```
### bielolopez@gmail.com  ###




mix_dataset <- data.frame(
  id=c(10,20,30,40,50),
  gender=c('male','female','female','male','female'),
  some_date=c('01/11/2012','04/12/2012','28/02/2013','17/06/2014','08/03/2015'),
  value=c(12.34, 32.2, 24.3, 83.1, 8.32),
  outcome=c(1,1,0,0,0))

write.csv(mix_dataset, 'mix_dataset.csv', row.names=FALSE)
library(readr)
mix_dataset <- read_csv('mix_dataset.csv')
mix_dataset$some_date <- as.Date(mix_dataset$some_date, format="%d/%m/%Y")
str(mix_dataset$some_date)
```

```
Fix_Date_Features <- function(data_set) {
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (feature_name in text_features) {
  feature_vector <- as.character(data_set[,feature_name])
  # assuming date pattern: '01/11/2012'
  date_pattern <- '[0-9][0-9]/[0-9][0-9]/[0-9][0-9][0-9][0-9]'
  if (max(nchar(feature_vector)) == 10) {
   if (sum(grepl(date_pattern, feature_vector)) > 0) {
    print(paste('Casting feature to date:',feature_name))
    data_set[,feature_name] <- as.Date(feature_vector, format="%d/%m/%Y")
   }
  }
 }
 return (data_set)
}



path_and_file_name <- 'mix_dataset.csv'
```

```
# quick peek at top lines
print(readLines(path_and_file_name, n=5))



mix_dataset <- read.csv(path_and_file_name, stringsAsFactor=FALSE)
mix_dataset1 <- print(head(Fix_Date_Features(mix_dataset)))

write.csv(mix_dataset1, 'mix_dataset1.csv', row.names=FALSE)



###############################################################################################################################################3



# Text
# load the data set in case you haven't already done so
Titanic_dataset <- read.table('http://math.ucdenver.edu/RTutorial/titanic.txt', sep='\t', header=TRUE, stringsAsFactors = FALSE)
head(Titanic_dataset)
```

```
Titanic_dataset_temp <- Titanic_dataset

Titanic_dataset_temp$Word_Count <- sapply(strsplit(Titanic_dataset_temp$Name, " "), length)

print(head(Titanic_dataset_temp$Word_Count))


Titanic_dataset_temp <- Titanic_dataset

Titanic_dataset_temp$Character_Count <- nchar(as.character(Titanic_dataset_temp$Name))

print(head(Titanic_dataset_temp$Character_Count))


Titanic_dataset_temp <- Titanic_dataset

Titanic_dataset_temp$First_Word <- sapply(strsplit(as.character(Titanic_dataset_temp$Name), " "), `[`, 1)

print(head(Titanic_dataset_temp$First_Word))


Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, features_to_ignore=c()) {

 # look for text entries that are mostly unique

 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))

 for (f_name in setdiff(text_features, features_to_ignore)) {

  f_vector <- as.character(data_set[,f_name])
```

```
   # treat as raw text if data over minimum_precent_unique unique
   if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_unique_threshold)) {
     data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vector, " "), length)
     data_set[,paste0(f_name, '_character_count')] <- nchar(as.character(f_vector))
     data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`, 1)
     # remove orginal field
     data_set[,f_name] <- NULL
   }
  }
  return(data_set)
}


Titanic_dataset_temp <- Get_Free_Text_Measures(data_set = Titanic_dataset, features_to_ignore = c())
str(Titanic_dataset_temp)



#######################################################################################################################
```

```
# Factor 1

survey <- data.frame(satisfaction=c('very unhappy','unhappy','neutral','happy','very happy'))
print(survey)


survey$satisfaction <- as.factor(survey$satisfaction)
survey$satisfaction_Level <- as.numeric(survey$satisfaction)
print(survey$satisfaction_Level)


survey$satisfaction <- as.factor(survey$satisfaction)
levels(survey$satisfaction) <- list('very unhappy'=1,'unhappy'=2,'neutral'=3,'happy'=4,'very happy'=5)
survey$satisfaction_Level <- as.numeric(survey$satisfaction)
print(survey$satisfaction_Level)


Titanic_dataset <- read.table('http://math.ucdenver.edu/RTutorial/titanic.txt', sep='\t', header=TRUE)
head(Titanic_dataset)
```

```
dim(Titanic_dataset)

unique(Titanic_dataset$Sex)

unique(Titanic_dataset$PClass)

Titanic_dataset_temp <- Titanic_dataset
Titanic_dataset_temp$Sex_Female <- ifelse(Titanic_dataset_temp$Sex=='female', 1, 0)
Titanic_dataset_temp$Sex_Male <- ifelse(Titanic_dataset_temp$Sex=='male', 1, 0)
head(Titanic_dataset_temp)


Titanic_dataset_temp <- Titanic_dataset
for (newcol in unique(Titanic_dataset_temp$PClass)) {
  feature_name <- 'PClass'
  Titanic_dataset_temp[,paste0(feature_name,"_",newcol)] <- ifelse(Titanic_dataset_temp[,feature_name]==newcol,1,0)
}
```

```
head(Titanic_dataset_temp)

Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE) {
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (feature_name in setdiff(text_features, features_to_ignore)) {
  feature_vector <- as.character(data_set[,feature_name])
  # check that data has more than one level
  if (length(unique(feature_vector)) == 1)
   next
  # We set any non-data to text
  feature_vector[is.na(feature_vector)] <- 'NA'
  feature_vector[is.infinite(feature_vector)] <- 'INF'
  feature_vector[is.nan(feature_vector)] <- 'NAN'
  # loop through each level of a feature and create a new column
  first_level=TRUE
  for (newcol in unique(feature_vector)) {
   if (first_level && leave_out_one_level) {
```

```
        # avoid dummy trap and skip first level
        first_level=FALSE
      } else {
        data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
      }
    }
    # remove original feature
    data_set <- data_set[,setdiff(names(data_set),feature_name)]
  }
  return (data_set)
}

Titanic_dataset_temp <- Binarize_Features(data_set = Titanic_dataset, features_to_ignore = c('Name'))
str(Titanic_dataset_temp)

# CARET ON_LINE BOOK  http://topepo.github.io/caret/index.html

###################################################################################################
```

FACTOR 2

Load the functions

```
Fix_Date_Features <- function(data_set) {
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (feature_name in text_features) {
  feature_vector <- as.character(data_set[,feature_name])
  # assuming date pattern: '01/11/2012'
  date_pattern <- '[0-9][0-9]/[0-9][0-9]/[0-9][0-9][0-9][0-9]'
  if (max(nchar(feature_vector)) == 10) {
   if (sum(grepl(date_pattern, feature_vector)) > 0) {
    print(paste('Casting feature to date:',feature_name))
    data_set[,feature_name] <- as.Date(feature_vector, format="%d/%m/%Y")
   }
  }
```

```
 }
 return (data_set)
}



Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, features_to_ignore=c()) {
 # look for text entries that are mostly unique
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (f_name in setdiff(text_features, features_to_ignore)) {
  f_vector <- as.character(data_set[,f_name])
  # treat as raw text if data over minimum_precent_unique unique
  if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_unique_threshold)) {
   data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vector, " "), length)
   data_set[,paste0(f_name, '_character_count')] <- nchar(as.character(f_vector))
   data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`, 1)
   # remove orginal field
   data_set[,f_name] <- NULL
  }
```

```
 }
 return(data_set)
}



Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE) {
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (feature_name in setdiff(text_features, features_to_ignore)) {
  feature_vector <- as.character(data_set[,feature_name])
  # check that data has more than one level
  if (length(unique(feature_vector)) == 1)
   next
  # We set any non-data to text
  feature_vector[is.na(feature_vector)] <- 'NA'
  feature_vector[is.infinite(feature_vector)] <- 'INF'
  feature_vector[is.nan(feature_vector)] <- 'NAN'
  # loop through each level of a feature and create a new column
  first_level=TRUE
  for (newcol in unique(feature_vector)) {
   if (first_level && leave_out_one_level) {
```

```
        # avoid dummy trap and skip first level
        first_level=FALSE
      } else {
        data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
      }
    }
    # remove original feature
    data_set <- data_set[,setdiff(names(data_set),feature_name)]
  }
  return (data_set)
}
```

```
Titanic_dataset <- read.table('http://math.ucdenver.edu/RTutorial/titanic.txt', sep='\t', header=TRUE, stringsAsFactors = FALSE)
Titanic_dataset_temp <- Titanic_dataset
# fix date field if any
Titanic_dataset_temp <- Fix_Date_Features(data_set = Titanic_dataset_temp)
# extra quantative value out of text entires
```

```
Titanic_dataset_temp <- Get_Free_Text_Measures(data_set = Titanic_dataset_temp)

# binarize categories

Titanic_dataset_temp <- Binarize_Features(data_set = Titanic_dataset_temp, features_to_ignore = c(), leave_out_one_level = TRUE)
```

```
Titanic_dataset <- read.table('http://math.ucdenver.edu/RTutorial/titanic.txt', sep='\t', header=TRUE, stringsAsFactors = FALSE)

Titanic_dataset_temp <- Titanic_dataset

# fix date field if any

Titanic_dataset_temp <- Fix_Date_Features(data_set = Titanic_dataset_temp)

# extra quantative value out of text entires

Titanic_dataset_temp <- Get_Free_Text_Measures(data_set = Titanic_dataset_temp)

# get the Name_first_word feature

temp_vect <- Titanic_dataset_temp$Name_first_word

# only give us the top 20 most popular categories

popularity_count <- 20
```

```
# install.packages('dplyr')
```

```
library(dplyr)

##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
## filter, lag
##
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union

temp_vect <- data.frame(table(temp_vect)) %>% arrange(desc(Freq)) %>% head(popularity_count)
Titanic_dataset_temp$Name_first_word <- ifelse(Titanic_dataset_temp$Name_first_word %in% temp_vect$temp_vect,
                                     Titanic_dataset_temp$Name_first_word, 'Other')
print(head(Titanic_dataset_temp$Name_first_word,40))
```

```
# binarize categories

Titanic_dataset_temp <- Binarize_Features(data_set = Titanic_dataset_temp, features_to_ignore = c(), leave_out_one_level = TRUE)

head(Titanic_dataset_temp, 2)
```

```
## Age Survived Name_word_count Name_character_count PClass_2nd PClass_3rd

## 1 29 1 4 28 0 0

## 2 2 0 4 27 0 0

## Sex_male Name_first_word_Brown, Name_first_word_Carlsson,

## 1 0 0 0

## 2 0 0 0

## Name_first_word_Carter, Name_first_word_Fortune, Name_first_word_Van

## 1 0 0 0

## 2 0 0 0

## Name_first_word_Williams, Name_first_word_Davies, Name_first_word_Kelly,
```

1 0 0 0

2 0 0 0

Name_first_word_Andersson, Name_first_word_Asplund,

1 0 0

2 0 0

Name_first_word_Ford, Name_first_word_Goodwin,

1 0 0

2 0 0

Name_first_word_Johansson, Name_first_word_Johnson,

1 0 0

2 0 0

Name_first_word_Kink, Name_first_word_Lefebre, Name_first_word_Panula,

1 0 0 0

2 0 0 0

Name_first_word_Rice, Name_first_word_Sage, Name_first_word_Skoog,

1 0 0 0

2 0 0 0

```
Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE, max_level_count=20) {
 require(dplyr)
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (feature_name in setdiff(text_features, features_to_ignore)) {
  feature_vector <- as.character(data_set[,feature_name])
  # check that data has more than one level
  if (length(unique(feature_vector)) == 1)
   next
  # We set any non-data to text
  feature_vector[is.na(feature_vector)] <- 'NA'
  feature_vector[is.infinite(feature_vector)] <- 'INF'
  feature_vector[is.nan(feature_vector)] <- 'NAN'
  # only give us the top x most popular categories
  temp_vect <- data.frame(table(feature_vector)) %>% arrange(desc(Freq)) %>% head(max_level_count)
  feature_vector <- ifelse(feature_vector %in% temp_vect$feature_vector, feature_vector, 'Other')
  # loop through each level of a feature and create a new column
  first_level=TRUE
  for (newcol in unique(feature_vector)) {
   if (leave_out_one_level & first_level) {
    # avoid dummy trap and skip first level
```

```
        first_level=FALSE
        next
      }
      data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
    }
    # remove original feature
    data_set <- data_set[,setdiff(names(data_set),feature_name)]
  }
  return (data_set)
}


Titanic_dataset <- read.table('http://math.ucdenver.edu/RTutorial/titanic.txt', sep='\t', header=TRUE, stringsAsFactors = FALSE)
Titanic_dataset_temp <- Titanic_dataset
# fix date field if any
Titanic_dataset_temp <- Fix_Date_Features(data_set = Titanic_dataset_temp)
# extra quantative value out of text entires
Titanic_dataset_temp <- Get_Free_Text_Measures(data_set = Titanic_dataset_temp)
# binarize categories
Titanic_dataset_temp <- Binarize_Features(data_set = Titanic_dataset_temp, features_to_ignore = c(), leave_out_one_level = TRUE, max_level_count = 10)
```

```
######################################################################################################################

# IMPUTING MISSING DATA

mix_dataset <- data.frame(
  id=c(1,NA,3,4,5),
  mood=c(0,20,20,Inf,50),
  value=c(12.34, 32.2, NaN, 83.1, 8.32),
  outcome=c(1,1,0,0,0))
head(mix_dataset)
```

```
## id mood value outcome
## 1 1 0 12.34 1
## 2 NA 20 32.20 1
## 3 3 20 NaN 0
## 4 4 Inf 83.10 0
## 5 5 50 8.32 0
```

```
mix_dataset_temp <- mix_dataset
# where are the NAs?
is.na(mix_dataset_temp)
```

```
## id mood value outcome
## [1,] FALSE FALSE FALSE FALSE
## [2,] TRUE FALSE FALSE FALSE
## [3,] FALSE FALSE TRUE FALSE
## [4,] FALSE FALSE FALSE FALSE
## [5,] FALSE FALSE FALSE FALSE
```

```
# impute column:
```

```
mix_dataset_temp$id[is.na(mix_dataset_temp$id)] <- 0
mix_dataset_temp
```

```
## id mood value outcome
## 1 1 0 12.34 1
## 2 0 20 32.20 1
## 3 3 20 NaN 0
## 4 4 Inf 83.10 0
## 5 5 50 8.32 0
```

```
# impute with mean
mix_dataset_temp$value[is.nan(mix_dataset_temp$value)] <- mean(mix_dataset_temp$value, na.rm = TRUE)
mix_dataset_temp
```

```
## id mood value outcome
## 1 1 0 12.34 1
```

```
## 2 0 20 32.20 1
## 3 3 20 33.99 0
## 4 4 Inf 83.10 0
## 5 5 50 8.32 0
```

```
Impute_Features <- function(data_set, features_to_ignore=c(),
                            use_mean_instead_of_0=TRUE,
                            mark_NAs=FALSE,
                            remove_zero_variance=FALSE) {
 for (feature_name in setdiff(names(data_set), features_to_ignore)) {
  print(feature_name)
  # remove any fields with zero variance
  if (remove_zero_variance) {
   if (length(unique(data_set[, feature_name]))==1) {
    data_set[, feature_name] <- NULL
    next
   }
  }
  if (mark_NAs) {
```

```
    # note each field that contains missing or bad data
    if (any(is.na(data_set[,feature_name]))) {
     # create binary column before imputing
     newName <- paste0(feature_name, '_NA')
     data_set[,newName] <- as.integer(ifelse(is.na(data_set[,feature_name]),1,0)) }
    if (any(is.infinite(data_set[,feature_name]))) {
     newName <- paste0(feature_name, '_inf')
     data_set[,newName] <- as.integer(ifelse(is.infinite(data_set[,
                                   feature_name]),1,0)) }
   }
   if (use_mean_instead_of_0) {
    data_set[is.infinite(data_set[,feature_name]),feature_name] <- NA
    data_set[is.na(data_set[,feature_name]),feature_name] <- mean(data_set[,feature_name], na.rm=TRUE)
   } else {
    data_set[is.na(data_set[,feature_name]),feature_name] <- 0
    data_set[is.infinite(data_set[,feature_name]),feature_name] <- 0
   }
  }
  return(data_set)
}
```

mix_dataset_temp <- Impute_Features(mix_dataset, use_mean_instead_of_0 = TRUE, mark_NAs = TRUE)

[1] "id"

[1] "mood"

[1] "value"

[1] "outcome"

head(mix_dataset_temp)

id mood value outcome id_NA mood_inf value_NA

```
## 1 1.00 0.0 12.34 1 0 0 0

## 2 3.25 20.0 32.20 1 1 0 0

## 3 3.00 20.0 33.99 0 0 0 1

## 4 4.00 22.5 83.10 0 0 1 0

## 5 5.00 50.0 8.32 0 0 0 0



#####################################################################################################################################################################
###3



#   PIPELINE CHECK



Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, features_to_ignore=c()) {

 # look for text entries that are mostly unique

 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))

 for (f_name in setdiff(text_features, features_to_ignore)) {

  f_vector <- as.character(data_set[,f_name])
```

```
    # treat as raw text if data over minimum_precent_unique unique
    if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_unique_threshold)) {
      data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vector, " "), length)
      data_set[,paste0(f_name, '_character_count')] <- nchar(as.character(f_vector))
      data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`, 1)
      # remove orginal field
      data_set[,f_name] <- NULL
    }
  }
  return(data_set)
}


Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE, max_level_count=20) {
  require(dplyr)
  text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
  for (feature_name in setdiff(text_features, features_to_ignore)) {
    feature_vector <- as.character(data_set[,feature_name])
    # check that data has more than one level
    if (length(unique(feature_vector)) == 1)
```

```
  next
# We set any non-data to text
feature_vector[is.na(feature_vector)] <- 'NA'
feature_vector[is.infinite(feature_vector)] <- 'INF'
feature_vector[is.nan(feature_vector)] <- 'NAN'
# only give us the top x most popular categories
temp_vect <- data.frame(table(feature_vector)) %>% arrange(desc(Freq)) %>% head(max_level_count)
feature_vector <- ifelse(feature_vector %in% temp_vect$feature_vector, feature_vector, 'Other')
# loop through each level of a feature and create a new column
first_level=TRUE
for (newcol in unique(feature_vector)) {
  if (leave_out_one_level & first_level) {
    # avoid dummy trap and skip first level
    first_level=FALSE
    next
  }
  data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
}
# remove original feature
data_set <- data_set[,setdiff(names(data_set),feature_name)]
```

```
  }
  return (data_set)
}


Impute_Features <- function(data_set, features_to_ignore=c(),
                            use_mean_instead_of_0=TRUE,
                            mark_NAs=FALSE,
                            remove_zero_variance=FALSE) {
  for (feature_name in setdiff(names(data_set), features_to_ignore)) {
    print(feature_name)
    # remove any fields with zero variance
    if (remove_zero_variance) {
      if (length(unique(data_set[, feature_name]))==1) {
        data_set[, feature_name] <- NULL
        next
      }
    }
    if (mark_NAs) {
      # note each field that contains missing or bad data
```

```
    if (any(is.na(data_set[,feature_name]))) {
     # create binary column before imputing
     newName <- paste0(feature_name, '_NA')
     data_set[,newName] <- as.integer(ifelse(is.na(data_set[,feature_name]),1,0)) }
    if (any(is.infinite(data_set[,feature_name]))) {
     newName <- paste0(feature_name, '_inf')
     data_set[,newName] <- as.integer(ifelse(is.infinite(data_set[, feature_name]),1,0)) }
   }
   if (use_mean_instead_of_0) {
    data_set[is.infinite(data_set[,feature_name]),feature_name] <- NA
    data_set[is.na(data_set[,feature_name]),feature_name] <- mean(data_set[,feature_name], na.rm=TRUE)
   } else {
    data_set[is.na(data_set[,feature_name]),feature_name] <- 0
    data_set[is.infinite(data_set[,feature_name]),feature_name] <- 0
   }
  }
  return(data_set)
}
```

```
mix_dataset <- data.frame(
  ids=c(1,NA,3,4,5),
  some_dates = c('01/11/2012','04/12/2012','28/02/2013','17/06/2014','08/03/2015'),
  mood=c(0,20,20,Inf,50),
  some_real_numbers = c(12.34, 32.2, NaN, 83.1, 8.32),
  some_text = c('sentence one','sentence two', 'mixing it up', 'sentence four', 'sentence five'))

head(mix_dataset)
```

```
## ids some_dates mood some_real_numbers some_text
## 1 1 01/11/2012 0 12.34 sentence one
## 2 NA 04/12/2012 20 32.20 sentence two
## 3 3 28/02/2013 20 NaN mixing it up
## 4 4 17/06/2014 Inf 83.10 sentence four
```

```
## 5 5 08/03/2015 50 8.32 sentence five

library(readr)
write_csv(mix_dataset, 'mix_dataset.csv')

# take a peek at the data
readLines('mix_dataset.csv', n=3)

## [1] "ids,some_dates,mood,some_real_numbers,some_text"
## [2] "1,01/11/2012,0,12.34,sentence one"
## [3] "NA,04/12/2012,20,32.2,sentence two"

# pick your reader
library(data.table)
mix_dataset <- fread('mix_dataset.csv', data.table = FALSE)
str(mix_dataset)
```

```
# format date field to be R compliant

mix_dataset$some_dates <- as.Date(mix_dataset$some_dates, format="%d/%m/%Y")

str(mix_dataset$some_dates)


## Date[1:5], format: "2012-11-01" "2012-12-04" "2013-02-28" "2014-06-17" ...


class(mix_dataset)
```

```
# extra quantative value out of text entires

mix_dataset <- Get_Free_Text_Measures(data_set = mix_dataset)

head(mix_dataset,2)
```

```
## ids some_dates mood some_real_numbers some_text_word_count
```

```
## 1 1 2012-11-01 0 12.34 2
## 2 NA 2012-12-04 20 32.20 2
## some_text_character_count some_text_first_word
## 1 12 sentence
## 2 12 sentence
```

```
# binarize categories
mix_dataset <- Binarize_Features(data_set = mix_dataset, features_to_ignore = c(), leave_out_one_level = TRUE)
```

```
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:data.table':
##
## between, last
##
## The following objects are masked from 'package:stats':
##
```

```
## filter, lag
##
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union
```

```
head(mix_dataset, 2)
```

```
## ids some_dates mood some_real_numbers some_text_word_count
## 1 1 2012-11-01 0 12.34 2
## 2 NA 2012-12-04 20 32.20 2
## some_text_character_count some_text_first_word_mixing
## 1 12 0
## 2 12 0
```

```
# impute missing data using 0
```

```
mix_dataset <- Impute_Features(mix_dataset, use_mean_instead_of_0 = FALSE, features_to_ignore = c('some_dates'))
```

```
## [1] "ids"
## [1] "mood"
## [1] "some_real_numbers"
## [1] "some_text_word_count"
## [1] "some_text_character_count"
## [1] "some_text_first_word_mixing"
```

```
mix_dataset
```

```
## ids some_dates mood some_real_numbers some_text_word_count
## 1 1 2012-11-01 0 12.34 2
## 2 0 2012-12-04 20 32.20 2
## 3 3 2013-02-28 20 0.00 3
## 4 4 2014-06-17 0 83.10 2
## 5 5 2015-03-08 50 8.32 2
## some_text_character_count some_text_first_word_mixing
```

```
## 1 12 0

## 2 12 0

## 3 12 1

## 4 13 0

## 5 13 0


mix_dataset <- Impute_Features(mix_dataset, use_mean_instead_of_0 = TRUE, features_to_ignore = c('some_dates'))

mix_dataset


####################################################################################################################



# NEARZEROVARIANCE


library(caret)


mix_dataset <- data.frame(
     id=sample(1:100, 100, replace = F),
     value=runif(100,1.0, 55.5),
     no_varaiance=rep(1,100)
```

```
)

summary(mix_dataset)

# id            value        no_varaiance
# Min.   :  1.00   Min.   : 1.168   Min.   :1
# 1st Qu.: 25.75   1st Qu.: 9.995   1st Qu.:1
# Median : 50.50   Median :27.633   Median :1
# Mean   : 50.50   Mean   :26.988   Mean   :1
# 3rd Qu.: 75.25   3rd Qu.:42.664   3rd Qu.:1
# Max.   :100.00   Max.   :55.036   Max.   :1

nearZeroVar(mix_dataset, saveMetrics = TRUE)


# freqRatio percentUnique zeroVar   nzv
# id                 1          100   FALSE FALSE
# value              1          100   FALSE FALSE
# no_varaiance       0            1    TRUE  TRUE
```

```
mix_dataset$little_varaiance <- c(rep(1,98), 2, 3)

summary(mix_dataset)

# id          value        no_varaiance little_varaiance
# Min.   :  1.00   Min.   : 1.168   Min.   :1     Min.   :1.00
# 1st Qu.: 25.75   1st Qu.: 9.995   1st Qu.:1     1st Qu.:1.00
# Median : 50.50   Median :27.633   Median :1     Median :1.00
# Mean   : 50.50   Mean   :26.988   Mean   :1     Mean   :1.03
# 3rd Qu.: 75.25   3rd Qu.:42.664   3rd Qu.:1     3rd Qu.:1.00
# Max.   :100.00   Max.   :55.036   Max.   :1     Max.   :3.00

nearZeroVar(mix_dataset, saveMetrics = TRUE)

# freqRatio percentUnique zeroVar   nzv
# id                 1          100   FALSE FALSE
# value              1          100   FALSE FALSE
```

```
# no_varaiance           0           1    TRUE  TRUE
# little_varaiance      98           3   FALSE  TRUE
```

```
nzv <- nearZeroVar(mix_dataset, saveMetrics = TRUE)
nzv$percentUnique
```

```
# [1] 100 100   1   3
```

```
##########################################################################################################################
```

```
# ENGINEERING DATES - GETTING ADDITIONAL FEATURES OUT OF DATES.
```

```
print(as.numeric(as.Date('1970-01-01')))
```

```
## [1] 0
```

```
Feature_Engineer_Dates <- function(data_set, remove_original_date=TRUE) {
 require(lubridate)
 data_set <- data.frame(data_set)
 date_features <- names(data_set[sapply(data_set, is.Date)])
 for (feature_name in date_features) {
   data_set[,paste0(feature_name,'_DateInt')] <- as.numeric(data_set[,feature_name])
   data_set[,paste0(feature_name,'_Month')] <- as.integer(format(data_set[, feature_name], "%m"))
   data_set[,paste0(feature_name,'_ShortYear')] <- as.integer(format(data_set[,feature_name], "%y"))
   data_set[,paste0(feature_name,'_LongYear')] <- as.integer(format(data_set[,feature_name], "%Y"))
   data_set[,paste0(feature_name,'_Day')] <- as.integer(format(data_set[,feature_name], "%d"))
   # week day number requires first pulling the weekday label, creating the 7 week day levels, and casting to integer
   data_set[,paste0(feature_name,'_WeekDayNumber')] <- as.factor(weekdays(data_set[,feature_name]))
   levels(data_set[,paste0(feature_name,'_WeekDayNumber')]) <- list(Monday=1, Tuesday=2, Wednesday=3, Thursday=4, Friday=5, Saturday=6, Sunday=7)
   data_set[,paste0(feature_name,'_WeekDayNumber')] <- as.integer(data_set[,paste0(feature_name,'_WeekDayNumber')])
```

```
    data_set[,paste0(feature_name,'_IsWeekend')] <- as.numeric(grepl("Saturday|Sunday", weekdays(data_set[,feature_name])))
    data_set[,paste0(feature_name,'_YearDayCount')] <- yday(data_set[,feature_name])
    data_set[,paste0(feature_name,'_Quarter')] <- lubridate::quarter(data_set[,feature_name], with_year = FALSE)
    data_set[,paste0(feature_name,'_Quarter')] <- lubridate::quarter(data_set[,feature_name], with_year = TRUE)
    if (remove_original_date)
      data_set[, feature_name] <- NULL
  }
  return(data_set)
}
```

```
mix_dataset <- data.frame(
  id=c(10,20,30,40,50),
  gender=c('male','female','female','male','female'),
  some_date=c('2012-01-12','2012-01-12','2012-12-01','2012-05-30','2013-12-12'),
  value=c(12.34, 32.2, 24.3, 83.1, 8.32),
  outcome=c(1,1,0,0,0))
```

```
library(readr)

write_csv(mix_dataset, 'mix_dataset.csv')
mix_dataset <- read_csv('mix_dataset.csv')
mix_dataset <- Feature_Engineer_Dates(mix_dataset)

## Loading required package: lubridate
head(mix_dataset)

## id gender value outcome some_date_DateInt some_date_Month
## 1 10 male 12.34 1 15351 1
## 2 20 female 32.20 1 15351 1
## 3 30 female 24.30 0 15675 12
## 4 40 male 83.10 0 15490 5
## 5 50 female 8.32 0 16051 12
## some_date_ShortYear some_date_LongYear some_date_Day
## 1 12 2012 12
```

2 12 2012 12

3 12 2012 1

4 12 2012 30

5 13 2013 12

some_date_WeekDayNumber some_date_IsWeekend some_date_YearDayCount

1 4 0 12

2 4 0 12

3 6 1 336

4 3 0 151

5 4 0 346

some_date_Quarter

1 2012.1

2 2012.1

3 2012.4

4 2012.2

5 2013.4

###

```
# NUMERICAL ENGINEERING - INTEGERS AND REAL NUMBERS

## INTEGER

print(is.integer(1))

## [1] FALSE

print(class(1))

## [1] "numeric"

print(class(1L))

## [1] "integer"

mix_dataset <- data.frame(
```

```
 id=c(1,2,3,4,5),
 mood=c(0,20,20,40,50),
 value=c(12.34, 32.2, 24.3, 83.1, 8.32),
 outcome=c(1,1,0,0,0))



library(readr)

write_csv(mix_dataset, 'mix_dataset.csv')

mix_dataset <- read_csv('mix_dataset.csv')



Feature_Engineer_Integers <- function(data_set, features_to_ignore=c()) {
 require(infotheo)
 data_set <- data.frame(data_set)
 for (feature_name in setdiff(names(data_set), features_to_ignore)) {
  if (class(data_set[,feature_name])=='numeric' | class(data_set[,feature_name])=='integer') {
   feature_vector <- data_set[,feature_name]
   if (all((feature_vector - round(feature_vector)) == 0)) {
```

```
        # make sure we have more than 2 values excluding NAs
        if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 2) {
         print(feature_name)
         data_set[,paste0(feature_name,'_IsZero')] <- ifelse(data_set[,feature_name]==0,1,0)
         data_set[,paste0(feature_name,'_IsPositive')] <- ifelse(data_set[,feature_name]>=0,1,0)
         # separate data into two bins
         data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=2)
         data_set[,paste0(feature_name,'_2Bins')] <- data_discretized$X
         if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 4) {
          # try 4 bins
          data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=4)
          data_set[,paste0(feature_name,'_4Bins')] <- data_dis
          cretized$X
         }
        }
       }
      }
     }
     return (data_set)
    }
```

```
mix_dataset <- read_csv('mix_dataset.csv')

Feature_Engineer_Integers(mix_dataset, features_to_ignore=c('id'))

## Loading required package: infotheo

## [1] "mood"

## id mood value outcome mood_IsZero mood_IsPositive mood_2Bins
## 1 1 0 12.34 1 1 1 1
## 2 2 20 32.20 1 0 1 1
## 3 3 20 24.30 0 0 1 1
## 4 4 40 83.10 0 0 1 2
## 5 5 50 8.32 0 0 1 2
```

```
## NUMBERS

Feature_Engineer_Numbers <- function(data_set, features_to_ignore=c()) {
 require(infotheo)
 data_set <- data.frame(data_set)
 date_features <- setdiff(names(data_set[sapply(data_set, is.numeric)]), features_to_ignore)
 for (feature_name in date_features) {
  feature_vector <- data_set[,feature_name]
  if (is.integer(feature_vector) | is.numeric(feature_vector)) {
   if (any((feature_vector - round(feature_vector)) != 0)) {
    # make sure we have more than 2 values excluding NAs
    if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 2) {
     print(feature_name)
     # polynomial transformation
     poly_vector <- poly(x=feature_vector, degree = 2)
     data_set[,paste0(feature_name, "_poly1")] <- poly_vector
     [,1]
     data_set[,paste0(feature_name, "_poly2")] <- poly_vector
```

```
        [,2]
        # log transform
        data_set[,paste0(feature_name, "_log")] <- log(x = feature_vector)
        # exponential transform
        data_set[,paste0(feature_name, "_exp")] <- exp(x = feature_vector)
        # rounding
        data_set[,paste0(feature_name, "_rnd")] <- round(x = feature_vector, digits = 0)
        # binning into 2 bins
        data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=2)
        data_set[,paste0(feature_name,'_2Bins')] <- data_discretized$X
      }
    }
  }
 }
 return(data_set)
}



mix_dataset <- data.frame(
```

```
  id=sample(1:100, 100, replace=F),
  value=runif(100, 1.0, 55.5)
)

write_csv(mix_dataset, 'mix_dataset.csv')

mix_dataset <- read_csv('mix_dataset.csv')

head(Feature_Engineer_Numbers(mix_dataset, features_to_ignore=c()))

## [1] "value"

## id value value_poly1 value_poly2 value_log value_exp value_rnd
## 1 87 19.974386 -0.043198041 -0.07035281 2.9944507 4.728959e+08 20
## 2 98 51.824357 0.184762399 0.19461439 3.9478603 3.213900e+22 52
## 3 25 18.063778 -0.056872870 -0.05327673 2.8939087 6.998408e+07 18
## 4 4 2.469639 -0.168485118 0.22562557 0.9040719 1.181818e+01 2
## 5 53 31.098497 0.036420785 -0.09565801 3.4371595 3.205574e+13 31
## 6 51 26.661576 0.004664319 -0.10073098 3.2832234 3.792934e+11 27
```

```
## value_2Bins
## 1 1
## 2 2
## 3 1
## 4 1
## 5 2
## 6 2
```

```
#########################################################################################################################################33

# PIPELINE CHECK

Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE, max_level_count=20) {
  require(dplyr)
```

```
text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
for (feature_name in setdiff(text_features, features_to_ignore)) {
 feature_vector <- as.character(data_set[,feature_name])
 # check that data has more than one level
 if (length(unique(feature_vector)) == 1)
  next
 # We set any non-data to text
 feature_vector[is.na(feature_vector)] <- 'NA'
 feature_vector[is.infinite(feature_vector)] <- 'INF'
 feature_vector[is.nan(feature_vector)] <- 'NAN'
 # only give us the top x most popular categories
 temp_vect <- data.frame(table(feature_vector)) %>% arrange(desc(Freq)) %>% head(max_level_count)
 feature_vector <- ifelse(feature_vector %in% temp_vect$feature_vector, feature_vector, 'Other')
 # loop through each level of a feature and create a new column
 first_level=TRUE
 for (newcol in unique(feature_vector)) {
  if (leave_out_one_level & first_level) {
   # avoid dummy trap and skip first level
   first_level=FALSE
   next
```

```
    }
    data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
   }
   # remove original feature
   data_set <- data_set[,setdiff(names(data_set),feature_name)]
  }
  return (data_set)
}



Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, features_to_ignore=c()) {
  # look for text entries that are mostly unique
  text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
  for (f_name in setdiff(text_features, features_to_ignore)) {
   f_vector <- as.character(data_set[,f_name])
   # treat as raw text if data over minimum_precent_unique unique
   if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_unique_threshold)) {
    data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vector, " "), length)
    data_set[,paste0(f_name, '_character_count')] <- nchar(as.character(f_vector))
```

```
      data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`, 1)
      # remove orginal field
      data_set[,f_name] <- NULL
    }
  }
  return(data_set)
}



Impute_Features <- function(data_set, features_to_ignore=c(),
                            use_mean_instead_of_0=TRUE,
                            mark_NAs=FALSE,
                            remove_zero_variance=FALSE) {
  for (feature_name in setdiff(names(data_set), features_to_ignore)) {
    print(feature_name)
    # remove any fields with zero variance
    if (remove_zero_variance) {
      if (length(unique(data_set[, feature_name]))==1) {
        data_set[, feature_name] <- NULL
```

```
    next
  }
}
if (mark_NAs) {
  # note each field that contains missing or bad data
  if (any(is.na(data_set[,feature_name]))) {
    # create binary column before imputing
    newName <- paste0(feature_name, '_NA')
    data_set[,newName] <- as.integer(ifelse(is.na(data_set[,feature_name]),1,0)) }
  if (any(is.infinite(data_set[,feature_name]))) {
    newName <- paste0(feature_name, '_inf')
    data_set[,newName] <- as.integer(ifelse(is.infinite(data_set[,feature_name]),1,0)) }
}
if (use_mean_instead_of_0) {
  data_set[is.infinite(data_set[,feature_name]),feature_name] <- NA
  data_set[is.na(data_set[,feature_name]),feature_name] <- mean(data_set[,feature_name], na.rm=TRUE)
} else {
  data_set[is.na(data_set[,feature_name]),feature_name] <- 0
  data_set[is.infinite(data_set[,feature_name]),feature_name] <- 0
}
```

```
  }
  return(data_set)
}



Feature_Engineer_Dates <- function(data_set, remove_original_date=TRUE) {
  require(lubridate)
  data_set <- data.frame(data_set)
  date_features <- names(data_set[sapply(data_set, is.Date)])
  for (feature_name in date_features) {
    data_set[,paste0(feature_name,'_DateInt')] <- as.numeric(data_set[,feature_name])
    data_set[,paste0(feature_name,'_Month')] <- as.integer(format(data_set[,feature_name], "%m"))
    data_set[,paste0(feature_name,'_ShortYear')] <- as.integer(format(data_set[,feature_name], "%y"))
    data_set[,paste0(feature_name,'_LongYear')] <- as.integer(format(data_set[,feature_name], "%Y"))
    data_set[,paste0(feature_name,'_Day')] <- as.integer(format(data_set[,feature_name], "%d"))
    # week day number requires first pulling the weekday label, creating the 7 week day levels, and casting to integer
    data_set[,paste0(feature_name,'_WeekDayNumber')] <- as.factor(weekdays(data_set[,feature_name]))
    levels(data_set[,paste0(feature_name,'_WeekDayNumber')]) <- list(Monday=1, Tuesday=2, Wednesday=3, Thursday=4, Friday=5, Saturday=6, Sunday=7)
    data_set[,paste0(feature_name,'_WeekDayNumber')] <- as.integer(data_set[,paste0(feature_name,'_WeekDayNumber')])
```

```
    data_set[,paste0(feature_name,'_IsWeekend')] <- as.numeric(grepl("Saturday|Sunday", weekdays(data_set[,feature_name])))
    data_set[,paste0(feature_name,'_YearDayCount')] <- yday(data_set[,feature_name])
    data_set[,paste0(feature_name,'_Quarter')] <- lubridate::quarter(data_set[,feature_name], with_year = FALSE)
    data_set[,paste0(feature_name,'_Quarter')] <- lubridate::quarter(data_set[,feature_name], with_year = TRUE)
    if (remove_original_date)
      data_set[, feature_name] <- NULL
  }
  return(data_set)
}
```

```
Feature_Engineer_Integers <- function(data_set, features_to_ignore=c()) {
  require(infotheo)
  data_set <- data.frame(data_set)
  for (feature_name in setdiff(names(data_set), features_to_ignore)) {
    if (class(data_set[,feature_name])=='numeric' | class(data_set[,feature_name])=='integer') {
      feature_vector <- data_set[,feature_name]
      if (all((feature_vector - round(feature_vector)) == 0)) {
        # make sure we have more than 2 values excluding NAs
```

```
    if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 2) {
     print(feature_name)
     data_set[,paste0(feature_name,'_IsZero')] <- ifelse(data_set[,feature_name]==0,1,0)
     data_set[,paste0(feature_name,'_IsPositive')] <- ifelse(data_set[,feature_name]>=0,1,0)
     # separate data into two bins
     data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=2)
     data_set[,paste0(feature_name,'_2Bins')] <- data_discretized$X
     if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 4) {
      # try 4 bins
      data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=4)
      data_set[,paste0(feature_name,'_4Bins')] <- data_discretized$X
     }
    }
   }
  }
 }
 return (data_set)
}
```

```
Feature_Engineer_Numbers <- function(data_set, features_to_ignore=c()) {
 require(infotheo)
 data_set <- data.frame(data_set)
 date_features <- setdiff(names(data_set[sapply(data_set, is.numeric)]), features_to_ignore)
 for (feature_name in date_features) {
  feature_vector <- data_set[,feature_name]
  if (is.integer(feature_vector) | is.numeric(feature_vector)) {
   if (any((feature_vector - round(feature_vector)) != 0)) {
    # make sure we have more than 2 values excluding NAs
    if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 2) {
     print(feature_name)
     # polynomial transformation
     poly_vector <- poly(x=feature_vector, degree = 2)
     data_set[,paste0(feature_name, "_poly1")] <- poly_vector
     [,1]
     data_set[,paste0(feature_name, "_poly2")] <- poly_vector
     [,2]
     # log transform
     data_set[,paste0(feature_name, "_log")] <- log(x = feature_vector)
     # exponential transform
```

```
        data_set[,paste0(feature_name, "_exp")] <- exp(x = feature_vector)
        # rounding
        data_set[,paste0(feature_name, "_rnd")] <- round(x = feature_vector, digits = 0)
        # binning into 2 bins
        data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=2)
        data_set[,paste0(feature_name,'_2Bins')] <- data_discreti
        zed$X
       }
      }
    }
  }
  return(data_set)
}



mix_dataset <- data.frame(
  id=c(1,2,3,4,5),
  gender=c('male','female','female','male','female'),
  some_date=c('2012-01-01','2013-01-01','2014-01-01','2015-01-01','2016-01-01'),
```

```
  mood=c(0,20,20,NA,50),
  value=c(12.34, 32.2, 24.3, 83.1, 8.32),
  outcome=c(1,1,0,0,0))


library(readr)
write_csv(mix_dataset, 'mix_dataset.csv')
mix_dataset <- as.data.frame(read_csv('mix_dataset.csv'))


# automated pipeline

mix_dataset <- Get_Free_Text_Measures(data_set = mix_dataset)


mix_dataset <- Binarize_Features(data_set = mix_dataset, leave_out_one_level = FALSE)


## Loading required package: dplyr
##
```

```
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
## filter, lag
##
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union
```

```
mix_dataset <- Impute_Features(data_set = mix_dataset)
```

```
## [1] "id"
## [1] "some_date"
## [1] "mood"
## [1] "value"
## [1] "outcome"
## [1] "gender_male"
```

```
## [1] "gender_female"

mix_dataset <- Feature_Engineer_Dates(data_set = mix_dataset)

## Loading required package: lubridate

mix_dataset <- Feature_Engineer_Integers(data_set = mix_dataset)

## Loading required package: infotheo
## [1] "id"
## [1] "some_date_DateInt"
## [1] "some_date_ShortYear"
## [1] "some_date_LongYear"
## [1] "some_date_WeekDayNumber"

mix_dataset <- Feature_Engineer_Numbers(data_set = mix_dataset)
```

[1] "mood"

[1] "value"

[1] "some_date_Quarter"

head(mix_dataset,2)

id mood value outcome gender_male gender_female some_date_DateInt

1 1 0 12.34 1 1 0 15340

2 2 20 32.20 1 0 1 15706

some_date_Month some_date_ShortYear some_date_LongYear some_date_Day

1 1 12 2012 1

2 1 13 2013 1

some_date_WeekDayNumber some_date_IsWeekend some_date_YearDayCount

1 7 1 1

2 2 0 1

some_date_Quarter id_IsZero id_IsPositive id_2Bins id_4Bins

1 2012.1 0 1 1 1

2 2013.1 0 1 1 1

some_date_DateInt_IsZero some_date_DateInt_IsPositive

1 0 1

2 0 1

some_date_DateInt_2Bins some_date_DateInt_4Bins

1 1 1

2 1 1

some_date_ShortYear_IsZero some_date_ShortYear_IsPositive

1 0 1

2 0 1

some_date_ShortYear_2Bins some_date_ShortYear_4Bins

1 1 1

2 1 1

some_date_LongYear_IsZero some_date_LongYear_IsPositive

1 0 1

2 0 1

some_date_LongYear_2Bins some_date_LongYear_4Bins

1 1 1

2 1 1

some_date_WeekDayNumber_IsZero some_date_WeekDayNumber_IsPositive

1 0 1

2 0 1

some_date_WeekDayNumber_2Bins some_date_WeekDayNumber_4Bins mood_poly1

1 2 4 -0.630126

2 1 1 -0.070014

mood_poly2 mood_log mood_exp mood_rnd mood_2Bins value_poly1

1 0.6323510 -Inf 1 0 1 -0.327724828

2 -0.3495951 2.995732 485165195 20 1 0.002460596

value_poly2 value_log value_exp value_rnd value_2Bins

1 0.2483090 2.512846 2.286620e+05 12 1

2 -0.6629308 3.471966 9.644558e+13 32 2

some_date_Quarter_poly1 some_date_Quarter_poly2 some_date_Quarter_log

1 -0.6324555 0.5345225 7.606934

2 -0.3162278 -0.2672612 7.607431

some_date_Quarter_exp some_date_Quarter_rnd some_date_Quarter_2Bins

1 Inf 2012 1

2 Inf 2013 1

summary(mix_dataset)

id mood value outcome gender_male

Min. :1 Min. : 0.0 Min. : 8.32 Min. :0.0 Min. :0.0

1st Qu.:2 1st Qu.:20.0 1st Qu.:12.34 1st Qu.:0.0 1st Qu.:0.0

Median :3 Median :20.0 Median :24.30 Median :0.0 Median :0.0

Mean :3 Mean :22.5 Mean :32.05 Mean :0.4 Mean :0.4

3rd Qu.:4 3rd Qu.:22.5 3rd Qu.:32.20 3rd Qu.:1.0 3rd Qu.:1.0

Max. :5 Max. :50.0 Max. :83.10 Max. :1.0 Max. :1.0

gender_female some_date_DateInt some_date_Month some_date_ShortYear

Min. :0.0 Min. :15340 Min. :1 Min. :12

1st Qu.:0.0 1st Qu.:15706 1st Qu.:1 1st Qu.:13

Median :1.0 Median :16071 Median :1 Median :14

Mean :0.6 Mean :16071 Mean :1 Mean :14

3rd Qu.:1.0 3rd Qu.:16436 3rd Qu.:1 3rd Qu.:15

Max. :1.0 Max. :16801 Max. :1 Max. :16

some_date_LongYear some_date_Day some_date_WeekDayNumber

Min. :2012 Min. :1 Min. :2.0

1st Qu.:2013 1st Qu.:1 1st Qu.:3.0

Median :2014 Median :1 Median :4.0

Mean :2014 Mean :1 Mean :4.2

3rd Qu.:2015 3rd Qu.:1 3rd Qu.:5.0

Max. :2016 Max. :1 Max. :7.0

some_date_IsWeekend some_date_YearDayCount some_date_Quarter id_IsZero

Min. :0.0 Min. :1 Min. :2012 Min. :0

1st Qu.:0.0 1st Qu.:1 1st Qu.:2013 1st Qu.:0

Median :0.0 Median :1 Median :2014 Median :0

Mean :0.2 Mean :1 Mean :2014 Mean :0

3rd Qu.:0.0 3rd Qu.:1 3rd Qu.:2015 3rd Qu.:0

Max. :1.0 Max. :1 Max. :2016 Max. :0

id_IsPositive id_2Bins id_4Bins some_date_DateInt_IsZero

Min. :1 Min. :1.0 Min. :1.0 Min. :0

1st Qu.:1 1st Qu.:1.0 1st Qu.:1.0 1st Qu.:0

Median :1 Median :1.0 Median :3.0 Median :0

Mean :1 Mean :1.4 Mean :2.6 Mean :0

3rd Qu.:1 3rd Qu.:2.0 3rd Qu.:4.0 3rd Qu.:0

Max. :1 Max. :2.0 Max. :4.0 Max. :0

some_date_DateInt_IsPositive some_date_DateInt_2Bins

Min. :1 Min. :1.0

1st Qu.:1 1st Qu.:1.0

Median :1 Median :1.0

Mean :1 Mean :1.4

3rd Qu.:1 3rd Qu.:2.0

Max. :1 Max. :2.0

some_date_DateInt_4Bins some_date_ShortYear_IsZero

Min. :1.0 Min. :0

1st Qu.:1.0 1st Qu.:0

Median :3.0 Median :0

Mean :2.6 Mean :0

3rd Qu.:4.0 3rd Qu.:0

Max. :4.0 Max. :0

some_date_ShortYear_IsPositive some_date_ShortYear_2Bins

Min. :1 Min. :1.0

1st Qu.:1 1st Qu.:1.0

Median :1 Median :1.0

Mean :1 Mean :1.4

3rd Qu.:1 3rd Qu.:2.0

Max. :1 Max. :2.0

some_date_ShortYear_4Bins some_date_LongYear_IsZero

Min. :1.0 Min. :0

1st Qu.:1.0 1st Qu.:0

Median :3.0 Median :0

Mean :2.6 Mean :0

3rd Qu.:4.0 3rd Qu.:0

Max. :4.0 Max. :0

some_date_LongYear_IsPositive some_date_LongYear_2Bins

Min. :1 Min. :1.0

1st Qu.:1 1st Qu.:1.0

Median :1 Median :1.0

Mean :1 Mean :1.4

3rd Qu.:1 3rd Qu.:2.0

Max. :1 Max. :2.0

some_date_LongYear_4Bins some_date_WeekDayNumber_IsZero

Min. :1.0 Min. :0

1st Qu.:1.0 1st Qu.:0

Median :3.0 Median :0

Mean :2.6 Mean :0

3rd Qu.:4.0 3rd Qu.:0

Max. :4.0 Max. :0

some_date_WeekDayNumber_IsPositive some_date_WeekDayNumber_2Bins

Min. :1 Min. :1.0

1st Qu.:1 1st Qu.:1.0

Median :1 Median :1.0

Mean :1 Mean :1.4

3rd Qu.:1 3rd Qu.:2.0

Max. :1 Max. :2.0

some_date_WeekDayNumber_4Bins mood_poly1 mood_poly2

Min. :1.0 Min. :-0.63013 Min. :-0.3870

1st Qu.:1.0 1st Qu.:-0.07001 1st Qu.:-0.3496

Median :3.0 Median :-0.07001 Median :-0.3496

Mean :2.6 Mean : 0.00000 Mean : 0.0000

3rd Qu.:4.0 3rd Qu.: 0.00000 3rd Qu.: 0.4538

Max. :4.0 Max. : 0.77015 Max. : 0.6324

mood_log mood_exp mood_rnd mood_2Bins

Min. :-Inf Min. :1.000e+00 Min. : 0.0 Min. :1.0

1st Qu.: 3 1st Qu.:4.852e+08 1st Qu.:20.0 1st Qu.:1.0

Median : 3 Median :4.852e+08 Median :20.0 Median :1.0

Mean :-Inf Mean :1.037e+21 Mean :22.4 Mean :1.4

3rd Qu.: 3 3rd Qu.:5.911e+09 3rd Qu.:22.0 3rd Qu.:2.0

Max. : 4 Max. :5.185e+21 Max. :50.0 Max. :2.0

value_poly1 value_poly2 value_log value_exp

Min. :-0.394560 Min. :-0.6629 Min. :2.119 Min. :4.105e+03

1st Qu.:-0.327725 1st Qu.:-0.3865 1st Qu.:2.513 1st Qu.:2.287e+05

Median :-0.128882 Median : 0.2483 Median :3.190 Median :3.576e+10

Mean : 0.000000 Mean : 0.0000 Mean :3.143 Mean :2.460e+35

3rd Qu.: 0.002461 3rd Qu.: 0.2809 3rd Qu.:3.472 3rd Qu.:9.645e+13

Max. : 0.848706 Max. : 0.5202 Max. :4.420 Max. :1.230e+36

value_rnd value_2Bins some_date_Quarter_poly1

Min. : 8.0 Min. :1.0 Min. :-0.6325

1st Qu.:12.0 1st Qu.:1.0 1st Qu.:-0.3162

Median :24.0 Median :1.0 Median : 0.0000

Mean :31.8 Mean :1.4 Mean : 0.0000

3rd Qu.:32.0 3rd Qu.:2.0 3rd Qu.: 0.3162

Max. :83.0 Max. :2.0 Max. : 0.6325

some_date_Quarter_poly2 some_date_Quarter_log some_date_Quarter_exp

Min. :-0.5345 Min. :7.607 Min. :Inf

1st Qu.:-0.2673 1st Qu.:7.607 1st Qu.:Inf

Median :-0.2673 Median :7.608 Median :Inf

Mean : 0.0000 Mean :7.608 Mean :Inf

3rd Qu.: 0.5345 3rd Qu.:7.608 3rd Qu.:Inf

Max. : 0.5345 Max. :7.609 Max. :Inf

some_date_Quarter_rnd some_date_Quarter_2Bins

Min. :2012 Min. :1.0

1st Qu.:2013 1st Qu.:1.0

Median :2014 Median :1.0

Mean :2014 Mean :1.4

3rd Qu.:2015 3rd Qu.:2.0

Max. :2016 Max. :2.0

##

CORRELATION

```
print(cor(1:5,1:5))

## [1] 1

print(cor(1:5,seq(100,500,100)))

## [1] 1

print(cor(1:5,5:1))

## [1] -1

print(cor(1:5,c(1,2,3,4,4)))

## [1] 0.9701425


# install.packages('dplyr')
```

```
library(dplyr)

##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
## filter, lag
##
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union


# install.packages('reshape2')

library(reshape2)

data_set <- mtcars
d_cor <- as.matrix(cor(data_set))
```

d_cor

```
## mpg cyl disp hp drat wt
## mpg 1.0000000 -0.8521620 -0.8475514 -0.7761684 0.68117191 -0.8676594
## cyl -0.8521620 1.0000000 0.9020329 0.8324475 -0.69993811 0.7824958
## disp -0.8475514 0.9020329 1.0000000 0.7909486 -0.71021393 0.8879799
## hp -0.7761684 0.8324475 0.7909486 1.0000000 -0.44875912 0.6587479
## drat 0.6811719 -0.6999381 -0.7102139 -0.4487591 1.00000000 -0.7124406
## wt -0.8676594 0.7824958 0.8879799 0.6587479 -0.71244065 1.0000000
## qsec 0.4186840 -0.5912421 -0.4336979 -0.7082234 0.09120476 -0.1747159
## vs 0.6640389 -0.8108118 -0.7104159 -0.7230967 0.44027846 -0.5549157
## am 0.5998324 -0.5226070 -0.5912270 -0.2432043 0.71271113 -0.6924953
## gear 0.4802848 -0.4926866 -0.5555692 -0.1257043 0.69961013 -0.5832870
## carb -0.5509251 0.5269883 0.3949769 0.7498125 -0.09078980 0.4276059
## qsec vs am gear carb
## mpg 0.41868403 0.6640389 0.59983243 0.4802848 -0.55092507
## cyl -0.59124207 -0.8108118 -0.52260705 -0.4926866 0.52698829
## disp -0.43369788 -0.7104159 -0.59122704 -0.5555692 0.39497686
## hp -0.70822339 -0.7230967 -0.24320426 -0.1257043 0.74981247
## drat 0.09120476 0.4402785 0.71271113 0.6996101 -0.09078980
```

```
## wt -0.17471588 -0.5549157 -0.69249526 -0.5832870 0.42760594

## qsec 1.00000000 0.7445354 -0.22986086 -0.2126822 -0.65624923

## vs 0.74453544 1.0000000 0.16834512 0.2060233 -0.56960714

## am -0.22986086 0.1683451 1.00000000 0.7940588 0.05753435

## gear -0.21268223 0.2060233 0.79405876 1.0000000 0.27407284

## carb -0.65624923 -0.5696071 0.05753435 0.2740728 1.00000000



d_cor_melt <- arrange(melt(d_cor), -(value))


# clean up

pair_wise_correlation_matrix <- filter(d_cor_melt, Var1 != Var2)

pair_wise_correlation_matrix <- filter(pair_wise_correlation_matrix, is.na(value)==FALSE)


# remove pair dups


dim(pair_wise_correlation_matrix)


## [1] 110 3
```

```
pair_wise_correlation_matrix <- pair_wise_correlation_matrix[seq(1, nrow(pair_wise_correlation_matrix), by=2),]

dim(pair_wise_correlation_matrix)

## [1] 55 3

plot(pair_wise_correlation_matrix$value)



Get_Fast_Correlations <- function(data_set, features_to_ignore=c(), size_cap=5000) {
  require(dplyr)
  require(reshape2)
  data_set <- data_set[,setdiff(names(data_set), features_to_ignore)]
  if (size_cap > nrow(data_set)) {
    data_set = data_set[sample(nrow(data_set), size_cap),]
  } else {
    data_set = data_set[sample(nrow(data_set), nrow(data_set)),]
  }
  d_cor <- as.matrix(cor(data_set))
  d_cor_melt <- arrange(melt(d_cor), -(value))
```

```
 # clean up
 pair_wise_correlation_matrix <- filter(d_cor_melt, Var1 != Var2)
 pair_wise_correlation_matrix <- filter(pair_wise_correlation_matrix, is.na(value)==FALSE)
 # remove pair dups
 dim(pair_wise_correlation_matrix)
 pair_wise_correlation_matrix <- pair_wise_correlation_matrix[seq(1, nrow(pair_wise_correlation_matrix), by=2),
 ]
 dim(pair_wise_correlation_matrix)
 plot(pair_wise_correlation_matrix$value)
 return(pair_wise_correlation_matrix)
}
```

```
# install.packages('psych')
library(psych)
data_set <- mtcars
featurenames_copy <- names(data_set)
```

```
# strip var names to index for pair wise identification

names(data_set) <- seq(1:ncol(data_set))

cor_data_df <- corr.test(data_set)



# apply var names to correlation matrix over index

rownames(cor_data_df$r) <- featurenames_copy

colnames(cor_data_df$r) <- featurenames_copy


names(cor_data_df)


## [1] "r" "n" "t" "p" "se" "adjust" "sym" "ci"

## [9] "Call"



# matrix of correlations


cor_data_df$r
```

mpg cyl disp hp drat wt

mpg 1.0000000 -0.8521620 -0.8475514 -0.7761684 0.68117191 -0.8676594

cyl -0.8521620 1.0000000 0.9020329 0.8324475 -0.69993811 0.7824958

disp -0.8475514 0.9020329 1.0000000 0.7909486 -0.71021393 0.8879799

hp -0.7761684 0.8324475 0.7909486 1.0000000 -0.44875912 0.6587479

drat 0.6811719 -0.6999381 -0.7102139 -0.4487591 1.00000000 -0.7124406

wt -0.8676594 0.7824958 0.8879799 0.6587479 -0.71244065 1.0000000

qsec 0.4186840 -0.5912421 -0.4336979 -0.7082234 0.09120476 -0.1747159

vs 0.6640389 -0.8108118 -0.7104159 -0.7230967 0.44027846 -0.5549157

am 0.5998324 -0.5226070 -0.5912270 -0.2432043 0.71271113 -0.6924953

gear 0.4802848 -0.4926866 -0.5555692 -0.1257043 0.69961013 -0.5832870

carb -0.5509251 0.5269883 0.3949769 0.7498125 -0.09078980 0.4276059

qsec vs am gear carb

mpg 0.41868403 0.6640389 0.59983243 0.4802848 -0.55092507

cyl -0.59124207 -0.8108118 -0.52260705 -0.4926866 0.52698829

disp -0.43369788 -0.7104159 -0.59122704 -0.5555692 0.39497686

hp -0.70822339 -0.7230967 -0.24320426 -0.1257043 0.74981247

```
## drat 0.09120476 0.4402785 0.71271113 0.6996101 -0.09078980
## wt -0.17471588 -0.5549157 -0.69249526 -0.5832870 0.42760594
## qsec 1.00000000 0.7445354 -0.22986086 -0.2126822 -0.65624923
## vs 0.74453544 1.0000000 0.16834512 0.2060233 -0.56960714
## am -0.22986086 0.1683451 1.00000000 0.7940588 0.05753435
## gear -0.21268223 0.2060233 0.79405876 1.0000000 0.27407284
## carb -0.65624923 -0.5696071 0.05753435 0.2740728 1.00000000
```

```
cor.plot(cor_data_df$r)
```

```
#install.packages('corrplot')
library(corrplot)
corrplot.mixed(cor_data_df$r, lower="circle", upper="color", tl.pos="lt", diag="n", order="hclust", hclust.method="complete")
```

```
Get_Top_Relationships <- function(data_set, correlation_abs_threshold=0.8,pvalue_threshold=0.01) {
    require(psych)
```

```
    require(dplyr)
    feature_names <- names(data_set)
    # strip var names to index for pair-wise identification
    names(data_set) <- seq(1:ncol(data_set))
    # calculate correlation and significance numbers
    cor_data_df <- corr.test(data_set)
    # apply var names to correlation matrix over index
    rownames(cor_data_df$r) <- feature_names
    colnames(cor_data_df$r) <- feature_names
    # top cor and sig
    relationships_set <- cor_data_df$ci[,c('r','p')]
    # apply var names to data over index pairs
    relationships_set$feature_1 <- feature_names[as.numeric(sapply(strsplit(rownames(relationships_set), "-"), `[`, 1))]

    relationships_set$feature_2 <- feature_names[as.numeric(sapply(strsplit(rownames(relationships_set), "-"), `[`, 2))]
    relationships_set <- select(relationships_set, feature_1, feature_2, r, p) %>% rename(correlaton=r, pvalue=p)
    # return only the most insteresting relationships
    return(filter(relationships_set, abs(correlaton) > correlation_abs_threshold |pvalue < pvalue_threshold) %>% arrange(pvalue))
}
dim(Get_Top_Relationships(mtcars))
```

```
## [1] 39 4

head(Get_Top_Relationships(mtcars))

## feature_1 feature_2 correlaton pvalue
## 1 cyl disp 0.9020329 1.803002e-12
## 2 disp wt 0.8879799 1.222311e-11
## 3 mpg wt -0.8676594 1.293958e-10
## 4 mpg cyl -0.8521620 6.112688e-10
## 5 mpg disp -0.8475514 9.380328e-10
## 6 cyl hp 0.8324475 3.477861e-09

###########################################################################################################################
```

```
# CARET LIBRARY

library(caret)

data_set <- mtcars

d_cor <- cor(data_set)

class(d_cor)

top_correlations <- findCorrelation(x = d_cor, cutoff = 0.8, verbose = FALSE)



names(data_set)

names(data_set)[top_correlations]
```

```
top_correlations <- findCorrelation(x = d_cor, cutoff = 0.8, verbose = TRUE)
```

```
###############################################################################################################3
```

```
# OUTLIER DETECTION

wt_mean <- mean(mtcars$wt)
print(wt_mean)
```

```
## [1] 3.21725
```

```
wt_sd <- sd(mtcars$wt)
print(wt_sd)
```

```
## [1] 0.9784574

sum( (mtcars$wt > (wt_mean + (wt_sd))) | (mtcars$wt < (wt_mean - (wt_sd))))

## [1] 9

mtcars$wt[(mtcars$wt > (wt_mean + (wt_sd))) | (mtcars$wt < (wt_mean - (wt_sd)))]

## [1] 5.250 5.424 5.345 2.200 1.615 1.835 1.935 2.140 1.513

Identify_Outliers <- function(data_set, features_to_ignore=c(),
                              outlier_sd_threshold = 2,
                              remove_outlying_features = FALSE) {
  # get standard deviation for each feature
```

```
require(dplyr)
outliers <- c()
for (feature_name in setdiff(names(data_set),features_to_ignore)) {
  feature_mean <- mean(data_set[,feature_name], na.rm = TRUE)
  feature_sd <- sd(data_set[,feature_name], na.rm = TRUE)
  outlier_count <- sum(
    data_set[,feature_name] > (feature_mean + (feature_sd * outlier_sd_threshold))
     |
      data_set[,feature_name] < (feature_mean - (feature_sd * outlier_sd_threshold))
  )
  if (outlier_count > 0) {
    outliers <- rbind(outliers, c(feature_name, outlier_count))
    if (remove_outlying_features)
      data_set[, feature_name] <- NULL
  }
}
outliers <- data.frame(outliers) %>% rename(feature_name=X1, outlier_count=X2) %>%
  mutate(outlier_count=as.numeric(as.character(outlier_count))) %>% arrange(desc(outlier_count))
if (remove_outlying_features) {
  return(data_set)
```

```
  } else {
    return(outliers)
  }
}
```

```
head(Identify_Outliers(mtcars, remove_outlying_features=FALSE))
```

```
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
## filter, lag
##
## The following objects are masked from 'package:base':
##
```

intersect, setdiff, setequal, union

feature_name outlier_count

1 wt 3

2 mpg 2

3 hp 1

4 drat 1

5 qsec 1

6 carb 1

plot(sort(mtcars$wt))

##

```
# functions ----------------------------------------------------------------

Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE) {
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (feature_name in setdiff(text_features, features_to_ignore)) {
  feature_vector <- as.character(data_set[,feature_name])
  # check that data has more than one level
  if (length(unique(feature_vector)) == 1)
   next
  # We set any non-data to text
  feature_vector[is.na(feature_vector)] <- 'NA'
  feature_vector[is.infinite(feature_vector)] <- 'INF'
  feature_vector[is.nan(feature_vector)] <- 'NAN'
  # loop through each level of a feature and create a new column
  first_level=TRUE
  for (newcol in unique(feature_vector)) {
   if (first_level && leave_out_one_level) {
```

```
      # avoid dummy trap and skip first level
      first_level=FALSE
     } else {
      data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
     }
   }
   # remove original feature
   data_set <- data_set[,setdiff(names(data_set),feature_name)]
  }
  return (data_set)
}



Impute_Features <- function(data_set, features_to_ignore=c(),
                            use_mean_instead_of_0=TRUE,
                            mark_NAs=FALSE,
                            remove_zero_variance=FALSE) {
  for (feature_name in setdiff(names(data_set), features_to_ignore)) {
```

```
print(feature_name)
# remove any fields with zero variance
if (remove_zero_variance) {
  if (length(unique(data_set[, feature_name]))==1) {
    data_set[, feature_name] <- NULL
    next
  }
}
if (mark_NAs) {
  # note each field that contains missing or bad data
  if (any(is.na(data_set[,feature_name]))) {
    # create binary column before imputing
    newName <- paste0(feature_name, '_NA')
    data_set[,newName] <- as.integer(ifelse(is.na(data_set[,feature_name]),1,0)) }
  if (any(is.infinite(data_set[,feature_name]))) {
    newName <- paste0(feature_name, '_inf')
    data_set[,newName] <- as.integer(ifelse(is.infinite(data_set[,feature_name]),1,0)) }
}
if (use_mean_instead_of_0) {
  data_set[is.infinite(data_set[,feature_name]),feature_name] <- NA
```

```
    data_set[is.na(data_set[,feature_name]),feature_name] <- mean(dataset[,feature_name], na.rm=TRUE)
   } else {
    data_set[is.na(data_set[,feature_name]),feature_name] <- 0
    data_set[is.infinite(data_set[,feature_name]),feature_name] <- 0
   }
  }
  return(data_set)
}
```

```
Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, features_to_ignore=c()) {
  # look for text entries that are mostly unique
  text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
  for (f_name in setdiff(text_features, features_to_ignore)) {
   f_vector <- as.character(data_set[,f_name])
   # treat as raw text if data over minimum_precent_unique unique
   if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_unique_threshold)) {
    data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vector, " "), length)
    data_set[,paste0(f_name, '_character_count')] <- nchar(as.character(f_vector))
```

```
      data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`, 1)
      data_set[,paste0(f_name, '_second_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`, 2)
      # remove orginal field
      data_set[,f_name] <- NULL
    }
  }
  return(data_set)
}


#END FUNCTION ----------------------------------------------------------------



#Let's load the Titanic data set again. Take a quick peek at it before loading it in memory with readLines:
  # data ------------------------------------------------------------------
# using dataset from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/)

readLines('http://math.ucdenver.edu/RTutorial/titanic.txt', n=5)
## [1] "Name\tPClass\tAge\tSex\tSurvived"
## [2] "\"Allen, Miss Elisabeth Walton\"\t1st\t29\tfemale\t1"
```

```
## [3] "\"Allison, Miss Helen Loraine\"\t1st\t2\tfemale\t0"
## [4] "\"Allison, Mr Hudson Joshua Creighton\"\t1st\t30\tmale\t0"
## [5] "\"Allison, Mrs Hudson JC (Bessie Waldo Daniels)\"\t1st\t25\tfemale\t0"
# With readLines, we now know that the file has a header row and 5 columns separated by tabs.

titanicDF <- read.csv('http://math.ucdenver.edu/RTutorial/titanic.txt', sep='\t',header = TRUE)
head(titanicDF)
## Name PClass Age Sex
## 1 Allen, Miss Elisabeth Walton 1st 29.00 female
## 2 Allison, Miss Helen Loraine 1st 2.00 female
## 3 Allison, Mr Hudson Joshua Creighton 1st 30.00 male
## 4 Allison, Mrs Hudson JC (Bessie Waldo Daniels) 1st 25.00 female
## 5 Allison, Master Hudson Trevor 1st 0.92 male
## 6 Anderson, Mr Harry 1st 47.00 male
## Survived
## 1 1
## 2 0
## 3 0
## 4 0
## 5 1
```

```
## 6 1
```

```
titanicDF <- Get_Free_Text_Measures(titanicDF)
titanicDF$Name_first_word <- NULL
titanicDF <- Binarize_Features(titanicDF, leave_out_one_level = TRUE)
titanicDF <- Impute_Features(titanicDF, use_mean_instead_of_0 = FALSE)
```

```
## [1] "Age"
## [1] "Survived"
## [1] "Name_word_count"
## [1] "Name_character_count"
## [1] "Name_second_word_Mr"
## [1] "Name_second_word_Mrs"
## [1] "Name_second_word_Master"
## [1] "Name_second_word_Colonel"
```

```
## [1] "Name_second_word_Dr"
## [1] "Name_second_word_Major"
## [1] "Name_second_word_(Bowerman),"
## [1] "Name_second_word_Captain"
## [1] "Name_second_word_Villiers,"
## [1] "Name_second_word_Gordon,"
## [1] "Name_second_word_y"
## [1] "Name_second_word_Jonkheer"
## [1] "Name_second_word_(Russell),"
## [1] "Name_second_word_the"
## [1] "Name_second_word_Col"
## [1] "Name_second_word_Derhoef,"
## [1] "Name_second_word_Ms"
## [1] "Name_second_word_(Icabod),"
## [1] "Name_second_word_Mlle"
## [1] "Name_second_word_Rev"
## [1] "Name_second_word_Brito,"
## [1] "Name_second_word_Carlo,"
## [1] "Name_second_word_(?Douton),"
## [1] "Name_second_word_(Nasrallah),"
```

```
## [1] "Name_second_word_(Schmidt),"
## [1] "Name_second_word_(Kalil),"
## [1] "Name_second_word_Ernst"
## [1] "Name_second_word_(Kareem),"
## [1] "Name_second_word_Messemaeker,"
## [1] "Name_second_word_Mulder,"
## [1] "Name_second_word_Thomas"
## [1] "Name_second_word_Hilda"
## [1] "Name_second_word_Delia"
## [1] "Name_second_word_Jenny"
## [1] "Name_second_word_Oscar"
## [1] "Name_second_word_Nils"
## [1] "Name_second_word_Eino"
## [1] "Name_second_word_(Borak),"
## [1] "Name_second_word_Albert"
## [1] "Name_second_word_W"
## [1] "Name_second_word_Sander"
## [1] "Name_second_word_Richard"
## [1] "Name_second_word_Mansouer"
## [1] "Name_second_word_Nikolai"
```

```
## [1] "Name_second_word_(Joseph),"

## [1] "Name_second_word_(Trembisky),"

## [1] "Name_second_word_Khalil"

## [1] "Name_second_word_Simon"

## [1] "Name_second_word_William"

## [1] "Name_second_word_(Sitik),"

## [1] "Name_second_word_(Thomas),"

## [1] "Name_second_word_Billiard,"

## [1] "Name_second_word_der"

## [1] "Name_second_word_de"

## [1] "Name_second_word_Impe,"

## [1] "Name_second_word_Leo"

## [1] "PClass_2nd"

## [1] "PClass_3rd"

## [1] "Sex_male"
```

```
# split data set

set.seed(1234)
```

```
random_splits <- runif(nrow(titanicDF))

train_data <- titanicDF[random_splits < .5,]

tune_data <- titanicDF[random_splits >= .5 & random_splits < .8,]

test_data <- titanicDF[random_splits >= .8,]
```

```
# install.packages('randomForest')

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

set.seed(1234)

outcome_name <- 'Survived'

feature_names <- setdiff(names(train_data), outcome_name)
```

```
# print(setdiff(names(train_data), outcome_name))

tnRF <- tuneRF(x=tune_data[,feature_names],
        y = as.factor(tune_data[,outcome_name]),
        mtryStart = 3, stepFactor = 0.5)


## mtry = 3 OOB error = 26.25%
## Searching left ...
## mtry = 6 OOB error = 19.25%
## 0.2666667 0.05
## mtry = 12 OOB error = 19.25%
## 0 0.05
## Searching right ...
## mtry = 1 OOB error = 33.75%
## -0.7532468 0.05


best_mtry <- tnRF[tnRF[, 2] == min(tnRF[, 2]), 1][[1]]
print(best_mtry)
```

```
## [1] 6

rf_model <- randomForest(x=train_data[,feature_names],
                         y=as.factor(train_data[,outcome_name]),
                         importance=TRUE, ntree=100, mtry = best_mtry)

print(importance(rf_model, type=1)[importance(rf_model, type=1)!=0,])

## Age Name_word_count Name_character_count
## 4.472367 4.586581 3.786784
## Name_second_word_Mr Name_second_word_Mrs Name_second_word_Master
## 6.880982 6.093252 1.222808
## Name_second_word_Dr Name_second_word_y Name_second_word_Ms
## 2.447301 2.817508 1.202244
## Name_second_word_Mlle Name_second_word_Rev PClass_2nd
## -1.145309 4.133762 3.677996
## PClass_3rd Sex_male
## 8.137837 7.619451
```

#Let’s test the model on our test_data and use the pROC library to get an AUC score:

```
 predictions <- predict(rf_model, newdata=test_data[,feature_names], type="prob")


# install.packages('pROC')
library(pROC)
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
## cov, smooth, var

print(roc(response = test_data[,outcome_name], predictor = predictions[,2]))

##
## Call:
## roc.default(response = test_data[, outcome_name], predictor = predictions[,2])
```

```
##
## Data: predictions[, 2] in 174 controls (test_data[, outcome_name] 0) < 92 cases (test_data[, outcome_name] 1)
## Area under the curve: 0.8815

################################################################################################################

#install.packages('dplyr')
library(dplyr)
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#install.packages('caret')

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

names(getModelInfo())

##   [1] "ada"             "AdaBag"          "AdaBoost.M1"
##   [4] "adaboost"        "amdai"           "ANFIS"
##   [7] "avNNet"          "awnb"            "awtan"
##  [10] "bag"             "bagEarth"        "bagEarthGCV"
##  [13] "bagFDA"          "bagFDAGCV"       "bam"
##  [16] "bartMachine"     "bayesglm"        "bdk"
##  [19] "binda"           "blackboost"      "blasso"
##  [22] "blassoAveraged"  "Boruta"          "bridge"
##  [25] "brnn"            "BstLm"           "bstSm"
##  [28] "bstTree"         "C5.0"            "C5.0Cost"
##  [31] "C5.0Rules"       "C5.0Tree"        "cforest"
##  [34] "chaid"           "CSimca"          "ctree"
##  [37] "ctree2"          "cubist"          "dda"
##  [40] "deepboost"       "DENFIS"          "dnn"
##  [43] "dwdLinear"       "dwdPoly"         "dwdRadial"
```

```
## [46] "earth"          "elm"            "enet"
## [49] "enpls.fs"       "enpls"          "evtree"
## [52] "extraTrees"     "fda"            "FH.GBML"
## [55] "FIR.DM"         "foba"           "FRBCS.CHI"
## [58] "FRBCS.W"        "FS.HGD"         "gam"
## [61] "gamboost"       "gamLoess"       "gamSpline"
## [64] "gaussprLinear"  "gaussprPoly"    "gaussprRadial"
## [67] "gbm_h2o"        "gbm"            "gcvEarth"
## [70] "GFS.FR.MOGUL"   "GFS.GCCL"       "GFS.LT.RS"
## [73] "GFS.THRIFT"     "glm.nb"         "glm"
## [76] "glmboost"       "glmnet_h2o"     "glmnet"
## [79] "glmStepAIC"     "gpls"           "hda"
## [82] "hdda"           "hdrda"          "HYFIS"
## [85] "icr"            "J48"            "JRip"
## [88] "kernelpls"      "kknn"           "knn"
## [91] "krlsPoly"       "krlsRadial"     "lars"
## [94] "lars2"          "lasso"          "lda"
## [97] "lda2"           "leapBackward"   "leapForward"
## [100] "leapSeq"       "Linda"          "lm"
## [103] "lmStepAIC"     "LMT"            "loclda"
```

```
## [106] "logicBag"          "LogitBoost"        "logreg"
## [109] "lssvmLinear"       "lssvmPoly"         "lssvmRadial"
## [112] "lvq"               "M5"                "M5Rules"
## [115] "manb"              "mda"               "Mlda"
## [118] "mlp"               "mlpML"             "mlpSGD"
## [121] "mlpWeightDecay"    "mlpWeightDecayML"  "multinom"
## [124] "nb"                "nbDiscrete"        "nbSearch"
## [127] "neuralnet"         "nnet"              "nnls"
## [130] "nodeHarvest"       "oblique.tree"      "OneR"
## [133] "ordinalNet"        "ORFlog"            "ORFpls"
## [136] "ORFridge"          "ORFsvm"            "ownn"
## [139] "pam"               "parRF"             "PART"
## [142] "partDSA"           "pcaNNet"           "pcr"
## [145] "pda"               "pda2"              "penalized"
## [148] "PenalizedLDA"      "plr"               "pls"
## [151] "plsRglm"           "polr"              "ppr"
## [154] "protoclass"        "pythonKnnReg"      "qda"
## [157] "QdaCov"            "qrf"               "qrnn"
## [160] "randomGLM"         "ranger"            "rbf"
## [163] "rbfDDA"            "Rborist"           "rda"
```

```
## [166] "relaxo"            "rf"               "rFerns"
## [169] "RFlda"             "rfRules"          "ridge"
## [172] "rlda"              "rlm"              "rmda"
## [175] "rocc"              "rotationForest"   "rotationForestCp"
## [178] "rpart"             "rpart1SE"         "rpart2"
## [181] "rpartCost"         "rpartScore"       "rqlasso"
## [184] "rqnc"              "RRF"              "RRFglobal"
## [187] "rrlda"             "RSimca"           "rvmLinear"
## [190] "rvmPoly"           "rvmRadial"        "SBC"
## [193] "sda"               "sddaLDA"          "sddaQDA"
## [196] "sdwd"              "simpls"           "SLAVE"
## [199] "slda"              "smda"             "snn"
## [202] "sparseLDA"         "spikeslab"        "spls"
## [205] "stepLDA"           "stepQDA"          "superpc"
## [208] "svmBoundrangeString" "svmExpoString"   "svmLinear"
## [211] "svmLinear2"        "svmLinear3"       "svmLinearWeights"
## [214] "svmLinearWeights2" "svmPoly"          "svmRadial"
## [217] "svmRadialCost"     "svmRadialSigma"   "svmRadialWeights"
## [220] "svmSpectrumString" "tan"              "tanSearch"
## [223] "treebag"           "vbmpRadial"       "vglmAdjCat"
```

```
## [226] "vglmContRatio"    "vglmCumulative"    "widekernelpls"
## [229] "WM"               "wsrf"              "xgbLinear"
## [232] "xgbTree"          "xyf"
```

```
 require(RCurl)
## Loading required package: RCurl
## Loading required package: bitops
binData <- getBinaryURL("https://archive.ics.uci.edu/ml/machine-learning-databases/00296/dataset_diabetes.zip",
                        ssl.verifypeer=FALSE)

conObj <- file("dataset_diabetes.zip", open = "wb")
writeBin(binData, conObj)
# don't forget to close it
close(conObj)

# open diabetes file
files <- unzip("dataset_diabetes.zip")
readLines(files[1], n=5)
```

```
## [1]
"encounter_id,patient_nbr,race,gender,age,weight,admission_type_id,discharge_disposition_id,admission_source_id,time_in_hospital,payer_code,medical
_specialty,num_lab_procedures,num_procedures,num_medications,number_outpatient,number_emergency,number_inpatient,diag_1,diag_2,diag_3,num
ber_diagnoses,max_glu_serum,A1Cresult,metformin,repaglinide,nateglinide,chlorpropamide,glimepiride,acetohexamide,glipizide,glyburide,tolbutamide,pi
oglitazone,rosiglitazone,acarbose,miglitol,troglitazone,tolazamide,examide,citoglipton,insulin,glyburide-metformin,glipizide-metformin,glimepiride-
pioglitazone,metformin-rosiglitazone,metformin-pioglitazone,change,diabetesMed,readmitted"

## [2] "2278392,8222157,Caucasian,Female,[0-10),?,6,25,1,1,?,Pediatrics-
Endocrinology,41,0,1,0,0,0,250.83,?,?,1,None,None,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,NO"

## [3] "149190,55629189,Caucasian,Female,[10-
20),?,1,1,7,3,?,?,59,0,18,0,0,0,276,250.01,255,9,None,None,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,Up,No,No,No,No,No,Ch,Yes,>30"

## [4] "64410,86047875,AfricanAmerican,Female,[20-
30),?,1,1,7,2,?,?,11,5,13,2,0,1,648,250,V27,6,None,None,No,No,No,No,No,No,Steady,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,Yes,NO"

## [5] "500364,82442376,Caucasian,Male,[30-
40),?,1,1,7,2,?,?,44,1,16,0,0,0,8,250.43,403,7,None,None,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,Up,No,No,No,No,No,Ch,Yes,NO"
```

```
#install.packages('readr')

library(readr)

diabetes <- data.frame(read_csv(files[1], na = '?'))

## Parsed with column specification:

## cols(
```

```
##   .default = col_character(),
##   encounter_id = col_integer(),
##   patient_nbr = col_integer(),
##   admission_type_id = col_integer(),
##   discharge_disposition_id = col_integer(),
##   admission_source_id = col_integer(),
##   time_in_hospital = col_integer(),
##   num_lab_procedures = col_integer(),
##   num_procedures = col_integer(),
##   num_medications = col_integer(),
##   number_outpatient = col_integer(),
##   number_emergency = col_integer(),
##   number_inpatient = col_integer(),
##   number_diagnoses = col_integer()
## )
## See spec(...) for full column specifications.
dim(diabetes)
## [1] 101766     50
head(diabetes)
##   encounter_id patient_nbr           race gender     age weight
```

```
## 1     2278392     8222157      Caucasian Female  [0-10)   <NA>
## 2      149190    55629189      Caucasian Female [10-20)   <NA>
## 3       64410    86047875 AfricanAmerican Female [20-30)   <NA>
## 4      500364    82442376      Caucasian   Male [30-40)   <NA>
## 5       16680    42519267      Caucasian   Male [40-50)   <NA>
## 6       35754    82637451      Caucasian   Male [50-60)   <NA>
##   admission_type_id discharge_disposition_id admission_source_id
## 1                 6                       25                   1
## 2                 1                        1                   7
## 3                 1                        1                   7
## 4                 1                        1                   7
## 5                 1                        1                   7
## 6                 2                        1                   2
##   time_in_hospital payer_code        medical_specialty num_lab_procedures
## 1                1       <NA> Pediatrics-Endocrinology                 41
## 2                3       <NA>                     <NA>                 59
## 3                2       <NA>                     <NA>                 11
## 4                2       <NA>                     <NA>                 44
## 5                1       <NA>                     <NA>                 51
## 6                3       <NA>                     <NA>                 31
```

```
##   num_procedures num_medications number_outpatient number_emergency
## 1           0            1              0             0
## 2           0           18              0             0
## 3           5           13              2             0
## 4           1           16              0             0
## 5           0            8              0             0
## 6           6           16              0             0
##   number_inpatient diag_1 diag_2 diag_3 number_diagnoses max_glu_serum
## 1                0 250.83   <NA>   <NA>                1          None
## 2                0    276 250.01    255                9          None
## 3                1    648    250    V27                6          None
## 4                0      8 250.43    403                7          None
## 5                0    197    157    250                5          None
## 6                0    414    411    250                9          None
##   A1Cresult metformin repaglinide nateglinide chlorpropamide glimepiride
## 1      None        No          No          No             No          No
## 2      None        No          No          No             No          No
## 3      None        No          No          No             No          No
## 4      None        No          No          No             No          No
## 5      None        No          No          No             No          No
```

```
## 6    None      No       No       No         No        No
##   acetohexamide glipizide glyburide tolbutamide pioglitazone rosiglitazone
## 1          No       No       No        No         No          No
## 2          No       No       No        No         No          No
## 3          No   Steady       No        No         No          No
## 4          No       No       No        No         No          No
## 5          No   Steady       No        No         No          No
## 6          No       No       No        No         No          No
##   acarbose miglitol troglitazone tolazamide examide citoglipton insulin
## 1       No       No           No         No      No          No      No
## 2       No       No           No         No      No          No      Up
## 3       No       No           No         No      No          No      No
## 4       No       No           No         No      No          No      Up
## 5       No       No           No         No      No          No  Steady
## 6       No       No           No         No      No          No  Steady
##   glyburide.metformin glipizide.metformin glimepiride.pioglitazone
## 1                  No                  No                       No
## 2                  No                  No                       No
## 3                  No                  No                       No
## 4                  No                  No                       No
```

```
## 5            No            No                No
## 6            No            No                No
##   metformin.rosiglitazone metformin.pioglitazone change diabetesMed
## 1                  No                  No     No          No
## 2                  No                  No     Ch         Yes
## 3                  No                  No     No         Yes
## 4                  No                  No     Ch         Yes
## 5                  No                  No     Ch         Yes
## 6                  No                  No     No         Yes
##   readmitted
## 1         NO
## 2        >30
## 3         NO
## 4         NO
## 5         NO
## 6        >30
```

```
 # drop useless variables
diabetes <- subset(diabetes,select=-c(encounter_id, patient_nbr, examide, citoglipton))

# fix our outcome variable to those readmitted under 30 days
diabetes$readmitted <- ifelse(diabetes$readmitted == "<30",'yes','no')



 # see what type of classes we have
charcolumns <- names(diabetes[sapply(diabetes, is.character)])
non_numeric_data_dim <- c()
for (colname in charcolumns)
  non_numeric_data_dim <- rbind(non_numeric_data_dim, c(colname, length(unique(diabetes[,colname]))))

non_numeric_data_dim <- data.frame(non_numeric_data_dim) %>%
  mutate(feature_name=as.character(X1), unique_counts=as.numeric(as.character(X2))) %>%
  select(feature_name, unique_counts) %>%
  arrange(desc(unique_counts))

head(non_numeric_data_dim, 10)
```

```
##        feature_name unique_counts
## 1           diag_3         790
## 2           diag_2         749
## 3           diag_1         717
## 4  medical_specialty          73
## 5       payer_code          18
## 6              age          10
## 7           weight          10
## 8             race           6
## 9     max_glu_serum           4
## 10        A1Cresult           4
```

```
Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_level=FALSE, max_level_count=20) {
  require(dplyr)
  text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
  for (feature_name in setdiff(text_features, features_to_ignore)) {
    feature_vector <- as.character(data_set[,feature_name])
```

```
# check that data has more than one level
if (length(unique(feature_vector)) == 1)
  next

# We set any non-data to text
feature_vector[is.na(feature_vector)] <- 'NA'
feature_vector[is.infinite(feature_vector)] <- 'INF'
feature_vector[is.nan(feature_vector)] <- 'NAN'

# only give us the top x most popular categories
temp_vect <- data.frame(table(feature_vector)) %>% arrange(desc(Freq)) %>% head(max_level_count)
feature_vector <- ifelse(feature_vector %in% temp_vect$feature_vector, feature_vector, 'Other')

# loop through each level of a feature and create a new column
first_level=TRUE
for (newcol in unique(feature_vector)) {
  if (leave_out_one_level & first_level) {
    # avoid dummy trap and skip first level
    first_level=FALSE
    next
```

```
    }

    data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector==newcol,1,0)
   }
   # remove original feature
   data_set <- data_set[,setdiff(names(data_set),feature_name)]
  }
  return (data_set)
}


diabetes <- Binarize_Features(data_set = diabetes, leave_out_one_level = TRUE,
                              max_level_count = 20, features_to_ignore = 'readmitted')
summary(diabetes)
##  admission_type_id discharge_disposition_id admission_source_id
##  Min.   :1.000     Min.   : 1.000           Min.   : 1.000
##  1st Qu.:1.000     1st Qu.: 1.000           1st Qu.: 1.000
##  Median :1.000     Median : 1.000           Median : 7.000
##  Mean   :2.024     Mean   : 3.716           Mean   : 5.754
##  3rd Qu.:3.000     3rd Qu.: 4.000           3rd Qu.: 7.000
```

```
## Max. :8.000 Max. :28.000 Max. :25.000
## time_in_hospital num_lab_procedures num_procedures num_medications
## Min. : 1.000 Min. : 1.0 Min. :0.00 Min. : 1.00
## 1st Qu.: 2.000 1st Qu.: 31.0 1st Qu.:0.00 1st Qu.:10.00
## Median : 4.000 Median : 44.0 Median :1.00 Median :15.00
## Mean : 4.396 Mean : 43.1 Mean :1.34 Mean :16.02
## 3rd Qu.: 6.000 3rd Qu.: 57.0 3rd Qu.:2.00 3rd Qu.:20.00
## Max. :14.000 Max. :132.0 Max. :6.00 Max. :81.00
## number_outpatient number_emergency number_inpatient number_diagnoses
## Min. : 0.0000 Min. : 0.0000 Min. : 0.0000 Min. : 1.000
## 1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 6.000
## Median : 0.0000 Median : 0.0000 Median : 0.0000 Median : 8.000
## Mean : 0.3694 Mean : 0.1978 Mean : 0.6356 Mean : 7.423
## 3rd Qu.: 0.0000 3rd Qu.: 0.0000 3rd Qu.: 1.0000 3rd Qu.: 9.000
## Max. :42.0000 Max. :76.0000 Max. :21.0000 Max. :16.000
## readmitted race_AfricanAmerican race_NA
## Length:101766 Min. :0.0000 Min. :0.00000
## Class :character 1st Qu.:0.0000 1st Qu.:0.00000
## Mode :character Median :0.0000 Median :0.00000
## Mean :0.1888 Mean :0.02234
```

```
##                 3rd Qu.:0.0000      3rd Qu.:0.00000
##                 Max.   :1.0000      Max.   :1.00000
##   race_Other      race_Asian      race_Hispanic     gender_Male
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.000000   Median :0.00000   Median :0.0000
##  Mean   :0.0148   Mean   :0.006299   Mean   :0.02002   Mean   :0.4624
##  3rd Qu.:0.0000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.000000   Max.   :1.00000   Max.   :1.0000
##  gender_Unknown/Invalid  age_[10-20)      age_[20-30)
##  Min.   :0.00e+00      Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00e+00      1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00e+00      Median :0.00000   Median :0.00000
##  Mean   :2.95e-05      Mean   :0.00679   Mean   :0.01628
##  3rd Qu.:0.00e+00      3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00e+00      Max.   :1.00000   Max.   :1.00000
##   age_[30-40)      age_[40-50)      age_[50-60)      age_[60-70)
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.0000
```

```
## Mean   :0.03709   Mean   :0.09517   Mean   :0.1696   Mean   :0.2209
## 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.0000
## Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##  age_[70-80)      age_[80-90)     age_[90-100)     weight_[75-100)
## Min.   :0.0000   Min.   :0.000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:0.00000
## Median :0.0000   Median :0.000   Median :0.00000   Median :0.00000
## Mean   :0.2562   Mean   :0.169   Mean   :0.02745   Mean   :0.01313
## 3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.:0.00000   3rd Qu.:0.00000
## Max.   :1.0000   Max.   :1.000   Max.   :1.00000   Max.   :1.00000
## weight_[50-75)     weight_[0-25)      weight_[100-125)
## Min.   :0.000000   Min.   :0.0000000   Min.   :0.000000
## 1st Qu.:0.000000   1st Qu.:0.0000000   1st Qu.:0.000000
## Median :0.000000   Median :0.0000000   Median :0.000000
## Mean   :0.008814   Mean   :0.0004717   Mean   :0.006142
## 3rd Qu.:0.000000   3rd Qu.:0.0000000   3rd Qu.:0.000000
## Max.   :1.000000   Max.   :1.0000000   Max.   :1.000000
## weight_[25-50)      weight_[125-150)   weight_[175-200)
## Min.   :0.0000000   Min.   :0.000000   Min.   :0.0000000
## 1st Qu.:0.0000000   1st Qu.:0.000000   1st Qu.:0.0000000
```

```
## Median :0.0000000 Median :0.000000 Median :0.0000000
## Mean :0.0009532 Mean :0.001425 Mean :0.0001081
## 3rd Qu.:0.0000000 3rd Qu.:0.000000 3rd Qu.:0.0000000
## Max. :1.0000000 Max. :1.000000 Max. :1.0000000
## weight_[150-175) weight_>200 payer_code_MC payer_code_MD
## Min. :0.0000000 Min. :0.00e+00 Min. :0.0000 Min. :0.00000
## 1st Qu.:0.0000000 1st Qu.:0.00e+00 1st Qu.:0.0000 1st Qu.:0.00000
## Median :0.0000000 Median :0.00e+00 Median :0.0000 Median :0.00000
## Mean :0.0003439 Mean :2.95e-05 Mean :0.3188 Mean :0.03471
## 3rd Qu.:0.0000000 3rd Qu.:0.00e+00 3rd Qu.:1.0000 3rd Qu.:0.00000
## Max. :1.0000000 Max. :1.00e+00 Max. :1.0000 Max. :1.00000
## payer_code_HM payer_code_UN payer_code_BC payer_code_SP
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :0.0000
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.0000
## Median :0.00000 Median :0.00000 Median :0.00000 Median :0.0000
## Mean :0.06165 Mean :0.02406 Mean :0.04574 Mean :0.0492
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.0000
## Max. :1.00000 Max. :1.00000 Max. :1.00000 Max. :1.0000
## payer_code_CP payer_code_SI payer_code_DM
## Min. :0.00000 Min. :0.0000000 Min. :0.000000
```

```
##  1st Qu.:0.00000   1st Qu.:0.0000000   1st Qu.:0.000000
##  Median :0.00000   Median :0.0000000   Median :0.000000
##  Mean   :0.02489   Mean   :0.0005405   Mean   :0.005395
##  3rd Qu.:0.00000   3rd Qu.:0.0000000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.0000000   Max.   :1.000000
##  payer_code_CM     payer_code_CH      payer_code_PO
##  Min.   :0.00000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.00000   Median :0.000000   Median :0.000000
##  Mean   :0.01903   Mean   :0.001435   Mean   :0.005817
##  3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.000000   Max.   :1.000000
##  payer_code_WC      payer_code_OT       payer_code_OG
##  Min.   :0.000000   Min.   :0.0000000   Min.   :0.00000
##  1st Qu.:0.000000   1st Qu.:0.0000000   1st Qu.:0.00000
##  Median :0.000000   Median :0.0000000   Median :0.00000
##  Mean   :0.001327   Mean   :0.0009335   Mean   :0.01015
##  3rd Qu.:0.000000   3rd Qu.:0.0000000   3rd Qu.:0.00000
##  Max.   :1.000000   Max.   :1.0000000   Max.   :1.00000
##  payer_code_MP      payer_code_FR     medical_specialty_NA
```

```
## Min.   :0.0000000   Min.   :0.0e+00   Min.   :0.0000
## 1st Qu.:0.0000000   1st Qu.:0.0e+00   1st Qu.:0.0000
## Median :0.0000000   Median :0.0e+00   Median :0.0000
## Mean   :0.0007763   Mean   :9.8e-06   Mean   :0.4908
## 3rd Qu.:0.0000000   3rd Qu.:0.0e+00   3rd Qu.:1.0000
## Max.   :1.0000000   Max.   :1.0e+00   Max.   :1.0000
## medical_specialty_InternalMedicine
## Min.   :0.0000
## 1st Qu.:0.0000
## Median :0.0000
## Mean   :0.1438
## 3rd Qu.:0.0000
## Max.   :1.0000
## medical_specialty_Family/GeneralPractice medical_specialty_Cardiology
## Min.   :0.00000                          Min.   :0.00000
## 1st Qu.:0.00000                          1st Qu.:0.00000
## Median :0.00000                          Median :0.00000
## Mean   :0.07311                          Mean   :0.05259
## 3rd Qu.:0.00000                          3rd Qu.:0.00000
## Max.   :1.00000                          Max.   :1.00000
```

```
## medical_specialty_Surgery-General medical_specialty_Orthopedics
## Min.   :0.00000                 Min.   :0.00000
## 1st Qu.:0.00000                 1st Qu.:0.00000
## Median :0.00000                 Median :0.00000
## Mean   :0.03045                 Mean   :0.01376
## 3rd Qu.:0.00000                 3rd Qu.:0.00000
## Max.   :1.00000                 Max.   :1.00000
## medical_specialty_Gastroenterology
## Min.   :0.000000
## 1st Qu.:0.000000
## Median :0.000000
## Mean   :0.005542
## 3rd Qu.:0.000000
## Max.   :1.000000
## medical_specialty_Surgery-Cardiovascular/Thoracic
## Min.   :0.000000
## 1st Qu.:0.000000
## Median :0.000000
## Mean   :0.006407
## 3rd Qu.:0.000000
```

```
## Max.   :1.000000
## medical_specialty_Nephrology medical_specialty_Orthopedics-Reconstructive
## Min.   :0.00000             Min.   :0.00000
## 1st Qu.:0.00000             1st Qu.:0.00000
## Median :0.00000             Median :0.00000
## Mean   :0.01585             Mean   :0.01212
## 3rd Qu.:0.00000             3rd Qu.:0.00000
## Max.   :1.00000             Max.   :1.00000
## medical_specialty_Psychiatry medical_specialty_Emergency/Trauma
## Min.   :0.000000            Min.   :0.00000
## 1st Qu.:0.000000            1st Qu.:0.00000
## Median :0.000000            Median :0.00000
## Mean   :0.008392            Mean   :0.07434
## 3rd Qu.:0.000000            3rd Qu.:0.00000
## Max.   :1.000000            Max.   :1.00000
## medical_specialty_Pulmonology medical_specialty_Surgery-Neuro
## Min.   :0.000000             Min.   :0.000000
## 1st Qu.:0.000000             1st Qu.:0.000000
## Median :0.000000             Median :0.000000
## Mean   :0.008559             Mean   :0.004599
```

```
## 3rd Qu.:0.000000            3rd Qu.:0.000000
## Max.   :1.000000            Max.   :1.000000
## medical_specialty_ObstetricsandGynecology medical_specialty_Urology
## Min.   :0.000000                         Min.   :0.000000
## 1st Qu.:0.000000                         1st Qu.:0.000000
## Median :0.000000                         Median :0.000000
## Mean   :0.006594                         Mean   :0.006731
## 3rd Qu.:0.000000                         3rd Qu.:0.000000
## Max.   :1.000000                         Max.   :1.000000
## medical_specialty_Oncology
## Min.   :0.00000
## 1st Qu.:0.00000
## Median :0.00000
## Mean   :0.00342
## 3rd Qu.:0.00000
## Max.   :1.00000
## medical_specialty_PhysicalMedicineandRehabilitation
## Min.   :0.000000
## 1st Qu.:0.000000
## Median :0.000000
```

```
## Mean   :0.003842
## 3rd Qu.:0.000000
## Max.   :1.000000
## medical_specialty_Surgery-Vascular medical_specialty_Radiologist
## Min.   :0.000000                   Min.   :0.0000
## 1st Qu.:0.000000                   1st Qu.:0.0000
## Median :0.000000                   Median :0.0000
## Mean   :0.005237                   Mean   :0.0112
## 3rd Qu.:0.000000                   3rd Qu.:0.0000
## Max.   :1.000000                   Max.   :1.0000
##   diag_1_276       diag_1_414       diag_1_428       diag_1_434
## Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
## Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
## Mean   :0.01856   Mean   :0.06467   Mean   :0.06743   Mean   :0.01993
## 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
## Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   diag_1_518       diag_1_410       diag_1_682       diag_1_V57
## Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
```

```
## Median :0.00000 Median :0.00000 Median :0.00000 Median :0.00000
## Mean :0.01096 Mean :0.03551 Mean :0.02007 Mean :0.01186
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000
## Max. :1.00000 Max. :1.00000 Max. :1.00000 Max. :1.00000
## diag_1_786 diag_1_427 diag_1_996 diag_1_584
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :0.00000
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000
## Median :0.00000 Median :0.00000 Median :0.00000 Median :0.00000
## Mean :0.03946 Mean :0.02718 Mean :0.01933 Mean :0.01494
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000
## Max. :1.00000 Max. :1.00000 Max. :1.00000 Max. :1.00000
## diag_1_486 diag_1_250.6 diag_1_715 diag_1_38
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :0.00000
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000
## Median :0.00000 Median :0.00000 Median :0.00000 Median :0.00000
## Mean :0.03447 Mean :0.01162 Mean :0.02114 Mean :0.01659
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000
## Max. :1.00000 Max. :1.00000 Max. :1.00000 Max. :1.00000
## diag_1_599 diag_1_491 diag_1_250.8 diag_1_780
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :0.00000
```

```
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.01567   Mean   :0.02236   Mean   :0.01651   Mean   :0.01984
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##  diag_2_250.01      diag_2_250        diag_2_411        diag_2_427
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.01497   Mean   :0.05966   Mean   :0.02521   Mean   :0.04949
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##    diag_2_403        diag_2_425        diag_2_401        diag_2_496
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.02774   Mean   :0.01409   Mean   :0.03671   Mean   :0.03248
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##    diag_2_428        diag_2_585       diag_2_250.02      diag_2_276
```

```
## Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
## Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
## Mean   :0.06546   Mean   :0.01839   Mean   :0.02038   Mean   :0.06635
## 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
## Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   diag_2_599        diag_2_491        diag_2_707        diag_2_414
## Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
## Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
## Mean   :0.03231   Mean   :0.01518   Mean   :0.01964   Mean   :0.02604
## 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
## Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   diag_2_285        diag_2_780        diag_2_584       diag_2_682
## Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
## Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
## Mean   :0.01494   Mean   :0.01465   Mean   :0.0162   Mean   :0.01408
## 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
## Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
```

```
## diag_3_Other diag_3_403 diag_3_250 diag_3_V45
## Min. :0.000 Min. :0.00000 Min. :0.0000 Min. :0.00000
## 1st Qu.:0.000 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.00000
## Median :0.000 Median :0.00000 Median :0.0000 Median :0.00000
## Mean :0.416 Mean :0.02316 Mean :0.1135 Mean :0.01365
## 3rd Qu.:1.000 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:0.00000
## Max. :1.000 Max. :1.00000 Max. :1.0000 Max. :1.00000
## diag_3_250.6 diag_3_427 diag_3_414 diag_3_428
## Min. :0.00000 Min. :0.00000 Min. :0.000 Min. :0.00000
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.000 1st Qu.:0.00000
## Median :0.00000 Median :0.00000 Median :0.000 Median :0.00000
## Mean :0.01061 Mean :0.03886 Mean :0.036 Mean :0.04498
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.000 3rd Qu.:0.00000
## Max. :1.00000 Max. :1.00000 Max. :1.000 Max. :1.00000
## diag_3_276 diag_3_401 diag_3_585 diag_3_250.02
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :0.00000
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000
## Median :0.00000 Median :0.00000 Median :0.00000 Median :0.00000
## Mean :0.05085 Mean :0.08145 Mean :0.01957 Mean :0.01345
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000
```

```
## Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   diag_3_707        diag_3_496       diag_3_599        diag_3_424
## Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
## Median :0.00000   Median :0.0000   Median :0.00000   Median :0.00000
## Mean   :0.01336   Mean   :0.0256   Mean   :0.01907   Mean   :0.01045
## 3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
## Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##   diag_3_425        diag_3_272        diag_3_780        diag_3_285
## Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
## Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
## Mean   :0.01116   Mean   :0.01935   Mean   :0.01311   Mean   :0.01179
## 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
## Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
## max_glu_serum_>300 max_glu_serum_Norm max_glu_serum_>200
## Min.   :0.00000    Min.   :0.00000    Min.   :0.00000
## 1st Qu.:0.00000    1st Qu.:0.00000    1st Qu.:0.00000
## Median :0.00000    Median :0.00000    Median :0.00000
## Mean   :0.01242    Mean   :0.02552    Mean   :0.01459
```

```
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   A1Cresult_>7      A1Cresult_>8     A1Cresult_Norm    metformin_Steady
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.03746   Mean   :0.08073   Mean   :0.04903   Mean   :0.1803
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##   metformin_Up     metformin_Down    repaglinide_Up     repaglinide_Steady
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.000000   Median :0.0000
##  Mean   :0.01048   Mean   :0.00565   Mean   :0.001081   Mean   :0.0136
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.000000   Max.   :1.0000
##  repaglinide_Down    nateglinide_Steady nateglinide_Down
##  Min.   :0.0000000   Min.   :0.000000   Min.   :0.0000000
##  1st Qu.:0.0000000   1st Qu.:0.000000   1st Qu.:0.0000000
##  Median :0.0000000   Median :0.000000   Median :0.0000000
```

```
## Mean   :0.0004422   Mean   :0.006564   Mean   :0.0001081
## 3rd Qu.:0.0000000   3rd Qu.:0.000000   3rd Qu.:0.0000000
## Max.   :1.0000000   Max.   :1.000000   Max.   :1.0000000
## nateglinide_Up      chlorpropamide_Steady chlorpropamide_Down
## Min.   :0.0000000   Min.   :0.0000000     Min.   :0.0e+00
## 1st Qu.:0.0000000   1st Qu.:0.0000000     1st Qu.:0.0e+00
## Median :0.0000000   Median :0.0000000     Median :0.0e+00
## Mean   :0.0002358   Mean   :0.0007763     Mean   :9.8e-06
## 3rd Qu.:0.0000000   3rd Qu.:0.0000000     3rd Qu.:0.0e+00
## Max.   :1.0000000   Max.   :1.0000000     Max.   :1.0e+00
## chlorpropamide_Up glimepiride_Steady glimepiride_Down
## Min.   :0.0e+00   Min.   :0.00000    Min.   :0.000000
## 1st Qu.:0.0e+00   1st Qu.:0.00000    1st Qu.:0.000000
## Median :0.0e+00   Median :0.00000    Median :0.000000
## Mean   :5.9e-05   Mean   :0.04589    Mean   :0.001906
## 3rd Qu.:0.0e+00   3rd Qu.:0.00000    3rd Qu.:0.000000
## Max.   :1.0e+00   Max.   :1.00000    Max.   :1.000000
## glimepiride_Up     acetohexamide_Steady glipizide_Steady
## Min.   :0.000000   Min.   :0.0e+00      Min.   :0.0000
## 1st Qu.:0.000000   1st Qu.:0.0e+00      1st Qu.:0.0000
```

```
## Median :0.000000 Median :0.0e+00 Median :0.0000
## Mean :0.003213 Mean :9.8e-06 Mean :0.1116
## 3rd Qu.:0.000000 3rd Qu.:0.0e+00 3rd Qu.:0.0000
## Max. :1.000000 Max. :1.0e+00 Max. :1.0000
## glipizide_Up glipizide_Down glyburide_Steady
## Min. :0.000000 Min. :0.000000 Min. :0.00000
## 1st Qu.:0.000000 1st Qu.:0.000000 1st Qu.:0.00000
## Median :0.000000 Median :0.000000 Median :0.00000
## Mean :0.007566 Mean :0.005503 Mean :0.09113
## 3rd Qu.:0.000000 3rd Qu.:0.000000 3rd Qu.:0.00000
## Max. :1.000000 Max. :1.000000 Max. :1.00000
## glyburide_Up glyburide_Down tolbutamide_Steady
## Min. :0.000000 Min. :0.000000 Min. :0.000000
## 1st Qu.:0.000000 1st Qu.:0.000000 1st Qu.:0.000000
## Median :0.000000 Median :0.000000 Median :0.000000
## Mean :0.007979 Mean :0.005542 Mean :0.000226
## 3rd Qu.:0.000000 3rd Qu.:0.000000 3rd Qu.:0.000000
## Max. :1.000000 Max. :1.000000 Max. :1.000000
## pioglitazone_Steady pioglitazone_Up pioglitazone_Down
## Min. :0.00000 Min. :0.000000 Min. :0.000000
```

```
## 1st Qu.:0.00000 1st Qu.:0.000000 1st Qu.:0.000000
## Median :0.00000 Median :0.000000 Median :0.000000
## Mean :0.06855 Mean :0.002299 Mean :0.001159
## 3rd Qu.:0.00000 3rd Qu.:0.000000 3rd Qu.:0.000000
## Max. :1.00000 Max. :1.000000 Max. :1.000000
## rosiglitazone_Steady rosiglitazone_Up rosiglitazone_Down
## Min. :0.00000 Min. :0.000000 Min. :0.0000000
## 1st Qu.:0.00000 1st Qu.:0.000000 1st Qu.:0.0000000
## Median :0.00000 Median :0.000000 Median :0.0000000
## Mean :0.05994 Mean :0.001749 Mean :0.0008549
## 3rd Qu.:0.00000 3rd Qu.:0.000000 3rd Qu.:0.0000000
## Max. :1.00000 Max. :1.000000 Max. :1.0000000
## acarbose_Steady acarbose_Up acarbose_Down
## Min. :0.000000 Min. :0.00e+00 Min. :0.00e+00
## 1st Qu.:0.000000 1st Qu.:0.00e+00 1st Qu.:0.00e+00
## Median :0.000000 Median :0.00e+00 Median :0.00e+00
## Mean :0.002899 Mean :9.83e-05 Mean :2.95e-05
## 3rd Qu.:0.000000 3rd Qu.:0.00e+00 3rd Qu.:0.00e+00
## Max. :1.000000 Max. :1.00e+00 Max. :1.00e+00
## miglitol_Steady miglitol_Down miglitol_Up
```

```
## Min.   :0.0000000   Min.   :0.00e+00   Min.   :0.00e+00
## 1st Qu.:0.0000000   1st Qu.:0.00e+00   1st Qu.:0.00e+00
## Median :0.0000000   Median :0.00e+00   Median :0.00e+00
## Mean   :0.0003046   Mean   :4.91e-05   Mean   :1.97e-05
## 3rd Qu.:0.0000000   3rd Qu.:0.00e+00   3rd Qu.:0.00e+00
## Max.   :1.0000000   Max.   :1.00e+00   Max.   :1.00e+00
## troglitazone_Steady tolazamide_Steady   tolazamide_Up
## Min.   :0.00e+00    Min.   :0.0000000   Min.   :0.0e+00
## 1st Qu.:0.00e+00    1st Qu.:0.0000000   1st Qu.:0.0e+00
## Median :0.00e+00    Median :0.0000000   Median :0.0e+00
## Mean   :2.95e-05    Mean   :0.0003734   Mean   :9.8e-06
## 3rd Qu.:0.00e+00    3rd Qu.:0.0000000   3rd Qu.:0.0e+00
## Max.   :1.00e+00    Max.   :1.0000000   Max.   :1.0e+00
##   insulin_Up     insulin_Steady    insulin_Down
## Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
## 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
## Median :0.0000   Median :0.0000   Median :0.0000
## Mean   :0.1112   Mean   :0.3031   Mean   :0.1201
## 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
## Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
## glyburide.metformin_Steady glyburide.metformin_Down
## Min.   :0.0000           Min.   :0.0e+00
## 1st Qu.:0.0000           1st Qu.:0.0e+00
## Median :0.0000           Median :0.0e+00
## Mean   :0.0068           Mean   :5.9e-05
## 3rd Qu.:0.0000           3rd Qu.:0.0e+00
## Max.   :1.0000           Max.   :1.0e+00
## glyburide.metformin_Up glipizide.metformin_Steady
## Min.   :0.00e+00      Min.   :0.0000000
## 1st Qu.:0.00e+00      1st Qu.:0.0000000
## Median :0.00e+00      Median :0.0000000
## Mean   :7.86e-05      Mean   :0.0001277
## 3rd Qu.:0.00e+00      3rd Qu.:0.0000000
## Max.   :1.00e+00      Max.   :1.0000000
## glimepiride.pioglitazone_Steady metformin.rosiglitazone_Steady
## Min.   :0.0e+00               Min.   :0.00e+00
## 1st Qu.:0.0e+00               1st Qu.:0.00e+00
## Median :0.0e+00               Median :0.00e+00
## Mean   :9.8e-06               Mean   :1.97e-05
## 3rd Qu.:0.0e+00               3rd Qu.:0.00e+00
```

```
## Max.   :1.0e+00            Max.   :1.00e+00
## metformin.pioglitazone_Steady   change_Ch     diabetesMed_Yes
## Min.   :0.0e+00            Min.   :0.000   Min.   :0.00
## 1st Qu.:0.0e+00            1st Qu.:0.000   1st Qu.:1.00
## Median :0.0e+00            Median :0.000   Median :1.00
## Mean   :9.8e-06            Mean   :0.462   Mean   :0.77
## 3rd Qu.:0.0e+00            3rd Qu.:1.000   3rd Qu.:1.00
## Max.   :1.0e+00            Max.   :1.000   Max.   :1.00
```

```
Feature_Engineer_Integers <- function(data_set, features_to_ignore=c()) {
  require(infotheo)
  data_set <- data.frame(data_set)

  for (feature_name in setdiff(names(data_set), features_to_ignore)) {
    if (class(data_set[,feature_name])=='numeric' | class(data_set[,feature_name])=='integer') {
      feature_vector <- data_set[,feature_name]

      if (all((feature_vector - round(feature_vector)) == 0)) {
```

```
        # make sure we have more than 2 values excluding NAs
        if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 2) {
         print(feature_name)
         data_set[,paste0(feature_name,'_IsZero')] <- ifelse(data_set[,feature_name]==0,1,0)
         data_set[,paste0(feature_name,'_IsPositive')] <- ifelse(data_set[,feature_name]>=0,1,0)
         # separate data into two bins
         data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=2)
         data_set[,paste0(feature_name,'_2Bins')] <- data_discretized$X

         if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 4) {
          # try 4 bins
          data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=4)
          data_set[,paste0(feature_name,'_4Bins')] <- data_discretized$X
         }
        }
       }
      }
     }
   return (data_set)
}
```

```
diabetes <- Feature_Engineer_Integers(data_set=diabetes, features_to_ignore=c('admission_type_id',
                                       'discharge_disposition_id',
                                       'admission_source_id'))
## Loading required package: infotheo
## [1] "time_in_hospital"
## [1] "num_lab_procedures"
## [1] "num_procedures"
## [1] "num_medications"
## [1] "number_outpatient"
## [1] "number_emergency"
## [1] "number_inpatient"
## [1] "number_diagnoses"
nzv <- nearZeroVar(diabetes, saveMetrics = TRUE)
length(rownames(nzv[nzv$nzv==FALSE,]))
## [1] 66
diabetes <- diabetes[,rownames(nzv[nzv$nzv==FALSE,])]
dim(diabetes)
```

```
## [1] 101766     66
```

Now, let's use caret to model this data set using GBM. Here we will split the data into two portions: a training and a testing portion. We'll use the built-in createDataPartition from caret to split the data set in two. By using the same seed you will always get the same split in subsequent runs:

```
  # prep our variables
outcome_name <- 'readmitted'

# cleanup all feature names - replace periods with underscores
predictor_names <- setdiff(names(diabetes), outcome_name)

set.seed(1234)
splitIndex <- createDataPartition(diabetes[,outcome_name], p = .75, list = FALSE, times = 1)
train_data <- diabetes[ splitIndex,]
test_data  <- diabetes[-splitIndex,]



objControl <- trainControl(method='cv', number=2, returnResamp='none',
```

```
                 summaryFunction = twoClassSummary, classProbs = TRUE)

 # make outcome variable a factor (required for caret's GBM model)
gbm_caret_model <- train(train_data[,predictor_names], as.factor(train_data[,outcome_name]),
                 method='gbm',
                 trControl=objControl,
                 metric = "ROC",
                 preProc = c("center", "scale"))

## Loading required package: gbm
## Loading required package: survival
##
## Attaching package: 'survival'
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
## Loading required package: plyr
## ------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.6964             nan     0.1000    0.0017
##      2        0.6938             nan     0.1000    0.0013
##      3        0.6917             nan     0.1000    0.0010
##      4        0.6901             nan     0.1000    0.0008
```

```
##     5       0.6883            nan     0.1000    0.0008
##     6       0.6868            nan     0.1000    0.0006
##     7       0.6859            nan     0.1000    0.0005
##     8       0.6850            nan     0.1000    0.0004
##     9       0.6843            nan     0.1000    0.0003
##    10       0.6835            nan     0.1000    0.0004
##    20       0.6784            nan     0.1000    0.0002
##    40       0.6739            nan     0.1000    0.0000
##    60       0.6717            nan     0.1000    0.0000
##    80       0.6703            nan     0.1000   -0.0000
##   100       0.6691            nan     0.1000    0.0000
##   120       0.6681            nan     0.1000   -0.0000
##   140       0.6673            nan     0.1000   -0.0000
##   150       0.6669            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       0.6955            nan     0.1000    0.0021
##     2       0.6923            nan     0.1000    0.0015
##     3       0.6893            nan     0.1000    0.0014
##     4       0.6871            nan     0.1000    0.0011
```

```
##     5        0.6852             nan     0.1000    0.0009
##     6        0.6834             nan     0.1000    0.0009
##     7        0.6821             nan     0.1000    0.0006
##     8        0.6807             nan     0.1000    0.0006
##     9        0.6799             nan     0.1000    0.0004
##    10        0.6791             nan     0.1000    0.0004
##    20        0.6742             nan     0.1000    0.0001
##    40        0.6697             nan     0.1000   -0.0000
##    60        0.6662             nan     0.1000    0.0000
##    80        0.6638             nan     0.1000   -0.0000
##   100        0.6621             nan     0.1000   -0.0000
##   120        0.6610             nan     0.1000   -0.0000
##   140        0.6600             nan     0.1000   -0.0000
##   150        0.6596             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        0.6944             nan     0.1000    0.0023
##     2        0.6906             nan     0.1000    0.0018
##     3        0.6877             nan     0.1000    0.0013
##     4        0.6853             nan     0.1000    0.0012
```

```
##     5        0.6833             nan     0.1000    0.0010
##     6        0.6814             nan     0.1000    0.0009
##     7        0.6799             nan     0.1000    0.0007
##     8        0.6790             nan     0.1000    0.0004
##     9        0.6780             nan     0.1000    0.0004
##    10        0.6773             nan     0.1000    0.0003
##    20        0.6715             nan     0.1000    0.0002
##    40        0.6657             nan     0.1000    0.0000
##    60        0.6626             nan     0.1000    0.0000
##    80        0.6603             nan     0.1000   -0.0000
##   100        0.6580             nan     0.1000   -0.0000
##   120        0.6565             nan     0.1000   -0.0000
##   140        0.6552             nan     0.1000   -0.0000
##   150        0.6544             nan     0.1000   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        0.6960             nan     0.1000    0.0019
##     2        0.6932             nan     0.1000    0.0015
##     3        0.6907             nan     0.1000    0.0013
##     4        0.6886             nan     0.1000    0.0010
```

```
##     5     0.6870          nan     0.1000    0.0008
##     6     0.6854          nan     0.1000    0.0008
##     7     0.6841          nan     0.1000    0.0006
##     8     0.6828          nan     0.1000    0.0006
##     9     0.6819          nan     0.1000    0.0004
##    10     0.6811          nan     0.1000    0.0003
##    20     0.6747          nan     0.1000    0.0002
##    40     0.6697          nan     0.1000    0.0001
##    60     0.6677          nan     0.1000    0.0000
##    80     0.6660          nan     0.1000    0.0000
##   100     0.6651          nan     0.1000   -0.0000
##   120     0.6640          nan     0.1000   -0.0000
##   140     0.6632          nan     0.1000    0.0000
##   150     0.6629          nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1     0.6948          nan     0.1000    0.0023
##     2     0.6906          nan     0.1000    0.0021
##     3     0.6874          nan     0.1000    0.0016
##     4     0.6847          nan     0.1000    0.0011
```

```
##     5       0.6827           nan     0.1000    0.0010
##     6       0.6809           nan     0.1000    0.0009
##     7       0.6793           nan     0.1000    0.0007
##     8       0.6778           nan     0.1000    0.0007
##     9       0.6765           nan     0.1000    0.0006
##    10       0.6755           nan     0.1000    0.0005
##    20       0.6698           nan     0.1000    0.0002
##    40       0.6648           nan     0.1000    0.0000
##    60       0.6621           nan     0.1000    0.0000
##    80       0.6606           nan     0.1000   -0.0000
##   100       0.6586           nan     0.1000   -0.0000
##   120       0.6571           nan     0.1000    0.0000
##   140       0.6561           nan     0.1000   -0.0000
##   150       0.6554           nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       0.6939           nan     0.1000    0.0029
##     2       0.6893           nan     0.1000    0.0023
##     3       0.6859           nan     0.1000    0.0017
##     4       0.6830           nan     0.1000    0.0014
```

```
##     5     0.6803          nan     0.1000    0.0013
##     6     0.6783          nan     0.1000    0.0009
##     7     0.6765          nan     0.1000    0.0009
##     8     0.6753          nan     0.1000    0.0006
##     9     0.6740          nan     0.1000    0.0006
##    10     0.6729          nan     0.1000    0.0005
##    20     0.6672          nan     0.1000    0.0001
##    40     0.6618          nan     0.1000    0.0000
##    60     0.6582          nan     0.1000    0.0000
##    80     0.6555          nan     0.1000    0.0000
##   100     0.6531          nan     0.1000    0.0000
##   120     0.6511          nan     0.1000    0.0000
##   140     0.6499          nan     0.1000   -0.0000
##   150     0.6493          nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1     0.6942          nan     0.1000    0.0027
##     2     0.6903          nan     0.1000    0.0020
##     3     0.6871          nan     0.1000    0.0015
##     4     0.6843          nan     0.1000    0.0013
```

```
##     5        0.6818           nan     0.1000    0.0011
##     6        0.6800           nan     0.1000    0.0009
##     7        0.6785           nan     0.1000    0.0007
##     8        0.6771           nan     0.1000    0.0006
##     9        0.6759           nan     0.1000    0.0005
##    10        0.6751           nan     0.1000    0.0003
##    20        0.6696           nan     0.1000    0.0001
##    40        0.6647           nan     0.1000    0.0001
##    60        0.6614           nan     0.1000    0.0000
##    80        0.6597           nan     0.1000    0.0000
##   100        0.6582           nan     0.1000   -0.0000
##   120        0.6572           nan     0.1000    0.0000
##   140        0.6559           nan     0.1000   -0.0000
##   150        0.6555           nan     0.1000   -0.0000
```

```
summary(gbm_caret_model)
```

```
##                                                    var
```

```
## number_inpatient                    number_inpatient
## discharge_disposition_id            discharge_disposition_id
## number_inpatient_IsZero             number_inpatient_IsZero
## num_medications                     num_medications
## number_emergency                    number_emergency
## time_in_hospital                    time_in_hospital
## num_lab_procedures                  num_lab_procedures
## number_diagnoses                    number_diagnoses
## diag_1_428                          diag_1_428
## diabetesMed_Yes                     diabetesMed_Yes
## insulin_Down                        insulin_Down
## num_procedures                      num_procedures
## number_emergency_IsZero             number_emergency_IsZero
## number_diagnoses_2Bins              number_diagnoses_2Bins
## admission_type_id                   admission_type_id
## diag_3_401                          diag_3_401
## number_diagnoses_4Bins              number_diagnoses_4Bins
## payer_code_MC                       payer_code_MC
## change_Ch                           change_Ch
## age_.50.60.                         age_.50.60.
```

```
## time_in_hospital_2Bins                    time_in_hospital_2Bins
## admission_source_id                        admission_source_id
## gender_Male                                  gender_Male
## number_outpatient                            number_outpatient
## metformin_Steady                             metformin_Steady
## medical_specialty_NA                        medical_specialty_NA
## payer_code_HM                                  payer_code_HM
## race_AfricanAmerican                        race_AfricanAmerican
## time_in_hospital_4Bins                    time_in_hospital_4Bins
## num_medications_2Bins                        num_medications_2Bins
## diag_2_428                                   diag_2_428
## medical_specialty_Family.GeneralPractice medical_specialty_Family.GeneralPractice
## diag_3_250                                   diag_3_250
## medical_specialty_Emergency.Trauma          medical_specialty_Emergency.Trauma
## A1Cresult_.8                               A1Cresult_.8
## diag_3_Other                                 diag_3_Other
## num_procedures_IsZero                        num_procedures_IsZero
## age_.80.90.                                age_.80.90.
## age_.70.80.                                age_.70.80.
## diag_2_250                                   diag_2_250
```

```
## medical_specialty_Cardiology              medical_specialty_Cardiology
## diag_2_276                                  diag_2_276
## rosiglitazone_Steady                      rosiglitazone_Steady
## glipizide_Steady                          glipizide_Steady
## insulin_Steady                             insulin_Steady
## medical_specialty_InternalMedicine        medical_specialty_InternalMedicine
## glyburide_Steady                            glyburide_Steady
## num_lab_procedures_4Bins                      num_lab_procedures_4Bins
## number_outpatient_IsZero                     number_outpatient_IsZero
## num_medications_4Bins                         num_medications_4Bins
## age_.60.70.                                 age_.60.70.
## age_.40.50.                                 age_.40.50.
## diag_3_276                                  diag_3_276
## diag_1_414                                  diag_1_414
## pioglitazone_Steady                        pioglitazone_Steady
## insulin_Up                                  insulin_Up
## num_lab_procedures_2Bins                      num_lab_procedures_2Bins
## num_procedures_2Bins                          num_procedures_2Bins
## num_procedures_4Bins                          num_procedures_4Bins
## number_outpatient_2Bins                      number_outpatient_2Bins
```

```
## number_outpatient_4Bins                  number_outpatient_4Bins
## number_emergency_2Bins                   number_emergency_2Bins
## number_emergency_4Bins                   number_emergency_4Bins
## number_inpatient_2Bins                   number_inpatient_2Bins
## number_inpatient_4Bins                   number_inpatient_4Bins
##                              rel.inf
## number_inpatient            40.87029566
## discharge_disposition_id    25.78755007
## number_inpatient_IsZero      7.08713378
## num_medications              3.34553353
## number_emergency             3.27630233
## time_in_hospital             2.39670481
## num_lab_procedures           2.02396561
## number_diagnoses             1.60928573
## diag_1_428                   0.81986351
## diabetesMed_Yes              0.80382936
## insulin_Down                 0.75637593
## num_procedures               0.74798614
## number_emergency_IsZero      0.64643401
## number_diagnoses_2Bins       0.59160864
```

```
## admission_type_id                     0.58099922
## diag_3_401                            0.53642310
## number_diagnoses_4Bins                0.53503844
## payer_code_MC                         0.53396743
## change_Ch                             0.50894603
## age_.50.60.                           0.42698394
## time_in_hospital_2Bins                0.40122918
## admission_source_id                   0.35695272
## gender_Male                           0.35524920
## number_outpatient                     0.35469761
## metformin_Steady                      0.31412816
## medical_specialty_NA                  0.29430156
## payer_code_HM                         0.28613605
## race_AfricanAmerican                  0.27946576
## time_in_hospital_4Bins                0.25046225
## num_medications_2Bins                 0.23805674
## diag_2_428                            0.22904435
## medical_specialty_Family.GeneralPractice  0.22881666
## diag_3_250                            0.22192417
## medical_specialty_Emergency.Trauma    0.22007722
```

```
## A1Cresult_.8                       0.21118887
## diag_3_Other                       0.18957520
## num_procedures_IsZero              0.17662963
## age_.80.90.                        0.17179113
## age_.70.80.                        0.15863568
## diag_2_250                         0.14562289
## medical_specialty_Cardiology       0.13563927
## diag_2_276                         0.13314114
## rosiglitazone_Steady               0.12594872
## glipizide_Steady                   0.10558765
## insulin_Steady                     0.08551410
## medical_specialty_InternalMedicine 0.08011728
## glyburide_Steady                   0.07735456
## num_lab_procedures_4Bins           0.07439326
## number_outpatient_IsZero           0.05008065
## num_medications_4Bins              0.04929032
## age_.60.70.                        0.04295513
## age_.40.50.                        0.04024712
## diag_3_276                         0.03051853
## diag_1_414                         0.00000000
```

```
## pioglitazone_Steady          0.00000000
## insulin_Up                   0.00000000
## num_lab_procedures_2Bins     0.00000000
## num_procedures_2Bins         0.00000000
## num_procedures_4Bins         0.00000000
## number_outpatient_2Bins      0.00000000
## number_outpatient_4Bins      0.00000000
## number_emergency_2Bins       0.00000000
## number_emergency_4Bins       0.00000000
## number_inpatient_2Bins       0.00000000
## number_inpatient_4Bins       0.00000000
```

```
print(gbm_caret_model)
```

```
## Stochastic Gradient Boosting
##
## 76325 samples
```

```
##    65 predictor
##     2 classes: 'no', 'yes'
##
## Pre-processing: centered (65), scaled (65)
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 38162, 38163
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  ROC        Sens       Spec
##   1                   50      0.6481373  0.9994986  0.003052360
##   1                  100      0.6554346  0.9990266  0.006339516
##   1                  150      0.6596561  0.9986874  0.008217892
##   2                   50      0.6568785  0.9992036  0.005165532
##   2                  100      0.6659640  0.9986579  0.008570087
##   2                  150      0.6676805  0.9984072  0.010565861
##   3                   50      0.6622102  0.9988792  0.007278704
##   3                  100      0.6683780  0.9983630  0.010565861
##   3                  150      0.6694874  0.9986285  0.011035454
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
```

```
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
predictions <- predict(object=gbm_caret_model, test_data[,predictor_names], type='raw')
head(predictions)
## [1] no no no no no no
## Levels: no yes
print(postResample(pred=predictions, obs=as.factor(test_data[,outcome_name])))
##    Accuracy       Kappa
## 0.888015408 0.009486957
```

```
prop.table(table(as.factor(diabetes[,outcome_name])))
##
##        no       yes
## 0.8884008 0.1115992
```

```
# probabilites

predictions <- predict(object=gbm_caret_model, test_data[,predictor_names], type='prob')
head(predictions)
##          no        yes
## 1 0.9472629 0.05273706
## 2 0.9408044 0.05919557
## 3 0.9217157 0.07828429
## 4 0.9120175 0.08798251
## 5 0.8374687 0.16253128
## 6 0.9286577 0.07134228
```

```
# install.packages('pROC')

library(pROC)

## Type 'citation("pROC")' for a citation.

##

## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':

##

##     cov, smooth, var

auc <- roc(ifelse(test_data[,outcome_name]=="yes",1,0), predictions[[2]])

print(auc$auc)

## Area under the curve: 0.6616



######################################################################################################################################################
```

```
# k-MEANS CLUSTERING

Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, features_to_ignore=c()) {
 # look for text entries that are mostly unique
 text_features <- c(names(data_set[sapply(data_set, is.character)]), names(data_set[sapply(data_set, is.factor)]))
 for (f_name in setdiff(text_features, features_to_ignore)) {
  f_vector <- as.character(data_set[,f_name])
  # treat as raw text if data over minimum_precent_unique unique
  if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_unique_threshold)) {
   data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vector, " "), length)
   data_set[,paste0(f_name, '_character_count')] <- nchar(as.character(f_vector))
   data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.character(f_vector), " "), `[`,
                                    1)
   # remove orginal field
   data_set[,f_name] <- NULL
  }
 }
 return(data_set)
}
```

```
# Impute_Features

Impute_Features <- function(data_set, features_to_ignore=c(),
                            use_mean_instead_of_0=TRUE,
                            mark_NAs=FALSE,
                            remove_zero_variance=FALSE) {
  for (feature_name in setdiff(names(data_set), features_to_ignore)) {
    print(feature_name)
    # remove any fields with zero variance
    if (remove_zero_variance) {
      if (length(unique(data_set[, feature_name]))==1) {
        data_set[, feature_name] <- NULL
        next
      }
    }
    if (mark_NAs) {
      # note each field that contains missing or bad data
      if (any(is.na(data_set[,feature_name]))) {
```

```
      # create binary column before imputing
      newName <- paste0(feature_name, '_NA')
      data_set[,newName] <- as.integer(ifelse(is.na(data_set[,feature_name]),1,0)) }
     if (any(is.infinite(data_set[,feature_name]))) {
      newName <- paste0(feature_name, '_inf')
      data_set[,newName] <- as.integer(ifelse(is.infinite(data_set[,feature_name]),1,0)) }
    }
    if (use_mean_instead_of_0) {
     data_set[is.infinite(data_set[,feature_name]),feature_name] <- NA
     data_set[is.na(data_set[,feature_name]),feature_name] <- mean(data_set[,feature_name], na.rm=TRUE)
    } else {
     data_set[is.na(data_set[,feature_name]),feature_name] <- 0
     data_set[is.infinite(data_set[,feature_name]),feature_name] <- 0
    }
   }
  return(data_set)
 }
```

```
AutoMpg_data <- read.csv("http://mlr.cs.umass.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data", na.strings = '?', header=FALSE, sep="",
as.is=TRUE, col.names = c("mpg", "cylinders", "displacement", "horsepower",
                                             "weight", "acceleration", "model", "origin", "car_name"), stringsAsFactors = FALSE)

AutoMpg_data <- Get_Free_Text_Measures(data_set = AutoMpg_data, minimum_unique_threshold=0.5)

AutoMpg_data <- Impute_Features(data_set = AutoMpg_data, use_mean_instead_of_0 = FALSE)
```

```
## [1] "mpg"

## [1] "cylinders"

## [1] "displacement"

## [1] "horsepower"

## [1] "weight"

## [1] "acceleration"

## [1] "model"

## [1] "origin"

## [1] "car_name_word_count"

## [1] "car_name_character_count"

## [1] "car_name_first_word"
```

str(AutoMpg_data)

'data.frame': 398 obs. of 11 variables:

$ mpg : num 18 15 18 16 17 15 14 14 14 15 ...

$ cylinders : num 8 8 8 8 8 8 8 8 8 8 ...

$ displacement : num 307 350 318 304 302 429 454 440 455 390 ...

$ horsepower : num 130 165 150 150 140 198 220 215 225 190 ...

$ weight : num 3504 3693 3436 3433 3449 ...

$ acceleration : num 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...

$ model : num 70 70 70 70 70 70 70 70 70 70 ...

$ origin : num 1 1 1 1 1 1 1 1 1 1 ...

$ car_name_word_count : num 3 3 2 3 2 3 2 3 2 3 ...

$ car_name_character_count: num 25 17 18 13 11 16 16 17 16 18 ...

$ car_name_first_word : chr "chevrolet" "buick" "plymouth" "amc" ...

```
library(dplyr)

##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
## filter, lag
##
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union

set.seed(1234)
```

```
km1 = kmeans(x = select(AutoMpg_data, weight, acceleration), centers = 3)

# Plot results

plot(select(AutoMpg_data, weight, acceleration),

    col =km1$cluster, main="K-Means result with 3 clusters",

    pch=20, cex=2)

# find each cluster's centroids

points(km1$centers, pch=6, col='blue', cex=6)

points(km1$centers, pch=6, col='blue', cex=4)

points(km1$centers, pch=6, col='blue', cex=2)
```

```
unique(AutoMpg_data$car_name_first_word)
```

```
## [1] "chevrolet" "buick" "plymouth" "amc"

## [5] "ford" "pontiac" "dodge" "toyota"

## [9] "datsun" "volkswagen" "peugeot" "audi"

## [13] "saab" "bmw" "chevy" "hi"
```

```
## [17] "mercury" "opel" "fiat" "oldsmobile"
## [21] "chrysler" "mazda" "volvo" "renault"
## [25] "toyouta" "maxda" "honda" "subaru"
## [29] "chevroelt" "capri" "vw" "mercedes-benz"
## [33] "cadillac" "mercedes" "vokswagen" "triumph"
## [37] "nissan"
```

```
brand_set <- select(AutoMpg_data, weight, acceleration, car_name_first_word) %>%
  group_by(car_name_first_word) %>% summarize_each(funs(mean)) %>% data.frame
row.names(brand_set) <- brand_set$car_name_first_word
brand_set <- dplyr::select(brand_set, -car_name_first_word)
set.seed(1234)
km1 = kmeans(x = brand_set, centers = 3)
# Plot results
plot(brand_set,
     col =km1$cluster, main="K-Means result with 3 clusters",
     pch=20, cex=2)
```

```
# install.packages('factoextra')
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.2.5
```

```
## Loading required package: ggplot2
```

```
set.seed(1234)
km1 = kmeans(x = brand_set, centers = 3)
print(km1)
```

K-means clustering with 3 clusters of sizes 4, 14, 19

##

Cluster means:

weight acceleration

1 4170.250 16.26250

2 3377.402 16.06959

3 2250.898 15.68187

##

Clustering vector:

amc audi bmw buick cadillac

2 3 3 2 1

capri chevroelt chevrolet chevy chrysler

3 1 2 2 1

datsun dodge fiat ford hi

3 2 3 2 1

honda maxda mazda mercedes mercedes-benz

3 3 3 2 2

mercury nissan oldsmobile opel peugeot

2 3 2 3 2

plymouth pontiac renault saab subaru

```
## 2 2 3 3 3
## toyota toyouta triumph vokswagen volkswagen
## 3 3 3 3 3
## volvo vw
## 2 3
##
## Within cluster sum of squares by cluster:
## [1] 457857.8 676136.5 885185.0
## (between_SS / total_SS = 89.7 %)
##
## Available components:
##
## [1] "cluster" "centers" "totss" "withinss"
## [5] "tot.withinss" "betweenss" "size" "iter"
## [9] "ifault"
```

```
set.seed(1234)
fviz_nbclust(brand_set, kmeans, method = "wss")
```

##