Predicting Diabetes Risk with Behavioral Risk Factor Surveillance System (BRFSS)

Jimmy Hwang & Sasha Libolt

Shiley-Marcos School of Engineering, University of San Diego

ADS 503: Applied Predictive Modeling

Professor Ebrahim Tarshizi

June 22, 2025

Table of Contents

Abstract

This study aims to predict diabetes diagnosis using data from the 2023 Behavioral Risk Factor Surveillance System, a comprehensive public health dataset containing over 400,000 observations. Five classification models were developed: logistic regression (LR), penalized logistic regression (PLR), random forest (RF), extreme gradient boosting (XGB), and neural network (NN). The models were evaluated using 10-fold cross-validation with area under the ROC curve (AUC) as the primary performance metric. The best performing model was extreme gradient boosting, with the highest AUC and most favorable confidence interval. Key predictors included age and blood pressure status, with both the presence and absence of high blood pressure contributing significantly to model performance. The findings demonstrate the potential of machine learning models in supporting early diabetes risk detection and informing public health strategies.

Keywords: diabetes prediction, BRFSS, machine learning, binary classification

Predicting Diabetes Risk with Behavioral Risk Factor Surveillance System (BRFSS)

Nearly 40 million people in the United States are diabetic, of whom 22.8% are undiagnosed (Centers for Disease Control and Prevention [CDC], 2024b). In 2022, diabetes accounted for 25% of healthcare spending, with $306 billion in direct medical costs and $106.3 billion in indirect costs (American Diabetes Association, 2023). Diabetes self-management education reduces comorbidity risks: glucose control cuts eye, kidney, and nerve disease by 40%, and blood pressure/cholesterol management cuts stroke risk by 20–50% (CDC, 2024c). The Behavioral Risk Factor Surveillance System (BRFSS) is a public health data collection system that completes over 400,000 interviews a year to gather information on health behaviors, chronic conditions, and preventive practices among adults in the United States (CDC, 2025). This project aims to predict diabetes risk by developing a machine learning model using BRFSS data.

Data Exploration and Preparation

The BRFSS 2023 dataset, obtained in XPT format from the CDC, initially contained 433,333 observations and 350 columns encompassing response data from U.S. adults across all 50 states and territories. For this study, 25,311 observations from 2024 were removed, focusing solely on 2023 data. A data dictionary with a full explanation of all 350 variables can be found in the BRFSS codebook (CDC, 2024a).

Outcome Variable Definition and Initial Filtering

DIABETE4 captures self-reported diabetes status and was selected as the outcome variable. It originally had six different response options. The majority of respondents (82.78%) reported “No” to a diabetes diagnosis, while 13.8% answered “Yes,” indicating a diagnosed condition. Other less frequent responses included “Yes, but female only told during pregnancy” (0.75%), “No, prediabetes or borderline” (2.44%), or “Refused” (0.07%). These additional

response categories were removed. This resulted in significant class imbalance, with 85.7% falling into the negative class and 14.3% in the positive.

Removal of Missing, Noisy or Non-Informative Fields

Due to storage limitations within our GitHub project, columns with over 10% missing data were excluded, consistent with recommendations to mitigate bias (Dong & Peng, 2013). This reduced the dataset from 350 to 127 features. Survey administration variables (e.g., sampling weights, sequence numbers, time stamps) were uninformative to the project and were removed. Features unrelated to physical health or socioeconomic status (e.g., seat belt usage) were also removed. The remaining features were inspected for duplication, with the most informative and least missing representative retained (e.g., retaining the exact age variable over categorized age ranges).

Statistical Analysis for Feature Selection

After data cleaning, 50 features remained. Categorical data were converted into factors, and continuous variables remained numeric. To reduce noise and identify important predictors, formal statistical tests were performed to quantify each feature's relationship with the DIABETE4 outcome variable. Cramer's V was used for categorical variables because it effectively measures the strength of association between categorical data. Features with a p-value greater than .05 or a Cramer's V below .05 were excluded, as values closer to 1 indicate stronger relationships (Geeks for Geeks, 2024). For numerical variables, the outcome variable was converted to binary, and then Pearson's correlation was used to quantify the linear relationships. Pearson's correlation coefficients were converted to absolute values, and features with a p-value greater than .05 or an absolute correlation below .05 were removed, as values closer to 1 indicate stronger linear relationships (Geeks for Geeks, 2025).

Anomaly Investigation

After eliminating statistically insignificant features and discarding rows with missing values, the refined dataset consisted of 363,396 observations and 32 predictors. Each numeric attribute was investigated with visual diagnostics and summary statistics to check for extreme or unlikely values. For the variable CHILDREN, Figure 1 shows a peak distribution of children at "88." Per the BRFSS data dictionary, this indicates that the respondent does not have children, and all these values were converted to "0." Most responses ranged from 0–5 children, with unusual spikes at 15 (1,581 cases) and 30 (4,612). Values above 10 were considered unlikely and removed, resulting in 9,363 dropped rows. The PHYSHLTH variable measures the number of days in the past 30 during which the respondent experienced poor physical health. Figure 2 highlights a concentration of values above 75. Once again, the data dictionary indicated that "88" represented 0 days, which was recoded to 0. Additionally, "77" indicated unsure, and "99" meant answer refused. These were excluded from the dataset.

Figure 1

Distribution of Children Response

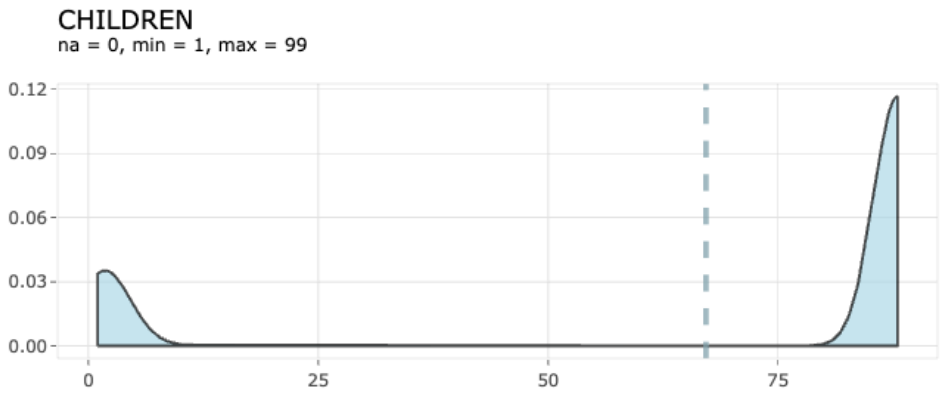
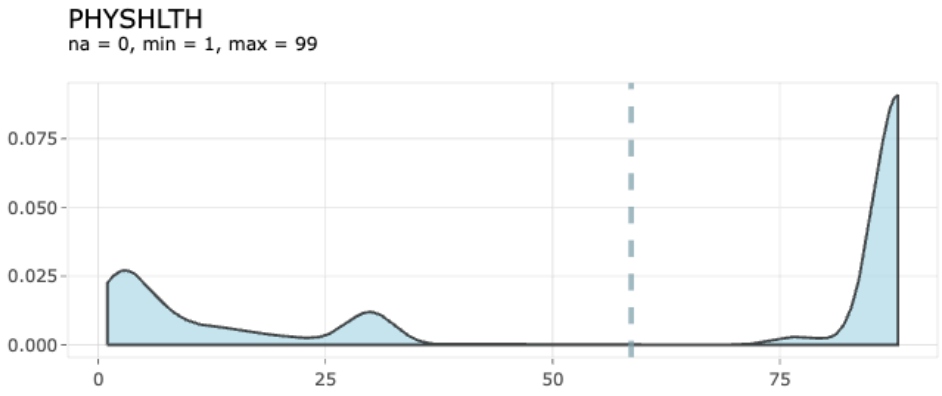


Figure 2

Distribution of Physical Health



Data Exploration

After data cleaning was completed, the distribution and relationships of the final predictor variables to the outcome were visually explored. Figure 3 depicts a clear age-related trend in diabetes outcomes, where “1” indicates a diabetic diagnosis and “3” indicates no diagnosis. Most adults under 50 did not have diabetes, but the proportion increased sharply around the mid-50s. Figure 4 presents diabetic cases by racial category. The lighter bar (DIABETE4 = 1) shows the number of respondents who reported having diabetes, and the darker bar (DIABETE4 = 3) shows the respondents who did not report diabetes. From top to bottom, Other (6), Hispanic (5), American Indian / Alaskan Native (4), Asian (3), Black (2) and White (1) are represented. White adults comprise the majority of survey respondents, but their share is somewhat lower among diabetics. On the other hand, Black respondents occupy a slightly higher percentage of diabetics in their cohort, indicating this population is disproportionately affected. These disparities highlight racial and ethnic differences in diabetic prevalence that warrant attention in both modeling and public health strategies.

Figure 3

Distributions of Age with Diabetes Status

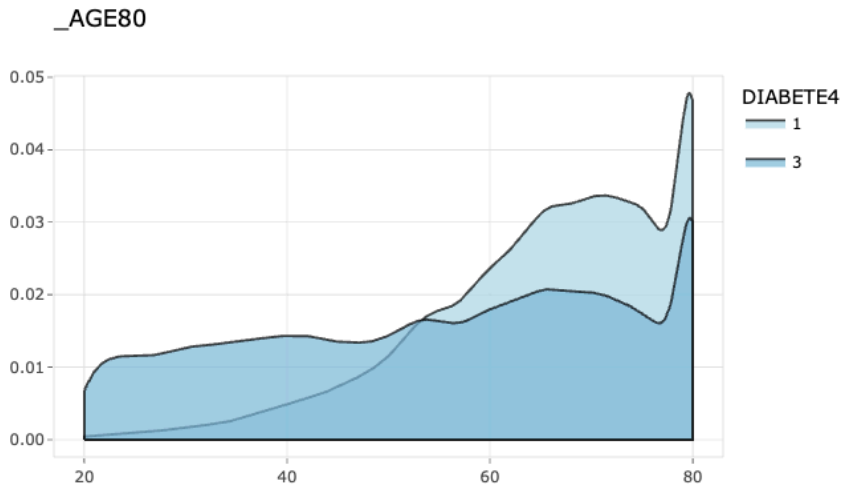
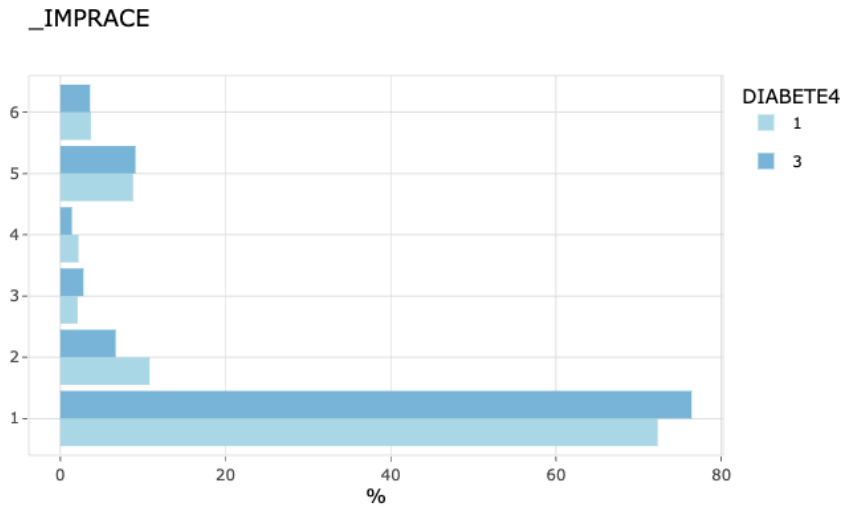


Figure 4

Diabetic Cases by Racial Category



Data Preparation

Prior to modeling, the outcome variable DIABETE4 was converted into a binary format ("Yes" or "No") to enable proper classification. The cleaned dataset was then evaluated for near-zero variance predictors, which can negatively impact model performance by adding noise

and redundancy without contributing meaningful information. A total of five such predictors were removed, leaving 26 predictors for modeling.

Given the large size of the dataset, which could lead to high computational costs, stratified sampling was applied to extract 10% of the data while preserving the outcome distribution. This resulted in a working dataset of 34,576 observations. Next, an 80–20 train-test split was performed. Depending on model requirements, datasets were maintained in raw, encoded, or transformed formats. For models that required standardized input, the training data was preprocessed using centering and scaling.

Data Modeling

To achieve the goal of binary classification using a dataset containing both categorical and continuous variables, five models were selected: logistic regression, penalized logistic regression, random forest, extreme gradient boosting, and neural network. Logistic regression was used as the baseline model for comparison. All models were evaluated using 10-fold cross-validation with a two-class summary metric appropriate for binary classification tasks.

Hyperparameter Tuning

Hyperparameter tuning was done for the penalized logistic regression, random forest, extreme gradient boosting, and neural network models to help improve performance. For penalized logistic regression, the best settings were an alpha of 0.375 and a lambda of 0.0022. This means the model used a mix of lasso and ridge regularization to help prevent overfitting. For the random forest model, the optimal value for mtry was 4, indicating that four randomly selected predictor variables were considered at each split in the decision trees. For extreme gradient boost, the best settings included 100 rounds of boosting, a tree depth of four, and a learning rate of 0.1 while using 70% of the data and 50% of the variables to grow each tree. For

the neural network model, the best setup had just one hidden unit and a decay value of 0.6. This way, the neural network model was kept very simple and used strong regularization.

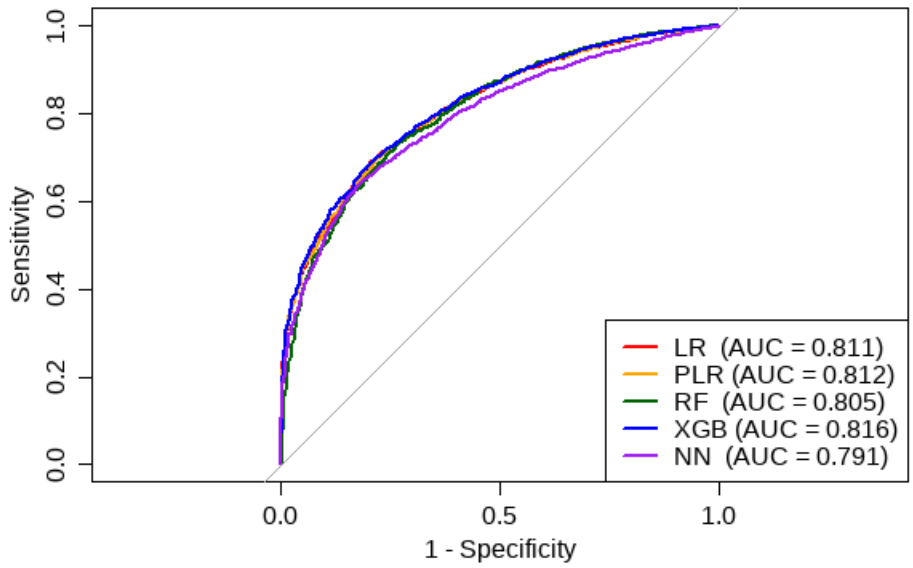
Results

ROC AUC

Model performance was evaluated using ROC curves, with the area under the curve (AUC) representing overall model strength. As shown in Figure 5, extreme gradient boosting achieved the highest AUC (0.816), followed closely by penalized logistic regression (0.812), logistic regression (0.811), random forest (0.805), and neural network (0.791).

Figure 5

ROC Curves for Five Classification Models



False Positive and False Negative Rates

Table 1 displays the false positive rates (FPR) and false negative rates (FNR) for the five models used to predict diabetes. In this context, a false positive rate represents the proportion of people without diabetes who were incorrectly predicted to have it, while a false negative rate reflects the proportion of actual diabetes cases missed by the model. The neural network had an

FPR of 0, meaning it never incorrectly predicted diabetes, but a perfect FNR of 1.0, indicating it failed to detect any true cases. Random forest had the lowest FPR (0.0150), though its FNR was relatively high (0.8669). Logistic regression showed the lowest FNR (0.8410) but the highest FPR (0.0223). Extreme gradient boosting and penalized logistic regression offered a similar trade-off, with moderate FPRs and FNRs.

Table 1

False Positives and False Negatives for Diabetes Prediction by Model

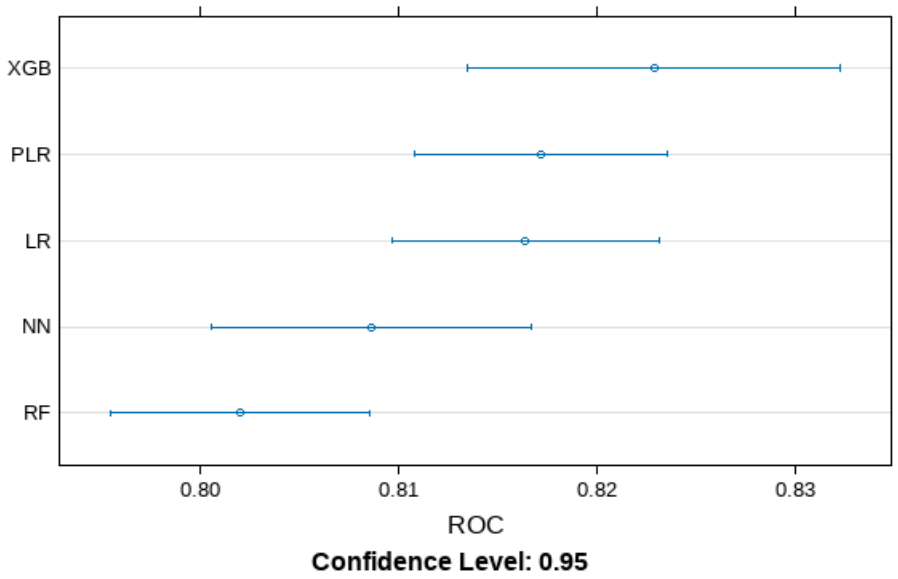
Model	**FPR**	**FNR**
LR	0.0223	0.8410
PLR	0.0193	0.8565
RF	0.0150	0.8669
XGB	0.0192	0.8565
NN	0	1

Confidence Intervals

Figure 6 presents the 95% confidence intervals for each model's AUC based on repeated cross-validation. Extreme gradient boosting not only had the highest mean AUC but also a narrow interval, indicating strong and consistent performance. Although penalized logistic regression and logistic regression had overlapping intervals with extreme gradient boosting, random forest and neural network had wider intervals and lower means, reflecting less stable results.

Figure 6

Confidence Intervals for Model AUC Using 10-Fold Cross-Validation



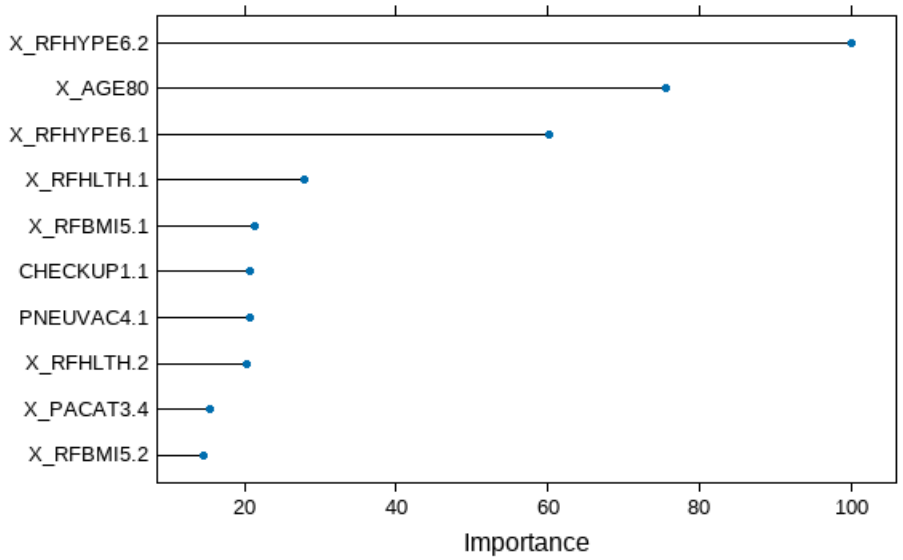
Final Model

Extreme gradient boosting was selected as the final model because it achieved the highest AUC and the most favorable confidence interval among all models tested. In addition to strong overall performance, XGBoost offered one of the lowest false positive rates (0.0192) and a relatively lower false negative rate (0.8565) compared to other models, showing a more effective balance between avoiding false alarms and correctly identifying individuals with diabetes.

The model's variable importance, shown in Figure 7, identified adults with high blood pressure (X_RFHYPE6.2) as the most important predictor of diabetes status, suggesting that hypertension plays a key role in classification. Interestingly, this differed from the other four models, which all ranked age (X_AGE80) as the most influential factor. Nonetheless, age still ranked second in extreme gradient boosting, and the presence or absence of high blood pressure (X_RFHYPE6.2 and X_RFHYPE6.1, respectively) dominated the top features, indicating that both the presence and absence of high blood pressure contributed significantly to the model's predictive power.

Figure 7

Top 10 Most Important Variables in the XGBoost Model for Predicting Diabetes



Note. Except for X_AGE80, all features in Figure 7 are dummy-encoded categorical variables derived from the 2023 BRFSS dataset. Each .1, .2, etc., suffix represents a specific response category within a factor variable (e.g., .1 for "No," .2 for "Yes").

Discussion

This study was limited by computational power and storage capability. The original dataset with over 400,000 observations and 350 features was too large to store in GitHub or process locally. To address this, we performed variable reduction by removing features that had a high level of missingness, appeared duplicative, or statistically insignificant. While this helped reduce dimensionality, it was guided more by practical limitations than optimal feature selection techniques. Future studies with more resources have the opportunity to apply more robust procedures. Even with our final reduced dataset, we encountered difficulty running models on local machines and had to take the 10% stratified sample of our cleaned dataset to further decrease the size. It is important to note this sample is from the already reduced dataset and not

the original one, potentially compounding the risk of excluding meaningful variation. Additionally, this study did not distinguish between Type 1 and Type 2 diabetes, missing an opportunity to explore important patterns specific to diabetic type. Despite these limitations, the models demonstrated strong performance suggesting that further refinement with a more robust dataset could be a worthwhile next step. This initial effort provides a useful benchmark and support for developing further more nuanced models.

Conclusion

This project demonstrated the effectiveness of using machine learning techniques to predict diabetes diagnosis based on self-reported health and behavioral data from the BRFSS. After comparing five classification models, extreme gradient boosting achieved the strongest performance, supported by the highest AUC and consistent cross-validation results. Hyperparameter tuning further improved model accuracy, emphasizing the importance of optimizing algorithms. Key variables influencing prediction included both the presence and absence of high blood pressure, as well as age. Overall, this approach gave useful insights and can be expanded in future research using the full dataset and better computing tools.

References

American Diabetes Association. (2023, November 1). New American Diabetes Association report finds annual costs of diabetes to be $412.9 billion. https://diabetes.org/newsroom/press-releases/new-american-diabetes-association-report-finds-annual-costs-diabetes-be

Centers for Disease Control and Prevention. (2024a, February 20). LLCP 2023 codebook report. Behavioral Risk Factor Surveillance System. https://www.cdc.gov/brfss/annual_data/annual_2023.html

Centers for Disease Control and Prevention. (2024b, May 15). Diabetes data and research. U.S. Department of Health and Human Services. https://www.cdc.gov/diabetes/php/data-research/index.html

Centers for Disease Control and Prevention. (2024c, May 15). Health and economic benefits of diabetes interventions. U.S. Department of Health and Human Services. https://www.cdc.gov/nccdphp/priorities/diabetes-interventions.html

Centers for Disease Control and Prevention. (2025, June 4). Behavioral Risk Factor Surveillance System. U.S. Department of Health and Human Services. https://www.cdc.gov/brfss/

Dong, Y., & Peng, C.-Y. J. (2013). Principled missing data methods for researchers. *SpringerPlus, 2*(1), 222. https://doi.org/10.1186/2193-1801-2-222

Geeks for Geeks. (2024, April 16). How to calculate Cramer's V in R. https://www.geeksforgeeks.org/how-to-calculate-cramers-v-in-r/

Geeks for Geeks. (2025, June 7). Pearson correlation coefficient. https://www.geeksforgeeks.org/pearson-correlation-coefficient/

Appendix

This appendix contains the code behind our project. Our full GitHub repository can be found here: https://github.com/slibolt/ads-503

ADS 503 Group 3 Final Project

Jimmy Hwang & Sasha Libolt

SETUP

Libraries

```
library(haven)
library(caret)
library(tidyverse)
library(dplyr)
library(dlookr)
library(naniar)
library(explore)
library(corrr)
library(gt)
library(pROC)
library(shiny)
library(bslib)
rseed = 100
```

Load Data

```
#file can be downloaded from: https://www.cdc.gov/brfss/annual_data/2023/files/LLCP2023XPT.zi
#file is too large for storage, you must download and update "file_path" to run
# file_path <- "/Users/sashalibolt/Desktop/df.XPT"
file_path <- "df.XPT"
brfss_orig <- read_xpt(file_path)
```

DATA CLEANING

Drop 2024 Data

```
#drop anything that is not 2023
df_drop_24 <- brfss_orig[ brfss_orig$IYEAR == 2023, ] #drop rows
```

Explore Outcome Variable

```
#explore outcome variable, diabetes
table(df_drop_24$DIABETE4, useNA = "ifany")
```

```
    1     2     3     4     7     9  <NA>
56282  3089 337785  9934   640   277     5
```

DIABETE4 values are explained in the table below. Drop any row that is not a "1" for Yes or a "3" for No.

Value	Value Label	Frequency
1	Yes	56,282
2	Yes, but female told only during pregnancy - Go to Section 08.01 AGE	3,089
3	No - Go to Section 08.01 AGE	337,785
4	No, pre-diabetes or borderline diabetes - Go to Section 08.01 AGE	9,934
7	Don't know/Not Sure - Go to Section 08.01 AGE	640
9	Refused - Go to Section 08.01 AGE	277
BLANK	Not asked or Missing	5

```
df_drop_diabetes <- df_drop_24 %>%
  filter(DIABETE4 %in% c(1, 3))
dqr_start <- diagnose(df_drop_diabetes)
```

Drop Columns Missing 10% or More

```r
#identify what has more than 10% missing
high_missing <- dqr_start %>%
  filter(missing_percent >= 10) %>%
  select(variables, missing_percent)

high_missing_col <- high_missing$variables #get all the high missing in the dataframe
#ensure that Diabtype & diabetes are not included in the list
high_missing_col <- setdiff(high_missing_col, c("DIABETE4", "DIABTYPE"))
df_drop_miss <- df_drop_diabetes[ , !(names(df_drop_diabetes) %in% high_missing_col)]
dqr_10_drop <- diagnose(df_drop_miss)
```

Drop Noisy Columns / Don't Have Explanatory Value

```r
#drop columns that are noise / don't add value
noise_to_drop <- c(
  # related to phone information
  "_DUALUSE",
  "_LLCPWT",
  "_LLCPWT2",
  "CPDEMO1C",
  "QSTVER",
  #related to survey identifcation information
  "_PSU",
  "_RAWRAKE",
  "_STRWT",
  "_STSTR",
  "_WT2RAKE",
  "FMONTH",
  "IDATE",
  "IDAY",
  "IMONTH",
  "IYEAR",
  "SEQNO",
  #related to seatbelt use
  "_RFSEAT2",
  "_RFSEAT3",
  "SEATBELT"
)
df_drop_noise <- df_drop_miss[ , !(names(df_drop_miss) %in% noise_to_drop)]
dqr_noise_drop <- diagnose(df_drop_noise)
```

Drop Duplicate Columns

#Look at columns that are duplicates of each other and determine which one to choose. Decisions were made based on data quality, data granularity and information that could be found.

Age Variables

Chose to use _AGE80 as it is the actual numerical value.

```
age_vars <- c(
  "_AGE_G", "_AGE65YR", "_AGE80", "_AGEG5YR"
)

for (varname in age_vars) {
  # Print a header line
  cat("----", varname, "----\n")

  # Use get() to extract the column by name
 print(table(df_drop_noise[[varname]], useNA = "ifany"))

  # Add a blank line for spacing
  cat("\n")
}
```

```
---- _AGE_G ----

     1      2      3      4      5      6
 24397  42229  50910  56058  70488 149985

---- _AGE65YR ----

     1      2      3
238806 148244   7017

---- _AGE80 ----

   18    19    20    21    22    23    24    25    26    27    28    29    30
 3006  3328  3323  3451  3612  3897  3780  4003  3729  3861  4121  3895  4824
   31    32    33    34    35    36    37    38    39    40    41    42    43
 4146  4688  4657  4305  5050  4639  4980  5190  4899  5842  4667  5580  5395
```

```
   44    45    46    47    48    49    50    51    52    53    54    55    56
 4668  5344  4856  5114  5019  4899  6083  5083  6552  6662  6446  6740  6107
   57    58    59    60    61    62    63    64    65    66    67    68    69
 6255  6418  6780  7847  6811  8004  7681  7845  9076  8234  8496  8328  7867
   70    71    72    73    74    75    76    77    78    79    80
 9074  7843  8634  7984  7384  7930  7215  6057  5376  5072 35415

---- _AGEG5YR ----

    1     2     3     4     5     6     7     8     9    10    11    12    13
24396 19601 22605 24596 25771 24580 28291 30950 38016 41736 39590 31518 35400
   14
 7017
```

Alcohol Variables

Chose _DRINKWK as it is a numerical quantification with best quality.

```r
alcohol_vars <- c(
  "_DRNKWK2", "_RFBING6", "_RFDRHV8",  "DRNKANY6", "DROCDY4_"
) #ALCDAY4 not included as it has high level of nulls

for (varname in alcohol_vars) {
  # Print a header line
  cat("----", varname, "----\n")

  # Use get() to extract the column by name
  print(table(df_drop_noise[[varname]], useNA = "ifany"))

  # Add a blank line for spacing
  cat("\n")
}
```

```
---- _DRNKWK2 ----

     0     23     47     70     93    100    117    140    163    187    200
173703  20390  17486   7882  11298   9907   3834   6842    696   4759  10649
   210    233    257    280    300    303    327    350    373    397    400
  1593   5197     12   3786   5971     14    957   3049   1948     28   9062
   420    443    467    490    500    513    537    560    583    600    607
   912      2   4713    391   2630     60     12   1585   1222   7166     72
```

630	653	677	700	747	770	793	800	817	840	887
87	563	56	8809	373	10	22	3143	113	518	5
900	910	933	980	1000	1027	1050	1073	1100	1120	1143
1888	13	2512	193	2624	52	737	22	7	397	30
1167	1190	1200	1213	1260	1283	1300	1307	1330	1353	1400
1225	17	2545	53	103	2	2	377	1	78	7202
1470	1493	1500	1517	1540	1587	1600	1610	1633	1680	1750
35	54	976	5	23	12	518	11	61	136	500
1800	1820	1867	1890	1960	1983	2000	2030	2053	2100	2147
528	23	383	38	188	3	569	56	8	2817	6
2200	2240	2287	2333	2380	2400	2427	2450	2500	2520	2567
2	54	5	366	1	366	11	35	147	28	13
2600	2613	2660	2683	2700	2707	2730	2777	2800	2917	2940
2	77	1	1	8	28	1	2	1378	69	7
2987	3000	3033	3080	3150	3173	3200	3220	3267	3300	3337
4	247	9	5	14	2	44	5	57	1	1
3360	3383	3400	3430	3500	3593	3600	3640	3733	3757	3780
38	7	1	4	738	2	116	6	44	1	7
3850	3900	3920	3967	4000	4060	4083	4107	4200	4247	4400
1	2	38	1	70	13	14	3	815	2	1
4410	4480	4500	4550	4573	4667	4737	4800	4853	4900	5000
3	6	26	1	7	70	2	33	2	161	32
5040	5133	5200	5227	5250	5367	5400	5413	5600	5833	5880
2	1	2	10	17	5	6	3	253	18	1
5973	6000	6067	6090	6300	6400	6417	6440	6500	6533	6720
1	46	2	2	44	2	1	2	1	8	1
6767	6800	7000	7187	7200	7280	7467	7500	7513	7560	7700
3	2	220	1	8	2	1	9	1	2	3
7840	8000	8120	8167	8400	8750	9000	9100	9333	9450	9473
4	6	4	2	196	7	9	2	7	1	1
9600	9660	9800	10000	10500	10733	10827	11200	11667	11900	11947
4	2	12	6	56	1	2	21	2	6	1
12000	12500	12600	13067	13300	14000	14210	14400	14583	14700	15000
3	1	19	2	1	32	1	1	1	2	4
15200	15400	16000	16100	16200	16800	17500	18000	18293	18900	19600
1	1	2	1	1	17	6	2	1	2	1
19623	20000	21000	22400	23333	24500	25200	26133	27067	28000	30000
1	1	35	1	1	1	1	1	1	5	4
31500	33600	35000	36000	41300	42000	43400	48300	49000	51100	52500
2	2	4	1	1	5	1	1	1	1	1
53200	99900									
2	29044									

```
---- _RFBING6 ----

     1      2      9
315840  48689  29538

---- _RFDRHV8 ----

     1      2      9
343184  21839  29044

---- DRNKANY6 ----

     1      2      7      9
194484 173144   2410  24029

---- DROCDY4_ ----

     0      3      7     10     13     14     17     20     23     27     29
173144  29953  22592  12782  10059  19304   8640   3468   2064   2863  16327
    30     33     37     40     43     47     50     53     57     60     63
   223   5653     42   1500  11830    498   4742    215   5921    190      9
    67     70     71     73     77     80     83     86     87     90     93
  5424    190   5484    138     69    198   2624   1834    137    181    790
    97    100    900
   259  18281  26439
```

Arthritis Variables

Chose to use _DRXAR2 which is a Y/N indicator of arthritis

```
arth_vars <- c(
  "_DRDXAR2", "HAVARTH4"
)

for (varname in arth_vars) {
  # Print a header line
  cat("----", varname, "----\n")

  # Use get() to extract the column by name
  print(table(df_drop_noise[[varname]], useNA = "ifany"))

  # Add a blank line for spacing
```

```
  cat("\n")
}
```

```
---- _DRDXAR2 ----

     1      2   <NA>
129061 262969   2037

---- HAVARTH4 ----

     1      2      7      9
129061 262969   1960     77
```

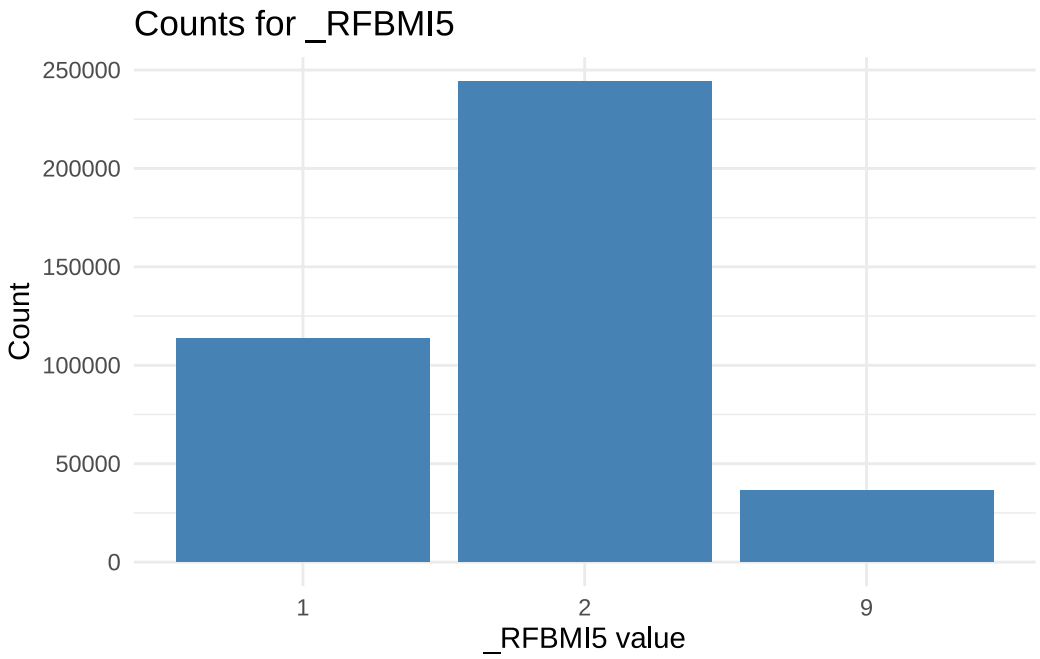
BMI, Height & Weight

BMI is simply a calculated ration between height and weight so evaluated all these as one to make a final determination. Determined that the best to keep would be _BMI5CAT as there were significant data quality issues with height and weight indicators.

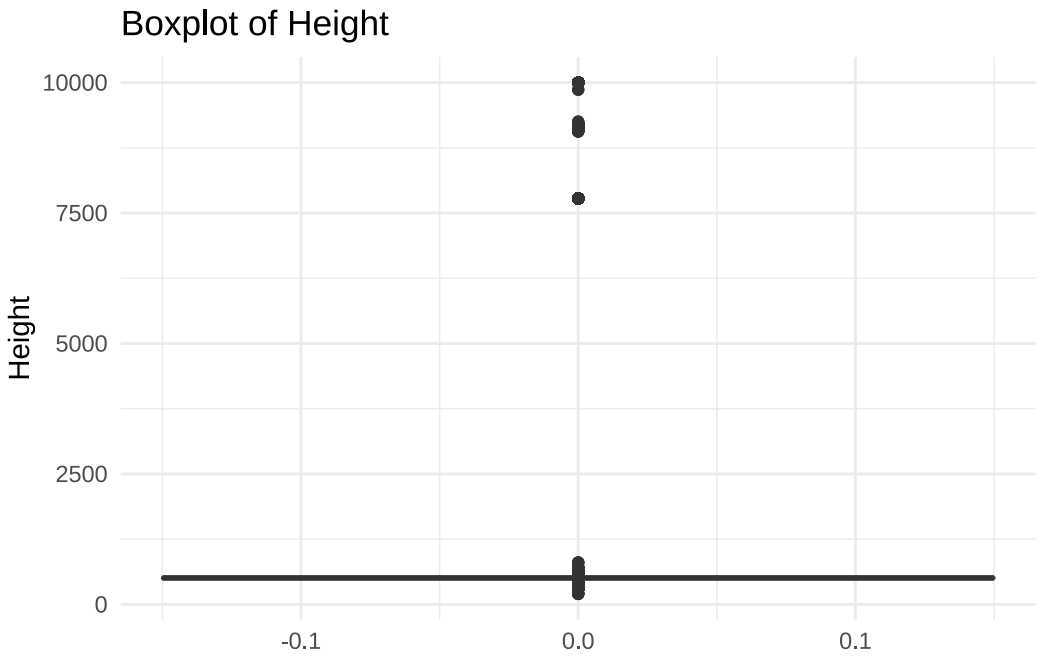
```
bmi_h_w_vars <- c(
  "_BMI5", "_BMI5CAT" , "_RFBMI5", "HEIGHT3", "HTIN4", "HTM4", "WEIGHT2", "WTKG3"
)

bmi_h_w_summary <- df_drop_noise [ , bmi_h_w_vars, drop = FALSE]

dqr_bmi_weight <- diagnose(bmi_h_w_summary)

dqr_bmi_weight
```

```
# A tibble: 8 x 6
  variables types   missing_count missing_percent unique_count unique_rate
  <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
1 _BMI5     numeric         36274            9.21         3980  0.0101
2 _BMI5CAT  numeric         36274            9.21            5  0.0000127
3 _RFBMI5   numeric             0            0               3  0.00000761
4 HEIGHT3   numeric         10260            2.60          174  0.000442
5 HTIN4     numeric         24686            6.26           59  0.000150
6 HTM4      numeric         19593            4.97          116  0.000294
7 WEIGHT2   numeric          9329            2.37          604  0.00153
8 WTKG3     numeric         30606            7.77          592  0.00150
```

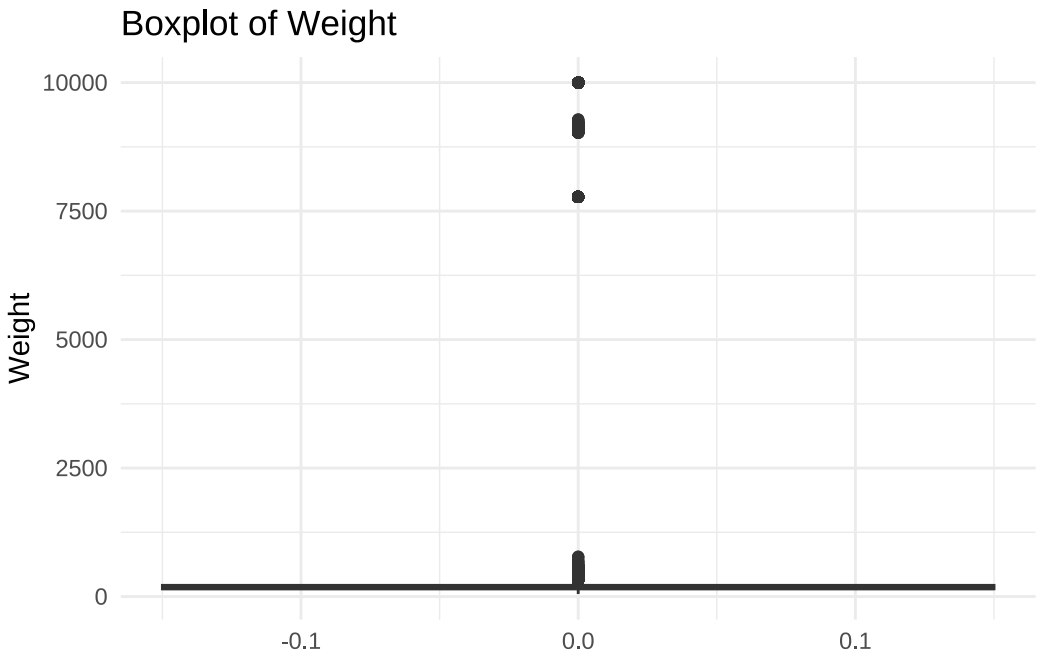
```r
#Bar chart of Obese
ggplot(df_drop_noise, aes(x = factor(`_RFBMI5`, levels = c(1, 2, 9)))) +
  geom_bar(na.rm = TRUE, fill = "steelblue") +
  labs(
    title = "Counts for _RFBMI5",
    x     = "_RFBMI5 value",
    y     = "Count"
  ) +
  theme_minimal()
```




```r
#boxplot of height
ggplot(df_drop_noise, aes(y = `HEIGHT3`)) +
  geom_boxplot(na.rm = TRUE,# ignore missing values
               fill  = "steelblue",
               width = 0.3) +
  labs(
    title = "Boxplot of Height",
    y     = "Height"
  ) +
  theme_minimal()
```




```
#histogram of weight

ggplot(df_drop_noise, aes(y = `WEIGHT2`)) +
  geom_boxplot(na.rm = TRUE,# ignore missing values
               fill  = "steelblue",
               width = 0.3) +
  labs(
    title = "Boxplot of Weight",
    y     = "Weight"
  ) +
  theme_minimal()
```



Diabetes Type

Diabetes type is potentially useful if we can narrow down between Type I and Type II. Investigating any time that DIABETES4 = 1, indicating a "YES" for a diabetic diagnosis the data quality revealed itself to be too poor for usage. Drop DIABTYPE.

```
df_diab_pos <- df_drop_noise[df_drop_noise$DIABETE4 == 1, ]
table(df_diab_pos$DIABTYPE, useNA = "ifany")
```

```
    1     2     7     9  <NA>
 1821 18804  2021    49 33587
```

Value	Value Label	Frequency
1	Type 1	1,821
2	Type 2	18,804
7	Don't know/Not Sure	2021
9	Refused	49
BLANK	Not asked or Missing	35,587
	Notes: Section 07.12, DIABETE4, is coded 2, 3, 4, 7, 9, or Missing	

Duplicate Column Removal

Besides the analysis above, some duplicate decisions were made between two columns based on which one provided the most information. Final removal list is below:

```
columns_to_drop <- c(
  # AGE
  "_AGEG5YR", "_AGE_G", "_AGE65YR",

  # ALCOHOL
  "ALCDAY4", "_RFBING6", "_RFDRHV8", "DRNKANY6", "DROCDY4_", "_DRNKDRV",

  # Arthritis
  "HAVARTH4",

  # BMI, Height, Weight
  "_BMI5", "_BMI5CAT", "HTIN4", "WTKG3", "HTM4", "HEIGHT3", "WEIGHT2",

  # Physical fitness
  "_PA150R4", "_PA30023", "_PA300R4", "_PAINDX3", "_PAREC3",
  "_PASTAE3", "_PASTRNG", "_PHYS14D", "_TOTINDA", "EXERANY2",
  "PAMISS3_", "STRENGTH", "STRFREQ_",

  # Race
  "_HISPANC", "_MRACE1", "_RACE", "_RACEG21", "_RACEGR3", "_RACEPRV",

  # Smoking
  "_RFSMOK3", "ECIGNOW2", "SMOKE100", "USENOW3", "_CURECI2",

  # Mental Health
  "_MENT14D", "MENTHLTH",

  # Asthma
  "_CASTHM1", "_LTASTH1", "ASTHMA3",

  # Heart
  "CVDCRHD4", "CVDINFR4",

  # Insurance
  "HCVU653", "PRIMINS1",

  # Miscellaneous
  "_CHLDCNT", "CHOLCHK3", "EDUCAG", "_SEX", "_INCOMG1", "HIVTST7", "BPHIGH6", "GENHLTH",
```

```
  "DIABTYPE", "_EDUCAG"
)
df_drop_dupe <- df_drop_noise %>%
  select(-any_of(columns_to_drop))
dqr_drop_dupe <- diagnose(df_drop_dupe)
```

Convert Categorical Variables to Factors

```
factor_cols <- dqr_drop_dupe %>%
  filter(unique_count <= 10) %>%
  pull(variables)

df_convert_factor <- df_drop_dupe %>%
  mutate(across(all_of(factor_cols), as.factor))
dqr_convert_factor <- diagnose(df_convert_factor)
```

Statistical Testing for Feature Importance

Cramer's V for Categorical Variables

Cramer's V is the a measure of relationship between categorical variables, 1 being perfect and 0 being no relationship.

```
library(vcd)
```

```
Warning: package 'vcd' was built under R version 4.4.3
```

```
Loading required package: grid
```

```
# Get all factor predictors (excluding the outcome)
factor_vars <- df_convert_factor %>%
  select(where(is.factor)) %>%
  select(-DIABETE4) %>%
  names()

# Loop through and run chi-square + cramer v, dropping NAs
chi_results <- map_dfr(factor_vars, function(var) {
```

```r
  # Drop rows where null
 temp_data <- df_convert_factor %>%
    select(all_of(var), DIABETE4) %>%
    filter(!is.na(.data[[var]]))

  # Create contingency table
  tbl <- table(temp_data[[var]], temp_data$DIABETE4)

  # Run chi-square test and get cramer v
  if (nrow(tbl) > 1 && ncol(tbl) > 1) {
    test <- suppressWarnings(chisq.test(tbl))
    cramers_v <- suppressWarnings(assocstats(tbl)$cramer)
    tibble(variable = var, p_value = test$p.value, cramers_v = cramers_v)
  } else {
    tibble(variable = var, p_value = NA, cramers_v = NA)
  }
}) %>%
  filter(!is.na(p_value)) %>%
  arrange(desc(cramers_v))

# View results
head(chi_results, 10)
```

```
# A tibble: 10 x 3
   variable p_value cramers_v
   <chr>      <dbl>     <dbl>
 1 _RFHYPE6       0     0.274
 2 _RFHLTH        0     0.226
 3 DIFFWALK       0     0.222
 4 EMPLOY1        0     0.217
 5 PNEUVAC4       0     0.189
 6 _MICHD         0     0.178
 7 CHCKDNY2       0     0.177
 8 _DRDXAR2       0     0.168
 9 _PACAT3        0     0.158
10 _HCVU653       0     0.153
```

```r
chi_results <- chi_results %>%
  mutate(selection = case_when(
    p_value >= 0.05 ~ "Drop",
   cramers_v >= 0.10 ~ "Keep",
```

```
   cramers_v >= 0.05 & cramers_v < 0.10 ~ "Maybe",
    TRUE ~ "Drop"
  ))
chi_results %>%
  count(selection)
```

```
# A tibble: 3 x 2
  selection     n
  <chr>     <int>
1 Drop         12
2 Keep         19
3 Maybe        10
```

```
#extract all the drops
cat_drop <- chi_results %>%
  filter(selection == "Drop") %>%
  pull(variable)

# View the list
cat_drop
```

```
 [1] "ADDEPEV3" "_HLTHPL1" "COVIDPO1" "_METSTAT" "CHCSCNC1" "_URBSTAT"
 [7] "RENTHOM1" "_AIDTST4" "DISPCODE" "QSTLANG"  "SEXVAR"   "MEDCOST1"
```

Pearson's Correlation for Numerical

```
#convert outcome to binary numerical for now
df_numeric <- df_convert_factor %>%
  mutate(diabetes_binary = ifelse(DIABETE4 == "1", 1, 0))

#get mumeric vars
numeric_vars <- df_numeric %>%
  select(where(is.numeric)) %>%
  select(-diabetes_binary) %>%
  names() # remove outcome column

#run pearson correlation
cor_results <- map_dfr(numeric_vars, function(var) {
  test <- cor.test(df_numeric[[var]], df_numeric$diabetes_binary, use = "complete.obs")
```

```
  tibble(
    variable = var,
    correlation = test$estimate,
    p_value = test$p.value,
    abs_correlation = abs(test$estimate)
  )
}) %>%
  arrange(desc(abs_correlation))
head(cor_results, 10)
```

```
# A tibble: 8 x 4
  variable correlation  p_value abs_correlation
  <chr>          <dbl>    <dbl>           <dbl>
1 _AGE80       0.217   0               0.217
2 CHILDREN     0.0928  0               0.0928
3 PHYSHLTH    -0.0833  0               0.0833
4 MAXVO21_    -0.0216  4.53e-42        0.0216
5 _DRNKWK2    -0.0163  1.76e-24        0.0163
6 FC601_      -0.00976 9.01e-10        0.00976
7 _STATE       0.00636 6.54e- 5        0.00636
8 INCOME3     -0.00511 1.48e- 3        0.00511
```

```
cor_results <- cor_results %>%
  mutate(selection = case_when(
    p_value >= 0.05 ~ "Drop",
   abs_correlation >= 0.10 ~ "Keep",
   abs_correlation >= 0.05 & abs_correlation < 0.10 ~ "Maybe",
    TRUE ~ "Drop"
  ))
cor_results %>%
  count(selection)
```

```
# A tibble: 3 x 2
  selection     n
  <chr>     <int>
1 Drop          5
2 Keep          1
3 Maybe         2
```

```r
#extract all the drops
num_drop  <- cor_results %>%
  filter(selection == "Drop") %>%
  pull(variable)

# View the list
num_drop
```

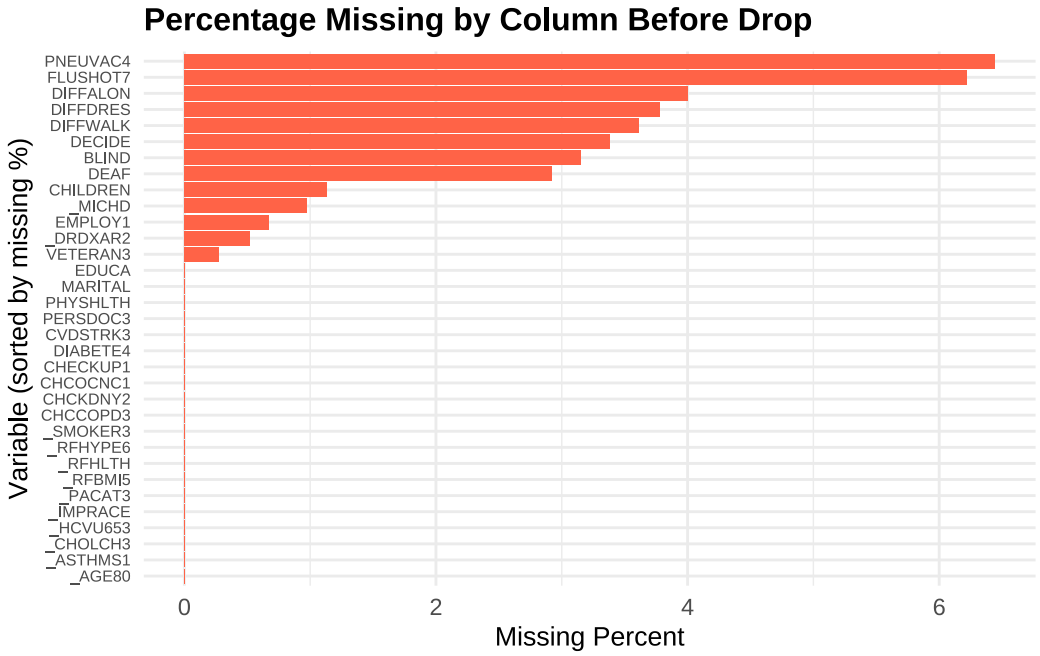
```
[1] "MAXVO21_" "_DRNKWK2" "FC601_"   "_STATE"   "INCOME3"
```

Drop Insignificant Columns

```r
#combine cat and num drops
all_drop <- c(cat_drop, num_drop)
#drop columns
df_insig_drop <- df_convert_factor %>%
  select(-all_of(all_drop))
dqr_start <- diagnose(df_insig_drop)
```

Explore Missing Data

```r
ggplot(dqr_start, aes(
       x = reorder(variables, missing_percent),
       y = missing_percent
     )) +
  geom_col(fill = "tomato") +
  coord_flip() +
  labs(
    title = "Percentage Missing by Column Before Drop",
    x     = "Variable (sorted by missing %)",
    y     = "Missing Percent"
  ) +
  theme_minimal() +
  theme(
    axis.text.y = element_text(size = 6),
    axis.title  = element_text(size = 10),
    plot.title  = element_text(size = 12, face = "bold")
  )
```

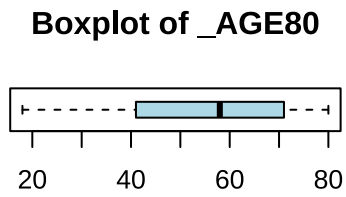



```
#drop all nulls
df_drop_null <- df_insig_drop %>%
  drop_na()
```

```
diagnose(df_drop_null)
```

```
# A tibble: 33 x 6
   variables types   missing_count missing_percent unique_count unique_rate
   <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
 1 PHYSHLTH  numeric             0               0           33  0.0000908
 2 PERSDOC3  factor              0               0            5  0.0000138
 3 CHECKUP1  factor              0               0            7  0.0000193
 4 CVDSTRK3  factor              0               0            4  0.0000110
 5 CHCOCNC1  factor              0               0            4  0.0000110
 6 CHCCOPD3  factor              0               0            4  0.0000110
 7 CHCKDNY2  factor              0               0            4  0.0000110
 8 DIABETE4  factor              0               0            2  0.00000550
 9 MARITAL   factor              0               0            7  0.0000193
10 EDUCA     factor              0               0            7  0.0000193
# i 23 more rows
```

```r
#get numeric varibles
numeric_final <- df_drop_null[, sapply(df_drop_null, is.numeric)]

par(mfrow = c(2, 2))

for (var_name in names(numeric_final)) {
  boxplot(numeric_final[[var_name]],
          main = paste("Boxplot of", var_name),
          horizontal = TRUE,
          col = "lightblue")
}
```




```r
summary(df_drop_null$PHYSHLTH)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   14.00   88.00   58.63   88.00   99.00
```

```r
summary(df_drop_null$CHILDREN)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   88.00   88.00   67.16   88.00   99.00
```

```r
summary(df_drop_null$'_AGE80')
```

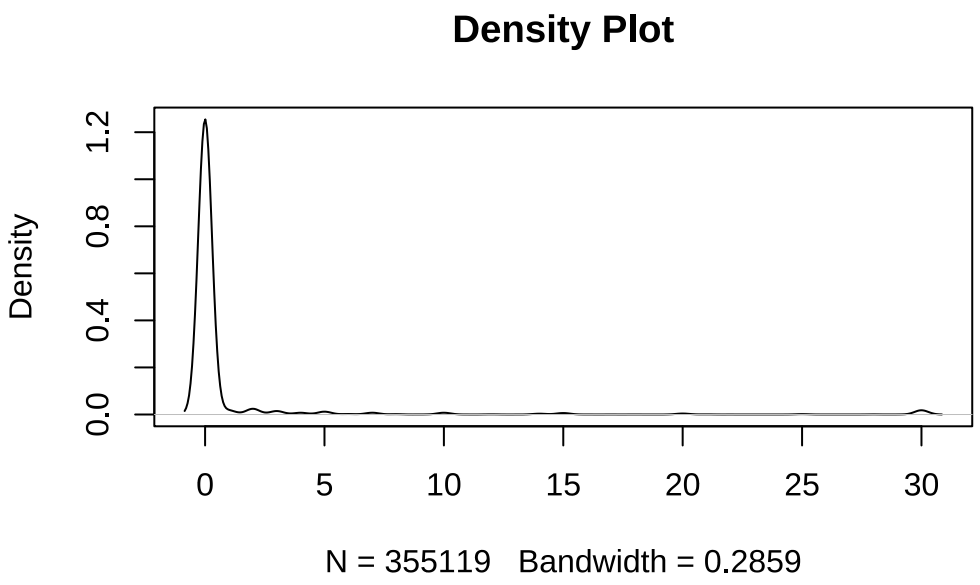
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   41.00   58.00   55.35   71.00   80.00
```

PHYSHLTH 1-30 is number of days. “88” means none,“77” mean’s don’t know. 99 means refused. Turn 88 to “0”, drop 77, drop 99.

```
#convert 88 to 0
df_convert80 <- df_drop_null %>%
  mutate(PHYSHLTH = ifelse(PHYSHLTH == 88, 0, PHYSHLTH))
df_drop_phys <- df_convert80 %>%
  filter(!(PHYSHLTH %in% c(77, 99)))
```

Children 1 - 87 means number of children, 88 means none, 99 means refused.

```
df_child_88 <- df_drop_phys %>%
  mutate(CHILDREN = ifelse(CHILDREN == 88, 0, PHYSHLTH))
plot(density(df_child_88$CHILDREN, na.rm = TRUE), main = "Density Plot")
```




```
children_pivot <- as.data.frame(table(df_child_88$CHILDREN))
colnames(children_pivot) <- c("CHILDREN", "count")
children_pivot
```

```
   CHILDREN  count
1         0 321699
```

2	1	4335
3	2	6219
4	3	3743
5	4	1883
6	5	3090
7	6	496
8	7	1938
9	8	357
10	9	76
11	10	1920
12	11	31
13	12	173
14	13	33
15	14	863
16	15	1581
17	16	44
18	17	47
19	18	61
20	19	13
21	20	1029
22	21	147
23	22	38
24	23	23
25	24	31
26	25	388
27	26	29
28	27	34
29	28	126
30	29	60
31	30	4612

Unlikely that people have 30 children. Most common distribution is between 0 - 5 children. There is an unusual spike at 15 (1,581 cases) and 30 children (4,612) suggesting a placeholder. Will drop anything less than 10.

```
sum(df_child_88$CHILDREN > 10, na.rm = TRUE)
```

```
[1] 9363
```

```
df_clean_a <- df_child_88[df_child_88$CHILDREN <= 10, ]
# Create a folder if it doesn't exist
```

```r
if (!dir.exists("data_files")) dir.create("data_files")

# Save the dataframe as CSV
write.csv(df_clean_a, "data_files/df_clean.csv", row.names = FALSE)
```

```r
df_clean_a %>% explore()
```

```
Warning in explore_shiny(data, ...): This function can only be used in an
interactive R session
```

DATA PRE-PROCESSING

Feature Engineering

Convert Outcome DIABETE4 to "Yes" or "No"

```r
# Add "X" in front of features that start with "_"
names(df_clean_a) <- sub("^_", "X_", names(df_clean_a))
```

```r
df_clean_a <- df_clean_a %>%
  mutate(
    DIABETE4 = as.numeric(as.character(DIABETE4)),
    DIABETE4 = case_when(
      DIABETE4 == 1 ~ "Yes",
      DIABETE4 == 3 ~ "No",
      TRUE ~ NA_character_
    )
  )

# Make DIABETE4 variable a factor
df_clean_a$DIABETE4 <- factor(df_clean_a$DIABETE4, levels = c("No", "Yes"))
```

Modeling Preparation

Check for Near Zero Variance

```
nzv <- nearZeroVar(df_clean_a[, setdiff(names(df_clean_a), "DIABETE4")])
df_clean_nzv <- df_clean_a[, -nzv]
```

Stratified Sampling for Large Data Set

```
set.seed(rseed)
df_sampled <- df_clean_nzv %>%
                  group_by(DIABETE4) %>%
                  sample_frac(size = 0.10)
```

```
#check distribution of new dataset
table(df_sampled$DIABETE4, useNA = "ifany")
```

```
   No   Yes
29765  4811
```

Train-Test Splitting

```
set.seed(rseed)

# Extract predictor X and outcome y
X <- df_sampled[, setdiff(names(df_sampled), "DIABETE4")]
y <- df_sampled$DIABETE4

# 80-20 split
train_index <- createDataPartition(y, p = 0.8, list = FALSE)
trainX <- X[train_index, ]
trainY <- y[train_index]
testX <- X[-train_index, ]
testY <- y[-train_index]

# Dummies + encoding for numeric-only models
dummies <- dummyVars("~ .", data = trainX)
trainX_dummies <- predict(dummies, newdata = trainX)
testX_dummies <- predict(dummies, newdata = testX)
```

Pre-processing

```
set.seed(rseed)

prep <- preProcess(trainX_dummies, method = c("center", "scale"))

# Train-test splits for factor-compatible models
trainX_trans <- predict(prep, trainX_dummies)
testX_trans <- predict(prep, testX_dummies)
```

MODELING

```
set.seed(rseed)

ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     number = 10,
                     classProbs = TRUE,
                     savePredictions = TRUE)
```

Logistic Regression (LR)

```
set.seed(rseed)

lrFit <- train(x = trainX_trans,
               y = trainY,
               method = "glm",
               metric = "ROC",
               trControl = ctrl)
```

```
varImp(lrFit)
```

```
glm variable importance

  only 20 most important variables shown (out of 81)
```

```
             Overall
X_AGE80       100.00
X_RFBMI5.1     63.17
X_MICHD.1      61.69
X_HCVU653.1    53.43
X_PACAT3.1     43.87
X_IMPRACE.3    30.30
X_DRDXAR2.1    30.23
X_PACAT3.2     29.39
CHCOCNC1.1     25.67
CHCOCNC1.7     24.81
X_HCVU653.2    23.70
CHCOCNC1.2     23.67
X_IMPRACE.2    22.50
X_CHOLCH3.3    22.49
X_RFBMI5.2     21.31
X_RFHYPE6.1    20.73
X_PACAT3.4     19.58
X_CHOLCH3.2    18.87
X_IMPRACE.1    18.11
X_CHOLCH3.1    17.65
```

Penalized Logistic Regression (PLR)

```
set.seed(rseed)

plrGrid = expand.grid(
  alpha = 0.375,
  lambda = 0.002154435
)

plrFit <- train(x = trainX_trans,
                y = trainY,
                method = "glmnet",
                tuneGrid = plrGrid,
                metric = "ROC",
                trControl = ctrl)
```

```
varImp(plrFit)
```

```
glmnet variable importance
```

```
  only 20 most important variables shown (out of 105)

            Overall
X_AGE80      100.00
X_RFBMI5.1    64.68
X_RFHYPE6.1   58.28
CHECKUP1.1    53.57
X_RFHYPE6.2   44.81
X_IMPRACE.1   41.18
PNEUVAC4.1    38.65
X_RFHLTH.1    33.98
X_PACAT3.4    31.25
PERSDOC3.3    26.64
X_HCVU653.9   25.84
X_RFBMI5.2    24.23
EDUCA.6       24.02
X_RFHLTH.2    22.72
X_CHOLCH3.3   20.86
X_CHOLCH3.1   20.01
X_CHOLCH3.2   17.84
CHECKUP1.4    17.57
X_PACAT3.9    16.33
PNEUVAC4.2    14.74
```

Random Forest (RF)

```r
set.seed(rseed)

trainX_df <- as.data.frame(trainX)
mtryValues = 4

rfFit <- train(
  x = trainX_df,
  y = trainY,
  method = "rf",
  ntree = 500,
  tuneGrid = data.frame(mtry = mtryValues),
  metric = "ROC",
  trControl = ctrl
)
```

```r
varImp(rfFit)
```

```
rf variable importance

  only 20 most important variables shown (out of 27)

          Overall
X_AGE80   100.000
PHYSHLTH   44.816
MARITAL    44.029
X_PACAT3   40.972
EDUCA      40.653
X_RFHYPE6  39.290
X_IMPRACE  29.820
X_SMOKER3  28.969
X_RFHLTH   21.775
PERSDOC3   21.254
X_RFBMI5   20.812
PNEUVAC4   20.509
FLUSHOT7   14.256
X_ASTHMS1  14.154
X_DRDXAR2  11.771
X_MICHD    10.072
X_HCVU653   9.271
CHCOCNC1    8.655
CHECKUP1    7.314
VETERAN3    7.010
```

Extreme Gradient Boosting (XGBoost)

```r
xgbGrid <- expand.grid(
  nrounds = 100,
  max_depth = 4,
  eta = 0.1,
  gamma = 0,
  colsample_bytree = 0.5,
  min_child_weight = 1,
  subsample = 0.7
)
```

```r
xgbFit <- train(
  x = trainX_dummies,
  y = trainY,
  method = "xgbTree",
  trControl = ctrl,
  tuneGrid = xgbGrid,
  metric = "ROC"
)
```

```r
varImp(xgbFit)
```

```
xgbTree variable importance

  only 20 most important variables shown (out of 105)

            Overall
X_RFHYPE6.2 100.000
X_AGE80      75.527
X_RFHYPE6.1  60.063
X_RFHLTH.1   27.959
X_RFBMI5.1   21.421
CHECKUP1.1   20.766
PNEUVAC4.1   20.659
X_RFHLTH.2   20.214
X_PACAT3.4   15.393
X_RFBMI5.2   14.507
X_IMPRACE.1  13.245
X_MICHD.1    10.846
PNEUVAC4.2    9.051
PHYSHLTH      7.468
EDUCA.6       7.144
X_DRDXAR2.1   5.236
X_CHOLCH3.1   4.515
PERSDOC3.3    4.068
X_PACAT3.1    3.694
X_HCVU653.1   3.566
```

Neural Network (NN)

```r
set.seed(rseed)

nnGrid <- expand.grid(
  decay = 0.6,
  size = 1
)

nnFit <- train(
  x = trainX_trans,
  y = trainY,
  method = "nnet",
  tuneGrid = nnGrid,
  trControl = ctrl,
  trace = FALSE,
  maxit = 100,
  metric = "ROC"
)
```

```r
varImp(nnFit)
```

```
nnet variable importance

  only 20 most important variables shown (out of 105)

            Overall
X_AGE80      100.00
PHYSHLTH      32.68
CHECKUP1.4    31.92
X_RFHYPE6.2   25.80
X_RFHLTH.1    24.78
X_RFHYPE6.1   24.75
CHCCOPD3.7    24.09
X_RFBMI5.2    22.06
X_IMPRACE.3   21.78
X_CHOLCH3.2   20.63
PNEUVAC4.1    20.47
X_CHOLCH3.1   20.14
X_SMOKER3.9   19.87
DIFFDRES.7    19.13
EDUCA.9       17.68
X_RFBMI5.9    16.07
```

```
X_HCVU653.1   15.58
X_PACAT3.1    15.55
X_SMOKER3.4   15.45
X_RFBMI5.1    14.87
```

RESULTS

ROC AUC

```
# Logistic regressiono
lr_probs <- predict(lrFit, newdata = testX_trans, type = "prob")
lrRoc <- roc(response = testY,
             predictor = lr_probs[, "Yes"],
             levels = rev(levels(testY)))
```

```
Setting direction: controls > cases
```

```
# Penalized logistic regression
plr_probs <- predict(plrFit, newdata = testX_trans, type = "prob")
plrRoc <- roc(response = testY,
             predictor = plr_probs[, "Yes"],
             levels = rev(levels(testY)))
```

```
Setting direction: controls > cases
```

```
# Random forest
rf_probs <- predict(rfFit, newdata = testX, type = "prob")
rfRoc <- roc(response = testY,
             predictor = rf_probs[, "Yes"],
             levels = rev(levels(testY)))
```

```
Setting direction: controls > cases
```

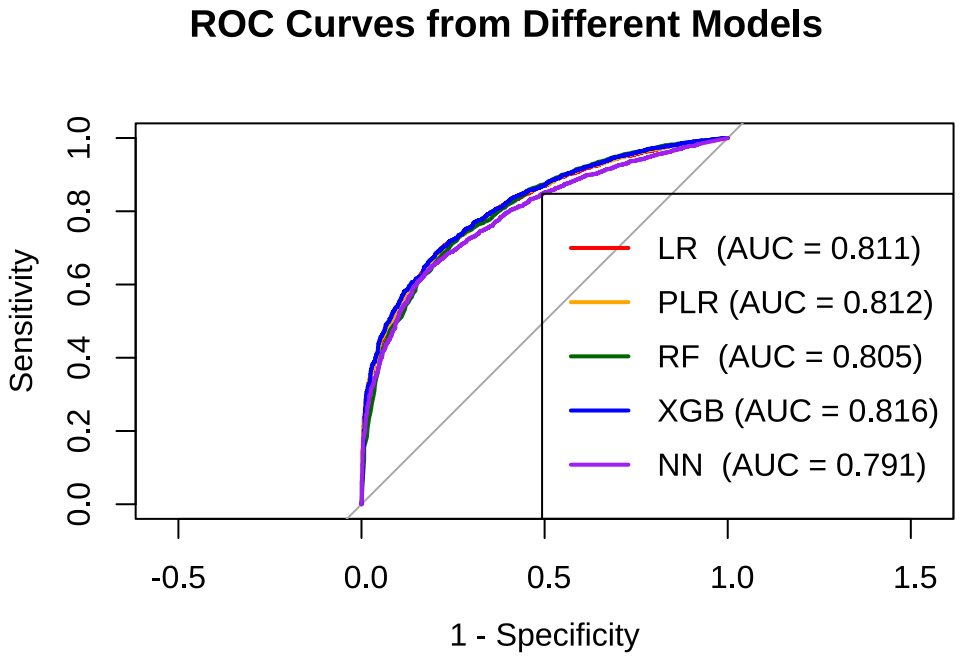
```
xgb_probs <- predict(xgbFit, newdata = testX_dummies, type = "prob")

# XGBoost
xgbRoc <- roc(response = testY,
             predictor = xgb_probs[, "Yes"],
             levels = rev(levels(testY)))
```

```
Setting direction: controls > cases
```

```
# Neural network
nn_probs <- predict(nnFit, newdata = testX_trans, type = "prob")
nnRoc <- roc(response = testY,
             predictor = nn_probs[, "Yes"],
             levels = rev(levels(testY)))
```

```
Setting direction: controls > cases
```

```
# Display ROC curves
par(oma= c(0, 0,1,0))
plot(lrRoc, col = "red", legacy.axes = TRUE)
plot(plrRoc, col = "orange", legacy.axes = TRUE, add = TRUE)
plot(rfRoc, col = "darkgreen", legacy.axes = TRUE, add = TRUE)
plot(xgbRoc, col = "blue", legacy.axes = TRUE, add = TRUE)
plot(nnRoc, col = "purple", legacy.axes = TRUE, add = TRUE)
legend("bottomright",
       legend = c(
         paste0("LR  (AUC = ", round(auc(lrRoc), 3), ")"),
         paste0("PLR (AUC = ", round(auc(plrRoc), 3), ")"),
         paste0("RF  (AUC = ", round(auc(rfRoc), 3), ")"),
         paste0("XGB (AUC = ", round(auc(xgbRoc), 3), ")"),
         paste0("NN  (AUC = ", round(auc(nnRoc), 3), ")")
       ),
       col = c("red", "orange", "darkgreen", "blue", "purple"),
       lwd = 2)
title(main = "ROC Curves from Different Models", outer = TRUE)
```



Confusion Matrix

```
lr_preds <- predict(lrFit, newdata = testX_trans)
lr_cm <- confusionMatrix(lr_preds, testY, positive = "Yes")

plr_preds <- predict(plrFit, newdata = testX_trans)
plr_cm <- confusionMatrix(plr_preds, testY, positive = "Yes")

testX_df <- as.data.frame(testX)
rf_preds <- predict(rfFit, newdata = testX_df)
rf_cm <- confusionMatrix(rf_preds, testY, positive = "Yes")

xgb_preds <- predict(xgbFit, newdata = testX_dummies)
xgb_cm <- confusionMatrix(xgb_preds, testY, positive = "Yes")

nn_preds <- predict(nnFit, newdata = testX_trans)
nn_cm <- confusionMatrix(nn_preds, testY, positive = "Yes")

lr_table   <- lr_cm$table
plr_table  <- plr_cm$table
rf_table   <- rf_cm$table
xgb_table  <- xgb_cm$table
```

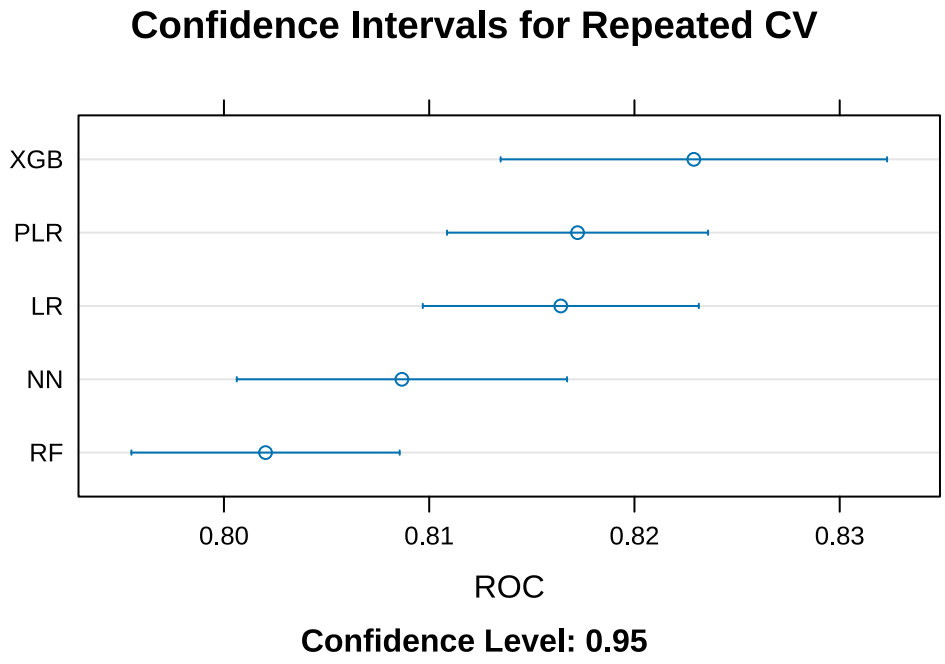
```r
nn_table   <- nn_cm$table

cm_results <- data.frame(
  Model = c("LR", "PLR", "RF", "XGB", "NN"),

  FPR = c(
    lr_table["Yes", "No"] / (lr_table["Yes", "No"] + lr_table["No", "No"]),
    plr_table["Yes", "No"] / (plr_table["Yes", "No"] + plr_table["No", "No"]),
    rf_table["Yes", "No"] / (rf_table["Yes", "No"] + rf_table["No", "No"]),
    xgb_table["Yes", "No"] / (xgb_table["Yes", "No"] + xgb_table["No", "No"]),
    nn_table["Yes", "No"] / (nn_table["Yes", "No"] + nn_table["No", "No"])
  ),

  FNR = c(
    lr_table["No", "Yes"] / (lr_table["No", "Yes"] + lr_table["Yes", "Yes"]),
    plr_table["No", "Yes"] / (plr_table["No", "Yes"] + plr_table["Yes", "Yes"]),
    rf_table["No", "Yes"] / (rf_table["No", "Yes"] + rf_table["Yes", "Yes"]),
    xgb_table["No", "Yes"] / (xgb_table["No", "Yes"] + xgb_table["Yes", "Yes"]),
    nn_table["No", "Yes"] / (nn_table["No", "Yes"] + nn_table["Yes", "Yes"])
  )
)

# Round for easier reading
cm_results$FPR <- round(cm_results$FPR, 4)
cm_results$FNR <- round(cm_results$FNR, 4)

cm_results
```

```
  Model    FPR    FNR
1    LR 0.0223 0.8410
2   PLR 0.0193 0.8565
3    RF 0.0150 0.8659
4   XGB 0.0192 0.8565
5    NN 0.0000 1.0000
```

Confidence Intervals for Cross-Validation

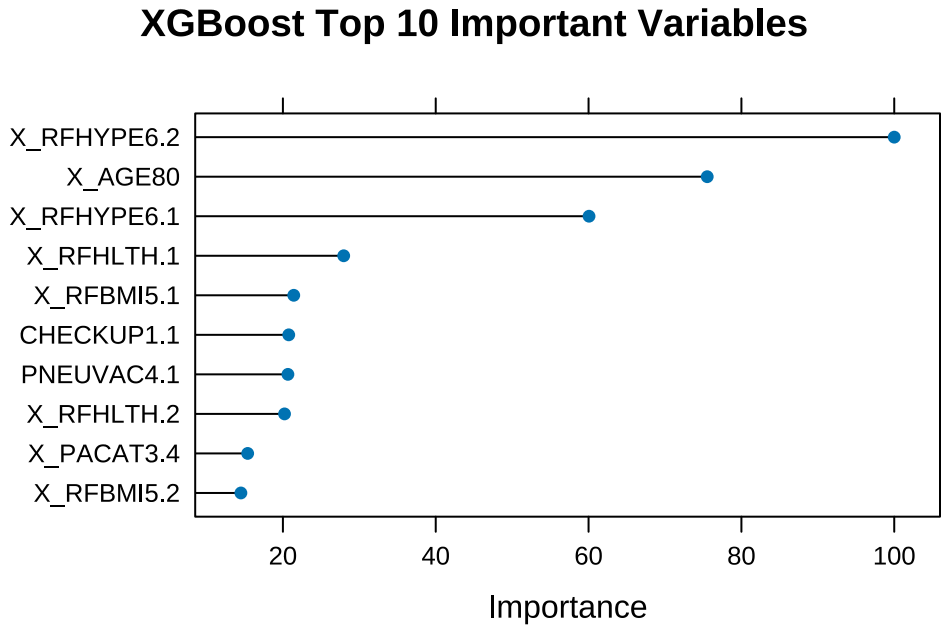
```r
train_metrics <- resamples(list(
  LR = lrFit,
  PLR = plrFit,
```

```
  RF = rfFit,
  XGB = xgbFit,
  NN = nnFit
))

dotplot(train_metrics, metric = "ROC", main = "Confidence Intervals for Repeated CV")
```



XGBoost Variable Importance

```
plot(varImp(xgbFit), top = 10, main = "XGBoost Top 10 Important Variables")
```



RShiny App (XGBoost Model)

```
colnames(trainX_dummies)
```

```
  [1] "PHYSHLTH"    "PERSDOC3.1"  "PERSDOC3.2"  "PERSDOC3.3"  "PERSDOC3.7"
  [6] "PERSDOC3.9"  "CHECKUP1.1"  "CHECKUP1.2"  "CHECKUP1.3"  "CHECKUP1.4"
 [11] "CHECKUP1.7"  "CHECKUP1.8"  "CHECKUP1.9"  "CHCOCNC1.1"  "CHCOCNC1.2"
 [16] "CHCOCNC1.7"  "CHCOCNC1.9"  "CHCCOPD3.1"  "CHCCOPD3.2"  "CHCCOPD3.7"
 [21] "CHCCOPD3.9"  "MARITAL.1"   "MARITAL.2"   "MARITAL.3"   "MARITAL.4"
 [26] "MARITAL.5"   "MARITAL.6"   "MARITAL.9"   "EDUCA.1"     "EDUCA.2"
 [31] "EDUCA.3"     "EDUCA.4"     "EDUCA.5"     "EDUCA.6"     "EDUCA.9"
 [36] "VETERAN3.1"  "VETERAN3.2"  "VETERAN3.7"  "VETERAN3.9"  "CHILDREN"
 [41] "BLIND.1"     "BLIND.2"     "BLIND.7"     "BLIND.9"     "DECIDE.1"
 [46] "DECIDE.2"    "DECIDE.7"    "DECIDE.9"    "DIFFDRES.1"  "DIFFDRES.2"
 [51] "DIFFDRES.7"  "DIFFDRES.9"  "DIFFALON.1"  "DIFFALON.2"  "DIFFALON.7"
 [56] "DIFFALON.9"  "FLUSHOT7.1"  "FLUSHOT7.2"  "FLUSHOT7.7"  "FLUSHOT7.9"
 [61] "PNEUVAC4.1"  "PNEUVAC4.2"  "PNEUVAC4.7"  "PNEUVAC4.9"  "X_IMPRACE.1"
 [66] "X_IMPRACE.2" "X_IMPRACE.3" "X_IMPRACE.4" "X_IMPRACE.5" "X_IMPRACE.6"
 [71] "X_RFHLTH.1"  "X_RFHLTH.2"  "X_RFHLTH.9"  "X_HCVU653.1" "X_HCVU653.2"
 [76] "X_HCVU653.9" "X_PACAT3.1"  "X_PACAT3.2"  "X_PACAT3.3"  "X_PACAT3.4"
 [81] "X_PACAT3.9"  "X_RFHYPE6.1" "X_RFHYPE6.2" "X_RFHYPE6.9" "X_CHOLCH3.1"
```

```
 [86] "X_CHOLCH3.2" "X_CHOLCH3.3" "X_CHOLCH3.9" "X_MICHD.1"   "X_MICHD.2"
 [91] "X_ASTHMS1.1" "X_ASTHMS1.2" "X_ASTHMS1.3" "X_ASTHMS1.9" "X_DRDXAR2.1"
 [96] "X_DRDXAR2.2" "X_AGE80"     "X_RFBMI5.1"  "X_RFBMI5.2"  "X_RFBMI5.9"
[101] "X_SMOKER3.1" "X_SMOKER3.2" "X_SMOKER3.3" "X_SMOKER3.4" "X_SMOKER3.9"
```

```
# Used ChatGPT on 6/18/25 to understand code structure of rshiny
xgb_features <- c(
  "PHYSHLTH", "PERSDOC3.1", "PERSDOC3.2", "PERSDOC3.3", "PERSDOC3.7", "PERSDOC3.9",
  "CHECKUP1.1", "CHECKUP1.2", "CHECKUP1.3", "CHECKUP1.4", "CHECKUP1.7", "CHECKUP1.8", "CHECKU
  "CHCOCNC1.1", "CHCOCNC1.2", "CHCOCNC1.7", "CHCOCNC1.9",
  "CHCCOPD3.1", "CHCCOPD3.2", "CHCCOPD3.7", "CHCCOPD3.9",
  "MARITAL.1", "MARITAL.2", "MARITAL.3", "MARITAL.4", "MARITAL.5", "MARITAL.6", "MARITAL.9",
  "EDUCA.1", "EDUCA.2", "EDUCA.3", "EDUCA.4", "EDUCA.5", "EDUCA.6", "EDUCA.9",
  "VETERAN3.1", "VETERAN3.2", "VETERAN3.7", "VETERAN3.9",
  "CHILDREN", "BLIND.1", "BLIND.2", "BLIND.7", "BLIND.9",
  "DECIDE.1", "DECIDE.2", "DECIDE.7", "DECIDE.9",
  "DIFFDRES.1", "DIFFDRES.2", "DIFFDRES.7", "DIFFDRES.9",
  "DIFFALON.1", "DIFFALON.2", "DIFFALON.7", "DIFFALON.9",
  "FLUSHOT7.1", "FLUSHOT7.2", "FLUSHOT7.7", "FLUSHOT7.9",
  "PNEUVAC4.1", "PNEUVAC4.2", "PNEUVAC4.7", "PNEUVAC4.9",
  "X_IMPRACE.1", "X_IMPRACE.2", "X_IMPRACE.3", "X_IMPRACE.4", "X_IMPRACE.5", "X_IMPRACE.6",
  "X_RFHLTH.1", "X_RFHLTH.2", "X_RFHLTH.9",
  "X_HCVU653.1", "X_HCVU653.2", "X_HCVU653.9",
  "X_PACAT3.1", "X_PACAT3.2", "X_PACAT3.3", "X_PACAT3.4", "X_PACAT3.9",
  "X_RFHYPE6.1", "X_RFHYPE6.2", "X_RFHYPE6.9",
  "X_CHOLCH3.1", "X_CHOLCH3.2", "X_CHOLCH3.3", "X_CHOLCH3.9",
  "X_MICHD.1", "X_MICHD.2",
  "X_ASTHMS1.1", "X_ASTHMS1.2", "X_ASTHMS1.3", "X_ASTHMS1.9",
  "X_DRDXAR2.1", "X_DRDXAR2.2",
  "X_AGE80",
  "X_RFBMI5.1", "X_RFBMI5.2", "X_RFBMI5.9",
  "X_SMOKER3.1", "X_SMOKER3.2", "X_SMOKER3.3", "X_SMOKER3.4", "X_SMOKER3.9"
)

ui <- fluidPage(

  # CSS
  tags$head(
  tags$style(HTML("
    .plot-box {
      border-radius: 20px;
      color: white;
```

```
    margin-right: 20px;
    margin-top: 20px;
    padding: 16px;
    box-shadow: 0 4px 20px rgba(0,0,0,0.1);
  }
")),

titlePanel(div(style = "margin-left: 20px;", "XGBoost Diabetes Prediction")),

# Layout below the title
sidebarLayout(
  sidebarPanel(

  numericInput("age", "Age:", value = 40, min = 18, max = 80),
  selectInput("hypertension", "Have you been told you have high blood pressure?",
              choices = c("Yes", "No", "I don't know")),
  selectInput("health", "Would you say your health is good or better?",
              choices = c("Yes", "No", "I don't know")),
  selectInput("bmi", "Is your BMI over 25 (overweight or obese)?",
              choices = c("Yes", "No", "I don't know")),
  selectInput("checkup", "When was your last routine checkup?",
              choices = c(
                "Within past year" = "1",
                "Within past 2 years" = "2",
                "Within past 5 years" = "3",
                "5 or more years ago" = "4",
                "Don't know / Not sure" = "7",
                "Never" = "8",
                "Refused" = "9"
              )),
  selectInput("pneumonia", "Have you ever had a pneumonia vaccine (age 65+)?",
              choices = c("Yes" = "1", "No" = "2", "I don't know or prefer not to say" = "9
  selectInput("cholesterol", "Have you had your cholesterol checked in the past 5 years?",
              choices = c(
                "Yes, within the past 5 years" = "1",
                "No, not within the past 5 years" = "2",
                "Never had it checked" = "3",
                "I don't know or prefer not to say" = "9"
              )),
  selectInput("activity", "What best describes your physical activity level?",
              choices = c(
                "Highly active" = "1",
```

```
                  "Active" = "2",
                  "Insufficiently active" = "3",
                  "Inactive" = "4",
                  "I don't know" = "9"
                )),
    selectInput("diffwalk", "Do you have serious difficulty walking or climbing stairs?",
                choices = c("Yes" = "1", "No" = "2", "I don't know" = "7", "I refuse to answe
    selectInput("heart", "Have you ever had coronary heart disease or a heart attack?",
                choices = c("Yes" = "1", "No" = "2")),
    selectInput("race", "What is your race/ethnicity?",
                choices = c(
                  "White" = "1",
                  "Black" = "2",
                  "Asian" = "3",
                  "American Indian or Alaska Native" = "4",
                  "Hispanic" = "5",
                  "Other" = "6"
                ))
    ),

    mainPanel(
      h3("Prediction Results"),
      div(class = "plot-box",
        plotOutput("probBarPlot")
      ),
      div(class = "plot-box",
        plotOutput("importancePlot")
      )
    )
  ))
)

server <- function(input, output) {

  prediction_data <- reactive({
    newdata <- as.data.frame(matrix(0, nrow = 1, ncol = length(xgb_features)))
    colnames(newdata) <- xgb_features
    newdata$X_AGE80 <- input$age

    # Hypertension
    if (input$hypertension == "Yes") newdata$X_RFHYPE6.2 <- 1
    else if (input$hypertension == "No") newdata$X_RFHYPE6.1 <- 1
```

```
  else newdata$X_RFHYPE6.9 <- 1

  # Health
  if (input$health == "Yes") newdata$X_RFHLTH.1 <- 1
  else if (input$health == "No") newdata$X_RFHLTH.2 <- 1
  else newdata$X_RFHLTH.9 <- 1

  # BMI
  if (input$bmi == "Yes") newdata$X_RFBMI5.2 <- 1
  else if (input$bmi == "No") newdata$X_RFBMI5.1 <- 1
  else newdata$X_RFBMI5.9 <- 1

  # Checkup
  newdata[[paste0("CHECKUP1.", input$checkup)]] <- 1

  # Pneumonia
  newdata$PNEUVAC4.7 <- 0
  newdata[[paste0("PNEUVAC4.", input$pneumonia)]] <- 1

  # Cholesterol
  if (input$cholesterol == "1") newdata$X_CHOLCH3.1 <- 1
  else if (input$cholesterol == "2") newdata$X_CHOLCH3.2 <- 1
  else if (input$cholesterol == "3") newdata$X_CHOLCH3.3 <- 1
  else if (input$cholesterol == "9") newdata$X_CHOLCH3.9 <- 1

  # Activity
  newdata[[paste0("X_PACAT3.", input$activity)]] <- 1

  # Difficulty walking
  newdata[[paste0("DIFFWALK.", input$diffwalk)]] <- 1

  # CHD/MI
  if (input$heart == "1") newdata$X_MICHD.1 <- 1
  else newdata$X_MICHD.2 <- 1

  # Race/Ethnicity
  newdata[[paste0("X_IMPRACE.", input$race)]] <- 1

  newdata
})

prediction <- reactive({
```

```
    req(xgbFit)
    pred <- predict(xgbFit, newdata = prediction_data(), type = "prob")
    pred
  })

  output$probBarPlot <- renderPlot({
    pred <- prediction()
    probs <- as.numeric(pred[1, c("Yes", "No")])
    bar_names <- c("Likely Has Diabetes", "Unlikely Has Diabetes")
    bar_colors <- c("red", "green")

    bar_locs <- barplot(
      height = probs,
      names.arg = bar_names,
      col = bar_colors,
      ylim = c(0, 1),
      ylab = "Probability"
    )

    title(main = "Predicted Probability of Diabetes")

    text(
      x = bar_locs,
      y = probs + 0.05,
      labels = paste0(round(probs * 100, 1), "%"),
      cex = 1.2
    )
  })

  output$importancePlot <- renderPlot({
    plot(varImp(xgbFit), top = 10, main = "Top 10 Important Features")
  })
}

shinyApp(ui = ui, server = server)
```

REFERENCES

Geeks for Geeks (2024a). *How to Calculate Cramer's V in R*. GeeksforGeeks. Retrieved June 4, 2025, from https://www.geeksforgeeks.org/how-to-calculate-cramers-v-in-r/

Geeks for Geeks (2024b). *Stratified Sampling in R*. GeeksforGeeks. Retrieved June 20, 2025 from https://www.geeksforgeeks.org/r-language/stratified-sampling-in-r/