BIOSTAT 216 Final Project

Vishakha Malhotra

0.1 Introduction:

Throughout the last several decades, the use of computed tomography (CT) and X-ray imaging-based technologies to detect, diagnose, and treat various dental illnesses has increased exponentially due to the need for precise imaging and diagnostic tools. However, manual X-ray film interpretation and analysis may take large amounts of crucial clinical hours. They may lead to misdiagnosis or underdiagnosis, owing to personal factors such as stress, fatigue, and experience levels. If intelligent dental X-ray film interpretation tools are created to aid dentists in enhancing dental treatment, these flaws can be reduced. This viewpoint elevates dental informatics, such as automatic teeth identification, dental abnormality identification, and annotation, to a crucial component of competent health care.[1]

One way of ensuring that an oral cavity is diseased or healthy would be to utilize machine-learning techniques for object detection. For segmenting teeth, traditional Machine Learning (ML) methods including mathematical morphology [2], active contour [3], and level set [4] have been applied. In addition, hand-crafted features extracted using Fourier descriptors [5], contours, and textures [4] have been combined with Bayesian approaches [5], linear models [6], and support vector machines to achieve classification. Unfortunately, most of these techniques require rigorous feature extractor engineering to convert the raw data into an acceptable representation for the algorithms to recognize or categorize the input photos.

These X-rays can be utilized to create a binary classifier that accurately determines whether an X-ray is of a diseased mouth. A machine learning algorithm is trained to recognize patterns and features in the X-rays that are linked with diseased teeth by using a large dataset of dental X-rays. A binary classifier that can categorize new X-rays as either unhealthy or healthy can be created using these patterns. This strategy could greatly increase the precision and effectiveness of dental diagnostics and support the creation of patient-specific treatment strategies.

The dataset being used is from The Tufts Dental Database and consists of 1000 panoramic dental radiography images with expert labeling of abnormalities and teeth. The classification of radiography images was performed based on five different levels: anatomical location, peripheral characteristics, radiodensity, effects on the surrounding structure, and the abnormality category.

0.2 Methods:

1000 de-identified digital panoramic radiographs were chosen at random from the electronic patient database (AxiUm) between January 2014 and December 2016 for this pilot study by Tufts University School of Dental Medicine. An expert and a student from the school of dental medicine annotated the chosen radiographs, and eye-tracking and audio data were gathered. The collection consists of six main parts: radiographs, annotated masks, maps created by eye trackers, text data, teeth masks, and masks for the maxillomandibular region of interest. To assure diagnostic quality, the radiographs were taken utilizing two pieces of radiography equipment with automatic exposure control. Each radiograph was examined by a professional and a fourth-year dentistry student. Recognizing abnormalities required knowledge of typical variations, which was the initial stage in the analysis.

Using the radiographs provided, image pre-processing was done.

The code seeks to diagnose teeth by extracting dental traits from radiographic images of full-mouth x-rays. Python is used to create the code, together with libraries like OpenCV, NumPy, Pandas, SciPy, and scikit-image.

All necessary libraries and modules are first loaded by the code. The function "build dental features" is then defined, and it accepts an input file, which is a radiological image of a tooth. The image is then read using OpenCV's imread function, converted to grayscale, then thresholding is applied to produce a binary image. Then, morphological techniques are used to eliminate noise and detach touching objects. By locating contours, drawing them, and applying them to crop, the function goes one step further and extracts individual teeth from the image. This step is important because contrast enhancement will make it simpler for our model to recognize minute variations in tissue density and spot anomalies. Similarly, an image's objects and structures can be recognized using edge detection and circularity metrics. Edges can be used to locate anatomical components including bones, blood arteries, and organs in radiography pictures. (Please check image_preprocess_1.py file attached)

The function first obtains individual teeth, after which it extracts a variety of features from each tooth, including features for the Haralick texture, the Gabor filter, the Canny Edge Detection, the Gaussian image, the Sobel image, the contour-based features, and the Hu Moments. These features make it possible to recognize and categorize several kinds of anomalies in the image using metrics like extent and Hu shape. For instance, a tumor's extension can offer important details regarding its size and potential for metastasis, while Hu shape measurements can be used to differentiate between various tumor types based on their shape and texture. Each radiograph is then given a label indicating whether it is diseased. It then saves a CSV file containing all the features that were extracted from each tooth.

For all the radiographs in each directory, the "create features" function invokes the "build dental features" function and saves the extracted features in a different directory.

After image pre-processing is done, we cleaned the .csv file data obtained further in R. Data cleaning was done and a few variables (namely, edges, edge entropy, sobel_img_1, and aspect ratio) were removed as they could not be handled in .csv format. We also removed the variables which only had one unique value (aspect ratio).

The algorithms chosen to train our models were logistic regression and support vector machine because they calculate the likelihood of a binary result based on the input features. Logistic regression is an effective approach for data that can be linearly separated. SVM, on the other hand, is a non-linear method that identifies the best hyperplane that equidistantly divides the two classes. When the data cannot be separated linearly or when the boundaries between the two classes are complex and non-linear, SVM is especially helpful.

```r
dental_data <- read.csv(
  file = "/Users/drvish/Desktop/Final_project_216/tooth_features.csv")
```

```r
#count unique values for each variable
sapply(lapply(dental_data, unique), length)
```

```
     contrast          energy     homogeneity      gabor_mean       gabor_std 
          603             575             605             250             247 
        edges    edge_density      edge_mean.        edge_std    edge_entropy 
           46             995             952             980              75 
        pixel  guassian_img_1     sobel_img_1     circularity    aspect_ratio 
         1000              54              25             615               1 
       extent        hu_shape           label            name 
          409             556               2            1000 
```

```r
# So, aspect_ratio is the variable with one unique value and
# removing name since it is just radiograph number.
# We will also be removing the variables pixel, guassian_img_1, sobel_img_1,
# edges and edge_entropy
dental_data_clean <- dental_data %>%
  select(-c(aspect_ratio, name, pixel,
            guassian_img_1,sobel_img_1,edges, edge_entropy))

# Converting label variable into factors
dental_data_clean$label <- factor(dental_data_clean$label,
                                  levels = c('Healthy', 'Diseased'))
str(dental_data_clean)
```

```
'data.frame':   1000 obs. of  12 variables:
 $ contrast    : num  311.6 29.1 186.4 0 0 ...
 $ energy      : num  0.0217 0.1486 0.029 0 0 ...
 $ homogeneity : num  0.0362 0.2716 0.0771 0 0 ...
 $ gabor_mean  : num  0.00226 0 0.00108 0 0 ...
 $ gabor_std   : num  0.0493 0 0.0328 0 0 ...
 $ edge_density: num  753 21647 1115 211792 307558 ...
 $ edge_mean.  : num  1.82 1.76 1.52 1.41 2.04 ...
 $ edge_std    : num  21.5 21.1 19.7 18.9 22.7 ...
 $ circularity : num  0.441 0.419 0.544 0.785 0.785 ...
 $ extent      : num  0.464 1.1 0.453 1 1 ...
 $ hu_shape    : num  0.532 0.336 0.681 0.222 0.222 ...
 $ label       : Factor w/ 2 levels "Healthy","Diseased": 2 1 1 2 1 2 1 1 1 2 ...
```

0.3 Results:

The histogram displays the relative distribution of the diseased vs healthy.

```r
dental_data_clean_2 = dental_data_clean
dental_data_clean_2$label <- as.numeric(dental_data_clean_2$label)
ggplot(dental_data_clean_2,aes(x = label)) +
  geom_histogram(binwidth = 0.5) +
  xlab('Healthy Mouth v/s Diseased')
```

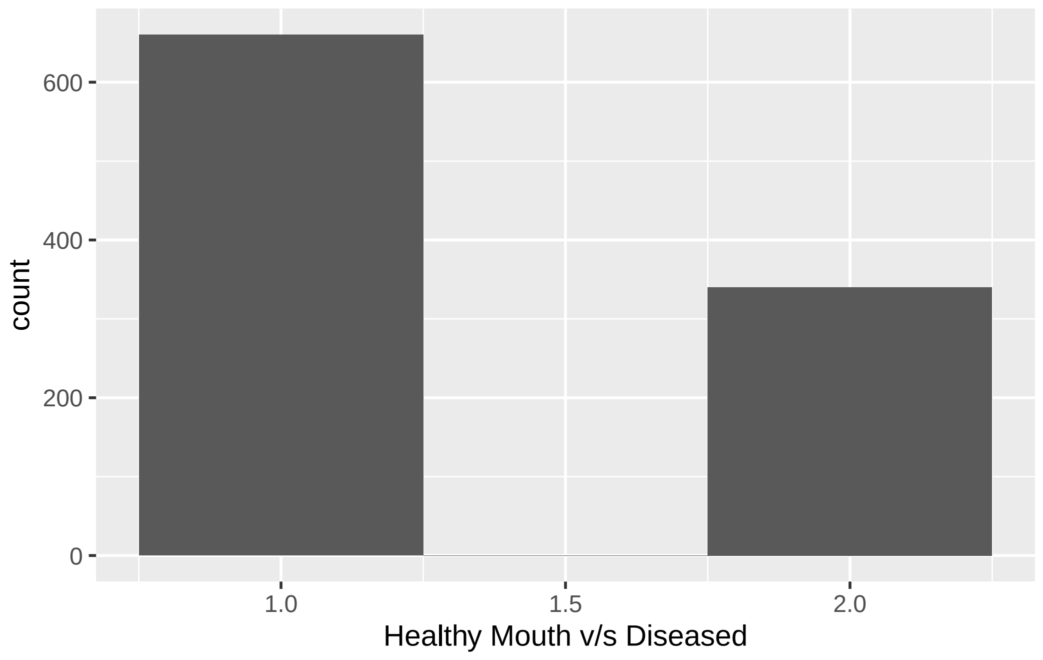
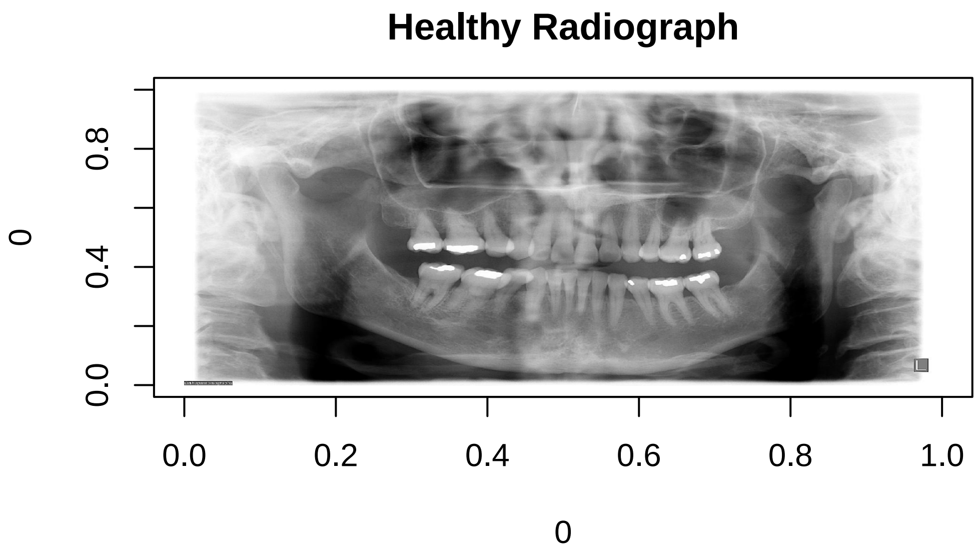
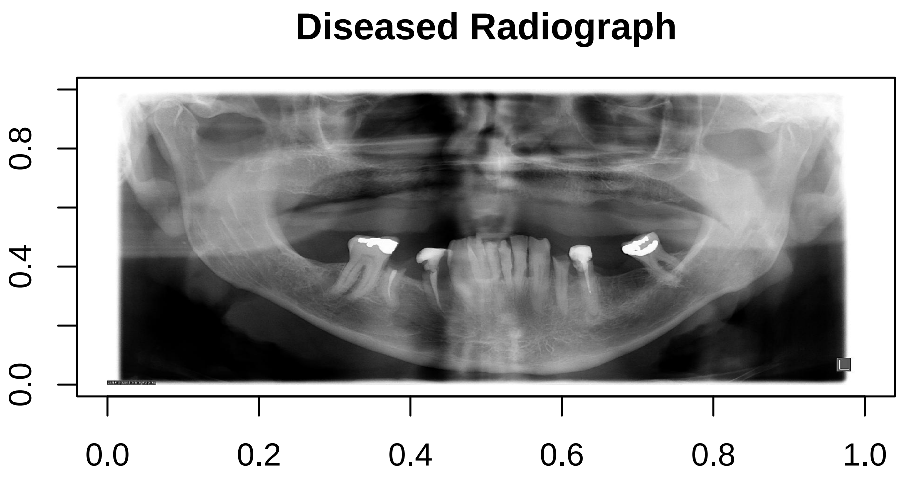


Image of healthy vs diseased radiograph:

```
healthy_radiograph <- readJPEG(
  "/Users/drvish/Desktop/TuftsDentalData/Radiographs/28.JPG")
plot(0, 0, type = "n", xlim = c(0, 1), ylim = c(0, 1))
rasterImage(healthy_radiograph, 0, 0, 1, 1)
title(main = "Healthy Radiograph", line = 1)
```



```
diseased_radiograph <- readJPEG(
  "/Users/drvish/Desktop/TuftsDentalData/Radiographs/854.JPG")
plot(0, 0, type = "n", xlim = c(0, 1), ylim = c(0, 1), xlab = "", ylab = "")
rasterImage(diseased_radiograph, 0, 0, 1, 1)
title(main = "Diseased Radiograph", line = 1)
```



After data cleaning is done, we will split the data into 70-30 (70% = 700 obs for training and 30% = 300 obs for testing.)

```r
# Split data into training and testing sets
set.seed(123)
train_index <- createDataPartition(
  dental_data_clean$label, p = 0.7, list = FALSE)
train_data <- dental_data_clean[train_index, ]
test_data <- dental_data_clean[-train_index, ]
```

Given that we are interested in a classification problem for a binary outcome, it was decided to use a logistic regression approach. This code is performing a classification analysis using the glmnet package in R, which uses Lasso or Ridge regression to perform variable selection and regularization. After loading the necessary packages, glmnet and ggplot2, we create a grid of alpha and lambda values. In this case, a grid with alpha values ranging from 0 to 1 in increments of 0.3, and lambda values ranging from 10^-5 to 10^1 in 25 steps was generated.

The trainControl function was set up for a repeated cross-validation scheme, where the data is split into training and testing sets (k = 10 and 3 repeats.) The training set and the predetermined grid of alpha and lambda values are used by the model glmnet function to fit a glmnet model to the data. The grid of hyperparameters to be tuned using cross-validation is specified by the tuneGrid argument. For each combination of alpha and lambda, the model glmnet$results line produces the cross-validation results, including the mean and standard deviation of the accuracy, kappa, and ROC AUC metrics across all cross-validation folds.

```r
set.seed(125)

# lambda and alpha values
grid <- expand.grid(alpha = seq(0, 1, by = 0.3),
                    lambda = 10^seq(-5, 1, length = 25))

train_control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)

# The model with the "glmnet" method and the alpha-lambda grid
model_glmnet <- train(label ~ .,
                      data = train_data,
                      method = "glmnet",
                      trControl = train_control,
                      tuneGrid = grid)

model_glmnet$results
```

```
    alpha       lambda  Accuracy         Kappa  AccuracySD     KappaSD
1     0.0 1.000000e-05 0.6590497 -1.030720e-03 0.006698961 0.013515016
2     0.0 1.778279e-05 0.6590497 -1.030720e-03 0.006698961 0.013515016
3     0.0 3.162278e-05 0.6590497 -1.030720e-03 0.006698961 0.013515016
4     0.0 5.623413e-05 0.6590497 -1.030720e-03 0.006698961 0.013515016
5     0.0 1.000000e-04 0.6590497 -1.030720e-03 0.006698961 0.013515016
6     0.0 1.778279e-04 0.6590497 -1.030720e-03 0.006698961 0.013515016
7     0.0 3.162278e-04 0.6590497 -1.030720e-03 0.006698961 0.013515016
8     0.0 5.623413e-04 0.6590497 -1.030720e-03 0.006698961 0.013515016
9     0.0 1.000000e-03 0.6590497 -1.030720e-03 0.006698961 0.013515016
10    0.0 1.778279e-03 0.6590497 -1.030720e-03 0.006698961 0.013515016
11    0.0 3.162278e-03 0.6590497 -1.030720e-03 0.006698961 0.013515016
12    0.0 5.623413e-03 0.6595259 -9.064966e-05 0.005977088 0.012502800
13    0.0 1.000000e-02 0.6600020  8.494208e-04 0.005109397 0.011320597
14    0.0 1.778279e-02 0.6604851  1.801802e-03 0.005028225 0.009868875
15    0.0 3.162278e-02 0.6604851  1.801802e-03 0.005028225 0.009868875
16    0.0 5.623413e-02 0.6604851  1.801802e-03 0.005028225 0.009868875
17    0.0 1.000000e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
18    0.0 1.778279e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
19    0.0 3.162278e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
20    0.0 5.623413e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
```

```
21    0.0 1.000000e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
22    0.0 1.778279e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
23    0.0 3.162278e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
24    0.0 5.623413e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
25    0.0 1.000000e+01 0.6600089  0.000000e+00 0.004615632 0.000000000
26    0.3 1.000000e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
27    0.3 1.778279e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
28    0.3 3.162278e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
29    0.3 5.623413e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
30    0.3 1.000000e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
31    0.3 1.778279e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
32    0.3 3.162278e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
33    0.3 5.623413e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
34    0.3 1.000000e-03 0.6599953  2.585403e-03 0.007667367 0.019487205
35    0.3 1.778279e-03 0.6599953  2.585403e-03 0.007667367 0.019487205
36    0.3 3.162278e-03 0.6604715  3.525474e-03 0.006978991 0.018611347
37    0.3 5.623413e-03 0.6600020  8.494208e-04 0.005109397 0.011320597
38    0.3 1.000000e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
39    0.3 1.778279e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
40    0.3 3.162278e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
41    0.3 5.623413e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
42    0.3 1.000000e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
43    0.3 1.778279e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
44    0.3 3.162278e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
45    0.3 5.623413e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
46    0.3 1.000000e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
47    0.3 1.778279e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
48    0.3 3.162278e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
49    0.3 5.623413e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
50    0.3 1.000000e+01 0.6600089  0.000000e+00 0.004615632 0.000000000
51    0.6 1.000000e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
52    0.6 1.778279e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
53    0.6 3.162278e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
54    0.6 5.623413e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
55    0.6 1.000000e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
56    0.6 1.778279e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
57    0.6 3.162278e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
58    0.6 5.623413e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
59    0.6 1.000000e-03 0.6599953  2.585403e-03 0.007667367 0.019487205
60    0.6 1.778279e-03 0.6604715  3.525474e-03 0.006978991 0.018611347
61    0.6 3.162278e-03 0.6600020  8.494208e-04 0.005109397 0.011320597
62    0.6 5.623413e-03 0.6600089  0.000000e+00 0.004615632 0.000000000
63    0.6 1.000000e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
```

```
64    0.6 1.778279e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
65    0.6 3.162278e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
66    0.6 5.623413e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
67    0.6 1.000000e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
68    0.6 1.778279e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
69    0.6 3.162278e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
70    0.6 5.623413e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
71    0.6 1.000000e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
72    0.6 1.778279e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
73    0.6 3.162278e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
74    0.6 5.623413e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
75    0.6 1.000000e+01 0.6600089  0.000000e+00 0.004615632 0.000000000
76    0.9 1.000000e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
77    0.9 1.778279e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
78    0.9 3.162278e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
79    0.9 5.623413e-05 0.6599953  2.585403e-03 0.007667367 0.019487205
80    0.9 1.000000e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
81    0.9 1.778279e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
82    0.9 3.162278e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
83    0.9 5.623413e-04 0.6599953  2.585403e-03 0.007667367 0.019487205
84    0.9 1.000000e-03 0.6599953  2.585403e-03 0.007667367 0.019487205
85    0.9 1.778279e-03 0.6600020  8.494208e-04 0.005109397 0.011320597
86    0.9 3.162278e-03 0.6600089  0.000000e+00 0.004615632 0.000000000
87    0.9 5.623413e-03 0.6600089  0.000000e+00 0.004615632 0.000000000
88    0.9 1.000000e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
89    0.9 1.778279e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
90    0.9 3.162278e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
91    0.9 5.623413e-02 0.6600089  0.000000e+00 0.004615632 0.000000000
92    0.9 1.000000e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
93    0.9 1.778279e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
94    0.9 3.162278e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
95    0.9 5.623413e-01 0.6600089  0.000000e+00 0.004615632 0.000000000
96    0.9 1.000000e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
97    0.9 1.778279e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
98    0.9 3.162278e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
99    0.9 5.623413e+00 0.6600089  0.000000e+00 0.004615632 0.000000000
100   0.9 1.000000e+01 0.6600089  0.000000e+00 0.004615632 0.000000000
```

```
# the best model
print(model_glmnet)
```

```
glmnet
```

```
700 samples
 11 predictor
  2 classes: 'Healthy', 'Diseased'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 630, 631, 630, 630, 630, 630, ...
Resampling results across tuning parameters:

  alpha  lambda        Accuracy   Kappa
  0.0    1.000000e-05  0.6590497  -1.030720e-03
  0.0    1.778279e-05  0.6590497  -1.030720e-03
  0.0    3.162278e-05  0.6590497  -1.030720e-03
  0.0    5.623413e-05  0.6590497  -1.030720e-03
  0.0    1.000000e-04  0.6590497  -1.030720e-03
  0.0    1.778279e-04  0.6590497  -1.030720e-03
  0.0    3.162278e-04  0.6590497  -1.030720e-03
  0.0    5.623413e-04  0.6590497  -1.030720e-03
  0.0    1.000000e-03  0.6590497  -1.030720e-03
  0.0    1.778279e-03  0.6590497  -1.030720e-03
  0.0    3.162278e-03  0.6590497  -1.030720e-03
  0.0    5.623413e-03  0.6595259  -9.064966e-05
  0.0    1.000000e-02  0.6600020   8.494208e-04
  0.0    1.778279e-02  0.6604851   1.801802e-03
  0.0    3.162278e-02  0.6604851   1.801802e-03
  0.0    5.623413e-02  0.6604851   1.801802e-03
  0.0    1.000000e-01  0.6600089   0.000000e+00
  0.0    1.778279e-01  0.6600089   0.000000e+00
  0.0    3.162278e-01  0.6600089   0.000000e+00
  0.0    5.623413e-01  0.6600089   0.000000e+00
  0.0    1.000000e+00  0.6600089   0.000000e+00
  0.0    1.778279e+00  0.6600089   0.000000e+00
  0.0    3.162278e+00  0.6600089   0.000000e+00
  0.0    5.623413e+00  0.6600089   0.000000e+00
  0.0    1.000000e+01  0.6600089   0.000000e+00
  0.3    1.000000e-05  0.6599953   2.585403e-03
  0.3    1.778279e-05  0.6599953   2.585403e-03
  0.3    3.162278e-05  0.6599953   2.585403e-03
  0.3    5.623413e-05  0.6599953   2.585403e-03
  0.3    1.000000e-04  0.6599953   2.585403e-03
  0.3    1.778279e-04  0.6599953   2.585403e-03
  0.3    3.162278e-04  0.6599953   2.585403e-03
```

```
0.3    5.623413e-04  0.6599953   2.585403e-03
0.3    1.000000e-03  0.6599953   2.585403e-03
0.3    1.778279e-03  0.6599953   2.585403e-03
0.3    3.162278e-03  0.6604715   3.525474e-03
0.3    5.623413e-03  0.6600020   8.494208e-04
0.3    1.000000e-02  0.6600089   0.000000e+00
0.3    1.778279e-02  0.6600089   0.000000e+00
0.3    3.162278e-02  0.6600089   0.000000e+00
0.3    5.623413e-02  0.6600089   0.000000e+00
0.3    1.000000e-01  0.6600089   0.000000e+00
0.3    1.778279e-01  0.6600089   0.000000e+00
0.3    3.162278e-01  0.6600089   0.000000e+00
0.3    5.623413e-01  0.6600089   0.000000e+00
0.3    1.000000e+00  0.6600089   0.000000e+00
0.3    1.778279e+00  0.6600089   0.000000e+00
0.3    3.162278e+00  0.6600089   0.000000e+00
0.3    5.623413e+00  0.6600089   0.000000e+00
0.3    1.000000e+01  0.6600089   0.000000e+00
0.6    1.000000e-05  0.6599953   2.585403e-03
0.6    1.778279e-05  0.6599953   2.585403e-03
0.6    3.162278e-05  0.6599953   2.585403e-03
0.6    5.623413e-05  0.6599953   2.585403e-03
0.6    1.000000e-04  0.6599953   2.585403e-03
0.6    1.778279e-04  0.6599953   2.585403e-03
0.6    3.162278e-04  0.6599953   2.585403e-03
0.6    5.623413e-04  0.6599953   2.585403e-03
0.6    1.000000e-03  0.6599953   2.585403e-03
0.6    1.778279e-03  0.6604715   3.525474e-03
0.6    3.162278e-03  0.6600020   8.494208e-04
0.6    5.623413e-03  0.6600089   0.000000e+00
0.6    1.000000e-02  0.6600089   0.000000e+00
0.6    1.778279e-02  0.6600089   0.000000e+00
0.6    3.162278e-02  0.6600089   0.000000e+00
0.6    5.623413e-02  0.6600089   0.000000e+00
0.6    1.000000e-01  0.6600089   0.000000e+00
0.6    1.778279e-01  0.6600089   0.000000e+00
0.6    3.162278e-01  0.6600089   0.000000e+00
0.6    5.623413e-01  0.6600089   0.000000e+00
0.6    1.000000e+00  0.6600089   0.000000e+00
0.6    1.778279e+00  0.6600089   0.000000e+00
0.6    3.162278e+00  0.6600089   0.000000e+00
0.6    5.623413e+00  0.6600089   0.000000e+00
0.6    1.000000e+01  0.6600089   0.000000e+00
```

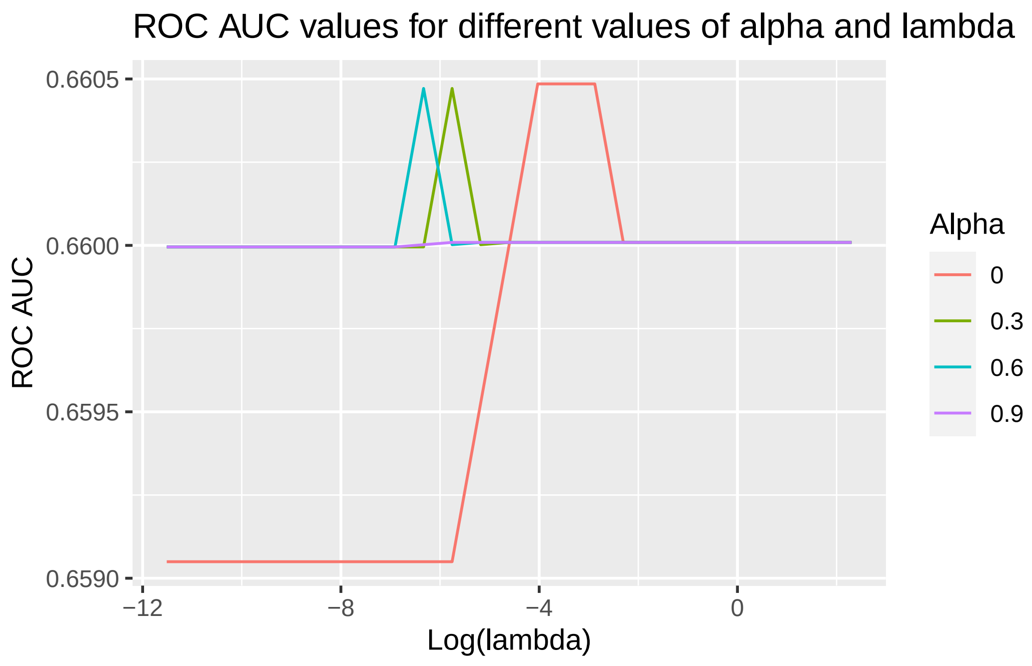
```
  0.9    1.000000e-05  0.6599953   2.585403e-03
  0.9    1.778279e-05  0.6599953   2.585403e-03
  0.9    3.162278e-05  0.6599953   2.585403e-03
  0.9    5.623413e-05  0.6599953   2.585403e-03
  0.9    1.000000e-04  0.6599953   2.585403e-03
  0.9    1.778279e-04  0.6599953   2.585403e-03
  0.9    3.162278e-04  0.6599953   2.585403e-03
  0.9    5.623413e-04  0.6599953   2.585403e-03
  0.9    1.000000e-03  0.6599953   2.585403e-03
  0.9    1.778279e-03  0.6600020   8.494208e-04
  0.9    3.162278e-03  0.6600089   0.000000e+00
  0.9    5.623413e-03  0.6600089   0.000000e+00
  0.9    1.000000e-02  0.6600089   0.000000e+00
  0.9    1.778279e-02  0.6600089   0.000000e+00
  0.9    3.162278e-02  0.6600089   0.000000e+00
  0.9    5.623413e-02  0.6600089   0.000000e+00
  0.9    1.000000e-01  0.6600089   0.000000e+00
  0.9    1.778279e-01  0.6600089   0.000000e+00
  0.9    3.162278e-01  0.6600089   0.000000e+00
  0.9    5.623413e-01  0.6600089   0.000000e+00
  0.9    1.000000e+00  0.6600089   0.000000e+00
  0.9    1.778279e+00  0.6600089   0.000000e+00
  0.9    3.162278e+00  0.6600089   0.000000e+00
  0.9    5.623413e+00  0.6600089   0.000000e+00
  0.9    1.000000e+01  0.6600089   0.000000e+00

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were alpha = 0 and lambda = 0.05623413.
```

```r
best_params <- model_glmnet$results[order(-model_glmnet$results$Accuracy),][1,]
best_params
```

```
   alpha     lambda  Accuracy       Kappa  AccuracySD     KappaSD
14     0 0.01778279 0.6604851 0.001801802 0.005028225 0.009868875
```

```r
# Plot the alpha-lambda combinations and their corresponding ROC AUC values
ggplot(model_glmnet$results, aes(x = log(lambda), y = Accuracy,
  color = factor(alpha))) +
  geom_line() +
  labs(x = "Log(lambda)", y = "ROC AUC", color = "Alpha") +
  scale_color_discrete(name = "Alpha") +
    ggtitle("ROC AUC values for different values of alpha and lambda")
```



Plot 1: Visualization of the trade-off between alpha and lambda and the impact of these hyperparameters on model performance.We can see how the ROC AUC values change for various combinations of alpha and lambda in the ggplot2 plot. It is clear that severe regularization lowers the model's performance because greater values of lambda result in smaller ROC AUC values. Furthermore, it is clear that varied alpha values result in varying ROC AUC values, with higher alpha values producing sparser models but possibly lower ROC AUC values.

The accuracy of the model_glmnet on the test data is 66%

```
# Predictions of model_glmnet on test data
pred_glm <- predict(model_glmnet, newdata = test_data)
table(pred_glm)
```

```
pred_glm
 Healthy Diseased
     299        1
```

```r
print(paste
      ("The accuracy of the model_glmnet on the test data is",
        mean(pred_glm==test_data$label)))
```

```
[1] "The accuracy of the model_glmnet on the test data is 0.656666666666667"
```

The confusion matrix has zero real positives and a lot of false negatives because the model appears to have predicted every event as healthy. As the model does not identify any instances of the Diseased class, it is not considered to be useful to predict the target variable.

```r
# Create the confusion matrix
confusionMatrix(pred_glm, test_data$label)
```

```
Confusion Matrix and Statistics

          Reference
Prediction Healthy Diseased
  Healthy      197      102
  Diseased       1        0

               Accuracy : 0.6567
                 95% CI : (0.5999, 0.7103)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.575

                  Kappa : -0.0066

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9949
            Specificity : 0.0000
         Pos Pred Value : 0.6589
         Neg Pred Value : 0.0000
             Prevalence : 0.6600
         Detection Rate : 0.6567
   Detection Prevalence : 0.9967
      Balanced Accuracy : 0.4975

       'Positive' Class : Healthy
```

Given that even though the accuracy for the logistic regression model was ~66%, a lot of false negatives were produced because the model appears to have predicted every event as healthy. So, we tried other models like SVM to answer our question.

Using the predictors hu shape, circularity, and contrast together with a radial basis kernel, this algorithm fits an SVM to predict the labels in a dental dataset. In order to specify a radial basis kernel, the method=svmRadial option of the caret package is being used to configure the SVM's hyperparameters C and sigma. With C taking the values 5e-2, 5e-1, 5, 50, 500, and 5000, and sigma taking the values 5e-5, 5e-4, 5e-3, 5e-2, and 5, the expand.grid function is used to construct a grid of hyperparameters to be searched across.

```r
# SVM
set.seed(1234)

dental_data_clean$label <-
  factor(dental_data_clean$label, levels = c("Healthy", "Diseased"))

train_control_svm <- trainControl(method="cv", number=5)

svm_grid <-  expand.grid(C = c(5e-2, 5e-1, 5, 50, 500, 5000),
                         sigma = c(5e-5, 5e-4, 5e-3, 5e-2, 5e-1, 5))
cv_svm <- train(
  label ~ hu_shape + circularity + contrast,
  data = train_data,
  trControl=train_control_svm,
  tuneGrid=svm_grid,
  method="svmRadial",
  )

# predictions of the SVM fit on the test dataset
predvals <- predict(cv_svm, newdata=test_data)
table(predvals)
```

```
predvals
 Healthy Diseased
     298        2
```

```r
print(paste("Accuracy", mean(predvals==test_data$label)))
```

```
[1] "Accuracy 0.66"
```

```r
# Create the confusion matrix
confusionMatrix(predvals, test_data$label)
```

```
Confusion Matrix and Statistics

          Reference
Prediction Healthy Diseased
  Healthy      197      101
  Diseased       1        1

               Accuracy : 0.66
                 95% CI : (0.6033, 0.7135)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.5269

                  Kappa : 0.0062

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.994949
            Specificity : 0.009804
         Pos Pred Value : 0.661074
         Neg Pred Value : 0.500000
             Prevalence : 0.660000
         Detection Rate : 0.656667
   Detection Prevalence : 0.993333
      Balanced Accuracy : 0.502377

       'Positive' Class : Healthy
```

Table 1: Shows the specific sensitivity and specificity values over diseased and healthy groups.

```r
table1 <- rbind(
  cbind('', 'Sensitivity', 'Specificity'),
  cbind('Healthy:', '0.965', '0.0098'),
  cbind('Diseased:', '0.5', '0.9949')
)

colnames(table1) <-
```

```r
  c('', 'Positive (Healthy)', 'Negative (Diseased)')

knitr::kable(table1, align = 'c')
```

	Positive (Healthy)	Negative (Diseased)
	Sensitivity	Specificity
Healthy:	0.965	0.0098
Diseased:	0.5	0.9949

From the table above, we can now see that there is overall ~66% accuracy, with >90% sensitivity but low levels of specificity for healthy category.

0.4 Discussion and Next Steps:

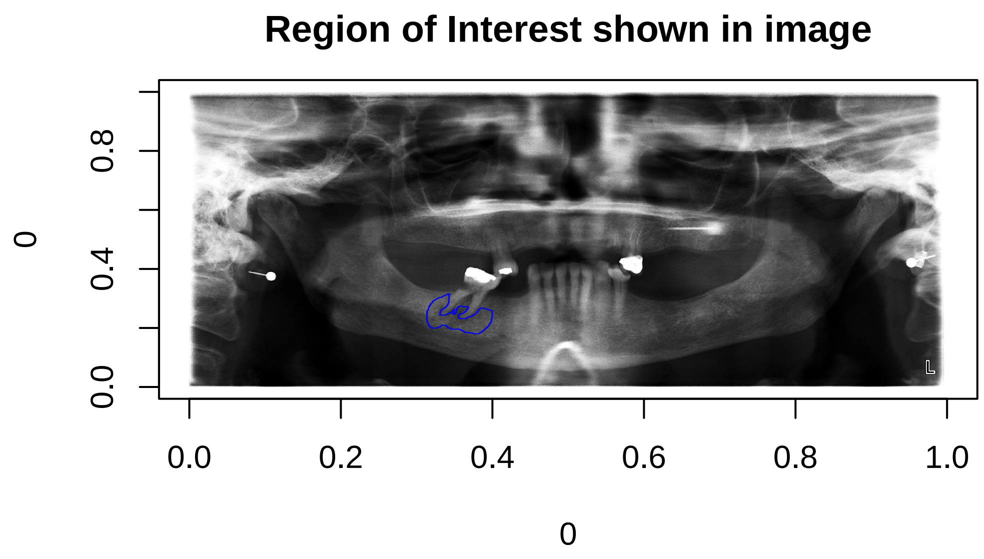
There are a variety of approaches that might improve the overall classification accuracy. First, we could collect and utilize more data to make it a balanced dataset. We could tune the SVM approach to identify the optimal value of candidate predictors, lambda and sigma. Although we used conventional values for these, examining other parameters may improve the prediction power of these. For image data like dental radiographs, a convolutional neural network (CNN or ConvNet) model could work better. Since they can automatically learn spatial hierarchies of characteristics from the original picture data, CNNs are a type of neural network that are particularly well suited for processing images. This qualifies them for tasks involving picture categorization, such as determining whether or not an image contains a specific object. They can be trained on a dataset of labeled images of teeth in the context of dental disease detection, with the labels indicating whether the teeth display symptoms of disease or not. They can learn complex patterns and traits and are an efficient tool for real-time dental disease detection because of their ability to interpret images rapidly and precisely.

Teeth segmentation masks, abnormality masks, maxillomandibular regions of interest masks, eye-tracking gaze maps, and text descriptions of the anomaly make up this dataset provided by the Tufts Dental Database. The next step I would like to pursue is to use the classification for the Tufts Dental Database to identify the specific disease that every radiograph has.

Two functions below I started with in Python, read data from JSON files that provide details about dental photos and their annotations. The functions create annotated copies of the associated images by drawing bounding boxes or polygons around particular ROIs using the PIL and OpenCV modules. This code could be helpful for displaying and analyzing dental pictures and associated annotations. (Please check region_of_interest.py to see the two functions.)

Example images after ROI extraction showing region of interest:

```r
ROI_1 <- readJPEG("/Users/drvish/Desktop/roi/roi_1.JPG")
plot(0, 0, type = "n", xlim = c(0, 1), ylim = c(0, 1))
rasterImage(ROI_1, 0, 0, 1, 1)
title(main = "Region of Interest shown in image", line = 1)
```



References:

[1] Panetta, K., Rajendran, R., Ramesh, A., Rao, S. P., & Agaian, S. (2021). Tufts dental database: a multimodal panoramic x-ray dataset for benchmarking diagnostic systems. IEEE Journal of Biomedical and Health Informatics, 26(4), 1650-1659.

[2] E. H. Said, D. E. M. Nassar, G. Fahmy, and H. H. Ammar, "Teeth segmentation in digitized dental X-ray films using mathematical morphology,"IEEE Trans. Inf. forensics Secur., vol. 1, no. 2, pp. 178–189,Jun. 2006.

[3] S. Shah, A. Abaza, A. Ross, andH.Ammar, "Automatic tooth segmentation using active contour without edges," in Proc. IEEE Biometrics Symp.,Special Session Res. Biometric Consortium Conf., 2006, pp. 1–6.

[4] A. E. Rad, M. S. M. Rahim, and A. Norouzi, "Digital dental X-ray image segmentation and feature extraction," TELKOMNIKA Indonesian J. Elect.Eng., vol. 11, no. 6, pp. 3109–3114, 2013.

[5] M. H. Mahoor and M. Abdel-Mottaleb, “Classification and numbering of teeth in dental bitewing images,” Pattern Recognit., vol. 38, no. 4,pp. 577–586, 2005.

[6] F. Aeini and F. Mahmoudi, “Classification and numbering of posterior teeth in bitewing dental images,” in Proc. 3rd Int. Conf. Adv. Comput.Theory Eng., 2010, vol. 6, pp. V6–66–V6–72.