Predicting Stroke

Leveraging Machine Learning Techniques in Electronic Health Records

Sue-Ellen Duffy

2024-12-14

Table of contents


```
library(tidyverse)
library(tidyr)
library(kableExtra)
```

```r
library(tidymodels)
library(stringr)
library(broom.mixed)
library(dotwhisker)
library(skimr)
library(GGally)
library(corrr)
library(visdat)
library(discrim)
library(rsample)
library(tidytext)
library(DataExplorer)
library(naniar)
library(xgboost)
library(corrplot)
library(ggplot2)
library(reshape2)
library(future)
library(pROC)
library(caret)
library(viridis)
library(inspectdf)
library(gridExtra)
```

1 Introduction and Data

Electronic health records (EHRs) are comprehensive medical datasets that capture information about patient visits, diagnoses, lab results, and more. A typical doctor's visit might include a survey about lifestyle factors, such as smoking and drinking, and potential diagnoses based on symptoms or lab results. These records are essential for understanding medical trends and outcomes but are protected under regulations like HIPAA to ensure patient privacy. Consequently, accessing real-world EHRs for research purposes can be challenging.

Synthetic datasets like those generated by Synthea provide a solution. These datasets simulate realistic healthcare interactions and patient data, enabling researchers to explore complex medical issues without compromising patient confidentiality. Such data is invaluable for testing machine learning models and preparing for large-scale studies.

This project focuses on predicting stroke risk using a synthetic EHR dataset that represents individuals aged 30-70 from Massachusetts. The dataset is designed to mimic real-world healthcare patterns and includes detailed patient information, such as symptoms, diagnoses, and lab results. It provides a unique opportunity to explore machine learning in the context of health outcomes.

1.1 Motivation

The primary goal of this project is to evaluate the effectiveness of machine learning models in predicting stroke risk. Key guiding questions include:

- Are patients providing symptoms that clearly indicate disease?
- Can doctors leverage patient-reported symptoms to improve early diagnosis?
- How can machine learning enhance diagnostic processes and support early, accurate identification of high-risk patients?

By identifying patterns in symptoms and diagnoses, this analysis aims to improve diagnostic tools, potentially leading to earlier intervention and better outcomes. Additionally, this project examines the influence of known high-risk factors on model performance. If a patient lacks these factors, can their EHR still predict stroke?

1.2 Research Question

How does the inclusion or exclusion of stroke-specific risk factors in electronic health records affect the ability of machine learning models to accurately predict stroke, considering both the accuracy of positive predictions (precision) and the ability to identify all true cases (recall)?

1.3 Data Overview

The dataset used in this project is part of Synthea's synthetic stroke series and includes 30,000 patients, with a 1:3 ratio of stroke to non-stroke cases. Synthea generates data through simulations designed to reflect real-world healthcare scenarios, including symptom reporting, lab results, and diagnoses.

1.3.0.1 Access to Data:

The dataset is available on the Harvard Dataverse:

https://dataverse.harvard.edu/file.xhtml?fileId=6707765&version=1.0

1.3.0.2 Data Curation by Original Authors:

The dataset was transformed many times from the original source of Synthea's synthetic data by the original authors (Chen, 2022) so that it could be utilized by other analysts. The following bulletpoints highlight the process of data collection and transformation performed by these authors:

- Extract 10 sets of patient populations of 15K living or deceased synthetic patients age 30 years or older from Synthea
  - Each population had about 15K living or deceased synthetic patients, and patient data were stored in different domain record files:
    1) patients.csv
    2) encounters.csv

3) conditions.csv
4) observations.csv
5) procedures.csv
6) medications.csv
7) immunizations.csv
8) allergies.csv

  - Each population of 15K contained ~5GB of data

- Create 5 subsets of 30K patients
- Convert patient records into a single row of data per patient.
  - **For Patients with Stroke**: Deduplicate datasets and captures the most recent patient data prior to stroke incident.
  - **For Patients without Stroke**: Average the patient data, choosing most common value for categorical data and the average value for numeric data. Age is determined by last piece of data collected.
- Update datasets using resampling for imbalanced datasets (patients with or without stroke)
- Convert values from continuous numeric format into categorical values based on commonly used ranges (e.g.,value of < 200 mg/dL total cholesterol was converted to the category "normal")

See Supplementary Information, accessible through this link for more information: https://www.nature.com/article
022-23011-4#MOESM1

1.4 Read in Data

Two datasets are required. One contains the patient data and electronic health records with column names that are medical codes. The other dataset contains the codebook to transpose these medical codes to medical names.

- **data**:
  - https://dataverse.harvard.edu/file.xhtml?fileId=6707764&version=1.0
- **codebook**:
  - https://dataverse.harvard.edu/file.xhtml?fileId=6707413&version=2.0

Each row represents one person, the outcome (stroke or no stroke) and 800 variables representing their electronic health records.

```
dim(data)
```

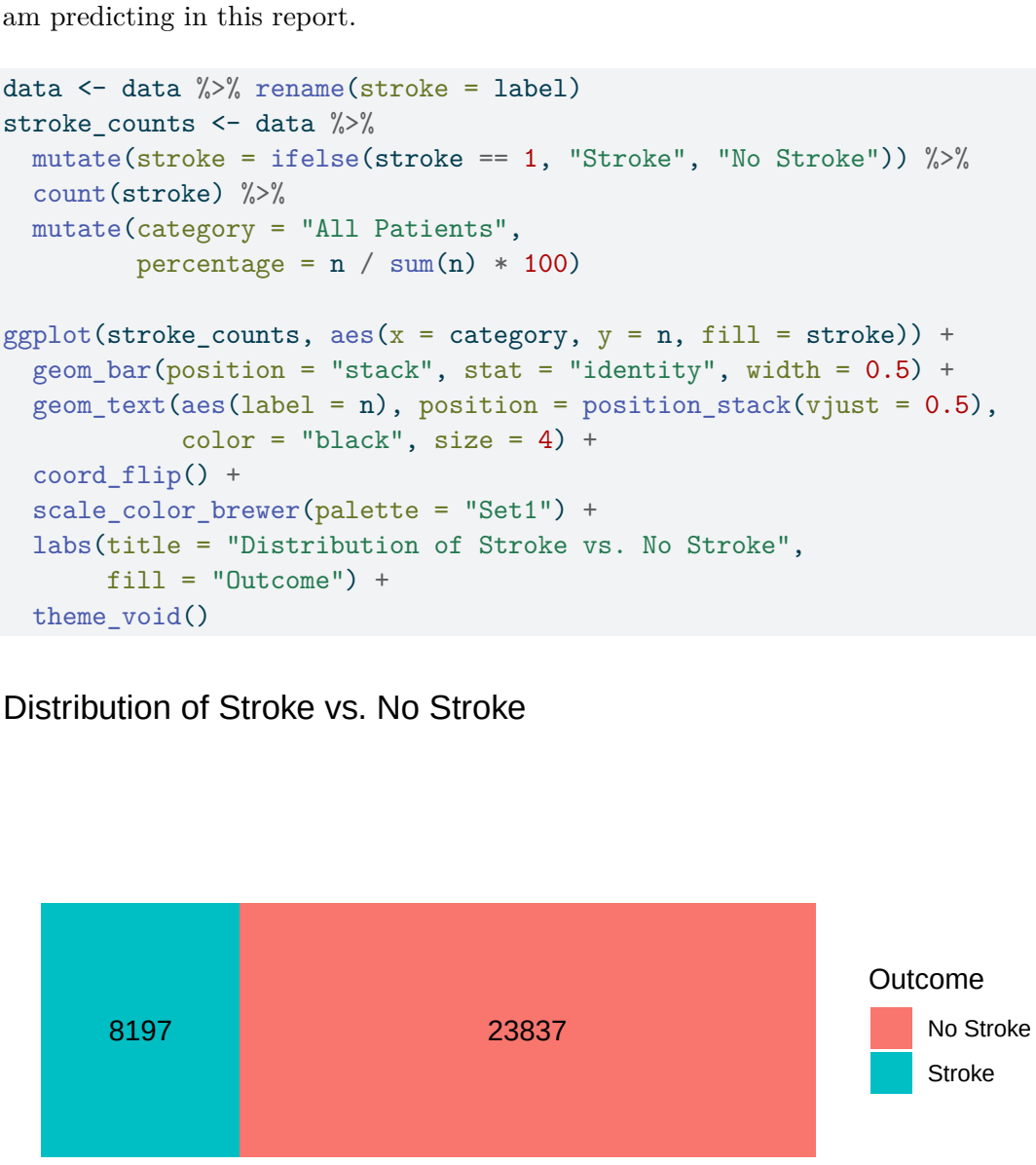
```
[1] 32034   803
```

There are 32034 patients in the data.

1.4.1 Stroke as Outcome

The **Outcome Variable** is a binary variable: Stroke or No Stroke. This captures the outcome I am predicting in this report.

```
data <- data %>% rename(stroke = label)
stroke_counts <- data %>%
  mutate(stroke = ifelse(stroke == 1, "Stroke", "No Stroke")) %>%
  count(stroke) %>%
  mutate(category = "All Patients",
         percentage = n / sum(n) * 100)

ggplot(stroke_counts, aes(x = category, y = n, fill = stroke)) +
  geom_bar(position = "stack", stat = "identity", width = 0.5) +
  geom_text(aes(label = n), position = position_stack(vjust = 0.5),
            color = "black", size = 4) +
  coord_flip() +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Distribution of Stroke vs. No Stroke",
       fill = "Outcome") +
  theme_void()
```



The outcome ratio of stroke to no stroke is 1:3 with 25% of the patients resulting in stroke and 75% of the patients resulting in no stroke.

2 Exploratory Data Analysis

In this section I will clean the data where it is necessary, perform column based transformations to prepare datasets for machine learning models, and after splitting the data for training and testing, I will explore the nature of the taraining data through visual and numerical analysis.

2.1 Data Cleaning

This dataset has been curated by a team of researchers to be ready for machine learning. The cleaning that needs to be done is minimal. There are some small errors in the codebook, so I have to update those values.

2.1.1 Medical Codes and Names

The data starts with 3 labeled columns and the rest of the data are coded columns. These codes are standard medical codes, utilized in the data so that there would be no confusion as to what a specific datapoint is measuring. The codes are a combination of LOINC and SNOMED-CT codes:

- LOINC is the world's most widely used terminology standard for health measurements, observations, and documents.
- SNOMED- CT is primarily a code system for recording diagnoses

The codebook provides a conversion from medical code to name of code in English.

Incomplete codenames, including shorthand of the code and open parenthesis, caused errors in when aligning the codebook data with the patient data. I searched the codes on LOINC.org and input the correct names.

```
#fix codebook duplicate names
codebook <- codebook %>%
  mutate(
    name =ifelse(code == "C-2028-9",
          "Carbon Dioxide [Mass/volume] in Serum or Plasma", name),
    name = ifelse(code == "C-2160-0",
          "Creatine [Mass/volume] in Serum or Plasma", name),
    name = ifelse(code == "C-17861-6",
          "Calcium [Mass/volume] in Serum or Plasma", name),
    name = ifelse(code == "C-3094-0",
          "Urea Nitrogen [Mass/volume] in Serum or Plasma", name),
    name = ifelse(code == "C-417181009",
         "Hormone receptor positive malignant neoplasm of breast (disorder)",
         name),
    name = ifelse(code == "C-311555007",
          "Speech and language therapy regime (regime/therapy)", name),
    name = ifelse(code == "C-80583007",
           "Severe anxiety (panic) (finding)", name),
```

```
    name = ifelse(code == "C-200346",
           "Cefdinir_a", name),
    name = ifelse(code == "C-25037",
           "Cefdinir_b", name))
#codebook to lowercase
codebook$name <- tolower(codebook$name)
```

I remove the "scc" column. The scc is a numerical value assigned to each individual. It is located right after ptnum and right before the stroke stroke. I am concerned that this is a weight assigned to patients. After looking through the codebook documentation I could not find a clear indicator as to what this data references, so I removed it from the dataset entirely to remove any pre-processing or data leakage that may come from this column.

```
#remove "scc". no documentation. might be a weight created by authors.
data <- data %>% select(!scc)
```

Next I assign the codebook names to the dataframe.

```
### Assign LOINC names to data columns
stroke_codes <- data
cols_to_rename <- names(stroke_codes)[-(1:2)]
new_names <- codebook$name[match(cols_to_rename, codebook$code)]
names(stroke_codes)[-(1:2)] <- new_names
```

2.1.2 Data Split Reproducibility

In order to create uniformity and reproducibility throughout transformations of the data and across various models, I create a data split for training data and testing data that can be used throughout this project.

I also create a column that represents the sample training data (50% of training data) for reproducibility.

```
#create data split that can be reproduced throughout the dataset
set.seed(123)
pre_transformation_stroke_data <- stroke_codes %>%
  group_by(stroke) %>%
  mutate(
    test = sample(c(1, 0), size = n(), replace = TRUE, prob = c(0.2, 0.8))
  ) %>%
  ungroup()

# Step 2: Split the training data into halves while maintaining proportions
set.seed(456)  # Seed for reproducibility
training_sample_a <- pre_transformation_stroke_data %>%
  filter(test == 0) %>%  # Use only training data
```

```
  group_by(stroke) %>%
  slice_sample(prop = 0.5) %>%  # Sample 50% of training data per stroke
  ungroup()
pre_transformation_stroke_data <- pre_transformation_stroke_data %>%
  mutate(
    train_sample = if_else(
      ptnum %in% training_sample_a$ptnum, 1,
      if_else(test == 0, 0, NA_real_)))
```

2.2 Data Transformations

In order to analyze the data in accordance to my research question, I need to create two datasets - a stroke risk factor inclusion dataset and a stroke risk factor exclusion dataset.

I will group the variables of the stroke risk inclusion dataset for a more thorough investigation. However to group the variables of the stroke risk factor exclusion dataset would require a medical expert.

2.2.1 Stroke Risk Factors

I utilize the John Hopkins list of top 22 risk factors for stroke to sort my data.

Variables that fit into these categories will become the Stroke Data Inclusion Dataset and variables that do not fit into these categories will become the Stroke Data Exclusion Dataset.

All 22 risk factors are listed below and risk factors in **bold** are available in the dataset, and therefore will be used in analysis.

Risk factors with a star (*) also contain a category for a possible connection with a stroke risk factor.

Categories like lack of exercise, obesity, history, heredity, location, and temperature will not be possible with the limited data.

Top 22 Risk Factors for Stroke

1. **High blood pressure**(*)
2. **Heart disease**(*)
3. **Diabetes**(*)
4. **Smoking**
5. **Birth control pills (oral contraceptives)**
6. History of TIAs (transient ischemic attacks) (mini-strokes)
7. **High red blood cell count**(*)
8. **High blood cholesterol and lipids**
9. Lack of exercise
10. Obesity(*)
11. **Excessive alcohol use**
12. **Illegal drugs**
13. **Abnormal heart rhythm (atrial fibrillation, irregular heartbeat)**

14. **Cardiac structural abnormalities, damaged heart valves (valvular heart disease)**
15. **Older age**
16. **Race**
17. **Gender**
18. History of prior stroke*
19. Heredity or genetics (family history of stroke)
20. Location
21. Temperature, season, climate
22. Social and economic factors

2.2.2 Risk Inclusion Data

Stroke Risk Factor Inclusion Dataset

Below, I manually sort the variables into their stroke risk groups, aligning with the John Hopkins top 22 stroke risks.

I utilized chatgpt, as well as manual searches of data using google, to sort this data into groups. This is a section that really would require a data expert in the field of stroke to sort through. However, for the purposes of a class project, this method is what I could manage.

```
### Assign Risk Factors Groups from EHR Variables
high_blood_pressure <- c(
  "diastolic blood pressure", "systolic blood pressure",
  "10 ml furosemide 10 mg/ml injection", "amlodipine 2.5 mg oral tablet",
  "captopril 25 mg oral tablet", "hydrochlorothiazide 25 mg oral tablet",
  "hypertension", "lisinopril", "lisinopril 10 mg oral tablet",
  "lisinopril 20 mg oral tablet", "mean blood pressure")
possible_high_blood_pressure <- c(
  "amlodipine 5 mg oral tablet", "losartan potassium 50 mg oral tablet",
  "referral to hypertension clinic", "verapamil hydrochloride 40 mg")
heart_disease <- c(
  "coronary heart disease", "chronic congestive heart failure (disorder)",
  "heart failure education (procedure)",  "heart failure (disorder)",
  "transplantation of heart (procedure)", "coronary artery bypass grafting",
  "myocardial infarction", "history of myocardial infarction (situation)",
  "nitroglycerin 0.4 mg/actuat mucosal spray",
  "acute pulmonary embolism (disorder)", "cardiac arrest",
  "carvedilol 25 mg oral tablet", "history of cardiac arrest (situation)",
  "percutaneous coronary intervention", "warfarin sodium 5 mg oral tablet",
  "sacubitril 97 mg / valsartan 103 mg oral tablet")
possible_heart_disease <- c(
  "echocardiography (procedure)", "electrocardiographic procedure",
  "hemoglobin / hematocrit / platelet count",
  "objective assessment of cardiovascular disease nyha", "nt-probnp",
  "1 ml vasopressin (usp) 20 unt/ml injection",
  "troponin i.cardiac [mass/volume] in serum or plasma by high sensitivity method",
  "1 ml heparin sodium  porcine 5000 unt/ml injection", "heart rate",
```

```
  "assessment using morse fall scale (procedure)",
  "cardiovascular stress testing (procedure)", "shock (disorder)",
  "clopidogrel 75 mg oral tablet")
diabetes <- c(
  "diabetes", "hemoglobin a1c/hemoglobin.total in blood",
  "neuropathy due to type 2 diabetes mellitus (disorder)",
  "diabetic retinopathy associated with type ii diabetes mellitus (disorder)",
  "nonproliferative diabetic retinopathy due to type 2 diabetes mellitus (disorder)",
  "microalbuminuria due to type 2 diabetes mellitus (disorder)",
  "macular edema and retinopathy due to type 2 diabetes mellitus (disorder)",
  "proliferative diabetic retinopathy due to type ii diabetes mellitus (disorder)",
  "proteinuria due to type 2 diabetes mellitus (disorder)" ,
  "blindness due to type 2 diabetes mellitus (disorder)",
  "24 hr metformin hydrochloride 500 mg extended release oral tablet",
  "3 ml liraglutide 6 mg/ml pen injector", "hyperglycemia (disorder)",
  "canagliflozin 100 mg oral tablet", "diabetic renal disease (disorder)",
  "insulin lispro 100 unt/ml injectable solution [humalog]")
possible_diabetes <- c(
  "amputation of left foot", "history of lower limb amputation (situation)",
  "amputation of left leg", "glucose [mass/volume] in serum or plasma",
  "oral glucose tolerance test", "glucose [presence] in urine by test strip",
  "glucose", "history of amputation of foot (situation)", "prediabetes")
smoking <- c(
  "tobacco smoking status nhis", "smokes tobacco daily",
  "24hr nicotine transdermal patch")
birth_control_oral <- c(
  "yaz 28 day pack", "ortho tri-cyclen 28 day pack", "camila 28 day pack",
  "errin 28 day pack", "estrostep fe 28 day pack", "jolivette 28 day pack",
  "levora 0.15/30 28 day pack", "mestranol / norethynodrel [enovid]",
  "seasonique 91 day pack", "trinessa 28 day pack" )
high_red_blood_cell_count <- c(
  "hemoglobin [mass/volume] in blood",
  "hematocrit [volume fraction] of blood by automated count",
  "erythrocytes [#/volume] in blood by automated count")
possible_high_red_blood_cell_count <- c(
  "1 ml epoetin alfa 4000 unt/ml injection [epogen]", "anemia (disorder)",
  "platelets [#/volume] in blood by automated count")
high_blood_cholesterol_and_lipids <- c(
  "low density lipoprotein cholesterol",
  "high density lipoprotein cholesterol", "total cholesterol",
  "triglycerides", "hyperlipidemia", "simvastatin 10 mg oral tablet",
  "simvastatin 20 mg oral tablet", "atorvastatin 80 mg oral tablet",
  "hypertriglyceridemia (disorder)")
possible_obesity <- c("body mass index")
alcohol <- c(
  "unhealthy alcohol drinking behavior (finding)", "alcoholism",
  "naltrexone hydrochloride 50 mg oral tablet")
```

```
drug_misuse <- c(
  "misuses drugs (finding)", "drug overdose",
  "methadone hydrochloride 10 mg oral tablet", "opioid abuse (disorder)")
abnormal_heart_rhythm <- c(
  "atrial fibrillation", "catheter ablation of tissue of heart",
  "insertion of biventricular implantable cardioverter defibrillator",
  "3 ml amiodarone hydrocholoride 50 mg/ml prefilled syringe",
  "atropine sulfate 1 mg/ml injectable solution",
  "digoxin 0.125 mg oral tablet", "electrical cardioversion")
cardiac_structural_abnormalities <- c(
  "injury of heart (disorder)", "left ventricular ejection fraction",
  "implantation of left ventricular assist device (procedure)")
possible_prior_stroke <- c(
  "0.3 ml enoxaparin sodium 100 mg/ml prefilled syringe",
  "0.4 ml enoxaparin sodium 100 mg/ml prefilled syringe",
  "1 ml enoxaparin sodium 150 mg/ml prefilled syringe")
age <- c("age")
race <- c("race", "ethnic")
gender <- c("gender")
```

2.2.2.1 Create Dataframe

In order to easily address the individual variables by their groups, add the group abbreviation as a prefix to each variable.

```
stroke_risk_inclusion_groups <- c("ptnum", "stroke", "test", "train_sample",
  high_blood_pressure, possible_high_blood_pressure, heart_disease,
  possible_heart_disease, diabetes, possible_diabetes, smoking,
  birth_control_oral, high_red_blood_cell_count,
  possible_high_red_blood_cell_count, high_blood_cholesterol_and_lipids,
  possible_obesity, alcohol, drug_misuse, abnormal_heart_rhythm,
  cardiac_structural_abnormalities, possible_prior_stroke, age, race, gender)

filtered_stroke_risk <- pre_transformation_stroke_data %>%
  select(all_of(stroke_risk_inclusion_groups))

filtered_stroke_risk <- filtered_stroke_risk %>%
  rename_with(~ case_when(
    . %in% high_blood_pressure ~ paste0("hbp_", .),
    . %in% possible_high_blood_pressure ~ paste0("hbp2_", .),
    . %in% heart_disease ~ paste0("hd_", .),
    . %in% possible_heart_disease ~ paste0("hd2_", .),
    . %in% diabetes ~ paste0("d_", .),
    . %in% possible_diabetes ~ paste0("d2_", .),
    . %in% smoking ~ paste0("s_", .),
    . %in% birth_control_oral ~ paste0("bco_", .),
    . %in% high_red_blood_cell_count ~ paste0("hrbcc_", .),
```

```
    . %in% possible_high_red_blood_cell_count ~ paste0("hrbcc2_", .),
    . %in% high_blood_cholesterol_and_lipids ~ paste0("hbcal_", .),
    . %in% possible_obesity ~ paste0("o2_", .),
    . %in% alcohol ~ paste0("a_", .),
    . %in% drug_misuse ~ paste0("dm_", .),
    . %in% abnormal_heart_rhythm ~ paste0("ahr_", .),
    . %in% cardiac_structural_abnormalities ~ paste0("csa_", .),
    . %in% possible_prior_stroke ~ paste0("ps2_", .),
    . %in% age ~ paste0("age_", .),
    . %in% race ~ paste0("race_", .),
    . %in% gender ~ paste0("gender_", .),
    TRUE ~ . ))
```

2.2.2.2 Transition Data to Numeric

First create a predictor, missing_count, that captures the missingness of data (NA values) in a person's stroke inclusion electronic health record variables.

Then, transition the logical data to a binary format. For all values that are true, abnormal, or essentially indicating "Yes, this variable present" I assign the value 1 and for all other values which indicate "No, this variable is not present" I assign the value 0.

```
filtered_stroke_risk2 <- filtered_stroke_risk %>% mutate(
    missing_count = rowSums(is.na(.)),
    across(where(~ all(. %in% c(TRUE, FALSE, NA))),
      ~ if_else(. == TRUE, 1, 0, NA)),
    across(where(~ all(. %in% c("abnormal", "normal", NA))),
      ~ if_else(. == "abnormal", 1,
                if_else(. == "normal", 0, NA))),
    across(where(~ all(. %in% c("former", "current every day smoker", "never", NA))),
      ~ if_else(. == "former" |. == "current every day smoker", 1,
                if_else(. == "never", 0, NA))),
    across(where(~ all(. %in% c("severe", "mod-severe", "minimal", NA))),
      ~ if_else(. == "severe"|. == "mod-severe", 1,
                if_else(. == "minimal", 0, NA))))
```

Now that all variables have been assigned a value, the group totals can be summed.

```
group_prefixes <- sub("(_.*)", "", colnames(filtered_stroke_risk2))

for (group in unique(group_prefixes)) {
  group_columns <- colnames(filtered_stroke_risk2)[group_prefixes ==
                    group & sapply(filtered_stroke_risk2, is.numeric)]
  new_colname <- paste0(group)
  if (!(new_colname %in% colnames(filtered_stroke_risk2))) {
    filtered_stroke_risk2[[new_colname]] <-
      rowSums(filtered_stroke_risk2[,group_columns,
```

```
                                        drop = FALSE], na.rm = TRUE)
  }
}
```

Bring together the essential details including the ptnum, stroke, test, train_sample, new variable:missing_count and all the stroke risk inclusion groups.

Adjust the weighting on possible* risk groups to 25% of the value to reduce their influence.

I also clean the demographic names to reduce repetition in naming.

```
stroke_risk_inclusion_data <- filtered_stroke_risk2 %>%
  select(ptnum, stroke, test, train_sample, missing_count, hbp, hbp2, hd, hd2,
         d, d2, s, bco, hrbcc, hrbcc2, hbcal, o2, a, dm, ahr, csa, ps2, age_age,
         gender_gender, race_race, race_ethnic)

stroke_risk_inclusion_data <- stroke_risk_inclusion_data %>%
    mutate(across(ends_with("2"),  #select the possible_ columns
      ~ . * .25, .names = "{.col}")) #reduce weight by 75%

stroke_risk_inclusion_data <- stroke_risk_inclusion_data %>%
  rename(age = age_age,
         race = race_race,
         ethnic = race_ethnic,
         gender = gender_gender)
```

Shorter or longer variable names are helpful in different contexts. I create a column_stroke vector for any situations where a longer variable name may be better suited.

```
# Create a named vector for column name mapping
column_strokes <- c(
  stroke = "outcome:stroke",
  missing_count = "missingness count",
  test = "test_train_split",
  train_sample = "training_sample_split",
  hbp = "high blood pressure",
  hbp2 = "possible high blood pressure",
  hd = "heart disease",
  hd2 = "possible heart disease",
  d = "diabetes",
  d2 = "possible diabetes",
  s = "smoking",
  bco = "birth control (pill)",
  hrbcc = "high red blood cell count",
  hrbcc2 = "possible high red blood cell count",
  hbcal = "high blood cholesterol and lipids",
  o2 = "possible obesity",
```

```
  a = "alcohol",
  dm = "drug misuse",
  ahr = "abnormal heart rhythm",
  csa = "cardiac structural abnormalities",
  ps2 = "possible prior stroke"
)
```

2.2.3 Risk Exclusion Data

Stroke Risk Factor Exclusion

2.2.3.1 Create Dataframe

Here I remove the variables that are used for the inclusion dataset.

```
risk_variables <- c(
  high_blood_pressure, possible_high_blood_pressure,
  heart_disease, possible_heart_disease, diabetes, possible_diabetes, smoking,
  birth_control_oral, high_red_blood_cell_count,
  possible_high_red_blood_cell_count, high_blood_cholesterol_and_lipids,
  possible_obesity, alcohol, drug_misuse, abnormal_heart_rhythm,
  cardiac_structural_abnormalities, possible_prior_stroke, age, race, gender)

exclusion_df <- pre_transformation_stroke_data %>%
  select(-all_of(risk_variables))
```

```
exclusion_df2 <- exclusion_df %>% mutate(
    across(where(~ all(. %in% c(TRUE, FALSE, NA))),
      ~ if_else(. == TRUE, 1, 0, NA)),
    across(where(~ all(. %in% c("abnormal", "normal", NA))),
      ~ if_else(. == "abnormal", 1,
                if_else(. == "normal", 0, NA))),
    across(where(~ all(. %in% c("positive", "negative", NA))),
      ~ if_else(. == "positive", 1,
                if_else(. == "negative", 0, NA))))
```

```
dim(exclusion_df2)
```

```
[1] 32034   682
```

There are 32034 people in the non_risk dataset each with a patient id, outcome variable and various entries amongst 680 electronic health record variables.

2.3 Data Split - Train and Test

I have created two datasets: risk inclusion and risk exclusion so I need to create separate data splits for each dataset.

2.3.1 Risk Inclusion Variables

Using the columns assigned earlier I create separate testing (20%), training (80%), and sample training (50% of training data) dataframes.

```
#order stroke as factor
inclusion_model_df <- stroke_risk_inclusion_data %>%
  mutate(stroke = as.factor(stroke))
inclusion_train<- inclusion_model_df %>% filter(test == "0")
inclusion_train_sample <- inclusion_model_df %>% filter(train_sample == "1")
inclusion_test <-inclusion_model_df %>% filter(test == "1")
```

```
dim(inclusion_train)
```

```
[1] 25682    26
```

```
dim(inclusion_train_sample)
```

```
[1] 12840    26
```

```
dim(inclusion_test)
```

```
[1] 6352   26
```

2.3.2 Risk Exclusion Variables

The XGBoost model can handle the NA variables so I leave them as is. The logistic regression model cannot manage the expanse of missing variables in the risk exclusion dataset, so I convert all NA to zero for the logistic regression model.

```
#order stroke as factor
exclusion_model_df <- exclusion_df2 %>%  mutate(stroke = as.factor(stroke))

#XGBoost Model
exclusion_train <- exclusion_model_df %>% filter(test == "0")
exclusion_train_sample <- exclusion_model_df %>% filter(train_sample == "1")
exclusion_test <-exclusion_model_df %>% filter(test == "1")

#Logistic Regression Model
```

```r
exclusion_model_df_lr <- exclusion_model_df %>%
  mutate(across(everything(), ~replace(., is.na(.), 0)))
exclusion_train_lr <- exclusion_model_df_lr %>% filter(test == "0")
exclusion_train_sample_lr <- exclusion_model_df_lr %>% filter(train_sample == "1")
exclusion_test_lr <-exclusion_model_df_lr %>% filter(test == "1")
```

```r
dim(exclusion_train)
```

```
[1] 25682   682
```

```r
dim(exclusion_train_sample)
```

```
[1] 12840   682
```

```r
dim(exclusion_test)
```

```
[1] 6352  682
```

2.4 Numerical and Visual Summary

The stroke risk inclusion data has 24 columns, and 32,034 rows. The first column, "ptnum" represents the id for each individual, and "stroke" represents the outcome: stroke or no stroke. The 22 additional columns are the variables that will be used for prediction.

The following exploratory analysis is performed on the full training data of the risk inclusion dataset (25626).

```r
inclusion_train %>% select(! c(ptnum, test, train_sample, stroke)) %>% skim()
```

Table 1: Data summary

Name	Piped data
Number of rows	25682
Number of columns	22
_____	
Column type frequency:	
character	4
numeric	18
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
age	0	1	5	5	0	2	0
gender	0	1	1	1	0	2	0
race	0	1	5	8	0	6	0
ethnic	0	1	8	11	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
missing_count	0	1	100.04	7.81	65	96.00	100.00	102.00	118.00	
hbp	0	1	1.05	1.47	0	0.00	0.00	2.00	8.00	
hbp2	0	1	0.05	0.11	0	0.00	0.00	0.00	0.75	
hd	0	1	0.62	1.32	0	0.00	0.00	1.00	10.00	
hd2	0	1	0.23	0.29	0	0.00	0.25	0.25	2.00	
d	0	1	0.87	1.30	0	0.00	1.00	1.00	11.00	
d2	0	1	0.08	0.12	0	0.00	0.00	0.25	0.75	
s	0	1	0.38	0.56	0	0.00	0.00	1.00	3.00	
bco	0	1	0.30	0.77	0	0.00	0.00	0.00	6.00	
hrbcc	0	1	0.28	0.55	0	0.00	0.00	0.00	3.00	
hrbcc2	0	1	0.08	0.12	0	0.00	0.00	0.25	0.50	
hbcal	0	1	1.65	1.23	0	1.00	1.00	2.00	8.00	
o2	0	1	0.21	0.09	0	0.25	0.25	0.25	0.25	
a	0	1	0.16	0.37	0	0.00	0.00	0.00	2.00	
dm	0	1	0.11	0.32	0	0.00	0.00	0.00	3.00	
ahr	0	1	0.29	0.97	0	0.00	0.00	0.00	7.00	
csa	0	1	0.06	0.23	0	0.00	0.00	0.00	2.00	
ps2	0	1	0.01	0.05	0	0.00	0.00	0.00	0.75	

The predictors present in the stroke risk inclusion data include:

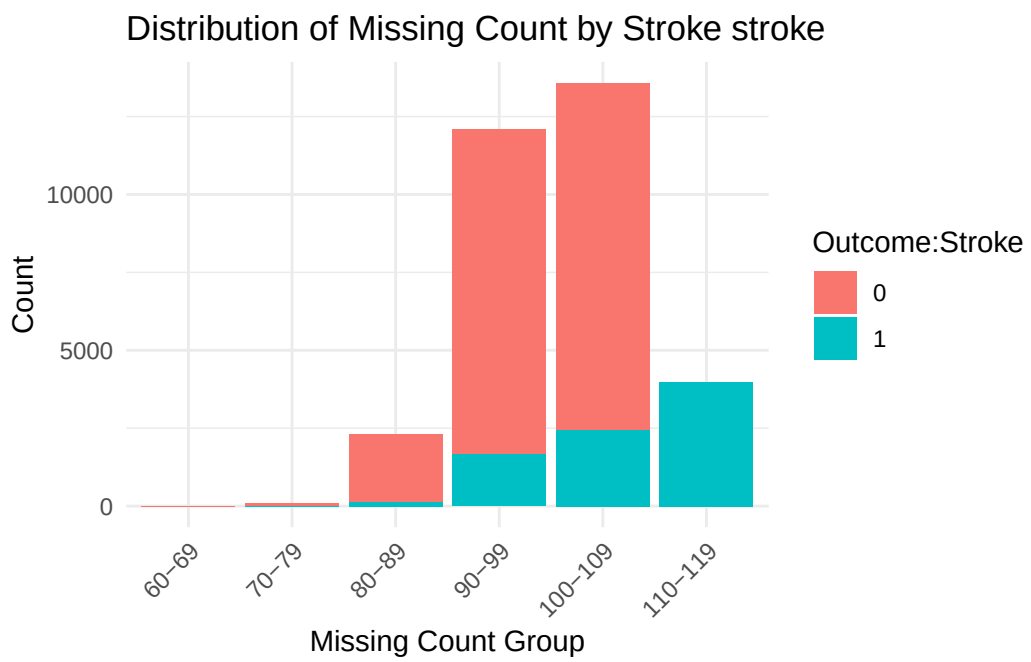
- Missing Data Count (continuous)
- Medical Data (continuous)
- Demographics (categorical)

2.4.1 Missing Data Count

Checking the distribution of the new missing_count variable

```
# Group missing_count into bins of 10
filtered_stroke_risk3 <- stroke_risk_inclusion_data %>%
  mutate(missing_count_group = case_when(
    missing_count >= 60 & missing_count <= 69 ~ "60-69",
    missing_count >= 70 & missing_count <= 79 ~ "70-79",
    missing_count >= 80 & missing_count <= 89 ~ "80-89",
```

```
    missing_count >= 90 & missing_count <= 99 ~ "90-99",
    missing_count >= 100 & missing_count <= 109 ~ "100-109",
    missing_count >= 110 & missing_count <= 119 ~ "110-119")) %>%
    mutate(missing_count_group = factor(missing_count_group,
          levels = c("60-69", "70-79", "80-89", "90-99",
                     "100-109", "110-119")))

missing_count_stroke <- filtered_stroke_risk3 %>%
  group_by(stroke, missing_count_group) %>%
  summarise(count = n(), .groups = "drop")

ggplot(missing_count_stroke, aes(x = missing_count_group, y = count,
                                 fill = factor(stroke))) +
  geom_bar(stat = "identity", position = "stack") +
  labs(title = "Distribution of Missing Count by Stroke stroke",
       x = "Missing Count Group",
       y = "Count",
       fill = "Outcome:Stroke") +
  theme_minimal() +
  scale_color_brewer(palette = "Set1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
summary_stats <- stroke_risk_inclusion_data %>%
  group_by(stroke) %>%
  summarise(Q1 = quantile(missing_count, 0.25, na.rm = TRUE),
            Q2 = quantile(missing_count, 0.5, na.rm = TRUE),
            Q3 = quantile(missing_count, 0.75, na.rm = TRUE),
```

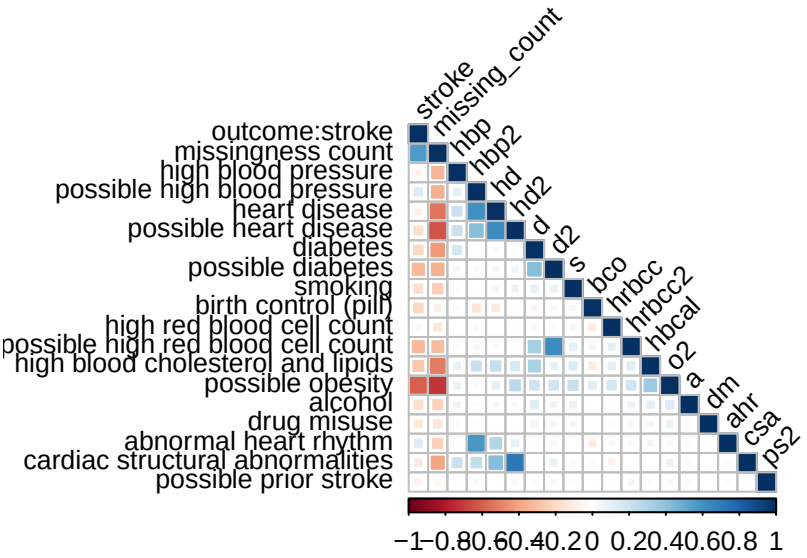
```
            avg = mean(missing_count, na.rm = TRUE),
            .groups = "drop")
print(summary_stats)
```

```
# A tibble: 2 x 5
  stroke    Q1    Q2    Q3   avg
   <dbl> <dbl> <dbl> <dbl> <dbl>
1      0    95    99   101  97.6
2      1   100   108   117 108.
```

This graph highlights a potential issue with the data. The outcome:stroke group is represented heavily at the tail end of the graph and not present in the first portion of the data. The table tells us that the outcome:stroke group has the highest average missingness count in each quarter. I believe the missingness has to do with how the original authors decided to collect data. The EHR for individuals with outcome:no stroke were actually the average of that persons data over time, whereas the EHR for outcome: stroke were the most recent medical data only. There's not much I can do with this information other than to say this variable would not be repeatable, but could be explored by researchers performing similar research. I will still use this variable for my analysis, as I don't have a clean fix for this issue at this time, but if I were to create my own dataset from EHRs I may choose to handle the avg record vs single date record issue differently.

2.4.2 Correlation Matrix

```
inclusion_train_eda <- inclusion_train %>%
  mutate(stroke = as.numeric(as.character(stroke))) %>%
  select(!c(test, train_sample))

numeric_columns <- inclusion_train_eda[, sapply(inclusion_train_eda, is.numeric)]
cor_matrix <- cor(numeric_columns, use = "complete.obs")  # Replace column names
#colnames(cor_matrix) <- column_strokes[colnames(cor_matrix)]
rownames(cor_matrix) <- column_strokes[rownames(cor_matrix)]

# Now plot using corrplot
library(corrplot)
# Plot only the lower triangle
corrplot(cor_matrix,
         method = "square",
         tl.col = "black",
         tl.cex = 0.8,
         tl.srt = 45,
         type= "lower")
```



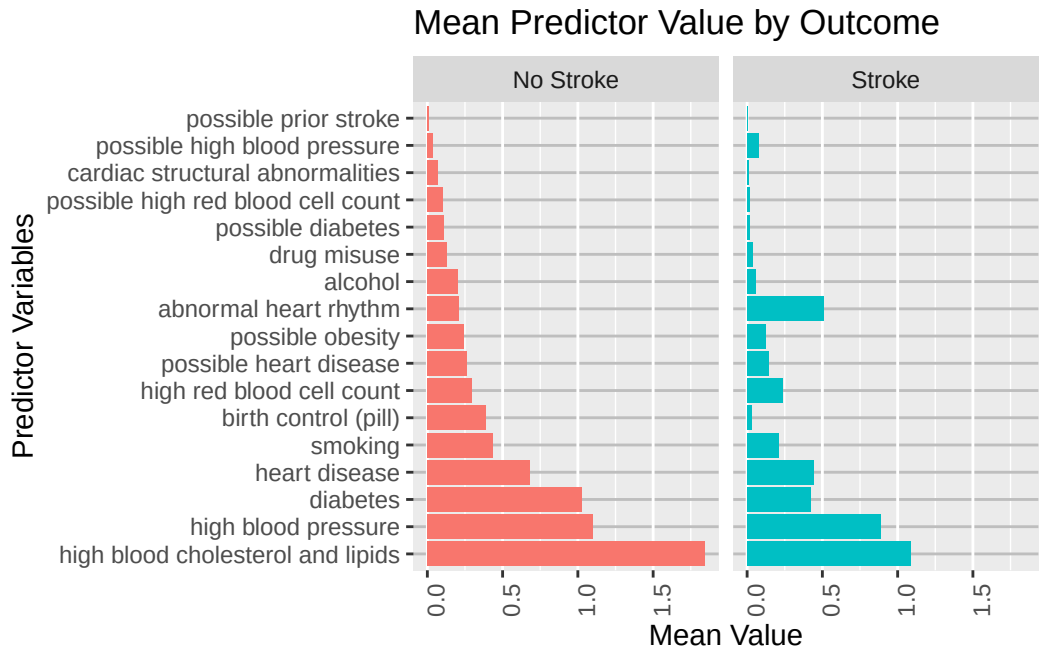
On the left side of this correlation are the variable names and on the top of the graph are the shorthand or abbreviations that are used throughout the data analysis. The first variable: outcome:stroke

Here we can see that missing_count has high negative correlation with multiple variables. What that means is that those variables are likely more 'complete' than other variables.

2.4.3 Medical Variables

```
# Ensure the 'stroke' is numeric
inclusion_train_eda <- inclusion_train_eda %>%
  mutate(stroke = as.numeric(as.character(stroke)))

# Convert data to long format
data_long <- inclusion_train_eda %>%
  select(where(is.numeric)) %>%
  pivot_longer(cols = -stroke, names_to = "variable", values_to = "value")

# Calculate the mean for each numeric variable grouped by 'stroke'
mean_values <- data_long %>%
  group_by(stroke, variable) %>%
  summarize(mean_value = mean(value, na.rm = TRUE))

mean_values <- mean_values %>%
  filter(variable != "missing_count") %>%
  mutate(variable_stroke = column_strokes[variable],
         variable_stroke = fct_reorder(variable_stroke, mean_value, .desc = TRUE))
```

```
mean_values$stroke <- factor(mean_values$stroke, levels = c(0, 1),
                             labels = c("No Stroke", "Stroke"))

ggplot(mean_values, aes(x = variable_stroke, y = mean_value,
                        fill = factor(stroke))) +
  geom_bar(stat = "identity", position = "dodge", show.legend = FALSE) +
  labs(x = "Predictor Variables",
       y = "Mean Value",
       fill = "stroke (No Stroke, Stroke)",
       title = "Mean Predictor Value by Outcome") +
  scale_color_brewer(palette = "Set1") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
       panel.grid.major.y = element_line(color = "gray", linetype = "solid")) +
  coord_flip() + facet_wrap(~ stroke)
```

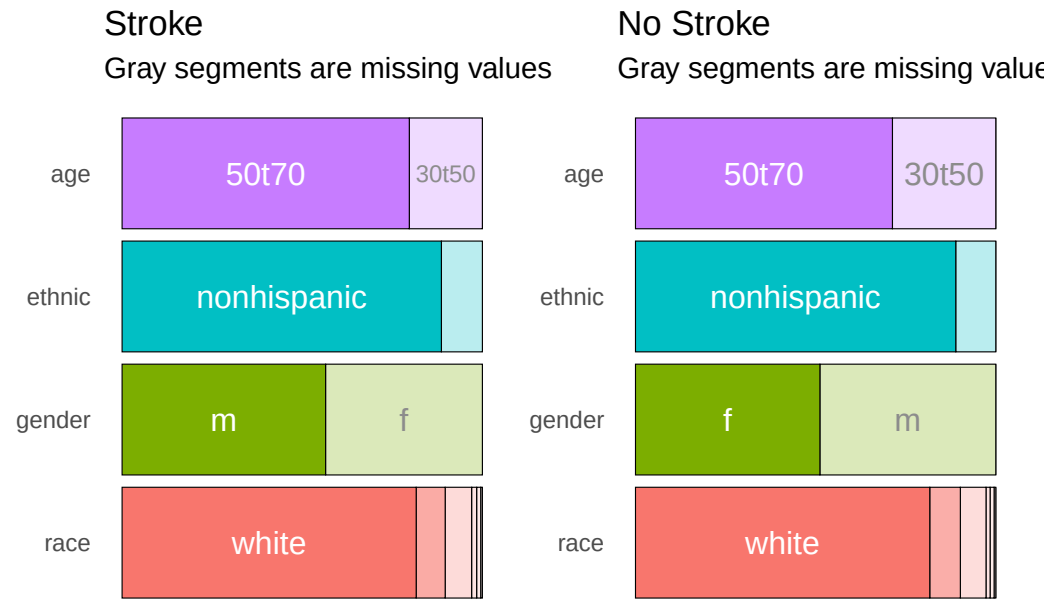


On the X axis, zero represents no instances of the variable in an persons medical history, 1 represents one instance, and so on. This graph shows the average score each individual has within the training dataset.

On average, individuals without an instance of stroke have a higher instance of heart disease than individuals with stroke. ### Demographic Data

```
df_stroke <- inclusion_train_eda %>% filter(stroke == 1) %>%
  select(!ptnum) %>% inspect_cat()
plot_stroke <- df_stroke %>% show_plot() +
  labs(title = "Stroke")

df_no_stroke <- inclusion_train_eda %>% filter(stroke == 0) %>%
```

```
  select(!ptnum) %>% inspect_cat()
plot_no_stroke <- df_no_stroke %>% show_plot() +
  labs(title = "No Stroke")

plot_stroke + plot_no_stroke
```



Two demographics that are presenting with difference between stroke and non-stroke outcome are age and gender. This is exactly what I expected to see in the data. Stroke is more prevalent in older individuals. Stroke is also more likely to affect men than women, although, women are more likely to die from stroke than men.

3 Evaluation Metric

My outcome variable is binary, therefore I will need to look at accuracy, recall, precision, or f1 as my primary indicator of model fit

I tune my models primarily on f1 score but pay attention to recall when choosing a best model overall.

Predicting a medical outcome requires a balance of focus on recall and accuracy.

4 Models

The three models I chose for my analysis are: 1. Logistic Regression 2. Support Vector Machines 3. XGBoost (Extreme Gradient Boosting)

These models were chosen for their ability to handle large, sparse datasets and for their distinct methodological approaches. Logistic Regression serves as a robust baseline for comparison and is well-suited for binary classification tasks. Support Vector Machines are known for their ability to perform well with high-dimensional data and complex decision boundaries. XGBoost, a gradient-boosted decision tree algorithm, has been frequently highlighted in the literature as achieving the best performance in stroke prediction tasks. For these reasons and for the opportunity to observe these methods myself, I set out to predict stroke using these three models.

4.1 Risk Inclusion Logistic Regression

4.1.1 Recipe

Many of the elements from this recipe are repeated throughout the other recipes. I assign the role of outcome to the column "stroke and the role of id to the column"ptnum" I remove the two columns "test" and "train_sample" which are columns noting if an individual is part of the test set, training set, or training sample set. We do not need them in analysis.

I add step_novel to all factor_predictors so that if there is a new variable in the test set unseen by the training data, the model can adjust to add that variable in.

I add step_dummy to all factors so that each factor variable is represented as 1 or zero

I add step_impute_mode for those factor variables that are now binary. I choose impute mode because I don't know well enough the data to trust another form of mutation. I would explore other options here if I were to work on this data again.

I add step_zv to all predictors which removes entire variables if there is no variance (meaning all data is assigned one value)

I add step_corr to all numeric predictors with a threshold of 0.8 to reduce multicollinearity in the dataset

I add step_scale to numeric predictors so that there is a sense of cohesion across the dataset. This would balance the binary variables with the continuous variables so that there wouldn't be an imbalance in how the model was understanding the data's importance.

I could use step_normalize here but I want to retain interpretability on an increase in a variable. Step_normalize would reduce stroke risk variables interpretability. This interpretability is particularly important in the medical field.

```
# Logistic Regression recipe
recipe_lr <- recipe(stroke ~ ., data = inclusion_train_sample) %>%
  update_role(stroke, new_role = "outcome") %>%
  update_role(ptnum, new_role = "id") %>%
  step_rm(test, train_sample) %>%
  step_novel(all_factor_predictors(), -all_outcomes()) %>%
  step_dummy(all_factor_predictors(), -all_outcomes(), one_hot = TRUE) %>%
  step_impute_mode(all_factor_predictors()) %>%
  step_zv(all_predictors()) %>%
```

```
  step_corr(all_numeric_predictors(), threshold = 0.8) %>%
  step_scale(all_numeric_predictors())
```

```
# Logistic Regression Workflow
logistic_spec <- logistic_reg() %>%
  set_engine("glm", family = binomial(link = "logit")) %>%
  set_mode("classification")

set.seed(123)
validation_split <- vfold_cv(inclusion_train_sample,
                             v = 5, repeats = 3, strata = "stroke")

logistic_workflow <- workflow() %>%
  add_model(logistic_spec) %>%
  add_recipe(recipe_lr)

f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")

plan(multisession)
#Logistic Regression Validation Fit
logistic_results <- fit_resamples(
  logistic_workflow,
  resamples = validation_split,
  metrics = metric_set(f_meas_sec_level))
plan(sequential)
```

```
# Extract best hyper parameters based on RMSE
best_logistic_results <- select_best(logistic_results, metric = "f_meas_sec_level")

best_f_meas_lr <- logistic_results %>%
  collect_metrics() %>%
  filter(.config == best_logistic_results$.config,
         .metric == "f_meas_sec_level")  %>% pull(mean)
best_f_meas_lr
```

```
[1] 0.789049
```

4.1.2 Fit Model and Predict

```
# Finalize model
final_logistic_workflow <- finalize_workflow(logistic_workflow,
  best_logistic_results)
# Fit final model on  training data
```

```
final_logistic_fit <- final_logistic_workflow %>% fit(data = inclusion_train)
# predict test data
aug_lr <- augment(final_logistic_fit, inclusion_test)
```

4.1.3 Performance

In order to maximize recall, I can use the "best" coordinates from the ROC curve.

```
roc_curve_lr <- roc(aug_lr$stroke, aug_lr$.pred_1)
roc_coords_lr <- coords(roc_curve_lr, "best")
roc_coords_lr
```

```
  threshold specificity sensitivity
1 0.3373084   0.8594276    0.889375
```

```
table(aug_lr$.pred_class, aug_lr$stroke)
```

```
      0    1
  0 4426  323
  1  326 1277
```

```
aug_lr_adj <- aug_lr %>%
  mutate(
    .pred_class = if_else(.pred_1 > roc_coords_lr$threshold, "1", "0"))
table(aug_lr_adj$.pred_class, aug_lr_adj$stroke)
```

```
      0    1
  0 4084  177
  1  668 1423
```

```
# Confusion matrix to compute metrics
conf_matrix_lr <- confusionMatrix(as.factor(aug_lr_adj$.pred_class),
                                  as.factor(aug_lr_adj$stroke),
                                  positive = "1")
Inclusion_Logistic_Regression_Model <- tibble(
    Model = "Inclusion Logistic Regression",
    ACCURACY = conf_matrix_lr$overall['Accuracy'],
    RECALL = conf_matrix_lr$byClass['Recall'],
    PRECISION = conf_matrix_lr$byClass['Precision'],
    F1 = conf_matrix_lr$byClass['F1']
    )
```

```
kbl(Inclusion_Logistic_Regression_Model %>%
      mutate(across(where(is.numeric), ~ round(., 4))),
    caption = "Logistic Regression Model Prediction Measures")  %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))
```

Table 4: Logistic Regression Model Prediction Measures

Model	ACCURACY	RECALL	PRECISION	F1
Inclusion Logistic Regression	0.867	0.8894	0.6805	0.7711

4.2 Risk Inclusion XG Boost

4.2.1 Recipe

The only difference with the XGBoost recipe is that it does not require an imputation on missing values.

```
xgb_rec <- recipe(stroke ~ ., data = inclusion_train_sample) %>%
  update_role(stroke, new_role = "outcome") %>%
  update_role(ptnum, new_role = "id") %>%
  step_rm(test, train_sample) %>%
  step_novel(all_factor_predictors(), -all_outcomes()) %>%
  step_dummy(all_factor_predictors(), -all_outcomes(), one_hot = TRUE) %>%
  step_zv(all_predictors()) %>%
  step_corr(all_numeric_predictors(), threshold = 0.8)
```

XGB Set Up

```
set.seed(8935)
validation_split <- vfold_cv(inclusion_train_sample,
                             v = 5, repeats = 3, strata = "stroke")
xgb_spec <- boost_tree(
    trees = tune(),               # Number of trees
    tree_depth = tune(),          # Maximum depth of trees
    learn_rate = tune(),          # Learning rate
    loss_reduction = tune(),      # Minimization of loss
    sample_size = tune(),         # Proportion of training data to sample
    min_n = tune()                # Minimum number of data points in a node
  ) %>%
  set_engine("xgboost", event_level = "second") %>%
  set_mode("classification")
#XGB Workflow
xgb_workflow <- workflow() %>%
  add_recipe(xgb_rec) %>%
  add_model(xgb_spec)
```

```
#XGB Grid
xgb_grid <- grid_random(
  trees(c(10, 50)),          # Range for number of trees
  tree_depth(c(1, 5)),       # Range for maximum depth
  learn_rate(c(0.01, 0.1)), # Range for learning rate
  loss_reduction(c(0, 5)),  # Range for loss reduction
  sample_size(c(0,1)),       # Use integer values
  min_n(c(1, 5)),            # Range for minimum number of data points in a node
  size = 15)                 # Number of combinations to try

f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")
#XGB Tune_Grid
plan(multisession)
xgb_tune_results <- tune_grid(
  xgb_workflow,
  resamples = validation_split,
  grid = xgb_grid,
  metrics = metric_set(f_meas_sec_level),
  control = tune::control_grid(verbose = TRUE))

plan(sequential)

tune_results <- xgb_tune_results %>%
  collect_metrics()
```

```
# Select the best hyperparameters
best_xgb_params <- select_best(xgb_tune_results, metric = "f_meas_sec_level")
best_f_meas_xgb <- xgb_tune_results %>%
  collect_metrics() %>%
  filter(.config == best_xgb_params$.config,
         .metric == "f_meas_sec_level") %>% pull(mean)
best_f_meas_xgb
```

4.2.2 Fit Model and Predict

```
# Finalize the workflow with best parameters
final_xgb_workflow <- finalize_workflow(xgb_workflow, best_xgb_params)
# Fit the final model on the training data
final_xgb_fit <- fit(final_xgb_workflow, data = inclusion_train)
# predict test data
aug_stroke_xgb <- augment(final_xgb_fit, inclusion_test)
```

4.2.3 Performance

In order to maximize recall, I can use the "best" coordinates from the ROC curve.

```
roc_curve_xgb <- roc(aug_stroke_xgb$stroke, aug_stroke_xgb$.pred_1)
# Get the threshold that maximizes recall
roc_coords_xgb <- coords(roc_curve_xgb, "best")
roc_coords_xgb
```

```
  threshold specificity sensitivity
1 0.4607417     0.90383    0.876875
```

```
table(aug_stroke_xgb$.pred_class, aug_stroke_xgb$stroke)
```

```
      0    1
  0 4367  234
  1  385 1366
```

```
aug_stroke_xgb_adj <- aug_stroke_xgb %>%
  mutate(.pred_class = if_else(.pred_1 > roc_coords_xgb$threshold, "1", "0"))
table(aug_stroke_xgb_adj$.pred_class, aug_stroke_xgb_adj$stroke)
```

```
      0    1
  0 4295  197
  1  457 1403
```

```
# Confusion matrix to compute metrics
conf_matrix_xgb <- confusionMatrix(as.factor(aug_stroke_xgb_adj$.pred_class),
                                   as.factor(aug_stroke_xgb_adj$stroke),
                                   positive = "1")
Inclusion_XGBoost_Model <- tibble(
    Model = "Inclusion XGBoost",
    ACCURACY = conf_matrix_xgb$overall['Accuracy'],
    RECALL = conf_matrix_xgb$byClass['Recall'],
    PRECISION = conf_matrix_xgb$byClass['Precision'],
    F1 = conf_matrix_xgb$byClass['F1'])
```

```
kbl(Inclusion_XGBoost_Model %>% mutate(across(where(is.numeric), ~ round(., 4))),
    caption = "XGBoost Model Prediction Measures")  %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))
```

Table 5: XGBoost Model Prediction Measures

Model	ACCURACY	RECALL	PRECISION	F1
Inclusion XGBoost	0.897	0.8769	0.7543	0.811

4.3 Risk Inclusion Supervised Learning Model

4.3.1 Recipe

This recipe is the same as is used in the Logistic Regression.

```
# SVM recipe
recipe_svm <- recipe(stroke ~ ., data = inclusion_train_sample) %>%
  update_role(stroke, new_role = "outcome") %>%
  update_role(ptnum, new_role = "id") %>%
  step_rm(test, train_sample) %>%
  step_novel(all_factor_predictors(), -all_outcomes()) %>%
  step_dummy(all_factor_predictors(), -all_outcomes(), one_hot = TRUE) %>%
  step_impute_mode(all_factor_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_corr(all_numeric_predictors(), threshold = 0.8) %>%
  step_scale(all_numeric_predictors())
```

```
# SVM Workflow
svm_linear_spec <- svm_linear() %>%
  set_engine("kernlab") %>%
  set_mode("classification")

set.seed(123)
validation_split <- vfold_cv(inclusion_train_sample, v = 5,
                             repeats = 3, strata = "stroke")

svm_workflow <- workflow() %>%
  add_model(svm_linear_spec) %>%
  add_recipe(recipe_svm)
f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")

# SVM Validation Fit
plan(multisession)

svm_results <- fit_resamples(
  svm_workflow,
  resamples = validation_split,
  metrics = metric_set(f_meas_sec_level))
```

```
plan(sequential)
best_svm_results <- select_best(svm_results, metric = "f_meas_sec_level")
```

4.3.2 Fit Model and Predict

```
best_f_meas_svm <- svm_results %>%
  collect_metrics() %>%
  filter(.config == best_svm_results$.config, .metric == "f_meas_sec_level") %>%
  pull(mean)
best_f_meas_svm
```

```
[1] 0.7788116
```

```
# SVM Fit Model and Predict
final_svm_workflow <- finalize_workflow(svm_workflow, best_svm_results)
# Fit final model on  training data
final_svm_fit <- final_svm_workflow %>% fit(data = inclusion_train)
```

```
 Setting default kernel parameters
```

```
# predict test data
aug_svm <- augment(final_svm_fit, inclusion_test)
```

4.3.3 Performance

In order to maximize recall, I can use the "best" coordinates from the ROC curve.

```
roc_curve_svm <- roc(aug_svm$stroke, aug_svm$.pred_1)
roc_coords_svm <- coords(roc_curve_svm, "best")
roc_coords_svm
```

```
  threshold specificity sensitivity
1 0.3377645   0.8773148     0.86375
```

```
table(aug_svm$.pred_class, aug_svm$stroke)
```

```
      0    1
  0 4590  450
  1  162 1150
```

```
aug_svm_adj <- aug_svm %>%
  mutate(.pred_class = if_else(.pred_1 > roc_coords_svm$threshold, "1", "0"))
table(aug_svm_adj$.pred_class, aug_svm_adj$stroke)
```

```
     0    1
  0 4169  218
  1  583 1382
```

```
conf_matrix_svm <- confusionMatrix(as.factor(aug_svm_adj$.pred_class),
                as.factor(aug_svm$stroke),
                positive = "1")
Inclusion_SVM_model <- tibble(
    Model = "Inclusion SVM",
    ACCURACY = conf_matrix_svm$overall['Accuracy'],
    RECALL = conf_matrix_svm$byClass['Recall'],
    PRECISION = conf_matrix_svm$byClass['Precision'],
    F1 = conf_matrix_svm$byClass['F1']
    )
```

```
kbl(Inclusion_SVM_model %>% mutate(across(where(is.numeric), ~round(.,4))),
    caption ="SVM Model Prediction Measures") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))
```

Table 6: SVM Model Prediction Measures

Model	ACCURACY	RECALL	PRECISION	F1
Inclusion SVM	0.8739	0.8638	0.7033	0.7753

4.4 Risk Exclusion Logistic Regression

4.4.1 Recipe

```
# Logistic Regression recipe
recipe_lr_nr <- recipe(stroke ~ ., data = exclusion_train_sample) %>%
  update_role(stroke, new_role = "outcome") %>%
  update_role(ptnum, new_role = "id") %>%
  step_rm(test, train_sample) %>%
  step_novel(all_factor_predictors(), -all_outcomes()) %>%
  step_dummy(all_factor_predictors(), -all_outcomes(), one_hot = TRUE) %>%
  step_impute_mode(all_factor_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_corr(all_numeric_predictors(), threshold = 0.8) %>%
  step_scale(all_numeric_predictors())
```

```
# Logistic Regression Workflow
logistic_spec_nr <- logistic_reg() %>%
  set_engine("glm", family = binomial(link = "logit")) %>%
  set_mode("classification")

set.seed(123)
validation_split_nr <- vfold_cv(exclusion_train_sample_lr,
                                v = 3, repeats = 2, strata = "stroke")

logistic_workflow_nr <- workflow() %>%
  add_model(logistic_spec_nr) %>%
  add_recipe(recipe_lr_nr)

f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")

plan(multisession)
options(future.globals.maxSize = 1e9)

#Logistic Regression Validation Fit
logistic_results_nr <- fit_resamples(
  logistic_workflow_nr,
  resamples = validation_split_nr,
  metrics = metric_set(f_meas_sec_level))
plan(sequential)
```

```
# Extract best hyper parameters based on RMSE
best_logistic_results_nr <- select_best(logistic_results_nr,
                                        metric = "f_meas_sec_level")
best_logistic_results_nr
```

```
# A tibble: 1 x 1
  .config
  <chr>
1 Preprocessor1_Model1
```

4.4.2 Fit Model and Predict

```
# Finalize model
final_logistic_workflow_nr <- finalize_workflow(logistic_workflow_nr,
  best_logistic_results_nr)
# Fit final model on  training data
final_logistic_fit_nr <- final_logistic_workflow_nr %>%
  fit(data = exclusion_train_lr)
```

```
# predict test data
aug_lr_nr <- augment(final_logistic_fit_nr, exclusion_test_lr)
```

4.4.3 Performance

In order to maximize recall, I can use the "best" coordinates from the ROC curve.

```
roc_curve_lr_nr <- roc(aug_lr_nr$stroke, aug_lr_nr$.pred_1)
roc_coords_lr_nr <- coords(roc_curve_lr_nr, "best")
roc_coords_lr_nr
```

```
  threshold specificity sensitivity
1       0.5   0.9701178     0.90125
```

```
table(aug_lr_nr$.pred_class, aug_lr_nr$stroke)
```

```
      0    1
  0 4610  158
  1  142 1442
```

```
aug_lr_adj_nr <- aug_lr_nr %>%
  mutate(
    .pred_class = if_else(.pred_1 > roc_coords_lr_nr$threshold, "1", "0"))
table(aug_lr_adj_nr$.pred_class, aug_lr_adj_nr$stroke)
```

```
      0    1
  0 4610  158
  1  142 1442
```

```
# Confusion matrix to compute metrics
conf_matrix_lr_nr <- confusionMatrix(as.factor(aug_lr_adj_nr$.pred_class),
                                     as.factor(aug_lr_adj_nr$stroke),
                                     positive = "1")
Exclusion_Logistic_Regression_Model <- tibble(
    Model = "Exclusion Logistic Regression",
    ACCURACY = conf_matrix_lr_nr$overall['Accuracy'],
    RECALL = conf_matrix_lr_nr$byClass['Recall'],
    PRECISION = conf_matrix_lr_nr$byClass['Precision'],
    F1 = conf_matrix_lr_nr$byClass['F1']
    )
```

```
kbl(Exclusion_Logistic_Regression_Model %>%
      mutate(across(where(is.numeric), ~round(.,4))),
    caption ="SVM Model Prediction Measures") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))
```

Table 7: SVM Model Prediction Measures

Model	ACCURACY	RECALL	PRECISION	F1
Exclusion Logistic Regression	0.9528	0.9012	0.9104	0.9058

This model performs incredibly well. Shockingly well really. Suspiciously well, considering the data does not include stroke risk data.

4.5 Risk Exclusion XGBoost

The specifications for the Exclusion XGBoost model is the same as the Inclusion XGBoost model. ### Recipe

```
xgb_rec_nr <- recipe(stroke ~ ., data = exclusion_train_sample) %>%
  update_role(stroke, new_role = "outcome") %>%
  update_role(ptnum, new_role = "id") %>%
  step_rm(test, train_sample) %>%
  step_novel(all_factor_predictors(), -all_outcomes()) %>%
  step_unknown(all_factor_predictors()) %>%
  step_dummy(all_factor_predictors(), -all_outcomes(), one_hot = TRUE) %>%
  step_unknown(all_factor_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_corr(all_predictors(), threshold = 0.8)
```

XGB Set Up

```
set.seed(8935)
validation_split_nr <- vfold_cv(exclusion_train_sample,
                                v = 5, repeats = 3, strata = "stroke")
xgb_spec_nr <- boost_tree(
    trees = tune(),               # Number of trees
    tree_depth = tune(),          # Maximum depth of trees
    learn_rate = tune(),          # Learning rate
    loss_reduction = tune(),      # Minimization of loss
    sample_size = tune(),         # Proportion of training data to sample
    min_n = tune()                # Minimum number of data points in a node
  ) %>%
  set_engine("xgboost", event_level = "second") %>%
  set_mode("classification")
```

```
#XGB Workflow
xgb_workflow_nr <- workflow() %>%
  add_recipe(xgb_rec_nr) %>%
  add_model(xgb_spec_nr)

#XGB Grid
xgb_grid_nr <- grid_random(
  trees(c(10, 50)),          # Range for number of trees
  tree_depth(c(1, 5)),       # Range for maximum depth
  learn_rate(c(0.01, 0.1)), # Range for learning rate
  loss_reduction(c(0, 5)),  # Range for loss reduction
  sample_size(c(0,1)),       # Use integer values
  min_n(c(1, 5)),            # Range for minimum number of data points in a node
  size = 15)                 # Number of combinations to try


f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")
#XGB Tune_Grid
plan(multisession)
options(future.globals.maxSize = 1e9)
xgb_tune_results_nr <- tune_grid(
  xgb_workflow_nr,
  resamples = validation_split_nr,
  grid = xgb_grid_nr,
  metrics = metric_set(f_meas_sec_level),
  control = tune::control_grid(verbose = TRUE))

plan(sequential)

tune_results_nr <- xgb_tune_results_nr %>%
  collect_metrics()
```

4.5.1 Fit Model and Predict

```
# Select the best hyperparameters
best_xgb_params_nr <- select_best(xgb_tune_results_nr, metric = "f_meas_sec_level")
# Finalize the workflow with best parameters
final_xgb_workflow_nr <- finalize_workflow(xgb_workflow_nr, best_xgb_params_nr)
# Fit the final model on the training data
final_xgb_fit_nr <- fit(final_xgb_workflow_nr, data = exclusion_train)
# predict test data
aug_stroke_xgb_nr <- augment(final_xgb_fit_nr, exclusion_test)
```

4.5.2 Performance

In order to maximize recall, I can use the "best" coordinates from the ROC curve.

```
roc_curve_xgb_nr <- roc(aug_stroke_xgb_nr$stroke, aug_stroke_xgb_nr$.pred_1)
# Get the threshold that maximizes recall
roc_coords_xgb_nr <- coords(roc_curve_xgb_nr, "best")
roc_coords_xgb_nr
```

```
  threshold specificity sensitivity
1 0.2675602   0.9343434     0.74375
```

```
table(aug_stroke_xgb_nr$.pred_class, aug_stroke_xgb_nr$stroke)
```

```
   0    1
0 4674  539
1   78 1061
```

```
aug_stroke_xgb_adj_nr <- aug_stroke_xgb_nr %>%
  mutate(.pred_class = if_else(.pred_1 > roc_coords_xgb_nr$threshold, "1", "0"))
table(aug_stroke_xgb_adj_nr$.pred_class, aug_stroke_xgb_adj_nr$stroke)
```

```
   0    1
0 4440  410
1  312 1190
```

```
# Confusion matrix to compute metrics
conf_matrix_xgb_nr <- confusionMatrix(as.factor(aug_stroke_xgb_adj_nr$.pred_class),
                                      as.factor(aug_stroke_xgb_adj_nr$stroke),
                                      positive = "1")
Exclusion_XGBoost_Model <- tibble(
    Model = "Exclusion XGBoost",
    ACCURACY = conf_matrix_xgb_nr$overall['Accuracy'],
    RECALL = conf_matrix_xgb_nr$byClass['Recall'],
    PRECISION = conf_matrix_xgb_nr$byClass['Precision'],
    F1 = conf_matrix_xgb_nr$byClass['F1'])
```

```
kbl(Exclusion_XGBoost_Model %>% mutate(across(where(is.numeric), ~ round(., 4))),
    caption = "XGBoost Exclusion Model Prediction Measures")  %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))
```

Table 8: XGBoost Exclusion Model Prediction Measures

Model	ACCURACY	RECALL	PRECISION	F1
Exclusion XGBoost	0.8863	0.7438	0.7923	0.7672

5 Compare Models

5.1 Model Metrics

```
Combined_Models <- rbind(Inclusion_Logistic_Regression_Model,
                         Inclusion_XGBoost_Model,
                         Inclusion_SVM_model,
                         Exclusion_Logistic_Regression_Model,
                         Exclusion_XGBoost_Model)
```

```
kbl(Combined_Models %>% mutate(across(where(is.numeric), ~round(.,4))),
    caption ="Model Prediction Measures") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))
```

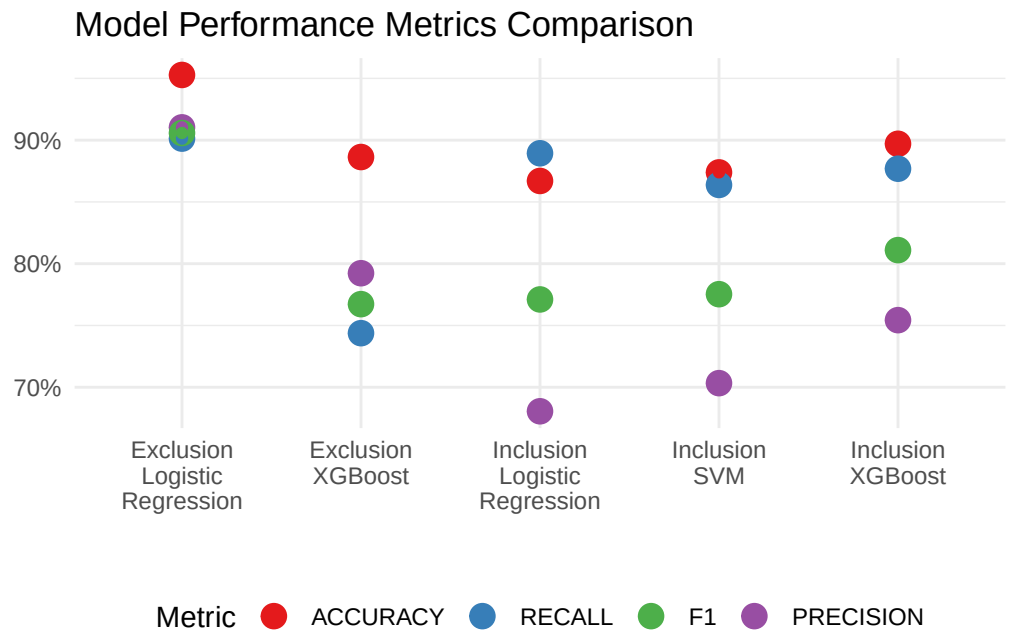
Table 9: Model Prediction Measures

Model	ACCURACY	RECALL	PRECISION	F1
Inclusion Logistic Regression	0.8670	0.8894	0.6805	0.7711
Inclusion XGBoost	0.8970	0.8769	0.7543	0.8110
Inclusion SVM	0.8739	0.8638	0.7033	0.7753
Exclusion Logistic Regression	0.9528	0.9012	0.9104	0.9058
Exclusion XGBoost	0.8863	0.7438	0.7923	0.7672

- The Exclusion Logistic Regression model outperforms all models across all metrics.
- The Inclusion XGBoost model has the second highest F1 score (0.8110).

```
# Pivot longer and assign level so it looks better in graph
Combined_Models_long <- Combined_Models %>%
  pivot_longer(cols = -Model, names_to = "Metric", values_to = "Value")%>%
  mutate(Metric = factor(Metric,
                         levels = c("ACCURACY", "RECALL", "F1", "PRECISION")))

# Modify Model column to add line breaks for two-line labeling
Combined_Models_long$Model <- gsub(" ", "\n", Combined_Models_long$Model)

ggplot(Combined_Models_long, aes(x = Model, y = Value, color = Metric)) +
  geom_point(size = 4) +
  geom_point(aes(group = Metric)) +
  theme_minimal() +
  labs(title = "Model Performance Metrics Comparison",
```

```
       x = "",
       y = "",
       color = "Metric") +
  scale_color_brewer(palette = "Set1") +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  theme(axis.text.x = element_text(hjust = .5),
       legend.position = "bottom")
```



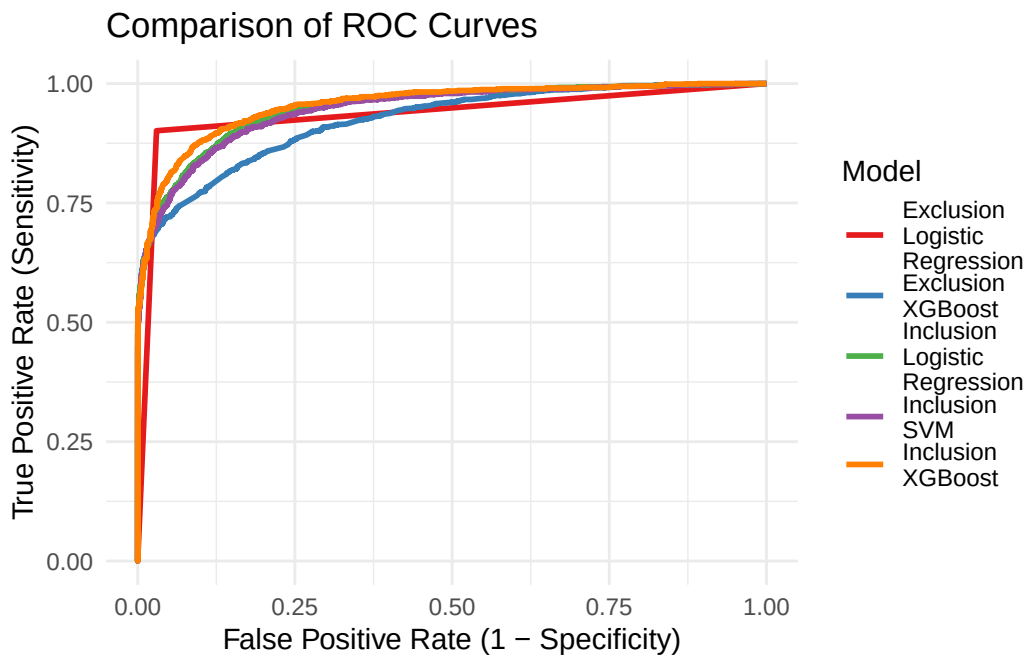
The Exclusion Logistic Regression Model has the highest accuracy, recall, f1, and precision and outperforms the Exclusion XGBoost Model when considering just metrics.

The Inclusion Logistic Regression Model achieved the highest recall across the Inclusion models however the Inclusion XGBoost model outperforms both SVM and XGBoost overall.

5.2 ROC curve

```
# Extract data for plotting
roc_data_lr <- data.frame(
  FPR = 1 - roc_curve_lr$specificities,
  TPR = roc_curve_lr$sensitivities,
  Model = "Inclusion Logistic Regression"
)
roc_data_xgb <- data.frame(
  FPR = 1 - roc_curve_xgb$specificities,
  TPR = roc_curve_xgb$sensitivities,
  Model = "Inclusion XGBoost"
```

```
)
roc_data_svm <- data.frame(
  FPR = 1 - roc_curve_svm$specificities,
  TPR = roc_curve_svm$sensitivities,
  Model = "Inclusion SVM"
)
roc_data_lr_ex <- data.frame(
  FPR = 1 - roc_curve_lr_nr$specificities,
  TPR = roc_curve_lr_nr$sensitivities,
  Model = "Exclusion Logistic Regression"
)
roc_data_xgb_ex <- data.frame(
  FPR = 1 - roc_curve_xgb_nr$specificities,
  TPR = roc_curve_xgb_nr$sensitivities,
  Model = "Exclusion XGBoost"
)
# Combine all ROC data
roc_data <- rbind(roc_data_lr, roc_data_xgb, roc_data_svm,
                  roc_data_lr_ex, roc_data_xgb_ex )

roc_data$Model <- gsub(" ", "\n", roc_data$Model)
# Plot combined ROC curves
ggplot(roc_data, aes(x = FPR, y = TPR, color = Model)) +
  geom_line(size = 1) +
  labs(
    title = "Comparison of ROC Curves",
    x = "False Positive Rate (1 - Specificity)",
    y = "True Positive Rate (Sensitivity)"
  )  +
  scale_color_brewer(palette = "Set1") +
  theme_minimal() +
  theme(legend.position = "right")
```



The ROC curve (Receiver Operating Characteristic curve) plot evaluates each of the 5 models for their ability to distinguish between positive and negative classes in classifying stroke or no stroke.

The Red Line (Exclusion Logisitic Regression Model) performs well but has a different shape than the other models. The sharp curve might be an indicator of the overfitting happening in this model. Essentially the model could be picking up really well on noise and random fluctuations in the dataset instead of truly identifying the underlying patterns.

The Blue Line (Exclusion XGBoost Model) fits slightly below the three Inclusion models (green, purple, and orange lines) indicating that the Exclusion XGBoost Model achieves a slightly lower true positive rate (recall) for the same positive rate.

The Green Line (Inclusion Logistic Regression Model) fits between the orange model (Inclusion XGBoost) and the purple model (Inclusion SVM).

Overall these plots are sharing the same insight - that of the Inclusion models, XGBoost performs best overall but the Logistic model has a higher recall, and that of the Exclusion models, while the Logistic model performs better than the XGBoost model, it is likely flawed.

5.3 Interpretability

In this section I will investigate the interpretability of both datasets (Inclusion and Exclusion) through their Logistic Regression and XGBoost models.

5.3.1 Inclusion Logistic Regression Interpretation

```r
coefs_lr <- data.frame(tidy(final_logistic_fit))
coefs_lr <- tidy(final_logistic_fit, conf.int = TRUE) %>%
  mutate(
    odds_ratio = exp(estimate),
    conf.low = exp(conf.low),
    conf.high = exp(conf.high),
    # Calculate the percentage increase in odds
    percentage_increase = (odds_ratio - 1) * 100
  )

coefs_lr %>%
  arrange(desc(estimate)) %>%
  select(term, estimate, p.value, odds_ratio, percentage_increase, conf.low, conf.high)
```

```
# A tibble: 28 x 7
   term     estimate   p.value odds_ratio percentage_increase conf.low conf.high
   <chr>       <dbl>     <dbl>      <dbl>               <dbl>    <dbl>     <dbl>
 1 missing~    4.45  4.70e-302      85.4                8436.     67.7     108.
 2 hd2         1.18  3.53e-100       3.27                227.      2.93      3.65
 3 hbp2        1.06  3.43e-126       2.89                189.      2.65      3.15
 4 d           0.942 1.97e-139       2.57                157.      2.38      2.76
 5 ahr         0.608 2.37e- 86       1.84                 83.6     1.73      1.95
 6 hbp         0.595 2.25e-105       1.81                 81.3     1.72      1.91
 7 age_X50~    0.470 4.52e- 48       1.60                 60.0     1.50      1.71
 8 hrbcc       0.272 1.23e- 38       1.31                 31.3     1.26      1.37
 9 hbcal       0.249 1.03e- 14       1.28                 28.2     1.20      1.37
10 gender_m    0.140 7.40e-  8       1.15                 15.0     1.09      1.21
# i 18 more rows
```

Of the 27 coefficients, 18 of them are statistically significant and their estimates are within an interpretable range.

According to the John Hopkins Medical reference I used for the choice in risk factors, high blood pressure was the top risk, heart disease the second and diabetes the third. The coefficients possible heart disease (hd2), possible high blood pressure (hbp2) and diabetes (d) have the three largest estimates, aside from missing_count. They affect the most change on the odds ratio holding all other values constant.

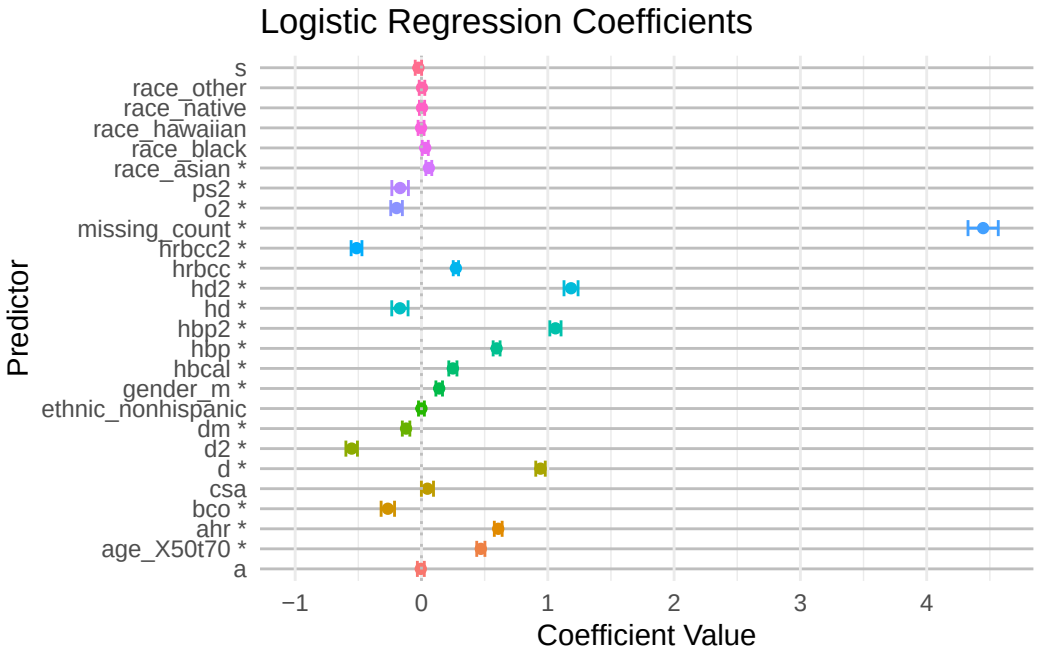
Heart Disease: For a one-standard-deviation increase in possible heart disease (hd2), the odds of stroke increase by a factor of 3.27 (227% increase in the odds), holding all other variables constant. The effect is highly statistically significant. Whereas, for a one-standard deviation increase in heart disease (hd), the odds of stroke decrease by 16% holding all other variables constant. The odds ratio of 0.84 suggests that heart disease has a protective effect or negative relationship with stroke. This effect is statistically significant. There may be a particular variable that is affecting the outcome here as the variables for possible heart disease are very positive.

High Blood Pressure: For a one-standard-deviation increase in possible high blood pressure (hbp2), the odds of stroke increase by a factor of 2.89 (188% increase in the odds), holding all

other variables constant. Additionally for a one-standard-deviation increase in high blood pressure (hbp), the odds of stroke increase by a factor of 1.81 (82% increase in the odds), holding all other variables constant.

Diabetes: For a one-standard-deviation increase in diabetes (d), the odds of stoke increase by a factor of 2.57 (157% increase in the odds), holding all other variables constant. This effect is highly statistically significant which means the association between diabetes and the outcome is not likely due to random chance.

```
coef_data <- coefs_lr %>%
  filter(!term == "(Intercept)") %>%
  mutate(term = ifelse(p.value < 0.05, paste0(term, " *"), term)) %>% na.omit()

# Plot
ggplot(coef_data, aes(x = estimate, y = term, color = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = estimate - std.error, xmax = estimate + std.error),
                 height = 0.8) +
  geom_vline(xintercept = 0, linetype = "dotted", color = "gray") +
  scale_x_continuous(limits = c(-1, NA)) +
  labs(
    title = "Logistic Regression Coefficients",
    x = "Coefficient Value",
    y = "Predictor"
  ) +
  theme_minimal() +
  theme(
    panel.grid.major.y = element_line(color = "gray", linetype = "solid"),
    legend.position = "none"
  )
```



Logistic Regression is inherently interpretable because it directly models the relationship between predictors (electronic health records) and the probability of the outcome (stroke). This transparency is crucial in healthcare for building trust and enabling informed decisions.

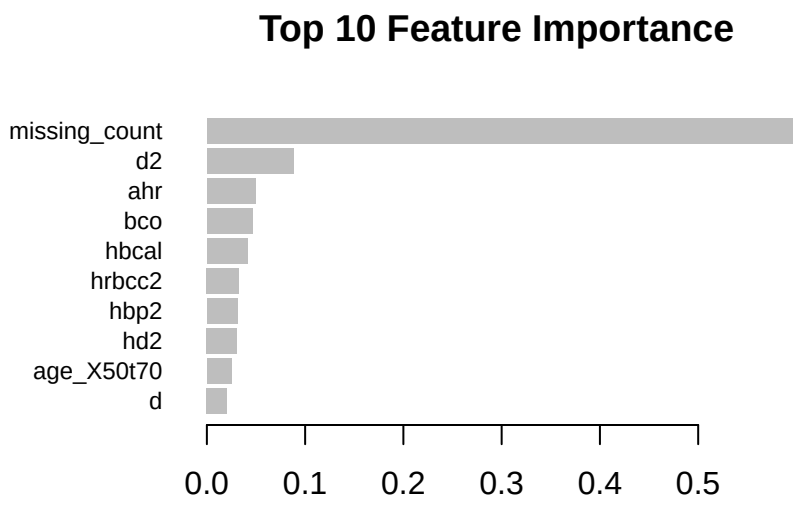
The missing data count has the largest coefficient estimate. This may have to do with how the data was curated by the original authors, where EHRs were averaged for patients without stroke, whereas patients with stroke only had their most recent EHR (potentially most recent doctors visit, blood work, emergency room visit) to represent them.

Some variables are on the negative side of zero, despite all variables being top risk factors for stroke. This could be due to a combination of my manual data sorting of variables not capturing the risk factors entirely and in some instances the risk group may have a lot of variables to choose from, like for instance birth control pills (bco). There were many different drug types to choose from that were available in pill format and easier to distinguish than medication for heart disease, which could be a mixture of different drugs to balance and counteract effects. Birth control pills are also a gendered data point, only used by women. Many women take birth control and never experience stroke.

5.3.2 Inclusion XGBoost Interpretation

```
feature_names <- final_xgb_fit$pre$recipe$term_info$variable
fit_model <- extract_fit_parsnip(final_xgb_fit)
bst <- fit_model$fit

#Training-logloss
importance <- xgb.importance(feature_names = feature_names, model = bst)
importance$Feature <- substr(importance$Feature, 1, 33)
```

```r
xgb.plot.importance(
  importance_matrix = importance,
  top_n = 10,
  main = "Top 10 Feature Importance")
```




```r
kbl(importance %>% mutate(Feature = substr(Feature, 1, 40),
                          across(where(is.numeric), ~ round(., 2))) %>%
      arrange(desc(Gain)))
```

Feature	Gain	Cover	Frequency	Importance
missing_count	0.60	0.25	0.23	0.60
d2	0.09	0.07	0.05	0.09
ahr	0.05	0.06	0.05	0.05
bco	0.05	0.05	0.04	0.05
hbcal	0.04	0.10	0.11	0.04
hrbcc2	0.03	0.02	0.02	0.03
hbp2	0.03	0.05	0.04	0.03
hd2	0.03	0.10	0.15	0.03
age_X50t70	0.02	0.04	0.05	0.02
d	0.02	0.07	0.06	0.02
hrbcc	0.02	0.03	0.04	0.02
hbp	0.01	0.05	0.04	0.01
hd	0.01	0.05	0.06	0.01
dm	0.00	0.01	0.01	0.00
gender_m	0.00	0.01	0.02	0.00
ps2	0.00	0.02	0.02	0.00
s	0.00	0.01	0.02	0.00
a	0.00	0.01	0.01	0.00
race_white	0.00	0.01	0.01	0.00

Here we see missing_count as the most dominant feature, with the highest gain (0.60), cover (0.25), and frequency (0.23). This highlights its significant contribution to the model's predictive power. Comparatively, all other variables exhibit much lower values across these metrics. To better understand its impact, a follow-up analysis excluding missing_count and relying solely on the risk factors would provide insights into how much this feature really drives the model's performance.

Beyond missing_count, d2 (possible diabetes) is the second most important feature, with a gain of 0.09, cover of 0.07 and frequency of 0.05. However, interpreting features in this imbalanced and sparse dataset poses challenges, as most variables contribute only partial data coverage, while missing_count is complete for every individual in the dataset. This completeness likely explains its outsized importance.

5.3.3 Exclusion Logisitic Regression Interpretation

```
coefs_lr_nr <- data.frame(tidy(final_logistic_fit_nr))
dim(coefs_lr_nr)
```

```
[1] 451   5
```

The Risk Exclusion Logistic Regression Model retained 451 coefficients after removing predictors for zero variance and correlation.

```
kbl(coefs_lr_nr %>%
  arrange(desc(estimate)) %>%
  head(5) %>%
  mutate(term = substr(term, 1, 20),
         across(where(is.numeric), ~ round(., 2))))
```

term	estimate	std.error	statistic	p.value
(Intercept)	2.232796e+15	8368196.8	266819209	0
'color of urine_brow	8.170916e+14	13716882.7	59568318	0
'non-small cell lung	3.722379e+14	12694020.2	29323879	0
'patient discharge (	2.060150e+14	749710.3	274792778	0
'professional / anci	1.745759e+14	1114544.8	156634216	0

```
kbl(coefs_lr_nr %>%
  arrange(estimate) %>%
  head(5) %>%
  mutate(term = substr(term, 1, 20),
         across(where(is.numeric), ~ round(., 2))))
```

term	estimate	std.error	statistic	p.value
'sars-cov-2 (covid-1	-5.279838e+14	559814.0	-943141455	0
'hearing examination	-4.617920e+14	10214475.5	-45209567	0
'oxygen saturation i	-4.520813e+14	1598028.2	-282899470	0
childbirth	-4.517888e+14	843118.7	-535854324	0
'ketones [presence]	-4.300492e+14	7628619.5	-56373140	0

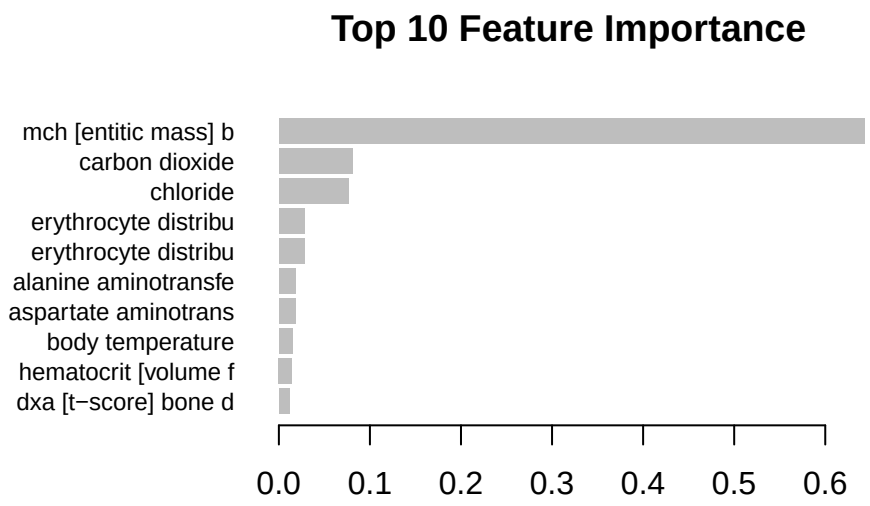
However, a glaring issue with the exclusion logistic regression is that nothing is statistically significant. Additionally, the estimates are on two polar ends with no middle ground and the standard errors are also large.

If there was a way to group these predictors, this model could be more useful for interpretation, but as it stands this model is uninterpretable.

5.3.4 Exclusion XGBoost Interpretation

```
feature_names <- final_xgb_fit_nr$pre$recipe$term_info$variable
fit_model <- extract_fit_parsnip(final_xgb_fit_nr)
bst <- fit_model$fit

#Training-logloss
importance <- xgb.importance(feature_names = feature_names, model = bst)
importance$Feature <- substr(importance$Feature, 1, 20)

xgb.plot.importance(
```

```
  importance_matrix = importance,
  top_n = 10,
  main = "Top 10 Feature Importance")
```




```
kbl(importance %>% mutate(Feature = substr(Feature, 1, 40),
                          across(where(is.numeric), ~ round(., 2))) %>%
      arrange(desc(Gain)) %>% head(10))
```

Feature	Gain	Cover	Frequency	Importance
mch [entitic mass] b	0.64	0.09	0.05	0.64
carbon dioxide	0.08	0.05	0.03	0.08
chloride	0.08	0.08	0.09	0.08
erythrocyte distribu	0.03	0.05	0.07	0.03
alanine aminotransfe	0.02	0.01	0.02	0.02
aspartate aminotrans	0.02	0.03	0.03	0.02
body temperature	0.02	0.04	0.04	0.02
hematocrit [volume f	0.01	0.03	0.02	0.01
dxa [t-score] bone d	0.01	0.03	0.05	0.01
respiratory rate	0.01	0.04	0.04	0.01

The Exclusion XGBoost Model performed very well, however we can imagine that the data is pretty uninterpretable.

A few observations from this feature importance chart.

- carbon dioxide: low levels can be associated with an increased risk of stroke
- aspartate aminotransferase: mildly elevated levels associates with stroke deaths

- dxa bone density: likely a great replacement for 'age'

Without performing a large analysis on this feature importance, I'm unclear on how much the model is grasping at straws and to do that, I'd need a medical expert to go through the dataset and gauge which of the 800 variables could be associated with stroke either lossely, associatively, or directly.

6 Ethical Implications

Stroke is a serious medical condition and one of the leading causes of death in the United States. If insurance companies had access to medical records that used predictive models to identify disease risks based on seemingly unrelated factors (e.g., egg allergy, anemia), they could increase premiums for high-risk customers without their consent or understanding of what makes them high-risk. This raises significant ethical concerns, especially if the data used in analysis is biased. Biases in synthetic or real datasets related to race, gender, location, or access to healthcare resources could unfairly label certain groups as higher risk, perpetuating inequality and injustice.

Moreover, synthetic data, while free from privacy concerns, carries its own biases as it is modeled on real-world patterns that may inherently reflect societal inequities. Synthetic data should not be used for drawing final conclusions but rather as a tool for developing and refining methodologies while being able to protect patient privacy. When transitioning to real-world data, safeguards must be in place to ensure transparency, fairness, and accountability in model predictions. This includes auditing models for bias, ensuring data diversity, and adhering to ethical guidelines to prevent harm or injustice.

As a final point, data analysis should not be done without the guidance of field experts. I have no medical training, and performed much of this analysis through intuition. In a real-world context this kind of research needs to be done with a team, medical experts, and ideally someone with a wide view of health equity in the United States and abroad.

7 References

- Chen, A. (2022). Synthea stroke synthetic patient data series for risk prediction ML. Harvard Dataverse. https://doi.org/10.7910/dvn/lbd9gu
- Johns Hopkins Medicine. (2024). Stroke. www.hopkinsmedicine.org.

https://www.hopkinsmedicine.org/health/conditions-and-diseases/stroke

- OpenAI. (2024). ChatGPT. Accessed from https://openai.com/chatgpt
- World Health Organization. (2024, August 7). The top 10 causes of death. World Health Organization; World Health Organization. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death