Using Statistics to Predict the Depression Statistic

Ash Kmetz (2240214), Tu Nguyen (2391093), Liam Reilly (2025217), Dionissios Kakissis (2160152)

MATH 4323

May 3, 2025

1. Introduction (Ash Kmetz)

Time and time again, people have said the same unfortunate mantra when tragedy strikes: “they seemed so happy”, “I never noticed anything wrong”, “they never told us anything was wrong”. And of course, when this outcome is reached, it is far too late to fix, as it is the one irreversible action.

Even for the times where depression does not result in tragedy, people suffer for months, years, even decades in silence, not knowing that they are depressed because it has not crossed their minds as a possibility. What if one could use factors of an individual’s life to predict if they have depression before they even see a doctor, however? What if schools could know if their students were potentially suffering based on a survey? These questions influenced our decision to ultimately attempt to pick how accurately a model could predict whether a student has depression or not.

With the goal in mind, we selected a dataset that focused on student depression. Our response variable was, unsurprisingly, whether or not the student had depression, and the rest of the variables which are: id, gender, age, city, profession, academic pressure, work pressure, CGPA, study satisfaction, job satisfaction, sleep duration, dietary habits, degree, whether the student has had suicidal thoughts before, work/study hours, financial stress, and if there is a family history of depression present. It is worth noting that not all of these predictors will end up used for a variety of reasons which will be gone over in the preprocessing subsection of this report. However, with a little over 28,000 records, we felt this was the best dataset to attempt to answer our big overarching question:

Can one predict whether a student has depression or not based on how they answer a survey and factors from their school life?

The hope is that, if the testing error rates come back good, we can figure out what predictors were most useful using post-hoc analysis. Being able to know what the most prominent signs of depression are can be useful in advising students who fit those signs in what steps to take next.

2. Methodology

We utilized SVM and KNN since we were doing supervised learning - in order to properly discuss both methods, this section has subsections.

2.1 KNN (K-Nearest Neighbors) (Tu Nguyen)

We identify the nearest K points to an observation, and the classification of that observation is determined by the majority class among the nearest K points.

Formula for conditional probability for each possible class j:

$$\Pr(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} \mathrm{I}(y_i = j)$$

Where $\mathrm{I}(y_i = j)$ is an indicator function that equals 1 if the i-th neighbor belongs to class j, and 0 otherwise. N_0: set of K nearest points

For classifying the observation:

Observation x_0 is assigned to the class with the highest estimated probability:

$$\widehat{C_{KNN}}(x_0) = \arg\max_j \quad \hat{p_j}(x_0)$$

And lastly, the distance function, in which we use Euclidean:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}$$

KNN provides an advantage over SVM in its simplicity in implementation. You only need to find the optimal value of K to get the best model - whereas SVM will involve a lot of tuning and kernel selection. However, KNN is insanely insensitive to outliers because all data points are considered, and can struggle in large spaces where distances between data points become less meaningful. Furthermore, if data is not scaled properly, some predictors will be treated much heavier than others regardless of actual impact on class prediction. It is our hope that we have scaled everything accordingly to make the distance meaningful.

2.2 SVM (Support Vector Machine) (Dionissios Kakissis, Liam Reilly)

Support vector machines construct a decision boundary (aka, a hyperplane) based on maximizing the margin between the support vectors. The "shape" of the hyperplane depends on the kernel used, with the three major ones being linear (straight lines), radial (circular), or polynomial (u shapes).

The equation depends on the type of kernel used - as our kernel utilizes the radial kernel, we will use the equations from that context of SVMs.

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma \textstyle\sum_{k=1}^{p} (x_{ik} - x_{jk})^2), \gamma > 0$$

This equation draws the radial support vectors that separate each class.

SVMs also come with two main parameters that influence how the model will end up being fit: cost and gamma. Gamma determines how "rigid" the boundaries of the support vectors will end up being. Higher gamma values result in more rigidity but lead to the potential of overfitting or overcomplicating the model. Cost, meanwhile, determines how many points get to fall inside of the margins where points should usually not be. Lower costs allow for more points in the margin, which means wider margins in the long run, while higher costs disallow points in the margins and results in smaller margins. As with gamma, there is the risk of under or overfitting at all possible points, and part of the SVM model fitting was trialing a few different cost and gamma combinations to see what would give the best result.

3. Data Analysis

3.1 Data Pre-processing (Ash Kmetz)

Before even beginning fitting models around our data, it is important to stop and analyze what exactly we are working with and the context, as well as cleaning the data up. For the obvious reason, the id column will not be considered in any analysis, as it is simply a primary key for the database and therefore has no actual meaning. The data does in fact come with columns that do not contain enough data collection to truly be meaningful. For example, out of nearly 28,000 records, only three students reported feeling some non-zero "work pressure" value. For that reason, the work pressure column will not be considered in the analysis. Similarly, job satisfaction also only has eight non-zero entries, so it will also not be considered. Both columns do not feel like major losses, as we wish to hone our focus on students and whether they have depression based on factors from education and their home life.

I also decided to remove 31 records from the dataset entirely – these records had their profession listed as something other than student. Considering that we are only really interested in attempting to predict student depression, they did not seem like necessary records for the question of interest. For that reason, the profession column will also not be included in the data analysis since it will be all students, however the column removal occurs after the removal of the records to make sure the data is cleaned. Furthermore, three rows were removed for having "null" values present specifically in the "financial stress" column. I placed the median value (3) in there, which makes sense given that financial stress is a scale from 1-5.

I check for duplicates, of which there are none, and convert all categorical variables to factors. Next, I checked the standard deviation of all the numeric variables to see if they were on similar enough scales (this happens before the factor changes in the code). The standard deviations are a little different, but the one that worries me the most ends up being age. Since we are using two methods that are sensitive to outliers, I decided to scale age due to the existence of outlier ages in the data. It is logical for students to mostly be young, but there are older students, and they end up forming outliers. By scaling, we reduce the risk of SVM and KNN being misled.

3.2 Optimal Parameters and the Test Error Rate (All)

For both models, we used the seed "123" and split the data into 80% training, 20% testing data to attempt to identify the optimal parameters. For the KNN model, values of K from 1-400, with steps = 5 (so testing 1, 6, 11, 16, etc), were tested. The outcome was that K = 196 was the best from all of the tested K values. Meanwhile, the SVM turned out to find radial as the best kernel, and then tested gammas of 0.01 and 0.1 with costs of 0.1, 1, and 10 (all possible combinations were tested). The outcome was that the optimal gamma was 0.01 while the optimal cost was 1.

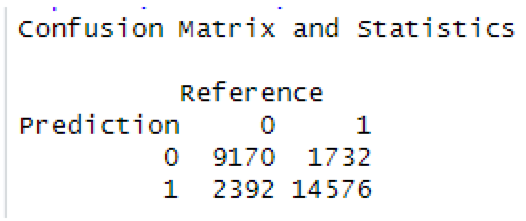
From there, we took the optimal parameters and compared the prediction test set errors from both models to figure out which model was performing better. The KNN had an accuracy of 78.42%, or in other words, a test error rate of 21.58%. Meanwhile, the SVM had an accuracy of 84.21%, or in other words, a test error rate of 15.79%. By a fairly impressive amount, the SVM with the radial kernel performs better than the KNN model.

While this will be explored more later, the SVM model ends up so complex it indicates that there are not clear cuts in the data. If this is the case, it would make sense why KNN ends up performing worse: most likely there are lots of points of both classes condensed close together in locations,which is also why K ends up so high to get such a good result.

3.3 Fitting the SVM on All Data & Interpretation (Liam Reilly)

With the best model identified, it was time to fit it to the entire dataset and not just a subsection. With the optimal parameters defined, we ran the SVM with the entire data set and examined how it performed. We started with no cross-validation or anything, just to get a flat number off the whole set. Later, we'll set up 10-fold cross-validation in order to get a better picture of what the SVM model is doing.

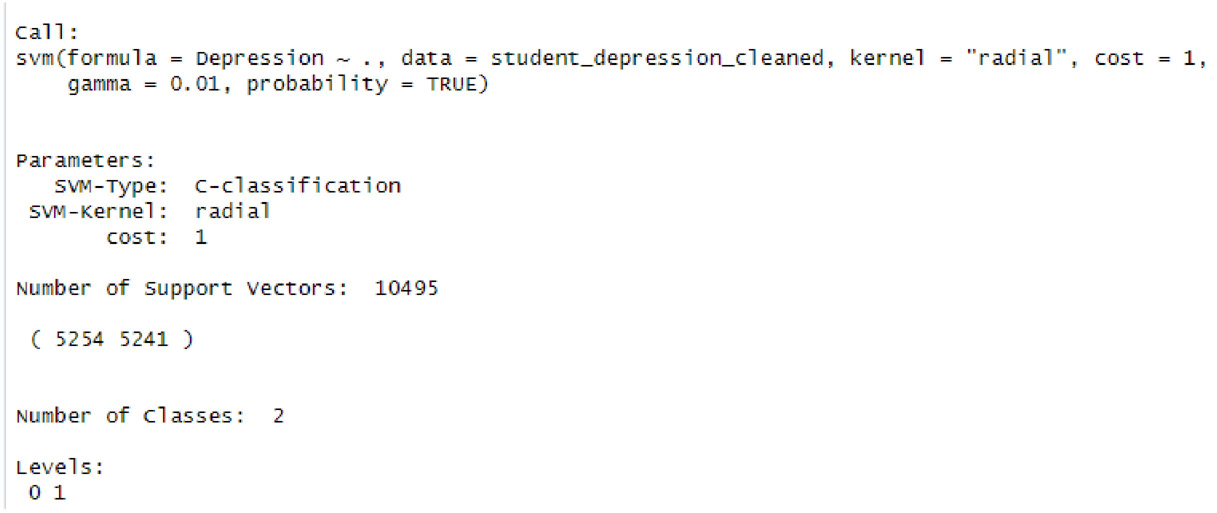
First and foremost, the confusion matrix of the best performing SVM model against the whole dataset:



```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  9170  1732
         1  2392 14576
```

The prediction rate of the best SVM fit, which showcases that there are 4,125 misclassifications total and then 23,746 correct classifications. It also helps us notice that there is a slight skew towards students having depression rather than not having depression.



```
Call:
svm(formula = Depression ~ ., data = student_depression_cleaned, kernel = "radial", cost = 1, 
    gamma = 0.01, probability = TRUE)


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  radial 
       cost:  1 

Number of Support Vectors:  10495

 ( 5254 5241 )


Number of Classes:  2 

Levels: 
 0 1
```

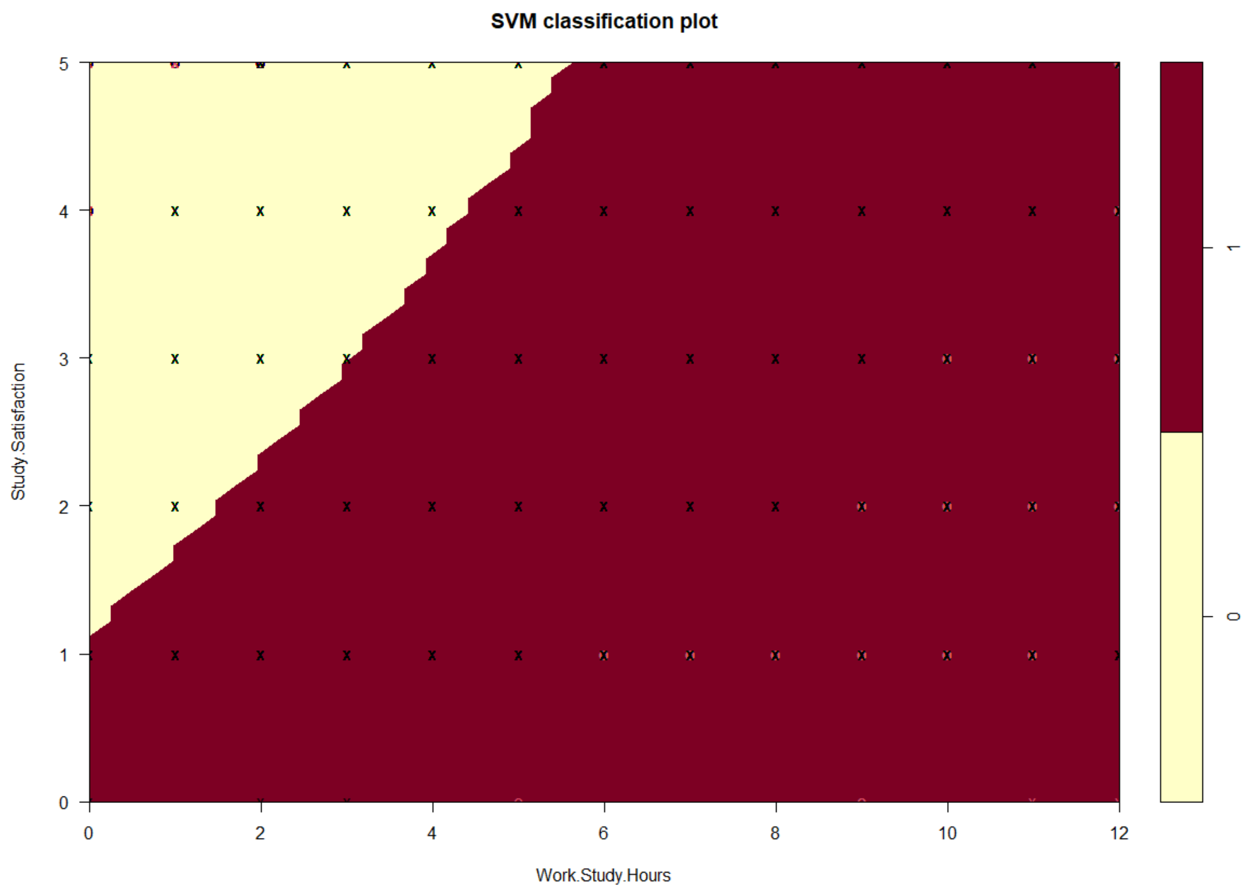
(0 - student predicted not to have depression / 1 - student predicted to have depression).

The model has a fairly equal number of support vectors in each class (5254 belonging to "no depression", 5241 belonging to "yes depression"). This leads us to believe that the model treats both classes fairly equally with the predictors it utilizes. However, this also indicates a complex model with perhaps an excessive amount of overlap, meaning that the classes are not as easily separable as we hoped. With this in mind, we decided to explore the SVM model in more detail by putting it through 10-Fold Cross-Validation. What follows is the average statistic for the folds:

CV Accuracy: 84.68%
Training Accuracy: 85.2%
Sensitivity: ~80%
Specificity: ~90%
Precision: ~85%

The model seems fairly strong for predicting whether a student has depression or not, but there is a caveat of a high false positive rate. With this in mind, there is a concern that a false positive rate means that the model is incorrectly identifying some students to have depression. However, with a roughly 85% accuracy rate total, there is a respectable job being done by the SVM here. That said, there is a concern that what we are dealing with is the medical field. The model producing a fairly high rate of false positives (around 14% of all positives are false) could actually do more harm than good. The model is by no means a substitute for a real psychiatrist or doctor. That said, it might be a good starting place to perhaps encourage one to seek more assistance. From here, it seems like the best thing to do is to see what predictors ended up being the most influential in the model.

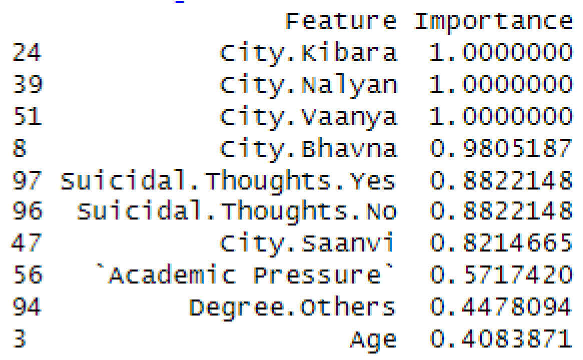
The way I first approached the idea of seeing what predictors were important was to try and draw the decision boundary. However, I quickly realized that this approach for judging variable importance might not be as useful as I had hoped:



As shown in the plot above, points overlap hard. We can sort of draw some conclusions however: people with lower study satisfaction and lower work/study hours LOOK like they are most likely to have depression, but it is hard to read this graph and even harder to get an idea.

So, given what I knew about SVM models, I chose to fit a linear model to it (knowing it would perform fairly badly) and check the coefficient weights of the linear model instead. While this is obviously not a one to one translation, it could at least indicate what predictors end up the most important.

What ended up coming out was that, unsurprisingly, suicidal thoughts was one of the most influential factors of all of them. It seemed that if someone had experienced suicidal thoughts before, they were indeed much more likely to be depressed. This is not very surprising when put into real world terms. What did however surprise me was how much location also seemed to matter. Students in certain cities were far more likely to experience depression than students in others. While I cannot necessarily know a lot about India, I have to imagine there might be financial differences or even educational differences in these cities.



```
                 Feature Importance
24           City.Kibara  1.0000000
39           City.Nalyan  1.0000000
51           City.Vaanya  1.0000000
8            City.Bhavna  0.9805187
97 Suicidal.Thoughts.Yes  0.8822148
96  Suicidal.Thoughts.No  0.8822148
47           City.Saanvi  0.8214665
56   `Academic Pressure`  0.5717420
94         Degree.Others  0.4478094
3                    Age  0.4083871
```

I was also surprised to see age get a high weight, but thinking about it more, it does make sense that teenagers are more likely to be depressed than adults, or perhaps they are more "visible" about it.

Something that made me wonder, and perhaps a different person could explore it: people who had "other" degrees (which does include undeclared) were more likely to be depressed. A part of me thinks there may be a correlation between not knowing what one wants with their future and depression. I think that would be an interesting topic to explore in more detail on a different project.

4. Conclusion

So with all of this in mind, it is time to revisit the basis of our entire research: how successfully can one predict whether a student has depression? Well, in a numerical stat, it's an 85% prediction rate. But what does that really mean in comparison to the real world? For us to judge that, we needed to see what happened in reality.

According to the Meadows Mental Health institute, roughly 74% of people with depression end up at just a "regular" primary care doctor instead of a mental health professional, resulting in a missed rate of 50%. This, on its own, makes the model look extremely successful.

Furthermore, the National Library of Medicine found that many people never even attempt to get diagnosed due to the medical costs here in America. Our model, while impersonal, is a cheap way to at least attempt to guess at whether someone is in line with other people who have depression. Encouraging someone who shows signs of depression to go to a mental health official is a great way to perhaps help the problem.

However, there are a number of problems faced with this type of data analysis: it bases itself mostly in student-answered surveys. People, in surveys, tend to be extremist. We are trained to want to answer things with 1s, 5s, or 10s. These surveys can end up skewed or inaccurate. We also have a different perspective of ourselves than how other people see us, which can mean we miss signs that other people (like a doctor, for example) can see.

Even beyond these human signs, there are problems with how some of the data ends up collected. So much of it is in factor form, and models tend to struggle with factors over numbers. Outliers, which still exist even after all the preprocessing we did to attempt to minimize them, throw the models off or create missed cases. Large datasets (like ours) take a long time to run - each SVM model took 15-20 minutes to train and test, a fairly long time to sit and wait for results each time.

Splitting the data into smaller sets or other validation approaches might help counteract the outliers and the long runtimes. Changing some of the factor variables to end up numeric at the risk of losing some data (like, instead of sleep hours being things like 7-8, simply calling them 8 and letting them be numerical) might also help the model perform better. To be honest, I would have conducted the study differently and would have had more of the factors based in numbers rather than factors.

That said, with what we had available to us, I think our model performed fairly well, and the project was mostly a success in predicting whether or not a student had depression.

5. References

National Library of Medicine's Study: https://pmc.ncbi.nlm.nih.gov/articles/PMC5769115/
Meadow's Mental Health Institute:
https://mmhpi.org/topics/educational-resources/the-cost-of-depression/
Student Depression Dataset: https://www.kaggle.com/datasets/hopesb/student-depression-dataset

6. Appendix (All the R Code)

6.1 KNN Model (coded by Tu Nguyen & Ash Kmetz)

```
data <- read.csv("Student Depression Dataset.csv", header=T)

student_depression_cleaned <- subset(data, select = -`Work.Pressure`)
student_depression_cleaned <- subset(student_depression_cleaned, select =
-`Job.Satisfaction`)
student_depression_cleaned <-
student_depression_cleaned[student_depression_cleaned$Profession ==
"Student", ]
student_depression_cleaned <- subset(student_depression_cleaned, select =
-`Profession`)
sdc <- subset(student_depression_cleaned, select = -`id`)
names(sdc)[10] <- "Suicidal.Thoughts"

colSums(is.na(sdc)) # Check for missing values - returns three missing
financial.stress values.
median(sdc$Financial.Stress, na.rm = TRUE) # grab median - returns 3
sdc$Financial.Stress[is.na(sdc$Financial.Stress)] <- 3
colSums(is.na(sdc)) # returns all 0s, so we're good there.

sapply(sdc, sd) # check to see if the data needs to be scaled.
boxplot(sdc$Age) # definitely has outliers
# I concluded age should be scaled
# concern for outliers being present while utilizing two outlier sensitive
methods
sdc$Age <- scale(sdc$Age)
names(sdc)[2] <- "Age"
sapply(sdc, sd) # check to see if the data needs to be scaled.
boxplot(sdc$Work.Study.Hours) # High standard dev but no outliers -
acceptable. No more scaling.

# Check for duplicated rows
any(duplicated(sdc)) # since this returns false, we're safe on dupe rows.
sdc$Gender <- as.factor(sdc$Gender)
sdc$City <- as.factor(sdc$City)
sdc$Sleep.Duration <- as.factor(sdc$Sleep.Duration)
sdc$Dietary.Habits <- as.factor(sdc$Dietary.Habits)
sdc$Degree <- as.factor(sdc$Degree)
sdc$Suicidal.Thoughts <- as.factor(sdc$Suicidal.Thoughts)
sdc$Family.History.of.Mental.Illness <-
as.factor(sdc$Family.History.of.Mental.Illness)
sdc$Depression <- as.factor(sdc$Depression)


#KNN:
# we are using numerical variables to predict
numeric_predictors = c("Age", "Academic.Pressure", "CGPA",
"Study.Satisfaction", "Work.Study.Hours", "Financial.Stress")
```

```
X = sdc[, numeric_predictors]
y = sdc$Depression

#dividing into train/test set
set.seed(123)
n = nrow(X)
train = sample(1:n, 0.8*n)
X.train = X[train, ]
y.train = y[train]
X.test = X[-train, ]
y.test = y[-train]

library(ISLR)
# Scale training set
X.train.scaled = scale(X.train)
# Save the mean and standard deviation
train.center = attr(X.train.scaled, "scaled:center")
train.scale = attr(X.train.scaled, "scaled:scale")
#test set is scaled the same as train set
X.test.scaled = scale(X.test, center = train.center, scale = train.scale)

library(class)
#validation set approach testing
K.set <-seq(1,401,by = 5)
knn.test.err<-numeric(length(K.set))
set.seed(123)
for (j in 1:length(K.set)){knn.pred<-knn(train = X.train.scaled, test =
X.test.scaled, cl= y.train, k = K.set[j]); knn.test.err[j]
<-mean(knn.pred!=y.test)}
#test error for best K
min(knn.test.err)

#value of best K
K.set[which.min(knn.test.err)]
```

6.2 SVM Model (coded by Liam Reilly & Dionissios Kakissis)

```
# Load packages
library(readr)
library(e1071)
library(caret)
```

```r
# Load and clean data
data <- read_csv("C:/Users/prett/Downloads/Student Depression Dataset.csv")
head(data)

student_depression_cleaned <- subset(data, select = -c(`Work Pressure`,
`Job Satisfaction`))
student_depression_cleaned <-
student_depression_cleaned[student_depression_cleaned$Profession ==
"Student", ]
student_depression_cleaned <- subset(student_depression_cleaned, select =
-c(Profession, id))
names(student_depression_cleaned)[10] <- "Suicidal.Thoughts"

# Handle missing values
colSums(is.na(student_depression_cleaned))
student_depression_cleaned$`Financial
Stress`[is.na(student_depression_cleaned$`Financial Stress`)] <- 3

# Scale numeric variables
student_depression_cleaned$Age <- scale(student_depression_cleaned$Age)
names(student_depression_cleaned)[2] <- "Age"

# Convert factors
factor_cols <- c("Gender", "City", "Sleep Duration", "Dietary Habits",
                 "Degree", "Suicidal.Thoughts", "Family History of Mental
Illness", "Depression")
student_depression_cleaned[factor_cols] <-
lapply(student_depression_cleaned[factor_cols], as.factor)

# Check for duplicates
any(duplicated(student_depression_cleaned))

# NEW CODE: 80/20 TRAIN-TEST SPLIT

set.seed(123)
split_index <- createDataPartition(student_depression_cleaned$Depression,
                                    p = 0.8, list = FALSE)
train_data <- student_depression_cleaned[split_index, ]
test_data <- student_depression_cleaned[-split_index, ]

# SVM MODEL WITH TUNING (WITH PROGRESS)

# Reduced tuning grid
```

```r
tune_grid <- expand.grid(
  cost = c(0.1, 1, 10),
  gamma = c(0.01, 0.1)
)

# Manual progress-tracking version
set.seed(123)
results <- list()
combinations <- nrow(tune_grid)

cat(paste("Tuning progress (0/", combinations, ")\n", sep=""))

for(i in 1:nrow(tune_grid)) {
  # Print progress
  cat(paste("Testing combination ", i, "/", combinations,
            " (cost=", tune_grid$cost[i],
            ", gamma=", tune_grid$gamma[i], ")\n", sep=""))

  # Train model
  model <- svm(
      Depression ~ .,
      data = train_data,
      kernel = "radial",
      cost = tune_grid$cost[i],
      gamma = tune_grid$gamma[i],
      cross = 3  # 3-fold CV
  )

  # Store results
  results[[i]] <- list(
      cost = tune_grid$cost[i],
      gamma = tune_grid$gamma[i],
      accuracy = 1 - model$tot.accuracy/100  # Convert to error rate
  )
}

# Convert results to data frame
tune_results <- do.call(rbind, lapply(results, as.data.frame))

# Find best parameters
best_params <- tune_results[which.min(tune_results$accuracy), ]

# Print the optimal values:
```

```r
cat("\nOptimal SVM Parameters:\n")
cat("Cost (C):", best_params$cost, "\n")
cat("Gamma (γ):", best_params$gamma, "\n")

# Train final model with best parameters
best_svm <- svm(
  Depression ~ .,
  data = train_data,
  kernel = "radial",
  cost = best_params$cost,
  gamma = best_params$gamma,
  probability = TRUE
)

# Print tuning results
print(best_svm)

# Evaluate on test set
svm_pred <- predict(best_svm, test_data)
conf_matrix <- confusionMatrix(svm_pred, test_data$Depression)

# Print results
print(conf_matrix)
cat("Tuned SVM Accuracy:", conf_matrix$overall['Accuracy'], "\n")

# Compare with default SVM
default_svm <- svm(Depression ~ ., data = train_data, kernel = "radial")
default_pred <- predict(default_svm, test_data)
default_acc <- mean(default_pred == test_data$Depression)
cat("Default SVM Accuracy:", default_acc, "\n")

# FINAL SVM MODEL ON FULL DATASET WITH OPTIMAL PARAMETERS

# Train final model on full dataset with best parameters
final_svm <- svm(
  Depression ~ .,
  data = student_depression_cleaned,  # Using full dataset
  kernel = "radial",
  cost = 1,        # Your optimal cost
  gamma = 0.01,    # Your optimal gamma
  probability = TRUE
)
```

```r
# 1. Summary of SVM object
cat("\nSUMMARY OF FINAL SVM MODEL:\n")
print(summary(final_svm))

# 2. Support vectors in each class
cat("\nNUMBER OF SUPPORT VECTORS IN EACH CLASS:\n")
print(final_svm$nSV)

library(caret)

# Set up 10-fold cross-validation
ctrl <- trainControl(method = "cv", number = 10)

# Re-train with cross-validation
svm_cv <- train(
  Depression ~ .,
  data = student_depression_cleaned,
  method = "svmRadial",
  trControl = ctrl,
  tuneGrid = data.frame(C = 1, sigma = 0.01),  # Your optimal parameters
  metric = "Accuracy"
)

# Print CV results
cat("\nCROSS-VALIDATION RESULTS:\n")
print(svm_cv$results)

# Key metrics:
cat("\nMEAN CV ACCURACY:", mean(svm_cv$resample$Accuracy), "\n")
cat("SD OF ACCURACY:", sd(svm_cv$resample$Accuracy), "\n")

# Predictions on same data (tends to be overly optimistic)
preds <- predict(final_svm, student_depression_cleaned)

# Confusion matrix
conf_mat <- confusionMatrix(preds, student_depression_cleaned$Depression)

cat("\nCONFUSION MATRIX (FULL DATA):\n")
print(conf_mat)

# Key metrics:
cat("\nOVERALL ACCURACY:", conf_mat$overall['Accuracy'], "\n")
cat("CLASS-SPECIFIC METRICS:\n")
```

```
print(conf_mat$byClass)

# In order to see some of the boundaries that ended up drawn, we segment
out two predictors
names(student_depression_cleaned) <-
make.names(names(student_depression_cleaned))
predictors <- c("Study.Satisfaction", "Work.Study.Hours")
svm_data_subset <- student_depression_cleaned[, c("Depression",
predictors)]

# Train SVM on two selected predictors, whatever boundary you want to see
svm_model_2d <- svm(
  Depression ~ .,
  data = svm_data_subset,
  kernel = "radial",
  cost = 1,
  gamma = 0.01,
  probability = TRUE
)

# Plot the decision boundary
plot(svm_model_2d, svm_data_subset, main = "Specific Decision Boundary")
# This proved fairly useless :( Talked about it in the report

# Create dummy variables (basically, split all factors into numericals)
dummy_data <- dummyVars(Depression ~ ., data = student_depression_cleaned)
X_dummy <- predict(dummy_data, newdata = student_depression_cleaned)
X_df <- as.data.frame(X_dummy)

# Put the response variable into this new data set
X_df$Depression <- student_depression_cleaned$Depression

# Train a linear SVM on the all-numeric values
linear_svm <- svm(
  Depression ~ .,
  data = X_df,
  kernel = "linear",
  cost = 1,
  scale = FALSE
)

# get all the coefficients on the model
weights <- t(linear_svm$coefs) %*% linear_svm$SV
```

```r
weights_df <- data.frame(
  Feature = colnames(X_df)[colnames(X_df) != "Depression"],
  Importance = as.vector(abs(weights))
)
weights_df <- weights_df[order(-weights_df$Importance), ]
head(weights_df, 10)
```