORRECTION = paste(SELECTED_CORRECTION, collapse = ", "))

combine1_Condensed <- combine1_Condensed[!duplicated(combine1_Condensed$DEVID), ]

master <- merge(master, combine1_Condensed, by = "DEVID", all.x = TRUE)



# Merging Pat_Tasks with master
Fixed_Pat_Tasks_Condensed <- Fixed_Pat_Tasks %>%
  group_by(DEVID) %>%
  mutate(ERROR = paste(ERROR, collapse = ", "),
         USERNAME = paste(USERNAME, collapse = ", "),
         NAME = paste(NAME, collapse = ", "),
         ACTION = paste(ACTION, collapse = ", "),
         STATUS = paste(STATUS, collapse = ", "))

Fixed_Pat_Tasks_Condensed <-
  Fixed_Pat_Tasks_Condensed[!duplicated(Fixed_Pat_Tasks_Condensed$DEVID), ]

master <- merge(master, Fixed_Pat_Tasks_Condensed, by = "DEVID", all.x = TRUE)
```

```
# Merging Pat_Documents with master

Fixed_Pat_Documents_Condensed <- Fixed_Pat_Documents %>%

  group_by(DEVID) %>%

  mutate(DOCUMENT_TYPE = paste(DOCUMENT_TYPE, collapse = ", "))


Fixed_Pat_Documents_Condensed <-

  Fixed_Pat_Documents_Condensed[!duplicated(Fixed_Pat_Documents_Condensed$DEVID), ]


master <- merge(master, Fixed_Pat_Documents_Condensed, by = "DEVID", all.x = TRUE)




# Merging Pat_Error_Lookup and Fixed_Pat_Category_Codes with master

combine2 <- merge(Fixed_Pat_Error_Lookup, Fixed_Pat_Category_Codes, by = "CATEGORY1_ID",
all = TRUE)

master <- merge(master, combine2, by = "ERROR_CODE", all.x = TRUE)




# Merging Pat_Users and Pat_Roles with master

ROLE_NAME = c()

roles = ""

current = ""

for (i in row(Fixed_Pat_Users)) {

  for (j in 1:nchar(Fixed_Pat_Users$ROLE_CODES[i])) {
    char = substr(Fixed_Pat_Users$ROLE_CODES[i], j, j)

    current = paste(char, " = ", subset(Fixed_Pat_Roles, ROLE_ID == char)[2], sep = "")

    if (roles == "") {

      roles = paste(roles, current, sep = "")

    } else {

      roles = paste(roles, current, sep = ", ")

    }

  }

  ROLE_NAME[i] = roles

  roles = ""

  j = 1

}

combine3 <- cbind(Fixed_Pat_Users, ROLE_NAME)

names(combine3)[1] <- "USERNAME"

master <- merge(master, combine3, by = "USERNAME", all.x = TRUE)


################################################ Reordering observations

master <- master[order(master$DEVID), ]


################################################ Reordering and renaming variables

master <- master[, c("ERROR_LEVEL_NUMBER", "ERROR_LEVEL_NAME", "CUSTOM_ERROR",
"ERROR_DESCRIPTION",

                     "IDENTIFIED_BY", "ASSIGNED_PERSON", "AFFECTED_TREATMENT",

                     "CORRECTED", "WHYNOTCORRECTED", "TX_INTENT", "TX_METHOD",
"MAIN_ERROR_TYPE",
                "ASSIGNED_USER", "ASSIGNED_ROLE", "APPROVAL_STAGE",
"LAST_CORRECTION_ROLE",

                "APPROVING_DESCRIPTION", "APPROVER_NAME", "APPROVAL_STATUS",

                "CUSTOM_CORRECTION", "SELECTED_CORRECTION", "DOCUMENT_TYPE",
"SEVERITY_LEVEL",

                "BEFORE/AFTER_TX", "APPROVAL_ORDER", "VIOLATION",
"POSSIBLE_VIOLATION",

                "CLINICALLY_SIGNIFICANT", "SIGNIFICANT", "MISADMINISTRATION",
"RECORDABLE_EVENT",

                "METHOD_OF_TREATMENT", "USUAL_CORRECTION_ROLE", "ROLE_NAME",
"ERROR", "USERNAME",

                "NAME", "ACTION", "STATUS", "ROLE_CODES")]

names(master) <- c("CATLEVEL", "DESCRIPTION (ERROR_LEVEL_NAME)", "DESCR_IF_MISC",

               "DESCRIPTION (ERROR_DESCRIPTION)", "IDENTIFIED_BY", "MODIFIED_BY",

               "AFFECTED_TREATMENT", "CORRECTED", "WHYNOTCORRECTED",
"TX_INTENT", "TX_METHOD",

               "DEV_TYPE", "ASSIGNED_USER", "ASSIGNED_ROLE", "STAGE", "ROLE",

               "DESCRIPTION (APPROVING_DESCRIPTION)", "APPROVER_BY", "ISARCHIVED",

               "DESCRIPTION (CUSTOM_CORRECTION)", "DESCRIPTION
(SELECTED_CORRECTION)", "DOC_TYPE",

               "SL", "IS_POST", "ROLE_CODES (APPROVAL_ORDER)", "REG_VIOLATION",

               "POSSIBLE_VIOLATION", "CLIN_SIGNIFICANT", "SIGNIFICANT_ERROR",
"MISADMIN",

               "RECORDABLE", "INTENT_METHOD_APPLICABLE", "CORRECTION_ROLES",
"RNAME", "MOREINFO",

               "USERID", "GROUPID", "ACTION", "STATUS", "ROLE_CODES")
```

```
################################################ Exporting final data set

write.csv(master,"/Users/leahrohde/Documents/Classes/st495/Project/Master.csv", row.names = FALSE)



#STAGE2

## correct data and begin tokenization process




########################## Removes punctuation

master2 <- master %>% mutate(across(`DESCRIPTION (ERROR_DESCRIPTION)`,
~gsub("[[:punct:]]", "", .x)))


################################################ Converts to lowercase

master2$`DESCRIPTION (ERROR_DESCRIPTION)` = tolower(master$`DESCRIPTION
(ERROR_DESCRIPTION)`)





######################## Splits observations into 3 larger categories

for(i in 1:293) {

  if(master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Billing' ||

     master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Quality Assurance' ||

     master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Radiation Safety') {

    master2$EVENT_CAT[i] = 'Standards'

  } else if(master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Computer Tx Planning' ||

            master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Dose Calculations' ||
        master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'R & V') {
    master2$EVENT_CAT[i] = 'Treatment and treatment preparations'
  } else if(master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Registration' ||
        master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Scheduling' ||
        master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Patient Docs/Notes' ||
        master2$`DESCRIPTION (ERROR_LEVEL_NAME)`[i] == 'Portal Images') {
    master2$EVENT_CAT[i] = 'Administration'
  }
}


############ tokenizing the description for ngrams = 1, 2, 3


errors <- tibble(txt = master2$`DESCRIPTION (ERROR_DESCRIPTION)`)
errors %>% unnest_tokens(word, txt, token = "ngrams", n = 1)



master2_tibble <- tibble(EVENT_CAT = master2$EVENT_CAT,
                         ERROR_DESCRIPTION = master2$`DESCRIPTION (ERROR_DESCRIPTION)`)
master2_words <- master2_tibble %>%
  unnest_tokens(word, ERROR_DESCRIPTION, token = "ngrams", n = 3, n_min = 1) %>%
  count(EVENT_CAT, word, sort = TRUE) %>%
  ungroup()
```

```
total_words <- master2_words %>%

  group_by(EVENT_CAT) %>%

  summarize(total = sum(n))

master2_words <- left_join(master2_words, total_words)

master2_words




freq_by_rank <- master2_words %>%

  group_by(EVENT_CAT) %>%

  mutate(rank = row_number(),

         `term frequency` = n/total)

freq_by_rank




master2_words <- master2_words %>%

  bind_tf_idf(word, EVENT_CAT, n)

master2_words


#creating corpus

master2corpus <- VCorpus(VectorSource(master2_tibble$ERROR_DESCRIPTION))


#LSA

tdm = TermDocumentMatrix(master2corpus, control = list(weighting = function(x) weightTfIdf(x,
normalize = TRUE), wordLengths=c(2,Inf)))

lsaSpace <- lsa(tdm)  # create LSA space
```

```
lsak = lsaSpace$dk #Docs  decomposed to k dimensions

dim(lsak)

lsaSpace$sk # these are like the variance

plot(lsaSpace$sk) # scree plot


#dtm

dtm = DocumentTermMatrix(master2corpus, control = list(

  weighting = function(x) weightTfIdf(x, normalize = TRUE),stopwords=TRUE))

dtm


#removing sparse terms

dtm2 = removeSparseTerms(dtm, 0.8)

dtm2

#creating dtm matrix

matrix = as.matrix(dtm2)

eventlabels = master2_tibble$EVENT_CAT

data.eventcat = data.frame(eventlabels, matrix)

data.eventcat$eventlabels = as.factor(data.eventcat$eventlabels)

table(data.eventcat$eventlabels)


#LSA

tdm = TermDocumentMatrix(master2corpus, control = list(weighting = function(x) weightTfIdf(x,
normalize = TRUE), wordLengths=c(2,Inf)))

lsaSpace <- lsa(tdm)  # create LSA space

lsak = lsaSpace$dk # dimension reduction
```

```
dim(lsak)

lsaSpace$sk # like the variance

plot(lsaSpace$sk) # plot


#support vector

library(e1071)

# ~70% of data for training, it is rounded to the nearest whole number

train = c(sample(88, 62), 88+sample(97, 68), 185+sample(108, 76))

data.train = data.eventcat[train,]

data.test = data.eventcat[-train,]


#predictions and summary

svm.model = svm(eventlabels ~. , data = data.train)

svm.predict = predict(svm.model, data.test[,-1])


table(svm.predict, data.test$eventlabels)

mean(svm.predict != data.test$eventlabels)

summary(svm.predict)
```