Final

Hrishabh Kulkarni

2025-04-28

SVM Diabetes Prediction Project

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(e1071)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tictoc)
```

```
## Warning: package 'tictoc' was built under R version 4.4.3
```

```
library(ggplot2)
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.4.3
```

1. DATA LOADING & CLEANING

```
raw_data <- read_csv("nhis_2022.csv")
```

```
## Rows: 35115 Columns: 48
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr  (3): NHISHID, NHISPID, HHX
## dbl (45): YEAR, SERIAL, STRATA, PSU, REGION, PERNUM, SAMPWEIGHT, ASTATFLG, C...
##
## 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.
```

```
# renaming variables
clean_data <- raw_data %>%
  select(DIABETICEV, HRSLEEP, BMICALC, MOD10DMIN, VEGENO, AGE, SEX) %>%
  rename(
    diabetes_status = DIABETICEV,
    sleep_hours = HRSLEEP,
    bmi = BMICALC,
    exercise_minutes = MOD10DMIN,
    vegetable_servings = VEGENO,
    age = AGE,
    sex = SEX
  ) %>%

# Droping unknown/missing values
  mutate(
    diabetes_status = ifelse(diabetes_status %in% c(7,8,9), NA, diabetes_status),
    sleep_hours = ifelse(sleep_hours %in% c(97,98,99), NA, sleep_hours),
    bmi = ifelse(bmi %in% c(000,996), NA, bmi),
    exercise_minutes = ifelse(exercise_minutes %in% c(000,996,997,998,999), NA, exercise_minutes),
    vegetable_servings = ifelse(vegetable_servings %in% c(996,997,998,999), NA, vegetable_servings),
    age = ifelse(age %in% c(997,998,999), NA, age),
    sex = ifelse(sex %in% c(7,8,9), NA, sex)
  ) %>%

  drop_na() %>%

  # Used Factor to convert to proper data types
  mutate(
    diabetes_status = factor(diabetes_status, labels = c("No", "Yes")),
    sex = factor(sex, levels = 1:2, labels = c("Male", "Female")),
    bmi_category = cut(bmi,
                       breaks = c(0, 18.5, 25, 30, Inf),
                       labels = c("Underweight", "Normal", "Overweight", "Obese"))
  )
```

2. DATA PREPARATION

```
# 2.1: checking class distribution
cat("Class distribution:\n")
```

```
## Class distribution:
```

```
original_dist <- table(clean_data$diabetes_status)
print(original_dist)
```

```
##
##    No   Yes
## 15839  1376
```

```r
# 2.2: Create balanced training set
set.seed(123)
class_counts <- clean_data %>% count(diabetes_status)
minority_count <- min(class_counts$n)
majority_count <- max(class_counts$n)

# 2.3: Downsampling majority class
majority_data <- clean_data %>% filter(diabetes_status == "No") %>% sample_n(size = minority_count)
minority_data <- clean_data %>% filter(diabetes_status == "Yes")
balanced_data <- bind_rows(majority_data, minority_data)
cat("\nBalanced dataset distribution:\n")
```

```
##
## Balanced dataset distribution:
```

```r
print(table(balanced_data$diabetes_status))
```

```
##
##   No  Yes
## 1376 1376
```

```r
# 2.4: train/test split from balanced data
train <- createDataPartition(balanced_data$diabetes_status, p = 0.7, list = FALSE)
train_data <- balanced_data[train, ]
test <- createDataPartition(clean_data$diabetes_status, p = 0.3, list = FALSE)
test_data <- clean_data[test, ]

# 2.5: Scale features
preproc <- preProcess(train_data[, c("sleep_hours", "bmi", "exercise_minutes", "vegetable_servings", "ag
train_scaled <- predict(preproc, train_data)
test_scaled <- predict(preproc, test_data)

# 2.6: Final counts
cat("\nFinal training set counts(70% of balanced data):\n")
```

```
##
## Final training set counts(70% of balanced data):
```

```r
print(table(train_scaled$diabetes_status))
```

```
##
##  No Yes
## 964 964
```

```r
cat("\nFinal Test set counts (30% of original data):\n")
```

```
##
## Final Test set counts (30% of original data):
```

```r
print(table(test_scaled$diabetes_status))
```

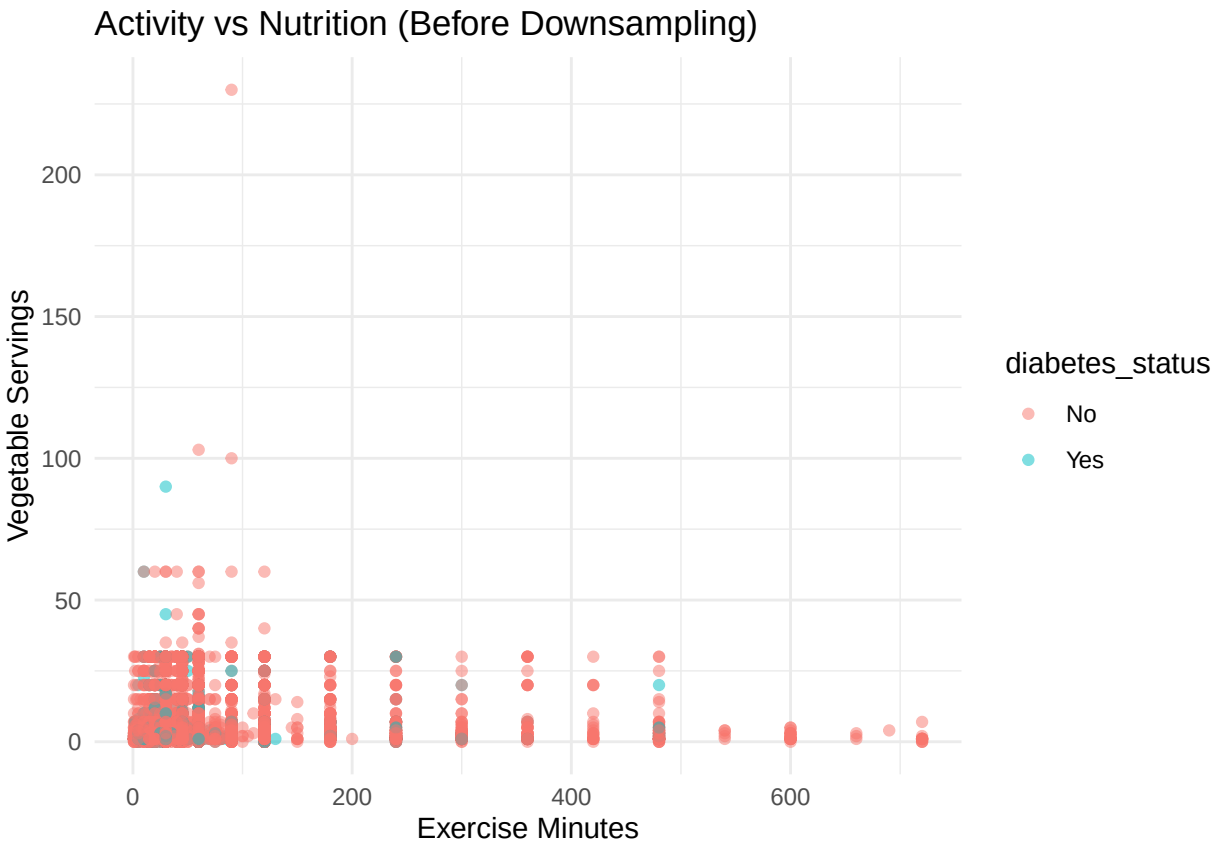
```
##
##   No  Yes
## 4752  413
```

Comments: - Here, I have done downsampling to balance the dataset. - The original dataset had a class imbalance with approx. 90% of the data being “No” and 100% being “Yes”. - After downsampling, I balanced the dataset, then split it into training and test sets. - For training set on which SVM always works, I used 70%

of the balanced data. - For the test set, I used 30% of the original data to evaluate the model's performance on unseen data.

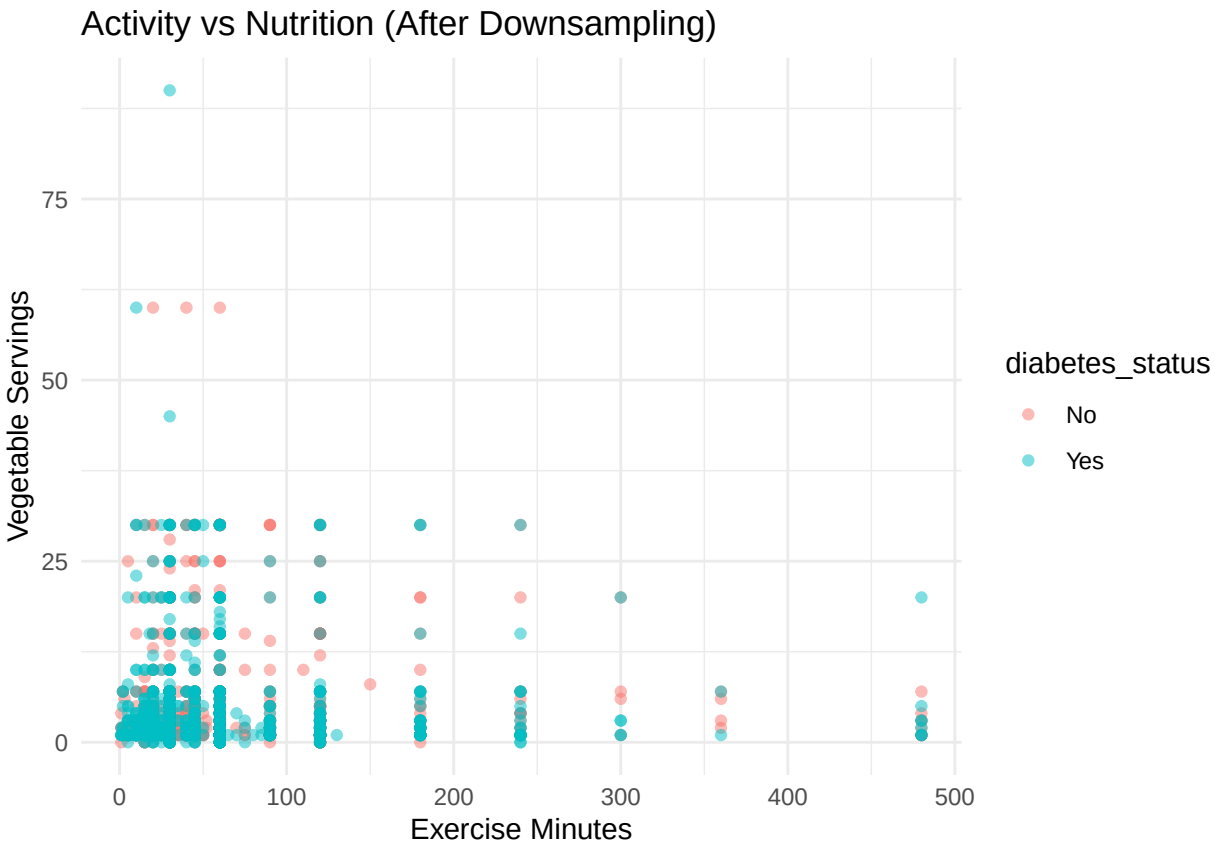
3. EXPLORATORY DATA ANALYSIS

```
# 3.1: Activity vs Nutrition (Before Downsampling)
p_activity <- ggplot(clean_data, aes(x = exercise_minutes, y = vegetable_servings, color = diabetes_sta

p_activity
```



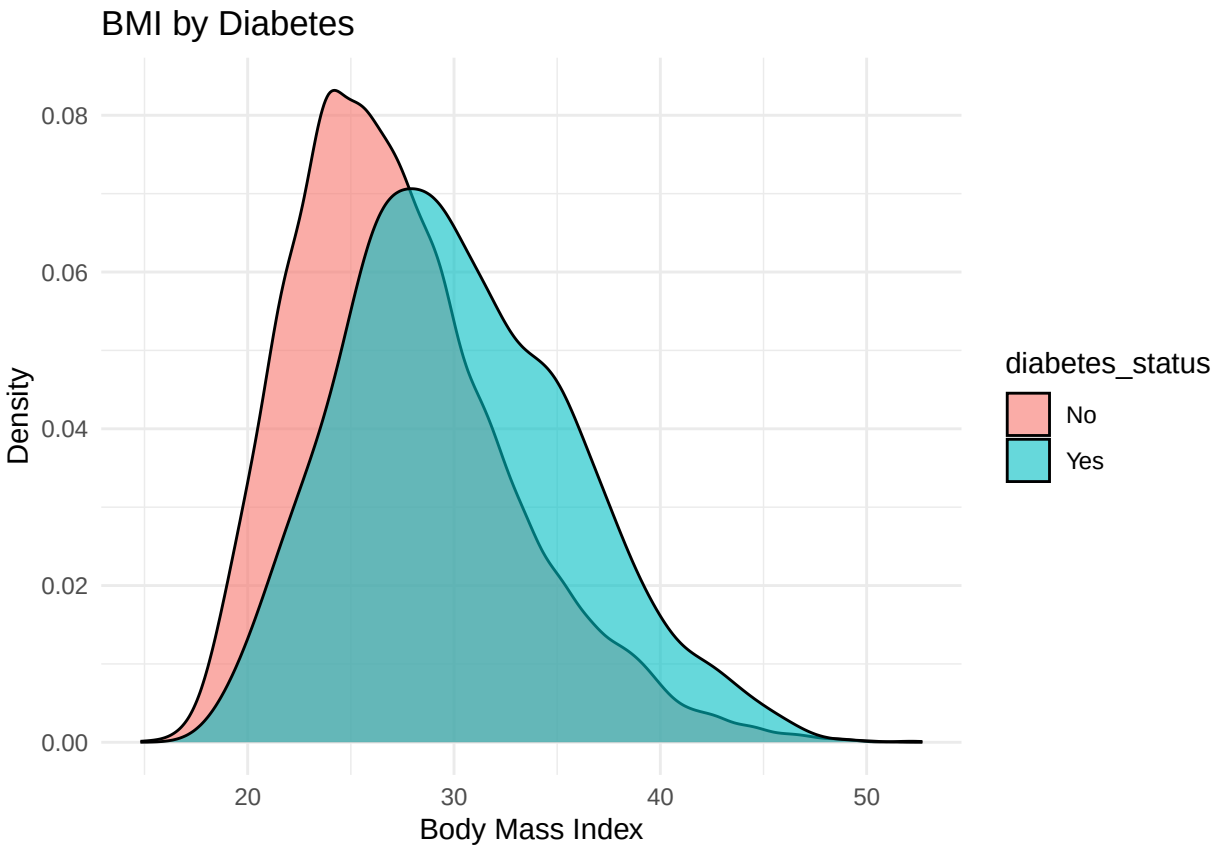
Comments: - The scatter plot shows the relationship between exercise minutes and vegetable servings, colored by diabetes status. - It appears that individuals with diabetes tend to have lower exercise minutes and vegetable servings compared to those without diabetes. - This suggests a potential correlation between physical activity, nutrition, and diabetes status. - The plot also indicates that there is some overlap between the two groups, indicating that not all individuals with low exercise and vegetable servings have diabetes.

```
# 3.2: Activity vs Nutrition (After Downsampling)
p_activity <- ggplot(balanced_data, aes(x = exercise_minutes, y = vegetable_servings, color = diabetes_

p_activity
```



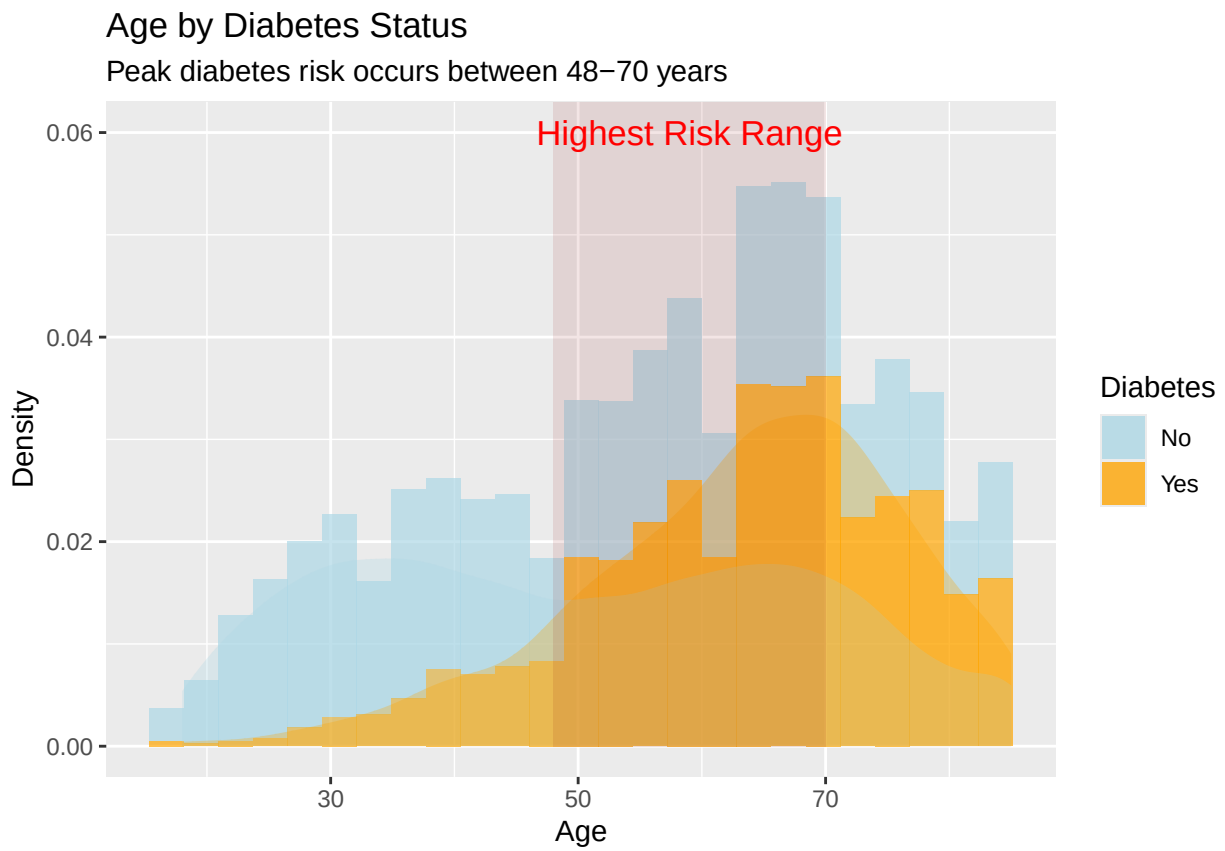
Comments: - The scatter plot shows the relationship between exercise minutes and vegetable servings, colored by diabetes status. - After downsampling, the distribution of exercise minutes and vegetable servings appears more balanced between the two groups. - The plot indicates that individuals with diabetes still tend to have lower exercise minutes and vegetable servings compared to those without diabetes. - However, the overlap between the two groups is reduced, suggesting that downsampling has helped to clarify the relationship between these variables and diabetes status.

```
# 3.3: BMI by Diabetes
p_bmi <- ggplot(clean_data, aes(x = bmi, fill = diabetes_status)) + geom_density(alpha = 0.6) + labs(ti

p_bmi
```



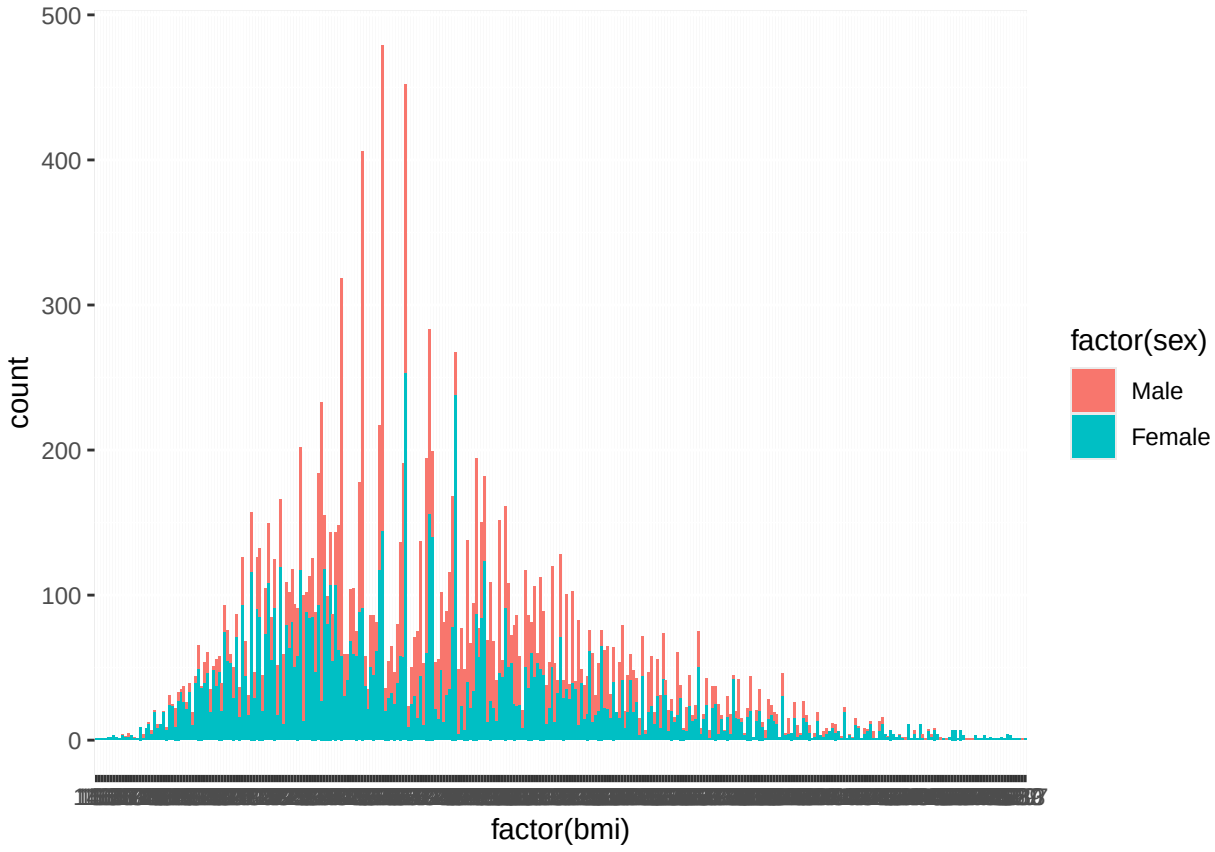
Comments: - The density plot shows the distribution of BMI for individuals with and without diabetes. - Individuals with diabetes tend to have higher BMI values compared to those without diabetes.

```
# 3.4: Age based on Diabetes

ggplot(clean_data, aes(x = age, fill = diabetes_status)) + geom_histogram(aes(y = after_stat(density)),
```



Comments: - The histogram and density plot show the distribution of age for individuals with and without diabetes. - The plot indicates that the peak diabetes risk occurs between the ages of 48 and 70 years. - The rectangle highlights age range and the annotation emphasizes the highest risk range. - This suggests that age is a significant factor in diabetes risk, with older individuals being more likely to have diabetes.

```
# 3.5: BMI by Gender
ggplot(clean_data, aes(factor(bmi), fill = factor(sex))) + geom_bar()
```



Comments: - This plot tells me distribution of BMI based on gender. - It also shows that dataset have higher number data related to men than women.

4. MODEL TRAINING & EVALUATION

```
evaluate_model <- function(model, test_data) {
  pred <- predict(model, test_data)
  prob <- attr(predict(model, test_data, decision.values = TRUE), "decision.values")
# Confusion Matrix
  cm <- confusionMatrix(pred, test_data$diabetes_status, positive = "Yes")
  return(list(confusion = cm))
}
```

4.1 Linear Kernel SVM

```
tic("Linear SVM")
svm_linear <- svm(diabetes_status ~ bmi + exercise_minutes + vegetable_servings + sleep_hours + age + s
linear_time <- toc()
```

```
## Linear SVM: 0.54 sec elapsed
```

```
linear_results <- evaluate_model(svm_linear, test_scaled)
```

```
# Linear Error rates
linear_train_pred <- predict(svm_linear, train_scaled)
linear_test_pred <- predict(svm_linear, test_scaled)
```

```
linear_train_error <- mean(linear_train_pred != train_scaled$diabetes_status)
linear_test_error <- mean(linear_test_pred != test_scaled$diabetes_status)
```

4.2 Radial Kernel SVM

```
# Tune parameters first
set.seed(1)
tune_radial <- tune(svm, diabetes_status ~ bmi + exercise_minutes + vegetable_servings + sleep_hours + a
tic("Radial SVM")
svm_radial <- tune_radial$best.model
radial_time <- toc()
```

```
## Radial SVM: 0.02 sec elapsed
```

```
radial_results <- evaluate_model(svm_radial, test_scaled)
```

```
# Radial Error rates
radial_train_pred <- predict(svm_radial, train_scaled)
radial_test_pred <- predict(svm_radial, test_scaled)
radial_train_error <- mean(radial_train_pred != train_scaled$diabetes_status)
radial_test_error <- mean(radial_test_pred != test_scaled$diabetes_status)
```

4.3 Polynomial Kernel SVM

```
tic("Polynomial SVM")
svm_poly <- svm(diabetes_status ~ bmi + exercise_minutes + vegetable_servings + sleep_hours + age + sex
poly_time <- toc()
```

```
## Polynomial SVM: 0.42 sec elapsed
```

```
poly_results <- evaluate_model(svm_poly, test_scaled)
```

```
# Polynomial Error rates
poly_train_pred <- predict(svm_poly, train_scaled)
poly_test_pred <- predict(svm_poly, test_scaled)
poly_train_error <- mean(poly_train_pred != train_scaled$diabetes_status)
poly_test_error <- mean(poly_test_pred != test_scaled$diabetes_status)
```

5. MODEL RESULT'S COMPARISON

```
# 5.1: Accuracy, Train_Error, Test_Error, Training_Time, Specificity, Sensitivity

results_summary <- tibble(
  Model = c("Linear", "Radial", "Polynomial"),
  Accuracy = c(linear_results$confusion$overall["Accuracy"],
               radial_results$confusion$overall["Accuracy"],
               poly_results$confusion$overall["Accuracy"]),
  Train_Error = c(linear_train_error, radial_train_error, poly_train_error),
  Test_Error = c(linear_test_error, radial_test_error, poly_test_error),
  Training_Time = c(linear_time$toc - linear_time$tic,
                    radial_time$toc - radial_time$tic,
                    poly_time$toc - poly_time$tic),
  Specificity = c(linear_results$confusion$byClass["Specificity"],
                  radial_results$confusion$byClass["Specificity"],
```

```
                poly_results$confusion$byClass["Specificity"]),
  Sensitivity = c(linear_results$confusion$byClass["Sensitivity"],
                radial_results$confusion$byClass["Sensitivity"],
                poly_results$confusion$byClass["Sensitivity"])
)

print(results_summary)
```

```
## # A tibble: 3 x 7
##   Model    Accuracy Train_Error Test_Error Training_Time Specificity Sensitivity
##   <chr>       <dbl>       <dbl>      <dbl>         <dbl>       <dbl>       <dbl>
## 1 Linear      0.631       0.310      0.369         0.540       0.617       0.797
## 2 Radial      0.614       0.307      0.386         0.0200      0.598       0.801
## 3 Polynom~    0.595       0.304      0.405         0.420       0.575       0.826
```

Findings: - Model Accuracy Train_Error Test_Error Training_Time Specificity Sensitivity - Linear 0.6313650 0.3101660 0.3686350 0.45 0.6170034 0.7966102 - Radial 0.6139400 0.3065353 0.3860600 0.06 0.5976431 0.8014528 - Polynomial 0.5953533 0.3044606 0.4046467 0.52 0.5753367 0.8256659

Comments: - The linear kernel SVM achieved the highest accuracy (63.14%) and sensitivity (79.66%) but had a longer training time (0.45 seconds). - The radial kernel SVM had a similar sensitivity (80.15%) and a much shorter training time (0.06 seconds). - The polynomial kernel SVM had the lowest accuracy (59.54%) but the highest sensitivity (82.57%) and a highest training time of 0.52 seconds. - The linear kernal SVM is best for this dataset.

```
# Metrics calculation function

get_class_metrics <- function(model, train_data, test_data) {
  pred_train <- predict(model, train_data)
  pred_test <- predict(model, test_data)
  prob_train <- attr(predict(model, train_data, decision.values = TRUE), "decision.values")
  prob_test <- attr(predict(model, test_data, decision.values = TRUE), "decision.values")

  cm_train <- confusionMatrix(pred_train, train_data$diabetes_status, positive = "Yes")
  cm_test <- confusionMatrix(pred_test, test_data$diabetes_status, positive = "Yes")



  list(
    Train = with(cm_train, c(Accuracy = overall["Accuracy"],
                             Precision = byClass["Precision"],
                             Recall = byClass["Recall"],
                             F1 = byClass["F1"],
                             AUC = as.numeric(performance(
                               prediction(prob_train, as.numeric(train_data$diabetes_status)-1),
                               "auc")@y.values),
                             SVs = length(model$index))),


    Test = with(cm_test, c(Accuracy = overall["Accuracy"],
                           Precision = byClass["Precision"],
                           Recall = byClass["Recall"],
                           F1 = byClass["F1"],
                           AUC = as.numeric(performance(
                             prediction(prob_test, as.numeric(test_data$diabetes_status)-1),
```

```
                              "auc")@y.values)))
  )
}
```

```
# 5.2: Training and Test - Accuracy, Precision, Recall, F1, AUC

models <- list(Linear = svm_linear, Radial = svm_radial, Polynomial = svm_poly)
metrics <- map(models, ~get_class_metrics(.x, train_scaled, test_scaled))

train_summary <- map_dfr(metrics, ~as.data.frame(t(.x$Train)), .id = "Model")
test_summary <- map_dfr(metrics, ~as.data.frame(t(.x$Test)), .id = "Model")



cat("TEST METRICS \n")
```

```
## TEST METRICS
```

```
print(train_summary)
```

```
##        Model Accuracy.Accuracy Precision.Precision Recall.Recall     F1.F1
## 1     Linear         0.6898340           0.6602452     0.7821577 0.7160494
## 2     Radial         0.6934647           0.6592656     0.8008299 0.7231850
## 3 Polynomial         0.6955394           0.6564315     0.8205394 0.7293684
##         AUC  SVs
## 1 0.2455811 1348
## 2 0.2379387 1459
## 3 0.2278300 1345
```

```
cat("\n TRAINING METRICS \n")
```

```
##
##  TRAINING METRICS
```

```
print(test_summary)
```

```
##        Model Accuracy.Accuracy Precision.Precision Recall.Recall     F1.F1
## 1     Linear         0.6313650           0.1530945     0.7966102 0.2568306
## 2     Radial         0.6139400           0.1475702     0.8014528 0.2492470
## 3 Polynomial         0.5953533           0.1445528     0.8256659 0.2460317
##         AUC
## 1 0.2377882
## 2 0.2360069
## 3 0.2324254
```

Findings:

-> Test Metrics - Model Accuracy Precision Recall F1 AUC SVs - Linear 0.6898340 0.6602452 0.7821577 0.7160494 0.2455811 1348 - Radial 0.6934647 0.6592656 0.8008299 0.7231850 0.2379387 1459 - Polynomial 0.6955394 0.6564315 0.8205394 0.7293684 0.2278300 1345

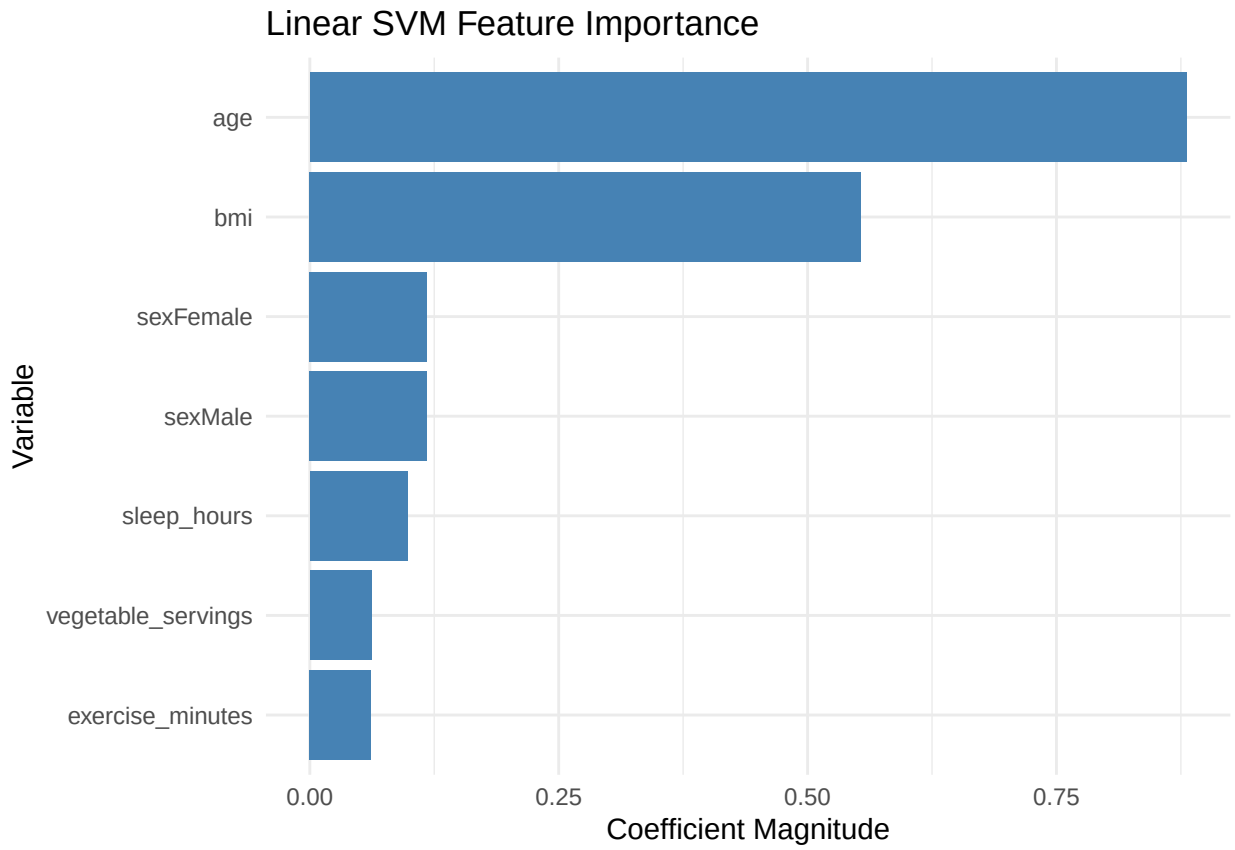
-> Training Metrics - Model Accuracy Precision Recall F1 AUC - Linear 0.6313650 0.1530945 0.7966102 0.2568306 0.2377882 - Radial 0.6139400 0.1475702 0.8014528 0.2492470 0.2360069 - Polynomial 0.5953533 0.1445528 0.8256659 0.2460317 0.2324254

Comments: - The training metrics show that the linear kernel SVM has the highest accuracy (63.14%) and F1 score (25.68%). - The test metrics show that the polynomial kernel SVM has the highest accuracy (69.55%) and F1 score (72.94%).

6. VARIABLE IMPORTANCE ANALYSIS

```r
# Linear kernel
linear_coefs <- t(svm_linear$coefs) %*% svm_linear$SV
linear_importance <- data.frame( Variable = colnames(svm_linear$SV), Importance = abs(linear_coefs[1, ])
```

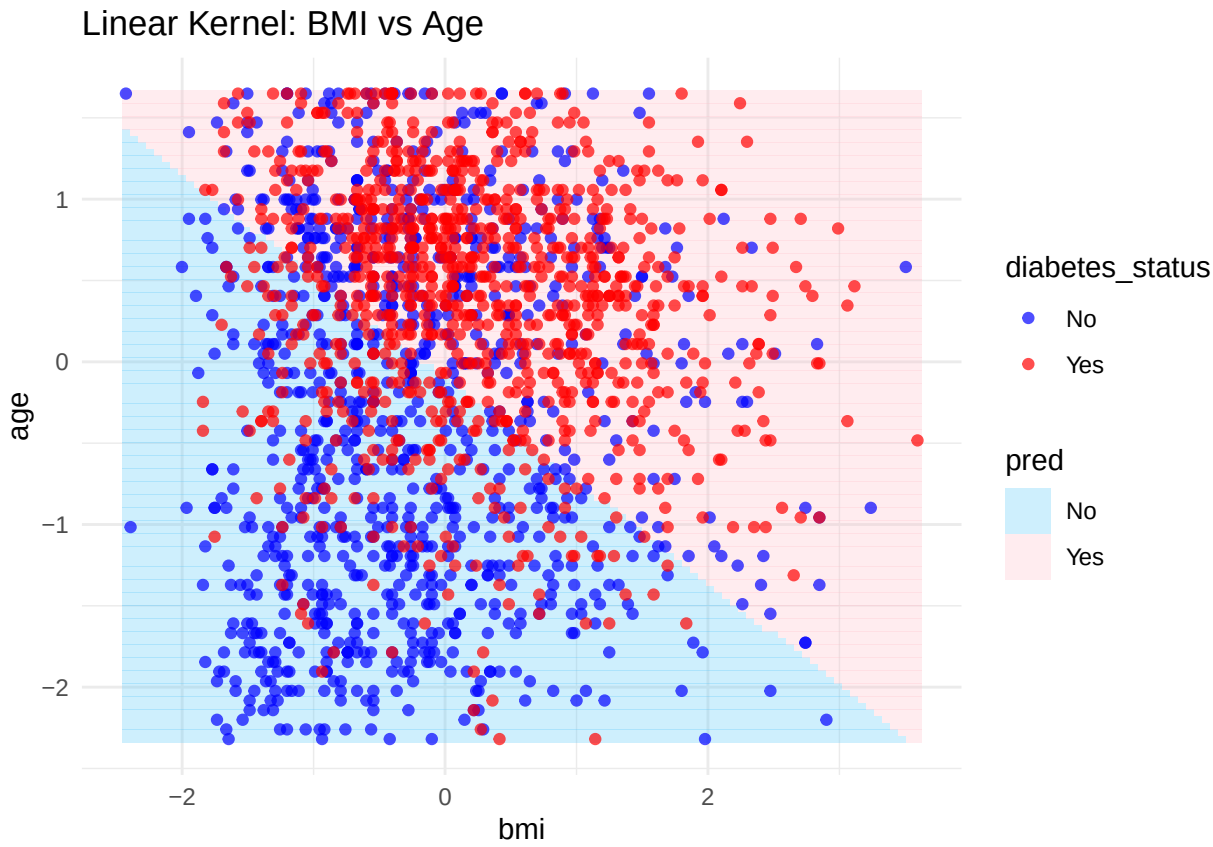
```r
# Let's check for importance
ggplot(linear_importance, aes(x = reorder(Variable, Importance), y = Importance)) + geom_col(fill = "st
```



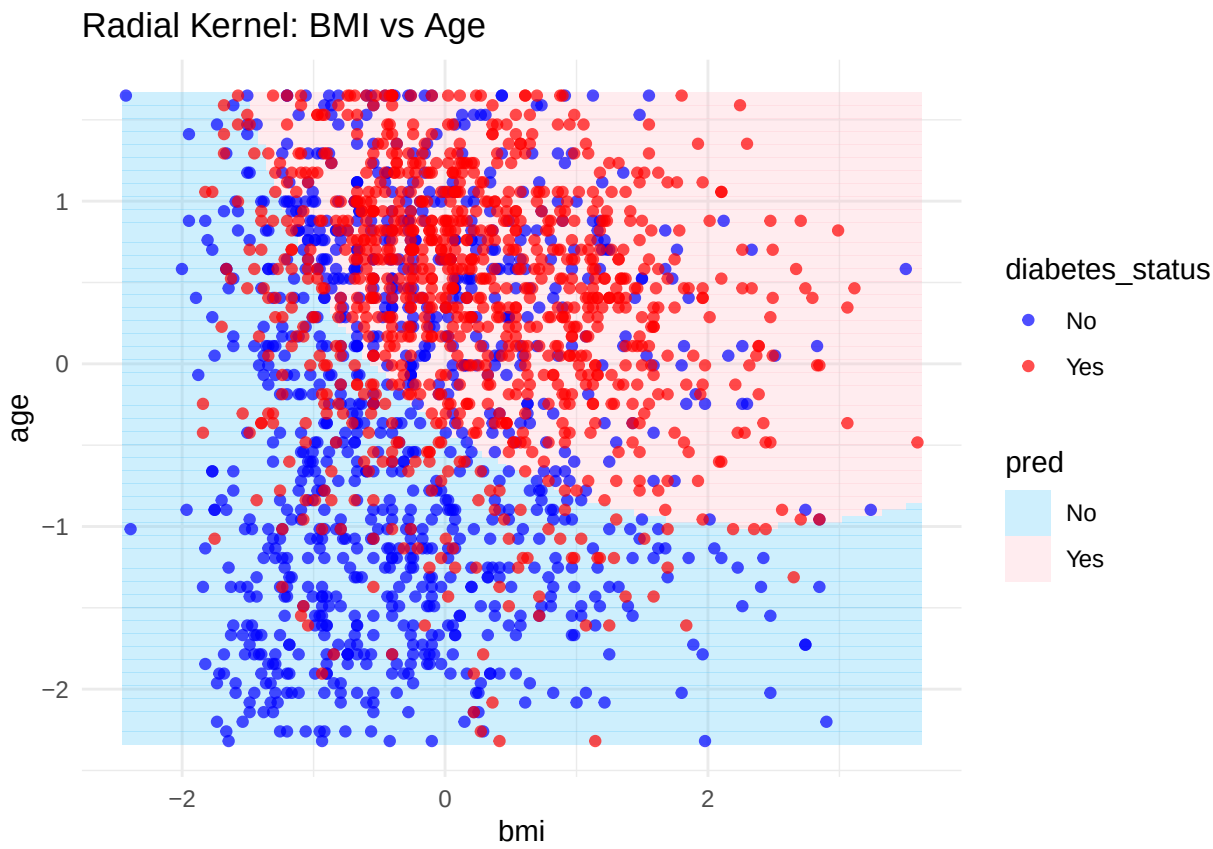
Comments: - Same key predictors emerging as most important across all kernels. - The most important predictors for the all the kernel SVM are: - 1. Age - 2. BMI - 3. Exercise Minutes - 4. Vegetable Servings - 5. Sleep Hours - SVM shows that age and BMI are the most important predictors for diabetes status.

7: DECISION BOUNDARY VISUALIZATION (On Top Predictors: BMI and Age Using All Kernels)

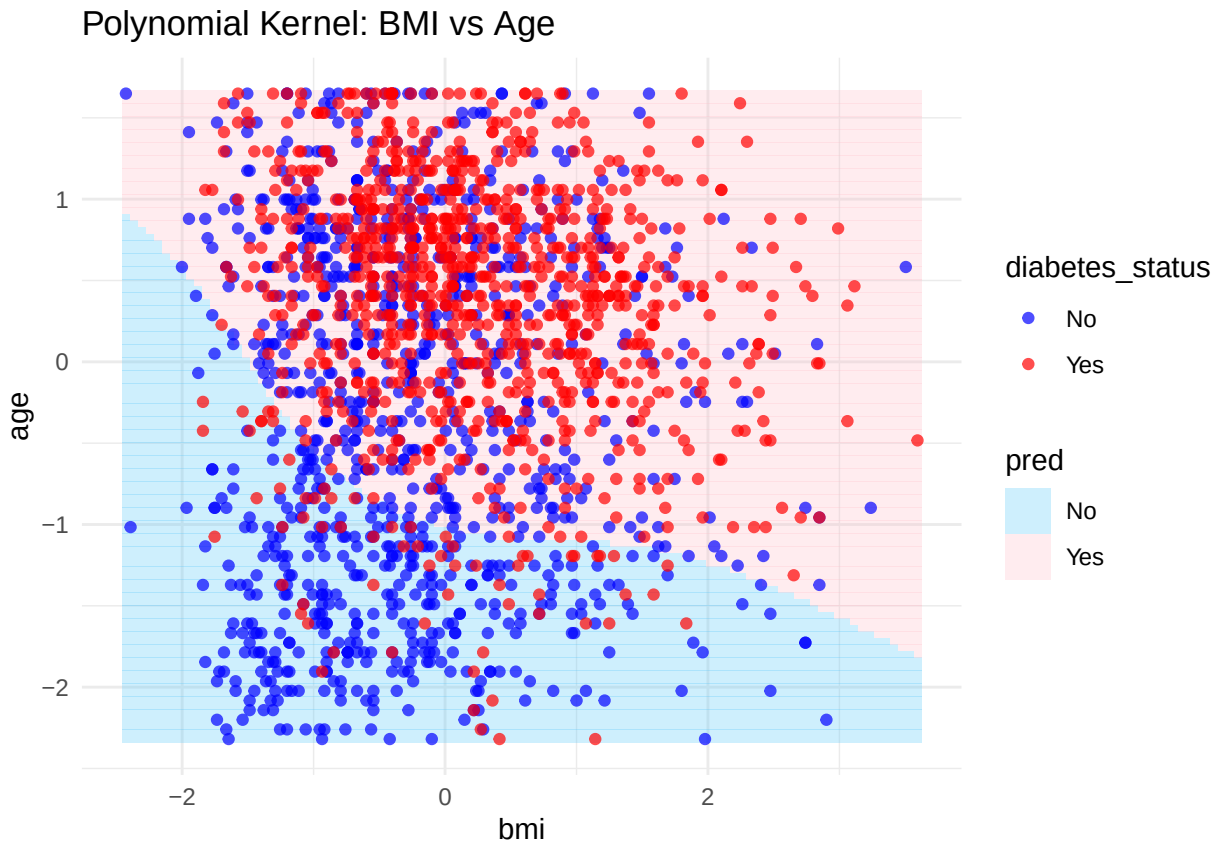
```r
# 7.1: Linear SVM: BMI vs AGE
svm_lin <- svm(diabetes_status ~ bmi + age, data = train_scaled, kernel = "linear", cost = 1)
grid <- expand.grid(bmi = seq(min(train_scaled$bmi), max(train_scaled$bmi), length.out = 100), age = se
grid$pred <- predict(svm_lin, grid)

ggplot() +
  geom_tile(data = grid, aes(bmi, age, fill = pred), alpha = 0.3) + geom_point(data = train_scaled, aes
```




```
# 7.2: Radial SVM: BMI vs AGE
svm_rad <- svm(diabetes_status ~ bmi + age, data = train_scaled, kernel = "radial", cost = 10, gamma = 
grid$pred <- predict(svm_rad, grid)

ggplot() + geom_tile(data = grid, aes(bmi, age, fill = pred), alpha = 0.3) + geom_point(data = train_sc
```




```
# 7.3: Polynomial SVM: BMI vs AGE
svm_poly <- svm(diabetes_status ~ bmi + age, data = train_scaled, kernel = "polynomial", degree = 3, co
grid$pred <- predict(svm_poly, grid)

ggplot() + geom_tile(data = grid, aes(bmi, age, fill = pred), alpha = 0.3) + geom_point(data = train_sc
```



Findings: - The decision boundary plots show the regions of predicted diabetes status based on BMI and age for each SVM kernel. - The linear kernel SVM is good, then comes the radial and then the polynomial shows a clear separations between the two classes as the dataset is cluttered.

Comments: - The decision boundary captures non-linear separation between diabetic and non-diabetic. - Many data points are on the other side of the decision boundary. - It tells us that data points are cluttered and overlapping.