# Homework 6

Prepare your answers as a **single PDF file**.
**Group work**: You may work in groups of 1-3. Include all group member names in the PDF file. Only **one person** in the group should submit to Canvas.
**Due**: check on Canvas.

**1.** Consider the toy dataset below, which shows if 4 subjects have diabetes or not, along with two diagnostic measurements. (Note: do **NOT** write any code for this problem. The answers are to be computed by hand.)

| **Preg** | **Glucose** | **HasDiabetes** | **Preg.Norm** | **Glucose.Norm** |
|---|---|---|---|---|
| 2 | 157 | No | (2 - 1) / (3 - 1) = 0.5 | (157 - 77) / (174 - 77) = 0.811 |
| 3 | 174 | Yes | (3 - 1) / (3 - 1) = 1.0 | (174 - 77) / (174 - 77) = 1.0 |
| 2 | 105 | Yes | (2 - 1) / (3 - 1) = 0.5 | (105 - 77) / (174 - 77) = 0.614 |
| 1 | 77 | No | (1 - 1) / (3 - 1) = 0.0 | (77 - 77) / (174 - 77) = 0.0 |
| 2 | 94 | ? | (2 - 1) / (3 - 1) = 0.5 | (94 - 77) / (174 - 77) = 0.425 |

a. Which variable is the “Class” variable? **HasDiabetes**
b. Normalize the Preg and Glucose values by scaling the minimum-maximum range of each column to 0-1. Fill in the empty columns in the table.

Preg.Norm = (Preg - min(Preg)) / (max(Preg) - min(Preg))
Glucose.Norm = (Glucose - min(Glucose)) / (max(Glucose) - min(Glucose))

c. Predict whether a subject with Preg=2, Glucose=94 will have diabetes using the 1-NN algorithm and
    i. Using Euclidean distance on the original variables

Distance to Row 1: sqrt((2 - 2)^2 + (94 - 157)^2) = sqrt(4225) = 65
Distance to Row 2: sqrt((2 - 3)^2 + (94 - 174)^2) = sqrt(18225) = 135
Distance to Row 3: sqrt((2 - 2)^2 + (94 - 105)^2) = sqrt(121) = 11
Distance to Row 4: sqrt((2 - 1)^2 + (94 - 77)^2) = sqrt(4225) = 65
The nearest neighbor is Row 3, with a distance of 11, and it has diabetes. Therefore, we predict that the subject with Preg=2 and Glucose=94 will have diabetes.

ii. Using Euclidean distance on the normalized variables

Distance to Row 1: sqrt((0.5 - 0.5)^2 + (0.425 - 1.0)^2) = sqrt(0.3025) = 0.55
Distance to Row 2: sqrt((0.5 - 1.0)^2 + (0.425 - 0.8)^2) = sqrt(0.3425) = 0.585
Distance to Row 3: sqrt((0.5 - 0.5)^2 + (0.425 - 0.614)^2) = sqrt(0.0356) = 0.189
Distance to Row 4: sqrt((0.5 - 0.0)^2 + (0.425 - 0.0)^2) = sqrt(0.325) = 0.57
The nearest neighbor is Row 3 with a distance of 0.189, and it has diabetes. Therefore, we predict that the subject with Preg.Norm = 0.5 and Glucose.Norm = 0.425 will have diabetes.

For each of these cases, give the nearest distance, nearest neighbor (e.g., “Row 1” or “Row 2”), and prediction.

**2.** The `pima-indians-diabetes-resampled.csv` file on Canvas contains records indicating whether the subjects have diabetes or not, along with certain diagnostic measurements. All subjects are of Pima Indian heritage and this dataset is called the Pima Indian Diabetes Database[1]. The goal is to see if it is possible to predict if a subject has diabetes given some of the diagnostic measurements. (**Note: this problem is an extension of the classwork assignment; R code from the class is also posted on Canvas.**)

a. Read the data file [code]

```
> survey <- read_csv("Downloads/pima-indians-diabetes-resampled.csv")
```

b. What does “Preg” represent in the dataset? (2-3 sentences. Search for the Pima Indian Diabetes Database online. Its background and the ethics issues it raises are also important.

In the dataset, "Preg" represents the number of times a subject has been pregnant. The Pima Indian Diabetes Database is a dataset used for studying the relationship between various health metrics, including the number of pregnancies, and the likelihood of diabetes in Pima Indian women. This database has raised ethical concerns due to its use in research involving a vulnerable population and the associated privacy and consent issues.

c. 0 values in the Glucose column indicate missing values. Remove rows which contain missing values in the Glucose column. You should have 763 rows. [code]

```
> diabetes_data <- survey[survey$Glucose != 0, ]
```

d. Create **three new columns/variables** which are the **normalized** versions of Preg, Pedigree, and Glucose columns, scaling the minimum-maximum range of each column to 0-1 (you can use the code developed in class). [code]

[1] https://github.com/jbrownlee/Datasets/blob/master/pima-indians-diabetes.names

```
> normalize <- function(x) {return ( (x-min(x))/(max(x)-min(x))  )}
> diabetes_data <- diabetes_data %>% mutate(Preg_Norm = normalize(Preg),
Pedigree_Norm = normalize(Pedigree), Glucose_Norm = normalize(Glucose))
```

e. Split the dataset into train and test datasets with the *first 500 rows* for training, and the remaining rows for test. Do NOT randomly sample the data (though resampling is usually done, this hw problem does not use this step for ease of grading).

```
> trainindex <- 1:500
> testindex <- -trainindex
```

f. Train and test a k-nearest neighbor classifier with the dataset. *Consider only the normalized Preg and Pedigree columns*. Set k=1. What is the error rate (number of misclassifications)? [code, error rate]

```
> trainfeatures <- diabetes_data[trainindex, c(10, 11)]
> traininglabels <- diabetes_data[trainindex, 9]

> testfeatures <- diabetes_data[testindex, c(10, 11)]
> testlabels <- diabetes_data[testindex, 9]

> predicted <- knn(train = trainfeatures, cl= traininglabels$HasDiabetes, test =
testfeatures)
> data.frame(testlabels, predicted) %>% View()
> table(testlabels$HasDiabetes, predicted)

predicted
     0   1
 0 120  50
 1  57  36
```

Errors: 57 + 50 = 107

Percentage: 107/263 = 0.407

g. Repeat part (f) but *consider the normalized Preg, Pedigree, and Glucose columns*. Set k=1. What is the error rate? Will the error rate always decrease with a larger number of features? Why or why not: answer in 2-3 sentences? [code, error rate, answer]

```
> trainfeatures <- diabetes_data[trainindex, c(10, 11, 12)]
> traininglabels <- diabetes_data[trainindex, 9]

> testfeatures <- diabetes_data[testindex, c(10, 11, 12)]
> testlabels <- diabetes_data[testindex, 9]
> predicted <- knn(train = trainfeatures, cl= traininglabels$HasDiabetes, test =
testfeatures)
> data.frame(testlabels, predicted) %>% View()
> table(testlabels$HasDiabetes, predicted)
   Predicted
      0   1
  0 128  42
  1  42  51
```

Errors: 42 + 42 = 84

```
> error_rate <- sum(predicted != testlabels$HasDiabetes) / nrow(testlabels)
> error_rate
[1] 0.3193916
```

Adding more features may lead to an increased risk of overfitting, where the model becomes too specific to the training data and performs poorly on new, unseen data. The relationship between the number of features and the error rate depends on the quality of the features, the amount of data available, and the complexity of the model. In some cases, adding relevant features can improve model performance, but in others, it can lead to overfitting, causing the error rate to increase. Therefore, it's essential to carefully consider the relevance and quality of features when building a machine-learning model.

h. Repeat part (g) but set k=5. What is the error rate? [code, error rate]

```
> predicted <- knn(train = trainfeatures, cl= traininglabels$HasDiabetes, test =
testfeatures, k = 5)
> data.frame(testlabels, predicted) %>% View()
> table(testlabels$HasDiabetes, predicted)
   predicted
      0   1
  0 149  21
  1  42  51
> error_rate <- sum(predicted != testlabels$HasDiabetes) / nrow(testlabels)
> error_rate
[1] 0.2395437
```

i. Repeat part (h) but set k=11. What is the error rate? [code, error rate]

```
> predicted <- knn(train = trainfeatures, cl= traininglabels$HasDiabetes, test =
testfeatures, k = 11)
> data.frame(testlabels, predicted) %>% View()
> table(testlabels$HasDiabetes, predicted)
   predicted
```

```
     0   1
  0 154  16
  1  42  51
> error_rate <- sum(predicted != testlabels$HasDiabetes) / nrow(testlabels)
> error_rate
[1] 0.2205323
```

j. Considering your observations from (g)-(i), which is the best value for k? [answer]

Considering the error rates observed for different values of k in parts (g), (h), and (i), the best value for k is 11. At k = 11 the error rate is the lowest among the values tested. A smaller value of k (k=5) resulted in a slightly higher error rate, and the smallest value of k (k=1) had the highest error rate. Therefore, based on the data and observations, k=11 is the best value for the k-nearest neighbor classifier in terms of minimizing the error rate.