Ryder Fiechter 1942883
Kobe Runnels 1873204
Jaskaran Singh 2027724
Moataz Altaweel 1800547
Brett Mclain 2130575

MATH 4323 Final Report

Introduction

The aim of our project is to use standard hospital testing and specific classifiers gathered from patients to predict whether a person has diabetes. Accurately detecting diabetes can improve patients' health through preventative care and medication, ultimately providing a higher standard of living. The data set we are working with was collected in 2019 and includes information such as age, gender, family history, blood pressure, BMI, drug abuse history, hours slept, junk food intake, and stress levels. This data will help create a system that can confidently recommend further testing to catch the disease early and also identify potential risk areas and habits that can be prevented. The final response variable is whether the patient has diabetes or not. The study includes 952 participants aged 18 and above, with 580 males and 372 females. Participants completed a questionnaire(see appendix) about factors that could lead to diabetes. The main question we seek to answer is which classifier(KNN or SVM) has better accuracy in predicting diabetes.

Methodology

We used KNN and SVM to help predict diabetes for the patients. KNN is a type of supervised machine learning algorithm. During the training phase of the KNN algorithm, the model is fitted to the training data by storing all the feature vectors and their corresponding labels. The feature vectors are essentially the input data points in the training set and their labels are the corresponding output values. When a new input data point is presented to the algorithm, KNN finds the K nearest neighbors of the new data point in the training set based on a distance metric such as Euclidean distance. The K nearest neighbors are determined by calculating the distance between the new data point and all the data points in the training set, and then selecting the K data points with the smallest distances. Once the K nearest neighbors are identified, the KNN algorithm predicts the output value of the new data point by taking the majority vote of the labels of its K nearest neighbors (for classification problems) or the mean of the labels (for regression problems).

KNN is a simple and easy-to-understand algorithm that is suitable for beginners, but some disadvantages are that KNN can be computationally expensive, especially when the dataset is large and the number of features is high. Also, KNN can be biased towards the majority class in imbalanced datasets, especially when the K parameter is small. If we have outliers and K is small, then KNN might choose the outlier’s category which will not be accurate.

Support Vector Machines (SVM) is a powerful supervised machine learning algorithm used for classification and regression analysis. SVM aims to find the hyperplane that best separates the data points into different classes. The hyperplane is selected in such a way that the distance between the hyperplane and the nearest data points (called support vectors) is maximized. This algorithm has the advantage of being able to handle high-dimensional data and perform well in cases where the number of features is greater than the number of samples. Additionally, SVM has a regularization parameter that helps to prevent overfitting. However, the performance of SVM is dependent on the choice of kernel function, and finding the optimal kernel function can be challenging. Some common kernel functions used in SVM include linear, polynomial, and radial basis function (RBF).

Linear Kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} x_{ik} x_{jk}$$ - linear kernel.

$$f(\mathbf{x}) = \beta_0 + \sum_{i \in S} \alpha_i K(\mathbf{x}, \mathbf{x}_i) = \beta_0 + \sum_{i \in S} \alpha_i \Big(\sum_{k=1}^{p} x_k x_{ik}\Big) =$$

$$= \beta_0 + \sum_{k=1}^{p} \Big(\sum_{i \in S} \alpha_i x_{ik}\Big) x_k \implies$$ **linear** function in features $x_1, \ldots, x_p$

Polynomial Kernel:

$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^d = (1 + \sum_{k=1}^{p} x_{ik} x_{jk})^d$ - polynomial kernel of degree d (d - positive integer). It is a non-linear kernel.

Ex. d=2

$$f(\mathbf{x}) = \beta_0 + \sum_{i \in S} \alpha_i K(\mathbf{x}, \mathbf{x}_i) = \beta_0 + \sum_{i \in S} \alpha_i \Big(1 + \sum_{k=1}^{p} x_k x_{ik}\Big)^2 =$$

$$\beta_0 + \sum_{i \in S} \alpha_i \Big(1 + 2\sum_{k=1}^{p} x_{ik} x_k + \Big(\sum_{k=1}^{p} x_{ik} x_k\Big)^2\Big)$$

Radial Kernel:

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

Intuition:
If obs. $\mathbf{x}^* = (x_1^*, \ldots, x_p^*)^T$ is far from obs. $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T \implies$
Euclidean distance $\sum_{k=1}^{p} (x_k^* - x_{ik})^2$ is large $\implies$
$K(\mathbf{x}^*, \mathbf{x}_i) = exp(-\gamma \sum_{k=1}^{p} (x_{ik}^* - x_{ik})^2)$ is very tiny.

This goes on to show that

- Radial kernel is a similarity measure between $\mathbf{x}^*$ & $\mathbf{x}_i$.
- A "far away" support vector $\mathbf{x}_i$ plays virtually no role in $f(\mathbf{x}^*)$

$$f(\mathbf{x}^*) = \beta_0 + \sum_{i \in S} \alpha_i K(\mathbf{x}^*, \mathbf{x}_i)$$

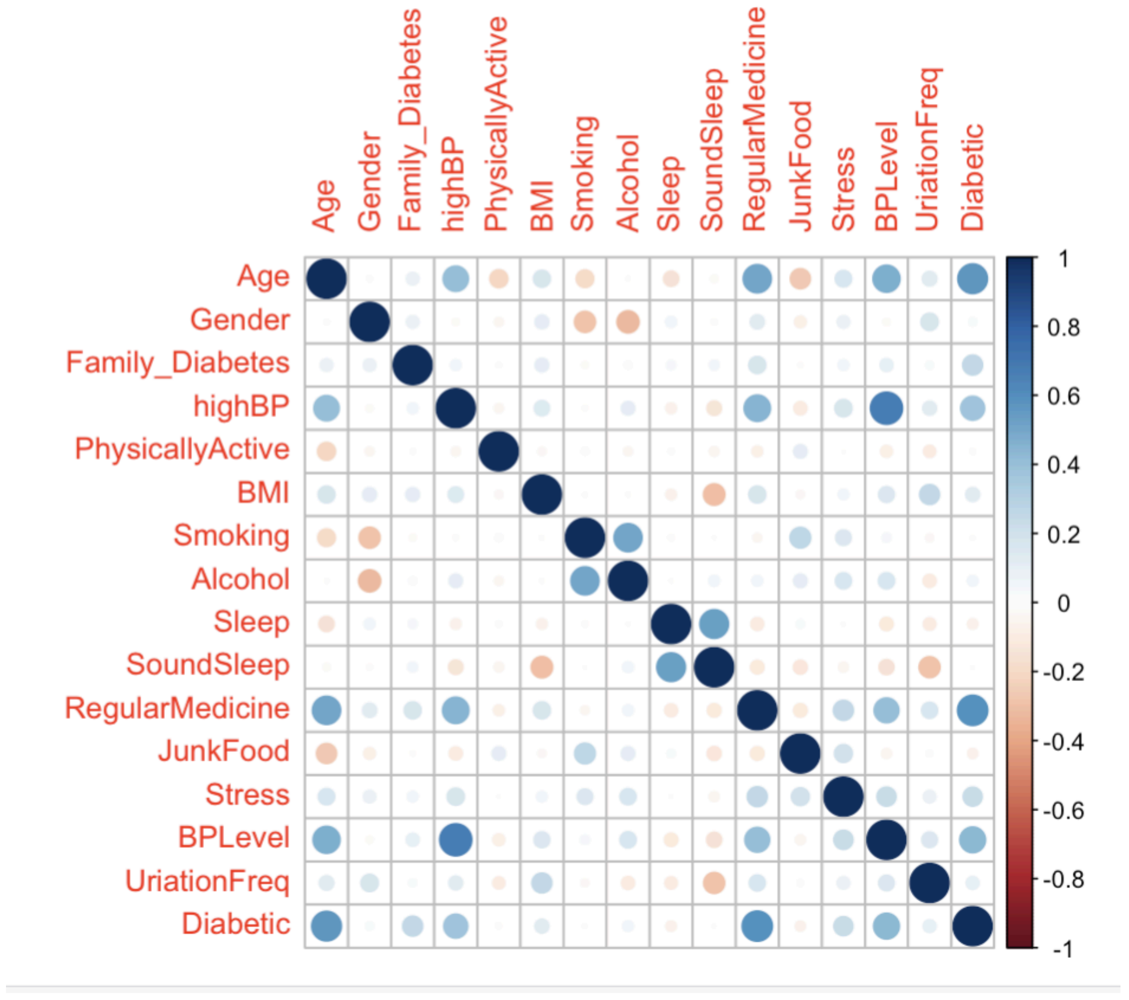
due to tiny value of $K(\mathbf{x}^*, \mathbf{x}_i)$.

In our analysis, we needed to rely on a response variable to answer our main research question, which is whether someone is diabetic or not. To do this, we used K-fold cross validation for both our KNN and SVM models. We opted for K-fold cross validation over Leave-One-Out Cross Validation (LOOCV) due to computational reasons. With the large amount of variables we have in our dataset, lumping some of the samples into subsets would help generalize some of the data that would not have too much effect on variability. Additionally, K-fold CV provides a good estimate of how well our classifiers would perform on new, unseen data. One of the main advantages of K-fold cross validation is that it is not losing any sort of quality of estimation compared to LOOCV, which is important when considering the specific estimation required for our research question. Furthermore, by using K-fold cross validation, we were able to obtain a more accurate estimation of the model's performance, which is critical for our research. However, we did encounter a disadvantage when using KNN, as it can be computationally expensive with large datasets, such as the one we were working with. Despite this, we believed that the benefits of using K-fold cross validation outweighed the potential drawbacks, as it allowed us to obtain more accurate and reliable results in answering our research question. Overall, using K-fold cross validation for our KNN and SVM models provided a robust approach for our analysis, allowing us to obtain reliable estimates of the model's performance and make informed decisions based on the results.

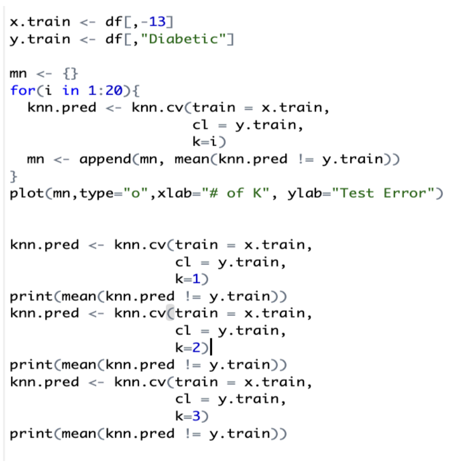
Data Analysis:

KNN: (Jaskaran Singh and Brett Mclain)

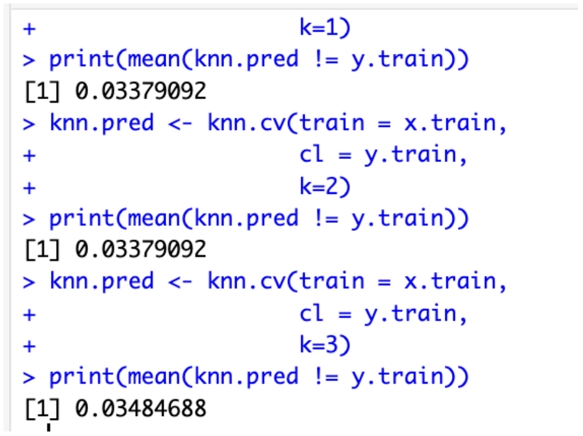
When creating our KNN model, we converted most of our categorical data down to numeric values as all of them were either binary or possessed some order to them. For example, JunkFood had different values like "occasionally" which we assigned to 2 which represents 2 times a week, ”often” =3, "very often"=4, "always"=7 days a week.For the Age column, we assigned a numerical value to each category, which was the average of the bounds. For example, for “under 40” we gave the value (40-0)/2 = 20. For the rest of the variables, please see the comments in the code file attached. Now as all variables are numeric, we are capable of creating a correlation heat map(see below) to show if any of our data is possibly related. From here we are ready to remove certain features. Firstly, we removed pregnancies and Pdiabetes(which is an indicator of Gestation Diabetes caused by pregnancies) because most of the people in the study are males(580 Males vs 372 females) and pregnancies are not relevant to males. Next we inspected the correlation heat map below to find any factors that need to be removed. We removed “highBP” first as it is highly correlated to “BPLevel” as well as a few others. After that we removed “Smoking” as it was related to “Alcohol” and removed “SoundSleep” as it was related to “Sleep.” Removing unnecessary features will help us reduce overfitting as adding too many attributes to a model can cause it to overfit the training data, meaning it fits the training data too well and may not generalize well to new, unseen data. By reducing the number of attributes, the model becomes simpler and less prone to overfitting. It also reduces the risk of including irrelevant or noisy data. Since KNN works on the basis of distances between data points, it is generally recommended to scale all variables to ensure that each variable has an equal influence on the distance metric used by the algorithm. This includes ordinal variables as well. With our predictors chosen, we scaled the data using standard scaling. (also known as Z-score normalization) instead of normalization (min-max scaling) as it centers the data around zero and scales it to have unit variance. This is important because KNN is a distance-based algorithm and standardization ensures that all variables are given equal importance when calculating distances. Min-max scaling, on the other hand, can lead to biased distance calculations if there are outliers or the distribution of the data is non-uniform.



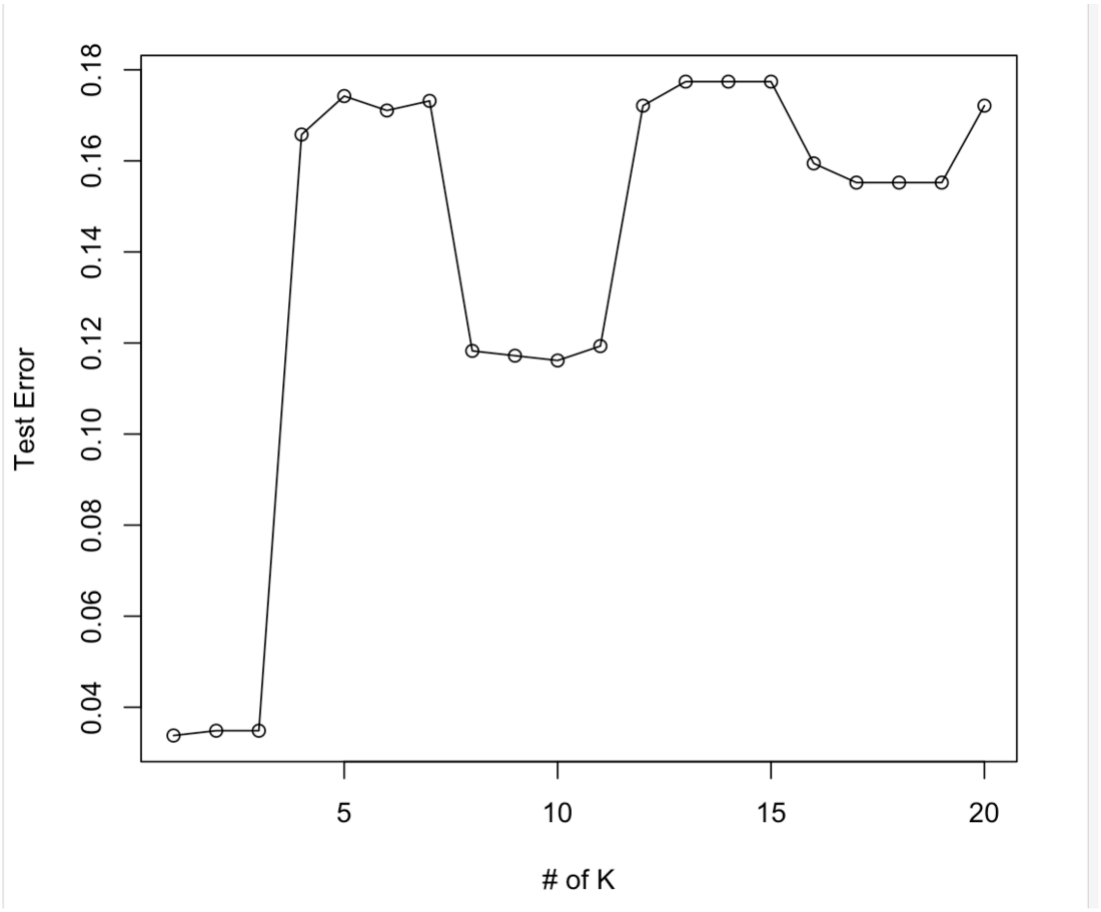
For KNN we tried using k values of 1 through 20 and used K-fold CV to produce the test errors estimates. Doing this, we saw that the test errors spike up quickly after K = 3 without recovering much. The optimal K value was 1 with the best test error of 0.03379 or approximately 3.379%.



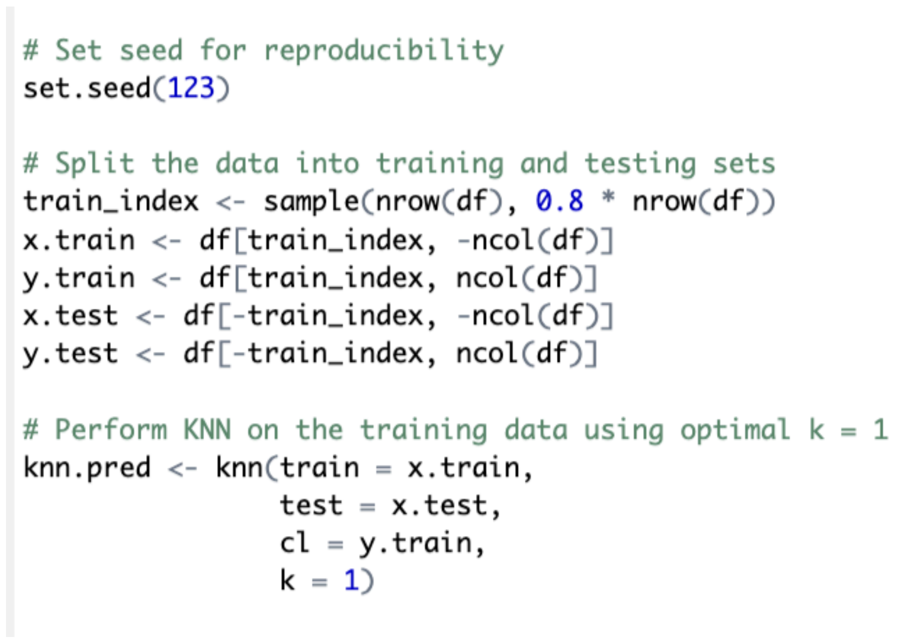
```
x.train <- df[,-13]
y.train <- df[,"Diabetic"]

mn <- {}
for(i in 1:20){
  knn.pred <- knn.cv(train = x.train,
                     cl = y.train,
                     k=i)
  mn <- append(mn, mean(knn.pred != y.train))
}
plot(mn,type="o",xlab="# of K", ylab="Test Error")


knn.pred <- knn.cv(train = x.train,
                   cl = y.train,
                   k=1)
print(mean(knn.pred != y.train))
knn.pred <- knn.cv(train = x.train,
                   cl = y.train,
                   k=2)
print(mean(knn.pred != y.train))
knn.pred <- knn.cv(train = x.train,
                   cl = y.train,
                   k=3)
print(mean(knn.pred != y.train))
```



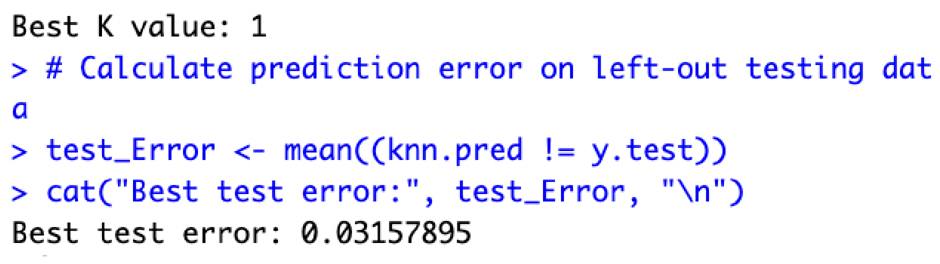
```
+                    k=1)
> print(mean(knn.pred != y.train))
[1] 0.03379092
> knn.pred <- knn.cv(train = x.train,
+                    cl = y.train,
+                    k=2)
> print(mean(knn.pred != y.train))
[1] 0.03379092
> knn.pred <- knn.cv(train = x.train,
+                    cl = y.train,
+                    k=3)
> print(mean(knn.pred != y.train))
[1] 0.03484688
```



Next, we randomly subdivided our data set in 80% for training, and 20% for testing and applied KNN with the optimal value of K =1 and recorded its prediction error on the 20% testing data.

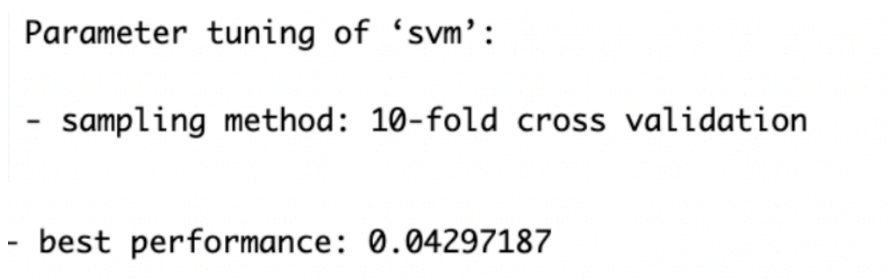


```
# Set seed for reproducibility
set.seed(123)

# Split the data into training and testing sets
train_index <- sample(nrow(df), 0.8 * nrow(df))
x.train <- df[train_index, -ncol(df)]
y.train <- df[train_index, ncol(df)]
x.test <- df[-train_index, -ncol(df)]
y.test <- df[-train_index, ncol(df)]

# Perform KNN on the training data using optimal k = 1
knn.pred <- knn(train = x.train,
                test = x.test,
                cl = y.train,
                k = 1)
```



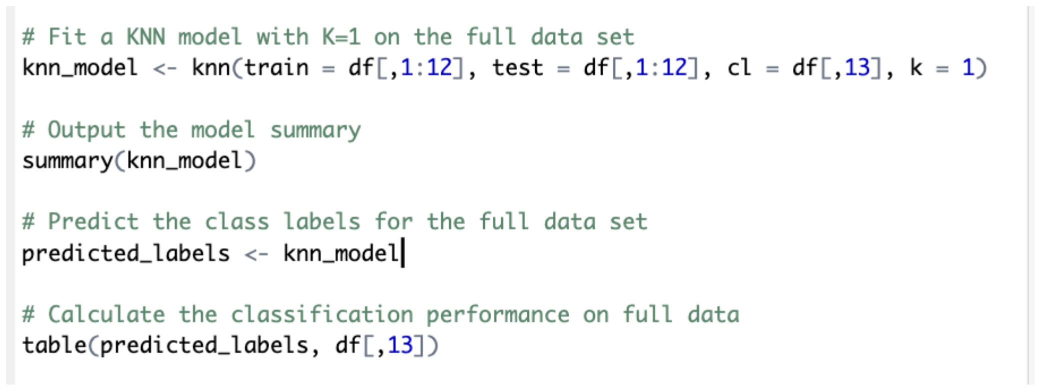
```
Best K value: 1
> # Calculate prediction error on left-out testing dat
a
> test_Error <- mean((knn.pred != y.test))
> cat("Best test error:", test_Error, "\n")
Best test error: 0.03157895
```

We utilized the K-Nearest Neighbors (KNN) algorithm to explore the relationship between the predictors and the diagnosis of diabetes. The main goal of using KNN was to investigate how well the features predict diabetes in a new patient. We used KNN to classify new observations as either diabetic or non-diabetic based on the most similar observation to the new data point in our training data. We analyzed the predictors and determined which ones had the strongest correlation with each other, which allowed us to identify which predictors were redundant and which ones were essential. This information was crucial in improving the accuracy of our model. By identifying which predictors were not necessary, we were able to reduce the complexity of the model and improve its interpretability. We were also able to optimize the model and ensure that it had the necessary variables to make accurate diagnoses. Overall, using KNN to identify the relationship between the predictors and the diagnosis of diabetes allowed us to gain a better understanding of the variables that were important for accurate diagnosis.

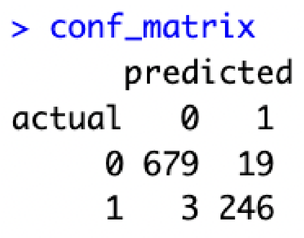


```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best performance: 0.04297187
```

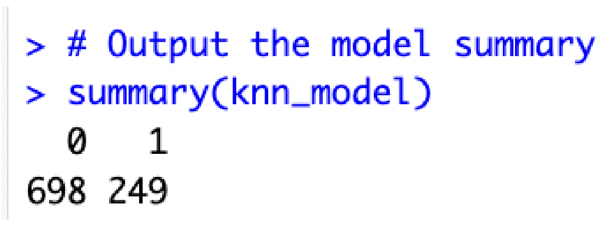
So KNN performed the best (0.034 < 0.043)
Fitting our best model on the full dataset:



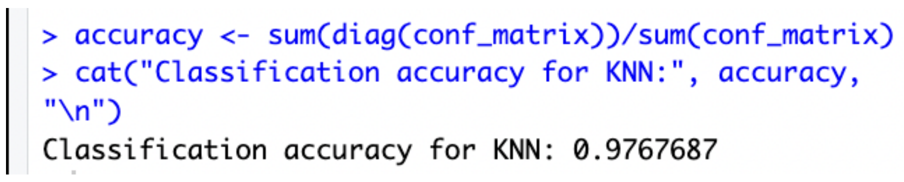
```
# Fit a KNN model with K=1 on the full data set
knn_model <- knn(train = df[,1:12], test = df[,1:12], cl = df[,13], k = 1)

# Output the model summary
summary(knn_model)

# Predict the class labels for the full data set
predicted_labels <- knn_model

# Calculate the classification performance on full data
table(predicted_labels, df[,13])
```



```
> conf_matrix
        predicted
actual    0   1
      0 679  19
      1   3 246
```

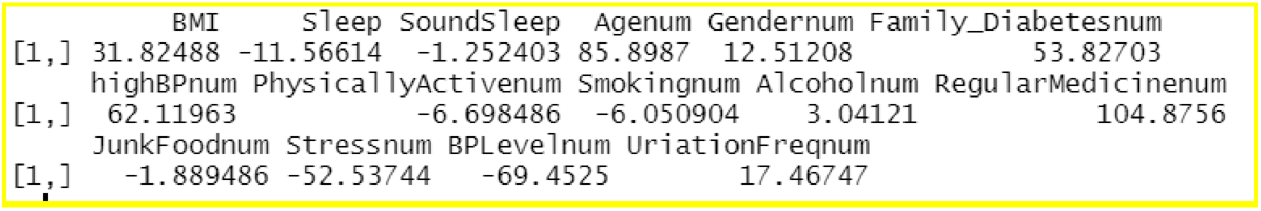


```
> # Output the model summary
> summary(knn_model)
  0   1
698 249
```



```
> accuracy <- sum(diag(conf_matrix))/sum(conf_matrix)
> cat("Classification accuracy for KNN:", accuracy,
"\n")
Classification accuracy for KNN: 0.9767687
```

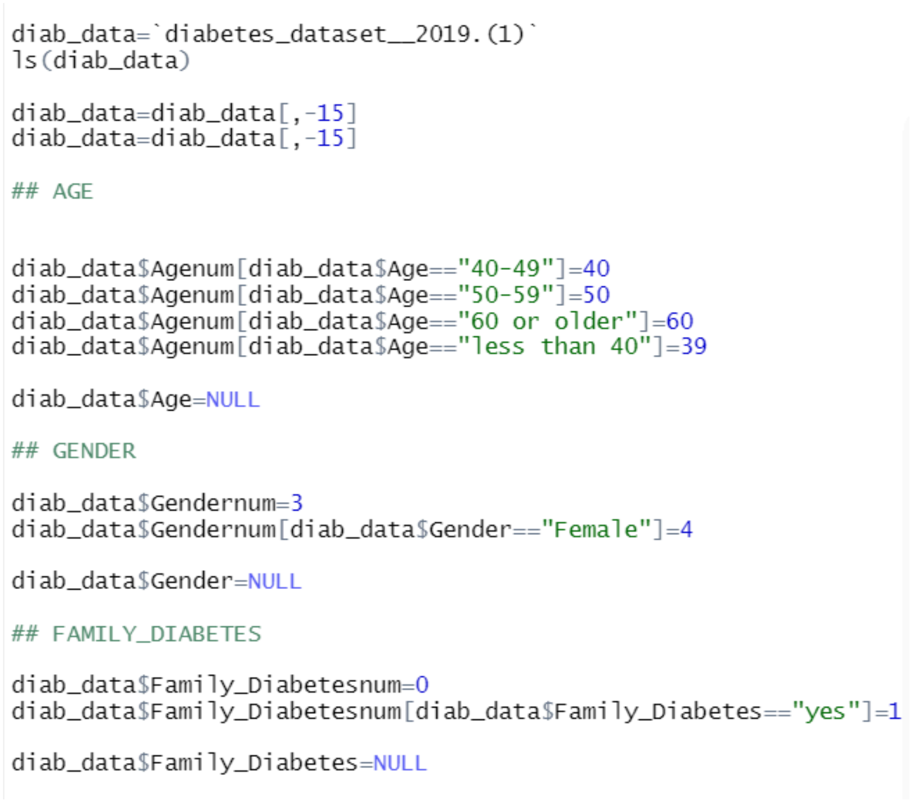
Here are the coefficients of the hyperplane: (Ryder Fiechter, Moataz Altaweel and Kobe Runnels)



```
          BMI     Sleep SoundSleep  Agenum Gendernum Family_Diabetesnum
[1,] 31.82488 -11.56614  -1.252403 85.8987  12.51208           53.82703
     highBPnum PhysicallyActivenum Smokingnum Alcoholnum RegularMedicinenum
[1,]  62.11963           -6.698486  -6.050904    3.04121           104.8756
     JunkFoodnum Stressnum BPLevelnum UriationFreqnum
[1,]   -1.889486 -52.53744   -69.4525        17.46747
```

For Linear SVM:
w^T + b = 0 where w and b are :

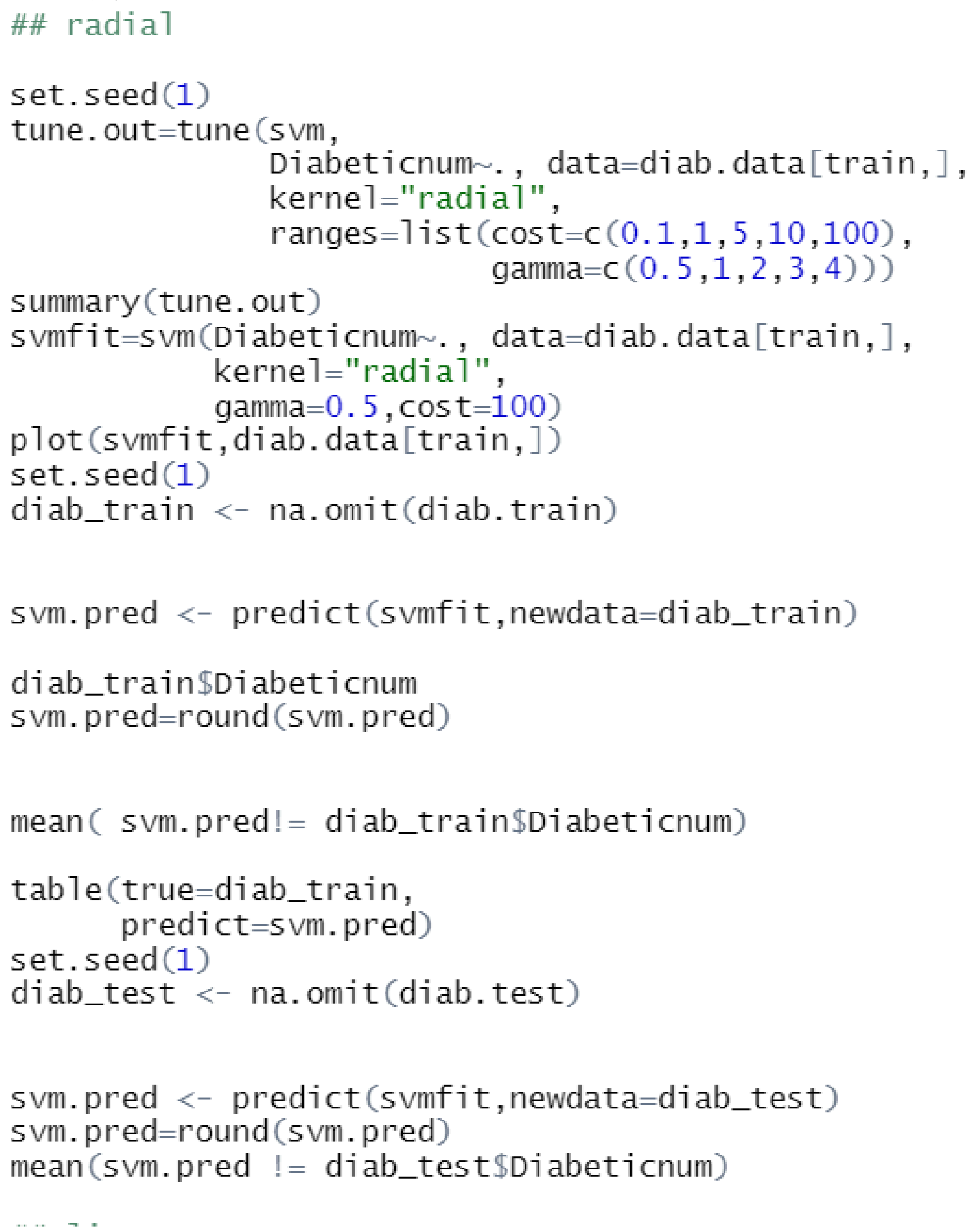
w <- t(svmfit$coefs) %*% svmfit$SV
b <- -svmfit$rho
We chose to use SVM because of its computability for the dataset size while keeping the data that we produce simple and easy to understand by not enlarging the feature space so we can see a definitive line on where people are presumably at risk for becoming diagnosed with Diabetes and people who are presumably not in as much danger of being diagnosed with Diabetes. Furthermore, we used the radial method of SVM because it gave the smallest test error of the three methods, being linear, polynomial, and radial.



```
diab_data=`diabetes_dataset__2019.(1)`
ls(diab_data)

diab_data=diab_data[,-15]
diab_data=diab_data[,-15]

## AGE


diab_data$Agenum[diab_data$Age=="40-49"]=40
diab_data$Agenum[diab_data$Age=="50-59"]=50
diab_data$Agenum[diab_data$Age=="60 or older"]=60
diab_data$Agenum[diab_data$Age=="less than 40"]=39

diab_data$Age=NULL

## GENDER

diab_data$Gendernum=3
diab_data$Gendernum[diab_data$Gender=="Female"]=4

diab_data$Gender=NULL

## FAMILY_DIABETES

diab_data$Family_Diabetesnum=0
diab_data$Family_Diabetesnum[diab_data$Family_Diabetes=="yes"]=1

diab_data$Family_Diabetes=NULL
```

Pregnancy and Pregnant AND have diabetes were omitted from the predictor list due to the fact that we considered them to be not as useful as a predictor as well as being too highly correlated with each other. We then used the tune out function in order to get the optimal set of hyperparameters, and then we ran an SVM function to find the best possible separation of the data, working to find the hyperplane equation.
We did not scale the data because the variables did not have too big of a range and would not provide too much more if any further knowledge for the data we are finding.

SVM: (Ryder Fiechter, Moataz Altaweel and Kobe Runnels)



```
## radial

set.seed(1)
tune.out=tune(svm,
              Diabeticnum~., data=diab.data[train,],
              kernel="radial",
              ranges=list(cost=c(0.1,1,5,10,100),
                          gamma=c(0.5,1,2,3,4)))
summary(tune.out)
svmfit=svm(Diabeticnum~., data=diab.data[train,],
           kernel="radial",
           gamma=0.5,cost=100)
plot(svmfit,diab.data[train,])
set.seed(1)
diab_train <- na.omit(diab.train)


svm.pred <- predict(svmfit,newdata=diab_train)

diab_train$Diabeticnum
svm.pred=round(svm.pred)


mean( svm.pred!= diab_train$Diabeticnum)

table(true=diab_train,
      predict=svm.pred)
set.seed(1)
diab_test <- na.omit(diab.test)


svm.pred <- predict(svmfit,newdata=diab_test)
svm.pred=round(svm.pred)
mean(svm.pred != diab_test$Diabeticnum)
```

Out of the three options: linear, polynomial, and radial, I proceeded to choose the radial SVM model with a cost parameter of 100 and a gamma parameter of 0.5 as it provided me with the smallest test error rate out of the three with a value of 0.02590674.

Conclusion

I believe we achieved a respectable predictive performance as the models are close to 97% accurate which would be really helpful in hospitals as it could determine whether a person's life is going to change.

The SVM model that we developed has an accuracy of 97.9% and is able to accurately predict whether a patient has diabetes or not based on their health characteristics. We chose to use all variables except for Pregnancy and Pdiabetes, as we found that these variables did not have a significant impact on the prediction accuracy.

One of the challenges we faced during this project was generating a plot for the SVM model. Although the code was running without any errors, we were not able to produce an output. Despite this setback, we were still able to obtain accurate predictions from the model using other methods.

Another difficulty we encountered was coming up with the hyperplane equations for the SVM model. The hyperplane is a key component of the SVM model, as it is used to separate the data into different classes.

Overall, we believe that our KNN model can be a useful tool for analyzing and testing patients for diabetes. It has a high accuracy rate and can help healthcare professionals make informed decisions about patient care. However, we acknowledge that there may be other models out there that could perform even better, and we would welcome the opportunity to compare our model with others to further improve our analysis. We tried one hot encoding on Age as we thought it might improve the data analysis and performance of the model, but since there wasn't significant improvement we decided to stick with the regular Age column which kept the number of the features reasonable in amount.

Based on what we've seen with the SVM feature coefficient measurements, there's an evident correlation between the age, regular medicine usage, the high blood pressure, the blood pressure levels, stress, and a history of family diabetes are strong correlated factors that play into potentially what separates diabetics from non-diabetics. These classifiers being shown to have a strong correlation is evident with how said parameters determine the slope of the line with relation to the classes defined, thus answering what factors could potentially lead to someone having diabetes.

In conclusion, we are proud of the work we have done in developing this KNN model for diabetes prediction. Despite some challenges along the way, we were able to produce a model with high accuracy and believe it has the potential to make a positive impact in the healthcare field.

References:

Dataset:
https://www.kaggle.com/datasets/tigganeha4/diabetes-dataset-2019?select=diabetes_dataset__2019.csv

Dr. Wang's Lectures

Appendix:

Our Dataset

	Parameters	Instances
	Total participants	952
1	Age	18 or above
2	Gender: • Male • Female	 580 372
3	Family history with diabetes	Yes/ No
4	Diagnosed with high blood pressure	Yes/ No
5	Walk/run/physically active	• None • Less than half an hour • More than half an hour • One hour or more
6	BMI	Numeric
7	Smoking	Yes/No
8	Alcohol consumption	Yes/No
9	Hours of sleep	Numeric
10	Hours of sound sleep	Numeric
11	Regular intake of medicine?	Yes/No
12	Junk food consumption	Yes/No
13	Stress	• Not at all • Sometimes • Often • Always
14	Blood pressure level	High/normal/low
15	Number of pregnancies	Numeric
16	Gestation diabetes	Yes/No
17	Frequency of urination	• Not much • Quite much
18	Diabetic?	• Diabetic - 267 • Non diabetic - 685