209 Project Proposal

Shijie Mao (smao027@ucr.edu), Kaiming Wang (kwang280@ucr.edu)

Research Objectives

What questions are you trying to answer? (Interested Questions)

Compared to traditional regression-based approaches, the use of modern machine learning techniques—such as Support Vector Machines (SVM), XGBoost, or even deep learning models like Convolutional Neural Networks (CNNs) and Transformers—has become increasingly prevalent in predictive modeling. In this final project, I aim to apply various modeling methods to healthcare data in order to systematically compare their predictive performance and interpretability, with a focus on metrics such as AUC. Specifically, the following sub-questions will be investigated:

1. **Variable Contribution and Interactions**
   Which variables contribute most significantly to the prediction of clinical events (e.g., disease occurrence)? Are there important interaction effects between risk factors (e.g., synergistic effects such as hypertension and high BMI)?

2. **Population Heterogeneity and Clustering**
   Do risk patterns vary across different subpopulations (e.g., by gender or age groups)? Can dimensionality reduction and clustering techniques (e.g., PCA) be employed to identify latent structures or risk profiles among patients, and further stratify them into low-, medium-, or high-risk categories?

3. **Model Comparison**
   How do different models—from traditional statistical models to machine learning and deep learning approaches—compare in terms of training/test AUC, interpretability, and robustness? What trade-offs exist between predictive performance and model transparency?

4. **Causal Inference for Lifestyle Interventions**
   Among modifiable variables, under what conditions should patients change certain behaviors (e.g., diet, exercise) to optimally reduce their risk of developing conditions such as diabetes? Based on model outputs, what personalized recommendations can be made to support preventive healthcare?

5. **Transfer Learning for Small Sample Settings**
   Can large-scale models trained on extensive datasets be fine-tuned or transferred to benefit smaller datasets? Specifically, does fine-tuning pre-trained deep learning models (e.g., LLMs or domain-specific Transformers) outperform training separate models on small-sample datasets?

What datasets will you use? (Including EDA)

Our **Diabetes prediction dataset** is from Kaggle:

https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset/data

The **dataset** is a collection of medical and demographic data from patients, along with their diabetes status (positive or negative). Various features include age, gender, body mass index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level. The dataset can be utilized to predict the likelihood of diabetes in patients based on their medical history and demographic details. This can be useful for healthcare professionals in identifying patients who may be at risk of developing diabetes and in developing personalized treatment plans. Additionally, the dataset can be used by researchers to explore the relationships between various medical and demographic factors and the likelihood of developing diabetes.

EDA

Dataset (At A Glance)

```r
diabetes_data <- read_csv(here("diabetes_prediction_dataset.csv"))
```

```
Rows: 100000 Columns: 9
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): gender, smoking_history
dbl (7): age, hypertension, heart_disease, bmi, HbA1c_level, blood_glucose_l...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
glimpse(diabetes_data)
```

```
Rows: 100,000
Columns: 9
$ gender              <chr> "Female", "Female", "Male", "Female", "Male", "Fem~
$ age                 <dbl> 80, 54, 28, 36, 76, 20, 44, 79, 42, 32, 53, 54, 78~
$ hypertension        <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ heart_disease       <dbl> 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ smoking_history     <chr> "never", "No Info", "never", "current", "current",~
$ bmi                 <dbl> 25.19, 27.32, 27.32, 23.45, 20.14, 27.32, 19.31, 2~
$ HbA1c_level         <dbl> 6.6, 6.6, 5.7, 5.0, 4.8, 6.6, 6.5, 5.7, 4.8, 5.0, ~
$ blood_glucose_level <dbl> 140, 80, 158, 155, 155, 85, 200, 85, 145, 100, 85,~
$ diabetes            <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

```
head(diabetes_data)
```

```
# A tibble: 6 x 9
  gender   age hypertension heart_disease smoking_history   bmi HbA1c_level
  <chr>  <dbl>        <dbl>         <dbl> <chr>           <dbl>       <dbl>
1 Female    80            0             1 never            25.2         6.6
2 Female    54            0             0 No Info          27.3         6.6
3 Male      28            0             0 never            27.3         5.7
4 Female    36            0             0 current          23.4         5  
5 Male      76            1             1 current          20.1         4.8
6 Female    20            0             0 never            27.3         6.6
# i 2 more variables: blood_glucose_level <dbl>, diabetes <dbl>
```

Missing Data

```
missing_values <- colSums(is.na(diabetes_data))
print(missing_values)
```

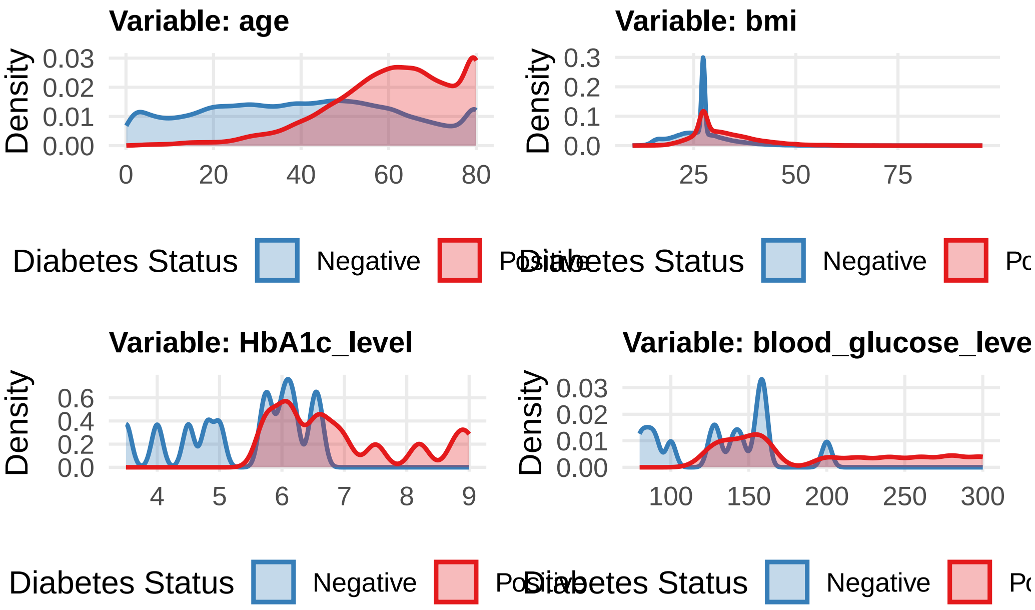
```
             gender                 age        hypertension       heart_disease 
                  0                   0                   0                   0 
    smoking_history                 bmi         HbA1c_level blood_glucose_level 
                  0                   0                   0                   0 
           diabetes 
                  0 
```

Analysis

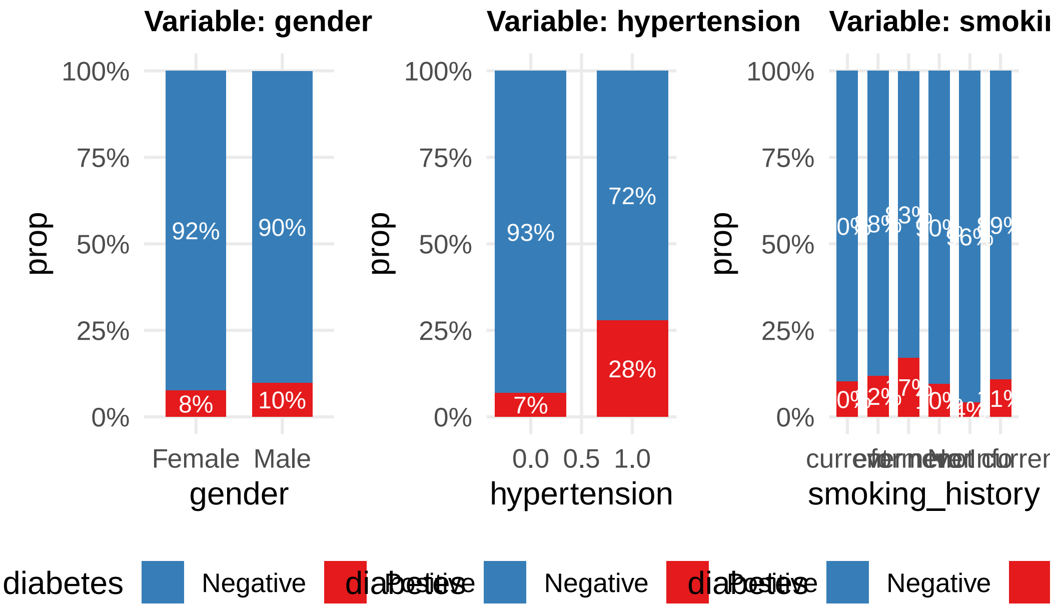
```
#library(ggplot2)
#library(patchwork)


diabetes_colors <- c("positive" = "#E41A1C", "negative" = "#377EB8")

diabetes_data <- diabetes_data %>%
     filter(gender != "Other" & hypertension!="0.5") %>%
     mutate(diabetes = factor(
         diabetes,
         levels = c(0, 1),
         labels = c("negative", "positive")
     ))

# ======================
# Modified Function: Overlaid density plots
# ======================
plot_univariate <- function(data, var, var_type) {

  base_plot <- ggplot(data) +
    labs(x = NULL, y = "Density", fill = "Diabetes Status", color = "Diabetes Status") +
    theme_minimal(base_size = 12) +
    theme(legend.position = "bottom",
          panel.grid.minor = element_blank(),
          plot.title = element_text(face = "bold", size = 11))

  # Numerical variables: Overlaid density plots
  if (var_type == "numeric") {
    p <- base_plot +
      geom_density(
        aes(x = .data[[var]], color = diabetes, fill = diabetes),
        alpha = 0.3,  # Fill transparency
        linewidth = 0.8  # Line thickness
      ) +
      scale_color_manual(
        values = diabetes_colors,
        labels = c("positive" = "Positive", "negative" = "Negative")
      ) +
      scale_fill_manual(
```

```r
        values = diabetes_colors,
        labels = c("positive" = "Positive", "negative" = "Negative")
      ) +
      labs(title = paste("Variable:", var))
  }

  # Categorical variables: Stacked proportion bars (unchanged)
  else if (var_type == "categorical") {
    p <- data %>%
      group_by(.data[[var]], diabetes) %>%
      summarise(n = n(), .groups = "drop") %>%
      group_by(.data[[var]]) %>%
      mutate(prop = n / sum(n)) %>%
      ggplot(aes(x = .data[[var]], y = prop, fill = diabetes)) +
      geom_col(position = "stack", width = 0.7) +
      geom_text(
        aes(label = scales::percent(prop, accuracy = 1)),
        position = position_stack(vjust = 0.5),
        size = 3, color = "white"
      ) +
      labs(title = paste("Variable:", var)) +
      scale_y_continuous(labels = scales::percent) +
      scale_fill_manual(
        values = diabetes_colors,
        labels = c("positive" = "Positive", "negative" = "Negative")
      ) +
      theme_minimal(base_size = 12) +
      theme(legend.position = "bottom",
            panel.grid.minor = element_blank(),
            plot.title = element_text(face = "bold", size = 11))
  }

  return(p)
}

# ======================
# Generate plots
# ======================
numeric_vars <- c("age", "bmi", "HbA1c_level", "blood_glucose_level")
categorical_vars <- c("gender", "hypertension", "smoking_history")
```

```r
num_plots <- map(numeric_vars, ~ plot_univariate(diabetes_data, .x, "numeric"))
cat_plots <- map(categorical_vars, ~ plot_univariate(diabetes_data, .x, "categorical"))

# Combine numerical plots (4 variables)
(num_plots[[1]] | num_plots[[2]]) / (num_plots[[3]] | num_plots[[4]])
```



```r
# Combine categorical plots (3 variables)
wrap_plots(cat_plots, ncol = 3)
```



Alternative Strategies / Backup Plans

1. Interpreting Complex Models (especially deep learning)

For the model comparison component, which focuses on evaluating different approaches, it is expected to proceed smoothly as this part is primarily application-based. However, for deep learning models, there is currently no universally accepted method for interpretation. To address this, we plan to review existing literature and attempt to interpret all possible results as thoroughly as possible. At the very least, we will apply model-X knockoff methods for variable selection. While this approach may not provide an unbiased estimator like traditional regression models, it can still offer some interpretative insights.

2. Clustering and Risk Stratification May Not Be Effective

Regarding the clustering analysis, I remain skeptical about whether the data will yield meaningful clusters. This heavily depends on the dataset itself. I plan to first try some traditional methods such as K-means, as well as relatively novel techniques like PCA and UMAP. If these methods do not produce clear or significant results, I will choose not to include the clustering component in the final analysis.

3. Fine-Tuning Pretrained Models May Be Difficult

The fine-tuning of large models is also a very interesting direction, but I do not have full confidence in its feasibility. As a first step, I plan to locally deploy and fine-tune some smaller pre-trained models. If this does not work well, my backup plan is to use the ChatGPT API for fine-tuning. Given the scale of ChatGPT, its performance after fine-tuning should not be too poor.