Predicting Autism in Adults Using KNN and SVM

Adriana Alvarez (2079939), Emily Gamez (2310823),

Lendy Varela (2170562), Grishma Vemireddy (2045611)

Autism spectrum disorder (ASD) is a neurological and developmental disorder that alters how a person learns, interacts, behaves, and thinks. Autism symptoms generally show up within the first 2 years of life but symptoms can become prevalent at any age (National Institute of Mental Health, 2024). Because of this, it is important that adults are screened if they begin to present any possible symptoms for ASD. However, with such a wide spectrum for the symptoms, it can be difficult to determine if an adult is presenting with ASD symptoms if they were not diagnosed at an early age. In addition, ASD is associated with costly healthcare expenditures. Waiting times for a diagnosis are lengthy and procedures are not cost effective for patients. The increase in the number of ASD cases shows that there is a need for easily implemented and effective screening methods that could help health professionals as well as individuals in deciding whether they should pursue a clinical diagnosis. This is why it is important to learn what the possible predictors are for ASD to help diagnose adults. So, can a dataset of adult information and their response to a questionnaire help predict their ASD diagnosis, and which classifier would be the best at predicting the diagnosis? If KNN and SVM is used as the supervised learning models to predict the diagnosis of adults, then the models, especially SVM, will accurately predict the ASD diagnoses.

Our dataset was obtained from the UC Irvine Machine Learning Repository. It contains 704 observations and 21 variables. Predictors A1- A10 are binary categorical variables that are answers to undisclosed screening questions. Gender and ethnicity are categorical variables. Jaundice is a binary variable based on whether the patient was born with jaundice. Country_of_res is the country where the patient resides and used_app_before notes if the patient has used a screening app before. The relation variable notes the relation of the person who completed the questionnaire. The autism variable is binary and notes if an immediate family member has been diagnosed with autism. The last categorical predictor is age_desc which notes if the patient is 18 or older. In total there are 18 categorical predictors. The dataset also contains 2 numerical variables, age and result. Where the result variable notes the score for the A1-A10 screening questions. The variable we are predicting is Class/ASD, a two-class label indicating if the patient has autism or not.

Methodology

We will use the two supervised learning models, K Nearest Neighbor (KNN) and Support Vector Machines (SVM). SVM will separate the data using a hyperplane. It will construct the optimal hyperplane that creates the greatest distance between the hyperplane and the observations. The observations that are closer to the decision boundary are support vectors. These are important because they help determine the position and direction of the separating hyperplane. SVM is also affected by the width of the margin. The margin is the distance between the decision boundary and the closest data points from each class. SVM can have a soft margin or a hard margin. A hyperplane with a soft margin will have a wider margin, meaning it will not perfectly separate the two classes. It will allow some observations to be on the incorrect

side of the margin or on the incorrect side of the hyperplane. This prevents overfitting so the model can generalize well to new data. A linear hyperplane will have one parameter, the cost value. A higher cost value indicates it is more tolerant to misclassifications (soft margin), while a lower cost value indicates it is less tolerant to misclassifications (hard margin). Furthermore, SVM can construct non-linear decision boundaries. It does so through kernel functions such as the polynomial or radial kernel. The polynomial function adds features based on the value of the degree specified. The kernel function determines the spread of the decision region based on the value of gamma specified. These kernels can make computations without explicitly working in the enlarged feature space. This is a great advantage, especially for high-dimensional data. Some advantages of the SVM model are that it works well with high dimensional data, and they can perform non linear classification with non linear data. This is because the non linear kernels don't enlarge the feature space, but rather they are computed without explicitly performing calculations on the enlarged space, which saves computation time.

Radial Kernel: $K(x_{i,}x_j) = \exp(-\gamma \sum_{k=1}^{p}(x_{ik} - x_{jk})^2), \gamma > 0$

Polynomial Kernel: $K(x_{i,}x_j) = (1 + \sum_{k=1}^{p} x_{ik}\, x_{jk})^d$

Linear Kernel: $K(x_{i,}x_j) = \sum_{k=1}^{p} x_{ik}\, x_{jk}$

Lastly, we evaluate the performance of the model through cross validation, which can also help us select the optimal parameter values. The KNN method can be used to determine the k-nearest region that results in a higher chance of being classified into "Class/ASD" or not. KNN classifies data points by finding the K closest points in the dataset based on Euclidean distance. The parameter K represents the number of nearest neighbors to be considered. Having a smaller K value could make the model more likely to be sensitive to noise, but having a larger K value could potentially oversimplify the classification. So, choosing the appropriate K value is crucial. We can use this method to determine non-linear decision boundaries that can be caused by different factors contributing to a higher chance of having an ASD diagnosis. One of the disadvantages of using this model is the effect noise or outliers will have on it. KNN could also be computationally expensive when dealing with larger datasets.

Data Analysis

Before we began training our models, we had to preprocess our data and understand what features we want influencing the model. First, we dealt with the issue of missing data. The only features with missing data were age, ethnicity, and relation. Only two observations had age as a missing feature, so we chose to exclude those observations from our final data set. While we could have averaged the age values or picked the median age, this may have skewed more with our data than simply omitting the observations. Now, for ethnicity and relation, these are both categorical variables so we cannot average the values. 95 observations had both features missing, so we also omitted these observations from our final data set. Now, we examine the data closely to ensure everything aligns with the expected parameters. First, there was an observation with an age value of 383. This is not logically correct, so this observation was emitted. Secondly, in the ethnicity feature, there were two overlapping factors: ‘others’ and ‘Others’. We combined the two factors into one as they represented the same value. Now that the data is filtered, the last step is to decide what features we want our model to be influenced by. We want to keep the features that represent the answers to the questionnaire (A1-A10) as these responses are the base for

helping diagnose ASD. The feature “age_desc” is a constant categorical predictor that describe the patient’s age. This is not a necessary feature since we already have an “age” feature, so this feature was omitted. There is also another feature, “used_app_before”, a categorical feature tracking if the patient used the screening app before. This feature was omitted as the scope of our project is simply helping diagnose adults with ASD. Similarly, the “relation” feature, a categorical variable that described who the observation was, was dropped for the same reason. While examining the relationship between a medical professional to an average person would be a good idea for another project, it is not in our scope. Lastly, we dropped “country_of_res” and “ethnicity” and kept the remaining features. These two categorical variables provided too much niche data, and we wanted to simplify our data to more simply base it off the questionnaire rather than their background and country of residence. Our final dataset consists of 608 observations and 16 variables.

Now that the data has been preprocessed, let’s look at the distribution of our data to make sure there is good distribution and a wide range of observations.

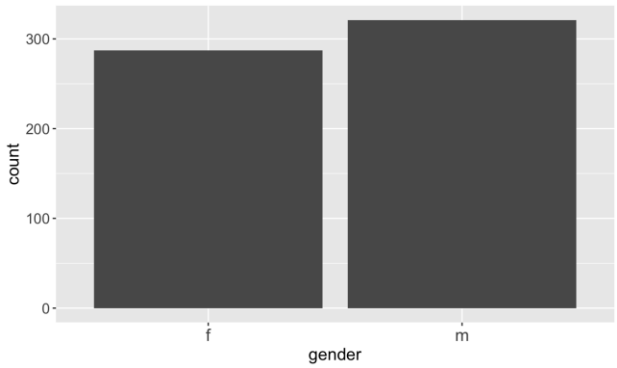

Figure 1: gender distribution

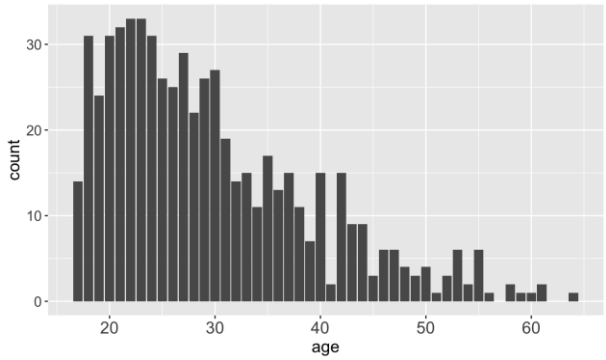

Figure 2: age distribution

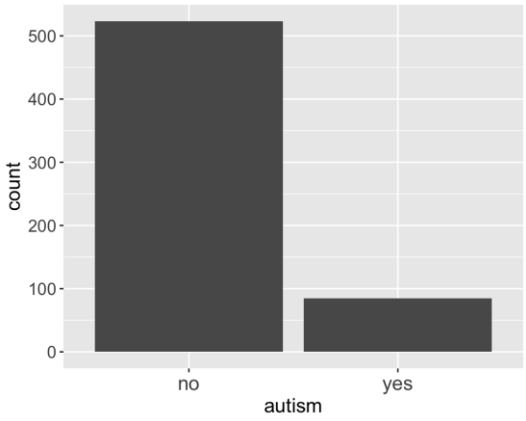

Figure 3: immediate family with autism distribution

Looking at these graphs, it is clear we have a good balance between female and male patients. In addition, age is widely dispersed but skewed to the right, with mostly concentrated in

the 20-30s range. There are some observations past age 60, which will provide some good nuance to the models. In our final data, there is a majority of people who do not have an immediate family of with autism—around 86% of the observations. This would be a good piece of data to understand if there is a genetic relationship between a family member having autism and the likelihood of being diagnosed with autism. For the scope of this project, it will be one of the many variables that will help predict an ASD diagnosis.

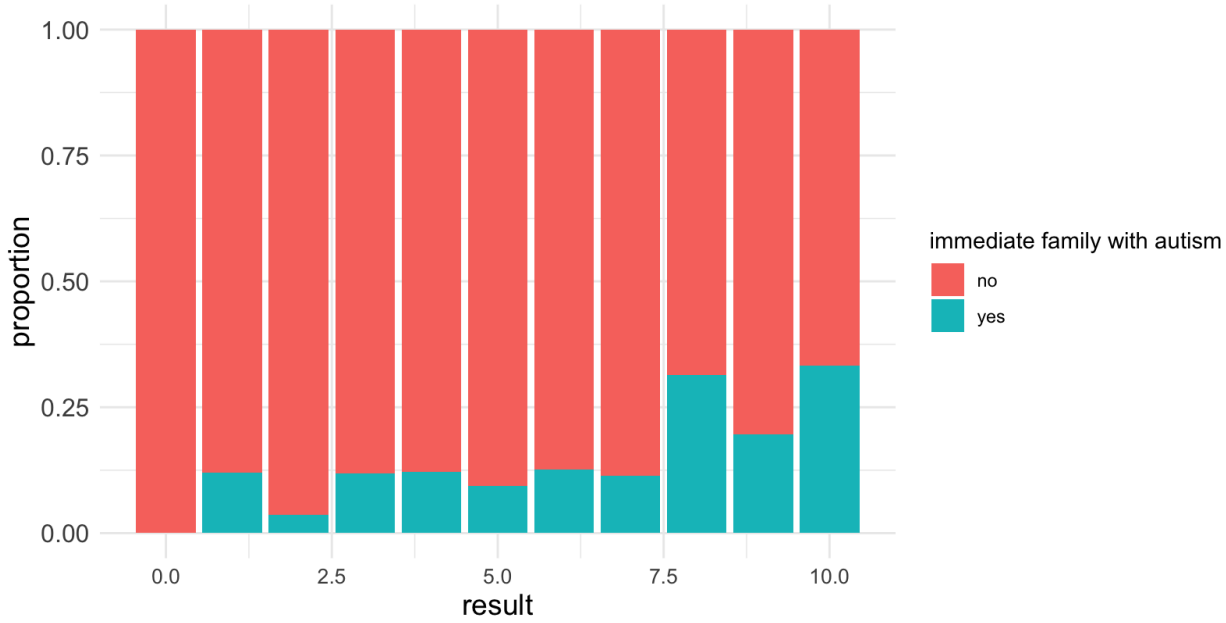

Figure 4: proportion of result and immediate family with autism

Lastly, we looked at the proportion between the observation's score on the questionnaire and if there is an immediate family with autism. There is a general trend that the higher the result the more likely an immediate family has autism, but only up to around 30%. Surprisingly, no observations that scored a 0 have an immediate family member with autism. This is very surprising because scoring a 1 brings the proportion up to 0.125. Now with a better understanding of how the data is distributed and the factors that will play into our algorithm's classifications, we began to train our data with two supervised learning algorithms: SVM and KNN.

1. SVM (Lendy Varela and Adriana Alvarez)

The SVM model was trained on 80% of the data and tested on 20% of the remaining data. Moreover, a set.seed(1) was used across all models and train-test splits to maintain consistent results.

Although SVM is one of many machine learning algorithms, it's hyper parameter tuning feature yields various models that perform differently based on the choice of hyperparameters. For this reason, cross validation was used as it helps with selecting the best choice of parameters that will yield the lowest training result. We implemented three different kernel functions, radial, polynomial and linear, which all have different hyper parameters.

The SVM model with radial kernel performed cross validation to find the best cost value among 0.001, 0.01, 0.1, 1, 5, 10, and 100, and the best gamma value among 0.5, 1, 2, 3 and 4. The tuning function created 35 different models, which it found that the cost and gamma values that yielded the lowest training error of 0.0432 are a cost of 5 and a gamma of 0.5. Looking at the summary of this optimal model, we see that there are 267 supporting vectors, 117 which are from the class NO and 90 which are from the class YES. To test the performance of this model on unseen data, we made predictions on the test data. The confusion matrix was created, which showed that 115 observations were classified correctly, with 7 observations being misclassified. Hence, this model had a test error of 0.06 and test accuracy of 0.94, which is pretty good, but we would like to see if we can get better results. One thing to note is that the model has a higher false negative rate than false positive rate, with 6 cases being false negative cases. This could be an indication that classifying Autism is a bit more challenging for this model. False negatives are particularly unwanted in medical scenarios, since we would like to detect Autism to begin treatment or medications to reduce the symptoms that interfere with the patient's daily life.

```
svmrad.tune = tune(METHOD = svm, `Class/ASD` ~ ., data = train_data,
                   kernel = "radial",
                   ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                                 gamma = c(0.5, 1, 2, 3, 4)))
         Actual
Predicted NO YES
      NO  82   6
      YES  1  33
```

The second SVM model with polynomial kernel also performed cross validation to find the optimal cost among 0.001, 0.01, 0.1, 1, 5, 10, and 100 and the optimal degree value among 2, 3, 4, 5, and 6. The tuning function created 35 different models, which found that the parameters that the parameters that yielded the lowest training error of 0 are a cost of 5 and a degree of 2. Looking at the summary of this optimal model, we observe that this model contains 75 support vectors, a great decrease from the number of support vectors than the SVM model from before. There are 39 support vectors from class NO and 36 support vectors from class YES. To test the performance of this model, predictions were made on the test data. The confusion matrix showed that the 122 observations were correctly classified. Therefore, this model has a tests error of 0 and a test accuracy of 1, an improvement from the previous SVM model.

```
svmpoly.tune = tune(METHOD = svm, `Class/ASD` ~ ., data = train_data,
                    kernel = "polynomial",
                    ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                                  degree = 2:6))
         Actual
Predicted NO YES
      NO  83   0
      YES  0  39
```

The third SVM model with linear kernel also performed cross validation to find the optimal cost value among 0.001, 0.01, 0.1, 1, 5, 10, and 100. The tuning function created 7 different models, and it found that the parameter that yielded the lowest training error of 0 is a cost of 1. Looking at the summary of this optimal model, we see that there are 58 supporting vectors. With 30 from the class NO and 28 from the class YES. This is a significant decrease in supporting

vectors from the previous two models. Since the training accuracy is perfect, it's important to test the performance of this model on unseen data, as we would like to determine if the model is overfitting the training data. After making predictions on the test data, the confusion matrix showed that the model had a test error of 0 and test accuracy of 1, meaning there was no misclassification on the test data either. This indicates that a simpler linear hyperplane performed better than using a more complex hyperplane.

```
svm_tune <- tune(METHOD = svm, `Class/ASD` ~ ., data = train_data,
                 kernel = "linear",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
```

```
         Actual
Predicted NO YES
      NO  83   0
      YES  0  39
```

2. KNN (Grishma Vemireddy and Emily Gamez)

For our KNN model, we manipulate our dataset so we can work better with the data. We clean the data to remove missing data and exclude columns such as age_desc, relation, country_of_res, ethnicity, and used_app_before as we thought they were not relevant. This cleaning process left us with 16 variables. Next, we will transform all the response variables to binary then to numeric. Since our variables are not on the same scale, we will also scale the data so it can produce more meaningful and accurate results. Scaling ensures that all features contribute equally to the distance computations in KNN. Without scaling, features with larger ranges could dominate the distance measurement, leading to biased results.

Validation Set Approach

Our initial step involved the meticulous process of choosing the most effective tuning parameters for the K-Nearest Neighbors (KNN) model. For that, we used the validation set approach initially. We randomly split the dataset with 80% for training set and 20% for testing set. This allows our model to predict responses from our test set and analyze the accuracy of our model on the K value. To determine the optimal K value, we decided to run a range of values from K = 1 to K = 50.

We proceeded by training the model and selecting the optimal K that yielded the smallest test error, code shown below:

```
k.set <- seq(1, 50, by=2)
mc.rates <- numeric(length(k.set)
for (i in seq_along(k.set)) {
  k <- k.set[i]
  knn_pred <- knn(train = X.train.aut, test = X.test.aut, cl = y.train.aut, k = k)
  mc.rates[i] <- mean(knn_pred != y.test.aut)}

> min(mc.rates)
[1] 0.01639344
> ind.min <- which.min(mc.rates)
```

```
> best_k <- k.set[ind.min]
> best_k
[1] 47
```

From our results, K = 47 gives us the best test error of 1.63% using the validation set approach.

Confusion Matrix

```
Confusion matrix for k = 5:
> print(confusion_matrices[[k_index]])
          Actual
Predicted  0  1
        0 81  4
        1  2 35
```

```
Confusion matrix for k = 47:
> print(confusion_matrices[[k_index]])
          Actual
Predicted  0  1
        0 83  2
        1  0 37
```

Overall fraction of correct predictions for k = 5 is 0.950819 based on the confusion matrix.

The confusion matrix illustrates the performance of K-Nearest Neighbors (KNN) algorithm for different k values in classifying ASD diagnosis as with or without autism. With k values ranging from 1 to 50, the model consistently demonstrates high accuracy, achieving an overall fraction of correct predictions between 94.26% and 98.36%. Notably, the confusion matrix for K = 5 shows 81 true negatives, 35 true positives, 4 false positives, and 2 false negatives indicating effective diagnosis of CLASS/ASD. These results suggest that the KNN algorithm performs well in accurately categorizing the diagnosis of ASD with minimal misclassification.

Leave-one-out Cross Validation (LOOCV) Approach

We proceed by implementing another cross-validation technique. In LOOCV, each observation in the training set is used as a validation set, while remaining data points are used for training. The model is trained by iteratively leaving out one data point. The error for each iteration is computed by comparing the predicted class label to the true class label, and the overall error is averaged. LOOCV is considered a better approach for validating the performance of a model because it generally produces a less biased estimate of model performance by using almost the entire data set for training.

Code below:

```
K.set2 <- 1:50
Knn.test.err2 <- numeric(length(K.set2))
for (j in 1:length(K.set.f)){
  print(j)
  set.seed(1)
  knn.pred.f <- knn.cv(train=X.train.full,
                       cl=y.train.full,
                       k=K.set.f[j])
  knn.test.err.full[j] <- mean(knn.pred.f != y.train.full)
}
# Finding the optimal K
> K.set.f[which.min(knn.test.err.full)]
```

```
[1] 32
> loocv_error_rate <- mean(knn.test.err.full)
> loocv_error_rate
[1] 0.02871711
```

For our results, the optimal K is 32 and the test error is 2.87% for LOOCV.

Comparing SVM with KNN

Comparing the SVM models with the KNN model we find that SVM with linear or polynomial kernel performed better than SVM with radial kernel and the KNN model using LOOCV. After tuning SVM with linear or polynomial kernel, it achieved a perfect accuracy score of 100%. However, the KNN model using LOOCV performed better than the SVM model using radial kernel. This could indicate that for this dataset, more complex models don't perform as well compared to models that are much simpler. It is likely that the more complex models are overfitting the training data by creating overly complex decision boundaries, which results in slightly higher testing error rates.

Best Model – SVM with a Linear Kernel

After comparing the two different supervised learning algorithms, it's clear that the SVM algorithm with a linear kernel performed the best with the highest test accuracy of 1. Now, we perform SVM on the entire dataset to get the following results.

```
svm(formula = `Class/ASD` ~ ., data = data_autism, kernel = "linear",
    cost = 1)
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
Number of Support Vectors:  67
 ( 37 30 )
Number of Classes:  2
Levels:
 NO YES
         Actual
Predicted  NO YES
      NO  428   0
      YES   0 180
```

The supervised model performed perfectly like with the original test data split, correctly classifying all the observations. This model performed significantly better than the application's classification that incorrectly classified 50% of the observations that have autism. There is a total of 67 support vectors—37 "NO" and 30 "YES". 428 observations were correctly classified as "NO" and 180 observations were correctly classified as "YES".

To visualize this model, we needed to reduce the dimensionality of our dataset. We cannot properly show the decision boundary with many features. We created a plot that behaves similarly to our actual data by reducing the dimensionality with Principal Component Analysis (PCA). With PCA, we reduced the number of features down to two to visualize the similar

product. Unfortunately, this did not create the exact model as two observations were incorrectly classified—one false positive and one false negative—and there were only 55 support vectors. . However, it does provide an almost accurate decision plot.

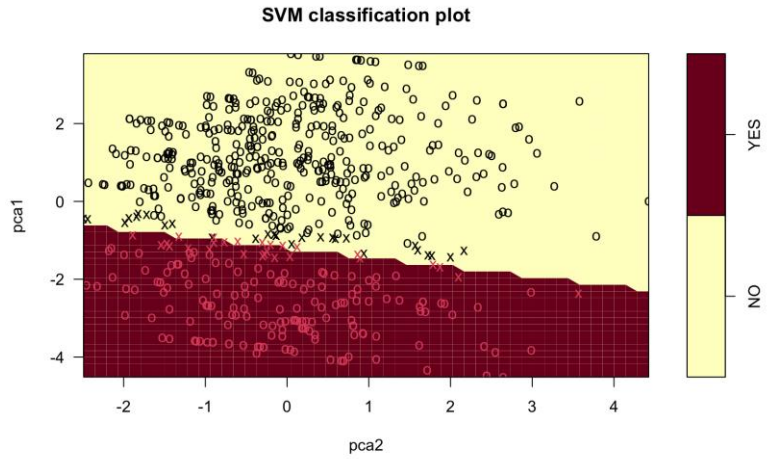

Figure 5: SVM Classification Plot

Looking at the figure above, there is a linear boundary separating the "NO" and "YES" observations. While not exactly clear, the two misclassified observations are likely along the decision boundary. The 55 support vectors lie closely to the decision boundary with 28 for "NO" and 27 for "YES".

Conclusion

Originally, the dataset's classifier "Class_ASD" had a poor performance, wrongly classifying 50% of the observations that have autism. We improved its performance by using the dataset of observations and training them on two different supervised learning algorithms: SVM and KNN. Overall, SVM performed the best with the linear kernel and a parameter cost of 1 having a test accuracy of 1. We used this model on the entire dataset and it correctly classified all of the observations. Thus, we can predict an observation's ASD diagnosis using personal data and responses to a questionnaire.

While the model did perform with 100% accuracy on the entire clean dataset, our data analysis could still be improved. The first issue was with missing values. By simply omitting the observations entirely, our model does not capture the information of those 95 observations. Even though the model did perform with a test accuracy of 1 on the test dataset, our model may not perform as well with new data than if we kept the observations. Our data analysis could be improved if we tested how cleaning the data would affect our final models. Overall, the final result of our project resulted in a model that correctly classified all of the observations in our final dataset, proving that an SVM classifier would be the best predictor of an adult's ASD diagnosis using a dataset of adult information and their response to a questionnaire.

References:

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.

National Institute of Mental Health. (2024). Autism spectrum disorder. National Institute of Mental Health. https://www.nimh.nih.gov/health/topics/autism-spectrum-disorders-asd

Thabtah, F. (2017). Autism Screening Adult [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5F019.

Appendix: additional figures, R code, etc.

R source code: **SVM and KNN R Code**

Link to dataset: https://archive.ics.uci.edu/dataset/426/autism+screening+adult

```
summary(data_autism)
 A1_Score A2_Score A3_Score A4_Score A5_Score A6_Score A7_Score A8_Score
 0:158    0:322    0:315    0:291    0:288    0:421    0:347    0:203
 1:450    1:286    1:293    1:317    1:320    1:187    1:261    1:405

 A9_Score A10_Score      age            gender         jundice
 0:400    0:244     Min.   :17.00   Min.   :0.000   Min.   :0.00000
 1:208    1:364     1st Qu.:22.00   1st Qu.:0.000   1st Qu.:0.00000
                    Median :27.00   Median :1.000   Median :0.00000
                    Mean   :29.63   Mean   :0.528   Mean   :0.09704
                    3rd Qu.:35.00   3rd Qu.:1.000   3rd Qu.:0.00000
                    Max.   :64.00   Max.   :1.000   Max.   :1.00000
    austim           result       Class/ASD
 Min.   :0.0000   Min.   : 0.000   NO :428
 1st Qu.:0.0000   1st Qu.: 3.000   YES:180
 Median :0.0000   Median : 5.000
 Mean   :0.1398   Mean   : 5.084
 3rd Qu.:0.0000   3rd Qu.: 7.000
 Max.   :1.0000   Max.   :10.000
```