ST310 Project

2025-06-03

1. Introduction

Drug consumption is known to be harmful, but many people still underestimate the risks. And the impact of drug use isn't limited to personal health, it can also lead to real danger for others. In March 2025, a tragic accident occurred near King's College London, where a student was killed and two others were injured after being hit by a van. The driver was arrested on suspicion of careless driving and drug-driving (BBC News, 2025).

In the UK, many drugs are classified as illegal such as cannabis, ecstasy, and cocaine. However, their usage persists across different age groups. Among all the drugs, cannabis is the most commonly used drug in England and Wales. According to the Crime Survey for England and Wales, around 6.8% of people aged 16 to 59 years reported using cannabis in the past 12 months, with an even higher rate of 13.8% among young adults aged 16 to 24 (Office for National Statistics, 2024).

In addition to the misuse of illegal drugs, the non-medical use of prescription drugs is also prohibited. For example, amphetamine is a stimulant that sometimes prescribed to treat conditions like ADHD or narcolepsy due to its effects on alertness and focus. However, misuse of amphetamine is associated with serious consequences, including addiction, cardiovascular problems, paranoia, and aggressive behavior. According to Figure 1, amphetamine is ranked among the ten most addictive drugs due to its high potential for psychological dependence and pleasure effects.

This project aims to use machine learning methods to predict the usage of amphetamine (*amphet_use*) based on a series of psychological, demographic, and behavioral variables. For demographic factors, we include age (*age*), gender (*gender*), education level (*education*), country of residence (*country*), and ethnicity (*ethnicity*). Psychological factors cover personality traits such as neuroticism (*nscore*), extraversion (*escore*), openness (*oscore*), agreeableness (*ascore*), conscientiousness (*cscore*), impulsivity (*impulsive*), and sensation-seeking (*ss*). Behavioral factors include consumption levels of other drugs: alcohol (*alcohol_level*), chocolate (*choc_level*), cannabis (*cannabis_level*), caffeine (*caff_level*), and nicotine (*nicotine_level*). Instead of including every drug variables from the original dataset, we selected these drugs due to their widespread use or relatively lower risk, which may help reveal broader patterns of drug consumption associated with amphetamine use. By identifying the key predictors of amphetamine use, the findings from this analysis can contribute to better strategies for prevention and early intervention.

2. Dataset

The dataset that we used for our analysis is the Drug Consumption dataset from the UCI Machine Learning Repository (link to our dataset: https://archive.ics.uci.edu/dataset/373/drug+consumption+quantified). The dataset comprises records from 1,885 respondents, each characterized by 12 quantified attributes. These attributes include psychological assessments such as the NEO-FFI-R (measuring neuroticism, extraversion, openness, agreeableness, and conscientiousness), BIS-11 (impulsivity), and ImpSS (sensation seeking), along with demographic information (education level, age, gender, country of residence, and ethnicity). While these input attributes are originally categorical, all features have been numerically quantified.

Respondents also reported their usage patterns across 18 legal and illegal substances, where drug use is categorized into seven classes: "Never Used", "Used over a Decade Ago", "Used in Last Decade", "Last Year", "Last Month", "Last Week", and Last Day".

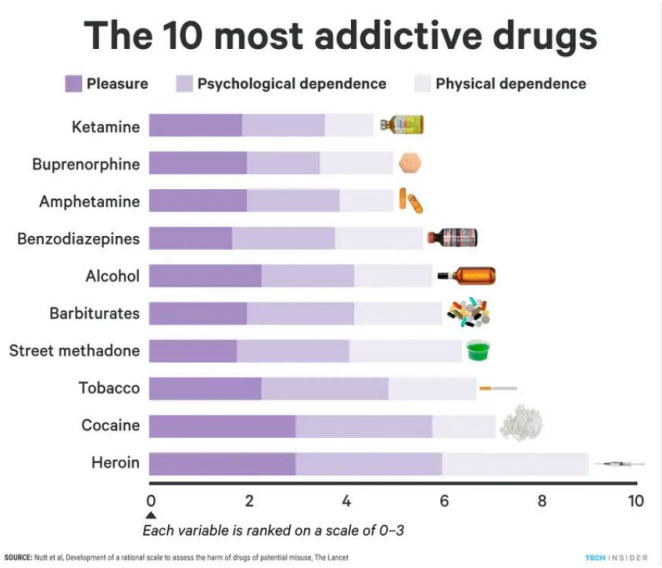


Figure 1: *Figure 1: The 10 most addictive drugs*

2.1 Load libraries and dataset

Load libraries:

```
suppressPackageStartupMessages({
library(caret)
library(ggplot2)
library(xgboost)
library(caret)
library(Matrix)
library(ROCR)
library(tidymodels)
library(dplyr)
library(yardstick)
library(smotefamily)
library(pROC)
library(lightgbm)
library(ROSE)
library(tibble)
})
```

Load data:

```
data <- read.csv("data/drug_consumption.csv")
#data <- read.csv("~/Desktop/ST310 Machine Learning/project/drug_consumption.csv")

# Keep only selected drug variables along with the original features
selected_vars <- c("age", "gender", "education", "country", "ethnicity", "nscore",
                   "escore", "oscore","ascore", "cscore", "impuslive", "ss",
                   "alcohol", "choc", "cannabis", "caff", "nicotine", "amphet")
drug_data <-  data[, selected_vars]
```

```r
# Rename 'impuslive' to 'impulsive'
colnames(drug_data)[colnames(drug_data) == "impuslive"] <- "impulsive"

head(drug_data)
```

```
##        age   gender education country ethnicity   nscore   escore   oscore
## 1  0.49788  0.48246  -0.05921 0.96082   0.12600  0.31287 -0.57545 -0.58331
## 2 -0.07854 -0.48246   1.98437 0.96082  -0.31685 -0.67825  1.93886  1.43533
## 3  0.49788 -0.48246  -0.05921 0.96082  -0.31685 -0.46725  0.80523 -0.84732
## 4 -0.95197  0.48246   1.16365 0.96082  -0.31685 -0.14882 -0.80615 -0.01928
## 5  0.49788  0.48246   1.98437 0.96082  -0.31685  0.73545 -1.63340 -0.45174
## 6  2.59171  0.48246  -1.22751 0.24923  -0.31685 -0.67825 -0.30033 -1.55521
##     ascore   cscore impulsive       ss alcohol choc cannabis caff nicotine
## 1 -0.91699 -0.00665  -0.21712 -1.18084     CL5  CL5      CL0  CL6      CL2
## 2  0.76096 -0.14277  -0.71126 -0.21575     CL5  CL6      CL4  CL6      CL4
## 3 -1.62090 -1.01450  -1.37983  0.40148     CL6  CL4      CL3  CL6      CL0
## 4  0.59042  0.58489  -1.37983 -1.18084     CL4  CL4      CL2  CL5      CL2
## 5 -0.30172  1.30612  -0.21712 -0.21575     CL4  CL6      CL3  CL6      CL2
## 6  2.03972  1.63088  -1.37983 -1.54858     CL2  CL4      CL0  CL6      CL6
##   amphet
## 1    CL2
## 2    CL2
## 3    CL0
## 4    CL0
## 5    CL1
## 6    CL0
```

```r
table(drug_data$amphet)
```

```
##
## CL0 CL1 CL2 CL3 CL4 CL5 CL6
## 976 230 243 198  75  61 102
```

Check data structure:

```r
str(drug_data)
```

```
## 'data.frame':    1885 obs. of  18 variables:
##  $ age      : num  0.4979 -0.0785 0.4979 -0.952 0.4979 ...
##  $ gender   : num  0.482 -0.482 -0.482 0.482 0.482 ...
##  $ education: num  -0.0592 1.9844 -0.0592 1.1637 1.9844 ...
##  $ country  : num  0.961 0.961 0.961 0.961 0.961 ...
##  $ ethnicity: num  0.126 -0.317 -0.317 -0.317 -0.317 ...
##  $ nscore   : num  0.313 -0.678 -0.467 -0.149 0.735 ...
##  $ escore   : num  -0.575 1.939 0.805 -0.806 -1.633 ...
##  $ oscore   : num  -0.5833 1.4353 -0.8473 -0.0193 -0.4517 ...
##  $ ascore   : num  -0.917 0.761 -1.621 0.59 -0.302 ...
##  $ cscore   : num  -0.00665 -0.14277 -1.0145 0.58489 1.30612 ...
##  $ impulsive: num  -0.217 -0.711 -1.38 -1.38 -0.217 ...
##  $ ss       : num  -1.181 -0.216 0.401 -1.181 -0.216 ...
##  $ alcohol  : chr  "CL5" "CL5" "CL6" "CL4" ...
```

```
##  $ choc     : chr  "CL5" "CL6" "CL4" "CL4" ...
##  $ cannabis : chr  "CL0" "CL4" "CL3" "CL2" ...
##  $ caff     : chr  "CL6" "CL6" "CL6" "CL5" ...
##  $ nicotine : chr  "CL2" "CL4" "CL0" "CL2" ...
##  $ amphet   : chr  "CL2" "CL2" "CL0" "CL0" ...
```

2.2 Initial data manipulation

Convert all drug variables into ordered factor (0–6):

```
drug_data2 <- drug_data
drug_vars <- c("alcohol", "choc", "cannabis", "caff", "nicotine", "amphet")

# Convert all to ordered numeric levels
for (var in drug_vars) {
  drug_data2[[paste0(var, "_level")]] <- as.numeric(factor(drug_data2[[var]],
                                                           levels = c("CL0", "CL1", "CL2", "CL3", "CL4"
}

# Remove the original categorical drug columns
drug_data2 <- drug_data2[ , !(names(drug_data2) %in% drug_vars)]
```

Convert *amphet* to binary factor:

- 0 = no use (CL0)
- 1 = used (CL1 to CL6)

```
# Create binary target: 0 = no use, 1 = used
drug_data2$amphet_use <- ifelse(drug_data2$amphet_level > 0, 1, 0)
drug_data2$amphet_use <- as.factor(drug_data2$amphet_use)

# Remove amphet_level
drug_data2$amphet_level <- NULL

head(drug_data2)
```

```
##        age   gender education country ethnicity   nscore   escore   oscore
## 1  0.49788  0.48246  -0.05921 0.96082   0.12600  0.31287 -0.57545 -0.58331
## 2 -0.07854 -0.48246   1.98437 0.96082  -0.31685 -0.67825  1.93886  1.43533
## 3  0.49788 -0.48246  -0.05921 0.96082  -0.31685 -0.46725  0.80523 -0.84732
## 4 -0.95197  0.48246   1.16365 0.96082  -0.31685 -0.14882 -0.80615 -0.01928
## 5  0.49788  0.48246   1.98437 0.96082  -0.31685  0.73545 -1.63340 -0.45174
## 6  2.59171  0.48246  -1.22751 0.24923  -0.31685 -0.67825 -0.30033 -1.55521
##     ascore   cscore impulsive       ss alcohol_level choc_level cannabis_level
## 1 -0.91699 -0.00665  -0.21712 -1.18084             5          5              0
## 2  0.76096 -0.14277  -0.71126 -0.21575             5          6              4
## 3 -1.62090 -1.01450  -1.37983  0.40148             6          4              3
## 4  0.59042  0.58489  -1.37983 -1.18084             4          4              2
## 5 -0.30172  1.30612  -0.21712 -0.21575             4          6              3
## 6  2.03972  1.63088  -1.37983 -1.54858             2          4              0
##   caff_level nicotine_level amphet_use
```

```
## 1          6              2          1
## 2          6              4          1
## 3          6              0          0
## 4          5              2          0
## 5          6              2          1
## 6          6              6          0
```

Check data structure:

```
str(drug_data2)
```

```
## 'data.frame':    1885 obs. of  18 variables:
##  $ age           : num  0.4979 -0.0785 0.4979 -0.952 0.4979 ...
##  $ gender        : num  0.482 -0.482 -0.482 0.482 0.482 ...
##  $ education     : num  -0.0592 1.9844 -0.0592 1.1637 1.9844 ...
##  $ country       : num  0.961 0.961 0.961 0.961 0.961 ...
##  $ ethnicity     : num  0.126 -0.317 -0.317 -0.317 -0.317 ...
##  $ nscore        : num  0.313 -0.678 -0.467 -0.149 0.735 ...
##  $ escore        : num  -0.575 1.939 0.805 -0.806 -1.633 ...
##  $ oscore        : num  -0.5833 1.4353 -0.8473 -0.0193 -0.4517 ...
##  $ ascore        : num  -0.917 0.761 -1.621 0.59 -0.302 ...
##  $ cscore        : num  -0.00665 -0.14277 -1.0145 0.58489 1.30612 ...
##  $ impulsive     : num  -0.217 -0.711 -1.38 -1.38 -0.217 ...
##  $ ss            : num  -1.181 -0.216 0.401 -1.181 -0.216 ...
##  $ alcohol_level : num  5 5 6 4 4 2 6 5 4 6 ...
##  $ choc_level    : num  5 6 4 4 6 4 5 4 6 6 ...
##  $ cannabis_level: num  0 4 3 2 3 0 1 0 0 1 ...
##  $ caff_level    : num  6 6 6 5 6 6 6 6 6 6 ...
##  $ nicotine_level: num  2 4 0 2 2 6 6 0 6 6 ...
##  $ amphet_use    : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 1 1 1 2 ...
```

Split data into training and testing set:

```
set.seed(38036)

train_indices <- createDataPartition(drug_data2$amphet_use, p = 0.8, list = FALSE)

train_data2 <- drug_data2[train_indices, ]
test_data2  <- drug_data2[-train_indices, ]

head(train_data2)
```

```
##        age   gender education country ethnicity   nscore   escore   oscore
## 1  0.49788  0.48246  -0.05921 0.96082   0.12600  0.31287 -0.57545 -0.58331
## 2 -0.07854 -0.48246   1.98437 0.96082  -0.31685 -0.67825  1.93886  1.43533
## 3  0.49788 -0.48246  -0.05921 0.96082  -0.31685 -0.46725  0.80523 -0.84732
## 4 -0.95197  0.48246   1.16365 0.96082  -0.31685 -0.14882 -0.80615 -0.01928
## 5  0.49788  0.48246   1.98437 0.96082  -0.31685  0.73545 -1.63340 -0.45174
## 6  2.59171  0.48246  -1.22751 0.24923  -0.31685 -0.67825 -0.30033 -1.55521
##     ascore   cscore impulsive       ss alcohol_level choc_level cannabis_level
## 1 -0.91699 -0.00665  -0.21712 -1.18084             5          5              0
## 2  0.76096 -0.14277  -0.71126 -0.21575             5          6              4
```

```
## 3 -1.62090 -1.01450  -1.37983  0.40148            6          4             3
## 4  0.59042  0.58489  -1.37983 -1.18084            4          4             2
## 5 -0.30172  1.30612  -0.21712 -0.21575            4          6             3
## 6  2.03972  1.63088  -1.37983 -1.54858            2          4             0
##   caff_level nicotine_level amphet_use
## 1          6              2          1
## 2          6              4          1
## 3          6              0          0
## 4          5              2          0
## 5          6              2          1
## 6          6              6          0
```

```
table(train_data2$amphet_use)
```

```
##
##   0   1
## 781 728
```

3. Methodology

3.1 Baseline model: logistic regression

3.1.1 Build Logistic regression model:

```
# Create a recipe
logit_recipe <- recipe(amphet_use ~ ., data = train_data2)

# Specify the logistic regression model
logit_spec <-
  logistic_reg(mode = "classification") %>%
  set_engine("glm")

# Combine recipe and model into a workflow
logit_workflow <-
  workflow() %>%
  add_model(logit_spec) %>%
  add_recipe(logit_recipe)

# Fit the workflow to the data
logit_fit <- fit(logit_workflow, data = train_data2)

# Full summary
logit_fit %>%
  extract_fit_parsnip() %>%
  pluck("fit") %>%
  summary()
```

```
##
## Call:
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.59832    0.49965  -3.199  0.00138 **
## age             0.50020    0.08321   6.012 1.84e-09 ***
## gender         -0.31950    0.13783  -2.318  0.02045 *
## education       0.08628    0.07146   1.207  0.22729
## country        -0.04844    0.10774  -0.450  0.65304
## ethnicity       0.62349    0.45586   1.368  0.17140
## nscore          0.09434    0.07449   1.266  0.20537
## escore         -0.04358    0.07748  -0.562  0.57380
## oscore          0.07281    0.07597   0.958  0.33782
## ascore          0.01123    0.06659   0.169  0.86612
## cscore         -0.05539    0.07494  -0.739  0.45981
## impulsive       0.09820    0.08420   1.166  0.24351
## ss              0.27064    0.09241   2.929  0.00340 **
## alcohol_level  -0.09862    0.04736  -2.082  0.03731 *
## choc_level     -0.08060    0.05918  -1.362  0.17321
## cannabis_level  0.35549    0.03965   8.967  < 2e-16 ***
## caff_level      0.15097    0.06031   2.503  0.01231 *
## nicotine_level  0.19910    0.02900   6.865 6.66e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2090.1  on 1508  degrees of freedom
## Residual deviance: 1609.6  on 1491  degrees of freedom
## AIC: 1645.6
##
## Number of Fisher Scoring iterations: 4
```

The result from logistic regression indicate that the most significant predictors of amphet use are age, gender, ss (sensation seeking), alcohol_level, cannabis level, caff_level, and nicotine_level. This finding is consistent with behavioral expectations: individuals who are older, male sex, and exhibit higher levels of sensation seeking are more inclined to experiment with substances like Amphetamine. Additionally, individuals who consume less alcohol and those who use more cannabis, caffeine, and nicotine are more likely use Amphetamine.

3.1.2 Compute accuracy:

```r
# Predict on test data
pred_class_test <- predict(logit_fit, new_data = test_data2, type = "class")$.pred_class
base_test_accuracy <- mean(pred_class_test == test_data2$amphet_use)
cat("Logistic Regression Test Accuracy:", round(base_test_accuracy, 3), "\n")
```

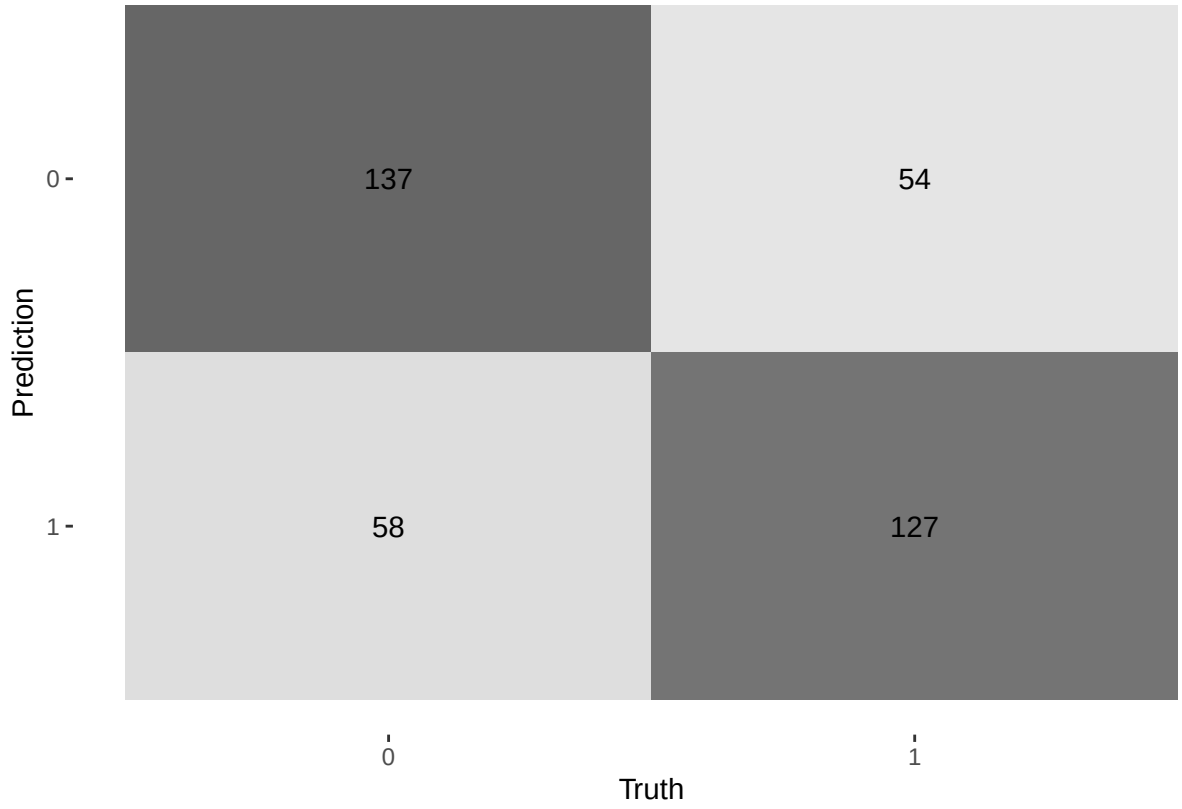
```
## Logistic Regression Test Accuracy: 0.702
```

```r
# Predict on train data
pred_class_train <- predict(logit_fit, new_data = train_data2, type = "class")$.pred_class
base_train_accuracy <- mean(pred_class_train == train_data2$amphet_use)
cat("Logistic Regression Training Accuracy:", round(base_train_accuracy, 3), "\n")
```

```
## Logistic Regression Training Accuracy: 0.73
```

3.1.3 Confusion matrix:

```
results_test <- tibble(
  truth = test_data2$amphet_use,
  pred = predict(logit_fit, new_data = test_data2, type = "class")$.pred_class
)

# Compute the confusion matrix
cm <- conf_mat(results_test, truth = truth, estimate = pred)

# Plot the confusion matrix
autoplot(cm, type = "heatmap")
```

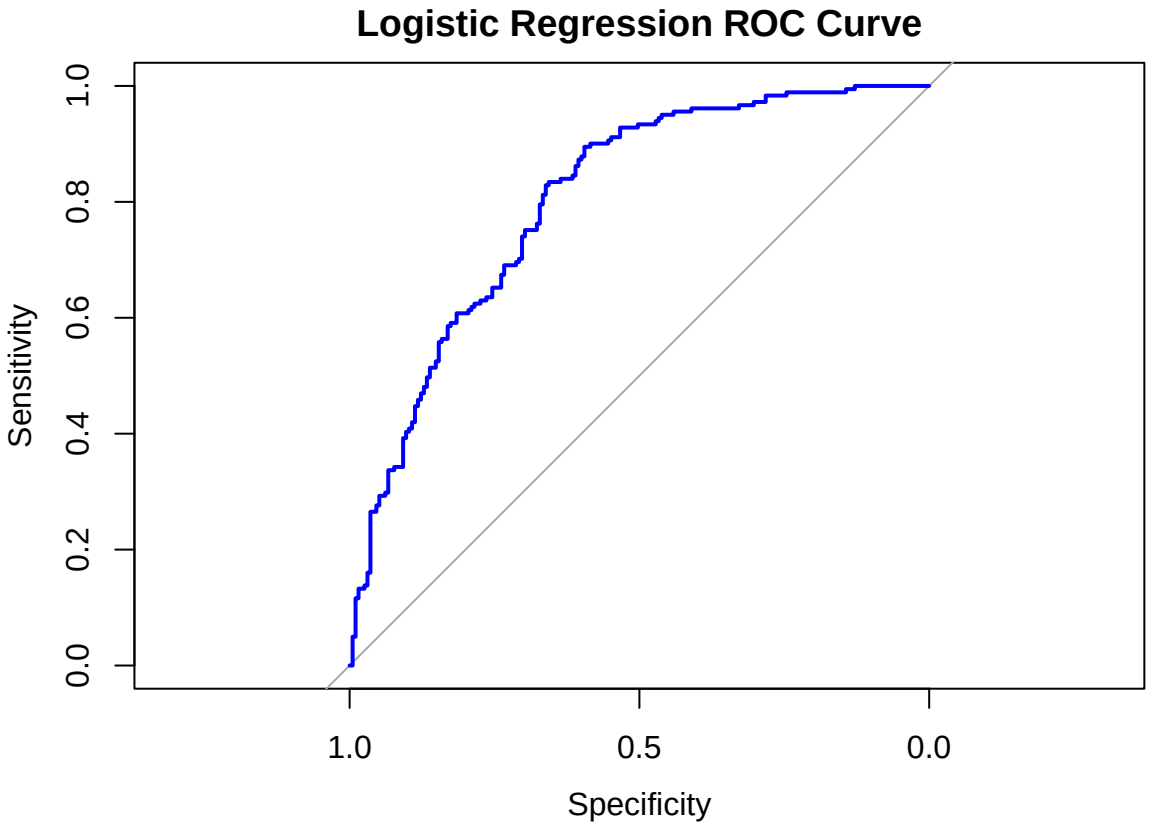


3.1.4 Compute ROC:

```
true_labels <- as.numeric(as.character(test_data2$amphet_use))
pred_probs_test <- predict(logit_fit, new_data = test_data2, type = "prob")$.pred_1
roc_obj <- roc(response = true_labels, predictor = pred_probs_test)

# Print AUC
log_auc <- auc(roc_obj)
print(log_auc)
```

```
## Area under the curve: 0.8031
```

```r
# Plot ROC curve
plot(roc_obj, main = "Logistic Regression ROC Curve", col = "blue")
```



The model's AUC is 0.80, indicating that there is a moderate 80% chance that the model ranks a random positive case (Amphetamine user) higher than a random negative case (non-user). Overall, the baseline model is performing relatively well with a test accuracy of 70%.

3.2 Custom Gradient Descent Model: Stochastic Gradient Descent (SGD) Model

We fitted the second model using our own implementation of stochastic gradient descent (SGD). We chose logistic regression as our model structure due to its simplicity and interpretability, and combined with the binary cross-entropy (BCE) loss function. This framework can keep the derivation and computation manageable, and better fit our binary classification task which predicting amphetamine use (amphet_use).

In our implementation, we first defined the sigmoid function to convert the linear combination of predictors into probabilities. Then, we calculated the gradient of the BCE loss, which depends on both the predicted probability and the true label. This satisfied the requirement that the gradient of the loss function should not be a constant because the gradient varies with each data point and changes during training.

We also applied mini-batch with a batch size of 32. In each epoch, the dataset was randomly shuffled and split into batches. For every mini-batch, we calculated the gradient, normalized it, and updated the weights (beta_hat). We tested several combinations of learning rates and number of epochs, and found out that a learning rate of 0.001 over 100 epochs had the most stable training and showed consistent learning. Our model aimed to minimize the BCE loss, which decreased from 0.6678 at epoch 1 to 0.5399 at epoch 100.

After training, we printed out the final weights, which indicated the importance and direction of each predictor related to the amphetamine use. For example:

- Cannabis consumption (0.297) and age (0.257) had positive associations with amphetamine use, and were two largest values. In other words, amphetamine use is more likely among individuals who use cannabis more frequently, and those are relatively older.
- Chocolate consumption (-0.162) and alcohol level (-0.140) showed negative associations with amphetamine use, which means that people who engage more frequently in these lower risk drugs are less reported to use amphetamines.

3.2.1 Build Stochastic Gradient Descent Model:

```
predictors <- c("age", "gender", "education", "country", "ethnicity",
                "nscore", "escore","oscore", "ascore", "cscore", "impulsive", "ss",
                "alcohol_level", "choc_level", "cannabis_level", "caff_level", "nicotine_level")
# Covert outcome variable to numeric
drug_data2$amphet_use <- as.numeric(as.character(drug_data2$amphet_use))

set.seed(38036)
batch_size <- 32 # Mini-batch size

# Sigmoid Function:
sigmoid <- function(z) {
  1 / (1 + exp(-z)) # map linear output to probabilities
}

# Binary Cross-Entropy (BEC) Loss:
BCE_loss <- function(y_true, y_pred) {
  epsilon <- 1e-8
  -mean(y_true * log(y_pred + epsilon) + (1 - y_true) * log(1 - y_pred + epsilon))
}

# Logistic Gradient:
logistic_gradient <- function(y_true, y_pred, X) {
  t(X) %*% (y_pred - y_true) / nrow(X)
}


# Define Stochastic Gradient Descent:
sgd_logistic_regression <- function(train_data2, lr = 0.01, epochs = 100) {
  X <- as.matrix(drug_data2[, predictors])
  y <- drug_data2$amphet_use

  X <- cbind(Intercept = 1, X) # Add intercept term
  beta_hat <- rep(0, ncol(X)) # Initialize weights

  n <- nrow(X)
  loss_history <- numeric(epochs)
  num_batches <- ceiling(n/batch_size)

  for (epoch in 1:epochs) {
    shuffled_indices <- sample(1:n, n, replace = TRUE) # Shuffle data

    for (batch in 1:num_batches) {
      batch_start <- 1 + batch_size * (batch - 1)
      batch_end <- min(batch * batch_size, n)
```

```
      batch_indices <- shuffled_indices[batch_start:batch_end]
      # Mini-batch
      X_batch <- X[batch_indices, , drop = FALSE]
      y_batch <- y[batch_indices]

      # Predict probabilities using sigmoid
      y_pred_batch <- sigmoid(X_batch %*% beta_hat)
      # Compute gradient
      gradient_batch <- logistic_gradient(y_batch, y_pred_batch, X_batch)
      # Normalize gradient
      gradient_norm <- sqrt(sum(gradient_batch^2))
      # Update beta_hat (weights)
      beta_hat <- beta_hat - lr * gradient_batch / (1e-8 + gradient_norm)
  }
    y_pred <- sigmoid(X %*% beta_hat)
    epoch_loss <- BCE_loss(y, y_pred)
    loss_history[epoch] <- epoch_loss

    # Print out loss every 10 epochs
    if (epoch %% 10 == 0 || epoch == 1) {
      cat("Epoch:", epoch, "- Loss:", round(epoch_loss, 4), "\n")
}
  }

  return(list(weights = beta_hat, loss_history = loss_history))
}

# Train the model
result <- sgd_logistic_regression(train_data2, lr = 0.001, epochs = 100)
```

```
## Epoch: 1 - Loss: 0.6678
## Epoch: 10 - Loss: 0.5693
## Epoch: 20 - Loss: 0.5539
## Epoch: 30 - Loss: 0.5496
## Epoch: 40 - Loss: 0.5468
## Epoch: 50 - Loss: 0.5452
## Epoch: 60 - Loss: 0.5436
## Epoch: 70 - Loss: 0.5425
## Epoch: 80 - Loss: 0.5413
## Epoch: 90 - Loss: 0.5411
## Epoch: 100 - Loss: 0.5399
```

```
# Extract final weights and loss history
final_weights <- result$weights
loss_history <- result$loss_history

# Display final weights
cat("Final Weights:\n")
```

```
## Final Weights:
```

```r
print(final_weights)
```

```
##                       [,1]
## Intercept      -0.071736553
## age             0.257226485
## gender         -0.142818738
## education       0.073192867
## country        -0.086108040
## ethnicity       0.033560247
## nscore          0.048026166
## escore         -0.074586378
## oscore          0.130962607
## ascore         -0.070836800
## cscore         -0.095000176
## impulsive       0.125746124
## ss              0.229079580
## alcohol_level  -0.139918536
## choc_level     -0.161973597
## cannabis_level  0.297031224
## caff_level     -0.007107689
## nicotine_level  0.188399724
```

```r
# Plot traing loss over epochs
plot(1:length(loss_history), loss_history,
     type = "b", pch = 19,
     xlab = "Epoch", ylab = "Loss (BCE)",
     main = "Training Loss over Epochs")
```

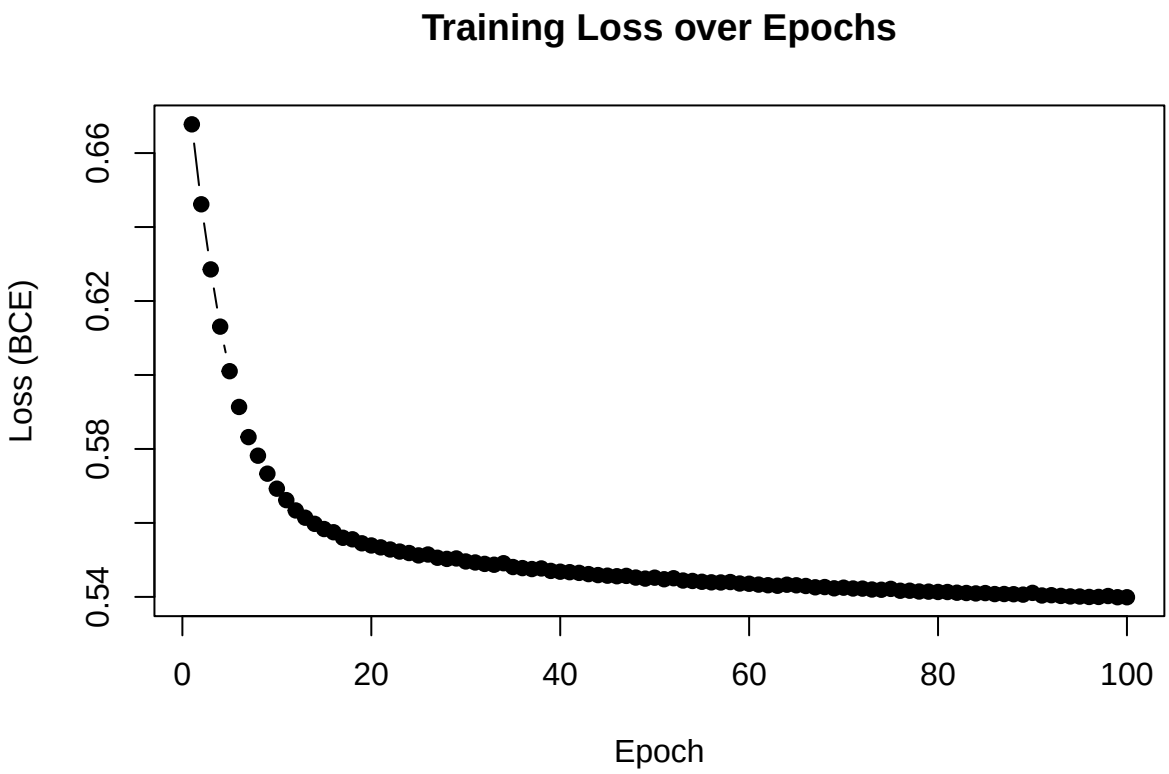


Figure 2. Training Loss over Epochs using custom SGD implementation. The BCE loss consistently decreases over the 100 epochs, reinforcing that our chosen learning rate (0.001) and number of epochs (100) led to a stable and convergent training process.

3.2.2 Evaluate on Test and Train Data:

```
# Test Data:
X_test <- as.matrix(test_data2[, -ncol(test_data2)])
y_test <- as.numeric(as.character(y_test <- test_data2$amphet_use))
# Add intercept term
X_test <- cbind(Intercept = 1, X_test)

# Compute predictions
z_test <- X_test %*% final_weights
y_hat_test <- sigmoid(z_test)
predictions_test <- ifelse(y_hat_test >= 0.5, 1, 0)

# Calculate accuracy
SGD_accuracy_test <- mean(predictions_test == y_test)
cat("SGD Model Accuracy (Test Set):", round(SGD_accuracy_test, 3), "\n")
```

```
## SGD Model Accuracy (Test Set): 0.729
```

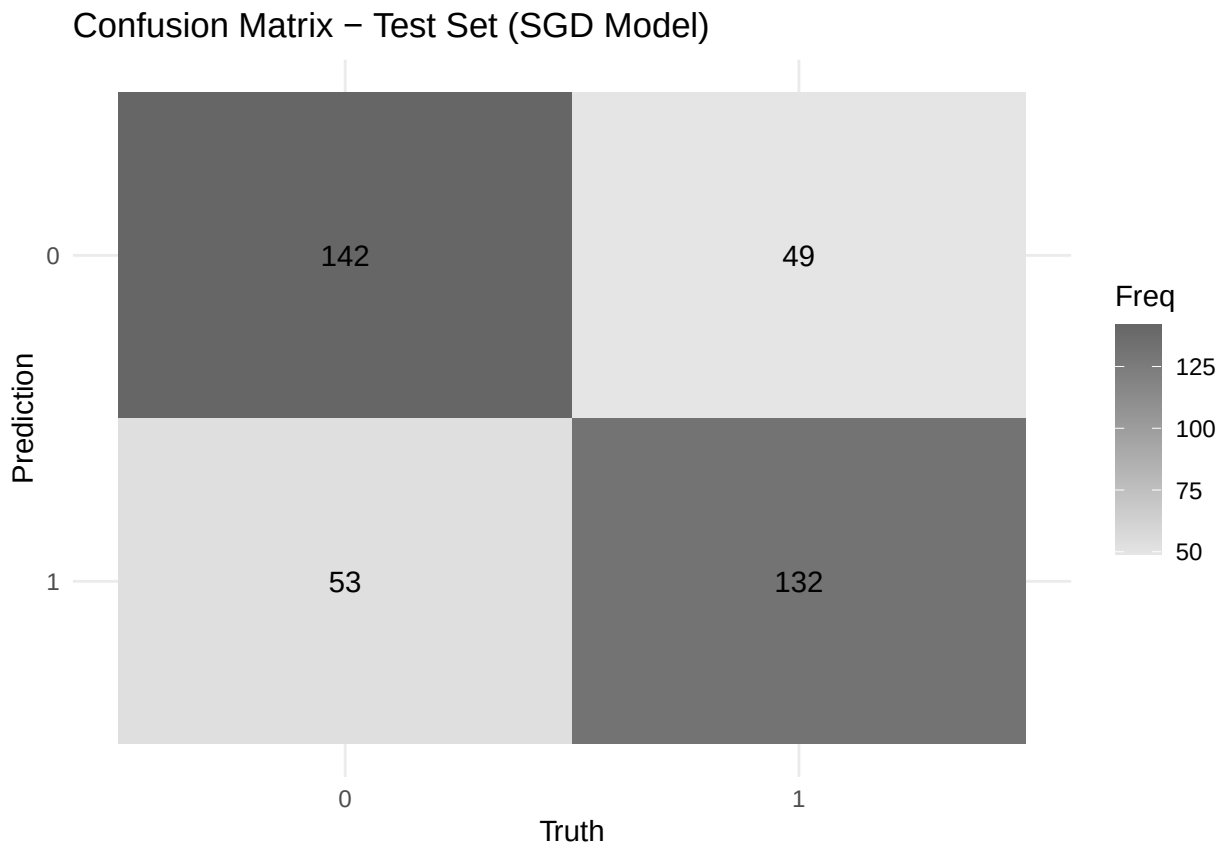
```
# Train Data:
X_train <- as.matrix(train_data2[, -ncol(train_data2)])
y_train <- as.numeric(as.character(y_train <- train_data2$amphet_use))
```

```r
# Add intercept term
X_train <- cbind(Intercept = 1, X_train)

# Compute predictions
z_test2 <- X_train %*% final_weights
y_hat_test2 <- sigmoid(z_test2)
predictions_test2 <- ifelse(y_hat_test2 >= 0.5, 1, 0)

# Calculate accuracy
SGD_accuracy_train <- mean(predictions_test2 == y_train)
cat("SGD Model Accuracy (Train Set):", round(SGD_accuracy_train, 3), "\n")
```

```
## SGD Model Accuracy (Train Set): 0.73
```

3.2.3 Confusion Matrix for SGD (Test Set):

```r
results_test <- tibble(
  truth = factor(y_test, levels = c(0, 1)),
  pred = factor(predictions_test, levels = c(0, 1))
)

# Compute the confusion matrix
cm_SGD <- conf_mat(results_test, truth = truth, estimate = pred)

# Plot the confusion matrix
autoplot(cm_SGD, type = "heatmap") + ggtitle("Confusion Matrix - Test Set (SGD Model)") + theme_minimal
```

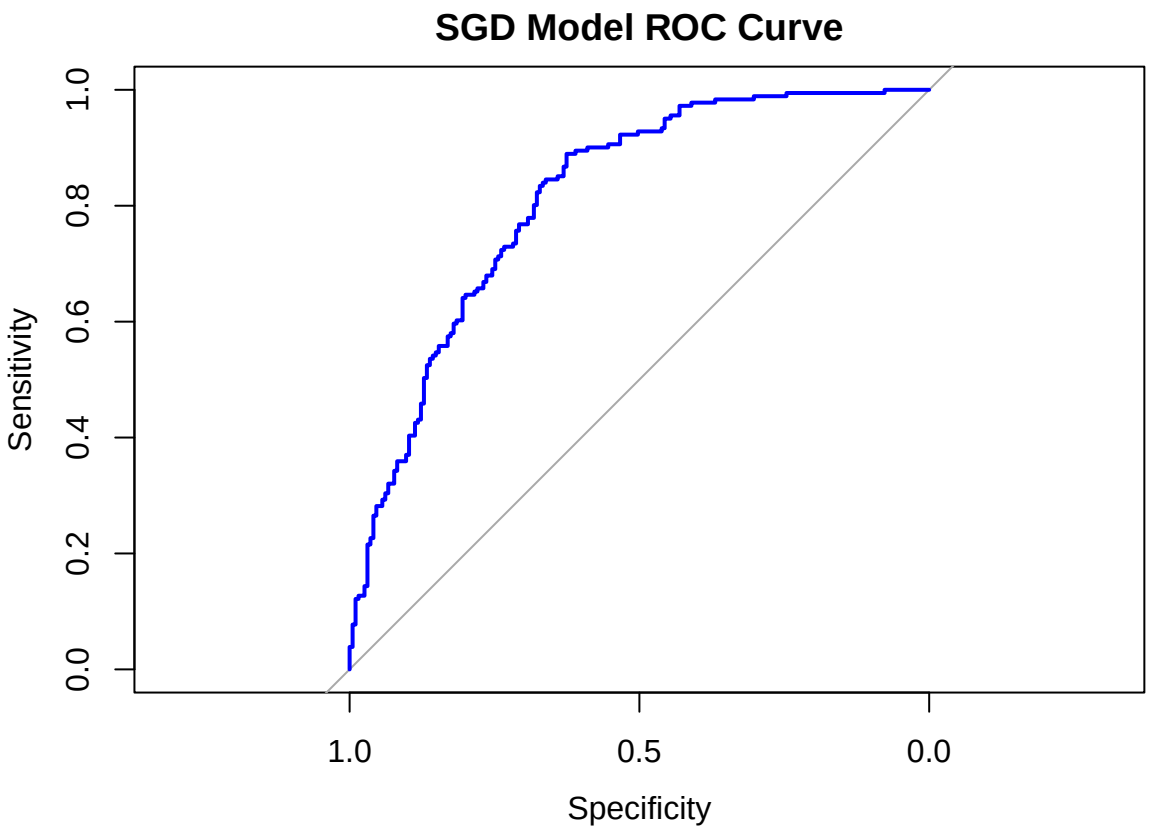


3.2.4 Compute ROC:

```
true_labels <- as.numeric(as.character(test_data2$amphet_use))
pred_probs_test <- y_hat_test
roc_obj <- roc(response = true_labels, predictor = pred_probs_test)

# Print AUC
SGD_auc <- auc(roc_obj)
print(SGD_auc)
```

```
## Area under the curve: 0.8132
```

```
# Plot ROC curve
plot(roc_obj, main = "SGD Model ROC Curve", col = "blue")
```



Our custom SGD model had an accuracy of 72.9% on the test set and 73% on the training set, which indicates this model does not have the overfitting problem. The ROC curve gave an AUC of 0.813, suggesting that the model can correctly distinguish between a user and a non-user of amphetamines about 81.3% of the time.

3.3 Interpretable Model: Lasso Regression

For our third relatively interpretable model, we used lasso regression to better understand what factors affect amphetamine use most. The lasso regression applied an L1 penalty to make the less important predictors more close to zero, which make it easier for us to identify the most significant predictors and interpret the results.

We used the structure of *tidymodels* to build our lasso regression, and performed a 10 folds cross-validation. After tuning, the best penalty value (λ) was 1×10^{-10}. We used this value to fit our final lasso regression model that included all predictors. The estimated coefficients of intercept and most significant predictors are as following (the significant predictors are sorted by magnitude from largest to smallest):

- Intercept (-0.125): This is the baseline value when all the other predictors are set to zero.
- Cannabis consumption (0.809): The strongest positive predictor in our lasso regression model. It indicates that individuals who use cannabis more frequently are much more likely to also use amphetamines.
- Nicotine consumption (0.479): The nicotine consumption is positively associated with amphetamine use, suggesting that higher nicotine use is linked to higher likelihood of amphetamine use.
- Age (0.427): Older individuals in this dataset are more likely to report the use of amphetamine.

- Sensation-seeking (0.253): Individuals with higher sensation-seeking scores are more likely to use amphetamines.
- Caffeine consumption (0.163): Individuals who consume more caffeine are also more likely to use amphetamines.
- Gender (-0.150): There is a negative relationship between gender and amphetamine use. The negative coefficient implies that female individuals (coded as +0.482) are less likely to use amphetamines compared to males (coded as -0.482).
- Alcohol level (-0.125): The alcohol consumption is negatively associated with amphetamine use, suggesting that people who drink alcohol more frequently show lower amphetamine use.

These seven significant predictors above consisted with the results from our baseline model, suggesting that psychological (sensation-seeking), demographic (age, gender), and behavioral (use of other drugs like cannabis, nicotine, caffeine, and alcohol) factors all contribute to amphetamine use. The results also supported that drug consumption behaviors are often interconnected, which means individuals who engage in one form of drug use are more likely to engage in others such as cannabis.

Compared to the baseline logistic model, this lasso regression model had the same test accuracy of 70.2% and train accuracy of 73.1%. The area under the ROC curve (AUC) was also similar (80.3%), and reinforced by the comparison ROC curves in Figure 3 which almost entirely overlapping. These findings showed that baseline logistic model and lasso regression model had similar classification ability, and applying L1 regularization did not improve predictive power. However, while both models are interpretable, the lasso model is easier for readers with less quantitative background to understand the most significant predictors because lasso regression reduced less important coefficients closer to zero instead of interpreting p-values or statistical significance tests.

3.3.1 Build Lasso regression model:

```
# Convert outcome variable to factor
train_data2$amphet_use <- as.factor(train_data2$amphet_use)
test_data2$amphet_use <- as.factor(test_data2$amphet_use)

# Build Lasso Model
lasso_recipe <- recipe(amphet_use ~ ., data = train_data2) %>%
  step_normalize(all_predictors()) # normalize all predictors

lasso_spec <- logistic_reg(
  mode = "classification",
  penalty = tune(),
  mixture = 1 # L1 penalty (lasso)
) %>%
  set_engine("glmnet")

# Cross-validation
set.seed(38036)
folds <- vfold_cv(train_data2, v = 10)

# Lasso workflow
lasso_workflow <- workflow() %>%
  add_recipe(lasso_recipe) %>%
  add_model(lasso_spec)

lasso_grid <- grid_regular(penalty(), levels = 30) # Create a grid of lambda (penalty) values
```

```r
# Tune the model over the grid using cross-validation
tuned_lasso <- tune_grid(
  lasso_workflow,
  resamples = folds,
  grid = lasso_grid,
  metrics = metric_set(accuracy, roc_auc)
)

# Find the best lambda based on accuracy
best_lasso <- select_best(tuned_lasso, metric = "accuracy")
print(best_lasso)
```

```
## # A tibble: 1 x 2
##        penalty .config
##          <dbl> <chr>
## 1 0.0000000001 Preprocessor1_Model01
```

```r
# Fit the final lasso model with the best lambda
final_lasso <- finalize_workflow(lasso_workflow, best_lasso)
lasso_fit <- fit(final_lasso, data = train_data2)
```

3.3.2 Evaluate on Test and Train Data:

```r
set.seed(38036)
# Test Data:
pred_test <- predict(lasso_fit, new_data = test_data2, type = "class") %>%
  pull(.pred_class)

# Predict probabilities
probs_test <- predict(lasso_fit, new_data = test_data2, type = "prob")

# Test Accuracy
lasso_test_acc <- mean(pred_test == test_data2$amphet_use)
cat("Lasso Regression Accuracy (Test Set):", round(lasso_test_acc, 3), "\n")
```

```
## Lasso Regression Accuracy (Test Set): 0.702
```

```r
# Train Data:
pred_train <- predict(lasso_fit, new_data = train_data2, type = "class") %>%
  pull(.pred_class)

# Train Accuracy
lasso_train_acc <- mean(pred_train == train_data2$amphet_use)
cat("Lasso Regression Accuracy (Train Set):", round(lasso_train_acc, 3), "\n")
```

```
## Lasso Regression Accuracy (Train Set): 0.731
```

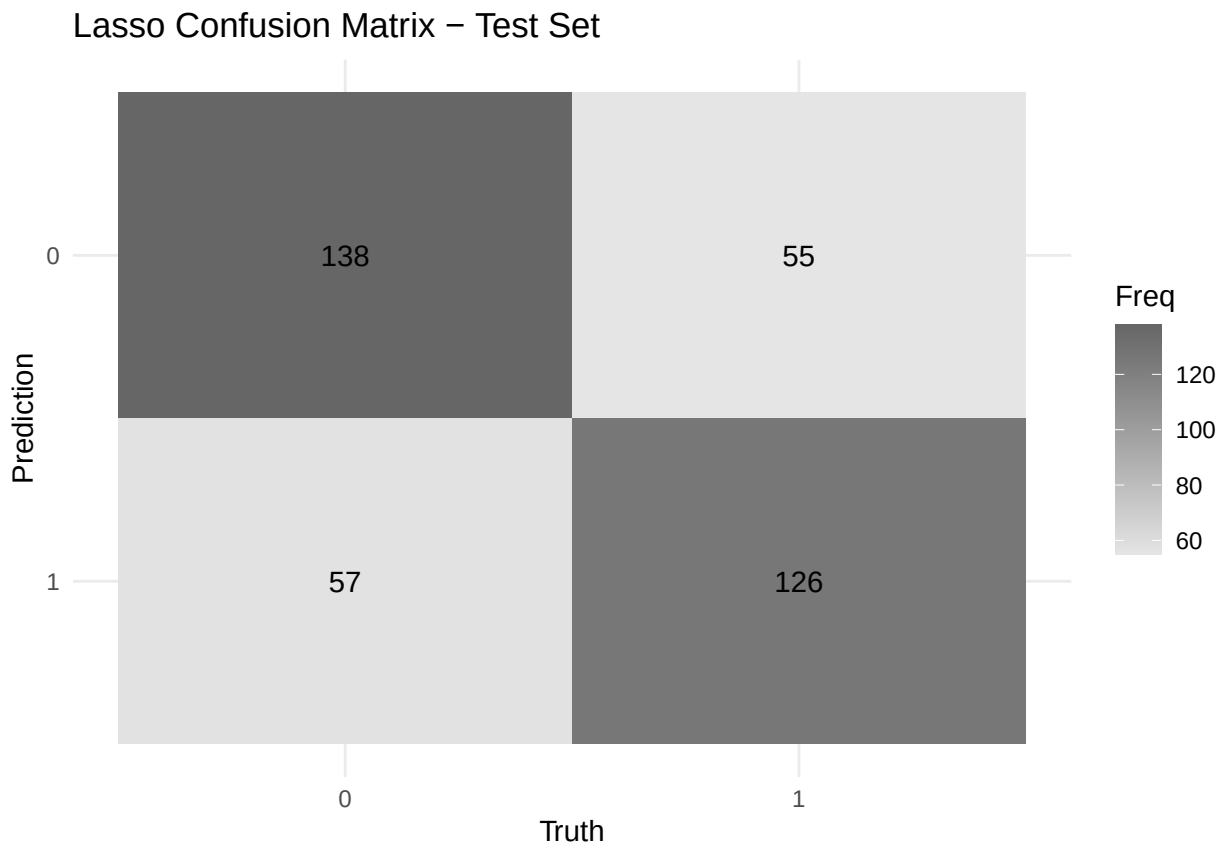
```r
lasso_coef <- lasso_fit %>%
  extract_fit_parsnip() %>%
  tidy() %>%
  filter(estimate != 0) %>%
```

```
  arrange(desc(abs(estimate)))

print(lasso_coef)
```

```
## # A tibble: 18 x 3
##    term           estimate      penalty
##    <chr>             <dbl>        <dbl>
##  1 cannabis_level  0.809   0.0000000001
##  2 nicotine_level  0.479   0.0000000001
##  3 age             0.427   0.0000000001
##  4 ss              0.253   0.0000000001
##  5 caff_level      0.163   0.0000000001
##  6 gender         -0.150   0.0000000001
##  7 alcohol_level  -0.128   0.0000000001
##  8 (Intercept)    -0.125   0.0000000001
##  9 ethnicity       0.0939  0.0000000001
## 10 impulsive       0.0916  0.0000000001
## 11 nscore          0.0888  0.0000000001
## 12 choc_level     -0.0808  0.0000000001
## 13 education       0.0736  0.0000000001
## 14 oscore          0.0709  0.0000000001
## 15 cscore         -0.0517  0.0000000001
## 16 escore         -0.0388  0.0000000001
## 17 country        -0.0327  0.0000000001
## 18 ascore          0.00324 0.0000000001
```

3.3.3 Confusion Matrix for Lasso (Test Set):

```
lasso_pred_class <- predict(lasso_fit, new_data = test_data2, type = "class") %>%
  bind_cols(test_data2)

cm_lasso <- conf_mat(lasso_pred_class, truth = amphet_use, estimate = .pred_class)
autoplot(cm_lasso, type = "heatmap") + ggtitle("Lasso Confusion Matrix - Test Set") +
  theme_minimal()
```



3.3.4 Compute ROC & Compare with Baseline Logistic Model:

```
# Baseline logistic model predictions
log_probs <- predict(logit_fit, test_data2, type = "prob") %>%
  bind_cols(test_data2) %>%
  mutate(model = "Logistic")

# Lasso model predictions
lasso_probs <- predict(lasso_fit, test_data2, type = "prob") %>%
  bind_cols(test_data2) %>%
  mutate(model = "Lasso")

true_labels <- test_data2$amphet_use
pred_probs_lasso <- probs_test %>% pull(.pred_1)
# Print AUC
roc_lasso <- roc(response = true_labels, predictor = pred_probs_lasso)
lasso_auc <- auc(roc_lasso)
print(lasso_auc)
```

```
## Area under the curve: 0.8033
```

```
# Plot ROC comparison curve
all_probs <- bind_rows(log_probs, lasso_probs)

roc_df <- all_probs %>%
  group_by(model) %>%
```

```
  roc_curve(truth = amphet_use, .pred_1)

autoplot(roc_df) +
  ggtitle("ROC Curve Comparison: Logistic vs Lasso") +
  theme_minimal()
```

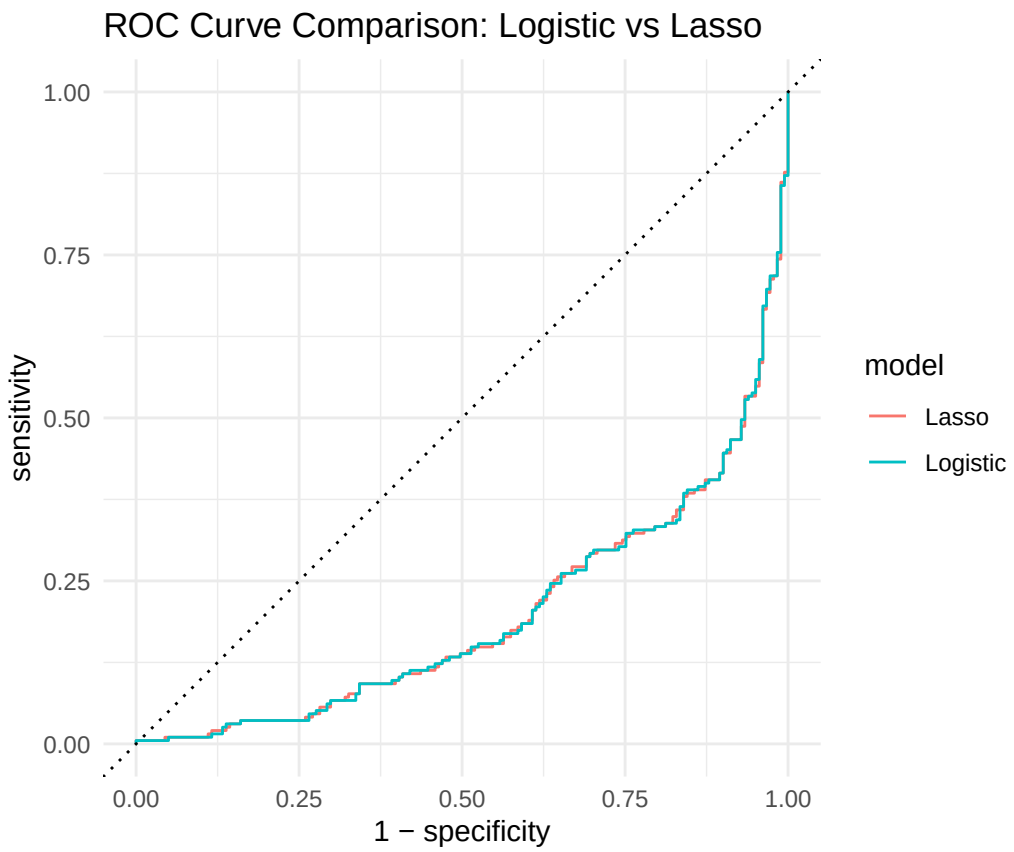


Figure 3. ROC curve comparison between the baseline logistic model and the lasso regression model. The two curves are nearly identical with AUC values around 0.803.

3.4 High Dimensional Model: Random Forest

The high-dimensional model was a Random Forest classifier built to predict binary amphetamine use (use vs. no use). We first applied oversampling to the minority class in the training data to mitigate class imbalance. After tuning the number of variables tried at each split (mtry in {2, 3, 4}) via 5-fold cross-validation, the final forest attained 73.7% accuracy on the held-out test set.

Examining the MeanDecreaseGini importances from the fitted forest, we found that other substance-use measures are the strongest predictors of amphetamine consumption. Cannabis_level was by far the top signal, followed by nicotine_level. Among personality dimensions, openness (oscore), conscientiousness (cscore) and neuroticism (nscore) emerged next in importance. Sensation-seeking (ss) and extraversion (escore) also contributed substantially, with agreeableness (ascore) trailing closely. Demographic features such as education, age and country had smaller—but still noticeable—effects. Together, these results suggest that patterns of other drug use, particularly cannabis and nicotine.

3.4.1 Set up cross-validation splits and recipe:

```r
set.seed(38036)
# Create 5-fold stratified cross-validation splits
cv_splits <- vfold_cv(train_data2, v = 5, strata = amphet_use)
```

```r
rf_recipe <- recipe(amphet_use ~ ., data = train_data2)
```

3.4.2 Build random forest model:

```r
# Define Random Forest model
rf_spec <-
  rand_forest(
    trees = 100,
    mtry  = tune()
  ) %>%
  set_mode("classification") %>%
  set_engine("randomForest")
```

```r
rf_wf <- workflow() %>%
  add_recipe(rf_recipe) %>%
  add_model(rf_spec)

mtry_grid <- tibble(mtry = c(2L, 3L, 4L))
```

```r
rf_res <- tune_grid(
  rf_wf,
  resamples = cv_splits,
  grid      = mtry_grid,
  metrics   = metric_set(accuracy, roc_auc)
)


# Select best mtry based on highest accuracy
best_rf <- rf_res %>% select_best(metric = "accuracy")
```

3.4.3 Fit final model:

```r
# Finalize model with best mtry and fit on full training data
rf_final <- rf_wf %>%
  finalize_workflow(best_rf) %>%
  fit(data = train_data2)
```

3.4.4 Model evaluation:

```r
# Evaluate model performance on training and test sets
pred_train <- predict(rf_final, train_data2) %>% pull(.pred_class)
pred_test  <- predict(rf_final, test_data2)  %>% pull(.pred_class)

rf_train_acc <- mean(pred_train == train_data2$amphet_use)
rf_test_acc  <- mean(pred_test  == test_data2$amphet_use)

cat("RF Accuracy (Train):", round(rf_train_acc,3), "\n")
```

```
## RF Accuracy (Train): 1
```

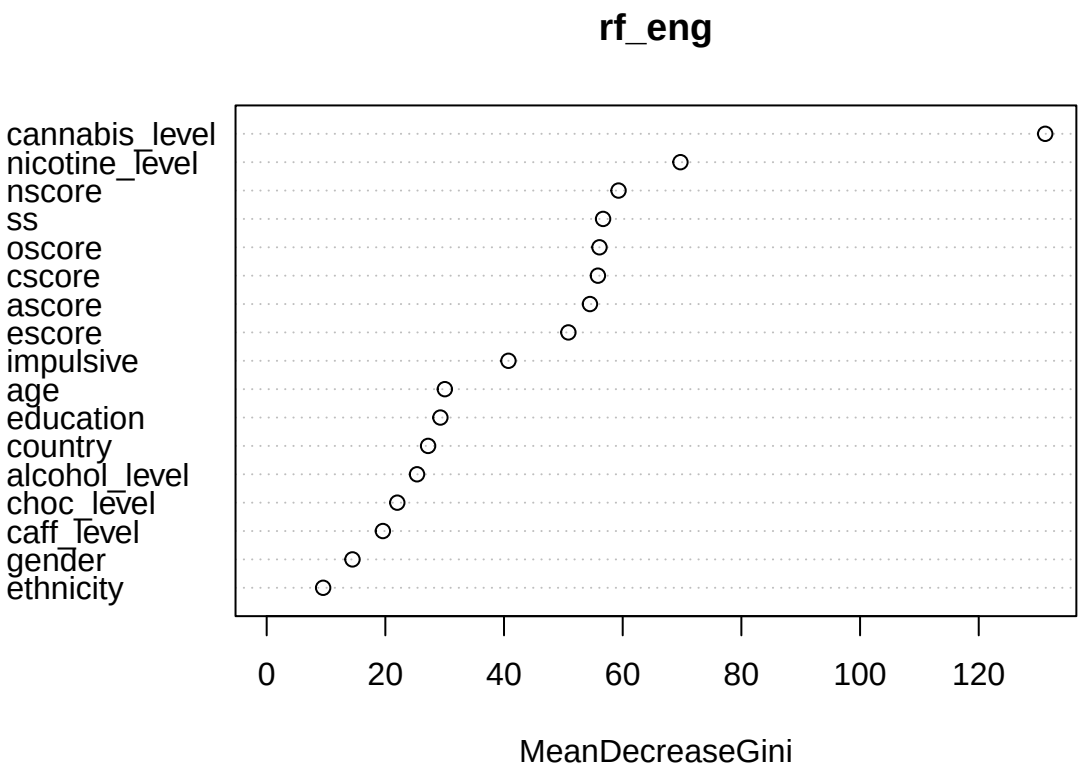
```
cat("RF Accuracy (Test) :", round(rf_test_acc,3), "\n")
```

```
## RF Accuracy (Test) : 0.737
```

```
rf_eng <- extract_fit_engine(rf_final)
print(importance(rf_eng))
```

```
##                 MeanDecreaseGini
## age                    30.023953
## gender                 14.456671
## education              29.275701
## country                27.205432
## ethnicity               9.516396
## nscore                 59.298662
## escore                 50.848187
## oscore                 56.087228
## ascore                 54.483545
## cscore                 55.821798
## impulsive              40.744082
## ss                     56.691296
## alcohol_level          25.340268
## choc_level             22.011596
## cannabis_level        131.203226
## caff_level             19.583919
## nicotine_level         69.729853
```

```
varImpPlot(rf_eng)
```




```
rf_probs <- predict(rf_final, new_data = test_data2, type = "prob") %>%
  bind_cols(test_data2 %>% select(amphet_use))

rf_auc <- roc_auc(rf_probs, truth = amphet_use, .pred_0)
print(rf_auc)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.811
```

3.5 Predictive model: XGB

3.5.1 Build Logistic regression model:

```
recipe <- recipe(amphet_use ~ ., data = train_data2)
cv <- vfold_cv(train_data2, v = 10, strata = amphet_use)

boost <- boost_tree(
  trees = tune(),
  learn_rate = tune(),
  tree_depth = tune(),
  loss_reduction = tune(),
  min_n = tune(),
```

```
  sample_size = tune(),
  mtry = tune()
) %>%
  set_mode("classification") %>%
  set_engine("xgboost", objective = "binary:logistic")

workflow_boost <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(boost)
```

3.5.2 Fit xgb with CV on training data:

```
fit_boost  <- tune_grid(
  workflow_boost,
  cv,
  metrics = metric_set(roc_auc)
  )
```

3.5.3 Fit best boosted tree model:

```
boost_best <- fit_boost %>%
  select_best() # best tuning parameters

boost_final <- finalize_model(
  boost,
  boost_best
  )

boost_final
```

```
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   mtry = 10
##   trees = 1980
##   min_n = 4
##   tree_depth = 14
##   learn_rate = 0.00119937603211951
##   loss_reduction = 0.000785408401368903
##   sample_size = 0.417912643856835
##
## Engine-Specific Arguments:
##   objective = binary:logistic
##
## Computational engine: xgboost
```

3.5.4 Test best boosted tree model:

```
final_fit <- workflow_boost %>%
  update_model(boost_final) %>%
  fit(data = train_data2)
```

```r
preds <- predict(final_fit, new_data = test_data2, type = "class") %>%
  bind_cols(test_data2 %>% select(amphet_use))

metrics(preds, truth = amphet_use, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.742
## 2 kap      binary         0.485
```

3.5.5 XGB accuracy:

```r
pred_train <- predict(final_fit, train_data2) %>% pull(.pred_class)
pred_test  <- predict(final_fit, test_data2)  %>% pull(.pred_class)

xgb_train_acc <- mean(pred_train == train_data2$amphet_use)
xgb_test_acc  <- mean(pred_test  == test_data2$amphet_use)

cat("XGB Accuracy (Train):", round(xgb_train_acc,3), "\n")
```

```
## XGB Accuracy (Train): 0.812
```

```r
cat("XGB Accuracy (Test) :", round(xgb_test_acc,3), "\n")
```

```
## XGB Accuracy (Test) : 0.742
```

3.5.6 XGB Confusion Matrix:

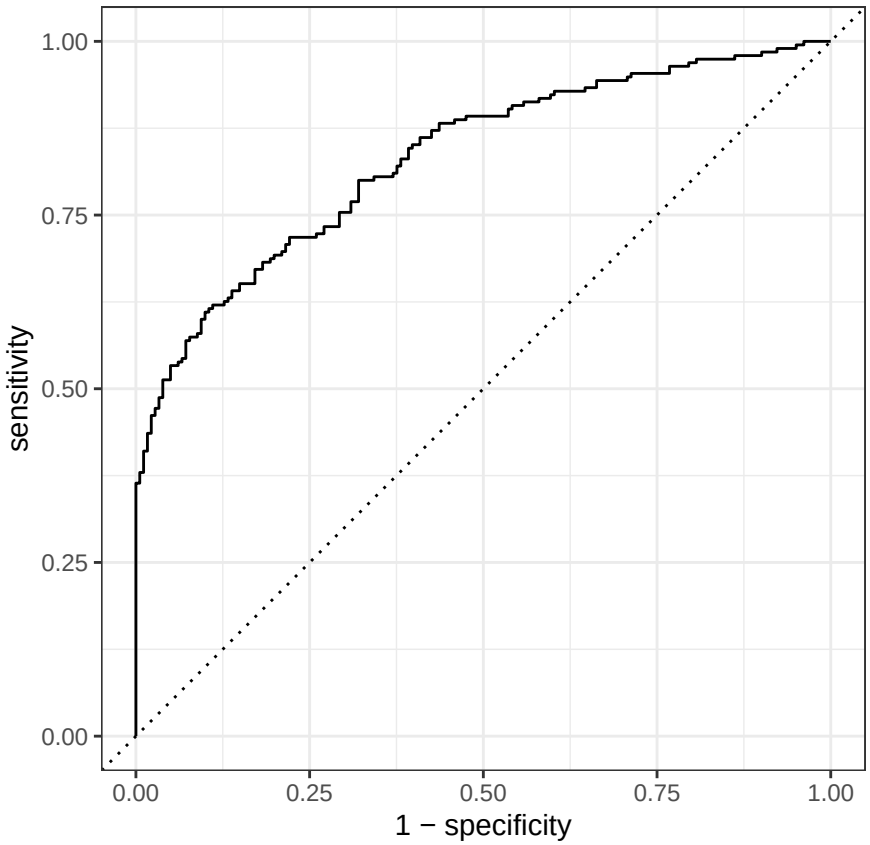
```r
conf_mat_test <- conf_mat(
  tibble(truth = test_data2$amphet_use, prediction = pred_test),
  truth = truth,
  estimate = prediction
)

conf_mat_test
```

```
##            Truth
## Prediction   0   1
##          0 138  40
##          1  57 141
```

The confusion matrix indicates that the XGBoost model successfully identified most true positives and true negatives. However, both the false negative and false positive rates remain non-negligible, suggesting room for improvement in classification precision and recall.

3.5.7 ROC:

```r
pred_probs <- predict(final_fit, new_data = test_data2, type = "prob") %>%
  bind_cols(test_data2 %>% select(amphet_use))

roc_data <- roc_curve(pred_probs, truth = amphet_use, .pred_0)
autoplot(roc_data)
```




```
xgb_auc <- roc_auc(pred_probs, truth = amphet_use, .pred_0)
xgb_auc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.832
```

The final XBGoost model yielded a test accuracy of 74.2% and a train accuracy of 81.2%. While resulting in a higher predictive accuracy, XGB is generally less intepretable than the other models. To achieve a higher predictive accuracy, we implemented a 10-fold stratified cross-validation to ensure that performance estimates were reliable and preserved the class balance of the amphet_use variable. We further tuned hyperparameters such as the number of trees, learning rate, and tree depth using automated optimization to find the best configuration. Together, these components allowed the XGBoost model to effectively capture complex non-linear relationships and interactions in the data, leading to a high predictive accuracy.

4. Model Comparisons

Summary of model performance:

```
log_auc_val   <- as.numeric(log_auc)
sgd_auc_val   <- as.numeric(SGD_auc)
lasso_auc_val <- as.numeric(lasso_auc)
rf_auc_val    <- rf_auc$.estimate
```

```r
xgb_auc_val   <- xgb_auc$.estimate

model_names <- c("Logistic Regression", "SGD", "Lasso Regression" , "Random Forest", "XGBoost")
accuracy <- c(base_test_accuracy, SGD_accuracy_test, lasso_test_acc, rf_test_acc, xgb_test_acc)
auc <- c(log_auc_val, sgd_auc_val, lasso_auc_val, rf_auc_val, xgb_auc_val)

# Create a data frame
summary_table <- data.frame(
  Model = model_names,
  Accuracy = accuracy,
  AUC = auc
)
print(summary_table)
```

```
##                 Model  Accuracy       AUC
## 1 Logistic Regression 0.7021277 0.8031449
## 2                 SGD 0.7287234 0.8131747
## 3    Lasso Regression 0.7021277 0.8032583
## 4       Random Forest 0.7367021 0.8111772
## 5             XGBoost 0.7420213 0.8317892
```

Our five models predict amphetamine use based on demographic, psychological, and behavioural variables. Our logistic regression identified age, gender, and sensation-seeking as significant predictors, achieving 70% accuracy. Our custom stochastic gradient descent (SGD) model demonstrated slightly better accuracy (72.9%). The Lasso regression provided interpretability by shrinking less important predictors towards zero, identifying cannabis, nicotine, and age as key predictors. Its performance was similar to logistic regression, with an accuracy of 70.2%. The Random Forest model achieved higher accuracy (73.7%) and highlighted cannabis and nicotine consumption as the most critical predictors, along with key personality traits (openness, conscientiousness, and neuroticism). Finally, the XGBoost model attained strong predictive accuracy(74.2%) and interpretability, slightly outperforming previous models.

All models demonstrated a reasonable AUC value of around 0.80, indicating that the models have an 80% chance of assigning a higher predicted probability to non-amphetamine users than to users. Overall, these models suggest that substance use patterns, especially cannabis and nicotine, along with personality and demographic factors, might be able to predict amphetamine usage.

5. Conclusion

In conclusion, by applying five machine learning models, we have summarised critical factors to predict amphetamine use. Among them, cannabis and nicotine consumption are the most dominant factors, alongside key demographic characteristics such as age and gender, and personality traits including sensation-seeking, openness, conscientiousness, and neuroticism. The XGBoost model achieved the highest accuracy (74%). These findings indicate potential value in focusing interventions on identified demographic and psychological risk factors alongside substance-use behaviors. Future research could further investigate additional predictors with larger databases and apply advanced modeling techniques to enhance predictive capabilities.

Reference

- BBC News. (2025, March 25). Student killed and two injured in Strand van crash named. https://www.bbc.co.uk/news/articles/cpdeg86ppdjo

- Office for National Statistics. (2024, December 12). Drug misuse in England and Wales: year ending March 2024. https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/articles/drugmisuseinenglandandwales/yearendingmarch2024