Allen Maredia 1964515
Dinh Bui 2220491
Devon Chao 1568011

Math 4323, Project: Final Report

1.Introduction (Devon Chao)

Stroke is a critical global health concern, standing as one of the most serious health crises worldwide. It stands as the second leading cause of death worldwide, accounting for approximately 11% of total deaths, as reported by the World Health Organization (WHO). Strokes not only affect individuals but also present significant challenges to healthcare systems and global public health initiatives worldwide. Given strokes' global impact, it's crucial to use data to establish risk thresholds, develop predictive models, and implement preventive measures. This report utilizes a dataset obtained from Kaggle, a platform hosting various datasets for research and analysis. Given its significant impact, this dataset is selected to predict the likelihood of strokes in patients, based on input parameters such as gender, age, existing health conditions, and smoking status.

2. Methodology (Allen Maredia, Dinh Bui, Devon Chao)

SVM and KNN are distinct classification algorithms in machine learning. SVM seeks the best hyperplane separating classes with maximum margin, handling linear and non-linear data via choosing a suitable kernel (e.g., linear, polynomial, or radial basis function). In contrast, KNN operates by predicting based on the majority class among 'k' nearest neighbors, its flexibility determined by 'k' and using distances to define data point similarity. The most common approach is to use a distance metric like Euclidean distance, but other metrics such as Manhattan distance or Minkowski distance can also be used. SVM prioritizes margins and support vectors, while KNN relies on local data relationships for predictions. Their selection depends on dataset characteristics and desired interpretability, with SVM excelling in complex separations and KNN's simplicity advantageous for smaller datasets.

SVM: Hyperplane B0 + B1X1 + B2X2 = Y

where B0, B1, B2 are some fixed parameters.
Y is response variable "stroke", X1 is predictor "bmi", X2 is predictor "age"

Advantages of SVM:
Effective in High-Dimensional Spaces:SVMs perform well in high-dimensional spaces, making them suitable for problems with a large number of features, such as predicting strokes using multiple patient attributes.

<u>Robust to Overfitting</u>:They are less prone to overfitting compared to more flexible models like decision trees.

Disadvantages of SVM:
<u>Sensitivity to Noise and Outliers</u>:SVMs are sensitive to noise in the dataset and outliers. Outliers can significantly impact the position of the hyperplane, leading to suboptimal results.
<u>Computational Intensity</u>: Training an SVM can be computationally intensive, especially when dealing with large datasetsSelection of Kernels:The choice of the kernel and associated parameters can significantly affect the performance of SVMs. Selecting the right kernel and tuning parameters may require experimentation.

Advantages of KNN:
<u>Simple and Intuitive</u>: KNN is easy to understand and implement. It's a simple algorithm that doesn't require complex assumptions or parameter tuning.
<u>Robust to Outliers</u>: KNN is less sensitive to outliers compared to some other algorithms. Outliers may have less influence on the model since predictions are based on the majority class or average of neighbors.

Disadvantages of KNN:
<u>Computational Complexity</u>: As the size of the dataset grows, the computational cost of finding the nearest neighbors increases. KNN needs to calculate distances for each query instance, which can be computationally expensive.
<u>Sensitive to Irrelevant Features</u>: KNN considers all features to calculate distances. If some features are irrelevant or noisy, they can negatively impact the performance of the algorithm.
Optimal k Selection: The choice of the hyperparameter k is crucial. An inappropriate value of k may lead to overfitting or underfitting. Optimal k selection often requires experimentation and tuning

3. Data Analysis (Allen Maredia, Dinh Bui)

3.a)
We chose to exclude predictors such as the ID since it is not expected to contribute to the outcome. Additionally we decided to scale the features age, hypertension, heart_disease, avg_glucose_level, and bmi. This was due to the fact that these features were on different scales which is detrimental to the KNN algorithm since it relies on distance calculations.

SVM: (Dinh Bui)

```
dat <- dat[!dat$bmi == "N/A", ] #ignore where bmi is null
dat$id <- NULL #id is not necessary

#Change data type for svm performance
cols <- c("gender", "hypertension", "heart_disease", "ever_married", "work_type",
"Residence_type", "smoking_status")
dat[cols] <- lapply(dat[cols], factor)
dat$bmi <- as.numeric(dat$bmi)
```

```
dat <- dat[!dat$gender == "Other",]

dat$stroke <- as.factor(dat$stroke)
```

3.b.i)

Split Data

```
n <- nrow(dat)
set.seed(1)
train <- sample(1:n, 0.8*n)

library(e1071)
```

3.b)

#Perform svm with different cost, gamma values

```
> set.seed(1)
> tune.out <- tune(svm, stroke~., data = dat[train,], kernel = "radial", ranges
=
+                    list(cost=c(0.1,1,10,100),gamma=c(0.5,1,2,3,4)))

> summary(svm.radial)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  0.1   0.5
- best performance: 0.04253583 #Cross-Validation Error Rate
```

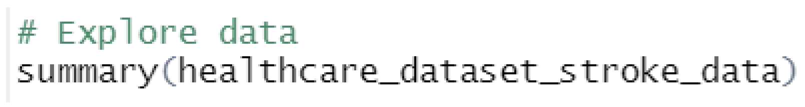
3.b.ii)

#Find test error

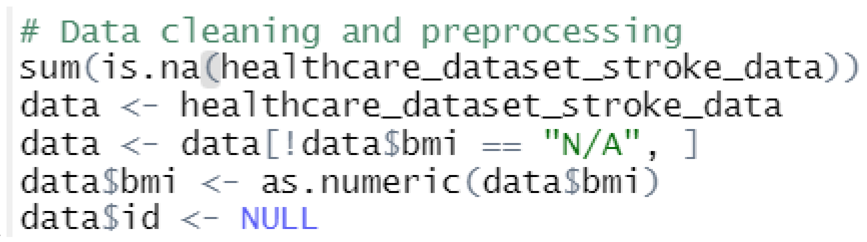
```
> svm.predict <- predict(tune.out$best.model, newdata = dat[-train,])
> mean(svm.predict != dat$stroke[-train])
[1] 0.04276986 #Test Error Rate
```

=> Accuracy = 95.72

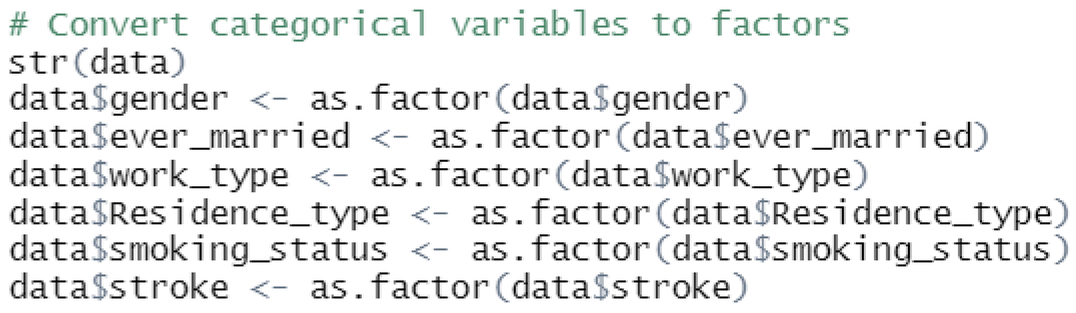
KNN: (Allen Maredia)



```
# Explore data
summary(healthcare_dataset_stroke_data)
```

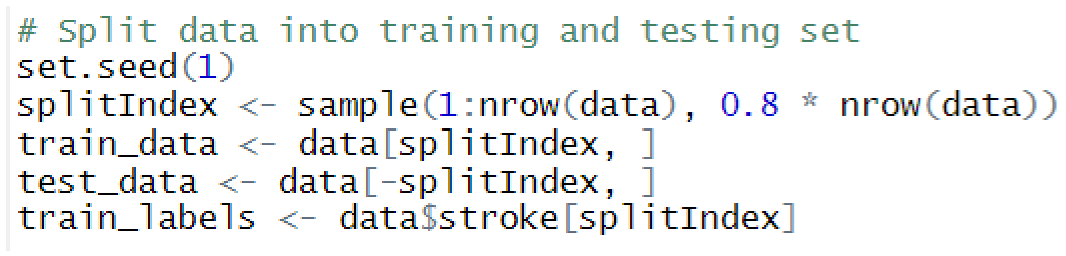


```
# Data cleaning and preprocessing
sum(is.na(healthcare_dataset_stroke_data))
data <- healthcare_dataset_stroke_data
data <- data[!data$bmi == "N/A", ]
data$bmi <- as.numeric(data$bmi)
data$id <- NULL
```

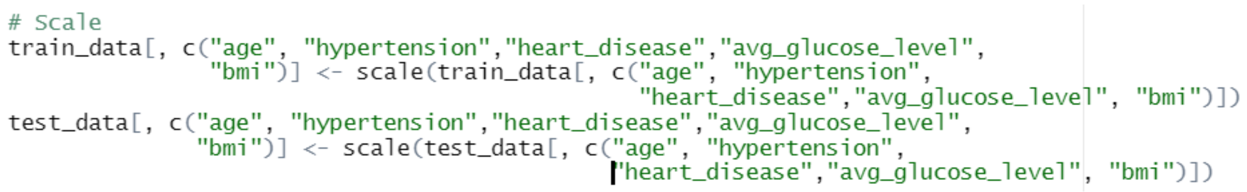


```
# Convert categorical variables to factors
str(data)
data$gender <- as.factor(data$gender)
data$ever_married <- as.factor(data$ever_married)
data$work_type <- as.factor(data$work_type)
data$Residence_type <- as.factor(data$Residence_type)
data$smoking_status <- as.factor(data$smoking_status)
data$stroke <- as.factor(data$stroke)
```

3.b.i)

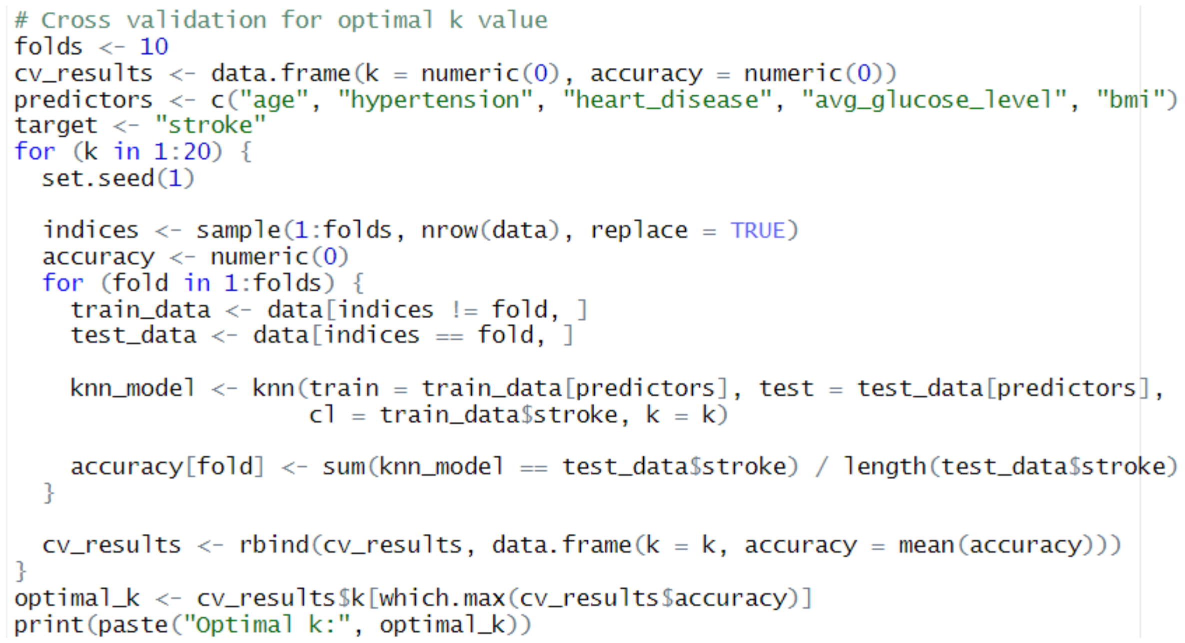


```
# Split data into training and testing set
set.seed(1)
splitIndex <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- data[splitIndex, ]
test_data <- data[-splitIndex, ]
train_labels <- data$stroke[splitIndex]
```

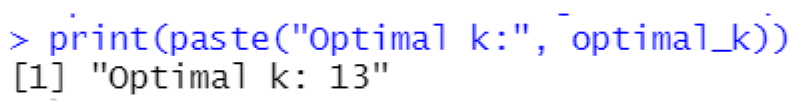


```
# Scale
train_data[, c("age", "hypertension","heart_disease","avg_glucose_level",
               "bmi")] <- scale(train_data[, c("age", "hypertension",
                                               "heart_disease","avg_glucose_level", "bmi")])
test_data[, c("age", "hypertension","heart_disease","avg_glucose_level",
              "bmi")] <- scale(test_data[, c("age", "hypertension",
                                             "heart_disease","avg_glucose_level", "bmi")])
```

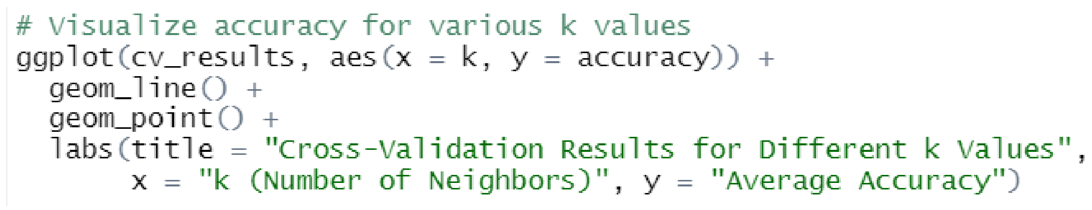
3.b)



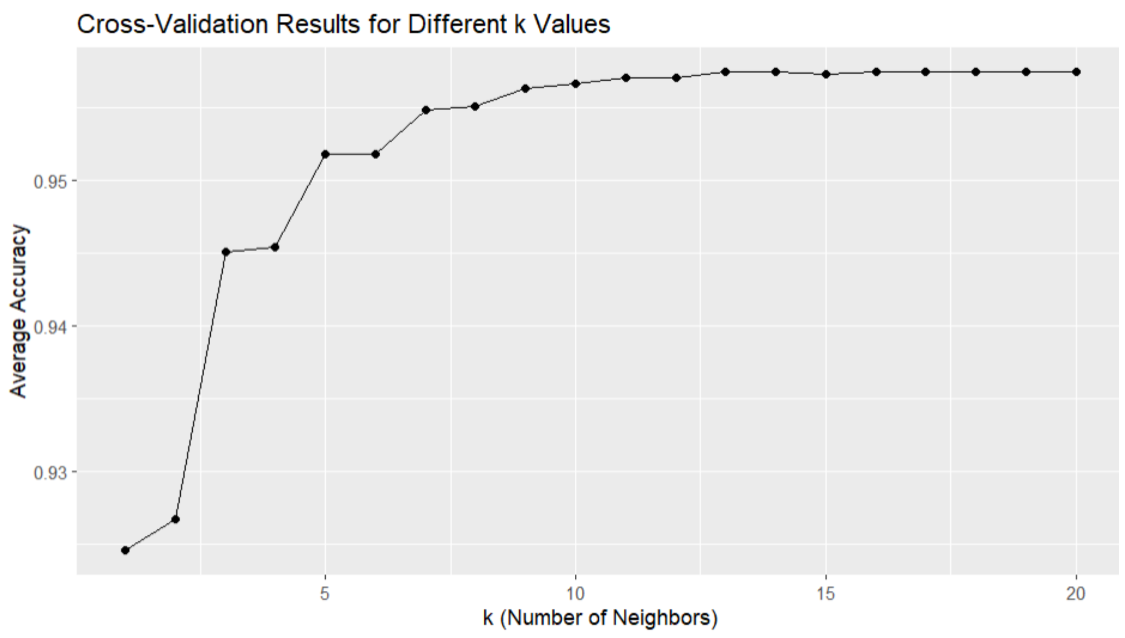
```
# Cross validation for optimal k value
folds <- 10
cv_results <- data.frame(k = numeric(0), accuracy = numeric(0))
predictors <- c("age", "hypertension", "heart_disease", "avg_glucose_level", "bmi")
target <- "stroke"
for (k in 1:20) {
  set.seed(1)

  indices <- sample(1:folds, nrow(data), replace = TRUE)
  accuracy <- numeric(0)
  for (fold in 1:folds) {
    train_data <- data[indices != fold, ]
    test_data <- data[indices == fold, ]

    knn_model <- knn(train = train_data[predictors], test = test_data[predictors],
                     cl = train_data$stroke, k = k)

    accuracy[fold] <- sum(knn_model == test_data$stroke) / length(test_data$stroke)
  }

  cv_results <- rbind(cv_results, data.frame(k = k, accuracy = mean(accuracy)))
}
optimal_k <- cv_results$k[which.max(cv_results$accuracy)]
print(paste("Optimal k:", optimal_k))
```



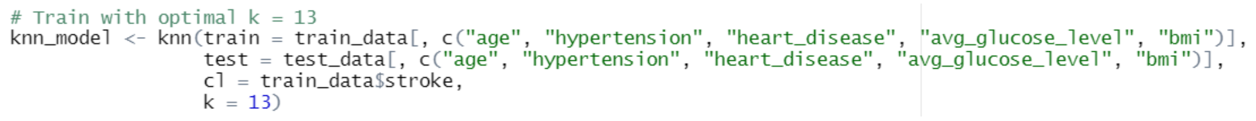
```
> print(paste("Optimal k:", optimal_k))
[1] "Optimal k: 13"
```



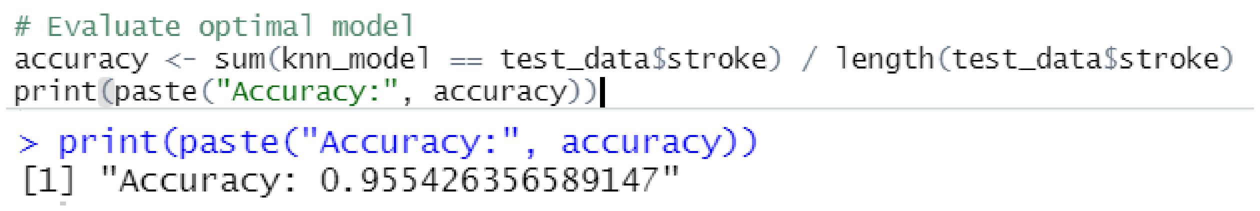
```
# Visualize accuracy for various k values
ggplot(cv_results, aes(x = k, y = accuracy)) +
  geom_line() +
  geom_point() +
  labs(title = "Cross-Validation Results for Different k Values",
       x = "k (Number of Neighbors)", y = "Average Accuracy")
```



3.b.ii)



```
# Train with optimal k = 13
knn_model <- knn(train = train_data[, c("age", "hypertension", "heart_disease", "avg_glucose_level", "bmi")],
                 test = test_data[, c("age", "hypertension", "heart_disease", "avg_glucose_level", "bmi")],
                 cl = train_data$stroke,
                 k = 13)
```



```
# Evaluate optimal model
accuracy <- sum(knn_model == test_data$stroke) / length(test_data$stroke)
print(paste("Accuracy:", accuracy))
```

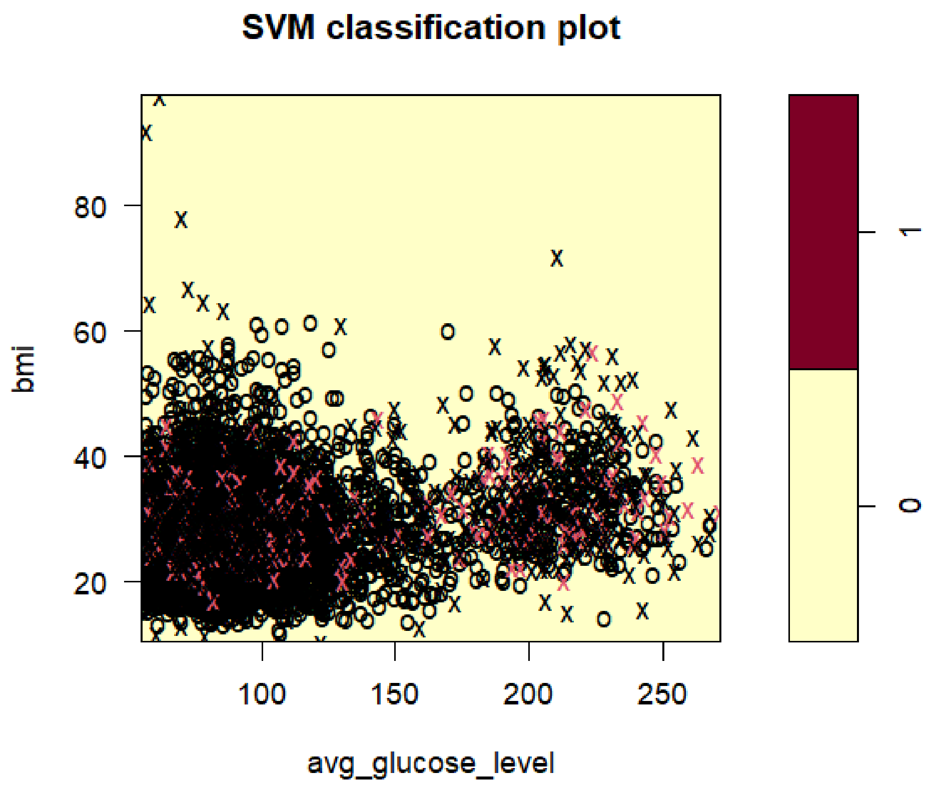
```
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.955426356589147"
```

3.c)
The prediction test error rate for the SVM model was 95.72 while the prediction test error rate for KNN was 95.542. The SVM model performed slightly better than the KNN model.

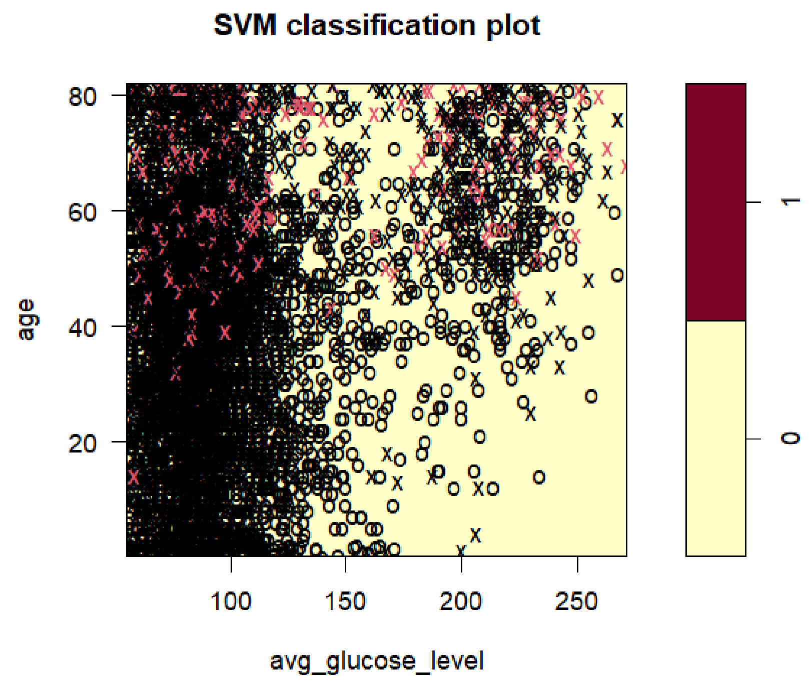
3.d)
SVM is best model in part c.

```
> svm.radial <- svm(stroke~., data=dat, kernel="radial", cost = 0.1, gamma = 0.5)
> summary(svm.radial)

Call:
svm(formula = stroke ~ ., data = dat, kernel = "radial", cost = 0.1,
    gamma = 0.5)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
```

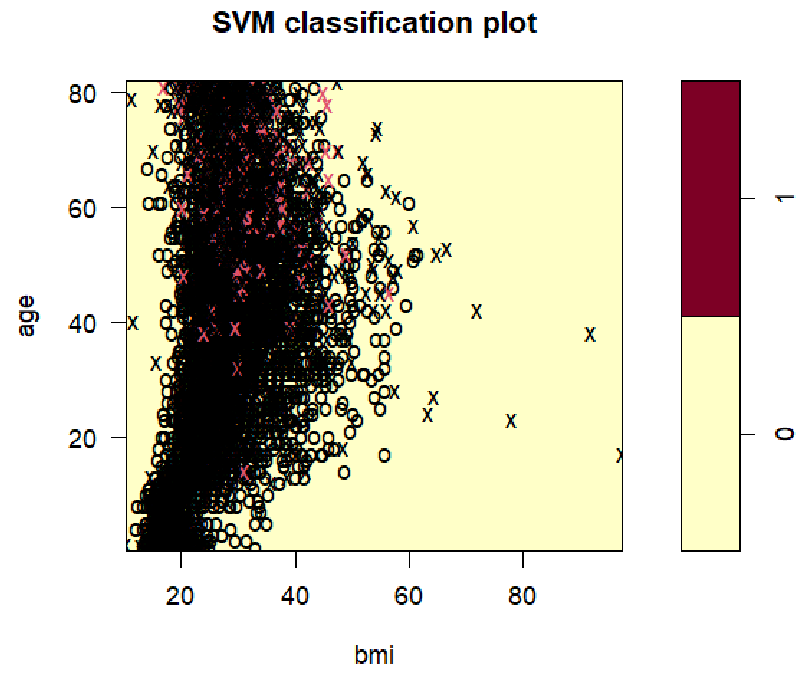
```
       cost:  0.1

Number of Support Vectors:  911

 ( 209 702 )


Number of Classes:  2

Levels:
 0 1
> plot(svm.radial, dat, bmi~avg_glucose_level)
```



```
> plot(svm.radial, dat, age~avg_glucose_level)
```



```
> plot(svm.radial, dat, age~bmi)
```



4. Conclusion (Devon Chao)

Based on the models created, factors like hypertension, BMI, and smoking had a direct impact on chance for stroke. There are many procedures that can improve data analysis, many of which have already been done during the making of the current model such as using optimal k for knn and kernel for svm. However there are a couple other procedures that can be taken to improve data analysis further.

For SVM:

Cross-Validation: Employ techniques like cross-validation to evaluate the model's performance and fine-tune hyperparameters effectively. Grid search or random search for hyperparameter optimization can enhance the model's generalization.

Handling Imbalanced Data: If dealing with imbalanced classes, use techniques like oversampling, undersampling, or employing class weights to handle the imbalance, improving the SVM's ability to predict minority classes.

For KNN:

Distance Metric Selection: Test different distance metrics (Euclidean, Manhattan, etc.) to identify the most suitable one for the dataset. Some datasets might exhibit better separation with a specific distance metric.

Dimensionality Reduction: Implement techniques like PCA (Principal Component Analysis) or LDA (Linear Discriminant Analysis) to reduce dimensionality, removing noise and improving KNN's performance.

Ensemble Methods: Consider ensemble techniques like bagging or boosting to combine multiple KNN models or improve the robustness of predictions.

Implementing these procedures while considering the characteristics of the dataset can help fine-tune SVM and KNN models, potentially enhancing their predictive accuracy.

5. References

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/discussion/343916